МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Комп'ютерних наук і технологій
(повне найменування факультету)
Системний аналіз та обчислювальна математика
(повне найменування кафедри)

Пояснювальна записка

до дипломної роботи
Бакалавра
(ступінь вищої освіти)

на тему Статистичний аналіз закладів освіти європейських країн
(назва теми)

Виконав(ла): студент(ка) 4 курсу, групи КНТ – 812

Спеціальності 124 Системний аналіз
(код і найменування спеціальності)

Освітня програма (спеціалізація)
Інтелектуальні технології та прийняття рішень у складних системах

ЧІЧА В. В.
(ПРІЗВИЩЕ та ініціали)

Керівник БАХРУШИН В. Є.
(ПРІЗВИЩЕ та ініціали)

Рецензент ЛОЗОВСЬКА Л. І.
(ПРІЗВИЩЕ та ініціали)

2026

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

Кафедра «Системний аналіз та обчислювальна математика»

Ступінь вищої освіти Бакалавр

Спеціальність 124 Системний аналіз

(код і найменування)

Освітня програма (спеціалізація) «Інтелектуальні технології та прийняття рішень в складних системах»

(назва освітньої програми (спеціалізації)

ЗАТВЕРДЖУЮ

Завідувач кафедри САОМ

Еліна ТЕРЕЩЕНКО

« 08 » червня 20 26 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Чічі Всеволода Володимирович

(ПРІЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Статистичний аналіз закладів освіти європейських країн

керівник проєкту (роботи) доктор фізико-математичних наук, професор,

(науковий ступінь, вчене звання, ПРІЗВИЩЕ, ім'я, по батькові)

БАХРУШИН Володимир Євгенович,

затверджені наказом закладу вищої освіти від « 23 » квітня 20 26 № 196

2. Строк подання студентом проєкту (роботи) « 05 » червня 20 26 року

3. Вихідні дані до проєкту (роботи) Дані ETER, літературні дані за темою.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Вступ. Огляд системи вищої освіти європейських країн. Методологія дослідження. Статистичний аналіз систем вищої освіти. Висновки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів)

6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
Нормоконтроль	Широкорад Д.В.	05.06.2026	06.06.2026

7. Дата видачі завдання « 04 » травня 20 26 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Сформулювати мету та основні завдання дипломної роботи	06.05.2026 - 09.05.2026	виконано
2	Збір та підготовка даних	10.05.2026 - 13.05.2026	виконано
3	Створення програмного середовища для аналізу	14.05.2026 - 22.05.2026	виконано
4	Аналіз результатів	23.05.2026 – 27.05.2026	виконано
5	Оформлення роботи	27.05.2026 – 04.06.2026	виконано

Студент(ка) _____ **Всеволод Чіча**
(підпис) (Ім'я ПРІЗВИЩЕ)

Керівник проєкту (роботи) _____ **Володимир Бахрушин**
(підпис) (Ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

Дипломна робота: 70 с., 23 рис. , 5 табл., 17 джерел.

Тема: Статистичний аналіз закладів вищої освіти європейських країн.

У роботі досліджено системи вищої освіти європейських країн на основі даних European Tertiary Education Register. Розглянуто кількість закладів вищої освіти, чисельність студентів, структуру студентів за рівнями ISCED – 2011, частку іноземних студентів, кадрові та фінансові показники, а також рівень концентрації студентів між закладами.

Практичну частину реалізовано мовою Python з використанням бібліотек Pandas, NumPy, Matplotlib, Seaborn та Scikit-learn. Побудовано описову статистику, здійснено візуалізацію показників, аналіз концентрації студентів і факторний аналіз методом головних компонент, що дозволило виявити основні статистичні відмінності між системами вищої освіти європейських країн. Отримано очищені таблиці, агреговані показники по країнах, типах закладів та їх правовому статусу, графіки та результати факторного аналізу для подальшого порівняльного дослідження.

СТАТИСТИЧНИЙ АНАЛІЗ, ВИЩА ОСВІТА, ETER, ЗАКЛАДИ ВИЩОЇ ОСВІТИ, ЄВРОПЕЙСЬКІ КРАЇНИ, ОПИСОВА СТАТИСТИКА, КОНЦЕНТРАЦІЯ СТУДЕНТІВ.

ЗМІСТ

ЗАВДАННЯ 2
РЕФЕРАТ 4
ВСТУП 6
РОЗДІЛ 1 СИСТЕМИ ВИЩОЇ ОСВІТИ ЄВРОПЕЙСЬКИХ КРАЇН 8
1.1 Системи вищої освіти як об’єкт статистичного аналізу 8
1.2. Особливості розвитку систем вищої освіти європейських країн 10
1.3 Основні статистичні характеристики систем та закладів вищої освіти 12
РОЗДІЛ 2 МЕТОДОЛОГІЯ ДОСЛІДЖЕННЯ 13
2.1 Створення програмного середовища для розрахунків 13
2.2. Характеристика інформаційної бази ETER 14
2.3 Формування, очищення та оцінювання якості аналітичної вибірки 16
2.4. Формування та характеристика аналітичних показників по країнах та роках 23
2.5 Описова статистика 29
РОЗДІЛ 3 СТАТИСТИЧНИЙ АНАЛІЗ СИСТЕМ ВИЩОЇ ОСВІТИ 41
3.1. Статистичний аналіз та перевірка гіпотези щодо інтернаціоналізації вищої освіти 41
3.2. Оцінювання концентрації та нерівномірності розподілу студентів 48
3.3. Факторний аналіз освітніх систем 57
ВИСНОВКИ 64
ПЕРЕЛІК ДЖЕРЕЛ 66
ДОДАТОК А 69

ВСТУП

Актуальність теми полягає в тому, що в Україні обговорюються різні варіанти модернізації мережі закладів вищої освіти. Одним з підходів до прийняття обґрунтованих рішень з цього питання є порівняльний аналіз мережі закладів вищої освіти України з аналогічними мережами закладів вищої освіти Європейських країн. Зокрема, таке порівняння має стосуватися основних статистичних параметрів мереж, таких як розміри закладів вищої освіти, їх фінансування вищої освіти, іноземних студентів тощо. Ці показники мають значні розкиди як в межах країн так і між країнами. Тому просте порівняння країн за кількістю університетів або студентів не дає повного уявлення про особливості їхніх систем вищої освіти. Для обґрунтованого аналізу необхідно враховувати не лише абсолютні, а й відносні показники: середній і медіанний розмір закладу, частку іноземних студентів, співвідношення студентів і персоналу, фінансування у розрахунку на одного студента та рівень концентрації студентів між закладами. Також важливими є розподіли цих показників.

Сучасні системи вищої освіти європейських країн суттєво відрізняються за кількістю закладів, чисельністю студентів, їх розподілом за освітніми рівнями, часткою іноземних студентів, кадровим забезпеченням і фінансуванням. У зв’язку з цим виникає потреба у використанні статистичних методів для аналізу та порівняння таких систем на основі єдиної інформаційної бази.

У роботі використовується набір даних European Tertiary Education Register (ETER), який містить веріфіковану інформацію про заклади вищої освіти європейських країн зібрані за єдиною методологією. Дані ETER дають змогу аналізувати географічні, інституційні, студентські, кадрові та фінансові характеристики ЗВО. Перед проведенням аналізу дані потребують попередньої підготовки: відбору релевантних змінних, очищення спеціальних значень і пропусків, перевірки дублікатів, приведення типів даних та формування агрегованих таблиць на рівні країни і року.

Об’єкт дослідження – системи вищої освіти європейських країн.

Предмет дослідження – статистичний аналіз мереж закладів вищої освіти на основі даних ETER.

Мета роботи – визначити типові статистичні характеристики мереж закладів вищої освіти європейських країн, придатні для застосування при прийнятті обґрунтованих рішень щодо модернізації мережі закладів вищої освіти України.

Для досягнення мети поставлено такі завдання:

– Проаналізувати структуру вихідного набору даних ETER.
– Виконати очищення даних, обробку спеціальних значень.
– Сформувати базову робочу таблицю даних у розрізі країн та років.
– Розрахувати похідні показники.
– Побудувати описову статистику й здійснити розвідувальний аналіз даних.
– Сформулювати і перевірити статистичні гіпотези.
– Виконати факторний аналіз.
– Сформулювати висновки за результатами аналізу.

Методи дослідження: очищення та попередня обробка даних, описова статистика, агрегування, кореляційний аналіз, аналіз концентрації, візуалізація даних, метод головних компонент. Обчислення виконуються засобами Python із використанням бібліотек pandas, numpy, matplotlib, seaborn і scikit-learn.

РОЗДІЛ 1 СИСТЕМИ ВИЩОЇ ОСВІТИ ЄВРОПЕЙСЬКИХ КРАЇН

1.1 Системи вищої освіти як об’єкт статистичного аналізу

Сучасний етап розвитку глобального суспільства знань характеризується посиленням ролі вищої освіти як стратегічного ресурсу соціально-економічного та науково-технічного прогресу [1]. У цьому контексті виникає нагальна потреба в об’єктивному, всебічному та систематичному оцінюванні функціонування освітніх систем. Ефективним інструментом для вирішення цього завдання виступає описова статистика та статистичний аналіз, що дозволяє трансформувати масиви первинних даних у структуровану інформацію для прийняття управлінських рішень.

Система вищої освіти як об’єкт статистичного аналізу є складною, багатоаспектною та динамічною системою, що поєднує в собі ознаки соціальної та економічної підсистем держави [1]. Головна особливість цього об’єкта полягає в тому, що він охоплює як кількісні характеристики: чисельність студентів, кількість закладів вищої освіти (ЗВО), обсяги фінансування. Так і якісні параметри: рівень кваліфікації професорсько-викладацького складу, відповідність навичок випускників вимогам ринку праці, рівень інтернаціоналізації.

З погляду статистики, система вищої освіти досліджується через призму трьох основних компонентів (входів, процесів та виходів) [2]:

– Ресурсне забезпечення (вхід системи): фінансові асигнування (державні та приватні), матеріально-технічна база ЗВО, людські ресурси (абітурієнти та науково-педагогічний персонал).

– Освітній процес (внутрішнє функціонування): структура підготовки за рівнями вищої освіти (бакалавр, магістр, доктор філософії), галузями знань та формами навчання, а також показники мобільності студентів і викладачів.

– Результативність (вихід системи): чисельність випускників, рівень їхнього працевлаштування, наукова продуктивність (публікації, патенти) та внесок у формування людського капіталу країни.

Специфіка статистичного аналізу систем вищої освіти європейських країн зумовлена процесами гармонізації в межах Європейського простору вищої освіти (ЄПВО) [3]. Це вимагає використання уніфікованих методологічних підходів для забезпечення порівнянності даних між різними національними системами. Основним інструментом такої уніфікації є Міжнародна стандартна класифікація освіти (МСКО / ISCED) [4], розроблена ЮНЕСКО, яка дозволяє зводити національні освітні рівні до єдиного міжнародного стандарту.

Основними джерелами статистичних даних для аналізу європейських систем вищої освіти є:

– Статистична служба Європейського Союзу (Євростат / Eurostat).

– Організація економічного співробітництва та розвитку (ОЕСР / OECD).

– Інститут статистики ЮНЕСКО (UIS).

– Національні статистичні відомства та міністерства освіти.

Таким чином, статистичний аналіз системи вищої освіти не обмежується простим збором та групуванням даних. Він передбачає розрахунок системи взаємопов'язаних індикаторів, виявлення структурних зрушень, моделювання тенденцій розвитку та проведення порівняльних (компаративних) досліджень. Це дозволяє визначити ефективність функціонування національних освітніх моделей та їхню конкурентоспроможність у глобальному вимірі.

1.2. Особливості розвитку систем вищої освіти європейських країн

Сучасний етап розвитку вищої освіти в європейських країнах характеризується глибокими трансформаційними процесами, спрямованими на створення єдиного Європейського простору вищої освіти (ЄПВО). Основним каталізатором цих змін став Болонський процес [5], який визначив спільні стратегічні орієнтири для національних освітніх систем. Аналіз розвитку вищої освіти в Європі дозволяє виділити кілька ключових особливостей.

Ступеневість та уніфікація освітніх рівнів. Більшість європейських країн успішно перейшли на трициклову систему вищої освіти: бакалавр, магістр, доктор філософії. Це забезпечило порівнянність дипломів та кваліфікацій між різними країнами. Для забезпечення прозорості оцінювання та визначення трудомісткості навчання використовується Європейська кредитно-трансферна система (ECTS), де один навчальний рік зазвичай еквівалентний 60 кредитам.

Забезпечення якості освіти (Quality Assurance). Особливу увагу в країнах Європейського Союзу (ЄС) приділяють створенню внутрішніх та зовнішніх систем контролю якості. Діяльність університетів оцінюється незалежними акредитаційними агенціями, які керуються Стандартами і рекомендаціями щодо забезпечення якості в Європейському просторі вищої освіти (ESG). Основний акцент зміщено з контролю процесу навчання на оцінювання його результатів.

Студентоцентроване навчання та індивідуальні освітні траєкторії. Європейська вища освіта базується на принципі студентоцентризму. Це передбачає:

- активну участь студентів у формуванні власного навчального плану;
- можливість вибору дисциплін (вибіркова складова часто становить понад 25% від загального обсягу програми);
- зміну ролі викладача від ретранслятора знань до ментора та фасилітатора.

Інтернаціоналізація та академічна мобільність. Інтеграційні процеси стимулюють мобільність студентів і науково-педагогічних працівників. Завдяки програмам фінансування європейські здобувачі мають можливість навчатися або проходити стажування в інших країнах без втрати часу, оскільки здобуті кредити автоматично зараховуються в рідному університеті.

Диверсифікація фінансування та автономія ЗВО. Сучасні європейські університети мають високий рівень академічної, організаційної, кадрової та фінансової автономії. Поряд із державним фінансуванням заклади вищої освіти (ЗВО) активно залучають кошти через грантові програми, наукові дослідження для бізнесу та створення стартапів.

Водночас розвиток вищої освіти в Європі не є однорідним. Дослідники виділяють кілька регіональних моделей [6]:

– Англо-саксонська модель (Велика Британія): орієнтована на високу ринкову конкуренцію, значну частку платного навчання та залучення іноземних студентів.

– Континентальна або Німецька модель (Німеччина, Австрія): традиційно безкоштовна або з мінімальними зборами, з акцентом на фундаментальну науку та поєднання навчання з дослідженнями (гумбольдтівський принцип).

– Скандинавська модель (Фінляндія, Швеція): повністю безкоштовна вища освіта, що фінансується державою, з високим рівнем соціального захисту студентів та орієнтацією на рівні можливості.

Таким чином, головною тенденцією розвитку європейської вищої освіти є баланс між збереженням кращих національних традицій та інтеграцією в єдиний глобалізований простір, де ключову роль відіграють якість, мобільність та інноваційність.

1.3 Основні статистичні характеристики систем та закладів вищої освіти

Аналіз та оцінювання ефективності функціонування систем і окремих закладів вищої освіти (ЗВО) базується на комплексі кількісних і якісних показників. На макрорівні спроможність освітньої системи характеризують такі показники, як загальна кількість ЗВО за типами й формами власності, чисельність студентів у розрахунку на 10 000 населення та загальний коефіцієнт охоплення вищою освітою. На рівні конкретного ЗВО комплексний моніторинг діяльності здійснюється за такими основними блоками статистичних характеристик [7]:

- кадровий потенціал
- контингент здобувачів освіти
- наукова та інноваційна діяльність
- результативність та якість освітнього процесу
- фінансово-економічна стабільність
- матеріально-технічна та цифрова інфраструктура

Системне відстеження та аналіз зазначених характеристик дозволяє здійснювати порівняльний аналіз університетів, оцінювати їхню конкурентоспроможність та приймати обґрунтовані управлінські рішення на рівні керівництва ЗВО та органів державної влади.

РОЗДІЛ 2 МЕТОДОЛОГІЯ ДОСЛІДЖЕННЯ

2.1 Створення програмного середовища для розрахунків

Для реалізації дослідження застосовано мову програмування Python та інтерактивне середовище розробки Jupyter Notebook [8]. Вибір мови Python[9] зумовлений наявністю розвиненої екосистеми інструментів для обробки даних, описової статистики, статистичного аналізу, візуалізації. Середовище Jupyter Notebook використано з метою забезпечення послідовного поєднання програмного коду, розрахункових таблиць, графічного матеріалу та текстових пояснень, що спрощує верифікацію проміжних результатів і забезпечує відтворюваність розрахунків.

Основними інструментами аналізу є такі спеціалізовані бібліотеки:

– pandas – обробка, структурування та агрегація даних [10];

– numpy – виконання високоефективних числових та матричних розрахунків [11];

– matplotlib та seaborn – візуалізація даних та побудова графіків [12], [13];

– scikit-learn – стандартизація змінних, факторний аналіз та кластеризація [14];

– pathlib – керування шляхами до файлів у файловій системі.

Процес розрахунків декомпоновано на послідовні логічні блоки, які реалізовано у вигляді окремих файлів-ноутбуків. Структуру та призначення цих файлів наведено далі.

Таблиця 2.1 – Структура та призначення розрахункових модулів Jupyter Notebook

Назва файлу (ноутбука)	Функціональне призначення
00_df_base_cleanup.ipynb	Попереднє очищення та підготовка початкових даних
01_country_year_profiles_and_eda_metrics.ipynb	Розрахунок базових аналітичних показників
02_descriptive_statistics_and_eda.ipynb	Описова статистика та розвідувальний аналіз даних (EDA)
03–04_hypothesis_*.ipynb	Статистична перевірка дослідницьких гіпотез
05_factor_analysis.ipynb	Проведення факторного аналізу

Кожен модуль виконує визначений етап аналізу та використовує масиви даних, отримані на попередньому кроці. Зазначена модульна структура унеможливлює дублювання обчислень та забезпечує прозорість перевірки результатів на кожній стадії дослідження.

2.2. Характеристика інформаційної бази ETER

Основою для дослідження європейських систем вищої освіти у цій роботі виступає Європейський реєстр вищої освіти (European Tertiary Education Register – ETER) [15]. Цей проєкт є найбільш авторитетним джерелом уніфікованих даних у європейському освітньому просторі.

Головною методологічною перевагою ETER є подолання проблеми дезагрегації даних. На відміну від макропоказників Eurostat чи ЮНЕСКО, ETER акумулює інформацію безпосередньо від закладів вищої освіти (ЗВО). Це дозволяє аналізувати внутрішню гетерогенність університетських систем як всередині країн, так і на міждержавному рівні.

Інформаційна архітектура ETER охоплює понад 3000 ЗВО з більше ніж 30 країн, включаючи членів ЄС, країни ЄАВТ та Велику Британію. До Реєстру включаються заклади, що офіційно визнані на національному рівні та мають право присуджувати ступені щонайменше бакалаврського рівня (ISCED 6) і

вище. База охоплює період з 2016 по 2023 роки, що дозволяє відстежити стійкі структурні тренди та оцінити трансформації в системах вищої освіти в динаміці [16].

Дані ETER оновлюються щорічно та структуровані за п'ятьма ключовими вимірами:

– інституційні характеристики: правовий статус (державний чи приватний), географічне розташування, рік заснування, тип закладу;

– студентський контингент: кількість студентів та випускників, диференційованих за рівнями освіти (ISCED 5–8) [17], статтю, галузями знань (ISCED-F 2013) та мобільністю;

– персонал та кадрова структура: кількість академічного та неакадемічного персоналу у фізичних особах (HC) та в еквіваленті повної зайнятості (FTE), із розподілом за статтю, громадянством та науковими ступенями;

– фінансові показники: структура доходів (державні субсидії, плата за навчання, гранти) та витрат (на персонал, капітальні та операційні витрати), де дані подаються в євро за паритетом купівельної спроможності (PPS) для коректного порівняння;

– дослідницька діяльність: залученість ЗВО у наукові проєкти, обсяги фінансування досліджень, кількість аспірантів (ISCED 8) та захищених дисертацій.

Методологія ETER узгоджена із загальноєвропейським стандартом збору даних UOE (UNESCO-OECD-Eurostat), що гарантує сувору відповідність дефініцій. Водночас використання цієї бази потребує врахування специфічних кодів пропущених та конфіденційних значень, що зумовлює необхідність етапу очищення даних.

2.3 Формування, очищення та оцінювання якості аналітичної вибірки

Перехід від первинного масиву сирих даних Європейського реєстру закладів вищої освіти до етапу статистичного аналізу передбачає попереднє формування аналітичної вибірки, її технічне очищення та оцінювання якісних характеристик. У межах цього дослідження зазначений етап реалізовано у вигляді окремого програмного модуля «00_df_base_cleanup.ipynb». Результатом його виконання є очищений масив даних, збережений у файлі «df_base_clean.csv». Цей набір даних є основою для подальшого розрахунку похідних показників, агрегування даних на рівні макросистем та проведення описової статистики та статистичного аналізу.

Основне завдання цього етапу полягало в трансформації повного вивантаження бази даних ETER у робочу таблицю «df_base», що містить виключно ті змінні, які безпосередньо залучені до аналізу систем вищої освіти європейських країн. Такий підхід дає змогу мінімізувати надлишковість інформації, уніфікувати структуру даних та забезпечити повну відтворюваність подальших обчислень.

Для формування аналітичної вибірки з початкового масиву ETER виокремлено такі групи змінних:

– ідентифікаційні параметри закладу: унікальний ідентифікатор закладу вищої освіти, повна офіційна назва закладу, країна та звітний рік спостереження;

– географічні характеристики: код країни, місто базування, регіональні ознаки за класифікацією NUTS, географічні координати (широта і довгота);

– інституційні характеристики: тип закладу, категорія ЗВО, правовий статус та форма власності;

– контингент здобувачів вищої освіти: загальна чисельність студентів, а також розподіл кількості студентів за рівнями Міжнародної стандартної класифікації освіти (ISCED 5, ISCED 6, ISCED 7 та ISCED 8);

– показники інтернаціоналізації: загальна чисельність іноземних студентів та їх розподіл за окремими рівнями освіти;

– кадрові показники: чисельність академічного та адміністративно-допоміжного персоналу, загальна чисельність працівників у еквіваленті повної зайнятості (Full-Time Equivalent, FTE);

– фінансові показники: обсяги поточних доходів, загальних витрат, фінансування зі сторонніх джерел (третьої сторони) та цільові витрати на науково-дослідну діяльність (R&D);

– індикатори якості даних: спеціалізовані службові змінні ETER (метадані), що відображають статус, ступінь повноти або методологічні особливості конкретних спостережень.

У межах програмної реалізації відбір релевантних змінних виконано шляхом ініціалізації списку цільових стовпців та формування відповідного об'єкта DataFrame:

Приклад програмного коду на мові Python:

python

```
selected_columns = [
    "institution_id",
    "institution_name",
    "country_code",
    "year",
    "city",
    "latitude",
    "longitude",
    "legal_status",
    "students_isced5",
    "students_isced6",
    "students_isced7",
    "students_isced8",
    "foreign_students_isced5_7_total",
```

```
    "academic_personnel_fte",
    "total_personnel_fte",
    "total_current_revenues_eur",
    "total_current_expenditure_eur"
]
df_base = df_raw[selected_columns].copy()
```

Після створення базової таблиці здійснено уніфікацію найменувань стовпців. Назви приведено до єдиного стандарту «snake_case». Це забезпечує зручність подальшого оперування змінними в середовищі Python та мінімізує ризик виникнення синтаксичних помилок під час звернення до атрибутів об'єкта.

Приклад програмного коду на мові Python:

```
python
df_base.columns = (
    df_base.columns
    .str.strip()
    .str.lower()
    .str.replace(" ", "_")
    .str.replace("-", "_")
)
```

Критично важливим кроком очищення масиву стала обробка спеціальних кодів ETER. У первинному наборі даних замість числових значень у стовпцях із кількісними показниками часто містяться буквені службові коди. Їх наявність унеможливлює виконання математичних операцій (знаходження сум, середніх значень, часток чи коефіцієнтів). Перелік та інтерпретацію цих кодів наведено в таблиці.

Таблиця 2.2 – Спеціальні коди та значення в базі даних ETER

Код	Зміст та інтерпретація службового коду
a	Показник не застосовується до цього об'єкта
m	Інформація відсутня (пропущені дані)
x	Деталізовані дані недоступні, проте значення враховано в загальному підсумку
xc	Значення інтегровано до іншої підколонки
xr	Значення інтегровано до іншого рядка (суміжного спостереження)
nc	Дані не збиралися або показник уведено в звітність пізніше
c	Конфіденційна інформація
s	Значення є додатним, але не перевищує 3 одиниць (закодовано для деперсоніфікації)

У числових аналітичних стовпцях зазначені коди ідентифіковано та замінено на системні пропуски NaN (Not a Number). Таке рішення є методологічно обґрунтованішим, ніж заміна кодів нулями, оскільки факт відсутності чи конфіденційності даних не еквівалентний нульовому значенню контингенту студентів, кадрового потенціалу чи фінансових потоків ЗВО.

Приклад програмного коду на мові Python:

```
python
import numpy as np
special_codes = ["a", "m", "x", "xc", "xr", "nc", "c", "s"]
df_base = df_base.replace(special_codes, np.nan)
```

Після заміни категоріальних службових кодів усі кількісні змінні трансформовано у числовий тип даних («float64» або «int64»). Це є обов'язковою умовою для коректного виконання процедур агрегування, групування та обчислення дескриптивних статистик.

Приклад програмного коду на мові Python:

```
python
numeric_columns = [
    "students_isced5",
    "students_isced6",
```

```
    "students_isced7",
    "students_isced8",
    "foreign_students_isced5_7_total",
    "academic_personnel_fte",
    "total_personnel_fte",
    "total_current_revenues_eur",
    "total_current_expenditure_eur"
]
for col in numeric_columns:
    df_base[col] = pd.to_numeric(df_base[col], errors="coerce")
```

Наступним кроком аналізу була верифікація масиву на наявність дублікатів. Враховуючи, що база даних ETER має структуру панельних даних, присутність одного і того самого закладу вищої освіти в таблиці є правомірною, якщо ці записи стосуються різних років. Отже, дублювання суто за ідентифікатором «institution_id» не вважається помилкою. Валідації підлягала унікальність комбінації ознак «ідентифікатор закладу – звітний рік».

Приклад програмного коду на мові Python:

```
python
duplicates_panel = df_base.duplicated(
    subset=["institution_id", "year"],
    keep=False
)
df_duplicates = df_base[duplicates_panel]
```

Виявлені випадки дублювання записів для одного об'єкта в межах одного часового зрізу підлягали детальному експертному аналізу з метою запобігання подвійному врахуванню кількісних параметрів під час подальшої просторової агрегації.

Паралельно проведено діагностику структури пропущених значень. Автоматичне вилучення рядків із пропусками (listwise deletion) не застосовувалося через те, що різні блоки дослідження мають відмінні вимоги до

критичних змінних. Наприклад, об'єкт із пустими фінансовими показниками може містити вичерпні й придатні для аналізу дані щодо структури студентського контингенту.

Приклад програмного коду на мові Python:

```
python
missing_summary = (
    df_base.isna()
    .mean()
    .sort_values(ascending=False)
    .reset_index()
)
missing_summary.columns = ["column", "missing_share"]
```

Для оптимізації подальших етапів моделювання на основі очищеної матриці сформовано базові розрахункові метрики. Зокрема, інтегральний показник чисельності студентів розраховувався як сума за всіма доступними рівнями ISCED:

Приклад програмного коду на мові Python:

```
python
df_base["students_total_calc"] = (
    df_base["students_isced5"].fillna(0)
    + df_base["students_isced6"].fillna(0)
    + df_base["students_isced7"].fillna(0)
    + df_base["students_isced8"].fillna(0)
)
```

Під час розрахунку відносних індикаторів застосовано умову додатного знаменника, що запобігає виникненню помилок ділення на нуль. Обчислення проводилися лише за умови наявності валідного значення, що перевищує нуль.

Приклад програмного коду на мові Python:

```
python
df_base["foreign_students_share"] = np.where(
```

```
    df_base["students_total_calc"] > 0,
    df_base["foreign_students_isced5_7_total"]
df_base["students_total_calc"],
    np.nan
)
```

Індикатори якості даних (метадані) було збережено у фінальній структурі вибірки. Вони виступають важливим джерелом інформації про надійність та повноту числових метрик, що враховується під час інтерпретації результатів статистичного оцінювання.

Після завершення процедур очищення та первинної трансформації сформований масив даних збережено у файл формату CSV:

Приклад програмного коду на мові Python:

```
python
df_base.to_csv(
    "data/processed/df_base_clean.csv",
    index=False
)
```

Сформований файл «df_base_clean.csv» є кінцевим продуктом підготовчого етапу і використовується як вхідна інформація для подальшої роботи з ним.

Таким чином, у межах підрозділу 2.3 розроблено та реалізовано процедуру формування, очищення та верифікації якості аналітичної вибірки ETER. Отриманий масив «df_base_clean.csv» забезпечує уніфікований методологічний базис для подальшого математичного моделювання, просторового аналізу, перевірки статистичних гіпотез та типізації систем вищої освіти європейських країн.

2.4. Формування та характеристика аналітичних показників по країнах та роках

Після очищення початкових даних Європейського реєстру вищої освіти (ETER) було сформовано систему аналітичних показників, необхідних для дослідження характеристик окремих закладів вищої освіти (ЗВО) та порівняльного аналізу національних систем вищої освіти європейських країн. Розрахунки виконувалися на основі набору даних «df_base_clean.csv».

Показники формувалися на двох основних рівнях. Перший рівень відповідав комбінації «заклад вищої освіти – рік» і використовувався для характеристики масштабу, структури студентського контингенту, інтернаціоналізації, кадрового забезпечення та фінансових ресурсів окремих ЗВО. Другий рівень відповідав комбінації «країна – рік» і був призначений для міжкраїнного порівняння систем вищої освіти.

У результаті було сформовано набір із 29 591 спостереження на рівні ЗВО–рік. Після агрегації отримано 376 профілів країна–рік, які охоплюють 40 країн за період 2011–2023 років. Для агрегованих таблиць обов'язково перевірялася унікальність складеного ключа «country_code + year».

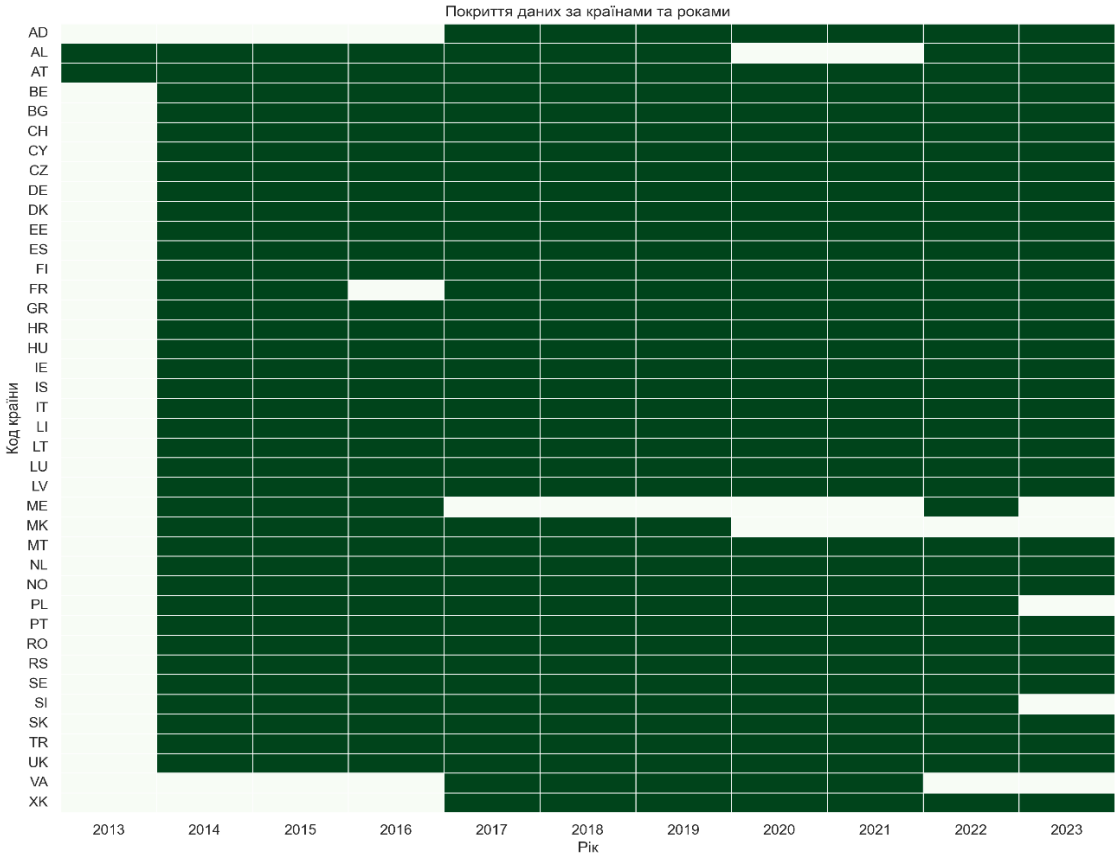

Рисунок 2.1 – Графік покриття даних за агрегацією року та країни

Основною таблицею інституційного рівня стала «institution_year_metrics». Вона містить базові характеристики закладів та розраховані аналітичні показники для кожного ЗВО за відповідний рік.

Одним із головних показників масштабу закладу є загальна кількість студентів «students_total». Насамперед вона розраховувалася як сума наявного в ETER агрегованого показника кількості студентів рівнів ISCED 5–7 та кількості студентів рівня ISCED 8:

$$StudentsTotal_{\{it\}} = StudentsISCED5-7_{\{it\}} + StudentsISCED8_{\{it\},} \quad (2.1)$$

де $StudentsTotal_{\{it\}}$ – загальна кількість студентів у певному закладі вищої освіти в певному році;

$StudentsISCED5-7_{\{it\}}$– кількість студентів рівнів ISCED 5–7 у певному закладі вищої освіти в певному році;

$StudentsISCED8_{\{it\}}$– кількість студентів рівня ISCED 8 (докторський рівень)у певному закладі вищої освіти в певному році;

Якщо агрегований показник для рівнів ISCED 5–7 був відсутній, використовувалася сума доступних значень за окремими підрівнями освіти. Аналогічний підхід застосовувався для визначення загальної кількості іноземних студентів («foreign_students_total»).

Під час підсумовування пропущені значення не замінювалися нулями автоматично. Якщо всі складові певного показника були відсутні, результат також залишався пропущеним «NaN». Це дозволило відрізнити реальне нульове значення від відсутності інформації у звітності закладу та запобігти штучному заниженню підсумкових результатів.

На основі загальної кількості студентів та чисельності іноземних студентів було розраховано частку іноземних студентів «foreign_students_share»:

$$ForeignStudentsShare_{\{it\}} = \frac{ForeignStudentsTotal_{\{it\}}}{StudentsTotal_{\{it\}}}, \qquad (2.2)$$

де $ForeignStudentsTotal_{\{it\}}$ – загальна кількість іноземних студентіву певному закладі вищої освіти в певному році;

Для характеристики освітньої структури кожного ЗВО також визначалися частки студентів рівнів ISCED 5, ISCED 6, ISCED 7 та ISCED 8 у загальній кількості студентів. Ці показники дають змогу порівнювати заклади за співвідношенням програм короткого циклу, бакалаврського, магістерського та докторського рівнів.

Усі частки та відносні показники розраховувалися лише за умови, що знаменник був наявним і перевищував нуль.

Якщо знаменник дорівнював нулю або був відсутній, результат зберігався як пропущене значення. Такий підхід застосовувався до показників інтернаціоналізації, структури студентського контингенту, кадрового забезпечення та фінансування.

Для характеристики кадрового забезпечення було сформовано два відносні показники:

$$StudentsPerAcademicStaff_{ct} = \frac{StudentsTotal_{ct}}{cademicPersonnelFTE_{ct}} \quad (2.3)$$

$$StudentsPerTotalStaff_{ct} = \frac{StudentsTotal_{ct}}{otalPersonnelFTE_{ct}} \quad (2.4)$$

AcademicPersonnelFTE – чисельність академічного персоналу конкретного закладу у конкретному році (в еквіваленті повної зайнятості, FTE);

TotalPersonnelFTE – загальна чисельність персоналу конкретного закладу у конкретному році (в еквіваленті повної зайнятості, FTE).

Перший показник характеризує кількість студентів у розрахунку на одного працівника академічного персоналу, а другий враховує весь штат закладу. Менші значення цих коефіцієнтів можуть свідчити про вищий рівень кадрового забезпечення. Водночас такі показники не можна безпосередньо трактувати як пряму оцінку якості освіти, оскільки правила обліку персоналу суттєво відрізняються між країнами.

Для міжнародних порівнянь використовувалися фінансові показники, скориговані за паритетом купівельної спроможності, що частково нівелює вплив відмінностей у рівнях цін між країнами. Проте навіть скориговані показники необхідно інтерпретувати обережно через різну повноту фінансових даних і відмінності в національних правилах фінансової звітності..

Після формування показників інституційного рівня дані було агреговано за кодом країни та роком. Абсолютні значення, зокрема кількість студентів, іноземних студентів, персоналу та обсяги фінансування, підсумовувалися за всіма доступними закладами відповідної країни.

Для подальшого аналізу було створено декілька взаємопов’язаних аналітичних таблиць.

Таблиця 2.3 – Перелік та призначення сформованих аналітичних таблиць

Назва аналітичної таблиці	Призначення таблиці
institution_year_metrics	Показники окремих ЗВО за кожний рік
student_structure_metrics	Структура студентів країни за рівнями ISCED
finance_staff_metrics	Кадрові та фінансові показники країни
concentration_metrics	Показники концентрації студентів між ЗВО
country_year_profiles	Об’єднаний профіль системи вищої освіти країни
analysis_readiness_report	Оцінювання повноти даних для подальшого аналізу

Головною агрегованою таблицею стала «country_year_profiles» Кожен її рядок характеризує систему вищої освіти певної країни в конкретному році. Таблиця об’єднує показники масштабу системи, структури студентів, інтернаціоналізації, кадрового та фінансового забезпечення, а також концентрації студентів між закладами.

Масштаб національних систем вищої освіти характеризувався такими показниками:

– «institutions_count» – кількість унікальних ЗВО;
– «total_students» – загальна кількість студентів країни;
– «mean_institution_size» – середня кількість студентів у закладі;
– «median_institution_size» – медіанна кількість студентів у закладі;
– «max_institution_size» – кількість студентів у найбільшому ЗВО країни.

Спільне використання середнього та медіанного розміру є важливим через асиметрію розподілу закладів за кількістю студентів. Наявність невеликої кількості дуже великих закладів суттєво зміщує середнє арифметичне значення

вправо, тому медіана використовується як додатковий, стійкіший до викидів показник.

Частка іноземних студентів на рівні країни розраховувалася як відношення сумарної кількості іноземних студентів до загальної кількості студентів усіх ЗВО країни.

Такий спосіб розрахунку враховує фактичний розмір кожного закладу (є зваженим показником). Він є більш обґрунтованим, ніж просте середнє значення інституційних часток, за якого малі та великі ЗВО мали б однаковий вплив на підсумковий результат країни.

Кадрові й фінансові показники рівня країни також визначалися через співвідношення агрегованих сум.

Отже, під час агрегації не використовувалося просте середнє інституційних коефіцієнтів. Це дозволило врахувати різний масштаб закладів і уникнути викривлення загального значення показника країни через однаковий внесок малих та великих ЗВО.

Окремий блок аналітичних показників було сформовано для оцінювання нерівномірності розподілу студентів між закладами країни. Для цього використовувалися показники часток концентрації «top_1_share», «top_5_share», «top_10_share» та коефіцієнт Джині «gini_students».

Показники «Top-N» визначають частку студентів країни, які навчаються відповідно в одному, п’яти або десяти найбільших ЗВО.

Чим вище значення відповідного показника, тим більша частина студентів зосереджена в найбільших закладах країни.

Коефіцієнт Джині використовувався як узагальнений показник нерівномірності розподілу студентів між ЗВО.

Значення коефіцієнта, близьке до нуля, характеризує відносно рівномірний розподіл студентів між закладами. Вищі значення свідчать про більшу концентрацію студентського контингенту у великих ЗВО. Для розрахунку використовувалися лише невід’ємні значення. Якщо дані були

відсутні або загальна кількість студентів в системі дорівнювала нулю, коефіцієнт не розраховувався.

Після формування аналітичних таблиць було проведено комплекс перевірок якості (валідації) розрахованих показників. На рівні ЗВО–рік перевірялася унікальність ключа «institution_id + year», а на рівні країна–рік – ключа «country_code + year». Додатково контролювалися відсутність від'ємних значень, належність значень часток інтервалу [0; 1] та наявність пропусків у ключових метриках. Розрахунок коефіцієнта Джині було незалежно перевірено за двома еквівалентними математичними формулами.

Для оцінювання придатності даних країн до окремих напрямів дослідження сформовано звітну таблицю «analysis_readiness_report». Вона містить кількість доступних років спостереження та кількість пропущених значень для показників загальної кількості студентів, інтернаціоналізації, кадрового забезпечення, фінансування та концентрації студентів. Ця таблиця не виключає країни з аналізу автоматично, а дає можливість обґрунтовано визначати робочу вибірку для кожного наступного аналітичного блоку.

На основі сформованих таблиць надалі створювалися описова статистика, рейтинги країн, графіки розподілів, кореляційні матриці та допоміжні таблиці для перевірки дослідницьких гіпотез. Основні показники при цьому повторно не розраховувалися. Описовий аналіз використовував уже підготовлені таблиці «institution_year_metrics» та «country_year_profiles», що забезпечило єдину методику розрахунку метрик у всіх частинах дослідження.

2.5 Описова статистика

Для виявлення основних відмінностей між системами вищої освіти європейських країн було проведено описовий статистичний аналіз сформованих показників рівня «країна–рік». Аналіз охоплює масштаби систем вищої освіти,

структуру студентів за рівнями ISCED, інтернаціоналізацію, кадрове забезпечення, фінансування та концентрацію студентів між закладами.

Для міжкраїнного порівняння використано останній доступний у базі спільний часовий зріз – 2023 рік. Залежно від повноти даних окремі показники доступні для 19–35 країн, тому кількість країн у різних тематичних блоках відрізняється. Під час розрахунку показників пропущені значення не замінювалися нулями. Частки та відносні показники обчислювалися лише за умови наявності додатного знаменника, що дозволило уникнути штучного заниження результатів і зберегти інформацію про неповноту даних.

Масштаб національних систем вищої освіти оцінювався за кількістю закладів вищої освіти, загальною кількістю студентів, а також середнім і медіанним розміром закладу.

У 2023 році середня кількість освітніх закладів на одну країну становила приблизно 90 закладів, тоді як медіанне значення дорівнювало 41 закладу. Значна різниця між середнім і медіанним значеннями свідчить про асиметричність розподілу: декілька великих систем суттєво підвищують загальне середнє. Найбільшу кількість закладів мали такі країни:

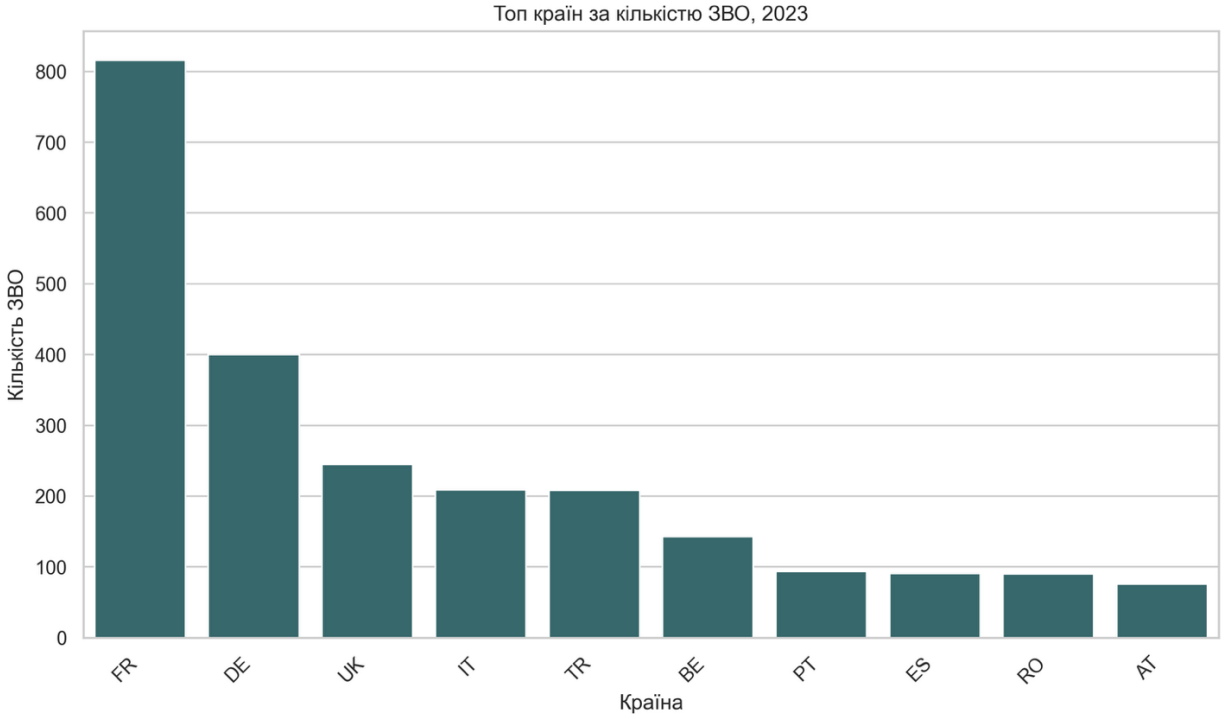

Рисунок 2.2 – Стовпчаста діаграма

– Франція – 816 закладів;
– Німеччина – 400 закладів;
– Велика Британія – 245 закладів;
– Італія – 209 закладів;
– Туреччина – 208 закладів.

За загальною кількістю студентів найбільшою була система Туреччини (понад 7,08 млн осіб). Далі розташувалися Велика Британія (2,88 млн осіб), Німеччина (2,87 млн осіб), Франція (2,56 млн осіб) та Італія (2,26 млн осіб).

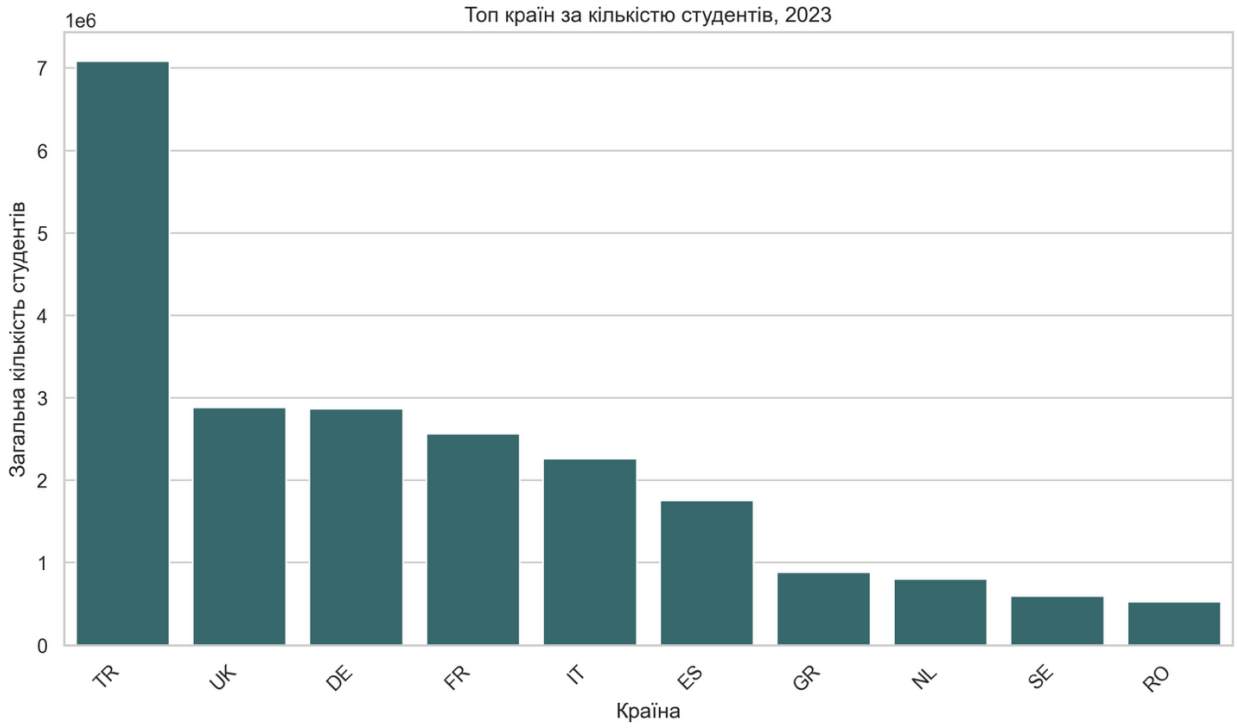

Рисунок 2.3 – Стовпчаста діаграма

Середня кількість студентів у розрахунку на одну країну становила близько 780 тис. осіб, однак медіанне значення дорівнювало лише 289 тис. осіб, що також вказує на значну нерівномірність масштабів систем.

Найбільший середній розмір закладу спостерігався у Туреччині (близько 34,4 тис. студентів), Греції (20,5 тис.), Іспанії (19,3 тис.), Швеції (15,3 тис.) та Нідерландах (14,1 тис. студентів). Отже, велика кількість студентів у країні не завжди пов’язана з великою кількістю ЗВО: окремі системи розвиваються за

інтенсивною моделлю і характеризуються меншою кількістю, але більшим середнім розміром закладів.

Структура студентів аналізувалася за частками студентів рівнів ISCED 5–8 у загальній кількості студентів. У більшості досліджуваних країн основну частину становили студенти рівня ISCED 6, який відповідає бакалаврським програмам. Середня частка студентів рівня ISCED 6 дорівнювала 62,5%, а медіанна – 62,4%. Найвищі значення цього показника спостерігалися у Греції (84,1%), Косові (81,0%) та Нідерландах (79,9%).

Середня частка студентів рівня ISCED 7 (магістерські програми) становила 22,2%. Найбільші частки студентів цього рівня зафіксовано в Люксембурзі (42,5%), на Кіпрі (41,5%), у Франції (35,8%) та Австрії (32,1%).

Студенти рівня ISCED 8 (докторські програми) у середньому становили близько 4,9% студентського контингенту. Порівняно високі частки спостерігалися у Ліхтенштейні, Люксембурзі та Швейцарії. Водночас для невеликих країн високі частки можуть бути пов’язані зі специфікою національних систем, вузькою спеціалізацією окремих закладів та малою загальною кількістю студентів.

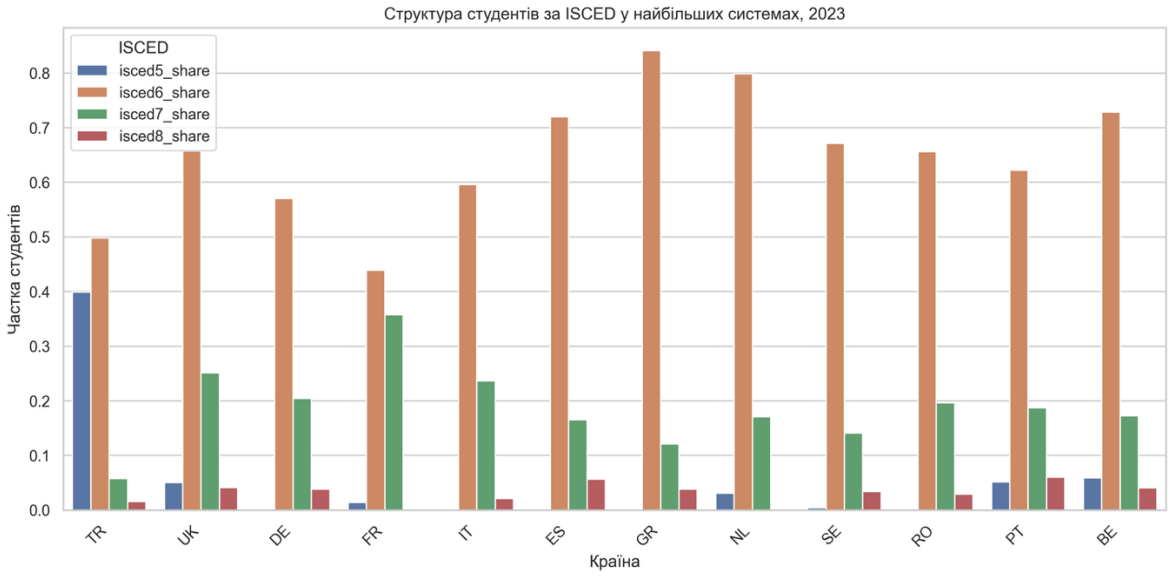

Рисунок 2.4 – Стовпчаста діаграма

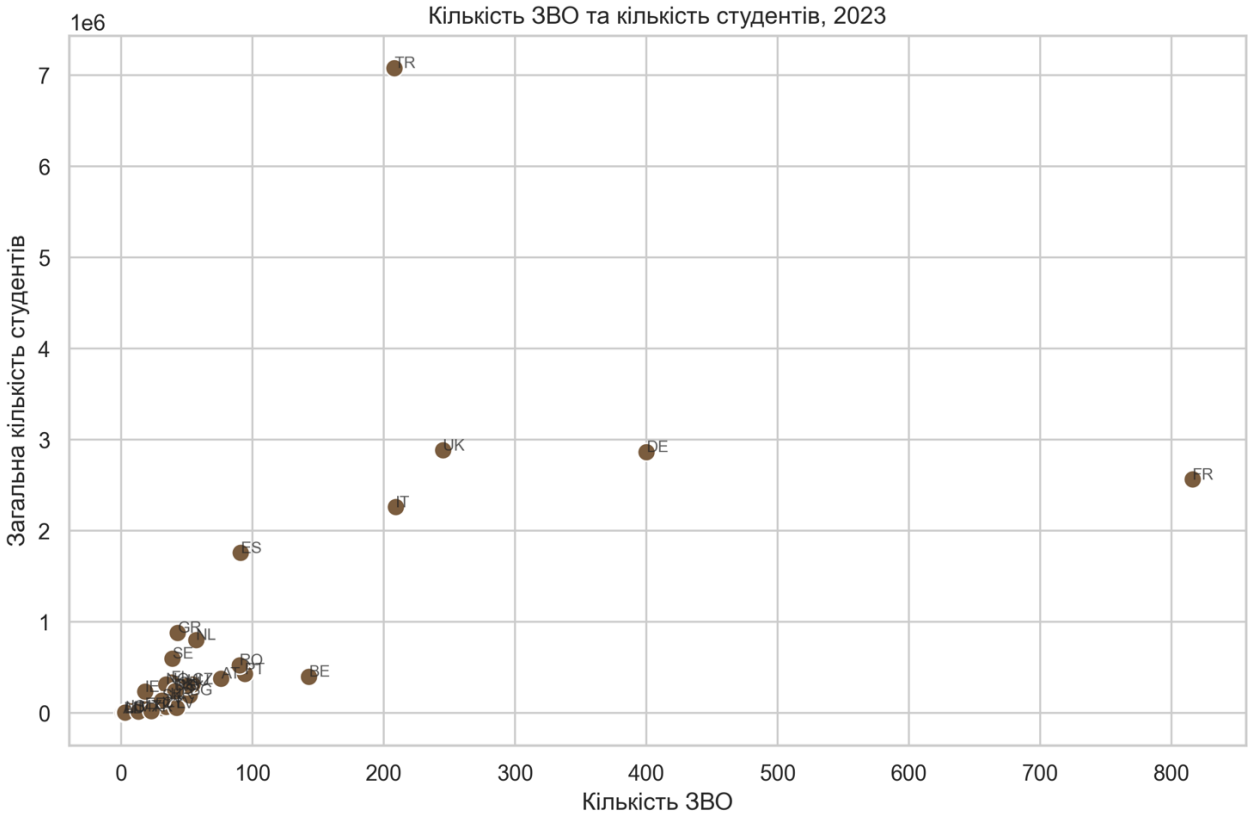

Рисунок 2.5 – Діаграма розсіювання

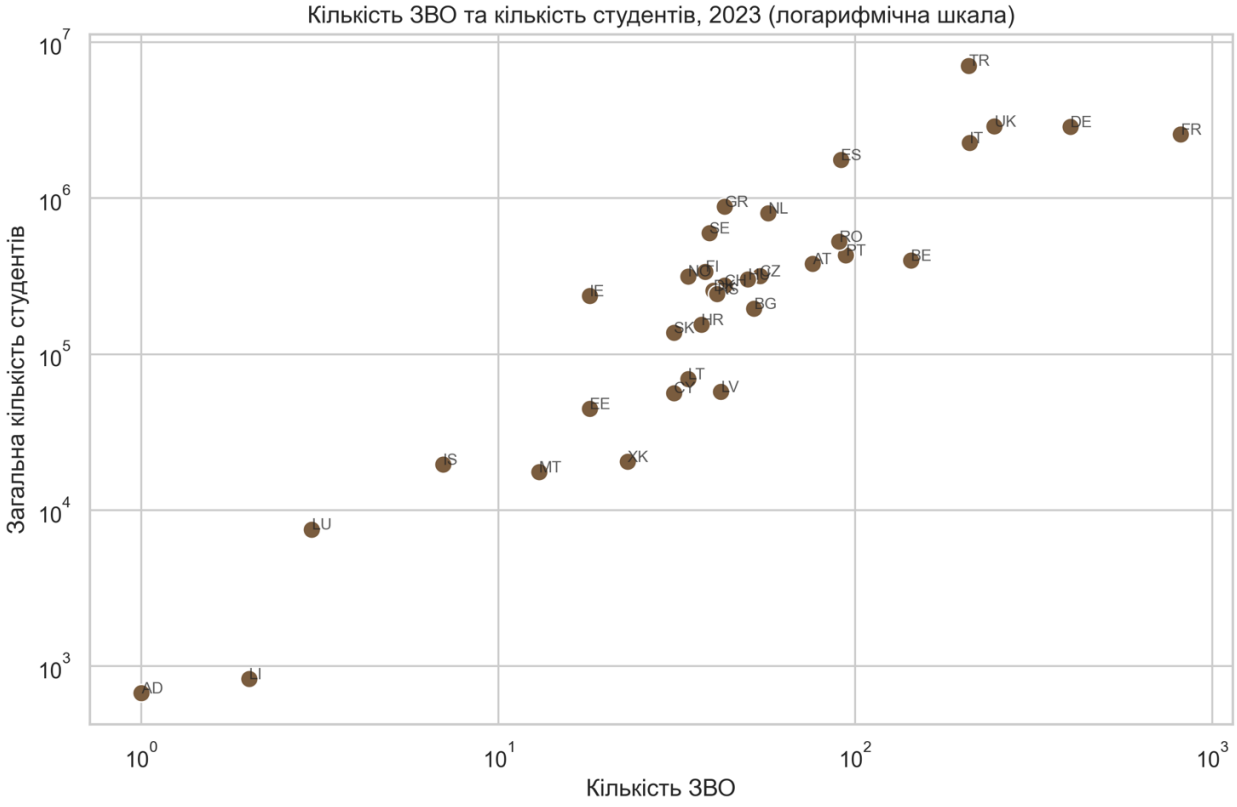

Рисунок 2.6 – діаграма розсіювання

На діаграмі розсіювання показано зв’язок між кількістю закладів вищої освіти та загальною кількістю студентів у країнах за останній доступний рік. Кожна точка відповідає окремій країні. На рисунку осі представлені в логарифмічному масштабі, що дозволяє одночасно порівнювати країни з малими та великими системами вищої освіти.

Графіки демонструють позитивний зв’язок: країни з більшою кількістю ЗВО переважно мають більшу загальну кількість студентів. Водночас точки не формують єдиної чіткої лінії, що свідчить про відмінності в середньому розмірі закладів між країнами.

Країни, розташовані вище від загальної тенденції, мають відносно велику кількість студентів за наявної кількості ЗВО. Це може вказувати на переважання великих закладів. Країни нижче загальної тенденції характеризуються меншою кількістю студентів на один заклад, що може свідчити про більш фрагментовану систему вищої освіти.

Отже, кількість ЗВО пов’язана із загальним масштабом системи вищої освіти, однак сама по собі не повністю пояснює відмінності в кількості студентів. Для детальнішої інтерпретації необхідно додатково враховувати середній та медіанний розмір закладу, а також повноту даних для кожної країни. Виявлений зв’язок є асоціацією та не підтверджує причинного впливу кількості ЗВО на кількість студентів.

Отримані результати показують, що структура студентів суттєво відрізняється між країнами, що відображає особливості національних освітніх моделей, тривалість програм, доступність наступних рівнів освіти, а також правила статистичної звітності.

Рівень інтернаціоналізації оцінювався за загальною кількістю іноземних студентів та їхньою часткою в загальному студентському контингенті. Середня частка іноземних студентів серед країн із доступними даними становила 16,8%, а медіанна – 11,1%. Різниця між середнім і медіанним значеннями пояснюється наявністю декількох країн із надзвичайно високим рівнем інтернаціоналізації:

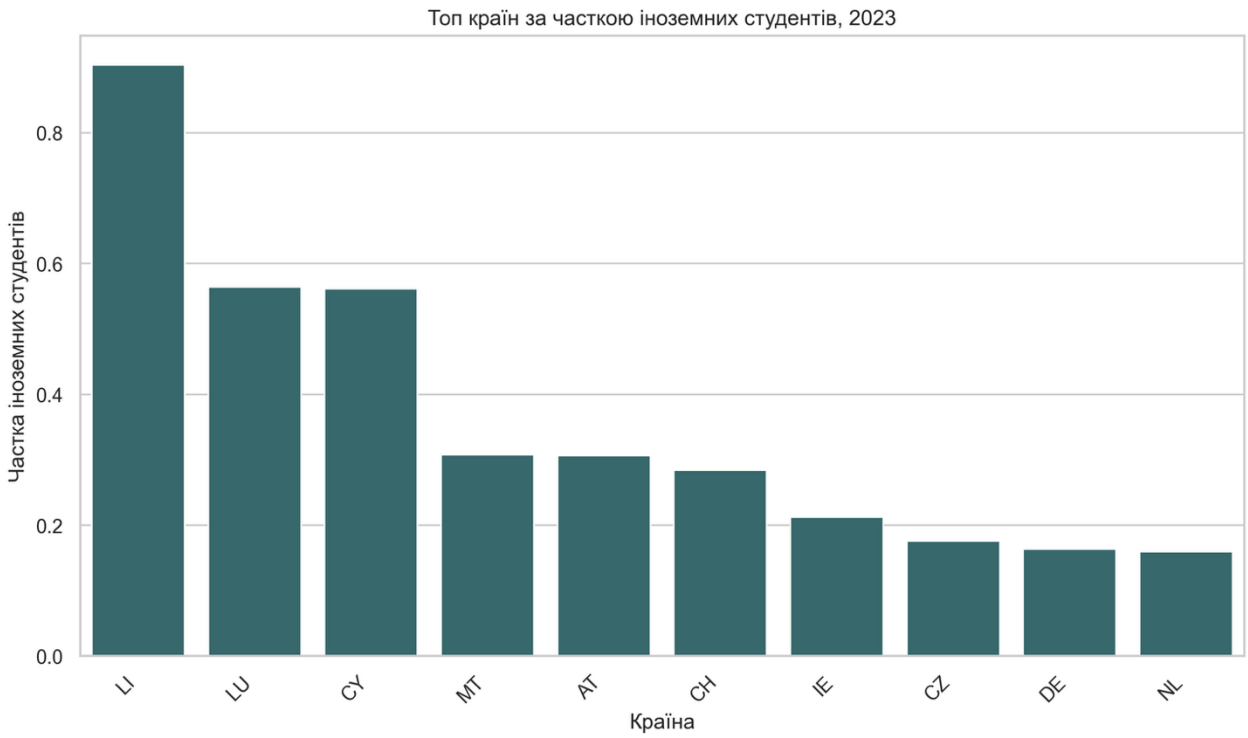

Рисунок 2.7 – Стовпчаста діаграма

- Ліхтенштейн – 90,4%;
- Люксембург – 56,4%;
- Кіпр – 56,2%;
- Мальта – 30,8%;
- Австрія – 30,7%.

Високі значення у малих країнах пояснюються їхньою географічною та економічною відкритістю, невеликим внутрішнім ринком, багатомовністю та орієнтацією ЗВО на міжнародний простір.

Між часткою іноземних студентів і часткою студентів рівня ISCED 8 спостерігався сильний позитивний зв’язок (близько 0,80), а з часткою студентів ISCED 7 – помірний (приблизно 0,55). Це вказує на те, що іноземні студенти більшою мірою орієнтовані на магістерський та докторський рівні освіти. Проте виявлені зв’язки є лише статистичними асоціаціями і не доводять причинно-наслідкового впливу.

Кадрове забезпечення оцінювалося за співвідношенням кількості студентів до академічного та загального персоналу ЗВО.

Медіанне співвідношення студентів до академічного персоналу становило близько 13,9 студента на одного працівника, тоді як середнє значення дорівнювало 24,2. Значне перевищення середнього над медіаною пов’язане з наявністю аномально високих значень в окремих країнах. Для співвідношення студентів до загального персоналу середнє значення становило 7,8, а медіанне – 7,4 студента на одного працівника. Найвищі значення цього показника серед доступних даних спостерігалися в Італії та на Кіпрі.

Показники кадрового забезпечення необхідно інтерпретувати обережно, оскільки вони суттєво залежать від повноти обліку персоналу (зокрема використання еквівалента повної зайнятості – FTE), національних правил звітності та структури закладів.

Розподіл ЗВО за стандартизованою категорією institution_category_std показує, що у 2023 році найбільшу частку вибірки становили заклади категорії «Other» – 1 354 установи, або 42,8%. До категорії «University» належали 1 214 закладів (38,4%), а до «University of applied sciences» – 556 закладів (17,6%). Для 39 установ (1,2%) категорію не було зазначено. Водночас в університетах навчалося 81,4% усіх студентів вибірки, хоча вони становили лише 38,4% ЗВО. Це свідчить, що університети в середньому є значно більшими за кількістю студентів порівняно з іншими типами закладів.

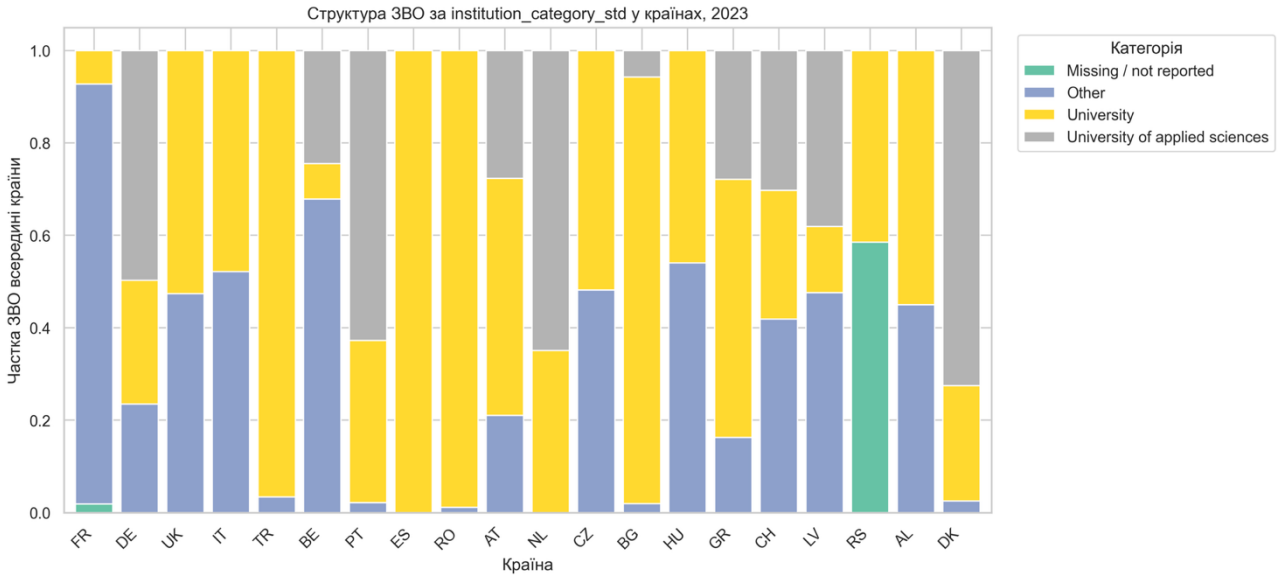

Рисунок 2.8 – Стовпчаста діаграма

Діаграма демонструє структуру ЗВО у 20 країнах із найбільшою кількістю закладів станом на 2023 рік. Університети є найбільшою категорією у 9 країнах, зокрема їхня частка становить 100% в Іспанії, 98,9% у Румунії та 96,6% у Туреччині. У Франції переважають заклади категорії «Other» – 90,9%, а в Бельгії їхня частка дорівнює 67,8%. Університети прикладних наук домінують у Данії – 72,5%, Нідерландах – 64,9% та Португалії – 62,8%. Для Сербії категорія не зазначена для 58,5% ЗВО, тому інтерпретація її структури є обмеженою. Загалом графік свідчить про суттєві відмінності в інституційній організації систем вищої освіти європейських країн.

Для міжнародного порівняння використано фінансові показники, скориговані за паритетом купівельної спроможності (далі – PPP). Основним показником виступали поточні витрати в розрахунку на одного студента.

Середні витрати на одного студента становили приблизно 11,96 тис. одиниць PPP, а медіанні – 10,50 тис. Найвищі значення спостерігалися в Люксембурзі (32,8 тис. одиниць PPP), Швейцарії (30,9 тис.), Ліхтенштейні (19,3 тис.), Данії (17,5 тис.) та Нідерландах (16,3 тис.).

Середня частка іноземних студентів у групі країн із вищими витратами на одного студента становила приблизно 27,0%, тоді як у групі з нижчими

витратами – 16,5%. Кореляція між витратами на одного студента та часткою іноземних студентів була помірно позитивною і становила близько 0,45.

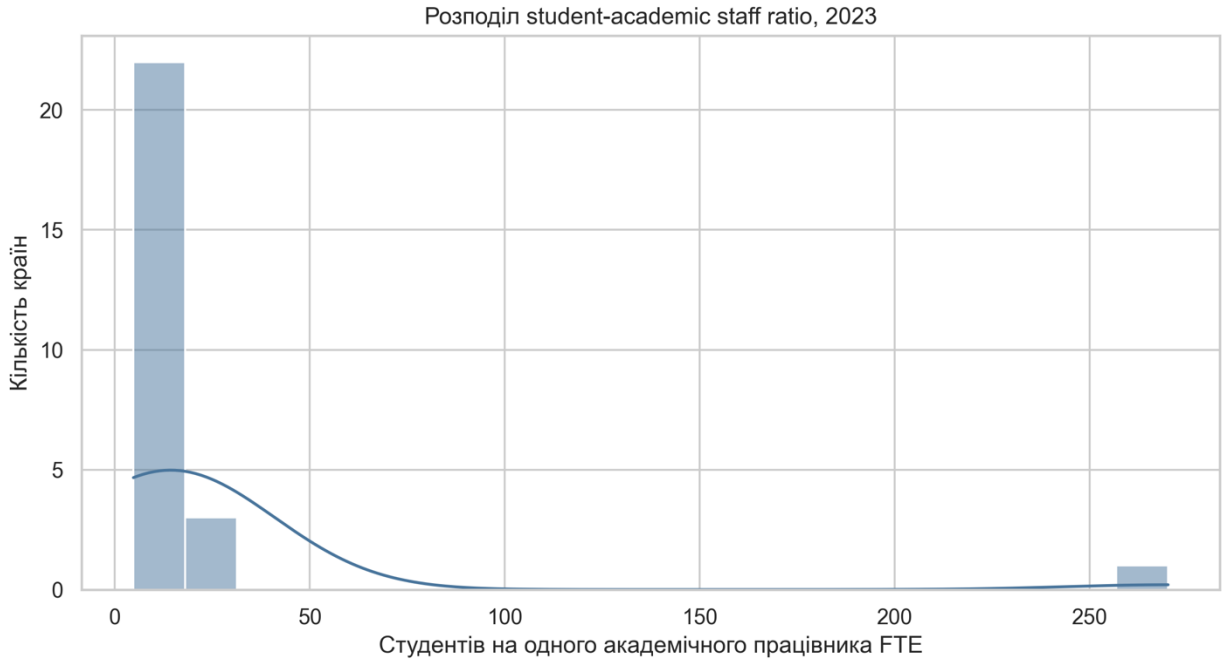

Рисунок 2.9 - Стовпчаста діаграма

Порівняльний аналіз фінансових результатів має суттєві обмеження через значний обсяг пропущених даних у 2023 році: показник витрат на одного студента був відсутній для 31,4% країн, доходів – для 34,3%, а витрат на дослідження і розробки (R&D) – для 45,7% країн.

Концентрація студентів оцінювалася за коефіцієнтом Джині та частками студентів, які навчаються у найбільших одному, п’яти та десяти ЗВО країни.

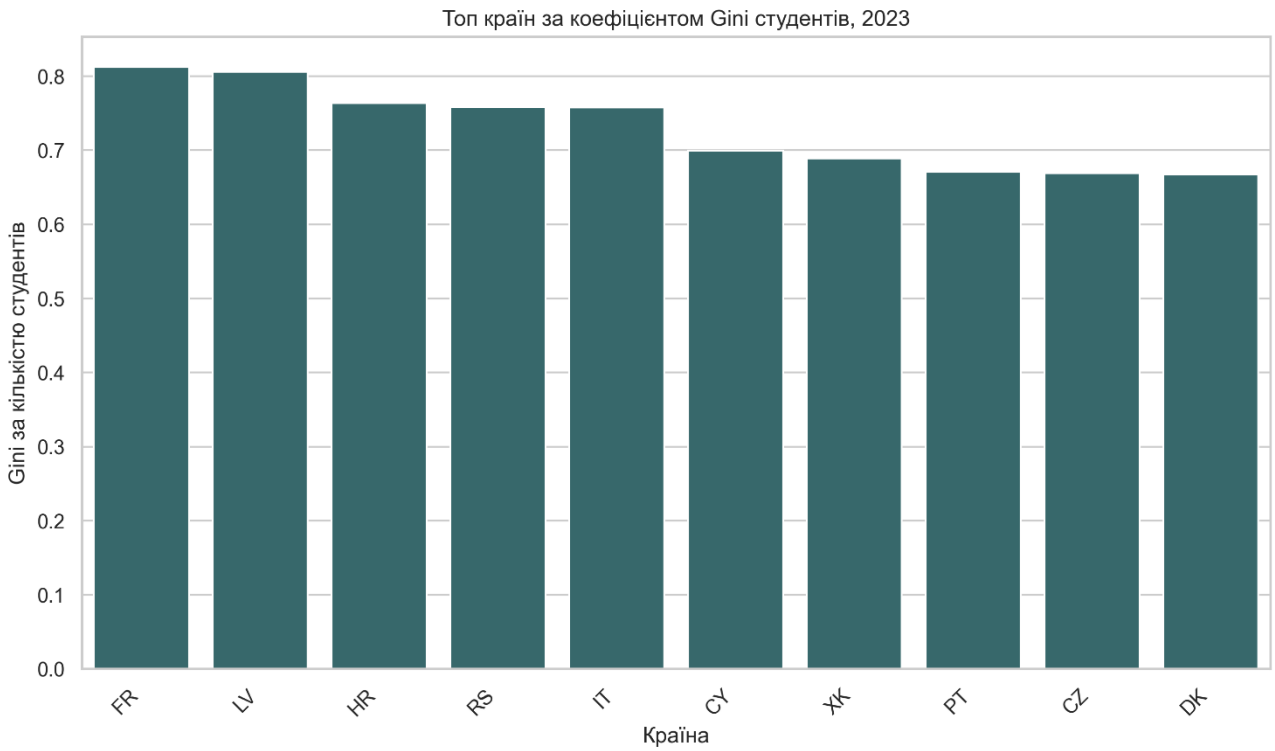

Рисунок 2.10 – Стовпчаста діаграма

Середнє значення коефіцієнта Джині становило 0,594, а медіанне – 0,640. Перевищення медіани над середнім свідчить, що для більшості досліджуваних країн характерна висока нерівномірність розподілу студентів між закладами, тоді як відносно низька концентрація в окремих країнах зміщує загальне середнє значення ліворуч. Найвищі значення коефіцієнта Джині спостерігалися у Франції (0,813), Латвії (0,806), Хорватії (0,764), Сербії (0,759) та Італії (0,758).

У цих країнах студентський контингент значною мірою зосереджений у відносно невеликій кількості великих університетських центрів.

У середньому по вибірці найбільший заклад країни охоплював близько 25,0% студентів, п’ять найбільших – 51,5%, а десять найбільших – 66,9%. Для малих освітніх систем значення часток найбільших закладів закономірно наближаються до 100%, тому такі показники необхідно аналізувати лише в поєднанні із загальною кількістю ЗВО. Між кількістю закладів і часткою студентів у десяти найбільших установах спостерігався очікуваний негативний зв’язок на рівні –0,68.

Проведений описовий статистичний аналіз засвідчив глибоку неоднорідність європейських систем вищої освіти за всіма досліджуваними вимірами (масштаб, структура, кадри, фінансування, інтернаціоналізація та рівень ринкової концентрації).

Результати підтверджують необхідність одночасного використання системи абсолютних і відносних показників, оскільки аналіз виключно агрегованих даних (наприклад, загальної кількості студентів) нівелює структурні особливості національних систем. Виявлені закономірності та статистичні зв'язки створюють підґрунтя для подальшого тестування дослідницьких гіпотез та проведення факторного аналізу країн.

РОЗДІЛ 3 СТАТИСТИЧНИЙ АНАЛІЗ СИСТЕМ ВИЩОЇ ОСВІТИ

3.1. Статистичний аналіз та перевірка гіпотези щодо інтернаціоналізації вищої освіти

Для оцінювання інтернаціоналізації систем вищої освіти використано частку іноземних студентів у загальній кількості студентів, яку розраховано за формулою:

$$ForeignStudentsShare_i = \frac{ForeignStudents_i}{TotalStudents_i}, \quad (3.1)$$

де $ForeignStudents_i$ – кількість іноземних студентів у країні;

$TotalStudents_i$ – загальна кількість студентів у країні.

Метою аналізу було визначення зв’язку між рівнем інтернаціоналізації та ресурсністю системи, структурою студентів за рівнями ISCED і розміром системи вищої освіти. Для цього сформульовано систему нульових та альтернативних гіпотез:

$$H_0: \rho = 0, \quad (3.2)$$

тобто статистично значущий зв’язок між часткою іноземних студентів та іншими характеристиками системи відсутній;

$$H_1: \rho \neq 0, \quad (3.3)$$

тобто між показниками існує статистично значущий зв’язок.

Аналіз виконано на основі агрегованих даних ETER для 32 країн за останній доступний 2023 рік. Перед проведенням розрахунків було перевірено унікальність спостережень на рівні «країна–рік», допустимість значень часток і

наявність пропусків. Показник витрат на одного студента був доступний лише для 22 країн, тому фінансові результати мають обмежену порівнюваність.

Рейтинговий аналіз показав значні відмінності між країнами. Найбільша частка іноземних студентів спостерігалася у Ліхтенштейні (90,36 %), Люксембурзі (56,44 %) та на Кіпрі (56,16 %). Високі показники також мали Мальта, Австрія та Швейцарія (Рис. 3.1).

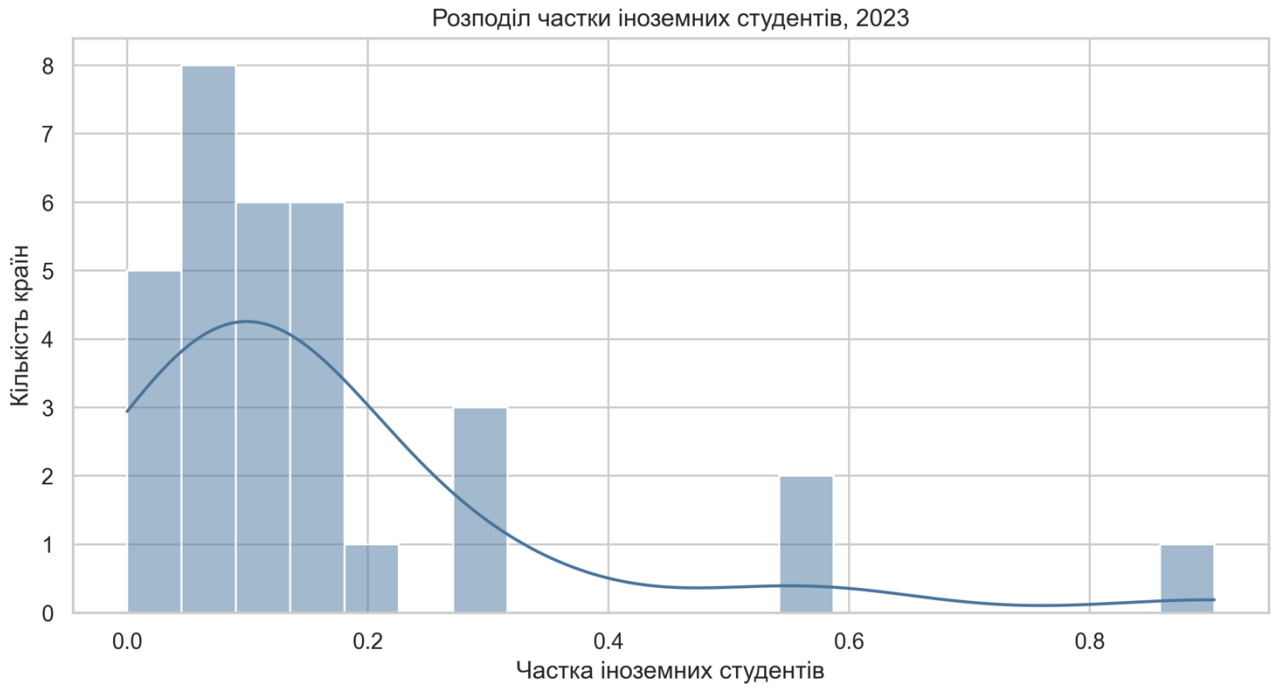

Рисунок 3.1 – Рейтинг країн за часткою іноземних студентів у 2023 році

Водночас рейтинг за абсолютною кількістю іноземних студентів відрізняється від рейтингу за їхньою часткою. Наприклад, Німеччина має нижчу частку іноземних студентів, ніж малі європейські країни, але посідає перше місце за їхньою абсолютною кількістю. Отже, для оцінювання інтернаціоналізації необхідно враховувати як відносні, так і абсолютні показники.

Для оцінювання статистичних зв’язків використано коефіцієнти кореляції Пірсона *r* та Спірмена

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})\,(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2}}, \tag{3.4}$$

$$r_s = 1 - \frac{6\sum_{i=1}^{n} d_i^2}{n(n^2-1)}, \tag{3.5}$$

де y_i, x_i – значення порівнюваних показників;

$\bar{x}, \bar{y}$ – середні значення показників;

d_i – різниця між рангами показників;

n – кількість спостережень.

Між часткою іноземних студентів і часткою студентів рівня ISCED 7 виявлено статистично значущий позитивний зв'язок (Рис. 3.2). Коефіцієнт Пірсона становив $r = 0{,}549$ при $p = 0{,}001$, а коефіцієнт Спірмена – $r_s = 0{,}527$ при $p = 0{,}002$. Отже, країни з більшою часткою студентів магістерського рівня переважно мають вищий рівень інтернаціоналізації.

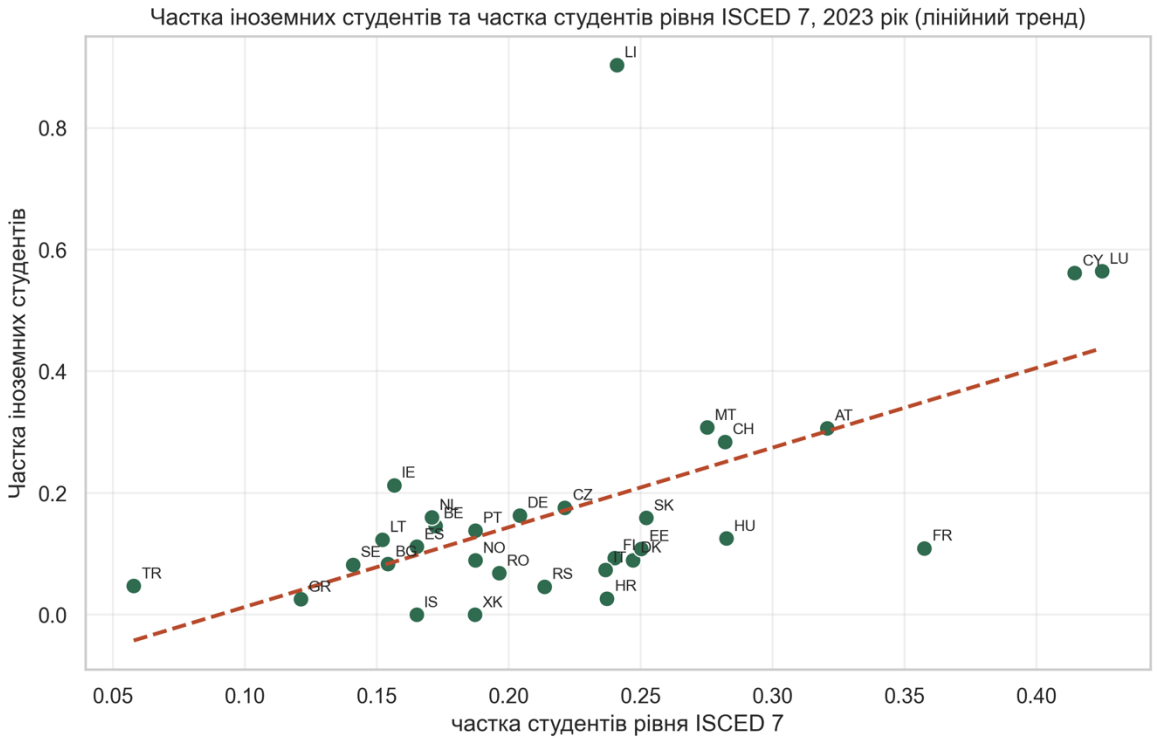

Рисунок 3.2 – Зв'язок між часткою студентів ISCED 7 і часткою іноземних студентів

Для рівня ISCED 8 також встановлено позитивний статистично значущий зв'язок (Рис. 3.3). Коефіцієнт Пірсона становив $r = 0{,}800$ при $p < 0{,}001$, а коефіцієнт Спірмена – $r_s = 0{,}459$ при $p = 0{,}009$. Різниця між коефіцієнтами може свідчити про вплив окремих країн із аномально високими значеннями показників.

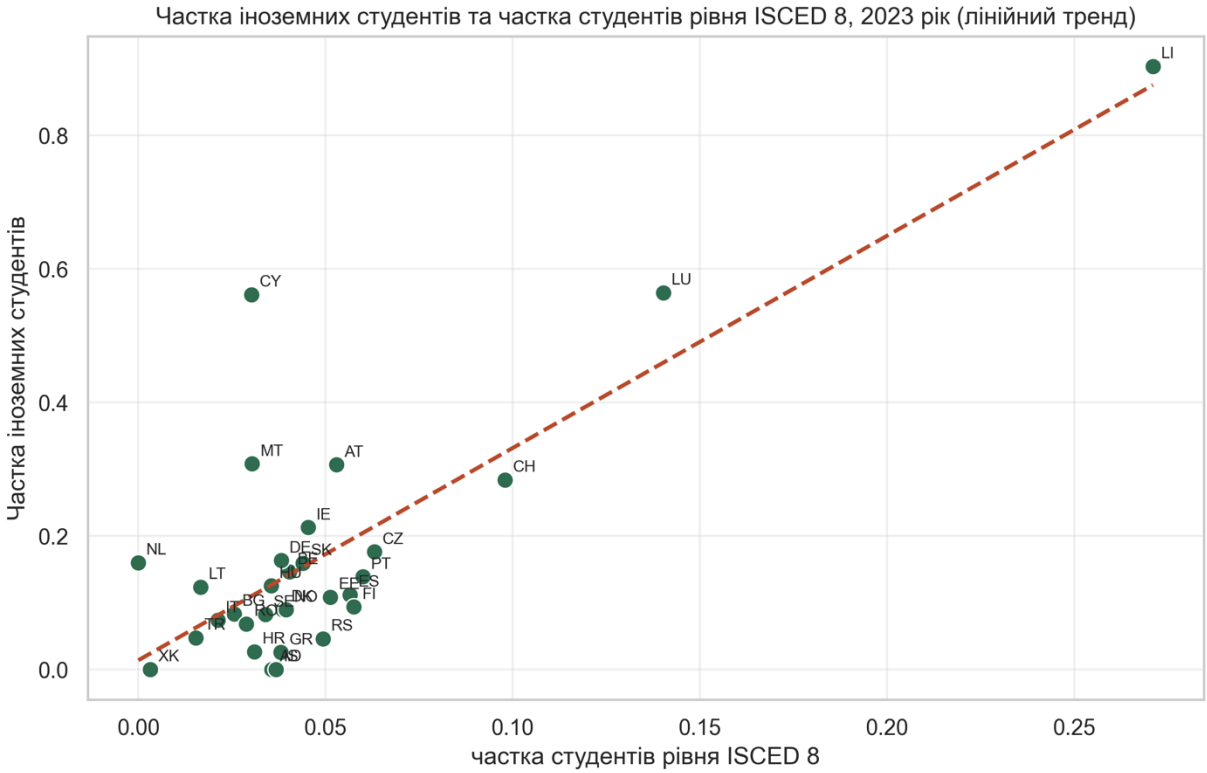

Рисунок 3.3 – Зв'язок між часткою студентів ISCED 8 і часткою іноземних студентів

Аналіз фінансової ресурсності виявив помірний позитивний зв'язок між часткою іноземних студентів і витратами на одного студента (Рис. 3.4). Коефіцієнт Пірсона становив $r = 0{,}450$ при $p = 0{,}035$. Проте коефіцієнт Спірмена дорівнював $r_s = 0{,}294$ при $p = 0{,}184$ і не був статистично значущим.

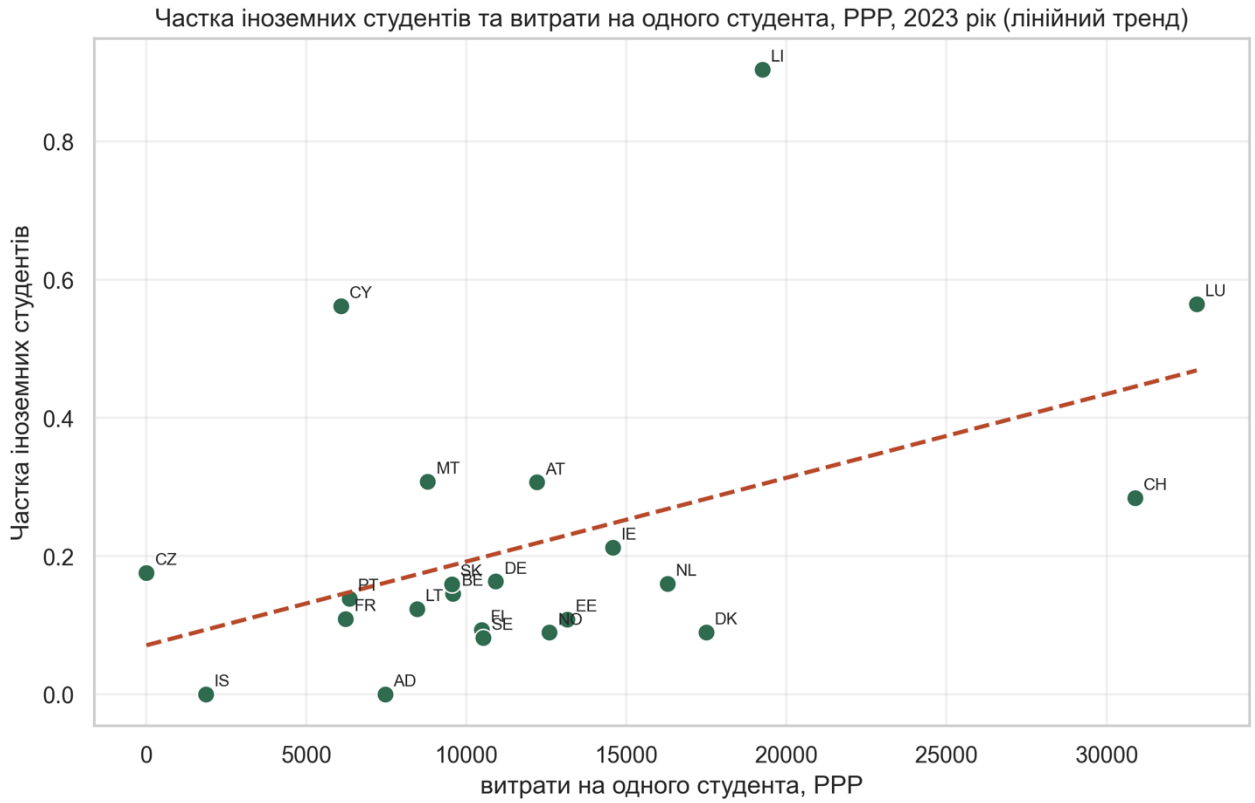

Рисунок 3.4 – Зв’язок між витратами на одного студента та часткою іноземних студентів

Статистично значущих зв’язків між часткою іноземних студентів і розміром системи не виявлено. Кореляції із середнім розміром ЗВО, кількістю студентів та кількістю закладів вищої освіти були слабкими й мали $p > 0{,}05$

Для додаткового аналізу країни було розподілено на три групи за рівнем інтернаціоналізації (Рис. 3.5). Медіанна частка ISCED 7 збільшувалася від 17,61 % у групі з низькою інтернаціоналізацією до 25,20 % у групі з високою. Медіанна частка ISCED 8 зростала від 3,09 % до 4,54 %, а медіанні витрати на одного студента – від 7 463,77 до 12 210,34 PPP.

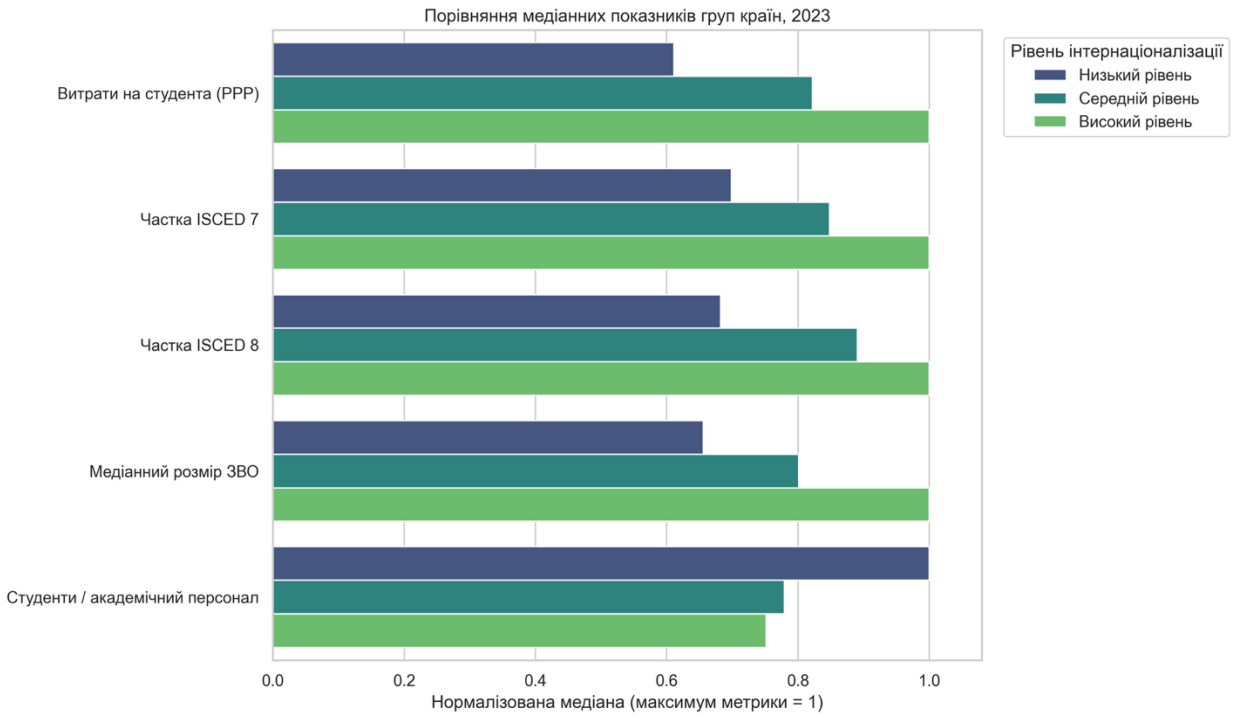

Рисунок 3.5 – Порівняння медіанних показників груп країн за рівнем інтернаціоналізації

Для перевірки стійкості результатів аналіз було повторено після виключення країн, у яких представлено менше ніж десять ЗВО. Позитивний зв'язок із часткою ISCED 7 зберігся та залишився статистично значущим: $r_s = 0{,}444$ при $p = 0{,}018$ (Рис. 3.6). Зв'язок із ISCED 8 послабився до $r_s = 0{,}357$ при $p = 0{,}068$, а зв'язок із витратами на одного студента практично зник. Зведені результати порівняння наведено в таблиці 3.2.

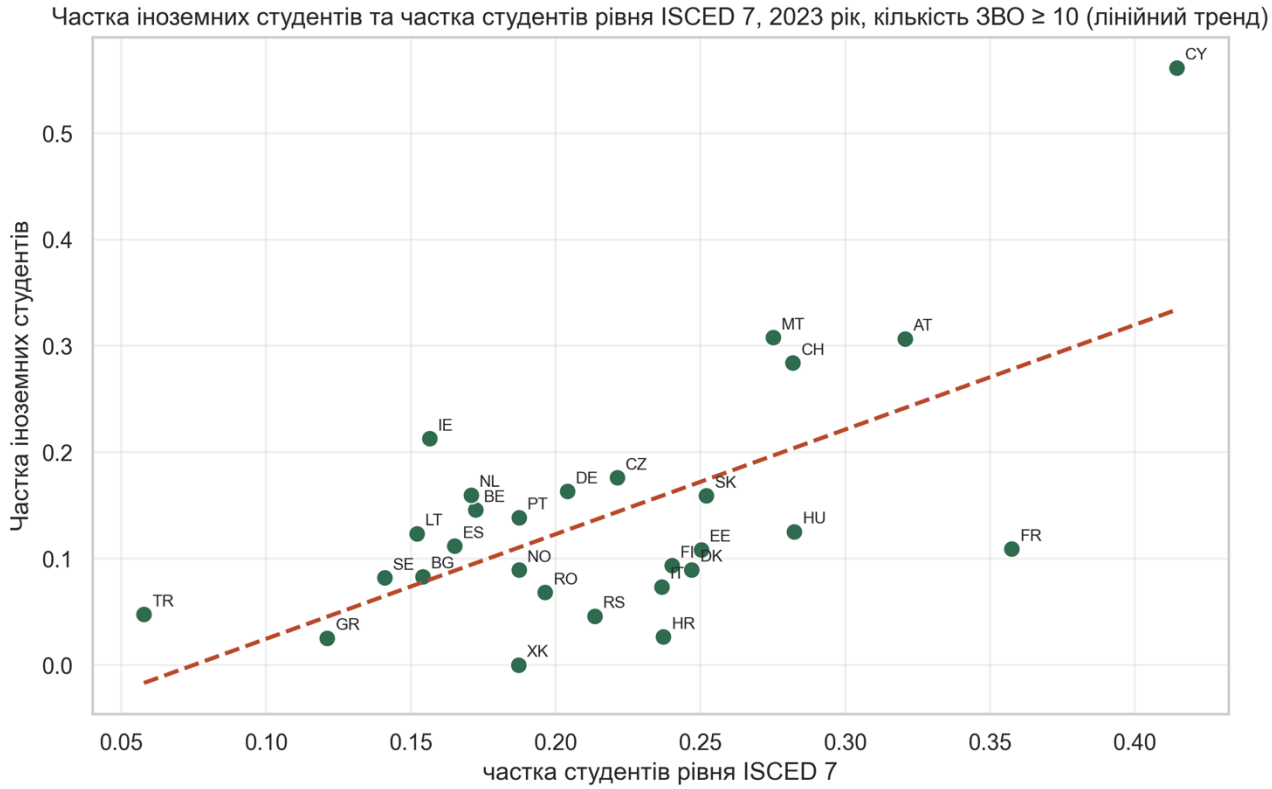

Рисунок 3.6 – Зв’язок між ISCED 7 і часткою іноземних студентів після виключення країн із менш ніж 10 ЗВО

Проведений аналіз дає підстави частково відхилити нульову гіпотезу. Найбільш стійкий позитивний зв’язок виявлено між часткою іноземних студентів і часткою студентів рівня ISCED 7. Зв’язки з ISCED 8 та фінансовою ресурсністю залежать від складу вибірки. Достатніх доказів зв’язку між інтернаціоналізацією та загальним розміром системи вищої освіти не отримано.

Отримані результати необхідно інтерпретувати з урахуванням пропусків у фінансових показниках, впливу малих країн, відмінностей у національних правилах звітності та використання лише одного часового зрізу. Виявлені статистичні асоціації не доводять наявність причинно-наслідкових зв’язків.

3.2. Оцінювання концентрації та нерівномірності розподілу студентів

У цьому підрозділі перевірено гіпотезу про наявність зв'язку між масштабами національних систем вищої освіти та нерівномірністю розподілу студентів між закладами вищої освіти. Масштаб системи характеризувався кількістю ЗВО, загальною кількістю студентів, а також середнім, медіанним і максимальним розміром закладу. Для оцінювання концентрації використано частки студентів у найбільшому, п'яти та десяти найбільших ЗВО і коефіцієнт Джині.

Робоча гіпотеза передбачала, що системи, у яких невелика кількість великих закладів навчає значну частину студентів, мають вищу нерівномірність розподілу. Окремо перевірено, чи супроводжується висока концентрація більшим розривом між середнім і медіанним розміром ЗВО. Такий розрив свідчить про наявність у системі великої кількості малих закладів та обмеженої кількості дуже великих установ.

Частку студентів у найбільших закладах розраховано за формулою:

$$TopNShare = \frac{\sum_{i=1} N\ Students_i}{\sum_{i=1} K\ Students_i}, \tag{3.6}$$

де N – кількість найбільших ЗВО;

K – загальна кількість закладів країни, для яких доступні дані про студентів.

Коефіцієнт Джині набуває значень від 0 до 1: значення, близьке до 0, відповідає рівномірному розподілу студентів, а більше значення вказує на вищу нерівномірність.

$$G = \frac{2\sum_{i=1}^{n} i\ x_i}{n\sum_{i=1}^{n} x_i} - \frac{n+1}{n}, \tag{3.7}$$

де x_i – кількість студентів у і-му ЗВО після впорядкування за зростанням;

n – кількість ЗВО.

Для коректного розрахунку використано лише невід'ємні значення кількості студентів. Якщо студентські дані були відсутні або їх сума дорівнювала нулю, концентраційні показники не розраховувалися. Для оцінювання статистичних зв'язків застосовано ранговий коефіцієнт кореляції Спірмена, оскільки розподіли досліджуваних показників є асиметричними та містять крайні значення.

Якість даних і склад вибірки

Аналіз проведено для останнього доступного року – 2023 року. У таблиці останнього року представлено 35 країн. Для 34 країн доступні значення коефіцієнта Джині та часток найбільших ЗВО. Для Албанії концентраційні показники за 2023 рік не розраховано через відсутність студентських даних на рівні окремих закладів.

Для 31 країни забезпечено повне покриття даними про студентів. Часткове покриття мають Бельгія – 43,4 %, Люксембург – 66,7 % і Туреччина – 99,0 %. Тому результати для Бельгії та Люксембургу необхідно інтерпретувати з особливою обережністю. Перевірка таблиць country_year_profiles і concentration_metrics не виявила дублікатів за ключем «країна–рік».

Загальна характеристика концентрації студентів

За результатами описової статистики середнє значення коефіцієнта Джині серед 34 країн становило 0,594, а медіанне – 0,640. Це вказує на помітну нерівномірність розподілу студентів між ЗВО в більшості досліджуваних систем. Середня частка найбільшого закладу становила 25,0 %, п'яти найбільших закладів – 51,5 %, а десяти найбільших – 66,9 %. Водночас значна варіативність цих показників між країнами свідчить про суттєві відмінності в організації національних систем вищої освіти.

Найвищі значення коефіцієнта Джині у 2023 році спостерігалися у Франції – 0,813, Латвії – 0,806, Хорватії – 0,764, Сербії – 0,759 та Італії – 0,758.

Найнижчі значення серед систем із щонайменше п’ятьма закладами зафіксовано в Ірландії – 0,350, Фінляндії – 0,400, Іспанії – 0,491, Словаччині – 0,508 та Швеції – 0,513.

Високе значення коефіцієнта Джині не завжди означає високу частку студентів у п’яти або десяти найбільших ЗВО. Наприклад, Франція має найвищий коефіцієнт Джині, однак п’ять найбільших закладів акумулюють лише 12,7 % студентів, а десять найбільших – 22,2 %. Це пояснюється великим масштабом французької системи, яка включає 816 закладів, і суттєвою різницею між їхніми розмірами. Середній розмір французького ЗВО становить 3143 студенти, тоді як медіанний – лише 489 студентів.

Іншу структуру демонструють Латвія, Хорватія та Сербія. У Латвії п’ять найбільших ЗВО охоплюють 77,7 % студентів, у Хорватії – 75,0 %, у Сербії – 73,2 %. Частка десяти найбільших закладів у цих країнах становить відповідно 87,6 %, 86,7 % та 84,0 %. Отже, у відносно невеликих системах високий рівень концентрації безпосередньо пов’язаний із домінуванням обмеженої кількості великих установ.

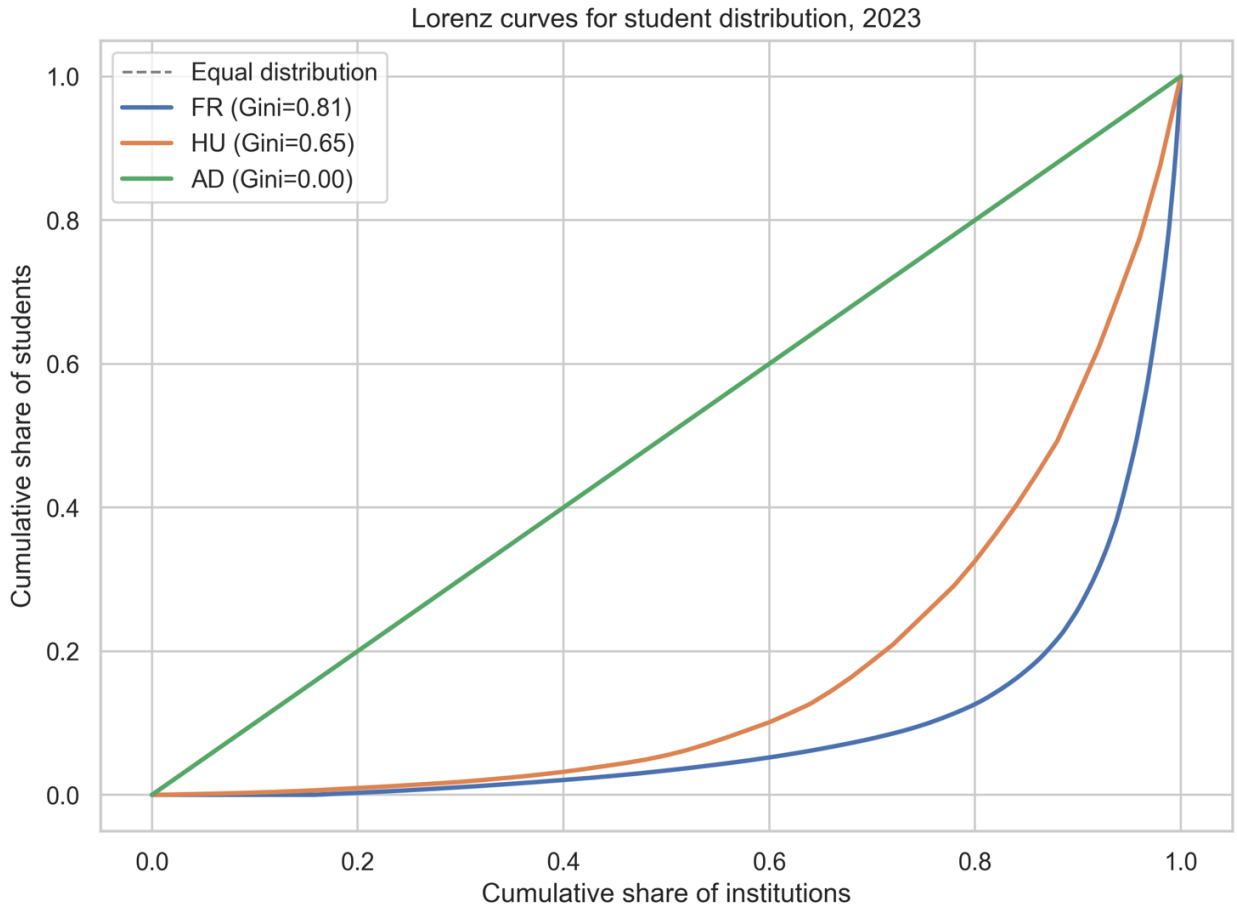

Рисунок 3.7 – Криві Лоренца для країн із різними рівнями концентрації

Криві Лоренца візуально підтверджують відмінності між країнами. Чим сильніше крива відхиляється від діагоналі рівномірного розподілу, тим більша частка студентів зосереджена в обмеженій кількості ЗВО. Графік доповнює коефіцієнт Джині, оскільки показує не лише підсумковий рівень нерівномірності, а й форму розподілу студентів між закладами.

Залежність концентрації від масштабів системи

Кореляційний аналіз показав, що частки найбільших ЗВО істотно залежать від кількості закладів у системі. Зв'язок між кількістю ЗВО та часткою найбільшого закладу є сильним від'ємним: $\rho = -0{,}743$, $p < 0{,}001$. Для частки п'яти найбільших закладів коефіцієнт становить $\rho = -0{,}848$, а для десяти найбільших – $\rho = -0{,}918$; в обох випадках $p < 0{,}001$.

Такий результат має очікуване структурне пояснення: у країнах із малою кількістю ЗВО п'ять або десять найбільших закладів можуть охоплювати майже всю систему. Наприклад, для Андорри, Ліхтенштейну та Люксембургу частка

десяти найбільших ЗВО дорівнює 100 %, оскільки загальна кількість закладів у цих країнах не перевищує десяти. Тому показники top_5_share і top_10_share не можна інтерпретувати окремо від інформації про кількість ЗВО.

Між кількістю ЗВО та коефіцієнтом Джині виявлено слабший додатний зв'язок: $\rho = 0{,}346$, $p = 0{,}045$. Це означає, що великі за кількістю закладів системи можуть мати вищу загальну нерівномірність розподілу студентів, навіть якщо частки окремих найбільших ЗВО в них є невисокими. Приклад Франції підтверджує, що велика кількість малих установ разом із групою великих закладів може формувати високий коефіцієнт Джині.

Загальна кількість студентів має сильний від'ємний зв'язок із часткою п'яти найбільших ЗВО: $\rho = -0{,}885$, а також із часткою десяти найбільших ЗВО: $\rho = -0{,}938$. Водночас статистично значущого зв'язку із коефіцієнтом Джині не виявлено: $\rho = 0{,}127$, $p = 0{,}473$. Отже, більший загальний контингент зменшує частку студентів, яку акумулює фіксована кількість найбільших закладів, проте сам по собі не визначає загальну нерівномірність розподілу.

Зв'язок концентрації з розривом між середнім і медіанним розміром ЗВО

Найсильніший результат отримано для зв'язку між коефіцієнтом Джині та відношенням середнього розміру ЗВО до медіанного. Коефіцієнт Спірмена становить $\rho = 0{,}880$ при $p < 0{,}001$. Таким чином, країни, у яких середній розмір закладу суттєво перевищує медіанний, зазвичай мають вищу нерівномірність розподілу студентів.

Водночас абсолютний розрив між середнім і медіанним розміром не має статистично значущого зв'язку з коефіцієнтом Джині: $\rho = 0{,}241$, $p = 0{,}170$. Це свідчить про те, що для міжнародних порівнянь відносний показник є інформативнішим за абсолютну різницю, оскільки країни суттєво відрізняються за типовим розміром закладів.

Для часток п'яти та десяти найбільших ЗВО зв'язок із відносним розривом між середнім і медіанним розміром не є статистично значущим: відповідно $\rho = -0{,}054$, $p = 0{,}763$ та $\rho = -0{,}080$, $p = 0{,}655$. Натомість ці показники мають значущі від'ємні кореляції з абсолютним розривом: $\rho = -0{,}588$ для частки

п’яти найбільших ЗВО і ρ = -0,654 для частки десяти найбільших ЗВО. Від’ємний напрямок зв’язку пояснюється тим, що найбільші абсолютні розриви частіше виникають у великих системах, де фіксована кількість провідних закладів охоплює меншу частку всіх студентів.

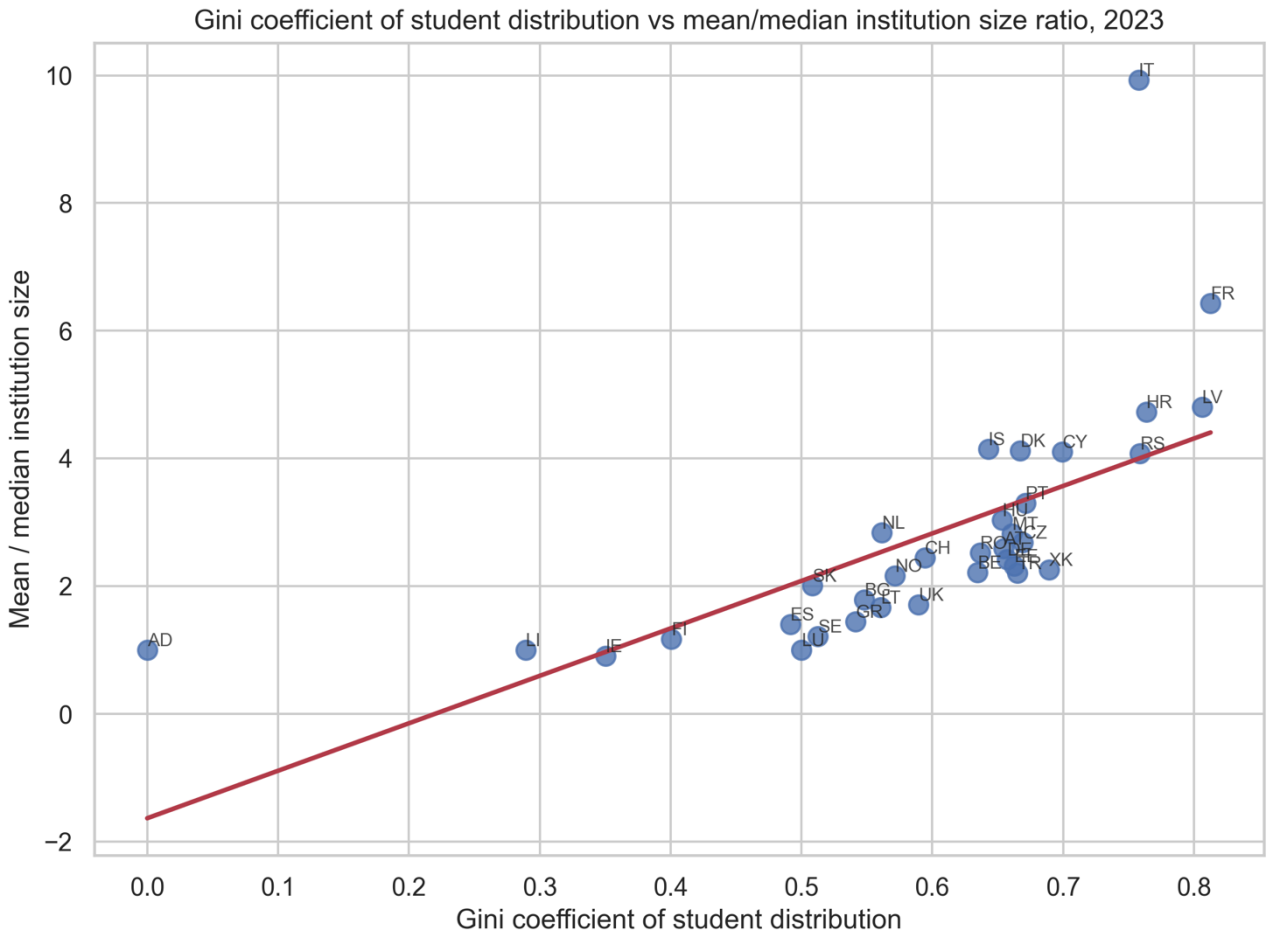

Рисунок 3.8 – Коефіцієнт Джині та відношення середнього розміру ЗВО до медіанного

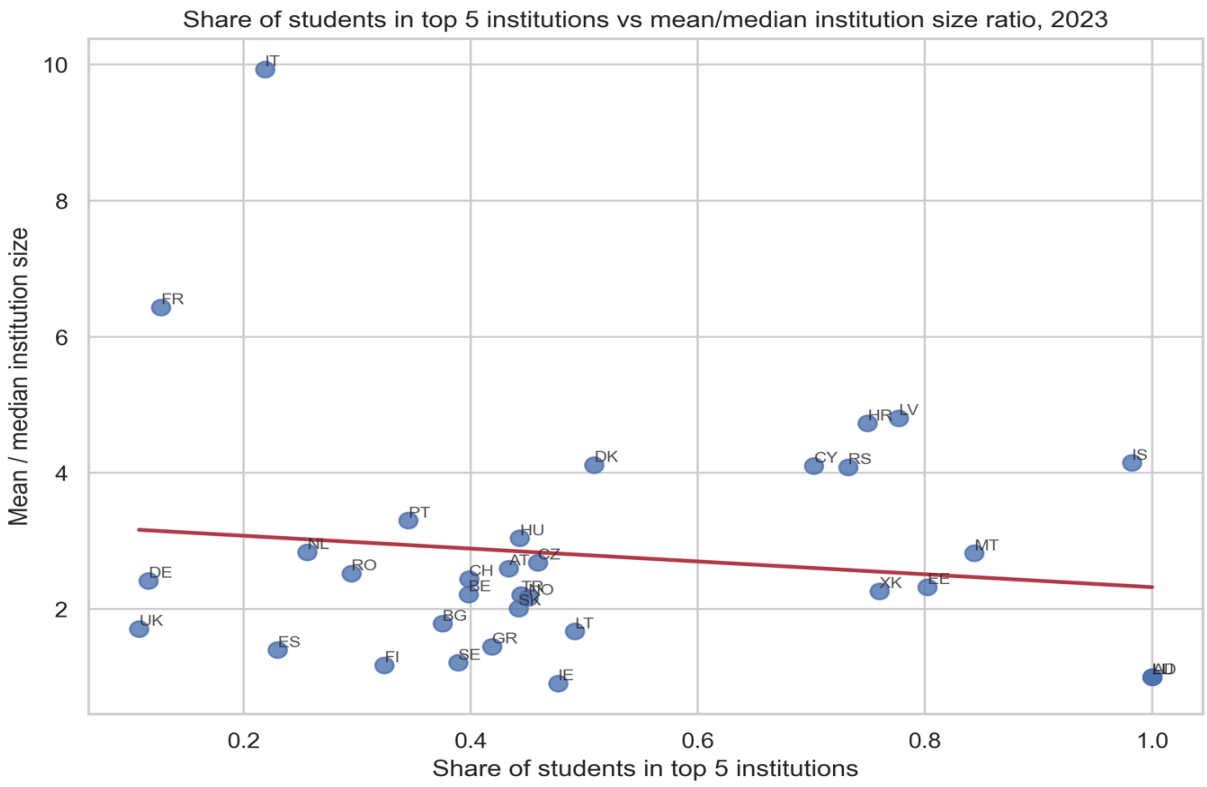

Рисунок 3.9 – Частка студентів у п’яти найбільших ЗВО та відношення середнього розміру до медіанного

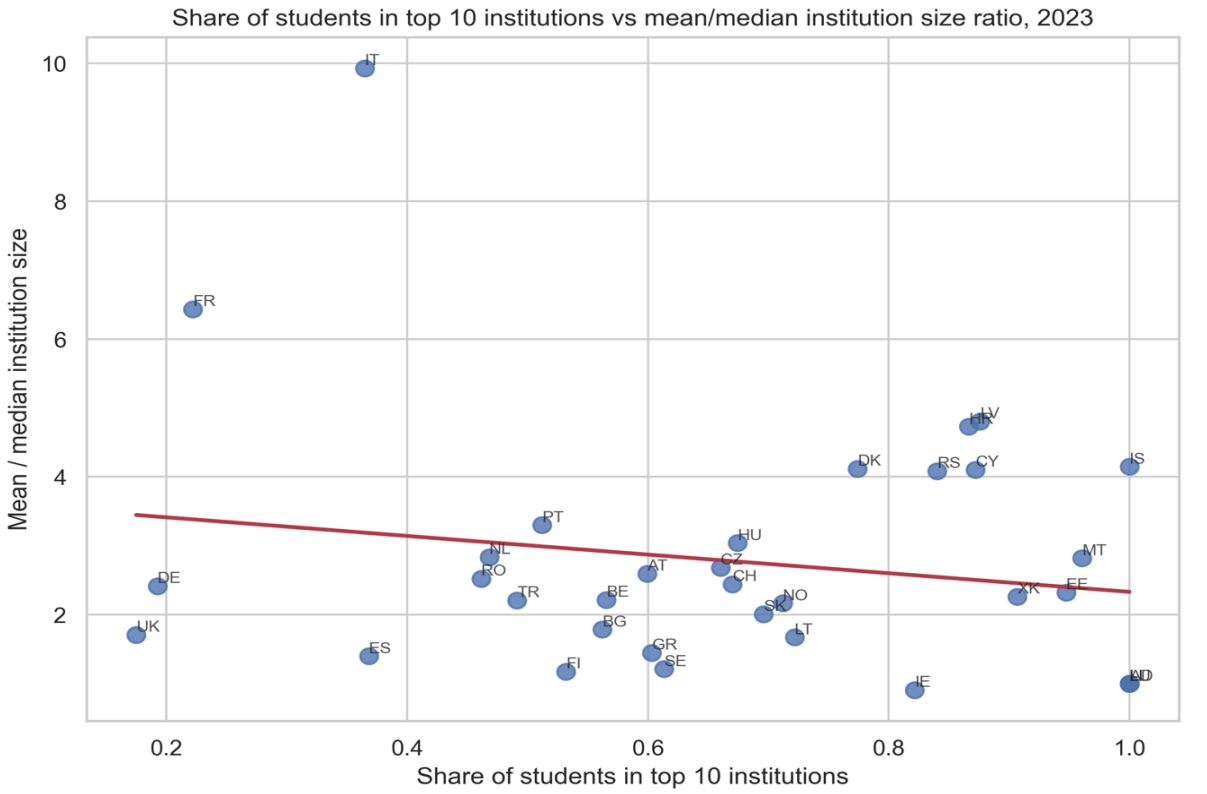

Рисунок 3.10 – Частка студентів у десяти найбільших ЗВО та відношення середнього розміру до медіанного

Розподіли на рівні окремих ЗВО додатково показують, що високий коефіцієнт Джині часто формується через довгий правий хвіст розподілу: у країні функціонує багато малих закладів і декілька установ із дуже великим контингентом. Саме тому середній розмір суттєво перевищує медіанний.

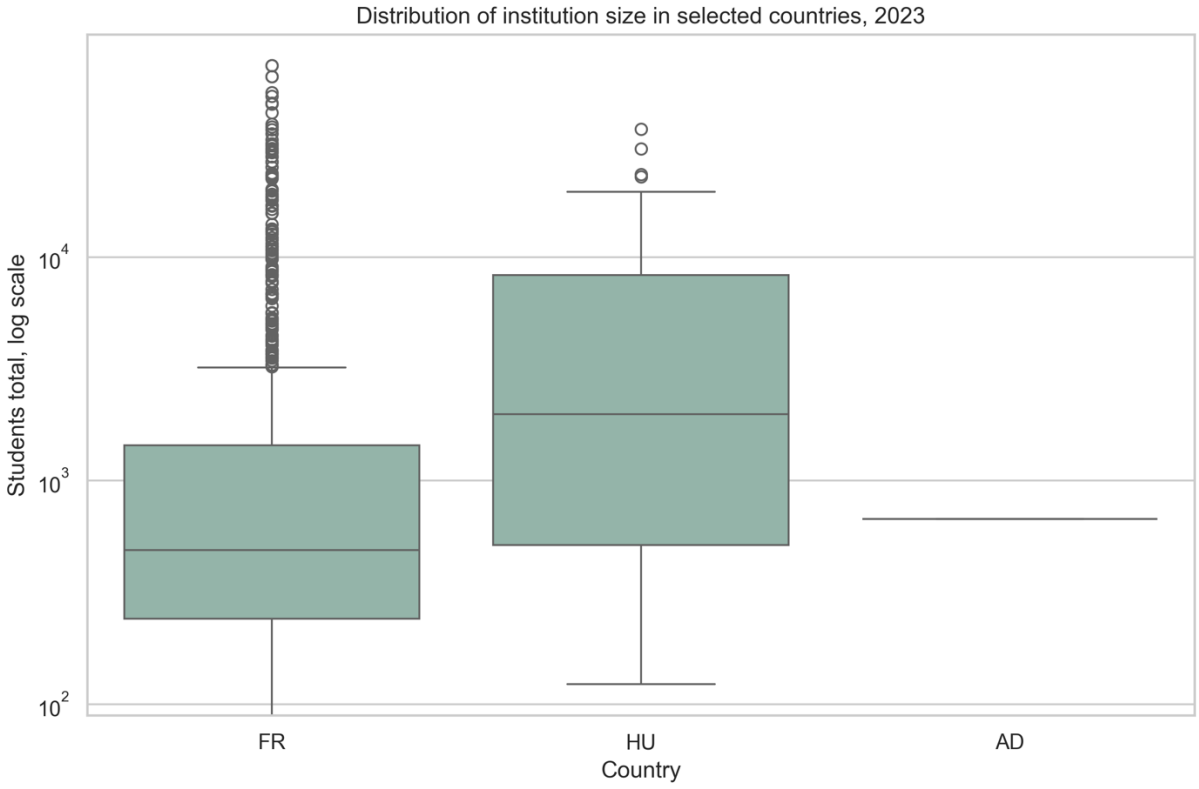

Рисунок 3.11 – Розподіл розміру ЗВО у вибраних країнах

Інтегральна оцінка впливу кількості та розміру закладів.

Для додаткової оцінки спільного зв’язку кількості та розміру закладів із концентрацією побудовано лінійні моделі зі стандартизованими показниками.

$$z_{ij} = \frac{x_{ij} - \bar{x}_j}{s_j} \tag{3.8}$$

Через значну асиметрію розподілу кількість ЗВО та середній розмір закладу було логарифмовано.

$$x_i^* = \ln(1 + x_i) \quad (3.9)$$

Моделі мають описовий характер і не використовуються для формування причинно-наслідкових висновків.

Модель для коефіцієнта Джині пояснює 41,8 % міжкраїнної варіації показника. За одночасного врахування двох факторів стандартизований коефіцієнт логарифма кількості ЗВО є додатним і статистично значущим: $\beta = 0{,}760$, $p < 0{,}001$. Коефіцієнт середнього розміру закладу є від'ємним, але статистично незначущим: $\beta = -0{,}269$, $p = 0{,}116$.

Для часток найбільших ЗВО моделі мають вищу пояснювальну здатність: 69,5 % для частки найбільшого ЗВО та 80,4 % для часток п'яти і десяти найбільших ЗВО. У всіх моделях кількість закладів має сильний від'ємний зв'язок із частками найбільших ЗВО. Це підтверджує, що top-N частки значною мірою відображають масштаб системи, тоді як коефіцієнт Джині краще характеризує загальну нерівномірність розподілу студентів між усіма закладами.

Результати аналізу частково підтверджують гіпотезу про зв'язок рівня концентрації студентів із масштабами систем вищої освіти. Кількість ЗВО має сильний від'ємний зв'язок із частками студентів у найбільшому, п'яти та десяти найбільших закладах. Водночас цей результат значною мірою зумовлений конструкцією показників top-N: у малих системах фіксована кількість закладів охоплює більшу частину всіх студентів.

Коефіцієнт Джині дає інший аспект оцінювання. Він має лише помірний зв'язок із кількістю ЗВО, але дуже сильну додатну асоціацію з відношенням середнього розміру закладу до медіанного. Це свідчить про те, що нерівномірність найбільше проявляється в системах, де поряд із великою кількістю малих установ функціонують окремі дуже великі ЗВО.

Отже, для комплексного оцінювання концентрації доцільно одночасно використовувати коефіцієнт Джині, частки найбільших закладів і показники масштабу системи. Top-N частки характеризують роль найбільших установ, тоді як коефіцієнт Джині відображає нерівномірність усього розподілу. Виявлені

зв’язки є статистичними асоціаціями та не доводять причинного впливу кількості або розміру ЗВО на концентрацію студентів.

До основних обмежень аналізу належать різна повнота студентських даних за країнами, структурна залежність top-N часток від кількості ЗВО, відмінності національних систем звітності та використання даних лише за останній доступний рік.

3.3. Факторний аналіз освітніх систем

Для виявлення узагальнених характеристик, які пояснюють відмінності між системами вищої освіти європейських країн для подальшої кластеризації, було застосовано метод головних компонент (Principal Component Analysis, PCA). Цей метод дає змогу скоротити початковий набір взаємопов’язаних показників до меншої кількості незалежних компонентів без значної втрати інформації.

Метод PCA у роботі використовується як практичний варіант факторного аналізу. На відміну від класичного факторного аналізу, PCA пояснює загальну варіацію показників, а не лише їхню спільну дисперсію. Тому виділені компоненти інтерпретуються як узагальнені статистичні характеристики освітніх систем.

Для аналізу використано дані з файла «country_year_profiles.csv», сформованого на попередніх етапах дослідження. Одиницею спостереження є освітня система окремої країни у певному році. Для забезпечення актуальності порівняння було обрано останній доступний рік – 2023 рік.

У 2023 році дані були доступні для 35 країн. Для PCA використовувалися лише країни, які не мали пропущених значень за всіма обраними показниками. У результаті до повної моделі увійшло 19 країн, тоді як 16 країн було виключено через пропуски. Найбільша частка пропущених значень спостерігалася для

витрат на одного студента за паритетом купівельної спроможності та співвідношення студентів до академічного персоналу. Розподіл пропущених значень наведено в таблиці.

Таблиця 3.1 – Повнота показників, використаних у PCA

Показник	Кількість пропусків	Частка пропусків, %
Витрати на одного студента	11	31,43
Співвідношення студентів і академічного персоналу	9	25,71
Частка іноземних студентів	3	8,57
Частка студентів ISCED 7	2	5,71
Частка студентів ISCED 8	2	5,71
Загальна кількість студентів	1	2,86
Середній розмір закладу	1	2,86
Медіанний розмір закладу	1	2,86
Коефіцієнт Джині	1	2,86

До факторного аналізу було включено 12 показників:

- кількість закладів вищої освіти;
- загальна кількість студентів;
- середній і медіанний розмір закладу;
- частка іноземних студентів;
- співвідношення студентів і академічного персоналу;
- витрати на одного студента за PPS;
- частки студентів рівнів ISCED 7 та ISCED 8;
- частки студентів у п’яти та десяти найбільших закладах;
- коефіцієнт Джині розподілу студентів.

Абсолютні показники розміру системи характеризувалися значною правосторонньою асиметрією. Тому для кількості закладів, загальної кількості

студентів, середнього та медіанного розміру закладу застосовано логарифмічне перетворення:

$$x_i^* = ln(1 + x_i), \tag{3.10}$$

де x_i – початкове значення показника;

x_i^* – трансформоване значення показника.

Після трансформації всі показники були стандартизовані:

$$z_{ij} = \frac{x_{ij} - x_i}{s_j}, \tag{3.11}$$

де x_{ij} – значення показника j для країни i;

x_i – середнє значення показника j;

s_j – стандартне відхилення показника j.

Стандартизація забезпечила порівнюваність показників, представлених у різних одиницях вимірювання.

Виділення головних компонентів

Кожна головна компонента є лінійною комбінацією стандартизованих показників:

$$PC_k = a_{\{1k\}z_1} + a_{\{2k\}z_2} + \cdots + a_{\{pk\}z_p}, \tag{3.12}$$

де PC_k – значення компоненти k.

Частка дисперсії, пояснена окремою компонентою, визначалася за формулою:

$$EVR_k = \frac{\lambda_k}{\sum_{j=1}^{p} \lambda_j}, \tag{3.13}$$

де λ_k – власне значення компоненти k.

Результати розподілу поясненої дисперсії за компонентами наведено в таблиці

Таблиця 3.2 – Пояснена дисперсія головних компонентів

Компонента	Пояснена дисперсія, %	Накопичена дисперсія, %
PC1	51,70	51,70
PC2	19,45	71,15
PC3	11,33	82,47
PC4	8,34	90,81
PC5	3,67	94,49
PC6	2,90	97,39

Перша компонента пояснює 51,70 % загальної варіації, друга – 19,45 %. Разом перші дві компоненти пояснюють 71,15 % варіації початкових показників. Три компоненти пояснюють 82,47 %, а чотири – 90,81 %. Візуальний розподіл часток дисперсії відображено на рисунку 3.3.1

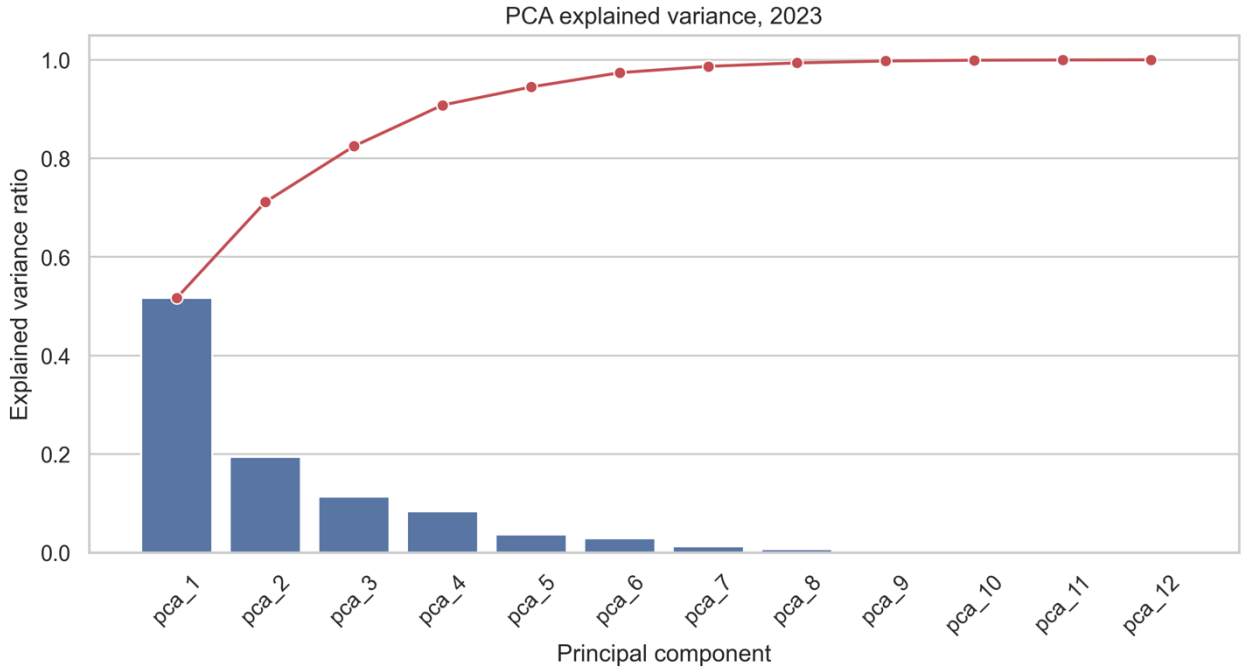

Рисунок 3.12 – Частка та накопичена частка дисперсії, поясненої компонентами PCA

Графік демонструє, що найбільше зростання накопиченої поясненої дисперсії забезпечують перші дві компоненти. Подальше додавання компонентів поступово збільшує пояснену дисперсію, але їхній внесок є значно меншим. Тому для основної змістовної інтерпретації було використано перші дві компоненти.

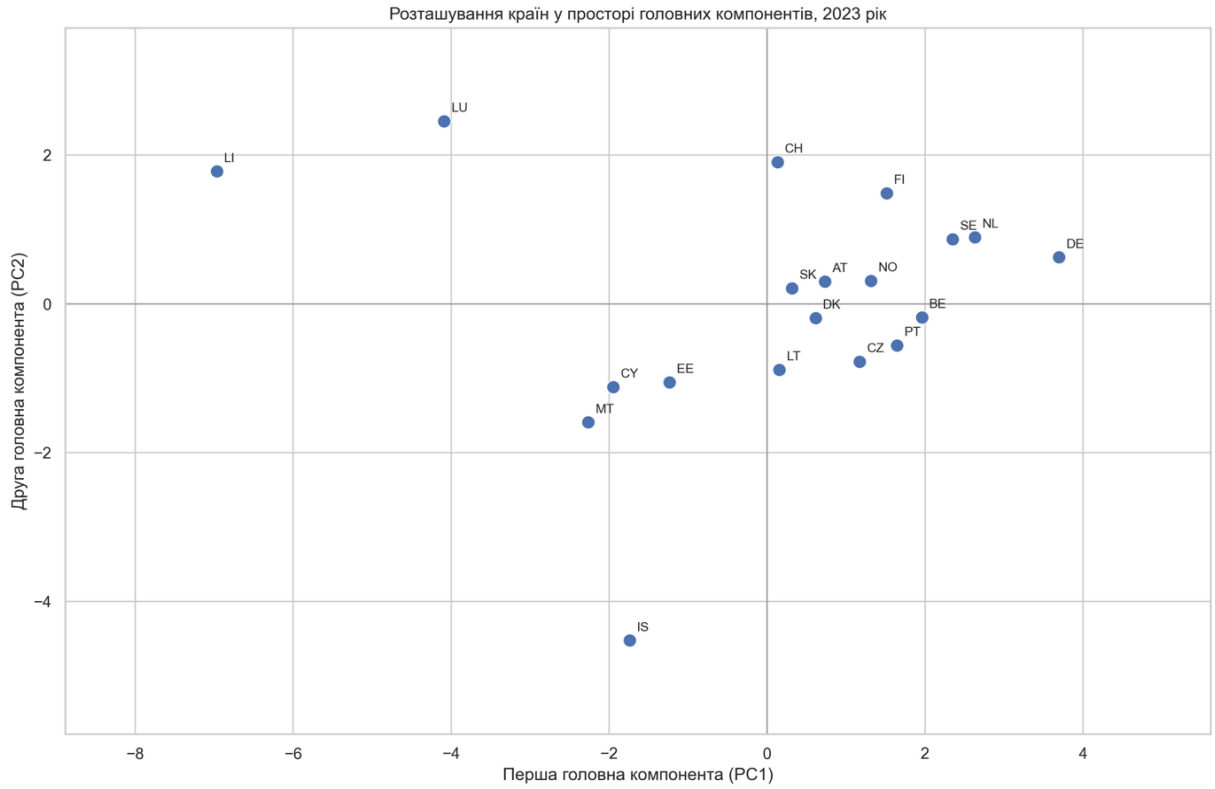

Рисунок 3.13 – Розташування освітніх систем країн у просторі перших двох головних компонентів

На графіку спостерігається помітна диференціація країн уздовж обох осей. Німеччина, Нідерланди, Швеція та Бельгія мають високі значення PC1, що характеризує їх як відносно масштабні та деконцентровані системи вищої освіти. Натомість Ліхтенштейн, Люксембург, Мальта та Кіпр розташовані в лівій частині графіка, що передусім зумовлено малим масштабом їхніх національних систем.

За другою компонентою (PC2) найвищі значення мають Люксембург, Швейцарія, Ліхтенштейн і Фінляндія. Це пов’язано з вищими фінансовими витратами на одного студента, меншим кадровим навантаженням на викладачів та відносно рівномірнішим розподілом студентського контингенту. Ісландія суттєво відокремлена від інших країн за віссю PC2, що може свідчити про специфічне поєднання досліджуваних характеристик або потенційний вплив аномальних значень (аутлаєрів).

Країни, розташовані близько одна до одної, мають подібні профілі за сукупністю використаних показників. Водночас варто враховувати, що цей графік відображає лише перші дві компоненти, які сумарно пояснюють 71,15 % загальної варіації, тому він не може повною мірою передати всі латентні відмінності між країнами. Перші дві головні компоненти вже виконали ключове завдання типізації, наочно диференціювавши країни за осями масштабу та ресурсного забезпечення, тому автоматичне кластерне групування лише продублює поточні результати без додавання нової змістовної інформації.

ВИСНОВКИ

У дипломній роботі проведено статистичний аналіз систем вищої освіти європейських країн на основі даних міжнародного реєстру «European Tertiary Education Register» (ETER). У процесі дослідження сформовано відтворюваний аналітичний підхід, що дозволяє перейти від первинних даних ETER до узагальнених показників на рівні закладів вищої освіти (ЗВО) та країн.

На першому етапі проаналізовано структуру вихідного набору даних, відібрано релевантні змінні, сформовано робочу таблицю на рівні «ЗВО–рік», виконано очищення спеціальних значень, перевірку дублікатів, приведення типів даних та оцінювання повноти ключових показників. Це забезпечило коректність подальших розрахунків і зменшило ризик викривлення результатів.

У роботі сформовано систему аналітичних показників, які характеризують масштаб освітніх систем, структуру студентів за рівнями ISCED - 2011, частку іноземних студентів, кадрове та фінансове забезпечення, а також рівень концентрації студентів між ЗВО. На основі цих показників побудовано агреговані таблиці на рівні «країна–рік».

Результати описового аналізу показали суттєву неоднорідність систем вищої освіти європейських країн. Виявлено, що країни відрізняються не лише за кількістю студентів і закладів, а й за середнім та медіанним розміром ЗВО, структурою освітніх рівнів, рівнем інтернаціоналізації та концентрацією студентів.

Аналіз інтернаціоналізації засвідчив, що високі частки іноземних студентів характерні переважно для малих і відкритих освітніх систем. Найбільш стійкий позитивний зв’язок виявлено між часткою іноземних студентів і часткою студентів рівня ISCED 7, що може свідчити про більшу привабливість магістерських програм для міжнародної студентської мобільності.

Оцінювання концентрації студентів показало, що для багатьох країн характерна нерівномірність розподілу студентського контингенту між ЗВО.

Показники $top - N$ залежать від кількості закладів у країні, тоді як коефіцієнт Джині краще відображає загальну нерівномірність розподілу студентів між усіма закладами.

Факторний аналіз методом головних компонент дозволив узагальнити відмінності між освітніми системами за сукупністю показників. Перші головні компоненти відображають насамперед масштаб системи, рівень концентрації, ресурсне забезпечення та структуру студентського контингенту.

Практична цінність роботи полягає у створенні відтворюваного програмного підходу до обробки й аналізу даних ETER. Отримані таблиці, показники та графічні матеріали можуть бути використані для подальших порівняльних досліджень систем вищої освіти, зокрема для зіставлення європейських показників із системою вищої освіти України.

До основних обмежень дослідження належать нерівномірна повнота даних за країнами, пропуски у фінансових і кадрових показниках, відмінності в національних правилах звітності та використання переважно останнього доступного часового зрізу. Тому отримані результати слід інтерпретувати як статистичні асоціації, а не як доказ причинно-наслідкових зв’язків.

ПЕРЕЛІК ДЖЕРЕЛ

1. Чайка І. Аналіз українського ринку вищої освіти. Галицький економічний вісник. 2020. № 1 (62). С. 91–101. DOI: https://doi.org/10.33108/galicianvisnyk_tntu2020.01.091. URL: https://galicianvisnyk.tntu.edu.ua/pdf/62/793.pdf (дата звернення: 07.05.2026).

2. Ящук Т. А., Чирва О. Г. Статистичний аналіз фінансування системи вищої освіти в Україні. Економіка і суспільство. 2018. Вип. 15. С. 193–199. URL: https://economyandsociety.in.ua/journals/15_ukr/30.pdf (дата звернення: 07.05.2026).

3. Європейський простір вищої освіти та Болонський процес : навч.-метод. посіб. / Т. М. Димань, О. А. Боньковський, А. Г. Вовкогон. Одеса : НУ «ОМА», 2017. 106 с. URL: https://foreign.udau.edu.ua/assets/files/docs/quantus/traning/european-higher-education-area-and-the-bologna-process_17_09_2017.pdf (дата звернення: 08.05.2026).

4. UNESCO Institute for Statistics. International Standard Classification of Education (ISCED). URL: https://www.uis.unesco.org/en/methods-and-tools/isced (дата звернення: 08.05.2026).

5. Болонський процес як засіб інтеграції і демократизації вищої освіти в Україні. URL: http://www.library.hneu.edu.ua/storage/vistavk/Bolon/page1.html (дата звернення: 08.05.2026).

6. The Nordic model of higher education from a comparative and historical perspective. In: Routledge Handbook of the Sociology of Higher Education. 2022. URL: https://www.researchgate.net/publication/359286110_The_Nordic_model_of_higher_education_from_a_comparative_and_historical_perspective_In_Routledge_Handbook_of_the_Sociology_of_Higher_Education_Ed_by_James_E_Cote_and_Sarah_Pickard_2022 (дата звернення: 09.05.2026).

7. Слюсаренко О. Інструменти вимірювання якості освітньої діяльності у контексті забезпечення її ефективності в умовах воєнного стану і післявоєнного відновлення країни. Вісник Черкаського національного університету імені Богдана Хмельницького. Серія: Педагогічні науки. 2023. № 2. С. 5–14. URL: https://ped-ejournal.cdu.edu.ua/article/view/4845 (дата звернення: 09.05.2026).

8. Jupyter Notebook Documentation / Project Jupyter. URL: https://jupyter-notebook.readthedocs.io/ (дата звернення: 05.05.2026).

9. Python 3.14.5 documentation / Python Software Foundation. URL: https://docs.python.org/3/ (дата звернення: 11.05.2026).

10. pandas 3.0.3 documentation. User Guide / pandas development team. URL: https://pandas.pydata.org/docs/user_guide/index.html (дата звернення: 11.05.2026).

11. NumPy user guide / NumPy Developers. URL: https://numpy.org/doc/stable/user/index.html (дата звернення: 12.05.2026).

12. Using Matplotlib / Matplotlib Development Team. URL: https://matplotlib.org/stable/users/index.html (дата звернення: 13.05.2026).

13. seaborn 0.13.2 documentation. User guide and tutorial / seaborn development team. URL: https://seaborn.pydata.org/tutorial.html (дата звернення: 13.05.2026).

14. scikit-learn: Machine Learning in Python / scikit-learn developers. URL: https://scikit-learn.org/stable/ (дата звернення: 13.05.2026).

15. ETER Technical Handbook v4.1 / European Tertiary Education Register. 2023. URL: https://national-policies.eacea.ec.europa.eu/sites/default/files/2025-01/ETERIV_Handbook_Complete_2023updated.pdf (дата звернення: 10.05.2026).

16. Lepori B., Bonaccorsi A., Daraio C. The use of European Tertiary Education Register (ETER) for polycentric higher education studies // Higher Education. – 2015. – Vol. 69, No. 5. – P. 715–733. – URL: https://eter-project.com/assets/pdf/ETER_final_report_exec_en.pdf (дата звернення: 15.05.2026).

17. International Standard Classification of Education: ISCED 2011 / UNESCO Institute for Statistics. – Montreal : UNESCO Institute for Statistics, 2012. – 84 p. – URL: https://www.openemis.org/wp-content/uploads/2018/04/unesco-international-standard-classification-education-isced-2011-en.pdf (дата звернення: 15.05.2026).

ДОДАТОК А

Код програми

```
#NOTEBOOK: 00_df_base_cleanup.ipynb
#CODE CELLS: 30



# Cell 1 (notebook cell 4)
from pathlib import Path

import pandas as pd
import seaborn as sns

pd.set_option("display.max_columns", 140)
pd.set_option("display.max_rows", 100)
sns.set_theme(style="whitegrid")

# Cell 2 (notebook cell 6)
PROJECT_ROOT = Path.cwd().parent if Path.cwd().name == "notebooks" else Path.cwd()
DATA_DIR = PROJECT_ROOT / "data" / "raw"
output_dir = PROJECT_ROOT / "data" / "processed"
output_dir.mkdir(parents=True, exist_ok=True)

path = DATA_DIR / "ETER_fullDump_27042023.csv"
df_raw = pd.read_csv(
    path,
    sep=";",
    encoding="utf-8-sig",
    decimal=",",
```

```
    low_memory=False,
)

# Cell 3 (notebook cell 8)
df_raw.info()

# Cell 4 (notebook cell 9)
df_raw.shape

# Cell 5 (notebook cell 10)
df_raw.head(5)

# Cell 6 (notebook cell 11)
df_raw.columns.tolist()

# Cell 7 (notebook cell 12)
df_raw.dtypes.value_counts()

# Cell 8 (notebook cell 14)
selected_columns = [
    # IDs
    "ETER ID Year",
    "ETER ID",
    "Institution Name",
    "English Institution Name",
    "Reference year",

    # Geography
    "Country Code",
    "Region of establishment (NUTS 2)",
```

"Region of establishment (NUTS 3)",
"Name of the city",
"Geographic coordinates - latitude",
"Geographic coordinates - longitude",
"Multi-site institution",

Institution type
"Legal status",
"Institution Category - English",
"Institution Category standardised",

Students
"Total students enrolled at ISCED 5",
"Total students enrolled at ISCED 6",
"Total students enrolled at ISCED 7",
"Total students enrolled ISCED 7 long degree",
"Total students enrolled ISCED 5-7",
"Total students enrolled at ISCED 8",

Foreign students
"Students enrolled at ISCED 5 - foreigner",
"Students enrolled at ISCED 6 - foreigner",
"Students enrolled at ISCED 7 - foreigner",
"Students enrolled ISCED 7 long degree - foreigner",
"Students enrolled at ISCED 5-7 - foreigner",
"Students enrolled at ISCED 8 - foreigner",

Staff
"Total academic personnel (FTE)",
"Total personnel (FTE)",

"Number of support and administrative personnel (FTE)",

Finance
"Total Current expenditure.1",
"Total Current expenditure.2",
"Total Current revenues.1",
"Total Current revenues.2",
"Total third party funding.1",
"Total third party funding.2",
"R&D Expenditure.1",
"R&D Expenditure.2",

Quality flags: students totals
"Flag Total students ISCED 5",
"Flag Total students ISCED 6",
"Flag Total students ISCED 7",
"Flag Total students ISCED 7 long degree",
"Flag Total students ISCED 5-7",
"Flag Total students ISCED 8",

Quality flags: foreign students / citizenship breakdowns
"Flag Students ISCED 5 - citizenship",
"Flag Students ISCED 6 - citizenship",
"Flag Students ISCED 7 - citizenship",
"Flag Students ISCED 7 long degree - citizenship",
"Flag students enrolled at ISCED 5-7 - citizenship",
"Flag Students ISCED 8 - citizenship",

Quality flags: staff
"Flag Total academic personnel (FTE)",

```
    "Flag Total personnel (FTE)",
    "Flag Number of support and administrative personnel (FTE)",

    # Quality flags: finance
    "Flag Total current expenditure",
    "Flag Total current revenues",
    "Flag Total third party funding",
    "Flag R&D Expenditure",
]

rename_map = {
    "ETER ID Year": "record_id",
    "ETER ID": "institution_id",
    "Institution Name": "institution_name",
    "English Institution Name": "institution_name_en",
    "Reference year": "year",
    "Country Code": "country_code",

    "Region of establishment (NUTS 2)": "nuts2",
    "Region of establishment (NUTS 3)": "nuts3",
    "Name of the city": "city",
    "Geographic coordinates - latitude": "latitude",
    "Geographic coordinates - longitude": "longitude",
    "Multi-site institution": "multi_site",

    "Legal status": "legal_status",
    "Institution Category - English": "institution_category_en",
    "Institution Category standardised": "institution_category_std",

    "Total students enrolled at ISCED 5": "students_isced5",
```

"Total students enrolled at ISCED 6": "students_isced6",

"Total students enrolled at ISCED 7": "students_isced7",

"Total students enrolled ISCED 7 long degree": "students_isced7_long_degree",

"Total students enrolled ISCED 5-7": "students_isced5_7_total",

"Total students enrolled at ISCED 8": "students_isced8",

"Students enrolled at ISCED 5 - foreigner": "foreign_students_isced5",

"Students enrolled at ISCED 6 - foreigner": "foreign_students_isced6",

"Students enrolled at ISCED 7 - foreigner": "foreign_students_isced7",

"Students enrolled ISCED 7 long degree - foreigner": "foreign_students_isced7_long_degree",

"Students enrolled at ISCED 5-7 - foreigner": "foreign_students_isced5_7_total",

"Students enrolled at ISCED 8 - foreigner": "foreign_students_isced8",

"Total academic personnel (FTE)": "academic_personnel_fte",

"Total personnel (FTE)": "total_personnel_fte",

"Number of support and administrative personnel (FTE)": "support_admin_personnel_fte",

"Total Current expenditure.1": "total_current_expenditure_eur",

"Total Current expenditure.2": "total_current_expenditure_ppp",

"Total Current revenues.1": "total_current_revenues_eur",

"Total Current revenues.2": "total_current_revenues_ppp",

"Total third party funding.1": "third_party_funding_eur",

"Total third party funding.2": "third_party_funding_ppp",

"R&D Expenditure.1": "rd_expenditure_eur",

"R&D Expenditure.2": "rd_expenditure_ppp",

```
        "Flag Total students ISCED 5": "flag_students_isced5",
        "Flag Total students ISCED 6": "flag_students_isced6",
        "Flag Total students ISCED 7": "flag_students_isced7",
        "Flag Total students ISCED 7 long degree": "flag_students_isced7_long_degree",
        "Flag Total students ISCED 5-7": "flag_students_isced5_7_total",
        "Flag Total students ISCED 8": "flag_students_isced8",

        "Flag Students ISCED 5 - citizenship": "flag_foreign_students_isced5",
        "Flag Students ISCED 6 - citizenship": "flag_foreign_students_isced6",
        "Flag Students ISCED 7 - citizenship": "flag_foreign_students_isced7",
        "Flag Students ISCED 7 long degree - citizenship": "flag_foreign_students_isced7_long_degree",
        "Flag students enrolled at ISCED 5-7 - citizenship": "flag_foreign_students_isced5_7_total",
        "Flag Students ISCED 8 - citizenship": "flag_foreign_students_isced8",

        "Flag Total academic personnel (FTE)": "flag_academic_personnel_fte",
        "Flag Total personnel (FTE)": "flag_total_personnel_fte",
        "Flag Number of support and administrative personnel (FTE)": "flag_support_admin_personnel_fte",

        "Flag Total current expenditure": "flag_total_current_expenditure",
        "Flag Total current revenues": "flag_total_current_revenues",
        "Flag Total third party funding": "flag_third_party_funding",
        "Flag R&D Expenditure": "flag_rd_expenditure",
    }

    # Cell 9 (notebook cell 16)
    existing_columns = [col for col in selected_columns if col in df_raw.columns]
```

```
missing_selected_columns = [col for col in selected_columns if col not in df_raw.columns]

if missing_selected_columns:
    print("Missing selected columns:")
    display(missing_selected_columns)
else:
    print("All selected columns are available in the raw dataset.")

# Cell 10 (notebook cell 18)
df_base = df_raw[existing_columns].copy()
df_base = df_base.rename(columns=rename_map)
df_base.shape

# Cell 11 (notebook cell 20)
df_base.head()

# Cell 12 (notebook cell 21)
df_base.info()

# Cell 13 (notebook cell 22)
df_base.columns.tolist()

# Cell 14 (notebook cell 23)
df_base.dtypes

# Cell 15 (notebook cell 24)
df_base.dtypes.value_counts()

# Cell 16 (notebook cell 26)
```

```
duplicates_report = pd.DataFrame({
    "check": [
        "full_row_duplicates",
        "record_id_duplicates",
        "institution_id_year_duplicates",
    ],
    "duplicates_count": [
        df_base.duplicated().sum(),
        df_base["record_id"].duplicated().sum(),
        df_base.duplicated(subset=["institution_id", "year"]).sum(),
    ],
})

duplicates_report

# Cell 17 (notebook cell 28)
missing_report = pd.DataFrame({
    "missing_count": df_base.isna().sum(),
    "missing_rate": df_base.isna().mean()
})

missing_report = missing_report.sort_values("missing_count", ascending=False)

missing_report

# Cell 18 (notebook cell 31)
special_codes = pd.read_excel(
    DATA_DIR / "CorrespondenceTable_NamesCodesToLabels.xlsx",
    sheet_name="SpecialCodes",
```

```
)

output_path = output_dir / "special_codes.csv"

special_codes.to_csv(
    output_path,
    index=False,
    encoding="utf-8-sig",
)

special_codes

# Cell 19 (notebook cell 33)
special_codes = ["a", "m", "x", "xc", "xr", "nc", "c", "s"]

special_code_report = pd.DataFrame({
    "special_code_count": df_base.astype("string").apply(
        lambda col: col.str.strip().isin(special_codes).sum()
    )
})

special_code_report["special_code_rate"] = (
    special_code_report["special_code_count"] / len(df_base)
)

special_code_report["special_code_pct"] = (
    special_code_report["special_code_rate"] * 100
)

special_code_report = special_code_report.sort_values(
```

```
    "special_code_count",
    ascending=False
)

special_code_report

# Cell 20 (notebook cell 35)
output_path = output_dir / "df_base.csv"

df_base.to_csv(
    output_path,
    index=False,
    encoding="utf-8-sig"
)

# Cell 21 (notebook cell 38)
flag_cols = [col for col in df_base.columns if col.startswith("flag_")]
non_flag_cols = [col for col in df_base.columns if col not in flag_cols]

df_base[non_flag_cols] = df_base[non_flag_cols].replace(
    r"^\s*(a|m|x|xc|xr|nc|c|s)\s*$",
    pd.NA,
    regex=True
)

# Cell 22 (notebook cell 40)
missing_after_cleaning = pd.DataFrame({
    "missing_count": df_base.isna().sum(),
    "missing_rate": df_base.isna().mean()
}).sort_values("missing_count", ascending=False)
```

```
missing_after_cleaning

# Cell 23 (notebook cell 42)
special_codes_left_non_flags = (
    df_base[non_flag_cols]
    .astype("string")
    .apply(lambda col: col.str.strip().isin(special_codes).sum())
    .sort_values(ascending=False)
)

special_codes_left_non_flags[special_codes_left_non_flags > 0]

# Cell 24 (notebook cell 44)
special_codes_in_flags = (
    df_base[flag_cols]
    .astype("string")
    .apply(lambda col: col.str.strip().isin(special_codes).sum())
    .sort_values(ascending=False)
)

special_codes_in_flags[special_codes_in_flags > 0]

# Cell 25 (notebook cell 46)
numeric_float_cols = [
    "students_isced5",
    "students_isced6",
    "students_isced7",
    "students_isced7_long_degree",
    "students_isced5_7_total",
```

```
    "students_isced8",

    "foreign_students_isced5",
    "foreign_students_isced6",
    "foreign_students_isced7",
    "foreign_students_isced7_long_degree",
    "foreign_students_isced5_7_total",
    "foreign_students_isced8",

    "academic_personnel_fte",
    "total_personnel_fte",
    "support_admin_personnel_fte",

    "total_current_expenditure_eur",
    "total_current_expenditure_ppp",
    "total_current_revenues_eur",
    "total_current_revenues_ppp",
    "third_party_funding_eur",
    "third_party_funding_ppp",
    "rd_expenditure_eur",
    "rd_expenditure_ppp",

    "latitude",
    "longitude",
]
df_base[numeric_float_cols] = (
    df_base[numeric_float_cols]
    .astype("string")
    .apply(lambda col: col.str.strip().str.replace(",", ".", regex=False))
    .apply(pd.to_numeric, errors="coerce")
```

```
)

text_cols = df_base.select_dtypes(include=["object", "string"]).columns

df_base[text_cols] = (
    df_base[text_cols]
    .astype("string")
    .apply(lambda col: col.str.strip())
)

# Cell 26 (notebook cell 47)
df_base.dtypes

# Cell 27 (notebook cell 49)
student_cols = [
    "students_isced5", "students_isced6", "students_isced7",
    "students_isced7_long_degree", "students_isced5_7_total",
    "students_isced8", "foreign_students_isced5",
    "foreign_students_isced6", "foreign_students_isced7",
    "foreign_students_isced7_long_degree",
    "foreign_students_isced5_7_total", "foreign_students_isced8",
]

staff_cols = [
    "academic_personnel_fte",
    "total_personnel_fte",
    "support_admin_personnel_fte",
]

finance_cols = [
```

```
    "total_current_expenditure_eur", "total_current_expenditure_ppp",
    "total_current_revenues_eur", "total_current_revenues_ppp",
    "third_party_funding_eur", "third_party_funding_ppp",
    "rd_expenditure_eur", "rd_expenditure_ppp",
]

check_cols = student_cols + staff_cols + finance_cols

anomaly_summary = pd.DataFrame({
    "negative_values": df_base[check_cols].lt(0).sum(),
    "missing_values": df_base[check_cols].isna().sum(),
})

anomaly_summary

# Cell 28 (notebook cell 50)
coordinate_anomalies = df_base[
    df_base["latitude"].notna()
    & df_base["longitude"].notna()
    & (
        ~df_base["latitude"].between(-90, 90)
        | ~df_base["longitude"].between(-180, 180)
    )
][[
    "record_id",
    "institution_id",
    "institution_name",
    "country_code",
    "city",
    "latitude",
```

```
    "longitude",
]]

coordinate_anomalies

# Cell 29 (notebook cell 51)
student_staff_diagnostic = df_base.assign(
    students_total_diagnostic=(
        df_base["students_isced5_7_total"].fillna(0)
        + df_base["students_isced8"].fillna(0)
    )
)

student_staff_diagnostic["student_staff_ratio_diagnostic"] = (
    student_staff_diagnostic["students_total_diagnostic"]
    / student_staff_diagnostic["academic_personnel_fte"]
)

student_staff_ratio_anomalies = student_staff_diagnostic[
    student_staff_diagnostic["academic_personnel_fte"].gt(0)
    & (
        student_staff_diagnostic["student_staff_ratio_diagnostic"].gt(100)
        | student_staff_diagnostic["student_staff_ratio_diagnostic"].lt(1)
    )
][[
    "record_id",
    "institution_id",
    "institution_name",
    "country_code",
    "year",
```

```
    "students_total_diagnostic",
    "academic_personnel_fte",
    "student_staff_ratio_diagnostic",
]]

student_staff_ratio_anomalies.sort_values(
    "student_staff_ratio_diagnostic",
    ascending=False
)

# Cell 30 (notebook cell 53)
output_path = output_dir / "df_base_clean.csv"

df_base.to_csv(
    output_path,
    index=False,
    encoding="utf-8-sig"
)

#NOTEBOOK: 01_country_year_profiles_and_eda_metrics.ipynb
#CODE CELLS: 15

# Cell 1 (notebook cell 3)
from pathlib import Path

import numpy as np
import pandas as pd

# Cell 2 (notebook cell 5)
```

```
PROJECT_ROOT = Path.cwd().parent if Path.cwd().name == "notebooks" else Path.cwd()
DATA_DIR = PROJECT_ROOT / "data"
PROCESSED_DIR = DATA_DIR / "processed"
FIGURES_DIR = PROJECT_ROOT / "figures"
TABLES_DIR = PROJECT_ROOT / "tables"

PROCESSED_DIR.mkdir(parents=True, exist_ok=True)
FIGURES_DIR.mkdir(parents=True, exist_ok=True)
TABLES_DIR.mkdir(parents=True, exist_ok=True)

INPUT_PATH = PROCESSED_DIR / "df_base_clean.csv"

INSTITUTION_YEAR_METRICS_PATH = PROCESSED_DIR / "institution_year_metrics.csv"
COUNTRY_YEAR_PROFILES_PATH = PROCESSED_DIR / "country_year_profiles.csv"
CONCENTRATION_METRICS_PATH = PROCESSED_DIR / "concentration_metrics.csv"
STUDENT_STRUCTURE_METRICS_PATH = PROCESSED_DIR / "student_structure_metrics.csv"
FINANCE_STAFF_METRICS_PATH = PROCESSED_DIR / "finance_staff_metrics.csv"
ANALYSIS_READINESS_REPORT_PATH = PROCESSED_DIR / "analysis_readiness_report.csv"

# Cell 3 (notebook cell 7)
if not INPUT_PATH.exists():
    raise FileNotFoundError(f"Input file not found: {INPUT_PATH}")
```

```
df_base_clean = pd.read_csv(INPUT_PATH)

print(f"Rows: {df_base_clean.shape[0]:,}")
print(f"Columns: {df_base_clean.shape[1]:,}")
print(f"Countries: {df_base_clean['country_code'].nunique() if 'country_code' in df_base_clean else 'missing column'}")
print(f"Years: {df_base_clean['year'].min()} - {df_base_clean['year'].max()}" if 'year' in df_base_clean else "Missing year column")

df_base_clean.head()

# Cell 4 (notebook cell 9)
required_columns = [
    "record_id",
    "institution_id",
    "institution_name",
    "institution_name_en",
    "year",
    "country_code",
    "students_isced5",
    "students_isced6",
    "students_isced7",
    "students_isced7_long_degree",
    "students_isced5_7_total",
    "students_isced8",
    "foreign_students_isced5",
    "foreign_students_isced6",
    "foreign_students_isced7",
    "foreign_students_isced7_long_degree",
    "foreign_students_isced5_7_total",
```

```
        "foreign_students_isced8",
        "academic_personnel_fte",
        "total_personnel_fte",
        "total_current_expenditure_ppp",
        "total_current_revenues_ppp",
        "rd_expenditure_ppp",
    ]

    missing_required_columns = sorted(set(required_columns) - set(df_base_clean.columns))
    if missing_required_columns:
        raise ValueError(f"Missing required columns: {missing_required_columns}")

    flag_columns = [col for col in df_base_clean.columns if col.startswith("flag_")]
    print(f"Required columns are available: {len(required_columns)}")
    print(f"Flag columns preserved: {len(flag_columns)}")

    # Cell 5 (notebook cell 11)
    def row_sum_min_count(df, columns):
        """Return row-wise sum without converting fully missing rows to zero."""
        available_columns = [col for col in columns if col in df.columns]
        if not available_columns:
            return pd.Series(np.nan, index=df.index)
        return df[available_columns].sum(axis=1, min_count=1)


    def safe_divide(numerator, denominator):
        numerator = pd.Series(numerator)
```

```
    denominator = pd.Series(denominator)
    return np.where(denominator > 0, numerator / denominator, np.nan)


def safe_scalar_divide(numerator, denominator):
    if pd.notna(denominator) and denominator > 0:
        return numerator / denominator
    return np.nan


def classify_size(students_total):
    if pd.isna(students_total):
        return np.nan
    if students_total < 1_000:
        return "small"
    if students_total < 10_000:
        return "medium"
    if students_total < 30_000:
        return "large"
    return "very_large"


def gini_non_negative(values):
    values = pd.Series(values).dropna()
    values = values[values >= 0]
    if values.empty or values.sum() <= 0:
        return np.nan

    sorted_values = np.sort(values.to_numpy(dtype=float))
    n = sorted_values.size
```

```
    index = np.arange(1, n + 1)
    return (2 * np.sum(index * sorted_values)) / (n * np.sum(sorted_values)) -
(n + 1) / n


def gini_pairwise_definition(values):
    """Calculate Gini with the pairwise-difference formula used in the thesis."""
    values = pd.Series(values).dropna()
    values = values[values >= 0].to_numpy(dtype=float)
    if values.size == 0 or values.sum() <= 0:
        return np.nan

    pairwise_absolute_differences = np.abs(values[:, None] - values[None,
:]).sum()
    return pairwise_absolute_differences / (2 * values.size ** 2 * values.mean())


def top_n_share(values, n):
    values = pd.Series(values).dropna()
    values = values[values >= 0]
    total = values.sum()
    if values.empty or total <= 0:
        return np.nan
    share = values.sort_values(ascending=False).head(n).sum() / total
    return min(max(share, 0), 1)


def sum_min_count(series):
    return series.sum(min_count=1)
```

```
# Cell 6 (notebook cell 13)
institution_year_metrics = df_base_clean.copy()

student_level_columns = [
    "students_isced5",
    "students_isced6",
    "students_isced7",
    "students_isced7_long_degree",
    "students_isced8",
]
foreign_student_level_columns = [
    "foreign_students_isced5",
    "foreign_students_isced6",
    "foreign_students_isced7",
    "foreign_students_isced7_long_degree",
    "foreign_students_isced8",
]

institution_year_metrics["students_total_by_levels"] = row_sum_min_count(
    institution_year_metrics,
    student_level_columns,
)
# Use the reported ISCED 5-7 total only when it is available.
# Otherwise, ISCED 8 alone must not become the overall student total.
institution_year_metrics["students_total_reported"] = (
    institution_year_metrics["students_isced5_7_total"]
    + institution_year_metrics["students_isced8"].fillna(0)
).where(institution_year_metrics["students_isced5_7_total"].notna())
institution_year_metrics["students_total"] = institution_year_metrics[
    "students_total_reported"
```

```
        ].combine_first(institution_year_metrics["students_total_by_levels"])

        institution_year_metrics["foreign_students_total_by_levels"] =
row_sum_min_count(
            institution_year_metrics,
            foreign_student_level_columns,
        )
        # Apply the same rule for foreign students: do not use ISCED 8 alone
        # as the reported total when the ISCED 5-7 total is missing.
        institution_year_metrics["foreign_students_total_reported"] = (
            institution_year_metrics["foreign_students_isced5_7_total"]
            + institution_year_metrics["foreign_students_isced8"].fillna(0)
        ).where(institution_year_metrics["foreign_students_isced5_7_total"].notna())
        institution_year_metrics["foreign_students_total"] = institution_year_metrics[
            "foreign_students_total_reported"
        ].combine_first(institution_year_metrics["foreign_students_total_by_levels"])

        institution_year_metrics["foreign_students_share"] = safe_divide(
            institution_year_metrics["foreign_students_total"],
            institution_year_metrics["students_total"],
        )

        for level in ["isced5", "isced6", "isced7", "isced8"]:
            institution_year_metrics[f"students_{level}_share"] = safe_divide(
                institution_year_metrics[f"students_{level}"],
                institution_year_metrics["students_total"],
            )

        institution_year_metrics["student_academic_staff_ratio"] = safe_divide(
            institution_year_metrics["students_total"],
```

```
            institution_year_metrics["academic_personnel_fte"],
        )
        institution_year_metrics["student_total_staff_ratio"] = safe_divide(
            institution_year_metrics["students_total"],
            institution_year_metrics["total_personnel_fte"],
        )
        institution_year_metrics["expenditure_per_student_ppp"] = safe_divide(
            institution_year_metrics["total_current_expenditure_ppp"],
            institution_year_metrics["students_total"],
        )
        institution_year_metrics["revenue_per_student_ppp"] = safe_divide(
            institution_year_metrics["total_current_revenues_ppp"],
            institution_year_metrics["students_total"],
        )
        institution_year_metrics["rd_expenditure_per_student_ppp"] = safe_divide(
            institution_year_metrics["rd_expenditure_ppp"],
            institution_year_metrics["students_total"],
        )

        institution_year_metrics["size_category"]                                    =
institution_year_metrics["students_total"].apply(classify_size)

        metric_columns = [
            "students_total",
            "students_total_by_levels",
            "foreign_students_total",
            "foreign_students_share",
            "students_isced5_share",
            "students_isced6_share",
            "students_isced7_share",
```

```
    "students_isced8_share",
    "student_academic_staff_ratio",
    "student_total_staff_ratio",
    "expenditure_per_student_ppp",
    "revenue_per_student_ppp",
    "rd_expenditure_per_student_ppp",
    "size_category",
]

institution_year_metrics[["country_code", "year", "institution_id"] +
metric_columns].head()

# Cell 7 (notebook cell 15)
institution_year_duplicate_count = institution_year_metrics.duplicated(
    subset=["institution_id", "year"]
).sum()

non_negative_metric_columns = [
    "students_total",
    "students_total_by_levels",
    "foreign_students_total",
    "student_academic_staff_ratio",
    "student_total_staff_ratio",
    "expenditure_per_student_ppp",
    "revenue_per_student_ppp",
    "rd_expenditure_per_student_ppp",
]
negative_counts = {
    col: int((institution_year_metrics[col] < 0).sum())
    for col in non_negative_metric_columns
```

```
    }

    share_columns = [
        "foreign_students_share",
        "students_isced5_share",
        "students_isced6_share",
        "students_isced7_share",
        "students_isced8_share",
    ]
    share_out_of_range_counts = {
        col: int(((institution_year_metrics[col] < 0) | (institution_year_metrics[col]
> 1)).sum())
        for col in share_columns
    }

    print(f"Duplicate institution_id + year rows:
{institution_year_duplicate_count}")
    print("Negative values:")
    print(pd.Series(negative_counts))
    print("Shares outside [0, 1]:")
    print(pd.Series(share_out_of_range_counts))
    print("Size categories:")
    print(institution_year_metrics["size_category"].value_counts(dropna=False))

    # Cell 8 (notebook cell 17)
    student_structure_totals = (
        institution_year_metrics
        .groupby(["country_code", "year"], as_index=False)
        .agg(
            students_isced5_total=("students_isced5", sum_min_count),
```

```
        students_isced6_total=("students_isced6", sum_min_count),
        students_isced7_total=("students_isced7", sum_min_count),
        students_isced8_total=("students_isced8", sum_min_count),
        students_total=("students_total", sum_min_count),
    )
)

for level in ["isced5", "isced6", "isced7", "isced8"]:
    student_structure_totals[f"{level}_share"] = safe_divide(
        student_structure_totals[f"students_{level}_total"],
        student_structure_totals["students_total"],
    )

student_structure_metrics = student_structure_totals[
    [
        "country_code",
        "year",
        "students_isced5_total",
        "students_isced6_total",
        "students_isced7_total",
        "students_isced8_total",
        "isced5_share",
        "isced6_share",
        "isced7_share",
        "isced8_share",
    ]
].copy()

student_structure_metrics.head()
```

```
# Cell 9 (notebook cell 19)
finance_staff_totals = (
    institution_year_metrics
    .groupby(["country_code", "year"], as_index=False)
    .agg(
        total_students=("students_total", sum_min_count),
        academic_personnel_fte_total=("academic_personnel_fte",
sum_min_count),
        total_personnel_fte_total=("total_personnel_fte", sum_min_count),
        total_current_expenditure_ppp_total=("total_current_expenditure_ppp",
sum_min_count),
        total_current_revenues_ppp_total=("total_current_revenues_ppp",
sum_min_count),
        rd_expenditure_ppp_total=("rd_expenditure_ppp", sum_min_count),
    )
)

finance_staff_metrics = finance_staff_totals[["country_code", "year"]].copy()
finance_staff_metrics["student_academic_staff_ratio"] = safe_divide(
    finance_staff_totals["total_students"],
    finance_staff_totals["academic_personnel_fte_total"],
)
finance_staff_metrics["student_total_staff_ratio"] = safe_divide(
    finance_staff_totals["total_students"],
    finance_staff_totals["total_personnel_fte_total"],
)
finance_staff_metrics["expenditure_per_student_ppp"] = safe_divide(
    finance_staff_totals["total_current_expenditure_ppp_total"],
    finance_staff_totals["total_students"],
)
```

```
finance_staff_metrics["revenue_per_student_ppp"] = safe_divide(
    finance_staff_totals["total_current_revenues_ppp_total"],
    finance_staff_totals["total_students"],
)
finance_staff_metrics["rd_expenditure_per_student_ppp"] = safe_divide(
    finance_staff_totals["rd_expenditure_ppp_total"],
    finance_staff_totals["total_students"],
)

finance_staff_metrics.head()

# Cell 10 (notebook cell 21)
concentration_metrics = (
    institution_year_metrics
    .groupby(["country_code", "year"])
    .agg(
        top_1_share=("students_total", lambda values: top_n_share(values, 1)),
        top_5_share=("students_total", lambda values: top_n_share(values, 5)),
        top_10_share=("students_total", lambda values: top_n_share(values, 10)),
        gini_students=("students_total", gini_non_negative),
    )
    .reset_index()
)

concentration_metrics.head()

# Cell 11 (notebook cell 23)
country_year_core = (
    institution_year_metrics
    .groupby(["country_code", "year"], as_index=False)
```

```
            .agg(
                institutions_count=("institution_id", "nunique"),
                total_students=("students_total", sum_min_count),
                foreign_students_total=("foreign_students_total", sum_min_count),
                mean_institution_size=("students_total", "mean"),
                median_institution_size=("students_total", "median"),
                max_institution_size=("students_total", "max"),
            )
        )

        country_year_core["foreign_students_share"] = safe_divide(
            country_year_core["foreign_students_total"],
            country_year_core["total_students"],
        )

        country_year_profiles = (
            country_year_core
            .merge(student_structure_metrics, on=["country_code", "year"], how="left")
            .merge(finance_staff_metrics, on=["country_code", "year"], how="left")
            .merge(concentration_metrics, on=["country_code", "year"], how="left")
        )

        country_year_profiles = country_year_profiles.sort_values(["country_code", "year"]).reset_index(drop=True)

        country_year_profiles.head()

        # Cell 12 (notebook cell 25)
        country_year_duplicate_count = country_year_profiles.duplicated(
```

```
        subset=["country_code", "year"]
    ).sum()
    print(f'Duplicate country_code + year rows: {country_year_duplicate_count}")

    if country_year_duplicate_count != 0:
        raise ValueError("country_year_profiles contains duplicated country_code + year keys")

    gini_formula_validation = (
        institution_year_metrics
        .groupby(["country_code", "year"])["students_total"]
        .agg(
            gini_ranked_formula=gini_non_negative,
            gini_pairwise_formula=gini_pairwise_definition,
        )
        .reset_index()
    )
    gini_formula_validation["absolute_difference"] = (
        gini_formula_validation["gini_ranked_formula"]
        - gini_formula_validation["gini_pairwise_formula"]
    ).abs()
    gini_missing_pattern_mismatches = (
        gini_formula_validation["gini_ranked_formula"].isna()
        != gini_formula_validation["gini_pairwise_formula"].isna()
    ).sum()
    gini_formula_mismatches = (
        gini_formula_validation["absolute_difference"].fillna(0) > 1e-12
    ).sum()
```

```
    gini_max_absolute_difference =
gini_formula_validation["absolute_difference"].max()

    print(f"Gini formula mismatches: {gini_formula_mismatches}")
    print(f"Gini missing-pattern mismatches:
{gini_missing_pattern_mismatches}")
    print(f"Maximum Gini formula difference: {gini_max_absolute_difference}")

    if gini_formula_mismatches != 0 or gini_missing_pattern_mismatches != 0:
      raise ValueError("Ranked and pairwise Gini formulas produce different
results")

    country_share_columns = [
      "foreign_students_share",
      "isced5_share",
      "isced6_share",
      "isced7_share",
      "isced8_share",
      "top_1_share",
      "top_5_share",
      "top_10_share",
      "gini_students",
    ]
    country_share_out_of_range_counts = {
      col: int(((country_year_profiles[col] < 0) | (country_year_profiles[col] >
1)).sum())
      for col in country_share_columns
    }

    country_non_negative_columns = [
```

```
    "institutions_count",
    "total_students",
    "foreign_students_total",
    "mean_institution_size",
    "median_institution_size",
    "max_institution_size",
    "student_academic_staff_ratio",
    "student_total_staff_ratio",
    "expenditure_per_student_ppp",
    "revenue_per_student_ppp",
    "rd_expenditure_per_student_ppp",
]
country_negative_counts = {
    col: int((country_year_profiles[col] < 0).sum())
    for col in country_non_negative_columns
}

print("Country-year shares outside [0, 1]:")
print(pd.Series(country_share_out_of_range_counts))
print("Country-year negative values:")
print(pd.Series(country_negative_counts))
print("Missing values in key metrics:")
print(country_year_profiles[[
    "total_students",
    "foreign_students_share",
    "student_academic_staff_ratio",
    "expenditure_per_student_ppp",
    "gini_students",
]].isna().sum())
```

```
# Cell 13 (notebook cell 27)
analysis_readiness_report = (
    country_year_profiles
    .assign(
        missing_total_students=country_year_profiles["total_students"].isna(),

missing_foreign_students_share=country_year_profiles["foreign_students_share"].is
na(),

missing_student_staff_ratio=country_year_profiles["student_academic_staff_ratio"].i
sna(),
        missing_finance_metric=country_year_profiles[
            [
                "expenditure_per_student_ppp",
                "revenue_per_student_ppp",
                "rd_expenditure_per_student_ppp",
            ]
        ].isna().any(axis=1),
        missing_gini_students=country_year_profiles["gini_students"].isna(),
    )
    .groupby("country_code", as_index=False)
    .agg(
        years_count=("year", "nunique"),
        missing_total_students=("missing_total_students", "sum"),
        missing_foreign_students_share=("missing_foreign_students_share",
"sum"),
        missing_student_staff_ratio=("missing_student_staff_ratio", "sum"),
        missing_finance_metric=("missing_finance_metric", "sum"),
        missing_gini_students=("missing_gini_students", "sum"),
    )
```

```
        .sort_values(["missing_total_students",  "missing_foreign_students_share",
"country_code"])
        .reset_index(drop=True)
    )

    analysis_readiness_report.head(10)

    # Cell 14 (notebook cell 29)
    output_tables = {
      "institution_year_metrics": institution_year_metrics,
      "country_year_profiles": country_year_profiles,
      "concentration_metrics": concentration_metrics,
      "student_structure_metrics": student_structure_metrics,
      "finance_staff_metrics": finance_staff_metrics,
      "analysis_readiness_report": analysis_readiness_report,
    }

    for table_name, table in output_tables.items():
      print(f"{table_name}:  {table.shape[0]:,}  rows  x  {table.shape[1]:,}
columns")

    assert                country_year_profiles.duplicated(subset=["country_code",
"year"]).sum() == 0
    assert                concentration_metrics.duplicated(subset=["country_code",
"year"]).sum() == 0
    assert            student_structure_metrics.duplicated(subset=["country_code",
"year"]).sum() == 0
    assert                 finance_staff_metrics.duplicated(subset=["country_code",
"year"]).sum() == 0
    assert analysis_readiness_report["country_code"].is_unique
```

```
print("Final key checks passed.")


# Cell 15 (notebook cell 31)
institution_year_metrics.to_csv(INSTITUTION_YEAR_METRICS_PATH, index=False)
country_year_profiles.to_csv(COUNTRY_YEAR_PROFILES_PATH, index=False)
concentration_metrics.to_csv(CONCENTRATION_METRICS_PATH, index=False)
student_structure_metrics.to_csv(STUDENT_STRUCTURE_METRICS_PATH, index=False)
finance_staff_metrics.to_csv(FINANCE_STAFF_METRICS_PATH, index=False)
analysis_readiness_report.to_csv(ANALYSIS_READINESS_REPORT_PATH, index=False)

print("Saved files:")
for path in [
    INSTITUTION_YEAR_METRICS_PATH,
    COUNTRY_YEAR_PROFILES_PATH,
    CONCENTRATION_METRICS_PATH,
    STUDENT_STRUCTURE_METRICS_PATH,
    FINANCE_STAFF_METRICS_PATH,
    ANALYSIS_READINESS_REPORT_PATH,
]:
    print(path.relative_to(PROJECT_ROOT))


#NOTEBOOK: 02_descriptive_statistics_and_eda.ipynb
```

```
#CODE CELLS: 46


# Cell 1 (notebook cell 3)
from pathlib import Path
import os

PROJECT_ROOT = Path.cwd().parent if Path.cwd().name == "notebooks" else Path.cwd()
os.environ.setdefault("MPLCONFIGDIR", str(PROJECT_ROOT / ".matplotlib_cache"))

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

sns.set_theme(style="whitegrid", context="notebook")
pd.set_option("display.max_columns", 100)
pd.set_option("display.float_format", lambda value: f"{value:,.4f}")

# Cell 2 (notebook cell 5)
DATA_DIR = PROJECT_ROOT / "data"
PROCESSED_DIR = DATA_DIR / "processed"
FIGURES_DIR = PROJECT_ROOT / "figures"
TABLES_DIR = PROJECT_ROOT / "tables"

FIGURES_DIR.mkdir(parents=True, exist_ok=True)
TABLES_DIR.mkdir(parents=True, exist_ok=True)
```

```
COUNTRY_YEAR_PROFILES_PATH = PROCESSED_DIR / "country_year_profiles.csv"
INSTITUTION_YEAR_METRICS_PATH = PROCESSED_DIR / "institution_year_metrics.csv"

COUNTRY_YEAR_PROFILES_PATH, INSTITUTION_YEAR_METRICS_PATH

# Cell 3 (notebook cell 7)
country_year_profiles = pd.read_csv(COUNTRY_YEAR_PROFILES_PATH)
institution_year_metrics = pd.read_csv(INSTITUTION_YEAR_METRICS_PATH)

print(f"country_year_profiles shape: {country_year_profiles.shape}")
print(f"institution_year_metrics shape: {institution_year_metrics.shape}")

# Cell 4 (notebook cell 9)
required_country_year_columns = [
    "country_code",
    "year",
    "institutions_count",
    "total_students",
    "foreign_students_total",
    "mean_institution_size",
    "median_institution_size",
    "max_institution_size",
    "foreign_students_share",
    "isced5_share",
    "isced6_share",
```

```
        "isced7_share",
        "isced8_share",
        "student_academic_staff_ratio",
        "student_total_staff_ratio",
        "expenditure_per_student_ppp",
        "revenue_per_student_ppp",
        "rd_expenditure_per_student_ppp",
        "top_1_share",
        "top_5_share",
        "top_10_share",
        "gini_students",
    ]

    required_institution_columns = [
        "record_id",
        "institution_id",
        "country_code",
        "year",
        "students_total",
        "foreign_students_share",
        "size_category",
    ]

    missing_country_year_columns = sorted(set(required_country_year_columns) - set(country_year_profiles.columns))
    missing_institution_columns = sorted(set(required_institution_columns) - set(institution_year_metrics.columns))

    country_year_duplicates = country_year_profiles.duplicated(subset=["country_code", "year"]).sum()
```

```
    gini_values_outside_range = int(
        ((country_year_profiles["gini_students"] < 0) | (country_year_profiles["gini_students"] > 1)).sum()
    ) if "gini_students" in country_year_profiles.columns else "missing column"

    quality_checks = pd.DataFrame(
        {
            "check": [
                "missing_country_year_columns",
                "missing_institution_columns",
                "country_year_duplicate_keys",
                "gini_values_outside_0_1",
            ],
            "result": [
                ", ".join(missing_country_year_columns) if missing_country_year_columns else "none",
                ", ".join(missing_institution_columns) if missing_institution_columns else "none",
                country_year_duplicates,
                gini_values_outside_range,
            ],
        }
    )

    quality_checks

    # Cell 5 (notebook cell 10)
    if missing_country_year_columns:
        raise ValueError(f"Missing columns in country_year_profiles: {missing_country_year_columns}")
```

```
    if missing_institution_columns:
        raise ValueError(f"Missing columns in institution_year_metrics: {missing_institution_columns}")
    if country_year_duplicates != 0:
        raise ValueError(f"country_code + year is not unique: {country_year_duplicates} duplicate rows")

    # Cell 6 (notebook cell 12)
    country_year_coverage = (
        institution_year_metrics
        .dropna(subset=["country_code", "year"])
        .groupby(["country_code", "year"], as_index=False)
        .agg(institutions=("institution_id", "nunique"))
    )

    coverage_matrix = (
        country_year_coverage
        .pivot_table(
            index="country_code",
            columns="year",
            values="institutions",
            aggfunc="sum",
        )
        .sort_index()
    )
    coverage_matrix = coverage_matrix.reindex(sorted(coverage_matrix.columns), axis=1)

    plt.figure(figsize=(12, max(6, 0.25 * len(coverage_matrix))))
    sns.heatmap(
```

```
        coverage_matrix.notna(),
        cmap="Greens",
        cbar=False,
        linewidths=0.3,
        linecolor="white",
    )
    plt.title("Покриття даних за країнами та роками")
    plt.xlabel("Рік")
    plt.ylabel("Код країни")
    plt.tight_layout()

    coverage_figure_path = FIGURES_DIR / "02_country_year_coverage_heatmap.png"
    plt.savefig(coverage_figure_path, dpi=300, bbox_inches="tight")
    plt.show()

    print(f"Saved figure: {coverage_figure_path}")
    print(f"Countries with at least one year: {coverage_matrix.shape[0]}")
    print(f"Years covered: {coverage_matrix.shape[1]}")

    # Cell 7 (notebook cell 15)
    latest_year = country_year_profiles["year"].max()
    latest_snapshot = country_year_profiles.query("year == @latest_year").copy()

    latest_snapshot_path = TABLES_DIR / "latest_snapshot.csv"
    latest_snapshot.to_csv(latest_snapshot_path, index=False)

    print(f"Selected year: {latest_year}")
    print(f"Countries in {latest_year} snapshot: {latest_snapshot['country_code'].nunique()}")
```

```
print(f"Saved: {latest_snapshot_path}")

# Cell 8 (notebook cell 17)
selected_metrics = [
    "institutions_count",
    "total_students",
    "mean_institution_size",
    "median_institution_size",
    "max_institution_size",
    "foreign_students_total",
    "foreign_students_share",
    "student_academic_staff_ratio",
    "student_total_staff_ratio",
    "expenditure_per_student_ppp",
    "revenue_per_student_ppp",
    "rd_expenditure_per_student_ppp",
    "isced5_share",
    "isced6_share",
    "isced7_share",
    "isced8_share",
    "top_1_share",
    "top_5_share",
    "top_10_share",
    "gini_students",
]

available_selected_metrics = [column for column in selected_metrics if column in latest_snapshot.columns]
descriptive_statistics = latest_snapshot[available_selected_metrics].describe().T
```

```
descriptive_statistics_path = TABLES_DIR / "descriptive_statistics_latest_year.csv"
descriptive_statistics.to_csv(descriptive_statistics_path)

descriptive_statistics

# Cell 9 (notebook cell 20)
def make_ranking_table(data, metric, top_n=10, ascending=False):
    columns = ["country_code", "year", metric]
    return (
        data.loc[:, columns]
        .dropna(subset=[metric])
        .sort_values(metric, ascending=ascending)
        .head(top_n)
        .reset_index(drop=True)
    )

ranking_specs = {
    "top_countries_by_students": "total_students",
    "top_countries_by_institutions": "institutions_count",
    "top_countries_by_foreign_share": "foreign_students_share",
    "top_countries_by_gini": "gini_students",
    "top_countries_by_expenditure_per_student": "expenditure_per_student_ppp",
}

ranking_tables = {}
for table_name, metric in ranking_specs.items():
    ranking_tables[table_name] = make_ranking_table(latest_snapshot, metric)
```

```
    ranking_tables[table_name].to_csv(TABLES_DIR / f"{table_name}.csv",
index=False)

ranking_tables["top_countries_by_students"]

# Cell 10 (notebook cell 21)
ranking_tables["top_countries_by_institutions"]

# Cell 11 (notebook cell 22)
ranking_tables["top_countries_by_foreign_share"]

# Cell 12 (notebook cell 23)
ranking_tables["top_countries_by_gini"]

# Cell 13 (notebook cell 24)
ranking_tables["top_countries_by_expenditure_per_student"]

# Cell 14 (notebook cell 26)
def save_current_figure(file_name):
    path = FIGURES_DIR / file_name
    plt.savefig(path, dpi=300, bbox_inches="tight")
    print(f"Saved figure: {path}")


def format_country_list(countries):
    return ", ".join(str(country) for country in countries)


def print_top_interpretation(data, metric, metric_label, top_n=3):
```

```
    top_data = data.dropna(subset=[metric]).sort_values(metric, ascending=False).head(top_n)
    if top_data.empty:
        print(f"Інтерпретація: для метрики {metric_label} немає достатньо даних у {latest_year} році.")
        return
    countries = format_country_list(top_data["country_code"].tolist())
    print(f"Інтерпретація: найвищі значення за метрикою '{metric_label}' мають країни: {countries}.")
    print("Обмеження: висновок є описовим і залежить від повноти даних у вибраному році.")


def plot_top_bar(data, metric, title, ylabel, file_name, top_n=10):
    plot_data = data.dropna(subset=[metric]).sort_values(metric, ascending=False).head(top_n)
    plt.figure(figsize=(10, 6))
    sns.barplot(data=plot_data, x="country_code", y=metric, color="#2F6F73")
    plt.title(title)
    plt.xlabel("Країна")
    plt.ylabel(ylabel)
    plt.xticks(rotation=45, ha="right")
    plt.tight_layout()
    save_current_figure(file_name)
    plt.show()
    print_top_interpretation(plot_data, metric, ylabel)


def plot_histogram(data, metric, title, xlabel, file_name, bins=20):
    plot_data = data.dropna(subset=[metric])
```

```
    plt.figure(figsize=(9, 5))
    sns.histplot(plot_data[metric], bins=bins, kde=True, color="#49759C")
    plt.title(title)
    plt.xlabel(xlabel)
    plt.ylabel("Кількість країн")
    plt.tight_layout()
    save_current_figure(file_name)
    plt.show()
    if plot_data.empty:
        print(f"Інтерпретація: для {xlabel} немає достатньо даних.")
    else:
        print(
            f"Інтерпретація: розподіл {xlabel} має медіану "
            f"{plot_data[metric].median():,.4f}; можливі крайні значення слід перевіряти з урахуванням пропусків."
        )


def plot_scatter(data, x, y, title, xlabel, ylabel, file_name):
    plot_data = data.dropna(subset=[x, y]).copy()
    plt.figure(figsize=(9, 6))
    sns.scatterplot(data=plot_data, x=x, y=y, s=80, color="#7A5C3E")
    for _, row in plot_data.iterrows():
        plt.text(row[x], row[y], row["country_code"], fontsize=8, alpha=0.75)
    plt.title(title)
    plt.xlabel(xlabel)
    plt.ylabel(ylabel)
    plt.tight_layout()
    save_current_figure(file_name)
    plt.show()
```

```
    if len(plot_data) < 3:
        print("Інтерпретація: недостатньо країн для змістовного порівняння.")
    else:
        corr = plot_data[[x, y]].corr().iloc[0, 1]
        print(f"Інтерпретація: кореляція між '{xlabel}' і '{ylabel}' становить {corr:,.4f}.")
        print("Обмеження: кореляція показує асоціацію, але не доводить причинний зв'язок.")


# Cell 15 (notebook cell 28)
plot_top_bar(
    latest_snapshot,
    metric="total_students",
    title=f"Топ країн за кількістю студентів, {latest_year}",
    ylabel="Загальна кількість студентів",
    file_name="top_countries_by_students.png",
)

# Cell 16 (notebook cell 30)
plot_top_bar(
    latest_snapshot,
    metric="institutions_count",
    title=f"Топ країн за кількістю ЗВО, {latest_year}",
    ylabel="Кількість ЗВО",
    file_name="top_countries_by_institutions.png",
)

# Cell 17 (notebook cell 32)
def plot_scatter_log(data, x, y, title, xlabel, ylabel, file_name):
```

```
    plot_data = data.dropna(subset=[x, y]).copy()
    plot_data = plot_data[(plot_data[x] > 0) & (plot_data[y] > 0)]

    plt.figure(figsize=(9, 6))
    sns.scatterplot(data=plot_data, x=x, y=y, s=80, color="#7A5C3E")

    for _, row in plot_data.iterrows():
        plt.text(row[x], row[y], row["country_code"], fontsize=8, alpha=0.75)

    plt.xscale("log")
    plt.yscale("log")
    plt.title(title)
    plt.xlabel(xlabel)
    plt.ylabel(ylabel)
    plt.tight_layout()
    save_current_figure(file_name)
    plt.show()

    if len(plot_data) < 3:
        print("Інтерпретація: недостатньо країн для змістовного порівняння.")
    else:
        corr = plot_data[[x, y]].corr(method="spearman").iloc[0, 1]
        print(f"Інтерпретація: Spearman correlation між '{x}' і '{y}' становить {corr:,.4f}.")
        print("Обмеження: логарифмічна шкала показує відносні відмінності; нульові та від’ємні значення не включаються.")

# Cell 18 (notebook cell 33)
plot_scatter_log(
```

```
    latest_snapshot,
    x="institutions_count",
    y="total_students",
    title=f"Кількість ЗВО та кількість студентів, {latest_year} (логарифмічна шкала)",
    xlabel="Кількість ЗВО",
    ylabel="Загальна кількість студентів",
    file_name="system_size_scatter_log.png",
)

# Cell 19 (notebook cell 36)
plot_histogram(
    latest_snapshot,
    metric="mean_institution_size",
    title=f"Розподіл середнього розміру ЗВО, {latest_year}",
    xlabel="Середній розмір ЗВО",
    file_name="mean_institution_size_histogram.png",
)

# Cell 20 (notebook cell 37)
plot_scatter_log(
    latest_snapshot,
    x="mean_institution_size",
    y="median_institution_size",
    title=f"Середній та медіанний розмір ЗВО, {latest_year} (логарифмічна шкала)",
    xlabel="Середній розмір ЗВО",
    ylabel="Медіанний розмір ЗВО",
    file_name="mean_vs_median_institution_size_log.png",
)
```

```
# Cell 21 (notebook cell 39)
students_size = institution_year_metrics["students_total"].dropna()
students_size_positive = students_size[students_size > 0]

if students_size_positive.empty:
    print("Недостатньо додатних значень students_total для побудови розподілу розміру ЗВО.")
else:
    p99_size = students_size_positive.quantile(0.99)
    students_size_main = students_size_positive[students_size_positive <= p99_size]

    fig, axes = plt.subplots(1, 2, figsize=(14, 5))

    sns.histplot(students_size_main, bins=60, ax=axes[0], color="#4C78A8")
    axes[0].axvline(
        students_size_positive.median(),
        color="#D55E00",
        linestyle="--",
        linewidth=1.5,
        label="Median",
    )
    axes[0].set_title("Institution size distribution, up to p99")
    axes[0].set_xlabel("Students total")
    axes[0].set_ylabel("Number of institution-year records")
    axes[0].legend(frameon=False)

    sns.histplot(students_size_positive, bins=60, log_scale=True, ax=axes[1], color="#72B7B2")
```

```
        axes[1].axvline(
            students_size_positive.median(),
            color="#D55E00",
            linestyle="--",
            linewidth=1.5,
            label="Median",
        )
        axes[1].set_title("Institution size distribution, log scale")
        axes[1].set_xlabel("Students total, log scale")
        axes[1].set_ylabel("Number of institution-year records")
        axes[1].legend(frameon=False)

        fig.suptitle(
            "Distribution excludes missing and zero student counts; left panel trims values above p99",
            y=1.03,
            fontsize=11,
        )

        plt.tight_layout()
        save_current_figure("institution_size_distribution_institution_year.png")
        plt.show()

        print(
            "Інтерпретація: розподіл розміру ЗВО є асиметричним; "
            "ліва панель показує основну масу значень без верхнього 1%, "
            "а права панель з log scale зберігає повний додатний діапазон. "
            "Висновок обмежений пропусками та виключенням нульових значень students_total."
        )
```

```
# Cell 22 (notebook cell 42)
institution_latest_snapshot = institution_year_metrics.query("year == @latest_year").copy()
institution_size_for_outliers = institution_latest_snapshot.loc[
    institution_latest_snapshot["students_total"] > 0,
    [
        "country_code",
        "institution_id",
        "institution_name_en",
        "institution_name",
        "legal_status",
        "institution_category_std",
        "students_total",
        "foreign_students_share",
        "expenditure_per_student_ppp",
    ],
].copy()

if institution_size_for_outliers.empty:
    df_outliers = pd.DataFrame()
    outlier_students_share_by_country = pd.DataFrame()
    print("Недостатньо додатних значень students_total для IQR-діагностики.")
else:
    q1 = institution_size_for_outliers["students_total"].quantile(0.25)
    q3 = institution_size_for_outliers["students_total"].quantile(0.75)
    iqr = q3 - q1
    outlier_threshold = q3 + 1.5 * iqr
```

```
        df_outliers = (
            institution_size_for_outliers
            .query("students_total > @outlier_threshold")
            .sort_values("students_total", ascending=False)
            .reset_index(drop=True)
        )
        df_outliers.to_csv(TABLES_DIR / "institution_size_outliers_latest_year.csv", index=False)

        country_students_totals = (
            institution_size_for_outliers
            .groupby("country_code", as_index=False)
            .agg(country_total_students=("students_total", "sum"))
        )
        outlier_students_share_by_country = (
            df_outliers
            .groupby("country_code", as_index=False)
            .agg(
                outlier_institutions_count=("institution_id", "nunique"),
                outlier_students_total=("students_total", "sum"),
            )
            .merge(country_students_totals, on="country_code", how="left")
        )
        outlier_students_share_by_country["outlier_students_share"] = np.where(
            outlier_students_share_by_country["country_total_students"] > 0,
            outlier_students_share_by_country["outlier_students_total"] / outlier_students_share_by_country["country_total_students"],
            np.nan,
        )
```

```
        outlier_students_share_by_country = outlier_students_share_by_country.sort_values(
            "outlier_students_share", ascending=False
        )
        outlier_students_share_by_country.to_csv(
            TABLES_DIR / "outlier_students_share_by_country_latest_year.csv",
            index=False,
        )

        print(f"Q1={q1:,.2f}, Q3={q3:,.2f}, IQR={iqr:,.2f}, outlier threshold={outlier_threshold:,.2f}")
        print(f"Кількість ЗВО-гігантів за правилом IQR: {len(df_outliers):,}")

    df_outliers.head(20)

    # Cell 23 (notebook cell 44)
    if institution_size_for_outliers.empty:
        print("Недостатньо даних для boxplot розміру ЗВО.")
    else:
        top_10_countries_for_boxplot = (
            latest_snapshot.dropna(subset=["total_students"])
            .sort_values("total_students", ascending=False)
            .head(10)["country_code"]
            .tolist()
        )
        boxplot_data = institution_size_for_outliers.query("country_code in @top_10_countries_for_boxplot").copy()

        plt.figure(figsize=(11, 7))
```

```
        ax = sns.boxplot(data=boxplot_data, x="students_total", y="country_code",
color="#8FB9A8")
        ax.set_xscale("log")
        ax.axvline(outlier_threshold, color="#C44E52", linestyle="--",
linewidth=1.5, label="IQR outlier threshold")
        ax.set_title(f"Institution size outliers by country, {latest_year} (log scale)")
        ax.set_xlabel("Students total, log scale")
        ax.set_ylabel("Country")
        ax.legend(frameon=False)
        plt.tight_layout()

save_current_figure("boxplot_institution_size_outliers_log_latest_year.png")
        plt.show()

        display(outlier_students_share_by_country.head(10))
        print(
          "Інтерпретація: якщо частка студентів у IQR-викидах висока,
середній розмір ЗВО для країни може бути "
          "суттєво зміщений вгору. У наступних гіпотезах варто порівнювати
mean і median."
        )

    # Cell 24 (notebook cell 46)
    institution_latest_snapshot = institution_year_metrics.query("year ==
@latest_year").copy()

    status_columns = [
      "country_code",
      "institution_id",
      "legal_status",
```

```
    "institution_category_std",
    "students_total",
]
missing_status_columns = [col for col in status_columns if col not in institution_latest_snapshot.columns]
if missing_status_columns:
    raise ValueError(f"Missing columns for legal status EDA: {missing_status_columns}")

legal_status_labels = {
    0: "Public",
    1: "Private",
    2: "Private government-dependent",
}

institution_category_std_labels = {
    0: "Other",
    1: "University",
    2: "University of applied sciences",
}


def label_eter_code(series, labels):
    numeric_codes = pd.to_numeric(series, errors="coerce")
    labelled = numeric_codes.map(labels)
    return labelled.fillna("Missing / not reported")


institution_latest_snapshot["legal_status_label"] = label_eter_code(
    institution_latest_snapshot["legal_status"],
```

```
        legal_status_labels,
    )
    institution_latest_snapshot["institution_category_std_label"] =
label_eter_code(
        institution_latest_snapshot["institution_category_std"],
        institution_category_std_labels,
    )

    status_missing_by_country = (
        institution_latest_snapshot
        .groupby("country_code", as_index=False)
        .agg(
            institutions_count=("institution_id", "nunique"),
            missing_legal_status=("legal_status", lambda x: x.isna().mean()),
            missing_institution_category_std=("institution_category_std", lambda x:
x.isna().mean()),
        )
        .sort_values(["missing_legal_status",  "missing_institution_category_std"],
ascending=False)
    )
    status_missing_by_country.to_csv(TABLES_DIR /
"legal_status_category_missing_by_country_latest.csv", index=False)

    print(f"Institution-year snapshot for status EDA:
{institution_latest_snapshot.shape[0]:,} rows, {latest_year}")
    print("Частка пропусків у дескрипторах:")
    display(
        institution_latest_snapshot[["legal_status", "institution_category_std"]]
        .isna()
        .mean()
```

```
    .rename("missing_share")
    .reset_index()
    .rename(columns={"index": "variable"})
)
status_missing_by_country.head(10)

# Cell 25 (notebook cell 48)
def make_distribution_table(data, group_column, output_file_name):
    distribution = (
        data
        .groupby(group_column, dropna=False)
        .agg(
            institutions_count=("institution_id", "nunique"),
            students_total=("students_total", lambda x: x.sum(min_count=1)),
        )
        .reset_index()
    )

    total_institutions = distribution["institutions_count"].sum()
    total_students = distribution["students_total"].sum()
    distribution["institution_share"] = np.where(
        total_institutions > 0,
        distribution["institutions_count"] / total_institutions,
        np.nan,
    )
    distribution["student_share"] = np.where(
        total_students > 0,
        distribution["students_total"] / total_students,
        np.nan,
    )
```

```
    distribution = distribution.sort_values("institutions_count",
ascending=False)
    distribution.to_csv(TABLES_DIR / output_file_name, index=False)
    return distribution


legal_status_distribution = make_distribution_table(
    institution_latest_snapshot,
    "legal_status_label",
    "legal_status_distribution_latest.csv",
)

institution_category_distribution = make_distribution_table(
    institution_latest_snapshot,
    "institution_category_std_label",
    "institution_category_distribution_latest.csv",
)

print("Розподіл за legal_status:")
display(legal_status_distribution)
print("Розподіл за institution_category_std:")
display(institution_category_distribution)

# Cell 26 (notebook cell 49)
fig, axes = plt.subplots(1, 2, figsize=(14, 5))

sns.barplot(
    data=legal_status_distribution,
    x="legal_status_label",
    y="institutions_count",
```

```
    color="#4F7C8A",
    ax=axes[0],
)
axes[0].set_title(f"Розподіл ЗВО за legal_status, {latest_year}")
axes[0].set_xlabel("Правовий статус")
axes[0].set_ylabel("Кількість ЗВО")
axes[0].tick_params(axis="x", rotation=25)

sns.barplot(
    data=legal_status_distribution,
    x="legal_status_label",
    y="student_share",
    color="#C17C54",
    ax=axes[1],
)
axes[1].set_title(f"Частка студентів за legal_status, {latest_year}")
axes[1].set_xlabel("Правовий статус")
axes[1].set_ylabel("Частка студентів")
axes[1].tick_params(axis="x", rotation=25)

plt.tight_layout()
save_current_figure("legal_status_distribution_latest.png")
plt.show()

main_legal_status = legal_status_distribution.iloc[0]
print(
    "Інтерпретація: найбільша група за кількістю ЗВО - "
    f"{main_legal_status['legal_status_label']} "
    f"({main_legal_status['institution_share']:.1%} установ)."
)
```

```
print(
    "Обмеження: legal_status описує юридичний тип закладу, але не дозволяє робити висновки "
    "про якість, ефективність або фактичний рівень державного фінансування."
)

# Cell 27 (notebook cell 50)
fig, axes = plt.subplots(1, 2, figsize=(14, 5))

sns.barplot(
    data=institution_category_distribution,
    x="institution_category_std_label",
    y="institutions_count",
    color="#5E8C61",
    ax=axes[0],
)
axes[0].set_title(f"Розподіл ЗВО за institution_category_std, {latest_year}")
axes[0].set_xlabel("Стандартизована категорія")
axes[0].set_ylabel("Кількість ЗВО")
axes[0].tick_params(axis="x", rotation=25)

sns.barplot(
    data=institution_category_distribution,
    x="institution_category_std_label",
    y="student_share",
    color="#B98B39",
    ax=axes[1],
)
```

```
axes[1].set_title(f"Частка студентів за institution_category_std, {latest_year}")
axes[1].set_xlabel("Стандартизована категорія")
axes[1].set_ylabel("Частка студентів")
axes[1].tick_params(axis="x", rotation=25)

plt.tight_layout()
save_current_figure("institution_category_distribution_latest.png")
plt.show()

main_category = institution_category_distribution.iloc[0]
print(
    "Інтерпретація: найбільша стандартизована категорія за кількістю ЗВО - "
    f"{main_category['institution_category_std_label']} "
    f"({main_category['institution_share']:.1%} установ)."
)
print(
    "Обмеження: institution_category_std є узагальненою ETER-класифікацією і може згладжувати "
    "національні відмінності у типах закладів."
)

# Cell 28 (notebook cell 52)
def make_country_distribution_table(data, group_column, output_file_name):
    country_distribution = (
        data
        .groupby(["country_code", group_column], dropna=False)
        .agg(
            institutions_count=("institution_id", "nunique"),
```

```
                students_total=("students_total", lambda x: x.sum(min_count=1)),
            )
            .reset_index()
        )
        country_totals = (
            country_distribution
            .groupby("country_code", as_index=False)
            .agg(
                country_institutions_count=("institutions_count", "sum"),
                country_students_total=("students_total", lambda x: x.sum(min_count=1)),
            )
        )
        country_distribution = country_distribution.merge(country_totals, on="country_code", how="left")
        country_distribution["institution_share_within_country"] = np.where(
            country_distribution["country_institutions_count"] > 0,
            country_distribution["institutions_count"] / country_distribution["country_institutions_count"],
            np.nan,
        )
        country_distribution["student_share_within_country"] = np.where(
            country_distribution["country_students_total"] > 0,
            country_distribution["students_total"] / country_distribution["country_students_total"],
            np.nan,
        )
        country_distribution = country_distribution.sort_values(["country_code", group_column])
```

```
    country_distribution.to_csv(TABLES_DIR / output_file_name, index=False)
    return country_distribution


def plot_country_stacked_structure(data, group_column, title, file_name, top_n=20):
    top_countries = (
        data[["country_code", "country_institutions_count"]]
        .drop_duplicates()
        .sort_values("country_institutions_count", ascending=False)
        .head(top_n)["country_code"]
        .tolist()
    )
    plot_data = data.query("country_code in @top_countries").copy()
    pivot_data = (
        plot_data
        .pivot_table(
            index="country_code",
            columns=group_column,
            values="institution_share_within_country",
            aggfunc="sum",
            fill_value=0,
        )
        .loc[top_countries]
    )

    ax = pivot_data.plot(
        kind="bar",
        stacked=True,
```

```
        figsize=(13, 6),
        colormap="Set2",
        width=0.82,
    )
    ax.set_title(title)
    ax.set_xlabel("Країна")
    ax.set_ylabel("Частка ЗВО всередині країни")
    ax.legend(title="Категорія", bbox_to_anchor=(1.02, 1), loc="upper left")
    plt.xticks(rotation=45, ha="right")
    plt.tight_layout()
    save_current_figure(file_name)
    plt.show()
    return pivot_data


country_legal_status_distribution = make_country_distribution_table(
    institution_latest_snapshot,
    "legal_status_label",
    "country_legal_status_distribution_latest.csv",
)

country_institution_category_distribution = make_country_distribution_table(
    institution_latest_snapshot,
    "institution_category_std_label",
    "country_institution_category_distribution_latest.csv",
)

country_legal_status_pivot = plot_country_stacked_structure(
    country_legal_status_distribution,
    "legal_status_label",
```

```
    f"Структура ЗВО за legal_status у країнах, {latest_year}",
    "country_legal_status_structure_latest.png",
)

country_legal_status_distribution.head(12)

# Cell 29 (notebook cell 54)
country_institution_category_pivot = plot_country_stacked_structure(
    country_institution_category_distribution,
    "institution_category_std_label",
    f"Структура ЗВО за institution_category_std у країнах, {latest_year}",
    "country_institution_category_structure_latest.png",
)

country_institution_category_distribution.head(12)

# Cell 30 (notebook cell 57)
plot_top_bar(
    latest_snapshot,
    metric="foreign_students_share",
    title=f"Топ країн за часткою іноземних студентів, {latest_year}",
    ylabel="Частка іноземних студентів",
    file_name="foreign_students_share_ranking.png",
)

# Cell 31 (notebook cell 58)
plot_histogram(
    latest_snapshot,
    metric="foreign_students_share",
    title=f"Розподіл частки іноземних студентів, {latest_year}",
```

```
    xlabel="Частка іноземних студентів",
    file_name="foreign_students_share_histogram.png",
)

# Cell 32 (notebook cell 59)
plot_scatter(
    latest_snapshot,
    x="mean_institution_size",
    y="foreign_students_share",
    title=f"Розмір ЗВО та частка іноземних студентів, {latest_year}",
    xlabel="Середній розмір ЗВО",
    ylabel="Частка іноземних студентів",
    file_name="foreign_share_vs_mean_institution_size.png",
)

# Cell 33 (notebook cell 62)
plot_top_bar(
    latest_snapshot,
    metric="gini_students",
    title=f"Топ країн за коефіцієнтом Gini студентів, {latest_year}",
    ylabel="Gini за кількістю студентів",
    file_name="student_concentration_gini.png",
)

# Cell 34 (notebook cell 63)
plot_scatter(
    latest_snapshot,
    x="top_5_share",
    y="gini_students",
    title=f"Top-5 share та Gini студентів, {latest_year}",
```

```
    xlabel="Частка студентів у 5 найбільших ЗВО",
    ylabel="Gini за кількістю студентів",
    file_name="top5_share_vs_gini.png",
)

# Cell 35 (notebook cell 64)
plot_scatter(
    latest_snapshot,
    x="top_10_share",
    y="gini_students",
    title=f"Top-10 share та Gini студентів, {latest_year}",
    xlabel="Частка студентів у 10 найбільших ЗВО",
    ylabel="Gini за кількістю студентів",
    file_name="top10_share_vs_gini.png",
)

# Cell 36 (notebook cell 67)
isced_share_columns = ["isced5_share", "isced6_share", "isced7_share",
"isced8_share"]

isced_plot_data = (
    latest_snapshot.dropna(subset=["total_students"])
    .sort_values("total_students", ascending=False)
    .head(12)
    .loc[:, ["country_code"] + isced_share_columns]
    .melt(id_vars="country_code",                    var_name="isced_level",
value_name="share")
    .dropna(subset=["share"])
)
```

```
plt.figure(figsize=(12, 6))
sns.barplot(data=isced_plot_data, x="country_code", y="share", hue="isced_level")
plt.title(f"Структура студентів за ISCED у найбільших системах, {latest_year}")
plt.xlabel("Країна")
plt.ylabel("Частка студентів")
plt.xticks(rotation=45, ha="right")
plt.legend(title="ISCED")
plt.tight_layout()
save_current_figure("isced_structure_top_countries.png")
plt.show()

if isced_plot_data.empty:
    print("Інтерпретація: недостатньо даних для порівняння структури ISCED.")
else:
    dominant_levels = (
        isced_plot_data.sort_values("share", ascending=False)
        .groupby("country_code")
        .head(1)
        .loc[:, ["country_code", "isced_level", "share"]]
    )
    display(dominant_levels)
    print("Інтерпретація: таблиця показує рівень ISCED з найбільшою часткою серед відібраних країн.")
    print("Обмеження: сума часток ISCED може бути меншою за 1 через пропуски або неповне покриття рівнів.")

# Cell 37 (notebook cell 70)
```

```
plot_histogram(
    latest_snapshot,
    metric="student_academic_staff_ratio",
    title=f"Розподіл student-academic staff ratio, {latest_year}",
    xlabel="Студентів на одного академічного працівника FTE",
    file_name="student_academic_staff_ratio_histogram.png",
)

# Cell 38 (notebook cell 71)
plot_scatter(
    latest_snapshot,
    x="student_academic_staff_ratio",
    y="mean_institution_size",
    title=f"Student-staff ratio та середній розмір ЗВО, {latest_year}",
    xlabel="Студентів на одного академічного працівника FTE",
    ylabel="Середній розмір ЗВО",
    file_name="student_staff_ratio_vs_mean_size.png",
)

# Cell 39 (notebook cell 72)
plot_scatter(
    latest_snapshot,
    x="expenditure_per_student_ppp",
    y="student_academic_staff_ratio",
    title=f"Витрати на студента PPP та student-staff ratio, {latest_year}",
    xlabel="Витрати на студента, PPP",
    ylabel="Студентів на одного академічного працівника FTE",
    file_name="expenditure_per_student_vs_staff_ratio.png",
)
```

```
# Cell 40 (notebook cell 75)
correlation_metrics = [
    "institutions_count",
    "total_students",
    "mean_institution_size",
    "median_institution_size",
    "foreign_students_share",
    "student_academic_staff_ratio",
    "expenditure_per_student_ppp",
    "isced7_share",
    "isced8_share",
    "top_5_share",
    "top_10_share",
    "gini_students",
]

available_correlation_metrics = [column for column in correlation_metrics if column in latest_snapshot.columns]
correlation_data = latest_snapshot[["country_code"] + available_correlation_metrics].copy()

pearson_correlation_matrix = correlation_data[available_correlation_metrics].corr(method="pearson")
spearman_correlation_matrix = correlation_data[available_correlation_metrics].corr(method="spearman")

pearson_correlation_matrix.to_csv(TABLES_DIR / "pearson_correlation_matrix_latest_year.csv")
spearman_correlation_matrix.to_csv(TABLES_DIR / "spearman_correlation_matrix_latest_year.csv")
```

```
fig, axes = plt.subplots(1, 2, figsize=(22, 9))
sns.heatmap(
    pearson_correlation_matrix,
    annot=True,
    fmt=".2f",
    cmap="BrBG",
    center=0,
    linewidths=0.5,
    ax=axes[0],
)
axes[0].set_title(f"Pearson correlations, {latest_year}")

sns.heatmap(
    spearman_correlation_matrix,
    annot=True,
    fmt=".2f",
    cmap="BrBG",
    center=0,
    linewidths=0.5,
    ax=axes[1],
)
axes[1].set_title(f"Spearman correlations, {latest_year}")

plt.tight_layout()
save_current_figure("correlation_heatmaps_latest_year.png")
plt.show()

plt.figure(figsize=(11, 9))
sns.heatmap(
```

```
        spearman_correlation_matrix,
        annot=True,
        fmt=".2f",
        cmap="BrBG",
        center=0,
        linewidths=0.5,
    )
    plt.title(f"Spearman correlation profile, {latest_year}")
    plt.tight_layout()
    save_current_figure("spearman_correlation_heatmap_latest_year.png")
    plt.show()

    print(
        "Інтерпретація: Spearman correlation є основною для цього EDA-блоку,
"
        "бо вона краще підходить для скошених розподілів і монотонних, але не обов'язково лінійних зв'язків."
    )

    spearman_correlation_matrix

    # Cell 41 (notebook cell 78)
    institution_latest_snapshot = institution_year_metrics.query("year == @latest_year").copy()
    institution_size_positive = institution_latest_snapshot.loc[
        institution_latest_snapshot["students_total"] > 0,
        ["country_code", "institution_id", "students_total"],
    ].copy()

    top_5_countries_by_students = (
```

```
            latest_snapshot.dropna(subset=["total_students"])
            .sort_values("total_students", ascending=False)
            .head(5)["country_code"]
            .tolist()
        )

        institution_size_top5 = institution_size_positive.query("country_code in @top_5_countries_by_students").copy()

        shape_diagnostics_records = []
        if not institution_size_positive.empty:
            all_size_values = institution_size_positive["students_total"].dropna()
            shape_diagnostics_records.append(
                {
                    "scope": "all_institutions_latest_year",
                    "country_code": "ALL",
                    "metric": "students_total",
                    "observations": all_size_values.count(),
                    "mean": all_size_values.mean(),
                    "median": all_size_values.median(),
                    "std": all_size_values.std(),
                    "skewness": all_size_values.skew(),
                    "kurtosis": all_size_values.kurtosis(),
                    "p90": all_size_values.quantile(0.90),
                    "p95": all_size_values.quantile(0.95),
                    "p99": all_size_values.quantile(0.99),
                    "max": all_size_values.max(),
                    "mean_to_median_ratio": np.where(all_size_values.median() > 0, all_size_values.mean() / all_size_values.median(), np.nan),
                    "top_5_percent_share": (
```

```
                    all_size_values.nlargest(max(1, int(np.ceil(all_size_values.count() *
0.05)))).sum() / all_size_values.sum()
                    if all_size_values.sum() > 0 else np.nan
                ),
            }
        )

    for country_code, group in institution_size_top5.groupby("country_code"):
        size_values = group["students_total"].dropna()
        shape_diagnostics_records.append(
            {
                "scope": "top_5_country_latest_year",
                "country_code": country_code,
                "metric": "students_total",
                "observations": size_values.count(),
                "mean": size_values.mean(),
                "median": size_values.median(),
                "std": size_values.std(),
                "skewness": size_values.skew(),
                "kurtosis": size_values.kurtosis(),
                "p90": size_values.quantile(0.90),
                "p95": size_values.quantile(0.95),
                "p99": size_values.quantile(0.99),
                "max": size_values.max(),
                "mean_to_median_ratio": np.where(size_values.median() > 0,
size_values.mean() / size_values.median(), np.nan),
                "top_5_percent_share": (
                    size_values.nlargest(max(1, int(np.ceil(size_values.count() *
0.05)))).sum() / size_values.sum()
                    if size_values.sum() > 0 else np.nan
```

```
            ),
        }
    )

finance_values = latest_snapshot["expenditure_per_student_ppp"].dropna()
finance_values = finance_values[finance_values > 0]
if not finance_values.empty:
    shape_diagnostics_records.append(
        {
            "scope": "country_year_latest_year",
            "country_code": "ALL",
            "metric": "expenditure_per_student_ppp",
            "observations": finance_values.count(),
            "mean": finance_values.mean(),
            "median": finance_values.median(),
            "std": finance_values.std(),
            "skewness": finance_values.skew(),
            "kurtosis": finance_values.kurtosis(),
            "p90": finance_values.quantile(0.90),
            "p95": finance_values.quantile(0.95),
            "p99": finance_values.quantile(0.99),
            "max": finance_values.max(),
            "mean_to_median_ratio": np.where(finance_values.median() > 0, finance_values.mean() / finance_values.median(), np.nan),
            "top_5_percent_share": np.nan,
        }
    )

institution_size_distribution_diagnostics = pd.DataFrame(shape_diagnostics_records)
```

```
    institution_size_distribution_diagnostics.to_csv(
        TABLES_DIR / "institution_size_distribution_diagnostics_latest_year.csv",
        index=False,
    )

    if institution_size_top5.empty:
        print("Недостатньо даних для top-5 країн за total_students.")
    else:
        kde_data = institution_size_top5.copy()
        kde_data["students_total_log10"] = np.log10(kde_data["students_total"])
        kde_data = kde_data.groupby("country_code").filter(lambda group: group["students_total_log10"].nunique() >= 2)

        if kde_data.empty:
            print("Недостатньо різних значень для KDE-графіка.")
        else:
            plt.figure(figsize=(11, 6))
            sns.kdeplot(
                data=kde_data,
                x="students_total_log10",
                hue="country_code",
                common_norm=False,
                linewidth=2,
            )
            plt.title(f"KDE of institution size for top-5 systems, {latest_year}")
            plt.xlabel("log10(students_total)")
            plt.ylabel("Density")
            plt.tight_layout()

save_current_figure("kde_institution_size_top5_countries_latest_year.png")
```

```
    plt.show()

    print(
      "Інтерпретація: правостороння асиметрія та довгий хвіст означають, що mean може переоцінювати "
      "типовий розмір ЗВО. Для гіпотез безпечніше порівнювати median та непараметричні асоціації."
    )

institution_size_distribution_diagnostics

# Cell 42 (notebook cell 81)
def calculate_lowess_curve(x_values, y_values, frac=0.65):
  """Small local-linear LOWESS smoother without an extra statsmodels dependency."""
  curve_data = pd.DataFrame({"x": x_values, "y": y_values}).dropna().sort_values("x")
  if len(curve_data) < 4:
    return pd.DataFrame(columns=["x", "y_smooth"])

  x = curve_data["x"].to_numpy(dtype=float)
  y = curve_data["y"].to_numpy(dtype=float)
  window = max(3, int(np.ceil(frac * len(curve_data))))
  smoothed = []

  for x0 in x:
    distances = np.abs(x - x0)
    nearest_idx = np.argsort(distances)[:window]
    x_window = x[nearest_idx]
    y_window = y[nearest_idx]
```

```
            max_distance = distances[nearest_idx].max()

            if max_distance == 0 or np.unique(x_window).size < 2:
                smoothed.append(np.nanmean(y_window))
                continue

            weights = (1 - (np.abs(x_window - x0) / max_distance) ** 3) ** 3
            try:
                slope, intercept = np.polyfit(x_window, y_window, deg=1, w=weights)
                smoothed.append(slope * x0 + intercept)
            except np.linalg.LinAlgError:
                smoothed.append(np.average(y_window, weights=weights))

        return pd.DataFrame({"x": x, "y_smooth": smoothed})


    def plot_lowess_scatter(data, x, y, title, xlabel, ylabel, file_name,
log_x=False):
        required_columns = ["country_code", x, y]
        plot_data = data.dropna(subset=[x, y]).loc[:, required_columns].copy()

        if log_x:
            plot_data = plot_data[plot_data[x] > 0].copy()
            transformed_x = f"log10_{x}"
            plot_data[transformed_x] = np.log10(plot_data[x])
            x_for_plot = transformed_x
            xlabel_for_plot = f"log10({xlabel})"
        else:
            x_for_plot = x
            xlabel_for_plot = xlabel
```

```
        if len(plot_data) < 4:
            print(f"Недостатньо спостережень для LOWESS-графіка {x} vs {y}.")
            return pd.DataFrame()

        lowess_curve = calculate_lowess_curve(plot_data[x_for_plot], plot_data[y], frac=0.65)

        plt.figure(figsize=(9, 6))
        sns.scatterplot(data=plot_data, x=x_for_plot, y=y, s=80, color="#7A5C3E", alpha=0.85)
        if not lowess_curve.empty:
            sns.lineplot(data=lowess_curve, x="x", y="y_smooth", color="#C44E52", linewidth=2.5, label="LOWESS")
        for _, row in plot_data.iterrows():
            plt.text(row[x_for_plot], row[y], row["country_code"], fontsize=8, alpha=0.75)
        plt.title(title)
        plt.xlabel(xlabel_for_plot)
        plt.ylabel(ylabel)
        plt.tight_layout()
        save_current_figure(file_name)
        plt.show()

        pearson_value = plot_data[[x_for_plot, y]].corr(method="pearson").iloc[0, 1]
        spearman_value = plot_data[[x_for_plot, y]].corr(method="spearman").iloc[0, 1]
        result = pd.DataFrame(
            [
```

```
                {
                    "x_metric": x,
                    "y_metric": y,
                    "x_transformation": "log10" if log_x else "none",
                    "observations": len(plot_data),
                    "pearson_correlation": pearson_value,
                    "spearman_correlation": spearman_value,
                }
            ]
        )
        print(
            f"Інтерпретація: Pearson={pearson_value:,.4f}, Spearman={spearman_value:,.4f}. "
            "LOWESS-лінія показує можливу нелінійну форму зв'язку, але не є причинним доказом."
        )
        return result

    bivariate_results = pd.concat(
        [
            plot_lowess_scatter(
                latest_snapshot,
                x="total_students",
                y="foreign_students_share",
                title=f"System size and foreign students share, {latest_year} (LOWESS)",
                xlabel="Загальна кількість студентів",
                ylabel="Частка іноземних студентів",
                file_name="lowess_total_students_vs_foreign_share_latest_year.png",
                log_x=True,
```

```
        ),
        plot_lowess_scatter(
            latest_snapshot,
            x="expenditure_per_student_ppp",
            y="foreign_students_share",
            title=f"Expenditure per student PPP and foreign students share, {latest_year} (LOWESS)",
            xlabel="Витрати на студента, PPP",
            ylabel="Частка іноземних студентів",
            file_name="lowess_expenditure_vs_foreign_share_latest_year.png",
        ),
        plot_lowess_scatter(
            latest_snapshot,
            x="gini_students",
            y="foreign_students_share",
            title=f"Student concentration and foreign students share, {latest_year} (LOWESS)",
            xlabel="Gini студентів",
            ylabel="Частка іноземних студентів",
            file_name="lowess_gini_vs_foreign_share_latest_year.png",
        ),
        plot_lowess_scatter(
            latest_snapshot,
            x="expenditure_per_student_ppp",
            y="student_academic_staff_ratio",
            title=f"Expenditure per student PPP and student-staff ratio, {latest_year} (LOWESS)",
            xlabel="Витрати на студента, PPP",
            ylabel="Студентів на одного академічного працівника FTE",
```

```
file_name="lowess_expenditure_vs_student_staff_ratio_latest_year.png",
                ),
            ],
            ignore_index=True,
        )

        bivariate_results.to_csv(TABLES_DIR / "bivariate_eda_correlations_latest_year.csv", index=False)
        bivariate_results

        # Cell 43 (notebook cell 84)
        grouped_latest_snapshot = latest_snapshot.copy()

        def add_median_group(data, metric, group_column, low_label, high_label):
            data[group_column] = pd.Series(pd.NA, index=data.index, dtype="object")
            valid_mask = data[metric].notna()
            median_value = data.loc[valid_mask, metric].median()
            data.loc[valid_mask, group_column] = np.where(
                data.loc[valid_mask, metric] >= median_value,
                high_label,
                low_label,
            )
            return median_value

        finance_median = add_median_group(
            grouped_latest_snapshot,
            "expenditure_per_student_ppp",
            "finance_group",
            "lower expenditure per student",
```

```
        "higher expenditure per student",
    )
    size_median = add_median_group(
        grouped_latest_snapshot,
        "total_students",
        "system_size_group",
        "smaller systems",
        "larger systems",
    )
    concentration_median = add_median_group(
        grouped_latest_snapshot,
        "gini_students",
        "concentration_group",
        "lower concentration",
        "higher concentration",
    )


    def plot_group_boxplot(data, group_column, metric, title, xlabel, ylabel, file_name):
        plot_data = data.dropna(subset=[group_column, metric]).copy()
        if plot_data[group_column].nunique() < 2:
            print(f"Недостатньо груп для графіка {group_column} vs {metric}.")
            return pd.DataFrame()

        plt.figure(figsize=(9, 6))
        sns.boxplot(data=plot_data, x=group_column, y=metric, color="#72B7B2")
        sns.stripplot(data=plot_data, x=group_column, y=metric, color="#333333", alpha=0.65, jitter=0.15)
        plt.title(title)
```

```
        plt.xlabel(xlabel)
        plt.ylabel(ylabel)
        plt.xticks(rotation=15, ha="right")
        plt.tight_layout()
        save_current_figure(file_name)
        plt.show()

        summary = (
            plot_data.groupby(group_column)[metric]
            .agg(["count", "mean", "median", "std", "min", "max"])
            .reset_index()
            .rename(columns={group_column: "group", metric: "metric"})
        )
        summary.insert(0, "grouping_variable", group_column)
        summary.insert(1, "target_metric", metric)
        return summary

    group_comparison_summary = pd.concat(
        [
            plot_group_boxplot(
                grouped_latest_snapshot,
                group_column="finance_group",
                metric="foreign_students_share",
                title=f"Foreign students share by expenditure group, {latest_year}",
                xlabel="Група за витратами на студента",
                ylabel="Частка іноземних студентів",

file_name="boxplot_foreign_share_by_finance_group_latest_year.png",
            ),
            plot_group_boxplot(
```

```
                grouped_latest_snapshot,
                group_column="system_size_group",
                metric="gini_students",
                title=f"Student concentration by system size group, {latest_year}",
                xlabel="Група за розміром системи",
                ylabel="Gini студентів",
                file_name="boxplot_gini_by_system_size_group_latest_year.png",
            ),
            plot_group_boxplot(
                grouped_latest_snapshot,
                group_column="concentration_group",
                metric="foreign_students_share",
                title=f"Foreign students share by concentration group, {latest_year}",
                xlabel="Група за концентрацією студентів",
                ylabel="Частка іноземних студентів",

file_name="boxplot_foreign_share_by_concentration_group_latest_year.png",
            ),
        ],
        ignore_index=True,
    )

    group_comparison_summary.to_csv(TABLES_DIR / "group_comparison_summary_latest_year.csv", index=False)
    print(f"Finance median: {finance_median:,.4f}")
    print(f"System size median: {size_median:,.4f}")
    print(f"Concentration median: {concentration_median:,.4f}")
    group_comparison_summary

    # Cell 44 (notebook cell 87)
```

```
top_10_systems_by_students = (
    latest_snapshot.dropna(subset=["total_students"])
    .sort_values("total_students", ascending=False)
    .head(10)["country_code"]
    .tolist()
)

trend_metrics = ["foreign_students_share", "total_students", "gini_students", "mean_institution_size"]
trend_data = country_year_profiles.query("country_code in @top_10_systems_by_students").copy()

def calculate_metric_delta(data, metric):
    records = []
    for country_code, group in data.dropna(subset=[metric]).groupby("country_code"):
        ordered = group.sort_values("year")
        if len(ordered) < 2:
            continue
        first_row = ordered.iloc[0]
        last_row = ordered.iloc[-1]
        records.append(
            {
                "country_code": country_code,
                "metric": metric,
                "first_year": int(first_row["year"]),
                "last_year": int(last_row["year"]),
                "first_value": first_row[metric],
                "last_value": last_row[metric],
                "absolute_delta": last_row[metric] - first_row[metric],
```

```
                "relative_delta": (
                    (last_row[metric] - first_row[metric]) / first_row[metric]
                    if pd.notna(first_row[metric]) and first_row[metric] != 0
                    else np.nan
                ),
                "years_observed": len(ordered),
            }
        )
    return pd.DataFrame(records)

key_metric_deltas_by_country = pd.concat(
    [calculate_metric_delta(country_year_profiles, metric) for metric in trend_metrics],
    ignore_index=True,
)
key_metric_deltas_by_country.to_csv(TABLES_DIR / "key_metric_deltas_by_country.csv", index=False)

yoy_metrics = ["foreign_students_share", "total_students"]
yoy_change_data = country_year_profiles[["country_code", "year"] + yoy_metrics].copy()
yoy_change_data = yoy_change_data.sort_values(["country_code", "year"])

for metric in yoy_metrics:
    previous_metric = f"previous_{metric}"
    yoy_metric = f"{metric}_yoy_pct_change"
    yoy_change_data[previous_metric] = yoy_change_data.groupby("country_code")[metric].shift(1)
    yoy_change_data[yoy_metric] = np.where(
        yoy_change_data[previous_metric] > 0,
```

```
            (yoy_change_data[metric] - yoy_change_data[previous_metric]) / yoy_change_data[previous_metric],
            np.nan,
        )

    yoy_change_data.to_csv(TABLES_DIR / "yoy_change_country_metrics.csv", index=False)

    yoy_long_records = []
    for metric in yoy_metrics:
        yoy_metric = f"{metric}_yoy_pct_change"
        previous_metric = f"previous_{metric}"
        metric_data = yoy_change_data.dropna(subset=[yoy_metric]).copy()
        if metric_data.empty:
            continue
        anomaly_threshold = metric_data[yoy_metric].abs().quantile(0.95)
        anomalies = metric_data[metric_data[yoy_metric].abs() >= anomaly_threshold].copy()
        for _, row in anomalies.iterrows():
            yoy_long_records.append(
                {
                    "country_code": row["country_code"],
                    "year": int(row["year"]),
                    "metric": metric,
                    "value": row[metric],
                    "previous_value": row[previous_metric],
                    "yoy_pct_change": row[yoy_metric],
                    "abs_yoy_pct_change": abs(row[yoy_metric]),
                    "anomaly_threshold_abs": anomaly_threshold,
                }
```

```
            )

        yoy_anomalies_country_metrics =
pd.DataFrame(yoy_long_records).sort_values(
            ["metric", "abs_yoy_pct_change"], ascending=[True, False]
        )
        yoy_anomalies_country_metrics.to_csv(TABLES_DIR /
"yoy_anomalies_country_metrics.csv", index=False)

        for metric in trend_metrics:
            plot_data = trend_data.dropna(subset=[metric]).copy()
            if plot_data.empty:
                print(f"Недостатньо даних для часової динаміки {metric}.")
                continue

            plt.figure(figsize=(11, 6))
            sns.lineplot(data=plot_data, x="year", y=metric, hue="country_code",
marker="o", linewidth=1.8)
            plt.title(f"Trend of {metric} for top-10 systems by students")
            plt.xlabel("Рік")
            plt.ylabel(metric)
            if metric == "total_students":
                plt.yscale("log")
            plt.tight_layout()
            save_current_figure(f"trends_{metric}_top10_systems.png")
            plt.show()

        spaghetti_period_data = country_year_profiles.query("2016 <= year <= 2021
and country_code in @top_10_systems_by_students").copy()
        for metric in yoy_metrics:
```

```
        plot_data = spaghetti_period_data.dropna(subset=[metric]).copy()
        if plot_data.empty:
            print(f"Недостатньо даних для spaghetti-графіка {metric} за 2016-2021.")
            continue

        plt.figure(figsize=(11, 6))
        sns.lineplot(data=plot_data, x="year", y=metric, hue="country_code", marker="o", linewidth=1.6)
        plt.title(f"Spaghetti plot of {metric}, 2016-2021")
        plt.xlabel("Рік")
        plt.ylabel(metric)
        if metric == "total_students":
            plt.yscale("log")
        plt.tight_layout()
        save_current_figure(f"spaghetti_{metric}_2016_2021.png")
        plt.show()

    print(
        "Інтерпретація: YoY-таблиця фіксує різкі зміни всередині країни. Такі стрибки потрібно перевіряти "
        "перед порівнянням latest year, бо вони можуть бути наслідком неповноти даних, реформ або змін звітності."
    )

    display(key_metric_deltas_by_country)
    yoy_anomalies_country_metrics

    # Cell 45 (notebook cell 90)
    missing_values_latest_year = (
```

```
        latest_snapshot[available_selected_metrics]
        .isna()
        .mean()
        .sort_values(ascending=False)
        .rename("missing_share")
        .reset_index()
        .rename(columns={"index": "metric"})
    )

    missing_values_path = TABLES_DIR / "missing_values_latest_year.csv"
    missing_values_latest_year.to_csv(missing_values_path, index=False)

    missing_values_latest_year

    # Cell 46 (notebook cell 93)
    institution_latest_snapshot = institution_year_metrics.query("year == @latest_year").copy()

    size_category_counts = (
        institution_latest_snapshot["size_category"]
        .value_counts(dropna=False)
        .rename_axis("size_category")
        .reset_index(name="institutions_count")
    )
    size_category_counts.to_csv(TABLES_DIR / "institution_size_category_counts_latest_year.csv", index=False)

    plt.figure(figsize=(9, 5))
    sns.barplot(data=size_category_counts, x="size_category", y="institutions_count", color="#8A6F3D")
```

```
plt.title(f"Розподіл ЗВО за size_category, {latest_year}")
plt.xlabel("Категорія розміру")
plt.ylabel("Кількість ЗВО")
plt.xticks(rotation=30, ha="right")
plt.tight_layout()
save_current_figure("institution_size_category_counts_latest_year.png")
plt.show()

size_category_counts


#NOTEBOOK: 03_hypothesis_2_internationalization_analysis.ipynb
#CODE CELLS: 23


# Cell 1 (notebook cell 3)
from pathlib import Path

import numpy as np
import pandas as pd
import matplotlib

try:
    get_ipython
    IN_INTERACTIVE_NOTEBOOK = get_ipython() is not None
except NameError:
    IN_INTERACTIVE_NOTEBOOK = False

# Safe backend for script/headless execution. Interactive notebooks keep their own backend.
```

```
if not IN_INTERACTIVE_NOTEBOOK:
    matplotlib.use("Agg")

import matplotlib.pyplot as plt
import seaborn as sns

try:
    from scipy.stats import pearsonr, spearmanr
    SCIPY_AVAILABLE = True
except ImportError:
    SCIPY_AVAILABLE = False

try:
    import statsmodels.api as sm
    STATSMODELS_AVAILABLE = True
except ImportError:
    STATSMODELS_AVAILABLE = False

sns.set_theme(style="whitegrid", context="notebook")
plt.rcParams["figure.figsize"] = (10, 6)
plt.rcParams["font.size"] = 11

# Cell 2 (notebook cell 5)
PROJECT_ROOT = Path.cwd().parent if Path.cwd().name == "notebooks" else Path.cwd()
DATA_DIR = PROJECT_ROOT / "data"
PROCESSED_DIR = DATA_DIR / "processed"
FIGURES_DIR = PROJECT_ROOT / "figures" / "hypotheses"
TABLES_DIR = PROJECT_ROOT / "tables" / "hypotheses"
```

```
COUNTRY_YEAR_PROFILES_PATH = PROCESSED_DIR / "country_year_profiles.csv"
STUDENT_STRUCTURE_METRICS_PATH = PROCESSED_DIR / "student_structure_metrics.csv"
FINANCE_STAFF_METRICS_PATH = PROCESSED_DIR / "finance_staff_metrics.csv"

FIGURES_DIR.mkdir(parents=True, exist_ok=True)
TABLES_DIR.mkdir(parents=True, exist_ok=True)

COUNTRY_YEAR_PROFILES_PATH,
STUDENT_STRUCTURE_METRICS_PATH,
FINANCE_STAFF_METRICS_PATH

# Cell 3 (notebook cell 7)
country_year_profiles = pd.read_csv(COUNTRY_YEAR_PROFILES_PATH)
student_structure_metrics = pd.read_csv(STUDENT_STRUCTURE_METRICS_PATH)
finance_staff_metrics = pd.read_csv(FINANCE_STAFF_METRICS_PATH)

print("country_year_profiles:", country_year_profiles.shape)
print("student_structure_metrics:", student_structure_metrics.shape)
print("finance_staff_metrics:", finance_staff_metrics.shape)

# Cell 4 (notebook cell 9)
required_columns = [
    "country_code",
    "year",
    "institutions_count",
```

```
            "total_students",
            "foreign_students_total",
            "foreign_students_share",
            "expenditure_per_student_ppp",
            "mean_institution_size",
            "median_institution_size",
            "isced7_share",
            "isced8_share",
            "student_academic_staff_ratio",
        ]

        missing_required_columns = [
            column for column in required_columns if column not in country_year_profiles.columns
        ]

        if missing_required_columns:
            raise ValueError(f"Missing required columns: {missing_required_columns}")

        country_year_duplicates = country_year_profiles.duplicated(
            subset=["country_code", "year"]
        ).sum()

        print("Duplicated country-year keys:", country_year_duplicates)

        if country_year_duplicates != 0:
            raise ValueError("country_year_profiles must have unique country_code + year keys")
```

```
country_year_profiles[required_columns].head()

# Cell 5 (notebook cell 10)
share_columns = ["foreign_students_share", "isced7_share", "isced8_share"]
share_quality = []

for column in share_columns:
    non_missing = country_year_profiles[column].dropna()
    share_quality.append(
        {
            "metric": column,
            "non_missing_count": non_missing.shape[0],
            "below_0_count": (non_missing < 0).sum(),
            "above_1_count": (non_missing > 1).sum(),
            "min": non_missing.min(),
            "max": non_missing.max(),
        }
    )

share_quality = pd.DataFrame(share_quality)
share_quality

# Cell 6 (notebook cell 13)
latest_year = country_year_profiles["year"].max()
latest_snapshot = country_year_profiles.query("year == @latest_year").copy()

print("Latest year:", latest_year)
print("Countries in latest snapshot:",
latest_snapshot["country_code"].nunique())
```

```
latest_snapshot[[
    "country_code",
    "year",
    "foreign_students_share",
    "foreign_students_total",
    "expenditure_per_student_ppp",
    "isced7_share",
    "isced8_share",
]].head()

# Cell 7 (notebook cell 15)
analysis_columns = [
    "foreign_students_share",
    "foreign_students_total",
    "expenditure_per_student_ppp",
    "isced7_share",
    "isced8_share",
    "mean_institution_size",
    "median_institution_size",
    "student_academic_staff_ratio",
    "institutions_count",
    "total_students",
]

missing_summary = (
    latest_snapshot[analysis_columns]
    .isna()
    .sum()
    .rename("missing_count")
    .to_frame()
```

```
    )
    missing_summary["missing_rate"] = missing_summary["missing_count"] / len(latest_snapshot)
    missing_summary = missing_summary.sort_values("missing_rate", ascending=False)
    missing_summary

    # Cell 8 (notebook cell 18)
    ranking_metrics = [
        "foreign_students_share",
        "foreign_students_total",
        "expenditure_per_student_ppp",
        "isced7_share",
        "isced8_share",
    ]

    ranking_tables = []

    for metric in ranking_metrics:
        ranking = (
            latest_snapshot[["country_code", "year", metric]]
            .dropna(subset=[metric])
            .sort_values(metric, ascending=False)
            .assign(metric=metric, rank=lambda df: np.arange(1, len(df) + 1))
            .rename(columns={metric: "value"})
        )
        ranking_tables.append(ranking[["metric", "rank", "country_code", "year", "value"]])

    hypothesis_2_rankings = pd.concat(ranking_tables, ignore_index=True)
```

```
hypothesis_2_rankings.to_csv(
    TABLES_DIR / "hypothesis_2_internationalization_rankings.csv",
    index=False,
)

hypothesis_2_rankings.head(15)

# Cell 9 (notebook cell 19)
top_foreign_share = hypothesis_2_rankings.query(
    "metric == 'foreign_students_share'"
).head(10)

top_foreign_share

# Cell 10 (notebook cell 22)
def plot_foreign_share_relationship(
    data,
    x_column,
    y_column,
    output_name,
    xlabel,
    ylabel,
    title_suffix="",
):
    plot_data = data[["country_code", x_column, y_column]].dropna().copy()

    fig, ax = plt.subplots(figsize=(10, 6))
    sns.scatterplot(
        data=plot_data,
        x=x_column,
```

```
        y=y_column,
        s=70,
        color="#2F6B4F",
        edgecolor="white",
        linewidth=0.7,
        ax=ax,
    )

    if plot_data.shape[0] >= 5:
        if STATSMODELS_AVAILABLE:
            sns.regplot(
                data=plot_data,
                x=x_column,
                y=y_column,
                scatter=False,
                lowess=True,
                color="#B84A2B",
                line_kws={"linewidth": 2, "label": "Лінія LOWESS"},
                ax=ax,
            )
        else:
            sns.regplot(
                data=plot_data,
                x=x_column,
                y=y_column,
                scatter=False,
                lowess=False,
                ci=None,
                color="#B84A2B",
```

```
                line_kws={"linewidth": 2, "linestyle": "--", "label": "Лінійний
тренд"},
                ax=ax,
            )

        for _, row in plot_data.iterrows():
            ax.annotate(
                row["country_code"],
                (row[x_column], row[y_column]),
                textcoords="offset points",
                xytext=(4, 4),
                fontsize=8,
            )

        trend_label = "LOWESS" if STATSMODELS_AVAILABLE else
"лінійний тренд"
        title_suffix_text = f", {title_suffix}" if title_suffix else ""
        ax.set_title(
            f"Частка іноземних студентів та {xlabel}, {latest_year} рік"
            f"{title_suffix_text} ({trend_label})"
        )
        ax.set_xlabel(xlabel)
        ax.set_ylabel(ylabel)
        ax.grid(True, alpha=0.3)

        handles, labels = ax.get_legend_handles_labels()
        if handles:
            ax.legend(loc="best")

        output_path = FIGURES_DIR / output_name
```

```
    fig.savefig(output_path, dpi=300, bbox_inches="tight")

    if IN_INTERACTIVE_NOTEBOOK:
        plt.show()
    else:
        plt.close(fig)

    return output_path

# Cell 11 (notebook cell 23)
plot_foreign_share_relationship(
    latest_snapshot,
    x_column="isced7_share",
    y_column="foreign_students_share",
    output_name="hypothesis_2_foreign_share_vs_isced7.png",
    xlabel="частка студентів рівня ISCED 7",
    ylabel="Частка іноземних студентів",
)

# Cell 12 (notebook cell 25)
plot_foreign_share_relationship(
    latest_snapshot,
    x_column="isced8_share",
    y_column="foreign_students_share",
    output_name="hypothesis_2_foreign_share_vs_isced8.png",
    xlabel="частка студентів рівня ISCED 8",
    ylabel="Частка іноземних студентів",
)

# Cell 13 (notebook cell 27)
```

```
plot_foreign_share_relationship(
    latest_snapshot,
    x_column="expenditure_per_student_ppp",
    y_column="foreign_students_share",
    output_name="hypothesis_2_foreign_share_vs_expenditure.png",
    xlabel="витрати на одного студента, PPP",
    ylabel="Частка іноземних студентів",
)

# Cell 14 (notebook cell 30)
def calculate_correlations(data, target_column, feature_columns):
    results = []

    for feature_column in feature_columns:
        pair_data = data[[target_column, feature_column]].dropna()
        n_observations = len(pair_data)

        result = {
            "target": target_column,
            "feature": feature_column,
            "n_observations": n_observations,
            "pearson_r": np.nan,
            "pearson_p_value": np.nan,
            "spearman_r": np.nan,
            "spearman_p_value": np.nan,
        }

        if n_observations >= 3 and pair_data[target_column].nunique() > 1 and
pair_data[feature_column].nunique() > 1:
            if SCIPY_AVAILABLE:
```

```
                    pearson_result = pearsonr(pair_data[target_column],
pair_data[feature_column])
                    spearman_result = spearmanr(pair_data[target_column],
pair_data[feature_column])
                    result["pearson_r"] = pearson_result.statistic
                    result["pearson_p_value"] = pearson_result.pvalue
                    result["spearman_r"] = spearman_result.statistic
                    result["spearman_p_value"] = spearman_result.pvalue
                else:
                    result["pearson_r"] =
pair_data[target_column].corr(pair_data[feature_column], method="pearson")
                    result["spearman_r"] =
pair_data[target_column].corr(pair_data[feature_column], method="spearman")

            results.append(result)

        return pd.DataFrame(results)

    correlation_features = [
        "expenditure_per_student_ppp",
        "isced7_share",
        "isced8_share",
        "mean_institution_size",
        "median_institution_size",
        "student_academic_staff_ratio",
        "total_students",
        "institutions_count",
    ]

    hypothesis_2_correlations = calculate_correlations(
```

```
    latest_snapshot,
    target_column="foreign_students_share",
    feature_columns=correlation_features,
)

hypothesis_2_correlations.to_csv(
    TABLES_DIR / "hypothesis_2_correlations.csv",
    index=False,
)

hypothesis_2_correlations.sort_values("spearman_r", ascending=False)

# Cell 15 (notebook cell 33)
group_data = latest_snapshot.dropna(subset=["foreign_students_share"]).copy()

group_code = pd.qcut(
    group_data["foreign_students_share"],
    q=3,
    labels=False,
    duplicates="drop",
)

available_groups = int(group_code.max() + 1) if group_code.notna().any() else 0

if available_groups == 3:
    group_label_map = {
        0: "low_internationalization",
        1: "medium_internationalization",
```

```
            2: "high_internationalization",
        }
    elif available_groups == 2:
        group_label_map = {
            0: "lower_internationalization",
            1: "higher_internationalization",
        }
    else:
        group_label_map = {0: "single_group"}

    group_data["internationalization_group"] =
group_code.map(group_label_map)

    group_comparison_metrics = [
        "foreign_students_share",
        "foreign_students_total",
        "expenditure_per_student_ppp",
        "isced7_share",
        "isced8_share",
        "mean_institution_size",
        "median_institution_size",
        "student_academic_staff_ratio",
        "total_students",
        "institutions_count",
    ]

    hypothesis_2_group_comparison = (
        group_data
        .groupby("internationalization_group",
observed=True)[group_comparison_metrics]
```

```
        .agg(["count", "mean", "median"])
    )

    hypothesis_2_group_comparison.columns = [
        f"{metric}_{statistic}" for metric, statistic in
hypothesis_2_group_comparison.columns
    ]

    hypothesis_2_group_comparison =
hypothesis_2_group_comparison.reset_index()

    hypothesis_2_group_comparison.to_csv(
        TABLES_DIR / "hypothesis_2_group_comparison.csv",
        index=False,
    )

    hypothesis_2_group_comparison

    # Cell 16 (notebook cell 35)
    median_comparison_metrics = {
        "expenditure_per_student_ppp_median": "Витрати на студента (PPP)",
        "isced7_share_median": "Частка ISCED 7",
        "isced8_share_median": "Частка ISCED 8",
        "median_institution_size_median": "Медіанний розмір ЗВО",
        "student_academic_staff_ratio_median": "Студенти / академічний
персонал",
    }

    group_order = [
        group_name
```

```
        for group_name in [
            "low_internationalization",
            "medium_internationalization",
            "high_internationalization",
            "lower_internationalization",
            "higher_internationalization",
            "single_group",
        ]
        if group_name in hypothesis_2_group_comparison["internationalization_group"].values
    ]

    group_label_ua = {
        "low_internationalization": "Низький рівень",
        "medium_internationalization": "Середній рівень",
        "high_internationalization": "Високий рівень",
        "lower_internationalization": "Нижчий рівень",
        "higher_internationalization": "Вищий рівень",
        "single_group": "Єдина група",
    }

    group_medians_plot = hypothesis_2_group_comparison[
        ["internationalization_group", *median_comparison_metrics]
    ].copy()

    for metric in median_comparison_metrics:
        metric_max = group_medians_plot[metric].max(skipna=True)
        group_medians_plot[metric] = np.where(
            metric_max > 0,
            group_medians_plot[metric] / metric_max,
```

```
            np.nan,
        )

    group_medians_plot = group_medians_plot.melt(
        id_vars="internationalization_group",
        var_name="metric",
        value_name="normalized_median",
    )
    group_medians_plot["metric"] = group_medians_plot["metric"].map(median_comparison_metrics)
    group_medians_plot["internationalization_group"] = group_medians_plot[
        "internationalization_group"
    ].map(group_label_ua)

    plot_group_order = [group_label_ua[group_name] for group_name in group_order]

    fig, ax = plt.subplots(figsize=(12, 7))
    sns.barplot(
        data=group_medians_plot,
        x="normalized_median",
        y="metric",
        hue="internationalization_group",
        hue_order=plot_group_order,
        palette="viridis",
        ax=ax,
    )

    ax.set_title(f"Порівняння медіанних показників груп країн, {latest_year}")
    ax.set_xlabel("Нормалізована медіана (максимум метрики = 1)")
```

```
        ax.set_ylabel("")
        ax.set_xlim(0, 1.08)
        ax.legend(title="Рівень інтернаціоналізації", bbox_to_anchor=(1.02, 1),
loc="upper left")
        fig.tight_layout()

        group_median_figure_path = FIGURES_DIR /
"hypothesis_2_group_median_comparison.png"
        fig.savefig(group_median_figure_path, dpi=300, bbox_inches="tight")
        plt.show()

        group_median_figure_path

        # Cell 17 (notebook cell 38)
        robustness_snapshot = latest_snapshot.query(
            "institutions_count >= 5 and total_students >= 10000"
        ).copy()

        print("Countries in main snapshot:",
latest_snapshot["country_code"].nunique())
        print("Countries after robustness filter:",
robustness_snapshot["country_code"].nunique())

        robustness_correlations = calculate_correlations(
            robustness_snapshot,
            target_column="foreign_students_share",
            feature_columns=[
                "expenditure_per_student_ppp",
                "isced7_share",
                "isced8_share",
```

```
        "mean_institution_size",
        "median_institution_size",
    ],
)

robustness_correlations

# Cell 18 (notebook cell 39)
correlation_comparison = hypothesis_2_correlations.merge(
    robustness_correlations,
    on=["target", "feature"],
    how="left",
    suffixes=("_main", "_robustness"),
)

correlation_comparison[[
    "feature",
    "n_observations_main",
    "spearman_r_main",
    "spearman_p_value_main",
    "n_observations_robustness",
    "spearman_r_robustness",
    "spearman_p_value_robustness",
]]

# Cell 19 (notebook cell 42)
minimum_institutions = 10

secondary_check_snapshot = latest_snapshot.query(
    "institutions_count >= @minimum_institutions"
```

```
    ).copy()

    excluded_small_systems = (
        latest_snapshot.query("institutions_count < @minimum_institutions")
        [["country_code", "year", "institutions_count", "total_students",
"foreign_students_share"]]
        .sort_values(["institutions_count", "country_code"])
        .reset_index(drop=True)
    )

    print("Countries in main snapshot:",
latest_snapshot["country_code"].nunique())
    print("Countries with at least 10 institutions:",
secondary_check_snapshot["country_code"].nunique())
    print("Excluded countries:",
excluded_small_systems["country_code"].nunique())

    excluded_small_systems

    # Cell 20 (notebook cell 45)
    plot_foreign_share_relationship(
        secondary_check_snapshot,
        x_column="isced7_share",
        y_column="foreign_students_share",

output_name="hypothesis_2_secondary_min_10_foreign_share_vs_isced7.png",
        xlabel="частка студентів рівня ISCED 7",
        ylabel="Частка іноземних студентів",
        title_suffix="кількість ЗВО ≥ 10",
    )
```

```
        # Cell 21 (notebook cell 47)
        plot_foreign_share_relationship(
            secondary_check_snapshot,
            x_column="isced8_share",
            y_column="foreign_students_share",

output_name="hypothesis_2_secondary_min_10_foreign_share_vs_isced8.png",
            xlabel="частка студентів рівня ISCED 8",
            ylabel="Частка іноземних студентів",
            title_suffix="кількість ЗВО ≥ 10",
        )

        # Cell 22 (notebook cell 49)
        plot_foreign_share_relationship(
            secondary_check_snapshot,
            x_column="expenditure_per_student_ppp",
            y_column="foreign_students_share",

output_name="hypothesis_2_secondary_min_10_foreign_share_vs_expenditure.png"
,
            xlabel="витрати на одного студента, PPP",
            ylabel="Частка іноземних студентів",
            title_suffix="кількість ЗВО ≥ 10",
        )

        # Cell 23 (notebook cell 52)
        secondary_correlation_features = [
            "expenditure_per_student_ppp",
            "isced7_share",
```

```
        "isced8_share",
        "mean_institution_size",
        "median_institution_size",
        "student_academic_staff_ratio",
    ]

    secondary_correlations = calculate_correlations(
        secondary_check_snapshot,
        target_column="foreign_students_share",
        feature_columns=secondary_correlation_features,
    )

    secondary_correlation_comparison = (
        hypothesis_2_correlations
        .query("feature in @secondary_correlation_features")
        .merge(
            secondary_correlations,
            on=["target", "feature"],
            how="left",
            suffixes=("_main", "_secondary_min_10"),
        )
    )

    secondary_correlation_comparison.to_csv(
        TABLES_DIR / "hypothesis_2_secondary_correlations_min_10_institutions.csv",
        index=False,
    )

    secondary_correlation_comparison[[
```

```
    "feature",
    "n_observations_main",
    "spearman_r_main",
    "spearman_p_value_main",
    "n_observations_secondary_min_10",
    "spearman_r_secondary_min_10",
    "spearman_p_value_secondary_min_10",
]]
```

```
    ============================================================
============================
    NOTEBOOK: 04_hypothesis_3_student_concentration_analysis.ipynb
    CODE CELLS: 13
    ============================================================
============================
```

```
    # Cell 1 (notebook cell 3)
    from pathlib import Path

    import numpy as np
    import pandas as pd
    import matplotlib.pyplot as plt
    import seaborn as sns
    from scipy.stats import spearmanr

    sns.set_theme(style="whitegrid", context="notebook")
    pd.set_option("display.max_columns", 100)
    pd.set_option("display.float_format", lambda value: f"{value:,.4f}")
```

```
# Cell 2 (notebook cell 5)
PROJECT_ROOT = Path.cwd().parent if Path.cwd().name == "notebooks" else Path.cwd()
DATA_DIR = PROJECT_ROOT / "data"
PROCESSED_DIR = DATA_DIR / "processed"
FIGURES_DIR = PROJECT_ROOT / "figures"
TABLES_DIR = PROJECT_ROOT / "tables"
HYPOTHESES_FIGURES_DIR = FIGURES_DIR / "hypotheses"
HYPOTHESES_TABLES_DIR = TABLES_DIR / "hypotheses"

PROCESSED_DIR.mkdir(parents=True, exist_ok=True)
HYPOTHESES_FIGURES_DIR.mkdir(parents=True, exist_ok=True)
HYPOTHESES_TABLES_DIR.mkdir(parents=True, exist_ok=True)

COUNTRY_YEAR_PROFILES_PATH = PROCESSED_DIR / "country_year_profiles.csv"
INSTITUTION_YEAR_METRICS_PATH = PROCESSED_DIR / "institution_year_metrics.csv"
CONCENTRATION_METRICS_PATH = PROCESSED_DIR / "concentration_metrics.csv"

RANKINGS_OUTPUT_PATH = HYPOTHESES_TABLES_DIR / "hypothesis_3_concentration_rankings.csv"
CORRELATIONS_OUTPUT_PATH = HYPOTHESES_TABLES_DIR / "hypothesis_3_correlations.csv"
DATA_COVERAGE_OUTPUT_PATH = HYPOTHESES_TABLES_DIR / "hypothesis_3_data_coverage.csv"

LORENZ_FIGURE_PATH = HYPOTHESES_FIGURES_DIR / "hypothesis_3_lorenz_curves.png"
```

```
GINI_SCATTER_FIGURE_PATH = HYPOTHESES_FIGURES_DIR / "hypothesis_3_gini_vs_mean_median_ratio.png"
TOP5_SCATTER_FIGURE_PATH = HYPOTHESES_FIGURES_DIR / "hypothesis_3_top5_vs_mean_median_ratio.png"
TOP10_SCATTER_FIGURE_PATH = HYPOTHESES_FIGURES_DIR / "hypothesis_3_top10_vs_mean_median_ratio.png"
BOXPLOT_FIGURE_PATH = HYPOTHESES_FIGURES_DIR / "hypothesis_3_boxplot_selected_countries.png"

# Cell 3 (notebook cell 7)
country_year_profiles = pd.read_csv(COUNTRY_YEAR_PROFILES_PATH)
institution_year_metrics = pd.read_csv(INSTITUTION_YEAR_METRICS_PATH)
concentration_metrics = pd.read_csv(CONCENTRATION_METRICS_PATH)

print("country_year_profiles:", country_year_profiles.shape)
print("institution_year_metrics:", institution_year_metrics.shape)
print("concentration_metrics:", concentration_metrics.shape)

# Cell 4 (notebook cell 9)
required_country_columns = [
    "country_code",
    "year",
    "institutions_count",
    "total_students",
    "mean_institution_size",
    "median_institution_size",
    "max_institution_size",
```

```
        "top_1_share",
        "top_5_share",
        "top_10_share",
        "gini_students",
    ]

    required_institution_columns = [
        "country_code",
        "year",
        "institution_id",
        "institution_name_en",
        "students_total",
    ]

    required_concentration_columns = [
        "country_code",
        "year",
        "top_1_share",
        "top_5_share",
        "top_10_share",
        "gini_students",
    ]

    missing_country_columns = sorted(set(required_country_columns) - set(country_year_profiles.columns))
    missing_institution_columns = sorted(set(required_institution_columns) - set(institution_year_metrics.columns))
    missing_concentration_columns = sorted(set(required_concentration_columns) - set(concentration_metrics.columns))
```

```
    if missing_country_columns:
        raise ValueError(f"Missing columns in country_year_profiles: {missing_country_columns}")
    if missing_institution_columns:
        raise ValueError(f"Missing columns in institution_year_metrics: {missing_institution_columns}")
    if missing_concentration_columns:
        raise ValueError(f"Missing columns in concentration_metrics: {missing_concentration_columns}")

    country_year_duplicates = country_year_profiles.duplicated(subset=["country_code", "year"]).sum()
    concentration_duplicates = concentration_metrics.duplicated(subset=["country_code", "year"]).sum()

    quality_checks = pd.DataFrame(
        {
            "check": [
                "country_year_profiles duplicate country_code-year keys",
                "concentration_metrics duplicate country_code-year keys",
                "country_year_profiles rows",
                "concentration_metrics rows",
            ],
            "value": [
                country_year_duplicates,
                concentration_duplicates,
                len(country_year_profiles),
                len(concentration_metrics),
            ],
        }
```

```
)
quality_checks

# Cell 5 (notebook cell 12)
latest_year = int(country_year_profiles["year"].max())
latest_snapshot = country_year_profiles.query("year == @latest_year").copy()
institution_latest = institution_year_metrics.query("year == @latest_year").copy()

print("Latest year:", latest_year)
print("Countries in latest snapshot:", latest_snapshot["country_code"].nunique())
print("Institution-year rows in latest year:", len(institution_latest))

# Cell 6 (notebook cell 14)
institution_student_coverage = (
    institution_latest
    .assign(has_student_data=lambda df: df["students_total"].notna() & (df["students_total"] >= 0))
    .groupby("country_code", as_index=False)
    .agg(
        institution_rows=("institution_id", "count"),
        institutions_with_student_data=("has_student_data", "sum"),
    )
)

analysis_snapshot = latest_snapshot.merge(institution_student_coverage, on="country_code", how="left")
analysis_snapshot["institution_rows"] = analysis_snapshot["institution_rows"].fillna(0).astype(int)
```

```
        analysis_snapshot["institutions_with_student_data"] =
analysis_snapshot["institutions_with_student_data"].fillna(0).astype(int)
        analysis_snapshot["student_data_coverage"] = np.where(
            analysis_snapshot["institutions_count"] > 0,
            analysis_snapshot["institutions_with_student_data"] /
analysis_snapshot["institutions_count"],
            np.nan,
        )
        analysis_snapshot["mean_median_gap"] = (
            analysis_snapshot["mean_institution_size"] -
analysis_snapshot["median_institution_size"]
        )
        analysis_snapshot["mean_median_ratio"] = np.where(
            analysis_snapshot["median_institution_size"] > 0,
            analysis_snapshot["mean_institution_size"] /
analysis_snapshot["median_institution_size"],
            np.nan,
        )

        analysis_columns = [
            "country_code",
            "year",
            "institutions_count",
            "institution_rows",
            "institutions_with_student_data",
            "student_data_coverage",
            "total_students",
            "mean_institution_size",
            "median_institution_size",
            "max_institution_size",
```

```
    "mean_median_gap",
    "mean_median_ratio",
    "top_1_share",
    "top_5_share",
    "top_10_share",
    "gini_students",
]

analysis_snapshot[analysis_columns].head()

# Cell 7 (notebook cell 16)
missing_metrics = [
    "gini_students",
    "top_1_share",
    "top_5_share",
    "top_10_share",
    "mean_median_gap",
    "mean_median_ratio",
    "student_data_coverage",
]

missing_summary = (
    analysis_snapshot[missing_metrics]
    .isna()
    .sum()
    .rename("missing_count")
    .reset_index()
    .rename(columns={"index": "metric"})
)
```

```
    missing_summary["missing_rate"] = missing_summary["missing_count"] /
len(analysis_snapshot)

    hypothesis_3_data_coverage = analysis_snapshot[
      [
        "country_code",
        "year",
        "institutions_count",
        "institution_rows",
        "institutions_with_student_data",
        "student_data_coverage",
        "gini_students",
        "top_5_share",
        "top_10_share",
      ]
    ].sort_values(["student_data_coverage", "gini_students"], ascending=[True,
False])

    hypothesis_3_data_coverage.to_csv(DATA_COVERAGE_OUTPUT_PATH,
index=False)

    missing_summary

    # Cell 8 (notebook cell 19)
    ranking_columns = [
      "country_code",
      "year",
      "institutions_count",
      "total_students",
      "mean_institution_size",
```

```
    "median_institution_size",
    "max_institution_size",
    "mean_median_gap",
    "mean_median_ratio",
    "student_data_coverage",
    "top_1_share",
    "top_5_share",
    "top_10_share",
    "gini_students",
]

rank_by_gini = (
    analysis_snapshot[ranking_columns]
    .sort_values("gini_students", ascending=False)
    .assign(ranking_metric="gini_students")
    .head(15)
)
rank_by_top5 = (
    analysis_snapshot[ranking_columns]
    .sort_values("top_5_share", ascending=False)
    .assign(ranking_metric="top_5_share")
    .head(15)
)
rank_by_top10 = (
    analysis_snapshot[ranking_columns]
    .sort_values("top_10_share", ascending=False)
    .assign(ranking_metric="top_10_share")
    .head(15)
)
rank_by_mean_median_ratio = (
```

```
        analysis_snapshot[ranking_columns]
        .sort_values("mean_median_ratio", ascending=False)
        .assign(ranking_metric="mean_median_ratio")
        .head(15)
    )

    hypothesis_3_concentration_rankings = pd.concat(
        [rank_by_gini, rank_by_top5, rank_by_top10, rank_by_mean_median_ratio],
        ignore_index=True,
    )

    hypothesis_3_concentration_rankings.to_csv(RANKINGS_OUTPUT_PATH, index=False)

    rank_by_gini[
        ["country_code", "gini_students", "top_5_share", "top_10_share", "mean_median_ratio", "student_data_coverage"]
    ]

    # Cell 9 (notebook cell 22)
    def lorenz_curve(values):
        clean_values = pd.Series(values).dropna()
        clean_values = clean_values[clean_values >= 0].sort_values().to_numpy()

        if len(clean_values) == 0 or clean_values.sum() <= 0:
            return np.array([0, 1]), np.array([0, np.nan])

        cumulative_students = np.cumsum(clean_values)
```

```
            cumulative_students = np.insert(cumulative_students / cumulative_students[-1], 0, 0)
            cumulative_institutions = np.linspace(0, 1, len(cumulative_students))
            return cumulative_institutions, cumulative_students

        eligible_for_lorenz = analysis_snapshot.dropna(subset=["gini_students", "student_data_coverage"]).copy()
        eligible_for_lorenz = eligible_for_lorenz.query("student_data_coverage >= 0.8")

        if len(eligible_for_lorenz) >= 3:
            high_country = eligible_for_lorenz.sort_values("gini_students", ascending=False).iloc[0]["country_code"]
            low_country = eligible_for_lorenz.sort_values("gini_students", ascending=True).iloc[0]["country_code"]
            median_gini = eligible_for_lorenz["gini_students"].median()
            middle_country = (
                eligible_for_lorenz
                .assign(distance_to_median=lambda df: (df["gini_students"] - median_gini).abs())
                .sort_values("distance_to_median")
                .iloc[0]["country_code"]
            )
            selected_lorenz_countries = list(dict.fromkeys([high_country, middle_country, low_country]))
        else:
            selected_lorenz_countries = (
                analysis_snapshot.dropna(subset=["gini_students"])
                .sort_values("gini_students", ascending=False)
                .head(3)["country_code"]
```

```
            .tolist()
        )

    plt.figure(figsize=(8, 6))
    plt.plot([0, 1], [0, 1], linestyle="--", color="gray", linewidth=1.2, label="Equal distribution")

    for country_code in selected_lorenz_countries:
        country_values = institution_latest.loc[
            institution_latest["country_code"] == country_code,
            "students_total",
        ]
        x_values, y_values = lorenz_curve(country_values)
        gini_value = analysis_snapshot.loc[
            analysis_snapshot["country_code"] == country_code,
            "gini_students",
        ].iloc[0]
        plt.plot(x_values, y_values, linewidth=2, label=f"{country_code} (Gini={gini_value:.2f})")

    plt.title(f"Lorenz curves for student distribution, {latest_year}")
    plt.xlabel("Cumulative share of institutions")
    plt.ylabel("Cumulative share of students")
    plt.legend()
    plt.tight_layout()
    plt.savefig(LORENZ_FIGURE_PATH, dpi=300, bbox_inches="tight")
    plt.show()

    selected_lorenz_countries
```

```
# Cell 10 (notebook cell 25)
def plot_concentration_scatter(x_column, x_label, figure_path):
    plot_data = analysis_snapshot.dropna(subset=[x_column, "mean_median_ratio"]).copy()

    plt.figure(figsize=(8, 6))
    ax = sns.regplot(
        data=plot_data,
        x=x_column,
        y="mean_median_ratio",
        scatter_kws={"s": 70, "alpha": 0.8},
        line_kws={"color": "#b23a48", "linewidth": 2},
        ci=None,
    )

    for _, row in plot_data.iterrows():
        ax.text(
            row[x_column],
            row["mean_median_ratio"],
            row["country_code"],
            fontsize=8,
            ha="left",
            va="bottom",
            alpha=0.8,
        )

    plt.title(f"{x_label} vs mean/median institution size ratio, {latest_year}")
    plt.xlabel(x_label)
    plt.ylabel("Mean / median institution size")
    plt.tight_layout()
```

```
        plt.savefig(figure_path, dpi=300, bbox_inches="tight")
        plt.show()

    plot_concentration_scatter("gini_students", "Gini coefficient of student distribution", GINI_SCATTER_FIGURE_PATH)
    plot_concentration_scatter("top_5_share", "Share of students in top 5 institutions", TOP5_SCATTER_FIGURE_PATH)
    plot_concentration_scatter("top_10_share", "Share of students in top 10 institutions", TOP10_SCATTER_FIGURE_PATH)

    # Cell 11 (notebook cell 28)
    correlation_pairs = [
        ("gini_students", "mean_median_gap"),
        ("gini_students", "mean_median_ratio"),
        ("top_1_share", "mean_median_gap"),
        ("top_1_share", "mean_median_ratio"),
        ("top_5_share", "mean_median_gap"),
        ("top_5_share", "mean_median_ratio"),
        ("top_10_share", "mean_median_gap"),
        ("top_10_share", "mean_median_ratio"),
    ]

    correlation_results = []
    for x_column, y_column in correlation_pairs:
        pair_data = analysis_snapshot[[x_column, y_column]].dropna()
        if len(pair_data) >= 3:
            statistic, p_value = spearmanr(pair_data[x_column], pair_data[y_column])
        else:
            statistic, p_value = np.nan, np.nan
```

```
            correlation_results.append(
                {
                    "x_metric": x_column,
                    "y_metric": y_column,
                    "method": "spearman",
                    "n_countries": len(pair_data),
                    "correlation": statistic,
                    "p_value": p_value,
                }
            )

        hypothesis_3_correlations = pd.DataFrame(correlation_results)
        hypothesis_3_correlations.to_csv(CORRELATIONS_OUTPUT_PATH,
index=False)
        hypothesis_3_correlations

        # Cell 12 (notebook cell 31)
        selected_boxplot_countries = selected_lorenz_countries
        boxplot_data = institution_latest[
            institution_latest["country_code"].isin(selected_boxplot_countries)
            & institution_latest["students_total"].notna()
            & (institution_latest["students_total"] >= 0)
        ].copy()

        plt.figure(figsize=(9, 6))
        sns.boxplot(
            data=boxplot_data,
            x="country_code",
            y="students_total",
```

```
    order=selected_boxplot_countries,
    color="#8fb9aa",
)
plt.yscale("log")
plt.title(f"Distribution of institution size in selected countries, {latest_year}")
plt.xlabel("Country")
plt.ylabel("Students total, log scale")
plt.tight_layout()
plt.savefig(BOXPLOT_FIGURE_PATH, dpi=300, bbox_inches="tight")
plt.show()

boxplot_data.groupby("country_code")["students_total"].describe()

# Cell 13 (notebook cell 34)
expected_outputs = [
    RANKINGS_OUTPUT_PATH,
    CORRELATIONS_OUTPUT_PATH,
    DATA_COVERAGE_OUTPUT_PATH,
    LORENZ_FIGURE_PATH,
    GINI_SCATTER_FIGURE_PATH,
    TOP5_SCATTER_FIGURE_PATH,
    TOP10_SCATTER_FIGURE_PATH,
    BOXPLOT_FIGURE_PATH,
]

output_check = pd.DataFrame(
    {
        "path": [str(path.relative_to(PROJECT_ROOT)) for path in
expected_outputs],
        "exists": [path.exists() for path in expected_outputs],
```

```
    }
)
output_check



#NOTEBOOK: 05_factor_analysis_and_clustering.ipynb
#CODE CELLS: 36



# Cell 1 (notebook cell 3)
from pathlib import Path
import os

PROJECT_ROOT = Path.cwd().parent if Path.cwd().name == "notebooks" else Path.cwd()
os.environ.setdefault("MPLCONFIGDIR", str(PROJECT_ROOT / ".matplotlib_cache"))

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans, AgglomerativeClustering
from sklearn.metrics import silhouette_score

sns.set_theme(style="whitegrid", context="notebook")
```

```
pd.set_option("display.max_columns", 120)
pd.set_option("display.float_format", lambda value: f"{value:,.4f}")

# Cell 2 (notebook cell 5)
DATA_DIR = PROJECT_ROOT / "data"
PROCESSED_DIR = DATA_DIR / "processed"
FIGURES_DIR = PROJECT_ROOT / "figures"
TABLES_DIR = PROJECT_ROOT / "tables"
CLUSTERING_FIGURES_DIR = FIGURES_DIR / "clustering"
FINAL_TABLES_DIR = TABLES_DIR / "final"

PROCESSED_DIR.mkdir(parents=True, exist_ok=True)
CLUSTERING_FIGURES_DIR.mkdir(parents=True, exist_ok=True)
FINAL_TABLES_DIR.mkdir(parents=True, exist_ok=True)

COUNTRY_YEAR_PROFILES_PATH = PROCESSED_DIR / "country_year_profiles.csv"
CLUSTER_RESULTS_PATH = PROCESSED_DIR / "cluster_results.csv"
CLUSTER_PROFILES_PATH = FINAL_TABLES_DIR / "cluster_profiles.csv"
PCA_LOADINGS_PATH = FINAL_TABLES_DIR / "pca_loadings.csv"
PCA_SCORES_PATH = FINAL_TABLES_DIR / "pca_scores.csv"
CLUSTERING_MISSINGNESS_PATH = FINAL_TABLES_DIR / "clustering_missingness.csv"
EXCLUDED_COUNTRIES_PATH = FINAL_TABLES_DIR / "clustering_excluded_countries.csv"

COUNTRY_YEAR_PROFILES_PATH

# Cell 3 (notebook cell 7)
```

```
    if not COUNTRY_YEAR_PROFILES_PATH.exists():
        raise FileNotFoundError(f"Input file not found: {COUNTRY_YEAR_PROFILES_PATH}")

    country_year_profiles = pd.read_csv(COUNTRY_YEAR_PROFILES_PATH)

    print(f"Rows: {country_year_profiles.shape[0]:,}")
    print(f"Columns: {country_year_profiles.shape[1]:,}")
    print(f"Countries: {country_year_profiles['country_code'].nunique():,}")
    print(f"Years: {country_year_profiles['year'].min()} - {country_year_profiles['year'].max()}")

    country_year_profiles.head()

    # Cell 4 (notebook cell 9)
    required_columns = [
        "country_code",
        "year",
        "institutions_count",
        "total_students",
        "mean_institution_size",
        "median_institution_size",
        "foreign_students_share",
        "student_academic_staff_ratio",
        "expenditure_per_student_ppp",
        "isced7_share",
        "isced8_share",
        "top_5_share",
        "top_10_share",
```

```
    "gini_students",
]

missing_required_columns = sorted(set(required_columns) - set(country_year_profiles.columns))
country_year_duplicates = country_year_profiles.duplicated(subset=["country_code", "year"]).sum()

print(f"Missing required columns: {missing_required_columns}")
print(f"Duplicated country_code + year rows: {country_year_duplicates}")

if missing_required_columns:
    raise ValueError(f"Missing required columns: {missing_required_columns}")
if country_year_duplicates != 0:
    raise ValueError(f"country_code + year is not unique: {country_year_duplicates} duplicate rows")

# Cell 5 (notebook cell 11)
clustering_metrics = [
    "institutions_count",
    "total_students",
    "mean_institution_size",
    "median_institution_size",
    "foreign_students_share",
    "student_academic_staff_ratio",
    "expenditure_per_student_ppp",
    "isced7_share",
    "isced8_share",
    "top_5_share",
```

```
        "top_10_share",
        "gini_students",
    ]

    coverage_by_year = (
        country_year_profiles

.assign(complete_for_clustering=country_year_profiles[clustering_metrics].notna().al
l(axis=1))
        .groupby("year", as_index=False)
        .agg(
            countries_count=("country_code", "nunique"),
            complete_countries=("complete_for_clustering", "sum"),
        )
    )
    coverage_by_year["complete_share"] = np.where(
        coverage_by_year["countries_count"] > 0,
        coverage_by_year["complete_countries"]                                    /
coverage_by_year["countries_count"],
        np.nan,
    )

    latest_year = int(country_year_profiles["year"].max())
    analysis_year = latest_year
    latest_snapshot          =          country_year_profiles.query("year          ==
@analysis_year").copy()

    print(f"Selected year for clustering: {analysis_year}")
    print(f"Countries                 in                 selected                 year:
{latest_snapshot['country_code'].nunique()}")
```

```
coverage_by_year.sort_values("year")

# Cell 6 (notebook cell 13)
missingness = (
    latest_snapshot[clustering_metrics]
    .isna()
    .agg(["sum", "mean"])
    .T
    .rename(columns={"sum": "missing_count", "mean": "missing_rate"})
    .sort_values("missing_rate", ascending=False)
)
missingness.to_csv(CLUSTERING_MISSINGNESS_PATH)

country_missingness = latest_snapshot[["country_code", "year"] + clustering_metrics].copy()
country_missingness["missing_metrics_count"] = country_missingness[clustering_metrics].isna().sum(axis=1)
country_missingness["missing_metrics"] = country_missingness[clustering_metrics].isna().apply(
    lambda row: ", ".join(row.index[row].tolist()), axis=1
)

excluded_countries = country_missingness.query("missing_metrics_count > 0").copy()
excluded_countries.to_csv(EXCLUDED_COUNTRIES_PATH, index=False)

print(f"Saved missingness table: {CLUSTERING_MISSINGNESS_PATH}")
print(f"Countries excluded because of missing clustering metrics: {excluded_countries.shape[0]}")
```

```
    missingness

    # Cell 7 (notebook cell 15)
    complete_snapshot = latest_snapshot.dropna(subset=clustering_metrics).copy()

    if complete_snapshot.shape[0] < 4:
        raise ValueError(
            "Not enough complete countries for clustering. "
            "Consider using another year or reducing the feature set."
        )

    log_transform_metrics = [
        "institutions_count",
        "total_students",
        "mean_institution_size",
        "median_institution_size",
    ]

    modeling_data = complete_snapshot[["country_code", "year"] + clustering_metrics].copy()
    feature_matrix = modeling_data[clustering_metrics].copy()

    for metric in log_transform_metrics:
        if (feature_matrix[metric] < 0).any():
            raise ValueError(f"Metric {metric} contains negative values and cannot be log-transformed.")
        feature_matrix[f"log1p_{metric}"] = np.log1p(feature_matrix[metric])
```

```
modeling_features = [
    f"log1p_{metric}" if metric in log_transform_metrics else metric
    for metric in clustering_metrics
]
feature_matrix = feature_matrix[modeling_features]

scaler = StandardScaler()
features_scaled = scaler.fit_transform(feature_matrix)
features_scaled_df = pd.DataFrame(
    features_scaled,
    columns=modeling_features,
    index=modeling_data["country_code"],
)

print(f"Countries used in clustering: {modeling_data['country_code'].nunique()}")
print(f"Features used: {len(modeling_features)}")

features_scaled_df.head()

# Cell 8 (notebook cell 17)
n_components = min(len(modeling_features), features_scaled_df.shape[0] - 1)
pca = PCA(n_components=n_components, random_state=42)
pca_scores = pca.fit_transform(features_scaled_df)

pca_columns = [f"pca_{i + 1}" for i in range(n_components)]
pca_scores_df = pd.DataFrame(pca_scores, columns=pca_columns)
pca_scores_df.insert(0, "country_code", modeling_data["country_code"].values)
pca_scores_df.insert(1, "year", modeling_data["year"].values)
```

```
        pca_scores_df.to_csv(PCA_SCORES_PATH, index=False)

        explained_variance = pd.DataFrame(
            {
                "component": pca_columns,
                "explained_variance_ratio": pca.explained_variance_ratio_,
                "cumulative_explained_variance":
np.cumsum(pca.explained_variance_ratio_),
            }
        )

        pca_loadings = pd.DataFrame(
            pca.components_.T,
            index=modeling_features,
            columns=pca_columns,
        )
        pca_loadings.to_csv(PCA_LOADINGS_PATH)

        print(f"Saved PCA scores: {PCA_SCORES_PATH}")
        print(f"Saved PCA loadings: {PCA_LOADINGS_PATH}")

        explained_variance.head(10)

        # Cell 9 (notebook cell 18)
        plt.figure(figsize=(9, 5))
        sns.barplot(
            data=explained_variance,
            x="component",
            y="explained_variance_ratio",
            color="#4C72B0",
```

```
    )
    sns.lineplot(
        data=explained_variance,
        x="component",
        y="cumulative_explained_variance",
        marker="o",
        color="#C44E52",
    )
    plt.title(f"PCA explained variance, {analysis_year}")
    plt.xlabel("Principal component")
    plt.ylabel("Explained variance ratio")
    plt.xticks(rotation=45)
    plt.tight_layout()

    pca_variance_path = CLUSTERING_FIGURES_DIR / "pca_explained_variance.png"
    plt.savefig(pca_variance_path, dpi=300, bbox_inches="tight")
    plt.show()

    print(f"Saved figure: {pca_variance_path}")

    # Cell 10 (notebook cell 19)
    pca_plot_data = pca_scores_df.copy()

    fig, ax = plt.subplots(figsize=(12, 8))
    sns.scatterplot(data=pca_plot_data, x="pca_1", y="pca_2", s=90, color="#4C72B0", ax=ax)

    x_min, x_max = pca_plot_data["pca_1"].min(), pca_plot_data["pca_1"].max()
    y_min, y_max = pca_plot_data["pca_2"].min(), pca_plot_data["pca_2"].max()
```

```
x_padding = max((x_max - x_min) * 0.18, 0.5)
y_padding = max((y_max - y_min) * 0.18, 0.5)
ax.set_xlim(x_min - x_padding, x_max + x_padding)
ax.set_ylim(y_min - y_padding, y_max + y_padding)

for _, row in pca_plot_data.iterrows():
    ax.annotate(
        row["country_code"],
        (row["pca_1"], row["pca_2"]),
        textcoords="offset points",
        xytext=(5, 5),
        fontsize=9,
        ha="left",
        va="bottom",
        clip_on=False,
    )

ax.axhline(0, color="gray", linewidth=0.8, alpha=0.7)
ax.axvline(0, color="gray", linewidth=0.8, alpha=0.7)
ax.set_title(f"Розташування країн у просторі головних компонентів, {analysis_year} рік")
ax.set_xlabel("Перша головна компонента (PC1)")
ax.set_ylabel("Друга головна компонента (PC2)")
fig.tight_layout()

pca_country_map_path = CLUSTERING_FIGURES_DIR / "pca_country_map.png"
fig.savefig(pca_country_map_path, dpi=300, bbox_inches="tight")
plt.show()
```

```
print(f"Saved figure: {pca_country_map_path}")

# Cell 11 (notebook cell 20)
def describe_component_loadings(loadings, component, top_n=5):
    component_loadings = loadings[component].sort_values(key=lambda values: values.abs(), ascending=False).head(top_n)
    return component_loadings.reset_index().rename(columns={"index": "metric", component: "loading"})

pca_1_interpretation = describe_component_loadings(pca_loadings, "pca_1")
pca_2_interpretation = describe_component_loadings(pca_loadings, "pca_2")

print("PCA 1 найбільше пов'язана з такими метриками:")
display(pca_1_interpretation)
print("PCA 2 найбільше пов'язана з такими метриками:")
display(pca_2_interpretation)

# Cell 12 (notebook cell 22)
max_k = min(6, features_scaled_df.shape[0] - 1)
k_values = list(range(2, max_k + 1))

silhouette_records = []
for k in k_values:
    kmeans = KMeans(n_clusters=k, random_state=42, n_init=20)
    labels = kmeans.fit_predict(features_scaled_df)
    score = silhouette_score(features_scaled_df, labels)
    silhouette_records.append({"k": k, "silhouette_score": score})

silhouette_scores = pd.DataFrame(silhouette_records)
```

```
silhouette_scores.to_csv(FINAL_TABLES_DIR / "silhouette_scores.csv", index=False)

best_k = int(silhouette_scores.sort_values("silhouette_score", ascending=False).iloc[0]["k"])
print(f"Selected k by silhouette score: {best_k}")

silhouette_scores

# Cell 13 (notebook cell 23)
plt.figure(figsize=(8, 5))
sns.lineplot(data=silhouette_scores, x="k", y="silhouette_score", marker="o", color="#4C72B0")
plt.axvline(best_k, color="#C44E52", linestyle="--", linewidth=1.5, label=f"selected k = {best_k}")
plt.title(f"Silhouette scores for K-Means, {analysis_year}")
plt.xlabel("Number of clusters")
plt.ylabel("Silhouette score")
plt.legend()
plt.tight_layout()

silhouette_path = CLUSTERING_FIGURES_DIR / "silhouette_scores.png"
plt.savefig(silhouette_path, dpi=300, bbox_inches="tight")
plt.show()

print(f"Saved figure: {silhouette_path}")

# Cell 14 (notebook cell 25)
kmeans_model = KMeans(n_clusters=best_k, random_state=42, n_init=50)
kmeans_labels = kmeans_model.fit_predict(features_scaled_df)
```

```
cluster_results = modeling_data.copy()
cluster_results["kmeans_cluster"] = kmeans_labels
cluster_results = cluster_results.merge(pca_scores_df, on=["country_code", "year"], how="left")

cluster_results[["country_code", "year", "kmeans_cluster", "pca_1", "pca_2"]].head()

# Cell 15 (notebook cell 26)
plt.figure(figsize=(10, 7))
sns.scatterplot(
    data=cluster_results,
    x="pca_1",
    y="pca_2",
    hue="kmeans_cluster",
    palette="Set2",
    s=100,
)
for _, row in cluster_results.iterrows():
    plt.text(row["pca_1"], row["pca_2"], row["country_code"], fontsize=9, ha="left", va="bottom")
plt.axhline(0, color="gray", linewidth=0.8, alpha=0.7)
plt.axvline(0, color="gray", linewidth=0.8, alpha=0.7)
plt.title(f"K-Means clusters in PCA space, {analysis_year}")
plt.xlabel("PCA 1")
plt.ylabel("PCA 2")
plt.legend(title="Cluster")
plt.tight_layout()
```

```
kmeans_pca_path = CLUSTERING_FIGURES_DIR / "kmeans_clusters_pca.png"
plt.savefig(kmeans_pca_path, dpi=300, bbox_inches="tight")
plt.show()

print(f"Saved figure: {kmeans_pca_path}")

# Cell 16 (notebook cell 28)
hierarchical_model = AgglomerativeClustering(n_clusters=best_k, linkage="ward")
hierarchical_labels = hierarchical_model.fit_predict(features_scaled_df)

cluster_results["hierarchical_cluster"] = hierarchical_labels

cluster_comparison = pd.crosstab(
    cluster_results["kmeans_cluster"],
    cluster_results["hierarchical_cluster"],
    rownames=["kmeans_cluster"],
    colnames=["hierarchical_cluster"],
)
cluster_comparison.to_csv(FINAL_TABLES_DIR / "cluster_method_comparison.csv")

cluster_comparison

# Cell 17 (notebook cell 29)
plt.figure(figsize=(8, 6))
sns.heatmap(cluster_comparison, annot=True, fmt="d", cmap="Blues", cbar=False)
plt.title("Comparison of K-Means and hierarchical clusters")
```

```
        plt.xlabel("Hierarchical cluster")
        plt.ylabel("K-Means cluster")
        plt.tight_layout()

        cluster_comparison_path = CLUSTERING_FIGURES_DIR / "cluster_method_comparison.png"
        plt.savefig(cluster_comparison_path, dpi=300, bbox_inches="tight")
        plt.show()

        print(f"Saved figure: {cluster_comparison_path}")

        # Cell 18 (notebook cell 31)
        cluster_profiles = (
            cluster_results
            .groupby("kmeans_cluster")
            .agg(
                countries_count=("country_code", "nunique"),
                countries=("country_code", lambda values: ", ".join(sorted(values.astype(str)))),
                institutions_count_mean=("institutions_count", "mean"),
                total_students_mean=("total_students", "mean"),
                mean_institution_size_mean=("mean_institution_size", "mean"),
                median_institution_size_mean=("median_institution_size", "mean"),
                foreign_students_share_mean=("foreign_students_share", "mean"),
                student_academic_staff_ratio_mean=("student_academic_staff_ratio", "mean"),
                expenditure_per_student_ppp_mean=("expenditure_per_student_ppp", "mean"),
                isced7_share_mean=("isced7_share", "mean"),
                isced8_share_mean=("isced8_share", "mean"),
```

```
            top_5_share_mean=("top_5_share", "mean"),
            top_10_share_mean=("top_10_share", "mean"),
            gini_students_mean=("gini_students", "mean"),
        )
        .reset_index()
    )

    cluster_profiles.to_csv(CLUSTER_PROFILES_PATH, index=False)

    print(f"Saved cluster profiles: {CLUSTER_PROFILES_PATH}")
    cluster_profiles

    # Cell 19 (notebook cell 32)
    profile_metric_columns = [column for column in cluster_profiles.columns if
column.endswith("_mean")]
    profile_for_heatmap                                                    =
cluster_profiles.set_index("kmeans_cluster")[profile_metric_columns].copy()
    profile_z_scores = pd.DataFrame(
        StandardScaler().fit_transform(profile_for_heatmap),
        index=profile_for_heatmap.index,
        columns=profile_for_heatmap.columns,
    )

    plt.figure(figsize=(13, max(4, 1.2 * profile_z_scores.shape[0])))
    sns.heatmap(profile_z_scores, annot=True, fmt=".2f", cmap="vlag", center=0)
    plt.title("Standardized cluster profiles based on original metrics")
    plt.xlabel("Metric")
    plt.ylabel("K-Means cluster")
    plt.xticks(rotation=45, ha="right")
    plt.tight_layout()
```

```
cluster_profiles_heatmap_path = CLUSTERING_FIGURES_DIR / "cluster_profiles_heatmap.png"
plt.savefig(cluster_profiles_heatmap_path, dpi=300, bbox_inches="tight")
plt.show()

print(f"Saved figure: {cluster_profiles_heatmap_path}")

# Cell 20 (notebook cell 33)
def build_cluster_description(profile_z_scores, top_n=3):
    descriptions = []
    for cluster_id, row in profile_z_scores.iterrows():
        high_metrics = row.sort_values(ascending=False).head(top_n)
        low_metrics = row.sort_values(ascending=True).head(top_n)
        descriptions.append(
            {
                "kmeans_cluster": cluster_id,
                "high_relative_metrics": ", ".join(high_metrics.index.tolist()),
                "low_relative_metrics": ", ".join(low_metrics.index.tolist()),
            }
        )
    return pd.DataFrame(descriptions)

cluster_descriptions = build_cluster_description(profile_z_scores)
cluster_descriptions.to_csv(FINAL_TABLES_DIR / "cluster_descriptions.csv", index=False)
cluster_descriptions

# Cell 21 (notebook cell 35)
cluster_results.to_csv(CLUSTER_RESULTS_PATH, index=False)
```

```
print(f"Saved cluster results: {CLUSTER_RESULTS_PATH}")
print(f"Rows: {cluster_results.shape[0]}")
print(f"Columns: {cluster_results.shape[1]}")

cluster_results.head()

# Cell 22 (notebook cell 37)
cluster_results_duplicates = cluster_results.duplicated(subset=["country_code", "year"]).sum()
cluster_sizes = (
    cluster_results["kmeans_cluster"]
    .value_counts()
    .sort_index()
    .rename_axis("kmeans_cluster")
    .reset_index(name="countries_count")
)
small_clusters = cluster_sizes.query("countries_count <= 1")

print(f"Duplicated country_code + year in cluster_results: {cluster_results_duplicates}")
print(f"Countries included: {cluster_results['country_code'].nunique()}")
print(f"Countries excluded due to missing metrics: {excluded_countries.shape[0]}")
print(f"Clusters with one or zero countries: {small_clusters.shape[0]}")

if cluster_results_duplicates != 0:
    raise ValueError(f"cluster_results has duplicate country_code + year rows: {cluster_results_duplicates}")
```

```
cluster_sizes


# Cell 23 (notebook cell 39)
reduced_clustering_metrics = [
    metric
    for metric in clustering_metrics
    if metric not in ["student_academic_staff_ratio",
"expenditure_per_student_ppp"]
]


reduced_paths = {
    "cluster_results": PROCESSED_DIR / "cluster_results_reduced.csv",
    "cluster_profiles": FINAL_TABLES_DIR / "cluster_profiles_reduced.csv",
    "pca_loadings": FINAL_TABLES_DIR / "pca_loadings_reduced.csv",
    "pca_scores": FINAL_TABLES_DIR / "pca_scores_reduced.csv",
    "missingness": FINAL_TABLES_DIR /
"clustering_missingness_reduced.csv",
    "excluded_countries": FINAL_TABLES_DIR /
"clustering_excluded_countries_reduced.csv",
    "silhouette_scores": FINAL_TABLES_DIR /
"silhouette_scores_reduced.csv",
    "cluster_method_comparison": FINAL_TABLES_DIR /
"cluster_method_comparison_reduced.csv",
    "cluster_descriptions": FINAL_TABLES_DIR /
"cluster_descriptions_reduced.csv",
    "pca_variance_figure": CLUSTERING_FIGURES_DIR /
"pca_explained_variance_reduced.png",
    "pca_country_map_figure": CLUSTERING_FIGURES_DIR /
"pca_country_map_reduced.png",
```

```
        "silhouette_figure": CLUSTERING_FIGURES_DIR / "silhouette_scores_reduced.png",
        "kmeans_pca_figure": CLUSTERING_FIGURES_DIR / "kmeans_clusters_pca_reduced.png",
        "cluster_method_comparison_figure": CLUSTERING_FIGURES_DIR / "cluster_method_comparison_reduced.png",
        "cluster_profiles_heatmap_figure": CLUSTERING_FIGURES_DIR / "cluster_profiles_heatmap_reduced.png",
    }

    reduced_missingness = (
        latest_snapshot[reduced_clustering_metrics]
        .isna()
        .agg(["sum", "mean"])
        .T
        .rename(columns={"sum": "missing_count", "mean": "missing_rate"})
        .sort_values("missing_rate", ascending=False)
    )
    reduced_missingness.to_csv(reduced_paths["missingness"])

    reduced_country_missingness = latest_snapshot[["country_code", "year"] + reduced_clustering_metrics].copy()
    reduced_country_missingness["missing_metrics_count"] = reduced_country_missingness[reduced_clustering_metrics].isna().sum(axis=1)
    reduced_country_missingness["missing_metrics"] = reduced_country_missingness[reduced_clustering_metrics].isna().apply(
        lambda row: ", ".join(row.index[row].tolist()), axis=1
    )
```

```
    reduced_excluded_countries = reduced_country_missingness.query("missing_metrics_count > 0").copy()
    reduced_excluded_countries.to_csv(reduced_paths["excluded_countries"], index=False)

    print(f"Reduced model metrics: {len(reduced_clustering_metrics)}")
    print(f"Countries available in {analysis_year}: {latest_snapshot['country_code'].nunique()}")
    print(f"Countries excluded in reduced model: {reduced_excluded_countries.shape[0]}")
    print(f"Saved reduced missingness table: {reduced_paths['missingness']}")

    reduced_missingness

    # Cell 24 (notebook cell 40)
    reduced_complete_snapshot = latest_snapshot.dropna(subset=reduced_clustering_metrics).copy()

    if reduced_complete_snapshot.shape[0] < 4:
        raise ValueError(
            "Not enough complete countries for reduced clustering. "
            "Consider using another year or reducing the feature set."
        )

    reduced_log_transform_metrics = [
        metric
        for metric in log_transform_metrics
        if metric in reduced_clustering_metrics
    ]
```

```
    reduced_modeling_data = reduced_complete_snapshot[["country_code", "year"] + reduced_clustering_metrics].copy()
    reduced_feature_matrix = reduced_modeling_data[reduced_clustering_metrics].copy()

    for metric in reduced_log_transform_metrics:
        if (reduced_feature_matrix[metric] < 0).any():
            raise ValueError(f"Metric {metric} contains negative values and cannot be log-transformed.")
        reduced_feature_matrix[f"log1p_{metric}"] = np.log1p(reduced_feature_matrix[metric])

    reduced_modeling_features = [
        f"log1p_{metric}" if metric in reduced_log_transform_metrics else metric
        for metric in reduced_clustering_metrics
    ]
    reduced_feature_matrix = reduced_feature_matrix[reduced_modeling_features]

    reduced_scaler = StandardScaler()
    reduced_features_scaled = reduced_scaler.fit_transform(reduced_feature_matrix)
    reduced_features_scaled_df = pd.DataFrame(
        reduced_features_scaled,
        columns=reduced_modeling_features,
        index=reduced_modeling_data["country_code"],
    )

    print(f"Countries used in reduced clustering: {reduced_modeling_data['country_code'].nunique()}")
```

```
print(f"Features used in reduced model: {len(reduced_modeling_features)}")

reduced_features_scaled_df.head()

# Cell 25 (notebook cell 41)
reduced_n_components = min(len(reduced_modeling_features), reduced_features_scaled_df.shape[0] - 1)
reduced_pca = PCA(n_components=reduced_n_components, random_state=42)
reduced_pca_scores = reduced_pca.fit_transform(reduced_features_scaled_df)

reduced_pca_columns = [f"pca_{i + 1}" for i in range(reduced_n_components)]
reduced_pca_scores_df = pd.DataFrame(reduced_pca_scores, columns=reduced_pca_columns)
reduced_pca_scores_df.insert(0, "country_code", reduced_modeling_data["country_code"].values)
reduced_pca_scores_df.insert(1, "year", reduced_modeling_data["year"].values)
reduced_pca_scores_df.to_csv(reduced_paths["pca_scores"], index=False)

reduced_explained_variance = pd.DataFrame(
    {
        "component": reduced_pca_columns,
        "explained_variance_ratio": reduced_pca.explained_variance_ratio_,
        "cumulative_explained_variance": np.cumsum(reduced_pca.explained_variance_ratio_),
    }
)
```

```
reduced_pca_loadings = pd.DataFrame(
    reduced_pca.components_.T,
    index=reduced_modeling_features,
    columns=reduced_pca_columns,
)
reduced_pca_loadings.to_csv(reduced_paths["pca_loadings"])

print(f"Saved reduced PCA scores: {reduced_paths['pca_scores']}")
print(f"Saved reduced PCA loadings: {reduced_paths['pca_loadings']}")

reduced_explained_variance.head(10)

# Cell 26 (notebook cell 42)
plt.figure(figsize=(9, 5))
sns.barplot(
    data=reduced_explained_variance,
    x="component",
    y="explained_variance_ratio",
    color="#5F8D4E",
)
sns.lineplot(
    data=reduced_explained_variance,
    x="component",
    y="cumulative_explained_variance",
    marker="o",
    color="#C44E52",
)
plt.title(f"Reduced PCA explained variance, {analysis_year}")
plt.xlabel("Principal component")
```

```
plt.ylabel("Explained variance ratio")
plt.xticks(rotation=45)
plt.tight_layout()
plt.savefig(reduced_paths["pca_variance_figure"], dpi=300, bbox_inches="tight")
plt.show()

print(f"Saved figure: {reduced_paths['pca_variance_figure']}")

# Cell 27 (notebook cell 43)
reduced_pca_plot_data = reduced_pca_scores_df.copy()

fig, ax = plt.subplots(figsize=(12, 8))
sns.scatterplot(data=reduced_pca_plot_data, x="pca_1", y="pca_2", s=90, color="#5F8D4E", ax=ax)

x_min, x_max = reduced_pca_plot_data["pca_1"].min(), reduced_pca_plot_data["pca_1"].max()
y_min, y_max = reduced_pca_plot_data["pca_2"].min(), reduced_pca_plot_data["pca_2"].max()
x_padding = max((x_max - x_min) * 0.18, 0.5)
y_padding = max((y_max - y_min) * 0.18, 0.5)
ax.set_xlim(x_min - x_padding, x_max + x_padding)
ax.set_ylim(y_min - y_padding, y_max + y_padding)

for _, row in reduced_pca_plot_data.iterrows():
    ax.annotate(
        row["country_code"],
        (row["pca_1"], row["pca_2"]),
        textcoords="offset points",
```

```
            xytext=(5, 5),
            fontsize=9,
            ha="left",
            va="bottom",
            clip_on=False,
        )

    ax.axhline(0, color="gray", linewidth=0.8, alpha=0.7)
    ax.axvline(0, color="gray", linewidth=0.8, alpha=0.7)
    ax.set_title(f"Countries in reduced PCA space, {analysis_year}")
    ax.set_xlabel("PCA 1")
    ax.set_ylabel("PCA 2")
    fig.tight_layout()
    fig.savefig(reduced_paths["pca_country_map_figure"], dpi=300,
bbox_inches="tight")
    plt.show()

    print(f"Saved figure: {reduced_paths['pca_country_map_figure']}")

    # Cell 28 (notebook cell 44)
    reduced_pca_1_interpretation =
describe_component_loadings(reduced_pca_loadings, "pca_1")
    reduced_pca_2_interpretation =
describe_component_loadings(reduced_pca_loadings, "pca_2")

    print("Reduced PCA 1 найбільше пов'язана з такими метриками:")
    display(reduced_pca_1_interpretation)
    print("Reduced PCA 2 найбільше пов'язана з такими метриками:")
    display(reduced_pca_2_interpretation)
```

```
# Cell 29 (notebook cell 45)
reduced_max_k = min(6, reduced_features_scaled_df.shape[0] - 1)
reduced_k_values = list(range(2, reduced_max_k + 1))

reduced_silhouette_records = []
for k in reduced_k_values:
    reduced_kmeans = KMeans(n_clusters=k, random_state=42, n_init=20)
    reduced_labels = reduced_kmeans.fit_predict(reduced_features_scaled_df)
    reduced_score = silhouette_score(reduced_features_scaled_df, reduced_labels)
    reduced_silhouette_records.append({"k": k, "silhouette_score": reduced_score})

reduced_silhouette_scores = pd.DataFrame(reduced_silhouette_records)
reduced_silhouette_scores.to_csv(reduced_paths["silhouette_scores"], index=False)

reduced_best_k = int(reduced_silhouette_scores.sort_values("silhouette_score", ascending=False).iloc[0]["k"])
print(f"Selected k by silhouette score for reduced model: {reduced_best_k}")

reduced_silhouette_scores

# Cell 30 (notebook cell 46)
plt.figure(figsize=(8, 5))
sns.lineplot(data=reduced_silhouette_scores, x="k", y="silhouette_score", marker="o", color="#5F8D4E")
plt.axvline(reduced_best_k, color="#C44E52", linestyle="--", linewidth=1.5, label=f"selected k = {reduced_best_k}")
```

```
        plt.title(f"Reduced model silhouette scores for K-Means, {analysis_year}")
        plt.xlabel("Number of clusters")
        plt.ylabel("Silhouette score")
        plt.legend()
        plt.tight_layout()
        plt.savefig(reduced_paths["silhouette_figure"], dpi=300, bbox_inches="tight")
        plt.show()

        print(f"Saved figure: {reduced_paths['silhouette_figure']}")

        # Cell 31 (notebook cell 47)
        reduced_kmeans_model = KMeans(n_clusters=reduced_best_k,
random_state=42, n_init=50)
        reduced_kmeans_labels =
reduced_kmeans_model.fit_predict(reduced_features_scaled_df)

        reduced_cluster_results = reduced_modeling_data.copy()
        reduced_cluster_results["kmeans_cluster"] = reduced_kmeans_labels
        reduced_cluster_results =
reduced_cluster_results.merge(reduced_pca_scores_df, on=["country_code", "year"],
how="left")

        plt.figure(figsize=(10, 7))
        sns.scatterplot(
            data=reduced_cluster_results,
            x="pca_1",
            y="pca_2",
            hue="kmeans_cluster",
            palette="Set2",
            s=100,
```

```
        )
        for _, row in reduced_cluster_results.iterrows():
            plt.text(row["pca_1"], row["pca_2"], row["country_code"], fontsize=9,
ha="left", va="bottom")
        plt.axhline(0, color="gray", linewidth=0.8, alpha=0.7)
        plt.axvline(0, color="gray", linewidth=0.8, alpha=0.7)
        plt.title(f"Reduced K-Means clusters in PCA space, {analysis_year}")
        plt.xlabel("PCA 1")
        plt.ylabel("PCA 2")
        plt.legend(title="Cluster")
        plt.tight_layout()
        plt.savefig(reduced_paths["kmeans_pca_figure"], dpi=300,
bbox_inches="tight")
        plt.show()

        print(f"Saved figure: {reduced_paths['kmeans_pca_figure']}")
        reduced_cluster_results[["country_code", "year", "kmeans_cluster", "pca_1",
"pca_2"]].head()

        # Cell 32 (notebook cell 48)
        reduced_hierarchical_model =
AgglomerativeClustering(n_clusters=reduced_best_k, linkage="ward")
        reduced_hierarchical_labels =
reduced_hierarchical_model.fit_predict(reduced_features_scaled_df)

        reduced_cluster_results["hierarchical_cluster"] = reduced_hierarchical_labels

        reduced_cluster_comparison = pd.crosstab(
            reduced_cluster_results["kmeans_cluster"],
            reduced_cluster_results["hierarchical_cluster"],
```

```
        rownames=["kmeans_cluster"],
        colnames=["hierarchical_cluster"],
    )
    reduced_cluster_comparison.to_csv(reduced_paths["cluster_method_comparison"])

    plt.figure(figsize=(8, 6))
    sns.heatmap(reduced_cluster_comparison, annot=True, fmt="d", cmap="Greens", cbar=False)
    plt.title("Comparison of reduced K-Means and hierarchical clusters")
    plt.xlabel("Hierarchical cluster")
    plt.ylabel("K-Means cluster")
    plt.tight_layout()
    plt.savefig(reduced_paths["cluster_method_comparison_figure"], dpi=300, bbox_inches="tight")
    plt.show()

    print(f"Saved figure: {reduced_paths['cluster_method_comparison_figure']}")
    reduced_cluster_comparison

    # Cell 33 (notebook cell 49)
    reduced_cluster_profiles = (
        reduced_cluster_results
        .groupby("kmeans_cluster")
        .agg(
            countries_count=("country_code", "nunique"),
            countries=("country_code", lambda values: ", ".join(sorted(values.astype(str)))),
            institutions_count_mean=("institutions_count", "mean"),
            total_students_mean=("total_students", "mean"),
```

```
            mean_institution_size_mean=("mean_institution_size", "mean"),
            median_institution_size_mean=("median_institution_size", "mean"),
            foreign_students_share_mean=("foreign_students_share", "mean"),
            isced7_share_mean=("isced7_share", "mean"),
            isced8_share_mean=("isced8_share", "mean"),
            top_5_share_mean=("top_5_share", "mean"),
            top_10_share_mean=("top_10_share", "mean"),
            gini_students_mean=("gini_students", "mean"),
        )
        .reset_index()
    )

    reduced_cluster_profiles.to_csv(reduced_paths["cluster_profiles"],
index=False)

    reduced_profile_metric_columns = [column for column in
reduced_cluster_profiles.columns if column.endswith("_mean")]
    reduced_profile_for_heatmap =
reduced_cluster_profiles.set_index("kmeans_cluster")[reduced_profile_metric_colum
ns].copy()
    reduced_profile_z_scores = pd.DataFrame(
        StandardScaler().fit_transform(reduced_profile_for_heatmap),
        index=reduced_profile_for_heatmap.index,
        columns=reduced_profile_for_heatmap.columns,
    )

    plt.figure(figsize=(12, max(4, 1.2 * reduced_profile_z_scores.shape[0])))
    sns.heatmap(reduced_profile_z_scores, annot=True, fmt=".2f", cmap="vlag",
center=0)
    plt.title("Standardized reduced cluster profiles based on original metrics")
```

```
        plt.xlabel("Metric")
        plt.ylabel("K-Means cluster")
        plt.xticks(rotation=45, ha="right")
        plt.tight_layout()
        plt.savefig(reduced_paths["cluster_profiles_heatmap_figure"], dpi=300,
bbox_inches="tight")
        plt.show()

        print(f"Saved reduced cluster profiles: {reduced_paths['cluster_profiles']}")
        print(f"Saved figure: {reduced_paths['cluster_profiles_heatmap_figure']}")
        reduced_cluster_profiles

        # Cell 34 (notebook cell 50)
        reduced_cluster_descriptions =
build_cluster_description(reduced_profile_z_scores)
        reduced_cluster_descriptions.to_csv(reduced_paths["cluster_descriptions"],
index=False)

        reduced_cluster_results.to_csv(reduced_paths["cluster_results"], index=False)

        reduced_cluster_results_duplicates =
reduced_cluster_results.duplicated(subset=["country_code", "year"]).sum()
        reduced_cluster_sizes = (
            reduced_cluster_results["kmeans_cluster"]
            .value_counts()
            .sort_index()
            .rename_axis("kmeans_cluster")
            .reset_index(name="countries_count")
        )
```

```
reduced_small_clusters = reduced_cluster_sizes.query("countries_count <= 1")

print(f"Saved reduced cluster results: {reduced_paths['cluster_results']}")
print(f"Duplicated country_code + year in reduced_cluster_results: {reduced_cluster_results_duplicates}")
print(f"Countries included in reduced model: {reduced_cluster_results['country_code'].nunique()}")
print(f"Countries excluded in reduced model: {reduced_excluded_countries.shape[0]}")
print(f"Reduced clusters with one or zero countries: {reduced_small_clusters.shape[0]}")

if reduced_cluster_results_duplicates != 0:
    raise ValueError(
        f"reduced_cluster_results has duplicate country_code + year rows: {reduced_cluster_results_duplicates}"
    )

reduced_cluster_sizes

# Cell 35 (notebook cell 52)
clustering_model_comparison = pd.DataFrame(
    [
        {
            "model_name": "full_model",
            "year": analysis_year,
            "metrics_count": len(clustering_metrics),
            "countries_available": latest_snapshot["country_code"].nunique(),
            "countries_included": cluster_results["country_code"].nunique(),
```

```
                "countries_excluded": excluded_countries.shape[0],
                "best_k": best_k,
                "best_silhouette_score": silhouette_scores["silhouette_score"].max(),
            },
            {
                "model_name": "reduced_model_without_staff_finance",
                "year": analysis_year,
                "metrics_count": len(reduced_clustering_metrics),
                "countries_available": latest_snapshot["country_code"].nunique(),
                "countries_included":
reduced_cluster_results["country_code"].nunique(),
                "countries_excluded": reduced_excluded_countries.shape[0],
                "best_k": reduced_best_k,
                "best_silhouette_score":
reduced_silhouette_scores["silhouette_score"].max(),
            },
        ]
    )

    clustering_model_comparison_path = FINAL_TABLES_DIR /
"clustering_model_comparison.csv"
    clustering_model_comparison.to_csv(clustering_model_comparison_path,
index=False)

    print(f"Saved model comparison: {clustering_model_comparison_path}")
    clustering_model_comparison

    # Cell 36 (notebook cell 54)
    print("Короткий підсумок:")
    print(f"- Рік аналізу: {analysis_year}")
```

```
print(f"- Повна модель: {cluster_results['country_code'].nunique()} країн, {len(clustering_metrics)} метрик, k = {best_k}, silhouette = {silhouette_scores['silhouette_score'].max():.3f}")
print(f"- Reduced-модель: {reduced_cluster_results['country_code'].nunique()} країн, {len(reduced_clustering_metrics)} метрик, k = {reduced_best_k}, silhouette = {reduced_silhouette_scores['silhouette_score'].max():.3f}")
print(f"- Повна модель збережена в: {CLUSTER_RESULTS_PATH}")
print(f"- Reduced-модель збережена в: {reduced_paths['cluster_results']}")
print(f"- Порівняння моделей збережено в: {clustering_model_comparison_path}")
print("Обмеження: висновки описові; кластери залежать від метрик, року аналізу і повноти даних.")
print("Reduced-модель має ширше покриття, але не враховує фінансову та кадрову метрики.")
```