Database of study data for Germany

Georg F.K. Höhn

Ironhack

23 May 2025

Overview

Introduction

Database

- source: Federal Statistical Office (www-genesis.destatis.de)
- 5 datasets concerning subjects of study (plus gender and – partly – nationality)
    1. total number of students
    2. number of incoming students
    3. number of final exams (with information about result: passed/not passed)
    4. number of personnel by subject cluster
    5. number of professors by subject group
- effectively used for now: 1 and 2
- selected time frame: 2018–2023
- also: taxonomies for subjects < subject clusters < subject groups
    - 2 distinct versions for study subjects and personnel (pdfs)

Questions for SQL queries

1. What was the total number of students in Germany in 2023/24?
2. Overall top 10 subjects in 2023/24
3. Overall gender distribution of students in 2023/24
4. Gender distribution in the 5 most studied subjects in 2023/24
5. What were the top subjects by gender?
6. Top 10 subjects studied by foreign students
7. Distribution of the student numbers across subject groups
8. Change of yearly intake by group over time period
9. Change of yearly intake for language related subjects
10. Change of yearly intake for linguistic subjects only
11. 10 subjects with highest percentual drop in intake
12. 10 subjects with highest percentual rise in intake

Data preparation

Challenge 1: Data formating and cleaning

- data available in several formats: xls, csv and csv-flat
  - first two not practical for automatic processing
- csv-flat: extensible, but verbose format
- every property described by 4 columns
  - `variable_code`: code for the encoded variable
  - `variable_label`: plain text label for variable
  - `variable_attribute_code`: code for attribute/value assigned to variable
  - `variable_attribute_label`: plain text label for the attribute/value
- similarly for time, target value

```
stud.head(10).T
```

✓ 0.0s

	0	1
statistics_code	21311	21311
statistics_label	Statistik der Studenten	Statistik der Studenten
time_code	SEMEST	SEMEST
time_label	Semester	Semester
time	2018-10P6M	2022-10P6M
1_variable_code	DINSG	DINSG
1_variable_label	Deutschland insgesamt	Deutschland insgesamt
1_variable_attribute_code	DG	DG
1_variable_attribute_label	Deutschland	Deutschland
2_variable_code	NAT	NAT
2_variable_label	Nationalität	Nationalität
2_variable_attribute_code	NATD	NATD
2_variable_attribute_label	Deutsche	Deutsche
3_variable_code	GES	GES
3_variable_label	Geschlecht	Geschlecht
3_variable_attribute_code	GESM	GESW
3_variable_attribute_label	männlich	weiblich
4_variable_code	BILSF1	BILSF1
4_variable_label	Studienfach	Studienfach
4_variable_attribute_code	SF241	SF220
4_variable_attribute_label	Kerntechnik/Kernverfahrenstechn.(ab 2020 zu SF211)	Milch- und Molkereiwirtschaft
value	3	21
value_unit	Anzahl	Anzahl
value_variable_code	BIL002	BIL002
value_variable_label	Studierende	Studierende

Figure 1: Original data format (transposed)

Tasks

- extract only relevant data
- normalise data
    - time marked by semesters for total students and incoming students table, plain year elsewhere
    - subject codes in student-related tables prefixed by ‘SF’
    - format for cluster and group codes in personnel/professorial data is even more messy

Data coding

- `time` (renamed to `year`):
  - in students and incoming student data provided as more complex string, elsewhere just string of year
  - normalised to year int everywhere
- `gender` coded binary: GESW, GESM
  - recoded as f, m
- `nationality` coded binary: NATD, NATD
  - recoded as domestic, foreign
- `subj_code`, `subj_name`: as strings
- `number`: aggregated student count for a particular combination of gender, nationality and subject

Challenge 2: Taxonomies

- grouping of subjects should follow established standards
- two distinct taxonomies
  - the personnel-related one has some overlap in the titles, but codes are completely distinct from the study-related scheme
- only available as pdfs
- translations would be helpful for presentation

Extraction

- employ LLMs to generate csv or JSON incl. translations
- JSON should be computationally more effective (less energy waste?)
- multiple incomplete attempts with ChatGPT, Claude
- only Gemini produced a well-formed and (almost) complete JSON (probably?)

Implementation

General workflow

- database setup (see below)
- dictionary to store information on each dataframe
    - location for csv
    - name for db table
    - after creation: reference to actual df
    - (also bool for whether student-related, but unused)
- transform JSON of taxonomies into dataframes
- append to database tables
- define and run cleaning function on 5 dataframes

Database setup

- using `sqlalchemy` to set up database connection and tables
- drop and recreate tables on each run (allows appending of data)

Figure 2: Database schema

Cleaning function

```
def clean_dat(in_df, dstore, dbengine, dropcols,
rencols,complexdate=False,insgesamt=False)
```

- input
  - source dataframe
  - dictionary entry for target dataframe
  - SQL engine
  - list of columns to drop
  - dictionary of column renames
  - switches for special data treatment
- performs necessary cleaning operations depending on dfs
  - drop and rename columns
  - replace deprecated subject codes where appropriate
- save cleaned csv file based on location info in `dstore` dictionary
- append table to database table defined in `dstore`
- return cleaned dataframe

```
df_s.head()
```

```
df_s.head()
```

✓ 0.0s

	year	nationality	gender	subj_code	subj_name	number
0	2018	domestic	m	211	Kerntechnik/Kernverfahrenstechn.(ab 2020 zu SF211)	3
1	2022	domestic	f	220	Milch- und Molkereiwirtschaft	21
2	2022	foreign	f	280	Kartografie	44
3	2023	foreign	m	086	Katholische Theologie, - Religionslehre	407
4	2023	foreign	f	272	Alte Geschichte	18

Figure 3: Head of a cleaned dataframe

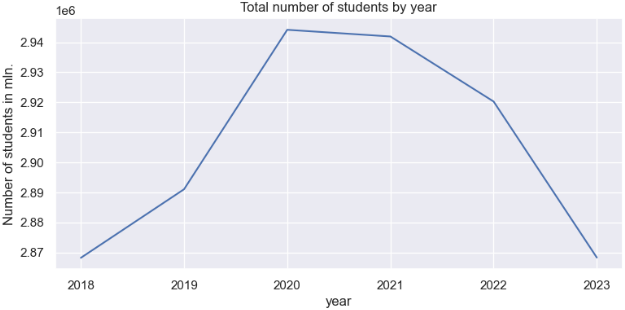
EDA

Practical complication

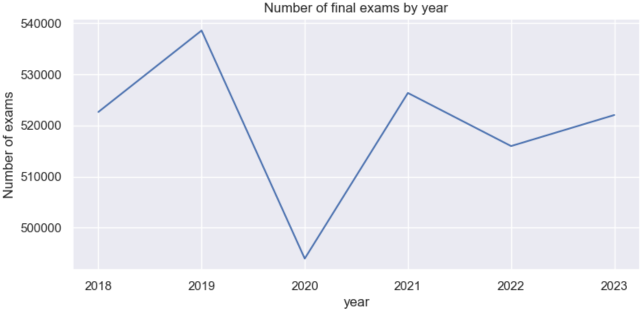
- data already aggregated by gender, (nationality), subject
- standard functions like `.describe()` or `.count()` of limited use
- instead: use `.sum()` aggregation function where relevant

General insights

- number of students actually picks up around COVID before dropping again (see next slide)
- as we'll see below, the number of incoming students actually dropped around the same time, so why did the overall number rise?
- answer: number of exams also had a very strong dip in 2020, so fewer people leaving university
- the notebook has the full story and graphs

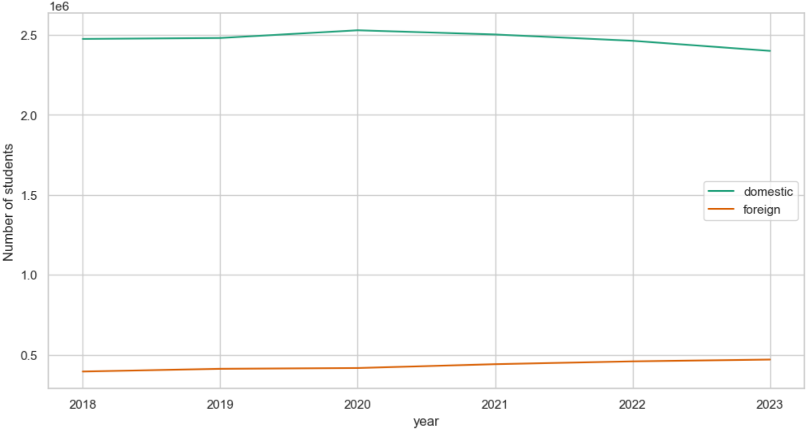


Total number of students by year
1e6
2.94
2.93
2.92
2.91
2.90
2.89
2.88
2.87
Number of students in mln.
2018
2019
2020
2021
2022
2023
year

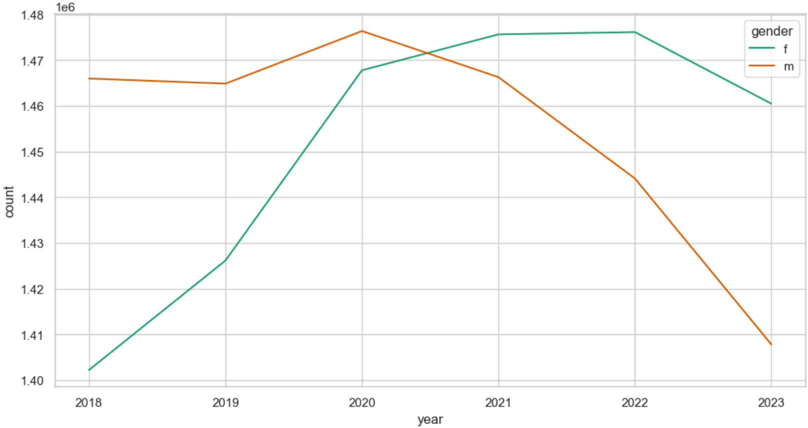


Number of final exams by year
540000
530000
520000
510000
500000
Number of exams
2018
2019
2020
2021
2022
2023
year

- COVID-drop among domestic students
- number of foreign student rose slightly over COVID (remote accessibility?)

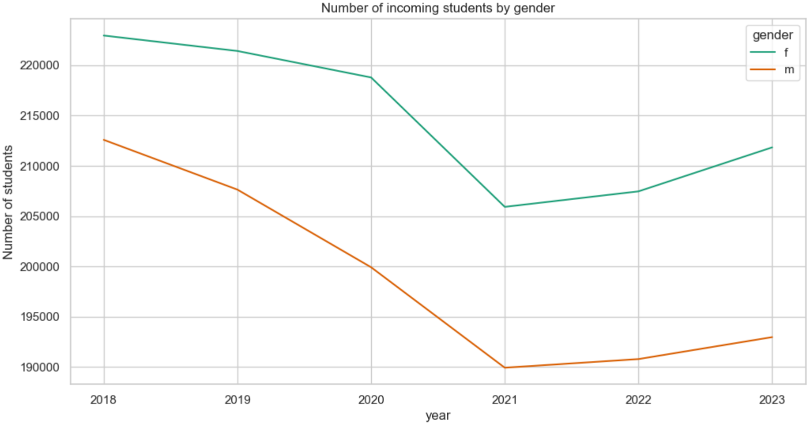


Gender over time

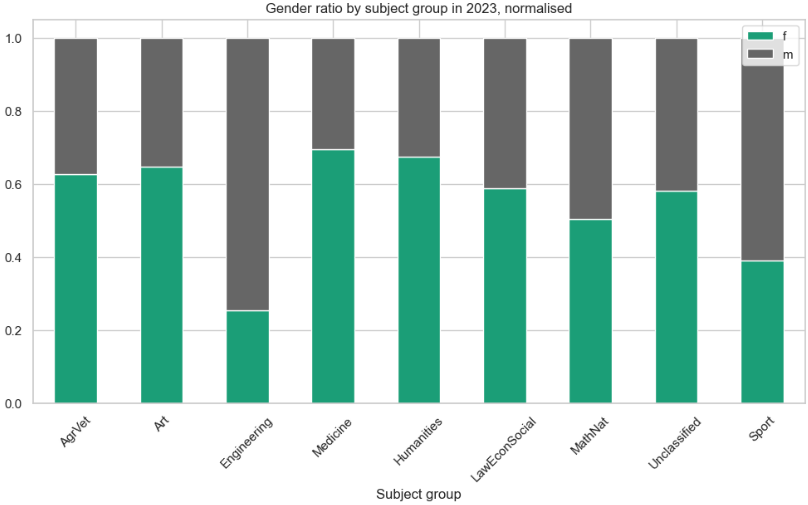


- total number of female student population overtook male population in 2021

- not directly due to COVID, probably longer trend



Gender x subject group



Chi²-test

Subject group	f	$f_{expected}$	m	$m_{expected}$
AgricForestNutriVet	37981	30922.9	22750	29808.1
Art, Art History	63354	49814.9	34480	48019.1
Engineering Sciences	189389	381224	559316	367481
Medicine/Health Sciences	143746	105417	63288	101617
Humanities	200420	151479	97078	146019
Law, Economics, Social Sciences	657197	568559	459425	548063
Mathematics, Natural Sciences	151683	153177	149149	147655
Outside Classification	4534	3975.66	3274	3832.34
Sport	12177	15910.3	19070	15336.7

$\chi^2 = 298192, df = 8, p < 0.01$

There is a statistically significant interaction between gender and subject group.

Selected SQL prompts

In my project, SQL prompts are provided in two ways:

- in a separate SQL script
- inside the jupyter notebook calling the database using `sqlalchemy`

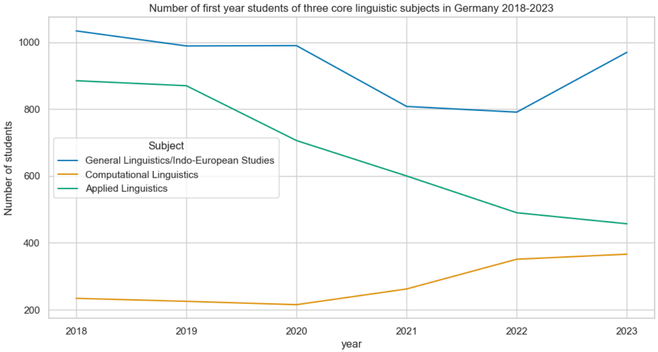
6. Which were the top 10 subjects studied by foreigners?

```sql
SELECT st.subj_name_en,SUM(number) AS total_number
FROM students s
LEFT JOIN subject_taxonomy st
   ON s.subj_code = st.subj_code
WHERE s.nationality = 'domestic' AND s.year = 2023
GROUP BY s.subj_code
ORDER BY total_number DESC
LIMIT 10;
```

	overall	domestic	foreign
1	Business Adm.	Business Adm.	Business Adm.
2	Computer Sc.	Computer Sc.	Computer Sc.
3	Law	Law	Elec. Engineering
4	Psychology	Psychology	Mech. Engineering
5	Medicine	Medicine	Int. Business Adm.
6	Economic Sc.	Social Work	Medicine
7	Social Work	Economic Sc.	Economic Sciences
8	Mech. Engineering	Mech. Engineering	Indust. Engineering
9	Business Informatics	German Studies	Civil Engineering
10	German Studies	Business Informatics	Law

10. Intake changes for linguistic subjects?

```
SELECT st.subj_name_en, is2.year, SUM(is2.number) AS student_num
FROM incoming_students is2
LEFT JOIN subject_taxonomy st ON is2.subj_code = st.subj_code
WHERE is2.subj_code IN ('152','284','160')
GROUP BY is2.subj_code, is2.year
ORDER BY is2.subj_code, is2.year;
```



Conclusion

- database, cleaned csv files, notebook: github.com/gfkpth/project-2-eda-sql
- (somewhat) cleaned up version of repository can be found on github.com/gfkpth/data-study-subjects

Technical lessons

- data integration is fun(ky)
- data formats can be a mess to sort out
- extraction from pdfs via LLMs can be viable, but beware limits and errors
  - best result with Gemini
  - still at least two missing data points
- keeping jupyter notebooks tidy is tricky
  - moving cells is a pain in VSCode
- dictionaries are great for central metadata storage

Content-wise

- many more interesting aspects in data
- inspecting longer timeframes would highlight trends more clearly

Thanks for your attention!