Heinrich Heine
University Düsseldorf

Institut für Informatik
Datenbanken und Informationssysteme
Prof. Dr. Stefan Conrad
Thomas Germer

WiSe 24/25
Assignment 2
Due October 29, 8:00 AM

Relational Databases and Data Analysis

- As usual, use `zipme.py` to create the archive to upload in ILIAS.
- We will use SQLite3 instead of PostgreSQL for this assignment since it is easier to get started. In general, PostgreSQL is preferable since it adheres more strictly to the SQL standard.

Quick start into SQLite3

- Later, we will interface with sqlite3 using Python, but to get started, we will use the command line tool `sqlite3`. If your operating system does not come with SQLite3 preinstalled, you might be able to install it with `sudo apt install sqlite3`. Alternatively, the binary is surprisingly easy to build from source. Python already includes the `sqlite3` module, so nothing to install there.
- Open the database in sqlite:

  ```
  sqlite3 data/database.db
  ```

- Run an SQL query:

  ```
  SELECT * FROM customer;
  ```

- Show prettier output (create file `~/.sqliterc` in your home path to make this permanent):

  ```
  .headers on
  .mode column
  .nullvalue NULL
  .separator ROW "\n"
  ```

- Show the database schema:

  ```
  .schema
  ```

- Quit:

  ```
  .quit
  ```

  Or alternatively, pressing `Ctrl + D` might also work.

General rules for SQL queries in this lecture

- If not otherwise stated, the result should have all attributes of the specified relation. For example, if a question asks for all stores, both the ID and the city should be returned.
- If you use a `GROUP BY` statement, make sure that you group by *all* referenced non-aggergate attributes. Some databases may be more lenient in that regard, but we will grade it as incorrect as demanded by the SQL 92 standard.

Exercise 1 *SQL queries* (7 Points)

You are given a relational model of a supermarket chain where underlined attributes indicate primary keys and overlined attributes indicate foreign keys. The foreign key `city_id` references `city.id`, `product_id` references `product.id`, `store_id` references `store.id` and `customer_id` references `customer.id`.

`product(id, price, productname)`

`city(id, cityname)`

`store(id, city_id)`

`customer(id, firstname, lastname)`

`sold_in(product_id, store_id)`

`purchase(id, customer_id, product_id, amount)`

Formulate the following queries using SQL. Write your queries in the corresponding Python files and run the tests.

(a) Find the names of all products.

(b) Find all customers who have a lastname starting with "Sch".

(c) How many products with *different names* are there?

(d) Find all cities with a store that sells `Potatoes`.

(e) Which customers made exactly 5 purchases with a total amount of at least 28 products?

(f) Which customers *never* purchased `Pizza`? Order them by last name in descending order.

(g) What is wrong with this query? Write your answer in `exercise_1_g.txt`.[1]

```sql
SELECT
    customer.firstname,
    customer.lastname,
    COUNT(*) AS n
FROM
    customer
GROUP BY
    customer.lastname;
```

Exercise 2 *SQL database* (3 Points)

Create your own SQLite3 database named `mydatabase.db` in `exercise_2.py` using Python. The database must have one table with 1000 rows and at least two columns of data in it.

[1] Hint: sqlite will incorrectly accept this query as valid.