Databases for data analytics

Home Assignments Description 2026

Assignments Submission Requirements:

You must submit **two files** as part of this assignment:

1. A **PDF file** containing the following for each question:
    - The SQL query you wrote.
    - The count of rows in the output.
    - A screenshot of the results of running the query.
    - A brief and clean explanation of your approach and reasoning, in English.
2. **A SQL file** containing your **SQL queries**.

- The file names must be **identical** in both submissions and must follow the format

  student_<id>_sql_<task_num> (e.g., student_123456789_sql_1.pdf and student_123456789_sql_1.sql). Task_num is its number in this document.

  When the task is a team task, the format should be team_<team_name>_student_<id>_sql_<task_num>. You should choose a unique team name (e.g, by verifying uniqueness with the others, concatenating your names).

Assignments

1 Introduction and settings (Mandatory, zero points)

The assignment is to install a personal working environment - MySql workbench and the IMDB database. Coming with a laptop with such an environment to class will allow you to practice writing in class. Therefore, **complete this task before the second lesson**.

GitHub

1. Read the course GitHub guide
2. Create a personal GitHub profile (if you have one already, you can use it) This is required before they can download anything from GitHub
3. Submit the name of your profile

MySql

1. Install MySql Workbench. See detailed instructions for Windows and for Mac
    a. **Keep the username and password, you will need them when accessing the database with code.**
    b. You might be asked to install Java too, used by MySql
2. Download the IMDB schema, open it with an editor that can handle large files (e.g., Notepad++)
3. Copy and Paste the contents into the workbench “Query window” and runny pressing the lightning icon
4. Run “select count(*) from imdb_ijs.movies;” and verify that that you get 388,269

2 SQL(Select) (4 points)

Please write queries on the IMDB database that extract the following

1. All the movies whose rank is at least 9.
    a. Using “rank” as a name of a column in MySql is problematic. Explain why and

explain how you cope with it.

(Hint: There are 49,573 movies whose rank is at least 5. Check your query by adapting it to this value and see that you get the same result.)

2. All the different role names that include the string ‘him’
    i. Take care of being case insensitive
    ii. Note that the same role might appear in multiple movies, yet should only appear once in the results

        (Hint: There are 46,686 roles that contain the string ‘her’. Check your query by adapting it to this value and see that you get the same result.)

    iii. Do you find the role name ‘Himself’ appropriate? To which problems it might lead? Explain why.
    iv. Suggest a way to improve the returned list.
3. All movies whose name is longer than 95 characters, ordered by length
    a. Explain the prevalence of names in each length. Why is certain length much more common than the others?
    b. Bonus: Suggest a way to identify some of the problematic names and implement it.
4. Find at least 3 **first names** in the actors table that are most likely to be mistakes. There are many ways to make mistakes so we are liberal and will accept any justified answer.
    i. Explain how you found each of the names
    ii. Suggest a possible cause of the mistake
    iii. Suggest a way that would prevent the problem in the first place or identify it afterwards. Bonus: Find mistakes where it is not easy to do so.

3 Joins and DML (4 points)

Please write queries on the IMDB database for the following

1. All **movies** with a role 'himself' or with a 'herself' role (case insensitive)
    a. A movie with multiple such roles should be counted once
    b. Hint: There are 2,731 such movies in the eighties. Check your query by adapting it to this value and see that you get the same result.
2. All actors that Alfred Hitchcock directed

    Note: Hitchcock appear in IMDB as ”34658, Alfred (I), Hitchcock”

3. Find all genres in which Hitchcock did not direct any movie
    a. In the original db, genres appear as a name per movie, making the lookup slow. Improve the performance by creating a dedicated table for genres where each

name will appear only once.

4. Copy movies to a table named sad_movies and delete all those that do not belong to the genre ‘Comedy’.

4 Aggregations and Grouping (Mandatory, 6 points)

Please write queries on the IMDB database for the following

1. The classic period of *film noir* **crime movies was the forties and the nineties**.
    a. Using the movies table, create a table of film noir movies from the classical period.
    b. Create all pairs of film noir movies with at least 3 common actors.
2. Actors per role distribution
    a. Create a table with the number of actors per role.
    b. Compute the distribution of actors per role.
    c. Can there be roles with zero actors? Explain.
    d. Write code that can handle zero roles.
3. Typical role sets- compute for each movie its list of roles- a string with the names of all its roles (names of all roles played in the movie). The group_concat function is handy.
    a. Take care of not considering a role twice if it is acted by two actors in the movie.
    b. Note that the order of the roles is not important in the cast. (A, B) is the same cast as (B, A).
    c. Use only roles that were played by at least 10 actors (in the entire database)
4. Compute the probability of roles to appear in a movie and in a genre (e.g., the number of movies with a clown role out of all movies and out of all comedies). Window functions can help you.
5. The interesting question - find an interesting question about movies and answer it with SQL
    a. bonus for the very interesting question
    b. Interesting is subjective and we would like not to judge it. In extreme cases of trivial or lazy questions, grades will be reduced.

5 Subqueries and SE (4 points)

This is a coding exercise.

You should implement it in Python.

In class, we built the table actors_per_movie using a sequence of queries.

1. Write code that runs automatically these queries (do use the SQL queries from the example, the emphasis is on implementing in code).

    a. Add suffix “_code” to all table names
2. Verify that actors_per_movie and actors_per_movie_code have similar content
    a. Check that any key in one table also appears in the other.
    b. Check that records of the same key in both tables are similar in the other columns too.

6 DDL (Mandatory, 6 points)

Modeling (60% - 25% as customer, 25% as designer, 10% as reviewer)

1. The goal of the exercise is to model an **organization in the traffic domain** database for a specific set of user requirements.
2. Students should submit the work in teams of three
3. Each student should be in each of the roles - customer, designer, and a reviewer
4. The **customer** should provide a **use case** for sales databases, **requirements**, and their **justification**.The customer can be an organization of any domain, size or type.
5. The customer should list the use case in an issue in the course repository with the label ‘Design exercise’
6. **Customer** grade will be based on:
    a. Use case being realistic and important
    b. Requirements fit to use case
    c. Sensible justification
7. Once the use case is ready, the customer should mention the designer in the issue. **Customers** and **designers** are **encouraged** to discuss the requirements and change them. The documentation of the discussion in the issue will **increase** grades.
8. After the discussion is over, the designer should explain the use case in their own words.
9. The reviewer should be mentioned and decided if both the customer and designer understand the use case the same.
10. The **designer** should build a database and explain how it fits the requirements
    a. Database fit to requirements. You should submit SQL queries creating your schema.
    b. Explanation of fitting to requirements.
    c. Normalization of database.
11. Once done, the reviewer should judge if the design fits the requirements and of high quality.
12. **Answer format:**
    a. The customer should submit the work of all the parts of its system (cursomer, designer, reviewer).
    b. The document should have the following sections:
    c. ids of the team members and roles in the system
    d. **Customer requirements, written by the customer** (final results, motion changes due to designer and reviewer comments in a subsection)

e. **System design, written by the designer** (final results, motion changes in requirements and design due to process in a subsection)
f. **Review, written by the reviewer** what disagreement where found, which improvements were done due to the review
g. A link to the issue, with the work document uploaded to it.
h. Each team should submit **three systems**, using the same team name.

Normalization Kata (40%)

For each of the cases below write:

1. What is the problem and its possible impact?
2. How to solve the problem?
3. Is it a normalization role violation and if so, which one?

Cases

1. Movies table with the id column removed
2. Movies table with the id column duplicated
3. Table columns in Hebrew (e.g., “shem”, “shana”)
4. Directors column in movies
5. A constant field (e.g., Pi) in movies table
6. Having id number and employee number in employees table
7. The minimal id of an playing actor column in movies table

7 Data integrity (Mandatory, 6 points)

1. Find all column names appearing in more than a single table.
    a. When is it valid or even recommended and when is it dangerous?
    b. Repeat with the same column appearing in different data types.
2. We need a representation of various animals (e.g., mammal, fish, birds) in a database
    a. Suggest a fixed representation (in dedicated tables)
    b. Suggest a flexible representation (using Json
    c. Compare the advantages and disadvantages of the representations
3. Real world commit improvements
    a. Go over on the real world examples https://github.com/evidencebp/databases-course/tree/main/sql_improvments
    b. Go over the commits whose group is the 2 last digits in your id in the file https://github.com/evidencebp/sweets/tree/main/single_sql_commit

    c. Create a PR adding the solution to the commits, as in the example, the the commits that you selected.
4. **10 points bonus**: Suggest 3 ideas for query improvement and submit them as in the question before.
5. Open question:
    a. Think of a **possible** data integrity problem in the schema
    b. Explain the problem, its possible causes, and its possible impact
    c. Explain how the problem can be prevented by an alternative design, if possible
    d. Write a SQL query to verify if the problem indeed exists in the schema

8 Performance (4 points)

1. The following query compute pairs of comedies by the same director:

```
# Comedies pairs of the same director

Select *

from

movies_genres as fmg

join movies as fm

  on fmg.movie_id = fm.id

join imdb_ijs.movies_directors as fmd

  on fm.id = fmd.movie_id

join imdb_ijs.movies_directors as smd

  on fmd.director_id = smd.director_id

join imdb_ijs.movies as sm

  on smd.movie_id = sm.id

join imdb_ijs.directors as d

  on fmd.director_id = d.id

join movies_genres as smg

  on sm.id = smg.movie_id

where
```

```
fmd.movie_id != smd.movie_id # Avoid reflexiveness
  and fmd.movie_id > smd.movie_id # No symmetry
  and fmg.genre = 'Comedy'
  and smg.genre = 'Comedy'
order by
d.first_name, d.last_name, fm.name, sm.name
;
```

There is a lot that can be done to improve the query performance. Consider and use in your explanation factors like data volume, data types (from the columns that are being joined), and existing indexes.

1. Compute the execution plan and time of the original query.
2. Try to improve the query in steps. Find at least 3 steps.
3. Per step:
    a. Explain the intuition explaining why it should improve the performance.
    b. Implement the change.
    c. Compute the execution plan and time of the new query.
    d. We should learn from attempts that do not improve performance too. Explain why these steps did not improve.

2. Answer question “Real world commit improvements” and add solutions for 3 performance queries.

Accepted pull requests (1 points, per PR)

Contributions to the course repository are welcomed.

They can contain links to websites of interest, suggestions for examples and explanations.

Any accepted pull request will get 1 point with a grade of 100, up to 5 points per student.

Extraordinary contributions might get more points.

You must discuss with me the planned PR before submitting it.