

Databases for data analytics

Home Assignments Description

Course requirements

1. All assignments are mandatory.
2. Each of the assignments should be submitted within two weeks of the class in which it was given.
3. Assignments not submitted within the allowed submission timeframe will be graded 0.
4. Submission of at least 9 of the tasks is a prerequisite for taking the exam. Students who do not submit at least 9 assignments will not be allowed to take the exam and fail the course.
5. Each assignment accredits a prespecified number of points. These points get summed up towards the overall home assignment grade.
6. The overall assignment grade for students who submit more than 9 assignments will be calculated based on the assignments with the highest grades.
7. Overall, one can accrue up to 40 points in home assignments, i.e. home assignments comprise 40% of the final grade.
8. Submission of the final project is mandatory.
9. Up to 3 points will be given for participation in class.

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 exactly **X SELECT queries**, one for each question.
- 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.

Assignments

1 Introduction (zero points, no need to submit)

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

1. Install [MySql Workbench](#)
 - a. You might be asked to install Java too
 - b. Keep the username and password, you will need them when accessing the database with code.
2. Download the [IMDB schema](#), open it with an editor that can handle large files (e.g., [Notepad++](#))
3. Paste content into workbench and run
4. Run “select count(*) from imdb_ijs.movies;” and see the you get 388,269

2 SQL(Select) (4 points)

Please write queries on the IMDB database for the following

1. All the movies from the eighties
2. All movies that have ‘star’, insensitive to case, in their name
3. All movies whose name is longer than 80 characters

3 DML (4 points)

Please write queries on the IMDB database for the following

1. All Hitchcock's movies with an 'Assassin' role (consider a movie with two of them as a corner case).
2. All movies with a 'detective' or a 'murderer' roles (case insensitive)
3. All actors that played with Marilyn Monroe
4. Copy directors to directors_tmp and delete all those that did not direct any movie

4 Aggregations and Grouping (8 points)

Please write queries on the IMDB database for the following

1. Movies pairs with at least 10 common roles
 - a. Running time on the original roles table will be long. Please run on the selected roles table created by the following queries:

```
DROP TABLE IF EXISTS imdb_ijs.selected_movies; # Drop is important for reruns
```

```
Create table
Imdb_ijs.selected_movies
As
select
m.*
from
imdb_ijs.movies as m
join
(
select movie_id
, count(*) as roles
from
imdb_ijs.roles
where role not in ('', 'Himself', 'Herself')
group by movie_id
) movie_roles
on
m.id = movie_roles.movie_id
where
year between 1980 and 1989
and roles between 5 and 50
```

;

DROP TABLE IF EXISTS imdb_ijs.selected_roles; # Drop is important for reruns

```
create table
imdb_ijs.selected_roles
as
select r.*
from
imdb_ijs.roles as r
join
imdb_ijs.selected_movies as s
on
r.movie_id = s.id
where
role not in ('', 'Himself', 'Herself')
;
```

b.

2. Actors per movie distribution, with zero actors
3. The interesting question - find an interesting question about movies and answer it with SQL (bonus for the very interesting question)

5 Subqueries and SE (4 points)

This is a coding exercise.

You can implement it in either Python or R.

[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 (8 points)

Modeling (70% - 35% as customer, 35% as designer)

1. The goal of the exercise is to model a sales database for a specific set of user requirements.
2. Each student should pair **twice** with others
3. One of the student should be a customer and the other is the designer
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.
Customer grade will be based on:
 - a. Use case being realistic and important
 - b. Requirements fit to use case
 - c. Sensible justification
5. 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. **Customer** and **designer** are **encouraged** to discuss the requirements and change them. Such a documented process will **increase** grades.
 - b. Explanation of fitting to requirements.
 - c. Normalization of database.
6. Each student should be with different people and in different role in each pair

Normalization (30%)

1. A database system for basketball games presents per game
 - a. The teams
 - b. The winning team and the score
 - c. The highest scorer per team
2. Build a normalized database that can present these results. You should submit SQL queries creating your schema and explanation of your choices.
3. Write a query on that database that present the results

7 Data integrity (8 points)

1. Find, using the information_schema, all string columns and their number of records of length as the column length.
 - a. For example, column A in table Tab has 0 such records, column B has 10 records where the length of the column value in the record equals the column maximal length.
2.
 - a. Think of a reasonable logic to define directors_genres.
 - b. implement it
 - c. See if your table matches the original one (hint: we checked similarity in the coding exercise. Use the same logic).

3. Open question:
 - a. Think of a **possible** data integrity problem in the schema
 - b. Explain the problem, its possible reasons, 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. Copy movies to a table without indices
2. Compare execution plan and time on both tables of
 - a. Select of a specific id
 - b. Select of all movie of odd id
 - c. Select of all movies where $5000 < id < 10000$
3. Compare execution plan and time of select distinct year and select count(*) from group by year

12 Final project - basic recommendation system (8 points)

The goal of this assignment is to build a basic movie-to-movie recommendation system.

Throughout the course we discussed many queries that identify relations between movies, that might suggest that people that like the first might like the second.

Examples are movies of the same director, movies with the same roles, etc.

You should use the **imdb_movielens** schema.

The results should be evaluated with respect to the ratings.

The recommendations **must not use the ratings**.

1. Submit at least 20 pairs of movies, with recommendation level (1 - bad recommendation, 5 - neutral, 10 - great), and a justification. The recommendations of the class will serve as the ground truth.
2. Build 5 queries defining a relation of similarity between two movies.
3. All the recommendations into a single table.
4. Aggregate all basic recommendations into a single one.
5. Evaluate the precision and recall of your recommendations per model and for the aggregation.

Note that this is not a course on ML and recommendation systems, though it is a great opportunity to

enter the domain.

Hint - taking care of data validity problems can allow you to keep the same logic yet boost the performance.

Extra credit will be given for creative ideas and well performing recommendation, yet is not a requirement.

You are not required to implement the system using code. However, that will allow you to do multiple quick iterations and therefore it is very recommended.

Accepted pull requests (2 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 2 points with a grade of 100.

Extraordinary contributions might get more points.

