CIS 5500: Database and Information Systems

Homework 1: SQL

This homework is based on material in Module 1 and Module 2 (corresponding questions denoted with **M1** and **M2** hereafter). Responses should be submitted via Gradescope using the template file (*homework1.py*). More detailed submission instructions and autograder specifications can be found below in the Submission and Autograder sections respectively.

Overview

This assignment comprises three parts.

Before starting Part 1, you should install DataGrip and any relevant drivers required to query an Oracle SQL or MySQL database - please refer to the ***DataGrip Handout*** for further instructions.

In Part 1 you will query the IMDb dataset on an existing AWS RDS[1] Oracle SQL database.

In Part 2 you will then create your own MySQL database on RDS using the RDS handout. You will create a database instance by defining the tables and uploading the OpenFlights dataset.

In Part 3 you will query the instance that you created in Part 2.

Note: For Parts 1 and 3, you may NOT use DDL statements or views. You may, however, use the 'WITH… AS…' statement as needed.

Note: Make sure your answers match the provided schema. Schema is not case sensitive in Oracle but is case sensitive in MySQL.

Advice to Students

While you should be familiar with SQL syntax from the lessons in **M1** and **M2**, you may need to spend additional time setting up, troubleshooting, and debugging errors with the IDE (Integrated Development Environment), AWS, data uploading, and so on. You should try to use internet resources such as the official documentation or Stackoverflow to solve setup or other issues. If these resources do not solve your problem, you can ask the course staff for assistance via Ed or by coming to OH.

We understand that for many of you this is the first time using cloud resources, querying a relational database, and setting up your own database - and these understandably have a steep

[1] Relational Database Service

learning curve. The staff is happy to help you throughout this learning process but we do advise you to ***please start early***.
You should also spend time thinking about the correctness of your queries. While we are happy to help you think through how to approach the query, we will not confirm the correctness of your answers

Finally, be aware that some of the queries are difficult and may take lots of brainstorming to complete!

Part 1: Querying an AWS Oracle Database

(Questions 1-7. 45 points)

IMDb Database. The database you will use consists of a portion of the Internet Movie Database. It has been uploaded to Amazon's AWS/RDS and conforms to the schema (where keys are underlined) outlined in **Appendix A: The IMDb Database** of this document.

Please refer to Section 5 of the ***DataGrip Handout*** for instructions on connecting to the database, if you have not already done so.

Question 1. (5 points, M1)

Print the name of all actors (of all genders) whose name starts with 'S' (case-sensitive) and ends with a lower-case vowel ('aeiou', case-sensitive). Eliminate duplicates, and print the result in **alphabetical** order.

Hint: See resources on like, substr, and in.

Schema: (name)

Question 2. (5 points, M1)

For each movie whose runtime is longer than 100 and released after 2017 (2017 not included), print all female actors (gender=1 in movie_cast).

Schema: (name)

Question 3. (6 points, M2)

For all directors who have directed at least one movie with more than a million ratings, print the director's name and the number of movies they directed that have more than a million ratings. You may assume that director names are unique.

Schema: (name, num_movies)

Question 4. (5 points, M2)

Print the total number of crew members who have not participated in any movies since 1950 (1950 included).

Schema: (num)

Question 5. (8 points, M2)

For all movies that have a greater rating than at least one movie directed by 'Christopher Nolan' and were released in the same year as “Inception”, print their title and rating. Do not hard code the release year of “Inception”.

Hint: As a warmup, try writing a query to find all the ratings of movies directed by Christopher Nolan.

Schema: (title, rating)

Question 6. (8 points, M2)

For each movie genre, print the genre name and a list of all the movies of this genre with more than 3 costume designers (these are people with job ‘Costume Design’ in crew_in). The list should separate movie titles by a semicolon and space (‘; ’), and should be in alphabetical order. Order the query result by genre name (column name ‘genre’), and name the list of movie titles ‘movies’.

Note: For those genres with no movies satisfying the condition, do not include the genre in the result.

As an example, there are 3 history movies each with more than 3 costume designers: ‘Bang Rajan’, ‘Napoleon’, ‘Samson and Delilah’, so the following tuple will appear in the result:

genre	movies
History	Bang Rajan; Napoleon; Samson and Delilah

To create the list of movie titles, use the Oracle built-in function LISTAGG. Note that Oracle 19c allows the DISTINCT keyword to be included directly in the LISTAGG function call, e.g. LISTAGG(DISTINCT …).

Hint: As a warm-up using just the material taught in class, you can write a query which returns all the movies with more than 3 costume designers.

Schema: (genre, movies)

Question 7. (8 points, M2)

We define the experience level of a director as 'Novice' if they have directed fewer than 5 movies, and 'Experienced' otherwise. Print out the average rating of movies directed by directors from each experience level. Round the average ratings to two decimal points, and print the results in a descending order based on the average rating.

Hint: See details on the round function and case.

Schema: (exp_level, avg_rating)

Part 2: Setting up an AWS MySQL Database

The goal of this part is to get experience with MySQL and AWS RDS (Amazon Web Services Relational Database Service), one of the most popular open-source databases and cloud-based relational DB services respectively.

The SQL syntax used in MySQL is similar to that used in Oracle in many ways, although there are some notable differences (for example, case sensitivity for object names and or keywords). A good reference for MySQL syntax can be found here.

OpenFlights Database. You will be creating three tables corresponding to the OpenFlights database that conform to the schema (where keys are underlined) outlined in **Appendix B: The OpenFlights Database** of this document.

If you have already not done so, install the MySQL driver for DataGrip as instructed in Section 4a of the ***DataGrip Handout***.

You should follow the ***AWS RDS Handout*** to create an AWS account and launch a MySQL database instance on RDS before proceeding. Follow this handout completely before proceeding.

Use the DataGrip console to interact with your AWS RDS instance for the following parts of the assignment.

Note: Please fill into db_config (in the template file) the connection details of the database you set up above. Please leave the instance running while the autograder is still running, but you may turn it off afterwards (i.e. when the autograder successfully runs all your queries) to avoid exceeding free tier limits.

Question 8. (15 points, M1)

Open an editor console corresponding to your MySQL instance and run the command **CREATE DATABASE flights**, and then **USE flights** to create and switch to a database named 'flights'. Refer to **Appendix B: The OpenFlights Database**, reading very carefully through it (including any footnotes or remarks). Reading through the specifications a few times is a good starting point.

Write SQL CREATE TABLE statements to create the database schema outlined in the appendix, enforcing the primary and foreign key constraints indicated. Fill in these CREATE TABLE statements as **answer8a** - **answer8c** of the template file, assuming that they will be executed in that order. Note that not all orders of creating tables work, and your response will receive a deduction for an invalid order!

Note: While designing the schema for these tables (especially when designating keys or other constraints), follow the relational model as closely as possible including primary/foreign keys and other constraints (note the requirements for the code_share attribute, which can assume only two values: 'Y' or '' (empty string)).

Part 3: Querying an AWS MySQL Database

(Questions 9-12. 35 points)

Download and extract the openFlights Dataset zip file, which can be found on Canvas. Once extracted, you will see three files, *Airports.csv*, *Airlines.csv* and *Routes.csv* corresponding to the three tables in the database. Follow Section 5 of the ***AWS RDS Handout*** to import these files into their respective tables.

You are encouraged to use CTEs (i.e. WITH … AS …) to solve the following problems.

Question 9. (8 points, M2)

Select the top 10 airlines whose country is "United States" by the number of routes by that airline between airports that are not in the United States or Canada. Print the airline name and number of such routes in decreasing order of number of routes.

Schema: (name, num_routes)

Question 10 (8 points, M2)

Consider only international routes, i.e. ones that go between two different countries. For each airline, count the number of distinct destinations of international routes (call this num_cities). Return all airlines with more than 150 num_cities, ordered by num_cities from largest to smallest, and secondarily by the airline's name in alphabetical order.

Schema: (airline_name, num_cities)

Question 11 (9 points, M2)

For each city in Mexico, return the number of other cities in Mexico that it cannot fly to by a direct flight (Routes). (By 'other cities', you should not count the city itself.) Order by the number of such cities from smallest to largest, and secondarily by the city's name in alphabetical order.

Schema: (city, num_cities)

Question 12. (10 points, M2)

For this question, you will restrict Routes to those using Airlines from the United Kingdom (call this UKRoutes). This reduces the size of the relation to make the query below run faster.

You have a friend in New Jersey who says Newark (iata code "EWR") is a "great airport". To prove it, he will pay for a free flight to take you from your home in Washington D.C. (city name "Washington") to Newark Airport. You want to know all cities that you can reach by either one total (direct) flight or two **total flights** (where the target_id of the first is the same as the source_id of the second).

For each city that is reachable from Washington (not including Newark or Washington) via one or two **total** flights, return the city name and the minimum number of **paid** flights (1 or 2), where all flights in the database are paid.

As an example, there is a direct flight from Washington to London, so the tuple ('London', 1) should be in the result. There is also a direct, connecting flight from this London airport to Brussels (but none from Newark or Washington) so the tuple ('Brussels', 2) should be in the result. There is a direct flight from Newark to Paris (though none from Washington), so the tuple ('Paris', 1) should be in the result, as we take the free flight from Washington to Newark and then the paid flight from Newark to Paris. However, even though there is a direct flight from the Paris airport to Naples, ('Naples', 2) is not in the result since it would entail three flights total, the first of which is free.

Hint: There are three cases that an eligible city can fall under while remaining at most two flights away: Washington -> City, Washington -> Newark -> City, Washington -> (some other airport) -> City. Consider how many paid flights are used for each.

Schema: (city, paid_flights)

Part 4: Generative AI Use

Question 13. (5 points)

Describe how, if at all, you used generative AI to assist with completing this homework. Recall that you cannot use generative AI to give you the answers to homework questions, but can use it to help understand the SQL constructs taught in class and/or introduced in this homework.

Submission

Please submit your responses using the template file (**homework1.py**). **Do not** rename this file. You may add comments to the file as required or change the order of answers if needed, but you **may not** rename any variables corresponding to answers (this includes the key parameters in *db_config* - fill in the required values only).

You must also fill in your PennKey and your 8-digit Penn ID at the top of the file.

As mentioned in Part 2, once the autograder has finished running and displays the results of your queries, you can terminate your RDS instance to avoid exceeding free tier limits. We will not be able to reimburse you for any costs incurred due to exceeding free-tier limits.

Autograder

You may submit your responses multiple times. Upon submission, the autograder will confirm if your queries were executed successfully; if a query is executed successfully it will also tell you the number of rows in the query result, and a (possibly condensed) preview of the result of executing your query. It will **not** tell you whether or not the query itself is correct.

Please note that queries that do not execute successfully will be heavily penalized.

Manual grading: Your responses are only partially autograded. Once the submission deadline is past, we will do a round of manual checking to adjust the autograded score to adjust (usually, add) credit based on the quality of the responses submitted.

Appendix A: The IMDb Database

	Attribute	Type
genre	name	varchar(255)

	Attribute	Type
movie	movie_id[2]	number(7)
	title	varchar(255)
	runtime	number(7)
	release_year	number(7)
	rating	float(2)
	num_ratings	number(7)

	Attribute	Type
crew	id	number(7)
	gender	number(7)
	name	varchar(255)

	Attribute	Type
movie_cast[3]	id	number(7)
	gender[4]	number(7)
	name	varchar(255)

	Attribute	Type
cast_in	movie_id	number(7)
	cast_id[5]	number(7)
	charac	varchar(255)

	Attribute	Type
crew_in	movie_id	number(7)
	crew_id[6]	number(7)
	job	varchar(255)

	Attribute	Type
movie_genre	movie_id	number(7)
	genre_name[7]	varchar(255)

	Attribute	Type
movie_keyword	kwd_name	varchar(255)
	movie_id	number(7)

* Filled colors in the 'Attribute' columns indicate references between relations
* Keys are underlined

Notes:

- In this part of the homework, "crew" refers to staff members (e.g. directors, electricians, music editors) whereas "cast" refers to actors and actresses in a movie.
- The relations **'crew'** and **'movie_cast'** contain information about the person who contributed as a staff or as an actor/actress, respectively.
- The relations **'crew_in'** and **'cast_in'** describe the participation between a staff and a movie, or between an actor/actress and a movie, respectively.

---

[2] **cast_in**(movie_id), **crew_in**(movie_id), **movie_genre**(movie_id), and **movie_keyword**(movie_id) - all reference **movie**(movie_id)
[3] Think of this as a relation of actors
[4] 0 = Other/Not specified, 1 = Female, 2 = Male
[5] cast_id references **movie_cast**(id)
[6] crew_id references **crew**(id)
[7] genre_name references **genre**(name)

- Recall the definition of primary key. In the relation **‘movie’**, the same movie name can correspond to multiple release years. We always identify ‘a movie’ by its movie_id, instead of the movie name, as suggested by the primary key constraint.
- You can ignore the table **keyword**, as it is empty.

Appendix B: The OpenFlights Database

	Attribute	Type
Airlines	id	int
	name	varchar(255)
	alias	varchar(255)
	iata	char(2)
	icao	char(3)
	callsign	varchar(255)
	country	varchar(255)
	active	char(1)

	Attribute	Type
Routes	airline_iata	char(3)
	airline_id[8]	int
	src_iata_icao	char(4)
	source_id[9]	int
	target_iata_icao	char(4)
	target_id[10]	int
	code_share	char(1)
	equipment	char(20)

	Attribute	Type
Airports	id	int
	name	varchar(255)
	city	varchar(255)
	country	varchar(255)
	iata	char(3)
	icao	char(4)
	lat	decimal(8,6)
	lon	decimal(9,6)
	alt	int
	timezone	decimal(3,1)
	dst	char(1)
	tz	varchar(255)

* Filled colors in the 'Attribute' columns indicate references between relations
* Keys are underlined
* Relation names are case-sensitive in MySQL

NOTE: The attribute codeshare in the table Routes can only assume two values: 'Y' or '' (empty string) corresponding to whether or not the flight is a codeshare, that is, not operated by Airline but another carrier. Read up on the 'CHECK' syntax and include it in your DDL statement(s).

[8] airline_id is a foreign key referencing **Airlines**(id)
[9] source_id is a foreign key referencing **Airports**(id)
[10] target_id is a foreign key referencing **Airports**(id)