CS 448 Project 1

SQL Queries - 100 points

Spring 2020

This is not a group project. Each student should prepare and submit her/his project independently.

This project is due on 02/14/2020 at 11:59PM on Blackboard. Note: There will be a 10% penalty for each late calendar day. After five calendar days, the Project will not be accepted.

1 Introduction & Setup

In this project, you are asked to answer 10 questions with the given dataset using MySQL queries. To create a MySQL account, please follow the instructions on this link.

Hint:

1. Please use tables.sql and data.sql to create tables and populate data. To do so, log in to your MySQL account and use command "Source tables.sql;" and "source data.sql;"
2. You can test your result against the expect_out.txt to check the correctness.
3. When grading, we will use a different dataset. You are welcome to add/modify the given data to test your queries for corner cases.

2 Schema

You are given a database for an online movie ticket selling system. The tables are defined as below:
Movies(Id, Name, *Gen_Id*, Description, Length, Release)

- Id: Movie Id number. Each movie has a unique id number.
- Name: Name of the movie.
- Gen_Id: Genre code. This refers to the ID number in Genres table.
- Description: A short description of the movie.

- Length: Length of the movie (in minutes).
- Release: Releasing year.

Theatres(Id, Name, Address, Zip)

- Id: Theatre Id number. Each theatre has a unique Id number.
- Name: Theatre name.
- Address: The address of the theatre (not including the zip-code).
- Zip: Zip-code of the theatre.

Genres(Id, Title)

- Id: Genre code. Each genre has a unique code.
- Title: Genre's title (e.g., Action, Comedy, ...)

Movie_Cast(Act_Id, *Mid*, Act_Name, Role)

- Act_Id: Actor/Actress Id number. Each actor/actress has a unique Id number.
- Mid: Movie Id number. This refers to the Id in the Movies table.
- Act_Name: Name of the actor/actress.
- Role: The role the actor/actress played in this movie.

Ticket_Sell(Serial_Id, *Mid*, *Tid*, Start_Time, Price, Type)

- Serial_Id: The serial number of the selling transaction. Each transaction has a unique serial number.
- Mid: Movie Id number. This refers to the Id in the Movies table.
- Tid: Theatre Id number. This refers to the Id in the Theatres table.
- Start_Time: The date and start time of this movie.
- Price: Price of this movie ticket.
- Type: Type of the movie. (e.g., Standard/3D/IMAX)

Ratings(*MId*, Reviewer, Star, Rev_Date)

- Mid: Movie Id number. This refers to the Id in the Movies table.
- Reviewer: The Reviewer Id number. Each reviewer has a unique Id number.
- Star: The rating this reviewer gave. Could be any one decimal float within 0 5.
- Rev_Date: The day of the comment.

3 Queries

Write one SQL statement that corresponds to each of the following queries. Nested SQL queries are allowed.

1. List the names of *Action* movies that are released in *2016*.

```
SELECT NAME as name FROM movies WHERE GEN_ID = 1 AND RELEASE_YEAR = 2016
```

2. List the name, genre(title), description, and rating of the movie with the highest overall rating. Assume that there are no ties in the ratings.

```
SELECT m.NAME as Name,g.title as Genre, m.DESCRIPTION as Description, avg(r.STAR) as Ratings FROM movies m JOIN ratings r ON m.ID = r.MID JOIN genres g on m.GEN_ID = g.id GROUP BY r.mid order BY AVG(r.star) DESC limit 1 ;
```

3. List the name, genre(title), and description of the movie that is watched by most people. Print all if there exists a tie.

```
SELECT NAME as Name, title as Title, DESCRIPTION as Description FROM ticket_sell t join movies m on t.MID = m.ID join genres g on m.GEN_ID = g.ID GROUP BY t.mid HAVING COUNT(t.tid) = (SELECT COUNT(t.tid) FROM ticket_sell t join movies m on t.MID = m.ID join genres g on m.GEN_ID = g.ID GROUP BY t.mid ORDER BY COUNT (t.tid) desc LIMIT 1);
```

4. For each movie theatre, print its name and number of movies shown at this theatre. Print the theatre with the most movies and the lowest alphabetical order first.

```
select th.name as Name, COUNT(t.TID) as Num_Of_Movies from ticket_sell t join movies m on t.mid = m.ID join theatres th on th.ID = t.TID group by t.TID ORDER BY COUNT(t.TID)  desc, Name
```

5. List the name of the actor/actress that makes the highest box office value. Print all if there exists a tie.

```
select mc.NAME from movie_cast mc join ticket_sell ts on ts.MID = mc.MID GROUP BY mc.ACT_ID having sum(ts.PRICE) = (select sum(ts.PRICE) from movie_cast mc join ticket_sell ts on ts.MID = mc.MID GROUP BY mc.ACT_ID ORDER BY sum(ts.PRICE) desc LIMIT 1);
```

6. Amy watched both *The Shawshank Redemption* and *Dangal* on the same day. Which theatre and in which day did she go? List all the possibilities.

```
SELECT th.NAME as Name, DATE(t1.START_TIME) as Day from ticket_sell t1 join ticket_sell t2 on t1.TID = t2.TID join theatres th on th.ID = t1.TID where t1.mid != t2.mid and DATE(t1.START_TIME) = DATE(t2.START_TIME) and t1.mid = 1 and t2.mid = 4;
```

7. *Lake Shore* theatre has moved to *27 Broad Ave, 47943*. Please update this information.

```
UPDATE theatres SET ADDRESS = "27 Broad Ave", ZIP = 47943 where `NAME` = "Lake Shore"
```

8. For some reason, all movies released in 2016 are actually 20 minutes shorter than the length recorded in the database. Please decrease the length of those movies by 20 minutes.

```
UPDATE movies SET LENGTH = LENGTH -20 where RELEASE_YEAR = 2016
```

9. Delete all the negative reviews (i.e., the ones that have lower than 2.0 in the Attribute Star) made in 2019.

```
DELETE FROM ratings WHERE STAR < 2.0 AND YEAR(REV_DATE) = 2019
```

10. Since some new movies will be released in early February, movie theatres decided to remove some of the movies showing currently. Find the movie(s) with overall rating lower than 3.5 and remove all that qualify.

```
delete from movies where ID in (select id from (select id FROM movies m join ratings r on r.MID = m.ID GROUP BY r.MID HAVING AVG(r.STAR) < 3.5)t);
```

4 Submission

Write all your queries in **answers.sql** and submit to **Blackboard**.
Comment each query in the following format:
/*Query 1*/
SELECT ...

/*Query 2*/
...
If there is any other comment that you wish to include, please add it at the top of your script.