# CSci 4707 Programming Assignment 2

**(Total Points: 350)**

The database schema as well as commands to load the data have been provided in **schema.sql**. There are two tables: **Ratings** with 1 million records, and **Movies** with 10,000 records. You are expected to turn in one file (PDF, txt or doc) with **SQL queries** along with answers to the questions below.

NOTE: If you are using **MySQL workbench**, then the default option limits the output to be only the first 1000 rows. To remove that restriction, please follow the steps mentioned here
There is also a 30 second timeout on queries. To remove that restriction, go to Edit → Preferences → SQL Editor → DBMS connection read time out (in seconds). Change this value to 0 instead of the default 30.

This assignment tests the following:

I. **Impact of Indexes (Points: 100)**

This task involves running queries and noting the query execution time. Commands to note execution time have been provided in **schema.sql**. You do not need to turn in SQL outputs if the output is anything besides count, instead write the number of records returned by each query.
Write the SQL query for the following and then run these queries on the Ratings table using the different scenarios given below

A. Print number of distinct movie ids with rating = 1
B. Print number of distinct movie ids with ratings in the range 1 to 3
C. Print number of users who have rated more than 100 movies
D. Find average movie rating per user

**Scenario 1**: Run the above queries (A-D) on the given table as is (without defining any key or index). Note execution time and number of records returned by the query.
**Scenario 2**: Create an index on **rating**. Now run the queries again and note execution time.
**Scenario 3**: Define (userid, movieid) as the primary key. Run the queries again and note execution time.

Based on the execution times, which field would you recommend creating an index on for the ratings table? Which of the two indexes in scenario 2 and 3 is clustered and unclustered?

II. **Table joins and query optimization (Points: 100)**

This task involves joining the two tables and then performing some select operations. Write the SQL query, run it and report query execution time and number of records returned:

A. Find average movie **popularity** per user. If you have defined primary keys on the tables, drop them before running this query. Note query execution time and report number of records returned.
    a. Now create primary key (id) on the **movies** table and run query A again. Don't forget to note the execution time.
    b. Now create primary key (userid, movieid) on the **ratings** table and run query A. Report execution time.
    c. What is the performance improvement obtained after creating the two indexes as compared to running the query without indexes defined.
B. Find number of movies which have higher **popularity** but lower **vote_average** than others (This query is only on movies table)
C. Find number of movies which have equal **popularity** but unequal **vote_average** than others (This query is only on movies table)
D. Find number of ratings of movies with **popularity** > 5 and **vote_average** > 5
    a. First try this query with a join condition on the 2 tables (join is on the common field in the two tables)
    b. Next, run the query without the join condition. Note the execution time as well as number of records returned.
    c. Are number of records same in parts a and b? Why?

## III. Access Permissions (Points: 80)

Perform the following operations first:

1. Login as root user
2. Create 2 users (User1, User2) Identified by Password.
3. Grant select permission on movies table with grant option to User1.
4. Logout root.

A. Show the output of the following by logging in as User1 and User2 in the database.
    a. SHOW GRANTS FOR `user1`@`localhost`
    b. SELECT count(*) FROM Movies;
    c. SELECT count(*) FROM Ratings;
B. Login as User1 and Pass SELECT permission on Movies to User2. Show the outputs of 3 commands listed in A by logging in as User2.
C. Login as root and Revoke access of movies table from User1. Show outputs of 3 commands listed in A by logging in as User1 and User2.

Submit the grant and revoke commands you used to give and revoke access from the user.

## IV. Transactions (Points: 70)

This task involves looking at the lifecycle of a transaction. Perform the following operations:

A. Find the number of entries in the ratings table that have rating = 5. Report the SQL query and output

B. Execute the following command to prevent the default autocommit operation:

```
SET autocommit = 0
```

C. Start **Transaction**; update the ratings of movies with rating 5 to rating 5.5; **Rollback** Transaction; Now find the number of entries in the ratings table that have rating = 5.5. Report the output.

D. Start **Transaction**; update the ratings of movies with rating 5 to rating 5.5; **Commit** Transaction; Now find the number of entries in the ratings table that have rating = 5.5. Report the output.

E. Explain the difference in output observed in parts C and D