ISE3235
Database Design
Assignment-03

For this assignment you will modify the give **Assignment-03_skeleton.py** file by adding SQL statements to it. Please adhere to the instructions written in the file and do not modify the sections it has asked not to.

For submission, rename the .py file with your ID after you are done editing. Do not modify import statements. Everything you need to complete this assignment has been imported for you. Do not use other libraries for this assignment.

Tasks: This is a description of all the functions you are working on. Please note that you are only required to write the SQL queries except for (a) which only asks for your ID. Use round function from sql to round upto 2 decimal points if there are any floating point values in the resultant table.

a. **InhaID**: update this method to send your id in return
b. **createTable_1**: Create a table named **movies** with columns having the indicated data types:
   ```
   movies
      1. id (integer)
      2. title (text)
      3. score (real)
   ```
c. **createTable_2**: Create a table named **movie_cast** with columns having the indicated data types:
   ```
   movie_cast
      1. movie_id (integer)
      2. cast_id (integer)
      3. cast_name (text)
      4. birthday (text)
      5. popularity (real)
   ```
d. **createIndex**: Create the following indexes. Indexes increase data retrieval speed; though the speed improvement may be negligible for this small database, it is significant for larger databases.
      1. movie_index for the id column in movies table
      2. cast_index for the cast_id column in movie_cast table
e. **calcProportion**: Find the proportion of movies with a score between 7 and 20 (both limits inclusive). The proportion should be calculated as a percentage and

should only be based on the total number of rows in the movies table. Format all decimals to two places using round.

f. **prolificActors**: Find the most prolific actors. List 5 cast members with the highest number of movie appearances that have a popularity > 10. Sort the results by the number of appearances in descending order, then by cast_name in alphabetical order. Below is an example of query outcome. Note that your result may be different as it is only an example. Make sure your query has this exact result outcome from running the query.

cast_name	appearance_count
Harrison Ford	2

g. **highScoringMovie**: Identify the highest scoring movies while favoring small cast size. List the 5 highest-scoring movies. In the case of a tie, prioritize movies with fewer cast members. Sort the intermediate result by score in descending order, then by number of cast members in ascending order, then by movie name in alphabetical order. Below is an example of query outcome. Note that your result may be different as it is only an example. Make sure your query has this exact result outcome from running the query. Format all decimals to two places using round.

movie_title	score	cast_count
Star Wars: Holiday Special	75.01	12
Games	58.49	33

h. **highScoringActor**: Find the top ten cast members who have the highest average movie scores. Sort the output by average score in descending order, then by cast_name in alphabetical order.
   - First exclude movies with score <25 in the average score calculation.
   - Next include only cast members who have appeared in three or more movies with score >= 25.

   Note that the above "score" references a score of one singular movie, while "average_score" is the calculated mean. Below is an example of query outcome. Note that your result may be different as it is only an example. Make sure your query has this exact result outcome from running the query. Format all decimals to two places using round.

cast_id	cast_name	average_score
8822	Julia Roberts	53.00