This HW is divided into 3 parts.  For Parts B & C, please download the schema and dataset provided. Create the database on Oracle server using SQLDeveloper, and then write the queries for each part. Your queries must work for arbitrary data, not just the provided data.

**Submission instructions:** For each Part submit a Part-X.sql file and a Part-X.pdf file, where X = A, B, C. Part-X.sql contains all your SQL queries. In Part-X.pdf copy/paste screenshots of each query and its answer from SQLDeveloper. In total you shall be submitting  6 files.

Screenshots instructions are [here](http://facsrv.cs.depaul.edu/~tmalik1/teaching/csc453/Resources/Screenshot.pdf). If you do not follow screenshot instructions , 40% of your HW points are going to be deducted. We need the history of commands run.

**Part A. (20 points)**

Lets say we have two relations of the form:

R(A,B) = {(0,1),(2,3),(0,1),(2,4),(3,4)}

S(B,C) = {(0,1),(2,4),(2,5),(3,4),(0,2),(3,4)}

Write SQL queries which do the following:

(i) [2pts] Finds A+B on R.

(ii) [2pts] Sorts R based on B,A.

(iii) [2pts] Turns R from a bag to a set.

(iv) [2pts] Computes the sum of B for each A value in R.

(v) [4pts] Joins R and S and computes the max value of C for each A.

(vi) [2pts] Finds tuples of R which match and unmatch with S.

(vii) [2pts] Finds tuples of S which match and unmatch with R.

(viii) [4pts] Finds tuples of R which match and unmatch with S but in which R.B is less than S.B.

Write the queries first, without creating the tables in SQLDeveloper. Then create the tables in the database and check how many queries were correctly written.

  Submit A.sql consisting of all SQL statements, and A.pdf consisting of screenshots of queries and results. Not following screenshot instructions will result in loss of points.

**Part B. (15 points)**

Dataset: Restaurants (Provided in D2L/Homeworks)

The schema is as follows:

|  |  |  |
| --- | --- | --- |
| Restaurants | restaurant | rID, name, address, cuisine |
| reviewer | vID, name |
| rating | vID, rID, stars, ratingdate |

**[SQL-SingleTable]**

1. [1pt] Find the name of all restaurants offering Indian cuisine
2. [2pt] Find restaurants which have a 'street' in their address.

**[SQL-Join]**

1. [2pt] Find restaurant names that received a rating of 4 or 5, sort them in increasing order of rating.
2. [2pt] Find the names of all restaurants that have no rating.
3. [2pt] Some reviewers didn't provide a date with their rating. Find the names of all reviewers who have ratings with a NULL value for the date.

      Note: Checking for NULL is performed with "<attribute\_name> IS NULL" condition and not  "<attribute\_name> = NULL"

      5. [3pt] For all cases where the same reviewer rated the same restaurant more than once and gave it a higher rating the second time, return the reviewer's name and the name of the restaurant.

      6. [3pt] Find reviewers with no ratings.

    Submit B.sql consisting of all SQL statements, and B.pdf consisting of screenshots of queries and results. Not following screenshot instructions will result in loss of points.

**Part C.** **(20 points)**

Dataset: Trips

The schema is as follows:

|  |  |  |
| --- | --- | --- |
| Trips | Trips | TID, TripState, TravelMode, Fare |
| ByCar | TID, Rental Company, Mileage |
| ByTrain | TID, Type, Coach, TrainSpeed, NumberofStops |
| ByPlane | TID, Airline, Class, LayoverTime |

The Trips table stores information about trips taken by various modes of travel and their respective fares. Each trip, owing to its mode has some relevant information specific to the mode. For instance, a car trip has the rental company and the mileage, whereas train trip is characterized by the trainspeed, type, coach, and number of stops. The TravelCompany is interested in answering the following queries.

**[SQL-SingleTable]**

1. [1pt] List car rental companies which have a mileage of at least 27 miles/gallon.
2. [1pt] List trip IDs taken on train costing strictly more than $150.
3. [1pt] Find trip IDs and their fare that are not taken in the US i.e., `Non-US` trips.
4. [2pt] Find the cheapest trip taken by each of air, rail, or car.

**[SQL-Join]**

     5. [1pt] Find the business class plane trip IDs that are greater than $1000.

     6. [2pt] Find any car trip more expensive than a trip taken on a train in the same state or outside the country.

     7. [2pt] List pairs of distinct car trips that have exactly the same value of mileage.

**Note a pair of distinct trips is of the format: (TID1, TID2).**

**This distinct pair is not the same as the pair (TID2, TID1)**

     8. [3pt] List pairs of distinct train trips that in which the speed of the first train trip is lower than the speed of the second train trip.

     9. [3pt] Find those pair of trips which occur in the same state and with the same mode of travel. List such pairs only once. In other words, given a pair (TID1,TID2) do NOT list (TID2,TID1).

    10. [4pt] Find a state in which trips have been taken by all three modes of transportation:  train, plane, and car. Note: Think 3-way self-joins with equality and non-equality join criteria.

Submit C.sql consisting of all SQL statements, and C.pdf consisting of screenshots of queries and results. Not following screenshot instructions will result in loss of points.