Assignment-2

Due Date: **Tuesday 1^{st} February 2022**

1 General Instructions

1. Please complete this assignment individually.
2. You will submit just 1 file: query.sql.
3. Use PostgreSQL 14 for your homework. See this link for instructions on how to download and install it on your OS. The .sql files are run automatically using the psql command using the `\i` option, so please ensure that there are no syntax errors in the file. If we are unable to run your file, you get an automatic reduction to 0 marks.To understand how to run many queries at once from text file, a dummy query file example.sql is available. To run example.sql in PostgreSQL, type the following command in the terminal:
   ```
   sudo -u postgres psql dbname
   \i /address/to/example.sql
   ```
   This command will run all the queries listed in example.sql at once.
4. The format of the file should be as follows. One line should identify the query number (note the two hyphens before and after the query number), followed by the actual, syntactically correct SQL query. Leave a blank line after each query.
   ```
   - -1- -
   SQL QUERY
   - -2- -
   SQL QUERY
   - -3- -
   SQL QUERY
   ```
5. Some of the queries below require an 'ORDER BY' clause. If you made an error in this clause, your answer may be evaluated as incorrect, giving you zero marks for that query.
6. The submission will be done on Moodle. No changes are allowed in attribute names or table names.
7. If unspecified, order in ascending order by column 1, then column 2 .. etc. In case of any doubts please ask on Piazza. The instructors ordering will be final and no queries will be entertained on the same.
8. There are no NULL values in the dataset, so you need not worry about that.
9. There is no data provided for these queries (except in the examples shown later). For testing, make your own dataset. The .sql file that you submit however, should contain only queries
10. If any query asks you to output top x rows and you are getting y rows, output min(x; y) rows.

2 Dataset 1

1. In the first part of the assignment, we'll work with the Railway dataset. We have shared the schema of the dataset here, but we won't share the actual dataset that will be used for evaluation.

2. The database have one table `train_info`:

`train_no`: integer	`train_name` : text	distance : integer (kms)
`source_station_name` : text	`departure_time` : time input	`day_of_departure` : text
`destination_station_name` : text	`arrival_time` : time input	`day_of_arrival` : text

3. Columns `arrival_time` (time when train reaches destination) and `departure_time` (time when train starts from source) have time input : ISO 8601 format. Read more about the format here.

4. Similarly `day_of_arrival` is day of the week when train reaches destination and `departure_time` is day of the week when train starts from source.

5. Keys:

   (a) **train no** is primary key for **train info** table

6. Some keywords used in queries:

   (a) **hop** : If two stations are 3 hops away that means a person would require four different trains to reach. For eg. Station `A -> B -> C -> D -> E`. Here station A and E are three hops away i.e. there are 3 intermediate stations. Station A and D are two hops away.

   (b) **feasible routes** : A feasible route is a route where departure time of connecting train 2 is greater than arrival time of train 1, where Monday is considered as day 1 and Sunday as day 7(so if there is connecting train from station X and if passenger reaches there at 12:30pm on Wednesday then he/she can take other train from 12:30pm on-wards for Wednesday till Sunday. Also the passenger should reach the destination within a week(before Monday 00:00:00).

   (c) **train operating on same day** : this implies that the day of departure and day of arrival for all connecting trains should be same.

7. You should use only these tables while writing solutions to the queries. You can create temporary views while handling any SQL query but you should include SQL queries for creating and deleting these temporary views at the starting and end of your SQL file respectively. Note - you don't have to define these tables in the submission file, these will already be present while evaluation.

2.1 Queries

1. Find all destinations which are **at-most** two hops away from the city *"KURLA"*, if the first train taken was `train_no` : **97131**.
   Note: one can return back to the source destination by taking two trains.
   Sort output by destination station name in ascending order (alphabetical).

2. Find all destinations which are **at-most** two hops away from the city *"KURLA"*, if the first train taken was `train_no` : **97131** and the connecting trains should operate on the same day of the week.
   Note: one can return back to the source destination by taking two trains.
   Sort output by destination station name in ascending order (alphabetical).

3. Find all destinations which are **at-most** two hops away from "*DADAR*". Also find total distance from *DADAR* if taken the route, for all possible trains, given that the connecting trains should operate on the same day of the week.
   Output : `destination_station_name`, distance, day **Sort output by destination station name in ascending order (alphabetical).**

4. Find all destination station names of **feasible** routes from city *DADAR* to destinations at most two hops away. **Sort output by destination station name in ascending order (alphabetical).**

5. Find **count** of the total number of combinations of trains possible between stations "*CST-MUMBAI*" and "*VASHI*", given that a passenger can take **at-most** 3 trains (2 hop distance).
   Output: Count

6. Find the **minimum** distance between all pairs of stations, if one can take the train a maximum of **6 times**.
   Sort output by destination station name in ascending order (alphabetical).

7. Find all pairs of stations which are **less than or equal to** 3 hops away, i.e. one has to change 4 or less trains to reach the destination.
   Sort output by source station name in ascending order (alphabetical).

8. Find all reachable destinations(irrespective of train arrival and departure times) from the source station "*SHIVAJINAGAR*" given that the connecting trains should operate on the same day of the week.
   Sort output by destination station name in ascending order (alphabetical).

9. Find **minimum** distance to all reachable destinations(irrespective of train arrival and departure times) from the source station "*LONAVLA*" given that the connecting trains should operate on the same day of the week.
   Output: 3 columns distance, destination and day.
   Sort output by distance in ascending order. If two destinations are at same distance then sort by destination station name in ascending order.

10. Find the longest(in terms of distance) circular chain for all stations, i.e. source and destination is the same city.
    Output: `source_station_name`, distance.
    Sort output by source station name in ascending order (alphabetical).

11. Find the stations from which all other stations are at max one hop distance (two trains).
    Output:`source_station_name`
    Sort output by source station name in ascending order (alphabetical).

3 Dataset 2

1. In this second part of the assignment, you'll work with a real **Football Matches dataset**. The schema of the same is described below, but we won't share the actual dataset that will be used for evaluation.

2. The database will include following five tables and you should use only these tables while writing solution of the queries. You can create temporary views while handling any SQL query but you should include SQL queries for creating and deleting these temporary views at the starting and end of your SQL file respectively. Note - you don't have to define these tables in the submission file, these will already be present while evaluation.

   (a) games

gameid : integer	leagueid : integer	hometeamid : integer	awayteamid : integer
year : integer	homegoals : integer	awaygoals : integer	

   (b) appearances

gameid : integer	playerid : integer	leagueid : integer	goals : integer
owngoals : integer	assists : integer	keypasses : integer	shots : integer

   (c) leagues

leagueid : integer	name : text

   (d) players

playerid : integer	name : text

   (e) teams

teamid : integer	name : text

3. Keys:

   (a) **gameid** is primary key for **games** table

   (b) **leagueid** is primary key for **leagues** table

   (c) **playerid** is primary key for **players** table

   (d) **teamid** is primary key for **teams** table

   (e) **hometeamid** is foreign key for **games** table in relation to **teams** table

   (f) **awayteamid** is foreign key for **games** table in relation to **teams** table

4. Match **m** is a football match between teams **A** and **B** if there exists a match **g** in **games** table such that **g.hometeamid = A, g.awayteamid = B** or **g.hometeamid = B, g.awayteamid = A**

5. Teams **hometeamid A and hometeamid B** are said to be having common teams played against, if [**hometeamid A, awayteamid C**] & [**hometeamid B, awayteamid C**] ϵ **games** table, where **A**, **B** and **C** all belongs to the **teams** table. Same goes for away teams also.

6. For the queries where length of a path between two teams is asked, **hometeamid A** is used as starting team and **awayteamid B** should be used as ending team. Such that **(hometeamid A, awayteamid** t_1**), (hometeamid** t_1**, awayteamid** t_2**), ... (hometeamid** t_n**, awayteamid B).**

7. In **games** table, **homegoals** is of **hometeamid** team and **awaygoals** is of **awayteamid** team.

8. In **games** table, **year** column is in **YYYY** format.

3.1 Queries

Football Matches dataset Football Matches database is an example of spatial network. Consider the collaboration network G formed by the teams and games tables. The nodes for this graph will be the games, and there will exist an edge from team A to B iff there is a match between A and B. There is an edge from team A to B if there exists a direct match between A and B.

Mathematically, G = *(V, E)* where:

- $V = \{$ team.teamid $\}$
- $E = \{ (u, v) : \exists$ g in games s.t. g.hometeamid = u and g.awayteamid = v $\}$

12. Find all the **hometeams** who played in common against hometeam **Arsenal** in ascending order alphabetically by teamname.
Output: teamnames

13. Find the first **hometeam** who played in common against **hometeam Arsenal**, with **max total goals**.
Output: teamnames, goals, year

14. Find all the teams with difference between homegoal and awaygoal greater than **3 (homegoal - awaygoal)** in the year **2015** in matches with teams common against **hometeam Leicester** in increasing order of goal difference.
Output: teamnames, goals

15. Find the name of players who scored the highest number of **goals** in the **awaymatches** with teams in common against hometeam **Valencia** (if more than 1 order them by decreasing order of goals).
Output: playernames, score

16. Find the name of players who **assisted** the most number of times in the home matches with teams in common against team **Everton** in alphabetical order of their names.
Output: playernames, assistscount

17. Find the name of players who shot the most number of **shots** in the away matches with teams in common against team **AC Milan** in **2016** in alphabetical order of their names.
Output: playernames, shotscount

18. Find **top 5** teams who has scored **0 awaygoals** in **2020** in the away matches with teams in common against team **AC Milan** in alphabetical order of teamname.
Output: teamname, year

19. Find name of players who scored top goals in the **away matches** with teams in common against the top scorer team of each league in **2019** in decending order of teamscore.
Output: leaguename, playernames, playertopscore, teamname, teamtopscore

20. The concepts of graphical analysis can also be applied to the dataset of such a kind where, Given two teams **Manchester United** as home team and **Manchester City** as away team, what is the **maximum length of path** calculated between two teams as nodes of a graph?
Output: count

21. Given two teams **Manchester United** as home team and **Manchester City** as away team, what is the **total number of paths** calculated between two teams as nodes of a graph?
Output: count

22. **In each League**, what is the **longest path between two teams** as nodes of a graph in alphabetical order of League names?
Output: leaguename, teamAname, teamBname, count