DSC 208R: Data Management for Analytics

Programming Assignment 2: Part 2 - Queries in PostgreSQL Database

Introduction

As part of this assignment, you have to complete SQL Queries.

Now that you have completed the setup required to do this assignment, you are ready to continue on to Programming Assignment 2: Part 2 where you will answer a series of questions by writing a single SQL query to answer that question (you can use subqueries this time), and save your submission in individual files hw2-q1.sql, hw2-q2.sql, etc.

Assignment Submission Details

Submit a single zipped file with the format LastName_FirstName_PID.zip

Note that the time taken to run the query might vary depending on the computer of an individual.

SQL Queries (100 points)

Flight Data Questions (50 points total)

1. **(10 points)** For each origin city, find the destination city (or cities) with the longest direct flight. By direct flight, we mean a flight with no intermediate stops. Judge the longest flight in time, not distance.

   Name the output columns `origin_city` and `dest_city`, with time representing the flight time between them. Do not include duplicates of the same origin/destination city pair. Order the result by `origin_city` and then `dest_city` (ascending, i.e. alphabetically).

   [Output relation cardinality: 334 rows]

Running Time in Postgres:

- Using subquery in **FROM: ~1s**
- Using subquery in WITH: ~1s
- Using subquery in WHERE:
- Without subquery:

Running Time in Postgres with Indexing:

- Using subquery in FROM: ~1s
- Using subquery in WITH: ~1s
- Using subquery in WHERE: ~80s
- Without subquery:

2. **(10 points)** Find all origin cities that only serve flights shorter than 3 hours. You can assume that flights with NULL actual_time are not 3 hours or more.
   Name the output column `city` and sort them. List each city only once in the result.

   [Output relation cardinality: 109]

   Running Time in Postgres: <1s

   Running Time in Postgres with indexing: <1s

3. **(10 points)** For each origin city, find the percentage of departing flights shorter than 3 hours. For this question, treat flights with NULL actual_time values as longer than 3 hours.

   Name the output columns `origin_city` and `percentage`. Order by `percentage` value, ascending.

   i. Be careful to handle cities without any flights shorter than 3 hours. We will accept either 0 or NULL as the result for those cities.

   ii. Report percentages as percentages, not decimals (e.g., report 75.25 rather than 0.7525).

   [Output relation cardinality: 327]

   Running Time in Postgres: <1s

   Running Time in Postgres with indexing: <1s

4. **(10 points)** List all cities that cannot be reached from San Diego through a direct flight but can be reached with one stop (i.e., with any two flights that go through an

intermediate city). Do not include San Diego as one of these destinations (even though you could get back with two flights).

Name the output column `city`. Order the output in ascending order by `city`.

[Output relation cardinality: 258]

Running Time in Postgres: ~75s

Running Time in Postgres with indexing: ~40s

5. **(5 points)** List the names of carriers that operate flights from San Diego to San Francisco, CA. Return each carrier's name only once. Use a nested query to answer this question.

   Name the output column `carrier`. Order the output ascending by `carrier`.

   [Output relation cardinality: 4]

   Running Time in Postgres:

   Running Time in Postgres with indexing: <1s

6. **(5 points)** Express the same query as above, but do so without using a nested query.

   Again, name the output column `carrier` and order ascending.

   [Output Cardinality: 4]

Running Time in Postgres:

Running Time in Postgres with indexing: <1s

Chinook Data (ORDER BY all the answers in ascending order: 50 points total)

7. **(10 points)** List all tracks that were never purchased by any customers. Return distinct track names. We only need the name attribute.

   [Output relation cardinality: 1458]

8. **(10 points)** List the names of all songs that do not belong to the 90s Music playlist. Return distinct track names (only the name attribute).

   [Output relation cardinality: 1943]

9. **(10 points)** List the artists who did not record any tracks of the Blues genre. Return distinct artist names (only the name attribute).

   [Output relation cardinality: 270]

10. **(10 points)** List all the playlists that do not have any track in the Rock or Blues genres. Return distinct playlist name (only the name attribute)

    [Output relation cardinality: 10]

11. **(10 points)** Find the list of artists that record at least in **3** different genres. Again, return only the artist nme.

    [Output relation cardinality: 7]