Homework 3 | Advanced SQL

Objectives: To practice advanced SQL. To get familiar with commercial database management systems (SQL Server) and using a database management system in the cloud (SQL Azure).

Resources

- SQL Server on Windows Azure through SQL Azure
    - SQL Server Management Studio has been installed on the CSE lab and VDI machines if you would like to use that instead of Azure's web interface.

Before You Begin

Introduction

This homework is a continuation of HW1 and HW2, but with the following differences

- The queries are more challenging
- You will use a commercial database system (i.e., no more SQLite)
    - SQLite simply cannot execute these queries in any reasonable amount of time; hence, we will use SQL Server, which has one of the most advanced query optimizers
- You will use the Microsoft Azure cloud

Prework

Setting Up an Azure SQL Database

Before any queries can be run, you must first set up your Azure SQL database; please follow the instructions here. While you are following this document, you must be **logged in to your @uw** account and also **logged out of your @cs** account. You will have a **very bad time** with that document if you are not logged into your @uw, or if you are logged into both your @cs and @uw.

✨✨✨✨✨✨✨✨✨✨✨✨✨✨✨✨✨✨✨✨✨✨✨✨✨✨✨✨✨✨✨

✨✨ **NOTE: The above document will take some time to complete, so start early!** ✨✨

✨✨✨✨✨✨✨✨✨✨✨✨✨✨✨✨✨✨✨✨✨✨✨✨✨✨✨✨✨✨✨

Problem Set

These instructions are a subset of HW2's. Please review them carefully:

- The predicates in your queries should correspond to the exact English descriptions in the question.
    - For example, if a question asks you to find flights by "Alaska Airlines Inc.", your query should check for that specific string instead of its matching `cid` (eg, do not use `cid='AS'`).
    - For example, if a question asks you to find flights on "Tuesday", your query should check for that specific string instead of its corresponding `did` (eg, do not use `did=2`).
- Return the output columns exactly as indicated. Do not change the output column names, the output order, or return more/fewer columns.
- Unless otherwise noted, include canceled flights in your output.
- When asked to output specific durations, report them in minutes instead of doing the minute-to-hour conversion.

1. (15 points) (Output relation cardinality: **334 rows**)
   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 using duration, not distance. (15 points)

   Name the output columns `origin_city`, `dest_city`, and `time` (the flight duration). Do not include duplicates of (origin city, destination city) pairs. Order the result by `origin_city` and then `dest_city` (ascending, i.e. alphabetically).

2. (15 points) (Output relation cardinality: **109 rows**)
   Find all origin cities that only serve flights shorter than 3 hours. You should not include canceled flights in your determination.

   Name the output column **`city`** and sort them in ascending order alphabetically. List each city only once in the result.

3. (15 points) (Output relation cardinality: **327 rows**)
   For each origin city, find the percentage of departed flights whose duration is shorter than 3 hours; canceled flights do not count as having departed. Be careful to handle cities which do not have any flights shorter than 3 hours; you should return 0 as the result for these cities, not NULL (which is shown as a blank cell in Azure).

Name the output columns `origin_city` and `percentage`. Order by percentage value, then city, ascending. Report percentages as percentages, not decimals (e.g., report 75.2534 rather than 0.752534). Do not round the percentages.

4. (15 points) (Output relation cardinality: **256 rows**)
   List all cities that can be reached from Seattle using exactly one stop. In other words, the flight itinerary should use an intermediate city, but cannot be reached through a direct flight. **Do not include Seattle as one of these destinations** (even though you could get back with two flights).

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

5. (15 points) (Output relation cardinality: **3 or 4 rows**, depending on what you consider to be the set of all cities)
   List all cities that can be reached from Seattle, but which require **two** intermediate stops **or more**. Warning: this query might take a while to execute; we will learn about how to speed this up in lecture. You can assume all cities to be the collection of all `origin_city` or all `dest_city`.

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

6. (7 points) (Output relation cardinality: **4 rows**)
   List the names of carriers that operate flights from Seattle to San Francisco, CA. Return each carrier's name only once, and use a nested query to answer this question. (7 points)
   Name the output column `carrier`. Order the output ascending by carrier.

7. (8 points) (Output relation cardinality: **4 rows**)
   Express the same query as above, but do so without using a subquery.

   As before, name the output column `carrier`. Order the output ascending by carrier.

8. (10 points)
   The DBMS that we use in this assignment, SQLServer, is running somewhere in one of Microsoft's data centers. Comment on your experience using a cloud-hosted DBMS. Would you recommend it to a fellow student for tinkering/experimentation? What about recommending it to future co-workers for their project? If your answer differed, why?

   Save your answer in a file called hw3-d.txt in the submission directory.

Submission Instructions

You should submit the questions on Gradescope under HW3. You can resubmit however many times until the due date, which will save your progress without submitting all answers. If you don't have access to Gradescope or are having issues with submitting, contact course staff as soon as possible.

For each question in the problem set, write a **single SQL query or paragraph**. Each answer should be in a separate file, named `hw3-q#.sql` (eg, `hw3-q1.sql`, `hw3-q2.sql`, etc) or `hw3-d.txt` for the reflection question. Each .sql file should include:

- A comment indicating the number of rows your query returns
- A comment indicating how long the query took
- A comment containing the first 20 rows of the result
    - If the result has fewer than 20 rows, output all of them

We want you to succeed in this class, so we strongly recommend that you verify your filenames before your final submission; our autograders hardcode the expected filenames, and your submission will receive no points if they do not match.