# Homework 3 | Advanced SQL and Azure

CSE D 514 - Data Management for Data Science

**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).

**Assignment tools:** SQL Azure on Microsoft Azure.

**What to turn in:** hw3-d.txt and hw3-q1.sql, hw3-q2.sql, etc (see below).

## Assignment Details

This homework is a continuation of homework 2 but with three changes:

- The queries are more challenging
- You will get to 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.

In this homework, you will do three things:

1. Create a database in the SQL Server database management system running as a service on Windows Azure; import data from an Azure public blob.
2. Write and test the SQL queries below; keep in mind that the queries are quite challenging, both for you and for the database engine.
3. Reflect on using a database management system running in a public cloud.

### A. Setting up an Azure SQL Database [0 points]

In this assignment, we want you to learn how to use an Azure SQL database from scratch. Your first step will thus be to setup a database in the Azure service and importing your data. This step may seem tedious but it is crucially important. We want you to be able to continue using Azure after the class ends. For this, you need to know how to use the system starting from nothing.

**NOTE: These steps will take some time to complete, so start early!**

**Step 1: Create an Azure account and log in to Azure portal**

Click on the "Accept lab assignment" link in the email "Action required: Accept your lab assignment", log in using your washington.edu account and password.

Afterwards, you will be forwarded to the Azure portal.

**Step 2: Learn about Azure SQL Server**

Spend some time clicking around, reading documentation, watching tutorials, and generally familiarizing yourself with Azure and SQL Server.

**Step 3: Create a database**

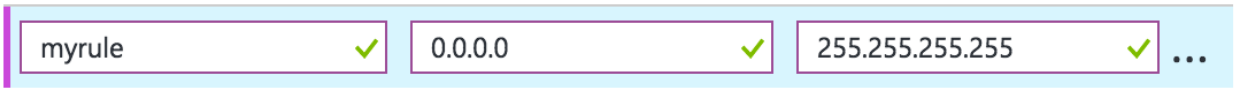
From the Azure portal, select "+ Create a resource", then select "Databases", then select "SQL Database". This will bring up a panel with configuration options for a new DB instance.

**Make sure to write down your database name, server name, and server password, you will need it later!**

Perform the following configuration:

- Create a new resource group (e.g. “csed514-20wi”).
- Choose a database name (e.g. “flights”).
- Create a new server by clicking on "Server". A second panel will appear to the right. Fill in the form as follows:
    - Choose a name for the server (e.g., "csed514-20wi-MyNetID"). Unlike your database name, the server name must be unique across the universe of Azure SQL databases.
    - Choose a server admin login and password. Remember this!
    - Set the location of your server to be somewhere in the US.
- Make sure "Want to use SQL elastic pool?" is set to "No".
- **This step is important because it controls how much money the database will cost.** Under "Compute + storage", click "Configure database". A new panel will open to the right. On this form,

    - Make sure you are on the “General Purpose” tab.
    - Make the Compute Tier “Serverless”.
    - Under Hardware Configuration, we recommend you slide the Max vCores to 8. Leave the Min vCores at the minimum. Note that the associated max memory available to your database is 24 GB in this case. This is probably significantly higher than what your laptop has available.
    - Leave the Auto-pause Delay at its default of 1 hour.
    - Leave the Data Max Size at its default of 32 GB.
- Select "Next: Networking"
    - Select "Connectivity method" : Public endpoint
    - Under Firewall rules, make sure "Allow azure services to access server" is set to "yes".
- Click "Next: Additional settings"
    - Make sure that "Choose a source" is set to "Blank". (Data Source is “None”.)
    - Make sure "Advanced Threat Protection" is set to "Not now".
- Click "Review + create"
    - Double-check that your settings are correct, and click "Create". This will likely take a few minutes to deploy.
- Once it's created, select your new database. You can find it on your dashboard or under the “SQL Databases” tab. Select the pushpin icon to "Pin to dashboard" so you can easily find it in the future.
- On the database page, click the “Set server firewall” tab. You need to change the firewall settings before you can upload data. The easiest option is to add a rule that allows connections from any client, which you can do as below and then click save.



**Step 4: Try out the database**

The simplest way to play with the database is using the built-in Query editor in the Azure portal. To launch this, go back to the dashboard, then click on the SQL database that you just set up. Enter the editor by clicking the "Query editor (preview)" on the side bar.

Enter the server username and password that you chose when you created your server in Step 3. Once you have done that, you can try entering SQL commands. Press the "Run" button to execute them.

Now you are ready to move on to the next part of the assignment!

## B. Ingesting Data (0 points)

Next, you will import all the data from HW2. Make sure that you execute your CREATE TABLE statements first so that the tables you will add tuples to already exist. Also, make sure that the types of the columns in the tables you created match the data. You are now in a Microsoft SQL Server environment so data types are now rigid. You can look them up here.

The data used in this assignment is the same as that in HW2, in the flights-small.csv, carriers.csv, months.csv, and weekdays.csv files. The easy, recommended way to ingest the data is by importing the data from a public blob storage container that we created for you. A "public blob" is that Microsoft calls its shareable storage hosted in the Azure cloud. Often it is easier to import data from within the same cloud (Azure), as opposed to data from outside the cloud (e.g., your local computer).

First make sure your CREATE TABLE statements have been run.

Next, run the following which defines the public blob endpoint:

```
CREATE EXTERNAL DATA SOURCE flightsblob
WITH (  TYPE = BLOB_STORAGE,
        LOCATION = 'https://csed514.blob.core.windows.net/flights'
);
```

Finally run each of these separately to import the data into each table:

```
bulk insert Carriers from 'carriers.csv'
with (ROWTERMINATOR = '0x0a',
DATA_SOURCE = 'flightsblob', FORMAT='CSV', CODEPAGE = 65001, --UTF-8 encoding
FIRSTROW=1,TABLOCK);
```

```
bulk insert Months from 'months.csv'
with (ROWTERMINATOR = '0x0a',
DATA_SOURCE = 'flightsblob', FORMAT='CSV', CODEPAGE = 65001, --UTF-8 encoding
FIRSTROW=1,TABLOCK);

bulk insert Weekdays from 'weekdays.csv'
with (ROWTERMINATOR = '0x0a',
DATA_SOURCE = 'flightsblob', FORMAT='CSV', CODEPAGE = 65001, --UTF-8 encoding
FIRSTROW=1,TABLOCK);

bulk insert Flights from 'flights-small.csv'
with (ROWTERMINATOR = '0x0a',
DATA_SOURCE = 'flightsblob', FORMAT='CSV', CODEPAGE = 65001, --UTF-8 encoding
FIRSTROW=1,TABLOCK);
```

Do some SELECT count(*) statements to check whether your imports were successful.

- Carriers has 1594 rows
- Months has 12 rows
- Weekdays has 8 rows
- Flights has 1148675 rows

Although you have extremely powerful hardware at your disposal, consider adding the following CREATE INDEX statements which organize your data in a way that may speed up certain queries. We will learn about the details of these later in the class.

CREATE INDEX flights_origin_dest ON flights(origin_city, dest_city);

CREATE INDEX flights_dest_origin ON flights(dest_city, origin_city);

## C. SQL Queries (90 points):

For each question below, write a single SQL query to answer that question (you can use subqueries this time), and save your submission in individual files hw3-q1.sql, hw3-q2.sql, etc.

Now answer the following questions:

1. 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. (15 points)

   Name the output columns origin_city, dest_city, and time representing the 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]

2. 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. (15 points)

   Name the output column city and sort them. List each city only once in the result.

   [Output relation cardinality: 109]

3. 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. (15 points)

   Name the output columns origin_city and percentage. Order by percentage value, ascending. 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. Report percentages as percentages not decimals (e.g., report 75.25 rather than 0.7525).

   [Output relation cardinality: 327]

4. List all cities that cannot be reached from Seattle 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 Seattle as one of these destinations (even though you could get back with two flights). (15 points)

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

   [Output relation cardinality: 256]

5. List all cities that cannot be reached from Seattle through a direct flight nor with one stop (i.e., with any two flights that go through an intermediate city). Warning: this query might take a while to execute. We will learn about how to speed this up in lecture. (15 points)

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

   (You can assume all cities to be the collection of all origin_city OR all dest_city)

   [Output relation cardinality: 3 or 4, depending on what you consider to be the set of all cities]

6. List the names of carriers that operate flights from Seattle to San Francisco, CA. Return each carrier's name only once. Use a nested query to answer this question. (7 points)

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

[Output relation cardinality: 4]

7. Express the same query as above, but do so without using a nested query. Again, name the output column carrier and order ascending. (8 points)

The final question is a text answer in part D:

### D. Using a Cloud Service (10 points)

The DBMS that we use in this assignment is running somewhere in one of Microsoft's data centers. Comment on your experience using this DBMS cloud service. What do you think about the idea of offering a DBMS as a service? What are the pros and cons to this infrastructure? Write a small paragraph to explain your observations and opinions. Save your answer in a file called hw3-d.txt in the submission directory.

# Submission Instructions

**Commit** and **push** all your submission files in your repository.

Your files should have the same structure as below:

```
\-- csed514-xxxx-hw3-[your uw username]
        \-- README
        \-- hw3-d.txt     # your discussion for part D
        \-- hw3-q1.sql  # your solution to question 1
        \-- hw3-q2.sql  # your solution to question 2
        \-- hw3-q3.sql  # your solution to question 3
        … (and so on)
```