synch_02

UCB MIDS W205 Summer 2018 - Kevin Crook's agenda for Synchronous Session #2

As always, remember to update the course-content repo in your docker container in your droplet

In your droplet, startup a container with volume mapping:

```
docker run -it --rm -v /home/science/w205:/w205 midsw205/base:latest bash
```

Using the bash shell running inside the container, change to the course content directory:

```
cd ~/course-content
```

Update your course-content repo:

```
git pull --all
```

Exit the docker container:

```
exit or control-D
```

Discuss the Query Project

Involves assignments 2, 3, 4, and 5

Warning: each assignment has the same header. The header applies to all of 2, 3, 4, and 5. Do only the detailed in the assignment part below the header.

SQL Tutorial

https://www.w3schools.com/sql/default.asp

Signup for Google Cloud account

https://cloud.google.com

Everyone should be signed up for the Google Cloud account before class as mentioned in slack.

Some tips: * Use the Chrome browser. Google makes Google Cloud. Google makes Chrome. Google tests Google Cloud using Chrome. Other browsers will hit things that won't work correctly. * You may have multiple Google accounts. Check and switch if necessary to the right account by clicking on the person icon in the upper right corner. * You must currently be on a project with billing enabled, otherwise something will work and others that require billing will not. * If you attempt to use the bike share dataset in BigQuery and it won't come up, it's probably one of the above.

We will be using Google BigQuery with the bay area bikeshare dataset

Google BigQuery Documentation: https://cloud.google.com/bigquery/

Google BigQuery public datasets documentation: https://cloud.google.com/bigquery/public-data/

Google BigQuery bay area bikeshare public dataset documentation: https://cloud.google.com/bigquery/public-data/bay-bike-share

Google BigQuery GUI based query page for the bay area bikeshare public dataset: https://bigquery.cloud.google.com/table/bigquery-public-data:san_francisco.bikeshare_trips

Legacy vs Standard SQL

We will be using Standard SQL instead of the Legacy SQL. It will be required for assignments and points will be deducted if it's not used. You **must** include the #standardSQL directive both in the query when running and also in assignments or points will be deducted, unless it's covered in other syntax.

Be careful if you are pulling code off the internet, as a lot of it may be specific to Legacy SQL.

Standard SQL supports more constructs and has more accurate results (is accuracy required for Big Data Analytics?)

Example of Legacy SQL (do **NOT** use)

```
SELECT *
FROM [bigquery-public-data:san_francisco.bikeshare_trips]
```

Same example in Standard SQL (use)

```
#standardSQL
SELECT *
FROM `bigquery-public-data.san_francisco.bikeshare_trips`
```

Standard SQL supports more constructs such as distinct():

```
#standardSQL
SELECT distinct(bikes_available)
FROM `bigquery-public-data.san_francisco.bikeshare_status`
```

Let's write and run some example queries against the bikeshare_status table

We will go to breakout and let each breakout group attempt to answer each question below. Scroll down to see the answers.

1. Select all column, all rows from the bikeshare_status table.
2. How many events are there? (hint: each row in the bikeshare_status table is considered an event)
3. How many stations are there? (hint: there is a station_id column in the bikeshare_status table, but remember each station_id may have more than 1 event. We only want to count each station once)
4. How long a time period do these data cover? (hint: there is a time column in the bikeshare_status table. Find the earliest time and the latest time for all times)
5. How many bikes does station 90 have (hint: total bikes should be docks_available + bikes_available. In this model, each bike has a dock and if the dock is empty, it means the bike is in use. Does the number of bikes at station 90 change over time?)

Queries which answer the previous questions

1. Select all columns, all rows from the bikeshare_status table:

```
#standardSQL
SELECT *
FROM `bigquery-public-data.san_francisco.bikeshare_status`
```

2. How many events are there?

```
#standardSQL
SELECT count(*)
FROM `bigquery-public-data.san_francisco.bikeshare_status`
```

3. How many stations are there?

```
#standardSQL
SELECT count(distinct station_id)
FROM `bigquery-public-data.san_francisco.bikeshare_status`
```

4. How long a time period do these data cover?

```
#standardSQL
SELECT min(time), max(time)
FROM `bigquery-public-data.san_francisco.bikeshare_status`
```

5. How many bikes does station 90 have (hint: total bikes should be docks_available + bikes_available)?

Does this query give us the answer?

```
#standardSQL
SELECT station_id,
(docks_available + bikes_available) as total_bikes
FROM `bigquery-public-data.san_francisco.bikeshare_status`
WHERE station_id = 90
```

No, it's time dependent. So, let's try this query:

```
#standardSQL
SELECT station_id, docks_available, bikes_available, time,
(docks_available + bikes_available) as total_bikes
FROM `bigquery-public-data.san_francisco.bikeshare_status`
WHERE station_id = 90
ORDER BY total_bikes
```

Create our own private dataset named bike_trip_data, create our own private table named total_bikes in our private dataset, run some queries against our private table

In the Google BigQuery user interface, on the left side panel, you will see the name of your project. To the right of the project name, you will see a dropdown arrow. Click on the dropdown arrow and choose "Create new dataset" and use this to create a new dataset named bike_trip_data.

Execute the following query. Once the results come back, towards the top right of the results panel, choose "Save as Table". Create a table named total_bikes in and put it in the dataset you just created.

```
#standardSQL
SELECT station_id, docks_available, bikes_available, time,
```

```
(docks_available + bikes_available) as total_bikes
FROM `bigquery-public-data.san_francisco.bikeshare_status`
```

Using the GUI examine the new table you created going through all of the tabs. Pay close attention to the naming and use it to create a similar queries to the ones below.

```
#standardSQL
SELECT distinct (station_id), total_bikes
FROM `xxxx.bike_trips_data.total_bikes`
```

```
#standardSQL
SELECT distinct station_id, total_bikes
FROM `xxxx.bike_trips_data.total_bikes`
WHERE station_id = 22
```

ssh - Secure Shell

scp - Secure Copy

Secure copy is a way to copy files from your laptop to your droplet and vice versa.

Windows

Windows users will probably want to use a utility such as WinSCP

https://winscp.net/eng/index.php

Mac

Mac users will use the command line utility scp. Here are a couple of basic examples (there are numerous variations on how to use scp):

Copy a file from your local directory to your droplet

```
scp my_file_local.txt science@ip_address:/home/science/my_file_host.txt
```

Copy a file in your droplet to your local directory

```
scp science@ip_address:/home/science/my_filehost.txt ~/my_file_local.txt
```

Using ssh to login without a password

In your droplet, change to the ~/.ssh directory:

```
cd ~/.ssh
```

Generate a pair of keys:

```
ssh-keygen -t rsa -b 2048
(hit return through the prompts)
```

This will create two files: * id_rsa - the private key file * id_rsa.pub - the public key file

Append the public key file to the end of the authorized_keys file:

```
cat id_rsa.pub >>authorized_keys
```

Windows

Windows users will use WinSCP to copy the private key file down to their laptop. Use the puttygen utility to translate it into putty key file format. When starting PuTTY, there is an option in the left panel: Connection => SSH => Auth click the Browse button to set the private key file.

Mac

Mac users will use the scp commands to copy the private key file down to theirlocal machine. It usually best to rename it (such as ucb_205.rsa) and place it in the ~/.ssh directory.

Change the mode of the file:

```
chmod 600 ~/.ssh/ucb_205.rsa
```

Specify the private key file on the command line when connecting:

```
ssh -i ~/.ssh/ucb_205.rsa science@ip_address
```