synch_03

UCB MIDS W205 Summer 2018 - Kevin Crook's agenda for Synchronous Session #3

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
```

Assignment 2 - answer the first 3 queries

Reminder - in class we have been querying bikeshare_status for our examples. For the first 3 queries, we will be using the bikeshare_trip table

1. What's the size of this dataset? (i.e., how many trips)

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

```
983648
```

2. What is the earliest start time and latest end time for a trip? (of all time, date and time, not a particular day)

```
#standardSQL
SELECT min(start_date)
FROM `bigquery-public-data.san_francisco.bikeshare_trips`
```

```
#standardSQL
SELECT max(end_date)
FROM `bigquery-public-data.san_francisco.bikeshare_trips`
```

```
2013-08-29 09:08:00 2016-08-31 23:48:00
```

Follow up - can you combine this into one query?

3. How many bikes are there?

```
#standardSQL
SELECT count(distinct bike_number)
FROM `bigquery-public-data.san_francisco.bikeshare_trips`
```

```
700
```

Using the explain shell website to help us understand what linux command line commands do

Explain shell uses the official man pages of linux and presents them in a graphical way that makes learning complicated commands much easier. Through this exercise, try each command in explain shell to help you understand.

https://explainshell.com/

Using a docker container in your droplet, we will download a dataset in csv (comma separated value) format, a dataset in json format, and manipulate them at the command line

Log into your droplet and start a docker container

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

The official instructions do this in the home directory, but I think it's a bit cleaner to make a subdirectory for this exercise.

```
mkdir synch_03
cd synch_03
```

Use the curl command line utility to download files from the internet to our current directory (synch_03)

```
curl -L -o annot_fpid.json https://goo.gl/rcickz
curl -L -o lp_data.csv https://goo.gl/rks6h3
```

Use the head command to see the first few lines of the file

```
head lp_data.csv
```

Use the tail command to see the last few lines of the file

```
tail lp_data.csv
```

Use the tail command to see only the first line of the csv file. What does the first line of a csv file tell us?

```
head -n1 lp_data.csv
```

Use a linux command line pipeline to cat (concatenate) the contents of a file to standard output and pipe standard output into standard input of a new process running the wc (word count) utility. Explain shell works well with this.

```
cat lp_data.csv | wc -l
```

Use a similar pipeline to sort and review in explain shell to see why output is sorted strangely

```
cat lp_data.csv | sort
```

Fix sorting using the -g and -n options. Look these up in explain shell.

```
cat lp_data.csv | sort -g
```

```
cat lp_data.csv | sort -n
```

Try a head on the json file. What happend and why?

```
head annot_fpid.json
```

(answer: json files can be 1 line for the whole file)

jq - a command line utility to format json

https://stedolan.github.io/jq/tutorial/

Use jq to format our json output

See what each of the following command do and how the syntax works. Consult the above documentation if necessary. Also use explain shell.

```
cat annot_fpid.json | jq .
```

```
cat annot_fpid.json | jq '.[][]'
```

```
cat annot_fpid.json | jq '.[][]' -r
```

```
cat annot_fpid.json | jq '.[][]' -r | sort
```

```
cat annot_fpid.json | jq '.[][]' -r | sort | uniq
```

```
cat annot_fpid.json | jq '.[][]' -r | sort | uniq -c
```

```
cat annot_fpid.json | jq '.[][]' -r | sort | uniq -c | sort -g
```

```
cat annot_fpid.json | jq '.[][]' -r | sort | uniq -c | sort -gr
```

```
cat annot_fpid.json | jq '.[][]' -r | sort | uniq -c | sort -gr | head -10
```

bq cli - Google BigQuery Command Line Utility

From a docker container running in our droplet, we will use the bq utility to connect externally to the Google Cloud and run BigQuery queries and pull the data down to our container.

Start in our home directory in our container

```
cd
```

Before we run bq, we have to set it up to communicate with Google Cloud. We will use the utility gcloud to do this. Follow the instructions given very carefully or it won't work. You will need to use your browser to call a URL and receive a token back as part of the security procedures. We should only have to do this once this semester as long as we mounted correctly.

```
gcloud init
```

Check and see the new configuration hidden directory structure:

```
ls -la
ls -la .config
```

The first time we run bq we will have to select a region and set a project. Be sure and pick the right project that you have been quering from in the GUI.

```
bq
```

Now that everything is setup, we can run some queries using bq to pull data from Google BigQuery.

```
bq query --use_legacy_sql=false '
SELECT count(*)
FROM `bigquery-public-data.san_francisco.bikeshare_status`'
```

Same command on 1 line to make copy and paste easier

```
bq query --use_legacy_sql=false 'SELECT count(*) FROM `bigquery-public-data.san_francisco.bikeshare_sta
```

```
107,501,619
```

How many stations?

```
bq query --use_legacy_sql=false '
SELECT count(distinct station_id)
FROM `bigquery-public-data.san_francisco.bikeshare_status`'
```

Same command on 1 line to make copy and paste easier

```
bq query --use_legacy_sql=false 'SELECT count(distinct station_id) FROM `bigquery-public-data.san_franci
```

How long a time period do these data cover?

```
bq query --use_legacy_sql=false '
SELECT min(time), max(time)
FROM `bigquery-public-data.san_francisco.bikeshare_status`'
```

Same command on 1 line to make copy and paste easier

```
bq query --use_legacy_sql=false 'SELECT min(time), max(time) FROM `bigquery-public-data.san_francisco.bi
```

```
2013-08-29 12:06:01.000 UTC 2016-08-31 23:58:59.000 UTC
```

sed and awk

sed (stream editor) and awk (pattern scanning and processing language are two utilities in linux command also used to format data

http://www.catonmat.net/blog/awk-one-liners-explained-part-one/ http://www.catonmat.net/blog/sed-one-liners-explained-part-one/

Change back into our synch_01 directory

```
cd synch_01
```

Example of a linux pipeline using awk. Check it out in explain shell to see what it does.

```
cat lp_data.csv | awk -F',' '{ print $2,$1 }' | sort
```

Example of a linux pipeline using sed. Check it out in explain shell to see what it does.

```
cat lp_data.csv  | awk -F',' '{ print $2,$1 }' | sed 's/"//' | sort | less
```

Question: how is sed related to vi? which mode?