CSCI 585 Summer 2019 HW4

Chin-Chou Ko
USC ID: 1511072586

Part 1: Google BigQuery (2pt)

Query #1
Using the “bigquery-public-data.usa_names.usa_1910_2013” table, query names, gender and total counts of the names which have the second letter “o” in the name (e.g. Rock) AND are from the gender ‘F’, limit the result to 10 rows.

```
SELECT name, count(*) AS total_counts
FROM `bigquery-public-data.usa_names.usa_1910_2013`
WHERE gender = 'F'
  AND SUBSTR(name, 2, 1) = 'o'
GROUP BY name
ORDER BY name DESC
LIMIT 10;
```

Results:

Row	name	total_counts
1	Zoya	116
2	Zowie	13
3	Zoua	11
4	Zorianna	2
5	Zoriah	5
6	Zoria	4
7	Zori	1
8	Zoraya	8
9	Zoraida	117
10	Zorah	1

Query #2
Using the “bigquery-public-data.usa_names.usa_1910_2013” table, find out the total counts of names starting with the letters “Al”.

```
SELECT name, COUNT(*) AS total_counts
FROM `bigquery-public-data.usa_names.usa_1910_2013`
WHERE STARTS_WITH(name, "Al") = true
GROUP BY name
ORDER BY total_counts DESC
LIMIT 5;
```

Results:

Row	name	total_counts
1	Allen	4806
2	Albert	4751
3	Alice	4662
4	Alex	4611
5	Alexander	4441

Part 2: DataLab and Notebooks (3pt)

1. Start with the BigQuery tutorial in notebook by go to datalab->docs->tutorials->BigQuery You need to run two notebooks: “BigQueryAPIs” and “BigQuery Commands.ipynb” as shown below. This step is just for you to get familiar with BigQuery in datalab. Simply take a screenshot of the result for the last cell you run for each notebook (For the BigQuery API notebook, take screenshot of the cell before delete resource) to prove that you have followed the tutorial(1pt).

Screenshots:

Google Cloud Datalab BigQuery APIs (unsaved changes)

Notebook | Add Code | Add Markdown | Delete | Move Up | Move Down | Run | Clear | Reset Session | Kernel: python2

```
# let's now try creating it from a list of records:
schema = [
  {'name': 'name', 'type': 'STRING'},
  {'name': 'value', 'type': 'INT64'},
  {'name': 'flag', 'type': 'BOOL', 'mode': 'NULLABLE'}
]
sample_schema = bq.Schema.from_data(schema)

sample_table = bq.Table("apisample.sample_table").create(schema = sample_schema, overwrite = True)
```

```
sample_table.schema
```

You can run the cell, below, to see the contents of the new dataset:

```
list(sample_dataset.tables())
```

Deleting Resources

```
# Clear out sample resources
sample_dataset.delete(delete_contents = True)
```

Looking Ahead

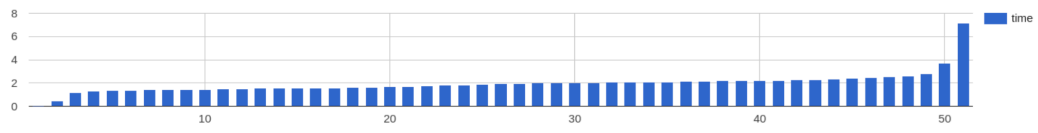
This notebook covered a small subset of the APIs. Subsequent notebooks cover additional capabilities, such as importing and exporting data into and from BigQuery tables.

Google Cloud Datalab BigQuery Commands (unsaved changes)

Notebook | Add Code | Add Markdown | Delete | Move Up | Move Down | Run | Clear | Reset Session | Kernel: python2

```
)
select row_number() over(order by time) as percentile, time from quantiles cross join unnest(quantiles.timearray) as time
order by percentile
```

```
%chart columns --data data --fields percentile,time
```



Looking Ahead

There are other commands, such as those that import (load) and export (extract) data or that handle tables and datasets.

Datalab allows queries to be constructed one step at a time to create **composite SQL queries** that use different constructs such as User Defined Functions and External Data Sources, in order to harness the full power of BigQuery SQL while managing authoring complexity.

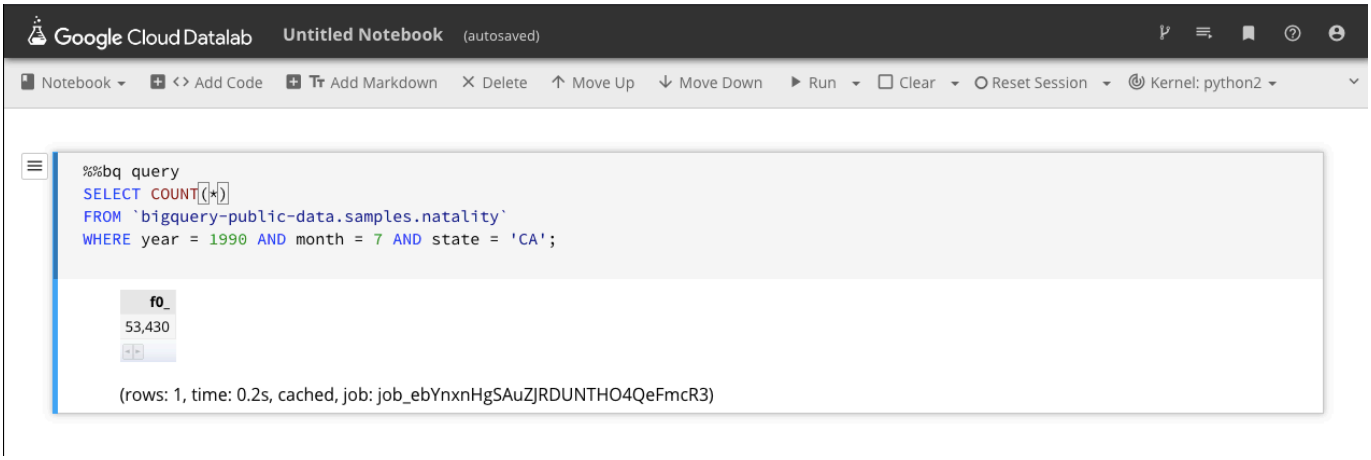
All of these BigQuery commands are implemented on top of **Python BigQuery APIs** (in the `google.datalab.bigquery` Python module). This implementation not only allows you to write arbitrary code and logic while working with BigQuery data, but also lets you integrate SQL and Python, and the Python data analysis libraries such as pandas and matplotlib, to perform sophisticated and custom data analysis and visualization tasks.

These topics are covered in other BigQuery tutorial notebooks that are included with Datalab.

2. Create a new notebook.
a) In the first cell, write a query using the natality dataset (you could find it in BigQuery public dataset and read the table description and schema by yourself, the table is "bigquery-public-data.samples.natality"), to count how many people has the birth year 1990 and birth month 7 in state 'CA'. Simply take a screenshot of the cell with the query and result (no need to include the text of the query) (1pt).

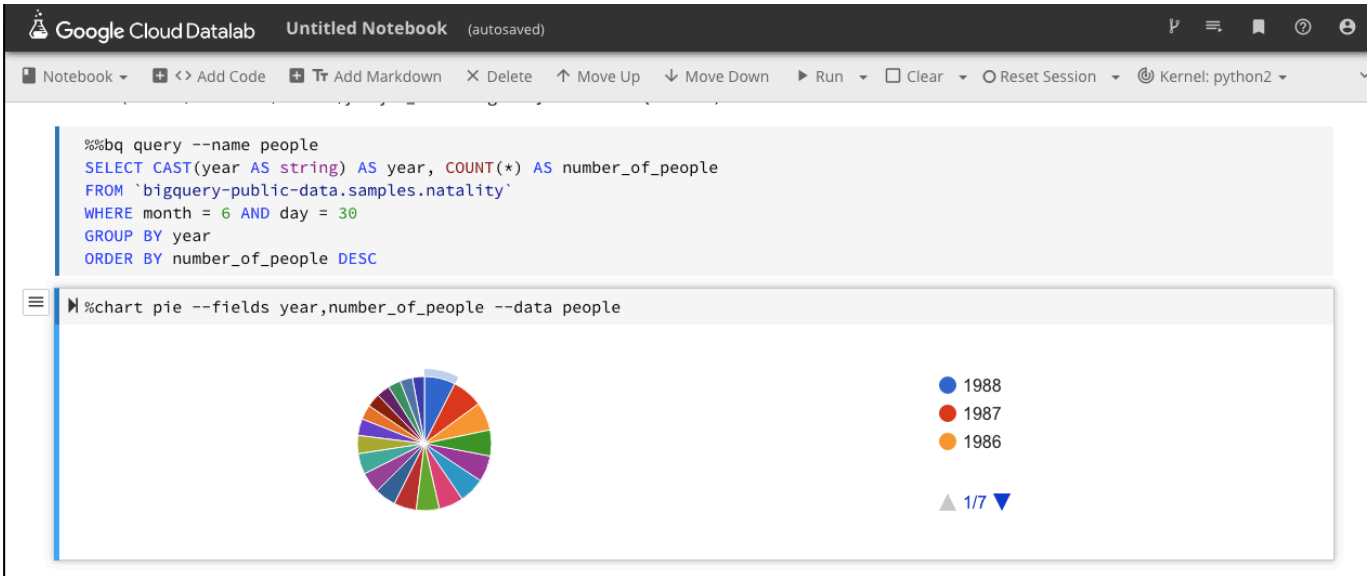
53430 people.

Screenshot:



b) In the second cell, write a query and find the number of people born on the June 30 in different years. There is no need to report the text result. You just need to visualize the data like what you did in tutorial but use a 'pie' graph. You do not have to show all the labels for different regions in the pie graph. Please use the method you learn from the "BigQuery Commands.ipynb". Simply take a screenshot of the cell with the query and graph (no need to include the text of the query) (1pt).

Screenshot:



Part 3: Big Public Data, Visualization and Interpretation (2pt)

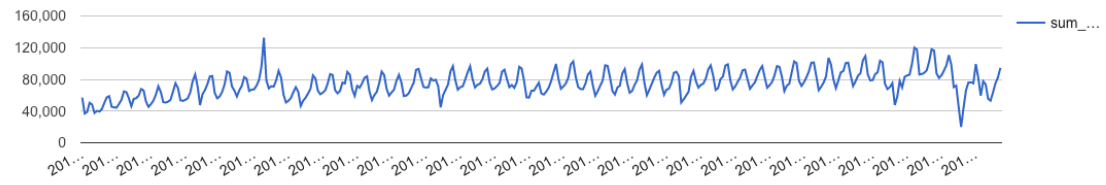
b) Visualization:

1).Use the method you learned from “BigQuery Commands.ipynb”. Write a query to retrieve the sum of trips for each single day for the year 2013, 2014, and 2015 and visualize in three figures. You need take three screenshots for the query and figure for the year 2013, 2014 and 2015 respectively. And answer the question above. (1pt)

The first big decrease is in Martin Luther King Jr. Day, the other one big decrease is in Christmas Eve. So we know during the long weekends, Chicago taxi trips are significantly less than normal weekdays.

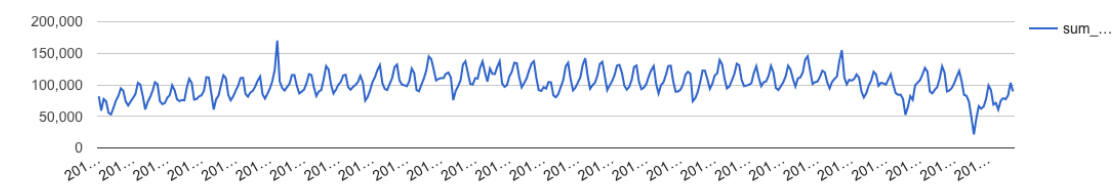
```
%%bq query --name trips
SELECT date, sum_of_trips
FROM (SELECT FORMAT_TIMESTAMP("%F", trip_start_timestamp) AS date, COUNT(*) AS sum_of_trips
  FROM `bigquery-public-data.chicago_taxi_trips.taxi_trips`
  GROUP BY date
  ORDER BY date)
WHERE date BETWEEN '2013-01-01' AND '2014-01-10'
ORDER BY date
```

```
%chart line --fields date,sum_of_trips --data trips
```



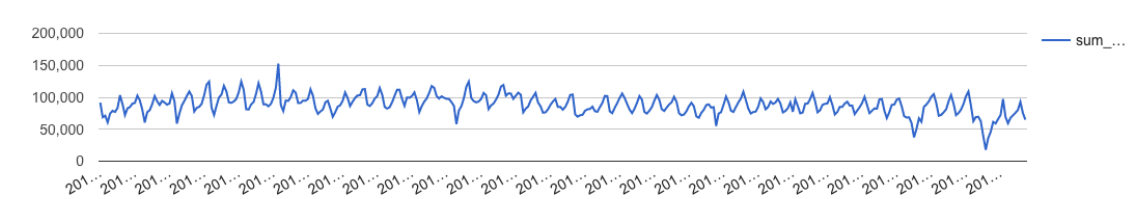

```
%%bq query --name trips
SELECT date, sum_of_trips
FROM (SELECT FORMAT_TIMESTAMP("%F", trip_start_timestamp) AS date, COUNT(*) AS sum_of_trips
  FROM `bigquery-public-data.chicago_taxi_trips.taxi_trips`
  GROUP BY date
  ORDER BY date)
WHERE date BETWEEN '2014-01-01' AND '2015-01-10'
ORDER BY date
```

```
%chart line --fields date,sum_of_trips --data trips
```




```
%%bq query --name trips
SELECT date, sum_of_trips
FROM (SELECT FORMAT_TIMESTAMP("%F", trip_start_timestamp) AS date, COUNT(*) AS sum_of_trips
  FROM `bigquery-public-data.chicago_taxi_trips.taxi_trips`
  GROUP BY date
  ORDER BY date)
WHERE date BETWEEN '2015-01-01' AND '2016-01-10'
ORDER BY date
```

```
%chart line --fields date,sum_of_trips --data trips
```



2) We want to investigate the pick-up and drop-off locations for expensive rides (trip_total between $300 and $400, include both) for future planning. You can use 2013 data (2013-01-01 till 2013-12-31). Feed in the coordinates (longitude, latitude) data into Google my maps (https://www.google.com/maps/d/). Differentiate between drop-off and pick-up locations using different marker colors. The final figure should be something like the following (the figure is just an example, you need take your own screenshot). Please note that the longitudes and latitudes values might be null, you just need to retrieve all the not null values and download as csv files, and the import to the map. You need to include the query (text format) for pickup/dropoff latitudes and longitudes and the screenshot of the marked map (there is no need to include all the locations, you could zoom in the map and take a screenshot that includes most locations) (1pt)

Query:

```
SELECT *
FROM `bigquery-public-data.chicago_taxi_trips.taxi_trips`
WHERE pickup_latitude IS NOT NULL
  AND dropoff_latitude IS NOT NULL
  AND trip_total BETWEEN 300 AND 400
  AND trip_start_timestamp BETWEEN "2013-01-01 00:00:00" AND "2013-12-31 23:59:59"
ORDER BY unique_key
```

Screenshot: