CS530 F23 Week 9 Lab
Prof. Feng

Amelia Miner
11/20/23

✅ indicates that a section was completed but did not request any screenshots or written answers.

TABLE OF CONTENTS:

Amelia Miner
11/20/23

I. Lab 9.1g: BigQuery, BigLake

1. BigQuery Lab #1 (Native Tables)

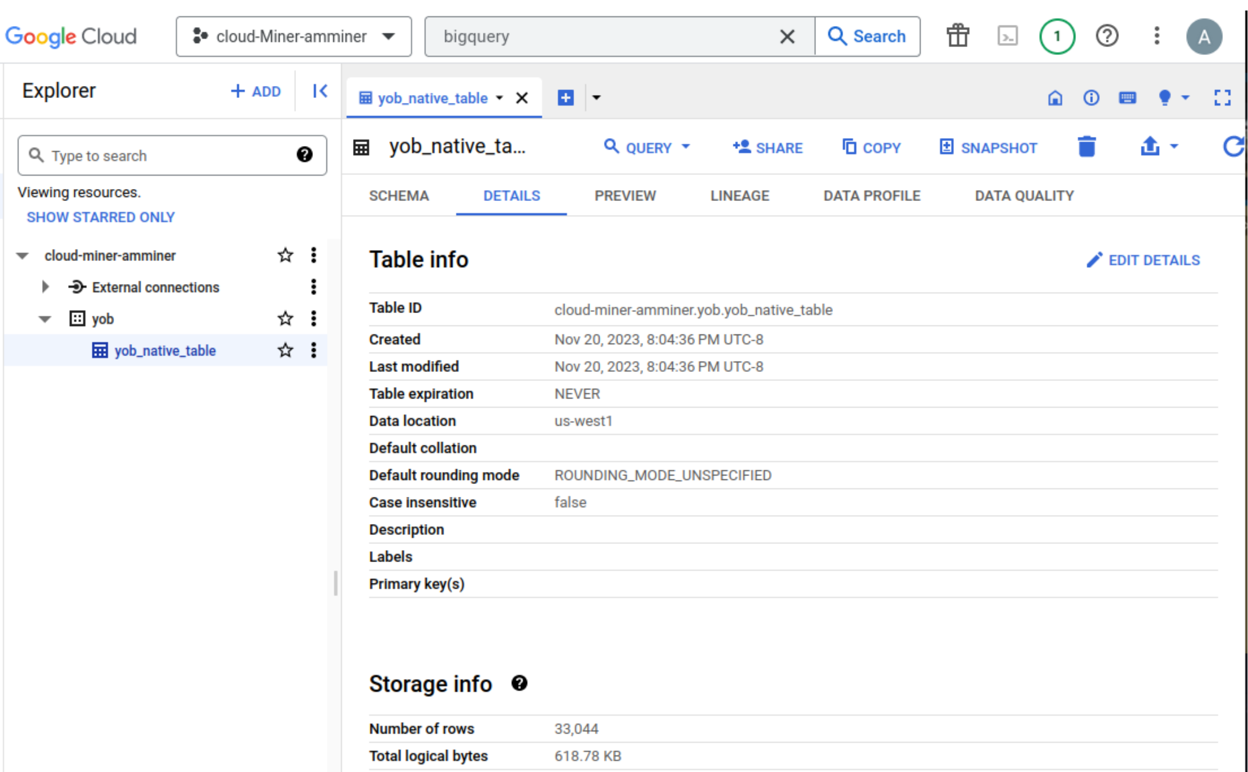
- Enable the BigQuery API and the "service" (Connection API).
  ✅

2. Examine Dataset

- In cloud shell, download this file containing the name, gender, and count of newborn babies in 2014 from the course site:
  https://thefengs.com/wuchang/courses/cs430/yob2014.csv
  Examine its format. Determine the data type for each field. Use wc to determine how many names it includes. Download the file to your local machine as well.
  name, gender, count; string, string, integer. 33044 names.
  ✅

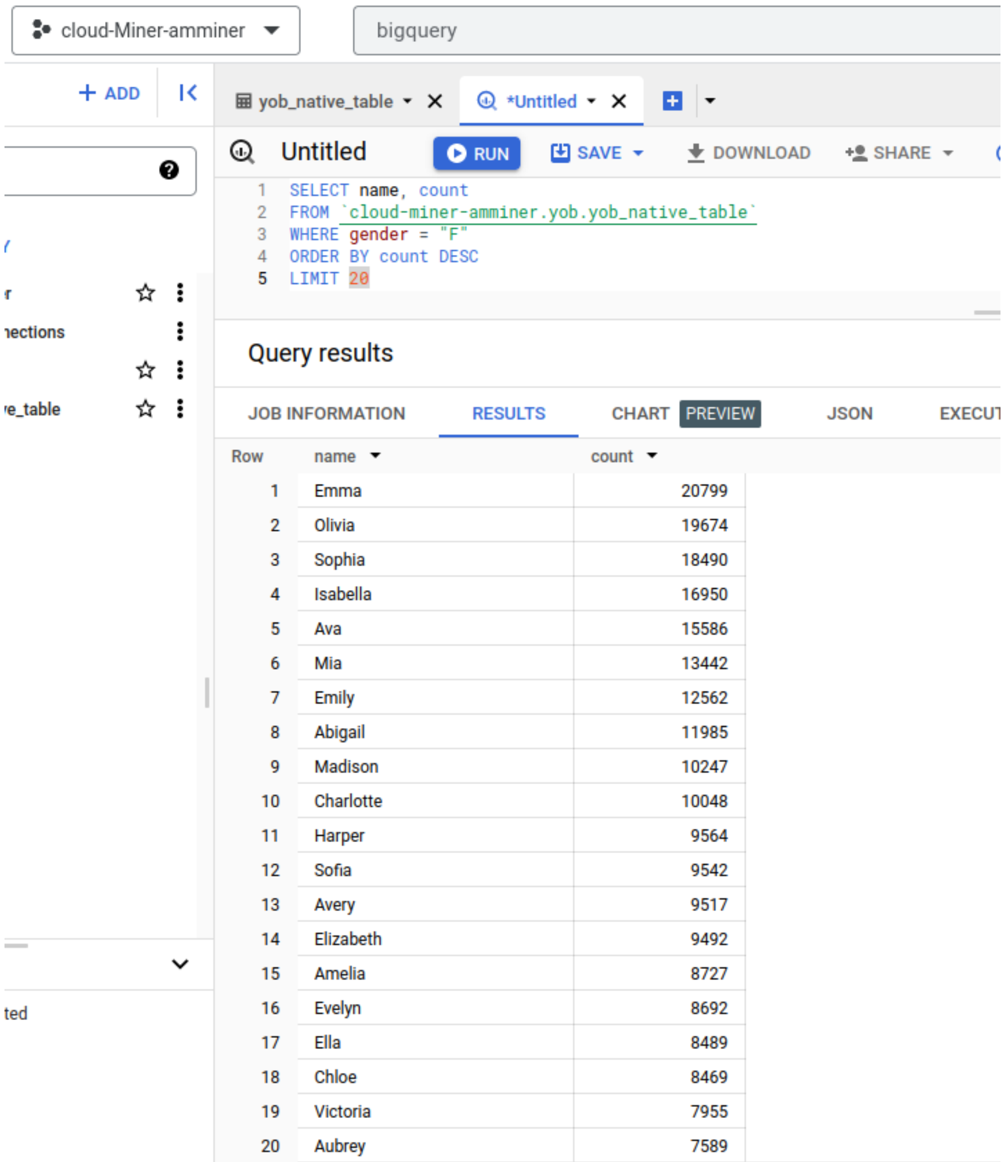
3. Create dataset

- Visit BigQuery in the web console and create a dataset. Give the dataset the ID yob and put it in us-west1. Create a table in the dataset named yob_native_table, uploading the CSV from step 2. Under schema, add fields that correspond to the columns in the file. Visit the table within the dataset and click preview, then details, and take a screenshot.



4. Query data

- BigQuery supports queries via
  - web console
    Select the table and compose a query that lists the 20 most popular female names in 2014. Table names must be escaped with back-ticks in the UI. Note that when given a valid query, the validator shows a green checkmark and the amount of data that the query will process, which is important to pay attention to so that you can eventually learn to optimize costs. Run the query.



Amelia Miner
11/20/23

- command line bq command
  In cloud shell, Use bq to get the 10 least popular boys names in 2014. Note that the tool requires the table name to be surrounded by square brackets, with periods separating the project name.
  The wording of this step is confusing, could use some attention/refinement.

```
amminer@cloudshell:~ (cloud-miner-amminer)$ bq query 'SELECT name, count FROM [cloud-miner-amminer:yob.yob_native_table] WHERE gender = "M" ORDER BY count ASC LIMIT 10'
+---------+-------+
|  name   | count |
+---------+-------+
| Aari    |     5 |
| Aaliyah |     5 |
| Aadian  |     5 |
| Aaroh   |     5 |
| Aarit   |     5 |
| Aadiv   |     5 |
| Aadhi   |     5 |
| Aarohan |     5 |
| Aariyan |     5 |
| Aamer   |     5 |
+---------+-------+
amminer@cloudshell:~ (cloud-miner-amminer)$
```

- interactive bq session
  Run a query to find the 10 most popular male names in 2014.

```
amminer@cloudshell:~ (cloud-miner-amminer)$ bq shell
Welcome to BigQuery! (Type help for more information.)
Cannot read termcap database;
using dumb terminal settings.
cloud-miner-amminer> SELECT name, count FROM [cloud-miner-amminer.yob.yob_native_table] WHERE gender = "M" ORDER BY count DESC LIMIT 10
+-----------+-------+
|   name    | count |
+-----------+-------+
| Noah      | 19144 |
| Liam      | 18342 |
| Mason     | 17092 |
| Jacob     | 16712 |
| William   | 16687 |
| Ethan     | 15619 |
| Michael   | 15323 |
| Alexander | 15293 |
| James     | 14301 |
| Daniel    | 13829 |
+-----------+-------+
cloud-miner-amminer>
```

- Query your own name. How popular was it in 2014?
  Pretty popular - it appeared in the GUI step above.

```
cloud-miner-amminer> SELECT name, gender, count FROM [cloud-miner-amminer.yob.yob_native_table] WHERE name = "Amelia"
+--------+--------+-------+
|  name  | gender | count |
+--------+--------+-------+
| Amelia | F      |  8727 |
| Amelia | M      |     8 |
+--------+--------+-------+
cloud-miner-amminer>
```

5. BigQuery Lab #2 (Lake Tables)

- In the above steps data is uploaded to BigQuery's storage layer or "warehouse". In the following steps we avoid duplicating the data between our local copy and the warehouse - with the data lake approach the data is kept in its original location and accessed per-query.
  Is that right? I need to learn more about big data, this goes over my head a little.
  ✅

6. Create external table

- Create a bucket in us-west1 and copy the csv file into it. In the bigquery GUI, select the dataset and create another table within it. Make the table external and specify the file stored in the bucket. Name the table yob_biglake_table and create a new connection for it as shown in the lab material's screenshots.
  ✅

7. -

- Specify the connection ID as biglake and add the schema manually as before.
  ✅

8. Configuring permissions

- The connection must have appropriate permissions to retrieve its data. Find its service account and grant the account access to view storage objects in IAM.
  ✅

9. query data

- Query for the 20 least popular female names in 2014.

```
cloud-miner-amminer> SELECT name, count FROM [cloud-miner-amminer.yob.yob_biglake_table] WHERE gender = "F" ORDER BY count ASC LIMIT 20
+-----------+-------+
|   name    | count |
+-----------+-------+
| Aaryah    |     5 |
| Aaniylah  |     5 |
| Aalimah   |     5 |
| Aayra     |     5 |
| Aarti     |     5 |
| Aamilah   |     5 |
| Aamyah    |     5 |
| Aashirya  |     5 |
| Aadrika   |     5 |
| Aashni    |     5 |
| Abagael   |     5 |
| Aarshi    |     5 |
| Aayusha   |     5 |
| Aania     |     5 |
| Aaiza     |     5 |
| Aabriella |     5 |
| Aavya     |     5 |
| Aarabella |     5 |
| Aarielle  |     5 |
| Aarion    |     5 |
+-----------+-------+
cloud-miner-amminer> 
```

10. clean up

- Delete the yob dataset and biglake connection in bigquery. Remove the storage bucket.
  ✅

II. Lab 9.2g: Jupyter Notebooks

1. Notebooks lab #1 (natality)

- In data science a backend like BigQuery is often paired with an interactive Python or R based Jupyter notebook. On GCP the Vertex AI service provides a managed notebook.

Enable the Vertex AI platform and its recommended APIs:

```
gcloud services enable notebooks.googleapis.com \
                       aiplatform.googleapis.com \
                       dataflow.googleapis.com \
                       storage.googleapis.com
```

Create a service account called cs430jupyter and add an IAM policy that associates the role of BigQuery User with this account.

```
gcloud iam service-accounts create cs430jupyter
gcloud projects add-iam-policy-binding $GOOGLE_CLOUD_PROJECT \
  --member
serviceAccount:cs430jupyter@${GOOGLE_CLOUD_PROJECT}.iam.gserviceaccount.com \
  --role roles/bigquery.user
```

Run the following command to bring up an instance, attaching the service account to it:

```
gcloud notebooks instances create bq-jupyter-instance \
  --vm-image-project=deeplearning-platform-release \
  --vm-image-family=tf2-2-2-cpu \
  --machine-type=e2-medium \
  --location=us-west1-b \

--service-account=cs430jupyter@${GOOGLE_CLOUD_PROJECT}.iam.gserviceaccount.com
```

✅

2. launch notebook

- Visit Vertex AI from the console. Find the instance in the workbench section and click “open jupyterlab”. Create a python3 notebook and leave it open.
  It’s confusing that this section refers to the notebook as an instance, but the notebook appears under the user-managed notebooks tab, not the instances tab.
  ✅

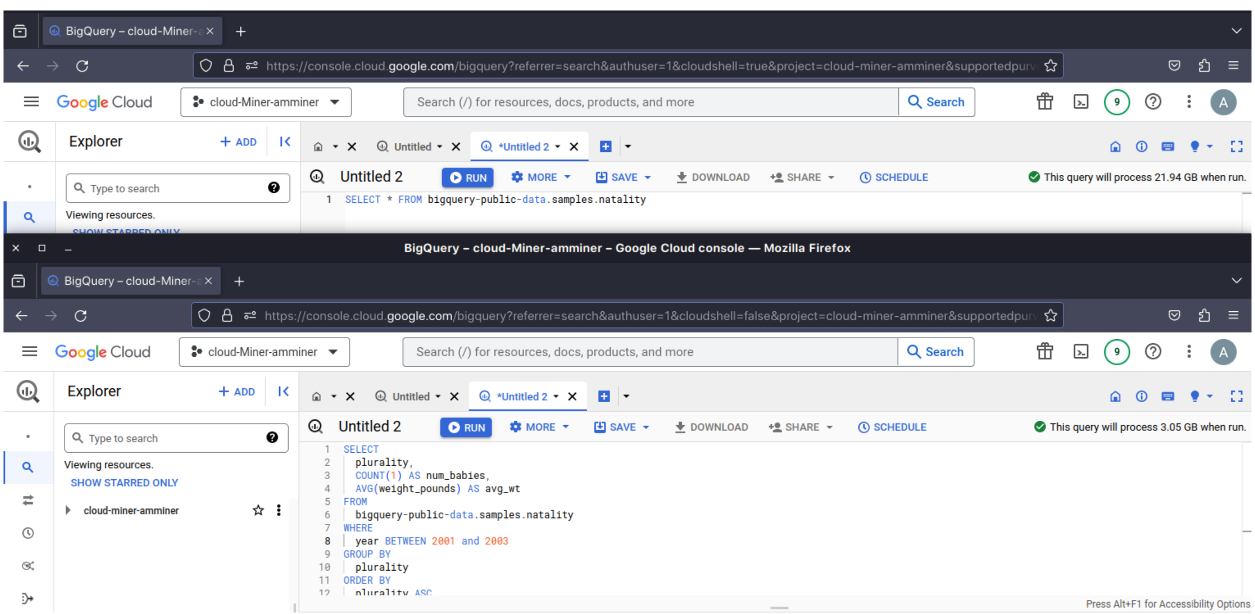
Amelia Miner
11/20/23

3. bigquery query

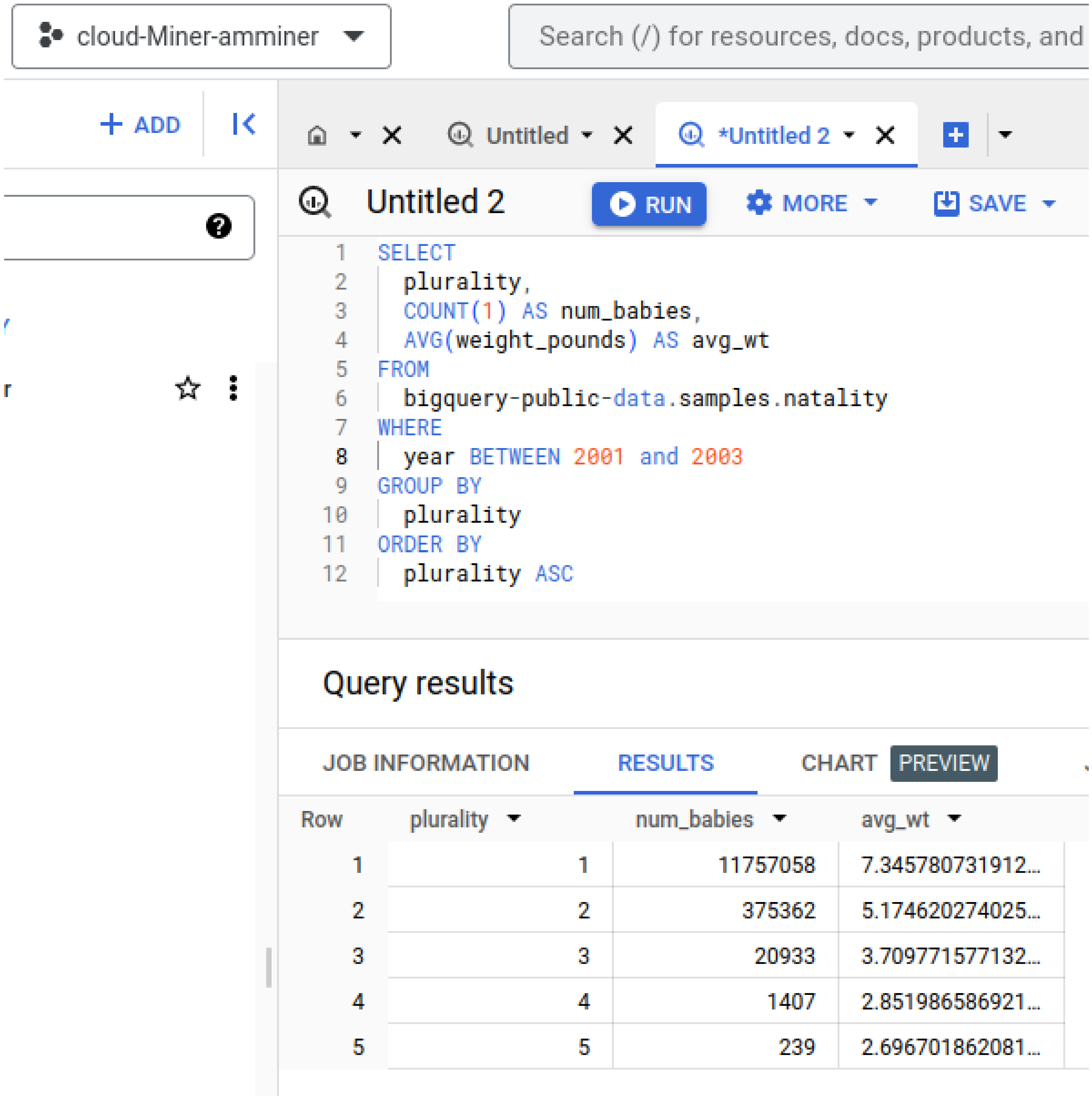
- Compose a query in the console that dumps the entire table, but don't run it.
  See the amount of data it will process - the size of the table.We'll run a query to obtain data on birthweight from publicly available data. Modify the query below to return the number of babies born, their average weight, and their plurality (single, twins, etc.) in ascending order between 2001 and 2003.

```
SELECT
  plurality,
  COUNT(1) AS num_babies,
  AVG(weight_pounds) AS avg_wt
FROM
  bigquery-public-data.samples.natality
WHERE
  year <FMI>
GROUP BY
  plurality
ORDER BY
  plurality ASC
```

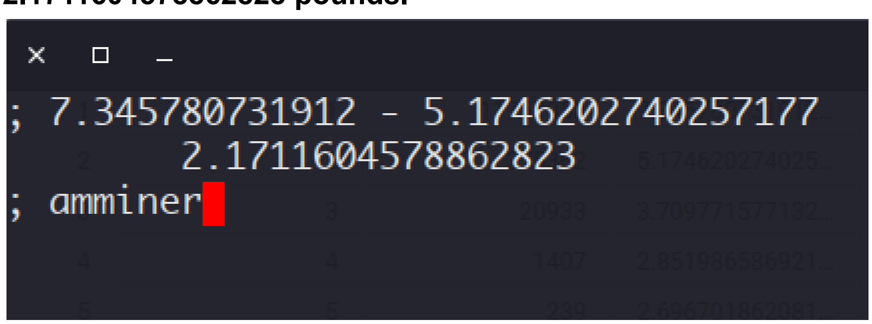
  Before running the query: How much less data does this query process?
  18.89 GB.



- Run the query. How many twins were born during this time range?
  375362.



- How much lighter on average are they compared to single babies?
  2.1711604578862823 pounds.



Amelia Miner
11/20/23

4. jupyter notebook query

- Go back to your notebook. We will now repeat the query in Python. Create the query_string variable in one cell, then run the following in another:

```
from google.cloud import bigquery
df = bigquery.Client().query(query_string).to_dataframe()
df.head(3)
```

Use built-in pandas functionality to create a scatter plot of the data:

```
df.plot(x='plurality', y='avg_wt', kind='scatter')
```

✅

5. exploring the dataset

- In a new cell run this:

```
query_string = """
SELECT
  weight_pounds,
  is_male,
  mother_age,
  plurality,
  gestation_weeks
FROM
  publicdata.samples.natality
WHERE year > 2000
"""
from google.cloud import bigquery
df = bigquery.Client().query(query_string + " LIMIT 100").to_dataframe()
df.head()
```

and in another new cell define this function:

```
def get_distinct_values(column_name):
  query_string = f"""
SELECT
  {column_name},
  COUNT(1) AS num_babies,
  AVG(weight_pounds) AS avg_wt
FROM
  publicdata.samples.natality
WHERE
  year > 2000
GROUP BY
  {column_name}
  """
  return bigquery.Client().query(query_string)\
         .to_dataframe().sort_values(column_name)
```

✅

Amelia Miner
11/20/23

6. run queries

- First, re-run the `plurality` query using the function, but generate a bar graph instead.
```
df = get_distinct_values('plurality')
df.plot(x='plurality', y='avg_wt', kind='bar')
```

  Then, run the query using gender:
```
df = get_distinct_values('is_male')
df.plot(x='is_male', y='avg_wt', kind='bar')
```

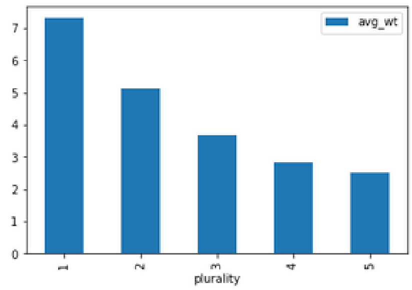
  Then, run the query using gestation time:
```
df = get_distinct_values('gestation_weeks')
df.plot(x='gestation_weeks', y='avg_wt', kind='bar')
```

  Finally, run the query using the mother's age:
```
df = get_distinct_values('mother_age')
df.plot(x='mother_age', y='avg_wt', kind='bar')
```

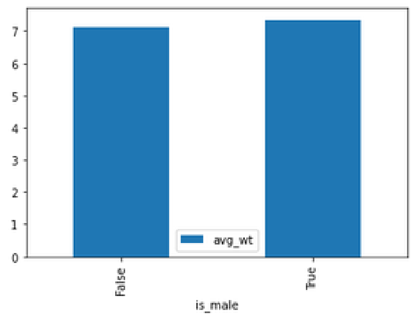
  In examining the plots, which two features are the strongest predictors for a newborn baby's weight?

gestation_weeks and plurality (third and first plots below, respectively)

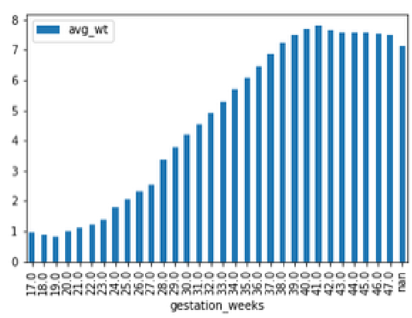
```
[6]: df = get_distinct_values('plurality')
     df.plot(x='plurality', y='avg_wt', kind='bar')  # amminer

[6]: <matplotlib.axes._subplots.AxesSubplot at 0x7f3bdc4c0fd0>
```



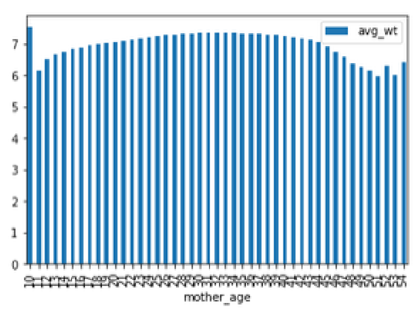

```
[7]: df = get_distinct_values('is_male')
     df.plot(x='is_male', y='avg_wt', kind='bar')

[7]: <matplotlib.axes._subplots.AxesSubplot at 0x7f3bdc497350>
```




```
[8]: df = get_distinct_values('gestation_weeks')
     df.plot(x='gestation_weeks', y='avg_wt', kind='bar')  # amminer

[8]: <matplotlib.axes._subplots.AxesSubplot at 0x7f3bcb7959d0>
```




```
[9]: df = get_distinct_values('mother_age')
     df.plot(x='mother_age', y='avg_wt', kind='bar')

[9]: <matplotlib.axes._subplots.AxesSubplot at 0x7f3bcb6b6510>
```



Amelia Miner
11/20/23

7. notebooks lab #2 (COVID-19 data)

- In the bigquery console explorer click ADD DATA and select bigquery-public-data. Select COVID-19. Scroll up to "Star" bigquery-public-data for ease of access.
  ✅

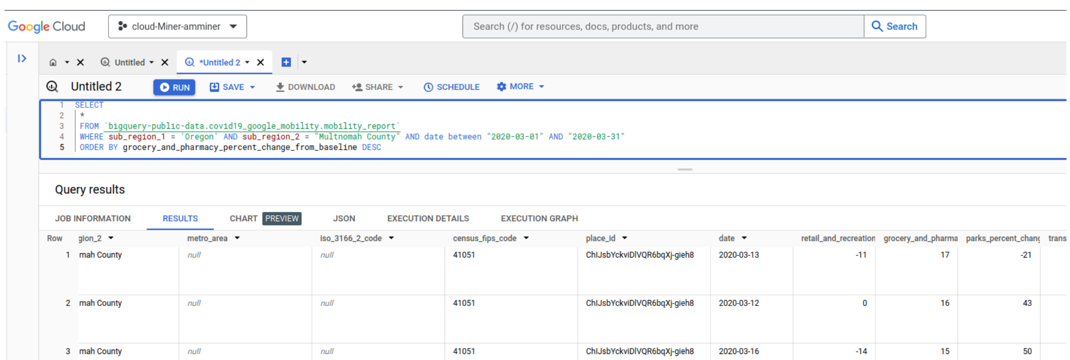
8. mobility

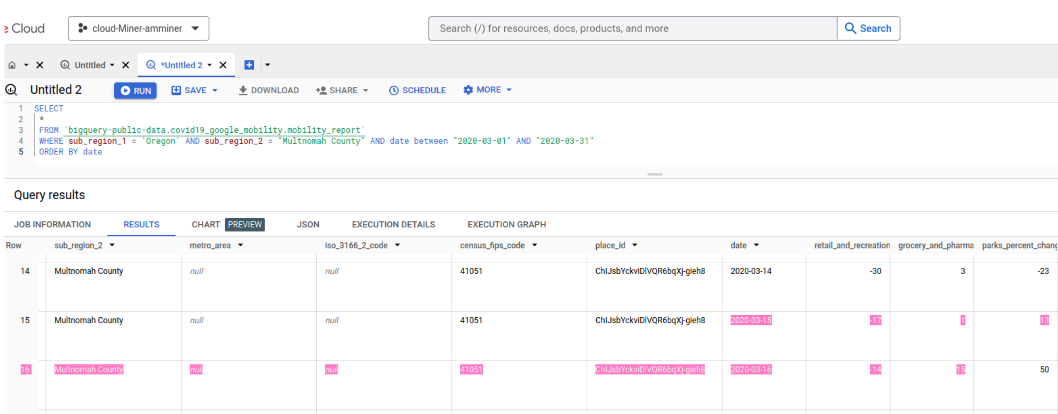
- Navigate to the covid19_google_mobility.mobility_report dataset and examine its columns. Run this query:

```
SELECT
 *
 FROM
`bigquery-public-data.covid19_google_mobility.mobility_report`
 WHERE sub_region_1 = 'Oregon' AND sub_region_2 = "Multnomah
County" AND date between "2020-03-01" AND "2020-03-31"
 ORDER BY date
```

- What day saw the largest spike in trips to grocery and pharmacy stores?

What does "largest spike" mean? Largest increase from one day to another? Highest percent change from baseline? 2020-03-13 had the highest percent change from baseline while the greatest day-to-day increase was 3/15-16.

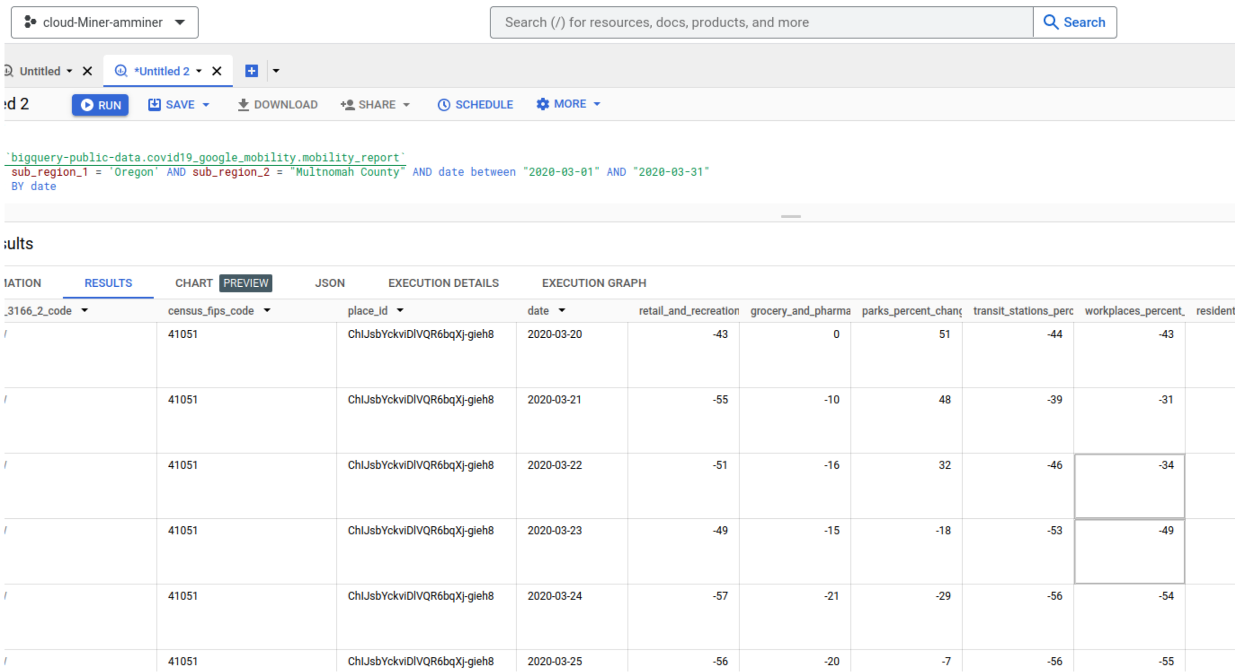




Amelia Miner
11/20/23

- On the day the stay-at-home order took effect (3/23/2020), what was the total impact on workplace trips?

  workplace presence decreased by 15% of baseline.



census_fips_code	place_id	date	retail_and_recreation	grocery_and_pharma	parks_percent_chang	transit_stations_perc	workplaces_percent_
41051	ChIJsbYckviDlVQR6bqXj-gieh8	2020-03-20	-43	0	51	-44	-43
41051	ChIJsbYckviDlVQR6bqXj-gieh8	2020-03-21	-55	-10	48	-39	-31
41051	ChIJsbYckviDlVQR6bqXj-gieh8	2020-03-22	-51	-16	32	-46	-34
41051	ChIJsbYckviDlVQR6bqXj-gieh8	2020-03-23	-49	-15	-18	-53	-49
41051	ChIJsbYckviDlVQR6bqXj-gieh8	2020-03-24	-57	-21	-29	-56	-54
41051	ChIJsbYckviDlVQR6bqXj-gieh8	2020-03-25	-56	-20	-7	-56	-55

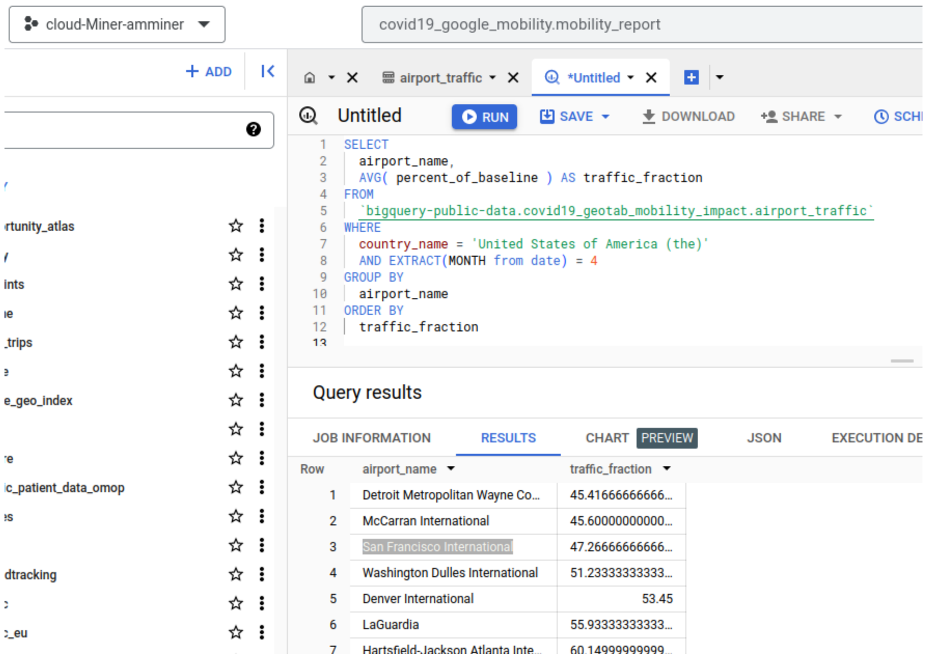
9. airport traffic

- Another dataset that is available is one that measures vehicle traffic changes. Find the column of covid19_geotab_mobility_impact.airport_traffic that gives us info on traffic impact. Adapt the following query:

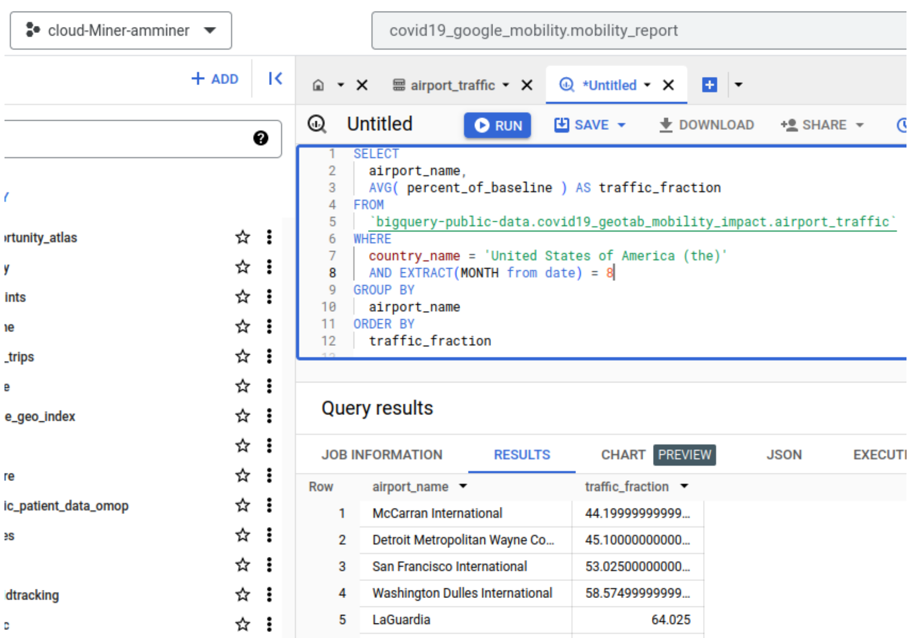
```
SELECT
  airport_name,
  AVG( ... ) AS traffic_fraction
FROM
  `bigquery-public-data.covid19_geotab_mobility_impact.airport_traffic`
WHERE
  country_name = 'United States of America (the)'
  AND EXTRACT(MONTH from date) = 4
GROUP BY
  airport_name
ORDER BY
  traffic_fraction
```

Amelia Miner
11/20/23

- Which three airports were impacted the most in April 2020 (the month when lockdowns became widespread)?
  Detroit Metropolitan Wayne County, McCarran International, and San Francisco International



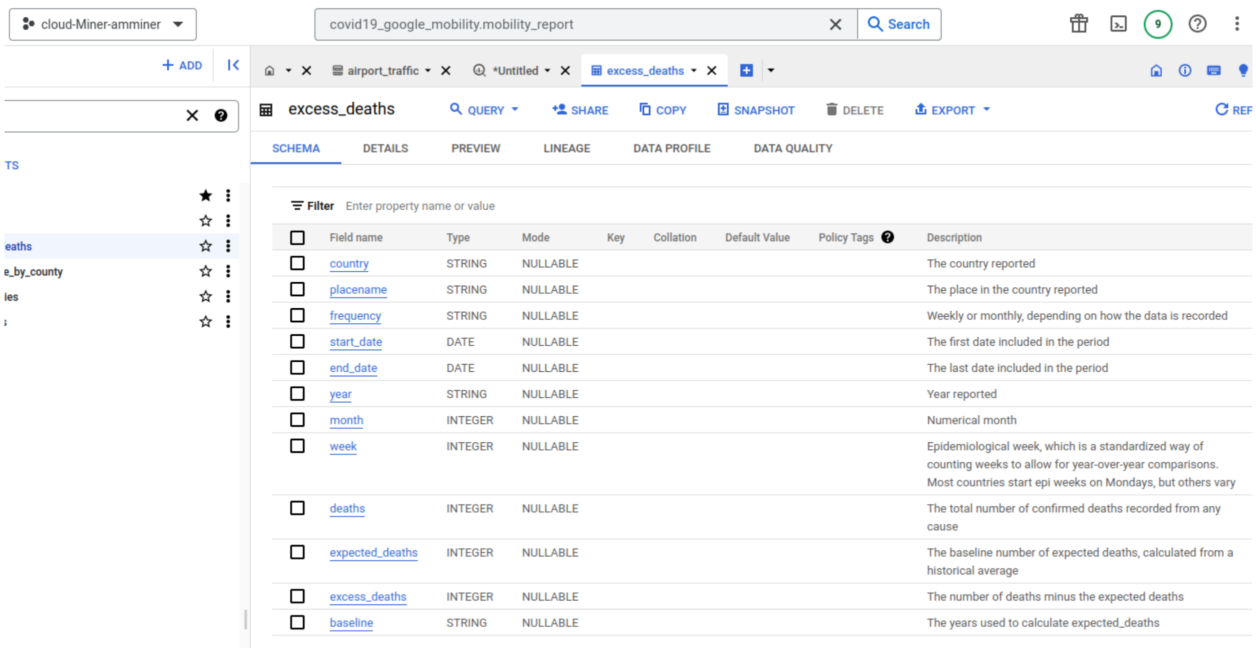
- Run the query again using the month of August 2020. Which three airports were impacted the most?
  The same 3 as above.



Amelia Miner
11/20/23

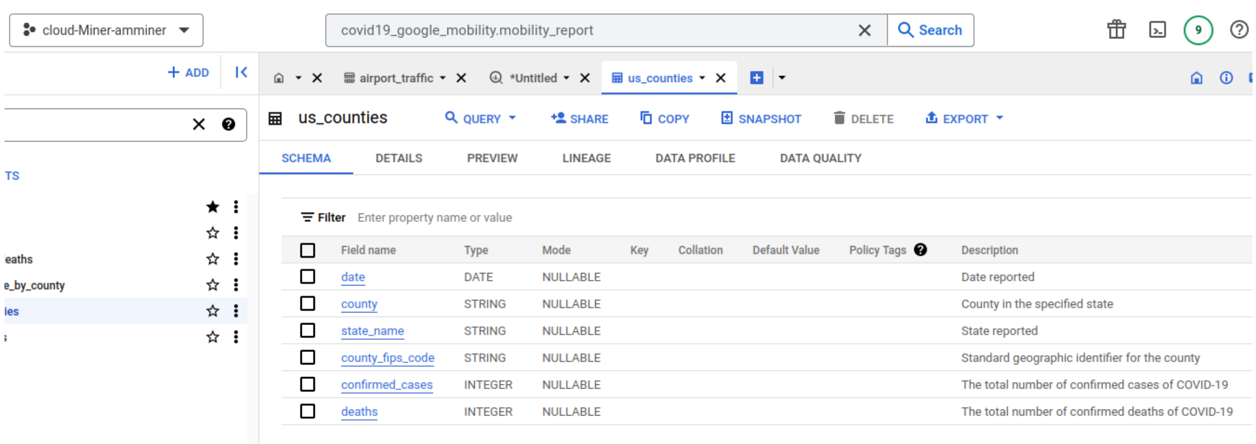
10. mortality

- In the console find the New York Times COVID-19 dataset and expand it. View each of the 4 tables therein. What table and columns identify the place name, the starting date, and the number of excess deaths from COVID-19?
  The table is excess_deaths and the columns are placename, start_date, and excess_deaths.



Field name	Type	Mode	Key	Collation	Default Value	Policy Tags	Description
country	STRING	NULLABLE					The country reported
placename	STRING	NULLABLE					The place in the country reported
frequency	STRING	NULLABLE					Weekly or monthly, depending on how the data is recorded
start_date	DATE	NULLABLE					The first date included in the period
end_date	DATE	NULLABLE					The last date included in the period
year	STRING	NULLABLE					Year reported
month	INTEGER	NULLABLE					Numerical month
week	INTEGER	NULLABLE					Epidemiological week, which is a standardized way of counting weeks to allow for year-over-year comparisons. Most countries start epi weeks on Mondays, but others vary
deaths	INTEGER	NULLABLE					The total number of confirmed deaths recorded from any cause
expected_deaths	INTEGER	NULLABLE					The baseline number of expected deaths, calculated from a historical average
excess_deaths	INTEGER	NULLABLE					The number of deaths minus the expected deaths
baseline	STRING	NULLABLE					The years used to calculate expected_deaths

- What table and columns identify the date, county, and deaths from COVID-19?
  us_counties: date, county, deaths

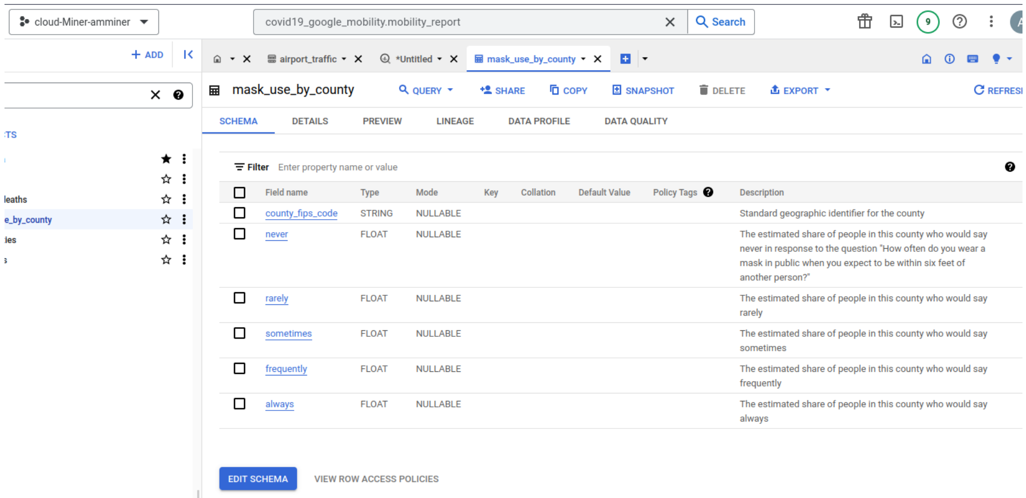


Field name	Type	Mode	Key	Collation	Default Value	Policy Tags	Description
date	DATE	NULLABLE					Date reported
county	STRING	NULLABLE					County in the specified state
state_name	STRING	NULLABLE					State reported
county_fips_code	STRING	NULLABLE					Standard geographic identifier for the county
confirmed_cases	INTEGER	NULLABLE					The total number of confirmed cases of COVID-19
deaths	INTEGER	NULLABLE					The total number of confirmed deaths of COVID-19

- What table and columns identify the date, state, and confirmed cases of COVID-19?
  us_counties: date, state, confirmed_cases (see above…)

Amelia Miner
11/20/23

- What table and columns identify a county code and the percentage of its residents that report they always wear masks?
  mask_use_by_county: county_fips_code, always



11. run example queries

- Run this query and graph generation:

```
SELECT date, confirmed_cases
FROM `bigquery-public-data.covid19_nyt.us_states`
WHERE state_name = 'Oregon'
ORDER BY date ASC
df.plot(x='date', y='confirmed_cases', kind='line', rot=45)
```

[2]:
```
query_string = '''  # amminer
SELECT date, confirmed_cases
FROM `bigquery-public-data.covid19_nyt.us_states`
WHERE state_name = 'Oregon'
ORDER BY date ASC
'''
```

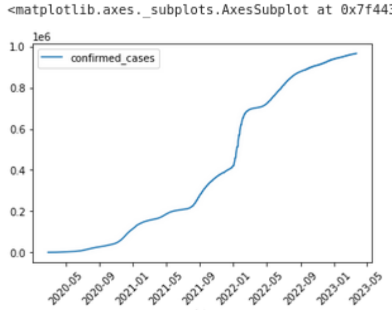
[3]:
```
from google.cloud import bigquery
df = bigquery.Client().query(query_string).to_dataframe()
df.head(3)
```

[3]:

	date	confirmed_cases
0	2020-02-28	1
1	2020-02-29	1
2	2020-03-01	2

[4]:
```
df.plot(x='date', y='confirmed_cases', kind='line', rot=45)
```

[4]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f443ab75a50>
```

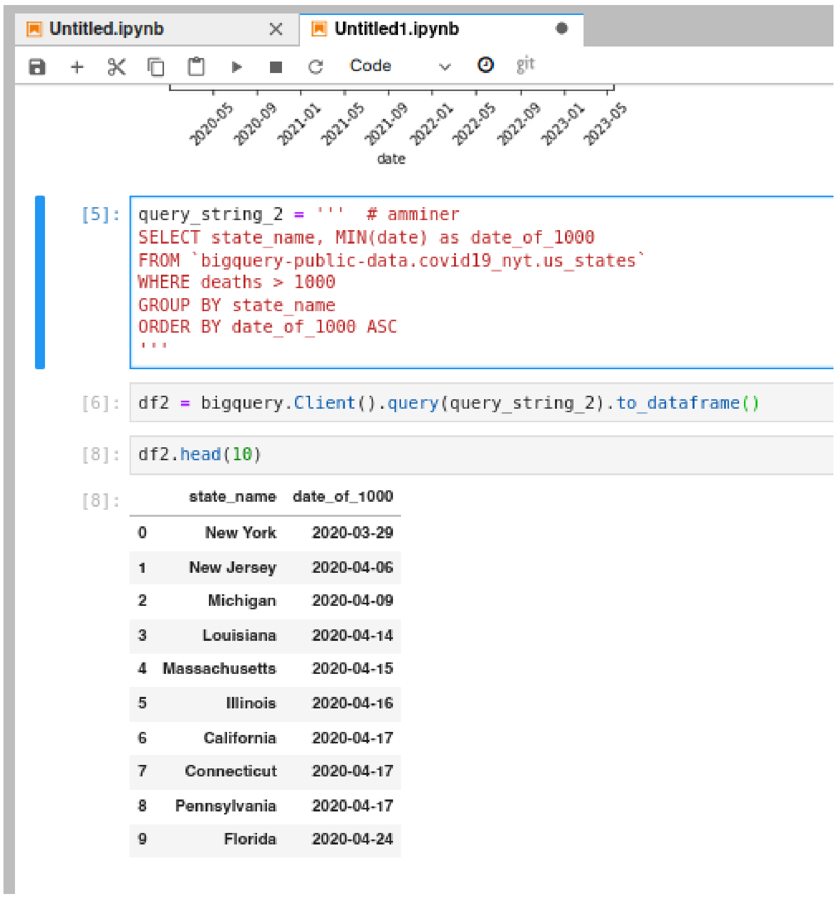


Amelia Miner
11/20/23

- Try this one:

```
SELECT state_name, MIN(date) as date_of_1000
FROM `bigquery-public-data.covid19_nyt.us_states`
WHERE deaths > 1000
GROUP BY state_name
ORDER BY date_of_1000 ASC
```

From within your Jupyter notebook, run the query and write code that shows the first 10 states that reached 1000 deaths from COVID-19.



[5]:
```
query_string_2 = '''  # amminer
SELECT state_name, MIN(date) as date_of_1000
FROM `bigquery-public-data.covid19_nyt.us_states`
WHERE deaths > 1000
GROUP BY state_name
ORDER BY date_of_1000 ASC
'''
```

[6]:
```
df2 = bigquery.Client().query(query_string_2).to_dataframe()
```

[8]:
```
df2.head(10)
```

[8]:

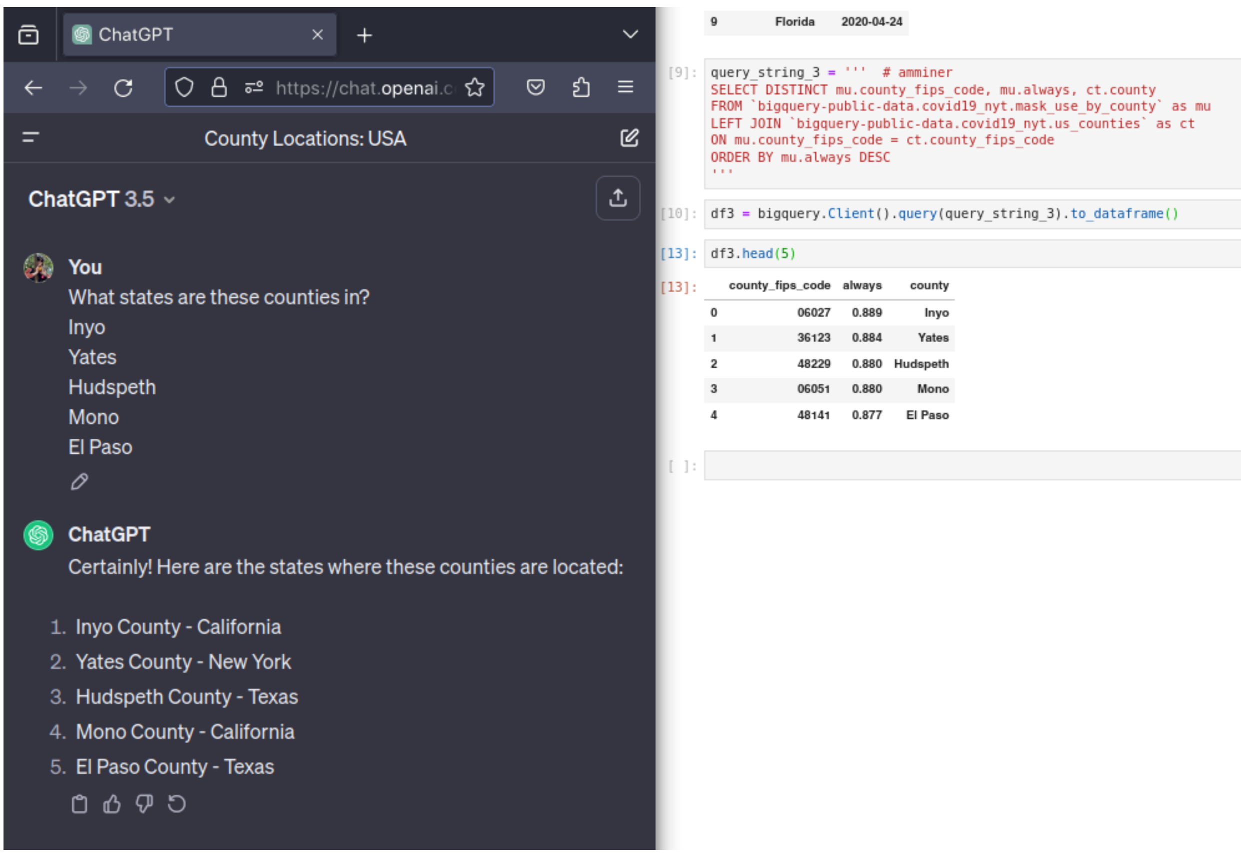
	state_name	date_of_1000
0	New York	2020-03-29
1	New Jersey	2020-04-06
2	Michigan	2020-04-09
3	Louisiana	2020-04-14
4	Massachusetts	2020-04-15
5	Illinois	2020-04-16
6	California	2020-04-17
7	Connecticut	2020-04-17
8	Pennsylvania	2020-04-17
9	Florida	2020-04-24

- Run the following query in your lab notebook that ranks the top counties in the US in which survey respondents always wear masks.

```
SELECT DISTINCT mu.county_fips_code, mu.always, ct.county
FROM `bigquery-public-data.covid19_nyt.mask_use_by_county` as mu
LEFT JOIN `bigquery-public-data.covid19_nyt.us_counties` as ct
ON mu.county_fips_code = ct.county_fips_code
ORDER BY mu.always DESC
```

Take a screenshot for your lab notebook of the Top 5 counties and the states they are located in.

Note that I was not asked to do that last part programmatically.



Amelia Miner
11/20/23

12. write queries

- Construct a query string that obtains the number of deaths from COVID-19 that have occurred in Multnomah county for each day in the dataset, ensuring the data is returned in ascending order of date. Run the query and obtain the results. Plot the results and take a screenshot for your lab notebook.

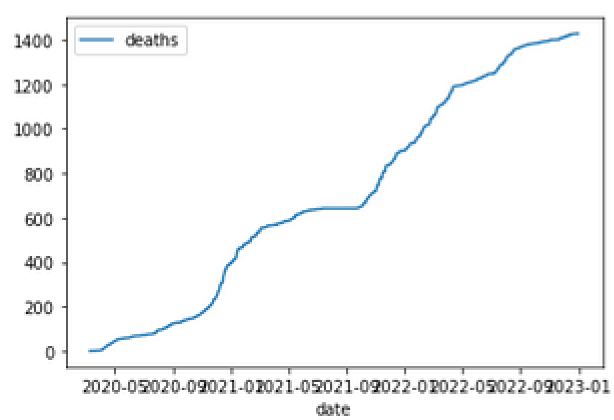
[26]:
```
def get_results(query):
    return bigquery.Client().query(query).to_dataframe()

query_string_4 = '''  # amminer
SELECT deaths, date
FROM `bigquery-public-data.covid19_nyt.us_counties`
WHERE county = 'Multnomah'
ORDER BY date ASC
'''

df4 = get_results(query_string_4)
df4.head(10)
```

[26]:

	deaths	date
0	0	2020-03-10
1	0	2020-03-11
2	0	2020-03-12
3	0	2020-03-13
4	1	2020-03-14
5	1	2020-03-15
6	1	2020-03-16
7	1	2020-03-17
8	1	2020-03-18
9	1	2020-03-19

[25]:
```
df4.plot(x='date', y='deaths', kind='line')
```

[25]: `<matplotlib.axes._subplots.AxesSubplot at 0x7f4432ee0310>`



- Construct a query string that obtains the number of deaths from COVID-19 that have occurred in Oregon for each day in the dataset, ensuring the data is returned in ascending order of date. Run the query and obtain the results. Plot the results and take a screenshot for your lab notebook.

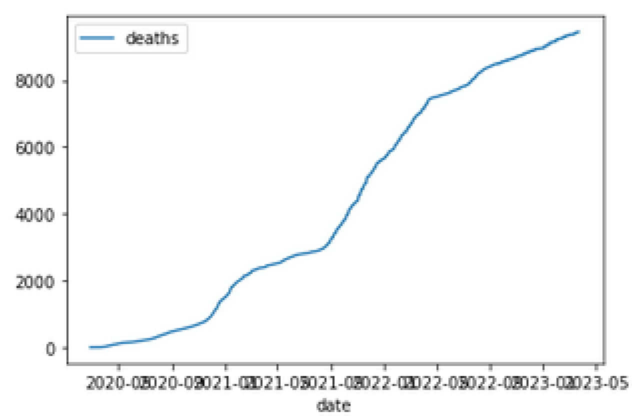
[28]:
```
query_string_5 = '''  # amminer
SELECT date, deaths
FROM `bigquery-public-data.covid19_nyt.us_states`
WHERE state_name = 'Oregon'
ORDER BY date ASC
'''

df5 = get_results(query_string_5)
df5.tail(10)
```

[28]:

	date	deaths
1110	2023-03-14	9378
1111	2023-03-15	9432
1112	2023-03-16	9432
1113	2023-03-17	9432
1114	2023-03-18	9432
1115	2023-03-19	9432
1116	2023-03-20	9432
1117	2023-03-21	9432
1118	2023-03-22	9451
1119	2023-03-23	9451

[30]:
```
df5.plot(x='date', y='deaths', kind='line')
```

[30]: `<matplotlib.axes._subplots.AxesSubplot at 0x7f441ff73dd0>`



13. clean up

- Delete the notebook:
  `gcloud notebooks instances delete bq-jupyter-instance --location us-west1-b`
  ✅

Amelia Miner
11/20/23

III. Lab 9.3g: Dataproc

1. Dataproc lab #1 (π)

- Cloud Dataproc manages and abstracts a Spark/Hadoop system of considerable complexity.
  ✅

2. Calculating π

- We'll use massively parallel dart throwing. For a unit circle centered over a 1x1 square, π = 4*darts_in_circle / total_darts.
  ✅

3. Code

- map: throw 1000 darts. Reduce: count the darts in the circle.

```
def inside(p):
  x, y = random.random(), random.random()
  return x*x + y*y < 1
count = sc.parallelize(xrange(0,
NUM_SAMPLES)).filter(inside).count()
print "Pi is roughly %f" % (4.0 * count / NUM_SAMPLES)
```

  where sc is an Apache Spark context.
  ✅

4. dataproc setup

- Enable the API:

```
gcloud services enable dataproc.googleapis.com
```

  Set the default zone and region for CE and dataproc:

```
gcloud config set compute/zone us-west1-b
gcloud config set compute/region us-west1
gcloud config set dataproc/region us-west1
```

  Set an env var CLUSTERNAME:

```
CLUSTERNAME=${USER}-dplab
```

  ✅

Amelia Miner
11/20/23

5. create compute engine cluster

- Create a cluster:

```
gcloud dataproc clusters create ${CLUSTERNAME} \
  --scopes=cloud-platform \
  --tags codelab \
  --region=us-west1 \
  --zone=us-west1-b \
  --master-machine-type=e2-medium \
  --worker-machine-type=e2-medium \
  --master-boot-disk-size=30GB \
  --worker-boot-disk-size=30GB
```

  View the cluster in the dataproc web console. View the nodes in CE.
  ✅

6. run computation

- Note the current time, then submit the job, specifying 1000 workers. We'll run the Java version of the program that comes included in the Apache Spark distribution.

```
date
gcloud dataproc jobs submit spark --cluster ${CLUSTERNAME} \
  --class org.apache.spark.examples.SparkPi \
  --jars file:///usr/lib/spark/examples/jars/spark-examples.jar -- 1000 \
  >& output.txt &
```

  You can check on it with

```
gcloud dataproc jobs list --cluster ${CLUSTERNAME}
date
```

- How long did the job take to execute?
  About 1 minute and 35 seconds. The console says 35 seconds, so I think I was a little slow to check.

```
amminer@cloudshell:~ (cloud-miner-amminer)$ date
Fri 24 Nov 2023 04:06:19 AM UTC
amminer@cloudshell:~ (cloud-miner-amminer)$ gcloud dataproc jobs submit spark --cluster ${CLUSTERNAME}   --class org.apache.spark.examples.SparkPi   --jars file:
///usr/lib/spark/examples/jars/spark-examples.jar -- 1000   >& output.txt &
[1] 1636
amminer@cloudshell:~ (cloud-miner-amminer)$ gcloud dataproc jobs list --cluster ${CLUSTERNAME}
JOB_ID: f179a6364080473db1a4785fa0ba2543
TYPE: spark
STATUS: SETUP_DONE
amminer@cloudshell:~ (cloud-miner-amminer)$ gcloud dataproc jobs list --cluster ${CLUSTERNAME} && date
JOB_ID: f179a6364080473db1a4785fa0ba2543
TYPE: spark
STATUS: SETUP_DONE
Fri 24 Nov 2023 04:06:41 AM UTC
amminer@cloudshell:~ (cloud-miner-amminer)$ gcloud dataproc jobs list --cluster ${CLUSTERNAME} && date
JOB_ID: f179a6364080473db1a4785fa0ba2543
TYPE: spark
STATUS: DONE
[1]+  Done                    gcloud dataproc jobs submit spark --cluster ${CLUSTERNAME} --class org.apache.spark.examples.SparkPi --jars file:///usr/lib/spark/e
xamples/jars/spark-examples.jar -- 1000 &> output.txt
Fri 24 Nov 2023 04:07:55 AM UTC
amminer@cloudshell:~ (cloud-miner-amminer)$
```

- Examine output.txt and show the estimate of π.

```
amminer@cloudshell:~ (cloud-miner-amminer)$ grep 3\\.14 output.txt -C 1
23/11/24 04:06:36 INFO com.google.cloud.hadoop.fs.gcs.GhfsStorageStatistics: Detected potential high latency for operation op_create. latencyMs=498; previousMaxL
atencyMs=0; operationCount=1; context=gs://dataproc-temp-us-west1-376082578160-nlveoe4x/d0f2c031-e816-4fa8-a506-e6da84833735/spark-job-history/application_170079
7065670_0001.inprogress
Pi is roughly 3.1416232714162327
23/11/24 04:06:53 INFO org.sparkproject.jetty.server.AbstractConnector: Stopped Spark@693de52b{HTTP/1.1, (http/1.1)}{0.0.0.0:0}
amminer@cloudshell:~ (cloud-miner-amminer)$
```

Amelia Miner
11/20/23

7. scale cluster

- Run `gcloud dataproc clusters describe ${CLUSTERNAME}` to find the number of instances used (**1 controller… manager… coordinator… seriously, computer scientists, can we pick another word besides master?... and 2 workers**).
- Allocate two additional pre-emptible machines to the cluster with
  `gcloud dataproc clusters update ${CLUSTERNAME} --num-secondary-workers=2`
  and repeat the earlier command to see that they show up in the listing. View them in the CE console.
  ✅

8. run computation again

- This time direct the output to a new file. How long did it take? How much faster was it?
  About 37 seconds; 58 seconds faster than before. The console shows 27 seconds, 8 seconds faster than the previous run's elapsed time field. I trust the console a lot more than I trust my sporadic check-ins.

```
amminer@cloudshell:~ (cloud-miner-amminer)$ date && gcloud dataproc jobs submit spark --cluster ${CLUSTERNAME}   --class org.apache.spark.examples.SparkPi   --ja
rs file:///usr/lib/spark/examples/jars/spark-examples.jar -- 1000   >& output2.txt &
[6] 1963
amminer@cloudshell:~ (cloud-miner-amminer)$ Fri 24 Nov 2023 04:35:57 AM UTC
gcloud dataproc jobs list --cluster ${CLUSTERNAME} && daters file:///usr/lib/spark/examples/jars/spark
JOB_ID: 28ee1d5285d149c88e408422989e0a94
TYPE: spark
STATUS: SETUP_DONE

JOB_ID: f179a6364080473db1a4785fa0ba2543
TYPE: spark
STATUS: DONE
[5]-  Exit 1                  date && gcloud dataproc jobs submit spark --cluster ${CLUSTERNAME} --class org.apache.spark.examples.SparkPi --jars file:///usr/lib
/spark/examples/jars/spark-examples.jar -- 1000 &> output2.txt
Fri 24 Nov 2023 04:36:06 AM UTC
amminer@cloudshell:~ (cloud-miner-amminer)$ gcloud dataproc jobs list --cluster ${CLUSTERNAME} && date
JOB_ID: 28ee1d5285d149c88e408422989e0a94
TYPE: spark
STATUS: SETUP_DONE

JOB_ID: f179a6364080473db1a4785fa0ba2543
TYPE: spark
STATUS: DONE
Fri 24 Nov 2023 04:36:15 AM UTC
amminer@cloudshell:~ (cloud-miner-amminer)$ gcloud dataproc jobs list --cluster ${CLUSTERNAME} && date
JOB_ID: 28ee1d5285d149c88e408422989e0a94
TYPE: spark
STATUS: DONE

JOB_ID: f179a6364080473db1a4785fa0ba2543
TYPE: spark
STATUS: DONE
[6]+  Done                    date && gcloud dataproc jobs submit spark --cluster ${CLUSTERNAME} --class org.apache.spark.examples.SparkPi --jars file:///usr/lib
/spark/examples/jars/spark-examples.jar -- 1000 &> output2.txt
Fri 24 Nov 2023 04:36:34 AM UTC
amminer@cloudshell:~ (cloud-miner-amminer)$
```

- Show the estimate of pi in the output.

```
amminer@cloudshell:~ (cloud-miner-amminer)$ grep 3\\.14 output2.txt -C 1
23/11/24 04:36:09 INFO com.google.cloud.hadoop.fs.gcs.GhfsStorageStatistics: Detected potential high latency for operation op_create. latencyMs=497; previousMaxL
atencyMs=0; operationCount=1; context=gs://dataproc-temp-us-west1-376082578160-nlveoe4x/d0f2c031-e816-4fa8-a506-e6da84833735/spark-job-history/application_170079
7065670_0008.inprogress
Pi is roughly 3.141499711414997
23/11/24 04:36:23 INFO org.sparkproject.jetty.server.AbstractConnector: Stopped Spark@1549bba7{HTTP/1.1, (http/1.1)}{0.0.0.0:0}
amminer@cloudshell:~ (cloud-miner-amminer)$
```

9. clean up

- Delete the cluster (`gcloud dataproc clusters delete $CLUSTERNAME`)
  ✅

Amelia Miner
11/20/23

IV. Lab 9.4g: Dataflow

1. dataflow lab #1 (Java package popularity)

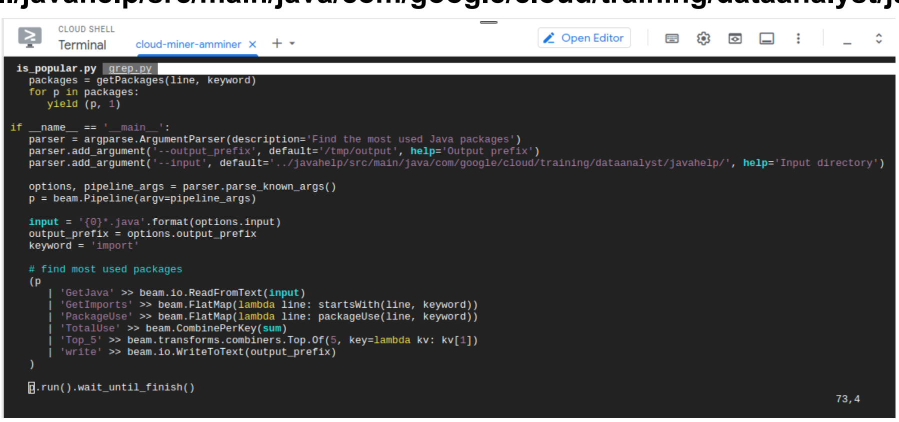
- Dataflow runs Apache Beam workloads. Beam does large scale processing of both stream and batch workloads using a transform-based approach. The programming paradigm is very functional and is expressed in graph-like form; **this reminds me of finite state automata**.
  One feature of dataflow is that it can run serverlessly, allocating compute capacity as needed without having to manage clusters of discreet machines.
  ✅

2. setup

- cd into training-data-analyst/courses/machine_learning/deepdive/04_features/dataflow/python/. Create a venv and activate it. pip install apache-beam[gcp] and oauth2client 3.0.0.
  ✅

3. beam code

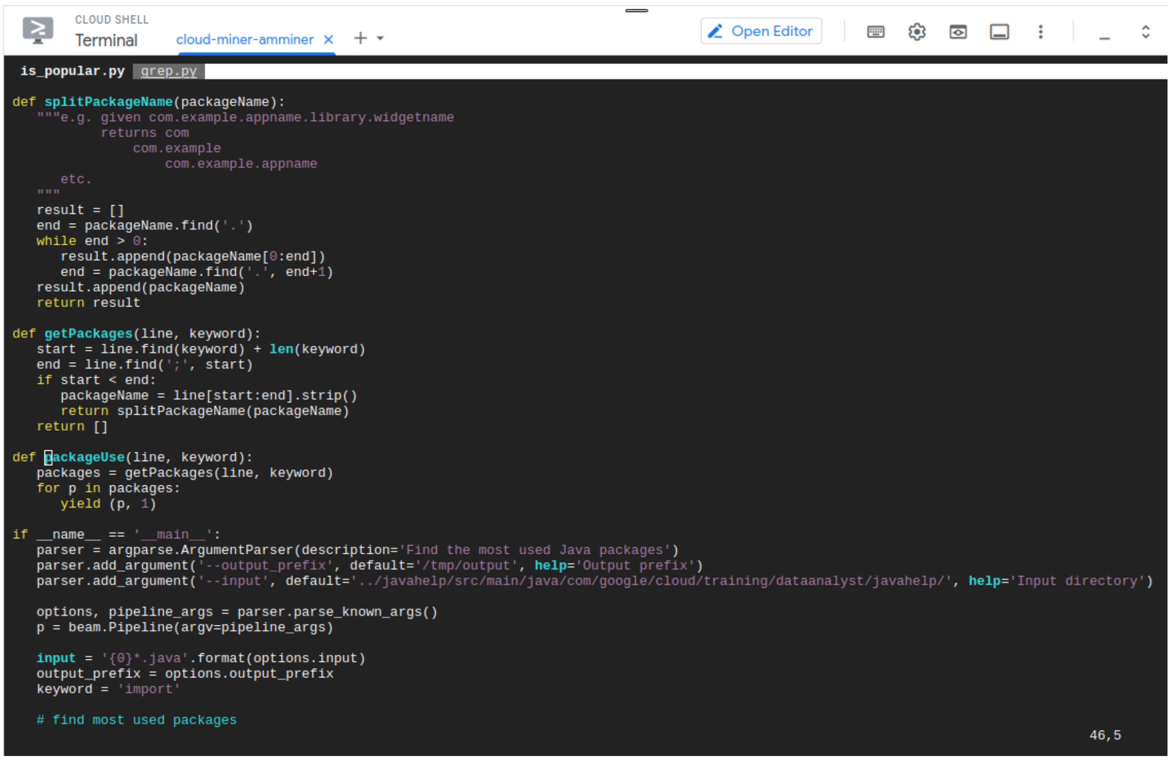
- Transforms can be mapped to their own compute nodes. Examine grep.py. It instantiates p, a bean pipeline, and configures some string variables for its input and output nodes (sinks), and a search term. The code then uses the | and >> operators, which the pipeline object has defined to be used for its configuration, to set the pipeline up. p.run().wait_until_finish() runs the pipeline and waits for it to finish.
  Now examine is_popular.py. Where is the input taken from by default?
  When is_popular is run as the main module from a terminal, it takes an --input argument which should be a directory containing a Java project's source code. "*.java" is appended to this path and the path is passed into the pipeline's configuration using that unusual `p | '…' >> _` syntax, so it looks like the pipeline takes input from any Java source code files in the directory passed to the --input argument from the terminal… I just realized I'm way over-explaining this. The default value of the input argument is another directory in this training repo, ../javahelp/src/main/java/com/google/cloud/training/dataanalyst/javahelp.



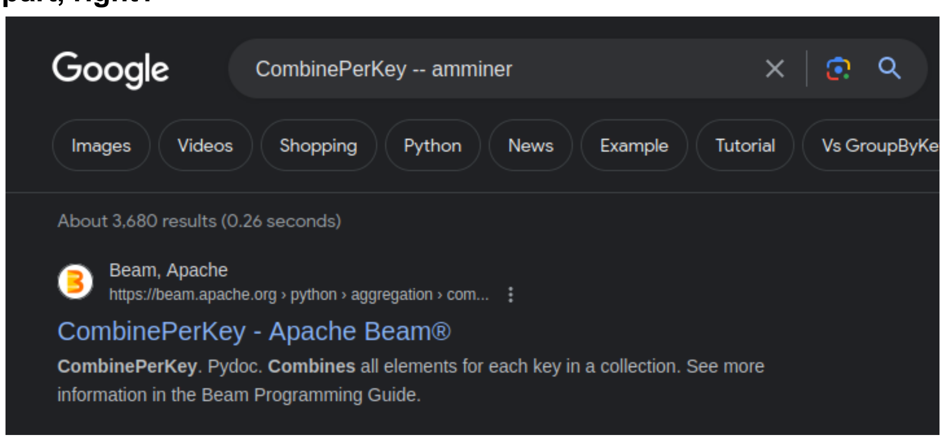
CS530 F23 Week 9 Lab
Prof. Feng

Amelia Miner
11/20/23

- Where does the output go by default?
  a file with the prefix /tmp/output. See above.
- Examine the getPackages and splitPackageName functions. What operation does the PackageUse transform implement?
  PackageUse uses the packageUse function, which uses getPackages, which uses splitPackageName; it implements retrieval of the name of each package and the number of times it is imported based on the packages found by the previous node in the graph, GetImports. This is the map in the map-reduce pattern, right?



- Look up Beam's CombinePerKey. What operation does the TotalUse operation implement?
  CombinePerKey "Combines all elements for each key in a collection". TotalUse implements the aggregation of the found package counts, which is the reduce part, right?



Amelia Miner
11/20/23

- Which operations correspond to a "Map"?
  GetImports and PackageUse.

amminer | Search | Create account Log in •••

MapReduce

20 languages

Article Talk | Read Edit View history Tools

From Wikipedia, the free encyclopedia

MapReduce is a programming model and an associated implementation for processing and generating big data sets with a parallel, distributed algorithm on a cluster.[1][2][3]

A MapReduce program is composed of a *map* procedure, which performs filtering and sorting (such as sorting students by first name into queues, one queue for each name), and a *reduce* method, which performs a summary operation (such as counting the number of students in each queue, yielding name frequencies). The "MapReduce

- Which operation corresponds to a "Shuffle-reduce"?
  PackageUse…?

amminer

A MapReduce framework (or system) is usually composed of three operations (or steps):

1. **Map:** each worker node applies the `map` function to the local data, and writes the output to a temporary storage. A master node ensures that only one copy of the redundant input data is processed.
2. **Shuffle:** worker nodes redistribute data based on the output keys (produced by the `map` function), such that all data belonging to one key is located on the same worker node.
3. **Reduce:** worker nodes now process each group of output data, per key, in parallel.

- Which corresponds to a "Reduce"?
  TotalUse… see above.

4. run pipeline locally

- Run the pipeline in cloud shell. cat the output file and show its contents.

```
(venv) amminer@cloudshell:~/training-data-analyst/courses/machine_learning/deepdive/04_features/dataflow/python (cloud-miner-amminer)$ python is_popular
.py
(venv) amminer@cloudshell:~/training-data-analyst/courses/machine_learning/deepdive/04_features/dataflow/python (cloud-miner-amminer)$ cat /tmp/output-0
0000-of-00001
[('org', 45), ('org.apache', 44), ('org.apache.beam', 44), ('org.apache.beam.sdk', 43), ('org.apache.beam.sdk.transforms', 16)]
(venv) amminer@cloudshell:~/training-data-analyst/courses/machine_learning/deepdive/04_features/dataflow/python (cloud-miner-amminer)$
```

- Explain what the data in the file corresponds to.
  The data shows the number of times each token appears in the java files in the default input directory.

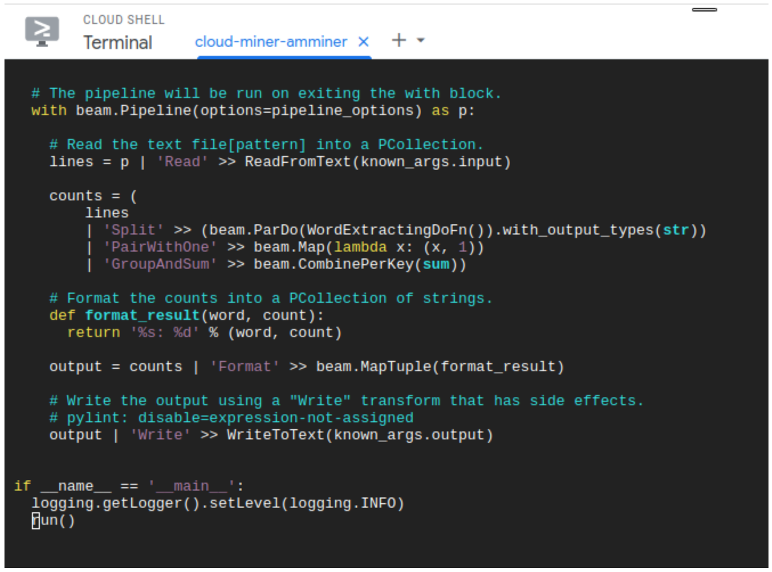
```
(venv) amminer@cloudshell:~/training-data-analyst/courses/machine_learning/deepdive/04_features/dataflow/python (cloud-miner-amminer)$ grep -r org.apach
e ../javahelp/src/main/java/com/google/cloud/training/dataanalyst/javahelp | wc -l
44
(venv) amminer@cloudshell:~/training-data-analyst/courses/machine_learning/deepdive/04_features/dataflow/python (cloud-miner-amminer)$
```

5. dataflow lab #2 (word count)

- Open venv/lib/python3.*/site-packages/apache_beam/examples/wordcount.py. A pipeline p is incrementally constructed with beam's funky syntax; the pipeline is assigned

Amelia Miner
11/20/23

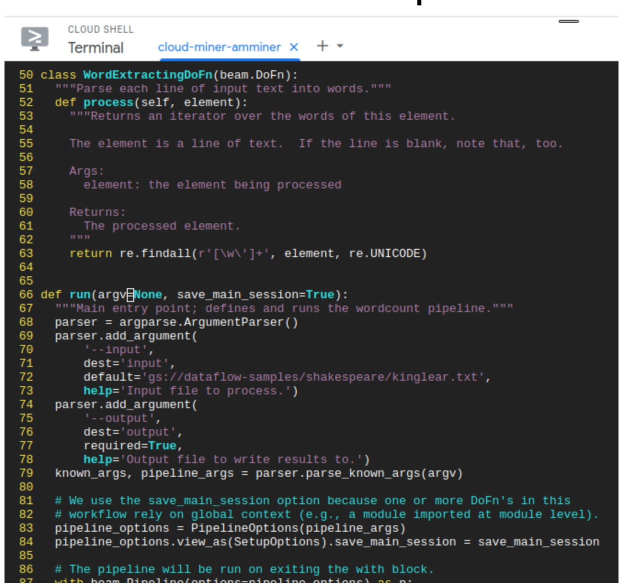
to a few different variables as the code progresses, lines, counts, and output, before it's run. What are the names of the stages in the pipeline?
Read, Split, PairWithOne, GroupAndSum, Format, and Write.



CLOUD SHELL
Terminal cloud-miner-amminer

```python
  # The pipeline will be run on exiting the with block.
  with beam.Pipeline(options=pipeline_options) as p:

    # Read the text file[pattern] into a PCollection.
    lines = p | 'Read' >> ReadFromText(known_args.input)

    counts = (
        lines
        | 'Split' >> (beam.ParDo(WordExtractingDoFn()).with_output_types(str))
        | 'PairWithOne' >> beam.Map(lambda x: (x, 1))
        | 'GroupAndSum' >> beam.CombinePerKey(sum))

    # Format the counts into a PCollection of strings.
    def format_result(word, count):
      return '%s: %d' % (word, count)

    output = counts | 'Format' >> beam.MapTuple(format_result)

    # Write the output using a "Write" transform that has side effects.
    # pylint: disable=expression-not-assigned
    output | 'Write' >> WriteToText(known_args.output)


if __name__ == '__main__':
  logging.getLogger().setLevel(logging.INFO)
  run()
```

- Describe what each stage does.
  Read reads text from an input file.
  Split splits each line of the text into words using regex.
  PairWithOne constructs key, value pairs (using tuples) where each word is mapped to the integer value 1.
  GroupAndSum is a shuffle-reduce step that aggregates identical keys and reduces them to a single instance of the key-value pair where the value is the sum of the other values (a count of how many times the word occurs).
  Format uses the format_result function to create more human-readable output showing the words and their counts.
  Write writes the formatted output a text file.



CLOUD SHELL
Terminal cloud-miner-amminer

```python
class WordExtractingDoFn(beam.DoFn):
  """Parse each line of input text into words."""
  def process(self, element):
    """Returns an iterator over the words of this element.

    The element is a line of text.  If the line is blank, note that, too.

    Args:
      element: the element being processed

    Returns:
      The processed element.
    """
    return re.findall(r'[\w\']+', element, re.UNICODE)


def run(argv=None, save_main_session=True):
  """Main entry point; defines and runs the wordcount pipeline."""
  parser = argparse.ArgumentParser()
  parser.add_argument(
      '--input',
      dest='input',
      default='gs://dataflow-samples/shakespeare/kinglear.txt',
      help='Input file to process.')
  parser.add_argument(
      '--output',
      dest='output',
      required=True,
      help='Output file to write results to.')
  known_args, pipeline_args = parser.parse_known_args(argv)

  # We use the save_main_session option because one or more DoFn's in this
  # workflow rely on global context (e.g., a module imported at module level).
  pipeline_options = PipelineOptions(pipeline_args)
  pipeline_options.view_as(SetupOptions).save_main_session = save_main_session

  # The pipeline will be run on exiting the with block.
  with beam.Pipeline(options=pipeline_options) as p:
```

6. run code locally

- Run the script in cloud shell:

  ```
  python -m apache_beam.examples.wordcount \
    --output outputs
  ```

  Use wc to determine the number of unique keywords in King Lear.

  4784

  ```
  amminer@cs-1054049896441-default:python$ wc -l outputs-00000-of-00001
  4784 outputs-00000-of-00001
  amminer@cs-1054049896441-default:python$
  ```

- Use sort to perform a numeric sort on the key field containing the count for each word in descending order. Pipe the output to head to show the top 3 words and their counts.

  ```
  amminer@cs-1054049896441-default:python$ sort -r -n -k 2 outputs-00000-of-00001 | head -3
  the: 786
  I: 622
  and: 594
  amminer@cs-1054049896441-default:python$
  ```

- The pipeline is case-sensitive. Edit the pipeline and, in the appropriate position, insert a stage that transforms all characters to lowercase like so:

  ```
       | 'lowercase' >> beam.Map(lambda x: x.lower())
  ```

  Repeat the above process with the case-insensitive version to show the top 3 words in King Lear.

  ```
  amminer@cs-1054049896441-default:python$ sort -r -n -k 2 outputs-00000-of-00001 | head -3
  the: 908
  and: 738
  i: 622
  amminer@cs-1054049896441-default:python$
  ```

7. setup for cloud dataflow

- Enable the necessary dataflow, compute_component, storage_component, and storage_api APIs. Throughout the lab we'll reference the same bucket:

  ```
  export BUCKET=${GOOGLE_CLOUD_PROJECT}
  export REGION=us-west1
  gsutil mb gs://${BUCKET}
  ```

  ✅

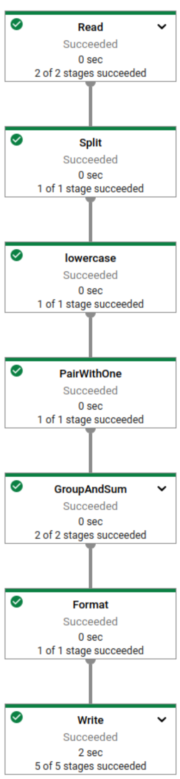
8. service account setup

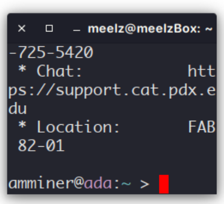
- cd into ~ and create a service account named df-lab. Add the following IAM roles:
  dataflow.admin to create and manage Dataflow jobs,
  dataflow.worker to create Compute Engine VMs (workers) on-demand,
  storage.admin to create storage buckets for the results

iam.serviceAccountUser to assign service accounts to Compute Engine VMs, and serviceusage.serviceUsageConsumer to use platform services.
Create a service account key:
```
gcloud iam service-accounts keys create df-lab.json --iam-account df-lab@${GOOGLE_CLOUD_PROJECT}.iam.gserviceaccount.com
```
finally, set this env var:
```
export GOOGLE_APPLICATION_CREDENTIALS=$PWD/df-lab.json
```
✅

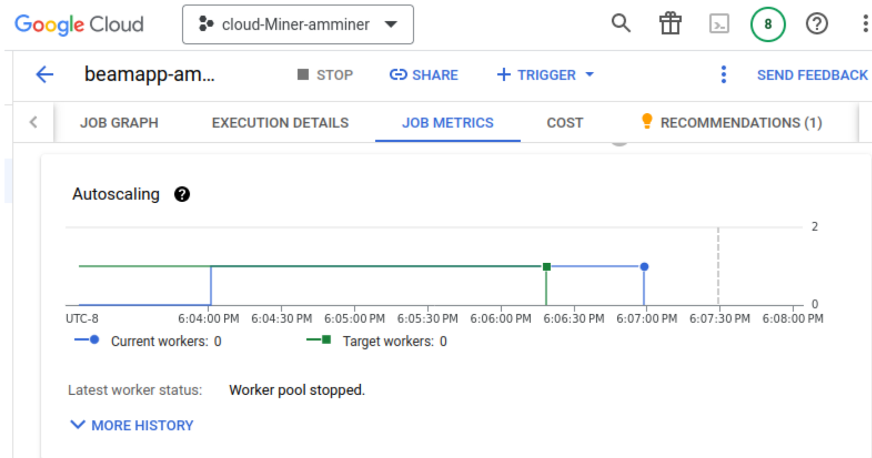
9. run code using dataflow runner

- Repeat our earlier execution, but specify the DataflowRunner:
```
python -m apache_beam.examples.wordcount \
  --region ${REGION} \
  --input gs://dataflow-samples/shakespeare/kinglear.txt \
  --output gs://$BUCKET/results/outputs \
  --runner DataflowRunner \
  --project ${GOOGLE_CLOUD_PROJECT} \
  --temp_location gs://${BUCKET}/tmp/
```
visit Dataflow in the web console and click on the Dataflow job that was executed. Examine both "Job Graph" and "Job Metrics". Include the following in your lab notebook:
    - The part of the job graph that has taken the longest time to complete.
      Write

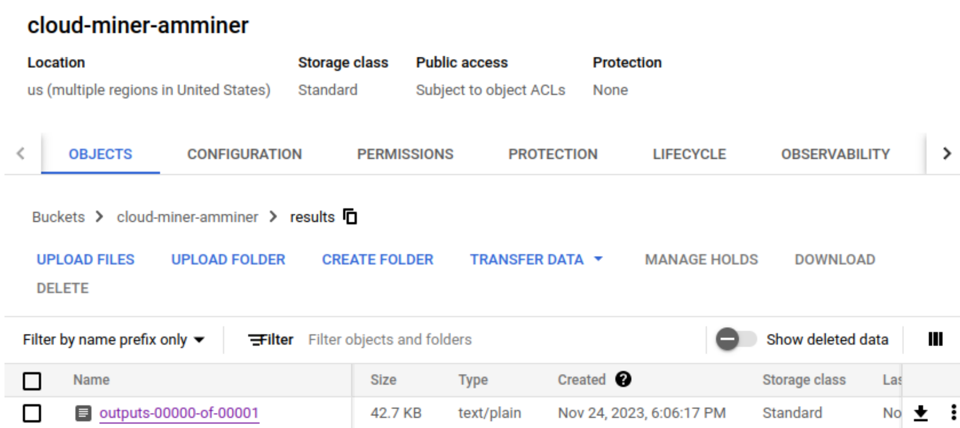




- The autoscaling graph showing when the worker was created and stopped.



- Examine the output directory in Cloud Storage. How many files has the final write stage in the pipeline created?
  One.



10. clean up

- Delete the IAM policies, the service account, and the bucket.
  ✅

11. dataflow lab #3 (taxi ETL pipeline)

- This lab will take raw data from a live pub-sub stream of taxi rides in NYC, extracting and cleaning the data in real time and inserting records into a BigQuery data warehouse. We'll then run some simple queries and explore visualization with Looker.
  ✅

Amelia Miner
11/20/23

12. view raw data from PubSub

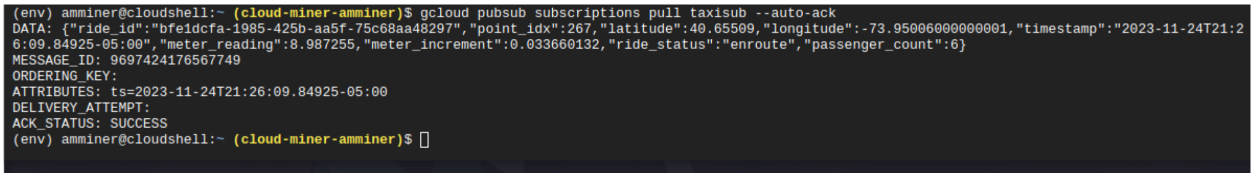
- Enable the pubsub service if you haven't already. Create a subscription:

```
gcloud pubsub subscriptions create taxisub \
  --topic=projects/pubsub-public-data/topics/taxirides-realtime
```

Pull and ack one message from the stream and examine the returned data.

```
gcloud pubsub subscriptions pull taxisub --auto-ack
```

Take a screenshot listing the different fields of this object and then delete the subscription.



13. BigQuery and Dataflow setup

- Make a BQ dataset: `bq mk taxirides`. Then make a table in it:

```
bq mk \
  --time_partitioning_field timestamp \
  --schema ride_id:string,point_idx:integer,latitude:float,longitude:float,\
timestamp:timestamp,meter_reading:float,meter_increment:float,ride_status:string,\
passenger_count:integer \
  -t taxirides.realtime
```

Finally, create a storage bucket to stage intermediate data.

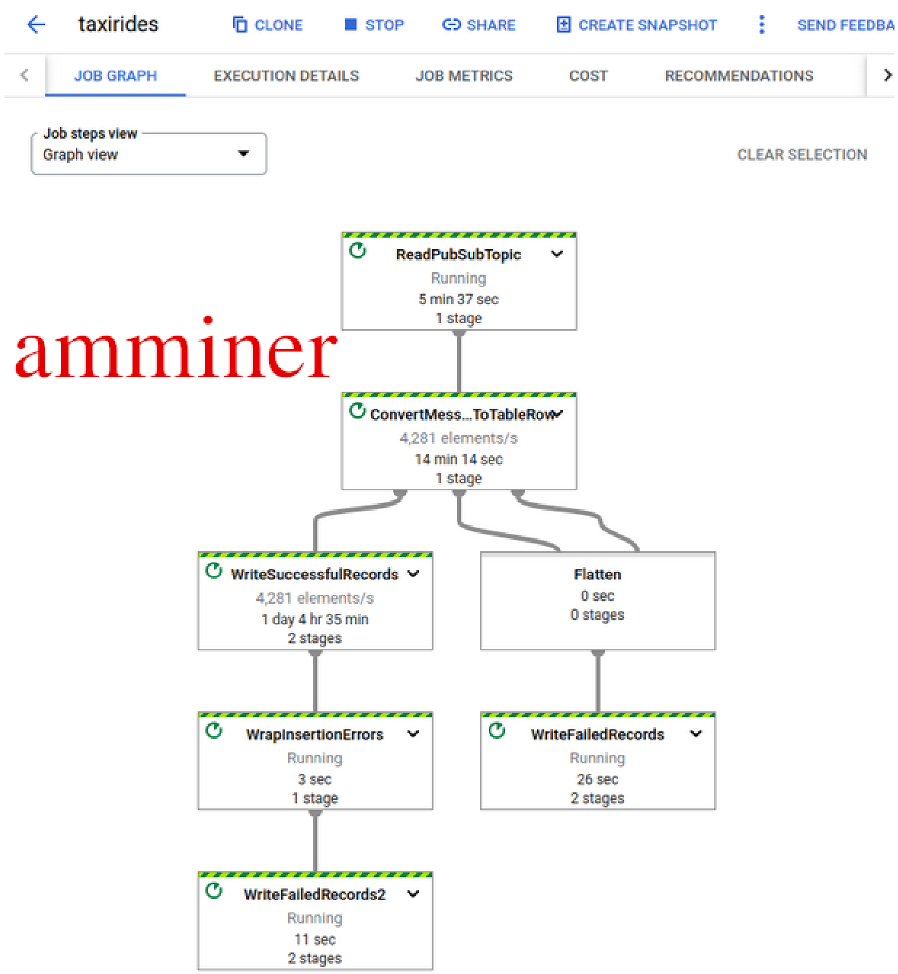
```
gsutil mb gs://${GOOGLE_CLOUD_PROJECT}-taxi
```

✅

Amelia Miner
11/20/23

14. Run Dataflow job from template

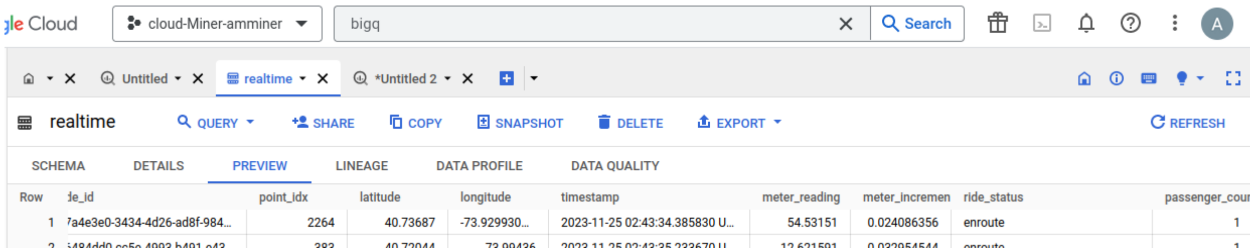
- Ingesting pubsub data into bigquery is so common that dataflow has a template for it. This is sometimes called zero-ETL since extraction and loading are done on the platform **(what about transformation? Seems to be done "on the platform" too?)**
Bring up a serverless Dataflow pipeline using this template:
```
gcloud dataflow jobs run taxirides \
        --gcs-location gs://dataflow-templates/latest/PubSub_to_BigQuery \
        --region us-west1 \
        --staging-location gs://${GOOGLE_CLOUD_PROJECT}-taxi/tmp \
        --parameters
inputTopic=projects/pubsub-public-data/topics/taxirides-realtime,\
outputTableSpec=${GOOGLE_CLOUD_PROJECT}:taxirides.realtime
```
Then visit the Cloud Dataflow console. Wait 5-10 minutes until substantial data has come through the pipeline. Take a screenshot of the pipeline that includes its stages and the number of elements per second being handled by individual stages.



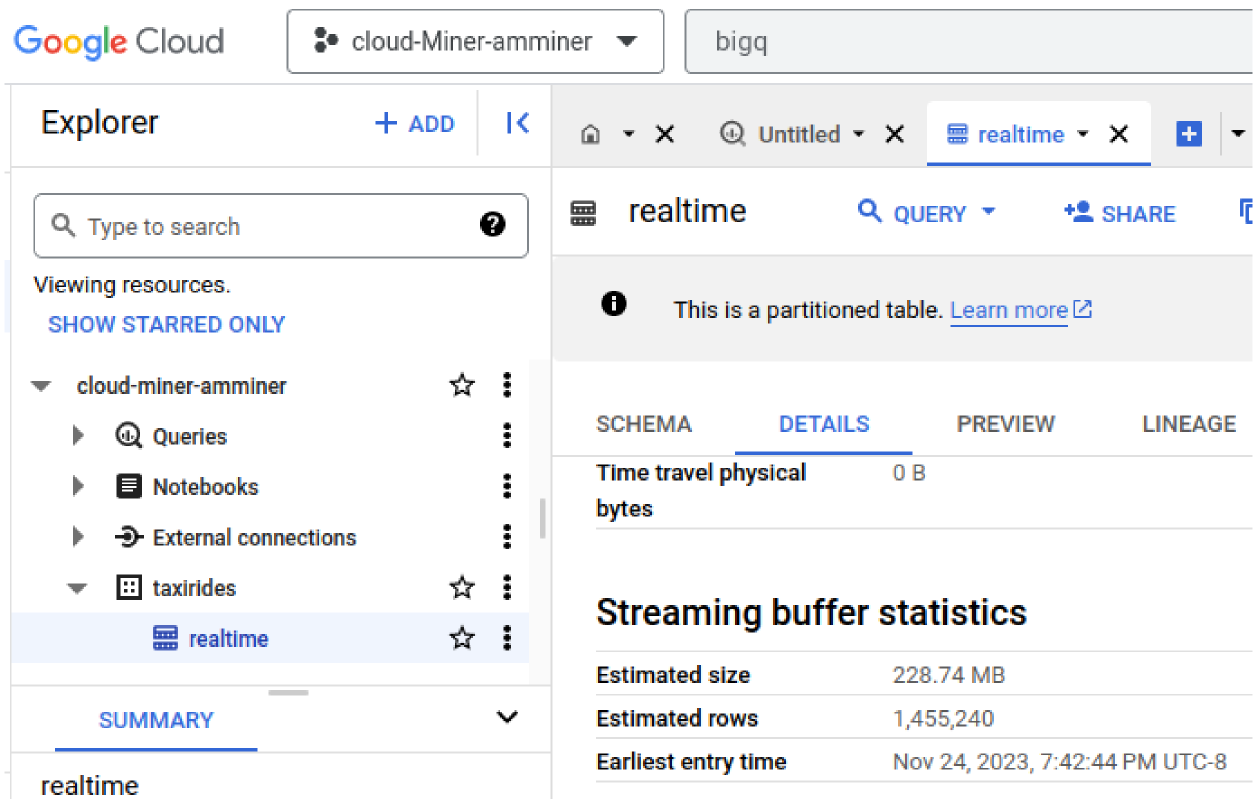
- Cancel the job:
```
gcloud dataflow jobs list --status=active
gcloud dataflow jobs cancel <JOB_ID> --region=us-west1
```

15. query data in BigQuery

- Visit the BigQuery web console and navigate to the table we've created. Show the number of passengers and the amount paid for the first ride.

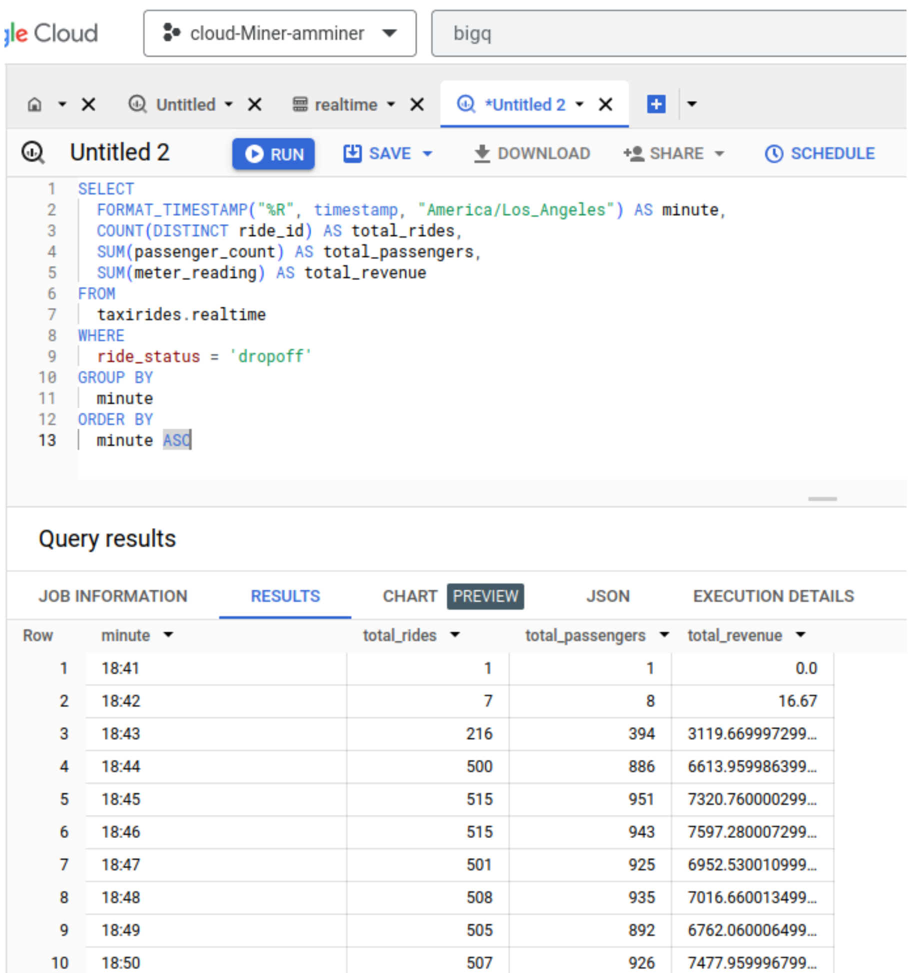


- Show the estimated number of rows in the table:
At this point I am forced to reveal that I accidentally got sidetracked and left the pipeline running for about an hour. It only cost 47 cents, so not a huge deal, but I am still not proud of letting it spin for that much longer than necessary.



- Query the table to obtain the number of rides, passengers, and amount of revenue for the rides taken during the collection period:

```
SELECT
  FORMAT_TIMESTAMP("%R", timestamp, "America/Los_Angeles") AS minute,
  COUNT(DISTINCT ride_id) AS total_rides,
  SUM(passenger_count) AS total_passengers,
  SUM(meter_reading) AS total_revenue
FROM
  taxirides.realtime
WHERE
  ride_status = 'dropoff'
GROUP BY
  minute
ORDER BY
  minute ASC
```

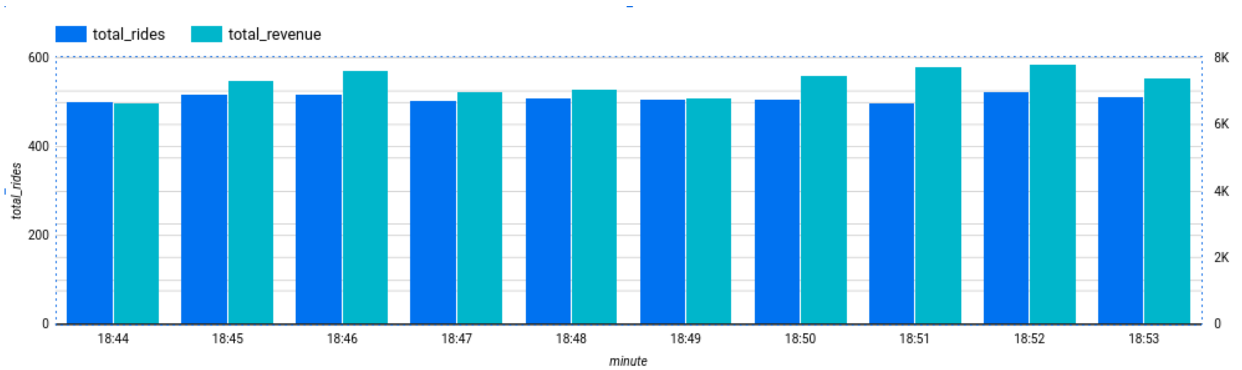


Row	minute	total_rides	total_passengers	total_revenue
1	18:41	1	1	0.0
2	18:42	7	8	16.67
3	18:43	216	394	3119.669997299...
4	18:44	500	886	6613.959986399...
5	18:45	515	951	7320.760000299...
6	18:46	515	943	7597.280007299...
7	18:47	501	925	6952.530010999...
8	18:48	508	935	7016.660013499...
9	18:49	505	892	6762.060006499...
10	18:50	507	926	7477.959996799...

Amelia Miner
11/20/23

16. Data Visualization

- From the query results of the prior query, click on "Explore Data" and bring up the query results in Looker Studio.
  Create a column chart that plots time in minutes over ascending time on the x axis and the total number of rides and revenue on the y axis.
  I excluded the first 3 minutes since the data seemed to taper up from them then sit pretty steady for the rest of the run.



17. Clean up

- Delete the BigQuery table, the dataset, and the bucket.
  ✅