

Week 9 Lab
Submitted by: Shrikrishna Bhat

Table of Contents

9.1g BigQuery, BigLake	2
Card 3 Create dataset.....	2
Card 4. Query data	3
Card 9 – Query Data.....	7
9.2g Jupyter Notebooks	8
Card 3 BigQuery query	8
Card 6 Run queries	8
Card 8 Mobility.....	9
Card 9 Airport Traffic	10
Card 10 Mortality	11
Card 11 – Run Queries	12
Card 12 Write queries	14
9.3g Dataproc Setup	16
Card 6 – Run Computation	16
9.4g Dataflow.....	18
Card 3 Beam Code	18
Card 4 - Run pipe locally	19
Card 5 – Dataflow Lab #2 (Word Count).....	20
Card 6 Run Code locally	21
Card 9 Run Code using Dataflow runner	22
Card 12 View raw data from PubSub.....	23
Card 14 Run Dataflow job from template	24
Card 15 Query data in BigQuery	25
Card 16 – Data visualization.....	28

9.1g BigQuery, BigLake

Card 3 Create dataset

9.1.1 Take a screenshot of the table's details that includes the number of rows in the table.

The screenshot shows the Google Cloud BigQuery interface. On the left, the sidebar lists various services like Analysis, SQL workspace, Data transfers, Scheduled queries, Analytics Hub, Dataform, Partner Center, Migration, Assessment, SQL translation, Administration, Monitoring, Capacity management, BI Engine, and Policy tags. The main area displays the 'Explorer' tab, which shows a tree view of workspaces and datasets. Under the 'cloud-Shrikrishna-shbhat' workspace, the 'yob' dataset is expanded, and the 'yob_native_table' table is selected. A modal window titled 'yob_native_table' provides detailed information about the table. The 'DETAILS' tab is active, showing the following details:

Created	May 30, 2023, 3:29:58 PM UTC-7
Last modified	May 30, 2023, 3:29:58 PM UTC-7
Table expiration	NEVER
Data location	us-west1
Default collation	
Default rounding mode	ROUNDING_MODE_UNSPECIFIED
Case insensitive	false
Description	
Labels	
Primary key(s)	

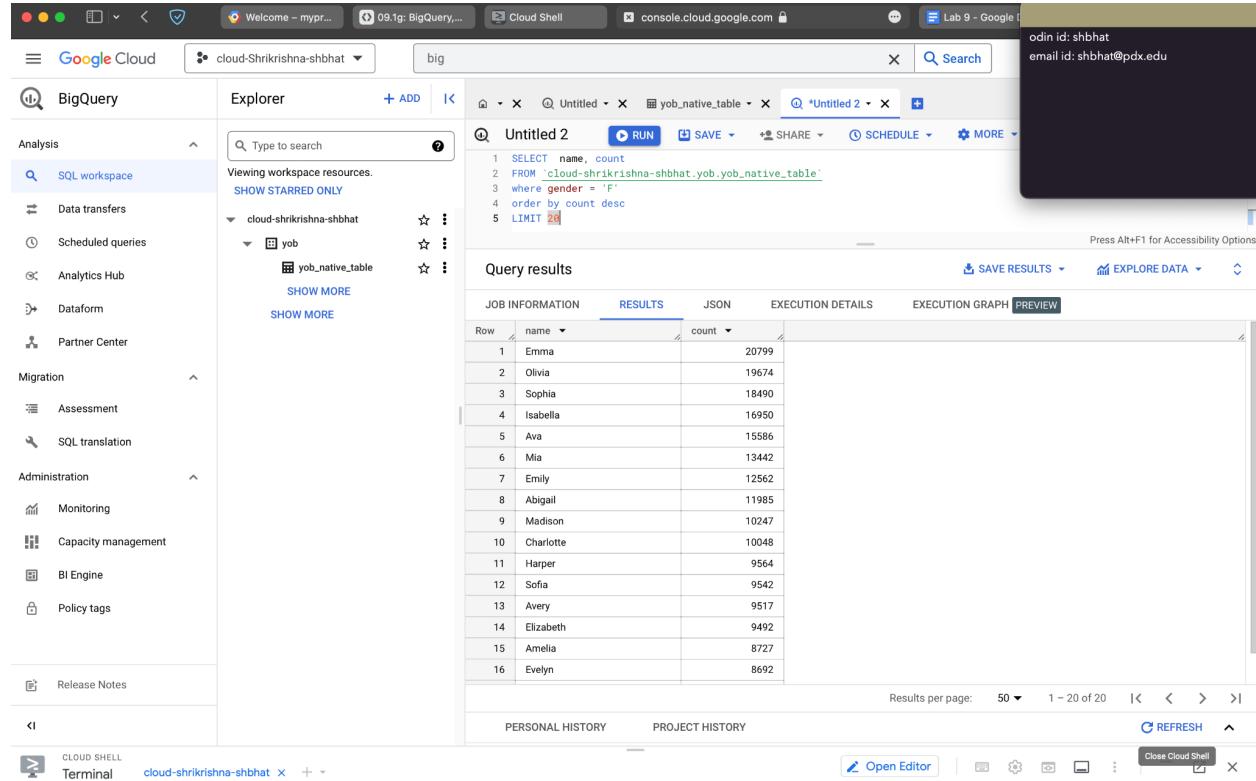
Below the details, the 'Storage info' section shows the following statistics:

Number of rows	33,044
Total logical bytes	618.78 KB
Active logical bytes	618.78 KB
Long term logical bytes	0 B
Total physical bytes	0 B
Active physical bytes	0 B
Long term physical bytes	0 B
Time travel physical bytes	0 B

A tooltip in the top right corner displays user information: 'odin id: shbhat' and 'email id: shbhat@pdx.edu'. At the bottom, there are tabs for 'PERSONAL HISTORY' and 'PROJECT HISTORY', along with a 'REFRESH' button. The bottom navigation bar includes 'CLOUD SHELL' and 'Terminal' tabs, with 'cloud-shrikrishna-shbhat' selected.

Card 4. Query data

9.1.2 Screenshot the query results and include it in your lab notebook



The screenshot shows the Google Cloud BigQuery interface. On the left, the sidebar lists various services: Analysis, SQL workspace, Data transfers, Scheduled queries, Analytics Hub, Dataform, Partner Center, Migration, Assessment, SQL translation, Administration, Monitoring, Capacity management, BI Engine, Policy tags, and Release Notes. The main area displays a query editor titled "Untitled 2" with the following SQL code:

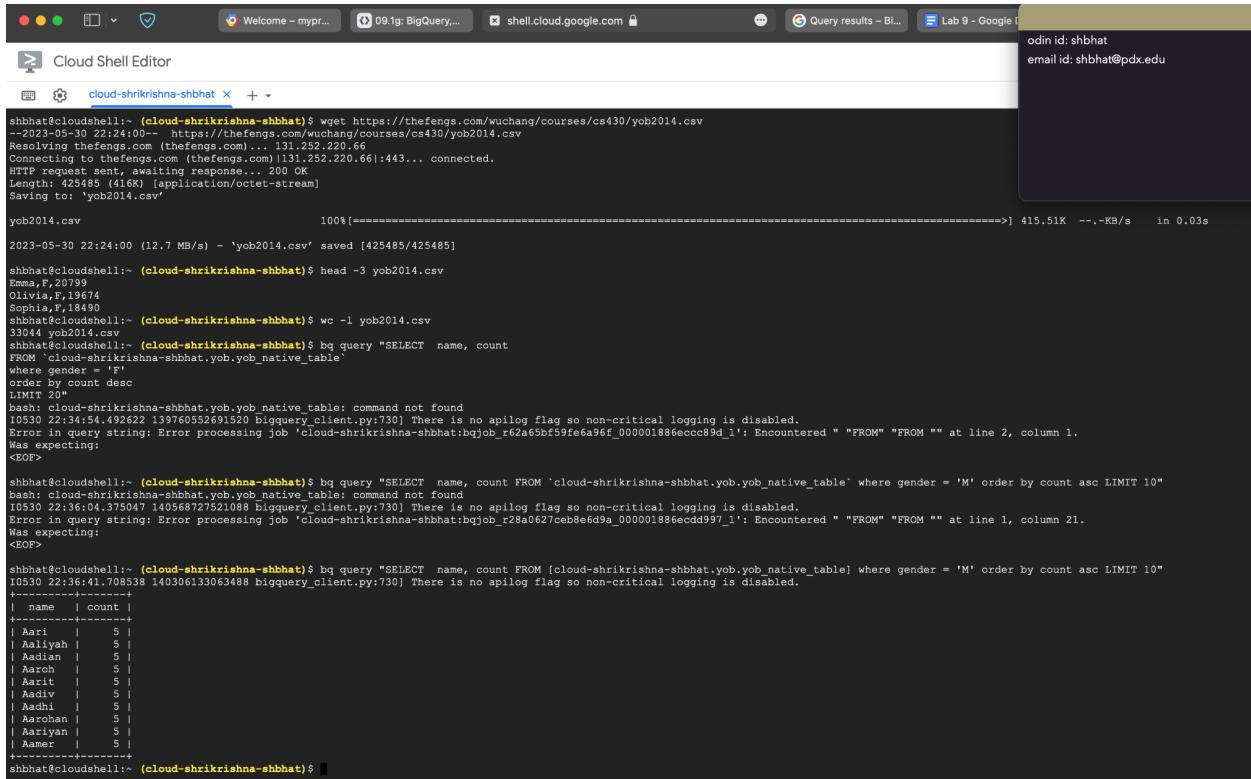
```
1 SELECT name, count
2 FROM `cloud-shrkrishna-shbhat.yob.yob_native_table`
3 WHERE gender = 'F'
4 ORDER BY count DESC
5 LIMIT 20
```

The results table shows the top 20 female names and their counts:

Row	name	count
1	Emma	20799
2	Olivia	19674
3	Sophia	18490
4	Isabella	16950
5	Ava	15586
6	Mia	13442
7	Emily	12562
8	Abigail	11985
9	Madison	10247
10	Charlotte	10048
11	Harper	9564
12	Sofia	9542
13	Avery	9517
14	Elizabeth	9492
15	Amelia	8727
16	Evelyn	8692

At the bottom, there are buttons for "PERSONAL HISTORY" and "PROJECT HISTORY", and a "REFRESH" button.

9.1.3 Screenshot your results and include it in your lab notebook



```
Cloud Shell Editor
cloud-shrikrishna-shbhat ~ + ~

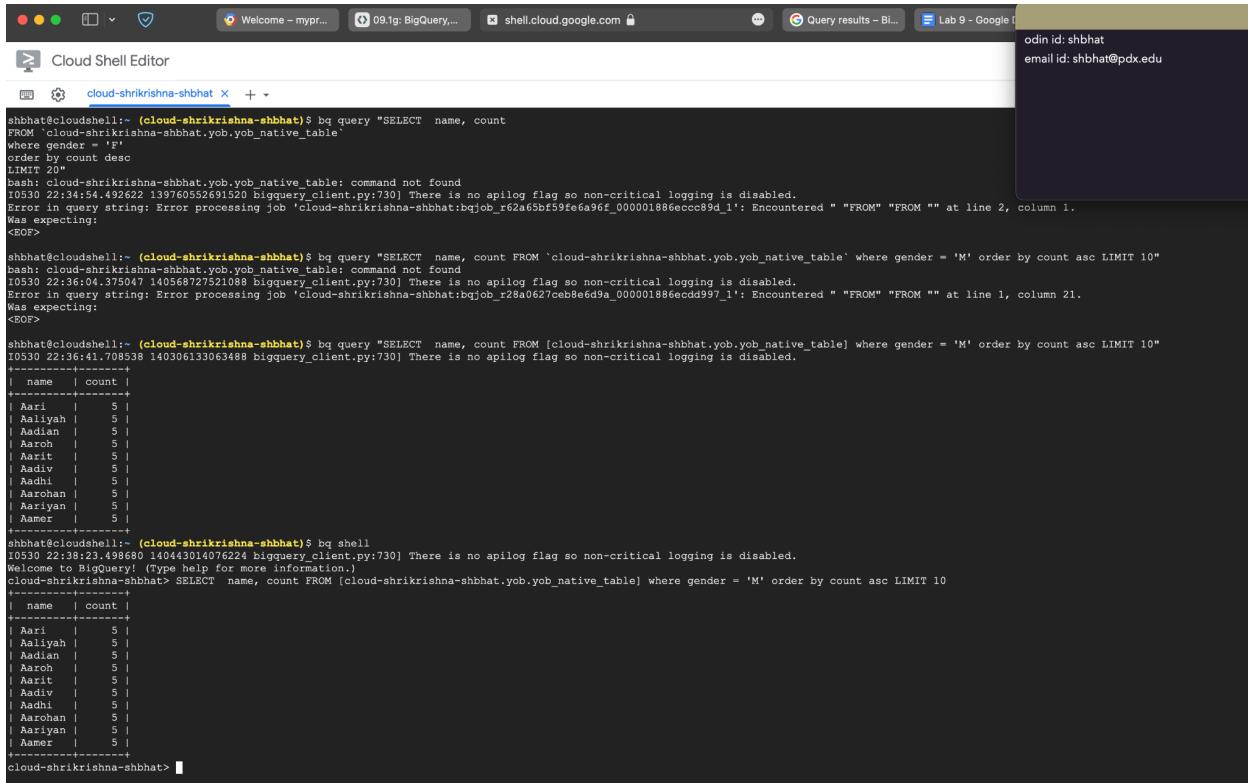
shbhat@cloudshell:~ (cloud-shrikrishna-shbhat)$ wget https://thefengs.com/wuchang/courses/cs430/yob2014.csv
--2023-05-30 22:24:00-- https://thefengs.com/wuchang/courses/cs430/yob2014.csv
Resolving thefengs.com (thefengs.com)... 131.252.220.66
Connecting to thefengs.com (thefengs.com)|131.252.220.66|:443...
HTTP request sent, awaiting response... 200 OK
Length: 425485 (416K) [application/octet-stream]
Saving to: 'yob2014.csv'

yob2014.csv          100%[=====] 415.51K --.-KB/s   in 0.03s

2023-05-30 22:24:00 (12.7 MB/s) - `yob2014.csv' saved [425485/425485]

shbhat@cloudshell:~ (cloud-shrikrishna-shbhat)$ head -3 yob2014.csv
Emma,F,20799
Olivia,F,18674
Sophia,F,18490
shbhat@cloudshell:~ (cloud-shrikrishna-shbhat)$ wc -l yob2014.csv
33044 yob2014.csv
shbhat@cloudshell:~ (cloud-shrikrishna-shbhat)$ bq query "SELECT name, count
FROM `cloud-shrikrishna-shbhat.yob.yob_native_table`
WHERE gender = 'F'
ORDER BY count DESC
LIMIT 20"
bq: command not found
shbhat@cloudshell:~ (cloud-shrikrishna-shbhat)yob.yob_native_table: command not found
10530 22:34:54.492622 139760552691520 bigquery_client.py:730] There is no apilog flag so non-critical logging is disabled.
Error in query string: Error processing job 'cloud-shrikrishna-shbhat:bqjob_r62a65bf59fe6a95f_000001886ecc89d_1': Encountered " "FROM" "FROM "" at line 2, column 1.
Was expecting:
<EOF>
shbhat@cloudshell:~ (cloud-shrikrishna-shbhat)$ bq query "SELECT name, count FROM `cloud-shrikrishna-shbhat.yob.yob_native_table` WHERE gender = 'M' ORDER BY count ASC LIMIT 10"
bq: command not found
shbhat@cloudshell:~ (cloud-shrikrishna-shbhat)yob.yob_native_table: command not found
10530 22:36:41.708538 140306133063488 bigquery_client.py:730] There is no apilog flag so non-critical logging is disabled.
Error in query string: Error processing job 'cloud-shrikrishna-shbhat:bqjob_r28a0627ceb8e6d9a_000001886ecd997_1': Encountered " "FROM" "FROM "" at line 1, column 21.
Was expecting:
<EOF>
shbhat@cloudshell:~ (cloud-shrikrishna-shbhat)$ bq query "SELECT name, count FROM [cloud-shrikrishna-shbhat.yob.yob_native_table] WHERE gender = 'M' ORDER BY count ASC LIMIT 10"
+-----+
| name | count |
+-----+
| Hari | 5 |
| Haliyah | 5 |
| Aadian | 5 |
| Aaroh | 5 |
| Aarit | 5 |
| Aadiv | 5 |
| Aadhi | 5 |
| Aashishan | 5 |
| Aaryan | 5 |
| Namee | 5 |
+-----+
shbhat@cloudshell:~ (cloud-shrikrishna-shbhat)$
```

9.1.4 Screenshot your results and include it in your lab notebook

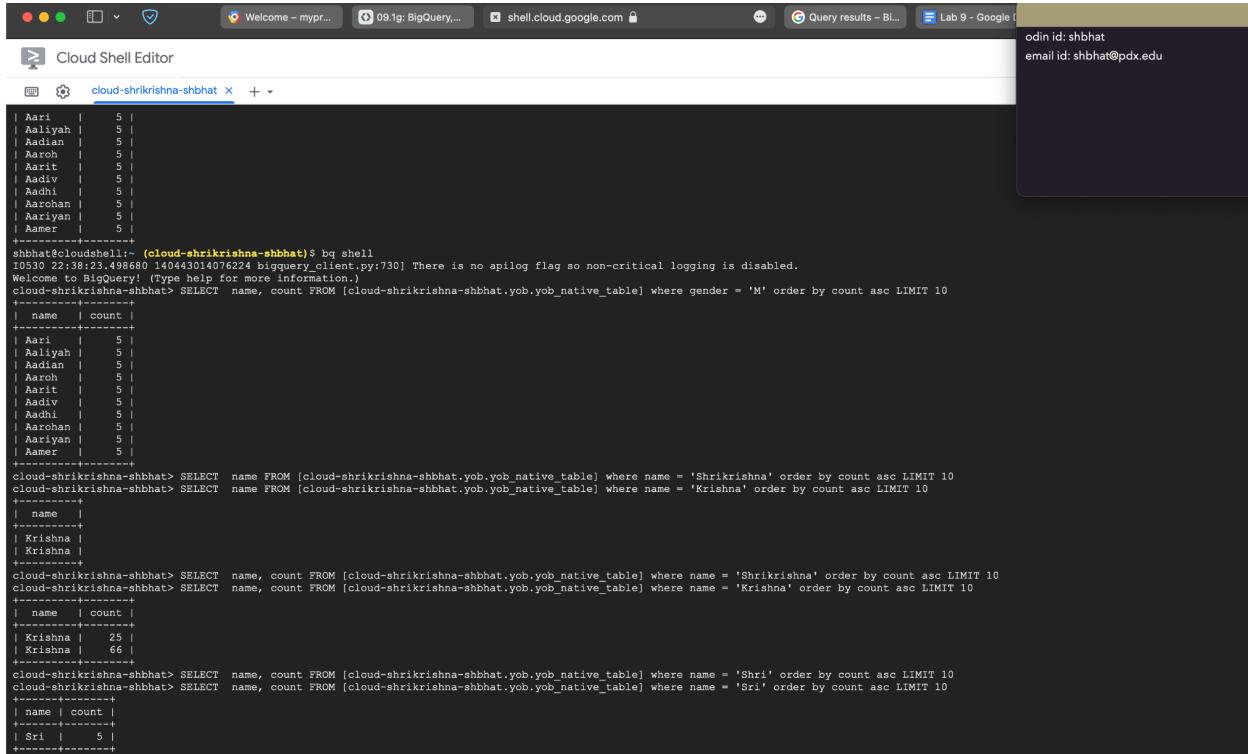


The screenshot shows a Cloud Shell Editor window with a terminal session. The terminal output is as follows:

```
shbhat@cloudshell:~ (cloud-shrikrishna-shbhat)$ bq query "SELECT name, count FROM `cloud-shrikrishna-shbhat.yob.yob_native_table` WHERE gender = 'M' ORDER BY count DESC LIMIT 20"
bash: cloud-shrikrishna-shbhat.yob.yob_native_table: command not found
10530 22:34:54.492622 139760552691520 bigquery_client.py:730] There is no apilog flag so non-critical logging is disabled.
Error in query string: Error processing job `cloud-shrikrishna-shbhat:bqjob_r62a65bf59fe6a9ef_000001886eccca89d_1`: Encountered "" "FROM" "FROM "" at line 2, column 1.
Was expecting:
<EOF>
shbhat@cloudshell:~ (cloud-shrikrishna-shbhat)$ bq query "SELECT name, count FROM `cloud-shrikrishna-shbhat.yob.yob_native_table` WHERE gender = 'M' ORDER BY count ASC LIMIT 10"
bash: cloud-shrikrishna-shbhat.yob.yob_native_table: command not found
10530 22:36:04.375047 140568727521088 bigquery_client.py:730] There is no apilog flag so non-critical logging is disabled.
Error in query string: Error processing job `cloud-shrikrishna-shbhat:bqjob_r28a0627ceb8e6d9a_000001886ecdd997_1`: Encountered "" "FROM" "FROM "" at line 1, column 21.
Was expecting:
<EOF>
shbhat@cloudshell:~ (cloud-shrikrishna-shbhat)$ bq query "SELECT name, count FROM [cloud-shrikrishna-shbhat.yob.yob_native_table] WHERE gender = 'M' ORDER BY count ASC LIMIT 10"
10530 22:36:41.708538 140306133063488 bigquery_client.py:730] There is no apilog flag so non-critical logging is disabled.
+-----+
| name | count |
+-----+
| Aari | 5 |
| Aaliyah | 5 |
| Aadilan | 5 |
| Aaroh | 5 |
| Aarit | 5 |
| Aadiv | 5 |
| Aadhi | 5 |
| Aarohan | 5 |
| Aariyan | 5 |
| Aamer | 5 |
+-----+
shbhat@cloudshell:~ (cloud-shrikrishna-shbhat)$ bq shell
10530 22:39:04.498680 140443014076224 bigquery_client.py:730] There is no apilog flag so non-critical logging is disabled.
Welcome to BigQuery! (Type help for more information.)
cloud-shrikrishna-shbhat> SELECT name, count FROM [cloud-shrikrishna-shbhat.yob.yob_native_table] WHERE gender = 'M' ORDER BY count ASC LIMIT 10
+-----+
| name | count |
+-----+
| Aari | 5 |
| Aaliyah | 5 |
| Aadilan | 5 |
| Aaroh | 5 |
| Aarit | 5 |
| Aadiv | 5 |
| Aadhi | 5 |
| Aarohan | 5 |
| Aariyan | 5 |
| Aamer | 5 |
+-----+
cloud-shrikrishna-shbhat>
```

In the top right corner of the terminal window, there is a sidebar with user information: odin id: shbhat and email id: shbhat@pdx.edu.

9.1.5 Screenshot your results and include it in your lab notebook



The screenshot shows a Cloud Shell Editor window with a dark theme. At the top, there are several tabs: "Welcome - mypr...", "09.1g: BigQuery...", "shell.cloud.google.com", "Query results - Bl...", and "Lab 9 - Google". On the right side of the editor, there is a sidebar with the text "odin id: shbhat" and "email id: shbhat@pdx.edu". The main area of the editor displays the following BigQuery session output:

```
shbhat@cloudshell:~ (cloud-shrikrishna-shbhat)$ bq shell
I0530 22:38:23.498680 140443014076224 bigquery_client.py:730] There is no apilog flag so non-critical logging is disabled.
Welcome to BigQuery! (Type help for more information.)
cloud-shrikrishna-shbhat> SELECT name, count FROM [cloud-shrikrishna-shbhat.yob.yob_native_table] where gender = 'M' order by count asc LIMIT 10
+-----+
| name | count |
+-----+
| Aari | 5 |
| Asaliyah | 5 |
| Aadian | 5 |
| Aaroh | 5 |
| Aarit | 5 |
| Aadv | 5 |
| Aadhi | 5 |
| Aarohan | 5 |
| Aariyan | 5 |
| Aame | 5 |
+-----+
cloud-shrikrishna-shbhat> SELECT name FROM [cloud-shrikrishna-shbhat.yob.yob_native_table] where name = 'Shrikrishna' order by count asc LIMIT 10
cloud-shrikrishna-shbhat> SELECT name FROM [cloud-shrikrishna-shbhat.yob.yob_native_table] where name = 'Krishna' order by count asc LIMIT 10
+-----+
| name |
+-----+
| Krishna |
| Krishna |
+-----+
cloud-shrikrishna-shbhat> SELECT name, count FROM [cloud-shrikrishna-shbhat.yob.yob_native_table] where name = 'Shrikrishna' order by count asc LIMIT 10
cloud-shrikrishna-shbhat> SELECT name, count FROM [cloud-shrikrishna-shbhat.yob.yob_native_table] where name = 'Krishna' order by count asc LIMIT 10
+-----+
| name | count |
+-----+
| Krishna | 29 |
| Krishna | 66 |
+-----+
cloud-shrikrishna-shbhat> SELECT name, count FROM [cloud-shrikrishna-shbhat.yob.yob_native_table] where name = 'Shri' order by count asc LIMIT 10
cloud-shrikrishna-shbhat> SELECT name, count FROM [cloud-shrikrishna-shbhat.yob.yob_native_table] where name = 'Sri' order by count asc LIMIT 10
+-----+
| name | count |
+-----+
| Sri | 5 |
+-----+
```

Card 9 – Query Data

9.1.6 Screenshot the query results and include it in your lab notebook

The screenshot shows the Google Cloud BigQuery interface. On the left, the sidebar navigation includes Analysis, SQL workspace (selected), Data transfers, Scheduled queries, Analytics Hub, Dataform, Partner Center, Migration, Assessment, SQL translation, Administration (Monitoring, Capacity management, BI Engine, Policy tags), and Release Notes. The main area displays a query editor titled "Untitled 3" with the following SQL code:

```
1 SELECT name, count
2 FROM `cloud-shrirkrishna-shbhat.yob.yob_biglake_table`
3 where gender = 'F'
4 order by count asc
5 LIMIT 20
```

The results pane shows a table with columns "Row" and "name". The data is as follows:

Row	name
5	Ava
6	Mia
7	Emily
8	Abigail
9	Madison
10	Charlotte
11	Harper
12	Sofia
13	Avery
14	Elizabeth
15	Amelia
16	Evelyn

Below the results table, there are buttons for "SAVE RESULTS" and "EXPLORE DATA". The bottom of the interface shows tabs for "PERSONAL HISTORY" and "PROJECT HISTORY", along with a "REFRESH" button.

9.2g Jupyter Notebooks

Card 3 BigQuery query

9.2.1 How much less data does this query process compared to the size of the table?

This query will run with 18 gb less data

9.2.2 How many twins were born during this time range?

375362

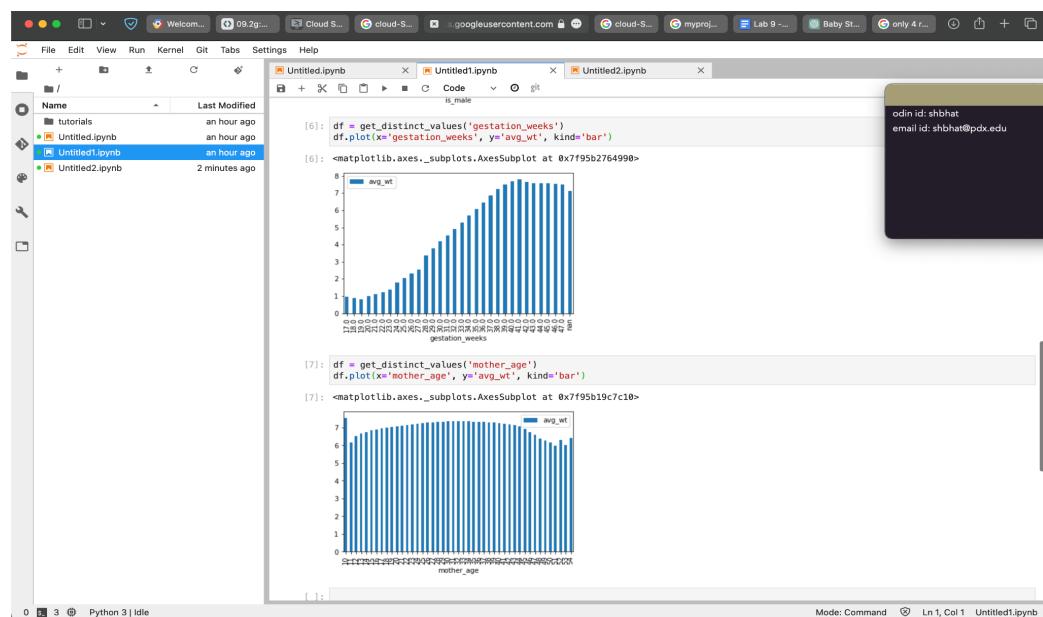
9.2.3 How much lighter on average are they compared to single babies?

2.17

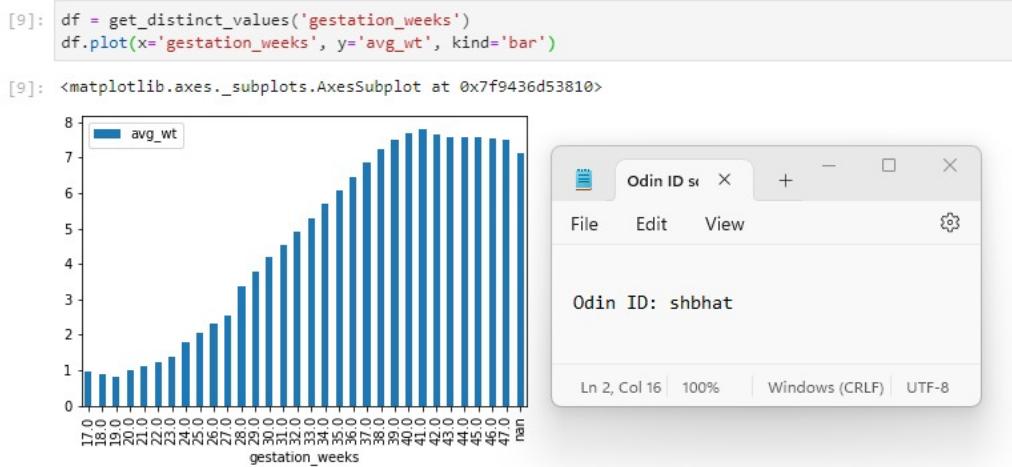
Card 6 Run queries

9.2.4 Show the plots generated for the two most important features for your lab notebook

Gestation_week and plurality



Initially had taken wrong screenshot hence had to run again and take screenshot so the screenshot differs from the rest.



Card 8 Mobility

9.2.5 What day saw the largest spike in trips to grocery and pharmacy stores? Answer – 2020-03-13

SANDBOX Set up billing to upgrade to the full BigQuery experience. [Learn more](#)

Explorer + ADD ↗

Untitled RUN SAVE SHARE SCHEDULE MORE

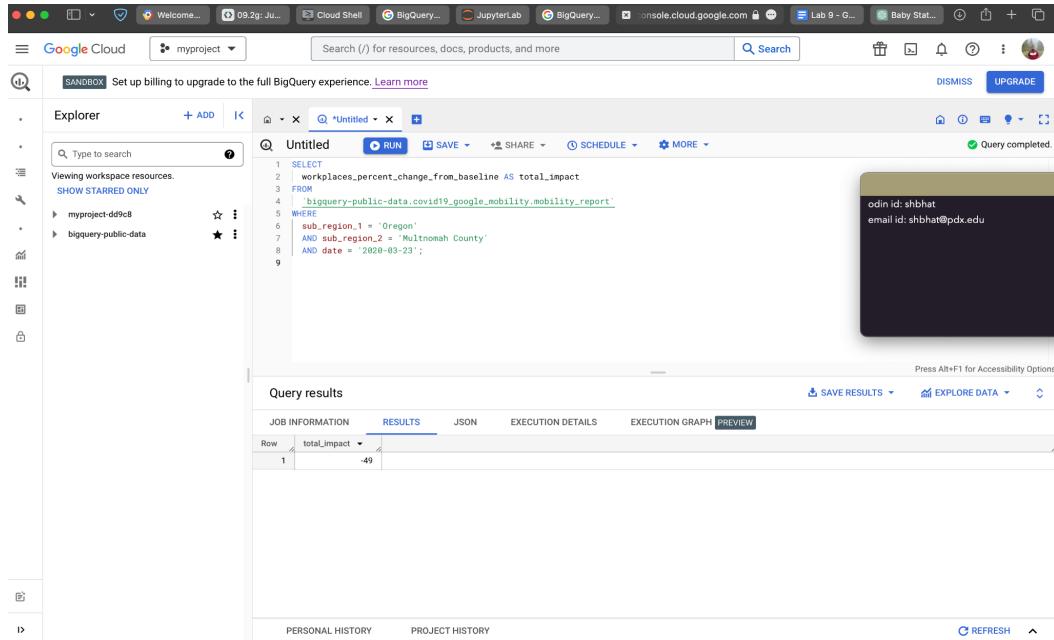
```
1 SELECT
2   date,
3   MAX(grocery_and_pharmacy_percent_change_from_baseline) AS max_spike
4   FROM
5     `bigquery-public-data.covid19_google_mobility.mobility_report`
6   WHERE
7     sub_region_1 = 'Oregon'
8     AND sub_region_2 = 'Multnomah County'
9     AND date BETWEEN '2020-03-01' AND '2020-03-31'
10 GROUP BY
11   date
12 ORDER BY
13   max_spike DESC
14 LIMIT 1;
```

odin id: shbhat
email id: shbhat@pdx.edu

Query results

Row	date	max_spike
1	2020-03-13	17

9.2.6 On the day the stay-at-home order took effect (3/23/2020), what was the total impact on workplace trips? Answer: -49



The screenshot shows the Google Cloud BigQuery interface. In the top navigation bar, there are tabs for 'Welcome...', 'Cloud Shell', 'BigQuery...', 'JupyterLab', 'BigQuery...', 'console.cloud.google.com', 'Lab 9 - G...', 'Baby Stat...', and a user icon. Below the navigation bar, there's a search bar and a 'DISMISS' button. The main area has a sidebar titled 'Explorer' with sections for 'myproject' and 'bigquery-public-data'. A query editor window titled 'Untitled' is open, showing the following SQL code:

```
1 SELECT
2   `workplaces_percent_change_from_baseline` AS total_impact
3 FROM
4   `bigquery-public-data.covid19_google_mobility.mobility_report`
5 WHERE
6   sub_region_1 = 'Oregon'
7   AND sub_region_2 = 'Multnomah County'
8   AND date = '2020-03-23';
9
```

To the right of the code editor, a preview pane displays the results of the query:

```
odin id: shbhat
email id: shbhat@pdx.edu
```

Below the code editor, the 'Query results' section shows a table with one row:

Row	total_impact
1	-49

Card 9 Airport Traffic

9.2.7 Which three airports were impacted the most in April 2020 (the month when lockdowns became widespread)?

Newark Liberty International

Daniel K. Inouye International

Chicago O'Hare International

9.2.8 Run the query again using the month of August 2020. Which three airports were impacted the most?

Newark Liberty International

Charlotte Douglas International

Dallas/Fort Worth International

Card 10 Mortality

9.2.10 What table and columns identify the place name, the starting date, and the number of excess deaths from COVID-19?

The table that identifies the place name, starting date, and number of excess deaths from COVID-19 is the "excess_deaths" table. The relevant columns are:

placename: The place in the country reported

start_date: The first date included in the period

excess_deaths: The number of deaths minus the expected deaths

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

The table that identifies the date, county, and deaths from COVID-19 is the "us_counties" table. The relevant columns are:

date: Date reported

county: County in the specified state

deaths: The total number of confirmed deaths of COVID-19

9.2.12 What table and columns identify the date, state, and confirmed cases of COVID-19?

The table that identifies the date, state, and confirmed cases of COVID-19 is the "us_states" table. The relevant columns are:

date: Date reported

state_name: State reported

confirmed_cases: The total number of confirmed cases of COVID-19

9.2.13 What table and columns identify a county code and the percentage of its residents that report they always wear masks?

The table that identifies a county code and the percentage of its residents that report they always wear masks is the "mask_use_by_county" table. The relevant columns are:

`county_fips_code`: Standard geographic identifier for the county

always: The estimated share of people in this county who would say always in response to the mask-wearing question

Card 11 – Run Queries

9.2.14 Show a screenshot of the plot and the code used to generate it for your lab notebook

The screenshot shows a Jupyter Notebook interface with three tabs open: Untitled.ipynb, Untitled1.ipynb, and Untitled2.ipynb. The Untitled2.ipynb tab is active, displaying the following code:

```
[2]: query_string = """
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 + " LIMIT 100").to_dataframe()
df.head()

[3]:   date  confirmed_cases
0 2020-02-28      1
1 2020-02-29      1
2 2020-03-01      2
3 2020-03-02      2
4 2020-03-03      2

[5]: df.plot(x='date', y='confirmed_cases', kind='line')

[5]: <matplotlib.axes._subplots.AxesSubplot at 0x7f9c405ce250>
```

A line plot titled "confirmed_cases" is displayed, showing the number of confirmed COVID-19 cases in Oregon from March 2020 to June 2020. The x-axis represents the date, and the y-axis represents the number of cases, ranging from 0 to 4,000. The plot shows a steady increase in cases over time.

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

The screenshot shows a Jupyter Notebook interface with several tabs open. The current tab is 'Untitled2.ipynb'. The code in the cell [25] is a SQL query to select state names and their first date of reaching 1000 deaths from the 'bigquery-public-data.covid19_nyt.us_states' dataset. The code in cell [26] imports the 'bigquery' module and runs the query with a limit of 10 rows. The resulting DataFrame is displayed, showing the top 10 states. Cell [8] shows a similar query but with a limit of 100 rows, resulting in a shorter list of states. A sidebar on the right displays user information: 'odin id: shbhat' and 'email id: shbhat@pdx.edu'.

```
[25]: query_string = """
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
"""

[26]: from google.cloud import bigquery
df = bigquery.Client().query(query_string + " LIMIT 10").to_dataframe()
df.head(10)

[26]:   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 Pennsylvania  2020-04-17
7 Connecticut  2020-04-17
8    California  2020-04-17
9      Florida  2020-04-24

[8]: from google.cloud import bigquery
df = bigquery.Client().query(query_string + " LIMIT 100").to_dataframe()
df.head()

[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
```

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

The screenshot shows a Jupyter Notebook interface with three tabs: Untitled.ipynb, Untitled1.ipynb, and Untitled2.ipynb. The Untitled2.ipynb tab is active, displaying the following Python code:

```
[8]: df = bigquery.Client().query(query_string + " LIMIT 100").to_dataframe()
df.head()

[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
```

[15]: query_string = """
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
"""

[16]: from google.cloud import bigquery
df = bigquery.Client().query(query_string + " LIMIT 5").to_dataframe()
df.head()

[16]: county_fips_code always county
0 06027 0.889 Inyo
1 36123 0.884 Yates
2 06051 0.880 Mono
3 48229 0.880 Hudspeth
4 48141 0.877 El Paso

The right pane shows the results of the last command, which is a DataFrame with columns: county_fips_code, always, and county. The data is as follows:

county_fips_code	always	county
06027	0.889	Inyo
36123	0.884	Yates
06051	0.880	Mono
48229	0.880	Hudspeth
48141	0.877	El Paso

Card 12 Write queries

9.2.17 Plot the results and take a screenshot for your lab notebook.

The screenshot shows a Jupyter Notebook interface with three tabs: Untitled.ipynb, Untitled1.ipynb, and Untitled2.ipynb. The Untitled2.ipynb tab is active, displaying the following Python code:

```
[17]: query_string = """
SELECT date, deaths
FROM `bigquery-public-data.covid19_nyt.us_counties`
WHERE county = 'Multnomah' AND state_name = 'Oregon'
ORDER BY date ASC
"""

[18]: from google.cloud import bigquery
df = bigquery.Client().query(query_string + " LIMIT 100").to_dataframe()
df.head()
```

[18]: date deaths
0 2020-03-10 0
1 2020-03-11 0
2 2020-03-12 0
3 2020-03-13 0
4 2020-03-14 1

[20]: df.plot(x='date', y='deaths', kind='line')

[20]: <matplotlib.axes._subplots.AxesSubplot at 0x7f9c400ce3d0>

The right pane shows a line plot titled "deaths" with the x-axis labeled "date" and the y-axis ranging from 0 to 70. The plot shows a sharp increase starting around March 12, 2020, reaching approximately 65 deaths by March 15, 2020.

9.2.18 Plot the results and take a screenshot for your lab notebook.

The screenshot shows a Jupyter Notebook interface with several tabs open. The current tab is 'Untitled2.ipynb'. The code cell [21] contains a SQL query to select 'date' and 'deaths' from the 'bigquery-public-data.covid19_nyt.us_states' table where the state name is 'Oregon' and the date is greater than or equal to '2020-03-19'. The code cell [22] imports the 'bigquery' module and runs the query, then converts the result to a DataFrame and prints its head. The code cell [23] plots the 'deaths' column against 'date' using a line plot. The resulting plot shows a sharp increase in deaths starting around March 20, 2020, reaching approximately 150 by June 1, 2020.

```
[21]: query_string = """
SELECT date, deaths
FROM `bigquery-public-data.covid19_nyt.us_states`
WHERE state_name = 'Oregon'
ORDER BY date ASC
"""

[22]: from google.cloud import bigquery
df = bigquery.Client().query(query_string + " LIMIT 100").to_dataframe()
df.head()

[23]: df.plot(x='date', y='deaths', kind='line')
<matplotlib.axes._subplots.AxesSubplot at 0x7f9c40306dd0>
```

date	deaths
2020-02-28	0
2020-02-29	0
2020-03-01	0
2020-03-02	0
2020-03-03	0

A line plot titled 'deaths' showing the number of deaths over time. The x-axis represents dates from March 19, 2020, to June 1, 2020. The y-axis represents the number of deaths, ranging from 0 to 160. The plot shows a steady increase in deaths, starting near zero and reaching approximately 150 by June 1, 2020.

9.3g Dataproc Setup

Card 6 – Run Computation

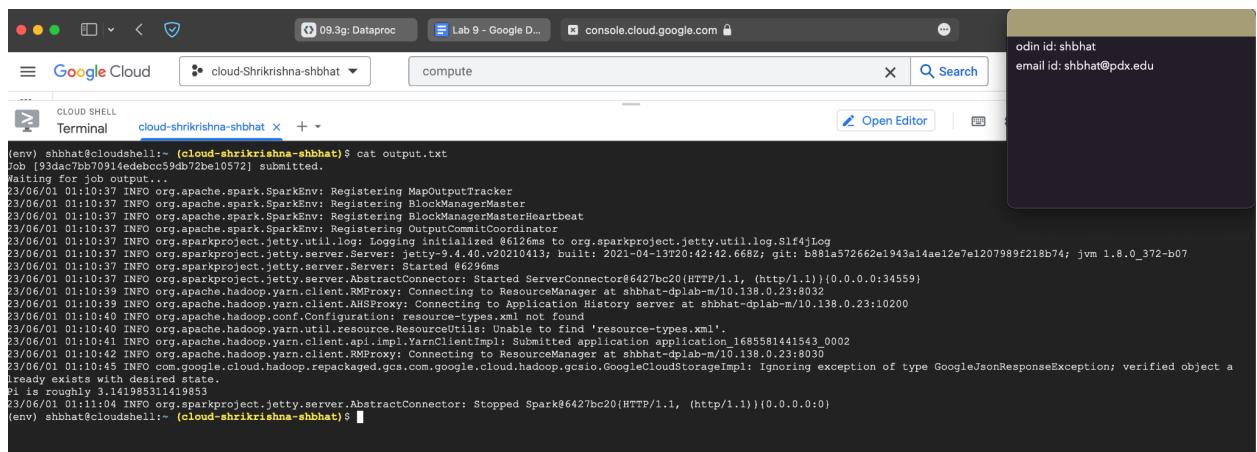
9.3.1 For your lab notebook:

- How long did the job take to execute?

About 30s

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

Pi is roughly **3.141985311419853**



```
(env) shbhat@cloudshell:~ (cloud-shrikrishna-shbhat)$ cat output.txt
Job (93dac7b70914edebcc59db72be10572) submitted.
Waiting for job output...
23/06/01 01:10:37 INFO org.apache.spark.SparkEnv: Registering MapOutputTracker
23/06/01 01:10:37 INFO org.apache.spark.SparkEnv: Registering BlockManagerMaster
23/06/01 01:10:37 INFO org.apache.spark.SparkEnv: Registering BlockManagerMasterHeartbeat
23/06/01 01:10:37 INFO org.apache.spark.SparkEnv: Registering ApplicationMaster
23/06/01 01:10:37 INFO org.apache.spark.SparkEnv: Registering ApplicationMasterHeartbeat
23/06/01 01:10:37 INFO org.apache.spark.project.jetty.util.log.Logging: Logging initialized @6126ms to org.sparkproject.jetty.util.log.Slf4jLog
23/06/01 01:10:37 INFO org.sparkproject.jetty.server.Server: jetty-9.4.40.v20210413; built: 2021-04-13T20:42:42.668Z; git: b881a572662e1943a14ae12e7e1207989f218b74; jvm 1.8.0_372-b07
23/06/01 01:10:37 INFO org.sparkproject.jetty.server.Server: Started @6296ms
23/06/01 01:10:37 INFO org.sparkproject.jetty.server.AbstractConnector: Started ServerConnector@6427bc20{HTTP/1.1, (http/1.1)}{0.0.0.0:34559}
23/06/01 01:10:39 INFO org.apache.hadoop.yarn.client.RMProxy: Connecting to ResourceManager at shbhat-dplab-m/10.138.0.23:8032
23/06/01 01:10:39 INFO org.apache.hadoop.yarn.client.AHSProxy: Connecting to Application History server at shbhat-dplab-m/10.138.0.23:10200
23/06/01 01:10:40 INFO org.apache.hadoop.conf.Configuration: resource-types.xml not found
23/06/01 01:10:40 INFO org.apache.hadoop.yarn.util.resource.Resources: Failed to find 'resource-types.xml'.
23/06/01 01:10:41 INFO org.apache.hadoop.yarn.util.ResourceLocalizationImpl: Submitting application application_16855814151493_0002
23/06/01 01:10:42 INFO org.apache.hadoop.yarn.client.RMProxy: Connecting to ResourceManager at shbhat-dplab-m/10.138.0.23:8030
23/06/01 01:10:45 INFO com.google.cloud.hadoop.repackaged.gcs.com.google.cloud.hadoop.GoogleCloudStorageImpl: Ignoring exception of type GoogleJsonResponseException; verified object already exists with desired state.
Pi is roughly 3.141985311419853
23/06/01 01:11:04 INFO org.sparkproject.jetty.server.AbstractConnector: Stopped Spark@6427bc20{HTTP/1.1, (http/1.1)}{0.0.0.0:0}
(env) shbhat@cloudshell:~ (cloud-shrikrishna-shbhat)$
```

Card 8 – Run Computation again

9.3.2 Answer the questions in the lab.

- How long did the job take to execute? How much faster did it take?

It took 11s, it was 19s faster

- Examine **output2.txt** and show the estimate of π calculated.

Pi is roughly **3.1415517914155178**

The screenshot shows a Google Cloud Shell terminal window titled "cloud-Shrikrishna-shbhat". The terminal is displaying log output from a DataProc job. The logs show several job entries, each with a unique ID, type (spark), and status (RUNNING). The log output includes timestamps, command-line inputs, and detailed logs from the Spark environment, such as log4j and YARN client configurations. The terminal interface includes a search bar and an "Open Editor" button.

```
state
JOB_ID: 9bb969ed0fa247c483465195ae068cd7
TYPE: spark
STATUS: RUNNING
Thu 01 Jun 2023 01:07:42 AM UTC
(env) shbhat@cloudshell:~ (cloud-shrikrishna-shbhat)$ gcloud dataproc jobs list --cluster ${CLUSTERNAME}

state
JOB_ID: 9bb969ed0fa247c483465195ae068cd7
TYPE: spark
STATUS: RUNNING
Thu 01 Jun 2023 01:07:46 AM UTC
(env) shbhat@cloudshell:~ (cloud-shrikrishna-shbhat)$ gcloud dataproc jobs list --cluster ${CLUSTERNAME}

state
JOB_ID: 9bb969ed0fa247c483465195ae068cd7
TYPE: spark
STATUS: RUNNING
Thu 01 Jun 2023 01:07:48 AM UTC
(env) shbhat@cloudshell:~ (cloud-shrikrishna-shbhat)$ gcloud dataproc jobs list --cluster ${CLUSTERNAME}

state
JOB_ID: 9bb969ed0fa247c483465195ae068cd7
TYPE: spark
STATUS: RUNNING
Thu 01 Jun 2023 01:07:50 AM UTC
(env) shbhat@cloudshell:~ (cloud-shrikrishna-shbhat)$ cat output
http://10.138.0.23:10200/jetty-0.9.2.v20110203140137
(env) shbhat@cloudshell:~ (cloud-shrikrishna-shbhat)$ cat output2.txt
job [9bb969ed0fa247c483465195ae068cd7] submitted.
Waiting for job output...
3/06/01 01:07:24 INFO org.apache.spark.SparkEnv: Registering MapOutputTracker
3/06/01 01:07:24 INFO org.apache.spark.SparkEnv: Registering BlockManagerMaster
3/06/01 01:07:24 INFO org.apache.spark.SparkEnv: Registering BlockManagerMasterHeartbeat
3/06/01 01:07:24 INFO org.apache.spark.SparkEnv: Registering OutputCommitCoordinator
3/06/01 01:07:25 INFO org.sparkproject.jetty.util.log: Logging initialized @80ms to org.sparkproject.jetty.util.log.Slf4jLog
3/06/01 01:07:25 INFO org.sparkproject.jetty-0.9.2.v20110203140137-Built! 2021-04-13T20:42:42.668Z git: b881a572662e1943a14ae12e7e1207989f218b74; jvm 1.8.0_372-b07
3/06/01 01:07:25 INFO org.sparkproject.jetty.Server: Started @6934ms
3/06/01 01:07:25 INFO org.sparkproject.jetty.server.AbstractConnector: Started ServerConnector@2104b36{HTTP/1.1, (http/1.1)}{0.0.0.0:34107}
3/06/01 01:07:27 INFO org.apache.hadoop.yarn.client.RMProxy: Connecting to ResourceManager at shbhat-dplab-m/10.138.0.23:8032
3/06/01 01:07:27 INFO org.apache.hadoop.yarn.client.AHSProxy: Connecting to Application History server at shbhat-dplab-m/10.138.0.23:10200
3/06/01 01:07:28 INFO org.apache.hadoop.conf.Configuration: resource-types.xml not found
3/06/01 01:07:28 INFO org.apache.hadoop.yarn.util.resource.ResourceUtils: Unable to find 'resource-types.xml'.
3/06/01 01:07:31 INFO org.apache.hadoop.yarn.client.api.impl.YarnClientImpl: Submitted application application_1685581441543_0001
3/06/01 01:07:32 INFO org.apache.hadoop.yarn.client.RMProxy: Connecting to ResourceManager at shbhat-dplab-m/10.138.0.23:8030
3/06/01 01:07:32 INFO org.apache.hadoop.yarn.recovery.HDFSResourceRecoveryService.cloud.hadoop.repackaged.gcs.com.google.cloud.hadoop.gcsio.GoogleCloudStorageImpl: Ignoring exception of type GoogleJsonResponseException; verified object already exists with desired state.
` is roughly 3.1415517914155178
3/06/01 01:07:59 INFO org.sparkproject.jetty.server.AbstractConnector: Stopped Spark@2104b36{HTTP/1.1, (http/1.1)}{0.0.0.0:0}
(env) shbhat@cloudshell:~ (cloud-shrikrishna-shbhat)$
```

9.4g Dataflow

Card 3 Beam Code

9.4.1 Answer the questions

- Where is the input taken from by default?

By default, the input for the script is taken from the
'..javahelp/src/main/java/com/google/cloud/training/dataanalyst/javahelp/' directory. The input files should have a '.java' extension.

- Where does the output go by default?

The output, by default, will be written to the '/tmp/output' directory with a filename generated by Beam.

- Examine both the `getPackages()` function and the `splitPackageName()` function. What operation does the `'PackageUse()'` transform implement?

The 'getPackages()' function takes a line of code and a keyword ('import' in this case) and extracts the Java package names from that line. It uses the 'splitPackageName()' function to split a package name into its individual components. The 'PackageUse()' transform takes a line of code and a keyword and yields a tuple of each package used and a count of 1.

- Look up Beam's `CombinePerKey`. What operation does the `TotalUse` operation implement?

The 'TotalUse' operation, using the 'CombinePerKey' transform, implements the "Reduce" operation. It combines the counts for each package and sums them up based on the package key.

9.4.2 Answer the questions

- Which operations correspond to a "Map"?

Operations that correspond to a "Map" in the pipeline are:

'GetImports': Extracts lines starting with the 'import' keyword.

'PackageUse': Extracts the packages from each line and yields them with a count of 1.

- Which operation corresponds to a "Shuffle-Reduce"?

The operation that corresponds to a "Shuffle-Reduce" is:

'TotalUse': Performs the shuffling of key-value pairs based on the package key and reduces the values by summing them up.

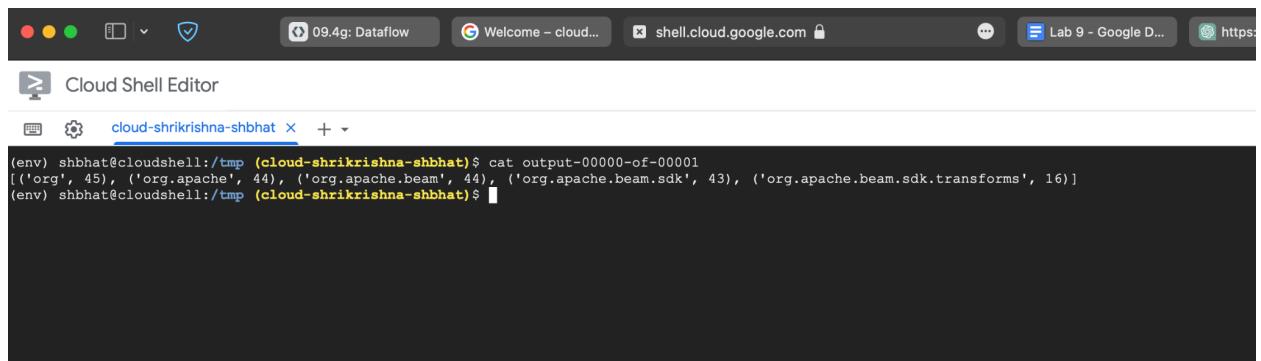
- Which operation corresponds to a "Reduce"?

The operation that corresponds to a "Reduce" is:

'TotalUse': It combines the counts for each package and sums them up based on the package key, reducing the values to a single count for each package.

Card 4 - Run pipe locally

9.4.3 Take a screenshot of its contents



A screenshot of a Mac OS X desktop showing a Cloud Shell Editor window. The window has a dark theme and displays a terminal session. The terminal title is 'cloud-shrikrishna-shbhat'. The command run was 'cat output-00000-of-00001'. The output shows a list of package names and their counts: ('org', 45), ('org.apache', 44), ('org.apache.beam', 44), ('org.apache.beam.sdk', 43), ('org.apache.beam.sdk.transforms', 16). The terminal prompt is '(env) shbhat@cloudshell:~/tmp {cloud-shrikrishna-shbhat}\$'.

```
(env) shbhat@cloudshell:~/tmp {cloud-shrikrishna-shbhat}$ cat output-00000-of-00001
[('org', 45), ('org.apache', 44), ('org.apache.beam', 44), ('org.apache.beam.sdk', 43), ('org.apache.beam.sdk.transforms', 16)]
(env) shbhat@cloudshell:~/tmp {cloud-shrikrishna-shbhat}$
```

9.4.4 Explain what the data in this output file corresponds to based on your understanding of the program.

Based on the program's logic, the output file will contain the top 5 most used Java packages in the input Java code files. Each line in the output file will consist of a package name and its corresponding count, separated by a tab or space.

The program scans the input Java code files, identifies the lines that start with the keyword 'import', and extracts the package names from those lines. It then counts the occurrences of each package and combines the counts using the 'CombinePerKey' transform.

The 'Top_5' transform selects the top 5 packages with the highest counts, based on the count values. Finally, the selected packages and their counts are written to the output file.

Therefore, the output file will provide insights into the most commonly used Java packages in the analyzed codebase, allowing developers to identify frequently utilized dependencies and potentially optimize their code or manage dependencies more effectively.

Card 5 – Dataflow Lab #2 (Word Count)

9.4.5 What are the names of the stages in the pipeline?

Answer is in the below question.

9.4.6 Describe what each stage does.

Both answers are written below where stages and explanation is given

In the given code, the stages in the pipeline are as follows:

Read: Reads the text file specified in the input argument using ReadFromText. Produces a PCollection named lines containing the lines of the input text file.

Split: Applies the WordExtractingDoFn function using ParDo. Splits each line of text into individual words using regular expression matching.

Produces a PCollection of words. PairWithOne: Maps each word to a key-value pair where the word is the key and the value is 1. Assigns an initial count of 1 to each word.

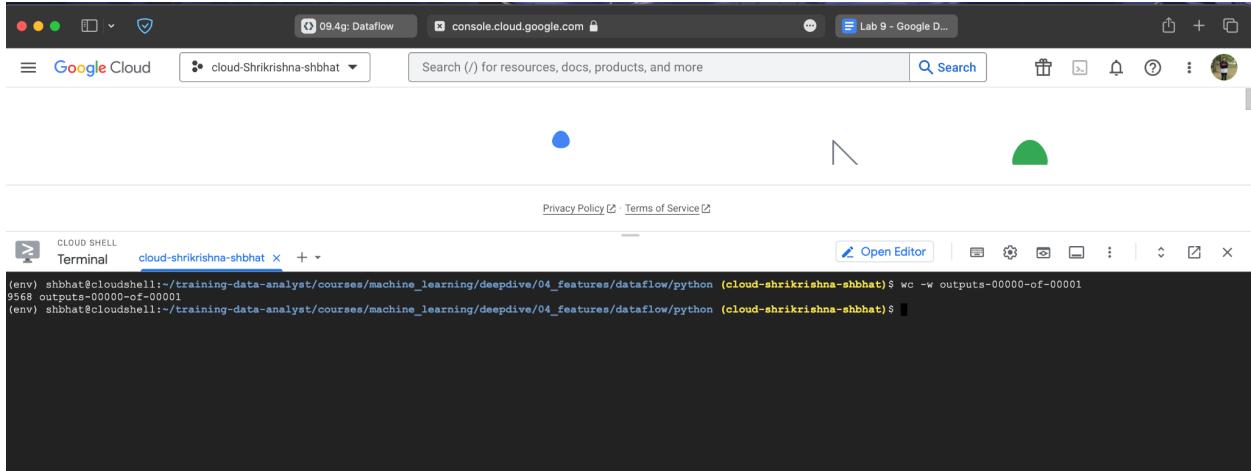
GroupAndSum: Groups the key-value pairs by key (word). Combines the values (counts) for each key (word) using the CombinePerKey transform with the sum function. Produces a PCollection of key-value pairs where the key is a unique word and the value is the sum of counts for that word.

Format: Maps each key-value pair to a formatted string using the format_result function. Formats the word and its count as "%s: %d" where %s is the word and %d is the count. Produces a PCollection of formatted strings.

Write: Writes the output PCollection to the output file specified in the --output argument using the WriteToText transform. Each stage in the pipeline performs a specific transformation or action on the data, such as reading from a text file, splitting lines into words, counting the occurrences of each word, formatting the results, and writing the output to a file

Card 6 Run Code locally

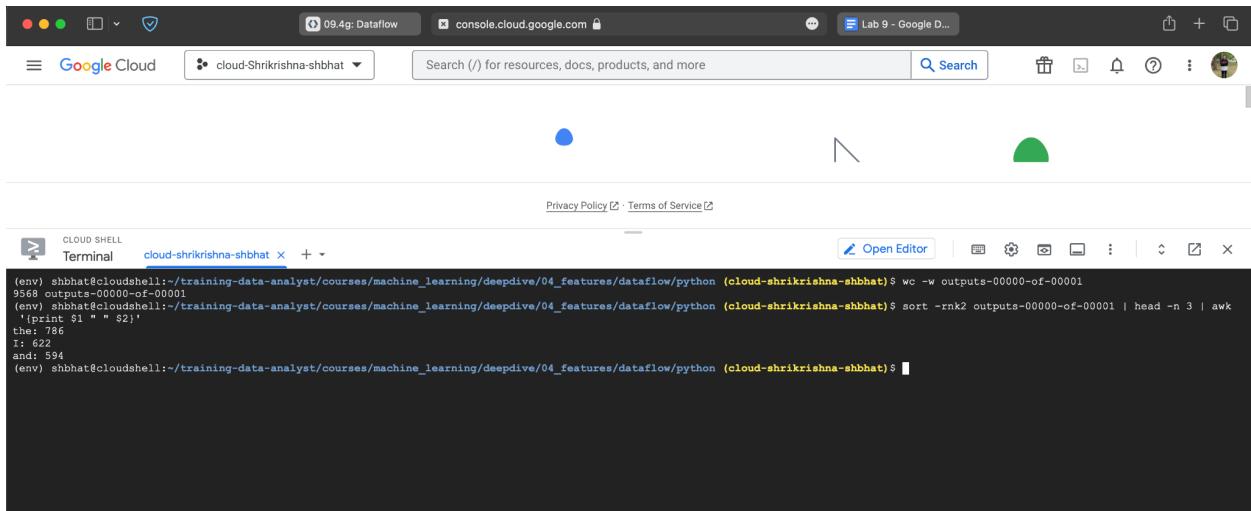
9.4.7 Use `wc` with an appropriate flag to determine the number of unique words in King Lear.



A screenshot of a Google Cloud Shell terminal window. The terminal title is "cloud-shrikrishna-shbhat". The command entered was "wc -w outputs-00000-of-00001". The output shows 9568 unique words. The terminal interface includes a toolbar at the top with icons for Open Editor, Share, and Help, and a status bar at the bottom.

```
(env) shbhat@cloudshell:~/training-data-analyst/courses/machine_learning/deepdive/04_features/dataflow/python (cloud-shrikrishna-shbhat)$ wc -w outputs-00000-of-00001
9568 outputs-00000-of-00001
(env) shbhat@cloudshell:~/training-data-analyst/courses/machine_learning/deepdive/04_features/dataflow/python (cloud-shrikrishna-shbhat)$
```

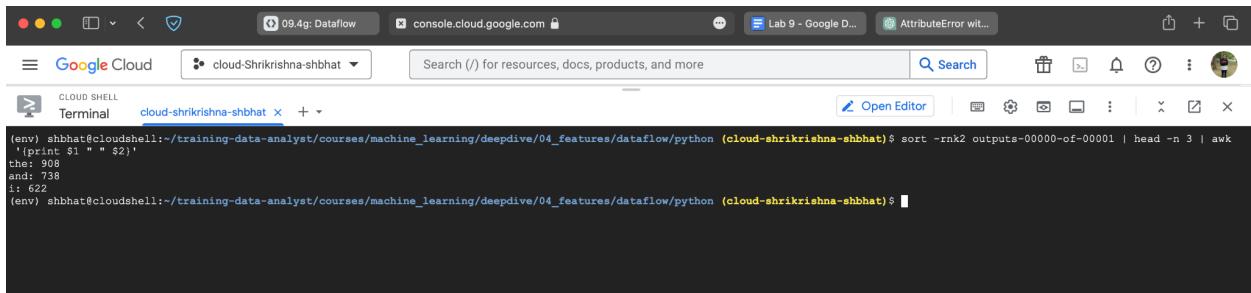
9.4.8 Use sort with appropriate flags to perform a *numeric sort* on the *key field* containing the count for each word in *descending order*. Pipe the output into `head` to show the top 3 words in King Lear and the number of times they appear



A screenshot of a Google Cloud Shell terminal window. The terminal title is "cloud-shrikrishna-shbhat". The command entered was "wc -w outputs-00000-of-00001 | sort -rnk2 | head -n 3 | awk '{print \$1 \" \" \$2}'". The output shows the top 3 words and their counts: the: 786, I: 622, and: 594. The terminal interface includes a toolbar at the top with icons for Open Editor, Share, and Help, and a status bar at the bottom.

```
(env) shbhat@cloudshell:~/training-data-analyst/courses/machine_learning/deepdive/04_features/dataflow/python (cloud-shrikrishna-shbhat)$ wc -w outputs-00000-of-00001
9568 outputs-00000-of-00001
(env) shbhat@cloudshell:~/training-data-analyst/courses/machine_learning/deepdive/04_features/dataflow/python (cloud-shrikrishna-shbhat)$ sort -rnk2 outputs-00000-of-00001 | head -n 3 | awk
'{print $1 " " $2}'
the: 786
I: 622
and: 594
(env) shbhat@cloudshell:~/training-data-analyst/courses/machine_learning/deepdive/04_features/dataflow/python (cloud-shrikrishna-shbhat)$
```

9.4.9 Use the previous method to show the top 3 words in King Lear, case-insensitive, and the number of times they appear.

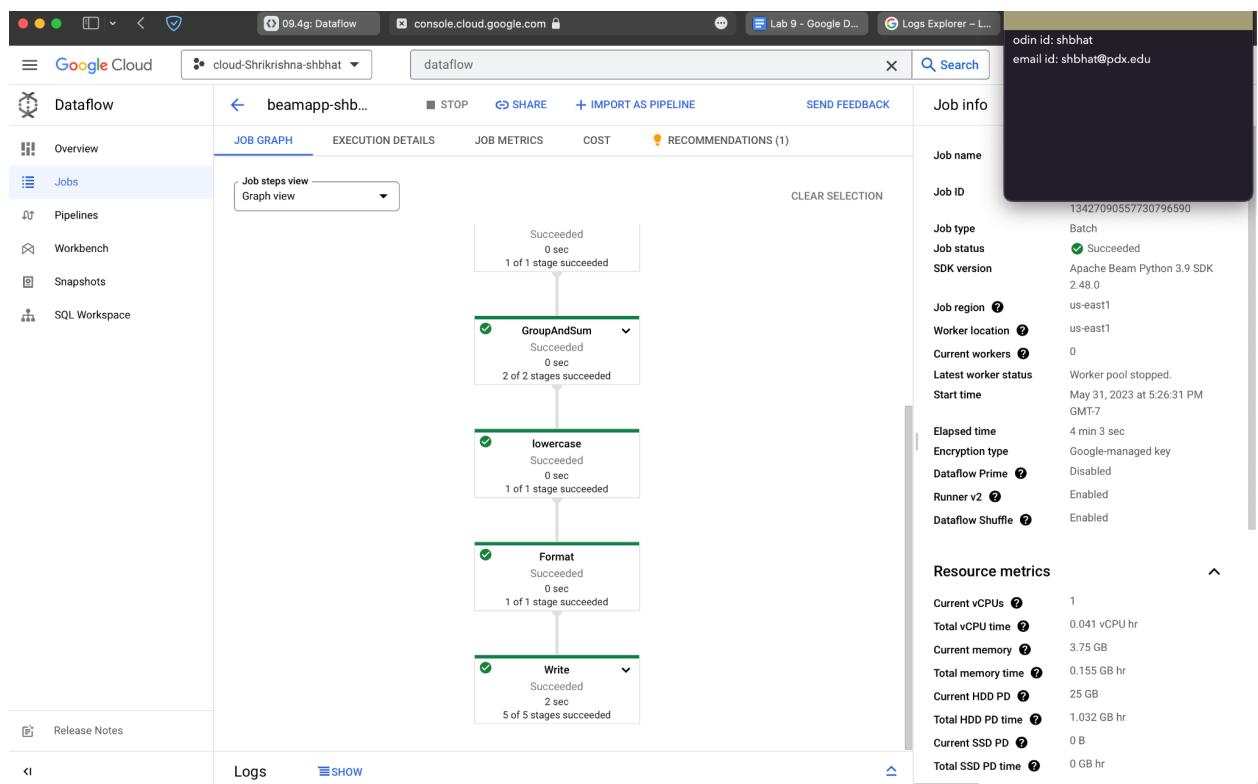


The screenshot shows a Google Cloud Terminal window titled "09.4g: Dataflow". The terminal session is running on a VM named "cloud-Shrikrishna-shbhat". The command entered was:

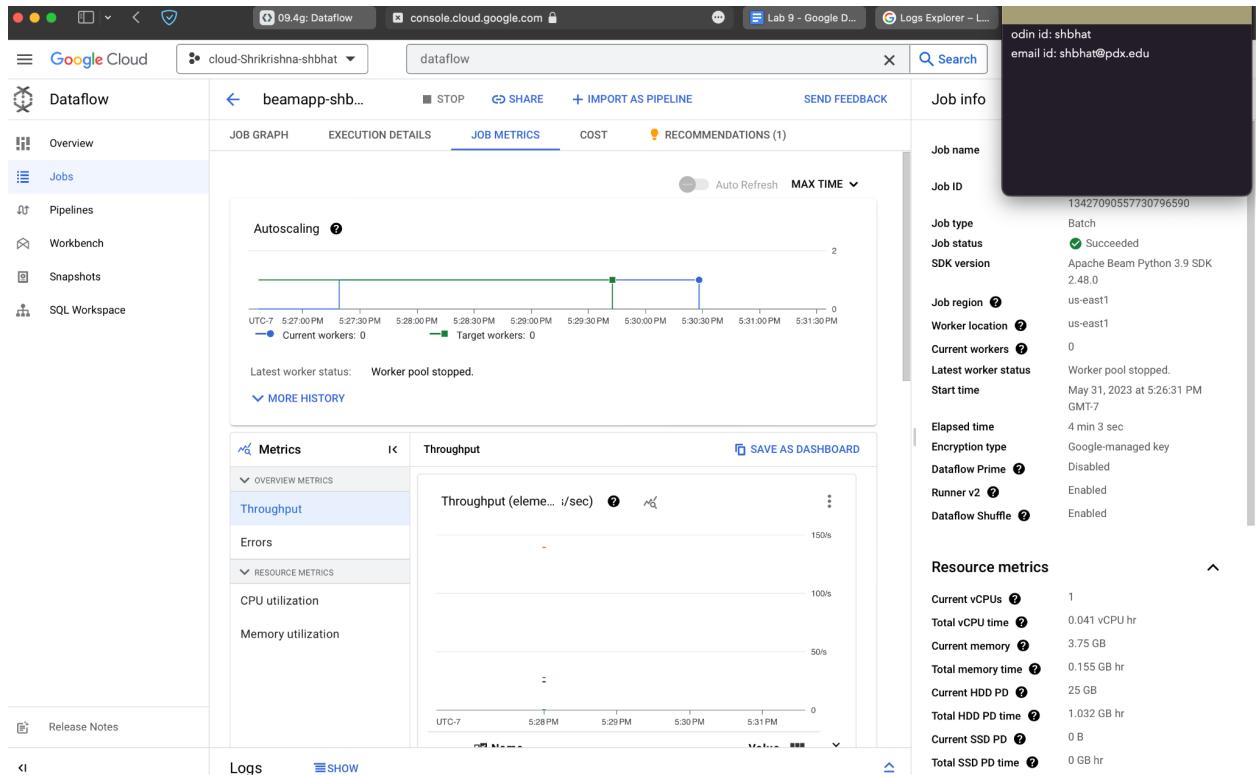
```
(env) shbhat@cloudshell:~/training-data-analyst/courses/machine_learning/deepdive/04_features/dataflow/python (cloud-shrikrishna-shbhat)$ sort -rnk2 outputs-00000-of-00001 | head -n 3 | awk '{print $1 " " $2}'  
the: 908  
and: 738  
is: 622  
(env) shbhat@cloudshell:~/training-data-analyst/courses/machine_learning/deepdive/04_features/dataflow/python (cloud-shrikrishna-shbhat)$
```

Card 9 Run Code using Dataflow runner

9.4.10 The part of the job graph that has taken the longest time to complete.



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



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

Final write stage has created 6 files.

Card 12 View raw data from PubSub

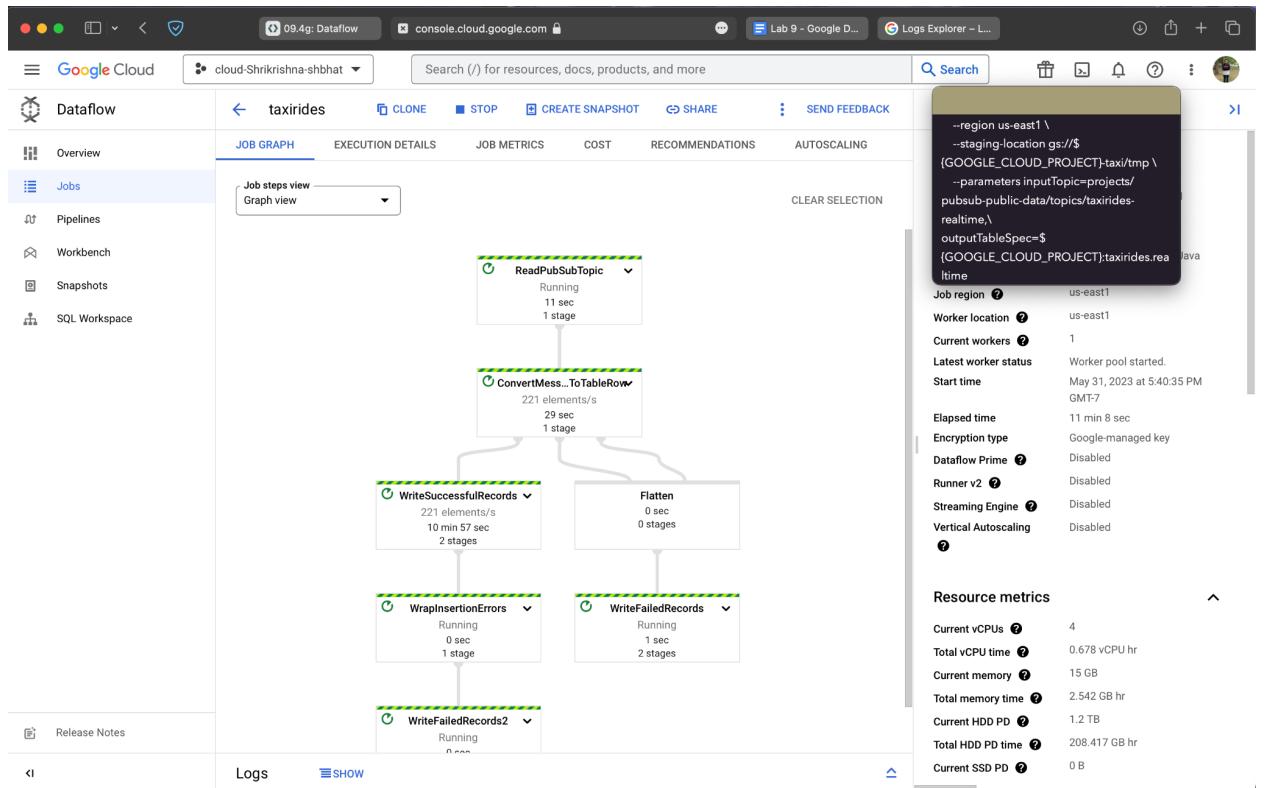
9.4.13 Take a screenshot listing the different fields of this object.

The screenshot shows the Google Cloud Logs Explorer interface. The 'Logs' tab is selected. A terminal window titled 'Terminal' shows raw data being pulled from a PubSub subscription named 'taxisub'. The data is in JSON format, showing fields like 'ride_id', 'latitude', 'longitude', 'meter_reading', 'meter_increment', 'ride_status', and 'passenger_count'. The terminal command used was 'gcloud pubsub subscriptions pull taxisub --auto-ack'.

```
(env) shbhat@cloudshell:~ (cloud-shrikrishna-shbhat)$ cd
(env) shbhat@cloudshell:~ (cloud-shrikrishna-shbhat)$ gcloud pubsub subscriptions create taxisub \
--topic=projects/pubsub-public-data/topics/taxirides-realtime
Created subscription [projects/cloud-shrikrishna-shbhat/subscriptions/taxisub].
(env) shbhat@cloudshell:~ (cloud-shrikrishna-shbhat)$ gcloud pubsub subscriptions pull taxisub --auto-ack
DATA: {"ride_id": "697924f-4a98-4179-8528-4e70b62fdcc2", "point_idx": 367, "latitude": 40.777190000000004, "longitude": -73.99027000000001, "timestamp": "2023-05-31T20:36:19.28338-04:00", "meter_reading": 3.32846, "meter_increment": 0.025566343, "ride_status": "enroute", "passenger_count": 1}
MESSAGE_ID: 697924f-4a98-4179-8528-4e70b62fdcc2
ORDERING KEY:
ATTRIBUTES: ts=2023-05-31T20:36:19.28338-04:00
DELIVERY_ATTEMPT:
ACK_STATUS: SUCCESS
(env) shbhat@cloudshell:~ (cloud-shrikrishna-shbhat)$
```

Card 14 Run Dataflow job from template

9.4.14 Take a screenshot of the pipeline that includes its stages and the number of elements per second being handled by individual stages.



Card 15 Query data in BigQuery

9.4.15 Take a screenshot showing the number of passengers and the amount paid for the first ride

The screenshot shows the Google Cloud BigQuery interface. On the left, the 'Explorer' sidebar lists a project named 'cloud-Shrikrishna-shbhat' and a dataset 'taxirides'. Within the 'taxirides' dataset, a table named 'realtime' is selected. The main area displays the schema and preview of the 'realtime' table. The schema includes columns: Row, ride_id, point_idx, latitude, longitude, timestamp, meter_reading, meter_incremen, and ride. The preview shows 21 rows of data. A SQL query editor on the right side of the screen shows a query being constructed:

```
--region us-east1 \
--staging-location gs://$ \
[GOOGLE_CLOUD_PROJECT]:taxirides \
--parameters inputTopic=projects/ \
pubsub-public-data/topics/taxirides- \
realtime, \
outputTableSpec=$ \
[GOOGLE_CLOUD_PROJECT]:taxirides.rea \
ltime
2
1
SELECT
ride_stat
us
FROM
`cloud-sh
rikrishna-
shbhat-
taxirides
`_realtime` \
WHERE
DATE
(timestamp
) =
"2023-05-
31"
LIMIT
1000
```

Below the preview, there are buttons for 'Results per page' (set to 50), '1 - 50 of many', and navigation arrows.

9.4.16 Take a screenshot showing the estimated number of rows in the table.

The screenshot shows the Google Cloud Dataflow console interface. The left sidebar has an 'Explorer' tab selected, showing a tree view of workspace resources under 'cloud-Shrikrishna-shbhat'. A table named 'realtime' is selected. The main panel displays the 'realtime' table details. The 'DETAILS' tab is active, showing the following information:

- Partition expiration: Partitions do not expire
- Partition filter: Not required

Below this, the 'Storage info' section provides detailed statistics:

Metric	Value
Number of rows	0
Number of partitions	0
Total logical bytes	0 B
Active logical bytes	0 B
Long term logical bytes	0 B
Total physical bytes	0 B
Active physical bytes	0 B
Long term physical bytes	0 B
Time travel physical bytes	0 B

The 'Streaming buffer statistics' section shows:

Statistic	Value
Estimated size	15.02 MB
Estimated rows	107,759
Earliest entry time	May 31, 2023, 5:42:28 PM UTC-7

A large portion of the right side of the screen is occupied by a code editor window displaying a SQL query:

```
--region us-east1 \
--staging-location gs://$(
(GOOGLE_CLOUD_PROJECT)taxi/tmp \
--parameters inputTopic=projects/
pubsub-public-data/topics/taxirides-
realtime, \
outputTableSpec=$(
(GOOGLE_CLOUD_PROJECT)taxirides.re
ltime

1 SELECT
ride_stat
us FROM
cloud-sh
rikrishna-
shbhat.
taxirides
realtime'
WHERE
DATE
(timestamp
p) =
"2023-05-
31"
LIMIT
1000
```

9.4.17 Take a screenshot showing the per-minute number of rides, passengers, and revenue for the data collected.

The screenshot shows the Google Cloud Dataflow interface. On the left, the Explorer sidebar displays a project named "cloud-Shrikrishna-shbhat" with a single dataset "taxirides" containing a table "realtime". A tooltip over the "realtime" table shows its schema and configuration details. The main workspace contains an untitled query editor with the following SQL code:

```
1 SELECT
2   FORMAT_TIMESTAMP("%R", timestamp, "America/Los_Angeles") AS minute,
3   COUNT(DISTINCT ride_id) AS total_rides,
4   SUM(passenger_count) AS total_passengers,
5   SUM(meter_reading) AS total_revenue
6 FROM
7   taxirides.realtime
8 WHERE
9   ride_status = 'dropoff'
10 GROUP BY
11   minute
12 ORDER BY
13   minute ASC
```

The "Query results" section shows the output of the query:

Row	minute	total_rides	total_passengers	total_revenue
1	17:39	1	1	0.0
2	17:40	138	234	2096.070003600...
3	17:41	122	196	1615.489998200...

Card 16 – Data visualization

9.4.18 Take a screenshot showing the plot for your data for your lab notebook

