Contents

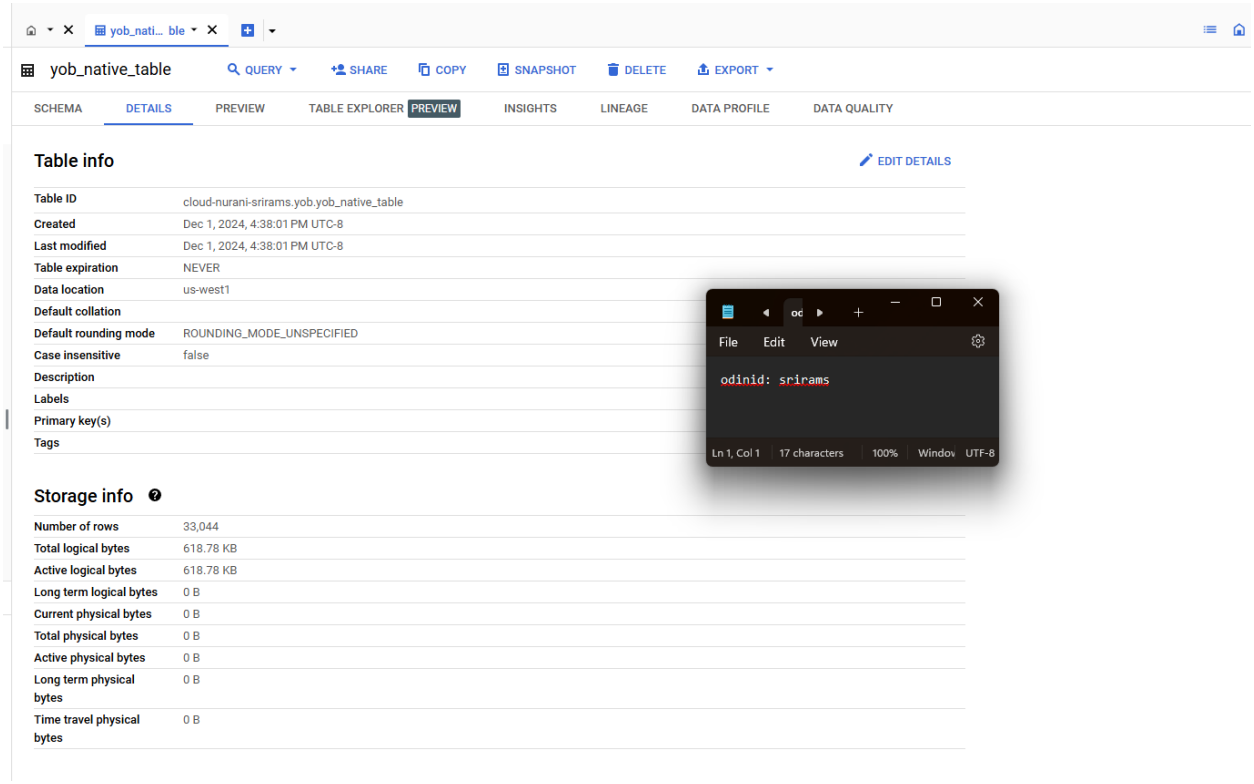
9.1g: BigQuery, BigLake

9.1.1 BigQuery Lab #1 (Native tables)

9.1.2 Examine dataset

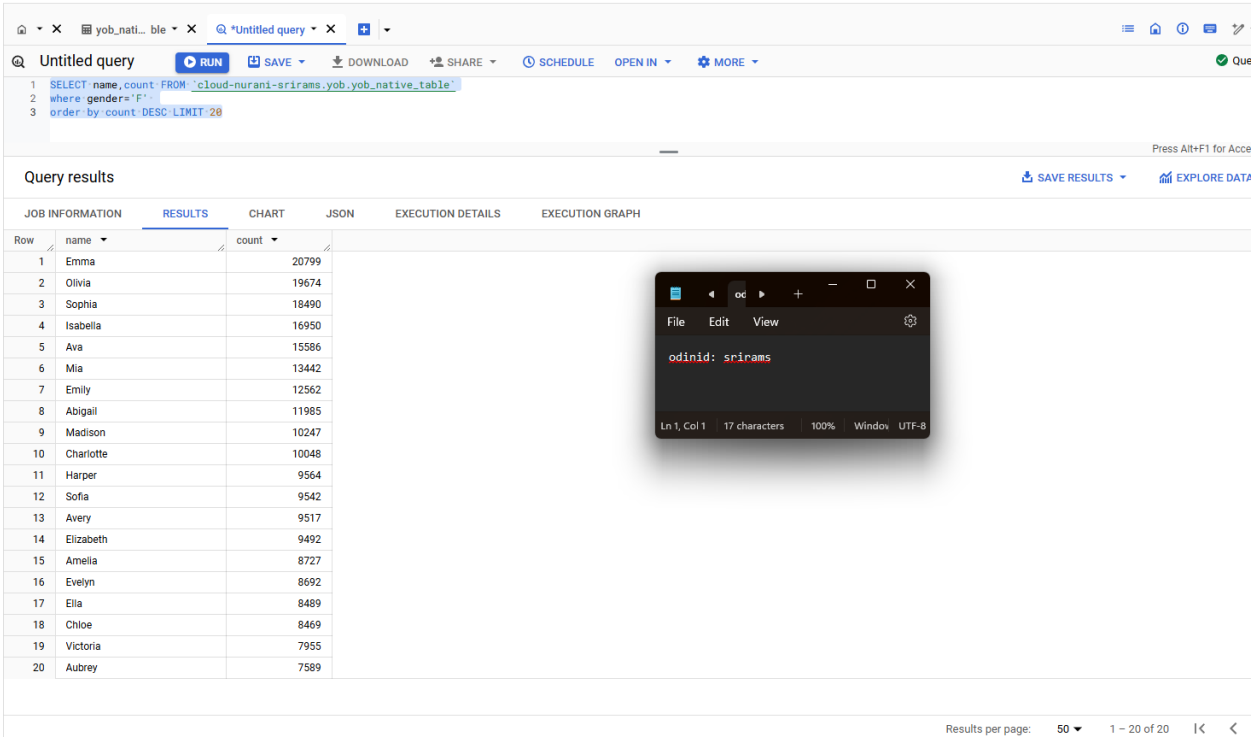
9.1.3 Create dataset

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



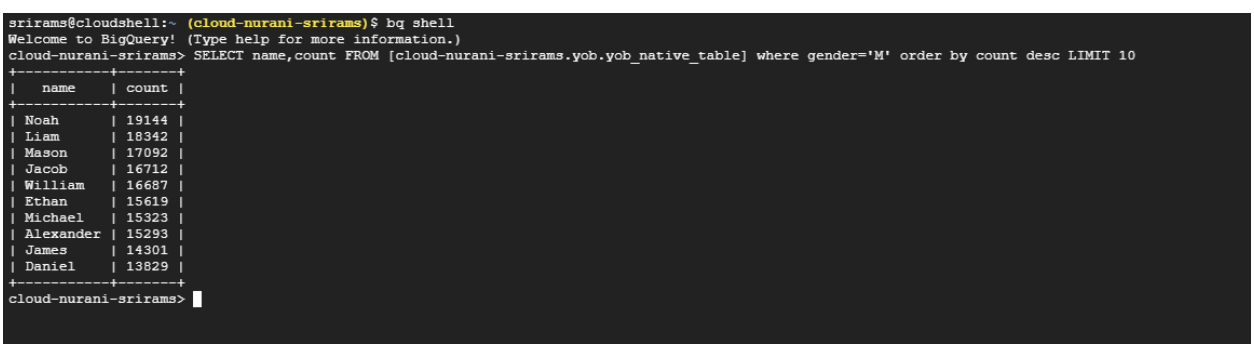
9.1.4 Query data

- **Screenshot the query results and include it in your lab notebook**



- **Screenshot your results and include it in your lab notebook**

- **Screenshot your results and include it in your lab notebook**



- **Screenshot your results and include it in your lab notebook**

```
cloud-nurani-srirams> SELECT name,count FROM [cloud-nurani-srirams.yob.yob_native_table] where name='Sriram'
+--------+-------+
|  name  | count |
+--------+-------+
| Sriram |    23 |
+--------+-------+
```

9.1.5 BigQuery Lab #2 (Lake tables)

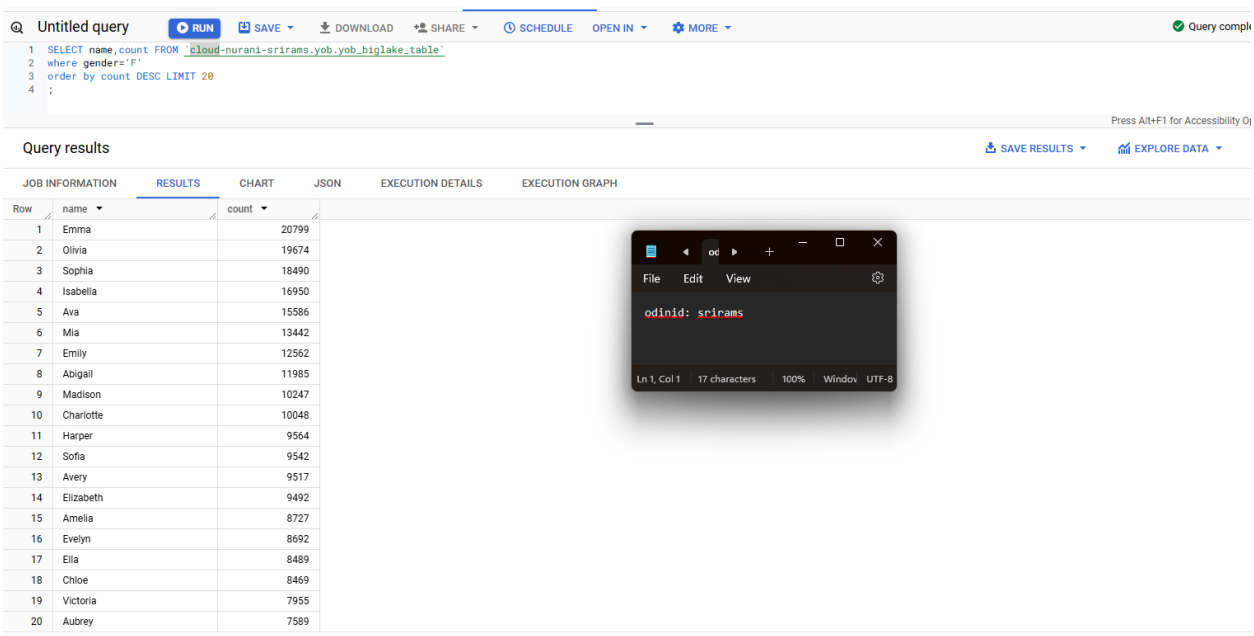
9.1.6 Create external table

9.1.7 –

9.1.8 Configuring permissions

9.1.9 Query data

- **Screenshot the query results and include it in your lab notebook**



9.1.10 Clean up

9.2g: Jupyter Notebooks

9.2.1 Notebooks Lab #1 (Natality)

9.2.2 Launch notebook

9.2.3 BigQuery query

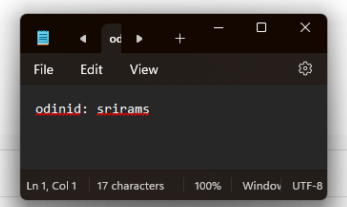
- **How much less data does this query process compared to the size of the table?**

This query will process 3.05 GB when run. That is almost 18.89 gb of less data

- **How many twins were born during this time range?**

  375362

```
SELECT
  plurality,
  COUNT(1) AS num_babies,
  AVG(weight_pounds) AS avg_wt
FROM
  bigquery-public-data.samples.natality
WHERE
  year between 2001 and 2003
GROUP BY
  plurality
ORDER BY
  plurality ASC
```



Query results

SAVE RESULTS

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS | EXECUTION GRAPH

row	plurality	num_babies	avg_wt
1	1	11757058	7.345780731912...
2	2	375362	5.174620274025...
3	3	20933	3.709771577132...
4	4	1407	2.851986586921...
5	5	239	2.696701862081...

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

On average, single babies (plurality 1) weigh approximately 2.17116046 units more than twins (plurality 2)

9.2.4 Jupyter notebook query

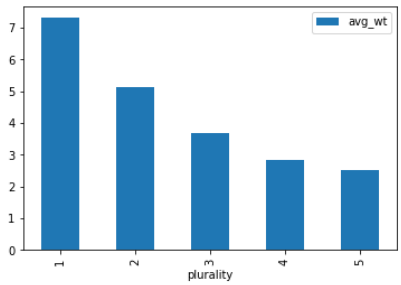
9.2.5 Exploring the dataset

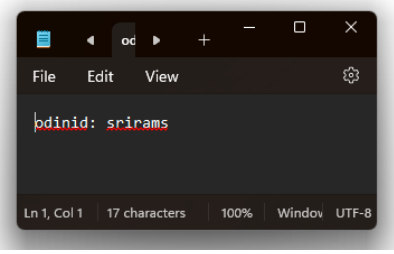
9.2.6 Run queries

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

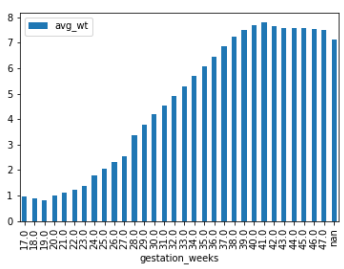
Plurality and gestation weeks

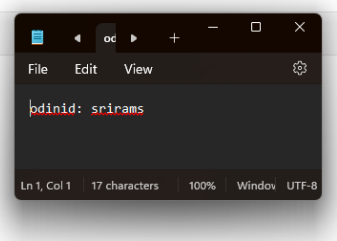
```
[8]: df = get_distinct_values('plurality')
     df.plot(x='plurality', y='avg_wt', kind='bar')

[8]: <matplotlib.axes._subplots.AxesSubplot at 0x7f3540909b50>
```






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

[10]: <matplotlib.axes._subplots.AxesSubplot at 0x7f354087df50>
```





9.2.7 Notebooks Lab #2 (COVID-19 data)

9.2.8 Mobility

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

  2020-03-13

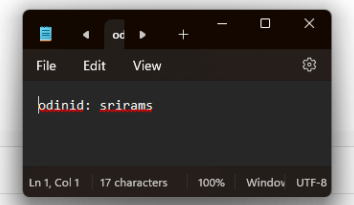
- **On the day the stay-at-home order took effect (3/23/2020), what was the total impact on workplace trips?**

-49

9.2.9 Airport traffic

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

```
SELECT
  airport_name,
  AVG(percent_of_baseline) 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
  AND EXTRACT(YEAR from date) = 2020
GROUP BY
  airport_name
ORDER BY
  traffic_fraction asc LIMIT 3;

  select * from   `bigquery-public-data.covid19_geotab_mobility_impact.airport_traffic` ;
```



Query results

SAVE RESULTS

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	airport_name	traffic_fraction
1	McCarran International	32.666666666666671
2	San Francisco International	38.599999999999994
3	Denver International	38.633333333333333

- McCarran International 32.666666666666671
- San Francisco International 38.599999999999994
- Denver International 38.633333333333333

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

```
SELECT
  airport_name,
  AVG(percent_of_baseline) 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) = 8
  AND EXTRACT(YEAR from date) = 2020
GROUP BY
  airport_name
ORDER BY
  traffic_fraction asc LIMIT 3;

  select * from  `bigquery-public-data.covid19_geotab_mobility_impact.airport_traffic` ;
```

pdinid: srirams

Query results

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	airport_name	traffic_fraction
1	McCarran International	40.93333333333...
2	Detroit Metropolitan Wayne Co...	46.13333333333...
3	San Francisco International	51.33333333333...

McCarran International 40.933333333333323

Detroit Metropolitan Wayne County 46.133333333333326

San Francisco International 51.333333333333336

9.2.10 Mortality

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

excess_deaths

QUERY | SHARE | COPY | SNAPSHOT | DELETE | EXPORT | REFRESH

SCHEMA | DETAILS | PREVIEW | TABLE EXPLORER PREVIEW | INSIGHTS | LINEAGE | DATA PROFILE | DATA QUALITY

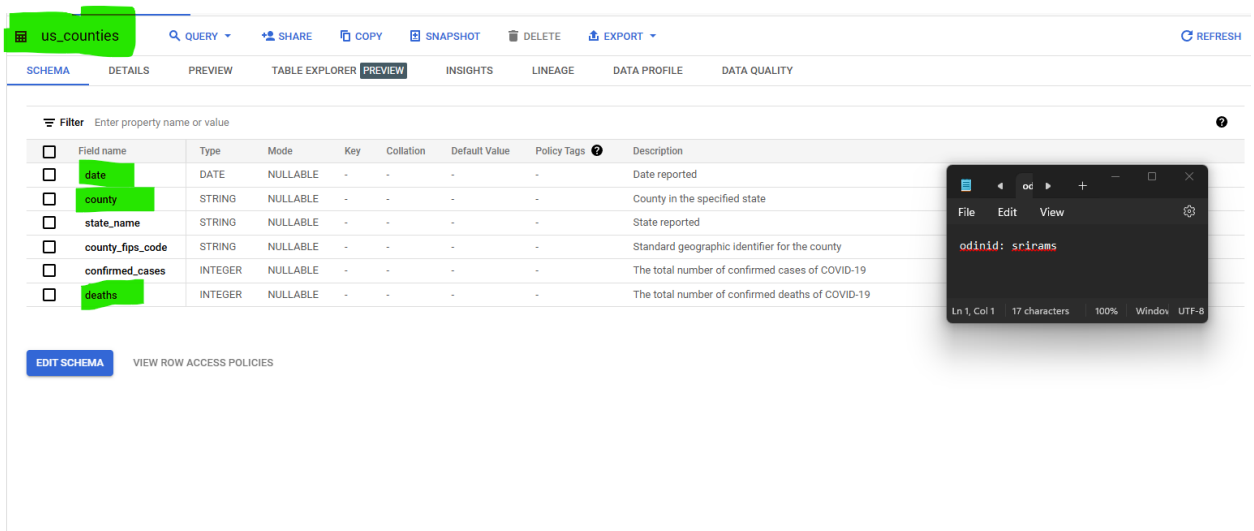
Filter Enter property name or value

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. M...
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

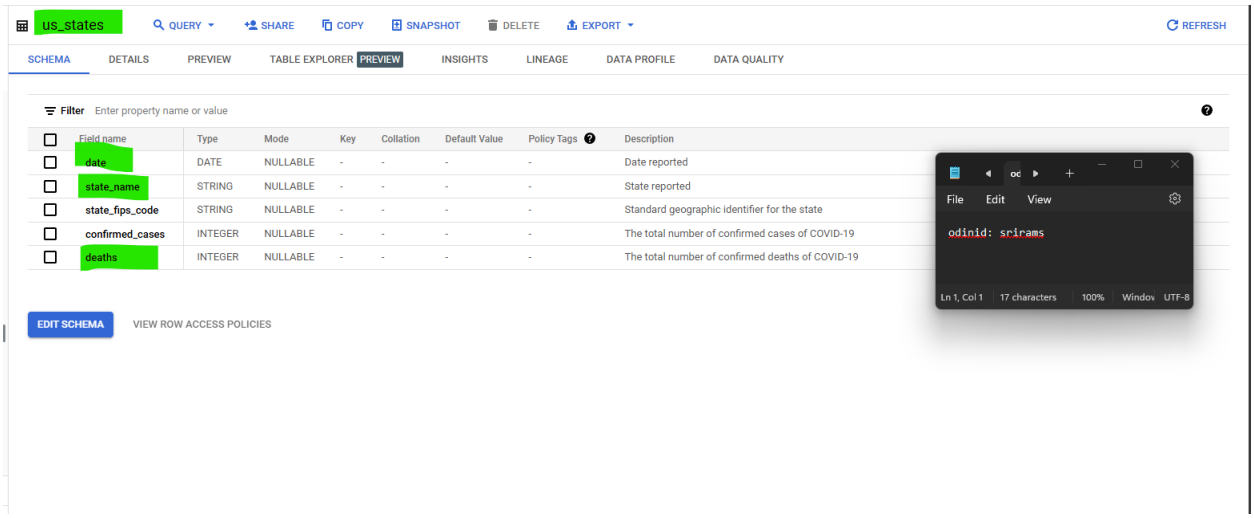
EDIT SCHEMA | VIEW ROW ACCESS POLICIES

pdinid: srirams

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



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



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

mask_use_by_county

QUERY SHARE COPY SNAPSHOT DELETE EXPORT REFRESH

SCHEMA DETAILS PREVIEW TABLE EXPLORER PREVIEW INSIGHTS LINEAGE DATA PROFILE DATA QUALITY

Filter Enter property name or value

Field name	Type	Mode	Key	Collation	Default Value	Policy Tags	Description
county_fips_code	STRING	NULLABLE	-	-	-	-	Standard geographic identifier for the county
never	FLOAT	NULLABLE	-	-	-	-	The estimated share of people in this county who would say never in response to the question "How often do you ...
rarely	FLOAT	NULLABLE	-	-	-	-	The estimated share of people in this county who would say rarely
sometimes	FLOAT	NULLABLE	-	-	-	-	The estimated share of people in this county who would say sometimes
frequently	FLOAT	NULLABLE	-	-	-	-	The estimated share of people in this county who would say frequently
always	FLOAT	NULLABLE	-	-	-	-	The estimated share of people in this county who would say always

EDIT SCHEMA VIEW ROW ACCESS POLICIES

File Edit View

odinid: srirams

Ln 1, Col 1 17 characters 100% Windov UTF-8

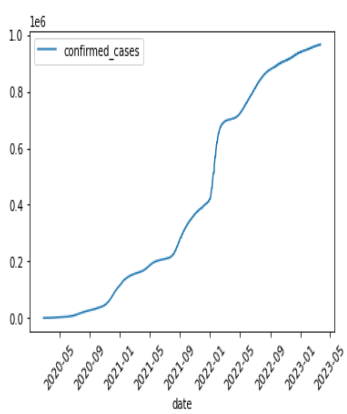
9.2.11 Run example queries

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

[15]:
```
query_string="""
SELECT date, confirmed_cases
FROM `bigquery-public-data.covid19_nyt.us_states`
WHERE state_name = 'Oregon'
ORDER BY date ASC
"""
from google.cloud import bigquery
df = bigquery.Client().query(query_string).to_dataframe()
```

[16]:
```
df.plot(x='date', y='confirmed_cases', kind='line', rot=45)
```

[16]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f354013dc90>
```



File Edit View

odinid: srirams

Ln 1, Col 1 17 characters 100% Windov UTF-8

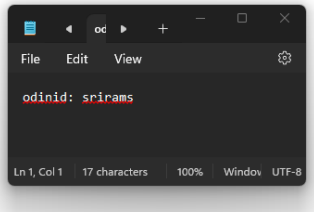
- **From within your Jupyter notebook, run the query and write code that shows the first 10 states that reached 1000 deaths from COVID-19. Take a screenshot for your lab notebook.**

[17]:
```
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
"""
from google.cloud import bigquery
df = bigquery.Client().query(query_string).to_dataframe()
```

[18]:
```
df.head(10)
```

[18]:

	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



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

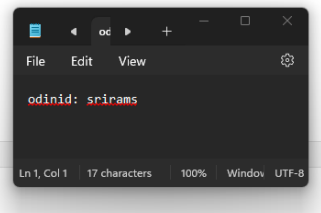
```
[21]: query_string="""
SELECT DISTINCT mu.county_fips_code, mu.always, ct.county
FROM `bigquery-public-data.covid19_nyt.mask_use_by_county` as mu
LEFT JOIN `bigquery-public-data.covid19_nyt.us_counties` as ct
ON mu.county_fips_code = ct.county_fips_code
ORDER BY mu.always DESC
"""
from google.cloud import bigquery
df = bigquery.Client().query(query_string+"LIMIT 5").to_dataframe()
```

```
[22]: df
```

[22]:

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

```
[ ]:
```



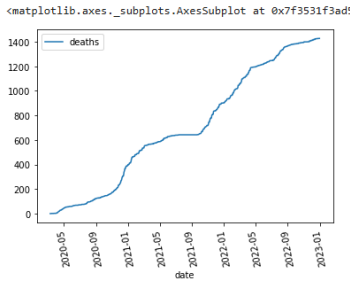
9.2.12 Write queries

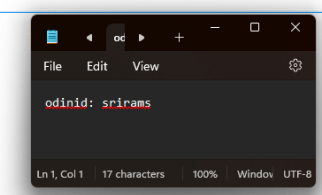
- **Plot the results and take a screenshot for your lab notebook.**

```
[24]: query_string="""
SELECT deaths,date FROM `bigquery-public-data.covid19_nyt.us_counties` where county="Multnomah" ORDER BY date asc;
"""
from google.cloud import bigquery
df = bigquery.Client().query(query_string).to_dataframe()
```

```
[30]: df.plot(x='date', y='deaths', kind='line' , rot="100")
```

```
[30]: <matplotlib.axes._subplots.AxesSubplot at 0x7f3531f3ad50>
```





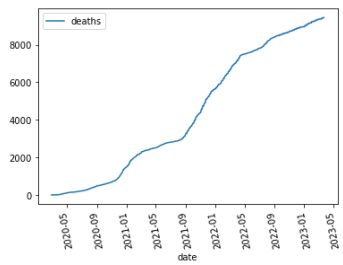
- **Plot the results and take a screenshot for your lab notebook.**

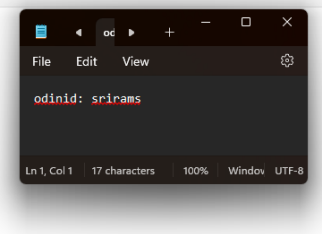
9.3.5 Create Compute Engine cluster

```
[31]: query_string="""
SELECT deaths,date FROM `bigquery-public-data.covid19_nyt.us_states` where state_name="Oregon" ORDER BY date asc;
"""
from google.cloud import bigquery
df = bigquery.Client().query(query_string).to_dataframe()
```

```
[32]: df.plot(x='date', y='deaths', kind='line' , rot="100")
```

```
[32]: <matplotlib.axes._subplots.AxesSubplot at 0x7f353217b710>
```






```
[ ]:
```

9.2.13 Clean up

9.3g: Dataproc

9.3.1 Dataproc Lab #1 (π)

9.3.2 Calculating π

9.3.3 Code

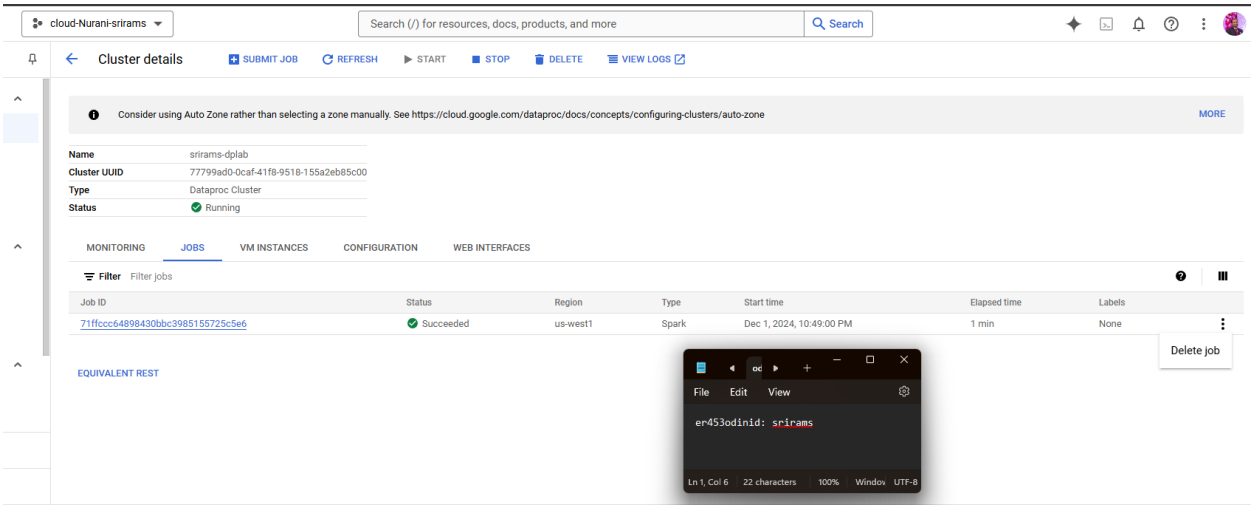
9.3.4 Dataproc setup

9.3.5 Create Compute Engine cluster

9.3.6 Run computation

- **How long did the job take to execute?**

```
srirams@cloudshell:~ (cloud-nurani-srirams)$ date

gcloud dataproc jobs submit spark --cluster ${CLUSTERNAME} \
  --class org.apache.spark.examples.SparkPi \
  --jars file:///usr/lib/spark/examples/jars/spark-examples.jar -- 1000 \
  >& output.txt &
Mon Dec  2 06:48:59 AM UTC 2024
[1] 1616
srirams@cloudshell:~ (cloud-nurani-srirams)$ gcloud dataproc jobs list --cluster ${CLUSTERNAME} ; date
JOB_ID: 71ffccc64898430bbc3985155725c5e6
TYPE: spark
STATUS: SETUP_DONE
Mon Dec  2 06:49:14 AM UTC 2024
srirams@cloudshell:~ (cloud-nurani-srirams)$ gcloud dataproc jobs list --cluster ${CLUSTERNAME} ; date
JOB_ID: 71ffccc64898430bbc3985155725c5e6
TYPE: spark
STATUS: RUNNING
Mon Dec  2 06:49:41 AM UTC 2024
srirams@cloudshell:~ (cloud-nurani-srirams)$ gcloud dataproc jobs list --cluster ${CLUSTERNAME} ; date
JOB_ID: 71ffccc64898430bbc3985155725c5e6
TYPE: spark
STATUS: RUNNING
Mon Dec  2 06:49:47 AM UTC 2024
srirams@cloudshell:~ (cloud-nurani-srirams)$ gcloud dataproc jobs list --cluster ${CLUSTERNAME} ; date
JOB_ID: 71ffccc64898430bbc3985155725c5e6
TYPE: spark
STATUS: RUNNING
Mon Dec  2 06:49:55 AM UTC 2024
srirams@cloudshell:~ (cloud-nurani-srirams)$ gcloud dataproc jobs list --cluster ${CLUSTERNAME} ; date
JOB_ID: 71ffccc64898430bbc3985155725c5e6
TYPE: spark
STATUS: DONE
Mon Dec  2 06:50:02 AM UTC 2024
srirams@cloudshell:~ (cloud-nurani-srirams)$
```



Elapsed minute: 1 minute

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

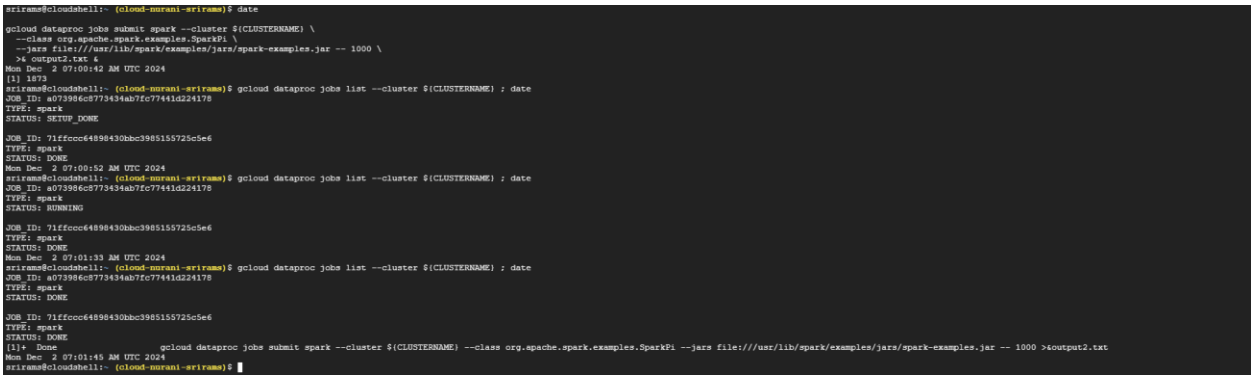
```
Job [71ffccc64898430bbc3985155725c5e6] submitted.
Waiting for job output...
24/12/02 06:49:09 INFO SparkEnv: Registering MapOutputTracker
24/12/02 06:49:09 INFO SparkEnv: Registering BlockManagerMaster
24/12/02 06:49:09 INFO SparkEnv: Registering BlockManagerMasterHeartbeat
24/12/02 06:49:10 INFO SparkEnv: Registering OutputCommitCoordinator
24/12/02 06:49:11 INFO MetricsConfig: Loaded properties from hadoop-metrics2.properties
24/12/02 06:49:11 INFO MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
24/12/02 06:49:11 INFO MetricsSystemImpl: google-hadoop-file-system metrics system started
24/12/02 06:49:11 INFO DataprocSparkPlugin: Registered 188 driver metrics
24/12/02 06:49:12 INFO DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at srirams-dplab-m.c.cloud-nurani-srirams.internal./10.138.0.22:8032
24/12/02 06:49:13 INFO AHSProxy: Connecting to Application History server at srirams-dplab-m.c.cloud-nurani-srirams.internal./10.138.0.22:10200
24/12/02 06:49:14 INFO Configuration: resource-types.xml not found
24/12/02 06:49:14 INFO ResourceUtils: Unable to find 'resource-types.xml'.
24/12/02 06:49:17 INFO YarnClientImpl: Submitted application application_1733121731148_0001
24/12/02 06:49:18 INFO DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at srirams-dplab-m.c.cloud-nurani-srirams.internal./10.138.0.22:8030
24/12/02 06:49:21 INFO RequestTracker: Detected high latency for [url=https://storage.googleapis.com/storage/v1/b/dataproc-temp-us-west1-40274584080-yqwvqko7/o?delimiter=/&fields=items(bucket,name,timeCreated,updated,generation,metagenerati
24/12/02 06:49:21 INFO GhfaGlobalStorageStatistics: periodic connector metrics: {gcs_api_client_non_found_response_count=1, gcs_api_client_side_error_count=1, gcs_api_time=1217, gcs_api_total_request_count=2, gcs_connector_time=1914, gcs_lis
24/12/02 06:49:21 INFO GoogleCloudStorageImpl: Ignoring exception of type GoogleJsonResponseException: verified object already exists with desired state.
24/12/02 06:49:23 INFO GoogleHadoopOutputStream: hflush(): No-op due to rate limit (RateLimiter[stableRate=0.2qps]): readers will *not* yet see flushed data for gs://dataproc-temp-us-west1-40274584080-yqwvqko7/77799ad0-0caf-41f8-9518-155a2eb
Pi is roughly 3.1414319514143136
24/12/02 06:49:56 INFO RequestTracker: Detected high latency for [url=https://storage.googleapis.com/upload/storage/v1/b/dataproc-temp-us-west1-40274584080-yqwvqko7/o?ifGenerationMatch=0&name=77799ad0-0caf-41f8-9518-155a2eb85c00/spark-job-hi
24/12/02 06:49:57 INFO GhfaGlobalStorageStatistics: Detected potential high latency for operation op_hflush. latencyMs=662; previousMaxLatencyMs=416; operationCount=15; context=gs://dataproc-temp-us-west1-40274584080-yqwvqko7/77799ad0-0caf-4
24/12/02 06:49:57 INFO DataprocSparkPlugin: Shutting down driver plugin. metrics=[action_http_patch_request=0, files_created=1, gcs_api_server_timeout_count=0, op_get_list_status_result_size=0, op_open=0, action_http_delete_request=3, gcs_ap
Job [71ffccc64898430bbc3985155725c5e6] finished successfully.
done: true
driverControlFilesUri: gs://dataproc-staging-us-west1-40274584080-wdfoaipw/google-cloud-dataproc-metainfo/77799ad0-0caf-41f8-9518-155a2eb85c00/jobs/71ffccc64898430bbc3985155725c5e6/
driverOutputResourceUri: gs://dataproc-staging-us-west1-40274584080-wdfoaipw/google-cloud-dataproc-metainfo/77799ad0-0caf-41f8-9518-155a2eb85c00/jobs/71ffccc64898430bbc3985155725c5e6/driveroutput
jobUuid: 22cb6558-f364-3121-9411-7012158f6dc3
placement:
  clusterName: srirams-dplab
  clusterUuid: 77799ad0-0caf-41f8-9518-155a2eb85c00
reference:
  jobId: 71ffccc64898430bbc3985155725c5e6
  projectId: cloud-nurani-srirams
sparkJob:
  args:
  - '1000'
  jarFileUris:
  - file:///usr/lib/spark/examples/jars/spark-examples.jar
  mainClass: org.apache.spark.examples.SparkPi
status:
  state: DONE
  stateStartTime: '2024-12-02T06:50:00.671834Z'
statusHistory:
- state: PENDING
  stateStartTime: '2024-12-02T06:49:00.727818Z'
- state: SETUP_DONE
  stateStartTime: '2024-12-02T06:49:00.758749Z'
- details: Agent reported job success
  state: RUNNING
  stateStartTime: '2024-12-02T06:49:01.122121Z'
yarnApplications:
```

Pi is roughly 3.1414319514143196

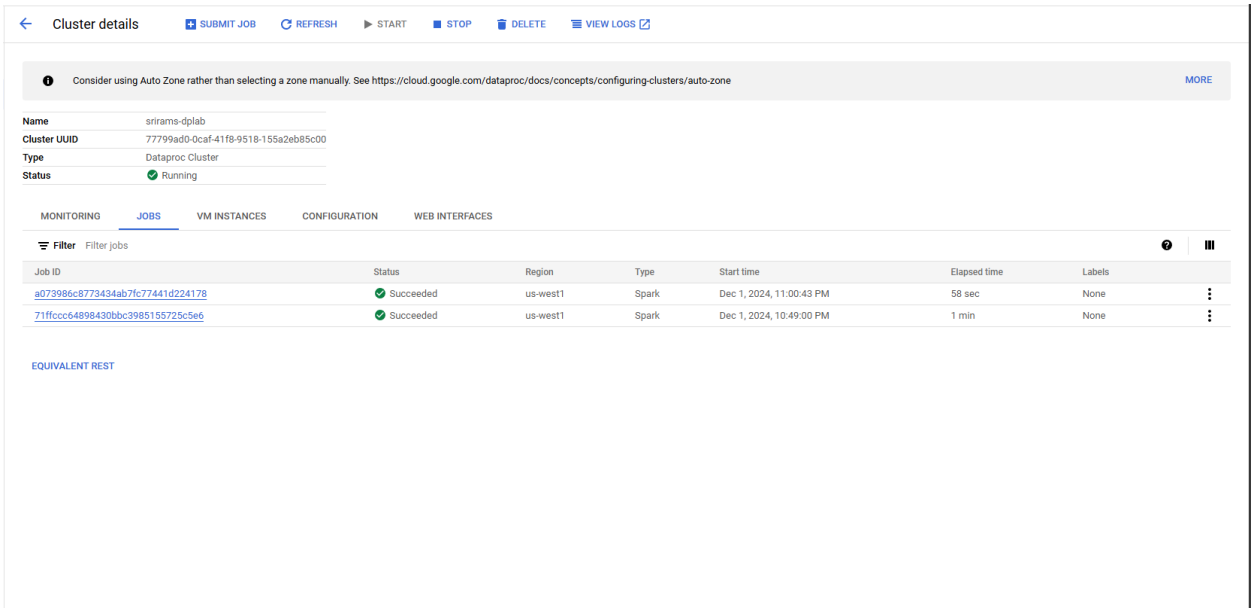
9.3.7 Scale cluster

9.3.8 Run computation again

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



```
sriramg@cloudshell:~ (cloud-murani-sriramg)$ date

gcloud dataproc jobs submit spark --cluster ${CLUSTERNAME} \
  --class org.apache.spark.examples.SparkPi \
  --jars file:///usr/lib/spark/examples/jars/spark-examples.jar -- 1000 \
  >4 output2.txt &
Mon Dec  2 07:00:42 AM UTC 2024
[1] 1873
sriramg@cloudshell:~ (cloud-murani-sriramg)$ gcloud dataproc jobs list --cluster ${CLUSTERNAME} ; date
JOB_ID: a073986c8773434ab7fc77441d224178
TYPE: spark
STATUS: SETUP_DONE

JOB_ID: 71ffccc64898430bbc3985155725c5e6
TYPE: spark
STATUS: DONE
Mon Dec  2 07:00:52 AM UTC 2024
sriramg@cloudshell:~ (cloud-murani-sriramg)$ gcloud dataproc jobs list --cluster ${CLUSTERNAME} ; date
JOB_ID: a073986c8773434ab7fc77441d224178
TYPE: spark
STATUS: RUNNING

JOB_ID: 71ffccc64898430bbc3985155725c5e6
TYPE: spark
STATUS: DONE
Mon Dec  2 07:01:33 AM UTC 2024
sriramg@cloudshell:~ (cloud-murani-sriramg)$ gcloud dataproc jobs list --cluster ${CLUSTERNAME} ; date
JOB_ID: a073986c8773434ab7fc77441d224178
TYPE: spark
STATUS: DONE

JOB_ID: 71ffccc64898430bbc3985155725c5e6
TYPE: spark
STATUS: DONE
[1]+  Done                    gcloud dataproc jobs submit spark --cluster ${CLUSTERNAME} --class org.apache.spark.examples.SparkPi --jars file:///usr/lib/spark/examples/jars/spark-examples.jar -- 1000 >4output2.txt
Mon Dec  2 07:01:45 AM UTC 2024
sriramg@cloudshell:~ (cloud-murani-sriramg)$
```



Elapsed Time is 58 seconds.

It was 2 seconds faster.

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

Pi is roughly 3.141640511416405

```
Job [a073986c8773434ab7fc77441d224178] submitted.
Waiting for job output...
24/12/02 07:00:53 INFO SparkEnv: Registering MapOutputTracker
24/12/02 07:00:54 INFO SparkEnv: Registering BlockManagerMaster
24/12/02 07:00:54 INFO SparkEnv: Registering BlockManagerMasterHeartbeat
24/12/02 07:00:54 INFO SparkEnv: Registering OutputCommitCoordinator
24/12/02 07:00:55 INFO MetricsConfig: Loaded properties from hadoop-metrics2.properties
24/12/02 07:00:55 INFO MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
24/12/02 07:00:55 INFO MetricsSystemImpl: google-hadoop-file-system metrics system started
24/12/02 07:00:56 INFO DataprocSparkPlugin: Registered 188 driver metrics
24/12/02 07:00:57 INFO DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at srirams-dplab-m.c.cloud-nurani-srirams.internal./10.138.0.22:8032
24/12/02 07:00:58 INFO AHSProxy: Connecting to Application History server at srirams-dplab-m.c.cloud-nurani-srirams.internal./10.138.0.22:10200
24/12/02 07:00:59 INFO Configuration: resource-types.xml not found
24/12/02 07:00:59 INFO ResourceUtils: Unable to find 'resource-types.xml'.
24/12/02 07:01:01 INFO YarnClientImpl: Submitted application application_1733121731148_0002
24/12/02 07:01:02 INFO DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at srirams-dplab-m.c.cloud-nurani-srirams.internal./10.138.0.22:8030
24/12/02 07:01:05 INFO RequestTracker: Detected high latency for [url=https://storage.googleapis.com/storage/v1/b/dataproc-temp-us-west1-40274584080-ygwvqko7/o/77799ad0-0caf-41f8-9518-155a2eb85c00%2Fspark-job-history?fields=bucket,name,time
24/12/02 07:01:05 INFO GhfsGlobalStorageStatistics: periodic connector metrics: {gcs_api_client_non_found_response_count=1, gcs_api_client_side_error_count=1, gcs_api_time=779, gcs_api_total_request_count=2, gcs_connector_time=1663, gcs_list
24/12/02 07:01:05 INFO GoogleCloudStorageImpl: Ignoring exception of type GoogleJsonResponseException; verified object already exists with desired state.
24/12/02 07:01:06 INFO GoogleHadoopOutputStream: hflush(): No-op due to rate limit [RateLimiter[stableRate=0.2qps]]: readers will *not* yet see flushed data for gs://dataproc-temp-us-west1-40274584080-ygwvqko7/77799ad0-0caf-41f8-9518-155a2eb
Pi is roughly 3.141640511416405
24/12/02 07:01:37 INFO DataprocSparkPlugin: Shutting down driver plugin. metrics=[action_http_patch_request=0, files_created=1, gcs_api_server_timeout_count=0, op_get_list_status_result_size=0, op_open=0, action_http_delete_request=4, gcs_ap
Job [a073986c8773434ab7fc77441d224178] finished successfully.
done: true
driverControlFilesUri: gs://dataproc-staging-us-west1-40274584080-vdfoaipw/google-cloud-dataproc-metainfo/77799ad0-0caf-41f8-9518-155a2eb85c00/jobs/a073986c8773434ab7fc77441d224178/
driverOutputResourceUri: gs://dataproc-staging-us-west1-40274584080-vdfoaipw/google-cloud-dataproc-metainfo/77799ad0-0caf-41f8-9518-155a2eb85c00/jobs/a073986c8773434ab7fc77441d224178/driveroutput
jobUuid: 75a06ffc-094e-3242-8260-48d89e756724
placement:
  clusterName: srirams-dplab
  clusterUuid: 77799ad0-0caf-41f8-9518-155a2eb85c00
reference:
  jobId: a073986c8773434ab7fc77441d224178
  projectId: cloud-nurani-srirams
sparkJob:
  args:
  - '1000'
  jarFileUris:
  - file:///usr/lib/spark/examples/jars/spark-examples.jar
  mainClass: org.apache.spark.examples.SparkPi
status:
  state: DONE
  stateStartTime: '2024-12-02T07:01:41.349162Z'
statusHistory:
- state: PENDING
  stateStartTime: '2024-12-02T07:00:43.211570Z'
- state: SETUP_DONE
  stateStartTime: '2024-12-02T07:00:43.240450Z'
- details: Agent reported job success
  state: RUNNING
  stateStartTime: '2024-12-02T07:00:43.427447Z'
yarnApplications:
- name: Spark Pi
  progress: 1.0
[ Read 55 lines ]
```

9.3.9 Clean up

9.4g: Dataflow

9.4.1 Dataflow Lab #1 (Java package popularity)

9.4.2 Setup

9.4.3 Beam code

- **Where is the input taken from by default?**

home > srirams > training-data-analyst > courses > data_analysis > lab2 > python > is_popular.py > ...

```python
def getPackages(line, keyword):
    start = line.find(keyword) + len(keyword)
    end = line.find(';', start)
    if start < end:
        packageName = line[start:end].strip()
        return splitPackageName(packageName)
    return []

def packageUse(line, keyword):
    packages = getPackages(line, keyword)
    for p in packages:
        yield (p, 1)

if __name__ == '__main__':
    parser = argparse.ArgumentParser(description='Find the most used Java packages')
    parser.add_argument('--output_prefix', default='/tmp/output', help='Output prefix')
    parser.add_argument('--input', default='../javahelp/src/main/java/com/google/cloud/training/dataanalyst/javahelp/', help='Input directory')

    options, pipeline_args = parser.parse_known_args()
    p = beam.Pipeline(argv=pipeline_args)

    input = '{0}*.java'.format(options.input)
    output_prefix = options.output_prefix
    keyword = 'import'
```

- **Where does the output go by default?**

Welcome | index.html | ~~index.js~~ 1 | is_popular.py

home > srirams > training-data-analyst > courses > data_analysis > lab2 > python > is_popular.py > ...

```python
def getPackages(line, keyword):
    start = line.find(keyword) + len(keyword)
    end = line.find(';', start)
    if start < end:
        packageName = line[start:end].strip()
        return splitPackageName(packageName)
    return []

def packageUse(line, keyword):
    packages = getPackages(line, keyword)
    for p in packages:
        yield (p, 1)

if __name__ == '__main__':
    parser = argparse.ArgumentParser(description='Find the most used Java packages')
    parser.add_argument('--output_prefix', default='/tmp/output', help='Output prefix')
    parser.add_argument('--input', default='../javahelp/src/main/java/com/google/cloud/training/dataanalyst/javahelp/', help='Input directory')

    options, pipeline_args = parser.parse_known_args()
    p = beam.Pipeline(argv=pipeline_args)

    input = '{0}*.java'.format(options.input)
    output_prefix = options.output_prefix
    keyword = 'import'

    # find most used packages
    (p
```

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

The packageUse() transform processes lines containing the specified keyword (import) by extracting package names, splitting them into their hierarchical components, and emitting each component along with a count of 1 for subsequent aggregation.

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

Beam's CombinePerKey is used to aggregate values for each key in a collection of key-value pairs. The TotalUse operation sums up the counts for each package name, effectively calculating the total number of occurrences of each package across all input lines.

- **Which operations correspond to a "Map"?**

GetImports , PackageUse

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

  TotalUse' > beam.CombinePerKey(sum)

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

  'TotalUse' >> beam.CombinePerKey(sum)

9.4.4 Run pipeline locally

- **Take a screenshot of its contents**

```
(env) srirams@cloudshell:~/training-data-analyst/courses/machine_learning/deepdive/04_features/dataflow/python (cloud-nurani-srirams)$ cat /tmp/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) srirams@cloudshell:~/training-data-analyst/courses/machine_learning/deepdive/04_features/dataflow/python (cloud-nurani-srirams)$
```

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

The output file contains the top 5 most commonly used Java package prefixes found in the analyzed Java files. Each entry in the output is a tuple comprising a package prefix and its associated usage count.

9.4.5 Dataflow Lab #2 (Word count)

- **What are the names of the stages in the pipeline?**

  Read Stage, Split Stage, PairWithOne Stage, GroupAndSum Stage, Format Stage, Write Stage

- **Describe what each stage does.**

  Read Stage: Loads the input text file into a PCollection.

  Split Stage: Breaks each line into individual words using a regular expression.

  PairWithOne Stage: Associates each word with a key-value pair, where the word is the key and the value is 1.

  GroupAndSum Stage: Groups the key-value pairs by word and computes the total count for each key.

  Format Stage: Converts the word count results into formatted strings.

Write Stage: Outputs the formatted word count to a text file.

9.4.6 Run code locally

- **Use wc with an appropriate flag to determine the number of different words in King Lear.**

```
(env) sriramg@cloudshell:~/training-data-analyst/courses/machine_learning/deepdive/04_features/dataflow/python (cloud-nurani-srirams)$ wc -l outputs-00000-of-00001
4784 outputs-00000-of-00001
```

- **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**

```
(env) sriramg@cloudshell:~/training-data-analyst/courses/machine_learning/deepdive/04_features/dataflow/python (cloud-nurani-srirams)$ sort -t: -k2,2nr outputs-00000-of-00001 |head -n 3
the: 786
I: 622
and: 594
```

The:786 I: 622 and: 594

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

```
(env) sriramg@cloudshell:~/training-data-analyst/courses/machine_learning/deepdive/04_features/dataflow/python (cloud-nurani-srirams)$ sort -t: -k2,2nr outputs-00000-of-00001 |head -n 3
the: 908
and: 738
i: 622
(env) sriramg@cloudshell:~/training-data-analyst/courses/machine_learning/deepdive/04_features/dataflow/python (cloud-nurani-srirams)$
```

The:908 I: 738 and: 622

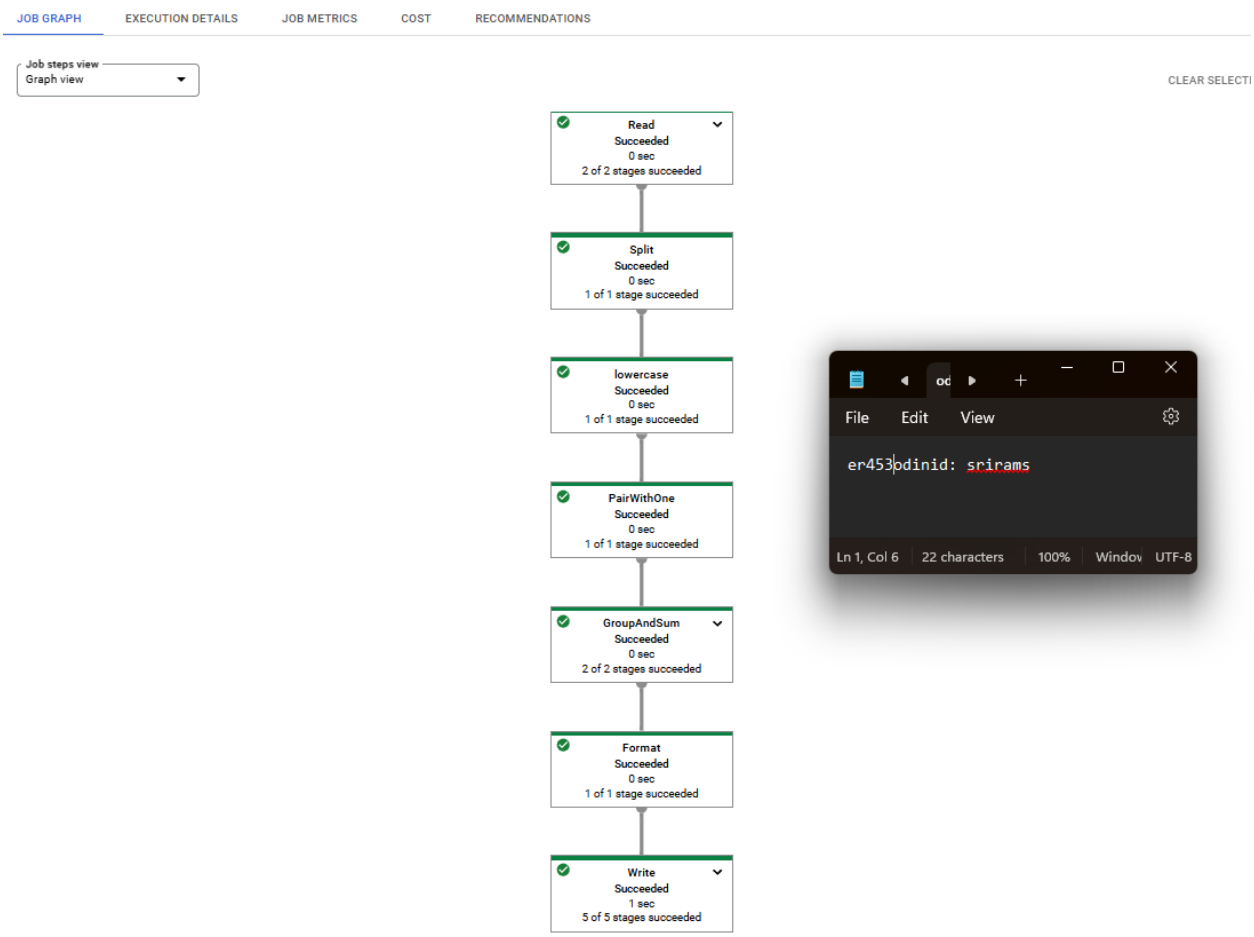
9.4.7 Setup for Cloud Dataflow

9.4.8 Service account setup

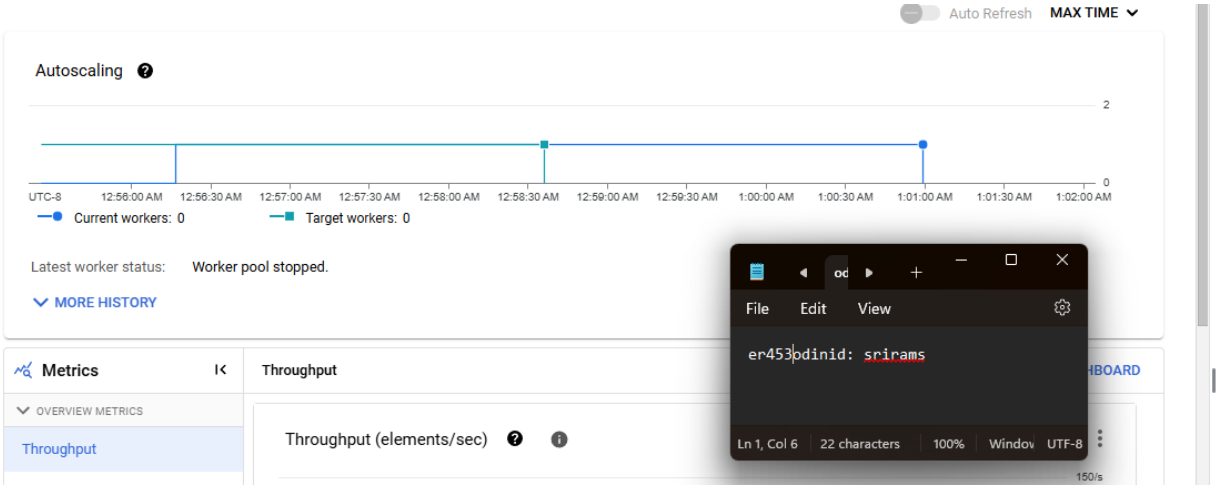
9.4.9 Run code using Dataflow runner

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

  Write succeeded it took 1 seconds

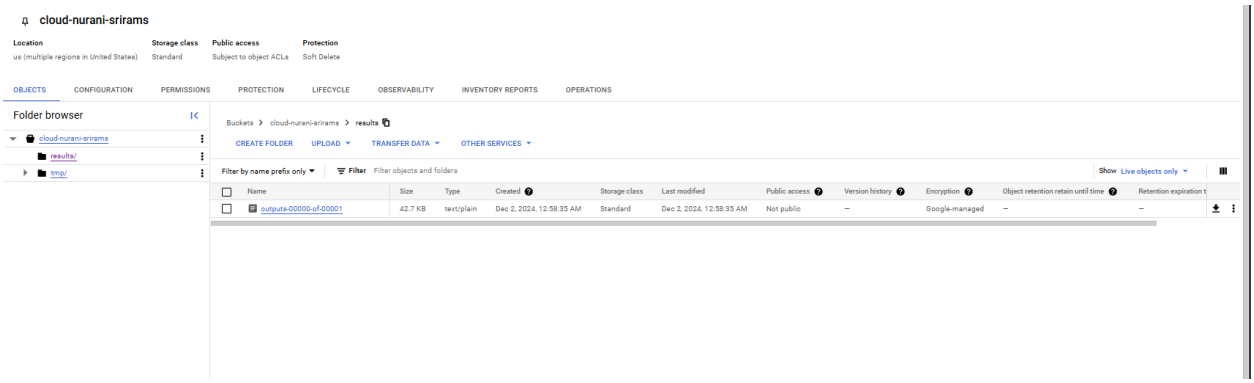


- **The autoscaling graph showing when the worker was created and stopped.**



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

One

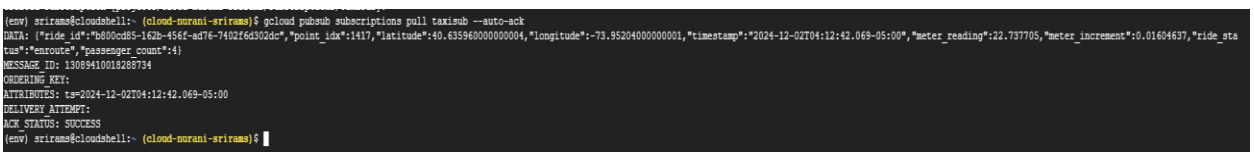


9.4.10 Clean up

9.4.11 Dataflow Lab #3 (Taxi ETL pipeline)

9.4.12 View raw data from PubSub

- **Take a screenshot listing the different fields of this object.**

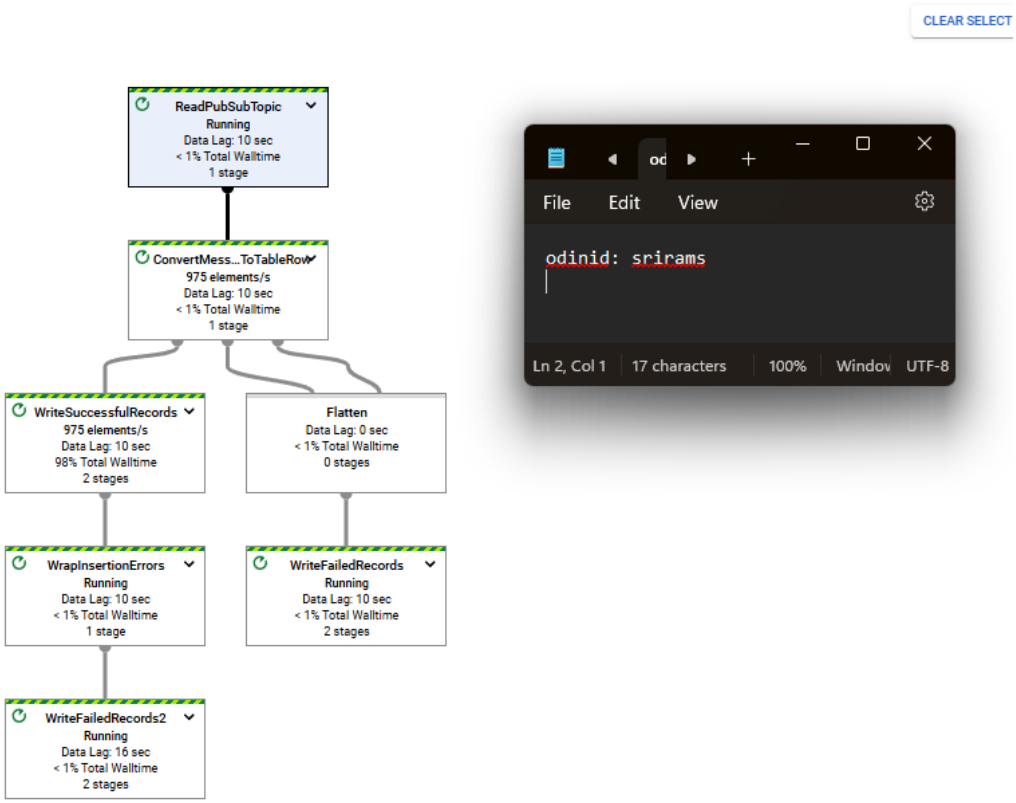


```
(env) srirams@cloudshell:~ (cloud-nurani-srirams)$ gcloud pubsub subscriptions pull taxisub --auto-ack
DATA: {"ride_id":"b800cd85-162b-456f-ad76-7402f6d302dc","point_idx":1417,"latitude":40.635960000000004,"longitude":-73.95204000000001,"timestamp":"2024-12-02T04:12:42.069-05:00","meter_reading":22.737705,"meter_increment":0.01604637,"ride_sta
tus":"enroute","passenger_count":4}
MESSAGE_ID: 13089410018288734
ORDERING_KEY:
ATTRIBUTES: ts=2024-12-02T04:12:42.069-05:00
DELIVERY_ATTEMPT:
ACK_STATUS: SUCCESS
(env) srirams@cloudshell:~ (cloud-nurani-srirams)$
```

9.4.13 BigQuery and Dataflow setup

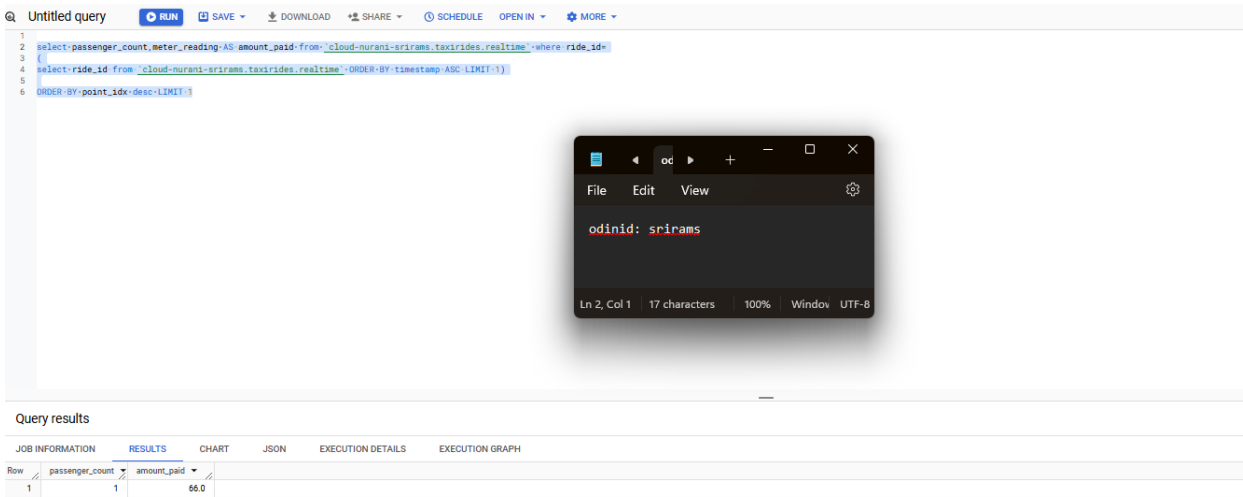
9.4.14 Run Dataflow job from template

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

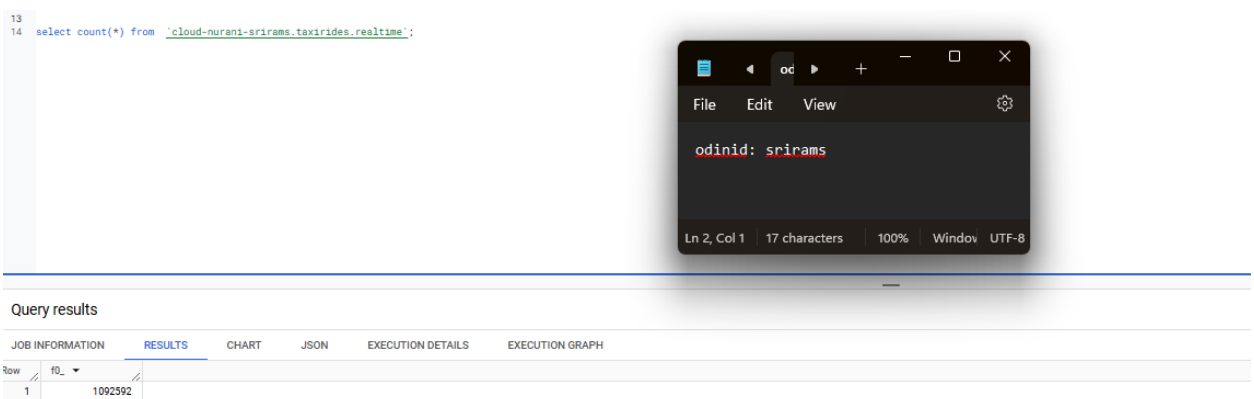


9.4.15 Query data in BigQuery

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

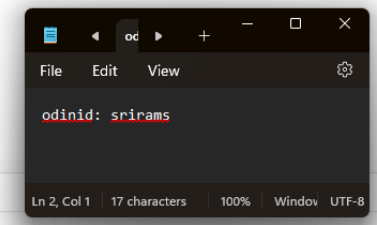


- **Take a screenshot showing the estimated number of rows in the table.**



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

```sql
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
`cloud-nurani-srirams.taxirides.realtime`
WHERE
  ride_status = 'dropoff'
GROUP BY
  minute
ORDER BY
  minute ASC;
```



Query results

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	minute	total_rides	total_passengers	total_revenue
1	02:12	30	33	645.2400003
2	02:13	71	101	1143.2000027
3	02:14	79	100	2065.8799901
4	02:15	81	115	1818.55000319...
5	02:16	89	120	1786.75000529...
6	02:17	66	95	1118.17000709...
7	02:18	93	147	1786.96000389...
8	02:19	83	133	1716.45000779...
9	02:20	79	123	1467.8899949
10	02:21	93	145	1600.48000429...
11	02:22	92	154	1585.67000010...
12	02:23	77	145	1458.7000033
13	02:24	102	170	1694.78000379...
14	02:25	93	142	1863.21999769...
15	02:26	94	161	1637.01999869...
16	02:27	99	148	1619.46999439...
17	02:28	100	147	1885.59000609...
18	02:29	96	168	1782.58000249...
19	02:30	98	149	1890.51000039...
20	02:31	2	7	17.6

9.4.16 Data visualization

- **Take a screenshot showing the plot for your data for your lab notebook**

9.4.17 Clean up