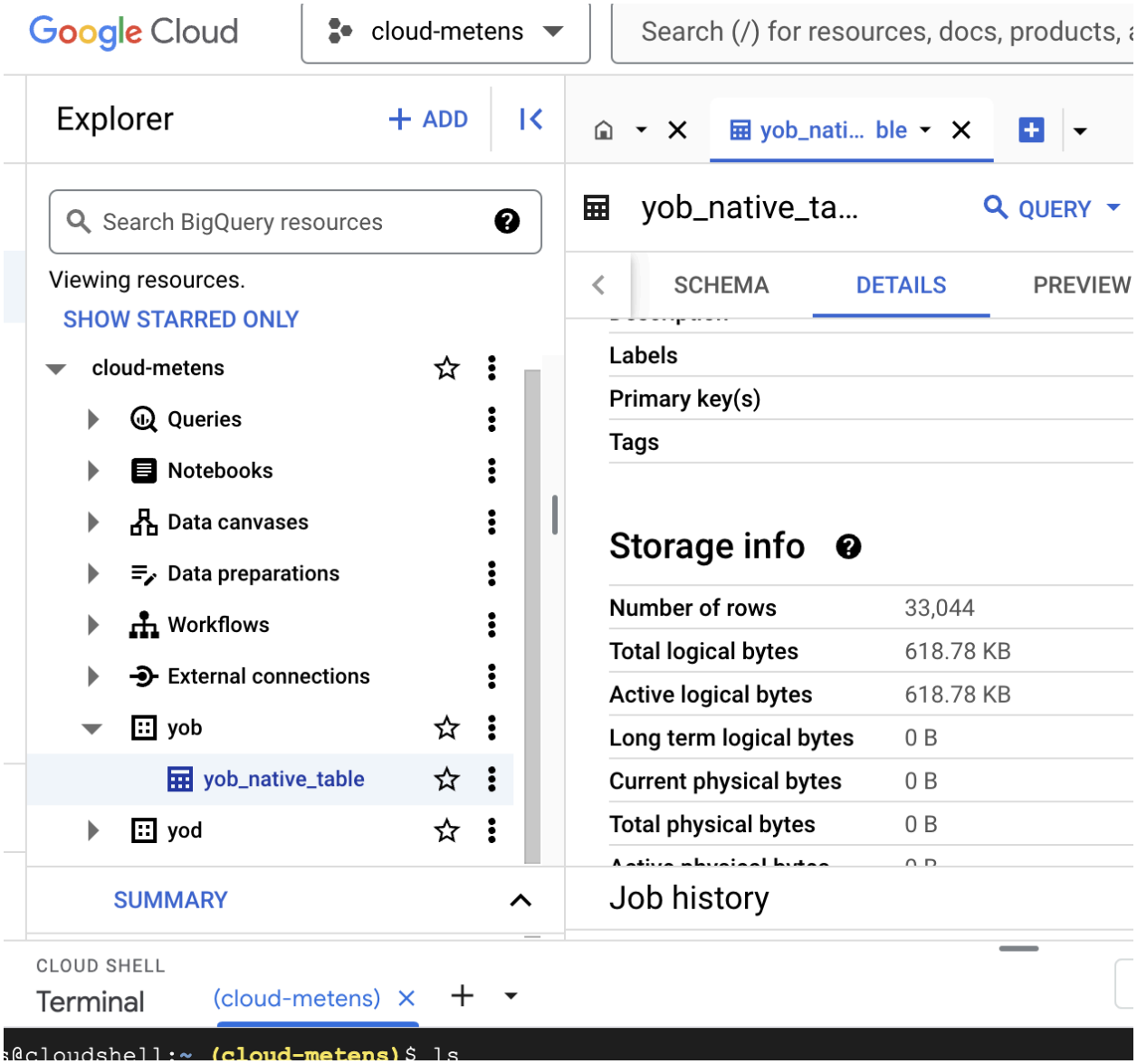
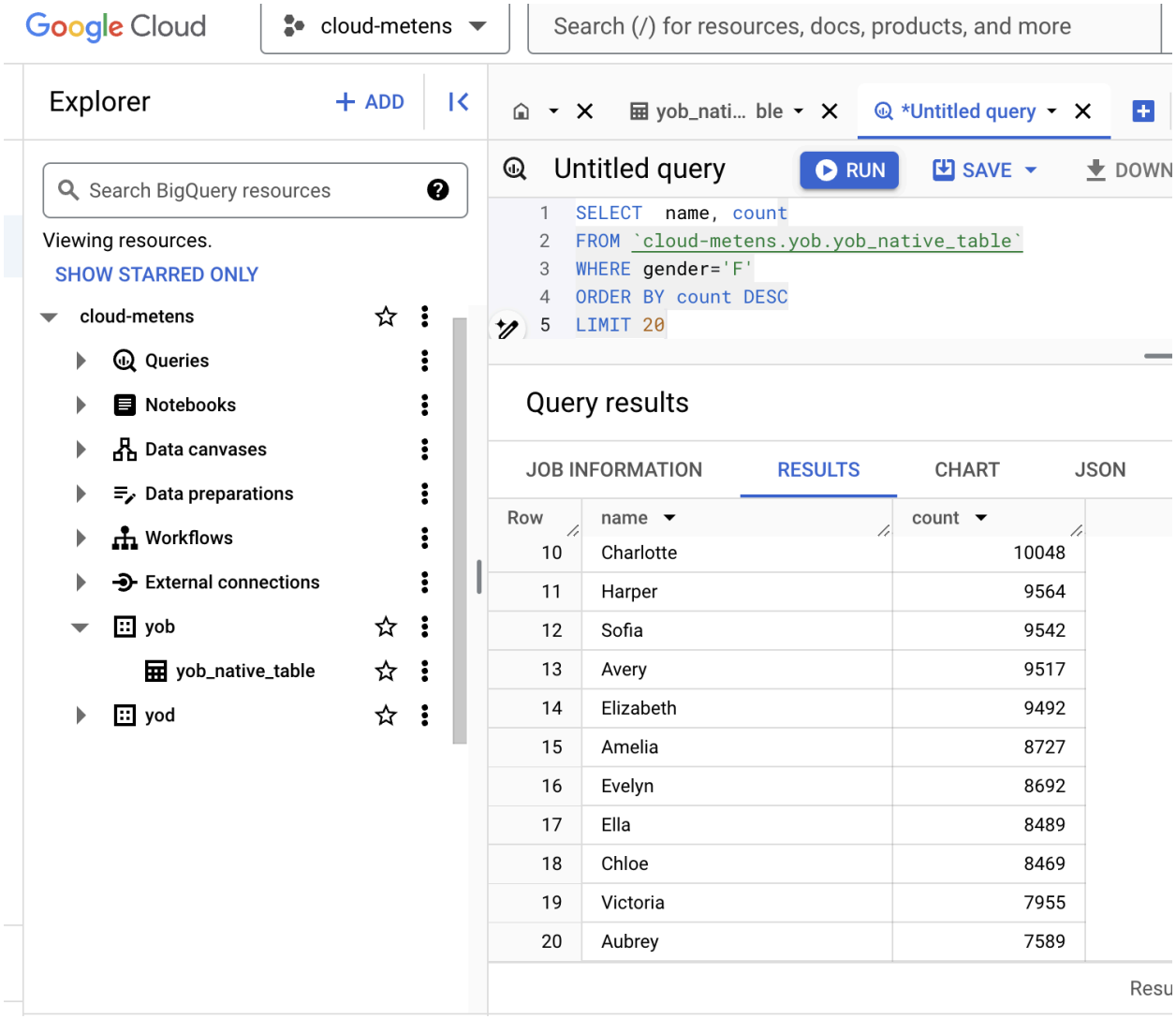
Week #9 Labs

9.1g: BigQuery, BigLake

3. Create dataset



4. Query data

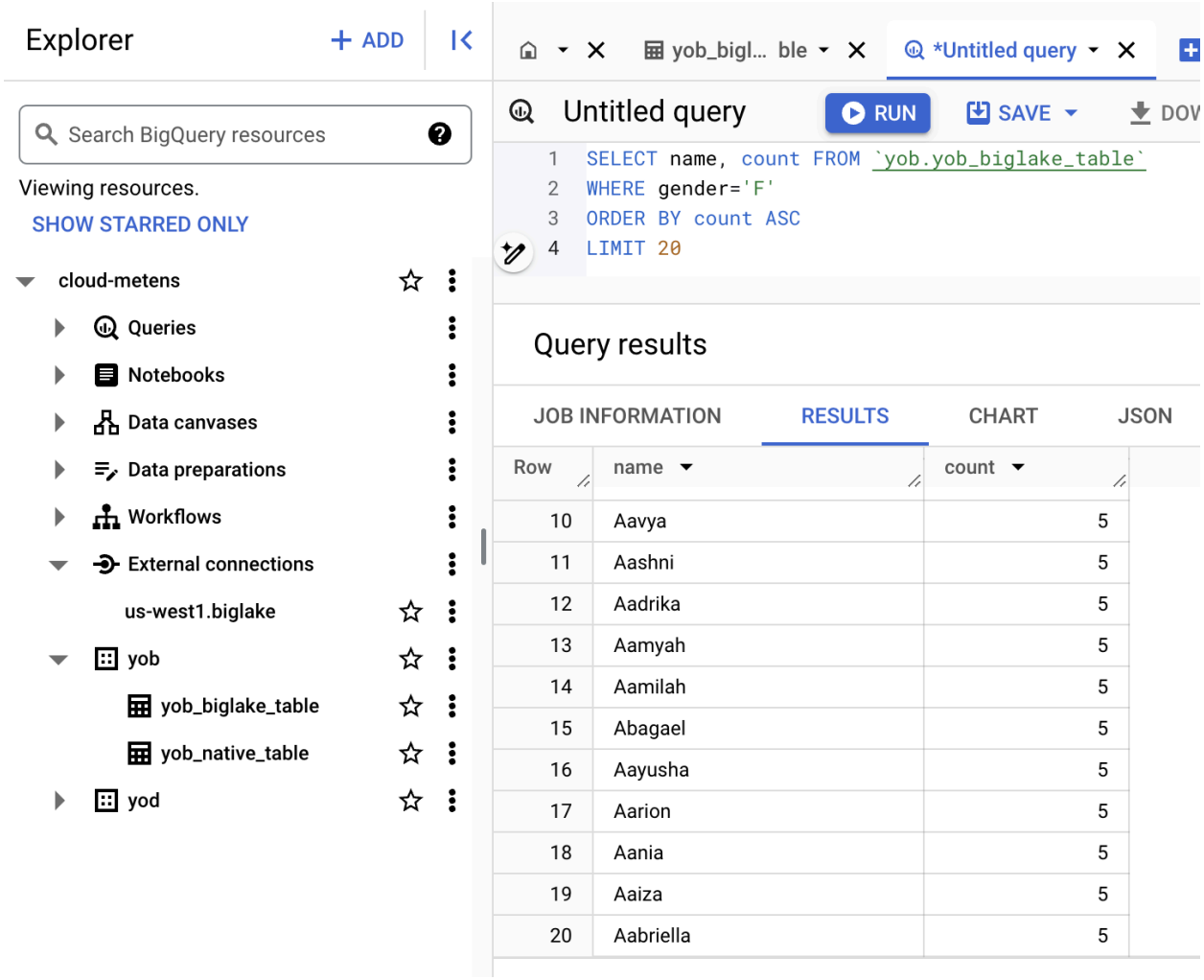

Google Cloud
cloud-metens
Search (/) for resources, docs, products, and more
Explorer
+ ADD
Search BigQuery resources
Viewing resources.
SHOW STARRED ONLY
cloud-metens
Queries
Notebooks
Data canvases
Data preparations
Workflows
External connections
yob
yob_native_table
yod
yob_nati... ble
*Untitled query
Untitled query
RUN
SAVE
DOWN
1 SELECT name, count
2 FROM `cloud-metens.yob.yob_native_table`
3 WHERE gender='F'
4 ORDER BY count DESC
5 LIMIT 20
Query results
JOB INFORMATION
RESULTS
CHART
JSON
Row
name
count
10 Charlotte 10048
11 Harper 9564
12 Sofia 9542
13 Avery 9517
14 Elizabeth 9492
15 Amelia 8727
16 Evelyn 8692
17 Ella 8489
18 Chloe 8469
19 Victoria 7955
20 Aubrey 7589
Resu

```
metens@cloudshell:~ (cloud-metens)$ wc -l yob2014.csv
33044 yob2014.csv
metens@cloudshell:~ (cloud-metens)$ bq query "SELECT  name, count
FROM [cloud-metens.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 |
+---------+-------+
metens@cloudshell:~ (cloud-metens)$
```

```
cloud-metens> SELECT  name, count FROM [cloud-metens.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-metens>
```

```
cloud-metens> SELECT name, count FROM [cloud-metens.yob.yob_native_table] WHERE name='Nathan'
+--------+-------+
|  name  | count |
+--------+-------+
| Nathan |     8 |
| Nathan |  8902 |
+--------+-------+
cloud-metens>
```

9. Query data



9.2g: Jupyter Notebooks

3. BigQuery query

The size of the table is about 22GB and the size of the query is about 3GB. So the query processes about 7 times less data than the size of the table.
Between 2001 and 2003, there were 375362 twins born.
On average, twins are about 5 pounds each while single babies are about 7 pounds in weight.

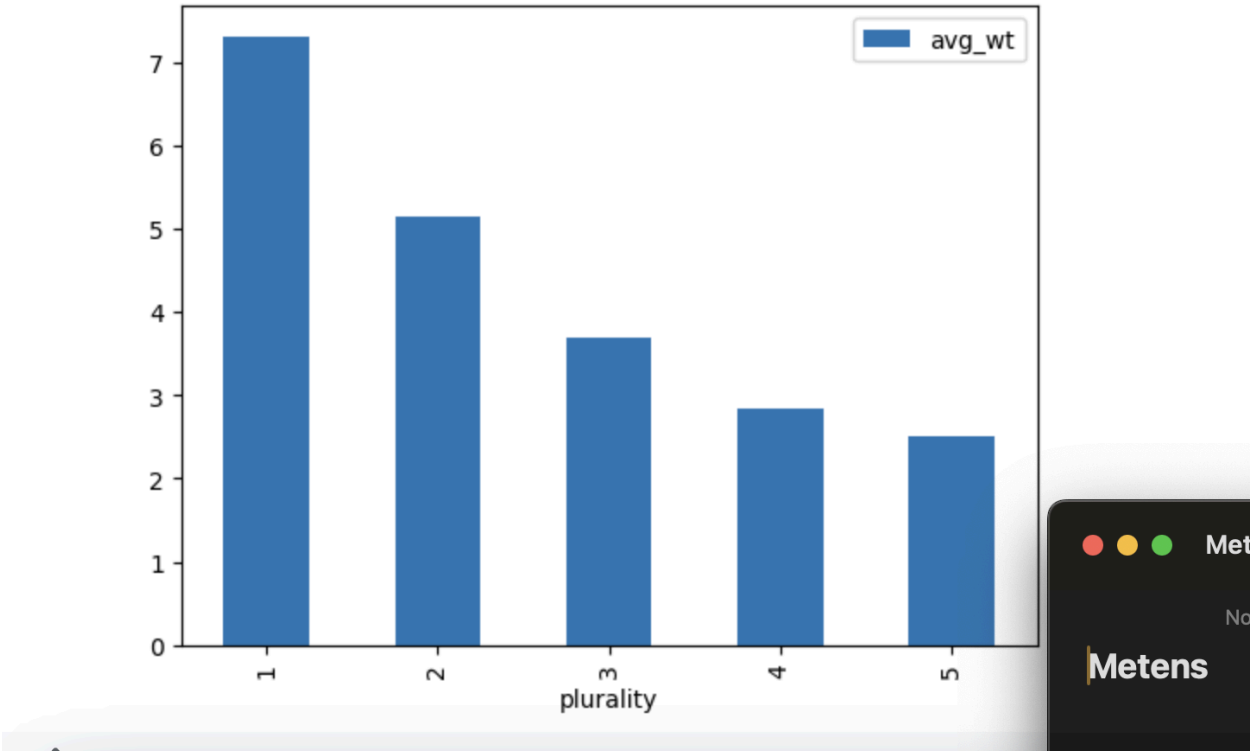
6. Run queries

The two features with the strongest prediction of a babies weight are: **plurality** and **gestation weeks**.

```
    """
    return bigquery.Client().query(query_string).to_dataframe().sort_values(column_n
```

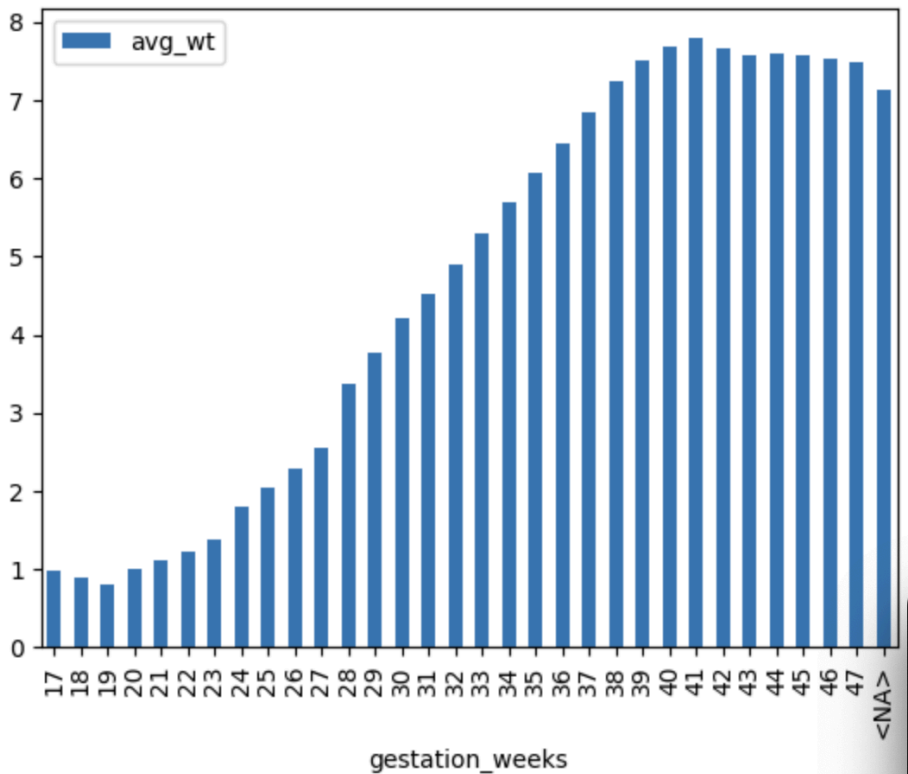
[7]:
```
df = get_distinct_values('plurality')
df.plot(x='plurality', y='avg_wt', kind='bar')
```

[7]:
```
<Axes: xlabel='plurality'>
```



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

```
[9]: <Axes: xlabel='gestation_weeks'>
```



8. Mobility

The day that had the largest spike in trips to grocery and pharmacy stores (17) was **2020-03-13**. On the day the stay-at-home order took effect (3/23/2020), the total impact on workplace trips was **-49 from baseline**.

9. Airport traffic

The three airports that were impacted the most in April were:

1) Detroit Metropolitan Wayne County with 45 % or normal traffic
2) McCarran International with 45 %
3) San Francisco International with 47 %

In August, the three airports were the same, but their percentages were only slightly different:

1) Detroit Metropolitan Wayne County with 45 % or normal traffic
2) McCarran International with 44 %
3) San Francisco International with 53 %

10. Mortality

The table **excess_deaths** has columns: **placename, start_date,** and **excess_deaths**.
The table **us_counties** has columns: **date**, **county**, and **deaths**.
The table **us_states** has columns: **date**, **state_name**, and **confirmed_cases**.
The table **mask_use_by_county** has columns: **county_fips_code** and **never, rarely, sometimes, frequently, always**.

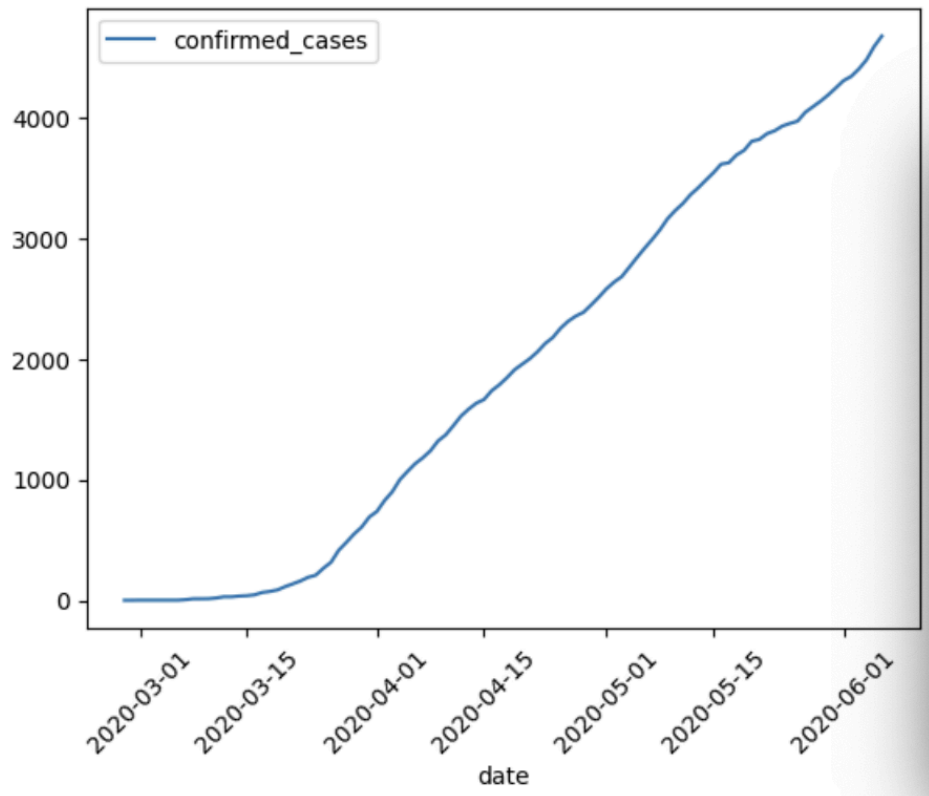
11. Run example queries

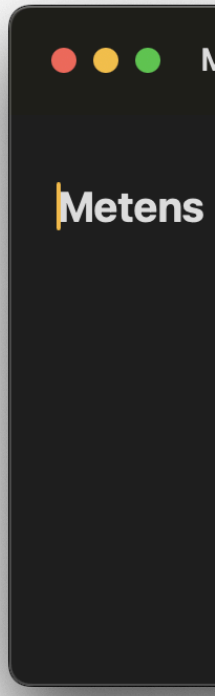
[16]:
```
query_string = """SELECT date, confirmed_cases
FROM `bigquery-public-data.covid19_nyt.us_states`
WHERE state_name = 'Oregon'
ORDER BY date ASC"""

from google.cloud import bigquery
df = bigquery.Client().query(query_string + " LIMIT 100").to_dataframe()
```

[17]:
```
df.plot(x='date', y='confirmed_cases', kind='line', rot=45)
```

[17]:
```
<Axes: xlabel='date'>
```

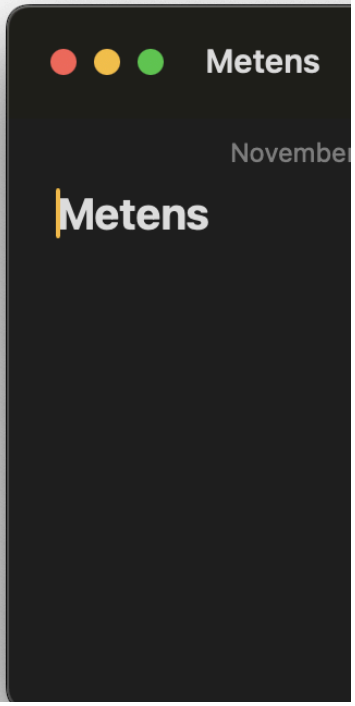




[20]:
```
query_string = """SELECT state_name, MIN(date) as date_of_1000
FROM `bigquery-public-data.covid19_nyt.us_states`
WHERE deaths > 1000
GROUP BY state_name
ORDER BY date_of_1000 ASC"""
from google.cloud import bigquery
df = bigquery.Client().query(query_string + " LIMIT 100").to_data
df.head(10)
```

[20]:

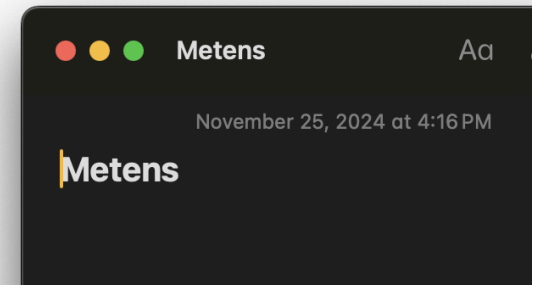
	state_name	date_of_1000
0	New York	2020-03-29
1	New Jersey	2020-04-06
2	Michigan	2020-04-09
3	Louisiana	2020-04-14
4	Massachusetts	2020-04-15
5	Illinois	2020-04-16
6	Connecticut	2020-04-17
7	Pennsylvania	2020-04-17
8	California	2020-04-17
9	Florida	2020-04-24



[27]:
```
query_string = """SELECT DISTINCT mu.county_fips_code, mu.always, ct.county, state_name
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 100").to_dataframe()
df.head()
```

[27]:

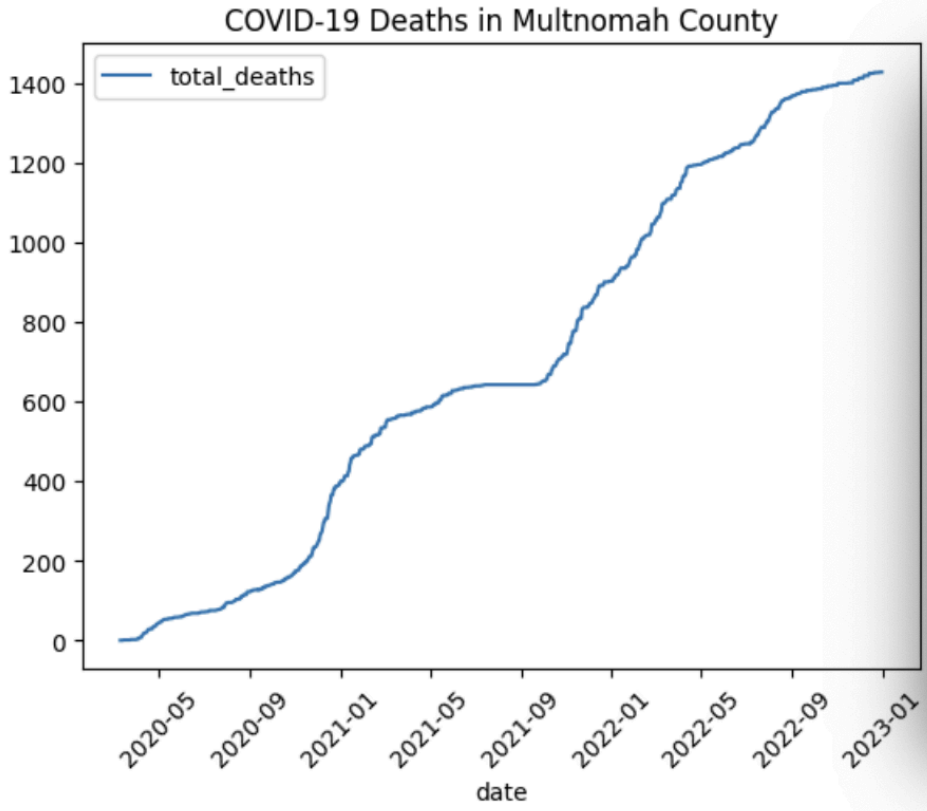
	county_fips_code	always	county	state_name
0	06027	0.889	Inyo	California
1	36123	0.884	Yates	New York
2	48229	0.880	Hudspeth	Texas
3	06051	0.880	Mono	California
4	48141	0.877	El Paso	Texas

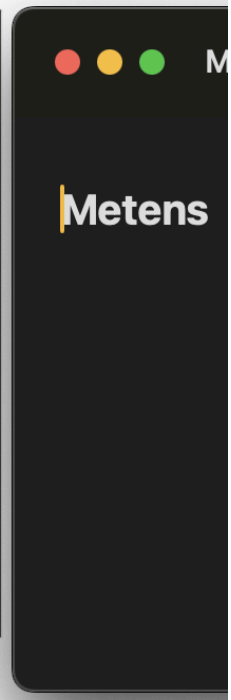


12. Write queries

[47]:
```
query_string = """
SELECT date, SUM(deaths) AS total_deaths
FROM `bigquery-public-data.covid19_nyt.us_counties`
WHERE county='Multnomah'
GROUP BY date
ORDER BY date ASC
"""
df = bigquery.Client().query(query_string).to_dataframe()
df.plot(x='date', y='total_deaths', rot=45, title='COVID-19 Deaths in Multnomah County'
```

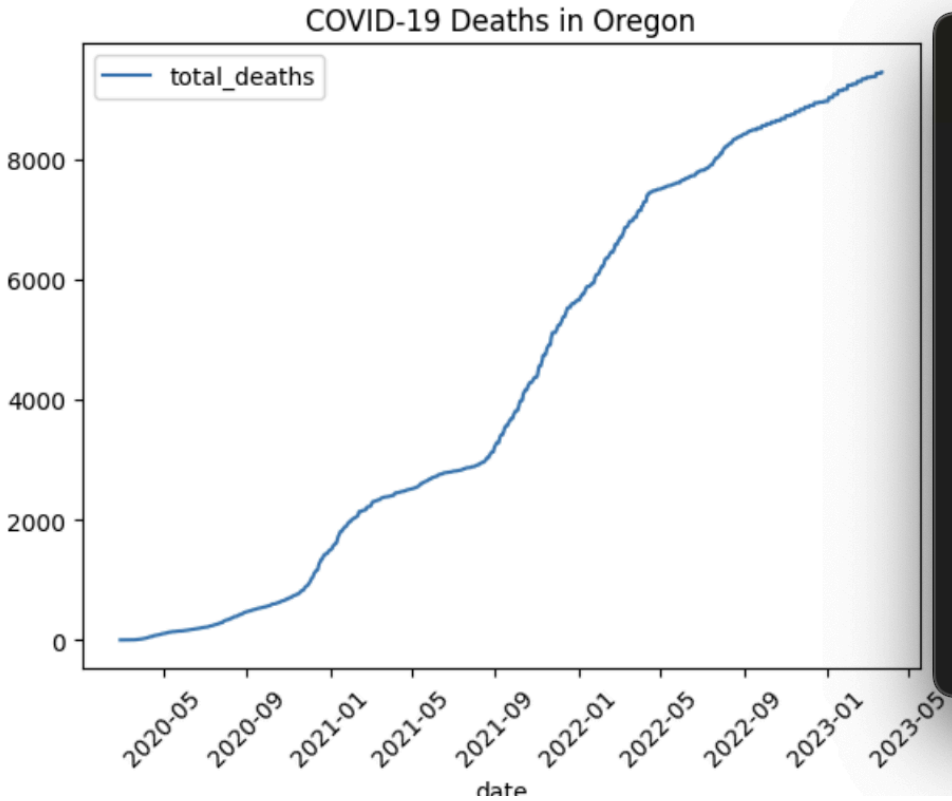
[47]:
```
<Axes: title={'center': 'COVID-19 Deaths in Multnomah County'}, xlabel='date'>
```





[49]:
```
query_string = """
SELECT date, SUM(deaths) AS total_deaths
FROM `bigquery-public-data.covid19_nyt.us_states`
WHERE state_name='Oregon'
GROUP BY date
ORDER BY date ASC
"""
df = bigquery.Client().query(query_string).to_dataframe()
df.plot(x='date', y='total_deaths', rot=45, title='COVID-19 Deaths in Oregon')
```

[49]:
```
<Axes: title={'center': 'COVID-19 Deaths in Oregon'}, xlabel='date'>
```



9.3g: Dataproc

6. Run computation

```
Waiting for cluster creation operation...done.
Created [https://dataproc.googleapis.com/v1/projects/cloud-metens/regions/us-west1/clusters/metens-dplab] Cluster placed in zone [us-west1-b].
metens@cloudshell:~ (cloud-metens)$ date

gcloud dataproc jobs submit spark --cluster ${CLUSTERNAME} \
  --class org.apache.spark.examples.SparkPi \
  --jars file:///usr/lib/spark/examples/jars/spark-examples.jar -- 1000 \
  >& output.txt &
Wed Nov 27 02:13:40 AM UTC 2024
[1] 2100
metens@cloudshell:~ (cloud-metens)$ gcloud dataproc jobs list --cluster ${CLUSTERNAME} ; date
JOB_ID: 85c330341393475480b9447e2c6f2e16
TYPE: spark
STATUS: SETUP_DONE
Wed Nov 27 02:13:49 AM UTC 2024
metens@cloudshell:~ (cloud-metens)$ gcloud dataproc jobs list --cluster ${CLUSTERNAME} ; date
JOB_ID: 85c330341393475480b9447e2c6f2e16
TYPE: spark
STATUS: SETUP_DONE
Wed Nov 27 02:14:00 AM UTC 2024
metens@cloudshell:~ (cloud-metens)$ gcloud dataproc jobs list --cluster ${CLUSTERNAME} ; date
JOB_ID: 85c330341393475480b9447e2c6f2e16
TYPE: spark
STATUS: SETUP_DONE
Wed Nov 27 02:14:13 AM UTC 2024
metens@cloudshell:~ (cloud-metens)$ gcloud dataproc jobs list --cluster ${CLUSTERNAME} ; date
JOB_ID: 85c330341393475480b9447e2c6f2e16
TYPE: spark
STATUS: RUNNING
Wed Nov 27 02:14:29 AM UTC 2024
metens@cloudshell:~ (cloud-metens)$ gcloud dataproc jobs list --cluster ${CLUSTERNAME} ; date
JOB_ID: 85c330341393475480b9447e2c6f2e16
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-exa
mples.jar -- 1000 >&output.txt
Wed Nov 27 02:15:17 AM UTC 2024
```

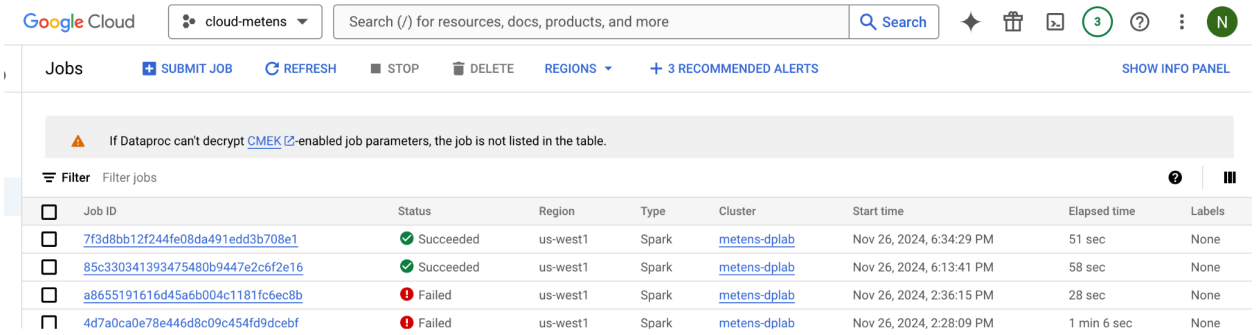
The job started at 02:13:40 and ended at 02:15:17. **It took less than 1 minute and 37 seconds to finish**.

Output.txt:

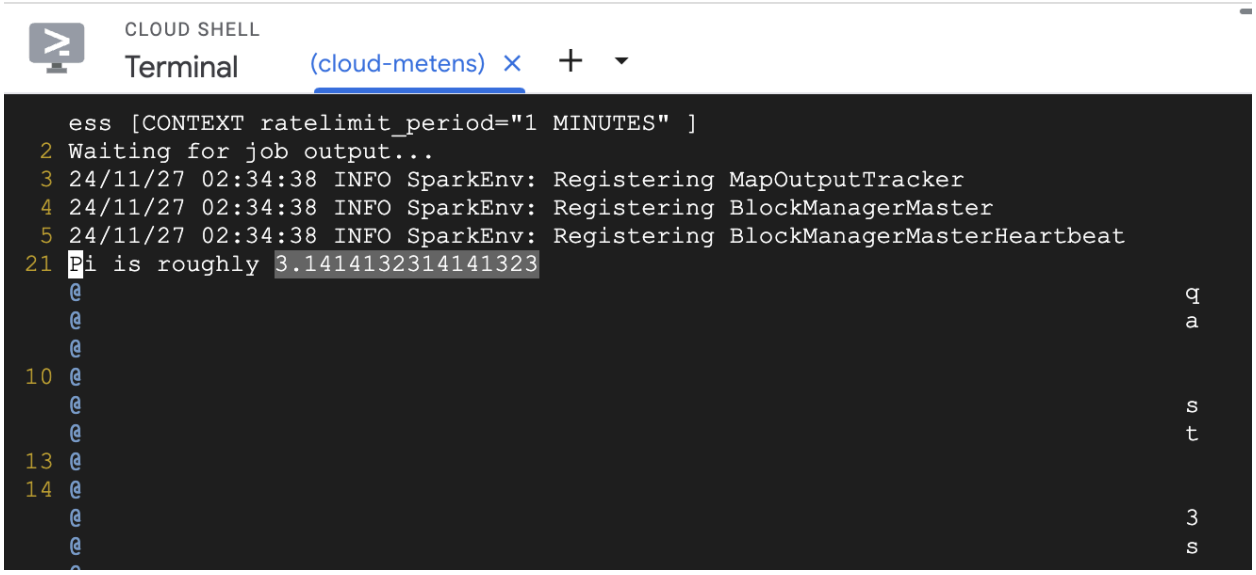
```
Terminal  (cloud-metens)                                                                                                    Open Editor
ean=71, gcs_met
adata_request_mean=59, op_list_status_mean=0, stream_read_close_operations_mean=0, op_xattr_get_named_map_mean=0, stream_read_vectored_operations_mean=0, op_mkdirs_mean
=315, action_ht
tp_post_request_mean=66, stream_write_close_operations_mean=0, action_http_put_request_mean=132, action_http_patch_request_mean=0, op_hsync_mean=0, delegation_tokens_is
sued_mean=0, ac
tion_http_delete_request_mean=60, stream_read_operations_mean=0, op_create_mean=169, op_delete_mean=0, op_create_non_recursive_mean=0, stream_readVectored_range_duratio
n_mean=0, op_xa
ttr_get_named_mean=0, gcs_get_media_request_mean=0, op_get_file_status_mean=1905, op_delete_duration=0, op_get_file_status_duration=1905, action_http_put_request_durati
on=398, stream_
write_operations_duration=0, op_hsync_duration=0, gcs_metadata_request_duration=1006, gcs_get_media_request_duration=0, gcs_list_file_request_duration=401, op_list_stat
us_duration=0,
op_mkdirs_duration=315, op_open_duration=0, op_create_duration=169, op_hflush_duration=1211, gcs_list_dir_request_duration=0, op_glob_status_duration=0, stream_read_ope
rations_duratio
n=0, action_http_delete_request_duration=120, action_http_post_request_duration=464, op_rename_duration=0]
Job [85c330341393475480b9447e2c6f2e16] finished successfully.
done: true
driverControlFilesUri: gs://dataproc-staging-us-west1-660633817405-kvwbi0ki/google-cloud-dataproc-metainfo/94e2d49e-37b8-4068-9c6b-980241e6ede9/jobs/85c330341393475480b
9447e2c6f2e16/
 21 Pi is roughly 3.141640591416406
 22 24/11/27 02:14:36 INFO RequestTracker: Detected high latency for [url=https://storage.googleapis.com/upload/storage/v1/b/dataproc-temp-us-west1-660633817405-sszii01
```

Pi is roughly 3.141640591416406.

8. Run computation again



The first run took 58 seconds, and this run took **51 seconds**. It was **7 seconds faster**.



Pi is roughly **3.1414132314141323**.

9.4g: Dataflow

3. Beam code

The input variable: `input = '{0}*.java'.format(options.input)` comes from `options, pipeline_args = parser.parse_known_args()` which has `parser = argparse.ArgumentParser(description='Find the most used Java packages')` . So the input is taken from java files in the directory (../javahelp/src/main/java/com/google/cloud/training/dataanalyst/javahelp/) which is captured by the parser variable.

The call: `'write' >> beam.io.WriteToText(output_prefix)` uses the variable: `output_prefix = options.output_prefix` which is derived from this: `output_prefix = options.output_prefixble` which holds the output prefix is this: `parser.add_argument('--output_prefix', default='/tmp/output', help='Output prefix')` Thus, the output goes to the default '**/tmp/output**' directory.

The **packageUse** method calls the **getPackes** method which calls the **splitPackageName** method. The **packageUse** method gets back a packages variable, which is a list of split packages (e.g. ["com", "com.example", "com.example.appname", "com.example.appname.library", "com.example.appname.library.widgetname"]) and then creates tuples using yield, which outputs all the split packages in tuple form: ("com", 1), ("com.example, 1), etc.

The **totalUse** operation found on line 68: `TotalUse' >> beam.CombinePerKey(sum)`

uses **beam.CombinePerKey** method passing a sum function that essentially calculates how many times the packages appear in the input dataset.

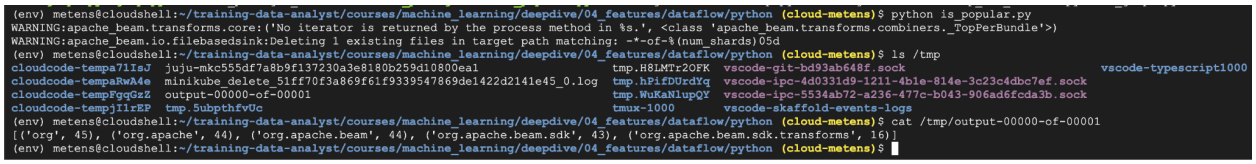
Operations that correspond to a Map are those who implement a one to one operation for each element in a list with all the elements receiving the same treatment. Both the **GetImports** and **PackageUse** operations correspond to a map:

```
'GetImports' >> beam.FlatMap(lambda line: startsWith(line, keyword))
'PackageUse' >> beam.FlatMap(lambda line: packageUse(line, keyword))
```

The **TotalUse** operation corresponds to a shuffle-reduce because each package (key) is assigned a value (1) in the tuple creations and then then TotalUse operation sums the total usage of each package pair using the beam's combinePerKey method.

The operation that corresponds to a reduce is **Top_5** operation. This operation transforms and combines, then captures the 5 most used packages.

4. Run pipeline locally



The output: `[('org', 45), ('org.apache', 44), ('org.apache.beam', 44), ('org.apache.beam.sdk', 43), ('org.apache.beam.sdk.transforms', 16)]` corresponds to the total times that the packages in each tuple in the list were used. The **'org'** Java package was used **45** times. The '**org.apache.beam.sdk.transforms**' package was used **16** times in the local java files.

5. Dataflow Lab #2 (Word count)

The names of the stages in the pipeline are **Read**, **Split**, **PairWithOne**, and **GroupAndSum**, **Format**, and **Write**.

Read: Read the text file[pattern] into a PCollection.
Split: Parse each line of input text into words. Using a regular expression and turns each word into a PCollection of strings.
PairWithOne: Map each string to a key value pair tuple ('string', 1).
GroupAndSum: Sums the toal usage of each string into the tuple for that string.
Format: Output each tuple into a PCollection of strings "word: count" format for easier readability.
Write: Outputs each PCollection of "word: count" into a file with each string in its own line.

6. Run code locally

Since the wc command outputs Lines, Words, and Characters, we can specify words:

```
(env) metens@cloudshell:~/training-data-analyst/courses/machine_learning/deepdive/04_features/dataflow/python (cloud-metens)$ cat outputs-00000-of-00001 | wc
   4784    9568   48944
(env) metens@cloudshell:~/training-data-analyst/courses/machine_learning/deepdive/04_features/dataflow/python (cloud-metens)$ cat outputs-00000-of-00001 | wc -w
9568
(env) metens@cloudshell:~/training-data-analyst/courses/machine_learning/deepdive/04_features/dataflow/python (cloud-metens)$
```

```
(env) metens@cloudshell:~/training-data-analyst/courses/machine_learning/deepdive/04_features/dataflow/python (cloud-metens)$ cat outputs-00000-of-00001 |sort -k2,2nr -t':'|head -n 3
the: 786
I: 622
and: 594
(env) metens@cloudshell:~/training-data-analyst/courses/machine_learning/deepdive/04_features/dataflow/python (cloud-metens)$
```

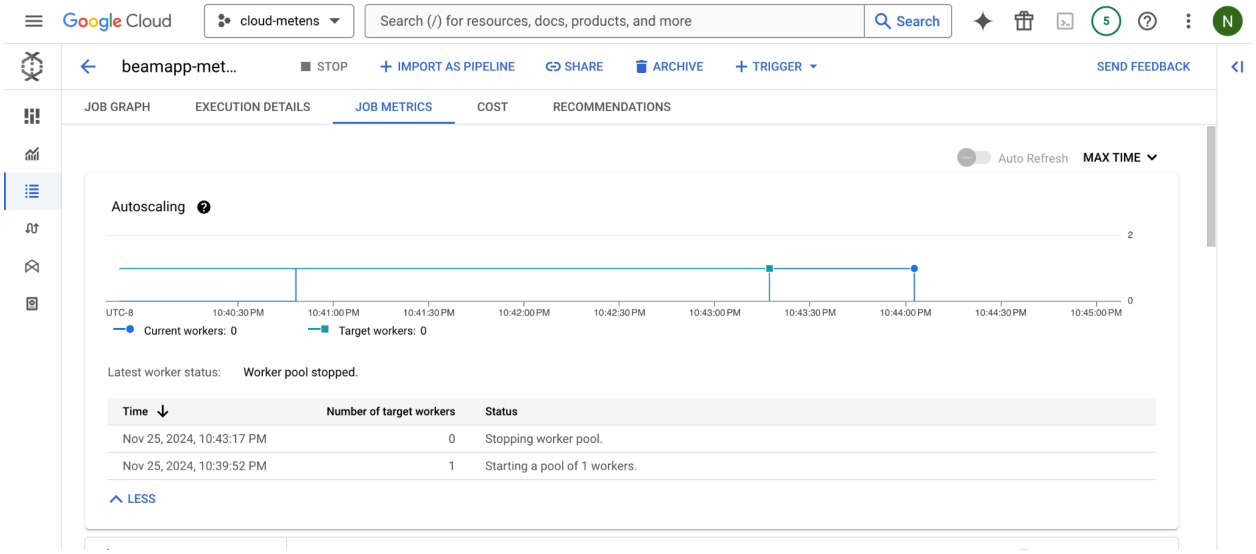
After adding `| 'lowercase' >> beam.Map(lambda x: x.lower())` to the pipeline, we get all words to lowercase:

```
(env) metens@cloudshell:~/training-data-analyst/courses/machine_learning/deepdive/04_features/dataflow/python (cloud-metens)$ cat outputs-00000-of-00001 |sort -k2,2nr -t':'|
head -n 3
the: 908
and: 738
i: 622
```

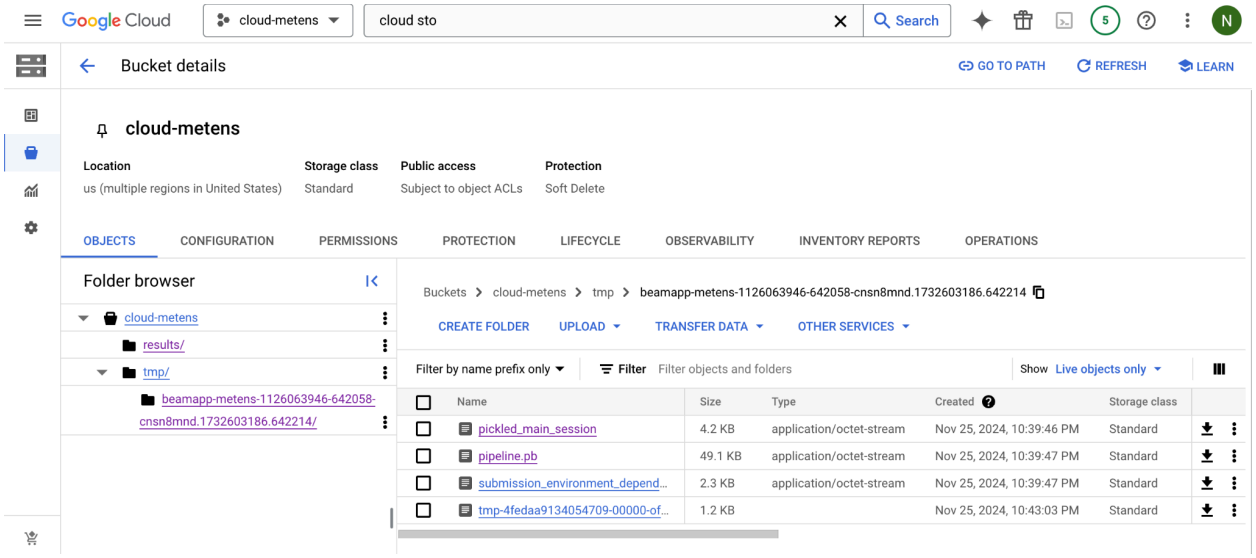
9. Run code using Dataflow runner

The part of the job graph that has taken the longest time to complete was the **Write** operation which took 1 second but had 5 stages, the most out of the other operations.

The autoscaling graph showing when the worker was created and stopped:



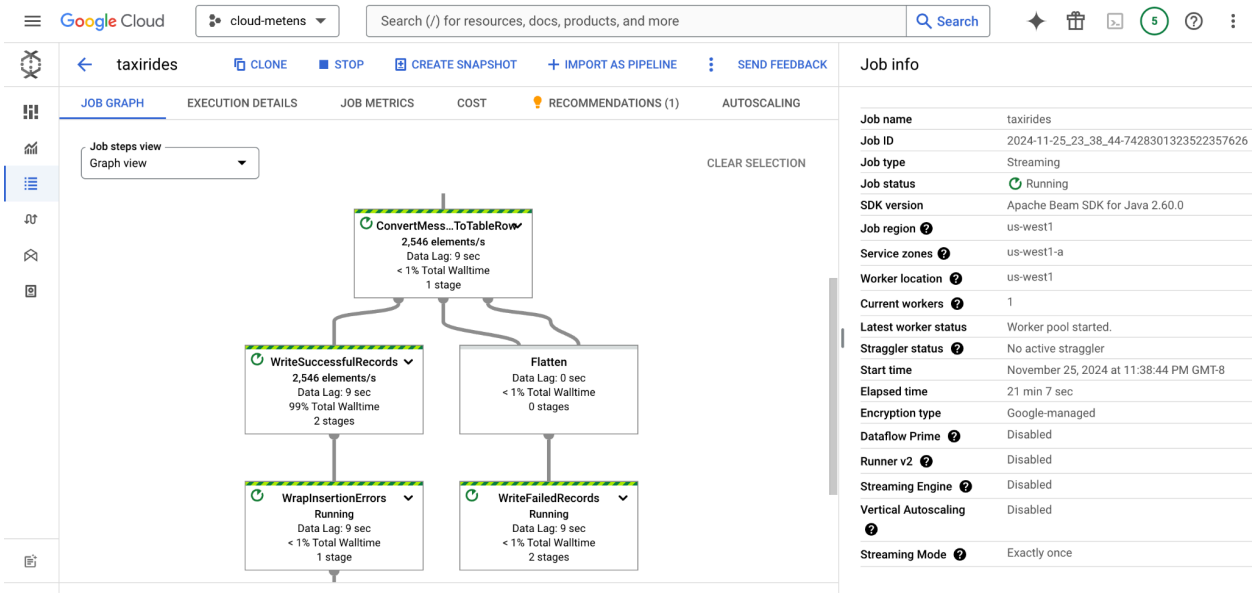
The final write stage in the pipeline created 4 files:



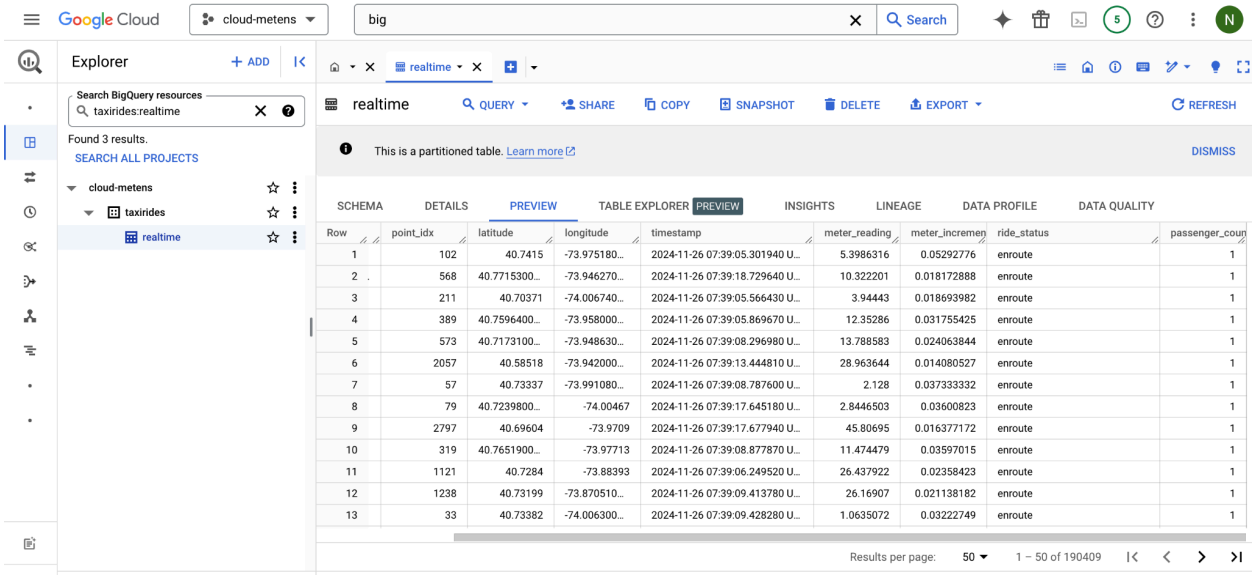
12. View raw data from PubSub

```
(env) metens@cloudshell:~ (cloud-metens)$ gcloud pubsub subscriptions pull taxisub --auto-ack
DATA: {"ride_id":"32140e60-ebf8-4fe8-9b45-861f201b3a3d","point_idx":371,"latitude":40.7657,"longitude":-73.96317,"timestamp":"2024-11-26T02:28:53.01506-05:00","meter_reading":10.516536,"meter_increment":0.028346457,"ride_status":"enroute","passenger_count":1}
MESSAGE_ID: 12752572141366057
ORDERING_KEY:
ATTRIBUTES: ts=2024-11-26T02:28:53.01506-05:00
DELIVERY_ATTEMPT:
ACK_STATUS: SUCCESS
```

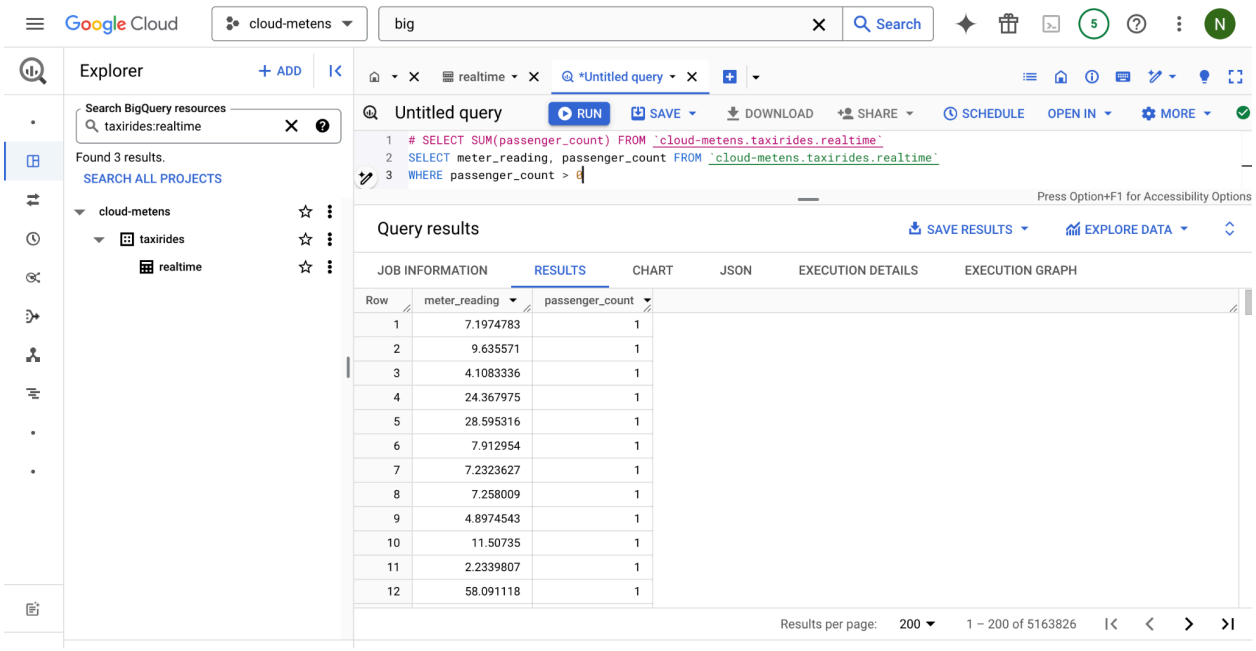
14. Run Dataflow job from template

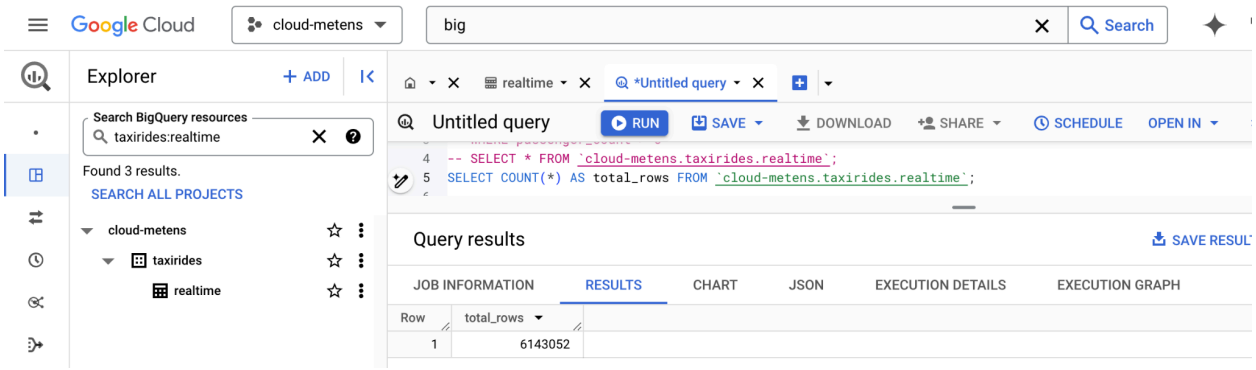


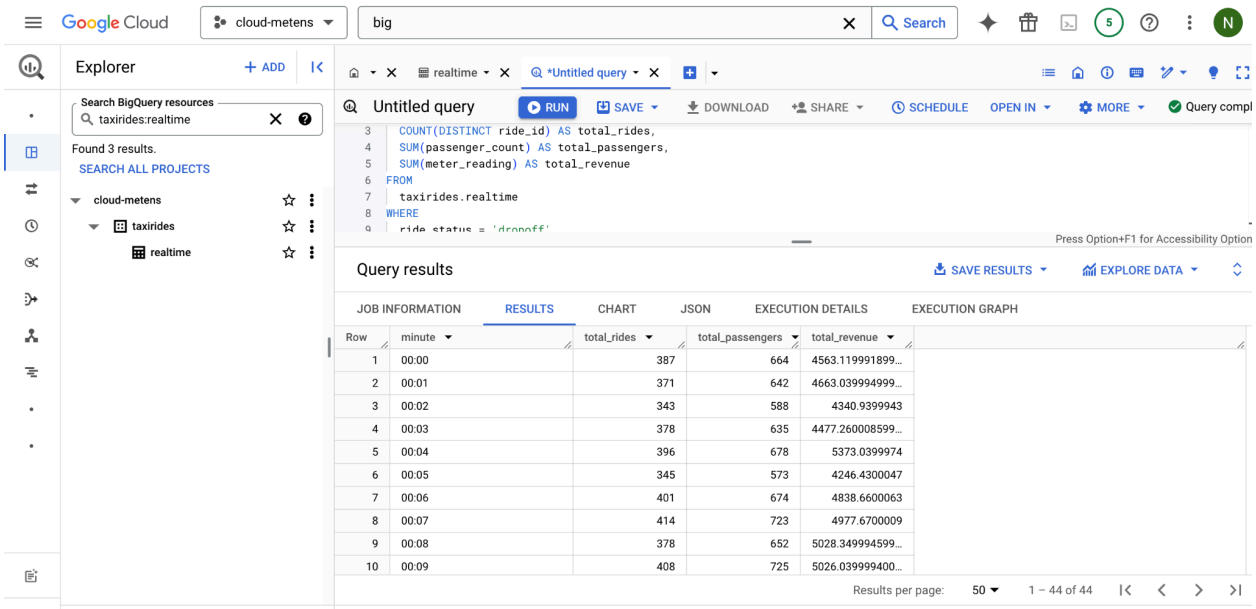
15. Query data in BigQuery



Assuming meter_count is the amount paid for the first ride:




Google Cloud
cloud-metens
big
Search
Explorer
+ ADD
Search BigQuery resources
taxirides:realtime
Found 3 results.
SEARCH ALL PROJECTS
cloud-metens
taxirides
realtime
realtime
*Untitled query
Untitled query
RUN
SAVE
DOWNLOAD
SHARE
SCHEDULE
OPEN IN
-- SELECT * FROM `cloud-metens.taxirides.realtime`;
SELECT COUNT(*) AS total_rows FROM `cloud-metens.taxirides.realtime`;
Query results
SAVE RESULT
JOB INFORMATION
RESULTS
CHART
JSON
EXECUTION DETAILS
EXECUTION GRAPH
Row
total_rows
1
6143052


Google Cloud
cloud-metens
big
Search
Explorer
+ ADD
Search BigQuery resources
taxirides:realtime
Found 3 results.
SEARCH ALL PROJECTS
cloud-metens
taxirides
realtime
realtime
*Untitled query
Untitled query
RUN
SAVE
DOWNLOAD
SHARE
SCHEDULE
OPEN IN
MORE
COUNT(DISTINCT ride_id) AS total_rides,
SUM(passenger_count) AS total_passengers,
SUM(meter_reading) AS total_revenue
FROM
taxirides.realtime
WHERE
Press Option+F1 for Accessibility Option
Query results
SAVE RESULTS
EXPLORE DATA
JOB INFORMATION
RESULTS
CHART
JSON
EXECUTION DETAILS
EXECUTION GRAPH
Row
minute
total_rides
total_passengers
total_revenue
1 00:00 387 664 4563.119991899...
2 00:01 371 642 4663.039994999...
3 00:02 343 588 4340.9399943
4 00:03 378 635 4477.260008599...
5 00:04 396 678 5373.0399974
6 00:05 345 573 4246.4300047
7 00:06 401 674 4838.6600063
8 00:07 414 723 4977.6700009
9 00:08 378 652 5028.349994599...
10 00:09 408 725 5026.039999400...
Results per page:
50
1 – 44 of 44

16. Data visualization