# Week9

## Lab9

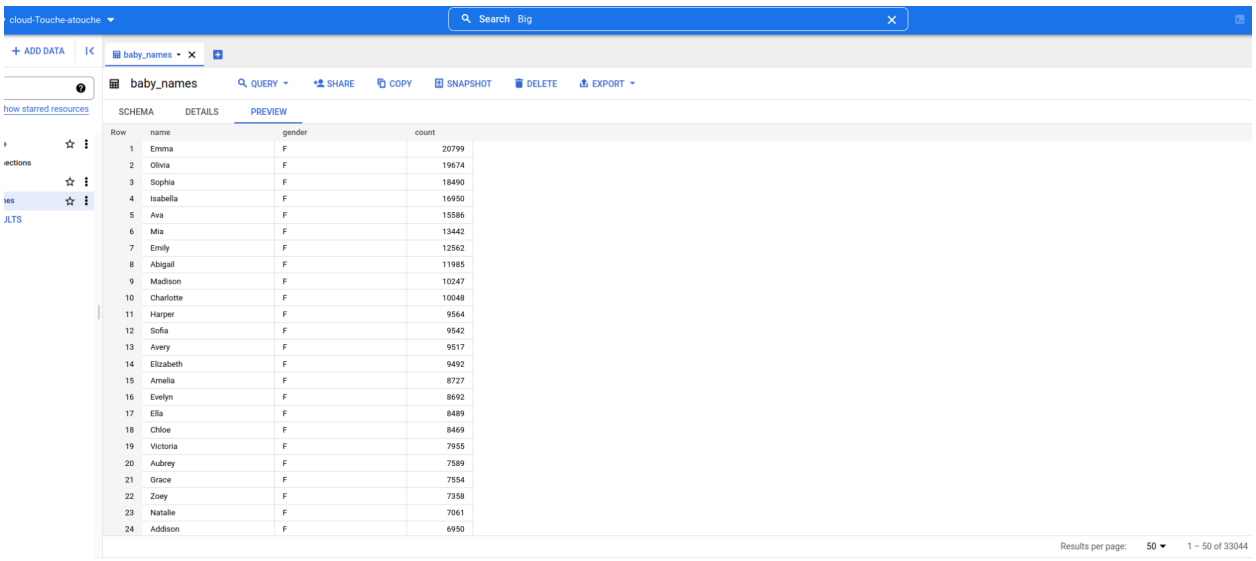
Armant Touche

**Class/Instructor:** CS430P/ Dr. Wu-Chang
**Date:** 12/05/22

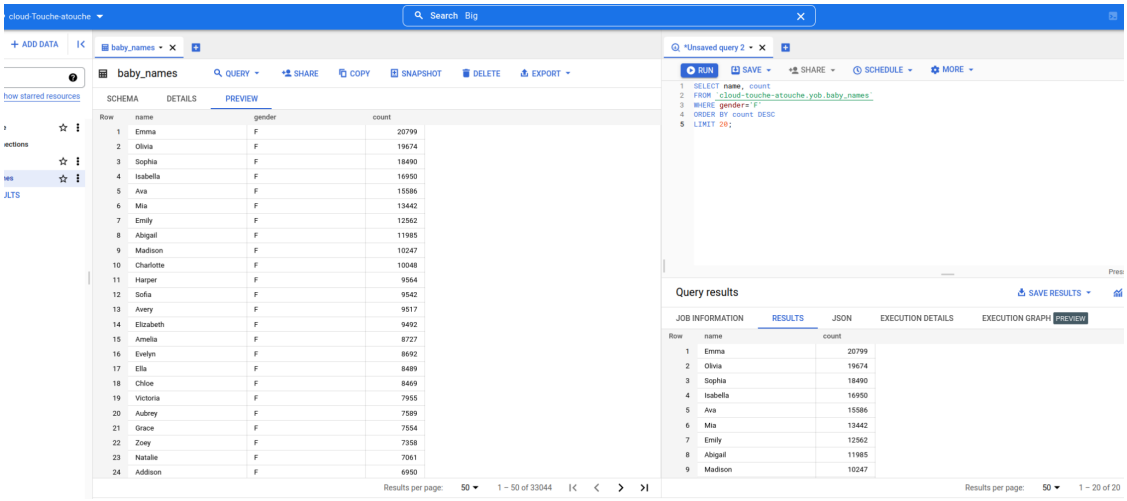
# Table of Contents

# BigQuery, JupyterLab (Link)

- [ ] BigQuery, Notebooks Lab #1
- [ ] Examine dataset
    - [ ] bucket: yobbigquery
- [ ] Create dataset
    - [ ] **Take a screenshot of the table's details that includes the number of rows in the table.**




- [ ] Query data
    - [ ] **Screenshot your results and include it in your lab notebook**



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

```
atouche@cloudshell:~/Documents/lab9/big_query (cloud-touche-atouche)$ bq query "SELECT name, count
FROM [cloud-touche-atouche:yob.baby_names]
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 |
+---------+-------+
```

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

```
Welcome to BigQuery! (Type help for more information.)
cloud-touche-atouche> select name, count from [cloud-touche-atouche:yob.baby_names] 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 |
+-----------+-------+
```

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

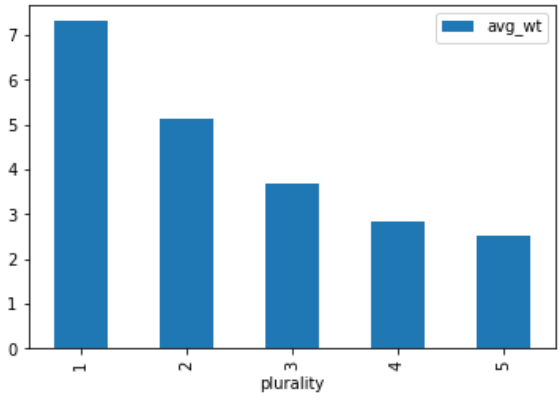
```
cloud-touche-atouche> select count(name) from [cloud-touche-atouche:yob.baby_names] where name='Armant'
+-----+
| f0_ |
+-----+
|   0 |
+-----+
```

- [ ] BigQuery, Notebooks Lab #2
- [ ] BigQuery query
  - [ ] **How many twins were born during this time?**
    - 375362
- [ ] Jupyter notebook query
- [ ] Exploring the dataset

- [ ] Run queries
  - [ ] **Show the plots generated for the two most important features for your lab notebook**
    - **1)** Plurality seems to be correlated w/ avg_wt. Higher the plurality, lower the avg_wt.

```
# step (9)
df = get_distinct_values('plurality')
df.plot(x='plurality', y='avg_wt', kind='bar')
```

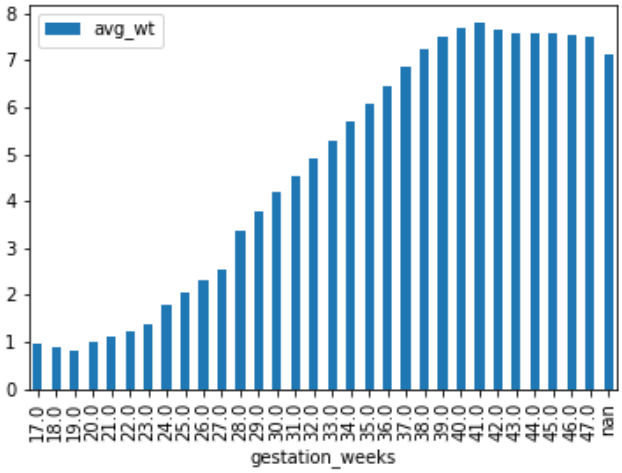
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f51354f3a90>
```




    - **2)** The further into gestation period, the higher the avg_wt.

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

```
[21]: <matplotlib.axes._subplots.AxesSubplot at 0x7f5134a07c50>
```

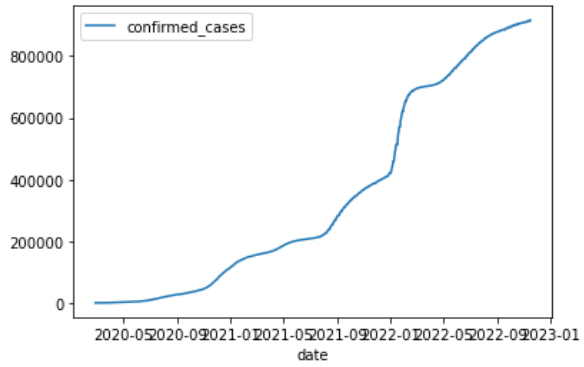


- [ ] BigQuery, Notebooks Lab #3
  - [ ] **What dates are used as a baseline for the mobility data?**
    - 2020-02-15 to 2022-10-15
  - [ ] **What day saw the largest spike in trips to grocery and pharmacy stores?**
    - 2020-03-13
  - [ ] **On the day the stay-at-home order took effect (3/23/2020), what was the total impact on workplace trips?**
    - -49% change from baseline meaning there were close to 50% less work trips being had on this date.
  - [ ] **Which three airports were impacted the most in April 2020 (the month when lockdowns became widespread)? Most to least.**
    - 1) Detroit Metropolitan Wayne County, 45.415% of normal traffic
    - 2) McCarran International, 45.599% of normal traffic
    - 3) San Francisco International, 47.266% of normal traffic
  - [ ] **Run the query again using the month of August 2020. Which three airports were impacted the most? Most to least**
    - 1) McCarran International, 44.200% of normal traffic
    - 2) Detroit Metropolitan Wayne County, 45.099% of normal traffic
    - 3) San Francisco International, 53.025% of normal traffic
- [ ] BigQuery, Notebooks Lab #4
  - [ ] **What table and columns identify the place name, the starting date, and the number of excess deaths from COVID-19?**
    - Table = excess_deaths
    - Place name= excess_deaths.placename (col)
    - Starting date = excess_deaths.start_date (col)
    - Number of excess deaths = excess_deaths.excess_deaths
  - [ ] **What table and columns identify the date, county, and deaths from COVID-19?**
    - Table = us_counties
    - Date = us_counties.date (col)
    - Count = us_counties.county (col)
    - Deaths = us_counties.deaths (col)
  - [ ] **What table and columns identify the date, state, and confirmed cases of COVID-19?**
    - Table = us_states
    - Date = us_counties.date (col)
    - State = us_counties.state_name (col)
    - Deaths = us_counties.confirmed_cases (col)
  - [ ] **What table and columns identify a county code and the percentage of its residents that report they always wear masks?**
    - Table = mask_use_by_county
    - Always wears = maske_use_by_county.always (col)

- [ ] Run example queries
  - [ ] **Show a screenshot of the plot and the code used to generate it for your lab notebook**

```
[26]: df = bigquery.Client().query(query_string).to_dataframe()
      df.plot(x='date', y='confirmed_cases', kind='line')
```

```
[26]: <matplotlib.axes._subplots.AxesSubplot at 0x7f51347d8510>
```




```
[ ]: atouche
```

  - [ ] **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.**

```
: # step 13
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 LIMIT 10"""
```

```
: df = bigquery.Client().query(query_string).to_dataframe()
df.head(10)
# atouche
```

:

| | 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 |
| **5** | 32009 | 0.872 | Esmeralda |
| **6** | 06023 | 0.865 | Humboldt |
| **7** | 36099 | 0.864 | Seneca |
| **8** | 36121 | 0.861 | Wyoming |
| **9** | 25015 | 0.857 | Hampshire |

...

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

```
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"""
```

```
df = bigquery.Client().query(query_string).to_dataframe()
df.head(5)
# atouche
```

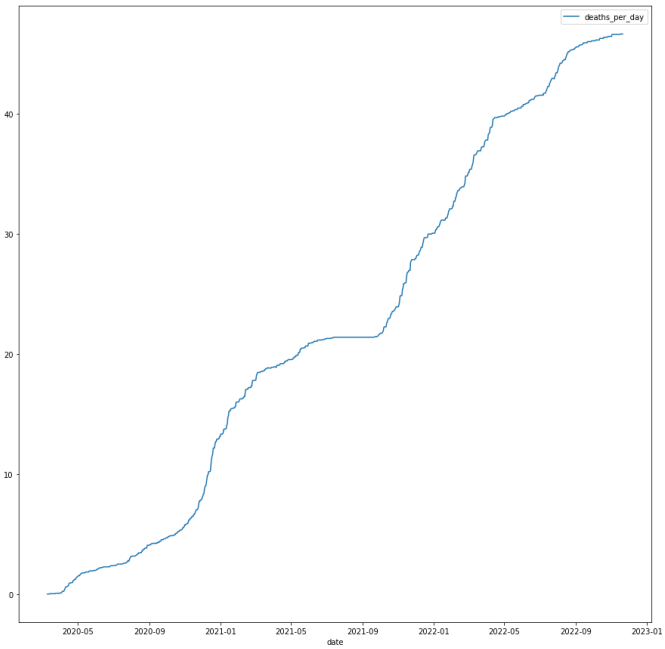
| | county_fips_code | always | county |
|---|---|---|---|
| **0** | 06027 | 0.889 | Inyo |
| **1** | 36123 | 0.884 | Yates |
| **2** | 48229 | 0.880 | Hudspeth |
| **3** | 06051 | 0.880 | Mono |
| **4** | 48141 | 0.877 | El Paso |

- [ ] Write queries
  - [ ] Deaths in Multnomah county
    - [ ] **Plot the results and take a screenshot for your lab notebook.**

```
[42]: # Deaths in Multnomah by atouche
query_string = """SELECT
(deaths/30) AS deaths_per_day, date
FROM `bigquery-public-data.covid19_nyt.us_counties`
WHERE county = "Multnomah" and state_name="Oregon"
ORDER BY date ASC"""
```

```
[43]: df = bigquery.Client().query(query_string).to_dataframe()
df.plot(x='date', y='deaths_per_day', kind='line', figsize=(15,15))
```

```
[43]: <matplotlib.axes._subplots.AxesSubplot at 0x7fe720875d90>
```

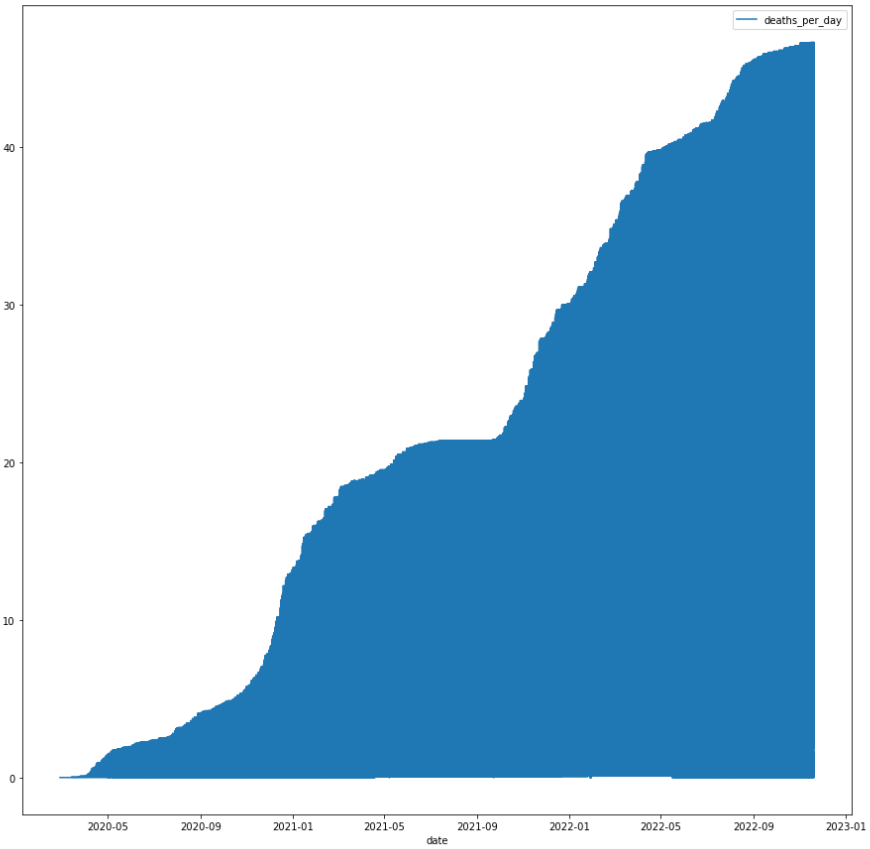


- [ ] Deaths in Oregon
    - [ ] **Plot the results and take a screenshot for your lab notebook.**

[44]:
```
# Deaths in Oregon by atouche
query_string = """SELECT
(deaths/30) AS deaths_per_day, date
FROM `bigquery-public-data.covid19_nyt.us_counties`
WHERE state_name="Oregon"
ORDER BY date ASC"""
```

[45]:
```
df = bigquery.Client().query(query_string).to_dataframe()
df.plot(x='date', y='deaths_per_day', kind='line', figsize=(15,15))
```

[45]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe720866d90>
```



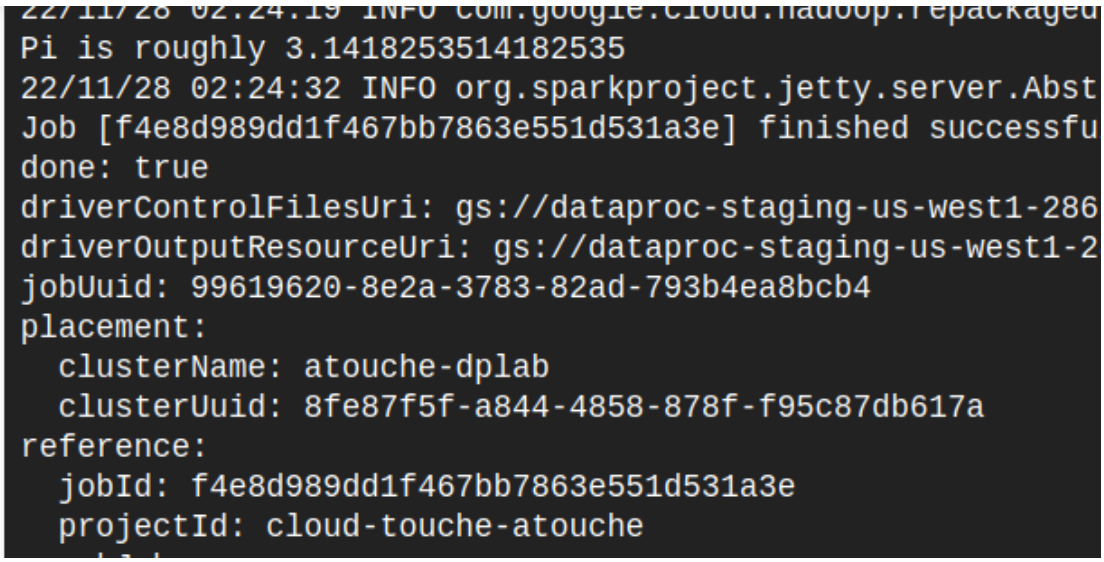

- [ ] Clean up

# Dataproc, Dataflow (Link)

- [ ] Dataproc Lab #1
- [ ] Calculating pi
- [ ] Code
- [ ] Dataproc setup
- [ ] Create Compute Engine cluster
- [ ] Run computation
    - [ ] **How long did the job take to execute?**
        - Job executed between 2:20:06 am  to 2:20:36 am so job took about 30 seconds.
    - [ ] **Examine output.txt and show the estimate of π calculated.**


- [ ] Scale cluster
- [ ] Run computation again
    - [ ] **How long did the job take to execute? How much faster did it take?**
        - Job executed between 2:23:59 am  to 2:24:39 am so the job took about 39 seconds.

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




- [ ] Clean up
- [ ] Dataflow Lab #1
- [ ] Setup
- [ ] Beam code
  - [ ] **Where is the input taken from by default?**
    - default is '../javahelp/src/main/java/com/google/cloud/training/dataanalyst/javahelp/'
  - [ ] **Where does the output go by default?**
    - default is '/tmp/output'
  - [ ] **Examine both the getPackages() function and the splitPackageName() function. What operation does the 'PackageUse()' transform implement?**
    - Assuming PackageUse() is referring to packageUse() is is_popular.py, the operation being implemented is to construct a Java packages used in.
  - [ ] **Look up Beam's CombinePerKey. What operation does the TotalUse operation implement?**
    - TotaleUse adds the total number of uses a package has from input (e.g. pkg=java.util.Scanner and therefore, TotalUse=10).
  - [ ] **Which operations correspond to a "Map"?**
    - GetImports and PackageUse correspond to "Map"
  - [ ] **Which operation corresponds to a "Shuffle-Reduce"?**
    - TotalUse corresponds "Shuffle-Reduce"
  - [ ] **Which operation corresponds to a "Reduce"?**
    - Top_5

- [ ] Run pipeline locally
    - [ ] **Take a screenshot of its contents**

```
atouche@cloudshell:~/.../dataflow/python (cloud-touche-atouche)$ 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)]
atouche@cloudshell:~/.../dataflow/python (cloud-touche-atouche)$
```

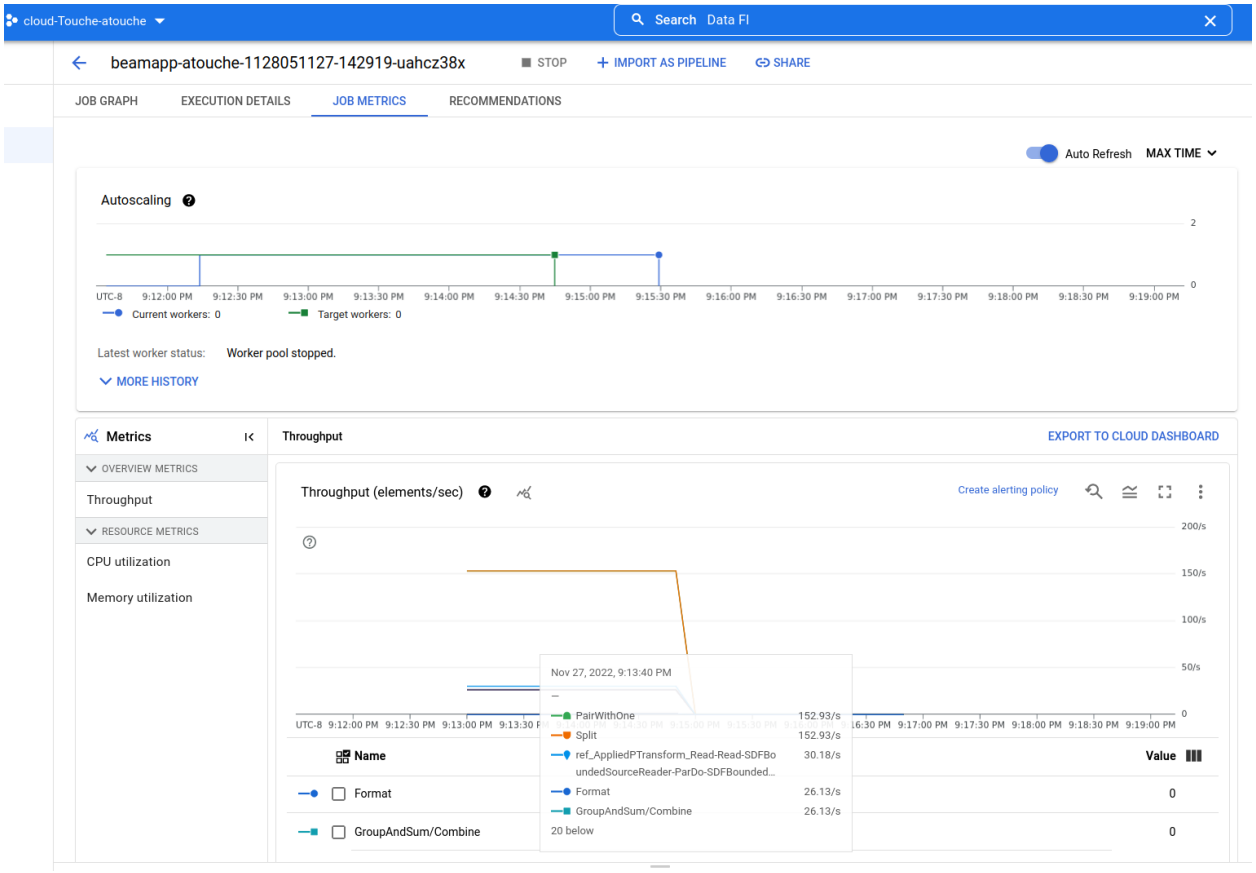
    - [ ] **Explain what the data in this output file corresponds to based on your understanding of the program.**
        - Using files from ~/javahelp,  Top 5 used packages from input:
            - org was used 45 times
            - org.apache used 44 times
            - org.apache.beam used 44 times
            - org.apache.beam.sdk used 43 times
            - org.apache.beam.sdk.transforms used 16 times
- [ ] Dataflow Lab #2
    - [ ] **What are the names of the stages in the pipeline?**
        - Read
        - Split
        - PairWithOne
        - GroupAndSum
        - Format
        - Write
    - [ ] **Describe what each stage does.**
        - Read: gets input where default is gs://dataflow-samples/shakespeare/kinglear.txt
        - Split: runs beam.pardo() which splits lines from input
        - PairWithOne: word count value is paired word
        - GroupAndSum: shuffle reduces each word to (word, totalSum)
        - Write: writes to file default='output'.
- [ ] Run code locally
    - [ ] **Use wc with an appropriate flag to determine the number of unique words in King Lear.**

```
e/04_features/dataflow/python (cloud-touche-atouche)$ sort -k 2r outputs-00000-of-00001 | uniq -c
 wc -l
4784
atouche@cloudshell:~/Documents/lab9/dataproc/training-data-analyst/courses/machine_learning/deepdi
e/04_features/dataflow/python (cloud-touche-atouche)$
```

- [ ] **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**

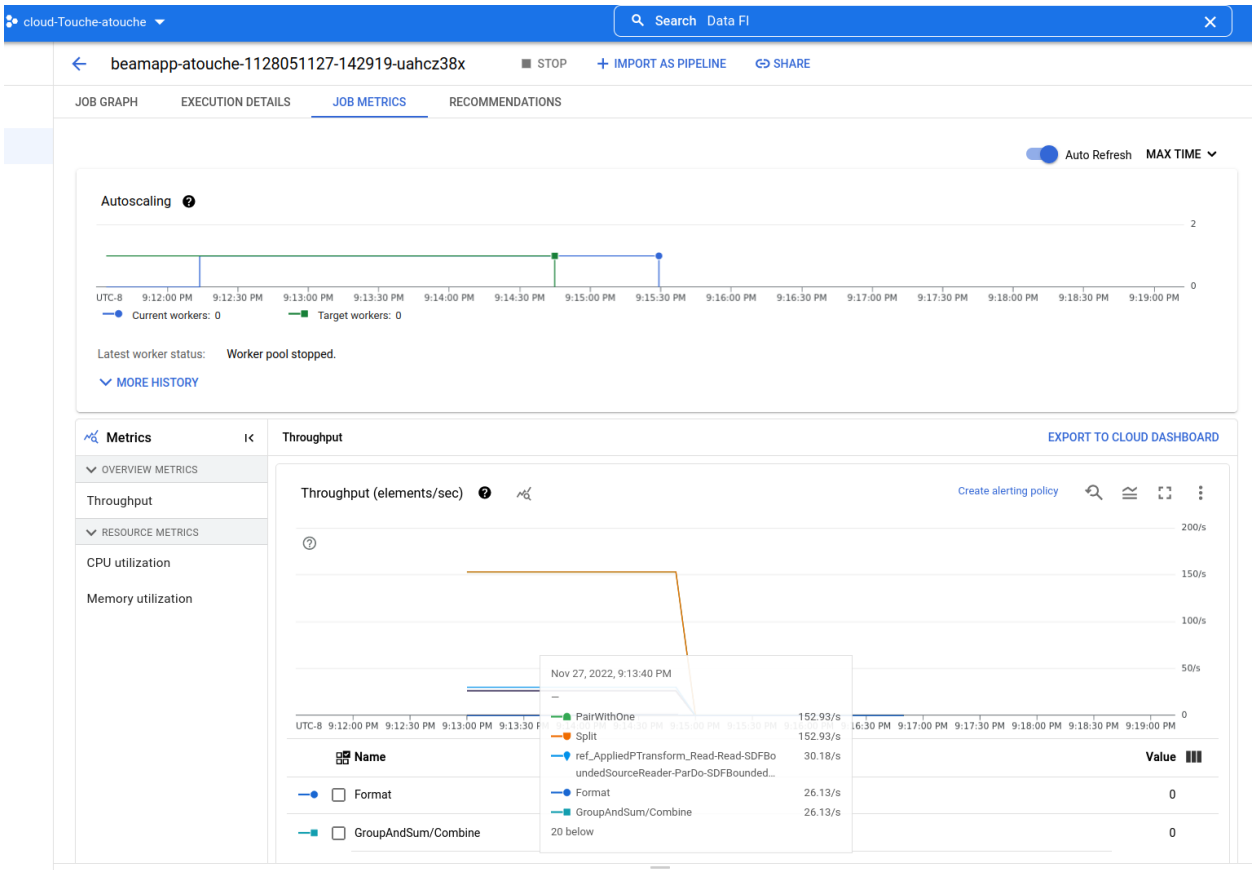
```
e/04_features/dataflow/python (cloud-touche-atouche)$ sort -k 2r outputs-00000-of-00001 | head -3
EDMUND: 99
That: 98
lord: 96
atouche@cloudshell:~/Documents/lab9/dataproc/training-data-analyst/courses/machine_learning/deepdi
```

- [ ] **Use the previous method to show the top 3 words in King Lear, case-insensitive, and the number of times they appear.**
    - Idk how
- [ ] Setup for Cloud Dataflow
- [ ] Service account setup
- [ ] Run code using Dataflow
    - [ ] **The part of the job graph that has taken the longest time to complete.**

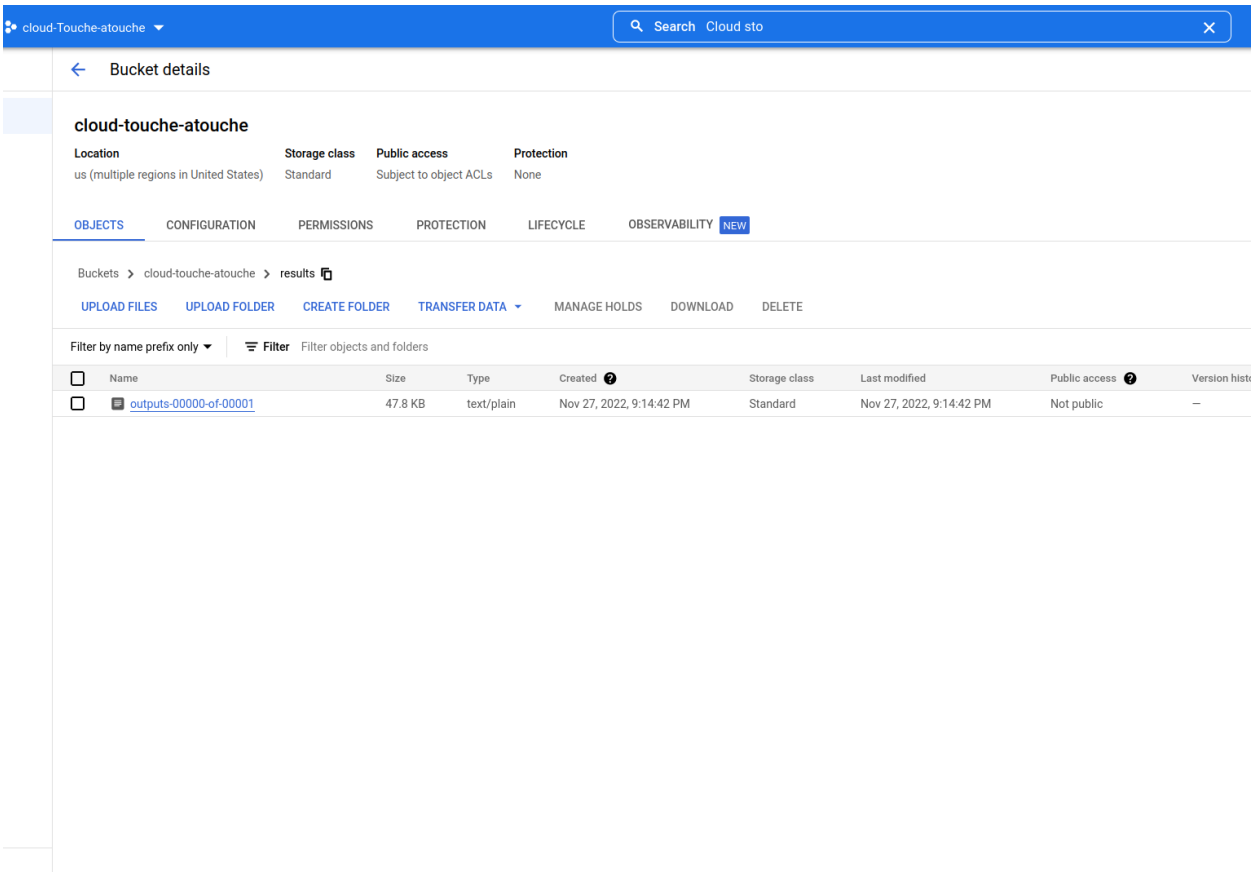



- PairWithOne seemed to take the longest to complete.

☐ **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?**




- The above screenshot shows the 1 file written.

☐ Clean up