Max Huecksteadt
CS 530
Week 9 Lab

1.

baby_names QUERY SHARE

SCHEMA DETAILS PREVIEW

Row	name	gender	co
1	Emma	F	
2	Olivia	F	
3	Sophia	F	
4	Isabella	F	
5	Ava	F	
6	Mia	F	
7	Emily	F	
8	Abigail	F	
9	Madison	F	
10	Charlotte	F	
11	Harper	F	
12	Sofia	F	

Results per page: 50 1 – 50 of 33044

2.

RUN SAVE SHARE SCHEDULE

```
SELECT name, count
FROM `cloud-huecksteadt-mhueck2.yob.baby_names`
WHERE gender='F'
ORDER BY count DESC
LIMIT 10
```

Query results SAVE RESULTS

JOB INFORMATION RESULTS JSON EXE

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

3.

```
mhueck2@cloudshell:~ (cloud-huecksteadt-mhueck2)$ bq query "SELECT name, count FROM [cloud-huecksteadt-mhueck2:yob.b
aby_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 |
+---------+-------+
```

4.

```
cloud-huecksteadt-mhueck2> SELECT name, count FROM [cloud-huecksteadt-mhueck2: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 |
+-----------+-------+
```

5.

```
cloud-huecksteadt-mhueck2> SELECT name, count FROM [cloud-huecksteadt-mhueck2:yob.baby_names] WHERE name='Max'
+------+-------+
| name | count |
+------+-------+
| Max  |    14 |
| Max  |  3468 |
+------+-------+
```

6. How many twins were born during this time?

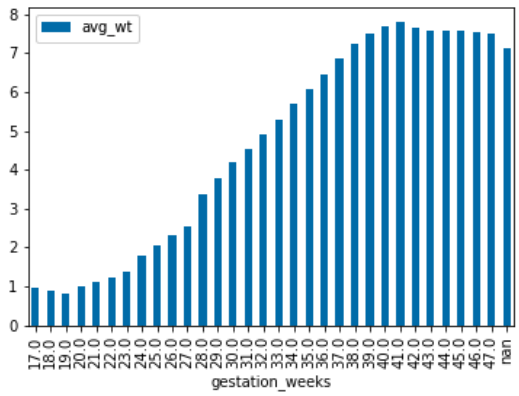
1	2	375362	5.17462027...

From the data returned, we can see that 375,362 babies were born during this time.

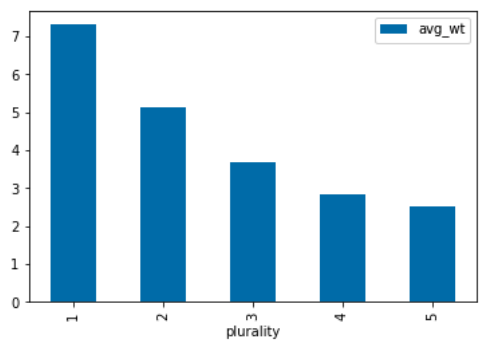
7. In examining the plots, which two features are the strongest predictors for a newborn baby's weight?

a. It appears that plurality and the gestation time are the strongest predictors:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7a7459bc10>
```




```
[7]: <matplotlib.axes._subplots.AxesSubplot at 0x7f7a746856d0>
```



8. What dates are used as a baseline for the mobility data?
    a. Jan 3–Feb 6, 2020
9. What day saw the largest spike in trips to grocery and pharmacy stores?
    a. 2020-03-13 with a spike of 17
10. On the day the stay-at-home order took effect (3/23/2020), what was the total impact on workplace trips?
    a. Workplace trips went down by 49
11. Which three airports were impacted the most in April 2020 (the month when lockdowns became widespread)?
    a. Detroit Metropolitan Wayne County, McCarran International, and San Francisco International
12. Run the query again using the month of August 2020. Which three airports were impacted the most?
    a. McCarran, Detroit, and SF (same as above just ordered differently)
13. What table and columns identify the place name, the starting date, and the number of excess deaths from COVID-19?
    a. excess_deaths: placename, start_date, excess_deaths
14. What table and columns identify the date, county, and deaths from COVID-19?
    a. us_counties: date, county, deaths
15. What table and columns identify the date, state, and confirmed cases of COVID-19?
    a. us_states: date, state_name, confirmed_cases

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

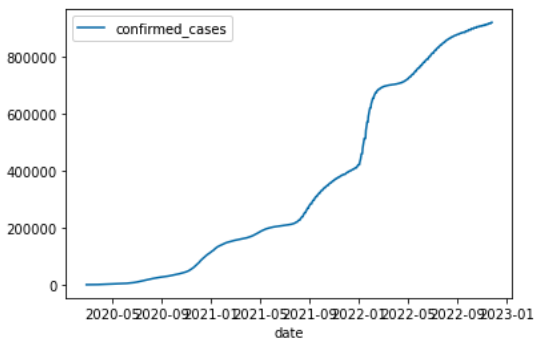
```
query_string = """
SELECT date, confirmed_cases
FROM `bigquery-public-data.covid19_nyt.us_states`
WHERE state_name = 'Oregon'
ORDER BY date ASC"""
```

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

	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

```
df.plot(x='date', y='confirmed_cases', kind='line')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f27f17e1710>
```



17.

```
[7]: 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"""
```

```
[9]: from google.cloud import bigquery
     df = bigquery.Client().query(query_string).to_dataframe()
     df.head(10)
```

[9]:

	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

18.

[12]:
```
query_string = """
SELECT DISTINCT mu.county_fips_code, mu.always, ct.county, st.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
LEFT JOIN `bigquery-public-data.covid19_nyt.us_states` as st
ON ct.state_name = st.state_name
ORDER BY mu.always DESC"""
```

[13]:
```
from google.cloud import bigquery
df = bigquery.Client().query(query_string).to_dataframe()
df.head()
```

[13]:

	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

19.

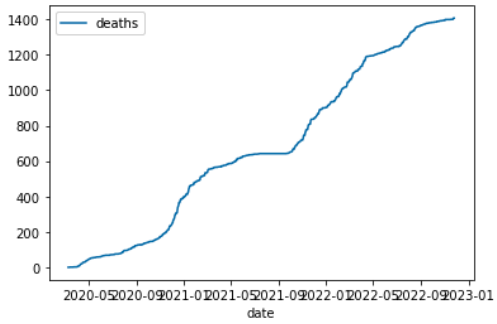
```
query_string = """
SELECT ct.date, ct.deaths, ct.county
FROM `bigquery-public-data.covid19_nyt.us_counties` as ct
WHERE ct.county = 'Multnomah'
ORDER BY ct.date ASC"""
```

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

	date	deaths	county
0	2020-03-10	0	Multnomah
1	2020-03-11	0	Multnomah
2	2020-03-12	0	Multnomah
3	2020-03-13	0	Multnomah
4	2020-03-14	1	Multnomah

```
df.plot(x='date', y='deaths', kind='line')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f27f0b1c990>
```



20.

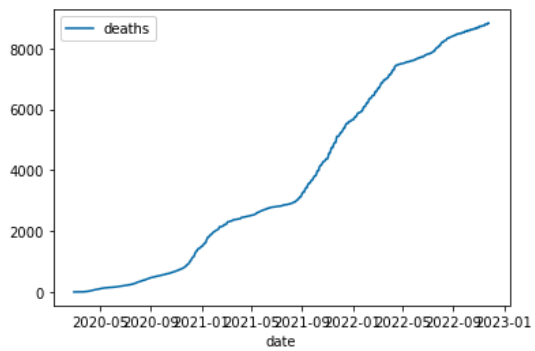
```
query_string = """
SELECT st.date, st.deaths, st.state_name
FROM `bigquery-public-data.covid19_nyt.us_states` as st
WHERE st.state_name = 'Oregon'
ORDER BY st.date ASC"""
```

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

	date	deaths	state_name
0	2020-02-28	0	Oregon
1	2020-02-29	0	Oregon
2	2020-03-01	0	Oregon
3	2020-03-02	0	Oregon
4	2020-03-03	0	Oregon

```
df.plot(x='date', y='deaths', kind='line')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f27f0c56690>
```



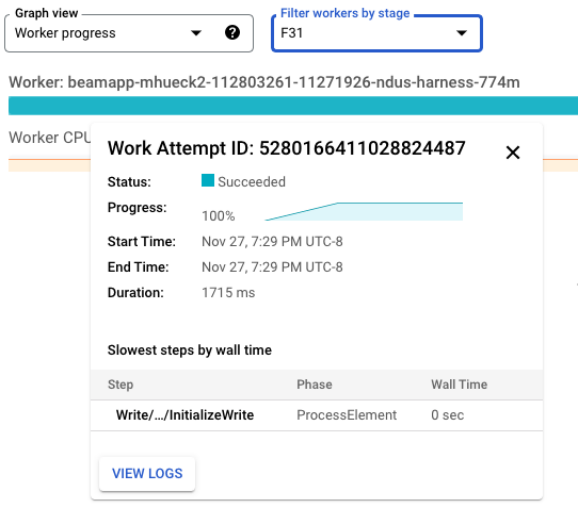
21.

9.3g

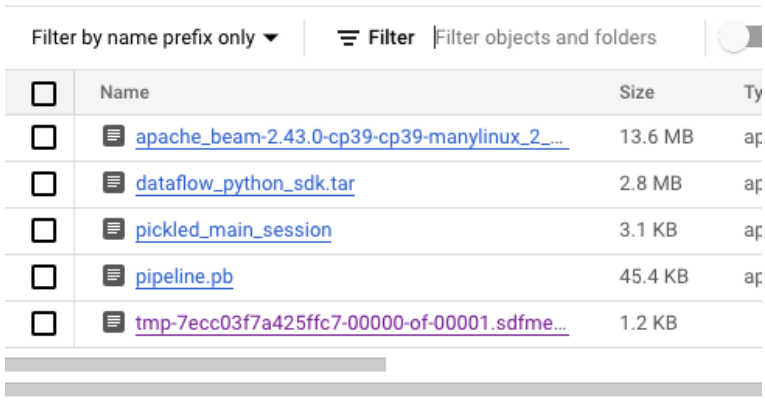
1. How long did the job take to execute?
    a. 56 seconds
2. 'Pi is roughly 3.1414826714148267'
3. The job with 2 extra workers took about 15 seconds
4. 'Pi is roughly 3.1415926535897936'
5. Default input:
    a. `../javahelp/src/main/java/com/google/cloud/training/dataanalyst/javahelp/`
6. Default output:
    a. `'/tmp/output`
7. Examine both the getPackages() function and the splitPackageName() function. What operation does the 'PackageUse()' transform implement?
    a. PackageUse() implements the getPackages operation (a mapping operation)
8. Look up Beam's CombinePerKey. What operation does the TotalUse operation implement?
    a. TotalUse implements the reduce operation

9. Which operations correspond to a "Map"?
    a. splitPackageName, getPackages, packageUse
10. Which operation corresponds to a "Shuffle-Reduce"?
    a. GetImports and TotalUse
11. Which operation corresponds to a "Reduce"?
    a. Top_5
12.
```
(env) mhueck2@cloudshell:/tmp (cloud-hueckstead-mhueck2)$ cat output*
[('org', 45), ('org.apache', 44), ('org.apache.beam', 44), ('org.apache.beam.sdk', 43), ('org.apache.beam.sdk.transf
orms', 16)]
```
13. The output is a count of the packages containing the leftmost entry branching to the rightmost (there are 45 pkgs with 'org' in them, 44 with org.apache, etc).
14. The pipeline stages are:
    a. Lines, counts (of lines), format_result->output, and write ->text
    b. First, the text is read into a pipe collection, then the words counts are formatted into strings, then the output is written as a transform to file.
15. Use wc with an appropriate flag to determine the number of unique words in King Lear.
    a.
```
d-hueckstead-mhueck2)$ wc -w outputs*
9568 outputs-00000-of-00001
```
16. 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.
```
(env) mhueck2@cloudshell:~/training-data-analyst/courses/machin
d-hueckstead-mhueck2)$ sort -n -k 2,2 -r outputs* | head -n 3
the: 786
I: 622
and: 594
```
17. Use the previous method to show the top 3 words in King Lear, case-insensitive, and the number of times they appear.
    a.
```
d-hueckstead-mhueck2)$ sort -n -k 2,2 -r outputs* | head -n 3
the: 908
and: 738
i: 622
```
18. The part of the job graph that has taken the longest time to complete.

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

a. 

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



a. 5 it seems: