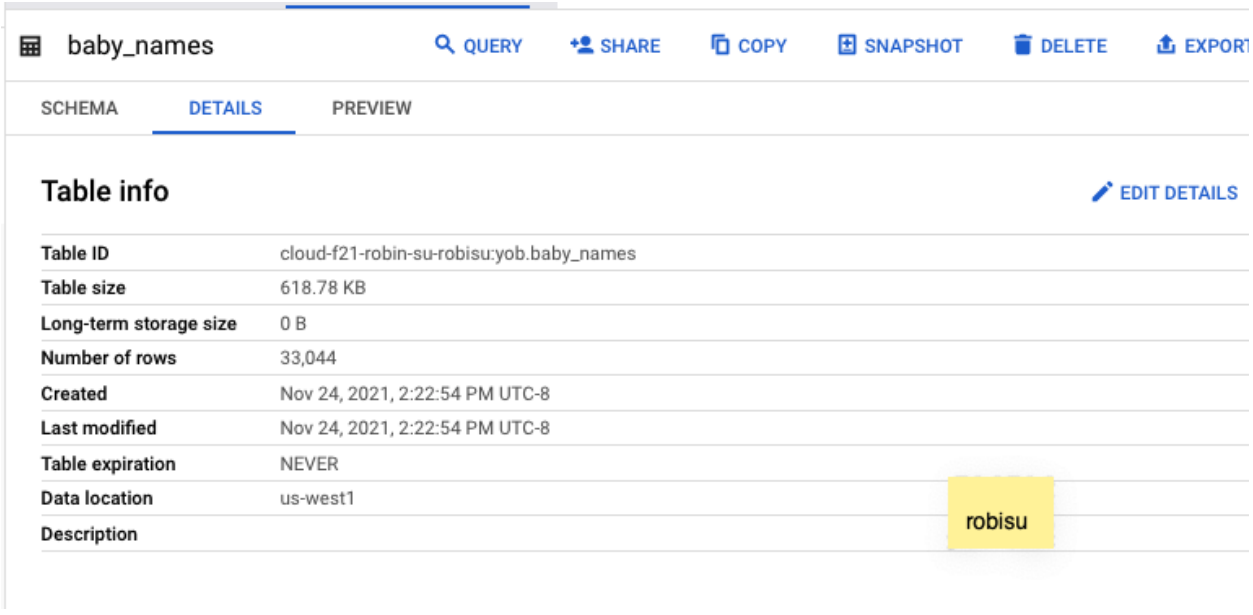
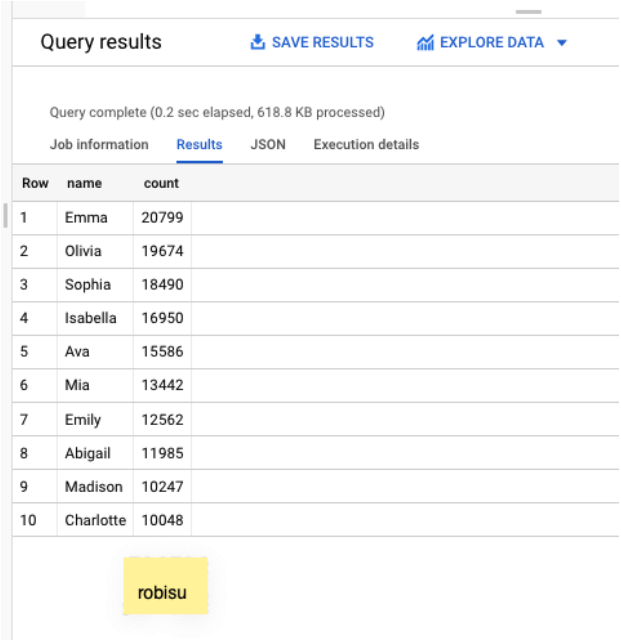
BigQuery



Most popular female names in 2014



Query results SAVE RESULTS EXPLORE DATA

Query complete (0.2 sec elapsed, 618.8 KB processed)

Job information | Results | JSON | Execution details

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

robisu

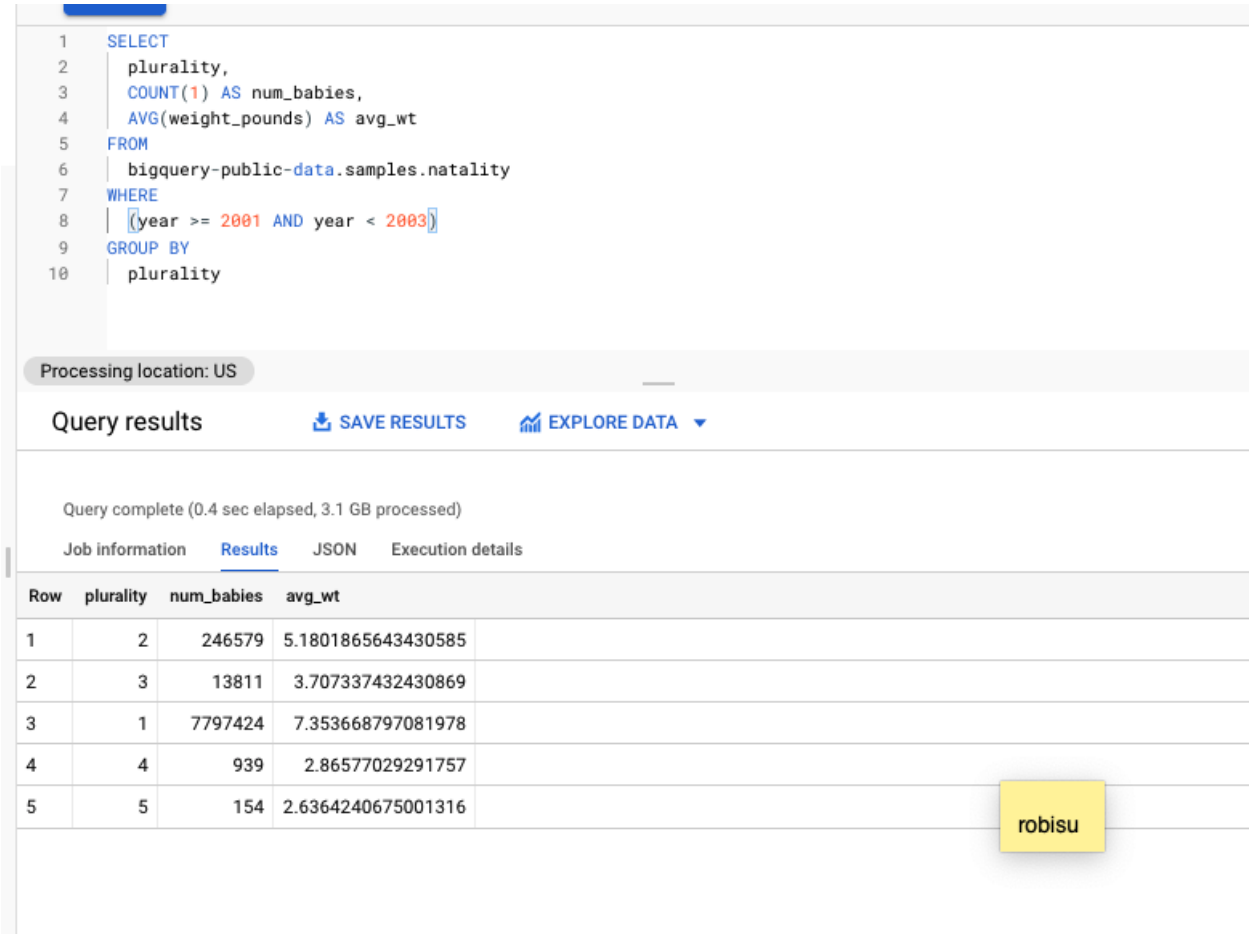
Least popular male names from 2014

```
robisu@cloudshell:~ (cloud-f21-robin-su-robisu)$ bq query "SELECT name, count FROM [cloud-f21-robin-s
u-robisu:yob.baby_names] WHERE gender='M' ORDER BY count ASC LIMIT 10"
Waiting on bqjob_r1ba4119d17aceeef_0000017d54135351_1 ... (0s) Current status: DONE
+---------+-------+
|  name   | count |
+---------+-------+
| Aari    |     5 |
| Aaliyah |     5 |
| Aadian  |     5 |
| Aaroh   |     5 |
| Aarit   |     5 |
| Aadiv   |     5 |
| Aadhi   |     5 |
| Aarohan |     5 |
| Aariyan |     5 |
| Aamer   |     5 |
+---------+-------+
```

Most popular male names from 2014

```
cloud-f21-robin-su-robisu> SELECT name, count FROM [cloud-f21-robin-su-robisu:yob.baby_names] WHERE g
ender='M' ORDER BY count DESC LIMIT 10
Waiting on bqjob_r75009048c58b7215_0000017d54154dca_1 ... (0s) Current status: DONE
+-----------+-------+
|   name    | count |
+-----------+-------+
| Noah      | 19144 |
| Liam      | 18342 |
| Mason     | 17092 |
| Jacob     | 16712 |
| William   | 16687 |
| Ethan     | 15619 |
| Michael   | 15323 |
| Alexander | 15293 |
| James     | 14301 |
| Daniel    | 13829 |
+-----------+-------+
cloud-f21-robin-su-robisu>
```

BigQuery, natality table / plurality query:

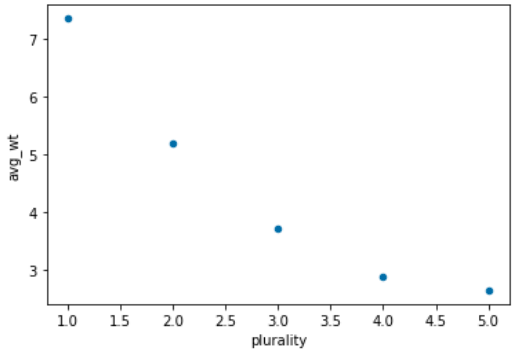


246579 twins were born between 2001 and 2003

plot of data from above (in the notebook):

```
[4]: df.plot(x='plurality', y='avg_wt', kind='scatter')
```

```
[4]: <matplotlib.axes._subplots.AxesSubplot at 0x7ff04bff8910>
```

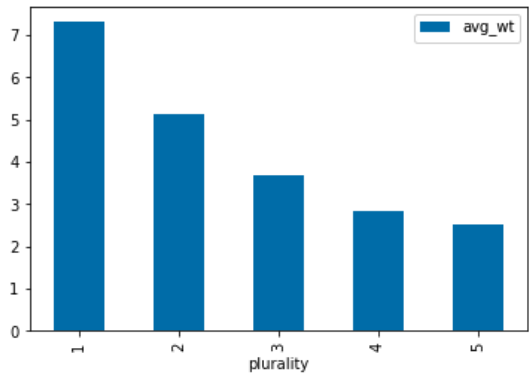


robisu

Two strongest predictors for birth weight: plurality, weeks of gestation:

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

```
[8]: <matplotlib.axes._subplots.AxesSubplot at 0x7ff0505ab710>
```

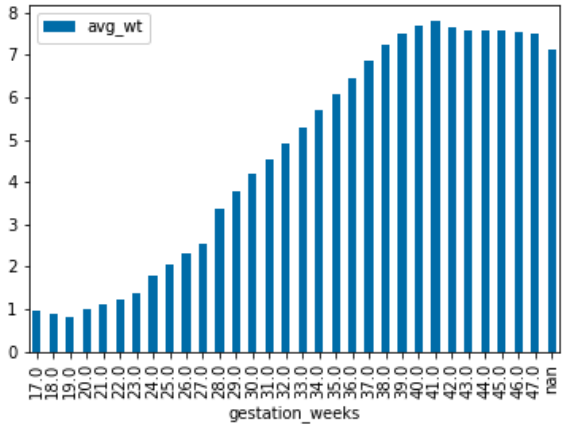


robisu

-

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

```
[10]: <matplotlib.axes._subplots.AxesSubplot at 0x7ff050664690>
```



robisu

Covid-19 Mobility Data

- *What dates are used as a baseline for the mobility data?*
  - **Jan 3–Feb 6, 2020**
- *What day saw the largest spike in trips to grocery and pharmacy stores?*
  - **3-13-2020**
- *On the day the stay-at-home order took effect (3/23/2020), what was the total impact on workplace trips?*
  - **There was <u>49%</u> dip in trips to the workplace.**

- *Which three airports were impacted the most in April 2020 (the month when lockdowns became widespread)?*
    - **Detroit Metropolitan Wayne County, McCarran International (Nevada), and San Francisco International**
- *Run the query again using the month of August 2020. Which three airports were impacted the most?*
    - **The same 3 as above were affected, except McCarran was the most, followed by Detroit, then followed by San Francisco.**

- *What table and columns identify the place name, the starting date, and the number of excess deaths from COVID-19?*
    - **Table: excess_deaths; columns: placename, start_date, excess_deaths**
- *What table and columns identify the date, county, and deaths from COVID-19?*
    - **Table: us_counties; columns: date, county, deaths**
- *What table and columns identify the date, state, and confirmed cases of COVID-19?*
    - **Table: us_states; columns: date, state_name, confirmed_cases**
- *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; columns: county_fips_code, always**

Top 10 States to exceed 1000 deaths:

```
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
"""

query = bigquery.Client().query(query_string + " LIMIT 10").to_dataframe()
print(query['state_name'])
```

```
0         New York
1       New Jersey
2         Michigan
3        Louisiana
4    Massachusetts
5         Illinois
6     Pennsylvania
7       California
8      Connecticut
9          Florida
Name: state_name, dtype: object
```

robisu

Name: state_name, dtype: object

[33]:
```
querystr = """
SELECT DISTINCT mu.county_fips_code, mu.always, ct.county, ct.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
"""

query = bigquery.Client().query(querystr + " LIMIT 5").to_dataframe()
print(f"Top 5 Counties and Their States In Mask Usage:")
count = 1
for county, state in zip(query['county'], query['state_name']):
    print(f"{count}. {county}, {state}")
    count += 1
```

robisu

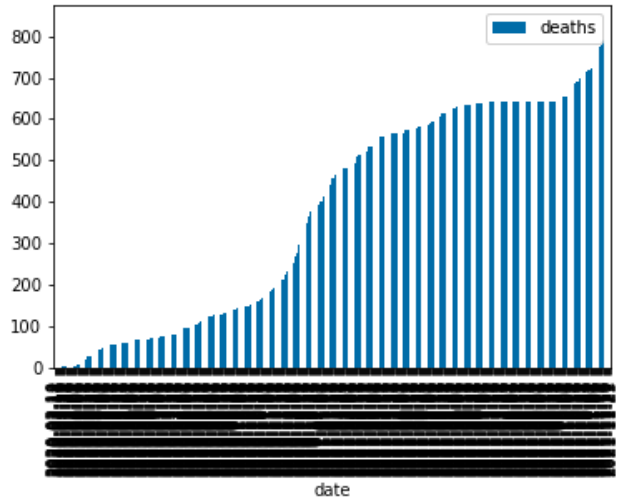
```
Top 5 Counties and Their States In Mask Usage:
1. Inyo, California
2. Yates, New York
3. Mono, California
4. Hudspeth, Texas
5. El Paso, Texas
```

Deaths in Multnomah County:

5. El Paso, Texas

[46]:
```
import pandas as pd
query_string = """
SELECT date, deaths
FROM `bigquery-public-data.covid19_nyt.us_counties`
WHERE county='Multnomah'
ORDER BY date ASC
"""
query = bigquery.Client().query(query_string).to_dataframe()
query.plot(x='date', y='deaths', kind='bar')
```

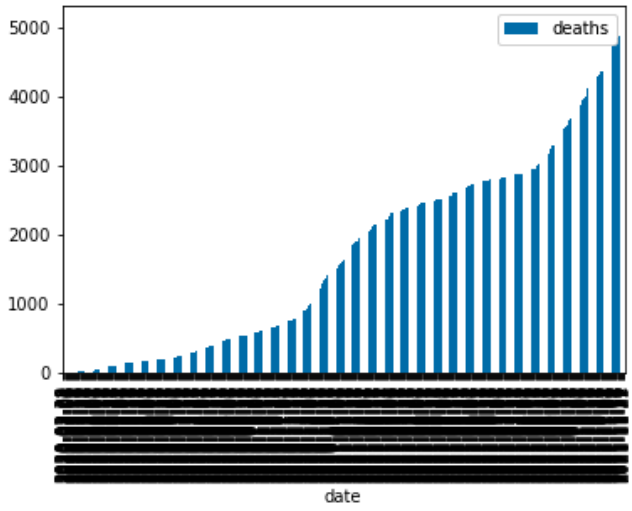
[46]: <matplotlib.axes._subplots.AxesSubplot at 0x7ff03efc27d0>



robisu

Deaths in Oregon:

[54]:
```
query_string = """
SELECT date, deaths
FROM `bigquery-public-data.covid19_nyt.us_states`
WHERE state_name='Oregon'
ORDER BY date ASC
"""
query = bigquery.Client().query(query_string).to_dataframe()
plt = query.plot(x='date', y='deaths', kind='bar')
```



robisu

Dataproc

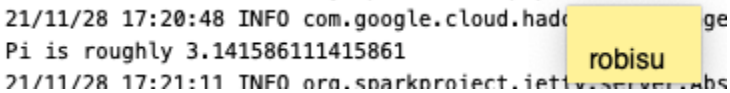
Calculating pi

- *How long did the job take to execute?* **It took about 2 minutes to execute**
  - **start time: Sun 28 Nov 2021 05:20:28 PM UTC**
  - **end time: Sun 28 Nov 2021 05:22:18 PM UTC**
- *Examine output.txt and show the estimate of π calculated.*

```
21/11/28 17:20:48 INFO com.google.cloud.hadoop
Pi is roughly 3.141586111415861
21/11/28 17:21:11 INFO org.sparkproject.jetty.server.AbstractConnector
```

robisu

- *How long did the job take to execute? How much faster did it take?*
    - **It only took 15 seconds to complete execution**
    - **start time: Sun 28 Nov 2021 05:29:01 PM UTC**
    - **end time: Sun 28 Nov 2021 05:29:16 PM UTC**
- *Examine output2.txt and show the estimate of π calculated.*



Dataflow #1

Apache Beam

is_popular.py

- *Where is the input taken from by default?*
- **`../javahelp/src/main/java/com/google/cloud/training/dataanalyst/javahelp/`**
- *Where does the output go by default?*
    - ***`/tmp/output`***
- *Examine both the getPackages() function and the splitPackageName() function. What operation does the 'PackageUse()' transform implement?*
    - **packageUse() is tallying the appearance of each package. For every package, it generates a tuple with the package name and a '1', which is summed (with beam.CombinePerKey) when the program is run**
- *Look up Beam's CombinePerKey. What operation does the TotalUse operation implement?*
    - **It takes all of the elements in the sequence, that were created in the packageUse function, and sums the elements of each distinct package name - giving us totals of times each package was imported and used.**

Map-Reduce Pattern

- *Which operations correspond to a "Map"?*
    - **'GetImports', 'PackageUse'**
- *Which operation corresponds to a "Shuffle-Reduce"?*
    - '**TotalUse**'
- *Which operation corresponds to a "Reduce"?*

- **‘Top5’**

contents of /tmp/output-00000-of-00001

```
(env) robisu@cloudshell:/tmp (cloud-f21-robin-su-robisu)$ 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)]
```

This file contains the counts of the top 5 packages that are imported and used in the input Java program. The packages were searched hierarchically, so the highest level module, ‘org’, has the most imports, followed by ‘org.apache’, and so on.

Dataflow #2

Word Count

- *What are the names of the stages in the pipeline?*
  - **‘Read’, ‘Split’, ‘PairWIthOne’, ‘GroupAndSum’, ‘Format’, ‘Write’**
- *Describe what each stage does.*
  - **‘Read’: reads in the input as specified from the cl arguments**
  - **‘Split’: takes strings and separates them into separate words that can be iterated over**
  - **‘PairWithOne’ (Map): pair each word with a count of 1 per occurrence**
  - **‘GroupAndSum’ (ShuffleReduce): for each distinct word, group each occurrence into a sum of occurrences**
  - **‘Format’: Takes each key/value pair of word and wordcount and formats into the string specified in format_result()**
  - **‘Write’: Write the result out to a file**

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

```
w/python (cloud-f21-robin-su-robisu)$ wc -w outputs-00000-of-00001
9568 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**

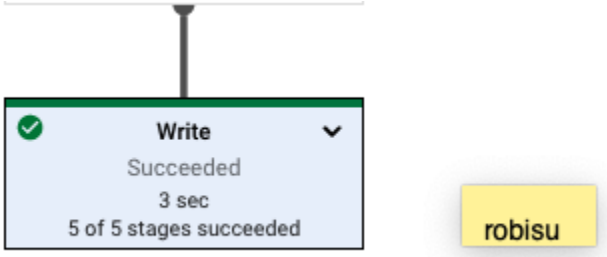
```
w/python (cloud-f21-robin-su-robisu)$ sort -n -k2rn outputs-00000-of-00001 | head -n 3
the: 786
I: 622
and: 594
```

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

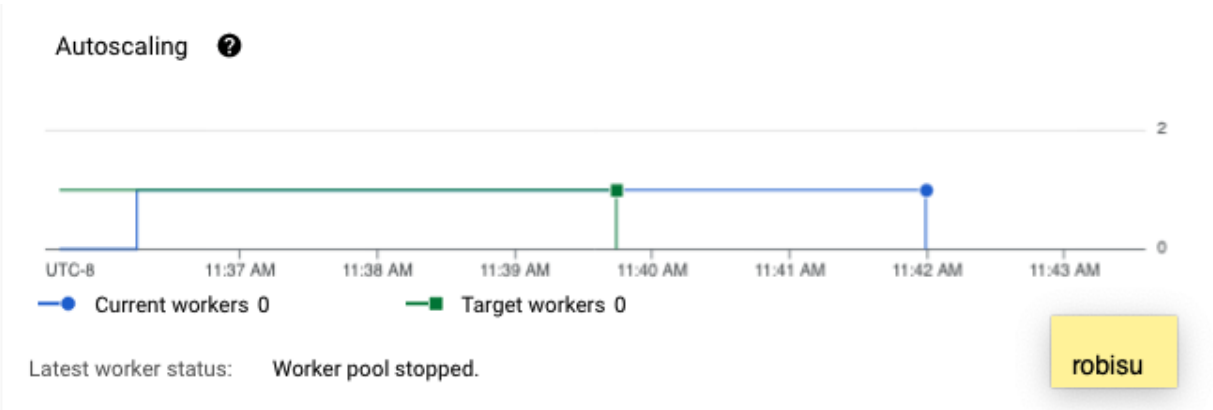
```
w/python (cloud-f21-robin-su-robisu)$ sort -n -k2rn outputs2-00000-of-00001 | head -n 3
the: 908
and: 738
i: 622
```

Dataflow Runner Execute

- *The part of the job graph that has taken the longest time to complete.*
  - **The Write stage too the longest (3 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?*
    - **There is a single file contained in results/, with a similar output to what we got when running the wordcount program locally.**