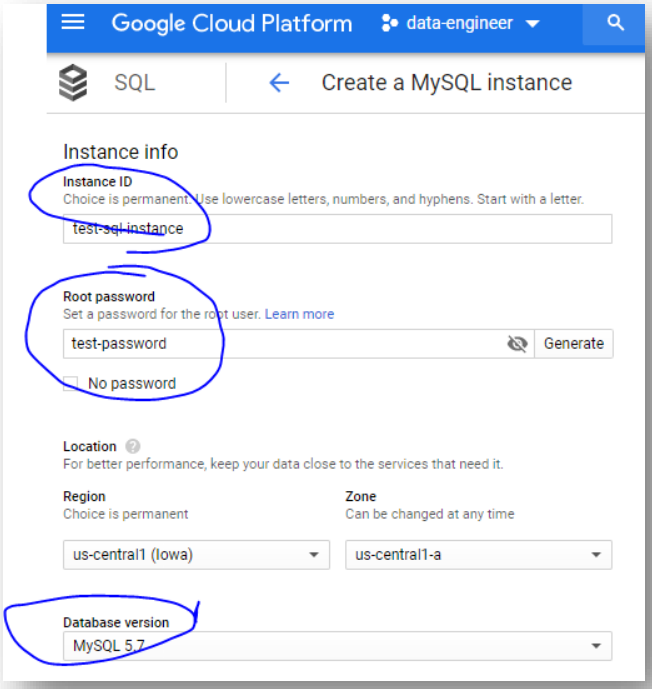
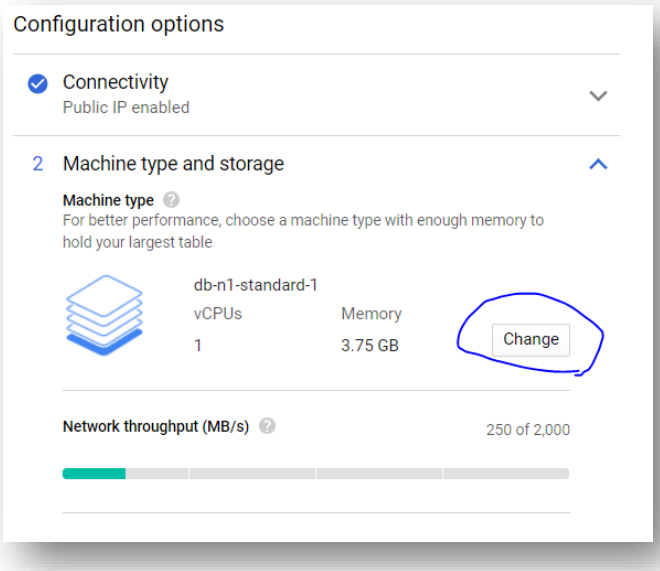
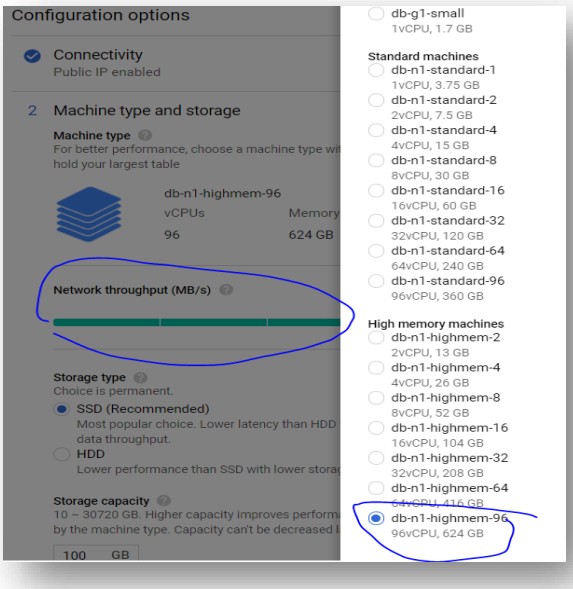
Lecture: Cloud SQL Hands On

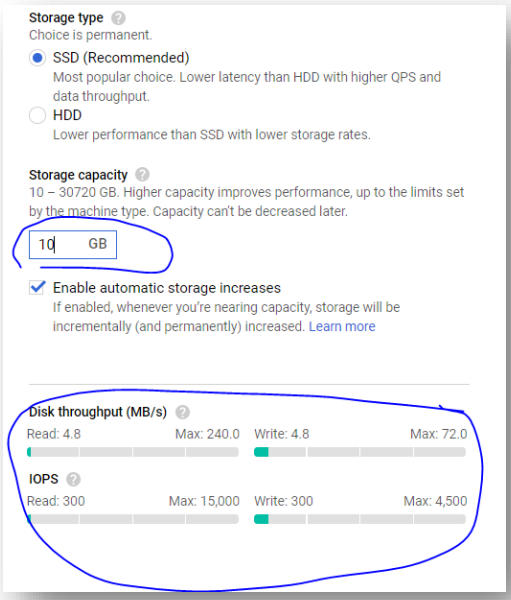


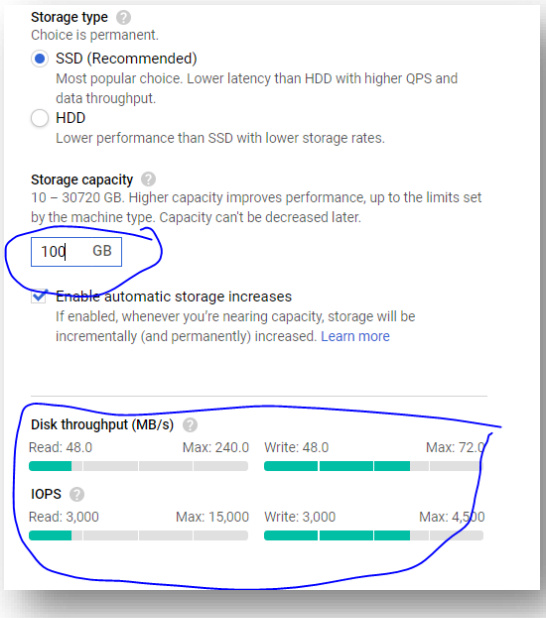
Check how **machine type** affect the **network throughput** rate:



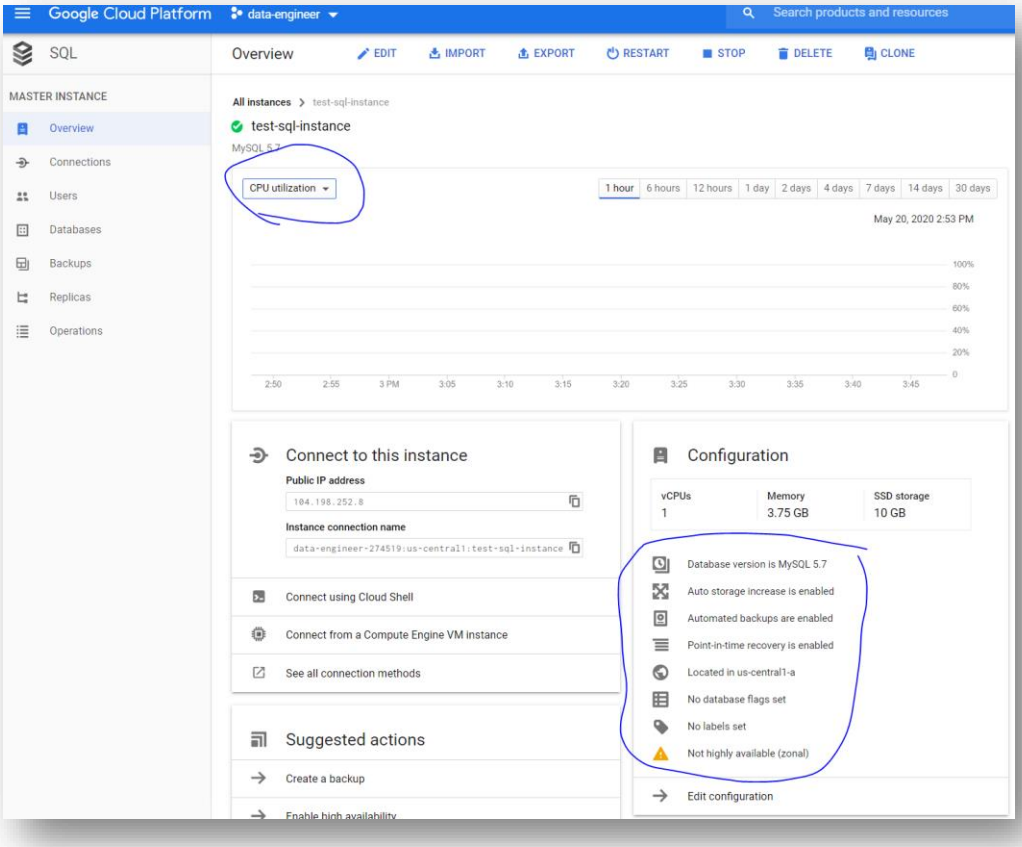


And the size of **disk** (storage) can affect performance:



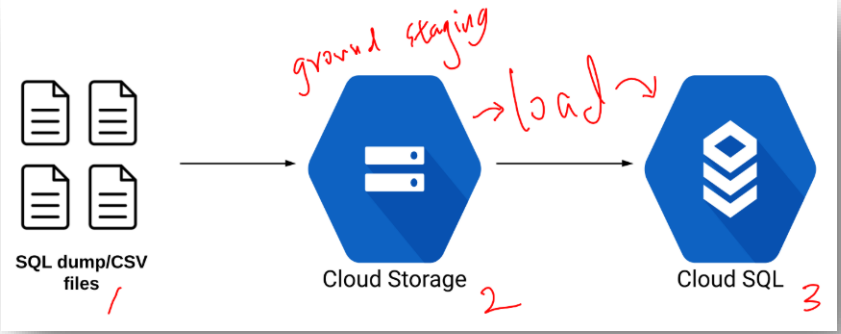


When create a SQL instance:
We can view ‘datasets’, ‘backups’, and ‘replicas’ in this instance.



Lecture: Importing Data

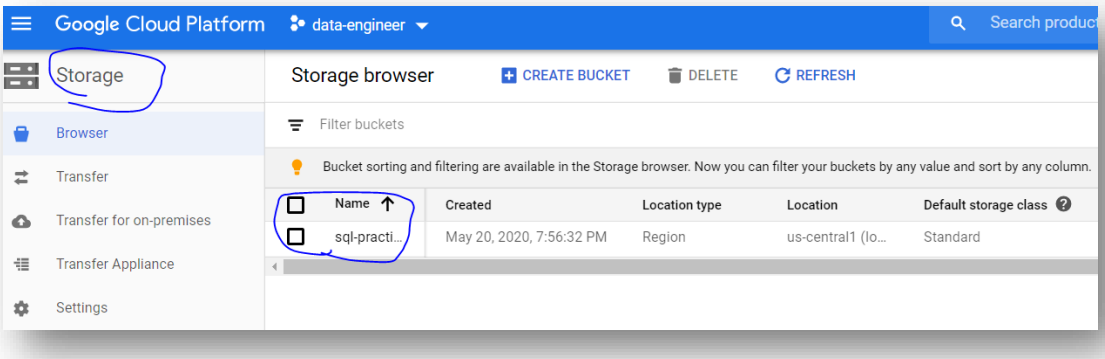
In this lesson, we will cover how to bulk import data into a Cloud SQL session, including SQL dump and CSV files. Detailed steps of what was performed in the lesson are below:
We will be using a sample data GitHub repo
here: https://github.com/linuxacademy/googledataengineer
The original source of this data from Google's GitHub can be viewed
here: https://github.com/GoogleCloudPlatform/training-data-analyst



1. Clone our sample data to Cloud Shell:
   - Open Cloud Shell.
   - Clone the data from *repo* to Cloud Shell.
   - `git clone https://github.com/linuxacademy/googledataengineer`

2. Create the **Cloud Storage bucket** (**as staging ground**) to copy data:
   - `gsutil mb -l (your_region) gs://(bucket_name)`

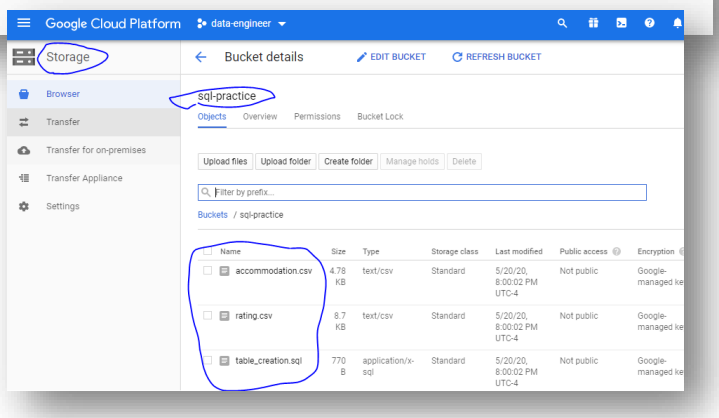
```
leonf_info@cloudshell:~/googledataengineer/CPB100/lab3a/cloudsql (data-engineer-274519)$ gsutil mb -l us-central1 gs://sql-practice
Creating gs://sql-practice/...
leonf_info@cloudshell:~/googledataengineer/CPB100/lab3a/cloudsql (data-engineer-274519)$
```



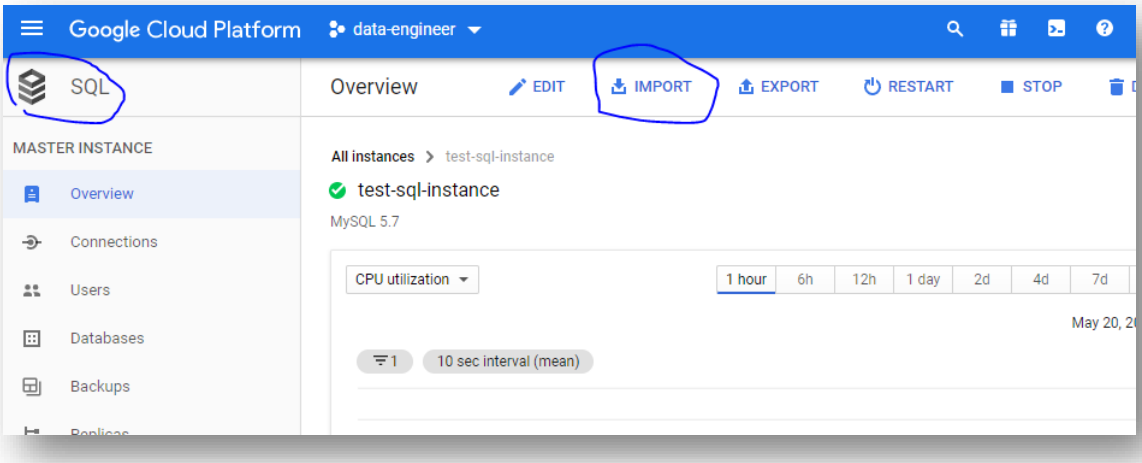
3. Browse to the CloudSQL sample data directory:
   - `cd googledataengineer/CPB100/lab3a/cloudsql`

4. Copy all data into your cloud bucket:
   - `gsutil cp * gs://(bucket_name)` **(‘*’ means transfer all the files)**

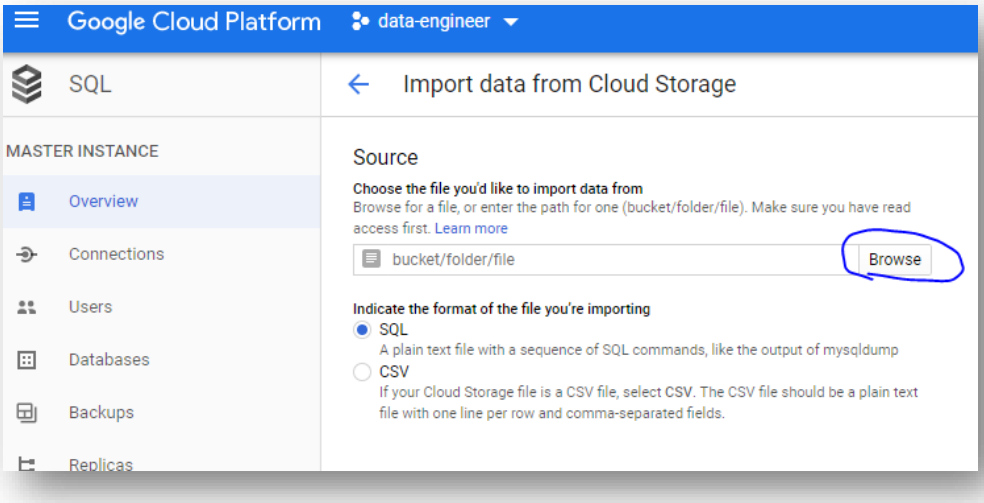
```
leonf_info@cloudshell:~/googledataengineer/CPB100/lab3a/cloudsql (data-engineer-274519)$ gsutil cp * gs://sql-practice
Copying file://accommodation.csv [Content-Type=text/csv]...
Copying file://rating.csv [Content-Type=text/csv]...
Copying file://table_creation.sql [Content-Type=application/x-sql]...
- [3 files][ 14.2 KiB/ 14.2 KiB]
Operation completed over 3 objects/14.2 KiB.
```



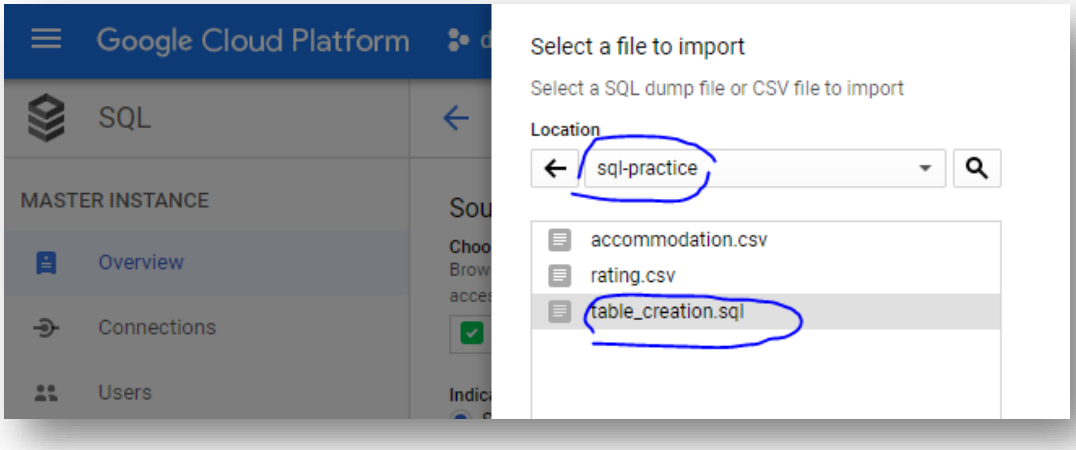
5. From **cloud storage bucket**, Import an SQL dump file into **Cloud SQL**:
   - From Cloud SQL, click on the instance and click the **Import** button



   - Click **Browse**, select **bucket**, browse to `table_creation.sql`, and click **Select**

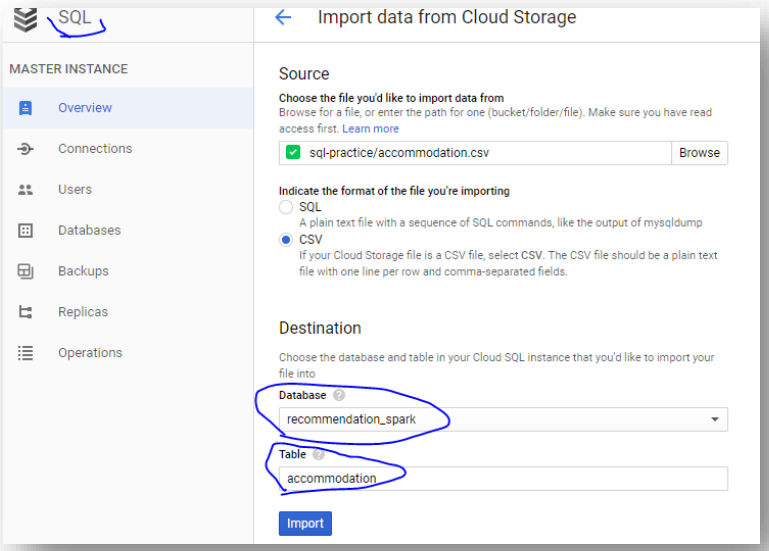


- Under the "Format of import" options, make sure **SQL** is selected as the import format (should be default).



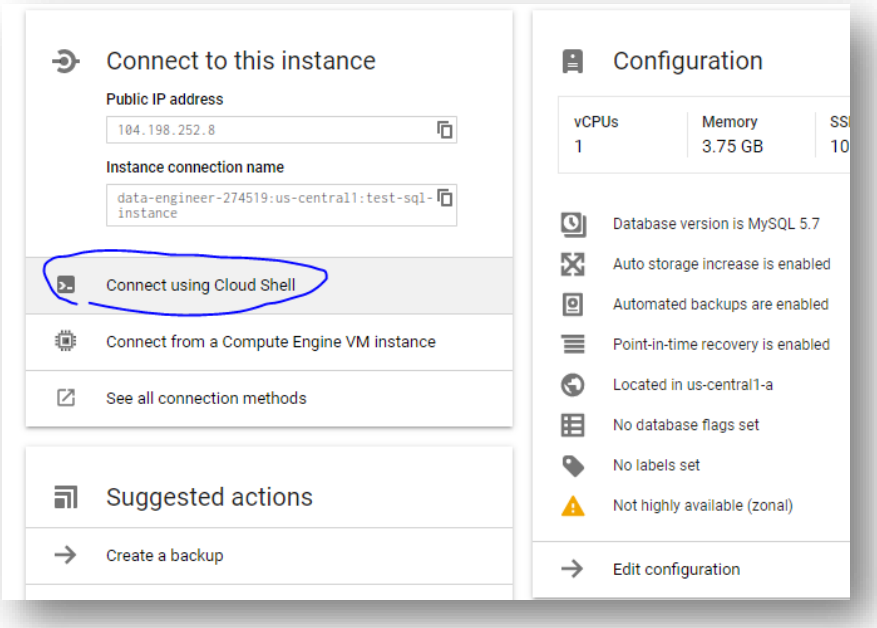
- Click **Import**.

6. Import CSV tables into SQL database (recommendation_spark):
   - From Cloud SQL, click on the instance and click the **Import** button.
   - Click **Browse**, select **bucket**, browse to `accommodation.csv`, and click **Select**.
   - Expand **advanced options**, and from the Database drop-down menu, select **recommendation_spark**.
   - Set the Table name to *Accommodation*.

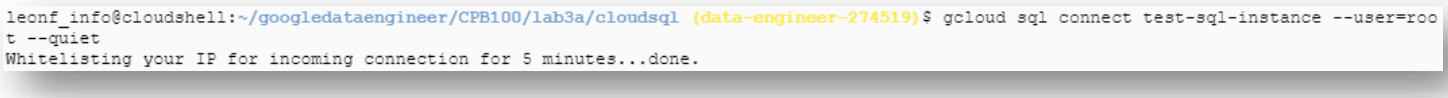


- Click **Import**.
- Perform the same actions for the `rating.csv` file as well, setting the Table name to **Rating**.

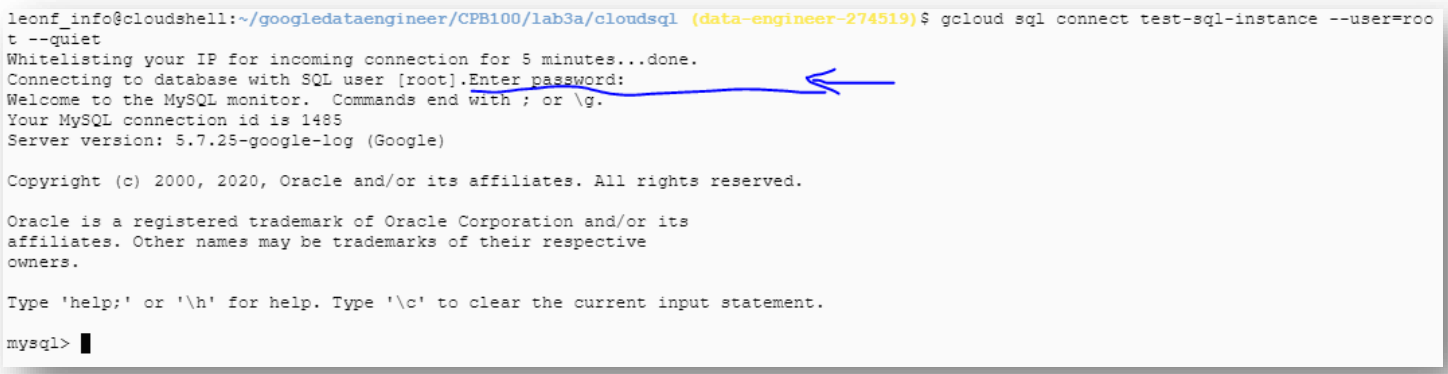
7. Connect to your Cloud SQL instance:
   - Click **Connect using Cloud Shell**.



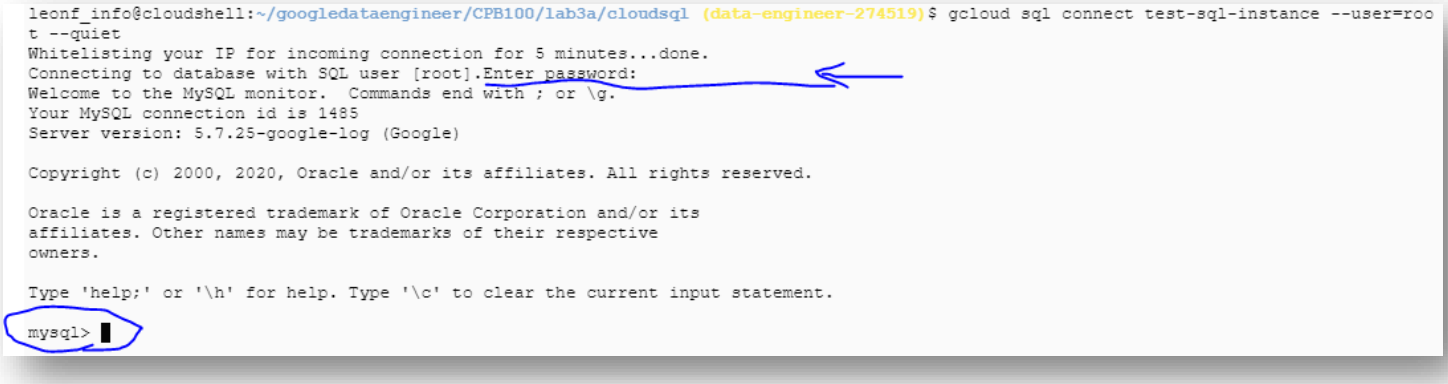
- In Cloud Shell, hit enter once the command is **populated**.



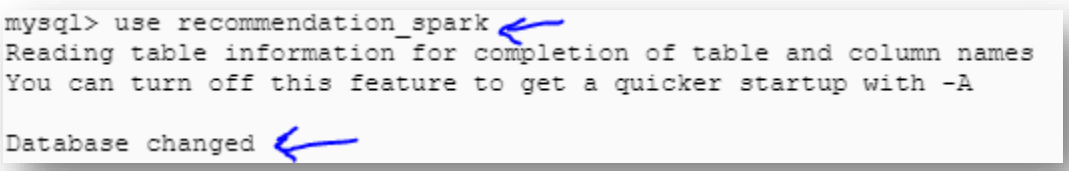
- Enter the root password when prompted.



- Now we are in MySQL!!



8. View tables and table data:
   - Switch to database:
     - `use recommendation_spark;`



- View tables:
  - `show tables;`

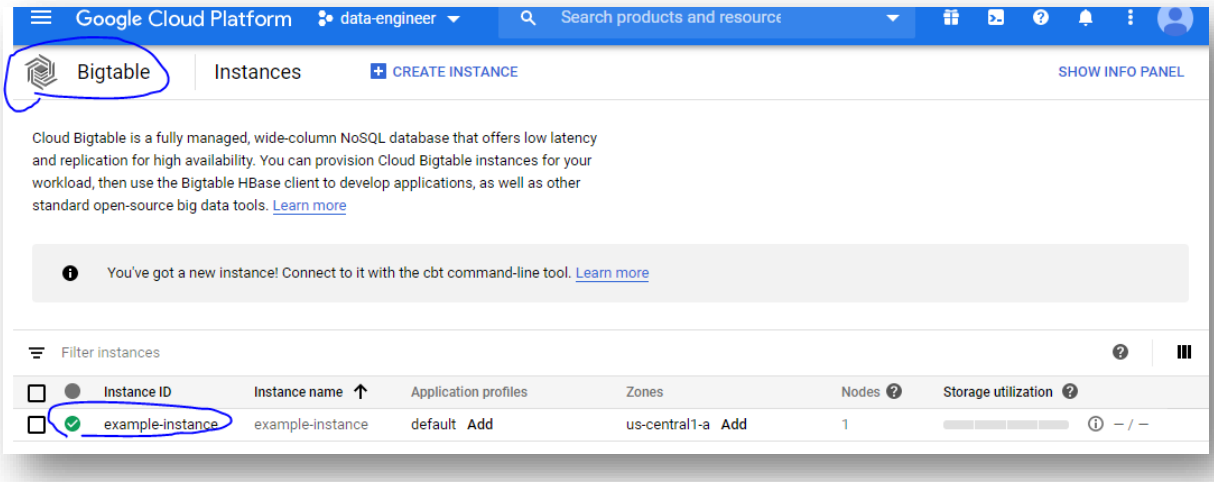
```
mysql> show tables;
+---------------------------------+
| Tables_in_recommendation_spark |
+---------------------------------+
| Accommodation                   |
| Rating                          |
| Recommendation                  |
+---------------------------------+
3 rows in set (0.03 sec)
```

- View contents of one of the tables:
  - `select * from Rating;`

```
mysql> select * from Accommodation;
+----+--------------------------------+---------------+-------+-------+--------+---------+
| id | title                          | location      | price | rooms | rating | type    |
+----+--------------------------------+---------------+-------+-------+--------+---------+
| 1  | Comfy Quiet Chalet             | Vancouver     |    50 |     3 |    3.1 | cottage |
| 10 | Sizable Calm Country House     | Auckland      |   650 |     9 |    4.9 | mansion |
| 11 | Homy Quiet Shanty              | Melbourne     |    50 |     1 |    2.8 | cottage |
| 12 | Beautiful Peaceful Villa       | Seattle       |    90 |     2 |    2.1 | house   |
| 13 | Enormous Peaceful Fortress     | Melbourne     |  3300 |    12 |    2.3 | castle  |
| 14 | Colossal Peaceful Palace       | Melbourne     |  1200 |    21 |    1.5 | castle  |
| 15 | Vast Private Fort              | London        |  1300 |    18 |    2.6 | castle  |
| 16 | Large Calm House               | Melbourne     |    45 |     3 |    4.1 | house   |
```

Lecture: Instance Configuration (Exam!)

In this lesson, we will go hands-on with creating a **Bigtable** instance (**expensive**!), and explore how to resize instances and some basic table interaction. The steps we took to interact with our table via `cbt` command line are below for your reference.



1. Install the `cbt` command in Google SDK:

- `gcloud components update`

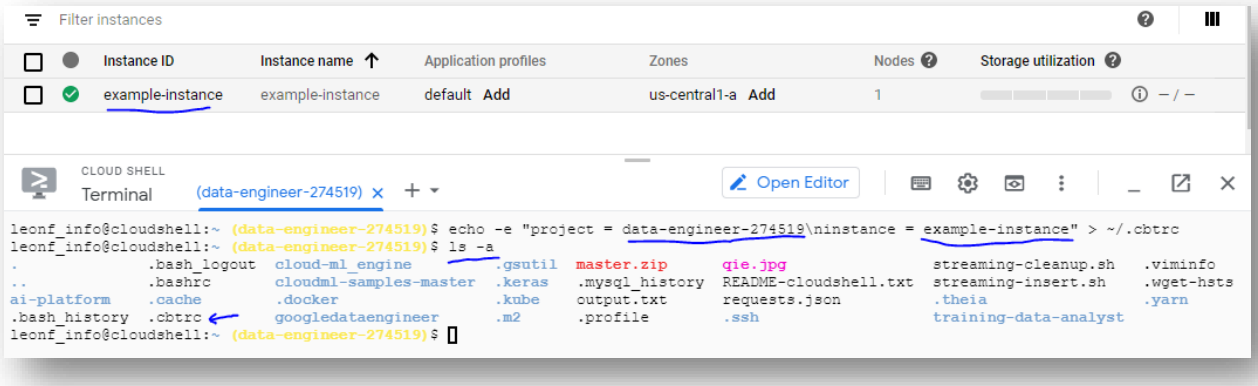
```
ERROR: (gcloud.components.update)
You cannot perform this action because the Cloud SDK component manager
is disabled for this installation. You can run the following command
to achieve the same result for this installation:

sudo apt-get update && sudo apt-get --only-upgrade install google-cloud-sdk-skaffold kubectl google-cloud-sdk-anthos-auth google-cloud-sdk-minikube google-cloud-sdk google-cloud-sdk-app-engine-grpc google-cloud-sdk-kind google-cloud-sdk-pubsub-emulator google-cloud-sdk-app-engine-go google-cloud-sdk-firestore-emulator google-cloud-sdk-cloud-build-local google-cloud-sdk-datastore-emulator google-cloud-sdk-kpt google-cloud-sdk-app-engine-python google-cloud-sdk-spanner-emulator google-cloud-sdk-cbt google-cloud-sdk-bigtable-emulator google-cloud-sdk-app-engine-python-extras google-cloud-sdk-datalab google-cloud-sdk-app-engine-java
```

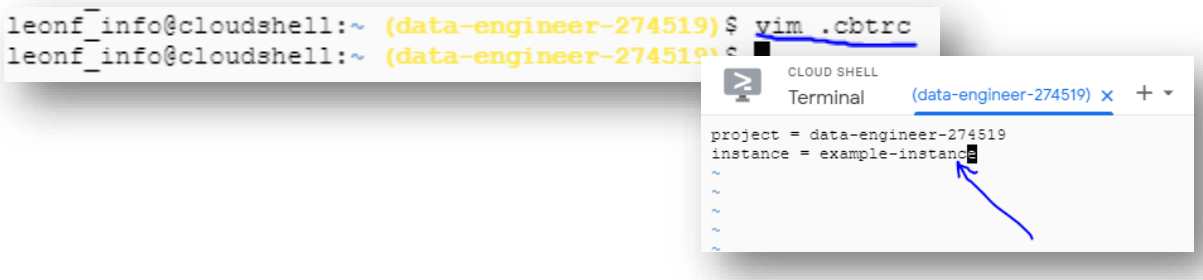
- `gcloud components install cbt`

```
leonf_info@cloudshell:~ (data-engineer-274519)$ gcloud components install cbt
ERROR: (gcloud.components.install)
You cannot perform this action because the Cloud SDK component manager
is disabled for this installation. You can run the following command
to achieve the same result for this installation:

sudo apt-get install google-cloud-sdk-cbt
```

2. Configure `cbt` to use your project and instance *via* `.cbtrc` file:

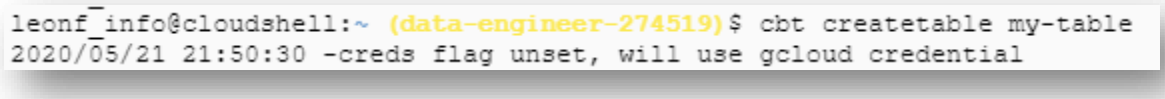
- `echo -e "project = [PROJECT_ID]\ninstance = [INSTANCE_ID]" > ~/.cbtrc`





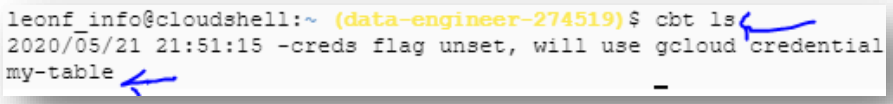
3. Create a table:

- `cbt createtable my-table`



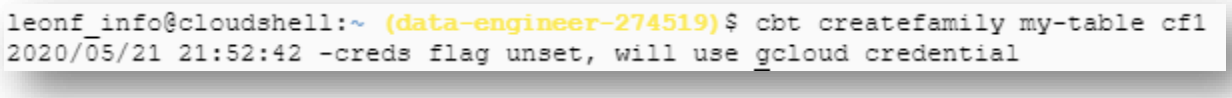
4. List the table (now we only have ‘my-table’):

- `cbt ls`



5. Add a column family:

- `cbt createfamily my-table cf1`



6. List column family

- `cbt ls my-table`

```
leonf_info@cloudshell:~ (data-engineer-274519)$ cbt ls my-table
2020/05/21 21:53:19 -creds flag unset, will use gcloud credential
Family Name     GC Policy
-----------     ---------
cf1             <never>
```

7. Add value to *row 1*, using column family *cf1* and column qualifier *c1*:

- `cbt set my-table r1 cf1:c1=test-value`

```
leonf_info@cloudshell:~ (data-engineer-274519)$ cbt set my-table r1 cf1:c1=test-value
2020/05/21 21:54:17 -creds flag unset, will use gcloud credential
```

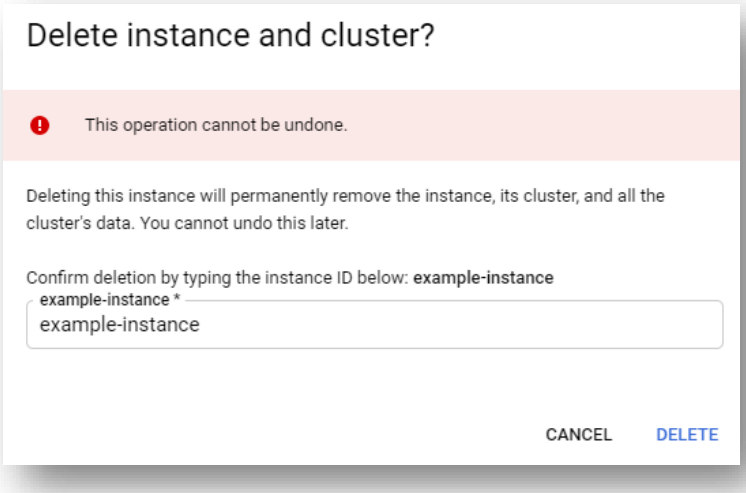
8. Use the `cbt read` command to read the data you added to the table:

- `cbt read my-table`

```
leonf_info@cloudshell:~ (data-engineer-274519)$ cbt read my-table
2020/05/21 21:54:45 -creds flag unset, will use gcloud credential
----------------------------------------
r1
  cf1:c1                                   @ 2020/05/21-21:54:18.618000
    "test-value"
```

9. Delete the table (if not deleting instance):

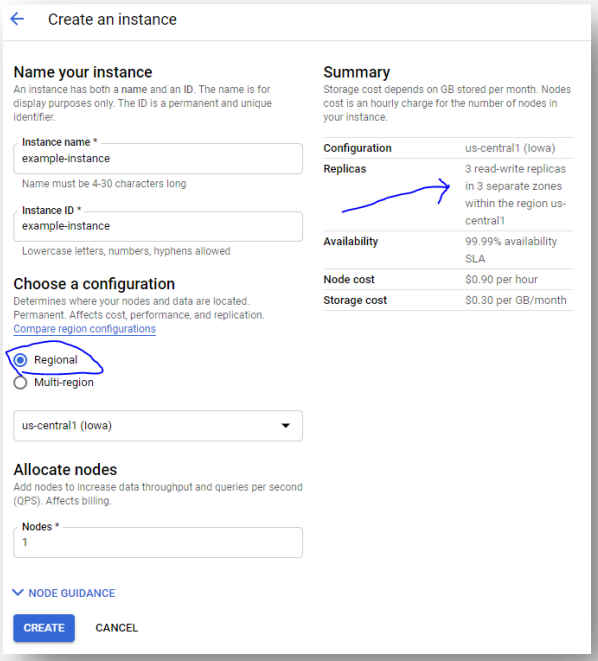
- `cbt deletetable my-table`



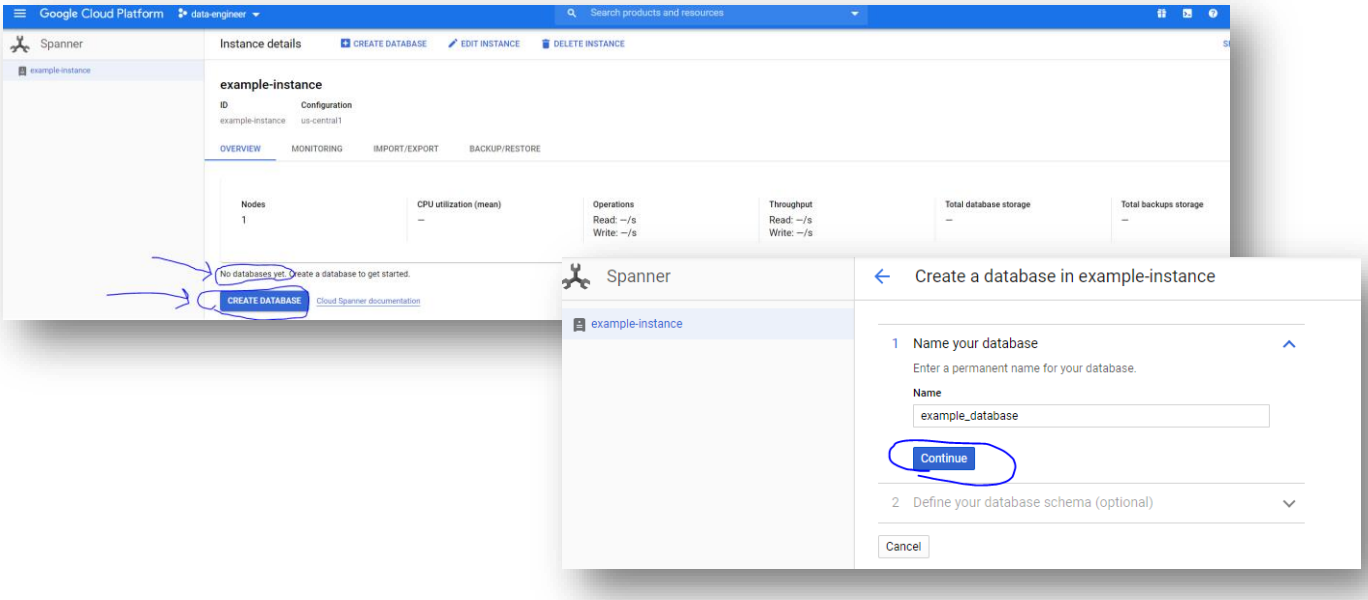
Lecture: Hands On and Viewing Examples (**Spanner**)

Create a Spanner instance, and populate another instance using Python scripts in order to exam a more complete database example.

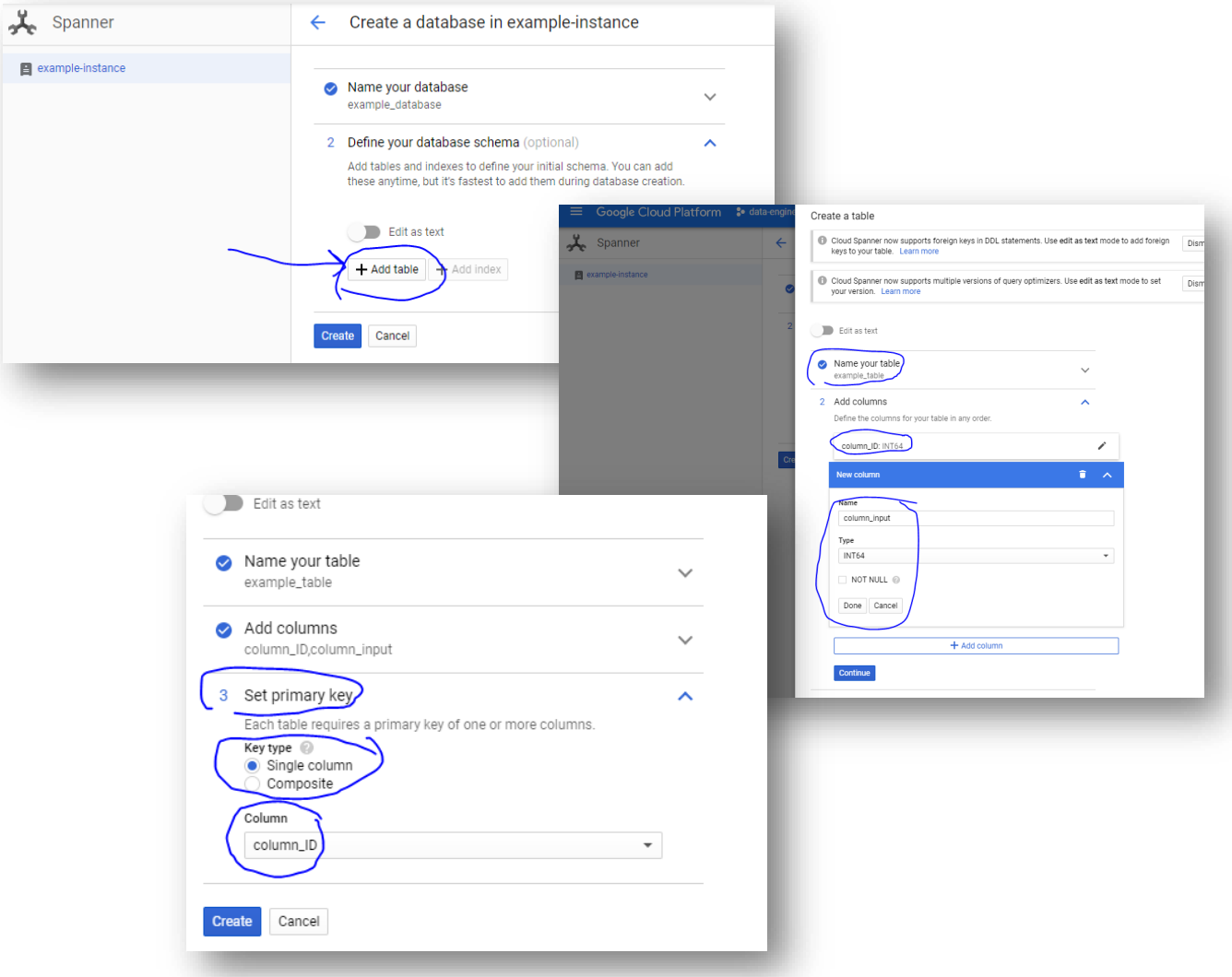
The below steps ran from Cloud Shell will create a Spanner instance and populate Singer and Album data for you.



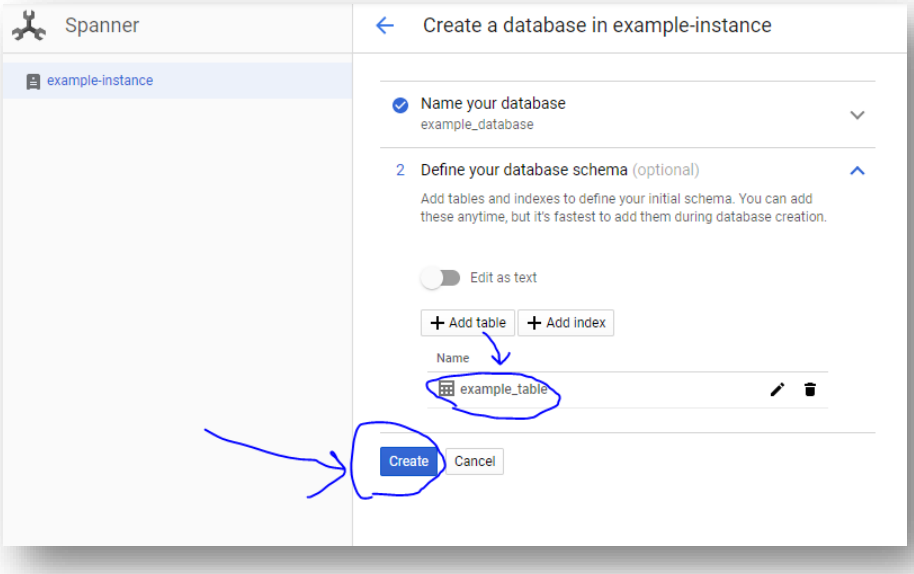
Create a **database** before we can add any **table**:



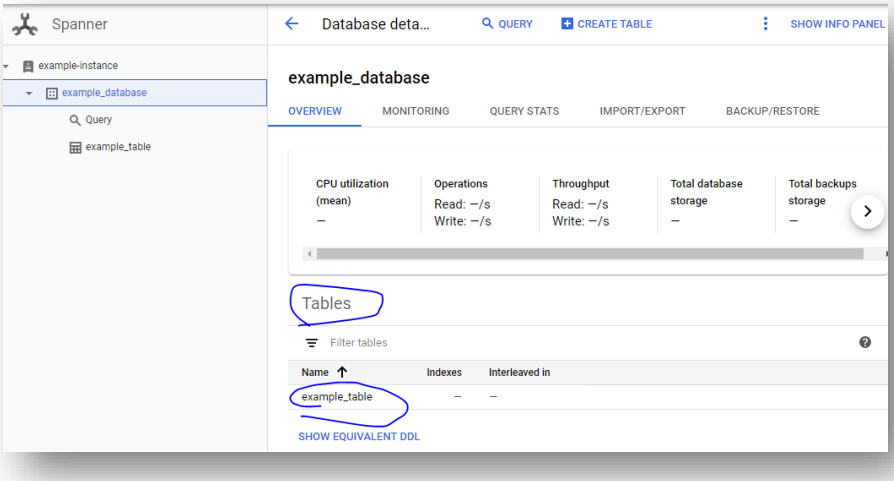
Now put **table** into the **database**:



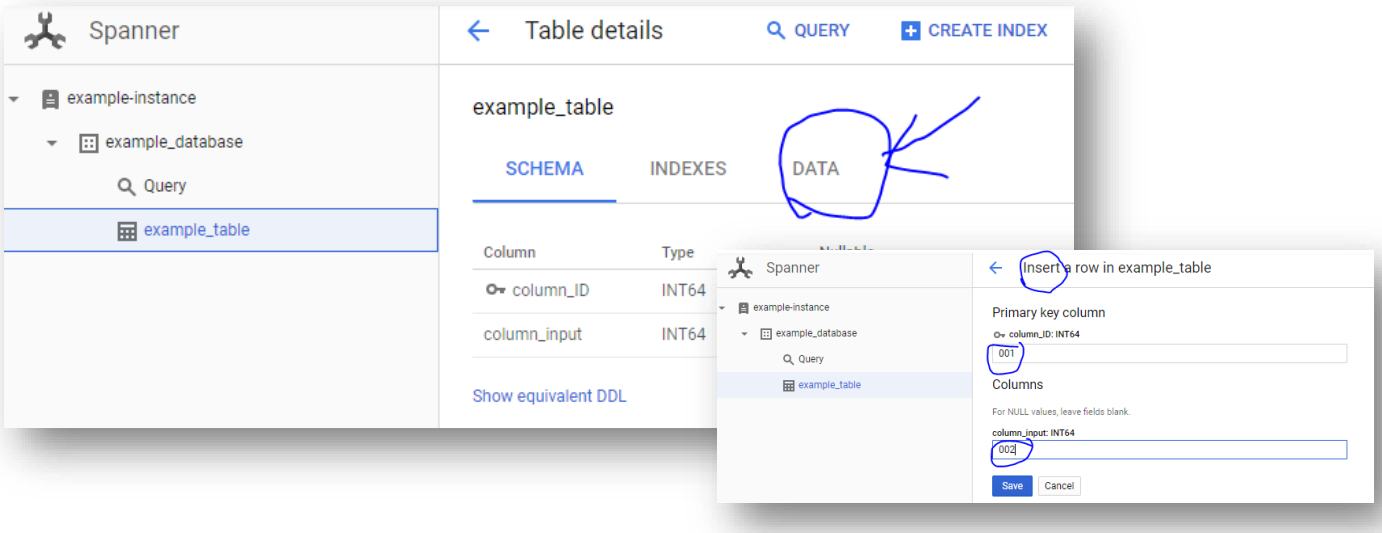
After all setting, click ‘create’:

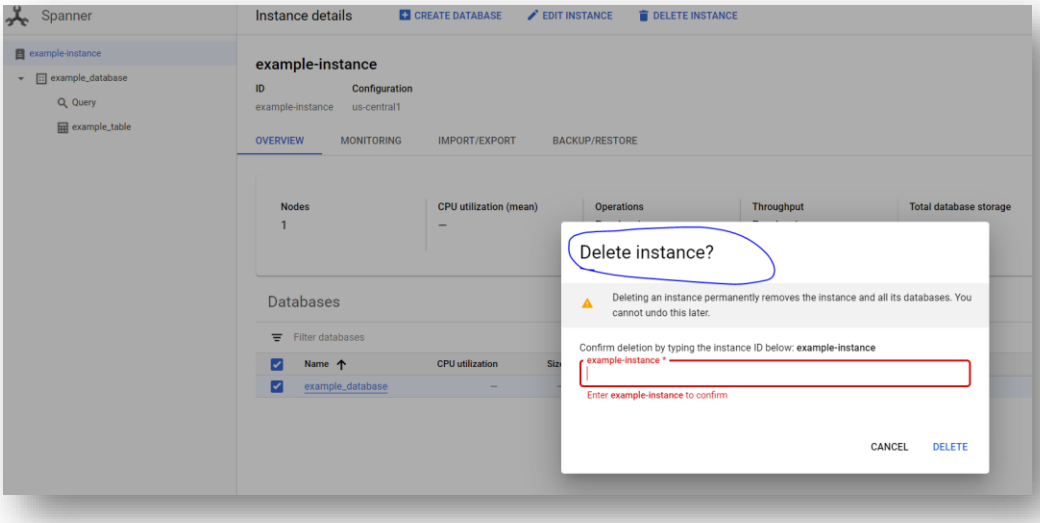


View the table we created:



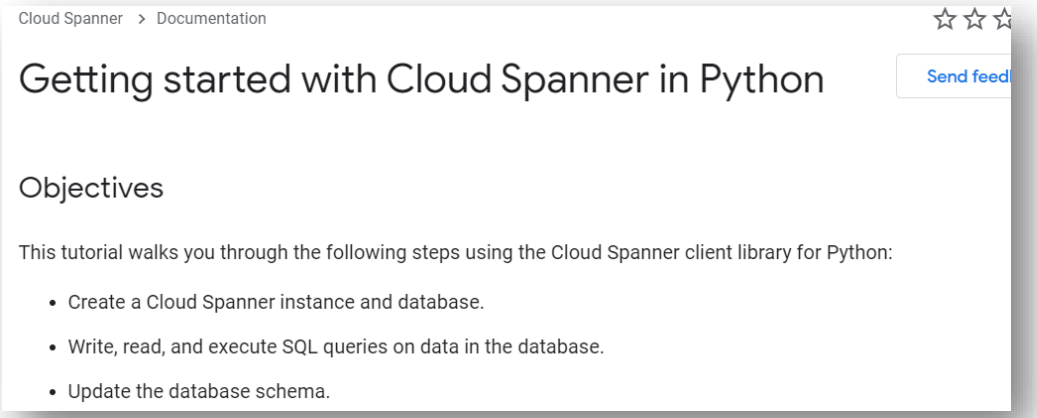
Add some data:





Another example with more data

1. Link Google Cloud documentation to follow along with for the Python example:
   - https://cloud.google.com/spanner/docs/getting-started/python/



2. Clone the GitHub repository to run scripts, and browse to the correct directory:
   - `git clone https://github.com/GoogleCloudPlatform/python-docs-samples.git`

```
leonf_info@cloudshell:~ (data-engineer-274519)$ git clone https://github.com/GoogleCloudPlatform/python-docs-samples.git

Cloning into 'python-docs-samples'...
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 31385 (delta 0), reused 4 (delta 0), pack-reused 31378
Receiving objects: 100% (31385/31385), 39.80 MiB | 42.76 MiB/s, done.
Resolving deltas: 100% (16846/16846), done.
```

   - `cd python-docs-samples/spanner/cloud-client`

```
leonf_info@cloudshell:~ (data-engineer-274519)$ cd python-docs-samples/spanner/cloud-client
leonf_info@cloudshell:~/python-docs-samples/spanner/cloud-client (data-engineer-274519)$
```

3. Create a Python environment and install dependencies:

4. `virtualenv env`

```
leonf_info@cloudshell:~/python-docs-samples/spanner/cloud-client (data-engineer-274519)$ virtualenv env
created virtual environment CPython3.7.3.final.0-64 in 759ms
  creator CPython3Posix(dest=/home/leonf_info/python-docs-samples/spanner/cloud-client/env, clear=False, global=False)
  seeder FromAppData(download=False, pip=latest, setuptools=latest, wheel=latest, via=copy, app_data_dir=/home/leonf_info/.local/share/v
irtualenv/seed-app-data/v1.0.1)
  activators BashActivator,CShellActivator,FishActivator,PowerShellActivator,PythonActivator,XonshActivator
```

5. `source env/bin/activate`

```
leonf_info@cloudshell:~/python-docs-samples/spanner/cloud-client (data-engineer-274519)$ source env/bin/activate
(env) leonf_info@cloudshell:~/python-docs-samples/spanner/cloud-client (data-engineer-274519)$
```

6. `pip install -r requirements.txt`

```
leonf_info@cloudshell:~/python-docs-samples/spanner/cloud-client (data-engineer-274519)$ source env/bin/activate
(env) leonf_info@cloudshell:~/python-docs-samples/spanner/cloud-client (data-engineer-274519)$ pip install -r requirements.txt
Ignoring futures: markers 'python_version < "3"' don't match your environment
Collecting google-cloud-spanner==1.16.0
  Downloading google_cloud_spanner-1.16.0-py2.py3-none-any.whl (249 kB)
     |████████████████████████████████| 249 kB 2.8 MB/s
Collecting google-cloud-core<2.0dev,>=1.0.3
  Downloading google_cloud_core-1.3.0-py2.py3-none-any.whl (26 kB)
Collecting grpc-google-iam-v1<0.13dev,>=0.12.3
  Downloading grpc-google-iam-v1-0.12.3.tar.gz (13 kB)
Collecting google-api-core[grpc,grpcgcp]<2.0.0dev,>=1.14.0
```

7. Create a Spanner instance named *test-instance*:
   - `gcloud spanner instances create test-instance --config=regional-us-central1 --description="Test Instance" --nodes=1`

```
(env) leonf_info@cloudshell:~/python-docs-samples/spanner/cloud-client (data-engineer-274519)$ gcloud spanner instances create test-inst
ance --config=regional-us-central1 --description="Test Instance" --nodes=1
Creating instance...done.
```

8. Create a database and insert data using the Python scripts from our GitHub clone:
   - `python snippets.py test-instance --database-id example-db create_database`

```
(env) leonf_info@cloudshell:~/python-docs-samples/spanner/cloud-client (data-engineer-274519)$ python snippets.py test-instance --databa
se-id example-db create_database
Waiting for operation to complete...
Created database example-db on instance test-instance
```

   - `python snippets.py test-instance --database-id example-db insert_data`

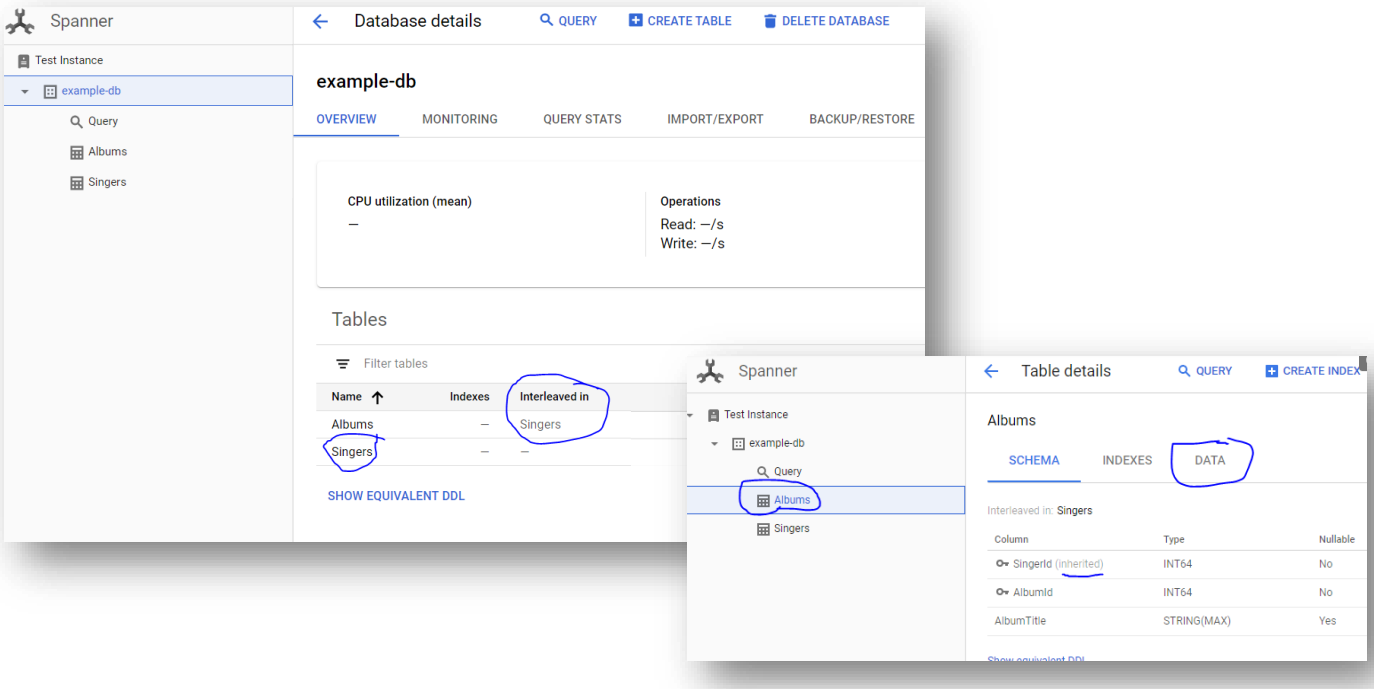
```
(env) leonf_info@cloudshell:~/python-docs-samples/spanner/cloud-client (data-engineer-274519)$ python snippets.py test-instance --databa
se-id example-db insert_data
Inserted data.
```

9. Run a query to read the values of all columns from the Albums table:
   - `gcloud spanner databases execute-sql example-db --instance=test-instance --sql='SELECT SingerId, AlbumId, AlbumTitle FROM Albums'`

```
(env) leonf_info@cloudshell:~/python-docs-samples/spanner/cloud-client (data-engineer-274519)$ gcloud spanner databases execute-sql exam
ple-db --instance=test-instance --sql='SELECT SingerId, AlbumId, AlbumTitle FROM Albums'
SingerId  AlbumId  AlbumTitle
1         1        Total Junk
1         2        Go, Go, Go
2         1        Green
2         2        Forever Hold Your Peace
2         3        Terrified
```

Check all the tables:

Albums table is **child** table of Singers table

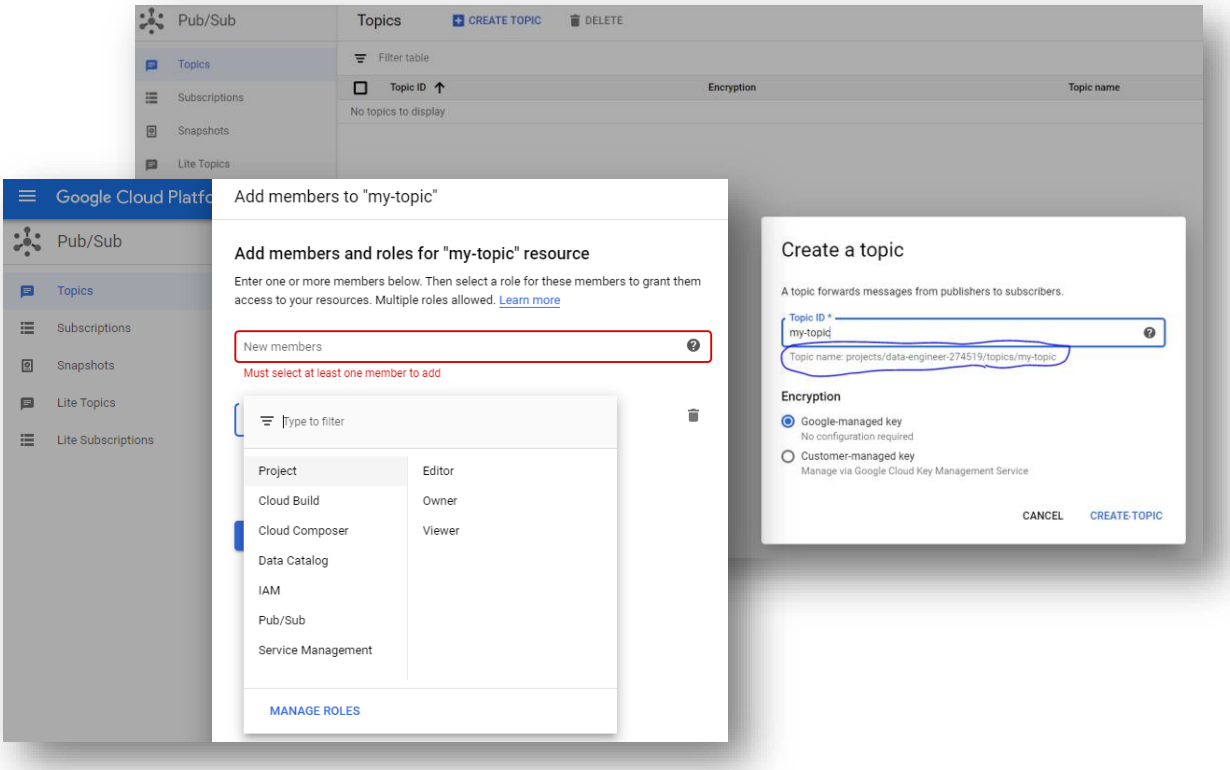


Lecture: Pub/Sub Hands On

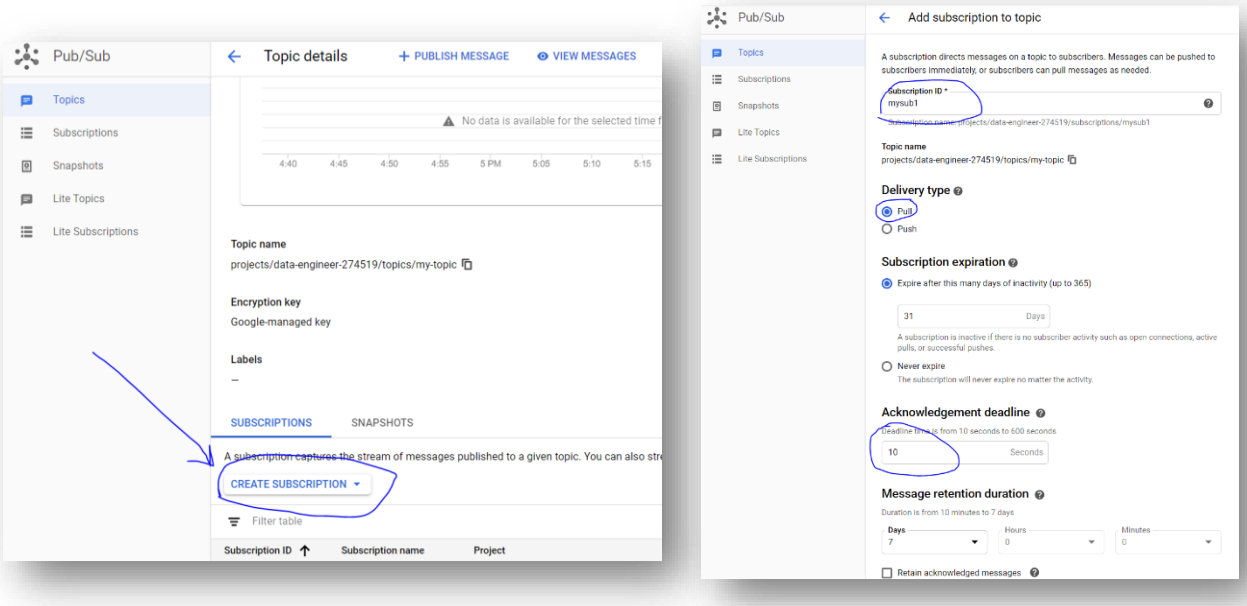
Demonstration of Cloud Pub/Sub using both the web console and gcloud command line.

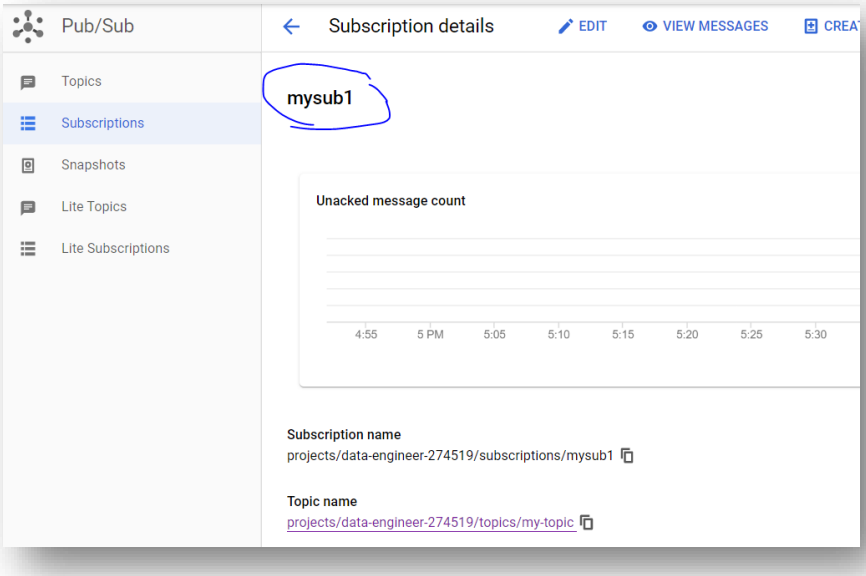
Web console demonstration:

1. Create a **topic** called *my-topic*:
   - `gcloud pubsub topics create my-topic`
   - We can also assign assess: project wide, topic wide, and subscription wide.

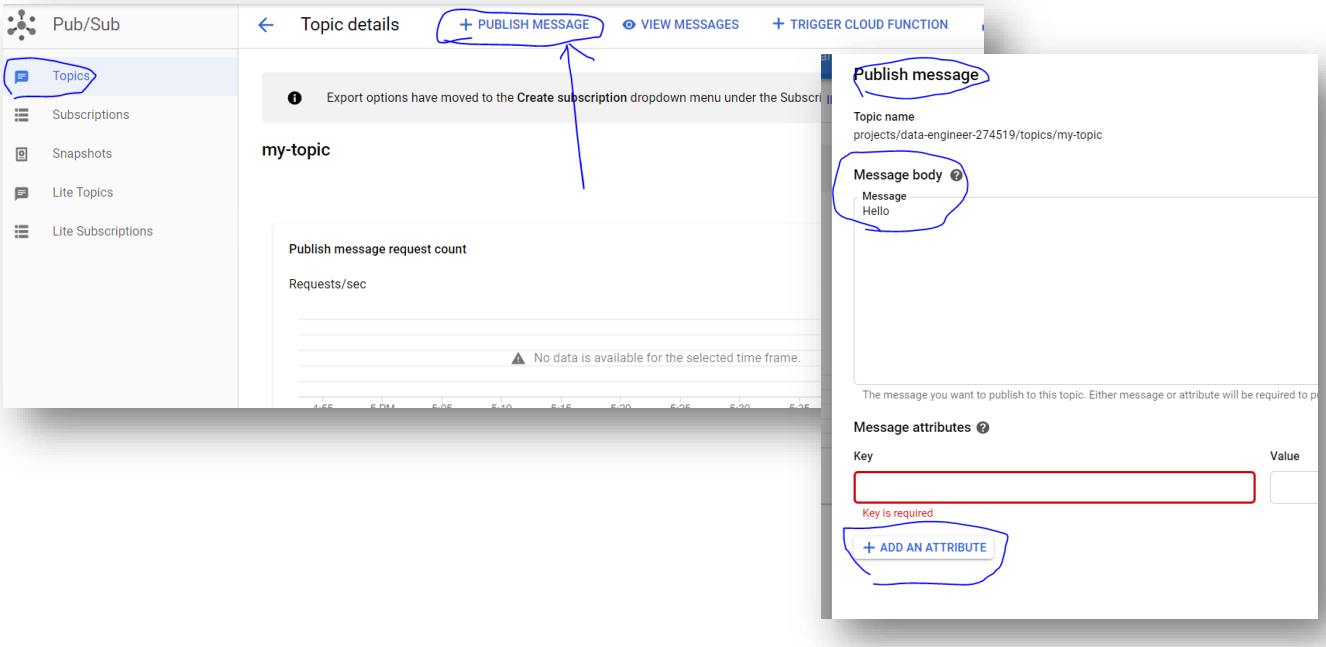


2. Create a **subscription** to topic *my-topic*:
   - `gcloud pubsub subscriptions create --topic my-topic mySub1`





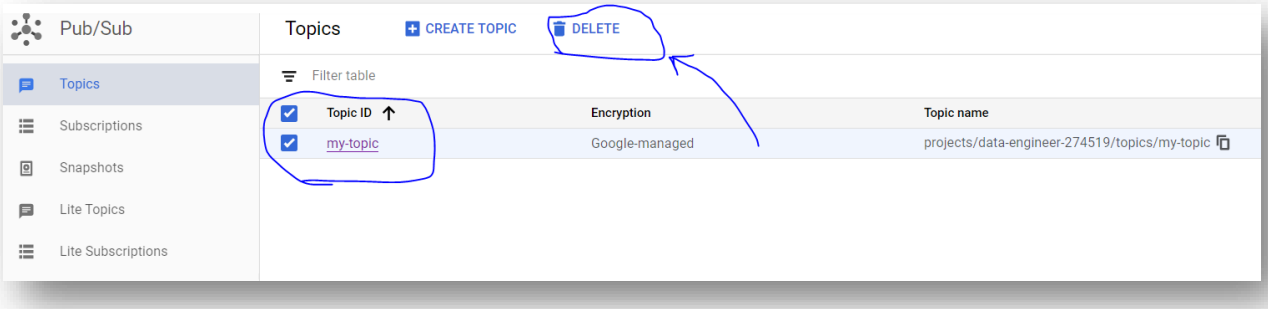
3. **Publish** a message to your topic:
    - `gcloud pubsub topics publish my-topic --message "hello"`



4. **Retrieve** *message* with your subscription, acknowledge receipt and remove *message* from the queue:
    - `gcloud pubsub subscriptions pull --auto-ack mySub1`
    - (**We have to use command line to do this step)**

```
leonf_info@cloudshell:~ (data-engineer-274519)$ gcloud pubsub subscriptions pull --auto-ack mysub1
┌──────┬─────────────────┬────────────┬──────────────────┐
│ DATA │   MESSAGE_ID    │ ATTRIBUTES │ DELIVERY_ATTEMPT │
├──────┼─────────────────┼────────────┼──────────────────┤
│ hello│ 448325824473138 │            │                  │
└──────┴─────────────────┴────────────┴──────────────────┘
```

5. Cancel subscription:
    - `gcloud pubsub subscriptions delete mySub1`



6. Simulated traffic ingest:
    - Clone GitHub data to Cloud Shell (or other SDK environments), and browse to the *publish* folder:
    - `cd ~ git clone https://github.com/linuxacademy/googledataengineer`

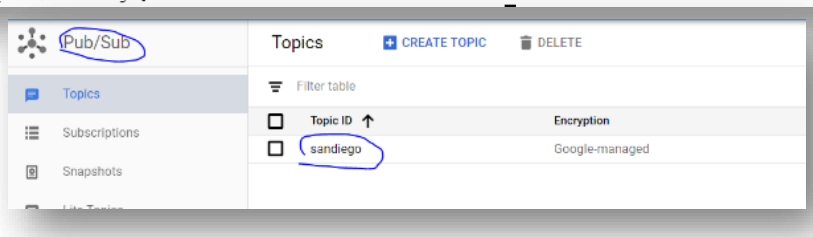
```
leonf_info@cloudshell:~ (data-engineer-274519)$ cd ~
leonf_info@cloudshell:~ (data-engineer-274519)$ git clone https://github.com/linuxacademy/googledataengineer
fatal: destination path 'googledataengineer' already exists and is not an empty directory.
```

    - `cd ~/googledataengineer/courses/streaming/publish`

```
leonf_info@cloudshell:~ (data-engineer-274519)$ cd ~/googledataengineer/courses/streaming/publish
leonf_info@cloudshell:~/googledataengineer/courses/streaming/publish (data-engineer-274519)$ ls
download_data.sh  pubsub_pull.py  README.txt  send_sensor_data.py  sensor_obs2008.csv.gz
```

7. Create a topic called *sandiego*:
    - `gcloud pubsub topics create sandiego`

```
leonf_info@cloudshell:~/googledataengineer/courses/streaming/publish (data-engineer-274519)$ gcloud pubsub topics create sandiego
Created topic [projects/data-engineer-274519/topics/sandiego].
```



8. Create a subscription to topic *sandiego*:
    - `gcloud pubsub subscriptions create --topic sandiego mySub1`

```
leonf_info@cloudshell:~/googledataengineer/courses/streaming/publish (data-engineer-274519)$ gcloud pubsub subscriptions create --topic sandiego mySub1
Created subscription [projects/data-engineer-274519/subscriptions/mySub1].
```

9. Run script to download sensor data:
    - `./download_data.sh`

```
leonf_info@cloudshell:~/googledataengineer/courses/streaming/publish (data-engineer-274519)$ vim download_data.sh
```

```
gsutil cp gs://la-gcloud-course-resources/sandiego/sensor_obs2008.csv.gz .
~
```

```
leonf_info@cloudshell:~/googledataengineer/courses/streaming/publish (data-engineer-274519)$ ./download_data.sh
Copying gs://la-gcloud-course-resources/sandiego/sensor_obs2008.csv.gz...
\ [1 files][ 34.6 MiB/ 34.6 MiB]
Operation completed over 1 objects/34.6 MiB.
leonf_info@cloudshell:~/googledataengineer/courses/streaming/publish (data-engineer-274519)$ ls
download_data.sh  pubsub_pull.py  README.txt  send_sensor_data.py  sensor_obs2008.csv.gz
```

(Optional). If you need to authenticate a shell to ensure we have the right permissions:

- `gcloud auth application-default login`

10. View script info:
    - `vim ./send_sensor_data.py or use viewer of your choice`
11. Run python script to simulate one hour of data per minute:
    - `./send_sensor_data.py --speedFactor=60 --project=YOUR-PROJECT-ID`

```
leonf_info@cloudshell:~/googledataengineer/courses/streaming/publish (data-engineer-274519)$ ./send_sensor_data.py --spee
*******************************************************************************
Python 2 is deprecated. Upgrade to Python 3 as soon as possible.
See https://cloud.google.com/python/docs/python2-sunset

To suppress this warning, create an empty ~/.cloudshell/no-python-warning file.
The command will automatically proceed in  seconds or on any key.
```

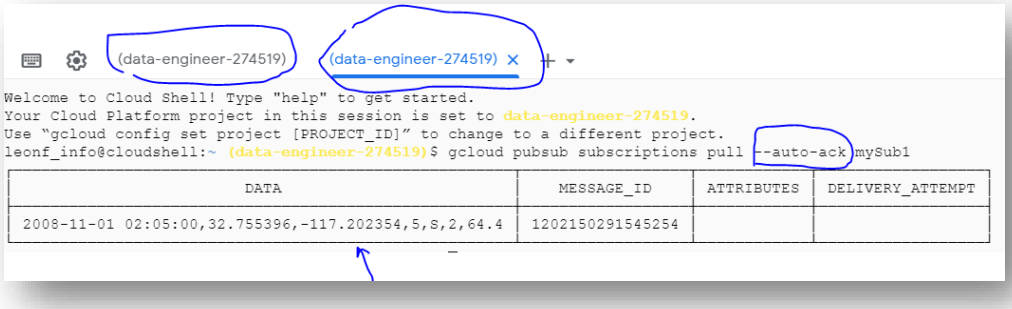
If you receive an error: *google.cloud.pubsub* cannot be found OR 'ImportError: No module named iterator', run the below `pip` command to install components then try again:

- `sudo pip install -U google-cloud-pubsub`

```
leonf_info@cloudshell:~/googledataengineer/courses/streaming/publish (data-engineer-274519)$ sudo pip install -U google-cloud-pubsub
********************************************************************************
Python 2 is deprecated. Upgrade to pip3 as soon as possible.
See https://cloud.google.com/python/docs/python2-sunset

To suppress this warning, create an empty ~/.cloudshell/no-pip-warning file.
The command will automatically proceed in 5 seconds or on any key.
********************************************************************************
DEPRECATION: Python 2.7 reached the end of its life on January 1st, 2020. Please upgrade your Python as Python 2.7 is no longer maintained. pip 21.0 will drop support for Python 2.7 in January 2021. More details about Python 2 support in
https://pip.pypa.io/en/latest/development/release-process/#python-2-support
Collecting google-cloud-pubsub
  Downloading google_cloud_pubsub-1.5.0-py2.py3-none-any.whl (137 kB)
     |████████████████████████████████| 137 kB 2.7 MB/s
Requirement already satisfied, skipping upgrade: enum34; python_version < "3.4" in /usr/local/lib/python2.7/dist-packages (from google-cloud-pubsub) (1.1.10)
Collecting google-api-core[grpc]<1.17.0,>=1.14.0
  Downloading google_api_core-1.16.0-py2.py3-none-any.whl (70 kB)
     |████████████████████████████████| 70 kB 3.9 MB/s
Requirement already satisfied, skipping upgrade: grpc-google-iam-v1<0.13dev,>=0.12.3 in /usr/local/lib/python2.7/dist-packages (from google-cloud-pubsub) (0.12.3)
Requirement already satisfied, skipping upgrade: six>=1.10.0 in /usr/local/lib/python2.7/dist-packages (from google-api-core[grpc]<1.17.0,>=1.14.0->google-cloud-pubsub) (1.14.0)
Requirement already satisfied, skipping upgrade: googleapis-common-protos<2.0dev,>=1.6.0 in /usr/local/lib/python2.7/dist-packages (from google-api-core[grpc]<1.17.0,>=1.14.0->google-cloud-pubsub) (1.51.0)
Requirement already satisfied, skipping upgrade: google-auth<2.0dev,>=0.4.0 in /usr/local/lib/python2.7/dist-packages (from google-api-core[grpc]<1.17.0,>=1.14.0->google-cloud-pubsub) (1.15.0)
Requirement already satisfied, skipping upgrade: setuptools>=34.0.0 in /usr/local/lib/python2.7/dist-packages (from google-api-core[grpc]<1.17.0,>=1.14.0->google-cloud-pubsub) (44.1.0)
Requirement already satisfied, skipping upgrade: protobuf>=3.4.0 in /usr/local/lib/python2.7/dist-packages (from google-api-core[grpc]<1.17.0,>=1.14.0->google-cloud-pubsub) (3.12.1)
Requirement already satisfied, skipping upgrade: requests<3.0.0dev,>=2.18.0 in /usr/local/lib/python2.7/dist-packages (from google-api-core[grpc]<1.17.0,>=1.14.0->google-cloud-pubsub) (2.23.0)
Requirement already satisfied, skipping upgrade: pytz in /usr/local/lib/python2.7/dist-packages (from google-api-core[grpc]<1.17.0,>=1.14.0->google-cloud-pubsub) (2020.1)
Requirement already satisfied, skipping upgrade: futures>=3.2.0; python_version < "3.2" in /usr/local/lib/python2.7/dist-packages (from google-api-core[grpc]<1.17.0,>=1.14.0->google-cloud-pubsub) (3.3.0)
Requirement already satisfied, skipping upgrade: grpcio<2.0dev,>=1.8.2; extra == "grpc" in /usr/local/lib/python2.7/dist-packages (from google-api-core[grpc]<1.17.0,>=1.14.0->google-cloud-pubsub) (1.29.0)
Requirement already satisfied, skipping upgrade: rsa<4.1,>=3.1.4 in /usr/local/lib/python2.7/dist-packages (from google-auth<2.0dev,>=0.4.0->google-api-core[grpc]<1.17.0,>=1.14.0->google-cloud-pubsub) (4.0)
Requirement already satisfied, skipping upgrade: pyasn1-modules>=0.2.1 in /usr/local/lib/python2.7/dist-packages (from google-auth<2.0dev,>=0.4.0->google-api-core[grpc]<1.17.0,>=1.14.0->google-cloud-pubsub) (0.2.8)
Requirement already satisfied, skipping upgrade: cachetools<5.0,>=2.0.0 in /usr/local/lib/python2.7/dist-packages (from google-auth<2.0dev,>=0.4.0->google-api-core[grpc]<1.17.0,>=1.14.0->google-cloud-pubsub) (3.1.1)
Requirement already satisfied, skipping upgrade: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python2.7/dist-packages (from requests<3.0.0dev,>=2.18.0->google-api-core[grpc]<1.17.0,>=1.14.0->google-cloud-pubsub) (1.25.9)
Requirement already satisfied, skipping upgrade: certifi>=2017.4.17 in /usr/local/lib/python2.7/dist-packages (from requests<3.0.0dev,>=2.18.0->google-api-core[grpc]<1.17.0,>=1.14.0->google-cloud-pubsub) (2020.4.5.1)
Requirement already satisfied, skipping upgrade: chardet<4,>=3.0.2 in /usr/local/lib/python2.7/dist-packages (from requests<3.0.0dev,>=2.18.0->google-api-core[grpc]<1.17.0,>=1.14.0->google-cloud-pubsub) (3.0.4)
Requirement already satisfied, skipping upgrade: idna<3,>=2.5 in /usr/local/lib/python2.7/dist-packages (from requests<3.0.0dev,>=2.18.0->google-api-core[grpc]<1.17.0,>=1.14.0->google-cloud-pubsub) (2.9)
Requirement already satisfied, skipping upgrade: pyasn1>=0.1.3 in /usr/local/lib/python2.7/dist-packages (from rsa<4.1,>=3.1.4->google-auth<2.0dev,>=0.4.0->google-api-core[grpc]<1.17.0,>=1.14.0->google-cloud-pubsub) (0.4.8)
Installing collected packages: google-api-core, google-cloud-pubsub
  Attempting uninstall: google-api-core
    Found existing installation: google-api-core 1.17.0
    Uninstalling google-api-core-1.17.0:
      Successfully uninstalled google-api-core-1.17.0
Successfully installed google-api-core-1.16.0 google-cloud-pubsub-1.5.0
```

```
leonf_info@cloudshell:~/googledataengineer/courses/streaming/publish (data-engineer-274519)$ ./send_sensor_data.py --speedFactor=60 --project=data-engineer-274519
********************************************************************************
Python 2 is deprecated. Upgrade to Python 3 as soon as possible.
See https://cloud.google.com/python/docs/python2-sunset

To suppress this warning, create an empty ~/.cloudshell/no-python-warning file.
The command will automatically proceed in  seconds or on any key.
********************************************************************************
INFO: Reusing pub/sub topic sandiego
INFO: Sending sensor data from 2008-11-01 00:00:00
INFO: Publishing 477 events from 2008-11-01 00:00:00
INFO: Sleeping 5.0 seconds
INFO: Publishing 477 events from 2008-11-01 00:05:00
INFO: Sleeping 5.0 seconds
```

12. Open new Cloud Shell tab (using + symbol):
    - Pull the message using the subscription *mySub1*
      - `gcloud pubsub subscriptions pull --auto-ack mySub1`



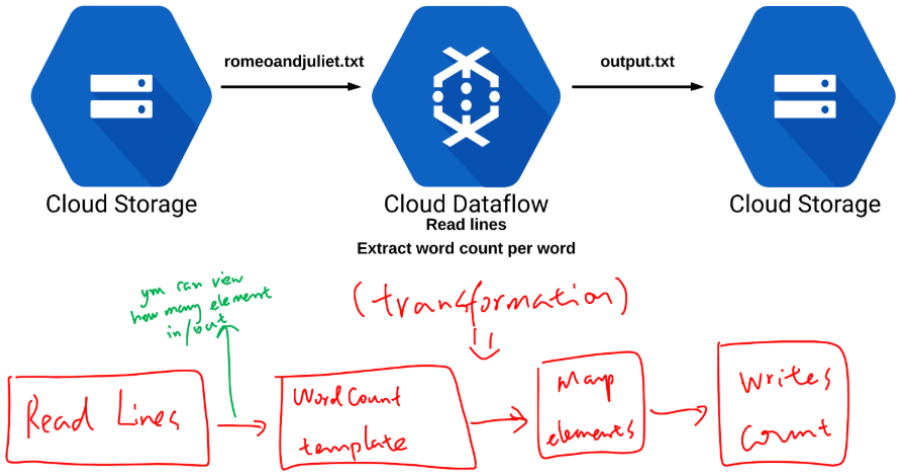
13. Create a new subscription and pull messages with it:
    - `gcloud pubsub subscriptions create --topic sandiego mySub2`
    - `gcloud pubsub subscriptions pull --auto-ack mySub2`

```
leonf_info@cloudshell:~ (data-engineer-274519)$ gcloud pubsub subscriptions create --topic sandiego mySub2
Created subscription [projects/data-engineer-274519/subscriptions/mySub2].
leonf_info@cloudshell:~ (data-engineer-274519)$ gcloud pubsub subscriptions pull --auto-ack mySub2
```

DATA	MESSAGE_ID	ATTRIBUTES	DELIVERY_ATTEMPT
2008-11-01 05:30:00,32.784015,-117.16195,163,S,1,77.1	1202128459831768		

Lecture: Template Hands On

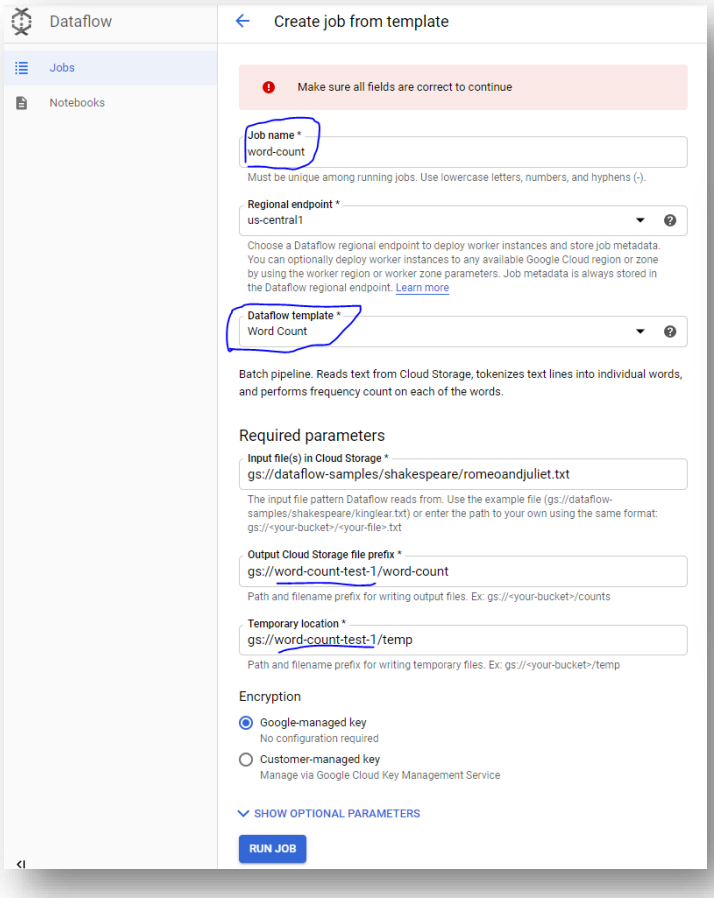
In the first of our two hands-on demonstrations, we are going to use a predefined template to conduct a word count on a classic Shakespeare play using Dataflow.



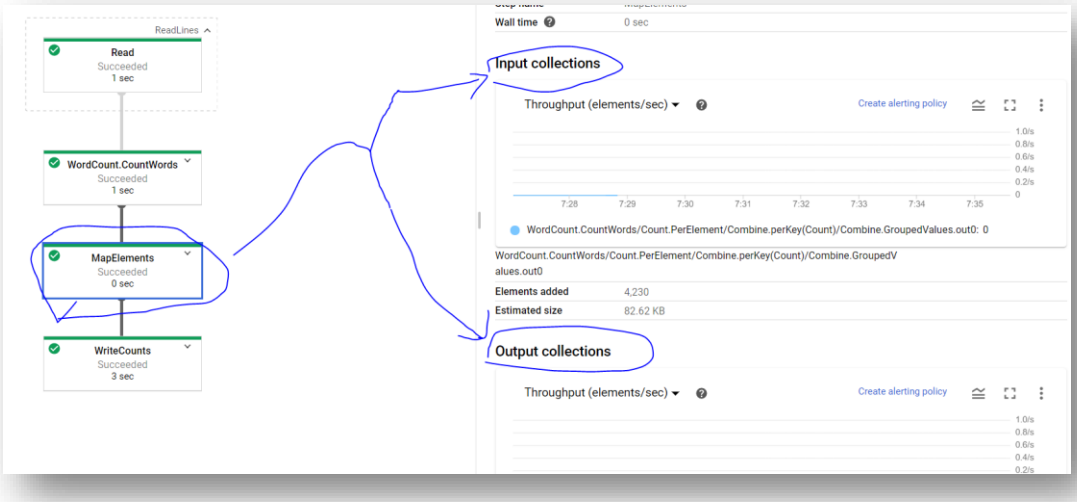
1. Create a storage bucket:

```
leonf_info@cloudshell:~/googledataengineer/courses/streaming/publish (data-engineer-274519)$ gsutil mb gs://word-count-test-1
Creating gs://word-count-test-1/...
```

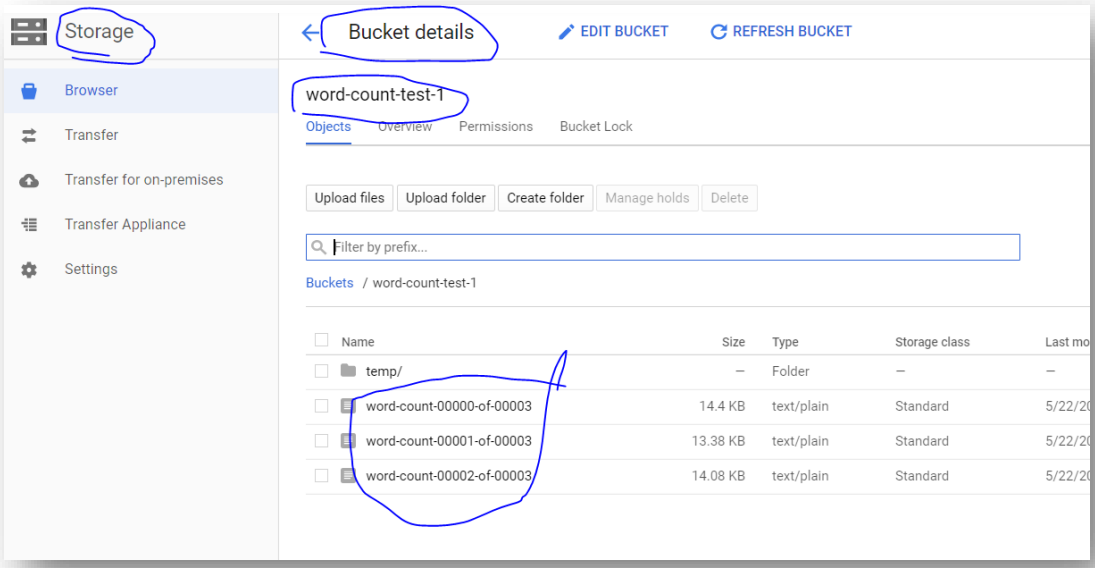
2. Create a job:
   - (specify the input and output location in storage buckets)

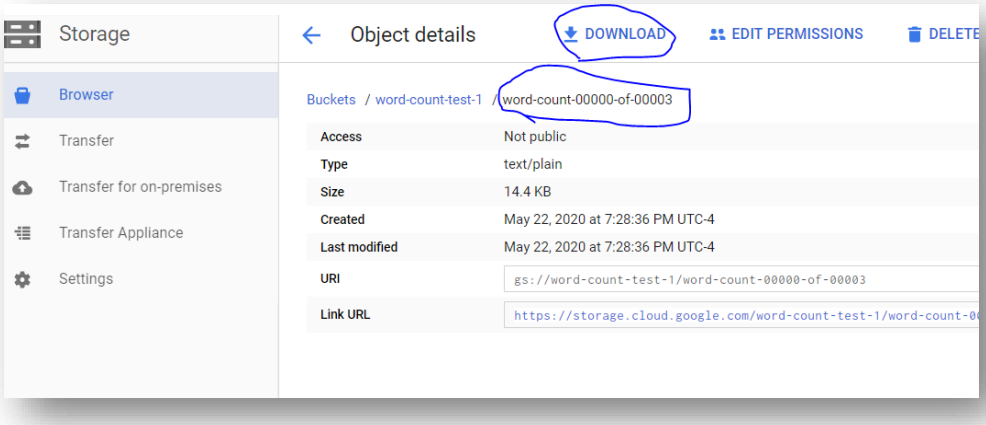


Watch the ‘batch processing’ job:



3. Check the result:

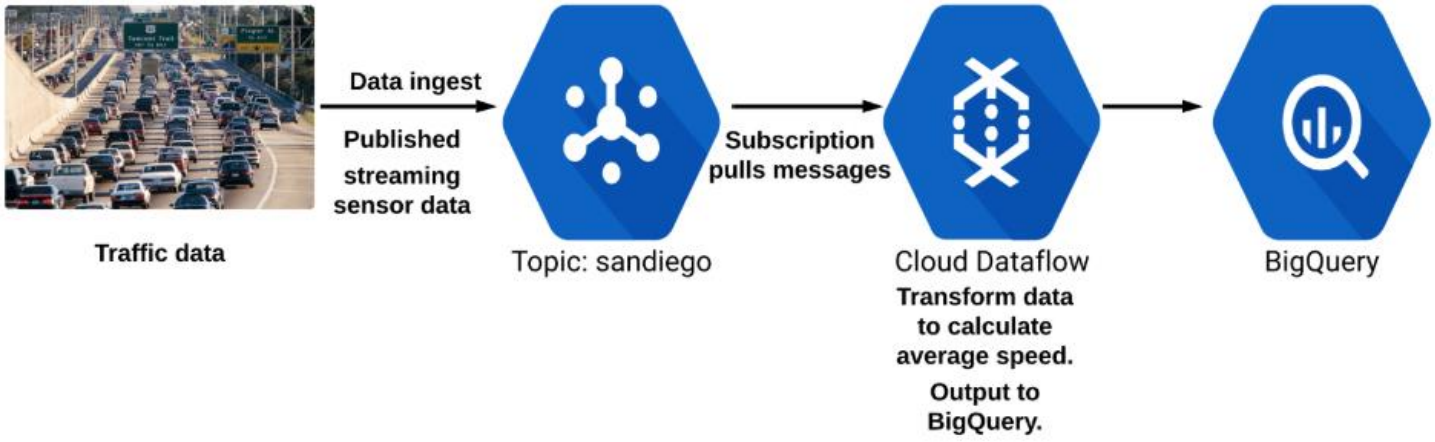





```
Montague: 25
child's: 1
feather'd: 1
mourners: 1
scatter'd: 1
she: 96
importuned: 1
lisping: 1
sudden: 7
More: 7
aspired: 1
'twill: 1
out: 48
mask: 3
believe: 3
extreme: 1
```

Lecture: Streaming Ingest Pipeline Hands On

How to take our streaming ingest of traffic sensor data from the previous section, and run it through a **Dataflow** pipeline to calculate average speeds and output it to **BigQuery**.



The command line reference for what we are demonstrating is below.

1. Create **BigQuery dataset** for processing pipeline output:
    - `bq mk --dataset $DEVSHELL_PROJECT_ID:demos`

```
leonf_info@cloudshell:~ (data-engineer-274519)$ bq mk --dataset $DEVSHELL_PROJECT_ID:demos
Dataset 'data-engineer-274519:demos' successfully created.
```

2. Create **Cloud Storage bucket** for Dataflow staging:
    - `gsutil mb gs://$DEVSHELL_PROJECT_ID`

```
leonf_info@cloudshell:~ (data-engineer-274519)$ gsutil mb gs://$DEVSHELL_PROJECT_ID
Creating gs://data-engineer-274519/...
```

3. Create a topic and publish messages (`sandiego example`):
    - `cd ~/googledataengineer/courses/streaming/publish`

```
leonf_info@cloudshell:~ (data-engineer-274519)$ cd ~/googledataengineer/courses/streaming/publish
leonf_info@cloudshell:~/googledataengineer/courses/streaming/publish (data-engineer-274519)$
```

    - `gcloud pubsub topics create sandiego`

```
leonf_info@cloudshell:~/googledataengineer/courses/streaming/publish (data-engineer-274519)$ gcloud pubsub topics create sandiego
Created topic [projects/data-engineer-274519/topics/sandiego].
```

- `./download_data.sh`

```
leonf_info@cloudshell:~/googledataengineer/courses/streaming/publish (data-engineer-274519)$ ./download_data.sh
Copying gs://la-gcloud-course-resources/sandiego/sensor_obs2008.csv.gz...
\ [1 files][ 34.6 MiB/ 34.6 MiB]
Operation completed over 1 objects/34.6 MiB.
```

- `sudo pip install -U google-cloud-pubsub`

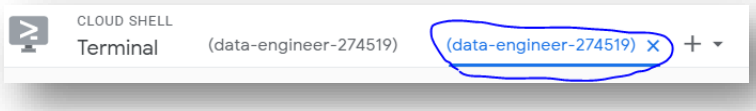
```
Installing collected packages: google-api-core, google-cloud-pubsub
  Attempting uninstall: google-api-core
    Found existing installation: google-api-core 1.17.0
    Uninstalling google-api-core-1.17.0:
      Successfully uninstalled google-api-core-1.17.0
Successfully installed google-api-core-1.16.0 google-cloud-pubsub-1.5.0
```

- `./send_sensor_data.py --speedFactor=60 --project=$DEVSHELL_PROJECT_ID`

```
leonf_info@cloudshell:~/googledataengineer/courses/streaming/publish (data-engineer-274519)$ ./send_sensor_data.py --speedFactor=60 --project=$DEVSHELL_PROJECT_ID
*******************************************************************************
Python 2 is deprecated. Upgrade to Python 3 as soon as possible.
See https://cloud.google.com/python/docs/python2-sunset

To suppress this warning, create an empty ~/.cloudshell/no-python-warning file.
The command will automatically proceed in  seconds or on any key.
*******************************************************************************
INFO: Reusing pub/sub topic sandiego
INFO: Sending sensor data from 2008-11-01 00:00:00
INFO: Publishing 477 events from 2008-11-01 00:00:00
INFO: Sleeping 5.0 seconds
```

In previous example, we create mySub1 and mySub2.
But this time, we create Dataflow pipeline instead

4. Open a new Cloud Shell tab



- Browse to the Dataflow directory and run the script to create a pipeline, passing along **our project ID**, **storage bucket**, and **Average Speeds file** to construct the pipeline.
- `cd ~/googledataengineer/courses/streaming/process/sandiego`
  (we use the 'run_oncloud.sh' to kick off the pipeline
  It will open some Java file to calculate the Average)

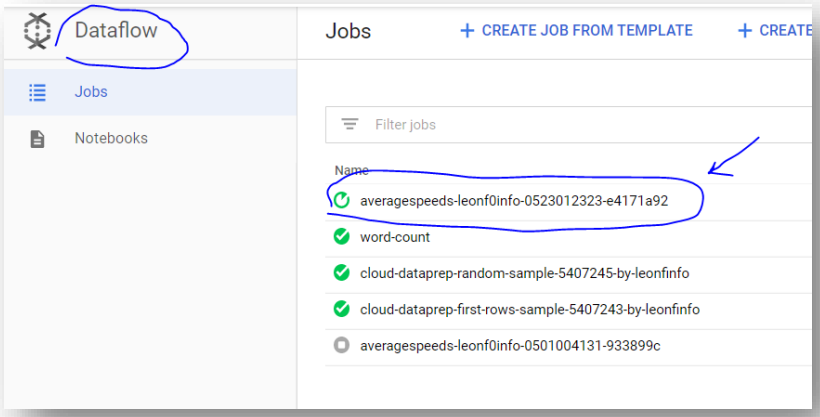
```
leonf_info@cloudshell:~ (data-engineer-274519)$ cd ~/googledataengineer/courses/streaming/process/sandiego
leonf_info@cloudshell:~/googledataengineer/courses/streaming/process/sandiego (data-engineer-274519)$ ls
create_cbt.sh  delete_cbt.sh  install_quickstart.sh  pom.xml  run_oncloud.sh  src
```

- `./run_oncloud.sh $DEVSHELL_PROJECT_ID $DEVSHELL_PROJECT_ID AverageSpeeds`

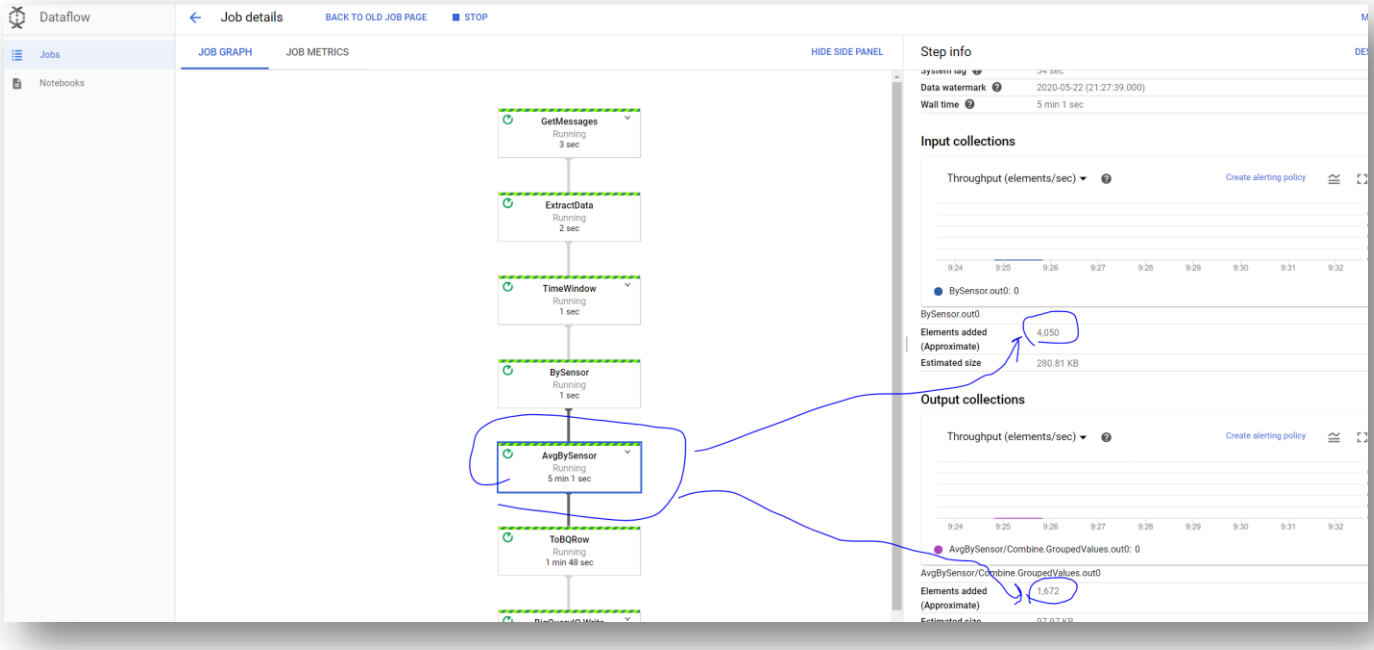
```
> gcloud dataflow jobs --project=data-engineer-274519 cancel --region=us-central1 2020-05-22_18_23_35-10788448147967333524
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  01:12 min
[INFO] Finished at: 2020-05-22T21:23:36-04:00
[INFO] ------------------------------------------------------------------------
```

Now, the pipeline is built successfully.

Go back to Dataflow to confirm it:

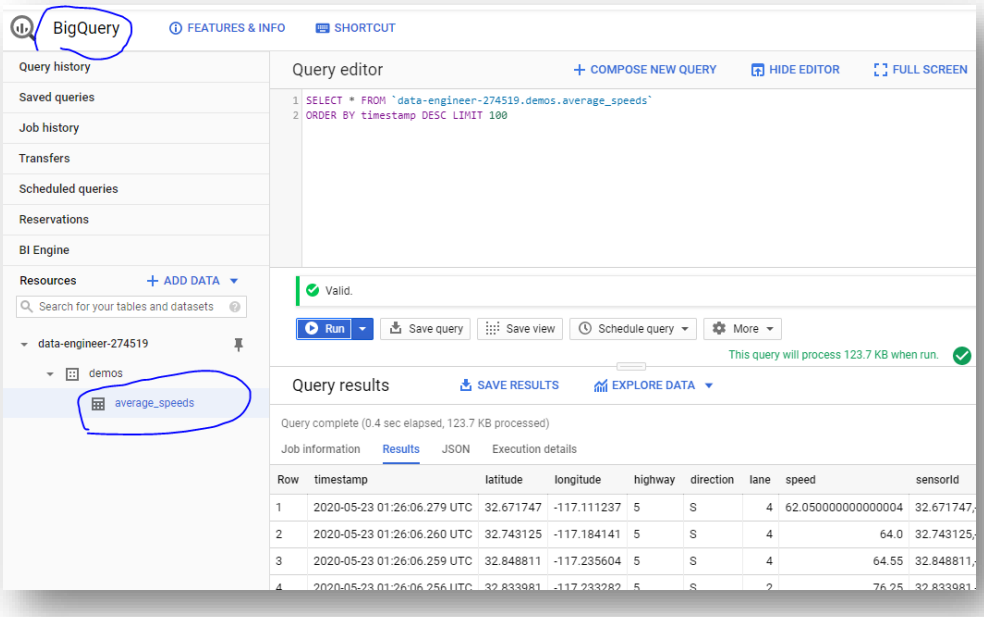


The transform is happening:

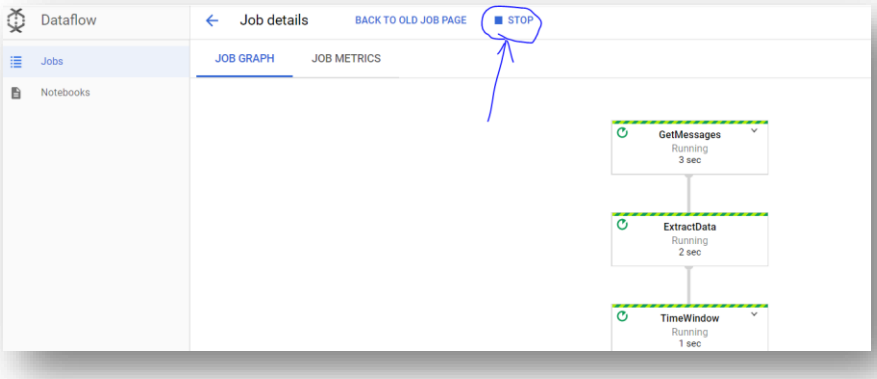


Run some SQL on BigQuery:

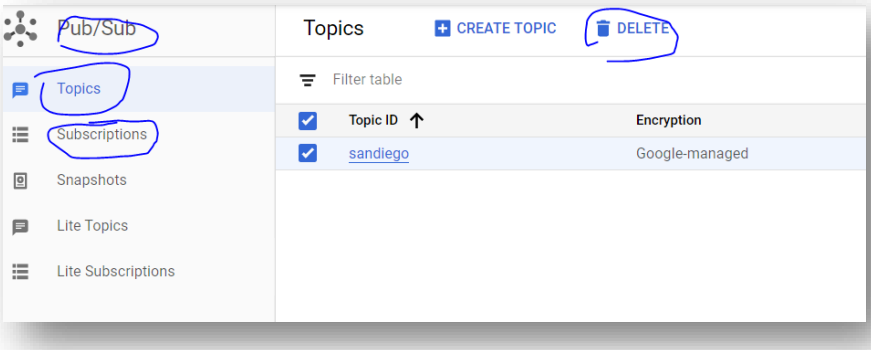
(**Even the data is in streaming buffer, we can see nothing in BigQuery, we can still write SQL query**)



Stop the job:



Go to PubSub to delete the **Job** and the **subscriptions**:



Lecture: Configure Dataproc Cluster and Submit Job – Part 1

- Creating a **Dataproc cluster** and **submitting a sample Spark job** to the cluster.
- Google-provided Dataproc initialization scripts:
  https://console.cloud.google.com/storage/browser/dataproc-initialization-actions/
- Reference for example job submission:
https://cloud.google.com/dataproc/docs/quickstarts/quickstart-console
- Creating a dataproc cluster: `gcloud dataproc clusters create [cluster_name] --zone [zone_name]`

```
leonf_info@cloudshell:~ (data-engineer-274519)$ gsutil mb -l us-central1 gs://$DEVSHELL_PROJECT_ID-data
Creating gs://data-engineer-274519-data/...
ServiceException: 409 Bucket data-engineer-274519-data already exists.
```

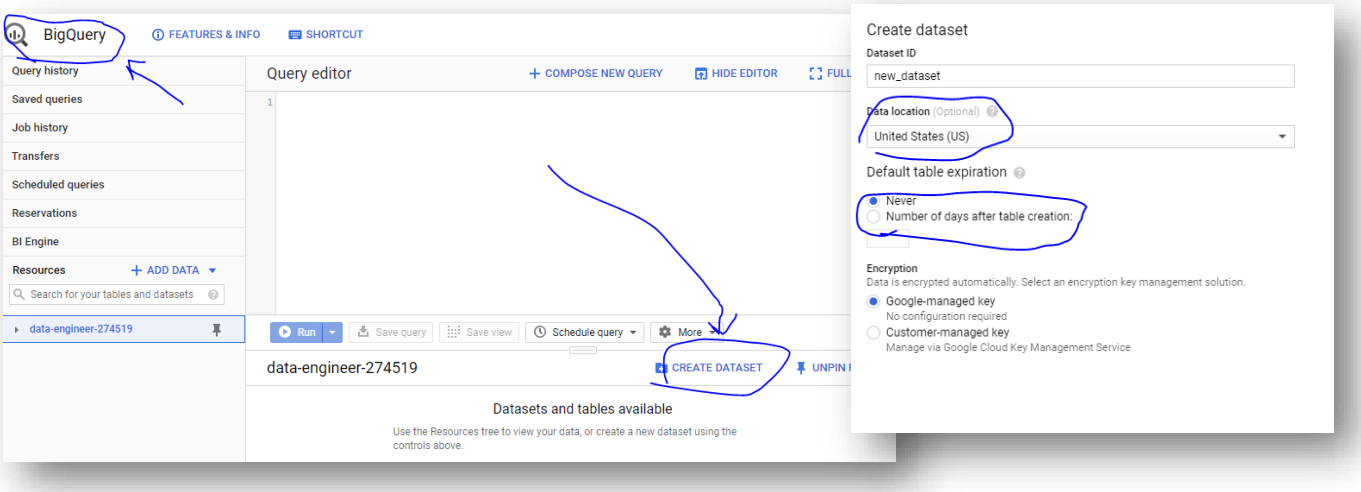
Lecture: Configure Dataproc Cluster and Submit Job – Part 2

This is the second half of our hands-on demonstration. Command line notes of actions taken in this lesson are below:

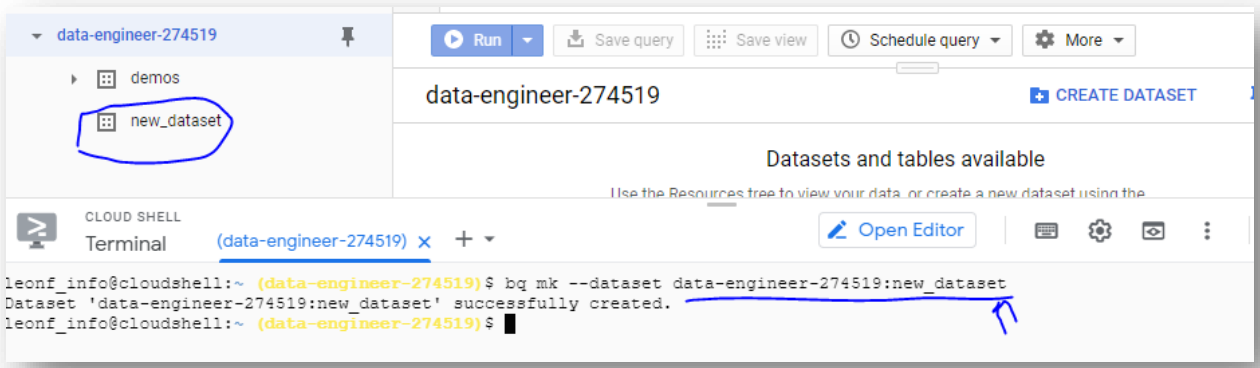
1. Updating a cluster with new workers/preemptible machines:
    - `gcloud dataproc clusters update [cluster_name] --num-workers [#] --num-preemptible-workers [#]`
    - SOCKS proxy configuration
2. From the local machine, SSH to the master to enable port forwarding:
    - `gcloud compute ssh master-host-name --project=project-id --zone=master-host-zone -- -D 1080 -N`
3. Open new terminal window and launch the web browser with parameters (varies by OS/browser):
    - `"/Applications/Google Chrome.app/Contents/MacOS/Google Chrome" --proxy-server="socks5://localhost:1080" --host-resolver-rules="MAP * 0.0.0.0 , EXCLUDE localhost" --user-data-dir=/tmp/cluster1-m`
        - Browse to http://[master]:port
        - 8088 - Hadoop
        - 9870 - HDFS
        - Using Cloud Shell (must use for each port)
    - `gcloud compute ssh master-host-name --project=project-id --zone master-host-zone -- -4 -N -L port1:master-host-name:port2`
4. Use Web Preview to choose port (8088/9870).

Lecture: Interacting with BigQuery

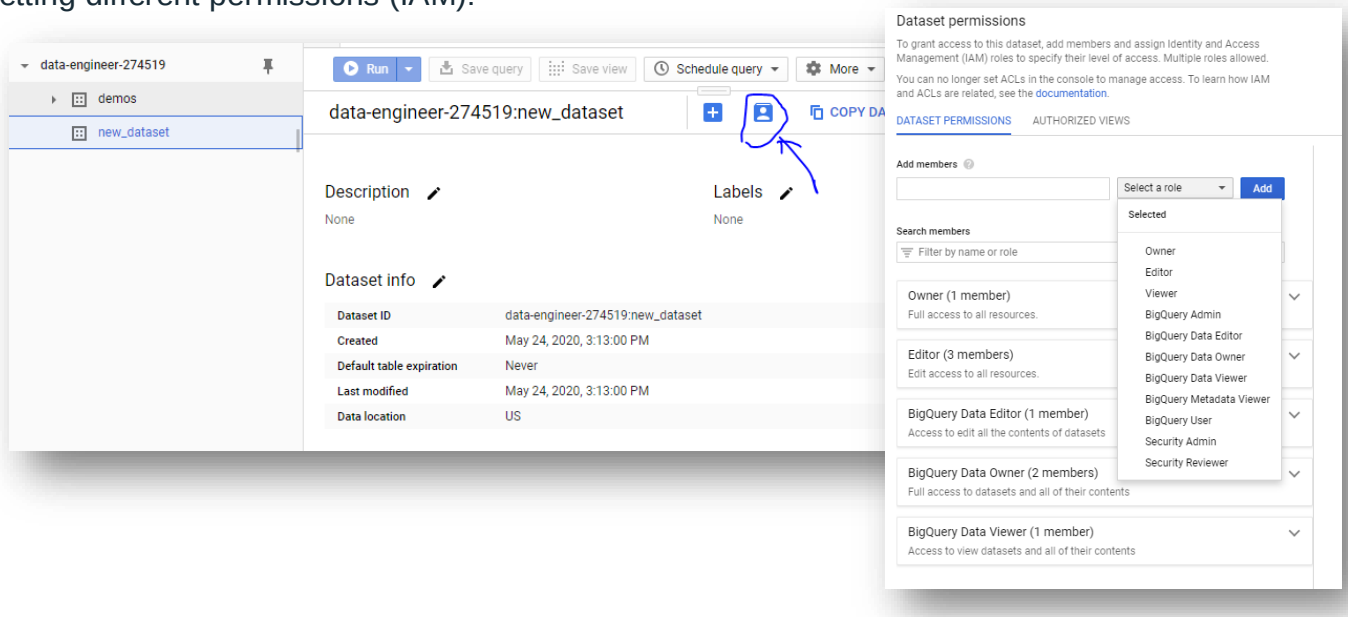
Create a dataset:



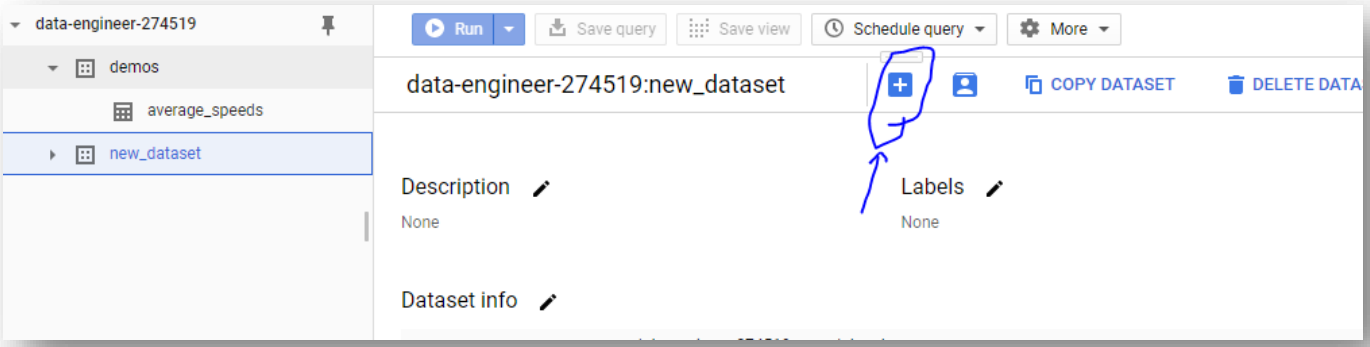
Or **using bq command**:



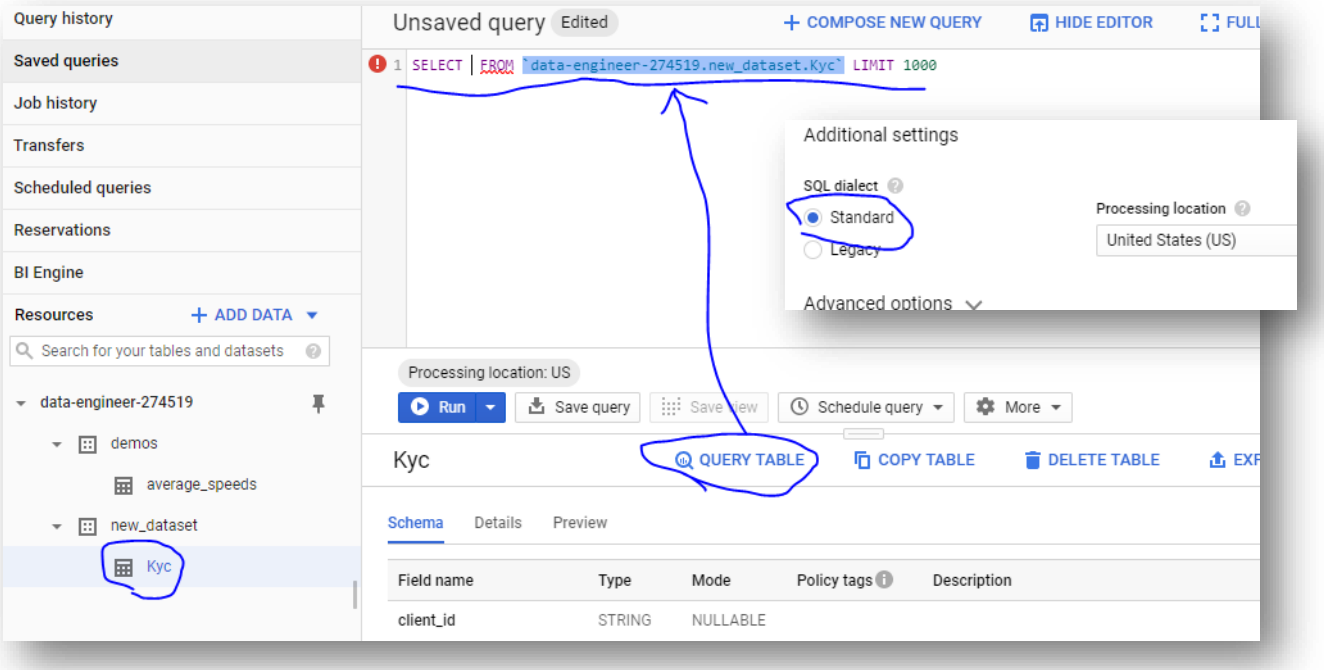
Setting different permissions (IAM):

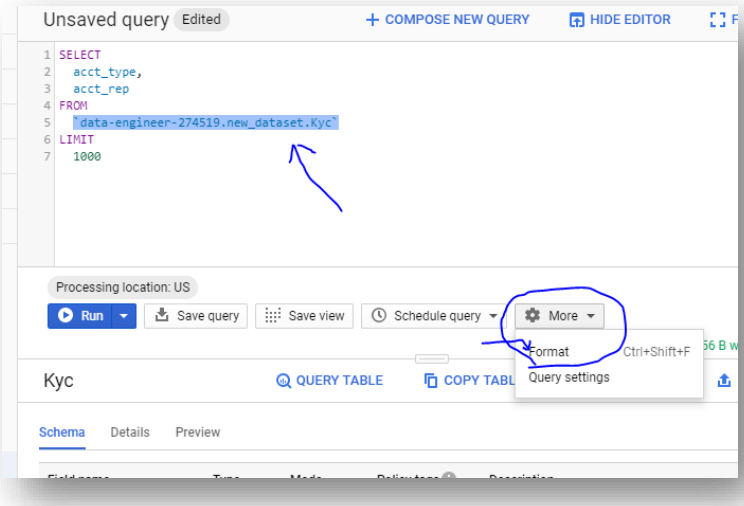


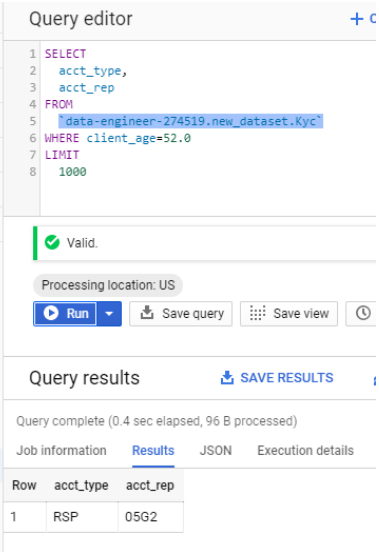
Add public dataset:

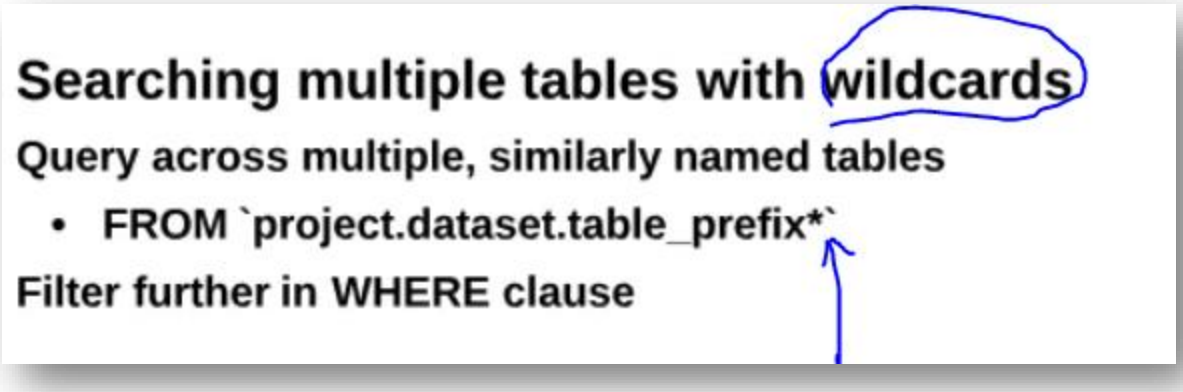


Write Query (Standard SQL mode):





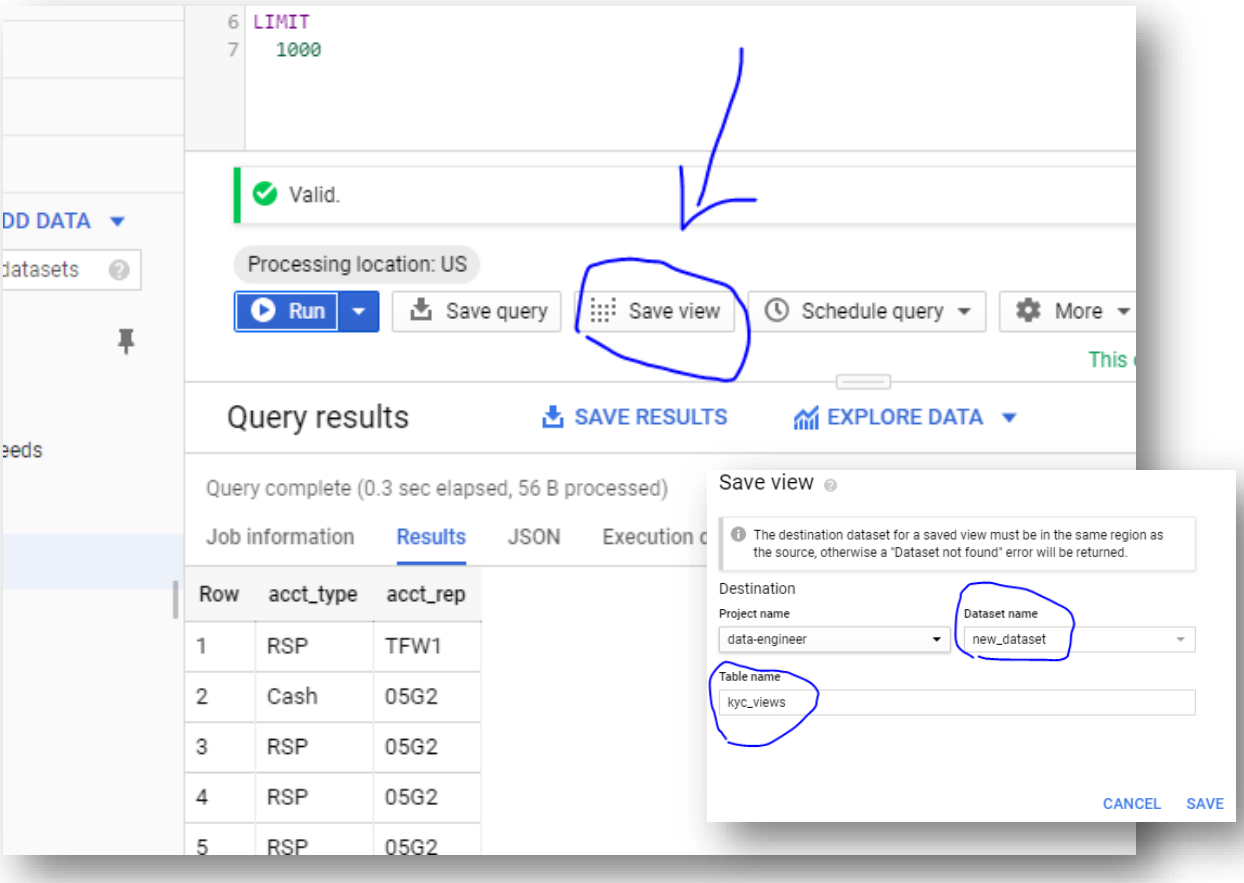




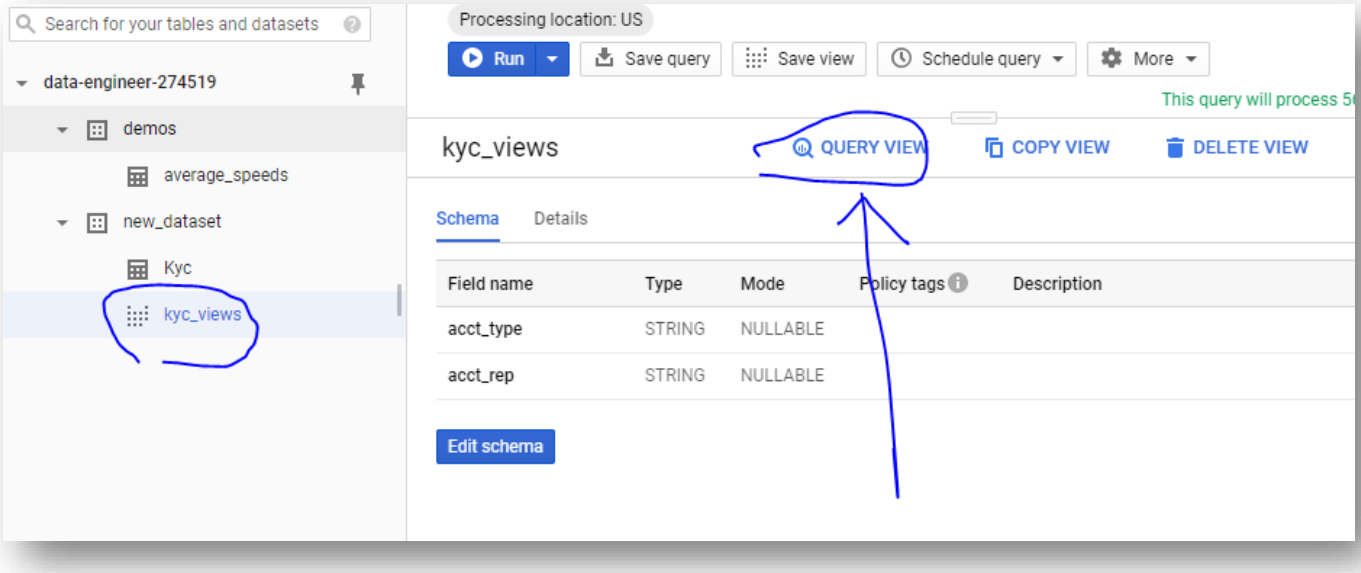
Query **using bq command** line format:

```
leonf_info@cloudshell:~ (data-engineer-274519)$ bq query --use_legacy_sql=false 'select acct_type, acct_rep from `data-engineer-274519.n
ew_dataset.Kyc`'

Waiting on bqjob_r38e00fa8ac7a651f_00000172483f51e2_1 ... (0s) Current status: DONE
+-----------+----------+
| acct_type | acct_rep |
+-----------+----------+
| RSP       | TFW1     |
| Cash      | 05G2     |
| RSP       | 05G2     |
| RSP       | 05G2     |
| RSP       | 05G2     |
+-----------+----------+
```

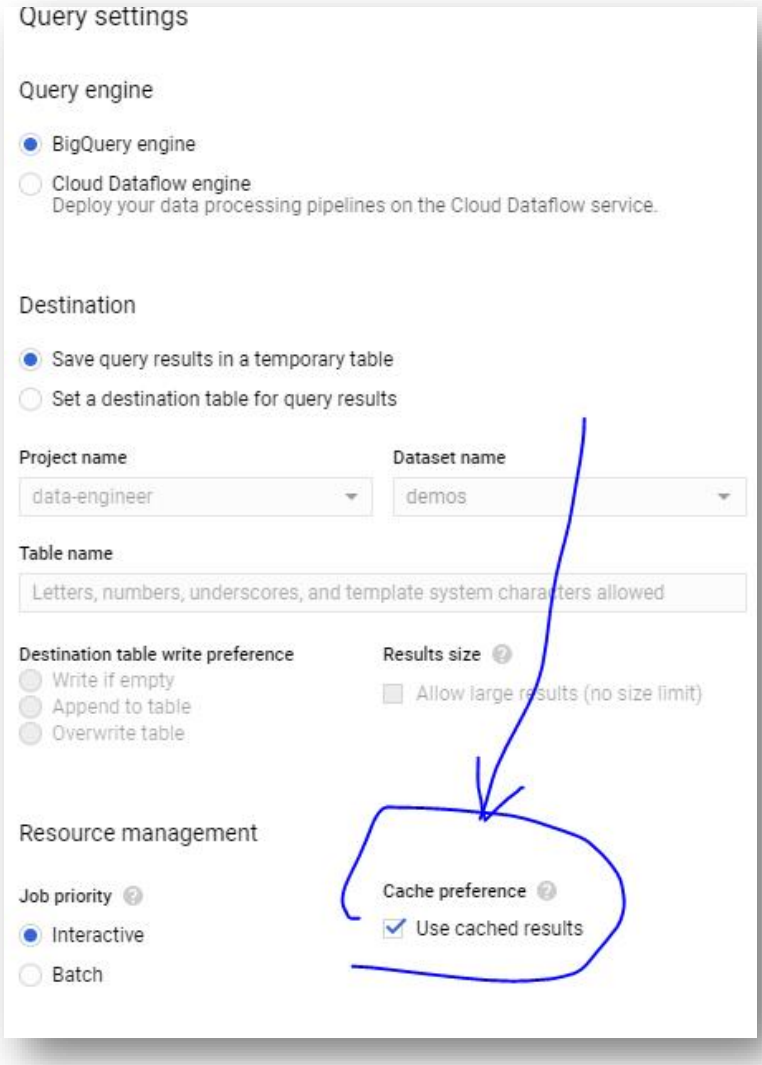
Create a **VIEW**:



If we query from the VIEW, it would be more much faster:



Cache a Query (per user only, somebody else can not use it at the same time):

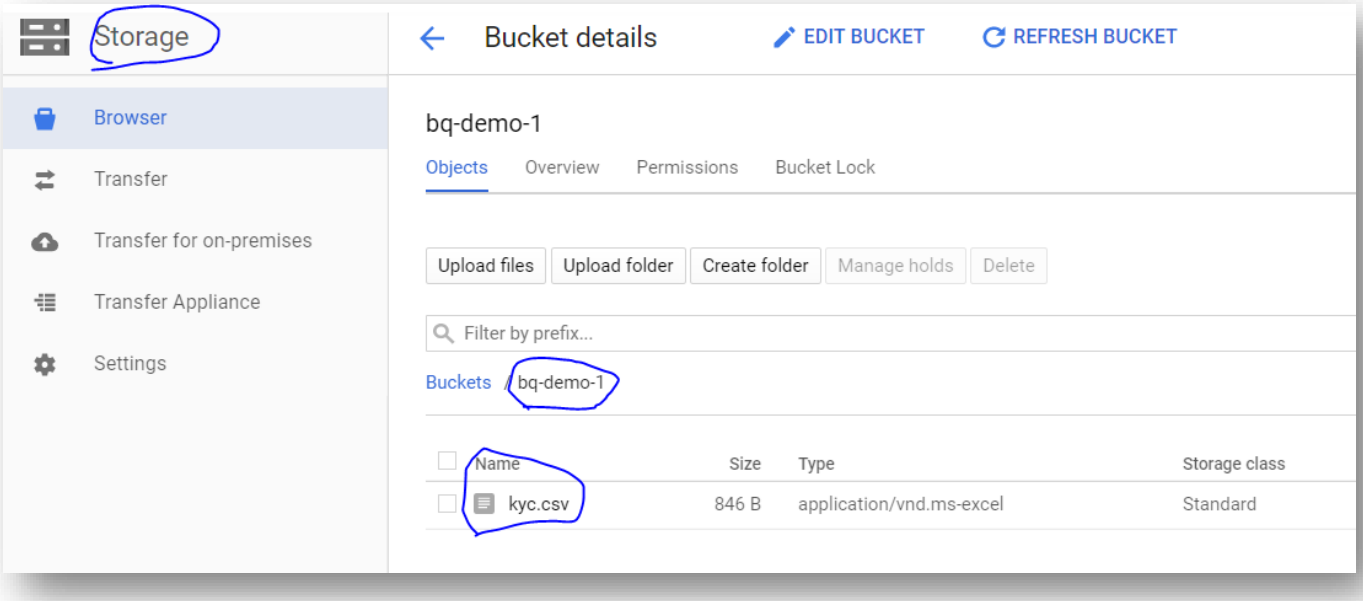


User Defined Functions (UDF)

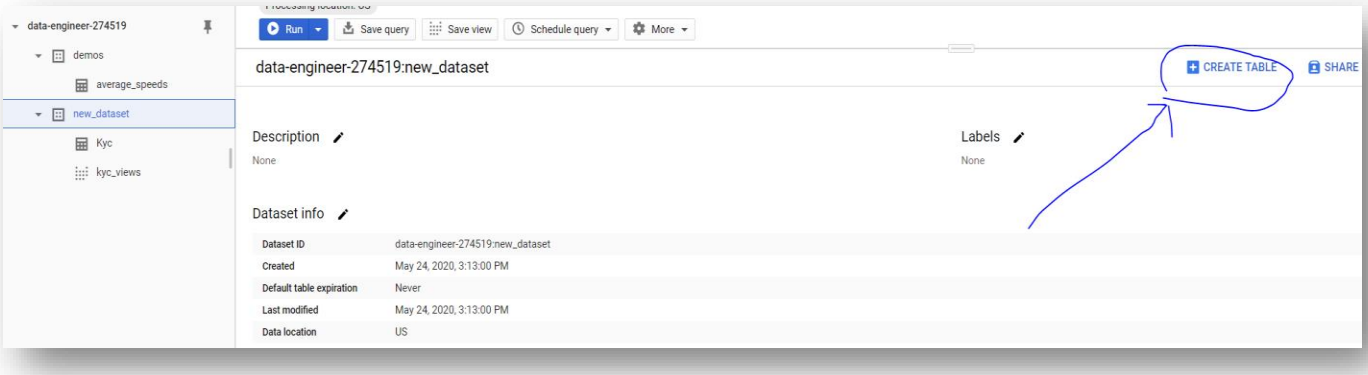
- Combine SQL code with JavaScript/SQL functions
- Combine SQL queries with programming logic
- Allow much more complex operations (loops, complex conditionals)
- WebUI only usable with Legacy SQL

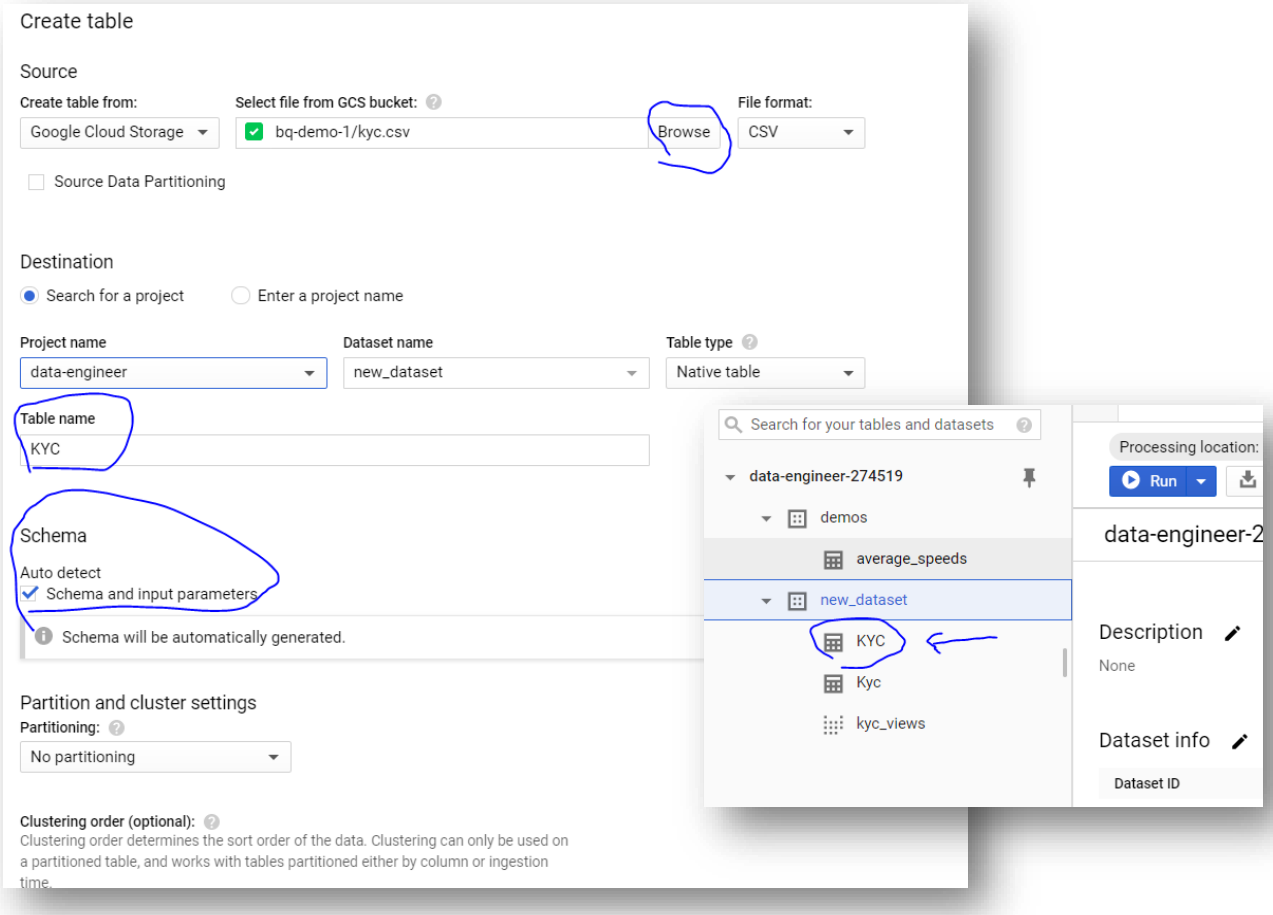
Lecture: Load and Export Data

Prepare data in **Storage Bucket**:



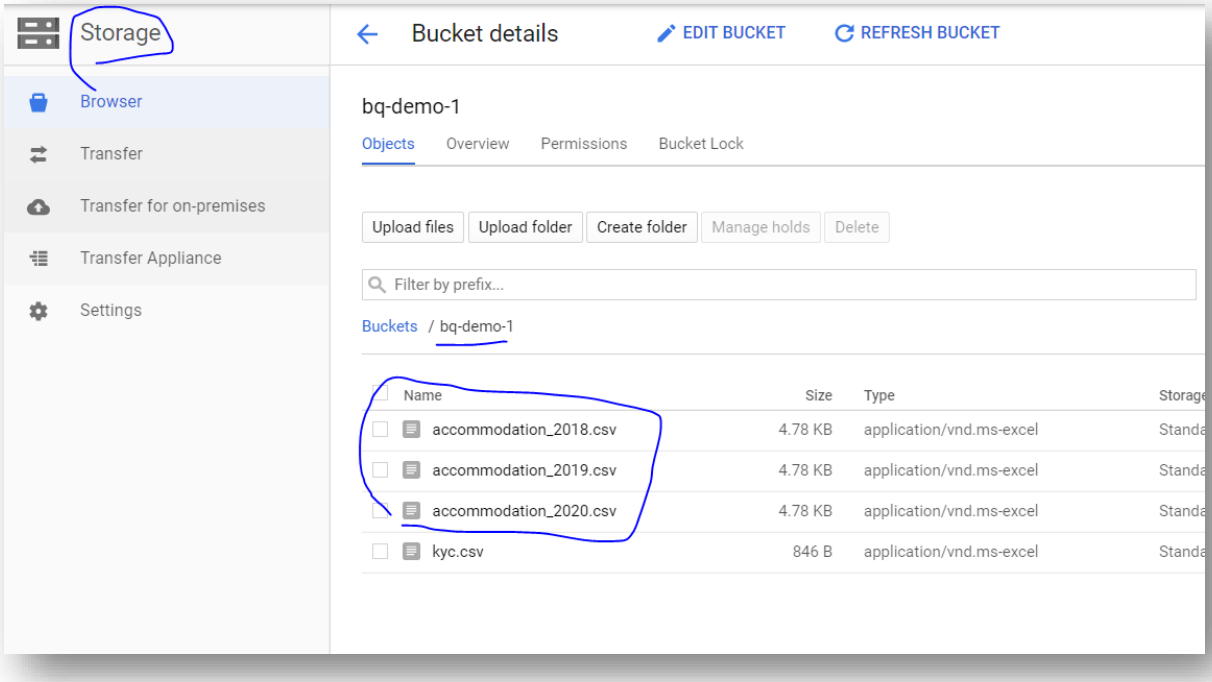
Create a new table in **BigQuery**:



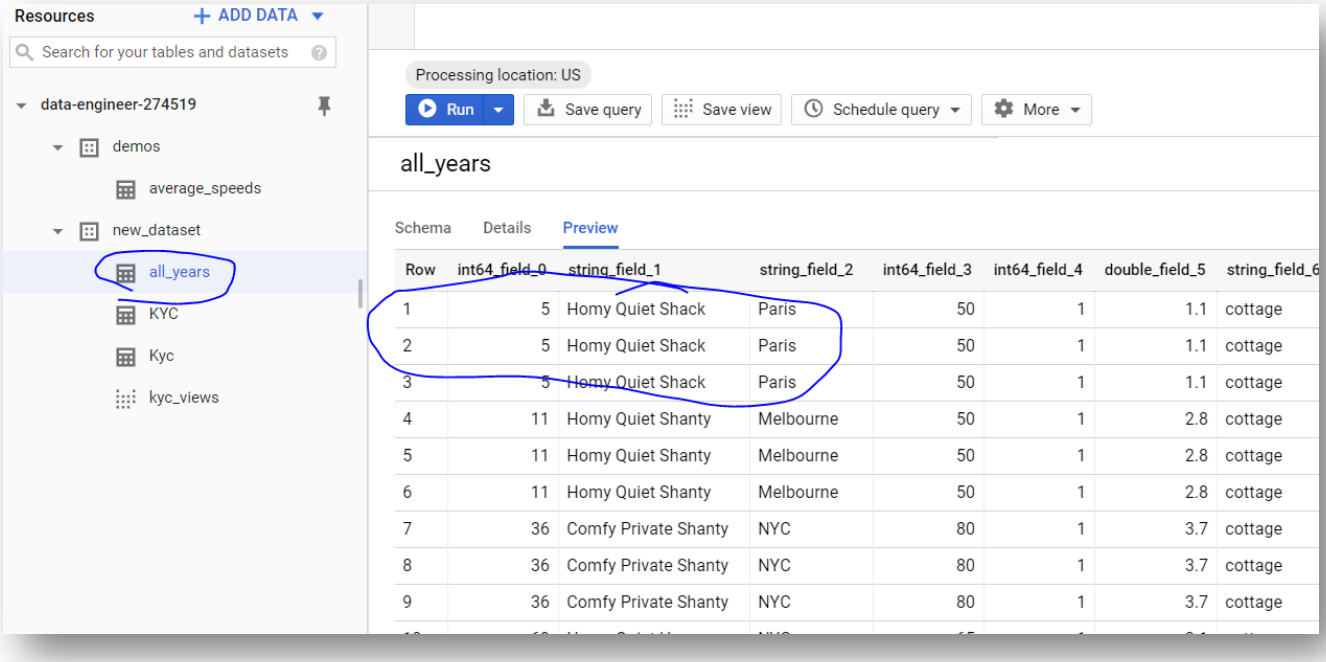


Lecture: Load and Export Data (Using **Wildcard**)

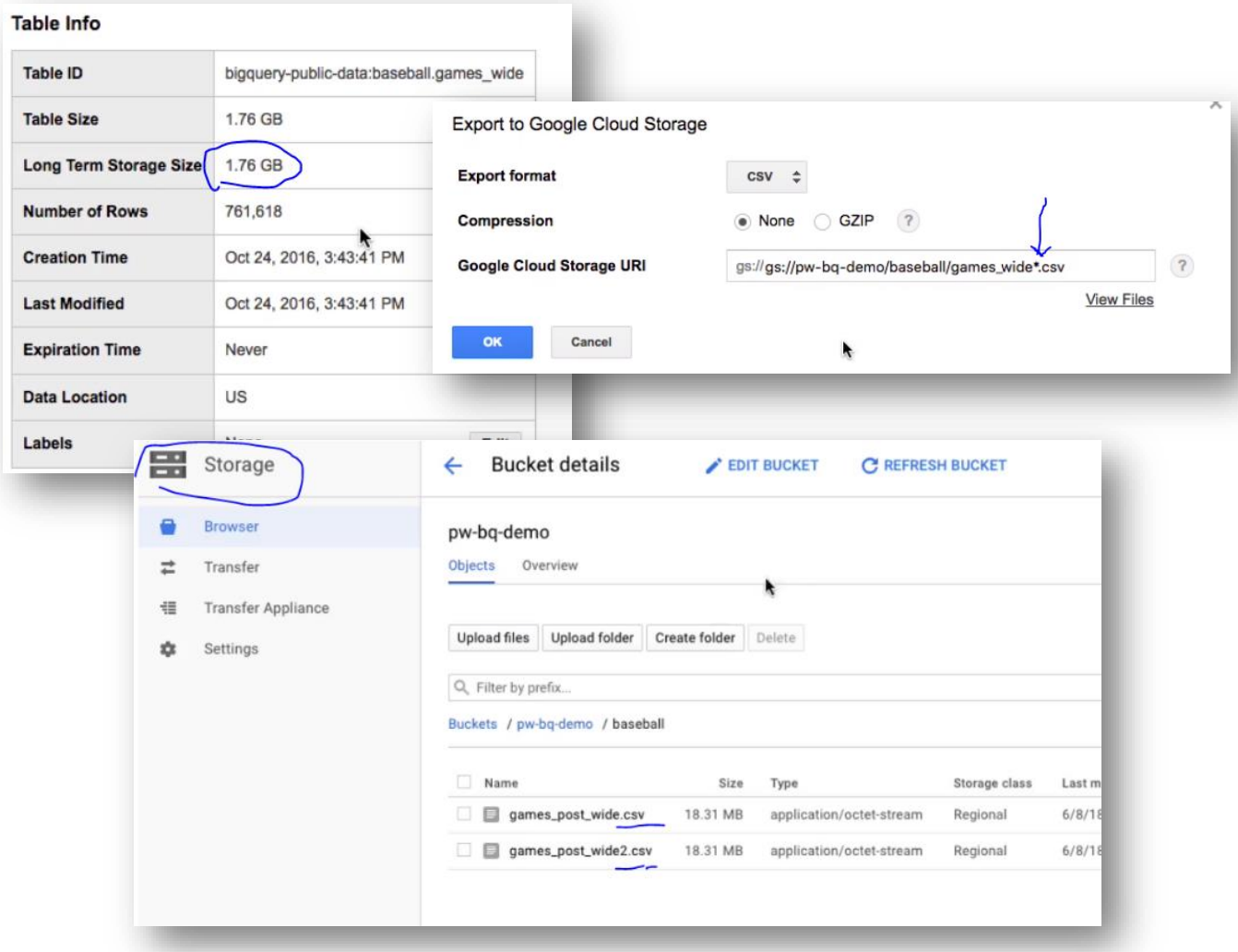
Prepare data in **Storage Bucket**:



Create a new table in **BigQuery** and load multiple files into **ONE** table:



- When Exporting, the maximum size is 1GB. If exporting is > 1GB, then use wildcard



Lecture: Load and Export Data (Using **Command Line**)

Loading Data:

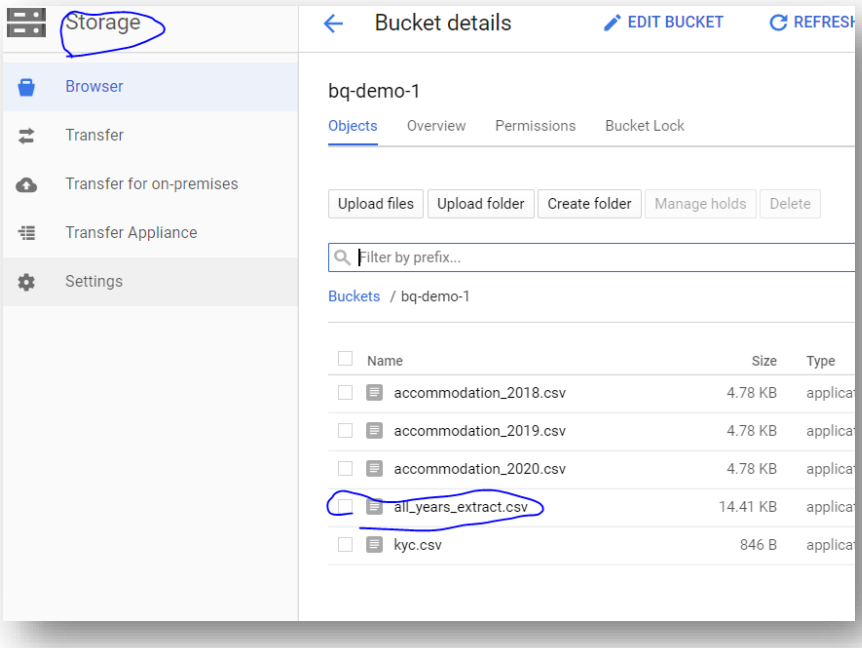
```
leonf_info@cloudshell:~ (data-engineer-274519)$ bq load --source_format=CSV new_dataset.kyc_from_command_line gs://bq-demo-1/accommodation_2018.csv
BigQuery error in load operation: Error processing job 'data-engineer-274519:bqjob_r422ea8856ce64e31_000001724d03d644_1': No schema specified on job or table.
```

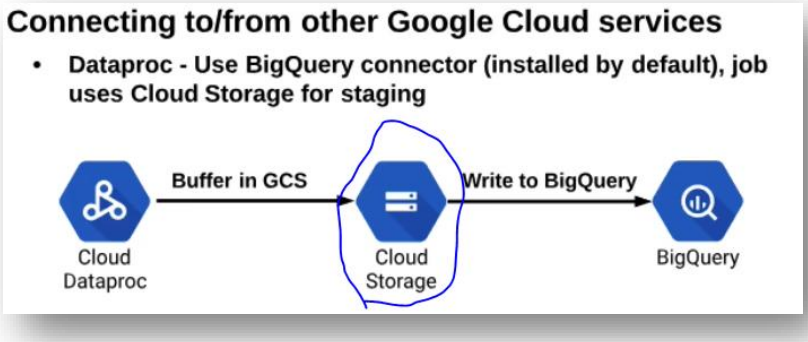
We need Schema:

```
leonf_info@cloudshell:~ (data-engineer-274519)$ bq load --source_format=CSV new_dataset.accom_from_command_line gs://bq-demo-1/accommoda
tion_2018.csv Index:INTEGER,Name:STRING,Location:STRING,Size:INTEGER,Room:INTEGER,Price:FLOAT,Type:STRING
Waiting on bqjob_rdb563f4599a2fef_000001724d0782ef_1 ... (0s) Current status: DONE
```

Exporting Data:

```
leonf_info@cloudshell:~ (data-engineer-274519)$ bq extract 'data-engineer-274519:new_dataset.all_years' gs://bq-demo-1/all_years_extract.csv
Waiting on bqjob_r1a6d629662014360_000001724d159bee_1 ... (0s) Current status: DONE
```





Lecture: Streaming Insert Example

Revisit our streaming Dataflow/PubSub pipeline and take a closer look at the end result, which is a streaming insert into BigQuery. For easy setup, the below commands and scripts will quickly create your pipeline so we can focus on BigQuery, as well as cleanup when finished.
Quick setup:

```
cd
gsutil cp -r gs://gcp-course-exercise-scripts/data-engineer/* .
```

```
leonf_info@cloudshell:~ (data-engineer-274519)$ gsutil cp -r gs://gcp-course-exercise-scripts/data-engineer/* .
Copying gs://gcp-course-exercise-scripts/data-engineer/streaming-cleanup.sh...
Copying gs://gcp-course-exercise-scripts/data-engineer/streaming-insert.sh...
- [2 files][  1.4 KiB/  1.4 KiB]
==> NOTE: You are performing a sequence of gsutil operations that may
run significantly faster if you instead use gsutil -m cp ... Please
see the -m section under "gsutil help options" for further information
about when gsutil -m can be advantageous.

Copying gs://gcp-course-exercise-scripts/data-engineer/ai-platform/1-local-train.sh...
Copying gs://gcp-course-exercise-scripts/data-engineer/ai-platform/2-single-instance-trainer.sh...
Copying gs://gcp-course-exercise-scripts/data-engineer/ai-platform/3-distributed-trainer.sh...
Copying gs://gcp-course-exercise-scripts/data-engineer/ai-platform/4-hyperparameter-distributed.sh...
Copying gs://gcp-course-exercise-scripts/data-engineer/ai-platform/5-create-model-send-predictions.sh...
Copying gs://gcp-course-exercise-scripts/data-engineer/ai-platform/AI Platform.pdf...
Copying gs://gcp-course-exercise-scripts/data-engineer/ai-platform/AI-Platform-Commands.md...
Copying gs://gcp-course-exercise-scripts/data-engineer/cloud-ml_engine/1-local-train.sh...
Copying gs://gcp-course-exercise-scripts/data-engineer/cloud-ml_engine/2-local-train-dist.sh...
Copying gs://gcp-course-exercise-scripts/data-engineer/cloud-ml_engine/3-single-instance-trainer.sh...
Copying gs://gcp-course-exercise-scripts/data-engineer/cloud-ml_engine/4-distributed-trainer.sh...
Copying gs://gcp-course-exercise-scripts/data-engineer/cloud-ml_engine/5-hyperparameter-distributed.sh...
Copying gs://gcp-course-exercise-scripts/data-engineer/cloud-ml_engine/6-create-model-send-predictions.sh...
Copying gs://gcp-course-exercise-scripts/data-engineer/cloud-ml_engine/7-superfun-script.sh...
\ [16 files][133.5 KiB/133.5 KiB]
==> NOTE: You are performing a sequence of gsutil operations that may
run significantly faster if you instead use gsutil -m cp ... Please
see the -m section under "gsutil help options" for further information
about when gsutil -m can be advantageous.


Operation completed over 16 objects/133.5 KiB.
leonf_info@cloudshell:~ (data-engineer-274519)$ ls
ai-platform             googledataengineer  python-docs-samples    requests.json         training-data-analyst
cloud-ml_engine         master.zip          qie.jpg                streaming-cleanup.sh
cloudml-samples-master  output.txt          README-cloudshell.txt  streaming-insert.sh
```

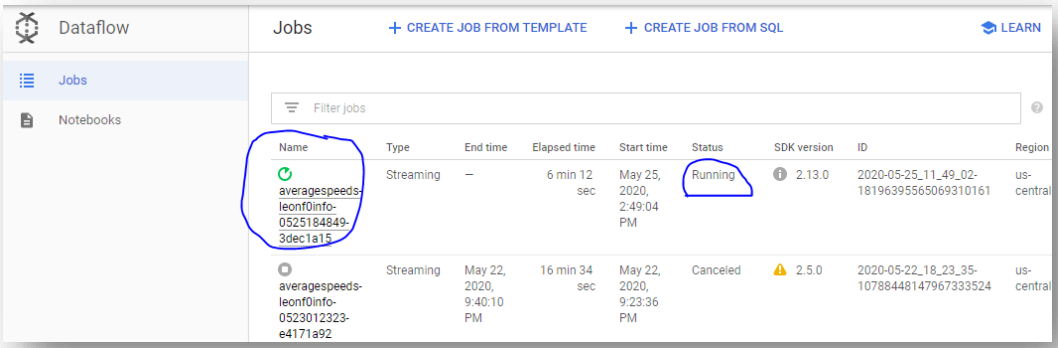
```
leonf_info@cloudshell:~ (data-engineer-274519)$ vim streaming-insert.sh
```

```
#!/bin/bash

# Enable Dataflow and Pub/Sub API's if not already enabled

gcloud services enable dataflow.googleapis.com
gcloud services enable pubsub.googleapis.com

# Go to home directory and clone GitHub data

cd ~
git clone https://github.com/GoogleCloudPlatform/training-data-analyst

# Create BigQuery dataset if it does not already exist
echo **Creating BigQuery dataset**
bq mk --dataset $DEVSHELL_PROJECT_ID:demos

# Create Cloud Storage Bucket for Dataflow staging
echo **Creating staging bucket**
gsutil mb gs://$DEVSHELL_PROJECT_ID

# Create Pub/Sub topic named San Diego
echo **Creating Pub/Sub topic**
gcloud pubsub topics create sandiego

# Go to Dataflow job directory and run script to subscribe to topic and create streaming pipeline
echo **Creating Dataflow streaming pipeline**
cd ~/training-data-analyst/courses/streaming/process/sandiego
./run_oncloud.sh $DEVSHELL_PROJECT_ID $DEVSHELL_PROJECT_ID AverageSpeeds

# Install python dependencies for Pub/Sub

cd ~/training-data-analyst/courses/streaming/publish
sudo pip install -U google-cloud-pubsub

# Download sample sensor data and begin simulated stream
echo **Starting sensor stream**
./download_data.sh
./send_sensor_data.py --speedFactor=60 --project=$DEVSHELL_PROJECT_I
```

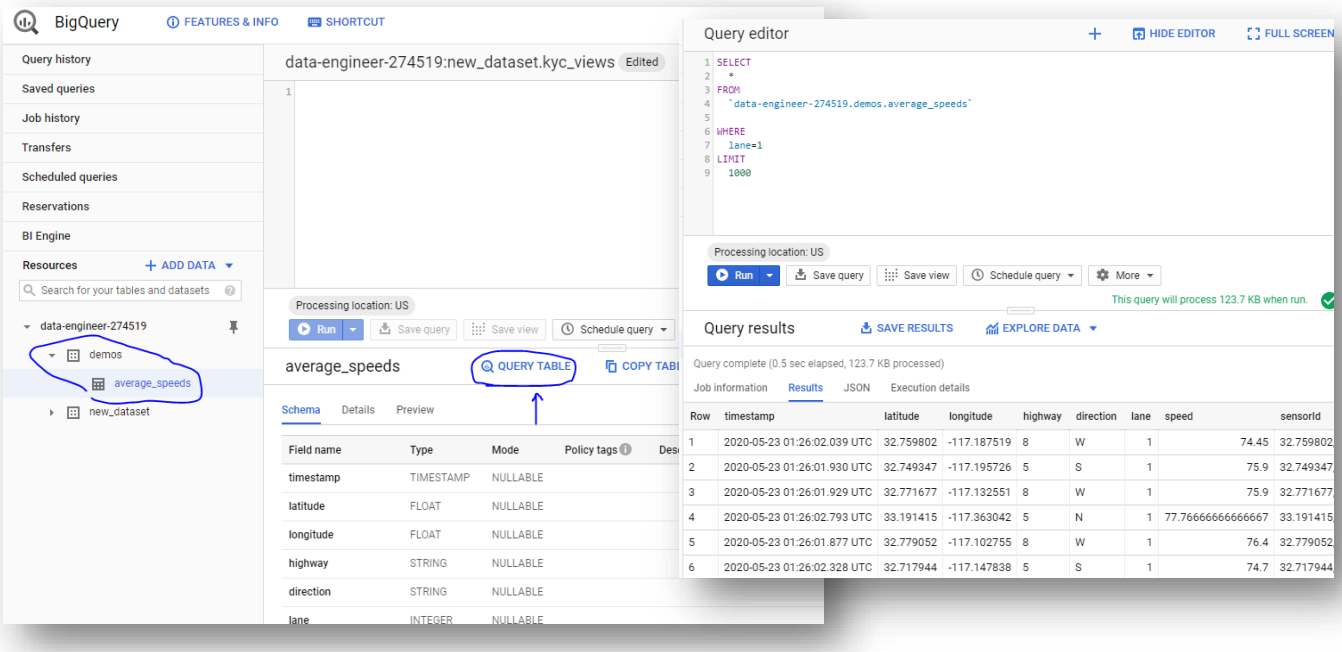
```
bash streaming-insert.sh
```

```
leonf_info@cloudshell:~ (data-engineer-274519)$ bash streaming-insert.sh
fatal: destination path 'training-data-analyst' already exists and is not an empty directory.
**Creating BigQuery dataset**
BigQuery error in mk operation: Dataset 'data-engineer-274519:demos' already exists.
**Creating staging bucket**
Creating gs://data-engineer-274519/...
ServiceException: 409 Bucket data-engineer-274519 already exists.
**Creating Pub/Sub topic**
Created topic [projects/data-engineer-274519/topics/sandiego].
**Creating Dataflow streaming pipeline**
Launching com.google.cloud.training.dataanalyst.sandiego.AverageSpeeds project=data-engineer-274519 bucket=data-engineer-274519
[INFO] Error stacktraces are turned on.
[INFO] Scanning for projects...
[INFO]
[INFO] ------< com.google.cloud.training.dataanalyst.sandiego:sandiego >-------
[INFO] Building sandiego [1.0.0,2.0.0]
[INFO] --------------------------------[ jar ]---------------------
Downloading from apache.snapshots: https://repository.apache.org/c
Downloading from central: https://repo.maven.apache.org/maven2/io/
Downloaded from central: https://repo.maven.apache.org/maven2/io/g
Downloading from apache.snapshots: https://repository.apache.org/c
INFO: Reusing pub/sub topic sandiego
INFO: Sending sensor data from 2008-11-01 00:00:00
INFO: Publishing 477 events from 2008-11-01 00:00:00
INFO: Sleeping 5.0 seconds
INFO: Publishing 477 events from 2008-11-01 00:05:00
INFO: Sleeping 5.0 seconds
INFO: Publishing 477 events from 2008-11-01 00:10:00
INFO: Sleeping 5.0 seconds
INFO: Publishing 477 events from 2008-11-01 00:15:00
INFO: Sleeping 5.0 seconds
```

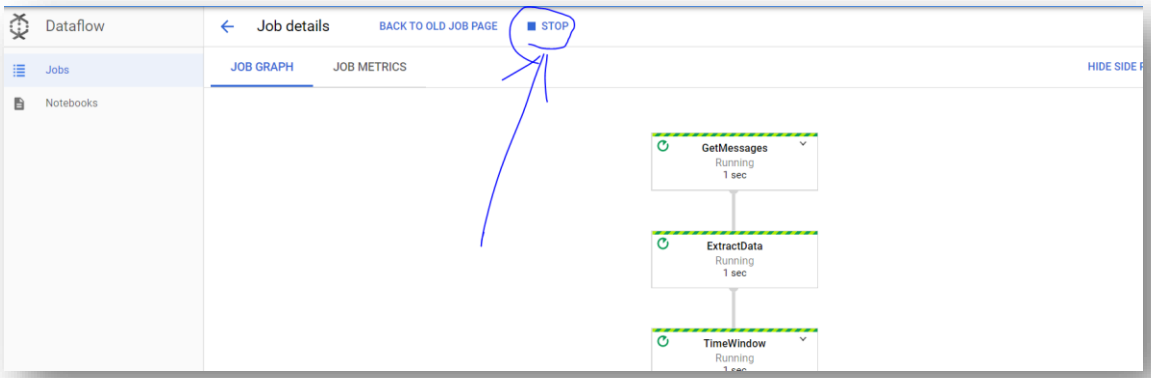
Go to **Dataflow**, can see the streaming pipeline:



Now jump to **BigQuery**:



Clean up:
Manually stop the Dataflow **job**:



Run **streaming-cleanup**.sh:

```
leonf_info@cloudshell:~ (data-engineer-274519)$ ls
ai-platform              googledataengineer  python-docs-samples    requests.json          training-data-analyst
cloud-ml_engine          master.zip          qie.jpg                streaming-cleanup.sh
cloudml-samples-master   output.txt          README-cloudshell.txt  streaming-insert.sh
```

```
leonf_info@cloudshell:~ (data-engineer-274519)$ vim streaming_cleanup.sh
```

```
#!/bin/bash

gcloud pubsub topics delete sandiego
gsutil rm -rf gs://$DEVSHELL_PROJECT_ID
bq rm -r -f -d $DEVSHELL_PROJECT_ID:demos
~
~
```

`bash streaming-cleanup.sh`

```
leonf_info@cloudshell:~ (data-engineer-274519)$ bash streaming-cleanup.sh
Deleted topic [projects/data-engineer-274519/topics/sandiego].
Removing gs://data-engineer-274519/google-cloud-dataproc-metainfo/0c0b8cb5-6018-4839-a0dc-73b022d5598e/cluster-1-m/dataproc-initializati
on-script-0_output#1590342723259981...
Removing gs://data-engineer-274519/google-cloud-dataproc-metainfo/0c0b8cb5-6018-4839-a0dc-73b022d5598e/cluster-1-m/dataproc-startup-scri
pt_output#1590342718630439...
Removing gs://data-engineer-274519/google-cloud-dataproc-metainfo/0c0b8cb5-6018-4839-a0dc-73b022d5598e/cluster-1-w-0/dataproc-initializa
tion-script-0_output#1590342775630606...
Removing gs://data-engineer-274519/google-cloud-dataproc-metainfo/0c0b8cb5-6018-4839-a0dc-73b022d5598e/cluster-1-w-0/dataproc-startup-sc
ript_output#1590342643266592...
/ [4 objects]
==> NOTE: You are performing a sequence of gsutil operations that may
run significantly faster if you instead use gsutil -m rm ... Please
see the -m section under "gsutil help options" for further information
about when gsutil -m can be advantageous.

Removing gs://data-engineer-274519/google-cloud-dataproc-metainfo/0c0b8cb5-6018-4839-a0dc-73b022d5598e/cluster-1-w-1/dataproc-initializa
```

Lecture: AI Platform Hands On Part 1

- Go over the process of training a pre-packaged machine learnng model on AI Platform, deploying the trained model, and running predictions against it.
- All commands used in these two demonstrations are below.
- If you want a set of scripts to automate much of the process, the **scripts** and a **PDF file** of the commands we used can be found at the following public link: https://console.cloud.google.com/storage/browser/gcp-course-exercise-scripts/data-engineer/ai-platform
- If you want to download all scripts and files in a terminal environment, use the below command:

```
gsutil cp gs://gcp-course-exercise-scripts/data-engineer/ai-platform/* .
```

Set region environment variable

```
REGION=us-central1
```

```
leonf_info@cloudshell:~ (data-engineer-274519)$ REGION=us-central1
```

Download and unzip ML github data for demo

```
wget https://github.com/GoogleCloudPlatform/cloudml-samples/archive/master.zip
```

```
leonf_info@cloudshell:~ (data-engineer-274519)$ wget https://github.com/GoogleCloudPlatform/cloudml-samples/archive/master.zip
--2020-05-25 21:32:45--  https://github.com/GoogleCloudPlatform/cloudml-samples/archive/master.zip
Resolving github.com (github.com)... 140.82.112.3
Connecting to github.com (github.com)|140.82.112.3|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://codeload.github.com/GoogleCloudPlatform/cloudml-samples/zip/master [following]
--2020-05-25 21:32:45--  https://codeload.github.com/GoogleCloudPlatform/cloudml-samples/zip/master
Resolving codeload.github.com (codeload.github.com)... 140.82.113.10
Connecting to codeload.github.com (codeload.github.com)|140.82.113.10|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [application/zip]
Saving to: 'master.zip.1'

master.zip.1                          [     <=>                                       ]  24.18M  7.63MB/s    in 3.2s

2020-05-25 21:32:48 (7.63 MB/s) - 'master.zip.1' saved [25353805]
```

```
unzip master.zip
```

```
leonf_info@cloudshell:~ (data-engineer-274519)$ unzip master.zip
Archive:  master.zip
54c14e0edc940a007cf045977c9e0a045a15d419
replace cloudml-samples-master/.github/ISSUE_TEMPLATE/bug_report.md? [y]es, [n]o, [A]ll, [N]one, [r]ename: y
```

```
leonf_info@cloudshell:~ (data-engineer-274519)$ ls
ai-platform               googledataengineer  output.txt           README-cloudshell.txt  streaming-insert.sh
cloud-ml-engine           master.zip          python-docs-samples  requests.json          training-data-analyst
cloudml-samples-master    master.zip.1        qie.jpg              streaming-cleanup.sh
```

Navigate to the cloudml-samples-master > census > estimator directory.

ALL Commands must be run from this directory

```
cd ~/cloudml-samples-master/census/estimator
```

```
leonf_info@cloudshell:~ (data-engineer-274519)$ cd ~/cloudml-samples-master/census/estimator
leonf_info@cloudshell:~/cloudml-samples-master/census/estimator (data-engineer-274519)$
```

Develop and validate trainer on local machine

Get training data from public GCS bucket

1. Create a directory called 'data'
2. Download the dataset

```
mkdir data
```

```
gsutil -m cp gs://cloud-samples-data/ml-engine/census/data/* data/
```

```
leonf_info@cloudshell:~/cloudml-samples-master/census/estimator (data-engineer-274519)$ gsutil -m cp gs://cloud-samples-data/ml-engine/c
ensus/data/* data/
Copying gs://cloud-samples-data/ml-engine/census/data/census.test.csv...
Copying gs://cloud-samples-data/ml-engine/census/data/test.json...
Copying gs://cloud-samples-data/ml-engine/census/data/adult.data.csv...
Copying gs://cloud-samples-data/ml-engine/census/data/census.train.csv...
Copying gs://cloud-samples-data/ml-engine/census/data/test.csv...
Copying gs://cloud-samples-data/ml-engine/census/data/adult.test.csv...
\ [6/6 files][ 10.7 MiB/ 10.7 MiB] 100% Done
Operation completed over 6 objects/10.7 MiB.
```

```
leonf_info@cloudshell:~/cloudml-samples-master/census/estimator (data-engineer-274519)$ cd data
leonf_info@cloudshell:~/cloudml-samples-master/census/estimator/data (data-engineer-274519)$ ls
adult.data.csv  adult.test.csv  census.test.csv  census.train.csv  test.csv  test.json
```

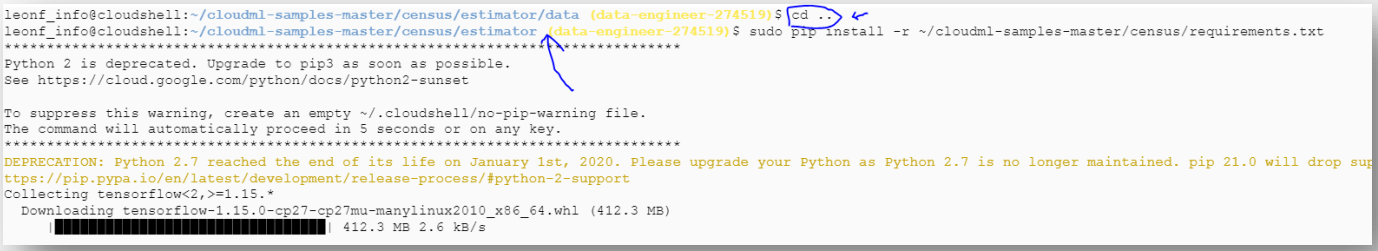
Set path variables for local file paths

These will change later when we use AI Platform

```
TRAIN_DATA=$(pwd)/data/adult.data.csv
EVAL_DATA=$(pwd)/data/adult.test.csv
```

Run sample requirements.txt to ensure we're using same version of TF as sample

```
sudo pip install -r ~/cloudml-samples-master/census/requirements.txt
```



Run a local trainer

Specify output directory, set as variable

```
MODEL_DIR=output
```

Best practice is to delete contents of output directory in case

data remains from previous training run

```
rm -rf $MODEL_DIR/*
```

Run local training using gcloud (not on AI Platform)

```
gcloud ai-platform local train --module-name trainer.task --package-path trainer/ \
--job-dir $MODEL_DIR -- --train-files $TRAIN_DATA --eval-files $EVAL_DATA \
--train-steps 1000 --eval-steps 100
```

```
leonf_info@cloudshell:~/cloudml-samples-master/census/estimator (data-engineer-274519)$ gcloud ai-platform local train --module-name tra
iner.task --package-path trainer/ \
> --job-dir $MODEL_DIR -- --train-files $TRAIN_DATA --eval-files $EVAL_DATA \
> --train-steps 1000 --eval-steps 100
********************************************************************************
Python 2 is deprecated. Upgrade to Python 3 as soon as possible.
See https://cloud.google.com/python/docs/python2-sunset

To suppress this warning, create an empty ~/.cloudshell/no-python-warning file.
The command will automatically proceed in  seconds or on any key.
********************************************************************************
WARNING:tensorflow:From /home/leonf_info/cloudml-samples-master/census/estimator/trainer/task.py:191: The name tf.logging.set_verbosity
is deprecated. Please use tf.compat.v1.logging.set_verbosity instead.

INFO:tensorflow:TF_CONFIG environment variable: {u'environment': u'cloud', u'cluster': {}, u'job': {u'args': [u'--train-files', u'/home/
leonf_info/cloudml-samples-master/census/estimator/data/adult.data.csv', u'--eval-files', u'/home/leonf_info/cloudml-samples-master/cens
us/estimator/data/adult.test.csv', u'--train-steps', u'1000', u'--eval-steps', u'100', u'--job-dir', u'output'], u'job_name': u'trainer.
task'}, u'task': {}}
Model dir output
INFO:tensorflow:Using config: {'_save_checkpoints_secs': 600, '_num_ps_replicas': 0, '_keep_checkpoint_max': 5, '_task_type': 'worker',
'_global_id_in_cluster': 0, '_is_chief': True, '_cluster_spec': <tensorflow.python.training.server_lib.ClusterSpec object at 0x7f1096d3d
bd0>, '_model_dir': 'output', '_protocol': None, '_save_checkpoints_steps': None, '_keep_checkpoint_every_n_hours': 10000, '_service': N
one, '_session_config': allow_soft_placement: true
graph_options {
  rewrite_options {
    meta_optimizer_iterations: ONE
  }
}
, '_tf_random_seed': None, '_save_summary_steps': 100, '_device_fn': None, '_session_creation_timeout_secs': 7200, '_experimental_distri
bute': None, '_num_worker_replicas': 1, '_task_id': 0, '_log_step_count_steps': 100, '_experimental_max_worker_delay_secs': None, '_eval
uation_master': '', '_eval_distribute': None, '_train_distribute': None, '_master': ''}
INFO:tensorflow:Not using Distribute Coordinator.
```

```
WARNING:tensorflow:Export includes no default signature!
INFO:tensorflow:Restoring parameters from output/model.ckpt-1000
INFO:tensorflow:Assets added to graph.
INFO:tensorflow:No assets to write.
INFO:tensorflow:SavedModel written to: output/export/census/temp-1590513459/saved_model.pb
INFO:tensorflow:Loss for final step: 16.124683.
```

```
# Viewing results in tensorboard
tensorboard --logdir=$MODEL_DIR --port=8080
```

```
leonf_info@cloudshell:~/cloudml-samples-master/census/estimator (data-engineer-274519)$ tensorboard --logdir=$MODEL_DIR --port=8080
W0526 13:22:25.694485 140590387840768 plugin_event_accumulator.py:294] Found more than one graph event per run, or there was a metagraph
 containing a graph_def, as well as one or more graph events.  Overwriting the graph with the newest event.
W0526 13:22:25.699318 140590387840768 plugin_event_accumulator.py:302] Found more than one metagraph event per run. Overwriting the meta
graph with the newest event.
TensorBoard 1.15.0 at http://cs-6000-devshell-vm-73658e62-4b5d-4301-afff-7f546f32260f:8080/ (Press CTRL+C to quit)
```

Run trainer on GCP AI Platform - single instance

Create regional Cloud Storage bucket used for all output and staging (make-bucket command)

```
REGION=us-central1 /

gsutil mb -l $REGION gs://$DEVSHELL_PROJECT_ID-aip-demo
```

```
leonf_info@cloudshell:~/cloudml-samples-master/census/estimator (data-engineer-274519)$ REGION=us-central1
leonf_info@cloudshell:~/cloudml-samples-master/census/estimator (data-engineer-274519)$ gsutil mb -l $REGION gs://$DEVSHELL_PROJECT_ID-aip-demo
Creating gs://data-engineer-274519-aip-demo/...
```

Upload training and test/eval data to bucket

```
cd ~/cloudml-samples-master/census/estimator

gsutil cp -r data gs://$DEVSHELL_PROJECT_ID-aip-demo/data
```

```
leonf_info@cloudshell:~/cloudml-samples-master/census/estimator (data-engineer-274519)$ cd ~/cloudml-samples-master/census/estimator
leonf_info@cloudshell:~/cloudml-samples-master/census/estimator (data-engineer-274519)$ gsutil cp -r data gs://$DEVSHELL_PROJECT_ID-aip-
demo/data
Copying file://data/test.json [Content-Type=application/json]...
Copying file://data/census.test.csv [Content-Type=text/csv]...
Copying file://data/adult.data.csv [Content-Type=text/csv]...
Copying file://data/test.csv [Content-Type=text/csv]...
- [4 files][  5.5 MiB/  5.5 MiB]
==> NOTE: You are performing a sequence of gsutil operations that may
run significantly faster if you instead use gsutil -m cp ... Please
see the -m section under "gsutil help options" for further information
about when gsutil -m can be advantageous.

Copying file://data/adult.test.csv [Content-Type=text/csv]...
Copying file://data/census.train.csv [Content-Type=text/csv]...
\ [6 files][ 10.7 MiB/ 10.7 MiB]
Operation completed over 6 objects/10.7 MiB.
```

Set data variables to point to storage bucket files

```
TRAIN_DATA=gs://$DEVSHELL_PROJECT_ID-aip-demo/data/adult.data.csv

EVAL_DATA=gs://$DEVSHELL_PROJECT_ID-aip-demo/data/adult.test.csv
```

```
leonf_info@cloudshell:~/cloudml-samples-master/census/estimator (data-engineer-274519)$ EVAL_DATA=gs://$DEVSHELL_PROJECT_ID-aip-demo/data/adult.test.csv
leonf_info@cloudshell:~/cloudml-samples-master/census/estimator (data-engineer-274519)$ TRAIN_DATA=gs://$DEVSHELL_PROJECT_ID-aip-demo/data/adult.data.csv
```

Additional set-up: Copy test.json to storage bucket (for prediction laster on)

```
gsutil cp ../test.json gs://$DEVSHELL_PROJECT_ID-aip-demo/data/test.json
```

```
leonf_info@cloudshell:~/cloudml-samples-master/census/estimator (data-engineer-274519)$ gsutil cp ../test.json gs://$DEVSHELL_PROJECT_ID
-aip-demo/data/test.json
Copying file://../test.json [Content-Type=application/json]...
/ [1 files][  314.0 B/  314.0 B]
Operation completed over 1 objects/314.0 B.
```

Set TEST_JSON to point to the same storage bucket file

```
TEST_JSON=gs://$DEVSHELL_PROJECT_ID-aip-demo/data/test.json
```

```
leonf_info@cloudshell:~/cloudml-samples-master/census/estimator (data-engineer-274519)$ TEST_JSON=gs://$DEVSHELL_PROJECT_ID-aip-demo/dat
a/test.json
```

Set variables for job name and output path (for AI Platform)

cloud storage staging bucket which is called as the job name

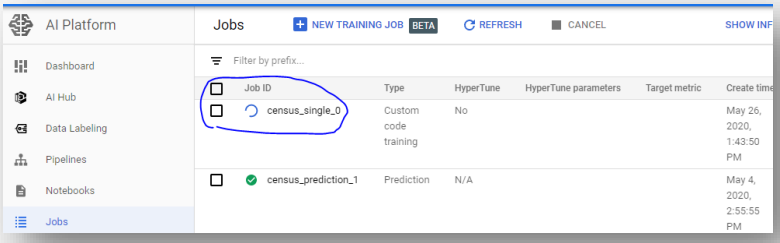
```
JOB_NAME=census_single_0

OUTPUT_PATH=gs://$DEVSHELL_PROJECT_ID-aip-demo/$JOB_NAME
```

```
leonf_info@cloudshell:~/cloudml-samples-master/census/estimator (data-engineer-274519)$ JOB_NAME=census_single_1
leonf_info@cloudshell:~/cloudml-samples-master/census/estimator (data-engineer-274519)$ OUTPUT_PATH=gs://$DEVSHELL_PROJECT_ID-aip-demo/$
JOB_NAME
```

Submit a single process job to AI Platform

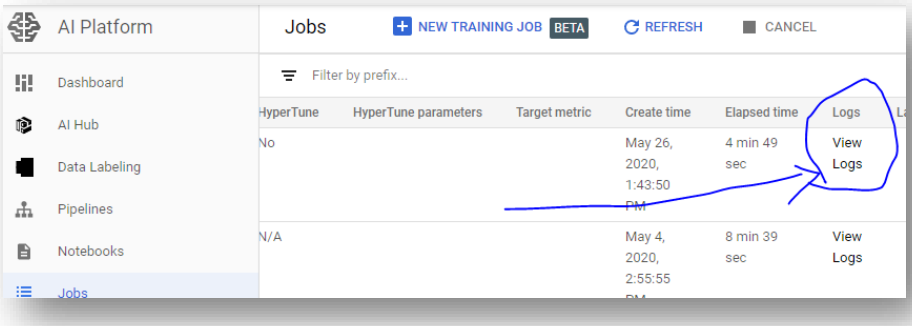
- Job name is JOB_NAME (census_single_1)
- Output path is our Cloud storage bucket/job_name
- Training and evaluation/test data is in our Cloud Storage bucket

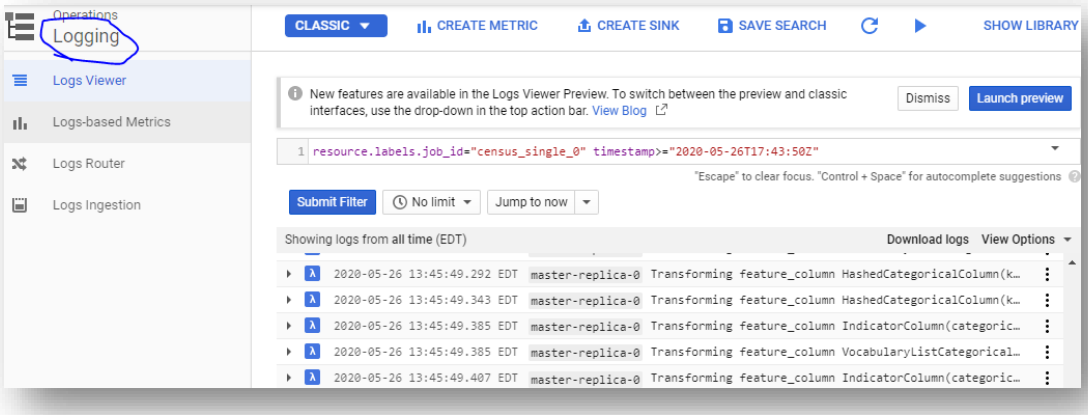
```
gcloud ai-platform jobs submit training $JOB_NAME \
--job-dir $OUTPUT_PATH \
--runtime-version 1.15 \
--module-name trainer.task \
--package-path trainer/ \
--region $REGION \
-- \
--train-files $TRAIN_DATA \
--eval-files $EVAL_DATA \
--train-steps 1000 \
--eval-steps 100 \
--verbosity DEBUG
```



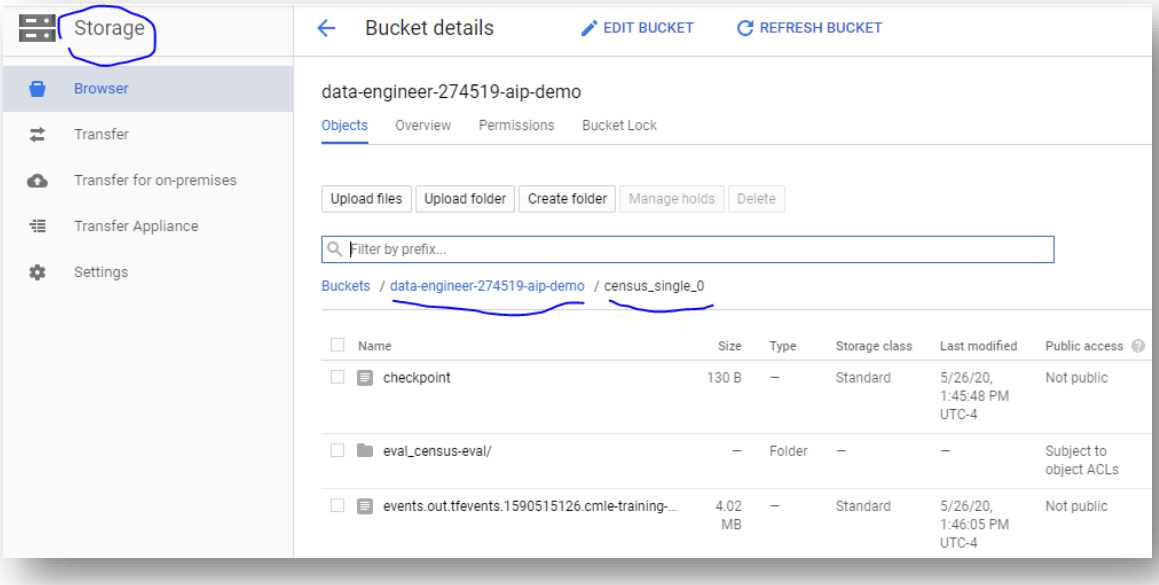

```
leonf_info@cloudshell:~/cloudml-samples-master/census/estimator (data-engineer-274519)$ JOB_NAME=census_single_0
leonf_info@cloudshell:~/cloudml-samples-master/census/estimator (data-engineer-274519)$ OUTPUT_PATH=gs://$DEVSHELL_PROJECT_ID-aip-demo/$JOB_NAME
leonf_info@cloudshell:~/cloudml-samples-master/census/estimator (data-engineer-274519)$ gcloud ai-platform jobs submit training $JOB_NAME \
> --job-dir $OUTPUT_PATH \
> --runtime-version 1.15 \
> --module-name trainer.task \
> --package-path trainer/ \
> --region $REGION \
> -- \
> --train-files $TRAIN_DATA \
> --eval-files $EVAL_DATA \
> --train-steps 1000 \
> --eval-steps 100 \
> --verbosity DEBUG
Job [census_single_0] submitted successfully.
Your job is still active. You may view the status of your job with the command

  $ gcloud ai-platform jobs describe census_single_0

or continue streaming the logs with the command

  $ gcloud ai-platform jobs stream-logs census_single_0
jobId: census_single_0
state: QUEUED
```

Can view streaming logs/output (stackdriver logging) with gcloud ai-platform jobs stream-logs $JOB_NAME





When complete, inspect output path with gsutil ls -r $OUTPUT_PATH (in cloud storage bucket with model.ckpt) and we will use those files for deployment



Run distributed training on AI Platform (on 4 machines)

Create variable for distributed job name

```
cd ~/cloudml-samples-master/census/estimator

JOB_NAME=census_dist_2
```

```
leonf_info@cloudshell:~/cloudml-samples-master/census/estimator (data-engineer-274519)$ cd ~/cloudml-samples-master/census/estimator
leonf_info@cloudshell:~/cloudml-samples-master/census/estimator (data-engineer-274519)$ JOB_NAME=census_dist_2
```

Set new output path to Cloud Storage location using new JOB_NAME variable

```
OUTPUT_PATH=gs://$DEVSHELL_PROJECT_ID-aip-demo/$JOB_NAME
```

```
leonf_info@cloudshell:~/cloudml-samples-master/census/estimator (data-engineer-274519)$ OUTPUT_PATH=gs://$DEVSHELL_PROJECT_ID-aip-demo/$
JOB_NAME
```

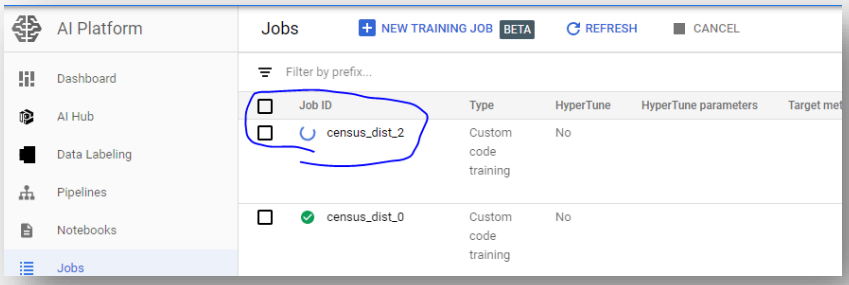
Submit a distributed training job

The '--scale-tier STANDARD_1' option is the new item that initiates distributed scaling

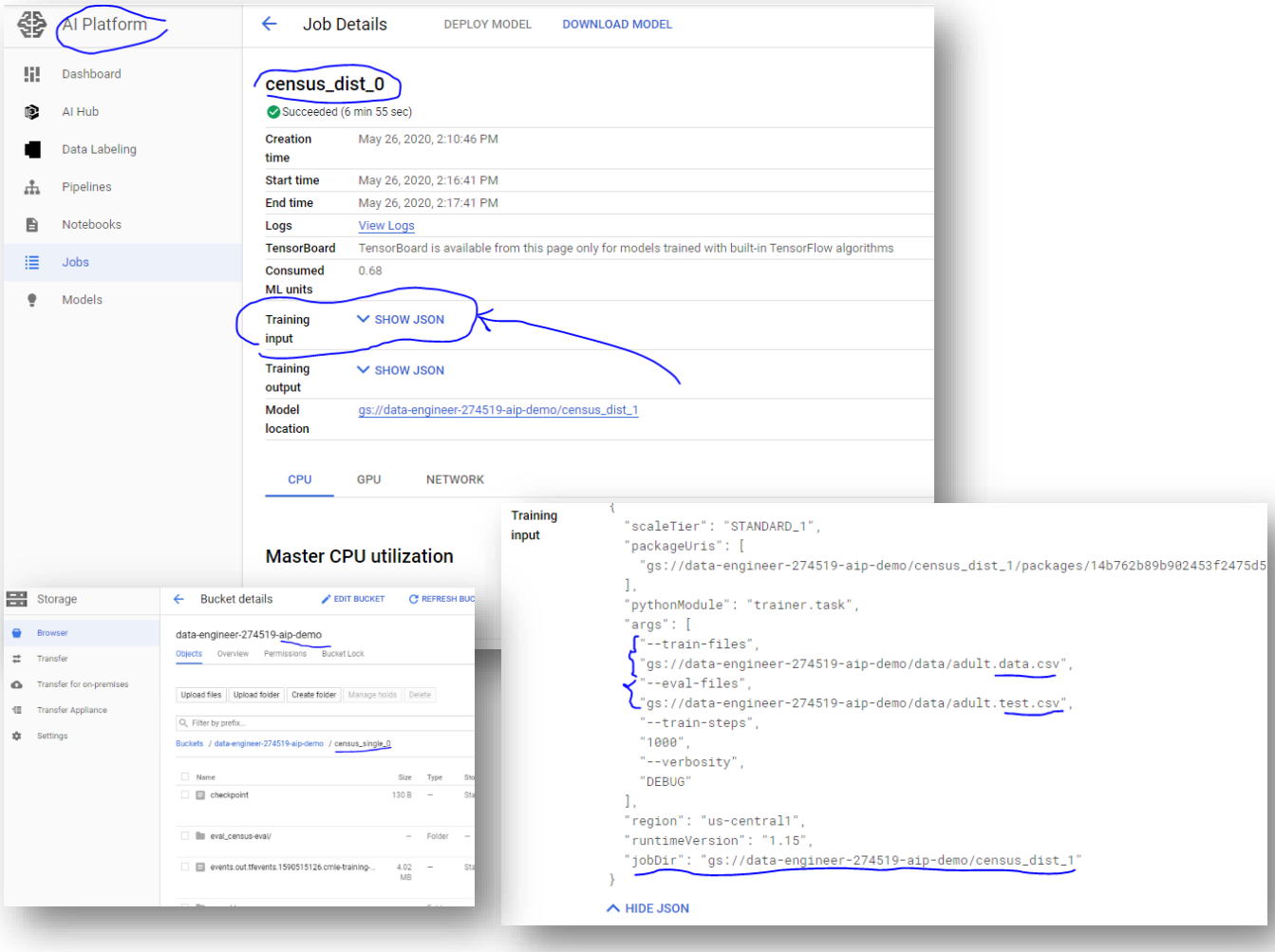
1 master, 4 workers, 3 parameter servers

```
leonf_info@cloudshell:~/cloudml-samples-master/census/estimator (data-engineer-274519)$ cd ~/cloudml-samples-master/census/estimator
leonf_info@cloudshell:~/cloudml-samples-master/census/estimator (data-engineer-274519)$ JOB_NAME=census_dist_2
leonf_info@cloudshell:~/cloudml-samples-master/census/estimator (data-engineer-274519)$ OUTPUT_PATH=gs://$DEVSHELL_PROJECT_ID-aip-demo/$JOB_NAME
leonf_info@cloudshell:~/cloudml-samples-master/census/estimator (data-engineer-274519)$ gcloud ai-platform jobs submit training $JOB_NAME \
> --job-dir $OUTPUT_PATH \
> --runtime-version 1.15 \
> --module-name trainer.task \
> --package-path trainer/ \
> --region $REGION \
> --scale-tier STANDARD_1 \
> -- \
> --train-files $TRAIN_DATA \
> --eval-files $EVAL_DATA \
> --train-steps 1000 \
> --verbosity DEBUG \
> --eval-steps 100

Job [census_dist_2] submitted successfully.
Your job is still active. You may view the status of your job with the command

  $ gcloud ai-platform jobs describe census_dist_2

or continue streaming the logs with the command

  $ gcloud ai-platform jobs stream-logs census_dist_2
jobId: census_dist_2
state: QUEUED
```

```
gcloud ai-platform jobs submit training $JOB_NAME \
--job-dir $OUTPUT_PATH \
--runtime-version 1.15 \
--module-name trainer.task \
--package-path trainer/ \
--region $REGION \
--scale-tier STANDARD_1 \
-- \
--train-files $TRAIN_DATA \
--eval-files $EVAL_DATA \
--train-steps 1000 \
--verbosity DEBUG \
--eval-steps 100
```



View the training input



Prediction phase (testing it out)

Deploy a model (Named census_2) for prediction, setting variables in the process

The deployed mode name: census

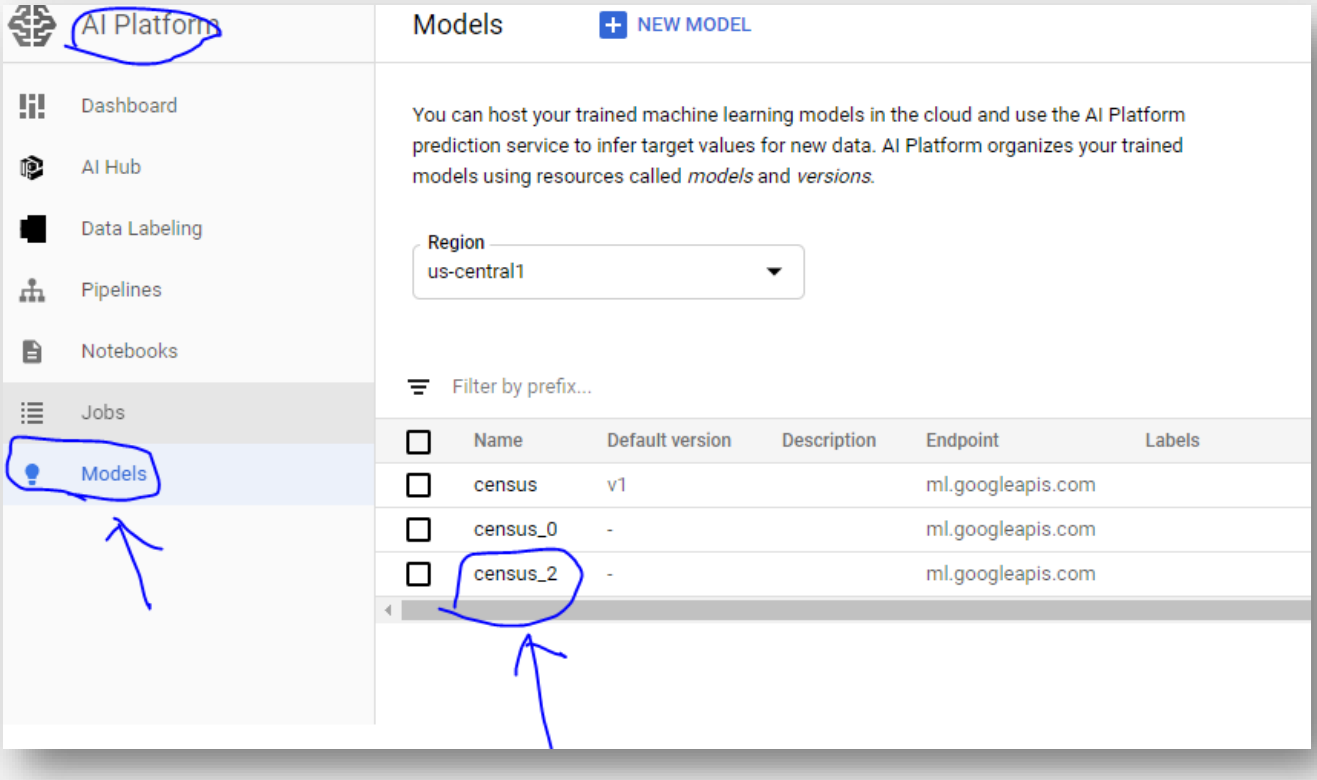
```
cd ~/cloudml-samples-master/census/estimator

MODEL_NAME=census_2
```

```
leonf_info@cloudshell:~/cloudml-samples-master/census/estimator (data-engineer-274519)$ cd ~/cloudml-samples-master/census/estimator
leonf_info@cloudshell:~/cloudml-samples-master/census/estimator (data-engineer-274519)$ MODEL_NAME=census_2
```

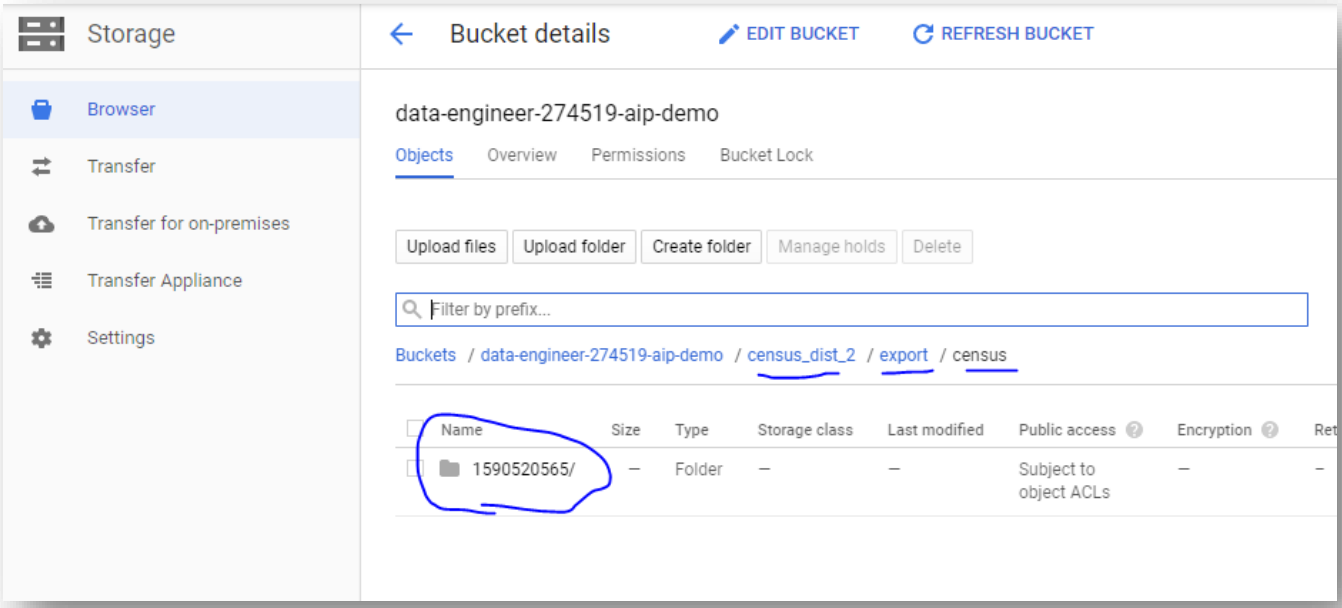
Create the ML Engine model (a platform)

```
gcloud ai-platform models create $MODEL_NAME --regions=$REGION
```

```
leonf_info@cloudshell:~/cloudml-samples-master/census/estimator (data-engineer-274519)$ gcloud ai-platform models create $MODEL_NAME --r
egions=$REGION
WARNING: Using endpoint [https://ml.googleapis.com/]
Created ml engine model [projects/data-engineer-274519/models/census_2].
```



Set the job output we want to use. This example uses census_dist_2

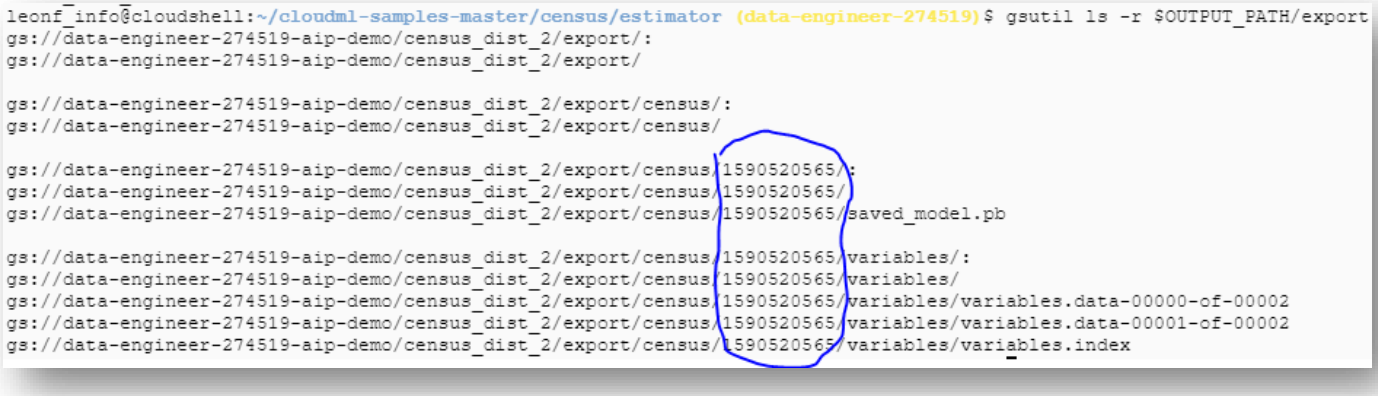



```
OUTPUT_PATH=gs://$DEVSHELL_PROJECT_ID-aip-demo/census_dist_2
```

CHANGE CENSUS_DIST_1 TO USE A DIFFERENT OUTPUT FROM PREVIOUS

IMPORTANT - Look up and **set full path** for **export trained model binaries**

```
gsutil ls -r $OUTPUT_PATH/export
```



Look for directory $OUTPUT_PATH/export/census/ and copy/paste **timestamp value (without colon)** into the below command

`MODEL_BINARIES=gs://$DEVSHELL_PROJECT_ID-aip-demo/census_dist_2/export/census/<timestamp>` ###CHANGE ME! (**blue circle**)

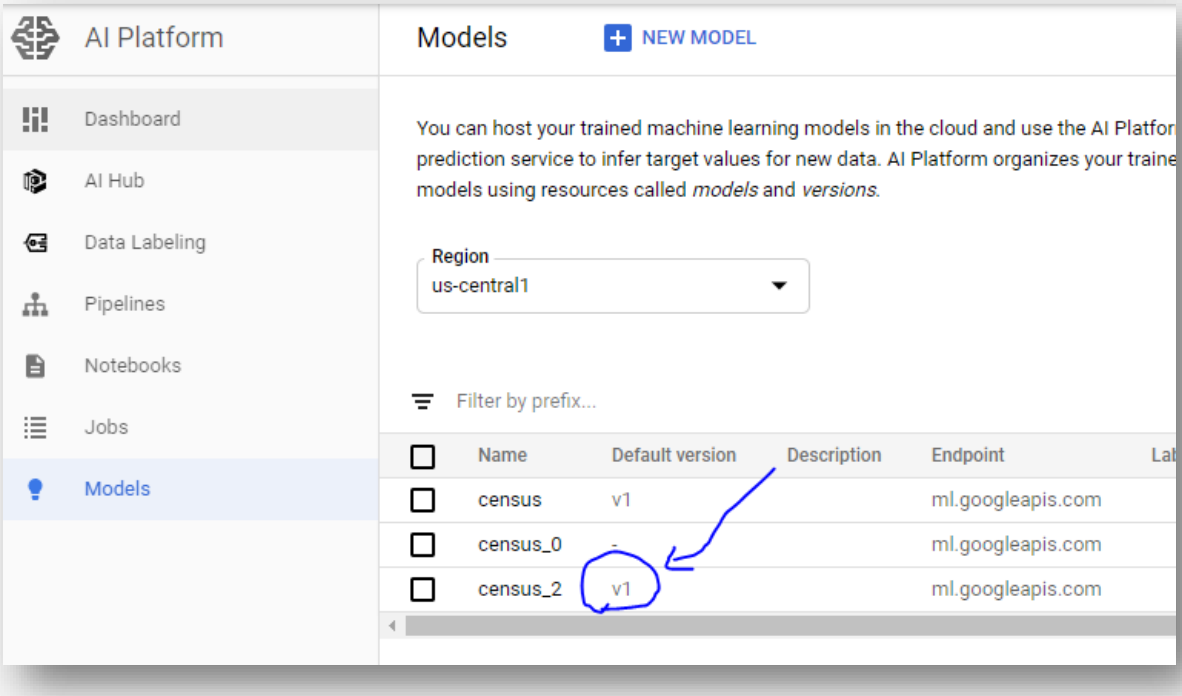
```
leonf_info@cloudshell:~/cloudml-samples-master/census/estimator (data-engineer-274519)$ MODEL_BINARIES=gs://$DEVSHELL_PROJECT_ID-aip-dem
o/census_dist_2/export/census/1590520565
```

Create Version 1 of your model

Now deploy the version 1 model

```
gcloud ai-platform versions create v1 \
--model $MODEL_NAME \
--origin $MODEL_BINARIES \
--runtime-version 1.15
```

```
leonf_info@cloudshell:~/cloudml-samples-master/census/estimator (data-engineer-274519)$ gcloud ai-platform versions create v1 \
> --model $MODEL_NAME \
> --origin $MODEL_BINARIES \
> --runtime-version 1.15
WARNING: Using endpoint [https://ml.googleapis.com/]
Creating version (this might take a few minutes)......done.
```



Send an online prediction request to our deployed model using test.json file

Results come back with a direct response (single data point)

```
gcloud ai-platform predict \
--model $MODEL_NAME \
--version v1 \
--json-instances \
../test.json
```

```
leonf_info@cloudshell:~/cloudml-samples-master/census/estimator (data-engineer-274519)$ gcloud ai-platform predict \
> --model $MODEL_NAME \
> --version v1 \
> --json-instances \
> ../test.json
WARNING: Using endpoint [https://ml.googleapis.com/]
ALL_CLASS_IDS  ALL_CLASSES  CLASS_IDS  CLASSES  LOGISTIC                LOGITS                 PROBABILITIES
[0, 1]         ['0', '1']   [0]        ['0']    [0.047716088593006134]  [-2.9935946464538574]  [0.9522839188575745, 0.047716084867715836
]
```

Send a batch prediction job using same test.json file

Results are exported to a Cloud Storage bucket location

Set job name and output path variables

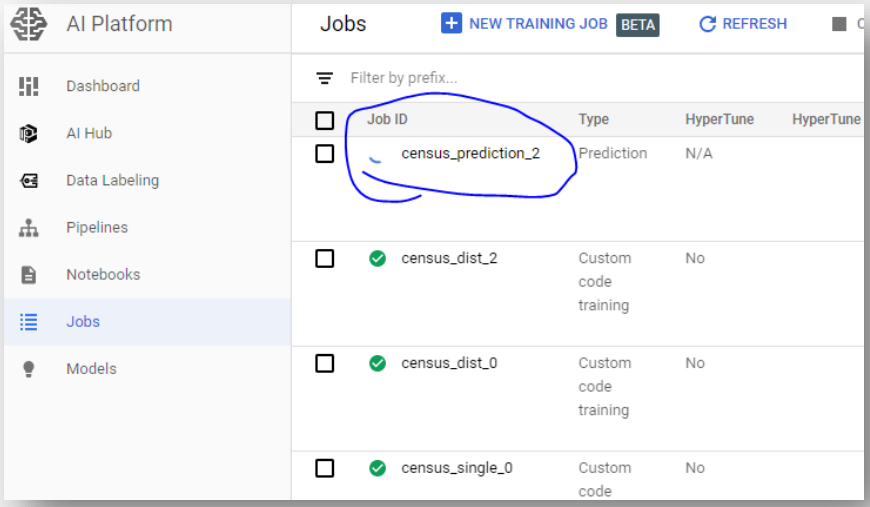
```
JOB_NAME=census_prediction_2
OUTPUT_PATH=gs://$DEVSHELL_PROJECT_ID-aip-demo/$JOB_NAME
```

```
leonf_info@cloudshell:~/cloudml-samples-master/census/estimator (data-engineer-274519)$ JOB_NAME=census_prediction_2
leonf_info@cloudshell:~/cloudml-samples-master/census/estimator (data-engineer-274519)$ OUTPUT_PATH=gs://$DEVSHELL_PROJECT_ID-aip-demo/$
JOB_NAME
```

Submit the prediction job

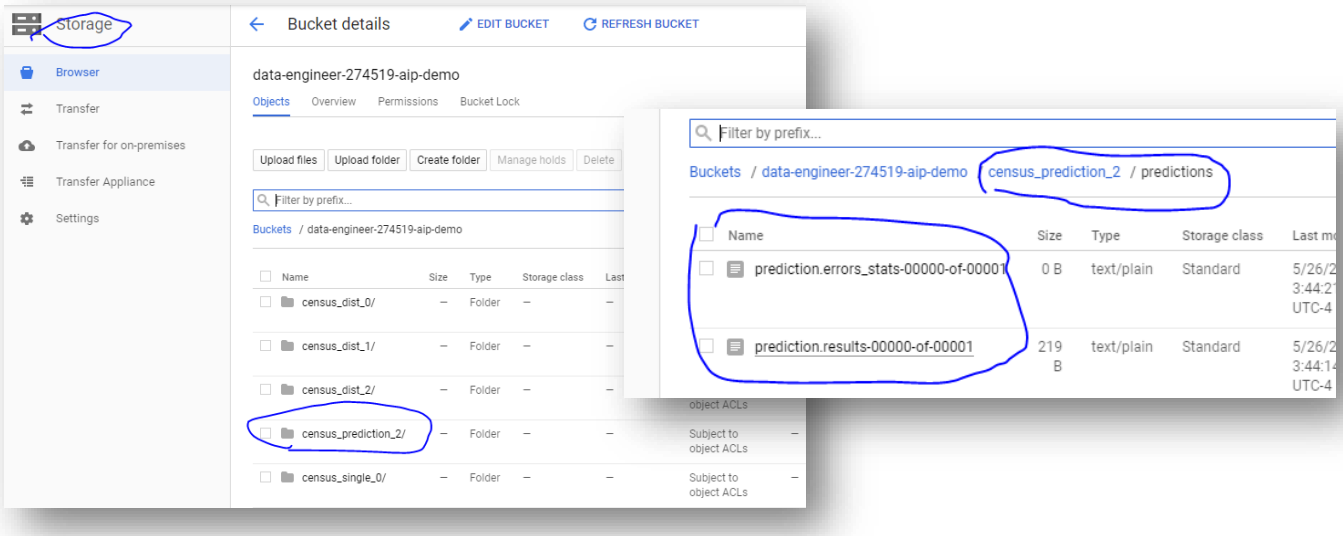
```
gcloud ai-platform jobs submit prediction $JOB_NAME \
--model $MODEL_NAME \
--version v1 \
--data-format TEXT \
--region $REGION \
--input-paths $TEST_JSON \
--output-path $OUTPUT_PATH/predictions
```

```
leonf_info@cloudshell:~/cloudml-samples-master/census/estimator (data-engineer-274519)$ gcloud ai-platform jobs submit prediction $JOB_N
AME \
> --model $MODEL_NAME \
> --version v1 \
> --data-format TEXT \
> --region $REGION \
> --input-paths $TEST_JSON \
> --output-path $OUTPUT_PATH/predictions

Job [census_prediction_2] submitted successfully.
Your job is still active. You may view the status of your job with the command

  $ gcloud ai-platform jobs describe census_prediction_2

or continue streaming the logs with the command

  $ gcloud ai-platform jobs stream-logs census_prediction_2
jobId: census_prediction_2
state: QUEUED
```



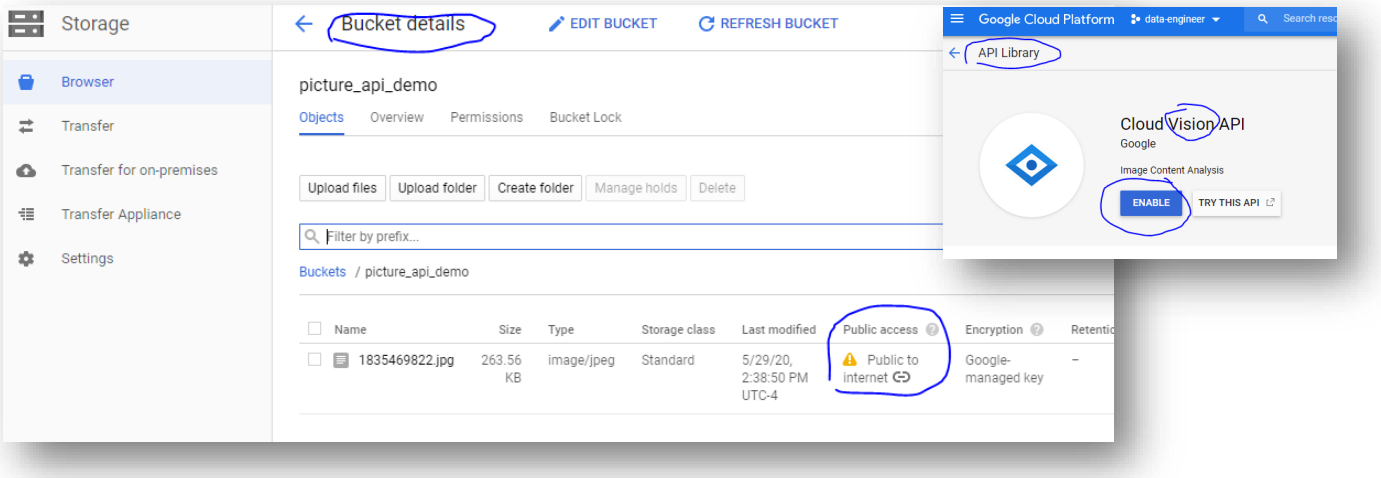
View results in web console at

`gs://$DEVSHELL_PROJECT_ID-aip-demo/$JOB_NAME/predictions/`

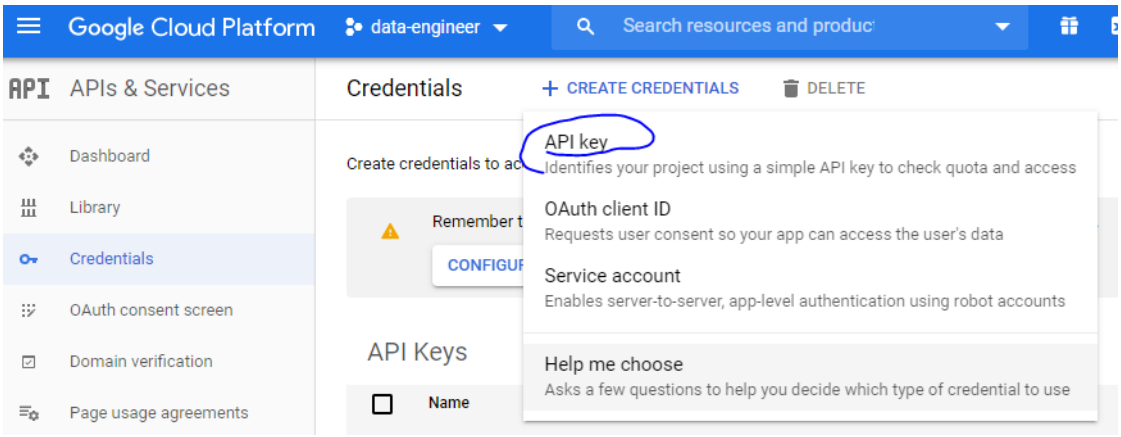


Lecture: Vision API Demo

The commands we used to authenticate with our API key, create *request.json* file, and the `curl` command used to call on the API are below for your reference.



1. Authenticate with your API key after **creating the API credential**:
   `export API_KEY=(your copied key)`



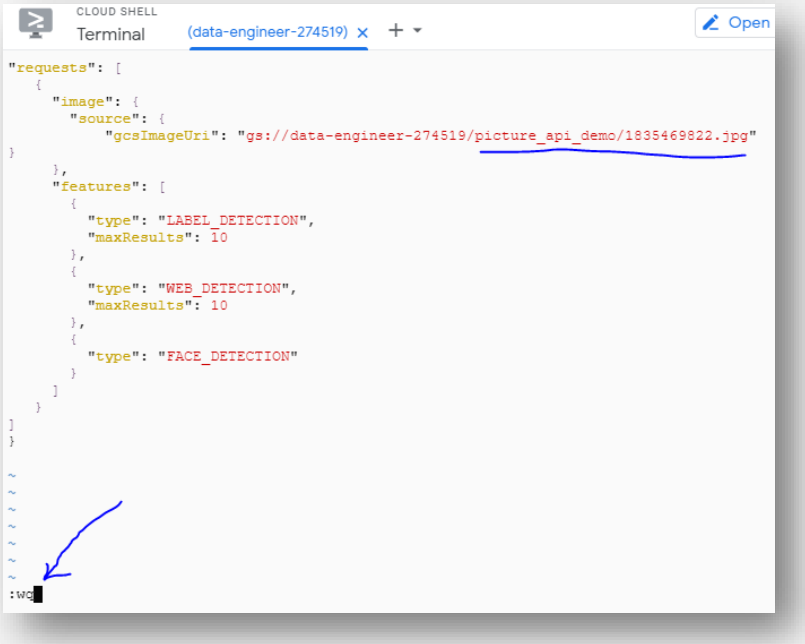
`AIzaSyB1hXl5vTTyea5skCP6PCOxqLOxNB-h2rg`

```
leonf_info@cloudshell:~ (data-engineer-274519)$ export API_KEY=AIzaSyB1hXl5vTTyea5skCP6PCOxqLOxNB-h2rg
```

2. Create your JSON file (points to img location on bucket) by typing:
   `vim request.json`

3. Then copy/paste the following text, substituting the Cloud Storage info with your actual Cloud Storage bucket and object info:

```
"requests": [
  {
    "image": {
      "source": {
          "gcsImageUri": "gs://(your_bucket)/(your_image_file)"
      }
    },
    "features": [
      {
        "type": "LABEL_DETECTION",
        "maxResults": 10
      },
      {
        "type": "WEB_DETECTION",
        "maxResults": 10
      },
      {
        "type": "FACE_DETECTION"
      }
    ]
  }
]
}
```



4. Use the below **curl command** to authenticate with the Vision API using your API key, and supplying the *request.json* file we just created:
   `curl -s -X POST -H "Content-Type: application/json" --data-binary @request.json https://vision.googleapis.com/v1/images:annotate?key=${API_KEY}`

Lecture: Datalab Demo

```
$ datalab create [-h] [--image-name IMAGE_NAME] [--disk-name DISK_NAME]
                 [--disk-size-gb DISK_SIZE_GB]  [--network-name NETWORK_NAME]
                 [--subnet-name SUBNET_NAME] [--idle-timeout IDLE_TIMEOUT]
                 [--machine-type MACHINE_TYPE] [--no-connect] [--no-swap]
                 [--no-backups] [--beta-no-external-ip] [--no-create-repository]
                 [--log-level {trace,debug,info,warn,error,fatal}]
                 [--for-user FOR_USER] [--service-account SERVICE_ACCOUNT]
                 [--port PORT] [--max-reconnects MAX_RECONNECTS]
                 [--connection-health-timeout-seconds CONNECTION_HEALTH_TIMEOUT_SECONDS]
                 [--ssh-log-level {quiet,fatal,error,info,verbose,debug,debug1,debug2,debug3}]
                 [--no-launch-browser] [--beta-internal-ip]
                 [--project PROJECT] [--quiet]
                 [--verbosity {debug,info,default,warning,error,critical,none}]
                 [--zone ZONE]
                 NAME
```

https://cloud.google.com/datalab/docs/reference/command-line/create

To open a datalab via command line, we need to use google **cloud shell**

Create datalab notebook:

(1) Go over different **options**
(2) Custom **machine type**
(3) Separate VPC / source repository created for you

Connecting / web preview:

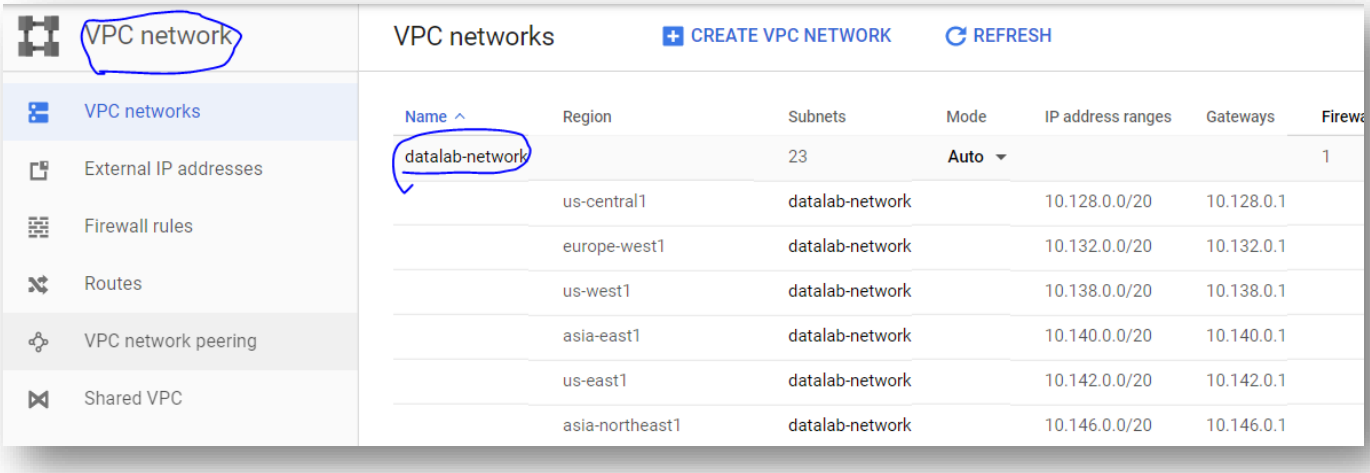
(1) Authenticated as **service account**, rather than individual google account

To create a Datalab notebook using Cloud Shell, type the below command:
`datalab create (instance-name)`

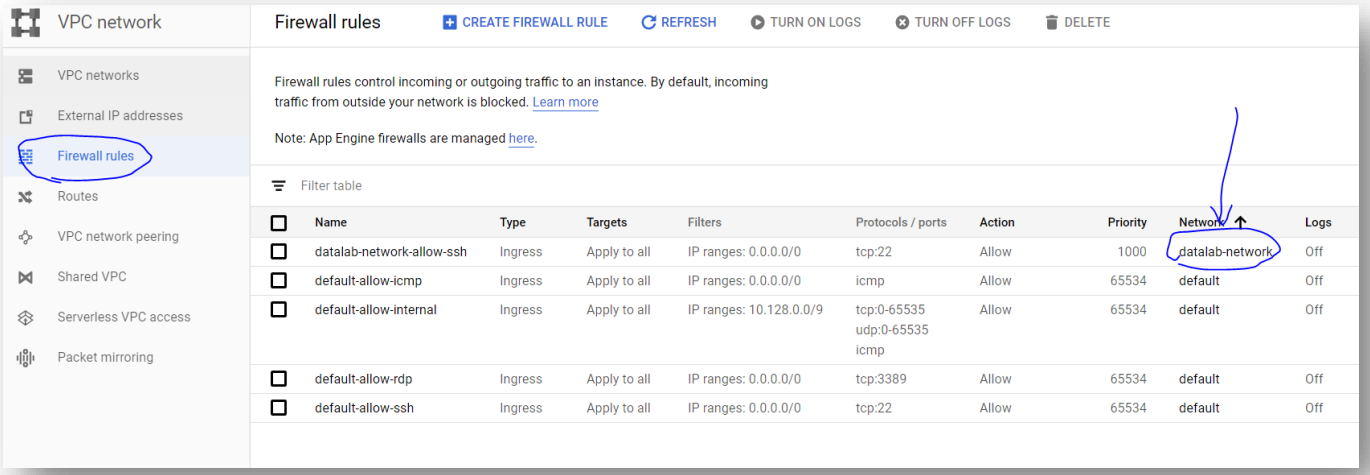
```
leonf_info@cloudshell:~ (data-engineer-274519)$ datalab create datalab-demo
Please specify a zone from one of:
 [1] us-east1-b
 [2] us-east1-c
 [3] us-east1-d
 [4] us-east4-c
 [5] us-east4-b
 [6] us-east4-a
 [7] us-central1-c
 [8] us-central1-a
 [9] us-central1-f
 [10] us-central1-b
 [11] us-west1-b
 [12] us-west1-c
 [13] us-west1-a
 [14] europe-west4-a
```

```
 [70] us-west4-c
Your selected zone: 8
Creating the instance datalab-demo
Created [https://www.googleapis.com/compute/v1/projects/data-engineer-274519/zones/us-central1-a/instances/datalab-demo].
Connecting to datalab-demo.
This will create an SSH tunnel and may prompt you to create an rsa key pair. To manage these keys, see https://cloud.google.com/compute/
docs/instances/adding-removing-ssh-keys
Waiting for Datalab to be reachable at http://localhost:8081/
```

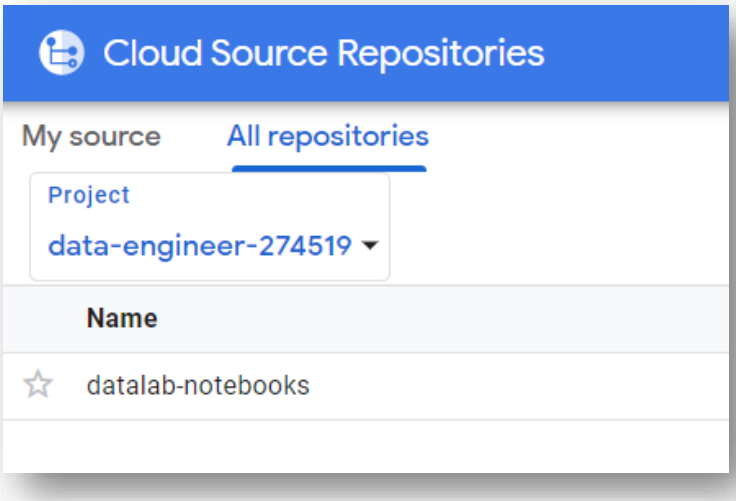
‘datalab-network’ is our default network



‘Port 22’ is for the datalab network



And we have a repository call **‘datalab-notebooks’** created for us

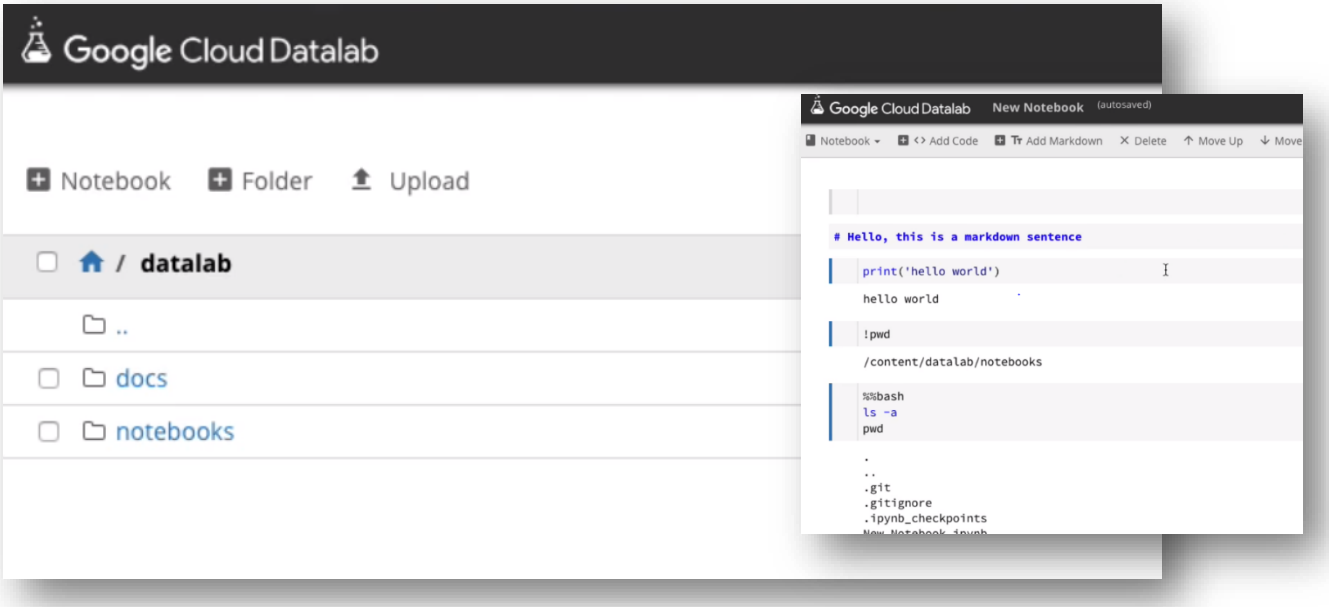


To connect to a Datalab notebook, type:
`datalab connect (instance-name)`

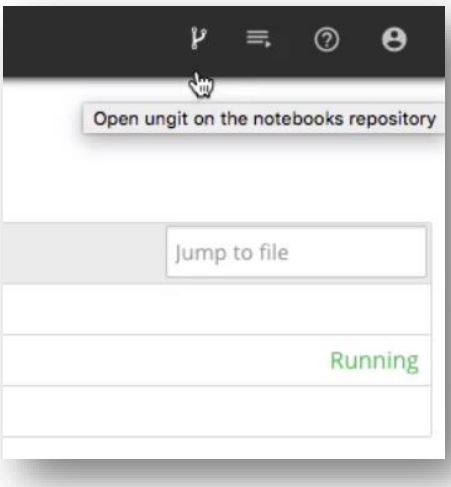
```
leonf_info@cloudshell:~ (data-engineer-274519)$ datalab connect datalab-demo
Connecting to datalab-demo.
This will create an SSH tunnel and may prompt you to create an rsa key pair. To manage these keys, see https://cloud.google.com/compute/
docs/instances/adding-removing-ssh-keys
Waiting for Datalab to be reachable at http://localhost:8081/

The connection to Datalab is now open and will remain until this command is killed.
Click on the *Web Preview* (square button at top-right), select *Change port > Port 8081*, and start using Datalab.
```

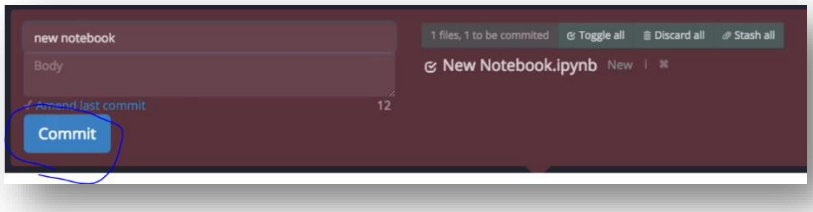
Working in a notebook

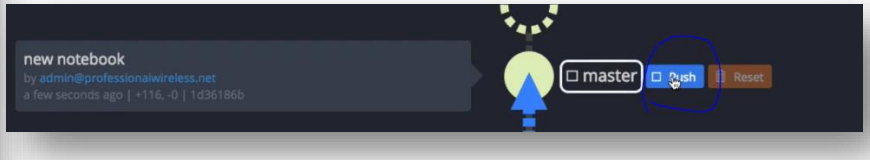
You can create a New notebook here, and share it by using the Pushing (next step)



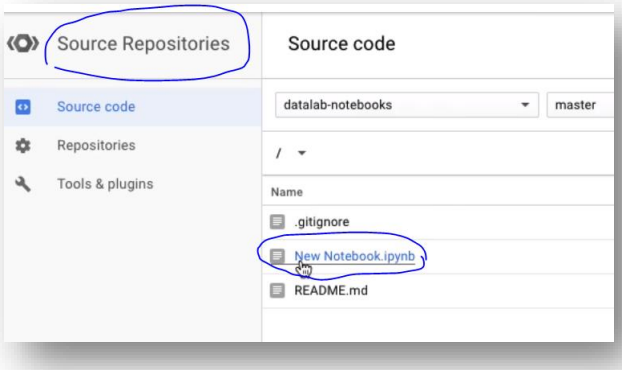
Pushing to source repository





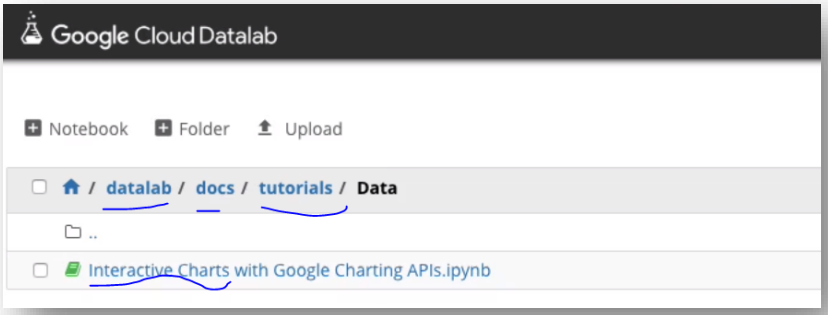


Go back to repo, now we have a new file:

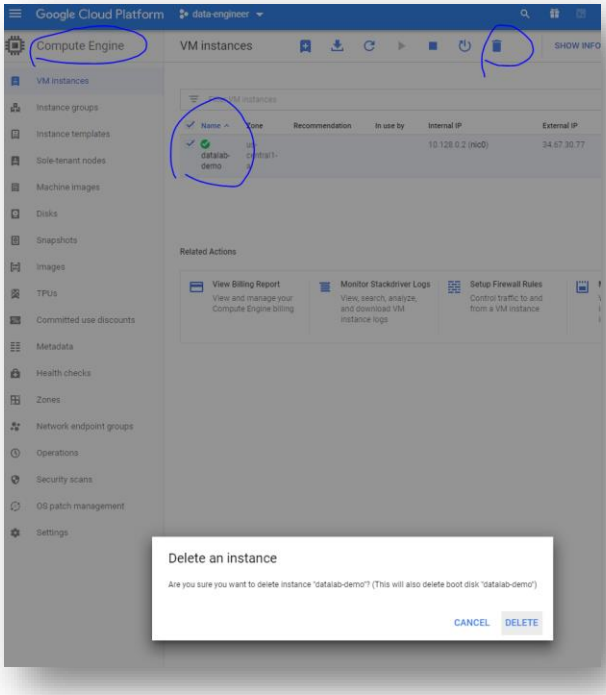


Examples

Lots of useful learning materials here



Clean up

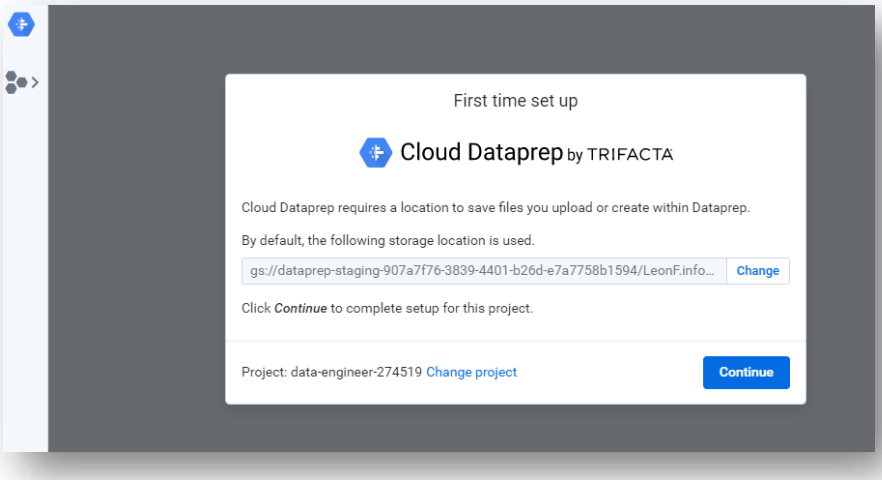


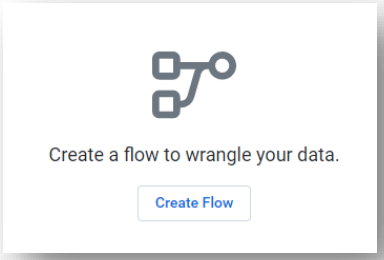
Lecture: Dataprep Demo Part 1

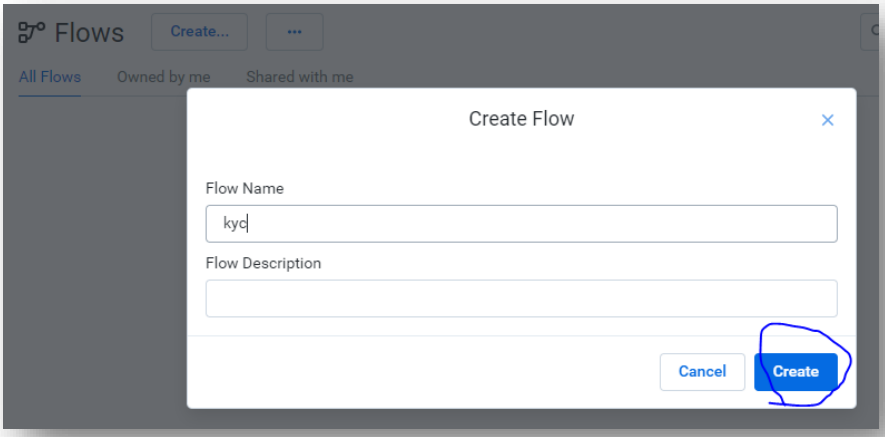
https://cloud.google.com/dataprep/docs/quickstarts/quickstart-dataprep

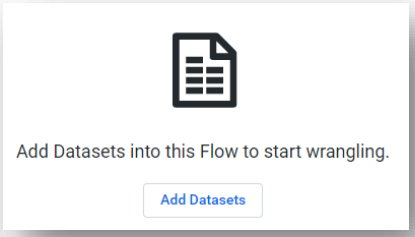
This is the first in a three-part demo of Dataprep.
The public cloud storage bucket we used for our datasets is `gs://la-dataprep-samples`.

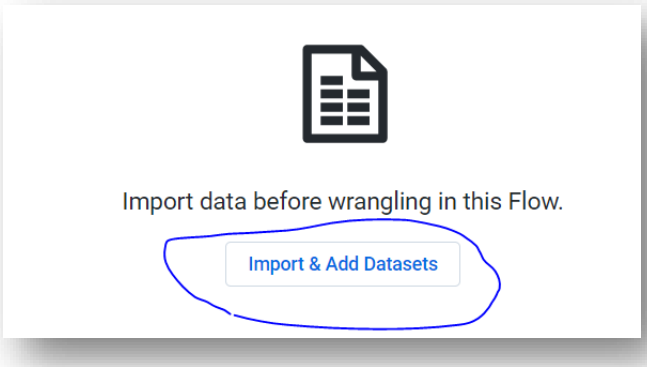
NOTE: Be aware that the above data is no longer available for the Customer 360 subfolder. You can find similar examples at `gs://la-dataprep-samples`

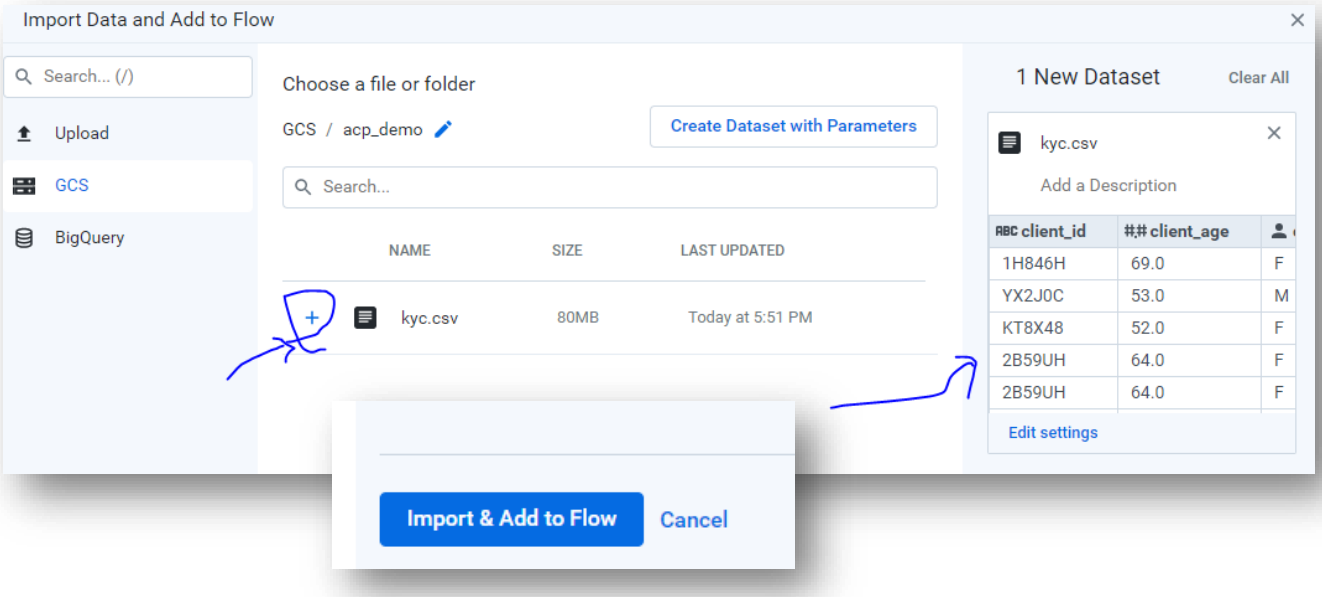






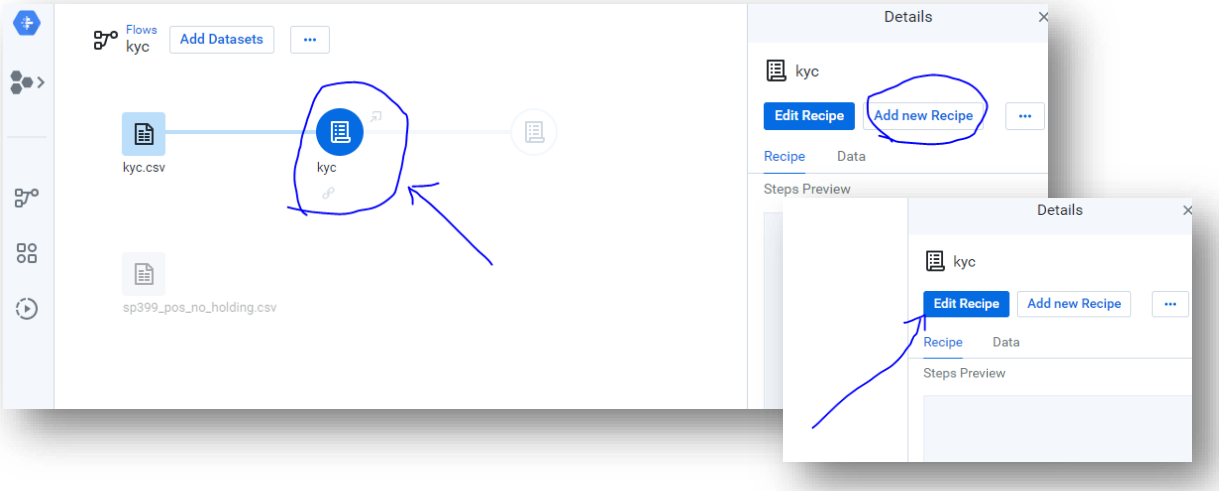


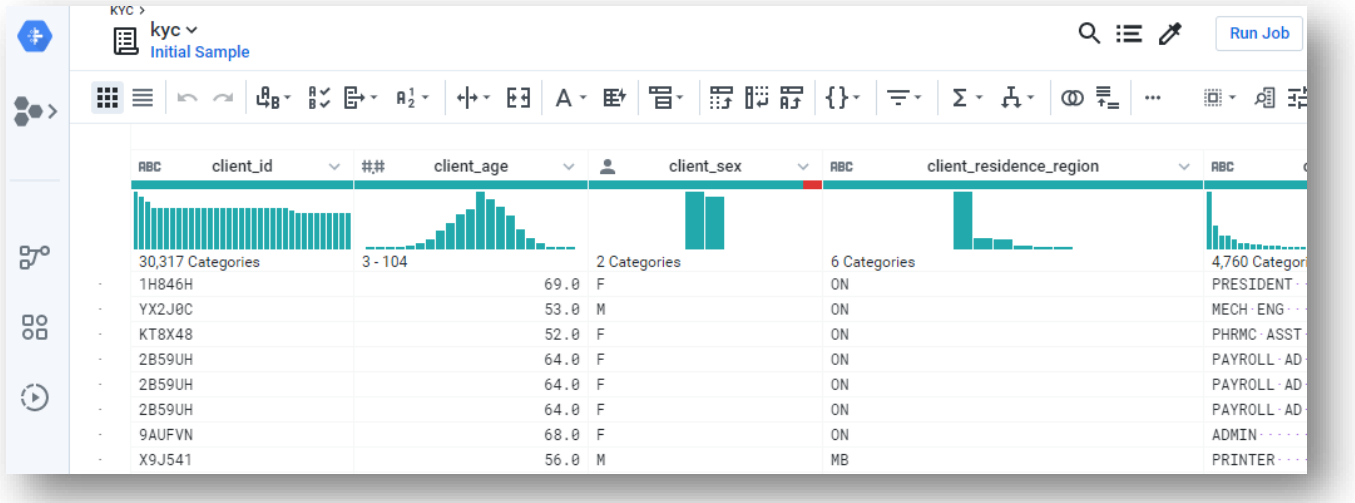




Click **add new recipe**, and then **edit recipe:**

(it using cloud storage for the staging process)

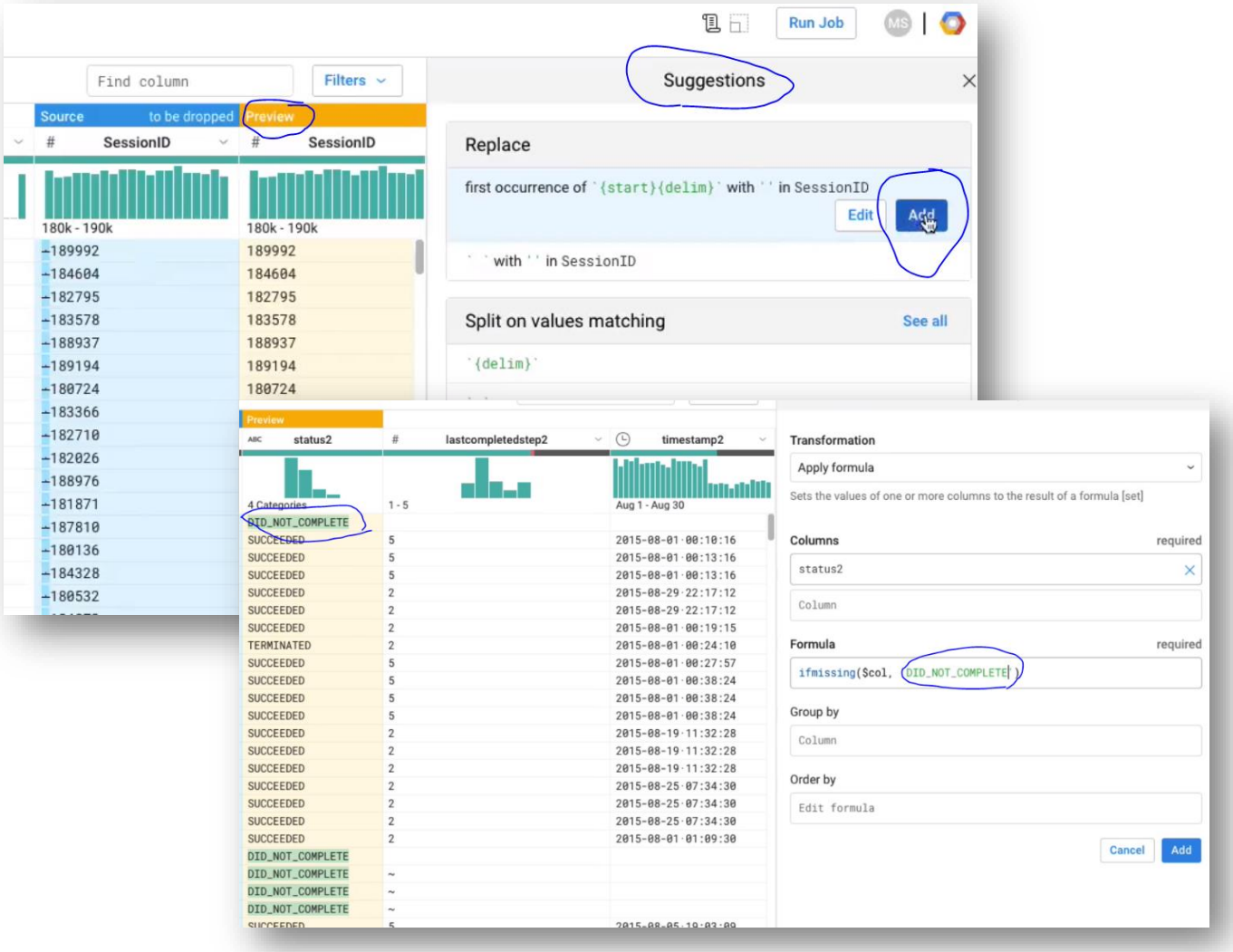




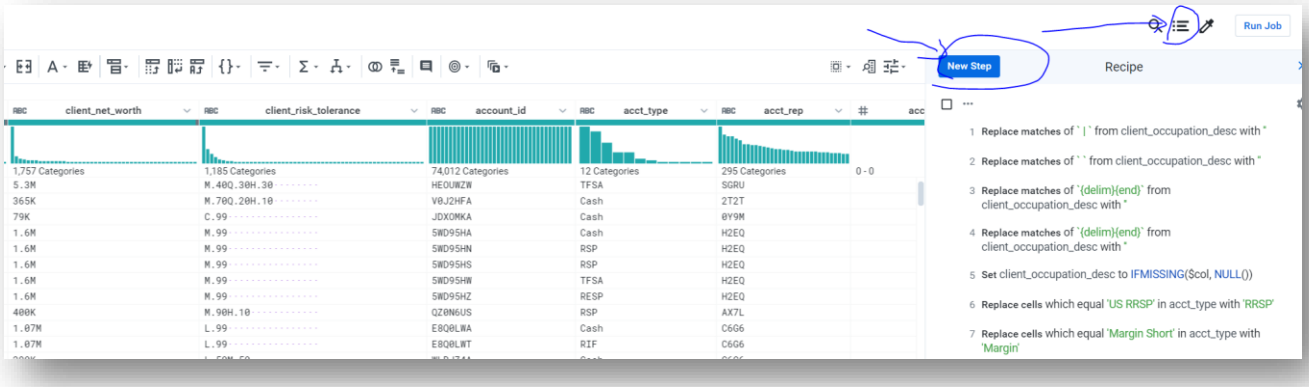
Lecture: Dataprep Demo Part 2

This is the second portion of our fairly extensive Dataprep demo.

Suggestion



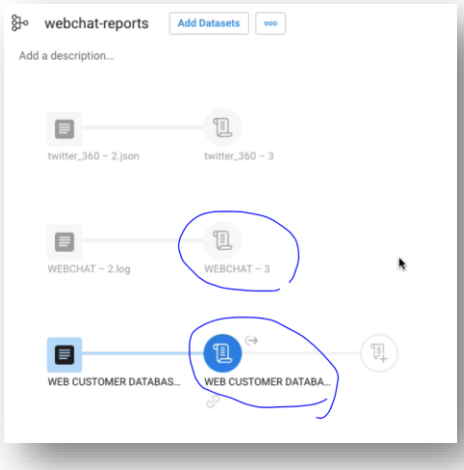
Manual recipe

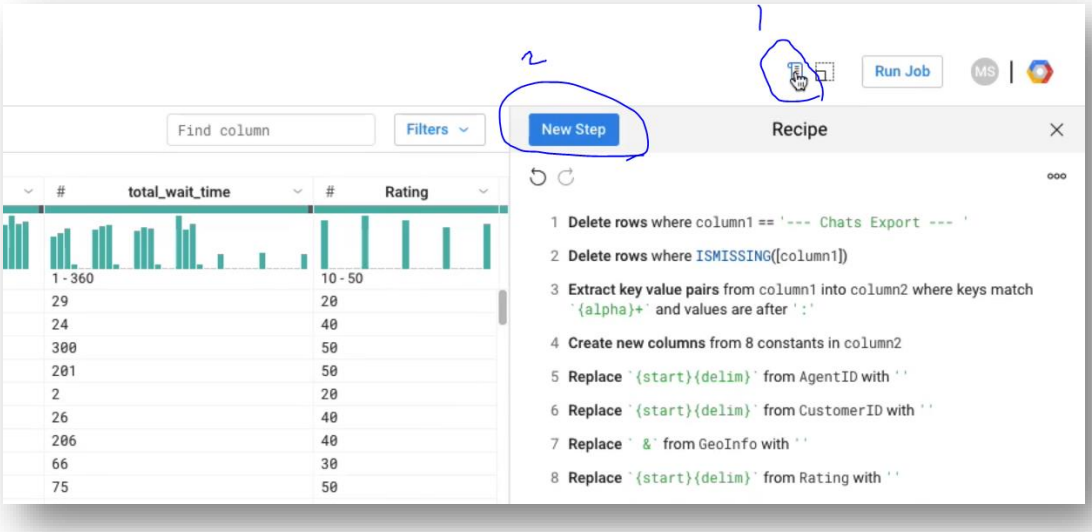


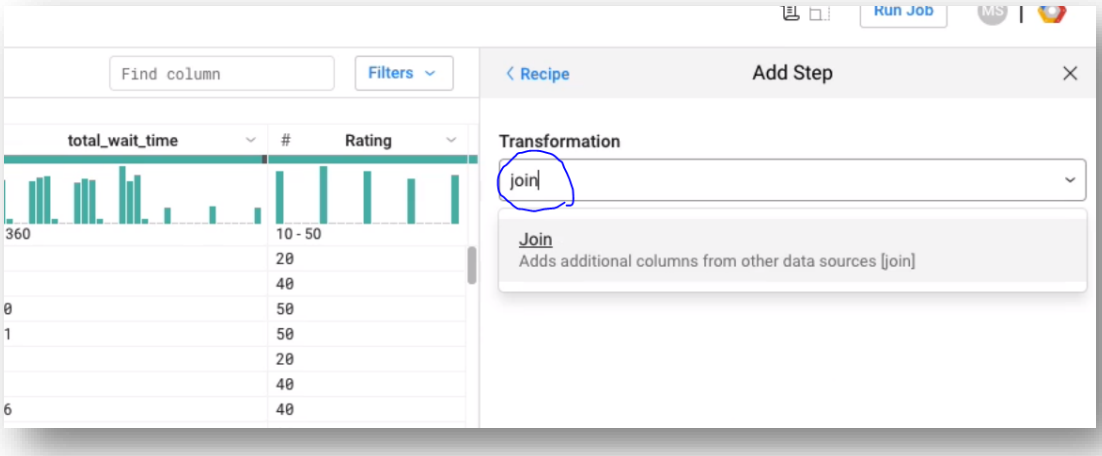
Lecture: Dataprep Demo Part 3

This is the third part of our Dataprep demo. In this lesson, we will take our joined datasets, and start a managed Dataflow job, which we will then insert into a BigQuery table.

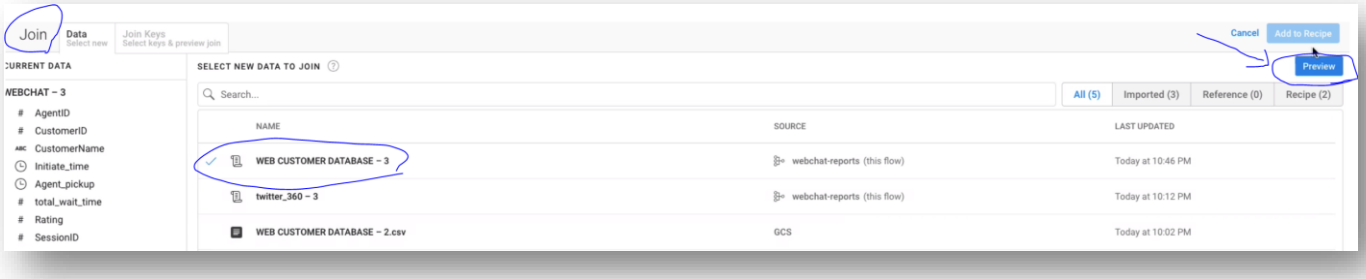
Join all the processed datasets together



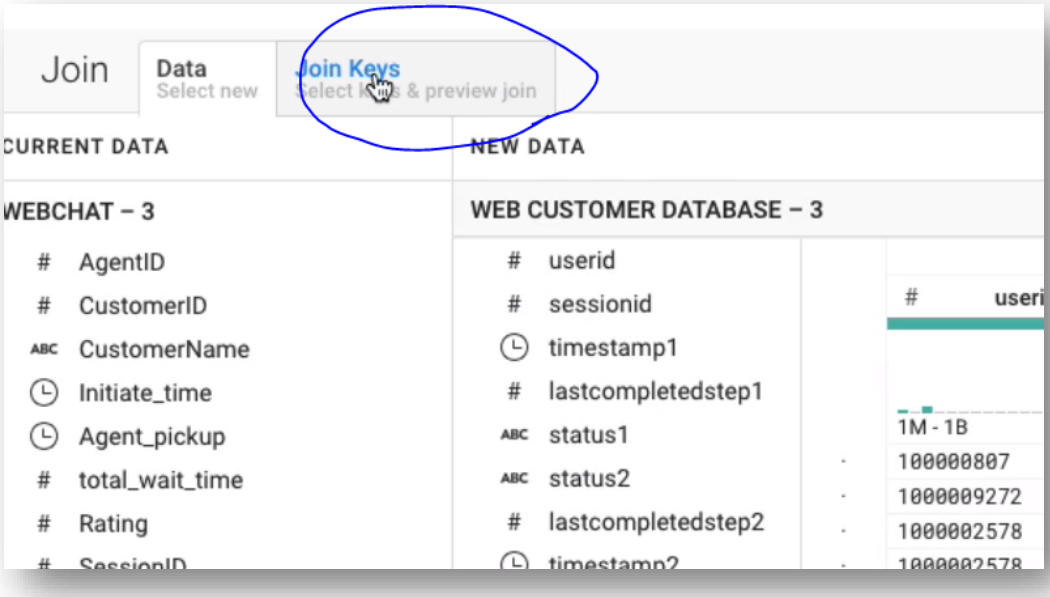




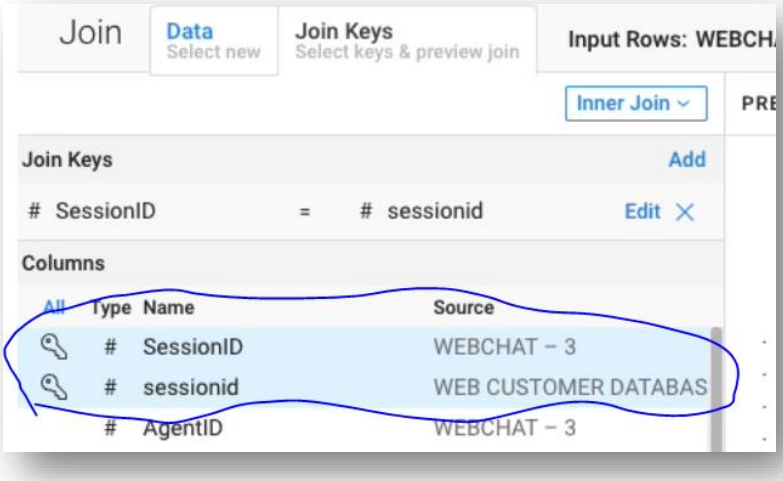
Then it goes to a new screen page:



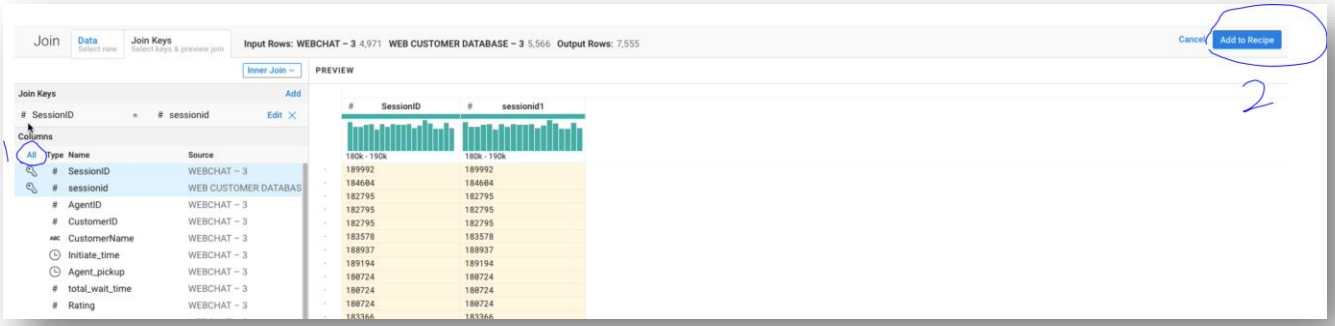
Next, set up the join key:



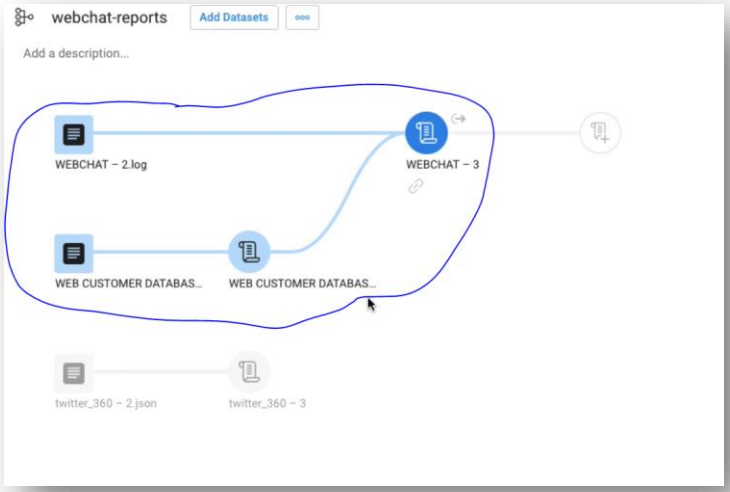
Automatically identify the joining keys for both datasets:



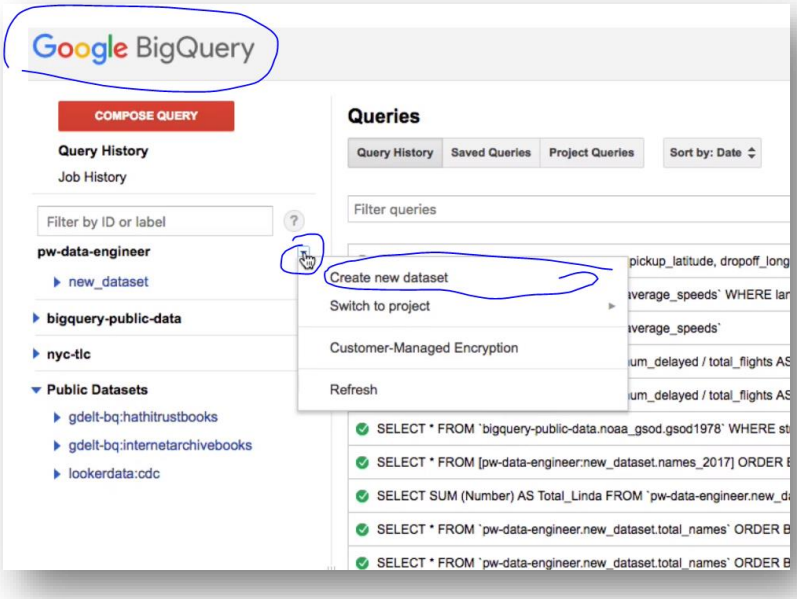
Click ‘all’ to add those different columns together:



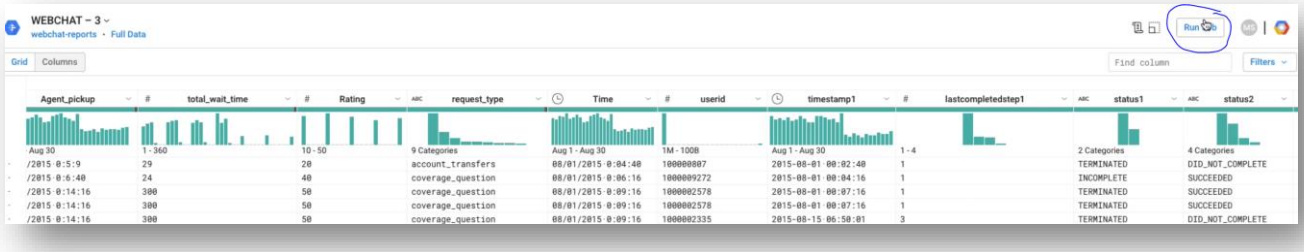
Now we have more data to work with:



Now move data to BigQuery:

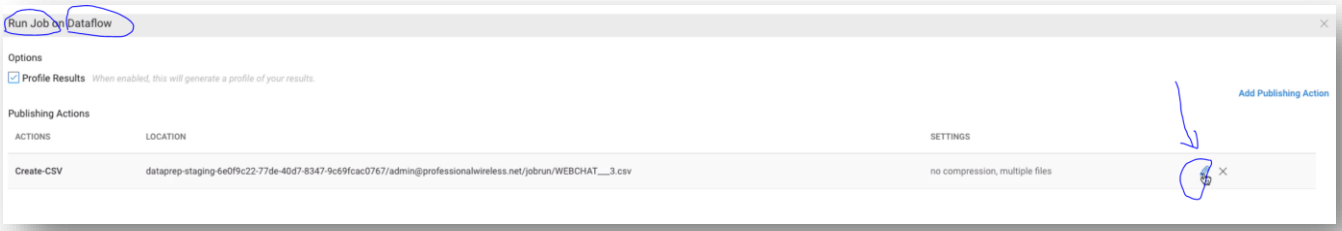


Back to table, click on ‘run job’:

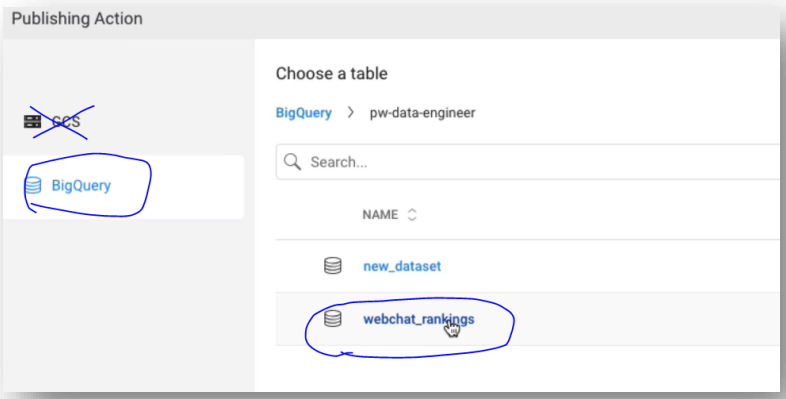


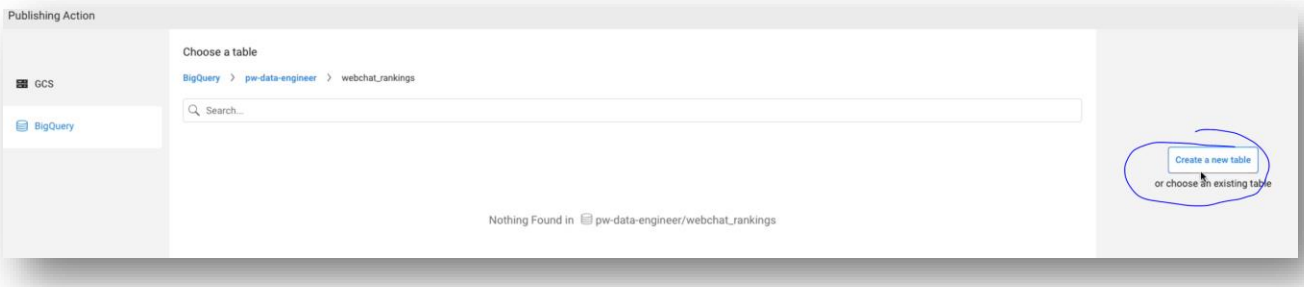
By default, it will create CSV file and will move to GCS. But this time, we don’t want the default since we need to move to BigQuery.

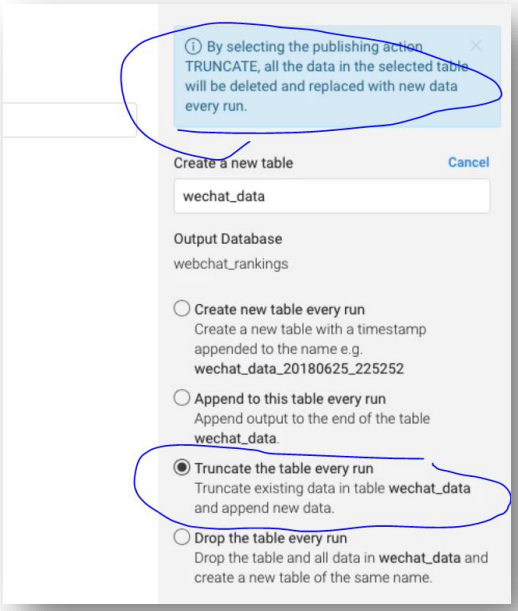
Change the default option:

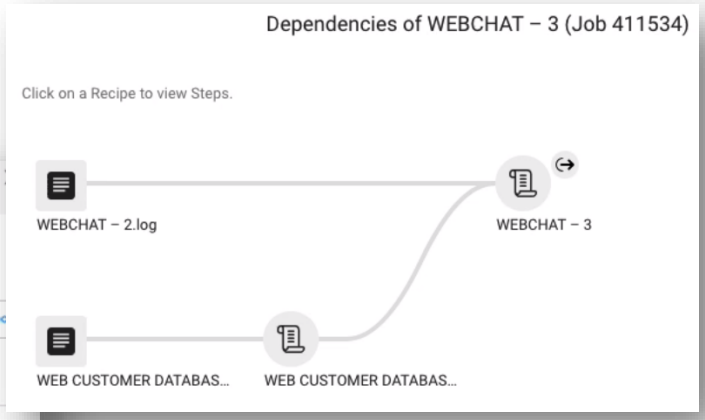


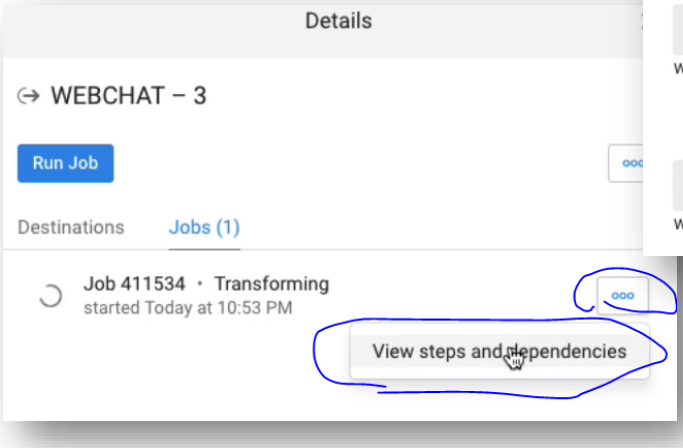
Choose **BigQuery**, create a new table, **save setting**, and click **run job**:

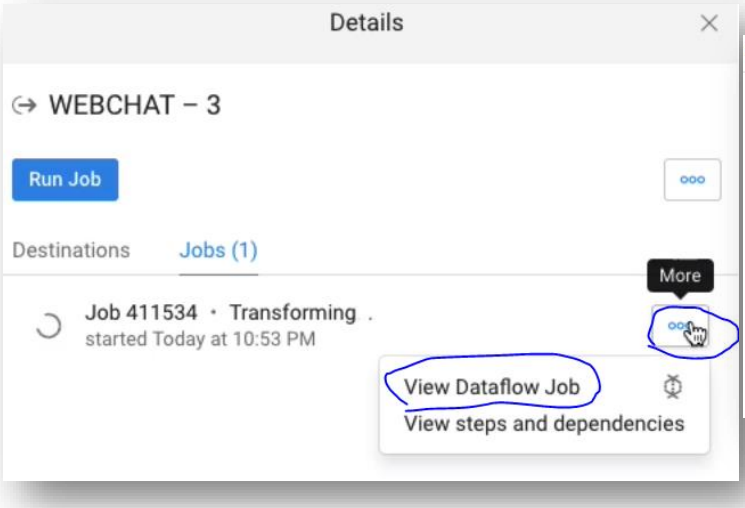


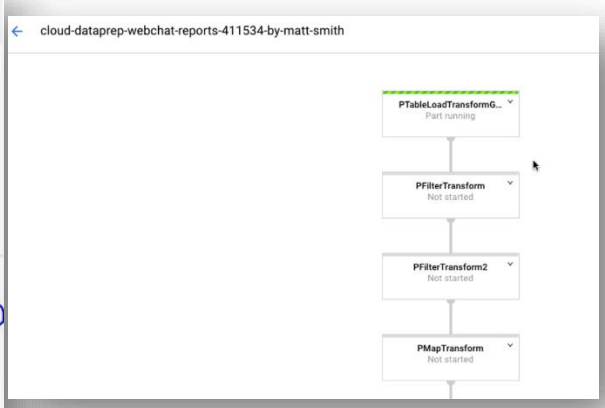



By selecting the publishing action TRUNCATE, all the data in the selected table will be deleted and replaced with new data every run.
Create a new table
Cancel
wechat_data
Output Database
webchat_rankings
Create new table every run
Create a new table with a timestamp appended to the name e.g. wechat_data_20180625_225252
Append to this table every run
Append output to the end of the table wechat_data.
Truncate the table every run
Truncate existing data in table wechat_data and append new data.
Drop the table every run
Drop the table and all data in wechat_data and create a new table of the same name.

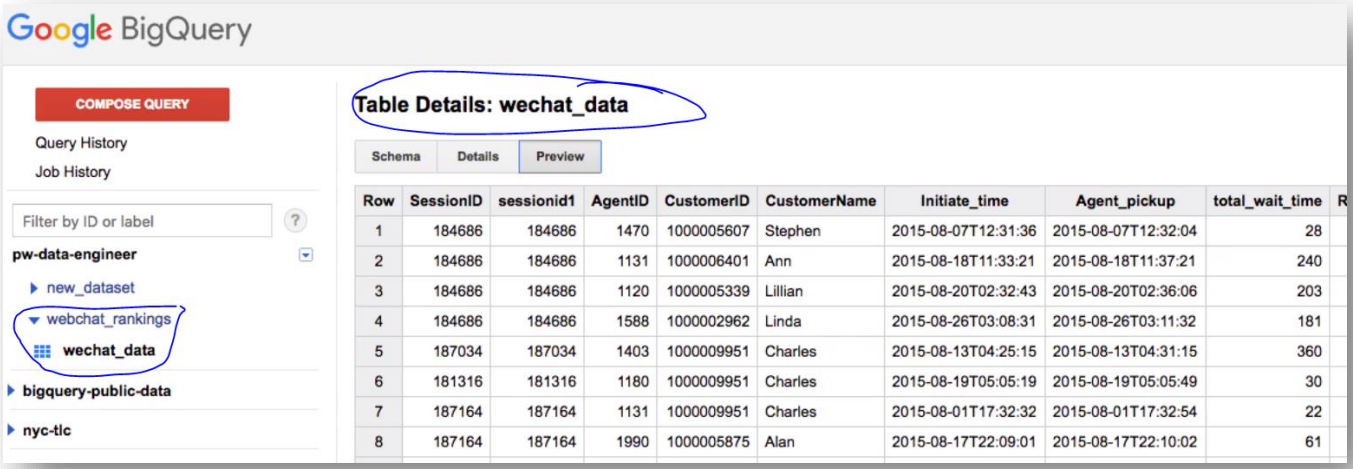

Dependencies of WEBCHAT – 3 (Job 411534)
Click on a Recipe to view Steps.
WEBCHAT – 2.log
WEBCHAT – 3
WEB CUSTOMER DATABAS...
WEB CUSTOMER DATABAS...


Details
WEBCHAT – 3
Run Job
Destinations
Jobs (1)
Job 411534 · Transforming
started Today at 10:53 PM
View steps and dependencies


Details
WEBCHAT – 3
Run Job
Destinations
Jobs (1)
Job 411534 · Transforming
started Today at 10:53 PM
More
View Dataflow Job
View steps and dependencies


cloud-dataprep-webchat-reports-411534-by-matt-smith
PTableLoadTransformG...
Part running
PFilterTransform
Not started
PFilterTransform2
Not started
PMapTransform
Not started

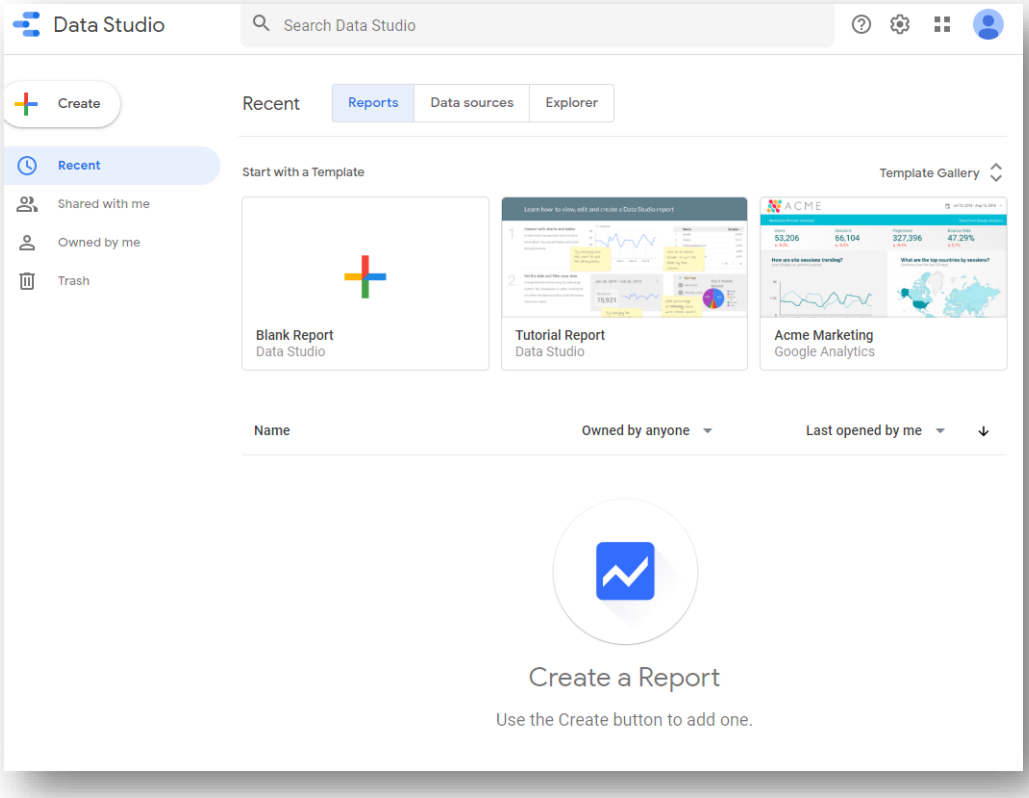
Done:



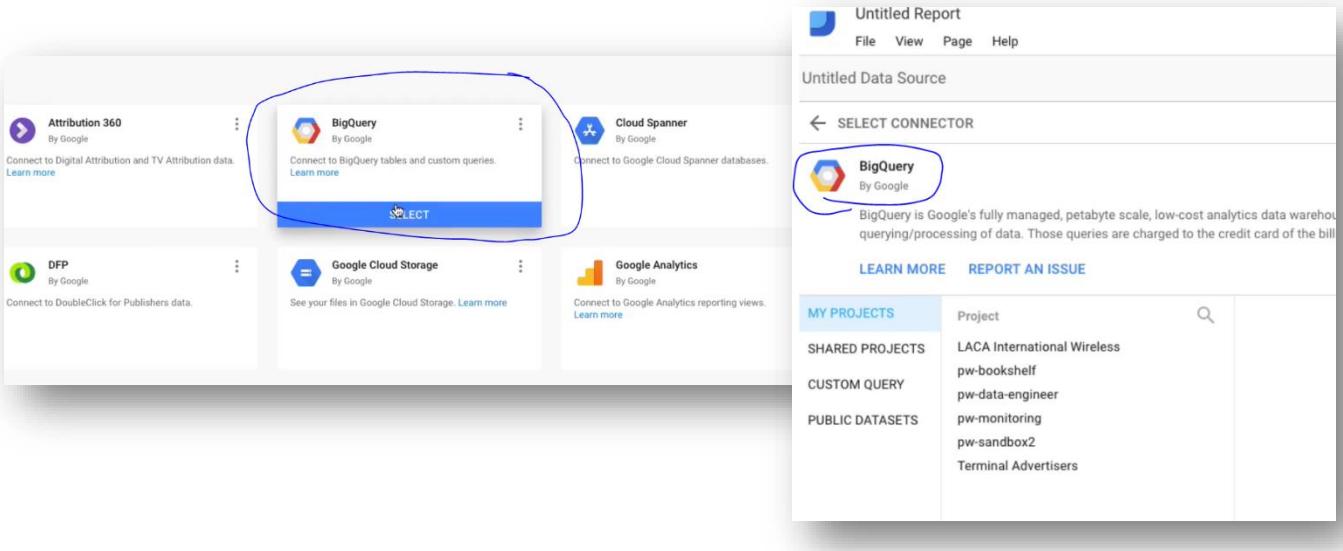
Row	SessionID	sessionid1	AgentID	CustomerID	CustomerName	Initiate_time	Agent_pickup	total_wait_time	R
1	184686	184686	1470	1000005607	Stephen	2015-08-07T12:31:36	2015-08-07T12:32:04	28	
2	184686	184686	1131	1000006401	Ann	2015-08-18T11:33:21	2015-08-18T11:37:21	240	
3	184686	184686	1120	1000005339	Lillian	2015-08-20T02:32:43	2015-08-20T02:36:06	203	
4	184686	184686	1588	1000002962	Linda	2015-08-26T03:08:31	2015-08-26T03:11:32	181	
5	187034	187034	1403	1000009951	Charles	2015-08-13T04:25:15	2015-08-13T04:31:15	360	
6	181316	181316	1180	1000009951	Charles	2015-08-19T05:05:19	2015-08-19T05:05:49	30	
7	187164	187164	1131	1000009951	Charles	2015-08-01T17:32:32	2015-08-01T17:32:54	22	
8	187164	187164	1990	1000005875	Alan	2015-08-17T22:09:01	2015-08-17T22:10:02	61	

Lecture: Data Studio Demo

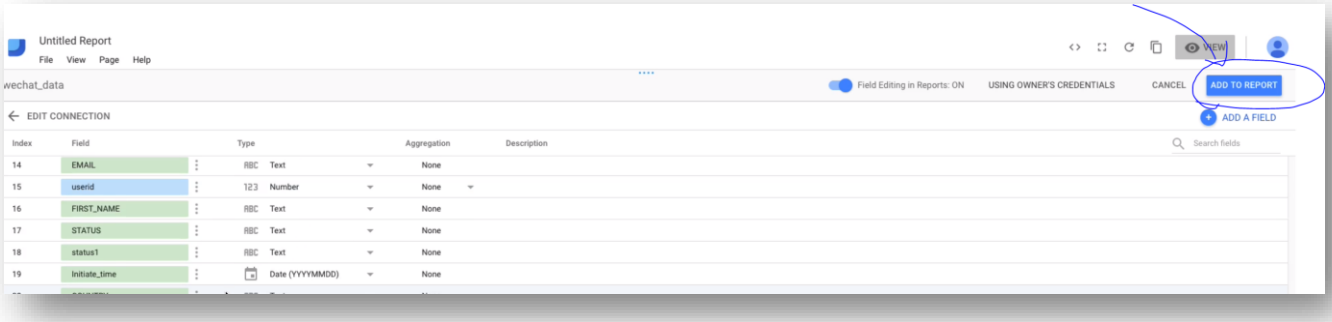
In this lesson, we will demonstrate working with Data Studio. We will take the cleaned Dataprep document from the previous section which we exported to BigQuery, and use that data source to get insights about our ratings.



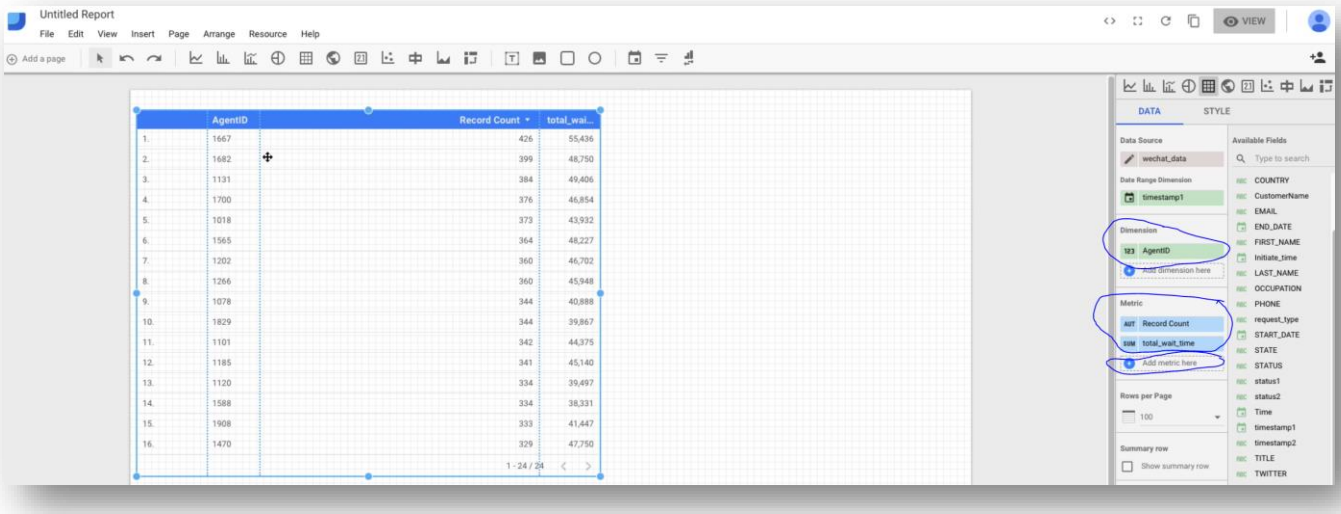
Can choose GCS or SQL



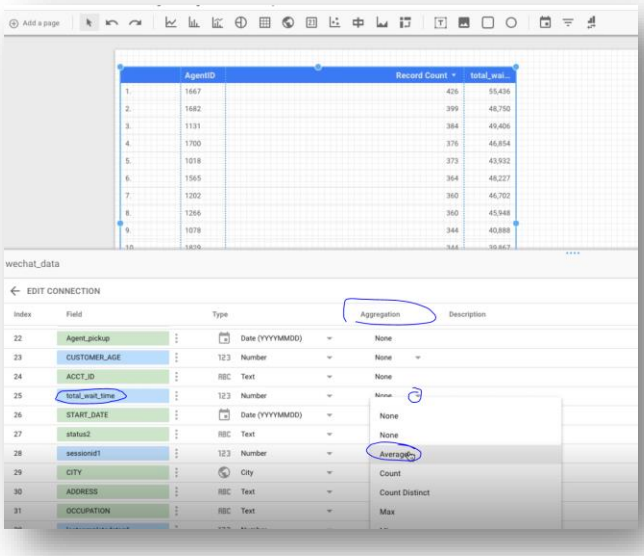
Detect the data type automatically, and click ‘add to report’

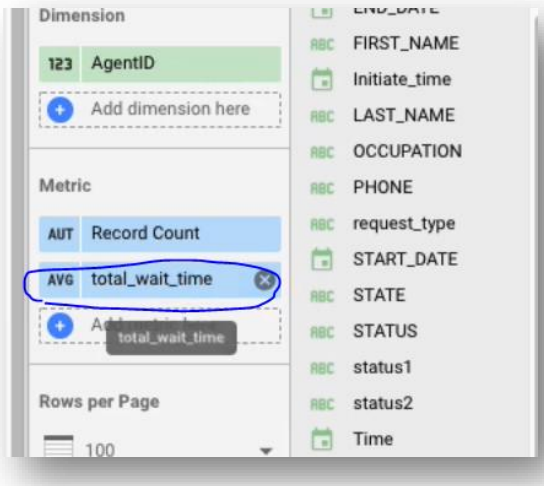


Make a table, click and drag

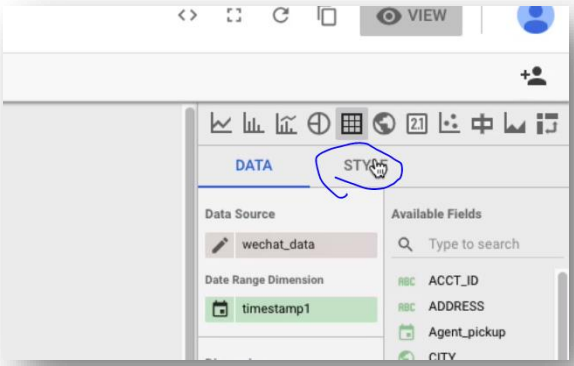


Since ‘adding’ is the default calculation. Need to create a new field (to use aggregation), click ‘create new field’ at the bottom. Go to the field (column) we need to aggregate. Re-add that field to the ‘metric’.

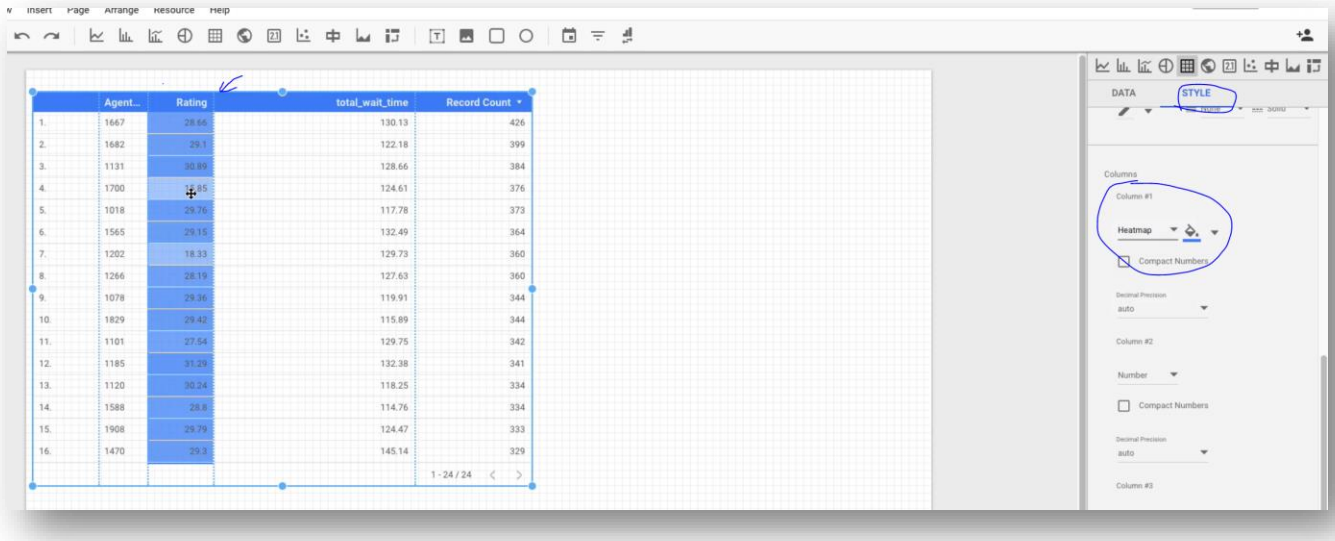




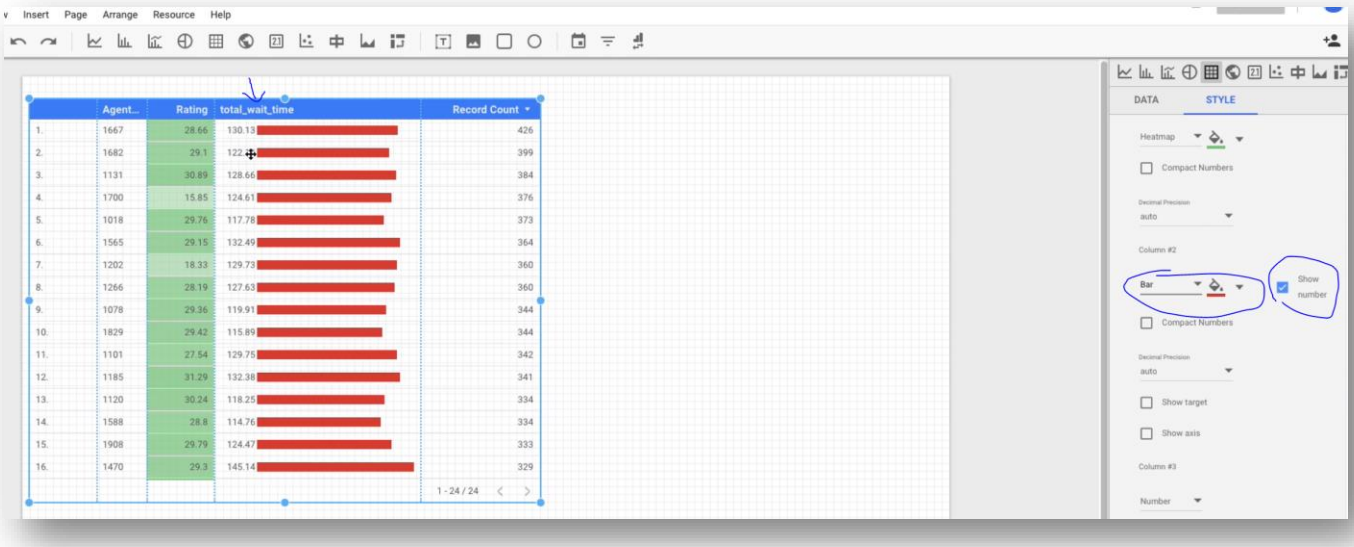
Change the table **style**



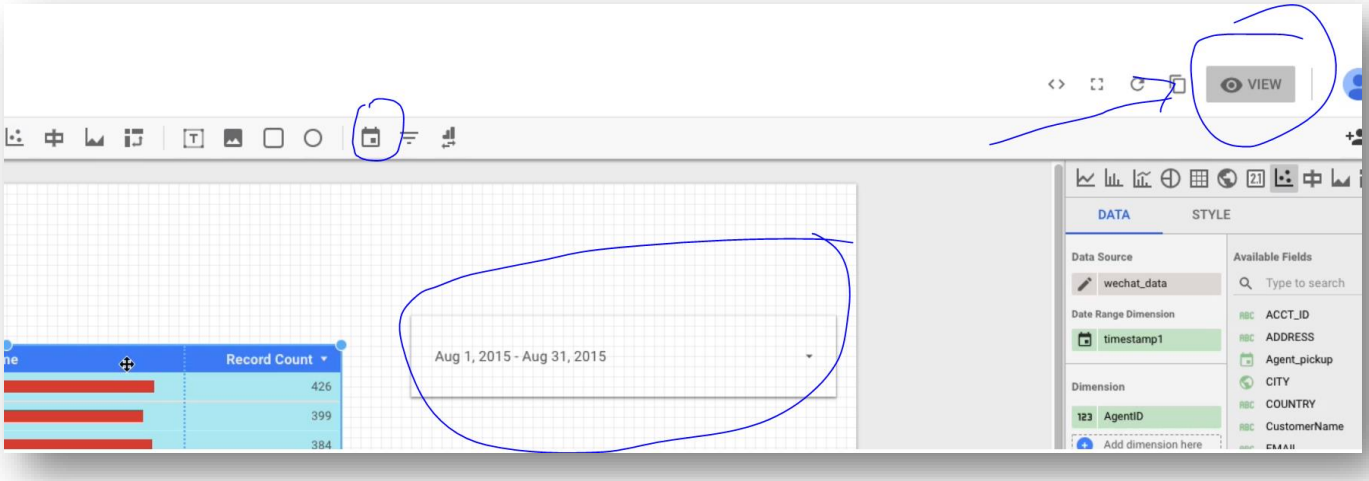
Change column to **heatmap**



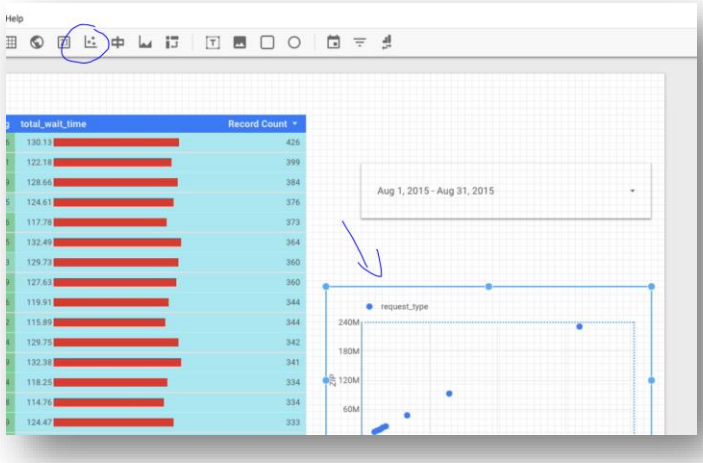
Change column to **bar (with showing the number aside)**



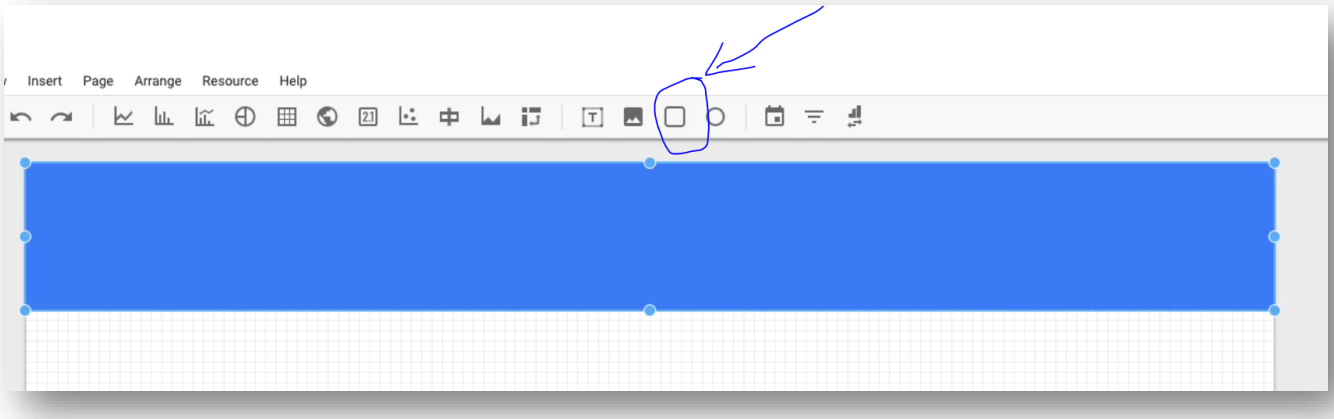
Add **time filter**, **click and drag. And use the view mode to check it out**



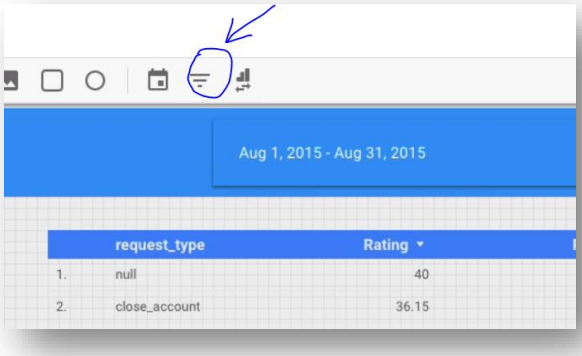
Make **scatter** plot

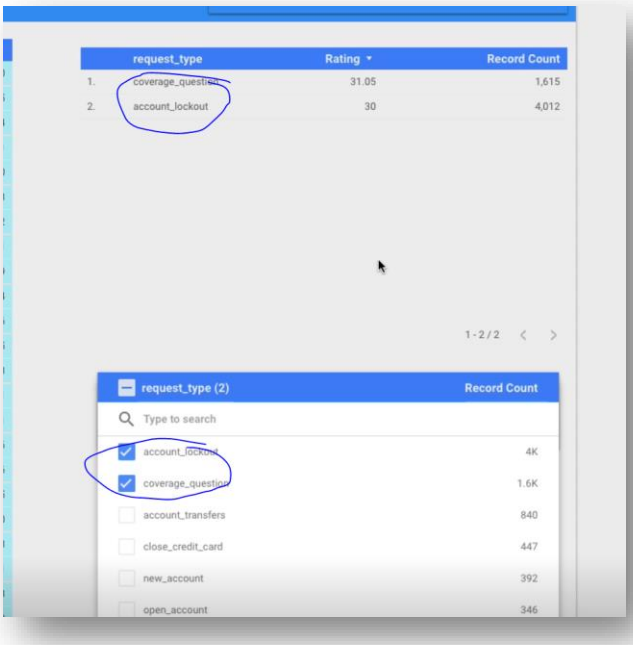


Make **report title**



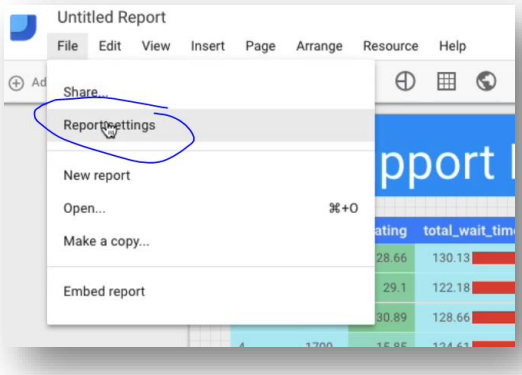
Add **control filters**

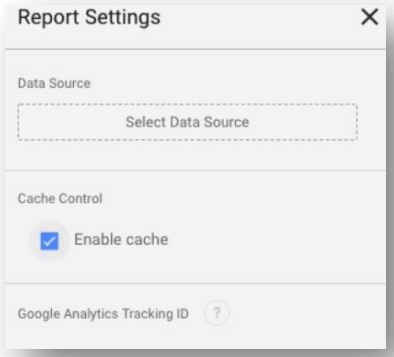




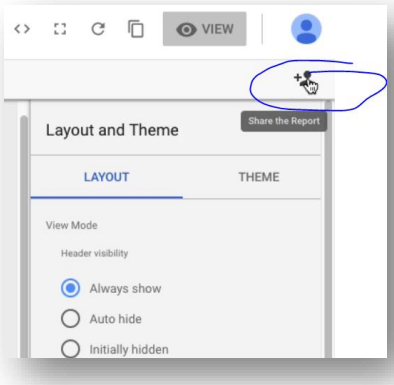
Finally, **report setting** (for sharing)

If the table need to change frequently, untick ‘enable cache’. Otherwise, just leave it.





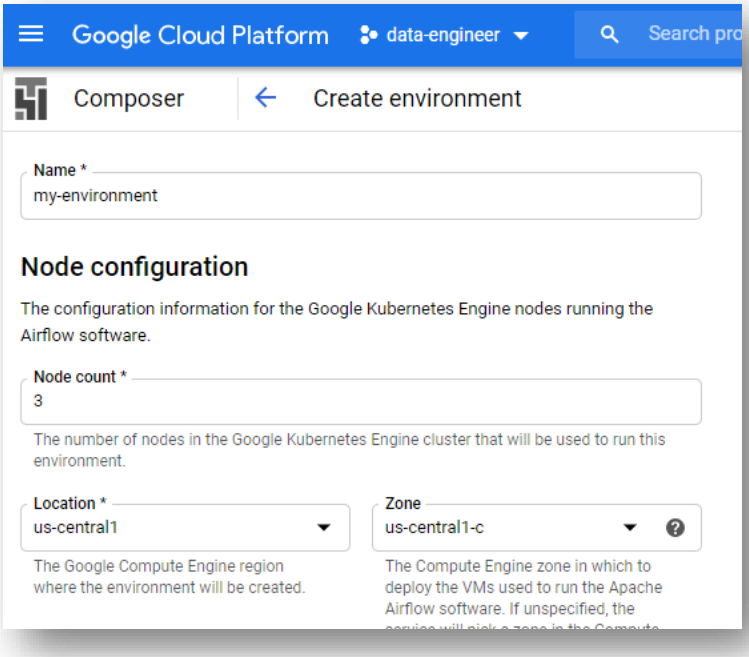
Share the report to people



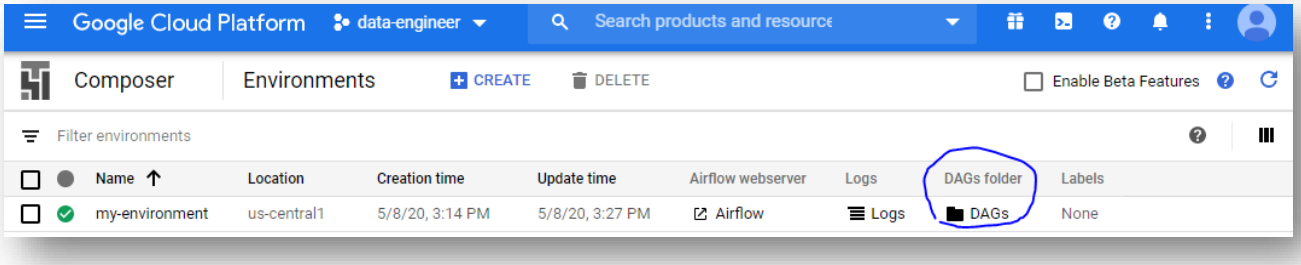
Lecture: Hands On - Cloud Composer

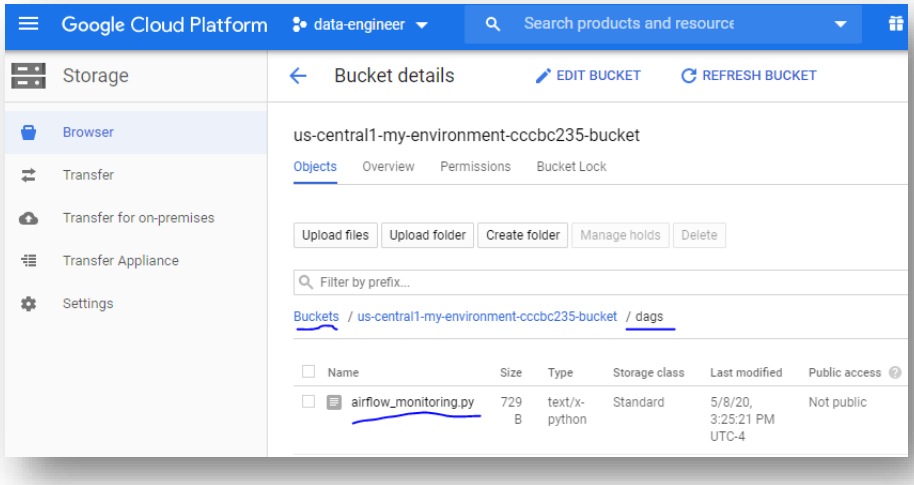
This lesson will be a hands on tour of Cloud Composer in action. The commands we used in this lesson are duplicated below:

Step1: Enable Composer / Dataproc API **First** **and** Use the default settings:

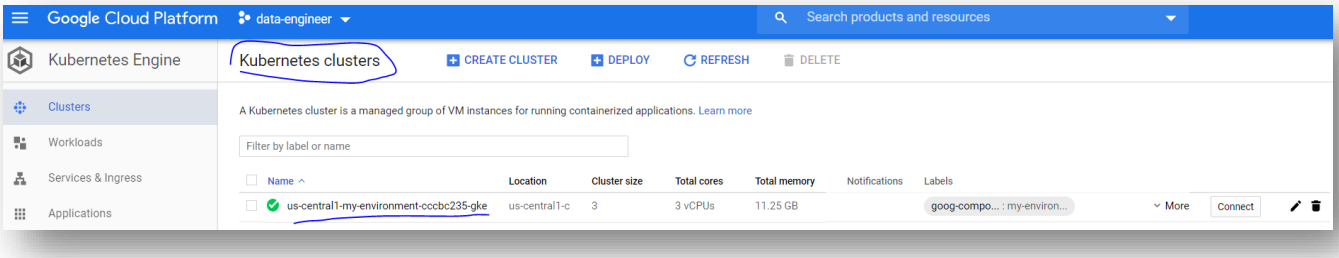


Done, the DAGs folder is a short-cut to storage bucket:



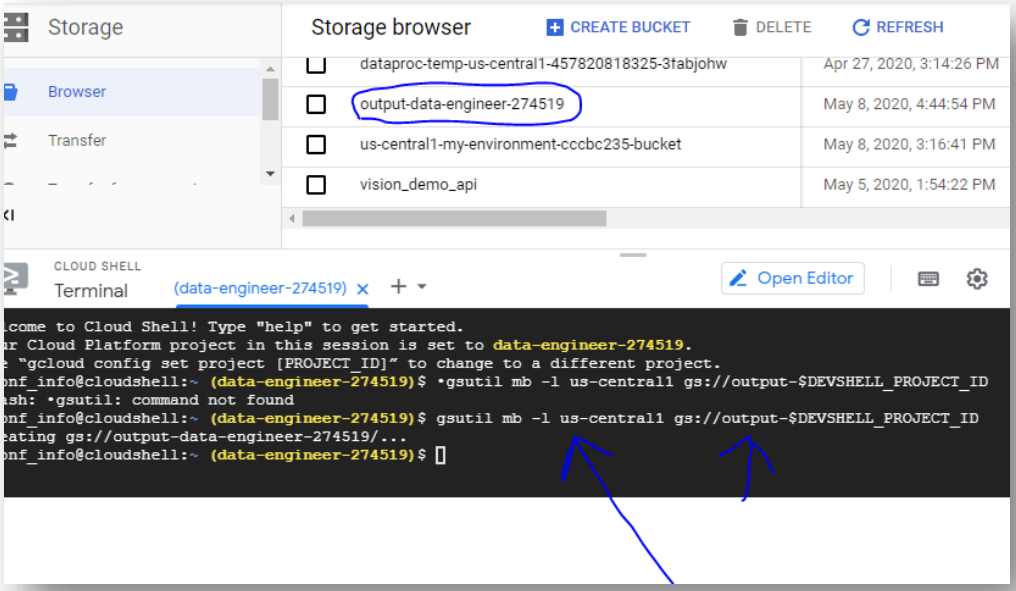


Also Kubernetes cluster is running at the same time:



Step 2: Create another **separate** GCS bucket to output **Dataproc** results (using Project ID shell variable in composer shell environment)

- gsutil mb -l us-central1 gs://output-$DEVSHELL_PROJECT_ID



Step 3: Before the workflow execute, set our Cloud Composer variables necessary.

Format

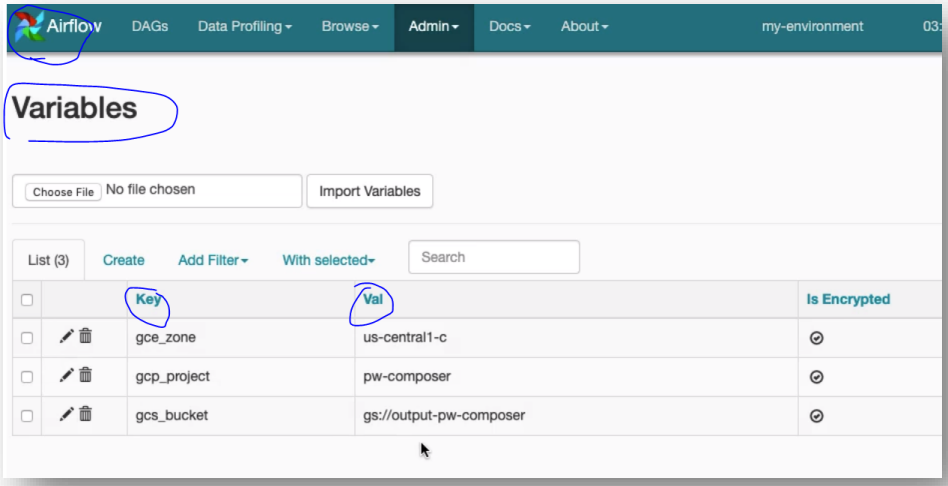
- **gcloud composer environments run (ENVIRONMENT_NAME) --location (LOCATION) variables -- --set (KEY VALUE)**
- **KEY VALUE pair example**: gcp_project $DEVSHELL_PROJECT_ID

Project ID will again be represented by the shell variable to auto-resolve to your unique Project ID:

- **gcloud composer environments run** my-environment --location us-central1 variables -- --set gcp_project $DEVSHELL_PROJECT_ID (set variable for project ID)
- **gcloud composer environments run** my-environment --location us-central1 variables -- --set gcs_bucket gs://output-$DEVSHELL_PROJECT_ID (set variable for cloud storage bucket)

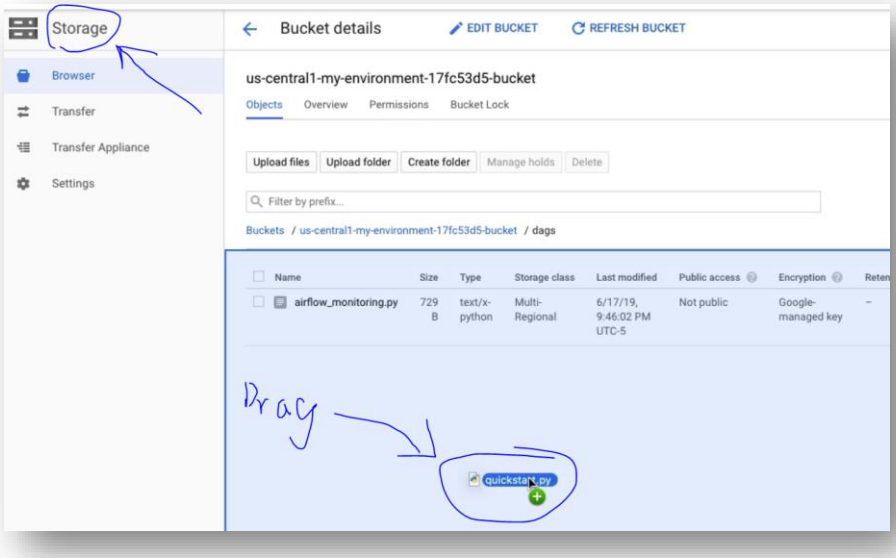
- **gcloud composer environments** run my-environment --location us-central1 variables -- --set gce_zone us-central1-c (set variable for zone)

```
leonf_info@cloudshell:~ (data-engineer-274519)$ gcloud composer environments run my-environment --location us-central1 variables -- --se
t gcs_bucket gs://output-$DEVSHELL_PROJECT_ID
kubeconfig entry generated for us-central1-my-environment-cccbc235-gke.
Executing within the following Kubernetes cluster namespace: composer-1-10-2-airflow-1-10-3-cccbc235
[2020-05-09 17:08:05,812] {settings.py:185} INFO - settings.configure_orm(): Using pool settings. pool_size=5, pool_recycle=1800, pid=10
043
[2020-05-09 17:08:06,170] {default_celery.py:90} WARNING - You have configured a result_backend of redis://airflow-redis-service.default
.svc.cluster.local:6379/0, it is highly recommended to use an alternative result_backend (i.e. a database).
[2020-05-09 17:08:06,175] {__init__.py:51} INFO - Using executor CeleryExecutor
[2020-05-09 17:08:06,584] {app.py:54} WARNING - Using default Composer Environment Variables. Overrides have not been applied.
[2020-05-09 17:08:06,599] {configuration.py:556} INFO - Reading the config from /etc/airflow/airflow.cfg
[2020-05-09 17:08:06,620] {settings.py:185} INFO - settings.configure_orm(): Using pool settings. pool_size=5, pool_recycle=1800, pid=10
043
```

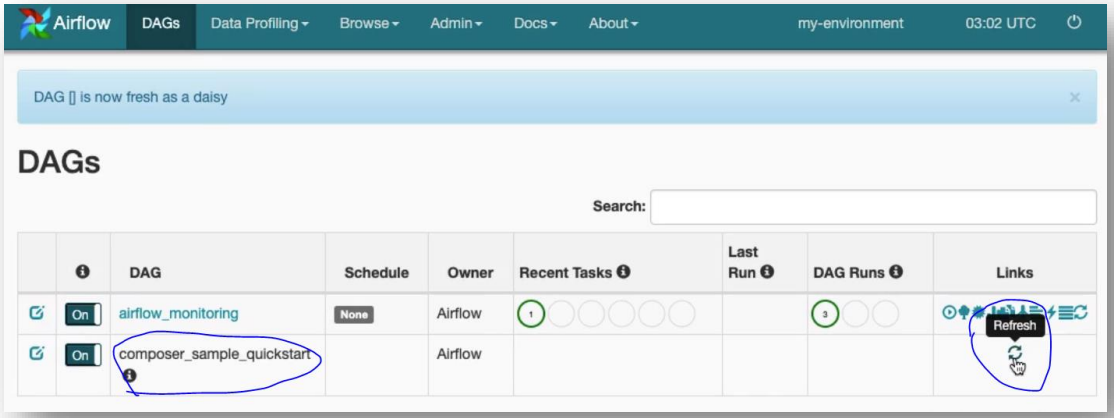


Step 4: Upload the example DAG file (Python file) to the DAG folder for Cloud Composer. A copy of the DAG file can be found at this link.

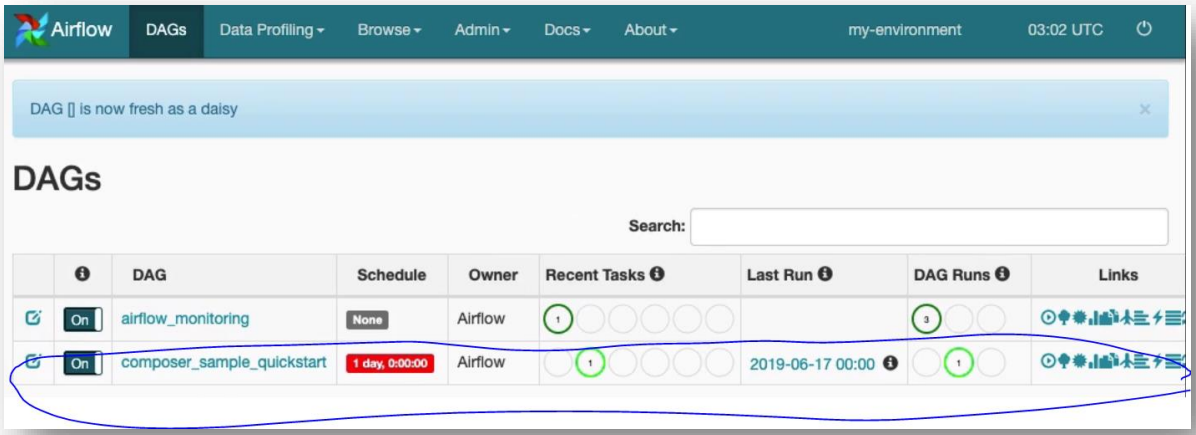
- Direct link for local download: https://storage.googleapis.com/la-gcloud-course-resources/data-engineer/cloud-composer/quickstart.py
- Public GCS location: gs://la-gcloud-course-resources/data-engineer/cloud-composer/quickstart.py
- GitHub link: https://github.com/GoogleCloudPlatform/python-docs-samples/blob/b80895ed88ba86fce223df27a48bf481007ca708/composer/workflows/quickstart.py
- Or if we have download the file to local directory, just drag it to storage bucket:



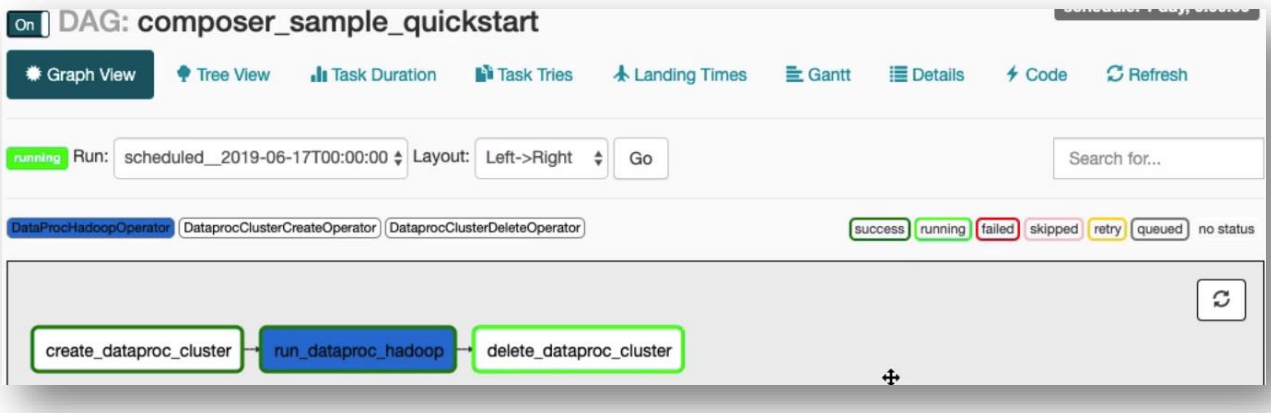
Cloud storage will automatically detect the quickstart.py:



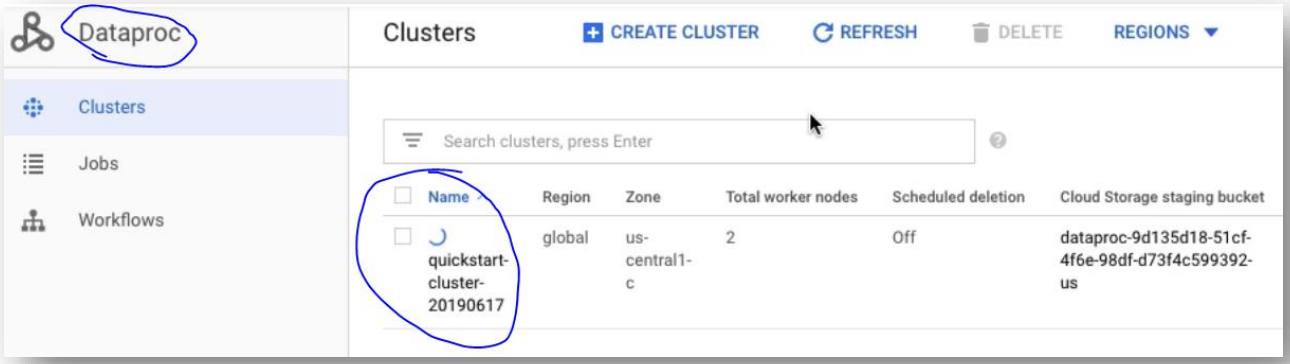
Refresh it, it shows it is processing:



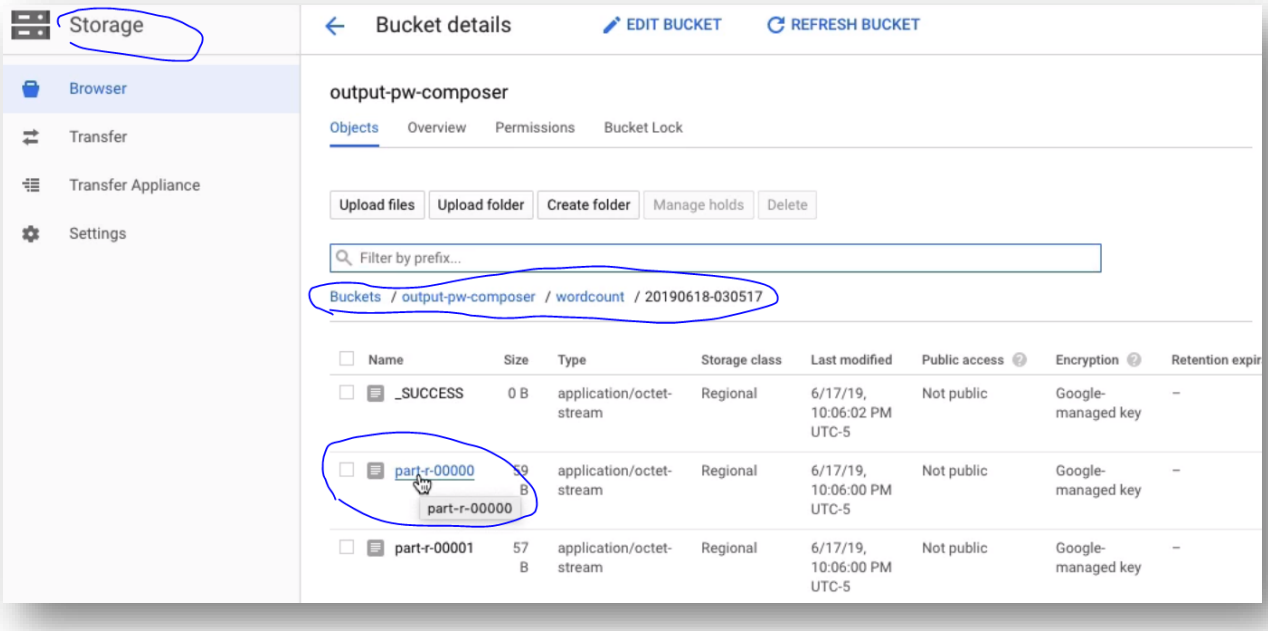
Doing the Hadoop job, and delete cluster after job done:



It create a dataproc cluster and do the work right now:



View result in bucket:



Delete composer instance and storage bucket!!!

Below are additional resources directly from Google that I highly recommend working with for even more hands-on practice:

Official Google Cloud Data Engineer practice exam:

https://cloud.google.com/certification/practice-exam/data-engineer

Google Cloud Solutions Center:

https://cloud.google.com/solutions/

Google **Codelabs** for hands-on practice across all products:

https://codelabs.developers.google.com/

Big Data and Machine Learning Blog:

https://cloud.google.com/blog/big-data/

BigQuery Tutorials:

https://cloud.google.com/bigquery/docs/tutorials

Dataflow Tutorials:

https://cloud.google.com/dataflow/examples/examples-beam

Dataproc samples and tutorials:

https://cloud.google.com/dataproc/docs/tutorials

Pub/Sub totorials:

https://cloud.google.com/pubsub/docs/tutorials

Cloud ML Engine tutorials:

https://cloud.google.com/ml-engine/docs/tensorflow/tutorials

Bigtable tutorials and samples:

https://cloud.google.com/bigtable/docs/samples

Cloud Spanner tutorials:

https://cloud.google.com/spanner/docs/tutorials

Additional study resources:
SQL deep dive
Course - SQL Primer
https://linuxacademy.com/cp/modules/view/id/52
Machine Learning
Google Machine Learning Crash Course (free)
https://developers.google.com/machine-learning/crash-course/
Hadoop
Hadoop Quick Start
https://linuxacademy.com/cp/modules/view/id/294
Apache Beam (Dataflow)
Google's guide to designing your pipeline with Apache Beam (using Java)
https://cloud.google.com/dataflow/docs/guides/beam-creating-a-pipeline
NOTE: The revised version of this course (and exam) as of the March 2019 refresh has the case studies removed.