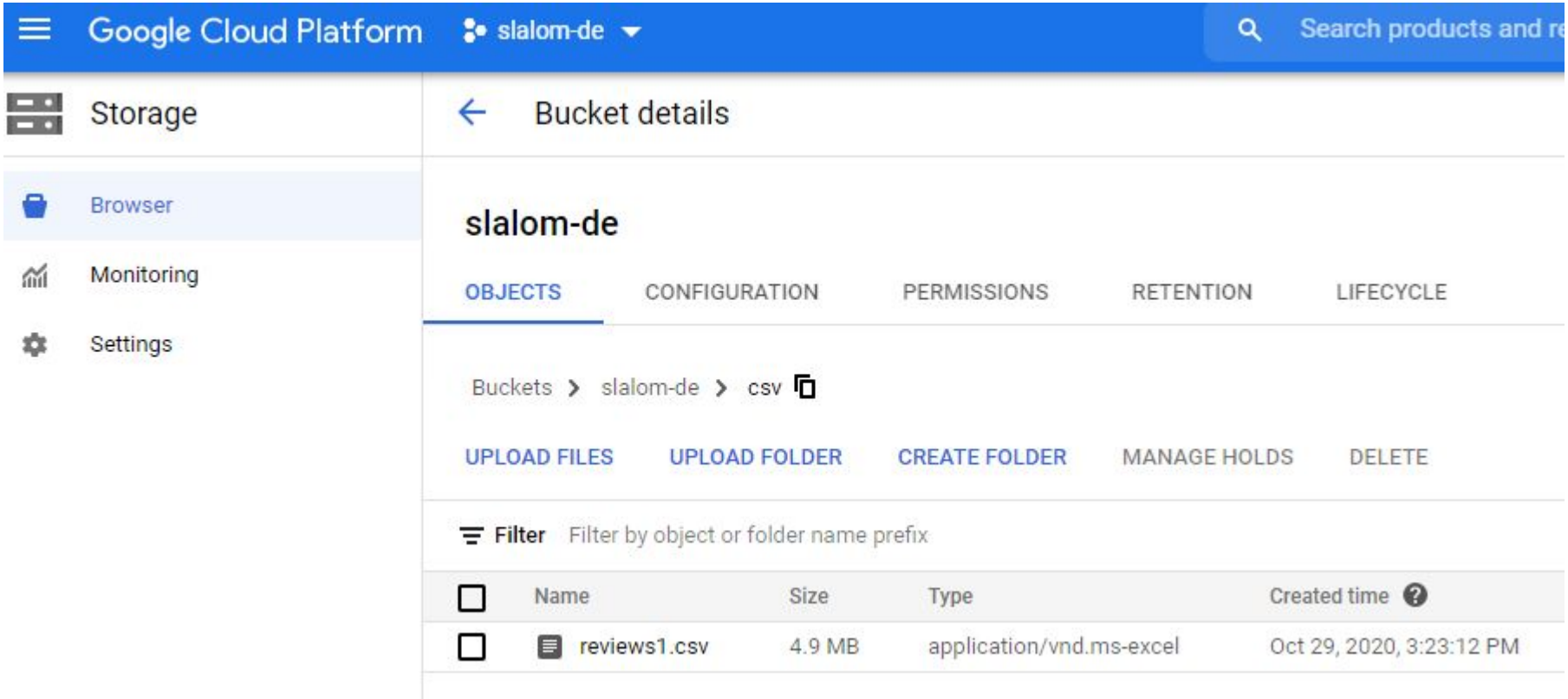
Slalom Data Engineering

Assignment

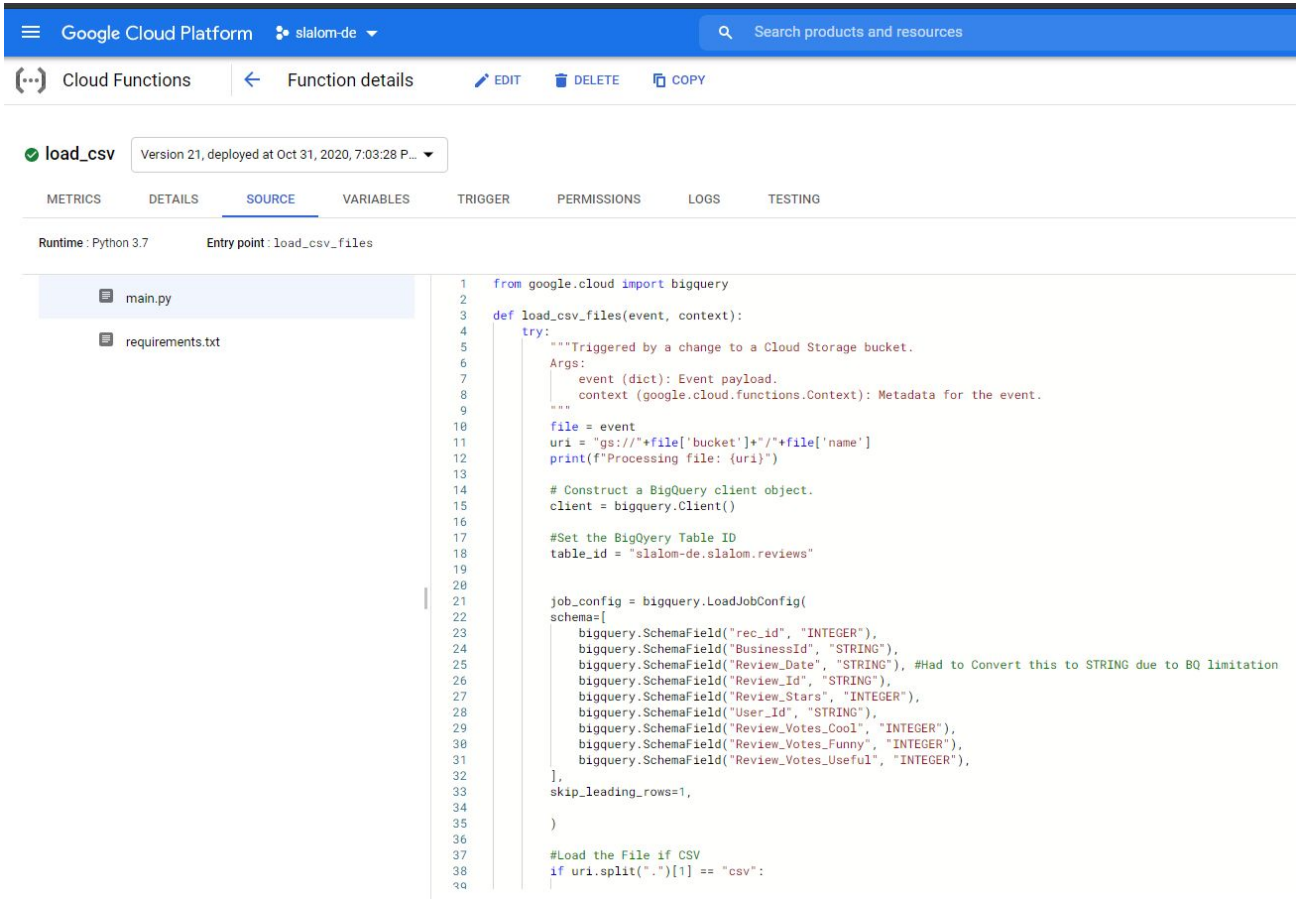
Serverless Event Driven Loads

1. CSV File lands in GCS
2. Cloud Function is Triggered
3. Data loaded to BQ

Snapshot of File load on GCS



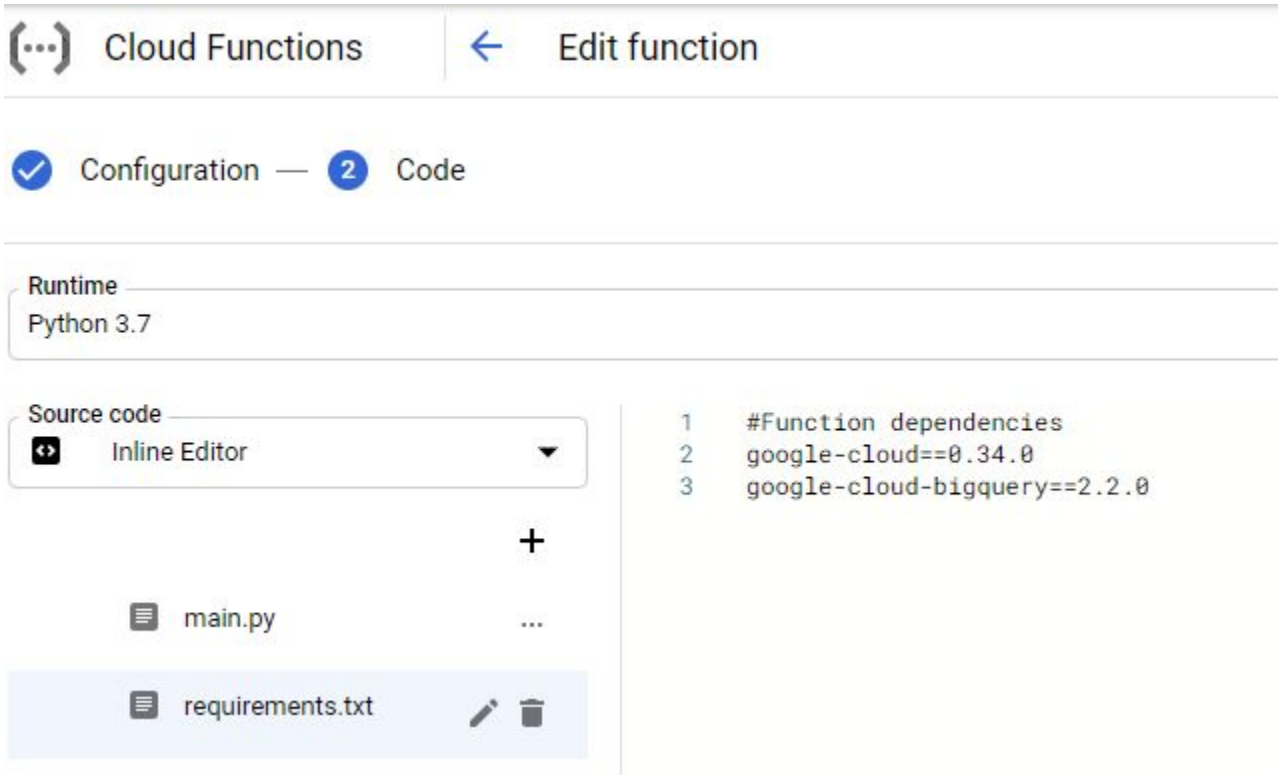
Snapshot of Cloud Function



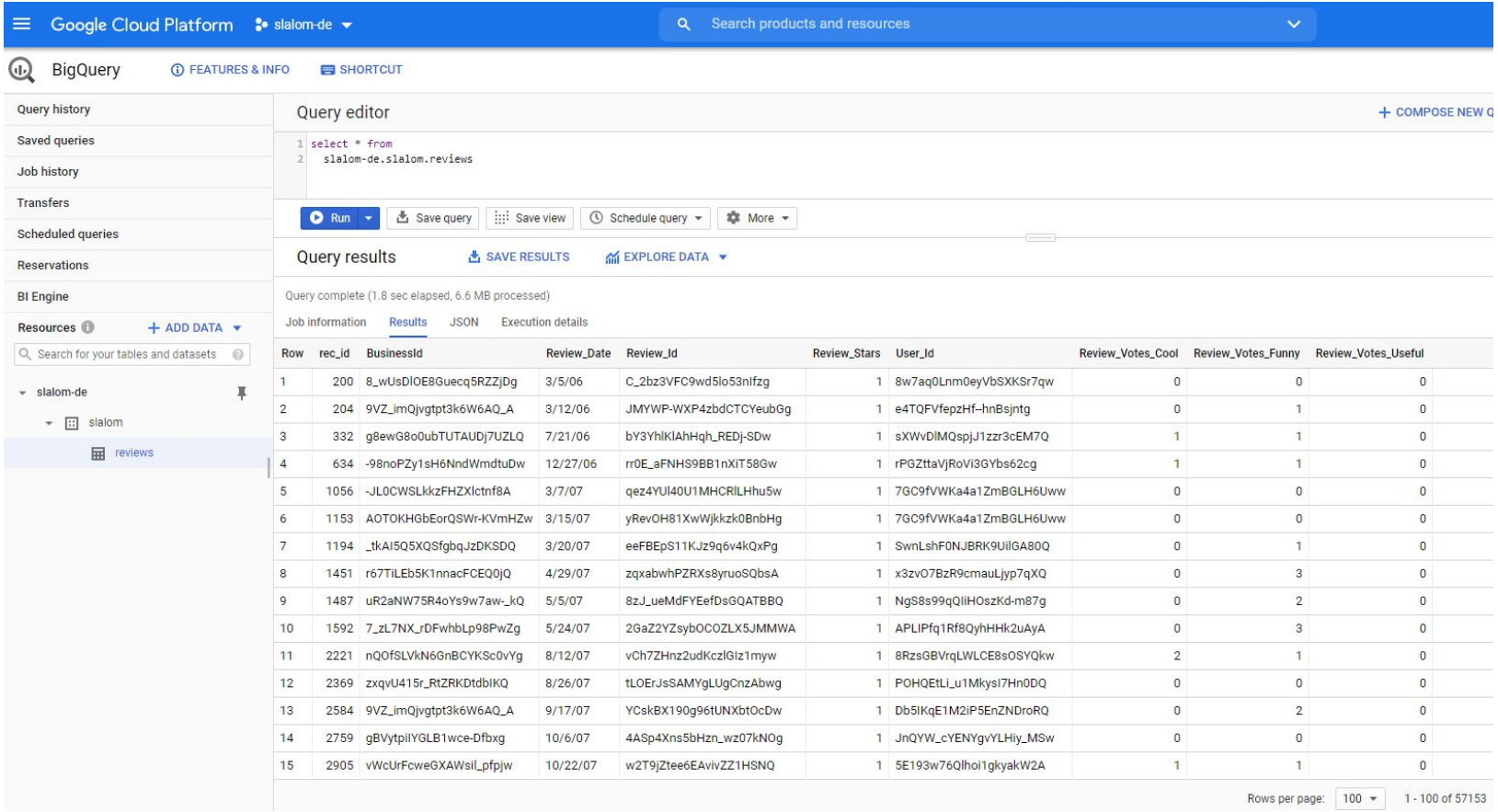
Snapshot of Cloud Function

```
35          )
36
37          #Load the File if CSV
38          if uri.split(".")[1] == "csv":
39
40              load_job = client.load_table_from_uri(
41              uri, table_id, job_config=job_config
42              )
43              load_job.result()  # Get Loaded Results
44              table = client.get_table(table_id)
45              print("Loaded {} rows to table {}".format(table.num_rows, table_id))
46          else:
47              print("File {} is not a CSV file".format(uri))
48      except:
49          print("Something went wrong")
50
```

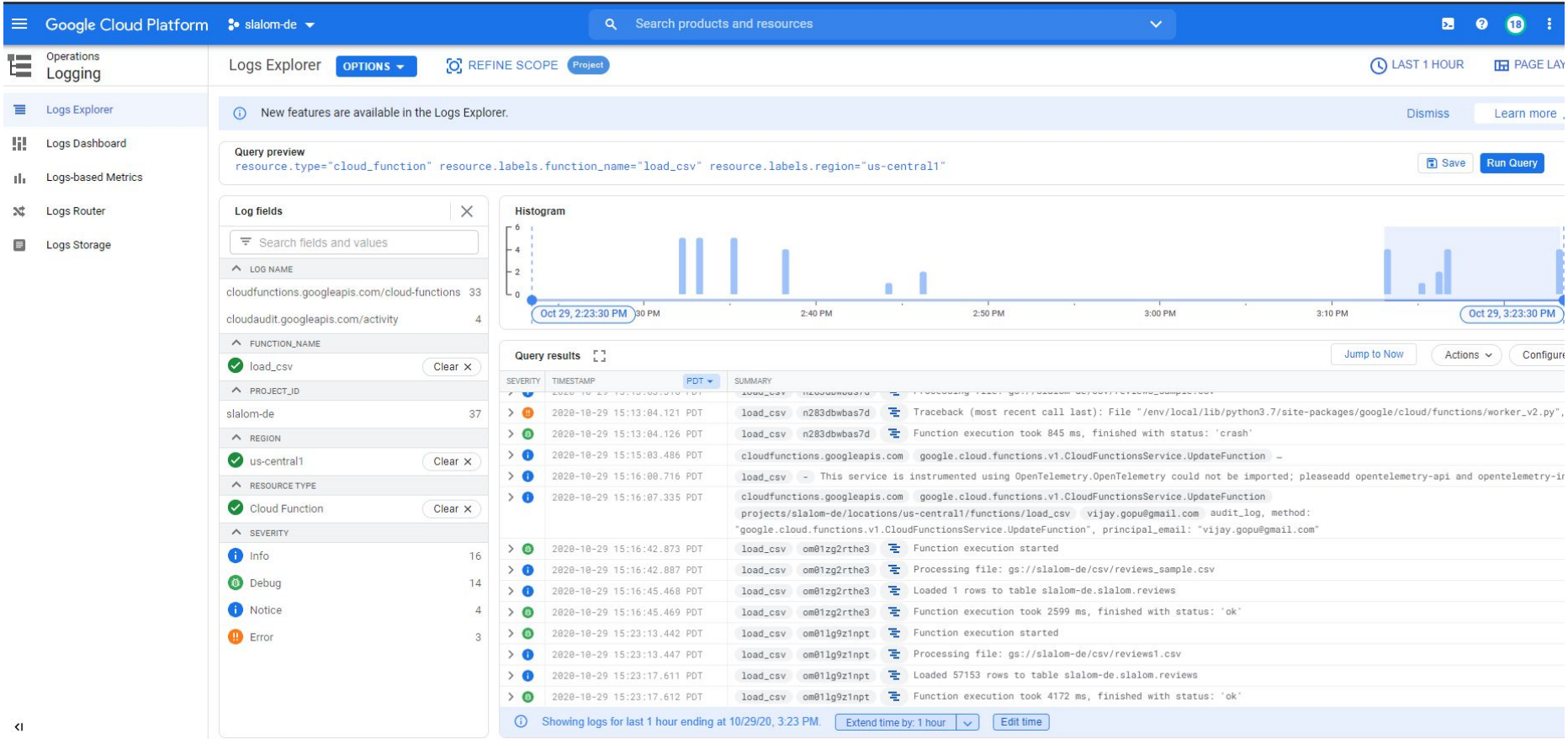
Cloud Function Dependency Packages



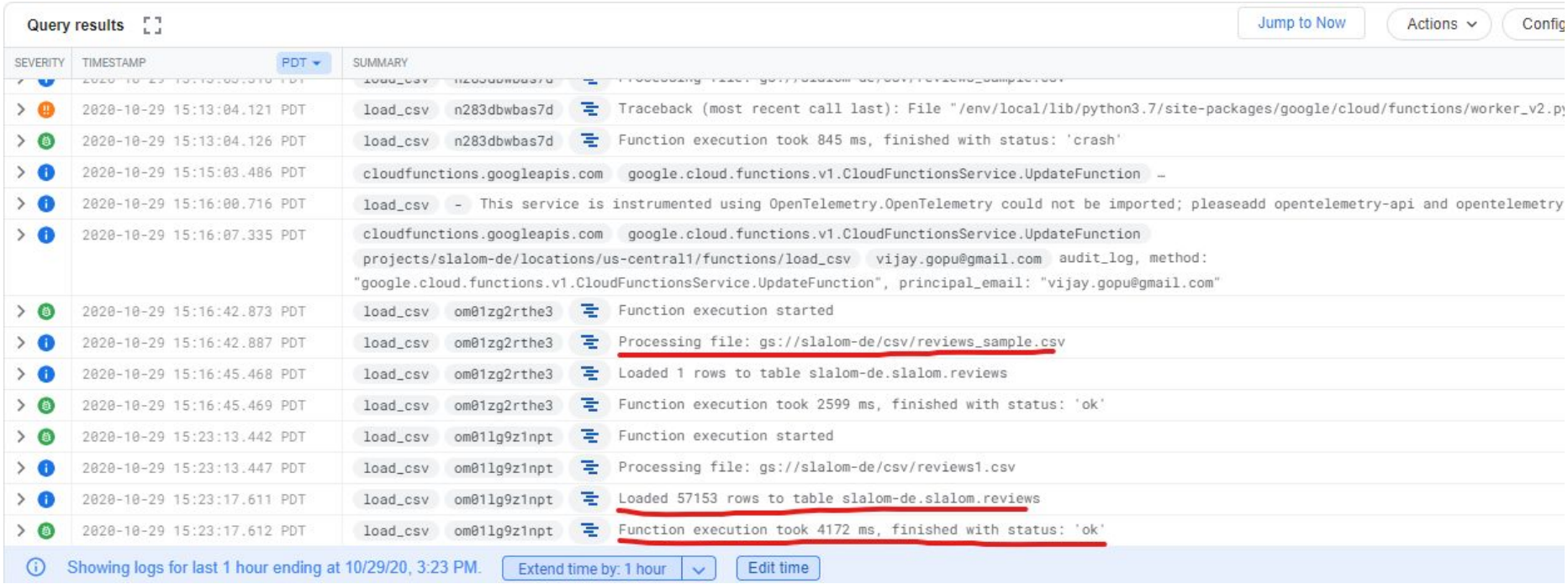
Data Load in BigQuery



StackDriver/Google Cloud Operations



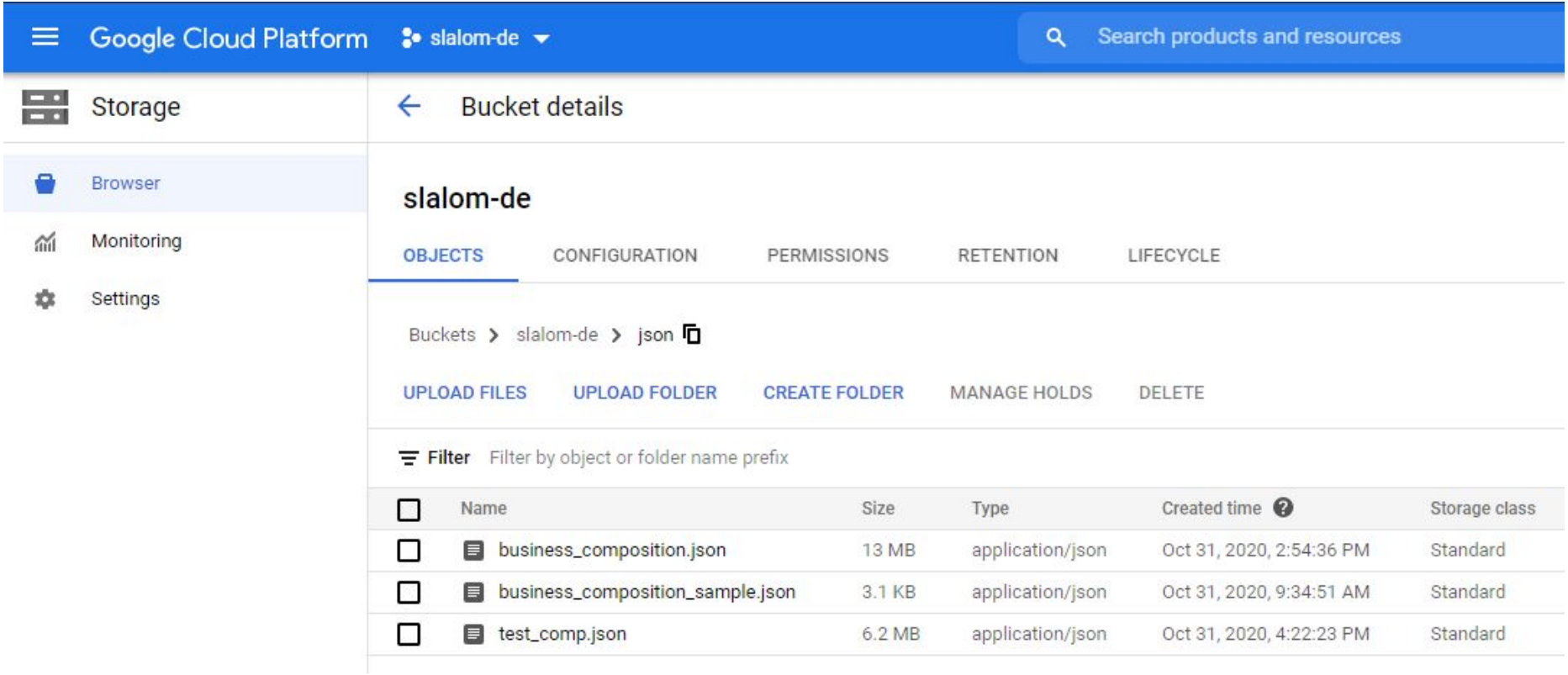
StackDriver/Google Cloud Operations



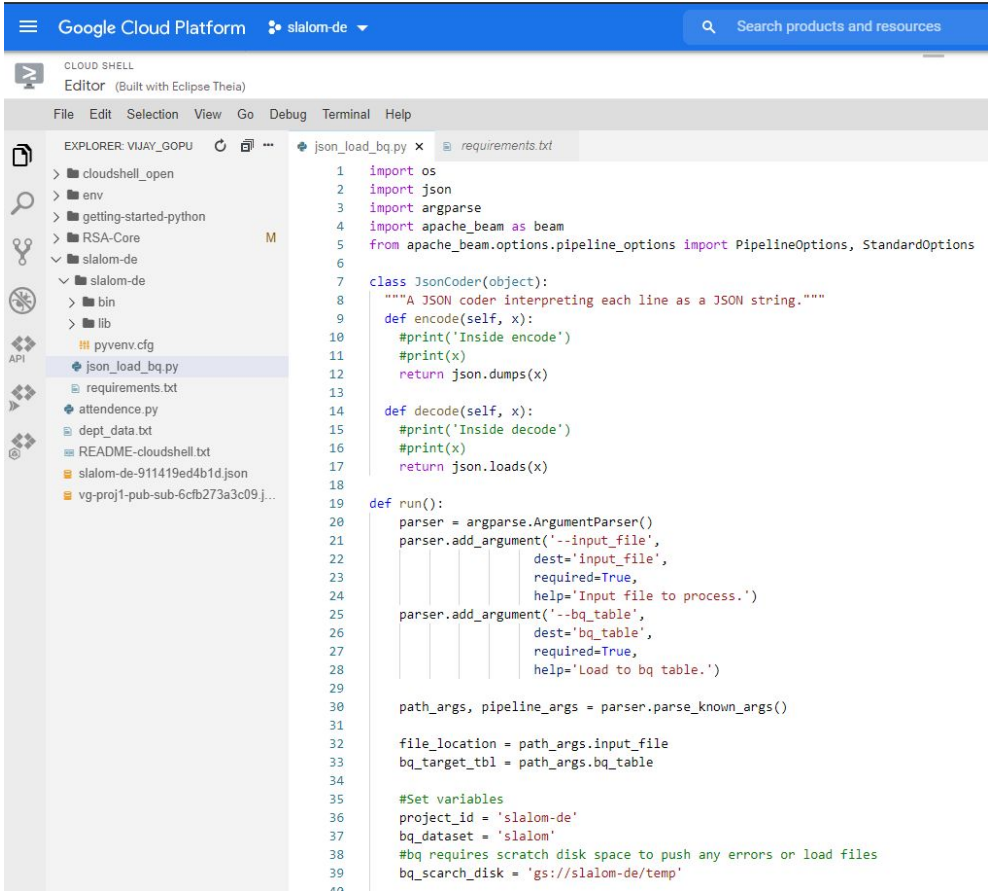
Serverless Batch Processing using Cloud Dataflow

1. JSON File Pushed to GCS
2. Batch Job is Started
3. Loaded to BigQuery

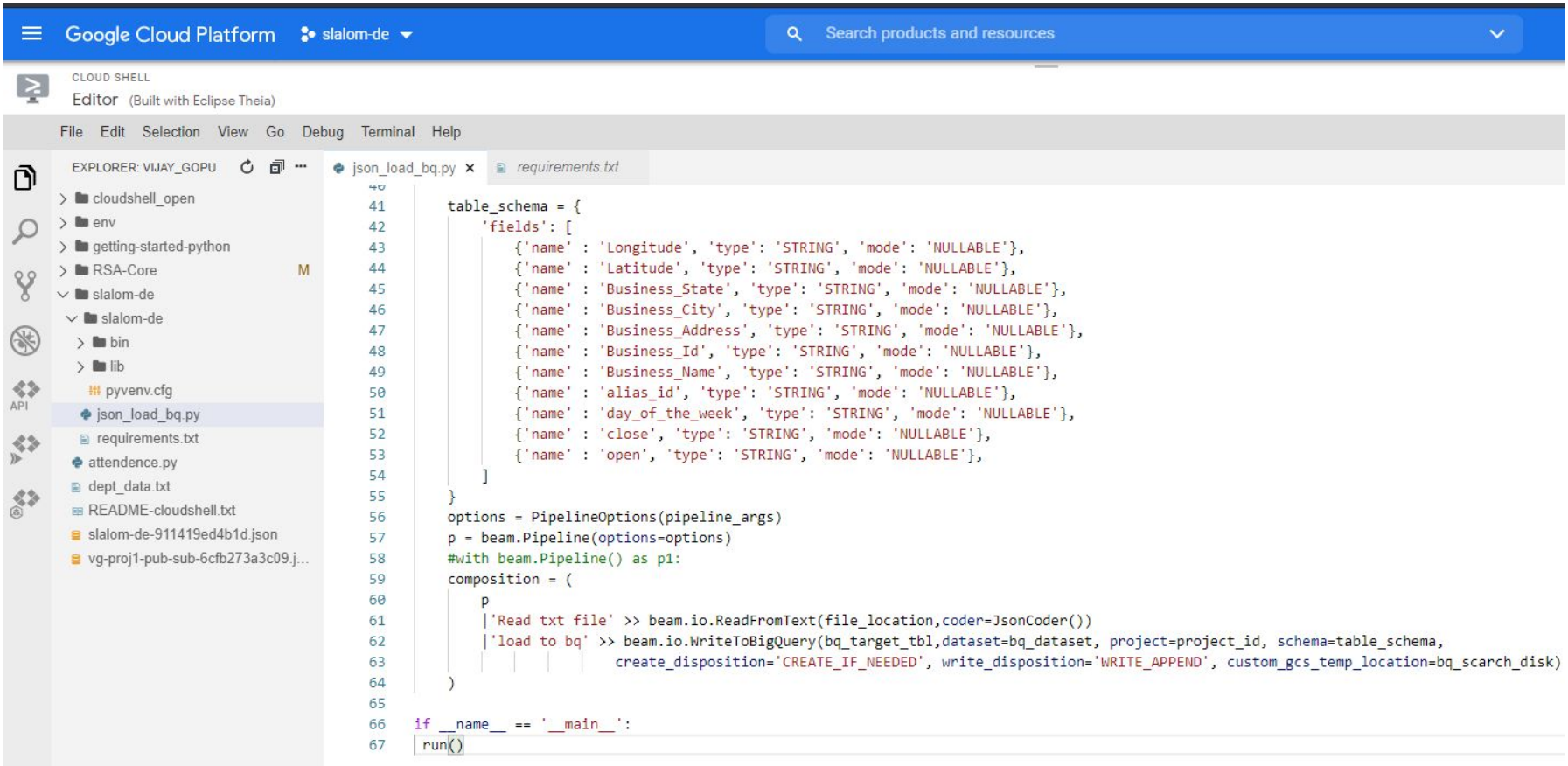
File Loaded to Google Cloud Storage



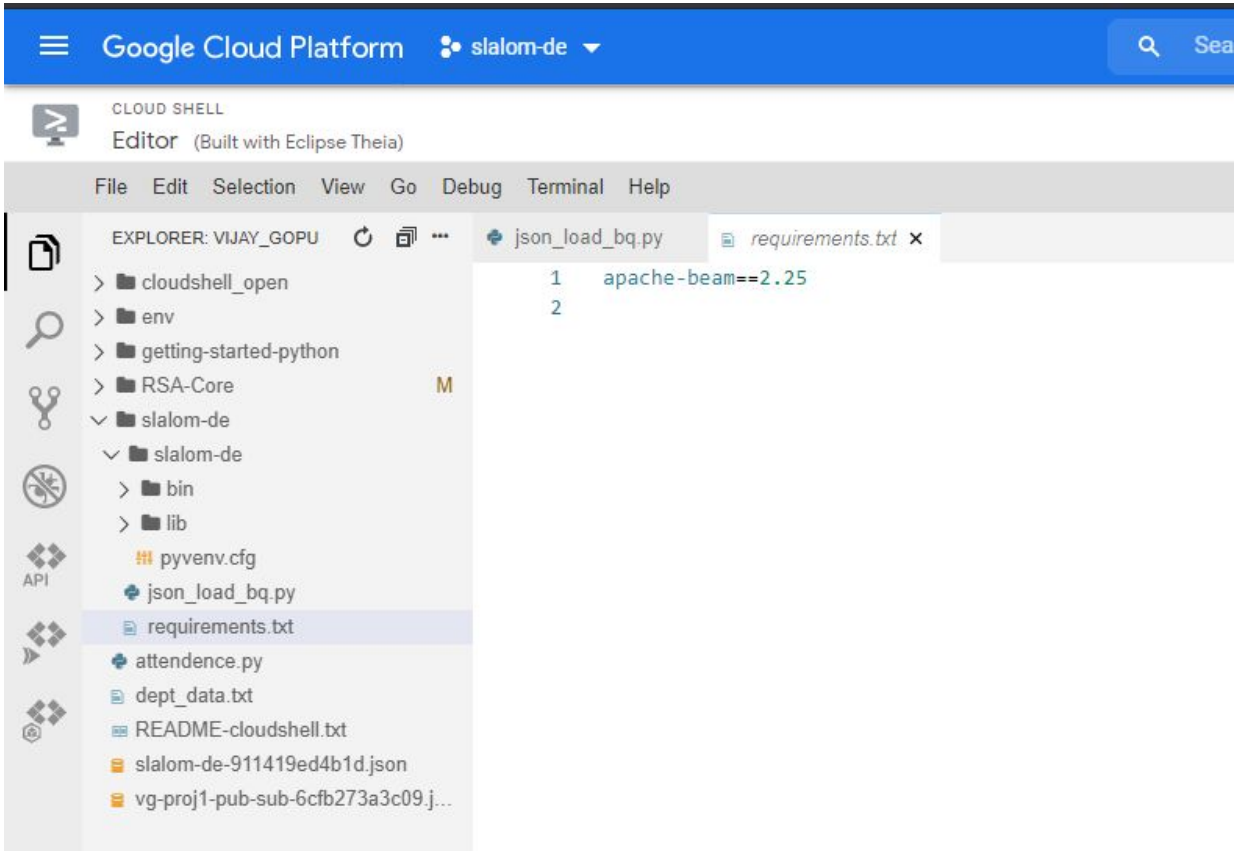
Pipeline Code



Pipeline Code



requirements.txt



Prerequisites

1. Create Service Account and Download the JSON API Key
2. Export the file location as GOOGLE_APPLICATION_CREDENTIALS

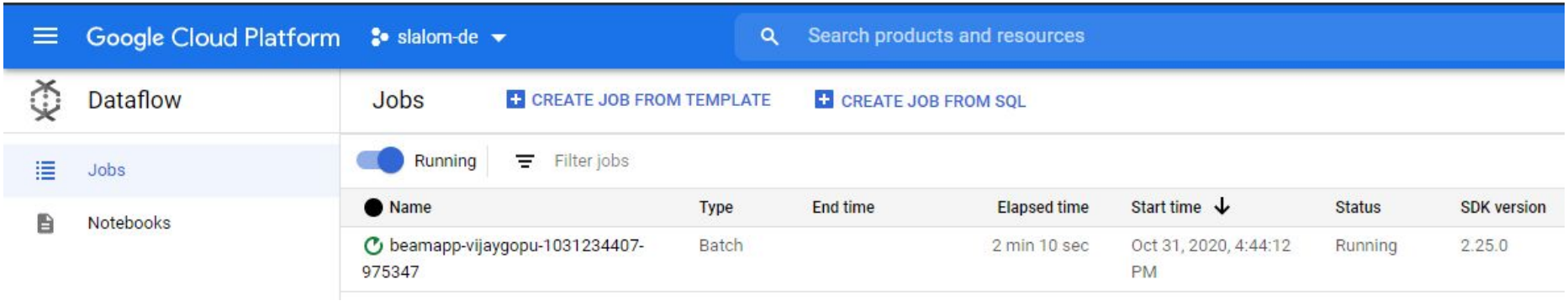
```
(slalom-de) vijay_gopu@cloudshell:~ (slalom-de)$ export GOOGLE_APPLICATION_CREDENTIALS="/home/vijay_gopu/slalom-de-911419ed4b1d.json"
(slalom-de) vijay_gopu@cloudshell:~ (slalom-de)$
```

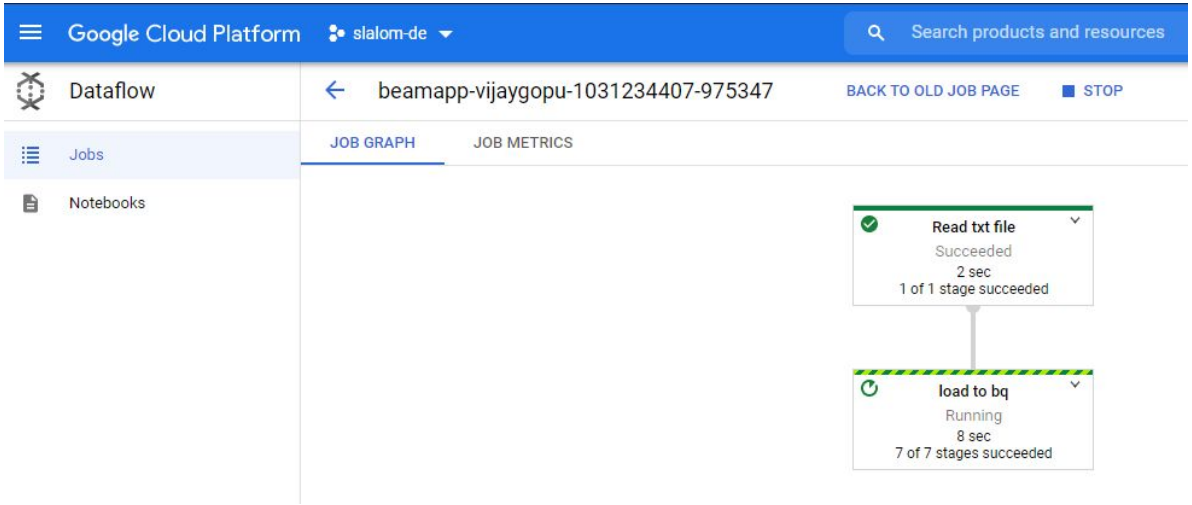
3. Create a virtual environment and perform installation of the requirements.txt
4. The JSON file need to be a file in the format such as each line represents a json object.

Command-line execution

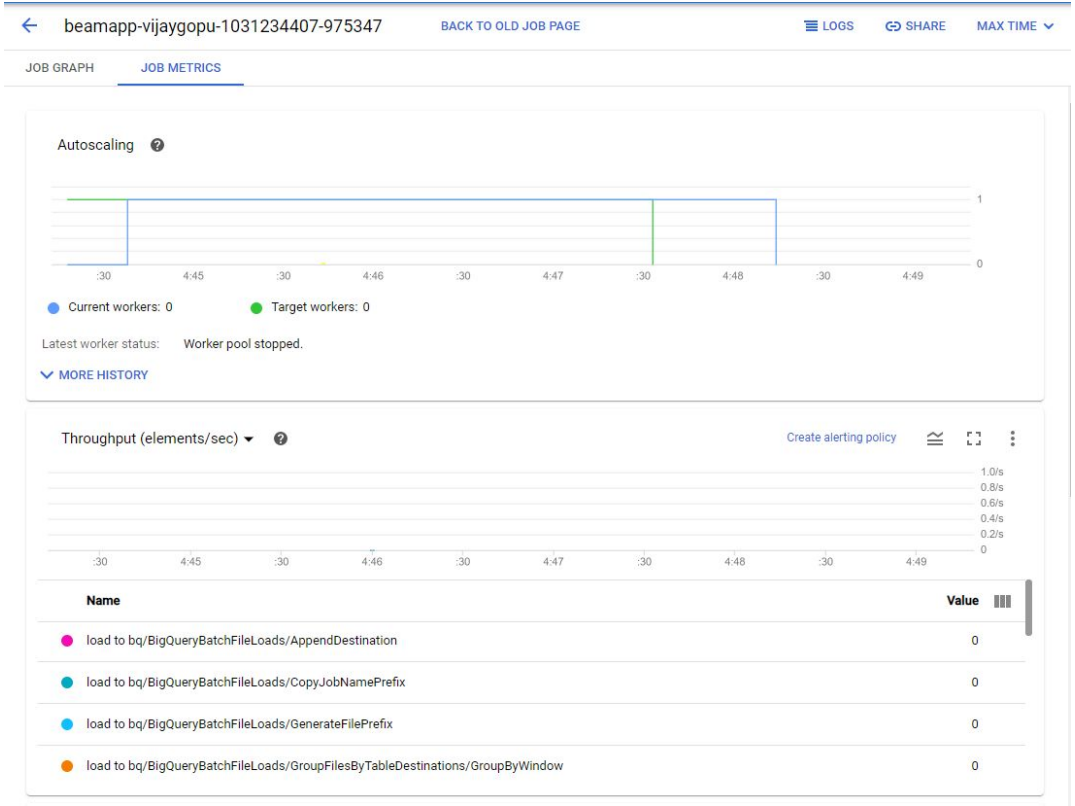
```
(slalom-de) vijay_gopu@cloudshell:~/slalom-de (slalom-de)$ python json_load_bq.py \
> --input_file gs://slalom-de/json/test_comp.json \
> --bq_table composition \
> --runner DataflowRunner \
> --project slalom-de \
> --temp_location gs://slalom-de/temp \
> --region us-central1
/home/vijay_gopu/slalom-de/slalom-de/lib/python3.7/site-packages/apache_beam/io/gcp/bigquery.py:1677: BeamDeprecationWarning: options is deprecated since Firs
t stable release. References to <pipeline>.options will not be supported
  experiments = p.options.view_as(DebugOptions).experiments or []
/home/vijay_gopu/slalom-de/slalom-de/lib/python3.7/site-packages/apache_beam/io/gcp/bigquery_file_loads.py:900: BeamDeprecationWarning: options is deprecated
since First stable release. References to <pipeline>.options will not be supported
  temp_location = p.options.view_as(GoogleCloudOptions).temp_location
WARNING: You are using pip version 20.2.3; however, version 20.2.4 is available.
You should consider upgrading via the '/home/vijay_gopu/slalom-de/slalom-de/bin/python -m pip install --upgrade pip' command.
WARNING: You are using pip version 20.2.3; however, version 20.2.4 is available.
You should consider upgrading via the '/home/vijay_gopu/slalom-de/slalom-de/bin/python -m pip install --upgrade pip' command.
WARNING:root:Make sure that locally built Python SDK docker image has Python 3.7 interpreter.
(slalom-de) vijay_gopu@cloudshell:~/slalom-de (slalom-de)$
```

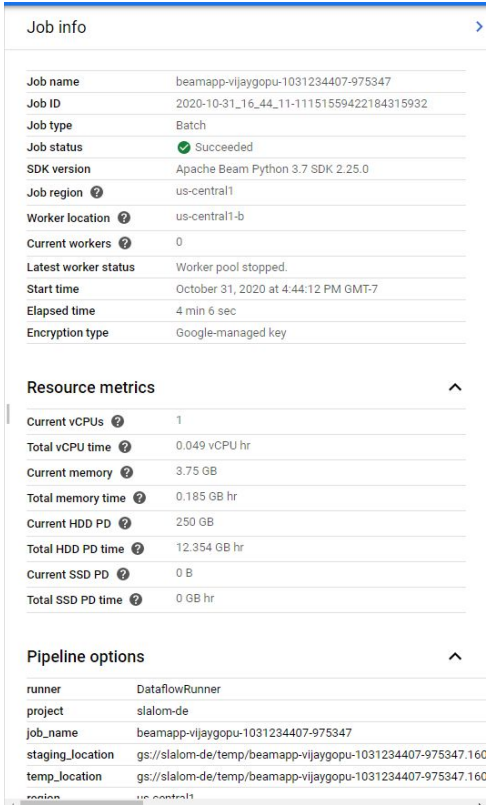
Dataflow Job Run



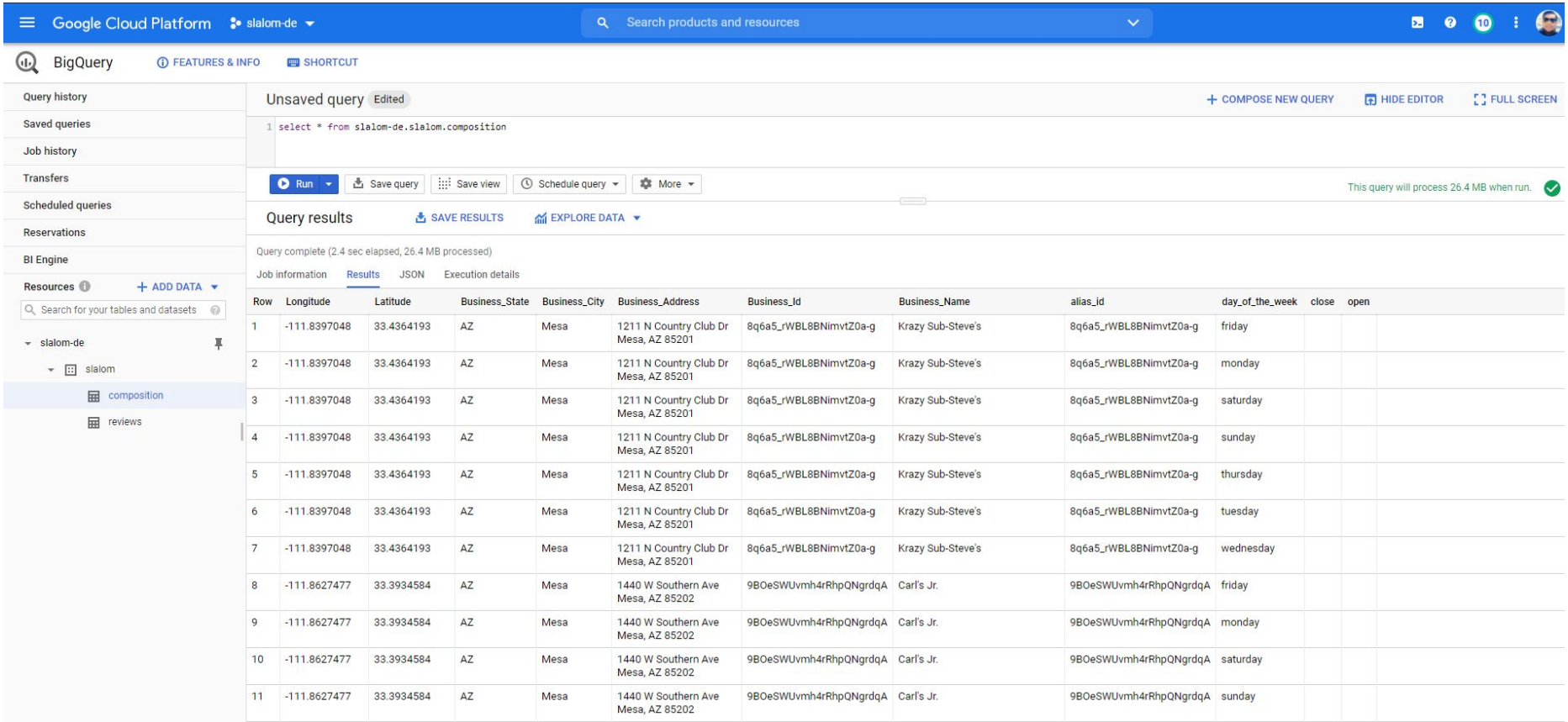


Dataflow Job Metrics



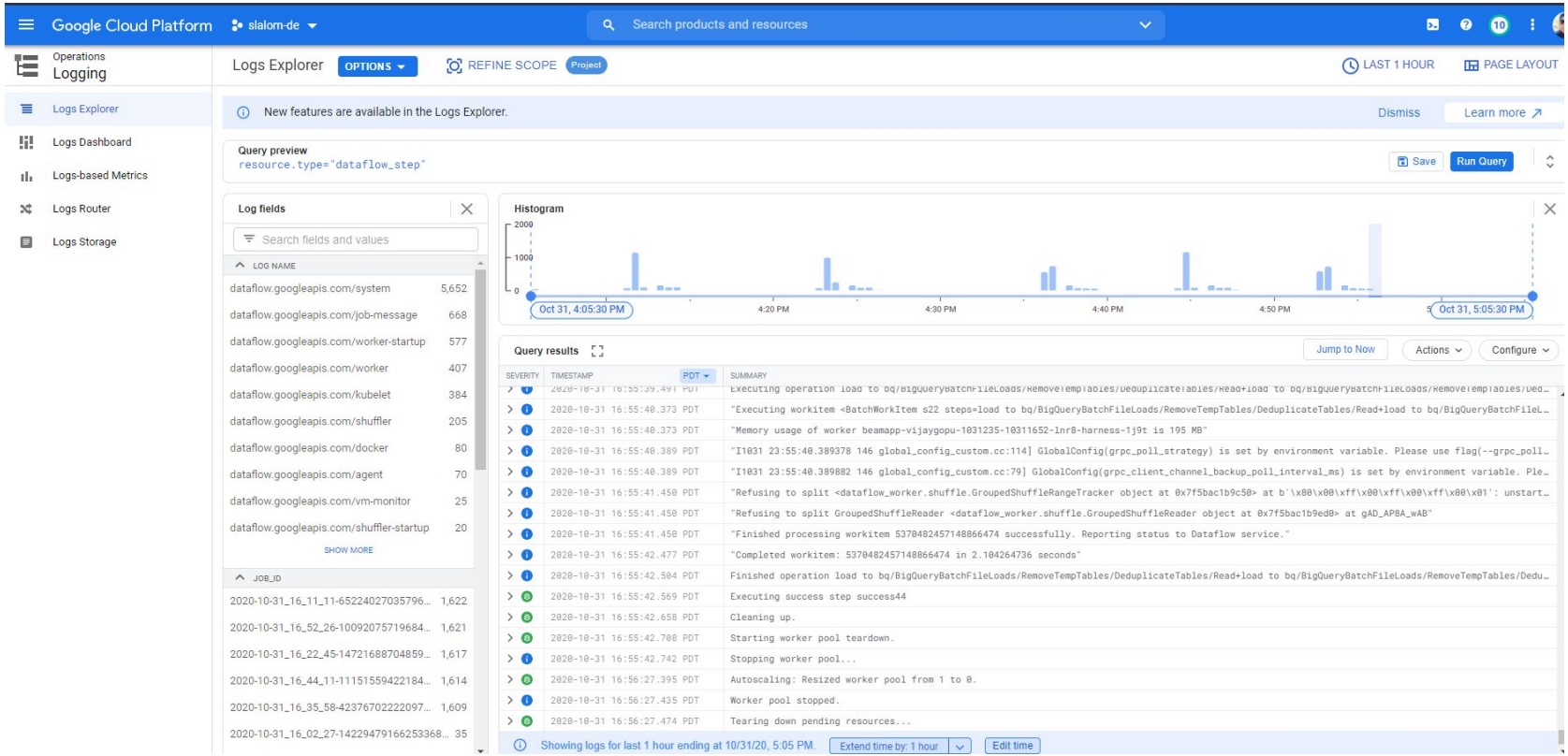


Data Landed in BigQuery



Row	Longitude	Latitude	Business_State	Business_City	Business_Address	Business_Id	Business_Name	alias_id	day_of_the_week	close	open
1	-111.8397048	33.4364193	AZ	Mesa	1211 N Country Club Dr Mesa, AZ 85201	8q6a5_rWBL8BNimvtZ0a-g	Krazy Sub-Steve's	8q6a5_rWBL8BNimvtZ0a-g	friday		
2	-111.8397048	33.4364193	AZ	Mesa	1211 N Country Club Dr Mesa, AZ 85201	8q6a5_rWBL8BNimvtZ0a-g	Krazy Sub-Steve's	8q6a5_rWBL8BNimvtZ0a-g	monday		
3	-111.8397048	33.4364193	AZ	Mesa	1211 N Country Club Dr Mesa, AZ 85201	8q6a5_rWBL8BNimvtZ0a-g	Krazy Sub-Steve's	8q6a5_rWBL8BNimvtZ0a-g	saturday		
4	-111.8397048	33.4364193	AZ	Mesa	1211 N Country Club Dr Mesa, AZ 85201	8q6a5_rWBL8BNimvtZ0a-g	Krazy Sub-Steve's	8q6a5_rWBL8BNimvtZ0a-g	sunday		
5	-111.8397048	33.4364193	AZ	Mesa	1211 N Country Club Dr Mesa, AZ 85201	8q6a5_rWBL8BNimvtZ0a-g	Krazy Sub-Steve's	8q6a5_rWBL8BNimvtZ0a-g	thursday		
6	-111.8397048	33.4364193	AZ	Mesa	1211 N Country Club Dr Mesa, AZ 85201	8q6a5_rWBL8BNimvtZ0a-g	Krazy Sub-Steve's	8q6a5_rWBL8BNimvtZ0a-g	tuesday		
7	-111.8397048	33.4364193	AZ	Mesa	1211 N Country Club Dr Mesa, AZ 85201	8q6a5_rWBL8BNimvtZ0a-g	Krazy Sub-Steve's	8q6a5_rWBL8BNimvtZ0a-g	wednesday		
8	-111.8627477	33.3934584	AZ	Mesa	1440 W Southern Ave Mesa, AZ 85202	9BOeSWUvmh4rRhpQNgrdqA	Carl's Jr.	9BOeSWUvmh4rRhpQNgrdqA	friday		
9	-111.8627477	33.3934584	AZ	Mesa	1440 W Southern Ave Mesa, AZ 85202	9BOeSWUvmh4rRhpQNgrdqA	Carl's Jr.	9BOeSWUvmh4rRhpQNgrdqA	monday		
10	-111.8627477	33.3934584	AZ	Mesa	1440 W Southern Ave Mesa, AZ 85202	9BOeSWUvmh4rRhpQNgrdqA	Carl's Jr.	9BOeSWUvmh4rRhpQNgrdqA	saturday		
11	-111.8627477	33.3934584	AZ	Mesa	1440 W Southern Ave Mesa, AZ 85202	9BOeSWUvmh4rRhpQNgrdqA	Carl's Jr.	9BOeSWUvmh4rRhpQNgrdqA	sunday		

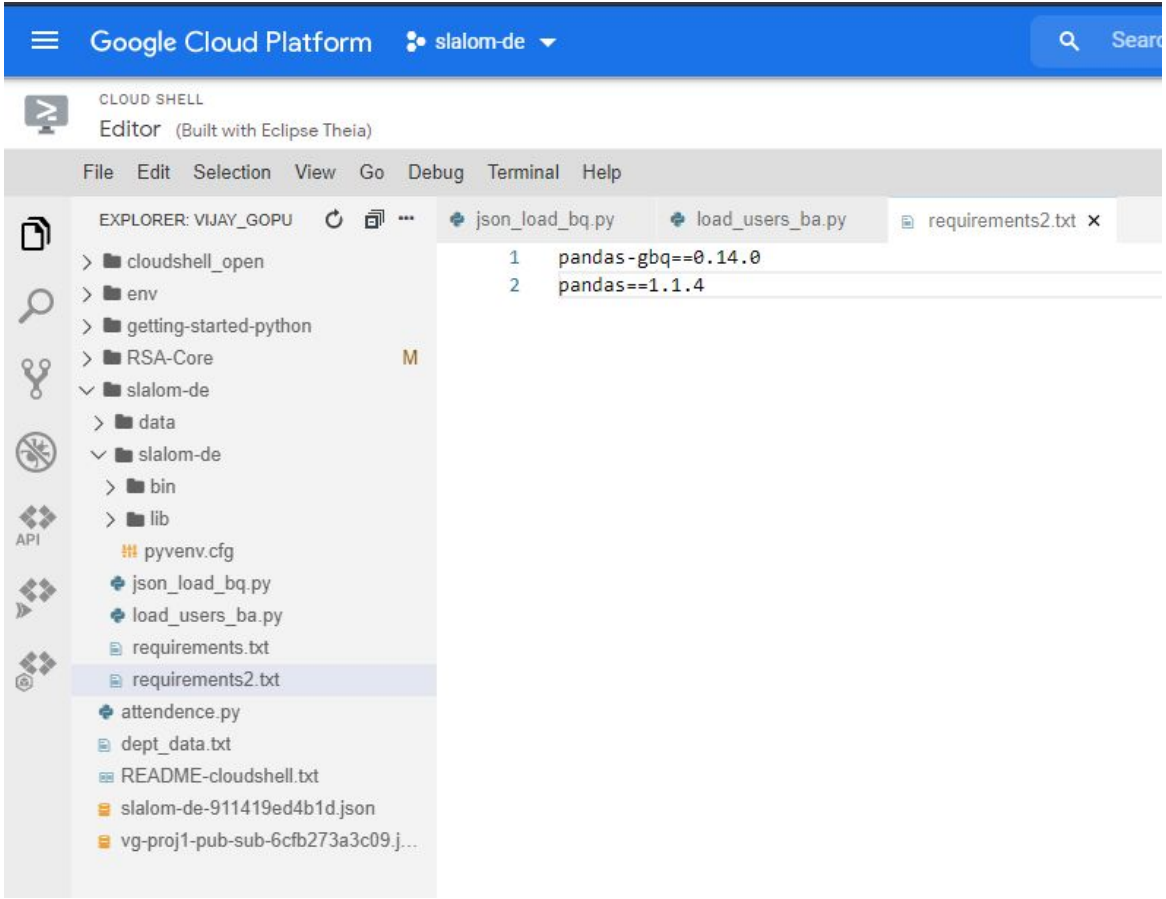
StackDriver Logs



Loading sqlite file to BigQuery

Using Pandas

requirements.txt



Load Job Code

json_load_bq.py | load_users_ba.py | requirements2.txt

```python
import os
import sqlite3
import pandas as pd
import argparse
#import pandas_gbq
from google.oauth2 import service_account

def load_users(conn,credentials):
  #using pandas read the USERS2 table using connection object
  df = pd.read_sql_query("SELECT * FROM USERS2", conn)

  #rename the df column name because bq will not accept spaces
  users_df = df.rename(columns={'User - Years Elite':'User_Years_Elite',
                                'Business - Id':'Business_Id',
                                'User - Compliments Cool':'User_Compliments_Cool',
                                'User - Compliments Cute':'User_Compliments_Cute',
                                'User - Compliments Funny':'User_Compliments_Funny',
                                'User - Compliments Hot':'User_Compliments_Hot',
                                'User - Compliments List':'User_Compliments_List',
                                'User - Compliments More':'User_Compliments_More',
                                'User - Compliments Note':'User_Compliments_Note',
                                'User - Compliments Photos':'User_Compliments_Photos',
                                'User - Compliments Plain':'User_Compliments_Plain',
                                'User - Compliments Profile':'User_Compliments_Profile',
                                'User - Compliments Writer':'User_Compliments_Writer',
                                'User - Fans':'User_Fans','User - Name':'User_Name',
                                'Review - Id':'Review_Id','User - Id':'User_Id',
                                'User - Votes Cool':'User_Votes_Cool',
                                'User - Votes Funny':'User_Votes_Funny',
                                'User - Votes Useful':'User_Votes_Useful',
                                'User - Yelping Since':'User_Yelping_Since'})

  #Load the data to bq using the dataframe method to_gbq
  #Provide table name along with the dataset name, project where the dataset resides, path to the service account key as credentials
  users_df.to_gbq(destination_table='slalom.users',project_id='slalom-de',credentials=credentials,if_exists='append')
```

Load Job Code

json_load_bq.py | load_users_ba.py | requirements2.txt

```
def load_business_attributes(conn,credentials):
  df = pd.read_sql_query("SELECT * FROM BUSINESS_ATTRIBUTES", conn)
  ba = df.rename(columns={'Business - Restaurant?':'Business_Restaurant',
                          'Business - Accepts Credit Cards':'Business_Accepts_Credit_Cards',
                          'Business - Accepts Insurance':'Business_Accepts_Insurance',
                          'Business - Ages Allowed':'Business_Ages_Allowed',
                          'Business - Alcohol':'Business_Alcohol',
                          'Business - Attire':'Business_Attire',
                          'Business - BYOB/Corkage':'Business_BYOB_Corkage',
                          'Business - BYOB':'Business_BYOB',
                          'Business - By Appointment Only':'Business_By_Appointment_Only',
                          'Business - Caters':'Business_Caters',
                          'Business - Coat Check':'Business_Coat_Check',
                          'Business - Corkage':'Business_Corkage',
                          'Business - Delivery':'Business_Delivery',
                          'Business - Dietary Restrictions':'Business_Dietary_Restrictions',
                          'Business - Dogs Allowed':'Business_Dogs_Allowed',
                          'Business - Drive-Thru':'Business_Drive_Thru',
                          'Business - Good For Dancing':'Business_Good_For_Dancing',
                          'Business - Good For Groups':'Business_Good_For_Groups',
                          'Business - Good For Kids':'Business_Good_For_Kids',
                          'Business - Good for Kids2':'Business_Good_for_Kids2',
                          'Business - Happy Hour':'Business_Happy_Hour',
                          'Business - Has TV':'Business_Has_TV',
                          'Business - Noise Level':'Business_Noise_Level',
                          'Business - Open 24 Hours':'Business_Open_24_Hours',
                          'Business - Order at Counter':'Business_Order_at_Counter',
                          'Business - Outdoor Seating':'Business_Outdoor_Seating',
                          'Business - Parking':'Business_Parking',
                          'Business - Payment Types':'Business_Payment_Types',
                          'Business - Price Range':'Business_Price_Range',
                          'Business - Smoking':'Business_Smoking',
                          'Business - Take-out':'Business_Take_out',
                          'Business - Takes Reservations':'Business_Takes_Reservations',
                          'Business - Waiter Service':'Business_Waiter_Service',
                          'Business - Wheelchair Accessible':'Business_Wheelchair_Accessible',
                          'Business - Wi-Fi':'Business_Wi_Fi',
                          'Business - Id':'Business_Id',
                          'Business - Categories':'Business_Categories',
```

Load Job Code

json_load_bq.py | load_users_ba.py | requirements2.txt

```
                                'Business - Categories':'Business_Categories',
                                'Business - Name':'Business_Name',
                                'Business - Neighborhoods':'Business_Neighborhoods',
                                'Business - Open?':'Business_Open'})
  #Load the data to bq using the dataframe method to_gbq
  #Provide table name along with the dataset name, project where the dataset resides, path to the service account key as credentials
  ba.to_gbq(destination_table='slalom.business_attributes',project_id='slalom-de',credentials=credentials,if_exists='append')

def create_connection(sqlite_file):
  conn = sqlite3.connect(sqlite_file)
  return conn

def close_connection(conn):
  conn.close()

def set_credentials(credentials_file):
  #Credentials required to load data to project using the service account
  credentials = service_account.Credentials.from_service_account_file(
    credentials_file,
  )
  return credentials

def run():
  parser = argparse.ArgumentParser()
  parser.add_argument('--sqlite_file',
                      dest='sqlite_file',
                      required=True,
                      help='Provide Sqlite file path')
  parser.add_argument('--credentials_file',
                      dest='credentials_file',
                      required=True,
                      help='Service Account Credentials file path')
  args = parser.parse_args()

  #Setting variables from parser
  sqlite_file = args.sqlite_file
  credentials_file = args.credentials_file
```

Load Job Code

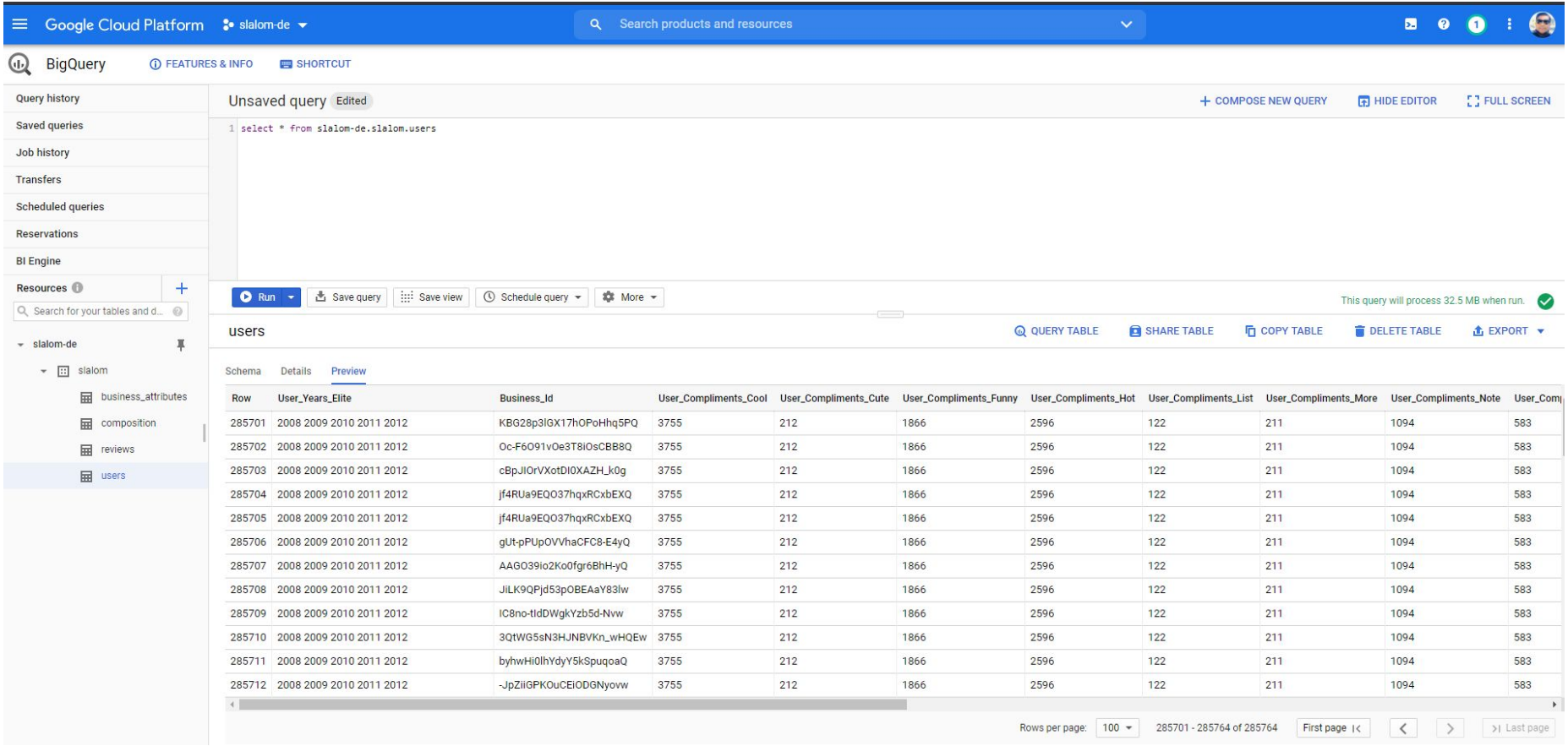
json_load_bq.py | load_users_ba.py | requirements2.txt

```
96
97  def run():
98      parser = argparse.ArgumentParser()
99      parser.add_argument('--sqlite_file',
100                         dest='sqlite_file',
101                         required=True,
102                         help='Provide Sqlite file path')
103     parser.add_argument('--credentials_file',
104                         dest='credentials_file',
105                         required=True,
106                         help='Service Account Credentials file path')
107     args = parser.parse_args()
108
109     #Setting variables from parser
110     sqlite_file = args.sqlite_file
111     credentials_file = args.credentials_file
112
113     #set credentials
114     credentials = set_credentials(credentials_file)
115
116     #create connection
117     conn = create_connection(sqlite_file)
118
119     #load_users_table_to_bq
120     load_users(conn,credentials)
121
122     #load_business_attributes_to_bq
123     load_business_attributes(conn,credentials)
124
125     #close connection
126     close_connection(conn)
127
128 if __name__ == '__main__':
129     run()
```

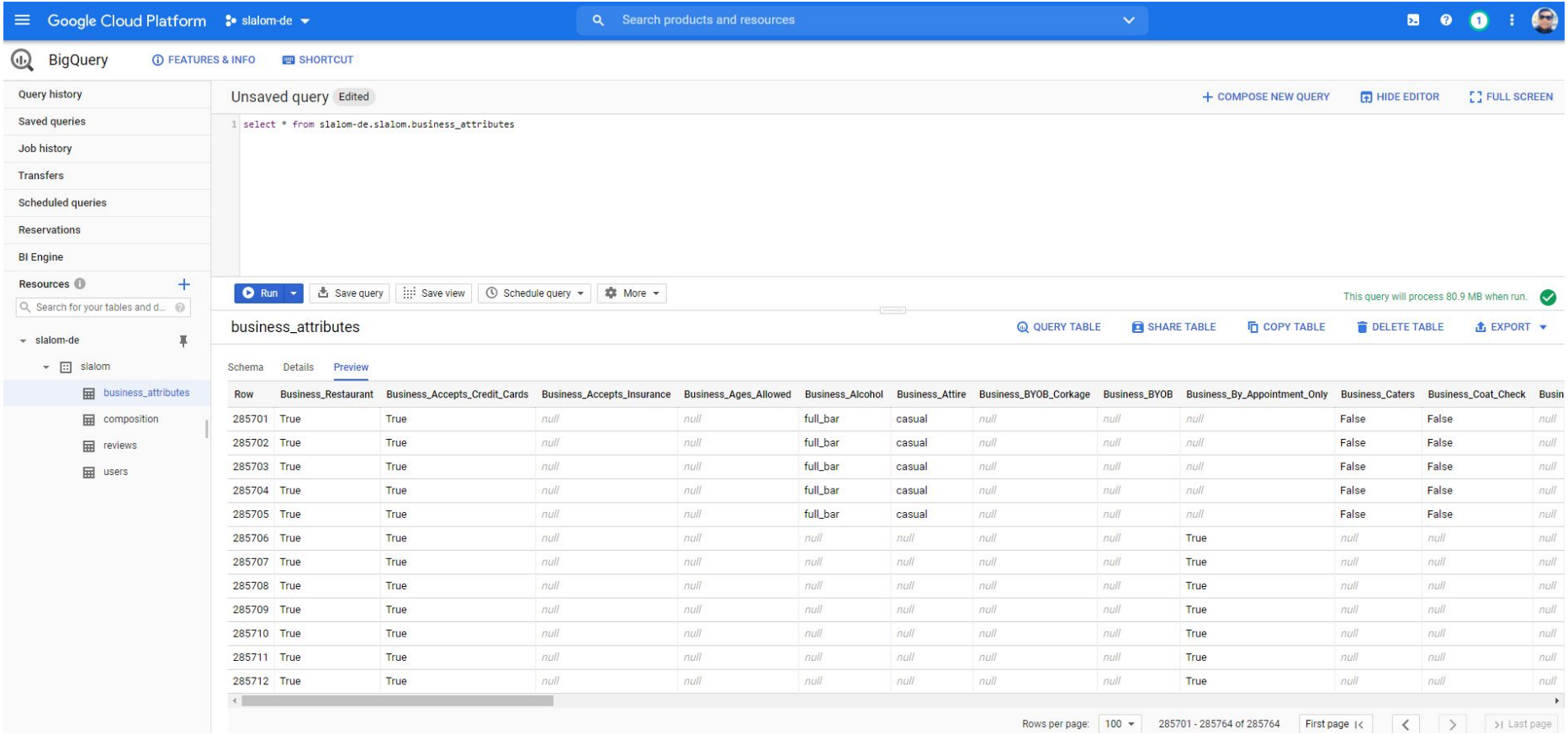
Command line execution

```
(slalom-de) vijay_gopu@cloudshell:~/slalom-de (slalom-de)$ python load_users_ba.py \
> --sqlite_file /home/vijay_gopu/slalom-de/data/user.sqlite \
> --credentials_file /home/vijay_gopu/slalom-de-911419ed4b1d.json
(slalom-de) vijay_gopu@cloudshell:~/slalom-de (slalom-de)$
```

Users In BigQuery

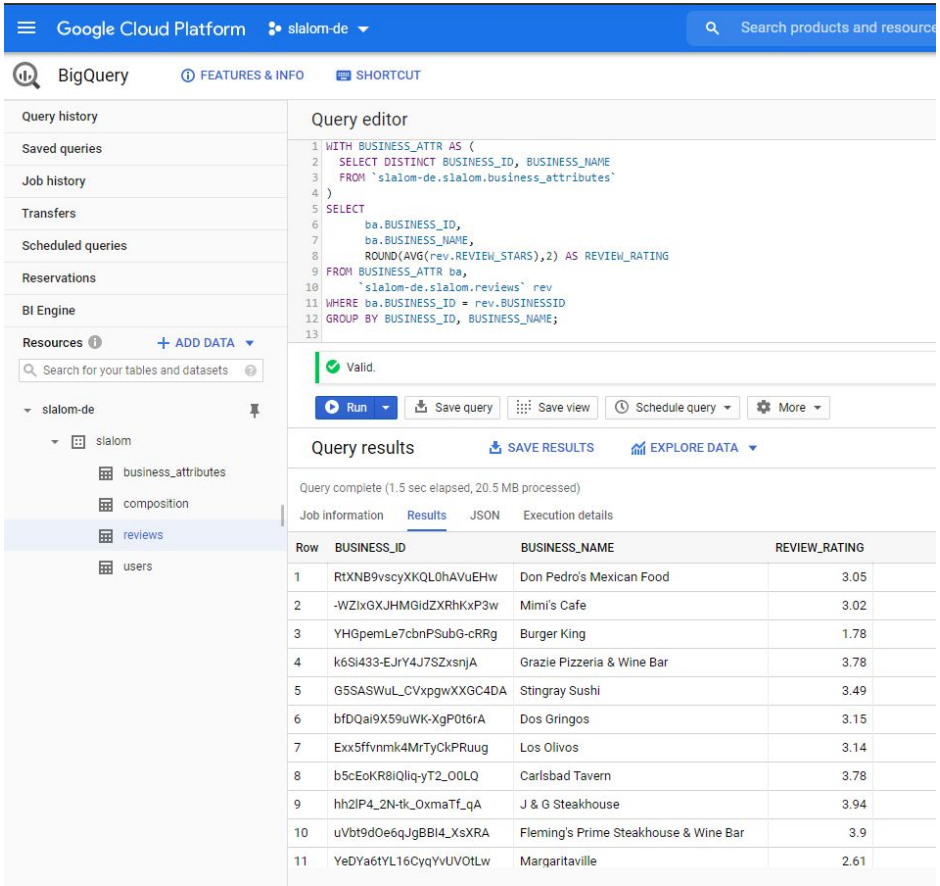


Business_Attributes In BigQuery

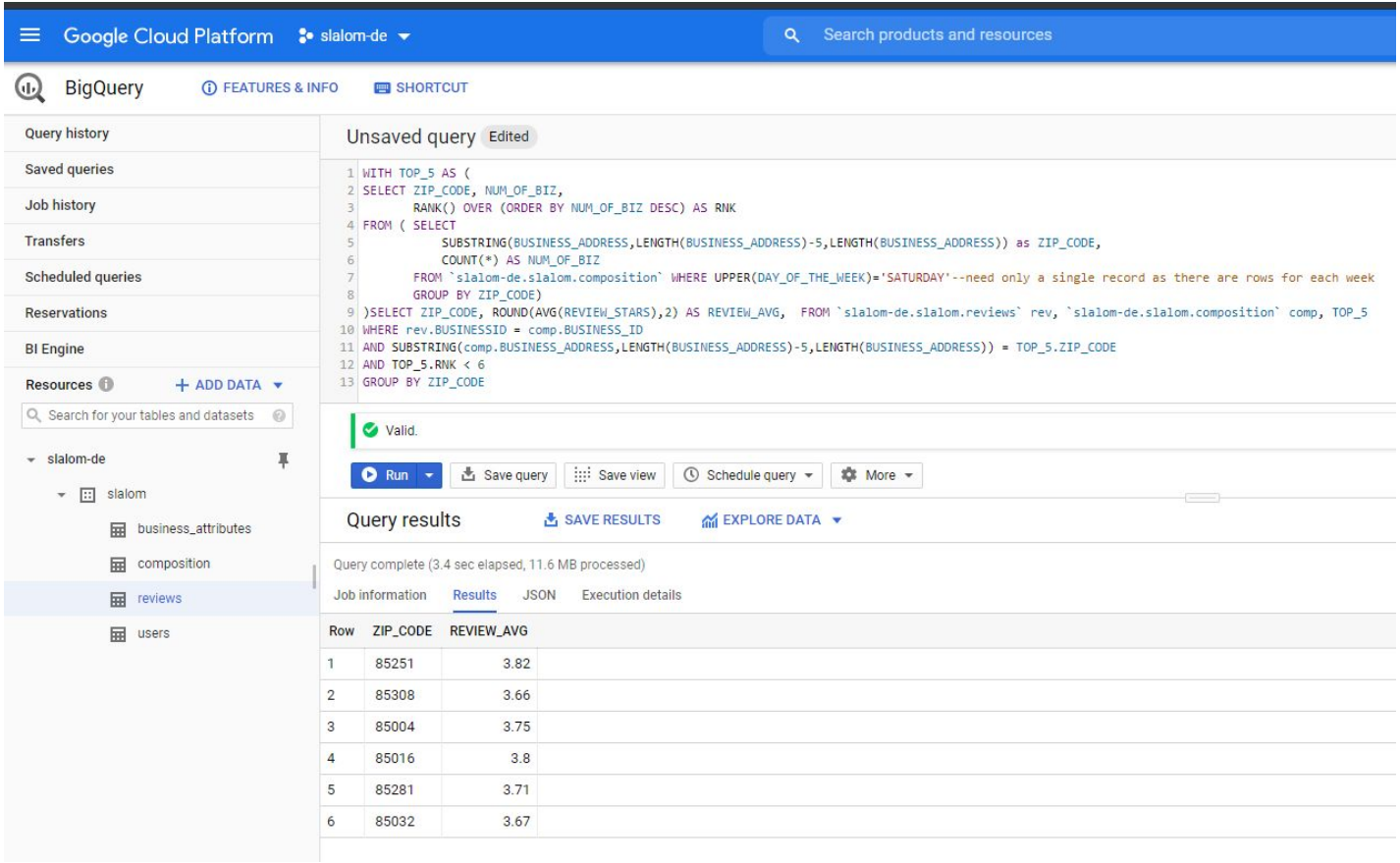


SQL Questions

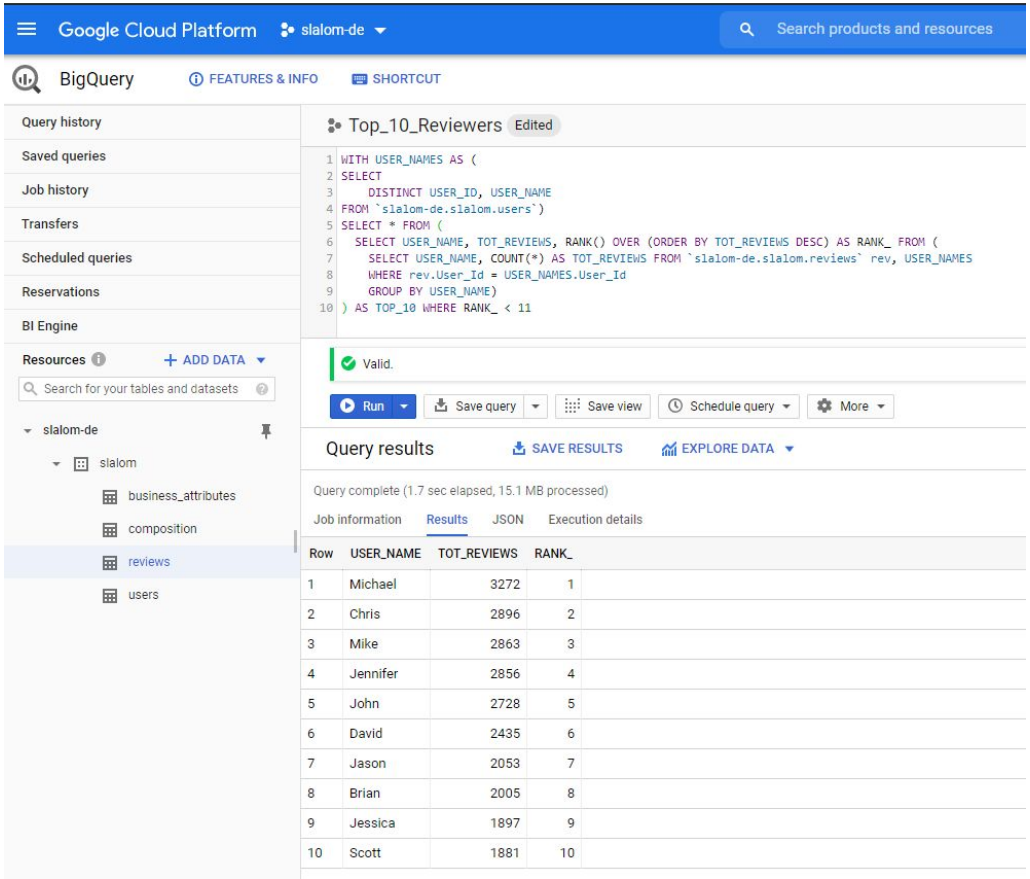
Mean Reviews of every Business



Mean Reviews By Top 5 Zip Codes



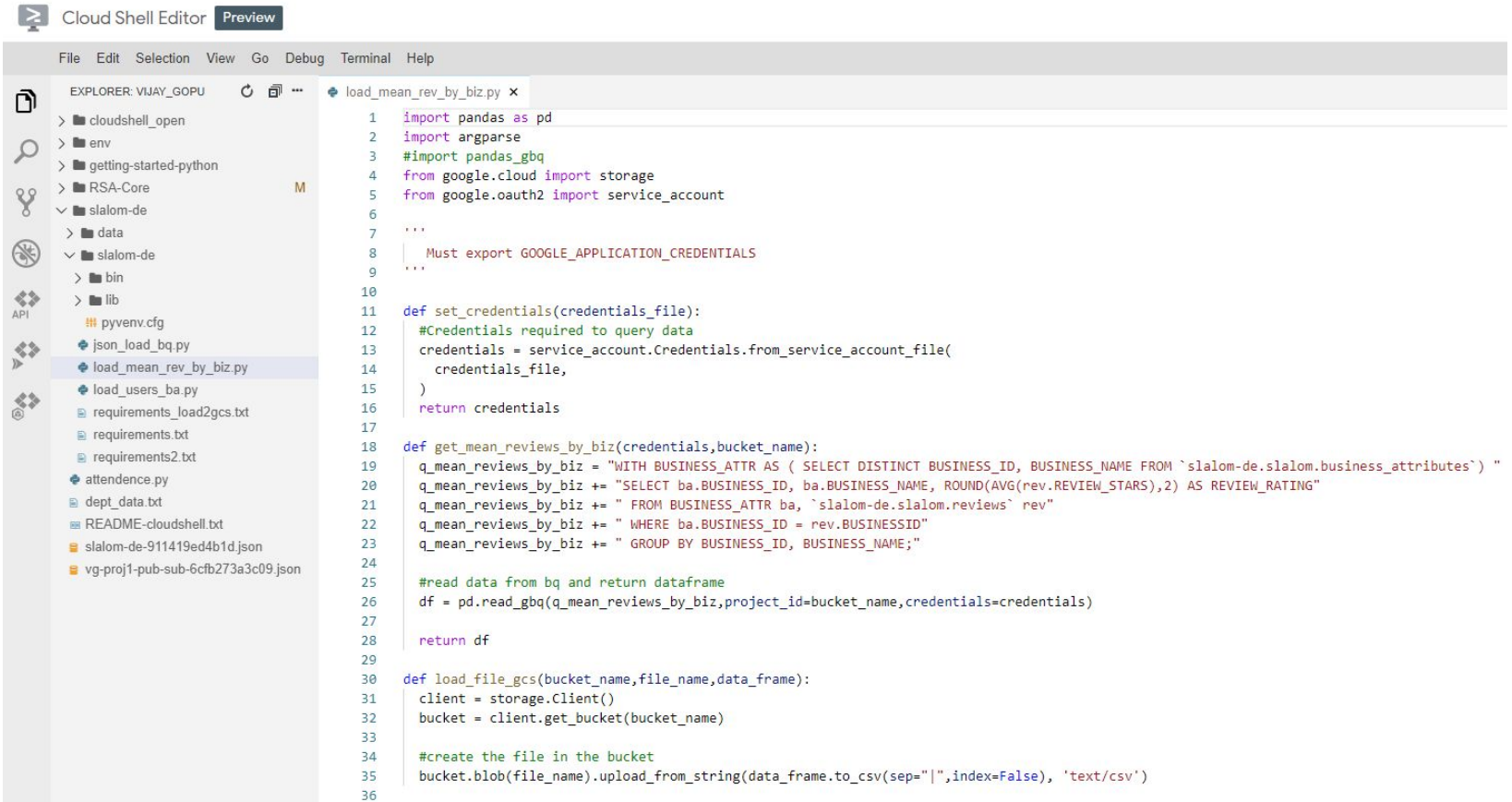
Top 10 Active Reviewers



Row	USER_NAME	TOT_REVIEWS	RANK_
1	Michael	3272	1
2	Chris	2896	2
3	Mike	2863	3
4	Jennifer	2856	4
5	John	2728	5
6	David	2435	6
7	Jason	2053	7
8	Brian	2005	8
9	Jessica	1897	9
10	Scott	1881	10

Loading To Cloud Storage

Mean Reviews By Business



Mean Reviews By Business

load_mean_rev_by_biz.py

```
30  def load_file_gcs(bucket_name,file_name,data_frame):
31    client = storage.Client()
32    bucket = client.get_bucket(bucket_name)
33
34    #create the file in the bucket
35    bucket.blob(file_name).upload_from_string(data_frame.to_csv(sep="|",index=False), 'text/csv')
36
37  def run():
38    parser = argparse.ArgumentParser()
39    parser.add_argument('--output_file',
40                        dest='output_file',
41                        required=True,
42                        help='Provide output file path')
43    parser.add_argument('--bucket_name',
44                        dest='bucket_name',
45                        required=True,
46                        help='bucket name for output file')
47    parser.add_argument('--credentials_file',
48                        dest='credentials_file',
49                        required=True,
50                        help='Service Account Credentials file path')
51    args = parser.parse_args()
52
53    bucket_name = args.bucket_name
54    output_file = args.output_file
55    credentials_file = args.credentials_file
56
57    #set credentials using the json file
58    credentials = set_credentials(credentials_file)
59
60    #execute the query to extract data from bq
61    df_mean_reviews_by_biz = get_mean_reviews_by_biz(credentials,bucket_name)
62
63    #create the file in gcs
64    load_file_gcs(bucket_name,output_file,df_mean_reviews_by_biz)
65
66  if __name__ == '__main__':
67    run()
68
```

Mean Reviews By Business - Prerequisites

1. requirements.txt

load_mean_rev_by_biz.py | requirements_load2gcs.txt

```
1  google-cloud-storage==1.32.0
2  #pandas-gbq==0.14.0 This is required if already not in your environment
```

2. Export Google Credentials

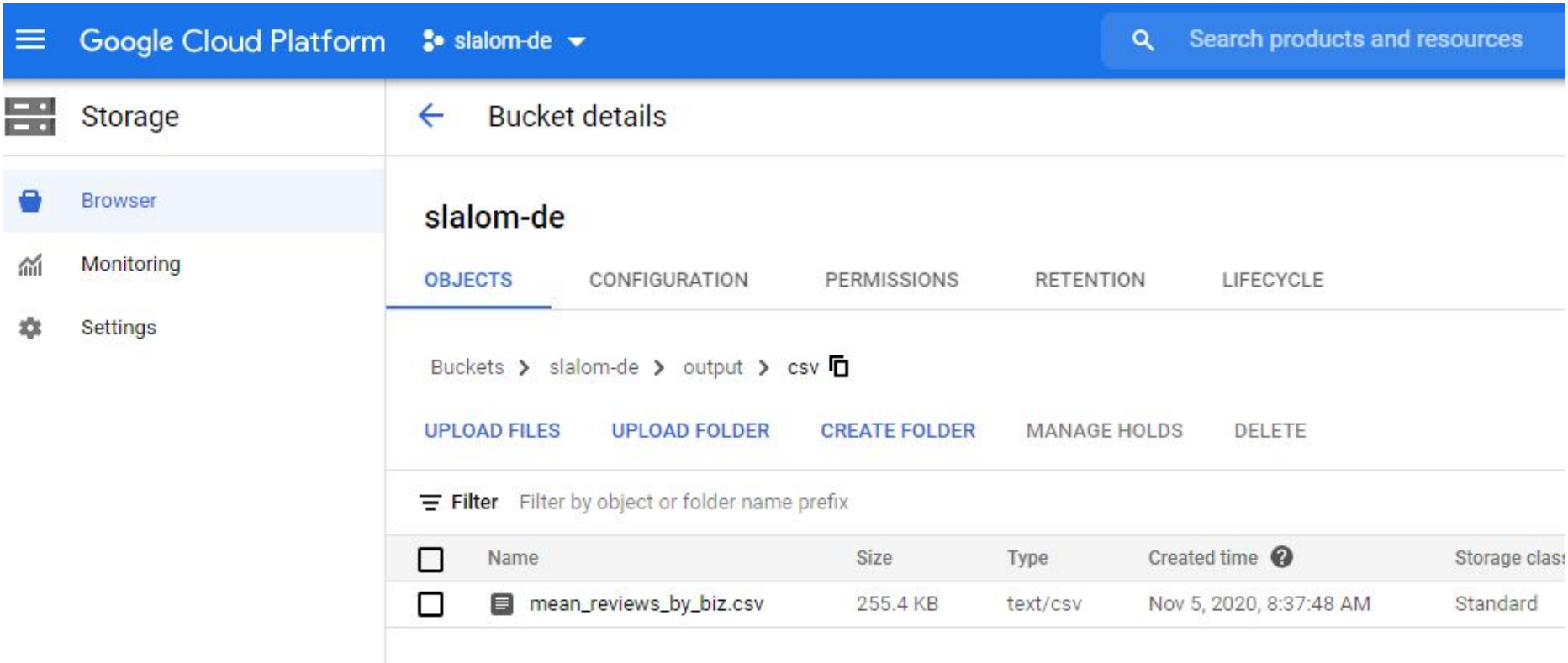
```
~/slalom-de (slalom-de)$ export GOOGLE_APPLICATION_CREDENTIALS="/home/vijay_gopu/slalom-de-911419ed4b1d.json"
```

Mean Reviews By Business

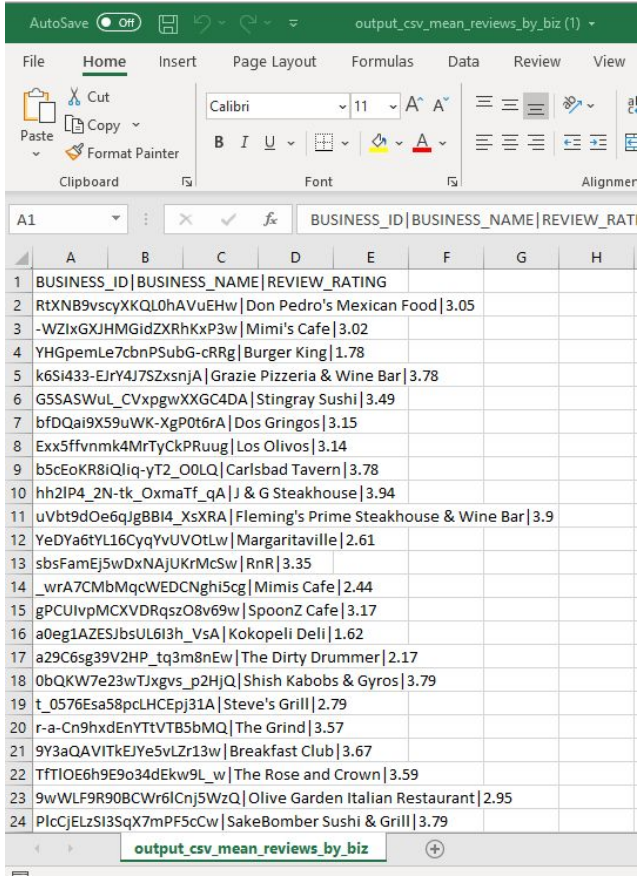
Command Line Execution

```
(slalom-de) vijay_gopu@cloudshell:~/slalom-de (slalom-de)$ python load_mean_rev_by_biz.py \
> --output_file output/csv/mean_reviews_by_biz.csv \
> --bucket_name slalom-de \
> --credentials_file /home/vijay_gopu/slalom-de-911419ed4b1d.json
(slalom-de) vijay_gopu@cloudshell:~/slalom-de (slalom-de)$
```

Mean Reviews By Business



Mean Reviews By Business



	A	B	C	D	E	F	G	H
1	BUSINESS_ID|BUSINESS_NAME|REVIEW_RATING							
2	RtXNB9vscyXKQL0hAVuEHw|Don Pedro's Mexican Food|3.05							
3	-WZIxGXJHMGidZXRhKxP3w|Mimi's Cafe|3.02							
4	YHGpemLe7cbnPSubG-cRRg|Burger King|1.78							
5	k6Si433-EJrY4J7SZxsnjA|Grazie Pizzeria & Wine Bar|3.78							
6	G5SASWuL_CVxpgwXXGC4DA|Stingray Sushi|3.49							
7	bfDQai9X59uWK-XgP0t6rA|Dos Gringos|3.15							
8	Exx5ffvnmk4MrTyCkPRuug|Los Olivos|3.14							
9	b5cEoKR8iQliq-yT2_O0LQ|Carlsbad Tavern|3.78							
10	hh2IP4_2N-tk_OxmaTf_qA|J & G Steakhouse|3.94							
11	uVbt9dOe6qJgBBI4_XsXRA|Fleming's Prime Steakhouse & Wine Bar|3.9							
12	YeDYa6tYL16CyqYvUVOtLw|Margaritaville|2.61							
13	sbsFamEj5wDxNAjUKrMcSw|RnR|3.35							
14	_wrA7CMbMqcWEDCNghi5cg|Mimis Cafe|2.44							
15	gPCUIvpMCXVDRqszO8v69w|SpoonZ Cafe|3.17							
16	a0eg1AZESJbsUL6I3h_VsA|Kokopeli Deli|1.62							
17	a29C6sg39V2HP_tq3m8nEw|The Dirty Drummer|2.17							
18	0bQKW7e23wTJxgvs_p2HjQ|Shish Kabobs & Gyros|3.79							
19	t_0576Esa58pcLHCEpj31A|Steve's Grill|2.79							
20	r-a-Cn9hxdEnYTtVTB5bMQ|The Grind|3.57							
21	9Y3aQAVITkEJYe5vLZr13w|Breakfast Club|3.67							
22	TfTlOE6h9E9o34dEkw9L_w|The Rose and Crown|3.59							
23	9wWLF9R90BCWr6lCnj5WzQ|Olive Garden Italian Restaurant|2.95							
24	PlcCjELzSI3SqX7mPF5cCw|SakeBomber Sushi & Grill|3.79							

Top 5 Mean Reviews By Zip

load_mean_rev_by_zip.py

```python
import pandas as pd
import argparse
#import pandas_gbq
from google.cloud import storage
from google.oauth2 import service_account

'''
   Must export GOOGLE_APPLICATION_CREDENTIALS
'''

def set_credentials(credentials_file):
  #Credentials required to query data
  credentials = service_account.Credentials.from_service_account_file(
    credentials_file,
  )
  return credentials

def get_mean_reviews_by_zip(credentials,bucket_name):
  q_mean_reviews_by_zip = "WITH TOP_5 AS ("
  q_mean_reviews_by_zip +="  SELECT ZIP_CODE, NUM_OF_BIZ, RANK() OVER (ORDER BY NUM_OF_BIZ DESC) AS RNK"
  q_mean_reviews_by_zip +="  FROM ( SELECT "
  q_mean_reviews_by_zip +="           SAFE.SUBSTRING(BUSINESS_ADDRESS,LENGTH(BUSINESS_ADDRESS)-5,LENGTH(BUSINESS_ADDRESS)) as ZIP_CODE, COUNT(*) AS NUM_OF_BIZ"
  q_mean_reviews_by_zip +="         FROM `slalom-de.slalom.composition` WHERE UPPER(DAY_OF_THE_WEEK)='SATURDAY'"
  q_mean_reviews_by_zip +="         GROUP BY ZIP_CODE)"
  q_mean_reviews_by_zip +=")SELECT ZIP_CODE, ROUND(AVG(REVIEW_STARS),2) AS REVIEW_AVG,  FROM `slalom-de.slalom.reviews` rev, `slalom-de.slalom.composition` comp, TOP_5 "
  q_mean_reviews_by_zip +="WHERE rev.BUSINESSID = comp.BUSINESS_ID "
  q_mean_reviews_by_zip +="AND SAFE.SUBSTRING(comp.BUSINESS_ADDRESS,LENGTH(BUSINESS_ADDRESS)-5,LENGTH(BUSINESS_ADDRESS)) = TOP_5.ZIP_CODE "
  q_mean_reviews_by_zip +="AND TOP_5.RNK < 6 "
  q_mean_reviews_by_zip +="GROUP BY ZIP_CODE;"
  #read data from bq and return dataframe
  df = pd.read_gbq(q_mean_reviews_by_zip,project_id=bucket_name,credentials=credentials)

  return df

def load_file_gcs(bucket_name,file_name,data_frame):
  client = storage.Client()
  bucket = client.get_bucket(bucket_name)

  #create the file in the bucket
  bucket.blob(file_name).upload_from_string(data_frame.to_csv(sep="|",index=False), 'text/csv')
```

Top 5 Mean Reviews By Zip

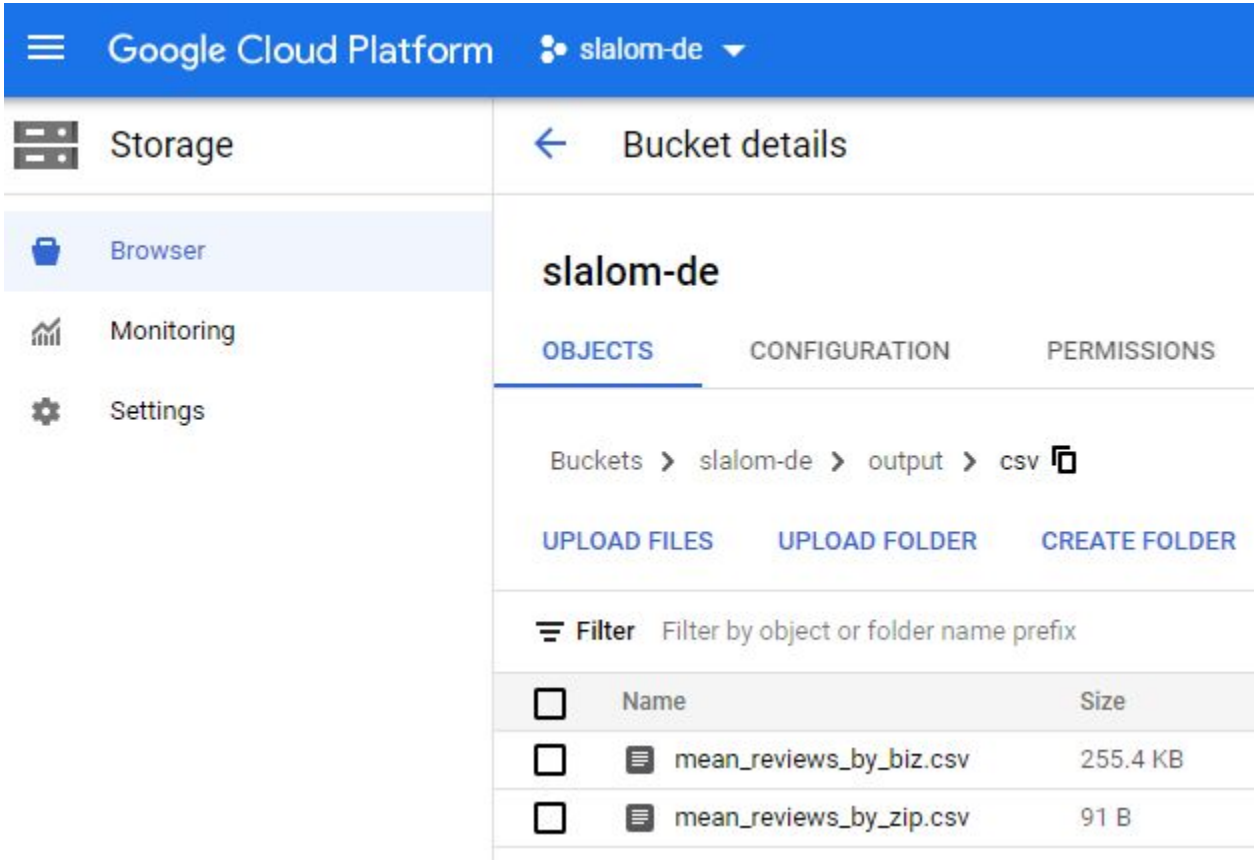
load_mean_rev_by_zip.py

```python
def run():
    parser = argparse.ArgumentParser()
    parser.add_argument('--output_file',
                        dest='output_file',
                        required=True,
                        help='Provide output file path')
    parser.add_argument('--bucket_name',
                        dest='bucket_name',
                        required=True,
                        help='bucket name for output file')
    parser.add_argument('--credentials_file',
                        dest='credentials_file',
                        required=True,
                        help='Service Account Credentials file path')
    args = parser.parse_args()

    bucket_name = args.bucket_name
    output_file = args.output_file
    credentials_file = args.credentials_file

    #set credentials using the json file
    credentials = set_credentials(credentials_file)

    #execute the query to extract data from bq
    df_mean_reviews_by_zip = get_mean_reviews_by_zip(credentials,bucket_name)

    #create the file in gcs
    load_file_gcs(bucket_name,output_file,df_mean_reviews_by_zip)

if __name__ == '__main__':
    run()

```

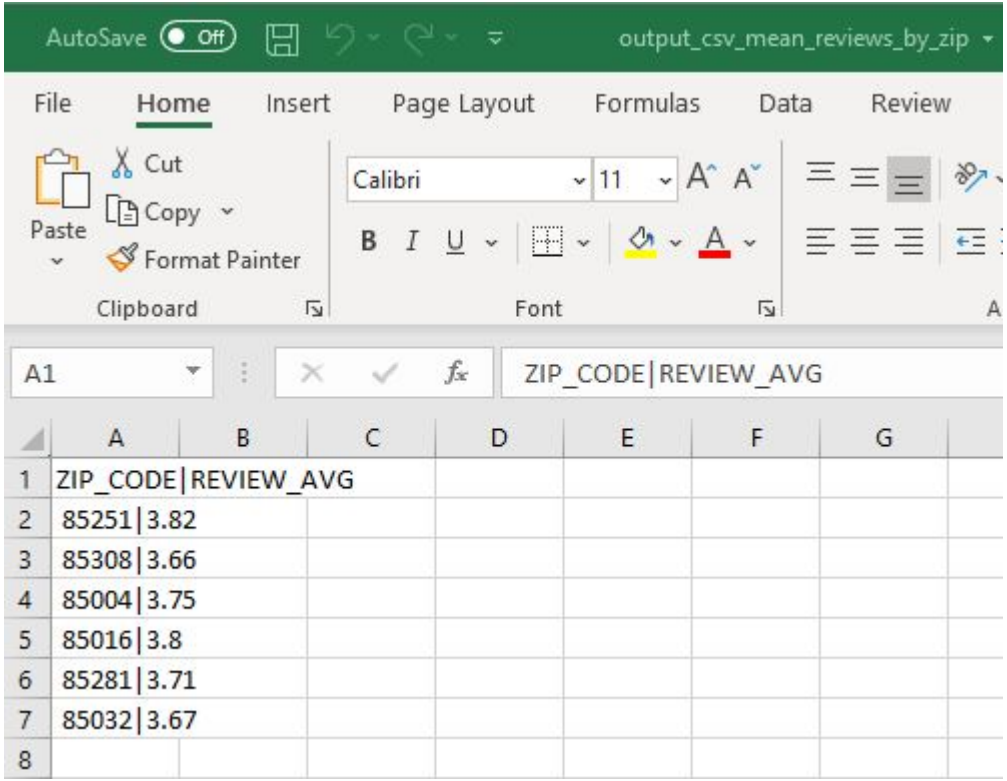
Top 5 Mean Reviews By Zip

```
(slalom-de) vijay_gopu@cloudshell:~/slalom-de (slalom-de)$ python load_mean_rev_by_zip.py \
> --output_file output/csv/mean_reviews_by_zip.csv \
> --bucket_name slalom-de \
> --credentials_file /home/vijay_gopu/slalom-de-911419ed4b1d.json
```

Top 5 Mean Reviews By Zip



Top 5 Mean Reviews By Zip



	A	B	C	D	E	F	G
1	ZIP_CODE|REVIEW_AVG						
2	85251|3.82						
3	85308|3.66						
4	85004|3.75						
5	85016|3.8						
6	85281|3.71						
7	85032|3.67						
8							

Top 10 Active Reviewers

load_top5_active_reviewers.py

```python
import pandas as pd
import argparse
#import pandas_gbq
from google.cloud import storage
from google.oauth2 import service_account

'''
  Must export GOOGLE_APPLICATION_CREDENTIALS
'''

def set_credentials(credentials_file):
  #Credentials required to query data
  credentials = service_account.Credentials.from_service_account_file(
    credentials_file,
  )
  return credentials

def get_top10_active_reviewers(credentials,bucket_name):
  q_top_10_reviewers = "WITH USER_NAMES AS ("
  q_top_10_reviewers += "SELECT"
  q_top_10_reviewers += "  DISTINCT USER_ID, USER_NAME "
  q_top_10_reviewers += "FROM `slalom-de.slalom.users`) "
  q_top_10_reviewers += "SELECT * FROM ("
  q_top_10_reviewers += " SELECT USER_NAME, TOT_REVIEWS, RANK() OVER (ORDER BY TOT_REVIEWS DESC) AS RANK_ FROM ("
  q_top_10_reviewers += "   SELECT USER_NAME, COUNT(*) AS TOT_REVIEWS FROM `slalom-de.slalom.reviews` rev, USER_NAMES"
  q_top_10_reviewers += "   WHERE rev.User_Id = USER_NAMES.User_Id"
  q_top_10_reviewers += "   GROUP BY USER_NAME)) AS TOP_10 WHERE RANK_ < 11"
  #read data from bq and return dataframe
  df = pd.read_gbq(q_top_10_reviewers,project_id=bucket_name,credentials=credentials)

  return df

def load_file_gcs(bucket_name,file_name,data_frame):
  client = storage.Client()
  bucket = client.get_bucket(bucket_name)

  #create the file in the bucket
  bucket.blob(file_name).upload_from_string(data_frame.to_csv(sep="|",index=False), 'text/csv')

```

Top 10 Active Reviewers

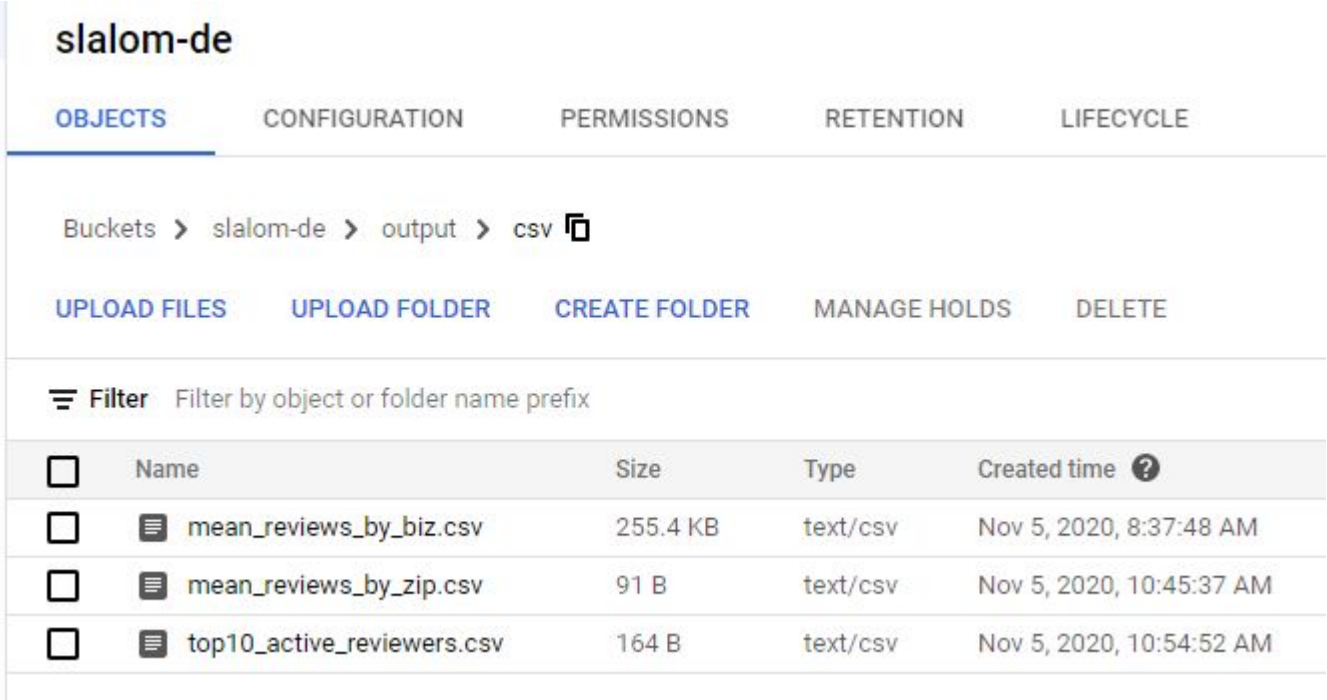
load_top5_active_reviewers.py

```
def run():
    parser = argparse.ArgumentParser()
    parser.add_argument('--output_file',
                        dest='output_file',
                        required=True,
                        help='Provide output file path')
    parser.add_argument('--bucket_name',
                        dest='bucket_name',
                        required=True,
                        help='bucket name for output file')
    parser.add_argument('--credentials_file',
                        dest='credentials_file',
                        required=True,
                        help='Service Account Credentials file path')
    args = parser.parse_args()

    bucket_name = args.bucket_name
    output_file = args.output_file
    credentials_file = args.credentials_file

    #set credentials using the json file
    credentials = set_credentials(credentials_file)

    #execute the query to extract data from bq
    df_top10_active_reviewers = get_top10_active_reviewers(credentials,bucket_name)

    #create the file in gcs
    load_file_gcs(bucket_name,output_file,df_top10_active_reviewers)

if __name__ == '__main__':
    run()
```

Top 10 Active Reviewers

```
(slalom-de) vijay_gopu@cloudshell:~/slalom-de (slalom-de)$ python load_top5_active_reviewers.py \
> --output_file output/csv/top10_active_reviewers.csv \
> --bucket_name slalom-de \
> --credentials_file /home/vijay_gopu/slalom-de-911419ed4b1d.json
```

Top 10 Active Reviewers



Top 10 Active Reviewers