Case Study 1

Amazon Books Catalog Search

Data Cleansing, Analytics, Pipeline &Reporting using Apache Beam, Python pySpark on a Google Data Proc Cluster.

PROBLEM STATEMENT:

1. Find the average user rating for Fiction books in pyspark.
2. Find the average user rating for Non-fiction books and for all books together. Please write the result in separate files. – python (analytics), PySpark
3. Sort the last 5 books in alphabetical order and print the output – Pyspark

Kaggle Dataset - https://www.kaggle.com/sootersaalu/amazon-top-50-bestselling-books-2009-2019

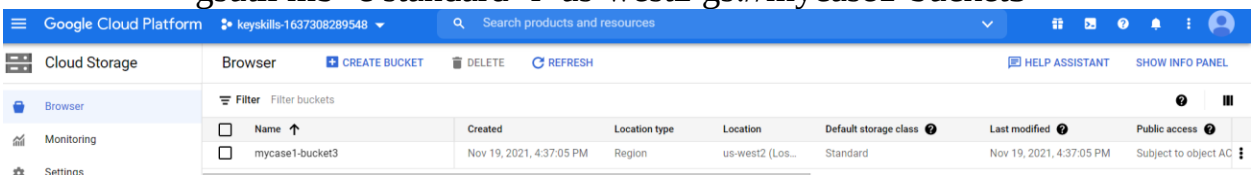
Notes - This is a dataset on Amazon's Top 50 bestselling books from 2009 to 2019., it contains 550+ books, and has been categorized into fiction and non-fiction using Goodreads.

[Note, one can add/multiple the data set to about 5000+] books to utilize power of GCP Cluster, Analytics, Spark Job, BigQuery, BigTable capabilities.

Answer:

Bucket creation using CLI:

- gsutil mb -c standard -l us-west2 gs://mycase1-bucket3

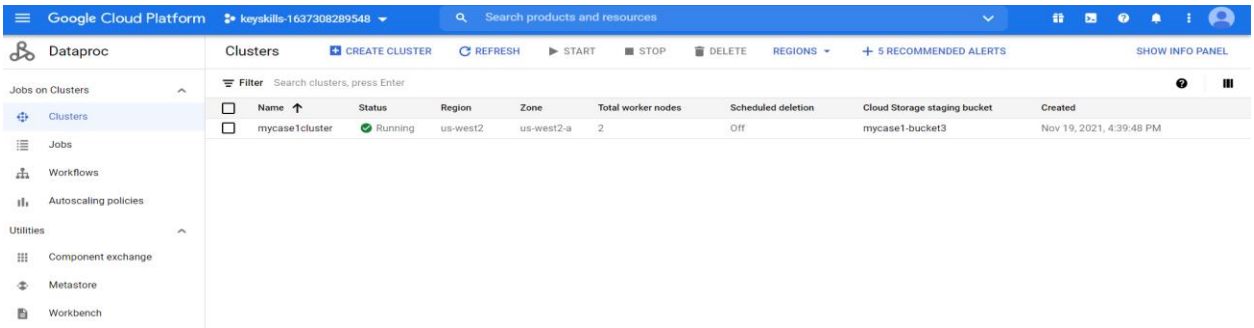


Cluster creation using CLI:

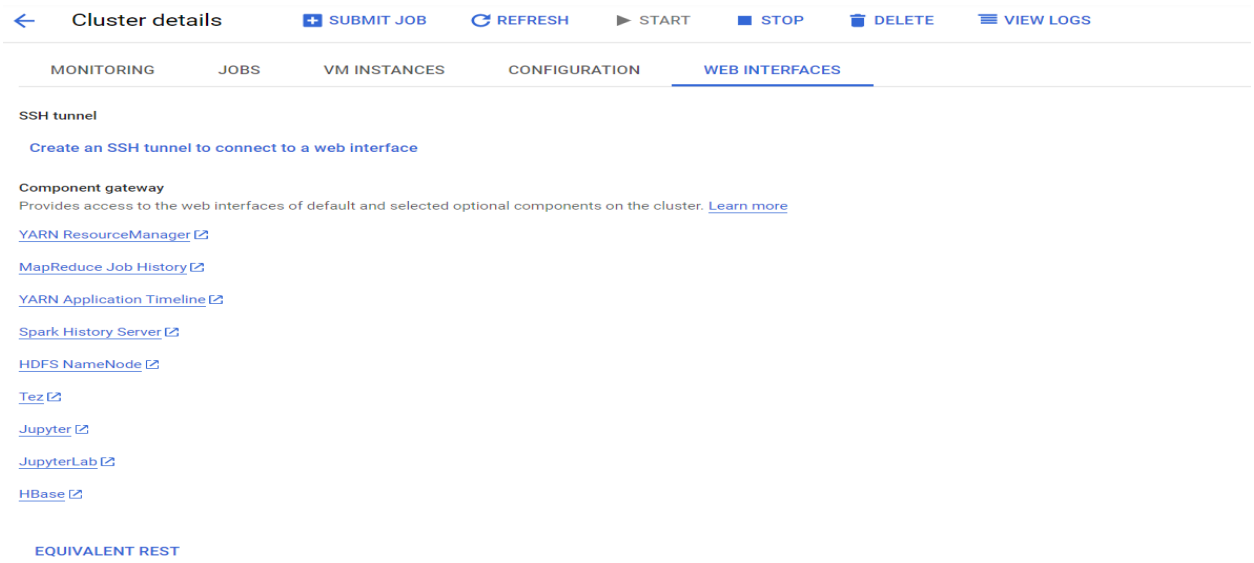
- gcloud dataproc clusters create mycase1cluster --region us-west2 --zone us-west2-a --master-machine-type n1-standard-2 --master-boot-disk-size 50 --num-workers 2 --worker-machine-type n1-standard-2 --worker-boot-disk-size 50 --image-version 2.0-ubuntu18 --optional-components JUPYTER,ZOOKEEPER,HBASE --bucket mycase1-bucket3 --enable-component-gateway --project keyskills-1637308289548

```
kshgcp_userid01@cloudshell:~ (keyskills-1637308289548)$ gsutil mb -c standard -l  us-west2 gs://mycase1-bucket3
Creating gs://mycase1-bucket3/...
kshgcp_userid01@cloudshell:~ (keyskills-1637308289548)$ gcloud dataproc clusters create mycase1cluster --region us-west2 --zone us-west2-a --master-machine-type n1-standard-2 --master-boot-disk
-size 50 --num-workers 2 --worker-machine-type n1-standard-2 --worker-boot-disk-size 50 --image-version 2.0-ubuntu18 --optional-components JUPYTER,ZOOKEEPER,HBASE --bucket  mycase1-bucket3 --en
able-component-gateway --project keyskills-1637308289548
API [dataproc.googleapis.com] not enabled on project [397076816615]. Would you like to enable and retry (this will take a few minutes)? (y/N)?  y

Enabling service [dataproc.googleapis.com] on project [397076816615]...
Operation "operations/acf.p2-397076816615-9b754a45-0572-4661-98c0-3ba3e49bd296" finished successfully.
Waiting on operation [projects/keyskills-1637308289548/regions/us-west2/operations/4959a072-a397-3296-9696-c082b97b0607].
Waiting for cluster creation operation...
WARNING: For PD-Standard without local SSDs, we strongly recommend provisioning 1TB or larger to ensure consistently high I/O performance. See https://cloud.google.com/compute/docs/disks/perfor
mance for information on disk I/O performance.
Waiting for cluster creation operation...done.
Created [https://dataproc.googleapis.com/v1/projects/keyskills-1637308289548/regions/us-west2/clusters/mycase1cluster] Cluster placed in zone [us-west2-a].
kshgcp_userid01@cloudshell:~ (keyskills-1637308289548)$
```

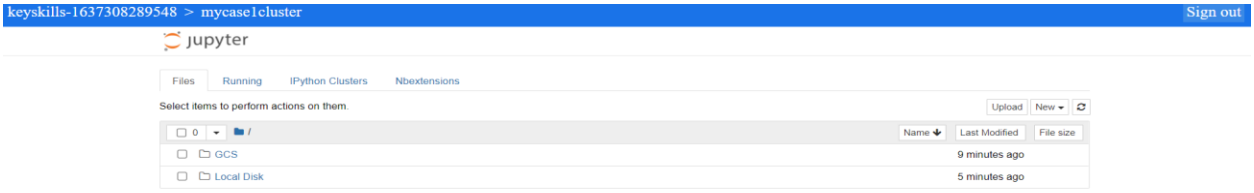
Cluster Created Using Dataproc



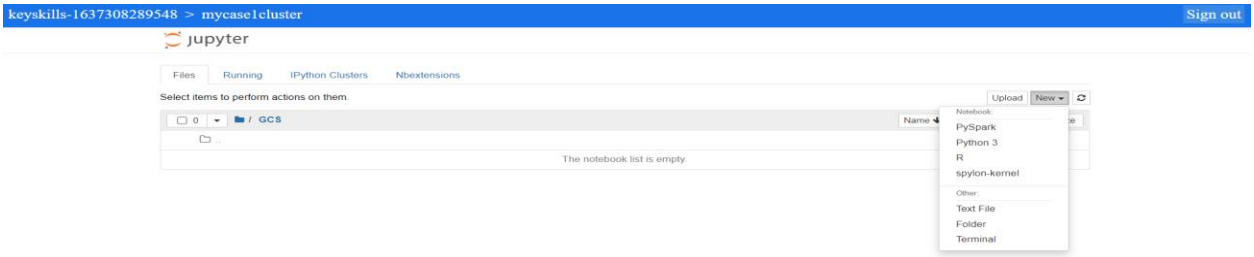
Cluster has all the web interfaces we opted



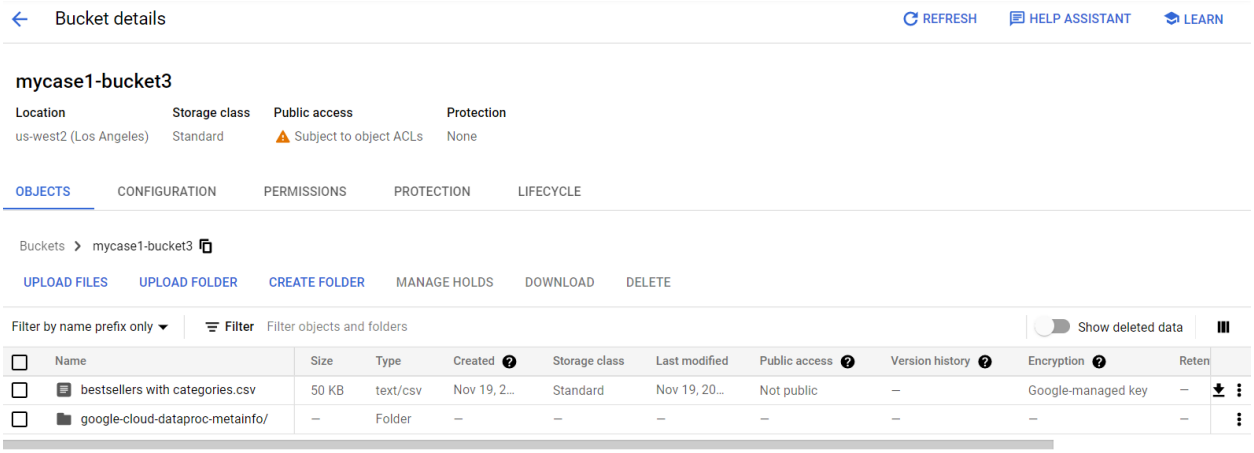
Click on Jupyter



Click on GCS and Select Pyspark



Upload the kaggle data set into the GCS bucket and copy its path



Upload the csv and perform data cleaning i.e replace the commas in the Name column with semicolons:

- import pyspark
- from pyspark.sql import SparkSession
- spark = SparkSession.builder.appName('temp').getOrCreate()
- df = spark.read.csv("gs://mycase1-bucket3/bestsellers with categories.csv", header=True, inferSchema = True)
- df.show(truncate=False)

File Edit View Insert Cell Kernel Widgets Help Trusted PySpark

In [1]:
```
import pyspark
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName('temp').getOrCreate()
df = spark.read.csv("gs://mycase1-bucket3/bestsellers with categories.csv", header=True, inferSchema = True)
df.show(truncate=False)
```

```
+-----------------------------------------------------------------------------------------------------------------------------------------+--------
-------------------+-----------+-------+-----+----+-----------+
|Name                                                                                                                                     |Author
|User Rating|Reviews|Price|Year|Genre      |
+-----------------------------------------------------------------------------------------------------------------------------------------+--------
-------------------+-----------+-------+-----+----+-----------+
|10-Day Green Smoothie Cleanse                                                                                                            |JJ Smith
|4.7        |17350  |8    |2016|Non Fiction|
|11/22/63: A Novel                                                                                                                        |Stephen
King               |4.6        |2052   |22   |2011|Fiction    |
|12 Rules for Life: An Antidote to Chaos                                                                                                  |Jordan
B. Peterson         |4.7        |18979  |15   |2018|Non Fiction|
|1984 (Signet Classics)                                                                                                                   |George O
rwell              |4.7        |21424  |6    |2017|Fiction    |
|5,000 Awesome Facts (About Everything!) (National Geographic Kids)                                                                       |National
Geographic Kids   |4.8        |7665   |12   |2019|Non Fiction|
|A Dance with Dragons (A Song of Ice and Fire)                                                                                            |George
R. R. Martin        |4.4        |12643  |11   |2011|Fiction    |
|A Game of Thrones / A Clash of Kings / A Storm of Swords / A Feast of Crows / A Dance with Dragons                                       |George
R. R. Martin        |4.7        |19735  |30   |2014|Fiction    |
|A Gentleman in Moscow: A Novel                                                                                                           |Amor Tow
les                |4.7        |19699  |15   |2017|Fiction    |
|A Higher Loyalty: Truth, Lies, and Leadership                                                                                            |James Co
mey                |4.7        |5983   |3    |2018|Non Fiction|
|A Man Called Ove: A Novel                                                                                                                |Fredrik
Backman            |4.6        |23848  |8    |2016|Fiction    |
|A Man Called Ove: A Novel                                                                                                                |Fredrik
Backman            |4.6        |23848  |8    |2017|Fiction    |
|A Patriot's History of the United States: From Columbus's Great Discovery to the War on Terror                                           |Larry Sc
hweikart           |4.6        |460    |2    |2010|Non Fiction|
|A Stolen Life: A Memoir                                                                                                                  |Jaycee D
ugard              |4.6        |4149   |32   |2011|Non Fiction|
|A Wrinkle in Time (Time Quintet)                                                                                                         |Madelein
e L'Engle          |4.5        |5153   |5    |2018|Fiction    |
|Act Like a Lady, Think Like a Man: What Men Really Think About Love, Relationships, Intimacy, and Commitment                             |Steve Ha
```

- from pyspark.sql.functions import translate
- df = df.withColumn('Name', translate('Name', ',', ';'))
- df.show(truncate=False)

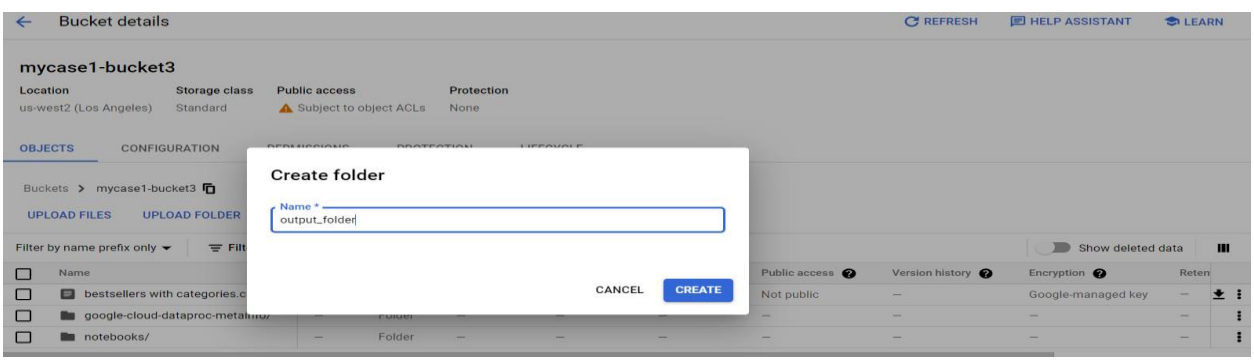
File Edit View Insert Cell Kernel Widgets Help Trusted PySpark

In [2]:
```
from pyspark.sql.functions import translate
df = df.withColumn('Name', translate('Name', ',', ';'))
df.show(truncate=False)
```

```
+-----------------------------------------------------------------------------------------------------------------------------------+--------
------------------+-----------+-------+-----+----+-----------+
|Name                                                                                                                               |Author
|User Rating|Reviews|Price|Year|Genre      |
+-----------------------------------------------------------------------------------------------------------------------------------+--------
------------------+-----------+-------+-----+----+-----------+
|10-Day Green Smoothie Cleanse                                                                                                      |JJ Smith
|4.7        |17350  |8    |2016|Non Fiction|
|11/22/63: A Novel                                                                                                                  |Stephen
King               |4.6        |2052   |22   |2011|Fiction    |
|12 Rules for Life: An Antidote to Chaos                                                                                            |Jordan
B. Peterson         |4.7        |18979  |15   |2018|Non Fiction|
|1984 (Signet Classics)                                                                                                             |George O
rwell              |4.7        |21424  |6    |2017|Fiction    |
|5;000 Awesome Facts (About Everything!) (National Geographic Kids)                                                                 |National
Geographic Kids   |4.8        |7665   |12   |2019|Non Fiction|
|A Dance with Dragons (A Song of Ice and Fire)                                                                                      |George
R. R. Martin        |4.4        |12643  |11   |2011|Fiction    |
|A Game of Thrones / A Clash of Kings / A Storm of Swords / A Feast of Crows / A Dance with Dragons                                |George
R. R. Martin        |4.7        |19735  |30   |2014|Fiction    |
|A Gentleman in Moscow: A Novel                                                                                                     |Amor Tow
les                |4.7        |19699  |15   |2017|Fiction    |
|A Higher Loyalty: Truth; Lies; and Leadership                                                                                      |James Co
mey                |4.7        |5983   |3    |2018|Non Fiction|
|A Man Called Ove: A Novel                                                                                                          |Fredrik
Backman            |4.6        |23848  |8    |2016|Fiction    |
|A Man Called Ove: A Novel                                                                                                          |Fredrik
Backman            |4.6        |23848  |8    |2017|Fiction    |
|A Patriot's History of the United States: From Columbus's Great Discovery to the War on Terror                                     |Larry Sc
hweikart           |4.6        |460    |2    |2010|Non Fiction|
|A Stolen Life: A Memoir                                                                                                            |Jaycee D
ugard              |4.6        |4149   |32   |2011|Non Fiction|
|A Wrinkle in Time (Time Quintet)                                                                                                   |Madelein
e L'Engle          |4.5        |5153   |5    |2018|Fiction    |
|Act Like a Lady; Think Like a Man: What Men Really Think About Love; Relationships; Intimacy; and Commitment                       |Steve Ha
rvey               |4.6        |5013   |17   |2009|Non Fiction|
|Adult Coloring Book Designs: Stress Relief Coloring Book: Garden Designs; Mandalas; Animals; and Paisley Patterns                  |Adult Co
loring Book Designs|4.5        |2313   |4    |2016|Non Fiction|
```

1. Finding average of user ratings for Genre = Fiction and saving the file in output_folder :

- df_fiction = df.filter("Genre == 'Fiction'")
- df_fiction.show(10)
- fiction_avg = df_fiction.agg({'User Rating': 'avg'})
- fiction_avg.show()
- fiction_avg.write.save("gs://mycase1-bucket3/output_folder/fiction_avg.csv", format = 'csv', header = True)



```
In [3]: df_fiction = df.filter("Genre == 'Fiction'")
        df_fiction.show(10)
        fiction_avg = df_fiction.agg({'User Rating': 'avg'})
        fiction_avg.show()
        fiction_avg.write.save("gs://mycase1-bucket3/output_folder/fiction_avg.csv", format = 'csv', header = True)
```

```
+--------------------+-------------------+-----------+-------+-----+----+-------+
|                Name|             Author|User Rating|Reviews|Price|Year|  Genre|
+--------------------+-------------------+-----------+-------+-----+----+-------+
|   11/22/63: A Novel|       Stephen King|        4.6|   2052|   22|2011|Fiction|
|1984 (Signet Clas...|      George Orwell|        4.7|  21424|    6|2017|Fiction|
|A Dance with Drag...|George R. R. Martin|        4.4|  12643|   11|2011|Fiction|
|A Game of Thrones...|George R. R. Martin|        4.7|  19735|   30|2014|Fiction|
|A Gentleman in Mo...|        Amor Towles|        4.7|  19699|   15|2017|Fiction|
|A Man Called Ove:...|    Fredrik Backman|        4.6|  23848|    8|2016|Fiction|
|A Man Called Ove:...|    Fredrik Backman|        4.6|  23848|    8|2017|Fiction|
|A Wrinkle in Time...|  Madeleine L'Engle|        4.5|   5153|    5|2018|Fiction|
|All the Light We ...|      Anthony Doerr|        4.6|  36348|   14|2014|Fiction|
|All the Light We ...|      Anthony Doerr|        4.6|  36348|   14|2015|Fiction|
+--------------------+-------------------+-----------+-------+-----+----+-------+
only showing top 10 rows

+-----------------+
| avg(User Rating)|
+-----------------+
|4.648333333333326|
+-----------------+
```

2. Finding average of user ratings for Genre =Non Fiction and saving the file in output_folder :

- df_non_fiction = df.filter("Genre == 'Non Fiction'")
- df_non_fiction.show(10)
- non_fiction_avg = df_non_fiction.agg({'User Rating': 'avg'})
- non_fiction_avg.show()
- non_fiction_avg.write.save("gs://mycase1-bucket3/output_folder/non_fiction_avg.csv", format = 'csv', header = True)

```
In [4]: df_non_fiction = df.filter("Genre == 'Non Fiction'")
        df_non_fiction.show(10)
        non_fiction_avg = df_non_fiction.agg({'User Rating': 'avg'})
        non_fiction_avg.show()
        non_fiction_avg.write.save("gs://mycase1-bucket3/output_folder/non_fiction_avg.csv", format = 'csv', header = True)
```

```
+--------------------+--------------------+-----------+-------+-----+----+-----------+
|                Name|              Author|User Rating|Reviews|Price|Year|      Genre|
+--------------------+--------------------+-----------+-------+-----+----+-----------+
|10-Day Green Smoo...|            JJ Smith|        4.7|  17350|    8|2016|Non Fiction|
|12 Rules for Life...|   Jordan B. Peterson|        4.7|  18979|   15|2018|Non Fiction|
|5;000 Awesome Fac...|National Geograph...|        4.8|   7665|   12|2019|Non Fiction|
|A Higher Loyalty:...|          James Comey|        4.7|   5983|    3|2018|Non Fiction|
|A Patriot's Histo...|     Larry Schweikart|        4.6|    460|    2|2010|Non Fiction|
|A Stolen Life: A ...|        Jaycee Dugard|        4.6|   4149|   32|2011|Non Fiction|
|Act Like a Lady; ...|         Steve Harvey|        4.6|   5013|   17|2009|Non Fiction|
|Adult Coloring Bo...|Adult Coloring Bo...|        4.5|   2313|    4|2016|Non Fiction|
|Adult Coloring Bo...|   Blue Star Coloring|        4.6|   2925|    6|2015|Non Fiction|
|Adult Coloring Bo...|   Blue Star Coloring|        4.4|   2951|    6|2015|Non Fiction|
+--------------------+--------------------+-----------+-------+-----+----+-----------+
only showing top 10 rows

+-----------------+
| avg(User Rating)|
+-----------------+
|4.595161290322579|
+-----------------+
```

Finding average of user ratings for all the books and saving the file in output_folder :

- avg_of_all = df.agg({'User Rating': 'avg'})

- avg_of_all.show()
- avg_of_all.write.save("gs://mycase1-bucket3/output_folder/avg_of_all.csv", format = 'csv', header = True)

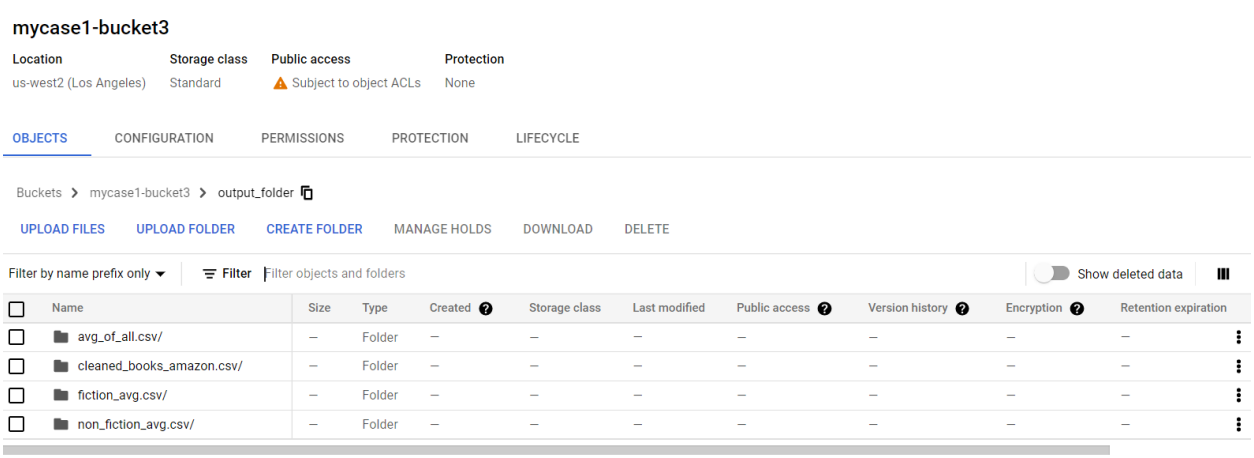
```
In [5]: avg_of_all = df.agg({'User Rating': 'avg'})
        avg_of_all.show()
        avg_of_all.write.save("gs://mycase1-bucket3/output_folder/avg_of_all.csv", format = 'csv', header = True)

        +-----------------+
        | avg(User Rating)|
        +-----------------+
        |4.618363636363641|
        +-----------------+
```

3. Sorting the last 5 books in alphabetical order and printing the output:

   - df.orderBy(df.Name.desc()).limit(5).show()
   - df.write.csv('gs://mycase1-bucket3/output_folder/cleaned_books_amazon.csv')

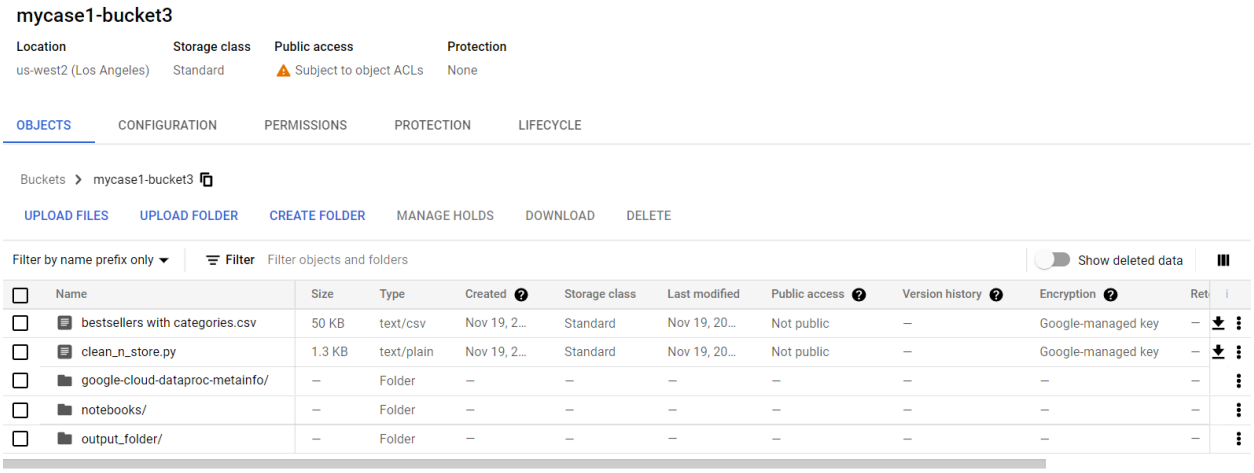
```
In [6]: df.orderBy(df.Name.desc()).limit(5).show()
        df.write.csv('gs://mycase1-bucket3/output_folder/cleaned_books_amazon.csv')

        +--------------------+-----------+-----------+-------+-----+----+-----------+
        |                Name|     Author|User Rating|Reviews|Price|Year|      Genre|
        +--------------------+-----------+-----------+-------+-----+----+-----------+
        |You Are a Badass:...|Jen Sincero|        4.7|  14331|    8|2018|Non Fiction|
        |You Are a Badass:...|Jen Sincero|        4.7|  14331|    8|2016|Non Fiction|
        |You Are a Badass:...|Jen Sincero|        4.7|  14331|    8|2017|Non Fiction|
        |You Are a Badass:...|Jen Sincero|        4.7|  14331|    8|2019|Non Fiction|
        |Wrecking Ball (Di...|Jeff Kinney|        4.9|   9413|    8|2019|    Fiction|
        +--------------------+-----------+-----------+-------+-----+----+-----------+
```

Output folder with all the output files:



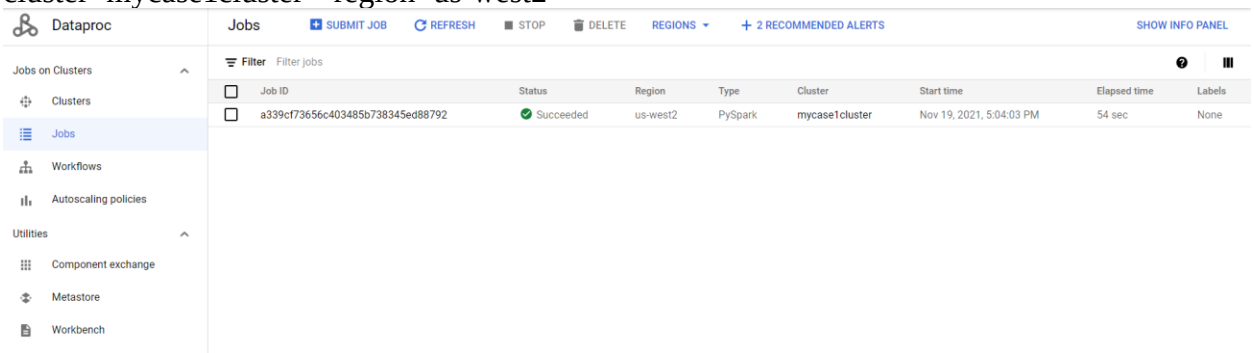
Job Running:

Save the Pyspark notebook file as clean_n_store.py and upload it in the mycase1-bucket3 bucket.



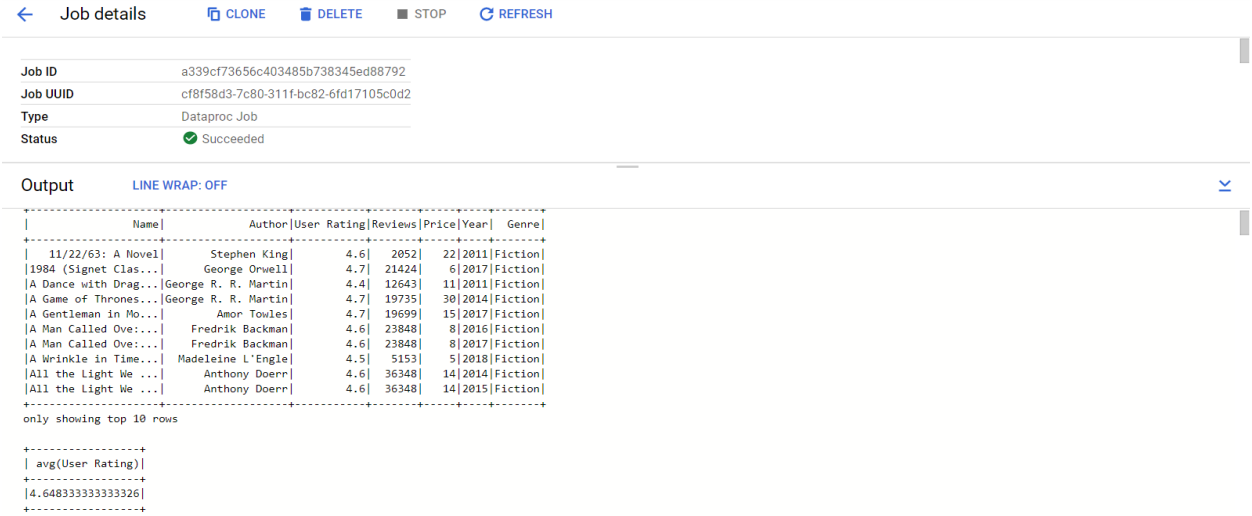
Running CLI command for Dataproc job:

- gcloud dataproc jobs submit pyspark gs://mycase1-bucket3/clean_n_store.py --cluster=mycase1cluster --region=us-west2



Outputs obtained from Dataproc job:

Average of Fiction:



Average of Non-Fiction:

Output LINE WRAP: OFF

```
+--------------------+--------------------+-----------+-------+-----+----+-----------+
|                Name|              Author|User Rating|Reviews|Price|Year|      Genre|
+--------------------+--------------------+-----------+-------+-----+----+-----------+
|10-Day Green Smoo...|            JJ Smith|        4.7|  17350|    8|2016|Non Fiction|
|12 Rules for Life...|  Jordan B. Peterson|        4.7|  18979|   15|2018|Non Fiction|
|5;000 Awesome Fac...|National Geograph...|        4.8|   7665|   12|2019|Non Fiction|
|A Higher Loyalty:...|          James Comey|        4.7|   5983|    3|2018|Non Fiction|
|A Patriot's Histo...|     Larry Schweikart|        4.6|    460|    2|2010|Non Fiction|
|A Stolen Life: A ...|        Jaycee Dugard|        4.6|   4149|   32|2011|Non Fiction|
|Act Like a Lady; ...|         Steve Harvey|        4.6|   5013|   17|2009|Non Fiction|
|Adult Coloring Bo...|Adult Coloring Bo...|        4.5|   2313|    4|2016|Non Fiction|
|Adult Coloring Bo...|  Blue Star Coloring|        4.6|   2925|    6|2015|Non Fiction|
|Adult Coloring Bo...|  Blue Star Coloring|        4.4|   2951|    6|2015|Non Fiction|
+--------------------+--------------------+-----------+-------+-----+----+-----------+
only showing top 10 rows

+-----------------+
| avg(User Rating)|
+-----------------+
|4.595161290322579|
+-----------------+
```

Average of all:

Output LINE WRAP: OFF

```
21/11/19 11:34:48 INFO com.google.cloud.hadoop.repackaged.gcs.com.google.cloud.hadoop.gcsio.GoogleCloudStorageFileSystem: Successfully repaired 'gs://mycase1-bucket3/output_folder/non_fict
+-----------------+
| avg(User Rating)|
+-----------------+
|4.618363636363641|
+-----------------+

21/11/19 11:34:51 INFO com.google.cloud.hadoop.repackaged.gcs.com.google.cloud.hadoop.gcsio.GoogleCloudStorageFileSystem: Successfully repaired 'gs://mycase1-bucket3/output_folder/avg_of_a
+--------------------+----------+-----------+-------+-----+----+-----------+
|                Name|    Author|User Rating|Reviews|Price|Year|      Genre|
+--------------------+----------+-----------+-------+-----+----+-----------+
|You Are a Badass:...|Jen Sincero|        4.7|  14331|    8|2018|Non Fiction|
|You Are a Badass:...|Jen Sincero|        4.7|  14331|    8|2016|Non Fiction|
|You Are a Badass:...|Jen Sincero|        4.7|  14331|    8|2017|Non Fiction|
|You Are a Badass:...|Jen Sincero|        4.7|  14331|    8|2019|Non Fiction|
|Wrecking Ball (Di...|Jeff Kinney|        4.9|   9413|    8|2019|    Fiction|
+--------------------+----------+-----------+-------+-----+----+-----------+

21/11/19 11:34:55 INFO com.google.cloud.hadoop.repackaged.gcs.com.google.cloud.hadoop.gcsio.GoogleCloudStorageFileSystem: Successfully repaired 'gs://mycase1-bucket3/output_folder/cleaned_
21/11/19 11:34:55 INFO org.sparkproject.jetty.server.AbstractConnector: Stopped Spark@3fee8cd1{HTTP/1.1, (http/1.1)}{0.0.0.0:0}
```

Output files in the output folder:

mycase1-bucket3

Location	Storage class	Public access	Protection
us-west2 (Los Angeles)	Standard	Subject to object ACLs	None

OBJECTS CONFIGURATION PERMISSIONS PROTECTION LIFECYCLE

Buckets > mycase1-bucket3 > output_folder

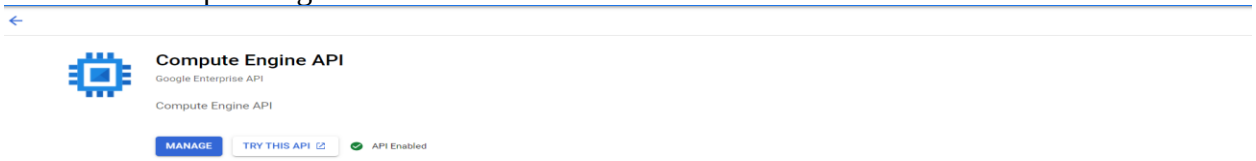
UPLOAD FILES UPLOAD FOLDER CREATE FOLDER MANAGE HOLDS DOWNLOAD DELETE

Filter by name prefix only Filter Filter objects and folders Show deleted data

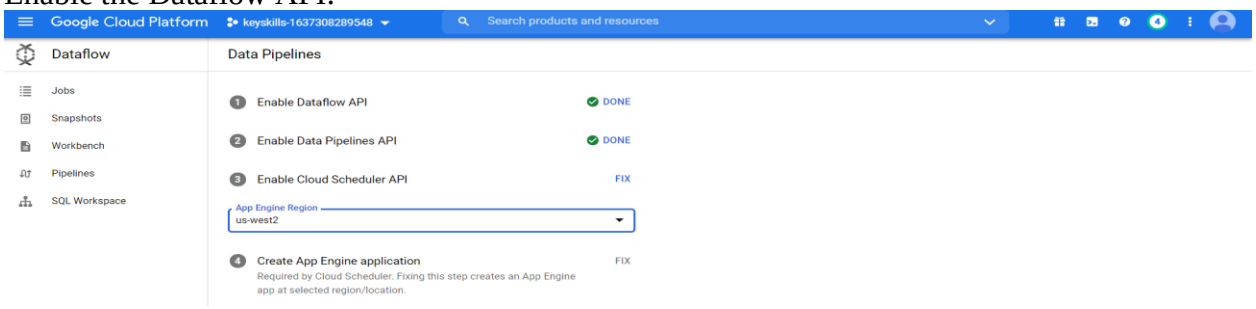
Name	Size	Type	Created	Storage class	Last modified	Public access	Version history	Encryption	Retention expiration
avg_of_all.csv/	–	Folder	–	–	–	–	–	–	–
avg_of_all.csv2/	–	Folder	–	–	–	–	–	–	–
cleaned_books_amazon.csv/	–	Folder	–	–	–	–	–	–	–
cleaned_books_amazon2.csv/	–	Folder	–	–	–	–	–	–	–
fiction_avg.csv/	–	Folder	–	–	–	–	–	–	–
fiction_avg2.csv/	–	Folder	–	–	–	–	–	–	–
non_fiction_avg.csv/	–	Folder	–	–	–	–	–	–	–
non_fiction_avg2.csv/	–	Folder	–	–	–	–	–	–	–

Create/Configure a pipeline:

Enable the compute engine API:

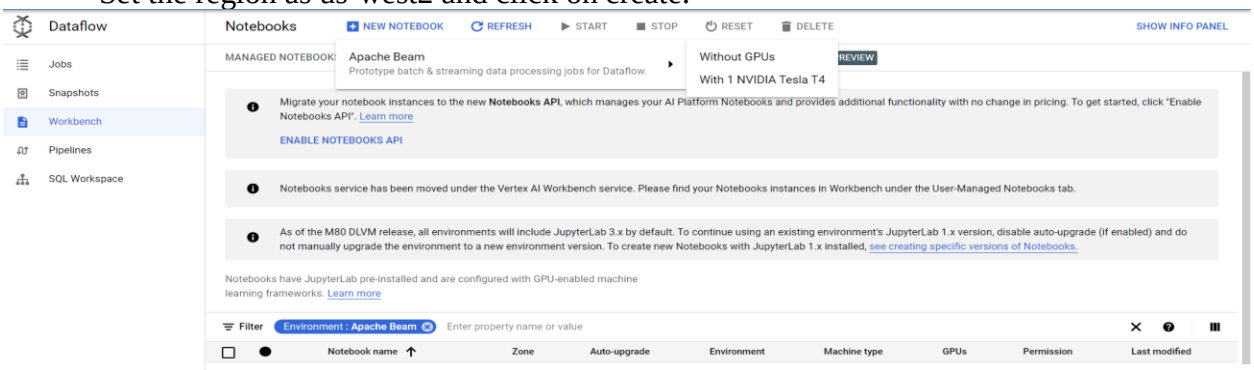


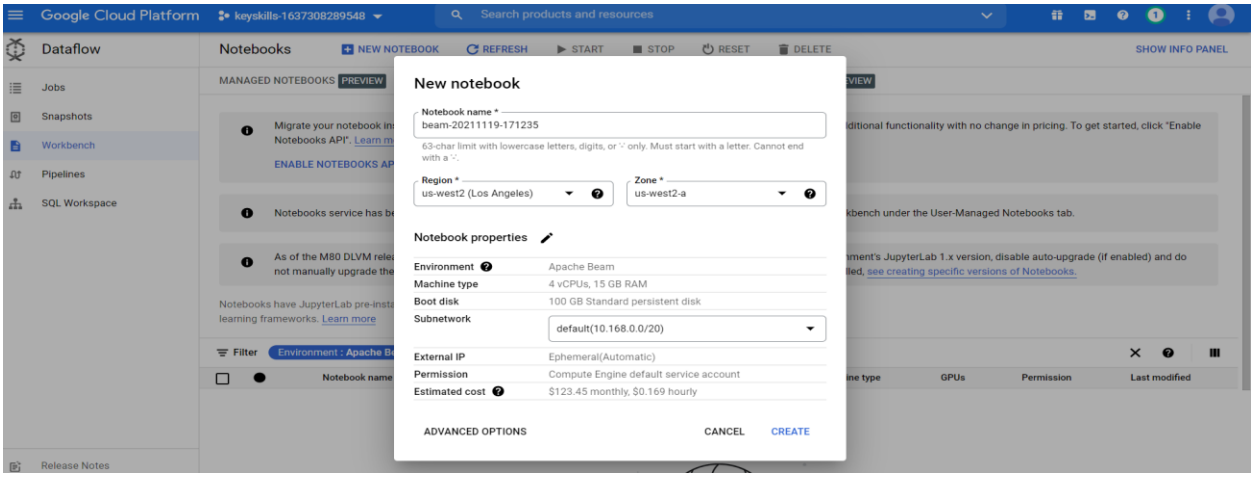
Enable the Dataflow API:



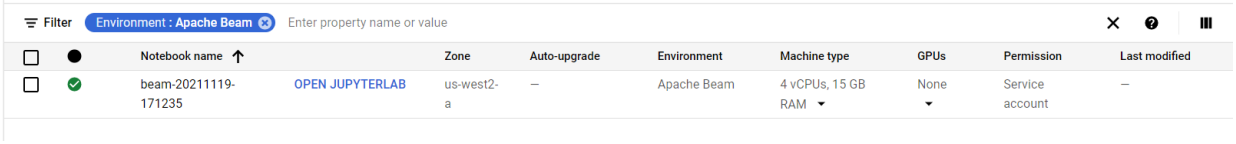
Configuring the Data Pipeline:

- Go to dataflow, click on workbench.
- Click on new notebook →Apache Beam →Without GPUs.
- Set the region as us-west2 and click on create.



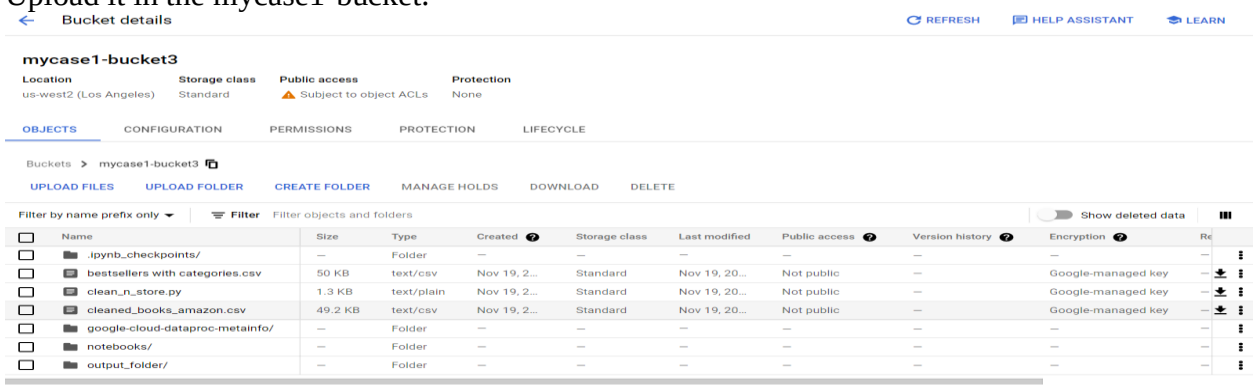


Click on open Jupyter lab:

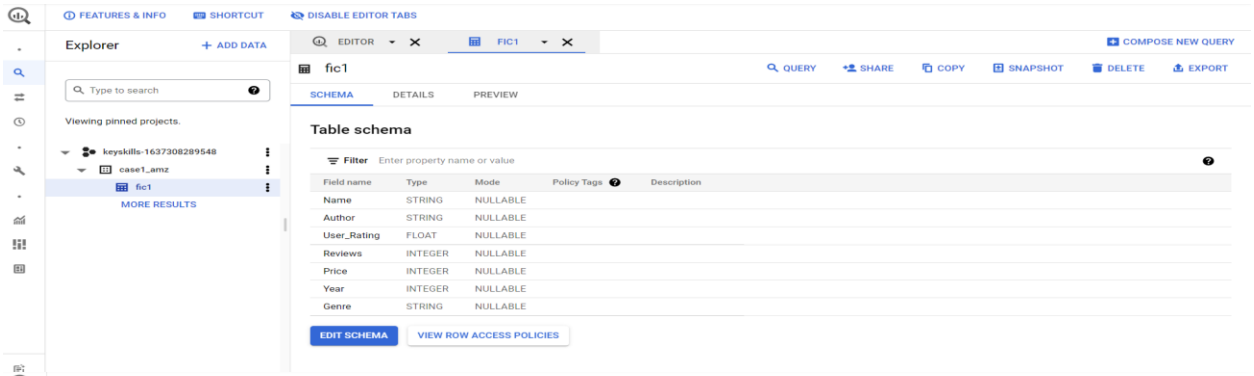


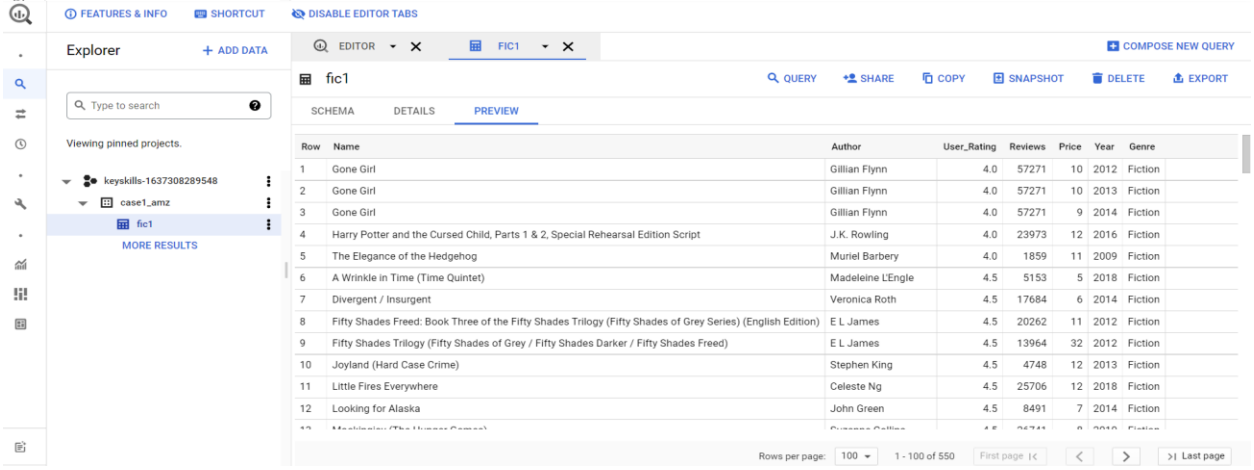
Before running the pipeline code we need to create a BigQuery view of the given dataset.

- Download the cleaned dataset file obtained in the output_folder after running the PySpark commands in notebook.
- Upload it in the mycase1-bucket.



- Go to BigQuery and create a dataset with dataset id case1_amz.
- Create table view for bestsellers with categories.csv file. Here the table name is set at fic1.





- Create a output bucket to store the outputs by name case_output_bucket1.

Run the pipeline code in jupyter lab:

```
from __future__ import absolute_import

import argparse

import logging

import re

import apache_beam as beam

from apache_beam.options.pipeline_options import PipelineOptions

import os

import google.auth

from apache_beam.options import pipeline_options

from apache_beam.options.pipeline_options import GoogleCloudOptions

from apache_beam.runners import DataflowRunner

from google.cloud import bigquery


def Split(element):

    return element.split(",")

#outputs text in a readable format

def FormatText(elem):

    return 'AVERAGE RATING OF BOOKS:'+str(elem[1])

#finds the average of given list of sets

class AverageFn(beam.CombineFn):

    def create_accumulator(self):

        return (0.0, 0)   # initialize (sum, count)

    def add_input(self, sum_count, inputt):
```

```
        (summ, count) = sum_count

        (summ2, c2) = inputt

        return summ + float(summ2)*c2, count + c2

options = PipelineOptions()

#p = beam.Pipeline(options=options)

p = beam.Pipeline(InteractiveRunner())

p2 = beam.Pipeline(InteractiveRunner())

# Setting up the Apache Beam pipeline options.

options = pipeline_options.PipelineOptions(flags=[])

# Sets the project to the default project in your current Google Cloud environment.

_, options.view_as(GoogleCloudOptions).project = google.auth.default()

# Sets the Google Cloud Region in which Cloud Dataflow runs.

options.view_as(GoogleCloudOptions).region = 'us-west2'

dataflow_gcs_location = 'gs://case_output_bucket1/dataflow'

options.view_as(GoogleCloudOptions).staging_location = '%s/staging' % dataflow_gcs_location

# Dataflow Temp Location. This location is used to store temporary files or intermediate results before finally outputting to the
sink.

options.view_as(GoogleCloudOptions).temp_location = '%s/temp' % dataflow_gcs_location

client = bigquery.Client()

dataset_id = "keyskills-1637308289548.case1_amz"

#dataset = bigquery.Dataset(dataset_id)

dataset.location = "us-west2"

dataset.description = "dataset books"

#dataset_ref = client.create_dataset(dataset, timeout = 30)

# split and Convert to json

def to_json(csv_str):

    fields = csv_str.split(',')
```

```
    json_str = {"Name":fields[0],

                "Author": fields[1],

                "User_Rating": fields[2],

                "Reviews": fields[3],

                "Price": fields[4],

                "Year": fields[5],

                "Genre": fields[6]

                }

    return json_str

table_schema =
'Name:STRING,Author:STRING,User_Rating:FLOAT,Reviews:INTEGER,Price:Integer,Year:Integer,Genre:STRING'

bs = (p2 | beam.io.ReadFromText("gs://mycase1-bucket3/cleaned_books_amazon.csv"))

(bs | 'cleaned_data to json' >> beam.Map(to_json)

| 'write to bigquery' >> beam.io.WriteToBigQuery(

"keyskills-1637308289548:case1_amz.fic1",

schema=table_schema,

create_disposition=beam.io.BigQueryDisposition.CREATE_IF_NEEDED,

write_disposition=beam.io.BigQueryDisposition.WRITE_APPEND,

custom_gcs_temp_location="gs://case_output_bucket1/dataflow/temp"

) )

from apache_beam.runners.runner import PipelineState

ret = p2.run()

if ret.state == PipelineState.DONE:

    print('Success!!!')

else:

    print('Error Running beam pipeline')

#read data and split based on ‘,’
```

```
books = ( p | beam.io.ReadFromText("gs://mycase1-bucket3/cleaned_books_amazon.csv") | beam.Map(Split) )
```

#Filter records having fiction , map each rating as a set (rating,1), use combineperkey to count the number of each rating, run the average function, write result to Fiction_avg

```
res1 = (

    books

    | beam.Filter(lambda rec : rec[6]=="Fiction")

    | beam.Map(lambda rec: (rec[2], 1))

    | "Grouping keys" >> beam.CombinePerKey(sum)

    | "Combine Globally" >> beam.CombineGlobally(AverageFn())

    | "write" >> beam.io.WriteToText("gs://case_output_bucket1/Fiction_avg") )
```

#Filter records having Non Fiction , map each rating as a set (rating,1), use combineperkey to count the number of each rating, run the average function, write result to N_Fiction_avg

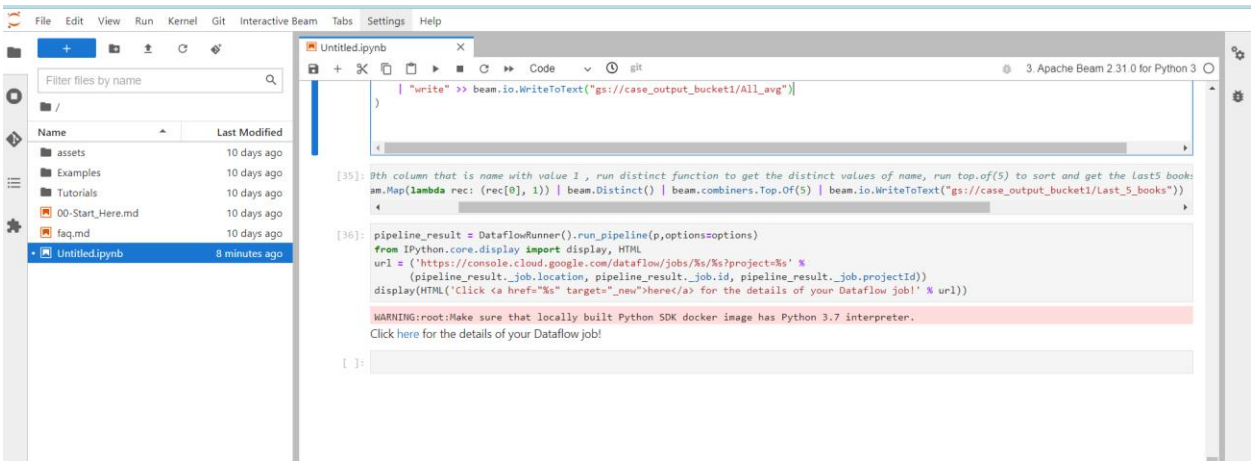
```
Res2 = (

    books

    | beam.Filter(lambda rec : rec[6]=="Non Fiction")

    | beam.Map(lambda rec: (rec[2], 1))

    | "Grouping keys" >> beam.CombinePerKey(sum)

    | "Combine Globally" >> beam.CombineGlobally(AverageFn())

    | "write" >> beam.io.WriteToText("gs://case_output_bucket1/N_Fiction_avg")

)
```

map each rating as a set (rating,1), use combineperkey to count the number of each rating, run the average function, write result to All_avg

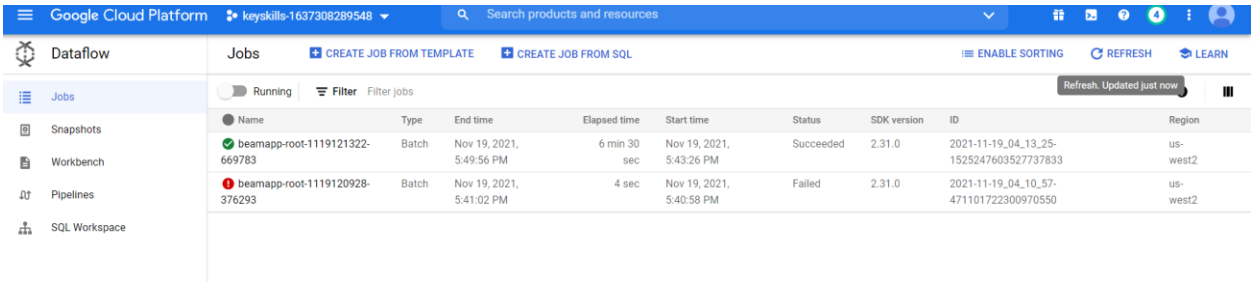
```
Res3 = (

    books

    | beam.Map(lambda rec: (rec[2], 1))

    | "Grouping keys" >> beam.CombinePerKey(sum)

    | "Combine Globally" >> beam.CombineGlobally(AverageFn())

    | "write" >> beam.io.WriteToText("gs://case_output_bucket1/All_avg") )
```

#map each record's 0th column that is name with value 1 , run distinct function to get the distinct values of name, run top.of(5) to sort and get the last5 books alphabetically and store in storage bucket last5

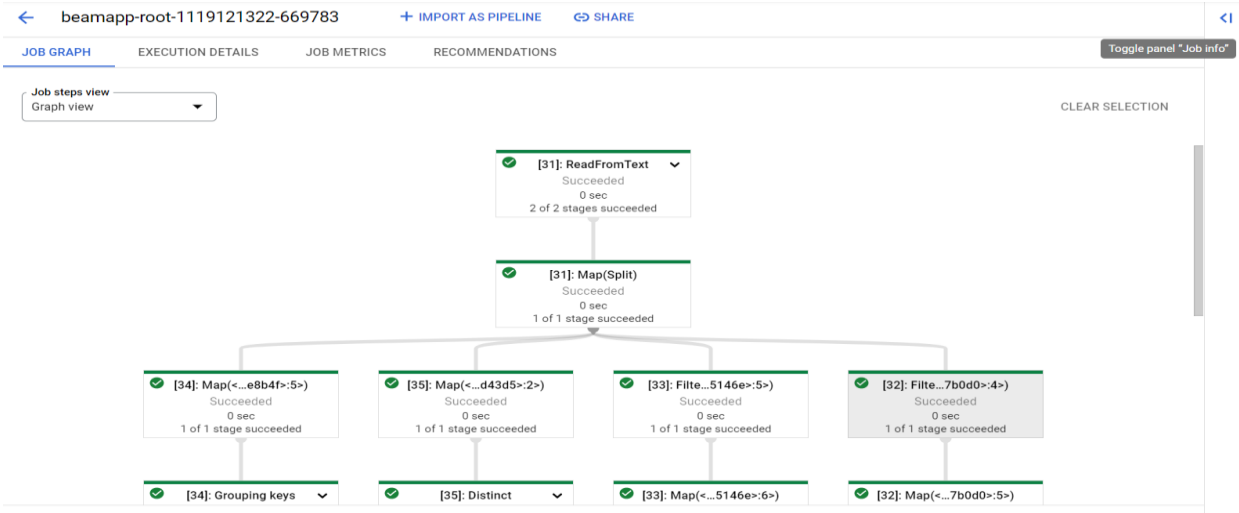
```
f_res = (books | beam.Map(lambda rec: (rec[0], 1)) | beam.Distinct() | beam.combiners.Top.Of(5) |
beam.io.WriteToText("gs://case_output_bucket1/Last_5_books"))

pipeline_result = DataflowRunner().run_pipeline(p,options=options)

from IPython.core.display import display, HTML

url = ('https://console.cloud.google.com/dataflow/jobs/%s/%s?project=%s' %

    (pipeline_result._job.location, pipeline_result._job.id, pipeline_result._job.projectId))

display(HTML('Click <a href="%s" target="_new">here</a> for the details of your Dataflow job!' % url))
```

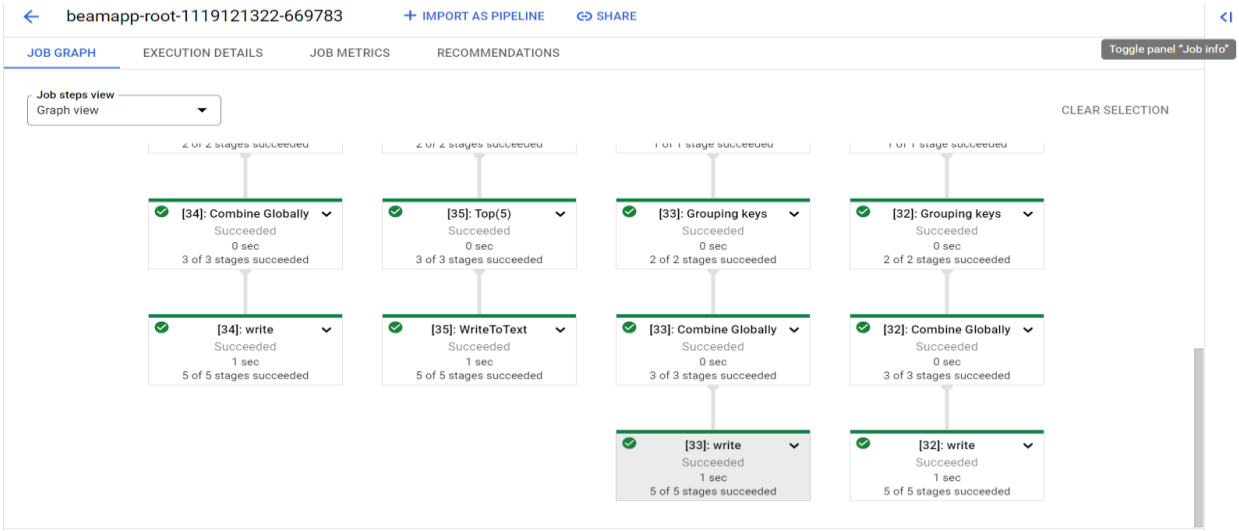


Dataflow job succeeded:

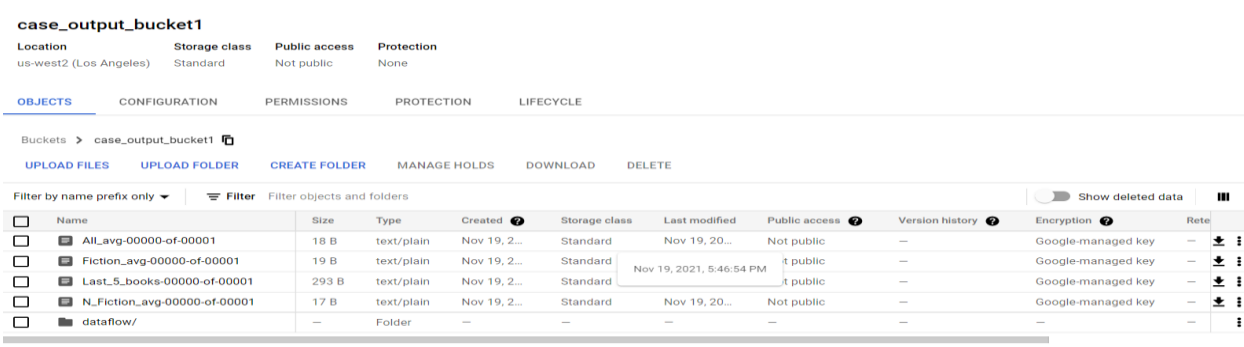


Below figures show the dataflow pipeline obtained after running the pipeline code in Apache beam for Python environment in Jupyter Lab.

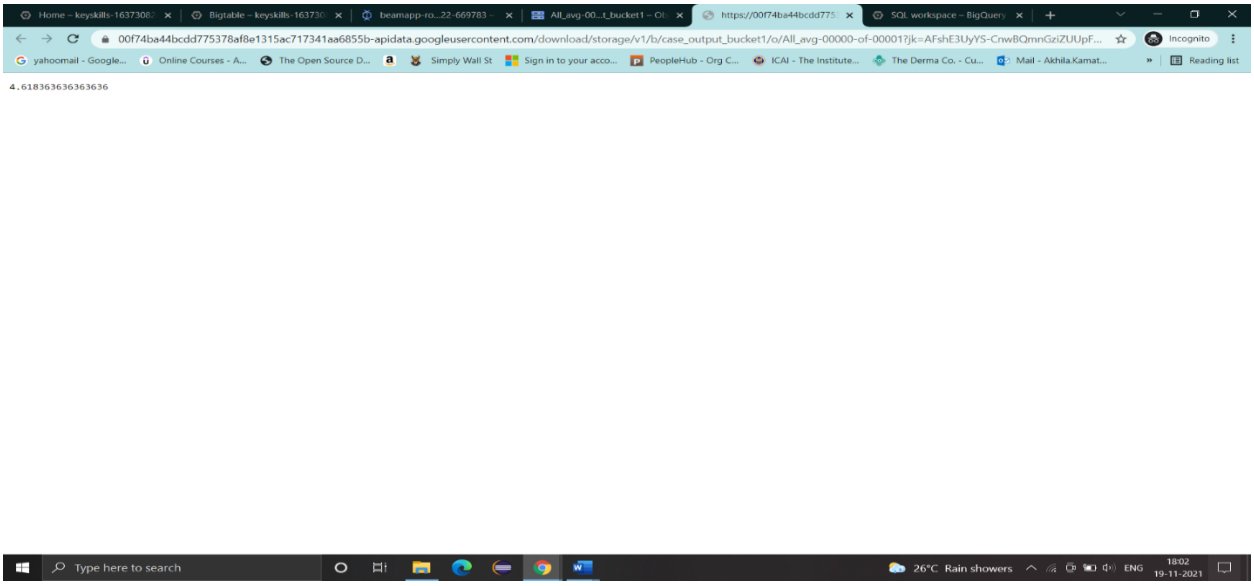




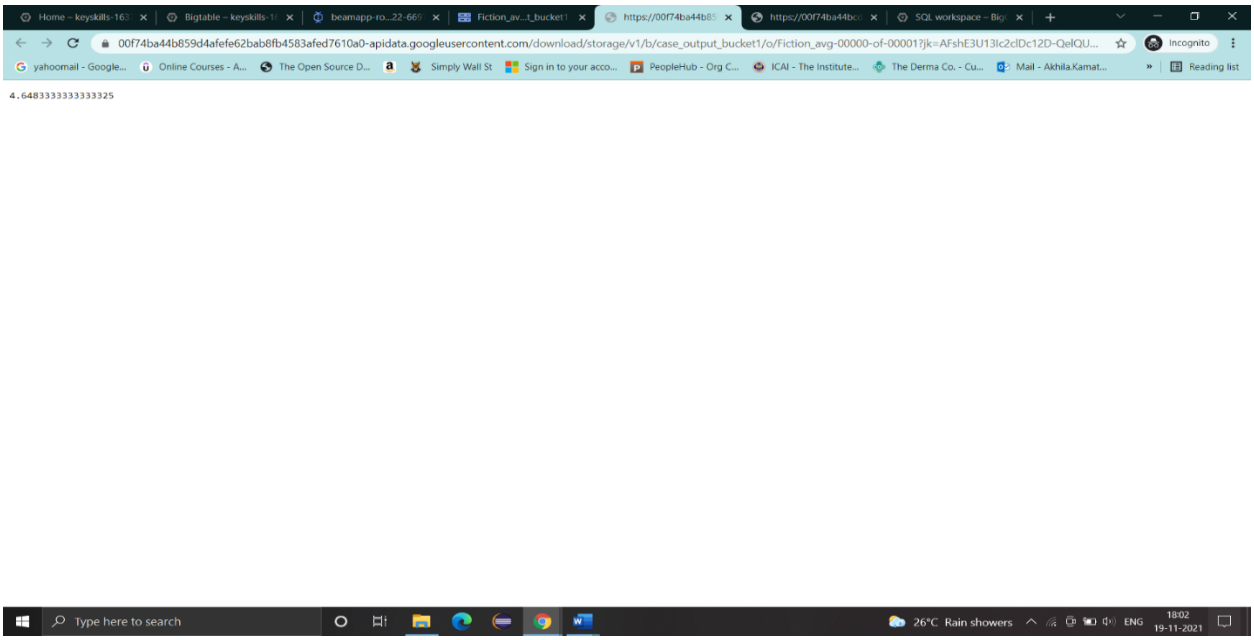
Output files obtained in the case_output_bucket1:



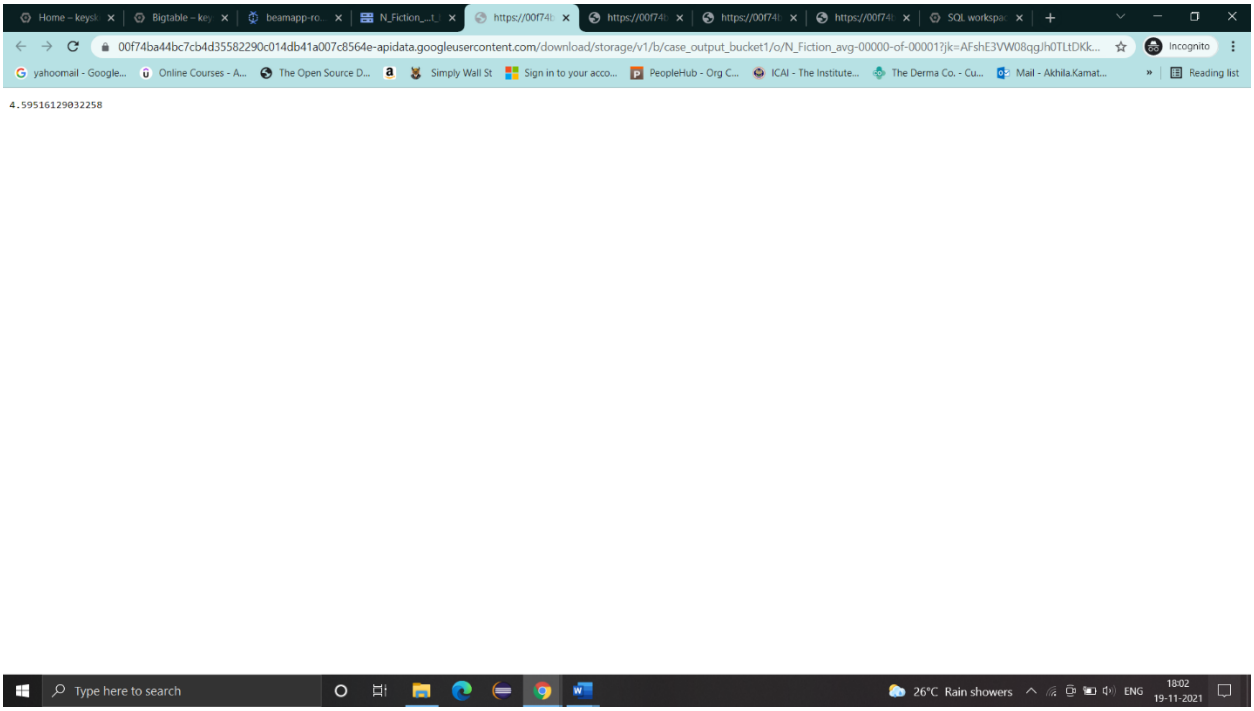
Average of all books :



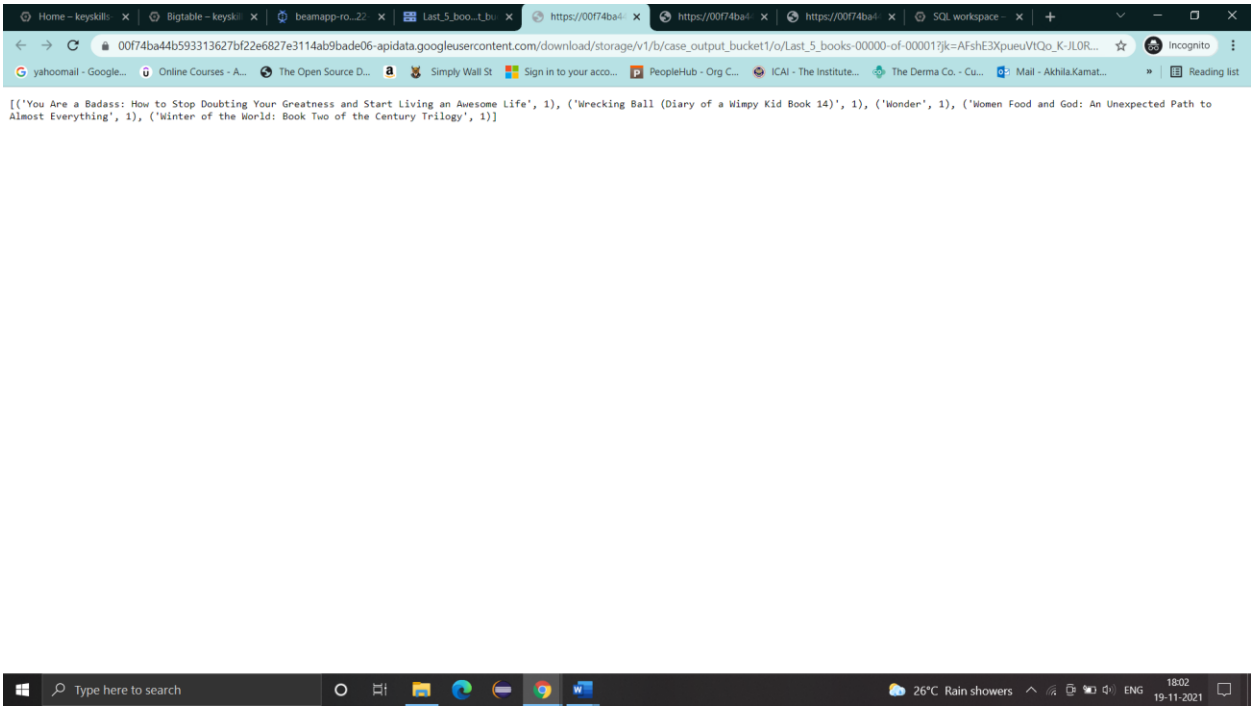
Average of Fiction Books:



Average of Non Fiction Books:



Last 5 Books:



Case study 2:

Big Data Analytics with Java using Cloud Dataproc, Google's Fully-Managed Spark and Hadoop Service, IBRD Statement Of Loans Data dataset, from Kaggle is used as input.

The International Bank for Reconstruction and Development (IBRD) loans are public and publicly guaranteed debt extended by the World Bank Group. IBRD loans are made to, or guaranteed by, countries that are members of IBRD. This dataset contains historical snapshots of the Statement of Loans including the latest available snapshots.

There are two data files available. The Statement of Loans latest available snapshots data file contains 8,713 rows of loan data (~3 MB), ideal for development and testing. The Statement of Loans historic data file contains approximately 750,000 rows of data (~265

MB). Although not exactly 'big data', the historic dataset is large enough to sufficiently explore Dataproc. Both IBRD files have an identical schema with 33 columns of data

PROBLEM STATEMENT:

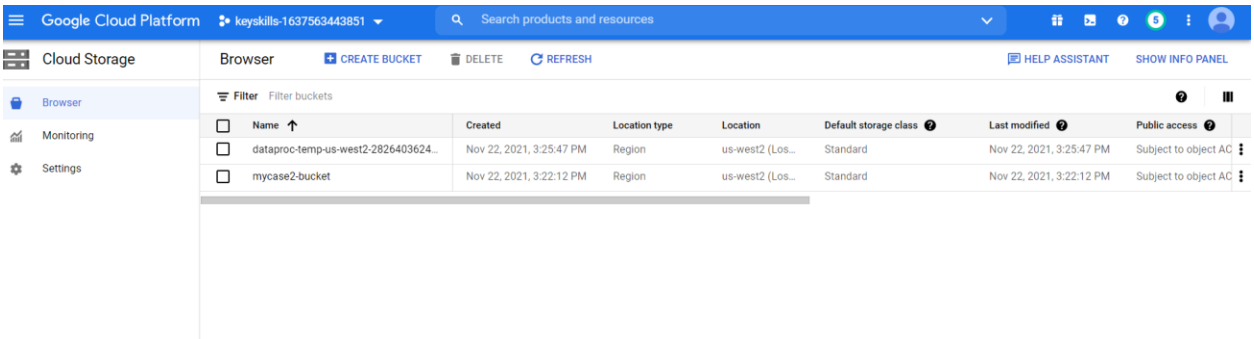
For the analysis, one has to

1. Ascertain the top 25 historic IBRD borrower;
2. Determine their total disbursements, current obligations, and the average interest rates they were charged.
3. Output/Persist the results in asked target endpoints*

Answer:

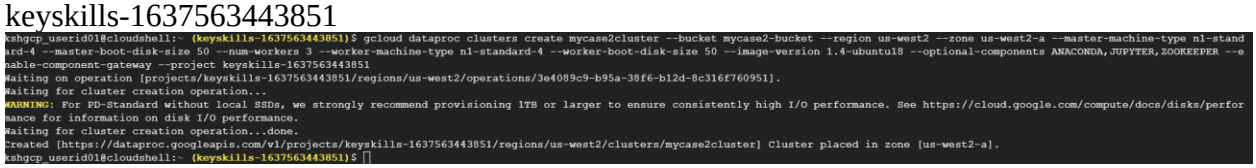
Bucket creation using CLI:

- gsutil mb -c standard -l us-west2 gs://mycase2-bucket

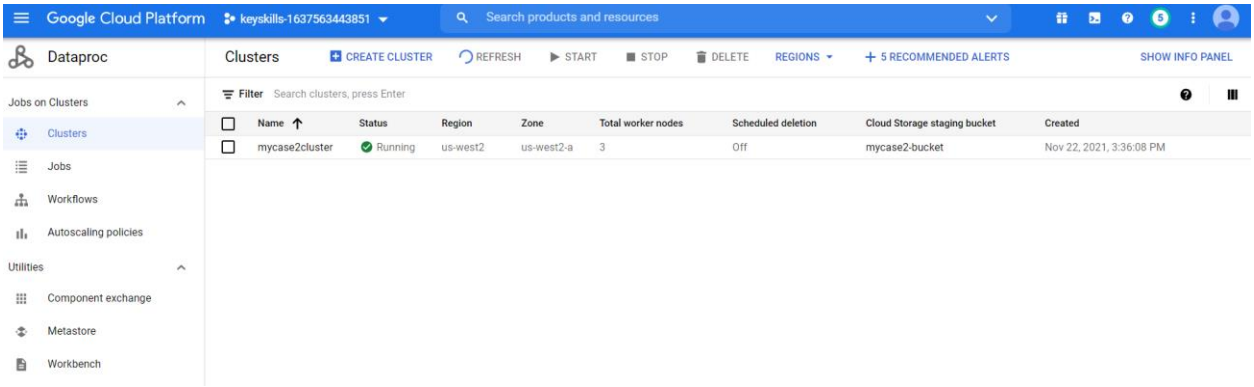


Cluster creation using CLI:

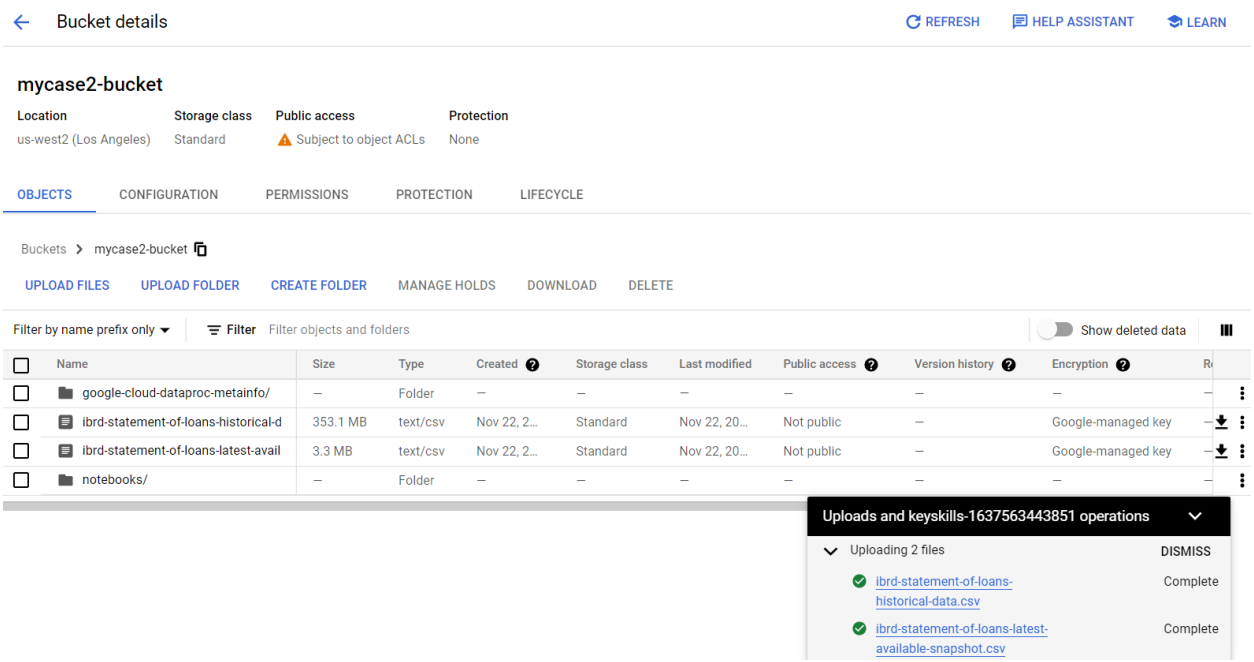
- gcloud dataproc clusters create mycase2cluster --bucket mycase2-bucket --region us-west2 --zone us-west2-a --master-machine-type n1-standard-4 --master-boot-disk-size 50 --num-workers 3 --worker-machine-type n1-standard-4 --worker-boot-disk-size 50 --image-version 1.4-ubuntu18 --optional-components ANACONDA,JUPYTER,ZOOKEEPER --enable-component-gateway --project keyskills-1637563443851



Dataproc cluster with 3 worker nodes:



Uploading the required csv files into the bucket:



Creating the JAR file:

- Java code:

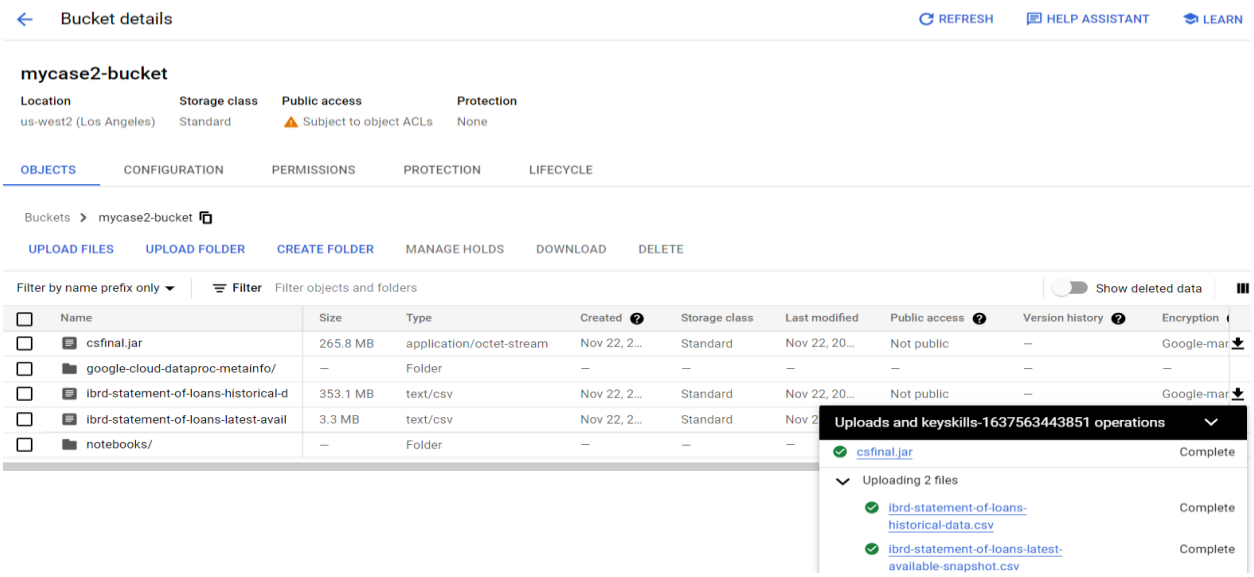
```
package dataproc;

import org.apache.spark.sql.Dataset;

import org.apache.spark.sql.Row;

import org.apache.spark.sql.SaveMode;

import org.apache.spark.sql.SparkSession;


public class InternationalLoansAppDataprocLarge {

   public static void main(String[] args) {            InternationalLoansAppDataprocLarge app = new
InternationalLoansAppDataprocLarge();

 app.start();

   }

   private void start() {

      SparkSession spark = SparkSession.builder()

            .appName("my-java-dataproc")

            .master("yarn")

            .getOrCreate();

      spark.sparkContext().setLogLevel("WARN");
```

```
// Loading the CSV file from GCS Bucket
    Dataset<Row> dfLoans = spark.read()
        .format("csv")
        .option("header", "true")
        .option("inferSchema", true)
        .load("gs://mycase2-bucket/ibrd-statement-of-loans-historical-data.csv");

    // Creates temporary view using DataFrame
    dfLoans.withColumnRenamed("Country", "country")
        .withColumnRenamed("Country Code", "country_code")
        .withColumnRenamed("Disbursed Amount", "disbursed")
        .withColumnRenamed("Borrower's Obligation", "obligation")
        .withColumnRenamed("Interest Rate", "interest_rate")
        .createOrReplaceTempView("loans");

 // Performs basic analysis of dataset
    Dataset<Row> dfDisbursement = spark.sql(
        "SELECT country, country_code, "
            + "format_number(total_disbursement, 0) AS total_disbursement, "
            + "format_number(ABS(total_obligation), 0) AS total_obligation, "
            + "format_number(avg_interest_rate, 2) AS avg_interest_rate "
            + "FROM ( "
            + "SELECT country, country_code, "
            + "SUM(disbursed) AS total_disbursement, "
            + "SUM(obligation) AS total_obligation, "
            + "AVG(interest_rate) AS avg_interest_rate "
            + "FROM loans "
            + "GROUP BY country, country_code "
            + "ORDER BY total_disbursement DESC "
            + "LIMIT 25)"
    );
    // Saves results to single CSV file in GCS Bucket
```

```
    dfDisbursement.repartition(1)
        .write()
        .mode(SaveMode.Overwrite)
        .format("csv")
        .option("header", "true")
        .save("gs://mycase2-bucket/ibrd-loan-summary-large");
    System.out.println("Results successfully written to CSV file");
  }
}
```

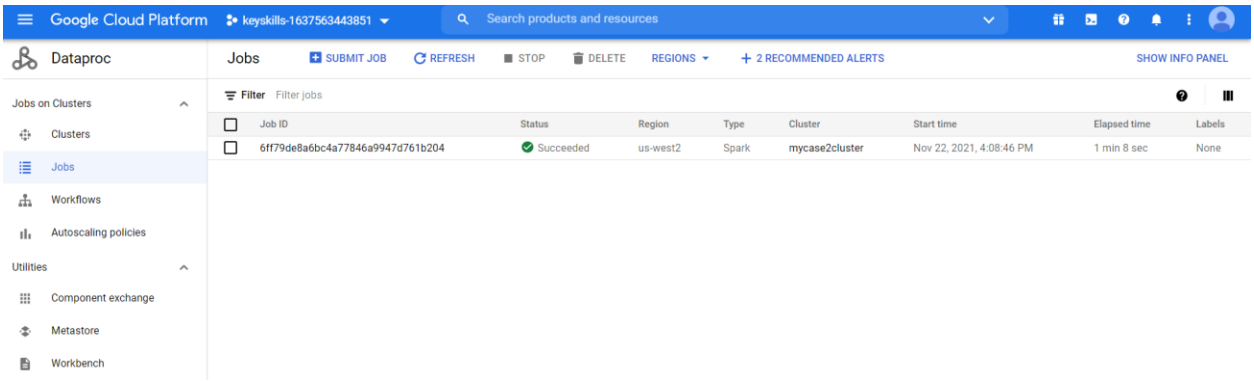
- The java code is saved as csfinal and exported as a jar file

Uploading the JAR file in the bucket:



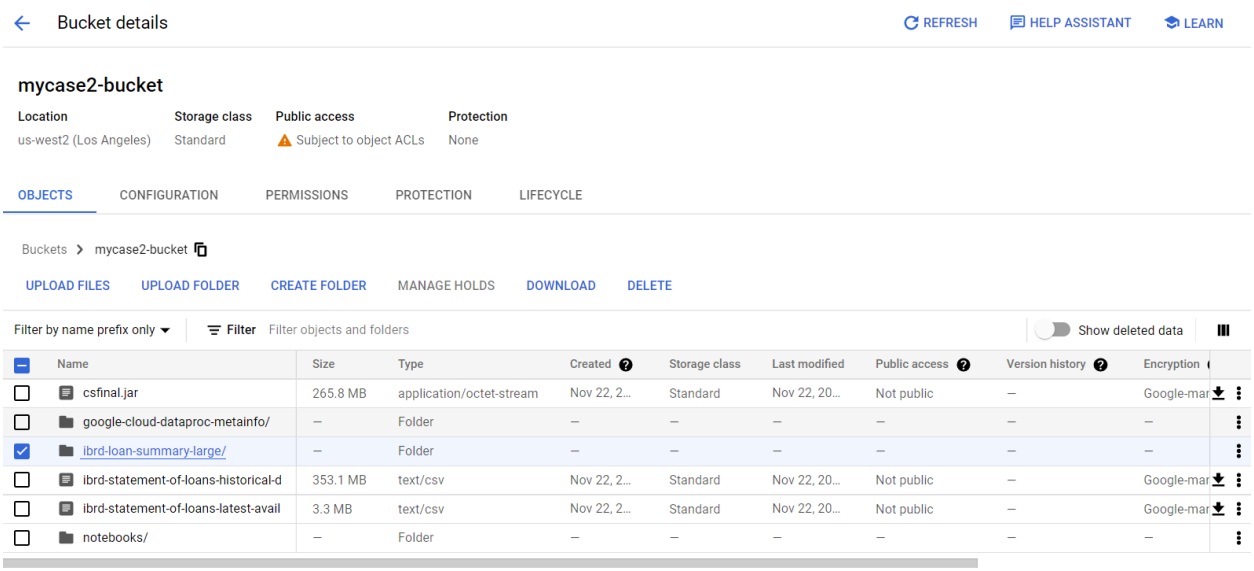
Submitting Dataproc job using CLI command:

- gcloud dataproc jobs submit spark --cluster=mycase2cluster --region=us-west2 --class=dataproc.InternationalLoansAppDataprocLarge --jars=gs://mycase2-bucket/csfinal.jar



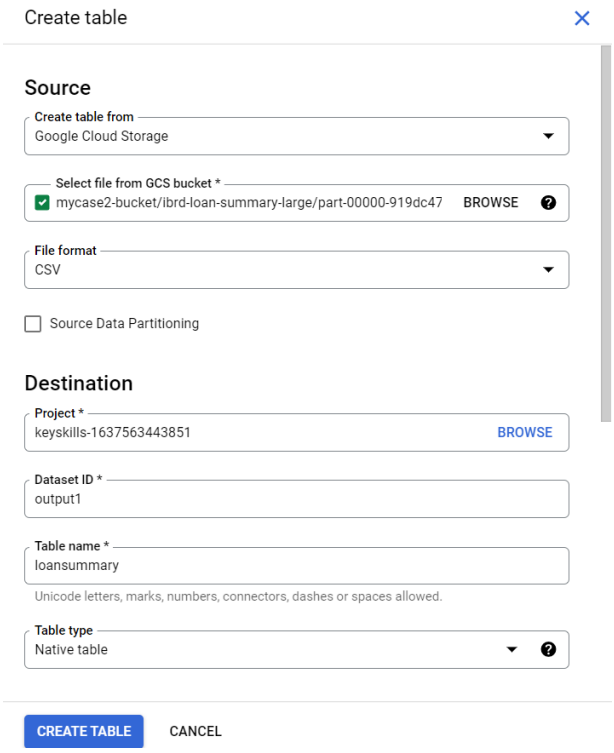
Dataproc job succeeded and the data is saved in the CSV file:

Output file added to the bucket:

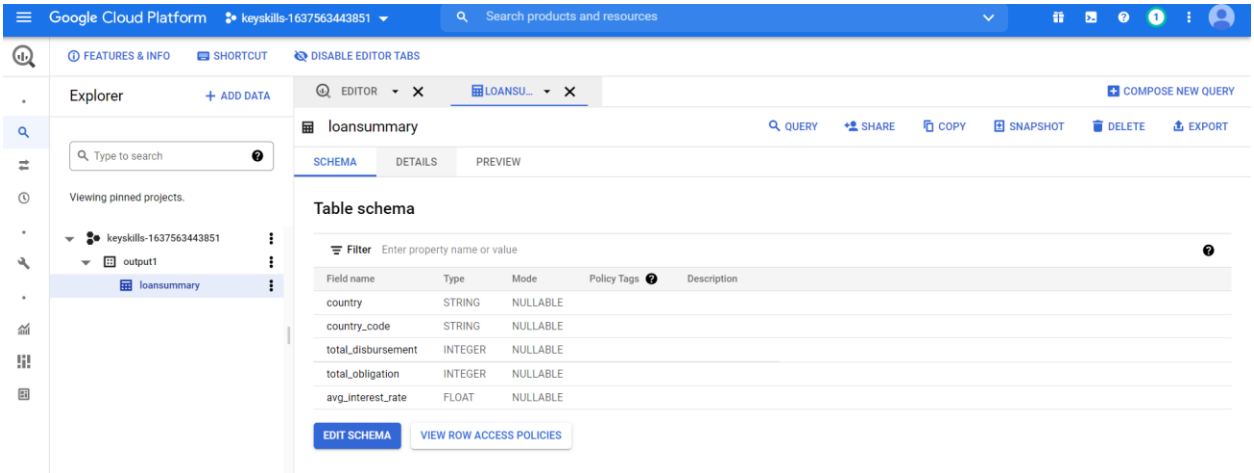


Creating a BigQuery table:

- Go to BigQuery and create a table under the project name's dataset (i.e output1) and select the output file from GCS and load it into table.
- Give the table name, in this case its loansummary.

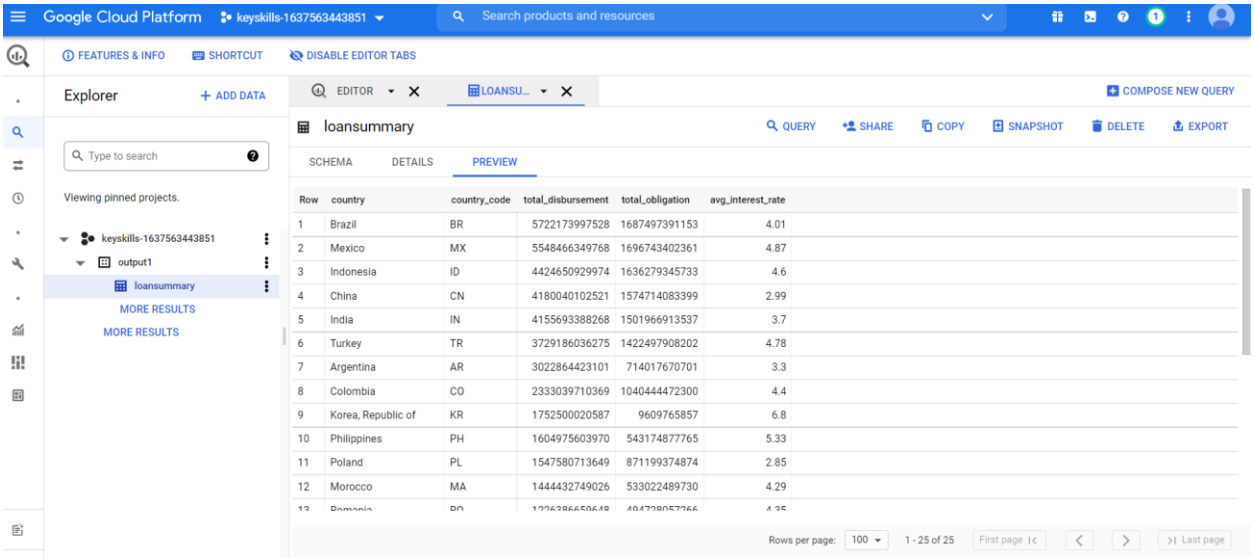


BigQuery table for output file is created:

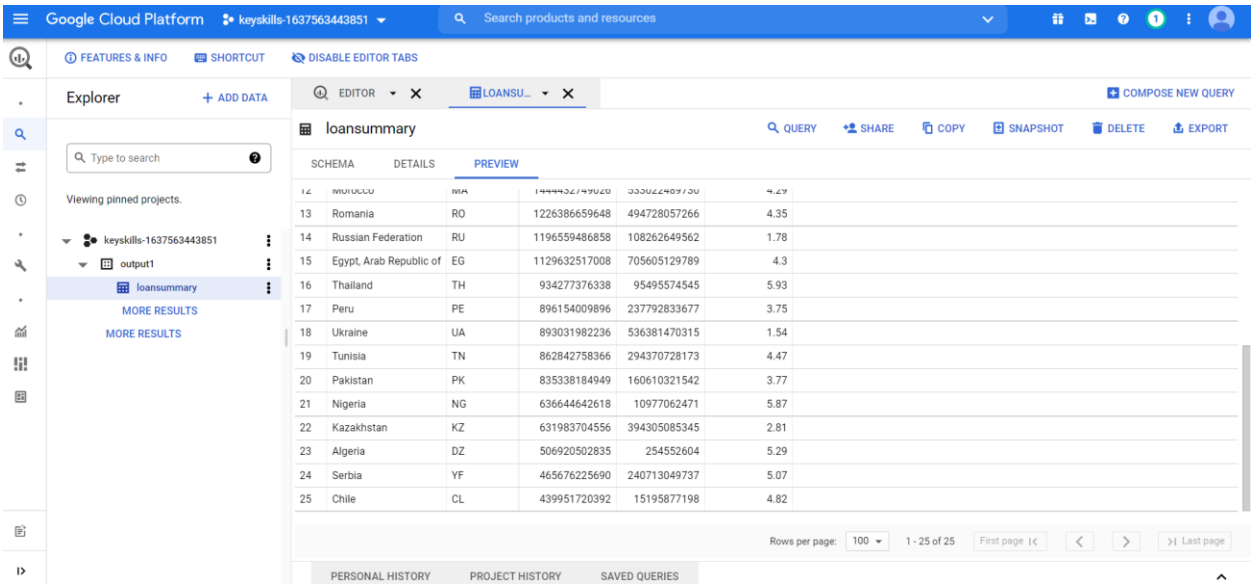


Click on preview to see the output:

- Output showing the top 25 historic IBRD borrowers.
- Also shows the total disbursements, current obligations, and the average interest rates they were charged.



Row	country	country_code	total_disbursement	total_obligation	avg_interest_rate
1	Brazil	BR	5722173997528	1687497391153	4.01
2	Mexico	MX	5548466349768	1696743402361	4.87
3	Indonesia	ID	4424650929974	1636279345733	4.6
4	China	CN	4180040102521	1574714083399	2.99
5	India	IN	4155693388268	1501966913537	3.7
6	Turkey	TR	3729186036275	1422497908202	4.78
7	Argentina	AR	3022864423101	714017670701	3.3
8	Colombia	CO	2333039710369	1040444472300	4.4
9	Korea, Republic of	KR	1752500020587	9609765857	6.8
10	Philippines	PH	1604975603970	543174877765	5.33
11	Poland	PL	1547580713649	871199374874	2.85
12	Morocco	MA	1444432749026	533022489730	4.29



12	Morocco	MA	1444432749026	533022489730	4.29
13	Romania	RO	1226386659648	494728057266	4.35
14	Russian Federation	RU	1196559486858	108262649562	1.78
15	Egypt, Arab Republic of	EG	1129632517008	705605129789	4.3
16	Thailand	TH	934277376338	95495574545	5.93
17	Peru	PE	896154009896	237792833677	3.75
18	Ukraine	UA	893031982236	536381470315	1.54
19	Tunisia	TN	862842758366	294370728173	4.47
20	Pakistan	PK	835338184949	160610321542	3.77
21	Nigeria	NG	636644642618	10977062471	5.87
22	Kazakhstan	KZ	631983704556	394305085345	2.81
23	Algeria	DZ	506920502835	254552604	5.29
24	Serbia	YF	465676225690	240713049737	5.07
25	Chile	CL	439951720392	15195877198	4.82

Output downloaded and viewed in XML format:

ibrd-loan-summary-large_part-00000-919dc47c-05dc-4f44-b334-af0a7b991fa6-c000 - Excel

country	country_c	total_disbursement	total_obligation	avg_interest_rate
Brazil	BR	57,22,17,39,97,528	16,87,49,73,91,153	4.01
Mexico	MX	55,48,46,63,49,768	16,96,74,34,02,361	4.87
Indonesia	ID	44,24,65,09,29,974	16,36,27,93,45,733	4.6
China	CN	41,80,04,01,02,521	15,74,71,40,83,399	2.99
India	IN	41,55,69,33,88,268	15,01,96,69,13,537	3.7
Turkey	TR	37,29,18,60,36,275	14,22,49,79,08,202	4.78
Argentina	AR	30,22,86,44,23,101	7,14,01,76,70,701	3.3
Colombia	CO	23,33,03,97,10,369	10,40,44,44,72,300	4.4
Korea, Rep	KR	17,52,50,00,20,587	9,60,97,65,857	6.8
Philippines	PH	16,04,97,56,03,970	5,43,17,48,77,765	5.33
Poland	PL	15,47,58,07,13,649	8,71,19,93,74,874	2.85
Morocco	MA	14,44,43,27,49,026	5,33,02,24,89,730	4.29
Romania	RO	12,26,38,66,59,648	4,94,72,80,57,266	4.35
Russian Fe	RU	11,96,55,94,86,858	1,08,26,26,49,562	1.78
Egypt, Aral	EG	11,29,63,25,17,008	7,05,60,51,29,789	4.3
Thailand	TH	9,34,27,73,76,338	95,49,55,74,545	5.93
Peru	PE	8,96,15,40,09,896	2,37,79,28,33,677	3.75
Ukraine	UA	8,93,03,19,82,236	5,36,38,14,70,315	1.54
Tunisia	TN	8,62,84,27,58,366	2,94,37,07,28,173	4.47
Pakistan	PK	8,35,33,81,84,949	1,60,61,03,21,542	3.77
Nigeria	NG	6,36,64,46,42,618	10,97,70,62,471	5.87
Kazakhstai	KZ	6,31,98,37,04,556	3,94,30,50,85,345	2.81
Algeria	DZ	5,06,92,05,02,835	25,45,52,604	5.29
Serbia	YF	4,65,67,62,25,690	2,40,71,30,49,737	5.07
Chile	CL	4,39,95,17,20,392	15,19,58,77,198	4.82