

Name : Manikandan Mohan

Case Study 1

Amazon Books Catalog Search, Data Cleansing, Analytics, Pipeline & Reporting using Apache Beam, Python pySpark on a Google Data Proc Cluster [ubuntu 20.04 VMs].

PROBLEM STATEMENT:

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

Implementation Steps -

- Pre-requisite – One should download this csv and do data cleaning (replace the commas in the Name column with semicolons). [Upload this csv in the GCP bucket] – Pyspark, Hive [10% credits]

One has to build a DataProc Cluster [at least 2 nodes] to run Python pySpark batch jobs [Kaggle dataset, from GCP bucket] on Google Cloud. – DataProc [15% credits]

- Advisable to connect to the nodes using CLI and provision the VMs with required configurations using [15% credits] o GCP Cloud SDK o gsutil tool
- Create/Configure a pipeline [10% credits] using Command Line

Pipeline = Reading input + Transforming Data + Writing Output

Output results should be stored as raw data in Hive[BigQuery] before stored finally into HBase[BigTable] for further processing & reporting to Analytics Engines. – Hbase, Hive [15% Credits]

- Run the pipeline – Apache Beam [10% credit]

- Print the results to console and store for further analytics – Hbase [10% credits]

Extract of the pipeline execution and output is here – (good to use HBase) – [10% credits]

Engineering Best Practices to be followed [30% credits]

1. Logging has to be implemented in all the Python Components
2. Exception handling framework need to be implemented
3. Unit testing of each Python component/service using Pytest/UnitTest/Mock Objects
4. Unit Testing of Spark Cluster, Data Pipeline [is desirable]
5. Usage of 4 to 5 Big Data Design Patterns using Python [is desirable]

Answer

Project Name : keyskills-1637220444729

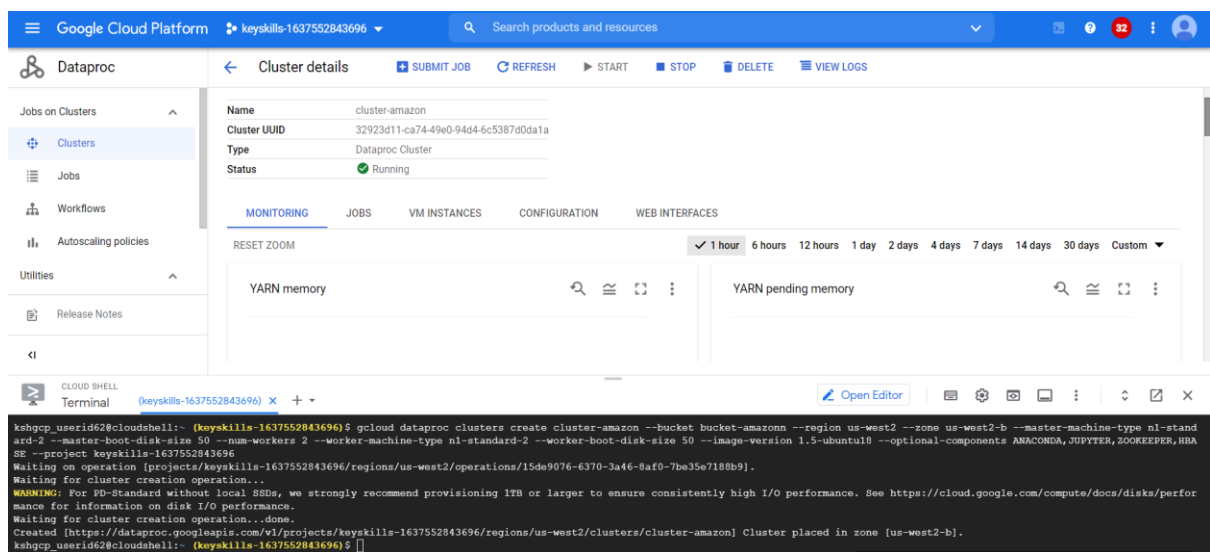
Creating a bucket storage named “bucket-amazonn” using CLI and uploading the kaggle dataset.

`gsutil mb -p keyskills-1637552843696 -l us-west2 -b on gs://bucket-amazonn`

The screenshot displays the Google Cloud Platform console interface. The top navigation bar shows the project name 'keyskills-1637552843696'. The left sidebar contains navigation options: Cloud Storage, Browser, Monitoring, Settings, Marketplace, and Release Notes. The main content area is titled 'Bucket details' for 'bucket-amazonn'. It shows the bucket's location as 'us-west2 (Los Angeles)', storage class as 'Standard', public access as 'Not public', and protection as 'None'. Below this, there are tabs for OBJECTS, CONFIGURATION, PERMISSIONS, PROTECTION, and LIFECYCLE. The 'OBJECTS' tab is active, showing a list of objects. One object is visible: 'bestsellers with categories.csv', which is 50 KB in size, of type 'application/vnd.ms-excel', and was created on Nov 22, 2022. At the bottom, a terminal window is open, showing the command `gsutil mb -p keyskills-1637552843696 -l us-west2 -b on gs://bucket-amazonn` being executed successfully.

Creating a cluster using Command Line Interface (CLI) :

gcloud dataproc clusters create cluster-amazon --bucket bucket-amazonn --region us-west2 --zone us-west2-b --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 1.5-ubuntu18 --optional-components ANACONDA,JUPYTER,ZOOKEEPER,HBASE --project keyskills-1637552843696



Creating a python (spark) file with the required queries and uploading it into the bucket.

Pyspark code :

amazon_book_analysis.py

import pyspark as sc

from pyspark import SparkConf, SparkContext

from pyspark.sql import SparkSession

```
from pyspark.sql.functions import translate
```

```
#Reading Data using SparkSession
```

```
spark = SparkSession.builder.master("local[*]").getOrCreate()
```

```
data =
```

```
spark.read.option("header",True).options(inferSchema=True,delimiter=',').csv("gs://bucket-amazonn/bestsellers with categories.csv" )
```

```
data.show(truncate=False)
```

```
data.printSchema()
```

```
#Replacing ',' with ';' in the name column
```

```
data = data.withColumn('Name', translate('Name', ',', ';'))
```

```
data.show(truncate=False)
```

```
#Find averages of fiction , non- fiction and overall
```

```
data.filter(data['Genre']=='Fiction').agg({'User Rating' : 'avg'}).show()
```

```
data.filter(data['Genre']=='Non Fiction').agg({'User Rating' : 'avg'}).show()
```

```
data.agg({'User Rating': 'avg'}).show()
```

```
avg_fict = data.filter(data['Genre']=='Fiction').agg({'User Rating' : 'avg'})
```

```
avg_non_fict = data.filter(data['Genre']=='Non Fiction').agg({'User Rating' : 'avg'})
```

```
avg_all = data.agg({'User Rating' : 'avg'})
```

```
avg_fict.write.csv('gs://bucket-amazonn/Avg_of_Fiction')
```

```
avg_non_fict.write.csv('gs://bucket-amazonn/Avg_of_Non_Fiction')
```

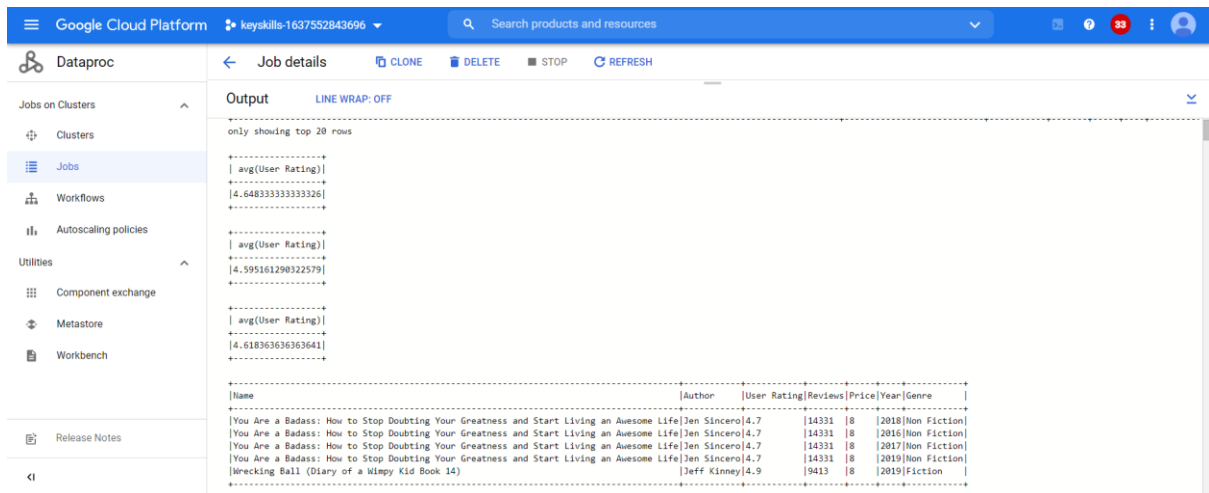
```
avg_all.write.csv('gs://bucket-amazonn/Avg_of_Allbooks')
```

```
data.orderBy(data.Name.desc()).limit(5).show(truncate=False)
```

```
gcloud dataproc jobs submit pyspark gs://bucket-amazonn
/amazon_book_analysis.py --cluster=cluster-amazon --region=us-west2
```

Job output : Displays the top 20 rows.

Displays the average rating of books and the last 5 rows of the dataset.



Google Cloud Platform | keyskills-1637552843696 | Search products and resources

Dataproc | Job details | CLONE | DELETE | STOP | REFRESH

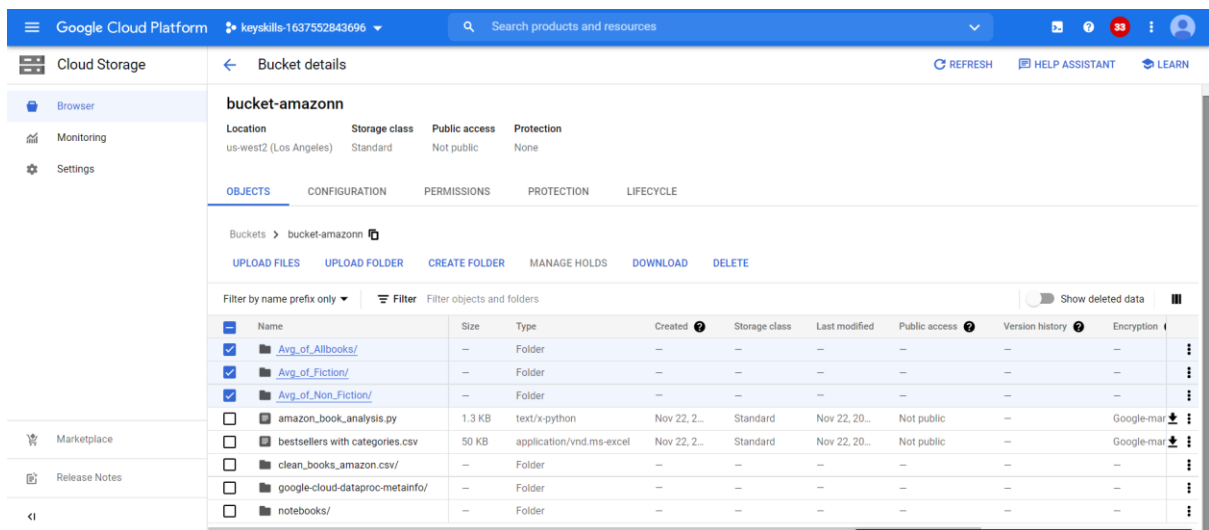
Output | LINE WRAP: OFF

only showing top 20 rows

```
| avg(User_Rating) |
| 4.648333333333326 |
| avg(User_Rating) |
| 4.595161290322579 |
| avg(User_Rating) |
| 4.618363636363641 |
```

Id	Author	User_Rating	Reviews	Price	Year	Genre
1	You Are a Badass: How to Stop Doubting Your Greatness and Start Living an Awesome Life	4.7	14331	8	2018	Non Fiction
2	You Are a Badass: How to Stop Doubting Your Greatness and Start Living an Awesome Life	4.7	14331	8	2016	Non Fiction
3	You Are a Badass: How to Stop Doubting Your Greatness and Start Living an Awesome Life	4.7	14331	8	2017	Non Fiction
4	You Are a Badass: How to Stop Doubting Your Greatness and Start Living an Awesome Life	4.7	14331	8	2019	Non Fiction
5	Wrecking Ball (Diary of a Wimpy Kid Book 14)	4.9	9413	8	2019	Fiction

Output CSV files written to the bucket.



Google Cloud Platform | keyskills-1637552843696 | Search products and resources

Cloud Storage | Bucket details | REFRESH | HELP ASSISTANT | LEARN

bucket-amazonn

Location: us-west2 (Los Angeles) | Storage class: Standard | Public access: Not public | Protection: None

OBJECTS | CONFIGURATION | PERMISSIONS | PROTECTION | LIFECYCLE

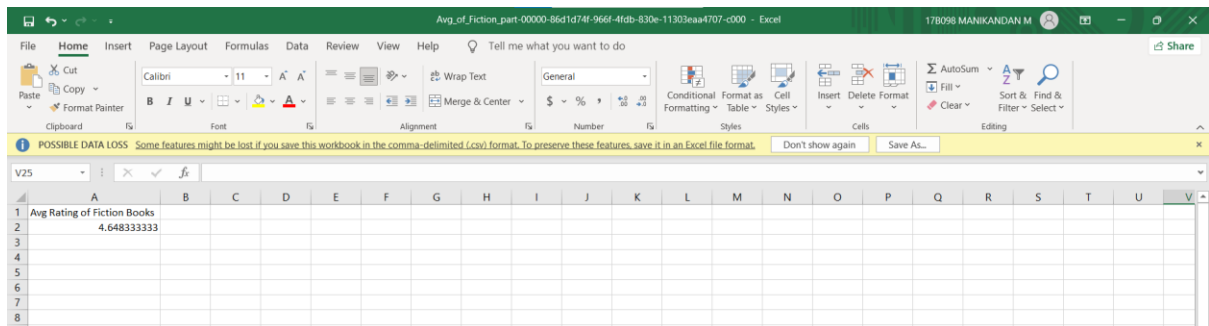
Buckets > bucket-amazonn

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
Avg_of_Allbooks/	-	Folder	-	-	-	-	-	-
Avg_of_Fiction/	-	Folder	-	-	-	-	-	-
Avg_of_Non-Fiction/	-	Folder	-	-	-	-	-	-
amazon_book_analysis.py	1.3 KB	text/x-python	Nov 22, 20...	Standard	Nov 22, 20...	Not public	-	Google-mar
bestsellers with categories.csv	50 KB	application/vnd.ms-excel	Nov 22, 20...	Standard	Nov 22, 20...	Not public	-	Google-mar
clean_books_amazon.csv/	-	Folder	-	-	-	-	-	-
google-cloud-dataproc-metainfo/	-	Folder	-	-	-	-	-	-
notebooks/	-	Folder	-	-	-	-	-	-

Average Rating of Fiction Books



Excel | 17B098 MANIKANDAN M

File | Home | Insert | Page Layout | Formulas | Data | Review | View | Help | Tell me what you want to do

Clipboard | Font | Alignment | Number | Styles | Cells | Editing

POSSIBLE DATA LOSS | Some features might be lost if you save this workbook in the comma-delimited (.csv) format. To preserve these features, save it in an Excel file format. | Don't show again | Save As...

V25 | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V

1	Avg Rating of Fiction Books																				
2		4.648333333																			
3																					
4																					
5																					
6																					
7																					
8																					

Average Rating of Non Fiction Books

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	Avg Rating of Non Fiction Books																				
2		4.59516129																			
3																					
4																					
5																					
6																					
7																					
8																					
9																					

Average Rating of all the Books.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
1	Avg Rating of All Books																					
2		4.618363636																				
3																						
4																						
5																						
6																						
7																						
8																						
9																						

Last 5 rows while the dataset in alphabetical order

[[('You Are a Badass: How to Stop Doubting Your Greatness and Start Living an Awesome Life', 1), ('Wrecking Ball (Diary of a Wimpy Kid Book 14)', 1), ('Wonder', 1), ('Women Food and God: An Unexpected Path to Almost Everything', 1), ('Winter of the World: Book Two of the Century Trilogy', 1)]]

Creating a dataflow pipeline job using apache beam (python) for executing all the queries and to write the table in the bigquery.

Creating a jupyter lab notebook with apache beam on dataflow workbench.

Pipeline code :

Dataflowpipeline.ipynb [Using Jupyter Lab on Google Cloud]

```
import apache_beam as beam

from apache_beam.runners.interactive.interactive_runner import InteractiveRunner
from apache_beam.options.pipeline_options import PipelineOptions
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
from apache_beam.runners.runner import PipelineState
from IPython.core.display import display, HTML

def Split(element):

    return element.split(",")

# Finding the average of given list of values
class avgRatingFn(beam.CombineFn):
```



```
def create_accumulator(self):  
    return (0.0, 0) # initialize (sum, count)
```

```
def add_input(self, sum_count, input1):  
    (sum1, count) = sum_count  
    (sum2, c2) = input1  
    return sum1 + float(sum2) * c2, count + c2
```

```
def merge_accumulators(self, accumulators):  
    ind_sums, ind_counts = zip(*accumulators)  
    return sum(ind_sums), sum(ind_counts)
```

```
def extract_output(self, sum_count):  
    (sum1, count) = sum_count  
    return sum1 / count if count else float('NaN')
```

```
pipe = beam.Pipeline(InteractiveRunner())
```

```
# Setting up the Apache Beam pipeline options.
```

```
options = pipeline_options.PipelineOptions(flags=[])
```

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

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

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

```
dataflow_gcs_location = 'gs://bucket-amazonn/dataflowAmz'
```

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

The Dataflow Temp Location location is used to store intermediate results before outputting the final result.

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

```
clientBQ = bigquery.Client()
```

```
dataset_id = "keyskills-1637552843696.dataflowpipelineAmz"
```

```
dataset = bigquery.Dataset(dataset_id)
```

```
dataset.location = "us-west2"
```

```
dataset.description = "Amazon dataset books"
```

```
clientBQ.create_dataset(dataset, timeout = 30)
```

#Converts csv to json

```
def to_json(csv_file):
```

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

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

```
                  "Author": fields[1],
```

```
                  "User_Rating": fields[2],
```

```
                  "Reviews": fields[3],
```

```
                  "Price": fields[4],
```

```
                  "Year": fields[5],
```

```
                  "Genre": fields[6]
```

```
    }
```

```
    return json_file
```

```
table_schema =
```

```
'Name:STRING,Author:STRING,User_Rating:FLOAT,Reviews:INTEGER,Price:  
INTEGER,Year: INTEGER ,Genre:STRING'
```

```

bookAmz = (pipe | beam.io.ReadFromText("gs://bucket-
amazonn/clean_books_amazon.csv/part-00000-44653796-5293-4a2a-b34e-
225c74788cd8-c000.csv"))

(bookAmz | 'cleaned_data to json' >> beam.Map(to_json)
| 'write to bigquery' >> beam.io.WriteToBigQuery(
    "keyskills-1637552843696:dataflowpipelineAmz.tableAmz",
    schema=table_schema,
    create_disposition=beam.io.BigQueryDisposition.CREATE_IF_NEEDED,
    write_disposition=beam.io.BigQueryDisposition.WRITE_APPEND,
    custom_gcs_temp_location="gs://bucket-amazonn/dataflowAmz/temp"
)
)

```

```

pipelineAmz = pipe.run()

if pipelineAmz.state == PipelineState.DONE:
    print("The pipeline is running successfully \n The table is sent to big query
successfully !!!")
else:
    print('Error running the pipeline')

```

```

# Reads data and split based on ','

pipe2 = beam.Pipeline(InteractiveRunner())

AmzBooks = (pipe2 | beam.io.ReadFromText("gs://bucket-
amazonn/clean_books_amazon.csv/part-00000-44653796-5293-4a2a-b34e-
225c74788cd8-c000.csv") | beam.Map(Split))

# Filter records having fiction. Map each rating as a set (rating,1) .Using
combineperkey to count the number of each rating. Running the average function
and write the result to Fiction_result1

```

```

avg_fict = (
    AmzBooks
    | beam.Filter(lambda rec: rec[6] == "Fiction")
    | beam.Map(lambda rec: (rec[2], 1))
    | "Fict Combine keys1" >> beam.CombinePerKey(sum)
    | "Fict Combine Global keys1" >> beam.CombineGlobally(avgRatingFn())
    | "Fict Write to bucket1" >> beam.io.WriteToText("gs://bucket-
amazonn/dataflowAmz/FictionAvg_Result")
)

```

#Same for "non fiction"

```

avg_non_fict = (
    AmzBooks
    | beam.Filter(lambda rec: rec[6] == "Non Fiction")
    | beam.Map(lambda rec: (rec[2], 1))
    | "N_Fict Combine keys" >> beam.CombinePerKey(sum)
    | "N_Fict Combine Global keys" >> beam.CombineGlobally(avgRatingFn())
    | "N_Fict Write to bucket" >> beam.io.WriteToText("gs://bucket-
amazonn/dataflowAmz/NonFictionAvg_Result")
)

```

#Same for "all genre"

```

avg_all = (
    AmzBooks
    | beam.Map(lambda rec: (rec[2], 1))
    | "All Combine keys" >> beam.CombinePerKey(sum)
    | "All Combine Global keys" >> beam.CombineGlobally(avgRatingFn())
    | "All Write to bucket" >> beam.io.WriteToText("gs://bucket-
amazonn/dataflowAmz/AllBookAvg_Result")
)

```

)

Map each record's 0th column that is name with value 1.

Running top.of(5) to sort and get the last5 books alphabetically and store it.

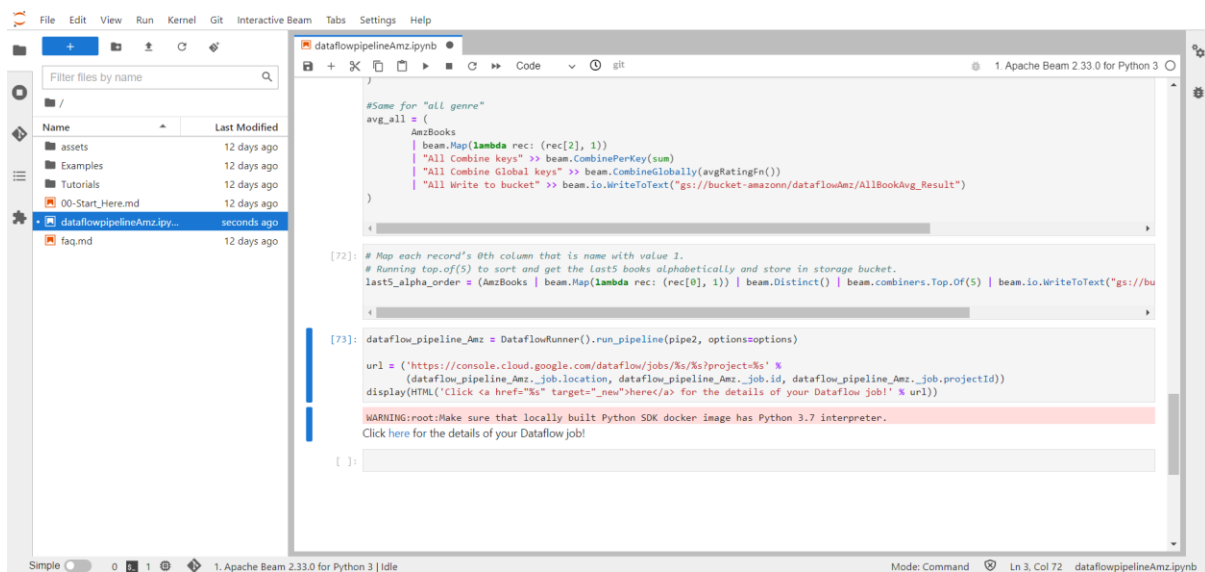
```
last5_alpha_order = (AmzBooks | beam.Map(lambda rec: (rec[0], 1)) |  
beam.Distinct() | beam.combiners.Top.Of(5) | beam.io.WriteToText("gs://bucket-  
amazonn/dataflowAmz/Last5_Result"))
```

```
dataflow_pipeline_Amz = DataflowRunner().run_pipeline(pipe2, options=options)
```

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

```
(dataflow_pipeline_Amz._job.location, dataflow_pipeline_Amz._job.id,  
dataflow_pipeline_Amz._job.projectId))
```

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



Dataflow Pipeline Job

Google Cloud Platform | keyskills-1637552843696 | Search products and resources

Dataflow | beamapp-root-1122070755-087384 | + IMPORT AS PIPELINE | SHARE

Jobs | Snapshots | Workbench | Pipelines | SQL Workspace

JOB GRAPH | EXECUTION DETAILS | JOB METRICS | RECOMMENDATIONS

Job steps view | Graph view | CLEAR SELECTION

```

graph TD
    A["[71]: ReadFromText  
Succeeded  
1 sec  
2 of 2 stages succeeded"] --> B["[71]: Map(Split)  
Succeeded  
0 sec  
1 of 1 stage succeeded"]
    B --> C["[71]: Filt...932f1>8>  
Succeeded  
0 sec  
1 of 1 stage succeeded"]
    B --> D["[71]: Filt...32f1>18>  
Succeeded  
0 sec  
1 of 1 stage succeeded"]
    B --> E["[71]: Map<...32f1>28>  
Succeeded  
0 sec  
1 of 1 stage succeeded"]
    B --> F["[72]: Map<...b3b8e>3>  
Succeeded  
0 sec  
1 of 1 stage succeeded"]
  
```

Logs | SHOW

Job info

Job name	beamapp-root-1122070755-087384
Job ID	2021-11-21_23_07_58-17752934408351991298
Job type	Batch
Job status	Succeeded
SDK version	Apache Beam Python 3.7 SDK 2.33.0
Job region	us-west2
Worker location	us-west2-b
Current workers	0
Latest worker status	Worker pool stopped
Start time	November 22, 2021 at 12:37:58 PM GMT+5
Elapsed time	6 min 34 sec
Encryption type	Google-managed key

Google Cloud Platform | keyskills-1637552843696 | Search products and resources

Dataflow | beamapp-root-1122070755-087384 | + IMPORT AS PIPELINE | SHARE

Jobs | Snapshots | Workbench | Pipelines | SQL Workspace

JOB GRAPH | EXECUTION DETAILS | JOB METRICS | RECOMMENDATIONS

Job steps view | Graph view | CLEAR SELECTION

```

graph TD
    A["[71]: Filt...932f1>8>  
Succeeded  
0 sec  
1 of 1 stage succeeded"] --> B["[71]: Map<...932f1>9>  
Succeeded  
0 sec  
1 of 1 stage succeeded"]
    A --> C["[71]: Filt...32f1>18>  
Succeeded  
0 sec  
1 of 1 stage succeeded"]
    A --> D["[71]: Map<...32f1>28>  
Succeeded  
0 sec  
1 of 1 stage succeeded"]
    A --> E["[72]: Map<...b3b8e>3>  
Succeeded  
0 sec  
1 of 1 stage succeeded"]
    B --> F["[71]: Fict...bine keys1  
Succeeded  
0 sec  
2 of 2 stages succeeded"]
    C --> G["[71]: N.Fic...mbine keys  
Succeeded  
0 sec  
2 of 2 stages succeeded"]
    D --> H["[71]: All C...lobal keys  
Succeeded  
0 sec  
3 of 3 stages succeeded"]
    E --> I["[72]: Top(5)  
Succeeded  
0 sec  
3 of 3 stages succeeded"]
    F --> J["[71]: Fict...lobal keys1  
Succeeded  
0 sec  
3 of 3 stages succeeded"]
    G --> K["[71]: N.Fic...lobal keys  
Succeeded  
0 sec  
3 of 3 stages succeeded"]
    H --> L["[71]: All W...to bucket  
Succeeded  
1 sec  
5 of 5 stages succeeded"]
    I --> M["[72]: WriteToText  
Succeeded  
1 sec  
5 of 5 stages succeeded"]
    J --> N["[71]: Fict...o bucket1  
Succeeded  
1 sec  
5 of 5 stages succeeded"]
    K --> O["[71]: N.Fic...to bucket  
Succeeded  
1 sec  
5 of 5 stages succeeded"]
  
```

Logs | SHOW

Job info

Job name	beamapp-root-1122070755-087384
Job ID	2021-11-21_23_07_58-17752934408351991298
Job type	Batch
Job status	Succeeded
SDK version	Apache Beam Python 3.7 SDK 2.33.0
Job region	us-west2
Worker location	us-west2-b
Current workers	0
Latest worker status	Worker pool stopped
Start time	November 22, 2021 at 12:37:58 PM GMT+5
Elapsed time	6 min 34 sec
Encryption type	Google-managed key

Google Cloud Platform | keyskills-1637552843696 | Search products and resources

Dataflow | beamapp-root-1122070755-087384 | + IMPORT AS PIPELINE | SHARE

Jobs | Snapshots | Workbench | Pipelines | SQL Workspace

JOB GRAPH | EXECUTION DETAILS | JOB METRICS | RECOMMENDATIONS

Job steps view | Graph view | CLEAR SELECTION

```

graph TD
    A["[71]: Fict...bine keys1  
Succeeded  
0 sec  
2 of 2 stages succeeded"] --> B["[71]: Fict...lobal keys1  
Succeeded  
0 sec  
3 of 3 stages succeeded"]
    A --> C["[71]: N.Fic...mbine keys  
Succeeded  
0 sec  
2 of 2 stages succeeded"]
    A --> D["[71]: All C...lobal keys  
Succeeded  
0 sec  
3 of 3 stages succeeded"]
    A --> E["[72]: Top(5)  
Succeeded  
0 sec  
3 of 3 stages succeeded"]
    B --> F["[71]: Fict...o bucket1  
Succeeded  
1 sec  
5 of 5 stages succeeded"]
    C --> G["[71]: N.Fic...lobal keys  
Succeeded  
0 sec  
3 of 3 stages succeeded"]
    C --> H["[71]: N.Fic...to bucket  
Succeeded  
1 sec  
5 of 5 stages succeeded"]
    D --> I["[71]: All W...to bucket  
Succeeded  
1 sec  
5 of 5 stages succeeded"]
    E --> J["[72]: WriteToText  
Succeeded  
1 sec  
5 of 5 stages succeeded"]
    F --> K["[71]: Fict...o bucket1  
Succeeded  
1 sec  
5 of 5 stages succeeded"]
    G --> H
    H --> I
    I --> J
  
```

Logs | SHOW

Job info

Job name	beamapp-root-1122070755-087384
Job ID	2021-11-21_23_07_58-17752934408351991298
Job type	Batch
Job status	Succeeded
SDK version	Apache Beam Python 3.7 SDK 2.33.0
Job region	us-west2
Worker location	us-west2-b
Current workers	0
Latest worker status	Worker pool stopped
Start time	November 22, 2021 at 12:37:58 PM GMT+5
Elapsed time	6 min 34 sec
Encryption type	Google-managed key

All the necessary dataflow files written to the bucket.

Google Cloud Platform | keyskills-1637552843696 | Search products and resources

Cloud Storage | Bucket details | bucket-amazonn

Location: us-west2 (Los Angeles) | Storage class: Standard | Public access: Not public | Protection: None

OBJECTS | CONFIGURATION | PERMISSIONS | PROTECTION | LIFECYCLE

Buckets > bucket-amazonn > dataflowAmz

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
AllBookAvg_Result-00000-of-00001	18 B	text/plain	Nov 22, 20...	Standard	Nov 22, 20...	Not public	—	Google-managed key	—
FictionAvg_Result-00000-of-00001	19 B	text/plain	Nov 22, 20...	Standard	Nov 22, 20...	Not public	—	Google-managed key	—
Last5_Result-00000-of-00001	293 B	text/plain	Nov 22, 20...	Standard	Nov 22, 20...	Not public	—	Google-managed key	—
NonFictionAvg_Result-00000-of-00001	17 B	text/plain	Nov 22, 20...	Standard	Nov 22, 20...	Not public	—	Google-managed key	—
staging/	—	Folder	—	—	—	—	—	—	—
temp/	—	Folder	—	—	—	—	—	—	—

The dataflow pipeline read the csv file from the bucket and wrote it as a table to the dataset in BigQuery.

Google Cloud Platform | keyskills-1637552843696 | Search products and resources

FEATURES & INFO | SHORTCUT | DISABLE EDITOR TABS

Explorer | + ADD DATA | EDITOR | TABLE | COMPOSE NEW QUERY

tableAmz | QUERY | SHARE | COPY | SNAPSHOT | DELETE | EXPORT

SCHEMA | DETAILS | PREVIEW

Row	Name	Author	User_Rating	Reviews	Price	Year	Genre
1	Gone Girl	Gillian Flynn	4.0	57271	10	2012	Fiction
2	Gone Girl	Gillian Flynn	4.0	57271	10	2013	Fiction
3	Gone Girl	Gillian Flynn	4.0	57271	9	2014	Fiction
4	Harry Potter and the Cursed Child; Parts 1 & 2; Special Rehearsal Edition Script	J.K. Rowling	4.0	23973	12	2016	Fiction
5	The Elegance of the Hedgehog	Muriel Barbery	4.0	1859	11	2009	Fiction
6	A Wrinkle in Time (Time Quintet)	Madeleine L'Engle	4.5	5153	5	2018	Fiction
7	Divergent / Insurgent	Veronica Roth	4.5	17684	6	2014	Fiction
8	Fifty Shades Freed: Book Three of the Fifty Shades of Grey Series (English Edition)	E L James	4.5	20262	11	2012	Fiction
9	Fifty Shades Trilogy (Fifty Shades of Grey / Fifty Shades Darker / Fifty Shades Freed)	E L James	4.5	13964	32	2012	Fiction
10	Joyland (Hard Case Crime)	Stephen King	4.5	4748	12	2013	Fiction
11	Little Fires Everywhere	Celeste Ng	4.5	25706	12	2018	Fiction

Rows per page: 100 | 1 - 100 of 550 | First page | < | > | Last page

PERSONAL HISTORY | PROJECT HISTORY | SAVED QUERIES