Step-by-Step Guide for Deployment on GCP with Correct File Paths

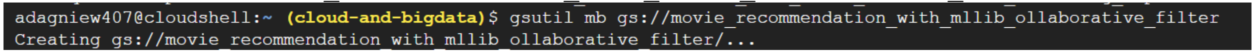
1. Upload Data and Scripts to GCS

Make sure you have already uploaded `movies.csv`, `ratings.csv`, and your PySpark script (e.g., `Recommendation_Engine_MovieLens.py`) to your GCS bucket as described earlier.

2. Create a Google Cloud Storage (GCS) Bucket

- Create a bucket in GCS to store your scripts and data.

```
gsutil mb gs://movie_recommendation_with_mllib_ollaborative_filter
```

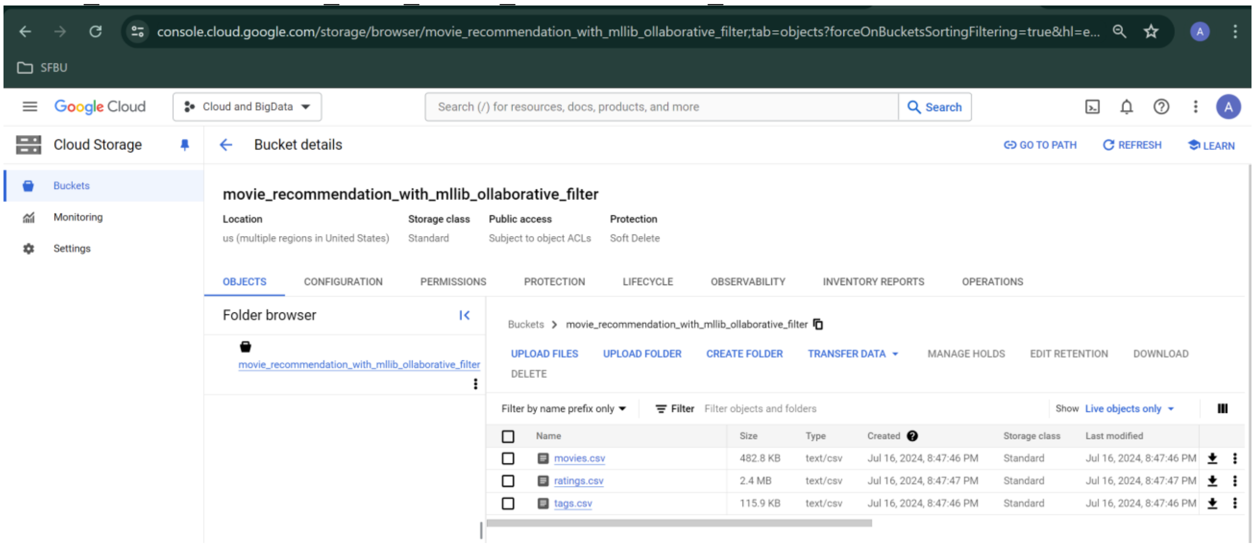


```
adagniew407@cloudshell:~ (cloud-and-bigdata)$ gsutil mb gs://movie_recommendation_with_mllib_ollaborative_filter
Creating gs://movie_recommendation_with_mllib_ollaborative_filter/...
```

3. Upload Data and Scripts to GCS

- Upload the `movies.csv`, `ratings.csv`, and your PySpark script (e.g., `Recommendation_Engine_MovieLens.py`) to your GCS bucket.

```
gsutil cp movies.csv gs://
movie_recommendation_with_mllib_ollaborative_filter/
gsutil cp ratings.csv gs://
movie_recommendation_with_mllib_ollaborative_filter/
gsutil cp Recommendation_Engine_MovieLens.py gs://
movie_recommendation_with_mllib_ollaborative_filter
```



console.cloud.google.com/storage/browser/movie_recommendation_with_mllib_ollaborative_filter;tab=objects?forceOnBucketsSortingFiltering=true&hl=e...
SFBU
Google Cloud
Cloud and BigData
Search (/) for resources, docs, products, and more
Search
Cloud Storage
Buckets
Monitoring
Settings
Bucket details
GO TO PATH
REFRESH
LEARN
OBJECTS
CONFIGURATION
PERMISSIONS
PROTECTION
LIFECYCLE
OBSERVABILITY
INVENTORY REPORTS
OPERATIONS
Folder browser
movie_recommendation_with_mllib_ollaborative_filter
Buckets > movie_recommendation_with_mllib_ollaborative_filter
UPLOAD FILES
UPLOAD FOLDER
CREATE FOLDER
TRANSFER DATA
MANAGE HOLDS
EDIT RETENTION
DOWNLOAD
DELETE
Filter by name prefix only
Filter
Filter objects and folders
Show Live objects only

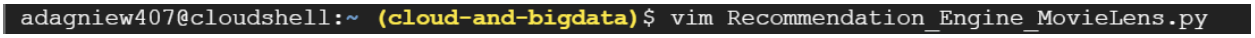
movie_recommendation_with_mllib_ollaborative_filter

Location	Storage class	Public access	Protection
us (multiple regions in United States)	Standard	Subject to object ACLs	Soft Delete

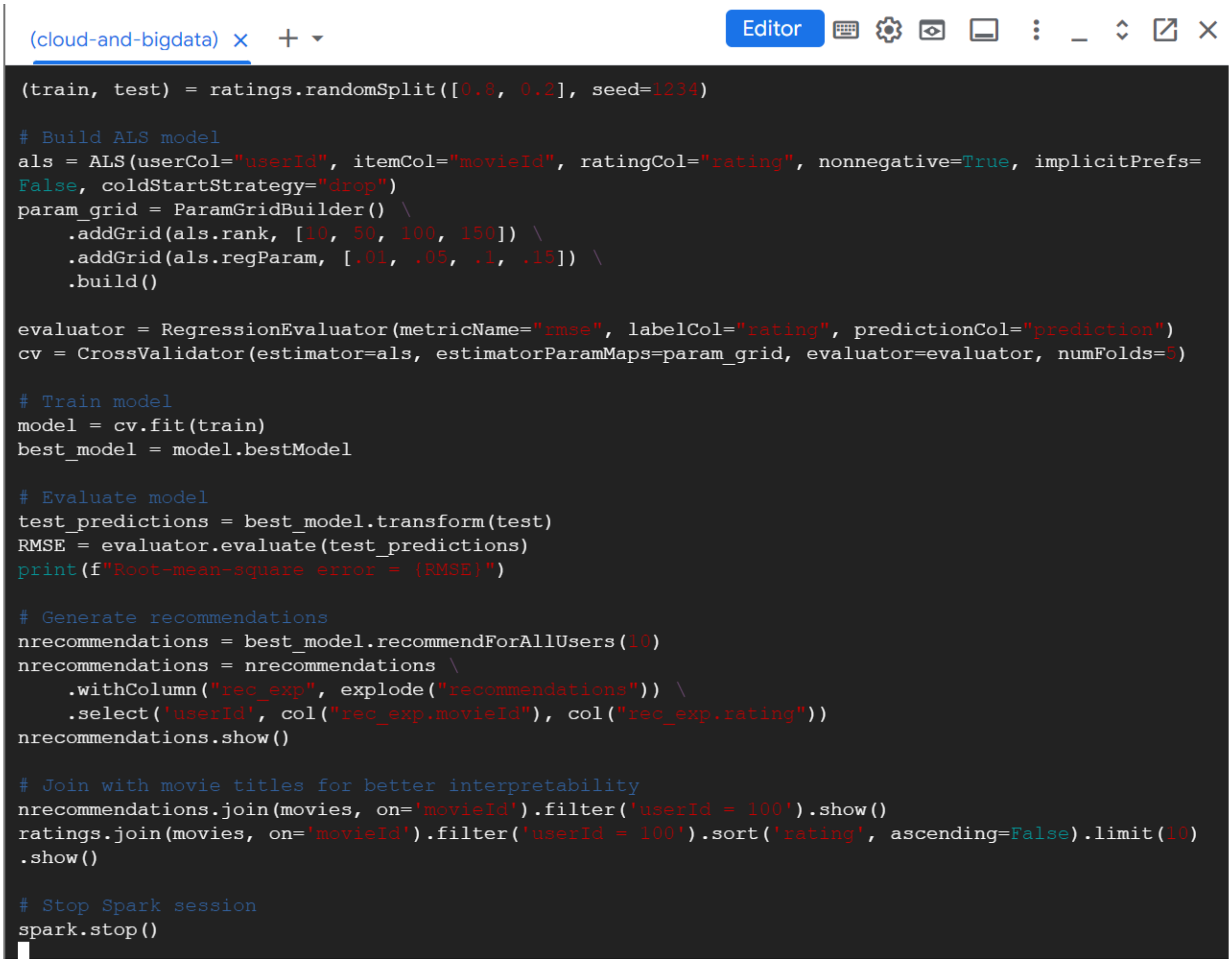
☐	Name	Size	Type	Created	Storage class	Last modified
☐	movies.csv	482.8 KB	text/csv	Jul 16, 2024, 8:47:46 PM	Standard	Jul 16, 2024, 8:47:46 PM
☐	ratings.csv	2.4 MB	text/csv	Jul 16, 2024, 8:47:47 PM	Standard	Jul 16, 2024, 8:47:47 PM
☐	tags.csv	115.9 KB	text/csv	Jul 16, 2024, 8:47:46 PM	Standard	Jul 16, 2024, 8:47:46 PM

2. Modify the PySpark Script to Use GCS Paths

Update your PySpark script to read the files from GCS. You can use command-line arguments to pass the paths of the CSV files, making the script more flexible.



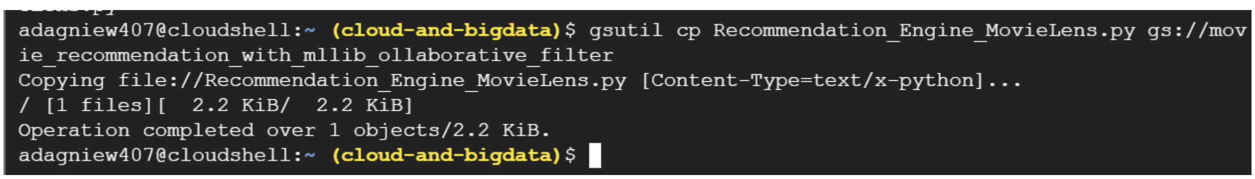
```
adagniew407@cloudshell:~ (cloud-and-bigdata)$ vim Recommendation_Engine_MovieLens.py
```



```
(train, test) = ratings.randomSplit([0.8, 0.2], seed=1234)

# Build ALS model
als = ALS(userCol="userId", itemCol="movieId", ratingCol="rating", nonnegative=True, implicitPrefs=
False, coldStartStrategy="drop")
param_grid = ParamGridBuilder() \
    .addGrid(als.rank, [10, 50, 100, 150]) \
    .addGrid(als.regParam, [.01, .05, .1, .15]) \
    .build()

evaluator = RegressionEvaluator(metricName="rmse", labelCol="rating", predictionCol="prediction")
cv = CrossValidator(estimator=als, estimatorParamMaps=param_grid, evaluator=evaluator, numFolds=5)

# Train model
model = cv.fit(train)
best_model = model.bestModel

# Evaluate model
test_predictions = best_model.transform(test)
RMSE = evaluator.evaluate(test_predictions)
print(f"Root-mean-square error = {RMSE}")

# Generate recommendations
nrecommendations = best_model.recommendForAllUsers(10)
nrecommendations = nrecommendations \
    .withColumn("rec_exp", explode("recommendations")) \
    .select('userId', col("rec_exp.movieId"), col("rec_exp.rating"))
nrecommendations.show()

# Join with movie titles for better interpretability
nrecommendations.join(movies, on='movieId').filter('userId = 100').show()
ratings.join(movies, on='movieId').filter('userId = 100').sort('rating', ascending=False).limit(10)
.show()

# Stop Spark session
spark.stop()
```

Then, upload it to the bucket.

```
gsutil cp Recommendation_Engine_MovieLens.py
gs://movie_recommendation_with_mllib_ollaborative_filter
```



```
adagniew407@cloudshell:~ (cloud-and-bigdata)$ gsutil cp Recommendation_Engine_MovieLens.py gs://mov
ie_recommendation_with_mllib_ollaborative_filter
Copying file://Recommendation_Engine_MovieLens.py [Content-Type=text/x-python]...
/ [1 files][  2.2 KiB/  2.2 KiB]
Operation completed over 1 objects/2.2 KiB.
adagniew407@cloudshell:~ (cloud-and-bigdata)$
```

Here is the script:

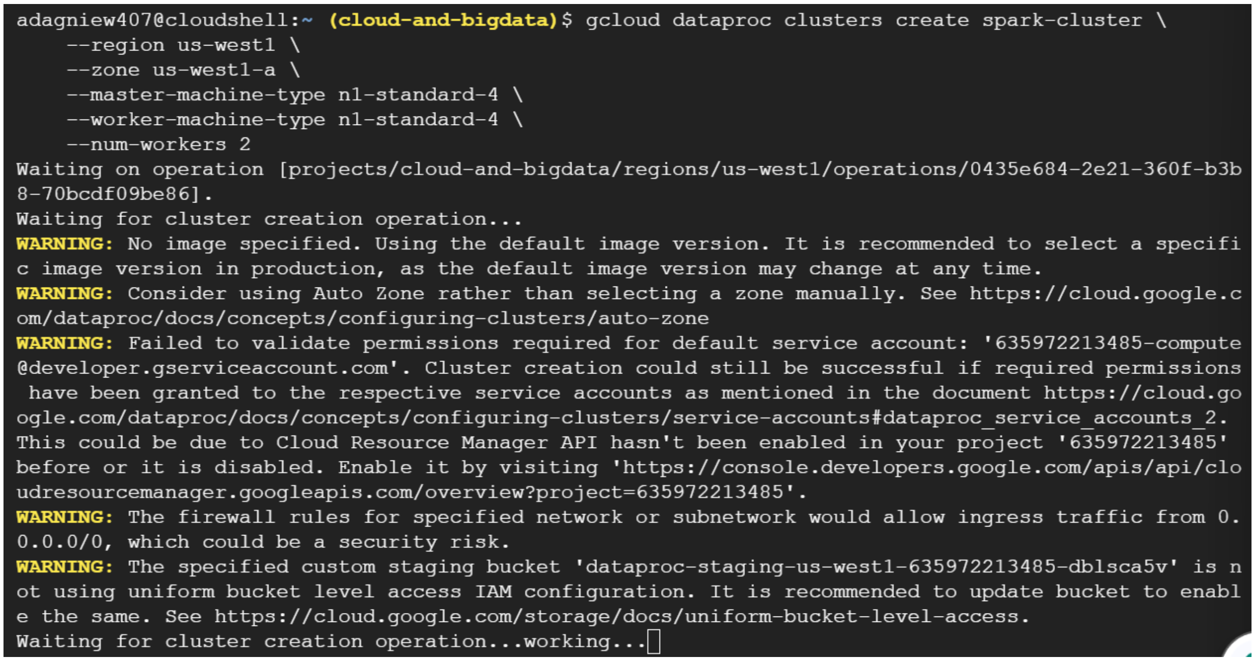
```
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, explode
from pyspark.ml.evaluation import RegressionEvaluator
from pyspark.ml.recommendation import ALS
from pyspark.ml.tuning import ParamGridBuilder, CrossValidator

import argparse
```

```
# Parse command-line arguments
parser = argparse.ArgumentParser()
parser.add_argument('--input_path_movies', required=True)
parser.add_argument('--input_path_ratings', required=True)
args = parser.parse_args()

# Initialize Spark session
spark = SparkSession.builder.appName('Recommendations').getOrCreate()

# Load data from GCS
movies = spark.read.csv(args.input_path_movies, header=True)
ratings = spark.read.csv(args.input_path_ratings, header=True)

# Preprocess data
ratings = ratings \
    .withColumn('userId', col('userId').cast('integer')) \
    .withColumn('movieId', col('movieId').cast('integer')) \
    .withColumn('rating', col('rating').cast('float')) \
    .drop('timestamp')

# Split data into training and testing sets
(train, test) = ratings.randomSplit([0.8, 0.2], seed=1234)

# Build ALS model
als = ALS(userCol="userId", itemCol="movieId", ratingCol="rating",
nonnegative=True, implicitPrefs=False, coldStartStrategy="drop")
param_grid = ParamGridBuilder() \
    .addGrid(als.rank, [10, 50, 100, 150]) \
    .addGrid(als.regParam, [.01, .05, .1, .15]) \
    .build()

evaluator = RegressionEvaluator(metricName="rmse", labelCol="rating",
predictionCol="prediction")
cv = CrossValidator(estimator=als, estimatorParamMaps=param_grid,
evaluator=evaluator, numFolds=5)

# Train model
model = cv.fit(train)
best_model = model.bestModel

# Evaluate model
test_predictions = best_model.transform(test)
RMSE = evaluator.evaluate(test_predictions)
print(f"Root-mean-square error = {RMSE}")

# Generate recommendations
nrecommendations = best_model.recommendForAllUsers(10)
nrecommendations = nrecommendations \
    .withColumn("rec_exp", explode("recommendations")) \
    .select('userId', col("rec_exp.movieId"), col("rec_exp.rating"))
nrecommendations.show()

# Join with movie titles for better interpretability
nrecommendations.join(movies, on='movieId').filter('userId = 100').show()
ratings.join(movies, on='movieId').filter('userId = 100').sort('rating',
ascending=False).limit(10).show()
```

```
# Stop Spark session
spark.stop()
```

3. create the cluster with the desired configuration:

```
gcloud dataproc clusters create spark-cluster \
    --region us-west1 \
    --zone us-west1-a \
    --master-machine-type n1-standard-4 \
    --worker-machine-type n1-standard-4 \
    --num-workers 2
```

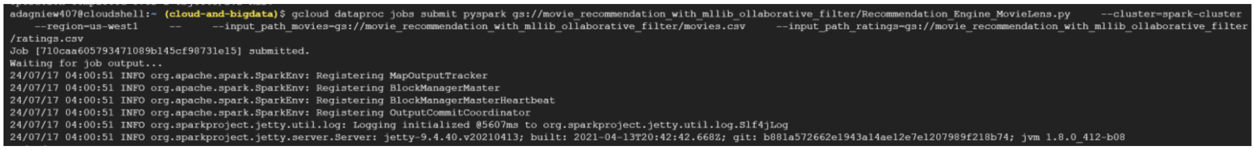


```
adagniew407@cloudshell:~ (cloud-and-bigdata)$ gcloud dataproc clusters create spark-cluster \
    --region us-west1 \
    --zone us-west1-a \
    --master-machine-type n1-standard-4 \
    --worker-machine-type n1-standard-4 \
    --num-workers 2
Waiting on operation [projects/cloud-and-bigdata/regions/us-west1/operations/0435e684-2e21-360f-b3b
8-70bcdf09be86].
Waiting for cluster creation operation...
WARNING: No image specified. Using the default image version. It is recommended to select a specifi
c image version in production, as the default image version may change at any time.
WARNING: Consider using Auto Zone rather than selecting a zone manually. See https://cloud.google.c
om/dataproc/docs/concepts/configuring-clusters/auto-zone
WARNING: Failed to validate permissions required for default service account: '635972213485-compute
@developer.gserviceaccount.com'. Cluster creation could still be successful if required permissions
 have been granted to the respective service accounts as mentioned in the document https://cloud.go
ogle.com/dataproc/docs/concepts/configuring-clusters/service-accounts#dataproc_service_accounts_2.
This could be due to Cloud Resource Manager API hasn't been enabled in your project '635972213485'
before or it is disabled. Enable it by visiting 'https://console.developers.google.com/apis/api/clo
udresourcemanager.googleapis.com/overview?project=635972213485'.
WARNING: The firewall rules for specified network or subnetwork would allow ingress traffic from 0.
0.0.0/0, which could be a security risk.
WARNING: The specified custom staging bucket 'dataproc-staging-us-west1-635972213485-dblsca5v' is n
ot using uniform bucket level access IAM configuration. It is recommended to update bucket to enabl
e the same. See https://cloud.google.com/storage/docs/uniform-bucket-level-access.
Waiting for cluster creation operation...working...
```

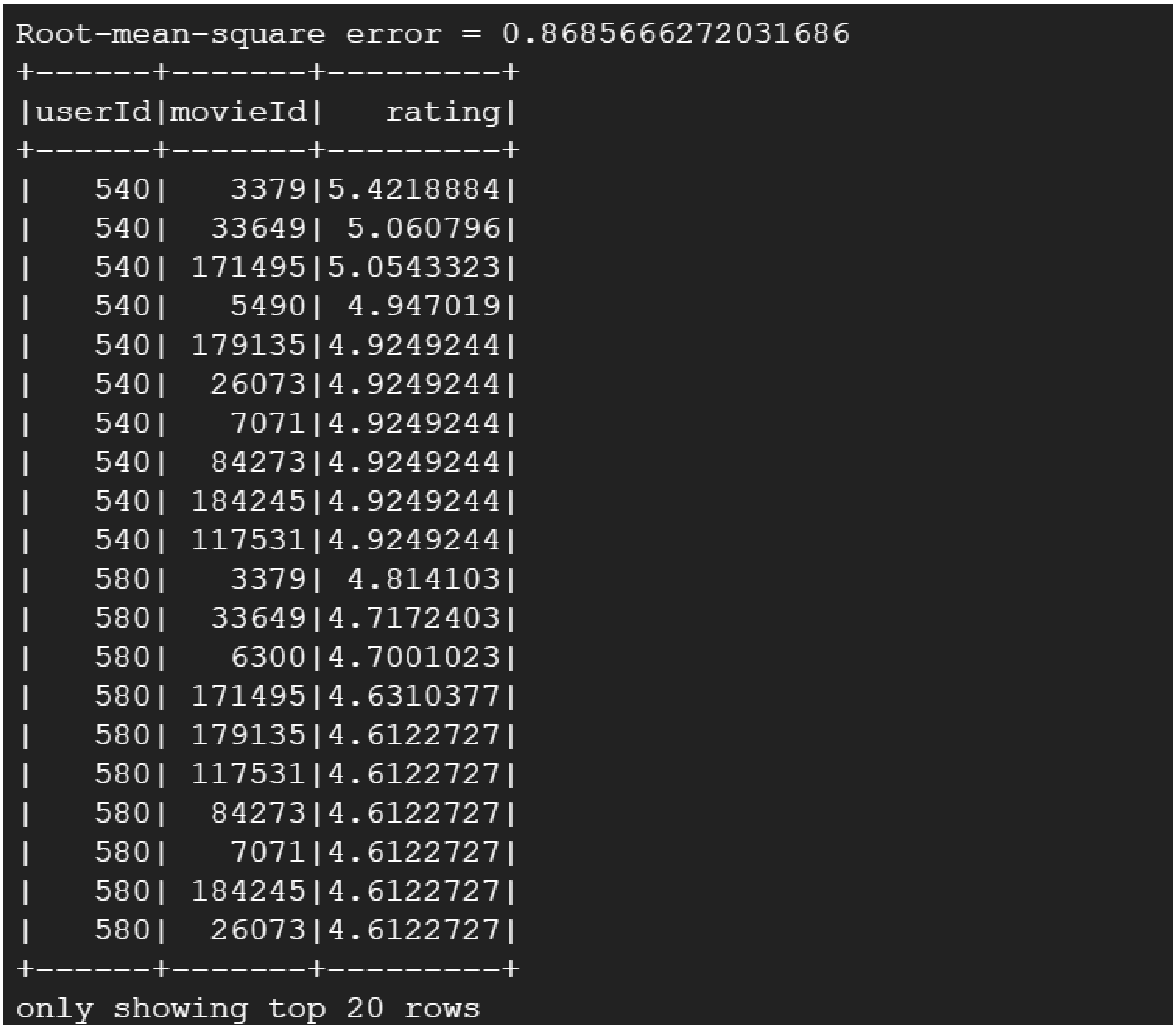
4. Submit the PySpark Job with GCS Paths

Submit your PySpark job to the Dataproc cluster, providing the GCS paths for the input files:

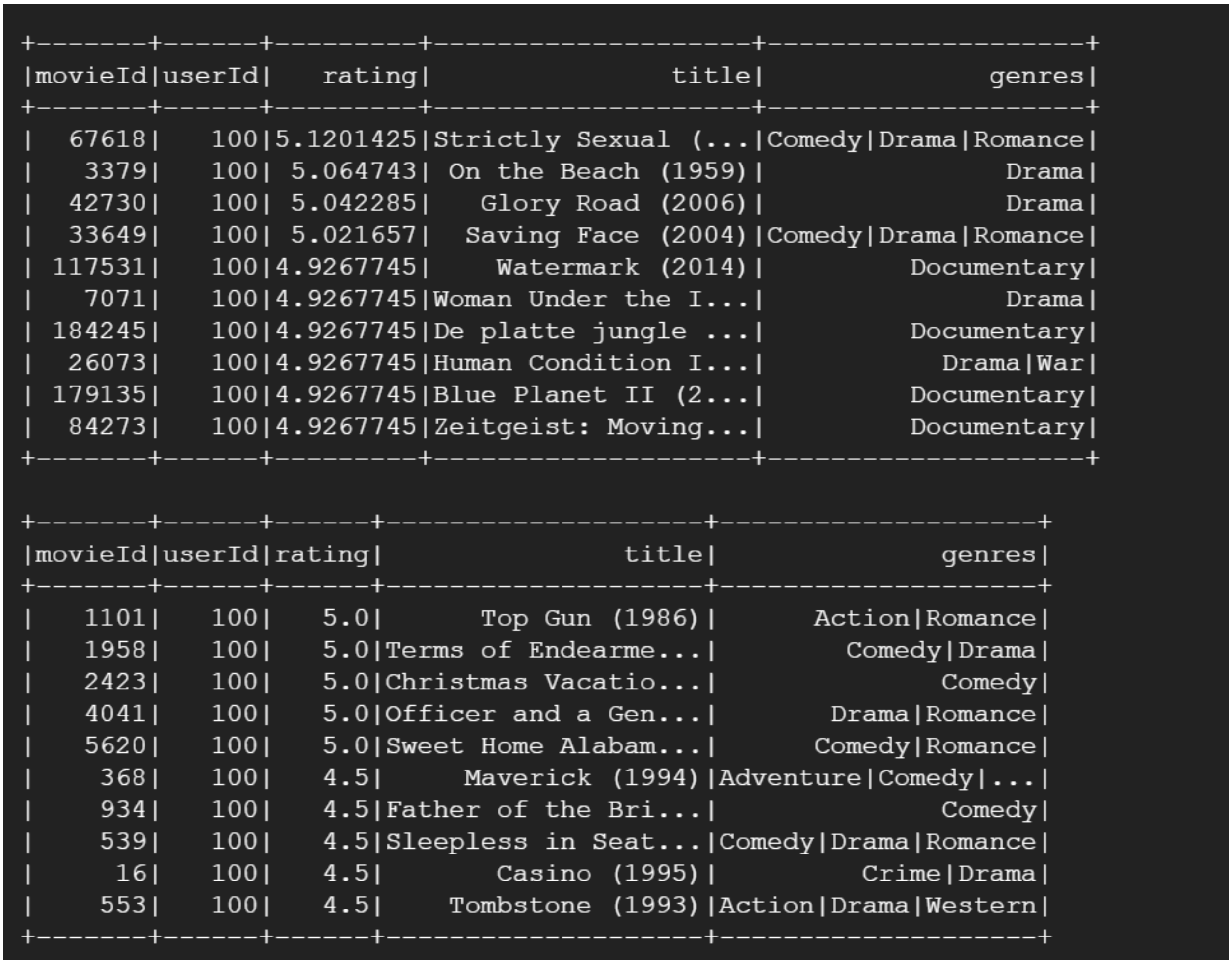
```
gcloud dataproc jobs submit pyspark gs://your-bucket-
name/Recommendation_Engine_MovieLens.py \
    --cluster=spark-cluster \
    --region=us-west1 \
    -- \
    --input_path_movies=gs://
movie_recommendation_with_mllib_ollaborative_filter/movies.csv \
    --input_path_ratings=gs://
movie_recommendation_with_mllib_ollaborative_filter/ratings.csv

Replace your-bucket-name with the actual name of your GCS bucket.
```



```
adagniew407@cloudshell:~ (cloud-and-bigdata)$ gcloud dataproc jobs submit pyspark gs://movie_recommendation_with_mllib_ollaborative_filter/Recommendation_Engine_MovieLens.py     --cluster=spark-cluster
    --region=us-west1     --     --input_path_movies=gs://movie_recommendation_with_mllib_ollaborative_filter/movies.csv     --input_path_ratings=gs://movie_recommendation_with_mllib_ollaborative_filter
/ratings.csv
Job [710caa605793471089b145cf98731e15] submitted.
Waiting for job output...
24/07/17 04:00:51 INFO org.apache.spark.SparkEnv: Registering MapOutputTracker
24/07/17 04:00:51 INFO org.apache.spark.SparkEnv: Registering BlockManagerMaster
24/07/17 04:00:51 INFO org.apache.spark.SparkEnv: Registering BlockManagerMasterHeartbeat
24/07/17 04:00:51 INFO org.apache.spark.SparkEnv: Registering OutputCommitCoordinator
24/07/17 04:00:51 INFO org.sparkproject.jetty.util.log: Logging initialized @5607ms to org.sparkproject.jetty.util.log.Slf4jLog
24/07/17 04:00:51 INFO org.sparkproject.jetty.server.Server: jetty-9.4.40.v20210413; built: 2021-04-13T20:42:42.668Z; git: b881a572662e1943a14ae12e7e1207989f218b74; jvm 1.8.0_412-b08
```



```
Root-mean-square error = 0.8685666272031686
+------+-------+---------+
|userId|movieId|   rating|
+------+-------+---------+
|   540|   3379|5.4218884|
|   540|  33649| 5.060796|
|   540| 171495|5.0543323|
|   540|   5490| 4.947019|
|   540| 179135|4.9249244|
|   540|  26073|4.9249244|
|   540|   7071|4.9249244|
|   540|  84273|4.9249244|
|   540| 184245|4.9249244|
|   540| 117531|4.9249244|
|   580|   3379| 4.814103|
|   580|  33649|4.7172403|
|   580|   6300|4.7001023|
|   580| 171495|4.6310377|
|   580| 179135|4.6122727|
|   580| 117531|4.6122727|
|   580|  84273|4.6122727|
|   580|   7071|4.6122727|
|   580| 184245|4.6122727|
|   580|  26073|4.6122727|
+------+-------+---------+
only showing top 20 rows
```



```
+-------+------+---------+--------------------+--------------------+
|movieId|userId|   rating|               title|              genres|
+-------+------+---------+--------------------+--------------------+
|  67618|   100|5.1201425|Strictly Sexual (...|Comedy|Drama|Romance|
|   3379|   100| 5.064743| On the Beach (1959)|               Drama|
|  42730|   100| 5.042285|   Glory Road (2006)|               Drama|
|  33649|   100| 5.021657|  Saving Face (2004)|Comedy|Drama|Romance|
| 117531|   100|4.9267745|    Watermark (2014)|         Documentary|
|   7071|   100|4.9267745|Woman Under the I...|               Drama|
| 184245|   100|4.9267745|De platte jungle ...|         Documentary|
|  26073|   100|4.9267745|Human Condition I...|           Drama|War|
| 179135|   100|4.9267745|Blue Planet II (2...|         Documentary|
|  84273|   100|4.9267745|Zeitgeist: Moving...|         Documentary|
+-------+------+---------+--------------------+--------------------+

+-------+------+------+--------------------+--------------------+
|movieId|userId|rating|               title|              genres|
+-------+------+------+--------------------+--------------------+
|   1101|   100|   5.0|      Top Gun (1986)|      Action|Romance|
|   1958|   100|   5.0|Terms of Endearme...|        Comedy|Drama|
|   2423|   100|   5.0|Christmas Vacatio...|              Comedy|
|   4041|   100|   5.0|Officer and a Gen...|       Drama|Romance|
|   5620|   100|   5.0|Sweet Home Alabam...|      Comedy|Romance|
|    368|   100|   4.5|     Maverick (1994)|Adventure|Comedy|...|
|    934|   100|   4.5|Father of the Bri...|              Comedy|
|    539|   100|   4.5|Sleepless in Seat...|Comedy|Drama|Romance|
|     16|   100|   4.5|       Casino (1995)|         Crime|Drama|
|    553|   100|   4.5|    Tombstone (1993)|Action|Drama|Western|
+-------+------+------+--------------------+--------------------+
```

By following these steps, your PySpark script will correctly read the files from GCS when running on GCP Dataproc.