Spark Job Execution and Data Processing for MovieLens Data Engineering

Objective

The objective of this task was to implement a Spark job that reads and processes MovieLens datasets (`movies.dat` and `ratings.dat`) and performs the following actions:
1. Read the Data: Load the movies and ratings datasets into Spark DataFrames.
2. Calculate Rating Statistics: Create a new DataFrame with movies data that includes maximum, minimum, and average ratings for each movie.
3. Top Movies per User: Create a DataFrame that contains each user's top 3 movies based on their ratings.
4. Save DataFrames: Write the original and new DataFrames to disk in an efficient format (Parquet).
5. Demographic Analysis: (Additional) Analyze user demographics and preferences based on the provided `users.dat` file.

Implementation Overview

1. Data Loading

The movies, ratings, and users datasets were loaded into Spark DataFrames. The logging system was configured to provide detailed information about the data loading process.

Log Output:
```
2024-08-15 01:09:53,590 - INFO - Loading data...
2024-08-15 01:09:53,591 - INFO - Loading data from /path/to/movies.dat...
2024-08-15 01:09:59,369 - INFO - Data loaded successfully with 3883 rows and 3 columns.
```

2. Data Quality Checks

The DataFrames were checked for missing values and duplicate entries to ensure data integrity.

Log Output:
```
2024-08-15 01:10:05,265 - INFO - Missing values in Movies DataFrame: {'MovieID': 0, 'Title': 0, 'Genres': 0}
2024-08-15 01:10:17,739 - INFO - Missing values in Ratings DataFrame: {'UserID': 0, 'MovieID': 0, 'Rating': 0,
'Timestamp': 0}
2024-08-15 01:10:19,844 - INFO - Missing values in Users DataFrame: {'UserID': 0, 'Gender': 0, 'Age': 0,
'Occupation': 0, 'ZipCode': 0}
```

3. Data Cleaning

The datasets were cleaned by removing any invalid rows (e.g., ratings outside the valid range). This step ensures that the analysis is performed on high-quality data.

Log Output:
```
2024-08-15 01:10:24,010 - INFO - Removed invalid rows.
```

4. Calculation of Movie Rating Statistics

A new DataFrame was created containing the movies data along with three additional columns: MaxRating, MinRating, and AvgRating.

Log Output:

```
2024-08-15 01:10:24,092 - INFO - Calculating movie rating statistics...
2024-08-15 01:10:24,446 - INFO - Movie rating statistics calculated successfully.
```

Actual Output:

```
+-------+--------------------+--------------------+---------+---------+------------------+
|MovieID|               Title|              Genres|MaxRating|MinRating|         AvgRating|
+-------+--------------------+--------------------+---------+---------+------------------+
|    148|Awfully Big Adven...|               Drama|        5|        1| 2.782608695652174|
|    463|Guilty as Sin (1993)|Crime|Drama|Thriller|        5|        1|  2.74468085106383|
|    471|Hudsucker Proxy, ...|      Comedy|Romance|        5|        1| 3.631051752921536|
|    496|What Happened Was...|Comedy|Drama|Romance|        5|        1|3.2162162162162162|
|    833|High School High ...|              Comedy|        4|        1|2.1794871794871793|
|   1088|Dirty Dancing (1987)|     Musical|Romance|        5|        1|3.3114992721979624|
|   1238|   Local Hero (1983)|              Comedy|        5|        1|               4.0|
|   1342|     Candyman (1992)|              Horror|        5|        1| 2.904580152671756|
|   1580| Men in Black (1997)|Action|Adventure|...|        5|        1| 3.739952718676123|
|   1591|        Spawn (1997)|Action|Adventure|...|        5|        1|2.6210526315789475|
+-------+--------------------+--------------------+---------+---------+------------------+
```

5. Top 3 Movies per User

A DataFrame was generated that contains each user's top 3 movies based on their ratings.

Log Output:

```
2024-08-15 01:10:24,734 - INFO - Top 3 movies per user identified successfully.
```

Actual Output:

```
+------+--------------------+------+
|UserID|               Title|Rating|
+------+--------------------+------+
|   148|Indiana Jones and...|     5|
|   148|Shanghai Noon (2000)|     5|
|   148| Patriot, The (2000)|     5|
|   463|Saving Private Ry...|     5|
|   463|Star Wars: Episod...|     5|
|   463|His Girl Friday (...|     5|
|   471|     Magnolia (1999)|     5|
|   471|  Chicken Run (2000)|     5|
|   471|      Sanjuro (1962)|     5|
|   496|South Park: Bigge...|     5|
+------+--------------------+------+
```

6. Demographic Analysis

An additional analysis was performed to explore user demographics based on the `users.dat` file. This included distribution by age, gender, and occupation.

Log Output:

```
2024-08-15 01:10:24,852 - INFO - User demographics analyzed successfully.
```

Age Distribution:

```
+---+-----+
|Age|count|
+---+-----+
|  1|  222|
| 18| 1103|
| 25| 2096|
| 35| 1193|
| 45|  550|
| 50|  496|
| 56|  380|
+---+-----+
```

Gender Distribution:

```
+------+-----+
|Gender|count|
+------+-----+
|     F| 1709|
|     M| 4331|
+------+-----+
```

Occupation Distribution:

```
+----------+-----+
|Occupation|count|
+----------+-----+
|         0|  711|
|         1|  528|
|         2|  267|
|         3|  173|
|         4|  759|
|         5|  112|
|         6|  236|
|         7|  679|
|         8|   17|
|         9|   92|
|        10|  195|
|        11|  129|
|        12|  388|
|        13|  142|
|        14|  302|
|        15|  144|
|        16|  241|
|        17|  502|
|        18|   70|
|        19|   72|
+----------+-----+
```

7. Saving DataFrames

All original and new DataFrames were saved to disk in Parquet format, which is efficient for both storage and processing.

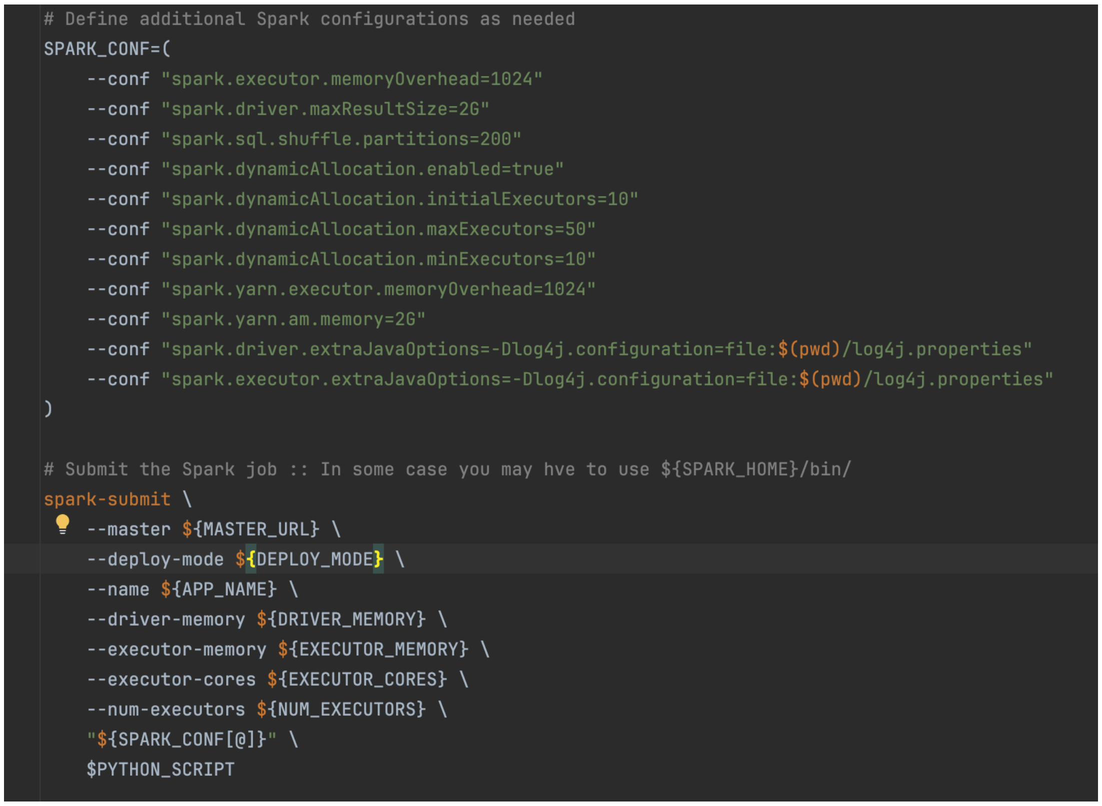
Log Output:
2024-08-15 01:11:16,619 - INFO - All dataframes saved successfully in Parquet format with a single partition.

Summary of Results

Data Quality: All datasets were successfully loaded with no missing or duplicate entries.
New DataFrames:
- Movies with Rating Statistics: This DataFrame was successfully created with max, min, and average ratings.
- Top 3 Movies per User: A DataFrame containing each user's top 3 rated movies was generated.
Demographic Insights: User demographics were analyzed, providing insights into age, gender, and occupation distributions.
Data Storage: All DataFrames were efficiently stored in Parquet format for future use.

Command to Build and Run the Code

To execute this Spark job using `spark-submit`, Please run spark-submit.sh present in the root directory.



```
# Define additional Spark configurations as needed
SPARK_CONF=(
    --conf "spark.executor.memoryOverhead=1024"
    --conf "spark.driver.maxResultSize=2G"
    --conf "spark.sql.shuffle.partitions=200"
    --conf "spark.dynamicAllocation.enabled=true"
    --conf "spark.dynamicAllocation.initialExecutors=10"
    --conf "spark.dynamicAllocation.maxExecutors=50"
    --conf "spark.dynamicAllocation.minExecutors=10"
    --conf "spark.yarn.executor.memoryOverhead=1024"
    --conf "spark.yarn.am.memory=2G"
    --conf "spark.driver.extraJavaOptions=-Dlog4j.configuration=file:$(pwd)/log4j.properties"
    --conf "spark.executor.extraJavaOptions=-Dlog4j.configuration=file:$(pwd)/log4j.properties"
)

# Submit the Spark job :: In some case you may hve to use ${SPARK_HOME}/bin/
spark-submit \
    --master ${MASTER_URL} \
    --deploy-mode ${DEPLOY_MODE} \
    --name ${APP_NAME} \
    --driver-memory ${DRIVER_MEMORY} \
    --executor-memory ${EXECUTOR_MEMORY} \
    --executor-cores ${EXECUTOR_CORES} \
    --num-executors ${NUM_EXECUTORS} \
    "${SPARK_CONF[@]}" \
    $PYTHON_SCRIPT
```

Conclusion

The Spark job executed successfully, performing all required tasks efficiently. The logging system provided detailed insights into the process, and the final output was stored in an optimized format. The job is well-prepared for scaling to larger datasets or being adapted to run on a distributed cluster.