**PySpark DataFrame Case Study: Movie Ratings Analysis**

Scenario: You have a dataset of movie ratings submitted by users. The goal is to analyze user preferences, average ratings, and identify popular movies. The dataset contains the following columns:

• user\_id: ID of the user

• movie\_id: ID of the movie

• rating: Rating given by the user (1-5 scale)

• timestamp: Time of the rating

**Sample Data**

+--------+---------+------+-------------------+  
|user\_id |movie\_id |rating|timestamp |  
+--------+---------+------+-------------------+  
|1 |101 |4 |1622388000000 |  
|1 |102 |3 |1622388020000 |  
|2 |101 |5 |1622388040000 |  
|2 |103 |4 |1622388060000 |  
|3 |101 |3 |1622388080000 |  
|3 |102 |4 |1622388100000 |  
|3 |103 |NULL |1622388120000 |  
|4 |101 |2 |1622388140000 |  
+--------+---------+------+-------------------+

**Code to Create DataFrame**

from pyspark.sql import SparkSession  
from pyspark.sql.functions import col, avg  
  
# Initialize Spark session  
spark = SparkSession.builder.appName("Movie Ratings Analysis").getOrCreate()  
  
# Sample data  
data = [  
 (1, 101, 4, 1622388000000),  
 (1, 102, 3, 1622388020000),  
 (2, 101, 5, 1622388040000),  
 (2, 103, 4, 1622388060000),  
 (3, 101, 3, 1622388080000),  
 (3, 102, 4, 1622388100000),  
 (3, 103, None, 1622388120000),  
 (4, 101, 2, 1622388140000)  
]  
  
# Create a DataFrame  
columns = ["user\_id", "movie\_id", "rating", "timestamp"]  
df = spark.createDataFrame(data, schema=columns)  
df.show()

**1. Filter**

Find all ratings that are greater than or equal to 4.

df\_filtered = df.filter(df.rating >= 4)  
df\_filtered.show()

**2. Handle Null Values**

Replace null values in the 'rating' column with the average rating.

average\_rating = df.selectExpr('avg(rating)').collect()[0][0]  
df\_filled = df.na.fill({'rating': average\_rating})  
df\_filled.show()

**3. Drop Duplicates**

Remove duplicate ratings by 'user\_id' and 'movie\_id'.

df\_no\_duplicates = df.dropDuplicates(['user\_id', 'movie\_id'])  
df\_no\_duplicates.show()

**4. Select Specific Columns**

Select 'user\_id' and 'rating' columns.

df\_selected = df.select('user\_id', 'rating')  
df\_selected.show()

**5. Grouping and Aggregating**

Calculate the average rating per movie.

df\_grouped = df.groupBy('movie\_id').agg({'rating': 'avg'})  
df\_grouped.show()

**6. Joining DataFrames**

Join the 'movie\_ratings' DataFrame with a 'movie\_details' DataFrame that contains 'movie\_id' and 'movie\_name'.

# Assuming df\_movies is another DataFrame that contains movie details  
df\_joined = df.join(df\_movies, on='movie\_id', how='inner')  
df\_joined.show()

**7. Union of DataFrames**

Union this DataFrame with another 'df\_new\_ratings' containing additional ratings data.

df\_union = df.union(df\_new\_ratings)  
df\_union.show()

**8. Temporary View and SQL**

Create a temp view and find the top-rated movies.

df.createOrReplaceTempView('ratings')  
sql\_result = spark.sql('SELECT movie\_id, AVG(rating) as avg\_rating FROM ratings GROUP BY movie\_id ORDER BY avg\_rating DESC LIMIT 10')  
sql\_result.show()