**PySpark DataFrame Case Study: Sales Transactions Analysis**

Scenario: You have a dataset of sales transactions, and the goal is to analyze total sales, popular products, and customer purchasing patterns. The dataset contains the following columns:

• transaction\_id: Transaction ID

• customer\_id: Customer ID

• product\_id: Product ID

• quantity: Quantity purchased

• price: Price per unit

• date: Transaction date

**Sample Data**

+---------------+------------+-----------+---------+-------+-------------+  
| transaction\_id| customer\_id| product\_id| quantity| price| date |  
+---------------+------------+-----------+---------+-------+-------------+  
| 1001 | 2001 | 3001 | 5 | 20.0 | 2024-09-01 |  
| 1002 | 2002 | 3002 | 2 | 50.0 | 2024-09-01 |  
| 1003 | 2003 | 3003 | 1 | 120.0| 2024-09-02 |  
| 1004 | 2001 | 3002 | 3 | 40.0 | 2024-09-03 |  
| 1005 | 2004 | 3001 | 10 | 15.0 | 2024-09-03 |  
| 1006 | 2005 | 3004 | NULL | 30.0 | 2024-09-03 |  
+---------------+------------+-----------+---------+-------+-------------+

**Code to Create DataFrame**

from pyspark.sql import SparkSession  
from pyspark.sql.functions import col, sum  
  
# Initialize Spark session  
spark = SparkSession.builder.appName("Sales Transactions Analysis").getOrCreate()  
  
# Sample data  
data = [  
 (1001, 2001, 3001, 5, 20.0, '2024-09-01'),  
 (1002, 2002, 3002, 2, 50.0, '2024-09-01'),  
 (1003, 2003, 3003, 1, 120.0, '2024-09-02'),  
 (1004, 2001, 3002, 3, 40.0, '2024-09-03'),  
 (1005, 2004, 3001, 10, 15.0, '2024-09-03'),  
 (1006, 2005, 3004, None, 30.0, '2024-09-03')  
]  
  
# Create DataFrame  
columns = ["transaction\_id", "customer\_id", "product\_id", "quantity", "price", "date"]  
df = spark.createDataFrame(data, schema=columns)  
df.show()

**1. Filter**

Find transactions where the total amount (quantity \* price) is greater than $100.

df\_filtered = df.filter((df.quantity \* df.price) > 100)  
df\_filtered.show()

**2. Handle Null Values**

Replace null values in the 'quantity' column with 0.

df\_filled = df.na.fill({'quantity': 0})  
df\_filled.show()

**3. Drop Duplicates**

Remove duplicate transactions by 'transaction\_id'.

df\_no\_duplicates = df.dropDuplicates(['transaction\_id'])  
df\_no\_duplicates.show()

**4. Select Specific Columns**

Select 'customer\_id', 'product\_id', and 'quantity'.

df\_selected = df.select('customer\_id', 'product\_id', 'quantity')  
df\_selected.show()

**5. Grouping and Aggregating**

Calculate total sales per product.

df\_grouped = df.groupBy('product\_id').agg({'quantity': 'sum'})  
df\_grouped.show()

**6. Joining DataFrames**

Join with a 'product\_details' DataFrame containing 'product\_id' and 'product\_name'.

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

**7. Union of DataFrames**

Union this DataFrame with another 'df\_new\_transactions'.

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

**8. Temporary View and SQL**

Create a temp view and calculate the total sales per customer.

df.createOrReplaceTempView('transactions')  
sql\_result = spark.sql('SELECT customer\_id, SUM(quantity \* price) as total\_spent FROM transactions GROUP BY customer\_id')  
sql\_result.show()