**PYSPARK EXERCISES 3.9.2024**

from pyspark.sql import SparkSession

from pyspark.sql.functions import col

**# Initialize SparkSession**

spark = SparkSession.builder \

    .appName("Product Sales Analysis") \

    .getOrCreate()

**# Sample data for products**

products = [

    (1, "Laptop", "Electronics", 50000),

    (2, "Smartphone", "Electronics", 30000),

    (3, "Table", "Furniture", 15000),

    (4, "Chair", "Furniture", 5000),

    (5, "Headphones", "Electronics", 2000),

]

**# Sample data for sales transactions**

sales = [

    (1, 1, 2),

    (2, 2, 1),

    (3, 3, 3),

    (4, 1, 1),

    (5, 4, 5),

    (6, 2, 2),

    (7, 5, 10),

    (8, 3, 1),

]

**# Define schema for DataFrames**

product\_columns = ["ProductID", "ProductName", "Category", "Price"]

sales\_columns = ["SaleID", "ProductID", "Quantity"]

**# Create DataFrames**

product\_df = spark.createDataFrame(products, schema=product\_columns)

sales\_df = spark.createDataFrame(sales, schema=sales\_columns)

**# Show the DataFrames**

print("Products DataFrame:")

product\_df.show()

print("Sales DataFrame:")

sales\_df.show()

**1. #Join the DataFrames:\*\***

**# Join the `product\_df` and `sales\_df` DataFrames on `ProductID` to create a combined DataFrame with product and sales data.**

combined\_df=product\_df.join(sales\_df,on="ProductID")

print("Combined DataFrame for product and sales data:")

combined\_df.show()

**2. #\*Calculate Total Sales Value:\*\***

**#For each product, calculate the total sales value by multiplying the price by the quantity sold**.

sales\_value\_df = combined\_df.withColumn("TotalSalesValue", col("Price") \* col("Quantity"))

print("Combined DataFrame with Total Sales Value:")

sales\_value\_df.show()

**3. #Find the Total Sales for Each Product Category:\*\***

**#Group the data by the `Category` column and calculate the total sales value for each product category.**

total\_sales\_per\_category\_df = sales\_value\_df.groupBy("Category") \

                                            .sum("TotalSalesValue") \

                                            .withColumnRenamed("sum(TotalSalesValue)", "TotalSalesValue")

total\_sales\_per\_category\_df.show()

**4. #Identify the Top-Selling Product:\*\***

**#Find the product that generated the highest total sales value.**

total\_sales\_per\_product\_df = sales\_value\_df.groupBy("ProductID", "ProductName", "Category") \

                                           .sum("TotalSalesValue") \

                                           .withColumnRenamed("sum(TotalSalesValue)", "TotalSalesValue")

sorted\_products\_df = total\_sales\_per\_product\_df.orderBy(col("TotalSalesValue").desc())

top\_selling\_product = sorted\_products\_df.limit(1)

print("Top-Selling Product:")

top\_selling\_product.show()

**5. \*Sort the Products by Total Sales Value:\*\***

**- Sort the products by total sales value in descending order.**

total\_sales\_per\_product\_df = sales\_value\_df.groupBy("ProductID", "ProductName", "Category") \

                                           .sum("TotalSalesValue") \

                                           .withColumnRenamed("sum(TotalSalesValue)", "TotalSalesValue")

sorted\_products\_df = total\_sales\_per\_product\_df.orderBy(col("TotalSalesValue").desc())

print("Products Sorted by Total Sales Value (Descending):")

sorted\_products\_df.show()

**6. #\*Count the Number of Sales for Each Product:\*\***

**# Count the number of sales transactions for each product.**

sales\_count\_with\_product\_df = sales\_df.groupBy("ProductID") \

.count() \

.join(product\_df, "ProductID") \

.select("ProductID", "ProductName", "Category", "count")

print("Number of Sales Transactions for Each Product:")

sales\_count\_with\_product\_df.show()

**7. #Filter the Products with Total Sales Value Greater Than ₹50,000:\*\***

**#Filter out the products that have a total sales value greater than ₹50,000.**

filtered\_products\_df = total\_sales\_per\_product\_df.filter(col("TotalSalesValue") > 50000)

print("Products with Total Sales Value Greater Than ₹50,000:")

filtered\_products\_df.show()