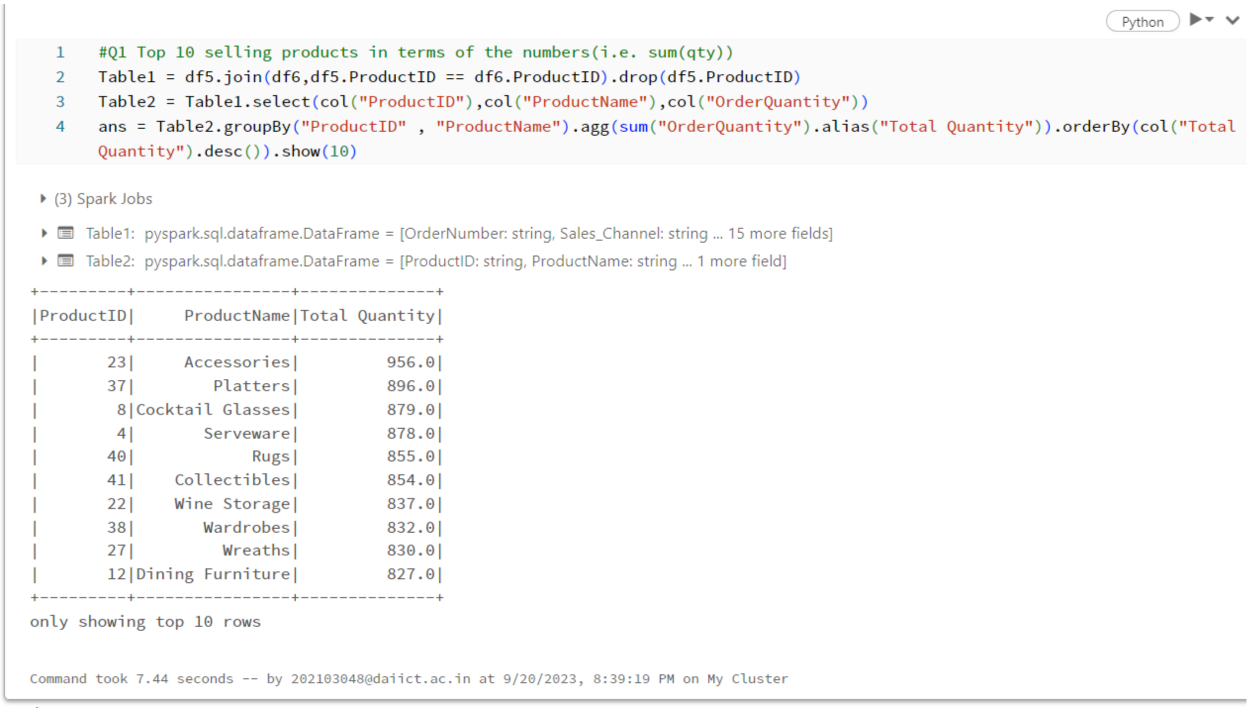
Name :- Raj Kariya
ID :- 202103048
Link:-https://databricks-prod-cloudfront.cloud.databricks.com/public/4027ec902e239c93eaaa8714f173bcfc/4164694028803888/686432533056833/5036389632654272/latest.html
Data(For accessing Customers Folder’s files in spark):-

```
df1 = spark.read.format("csv").option("header", "true").load("dbfs:/FileStore/shared_uploads/202103048@daiict.ac.in/customers.csv")
df2 = spark.read.format("csv").option("header", "true").load("dbfs:/FileStore/shared_uploads/202103048@daiict.ac.in/regions.csv")
df3 = spark.read.format("csv").option("header", "true").load("dbfs:/FileStore/shared_uploads/202103048@daiict.ac.in/sales_teams.csv")
df4 = spark.read.format("csv").option("header", "true").load("dbfs:/FileStore/shared_uploads/202103048@daiict.ac.in/stores.csv")
df5 = spark.read.format("csv").option("header", "true").load("dbfs:/FileStore/shared_uploads/202103048@daiict.ac.in/orders.csv")
df6 = spark.read.format("csv").option("header", "true").load("dbfs:/FileStore/shared_uploads/202103048@daiict.ac.in/products.csv")
```

Lab-4

Q1) Compute top-10 selling products in terms of numbers (i. e. sum(qty))
Solution:

Code And Output



```
#Q1 Top 10 selling products in terms of the numbers(i.e. sum(qty))
Table1 = df5.join(df6,df5.ProductID == df6.ProductID).drop(df5.ProductID)
Table2 = Table1.select(col("ProductID"),col("ProductName"),col("OrderQuantity"))
ans = Table2.groupBy("ProductID" , "ProductName").agg(sum("OrderQuantity").alias("Total Quantity")).orderBy(col("Total Quantity").desc()).show(10)
```

▸ (3) Spark Jobs

▸ Table1: pyspark.sql.dataframe.DataFrame = [OrderNumber: string, Sales_Channel: string ... 15 more fields]

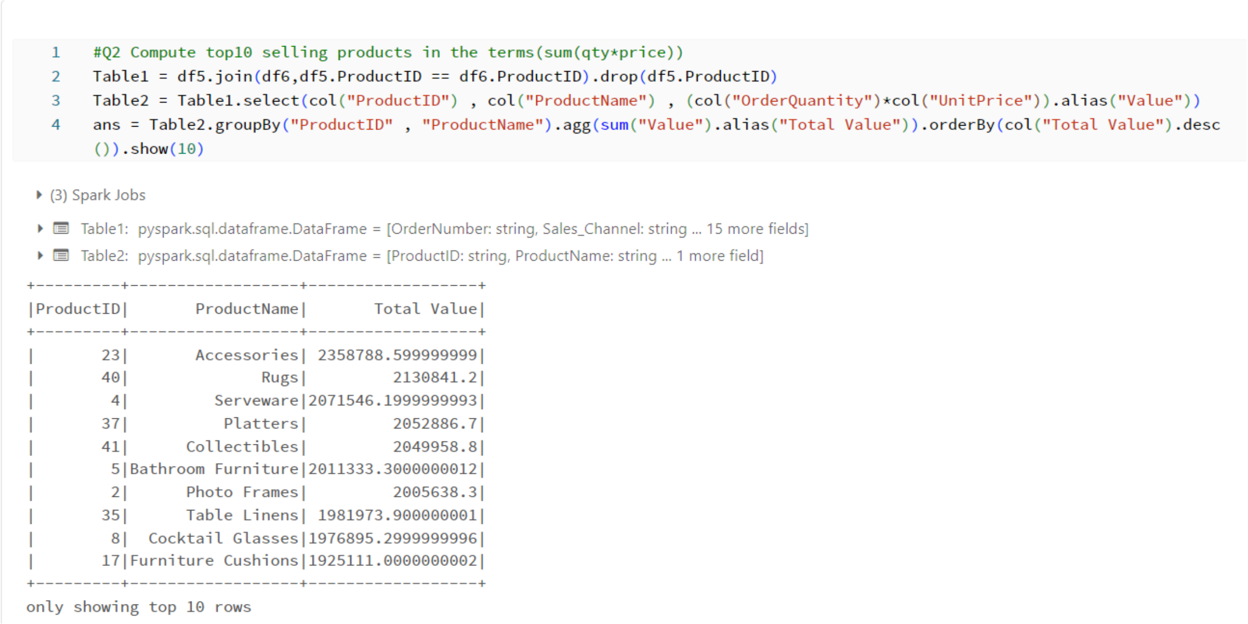
▸ Table2: pyspark.sql.dataframe.DataFrame = [ProductID: string, ProductName: string ... 1 more field]

```
+---------+----------------+--------------+
|ProductID|     ProductName|Total Quantity|
+---------+----------------+--------------+
|       23|     Accessories|         956.0|
|       37|        Platters|         896.0|
|        8|Cocktail Glasses|         879.0|
|        4|       Serveware|         878.0|
|       40|            Rugs|         855.0|
|       41|     Collectibles|         854.0|
|       22|    Wine Storage|         837.0|
|       38|       Wardrobes|         832.0|
|       27|         Wreaths|         830.0|
|       12|Dining Furniture|         827.0|
+---------+----------------+--------------+
only showing top 10 rows

Command took 7.44 seconds -- by 202103048@daiict.ac.in at 9/20/2023, 8:39:19 PM on My Cluster
```

Q2) Compute top-10 selling products in terms of value (i. e. sum (qty*price))

Code And Output



```
#Q2 Compute top10 selling products in the terms(sum(qty*price))
Table1 = df5.join(df6,df5.ProductID == df6.ProductID).drop(df5.ProductID)
Table2 = Table1.select(col("ProductID") , col("ProductName") , (col("OrderQuantity")*col("UnitPrice")).alias("Value"))
ans = Table2.groupBy("ProductID" , "ProductName").agg(sum("Value").alias("Total Value")).orderBy(col("Total Value").desc()).show(10)
```

▸ (3) Spark Jobs

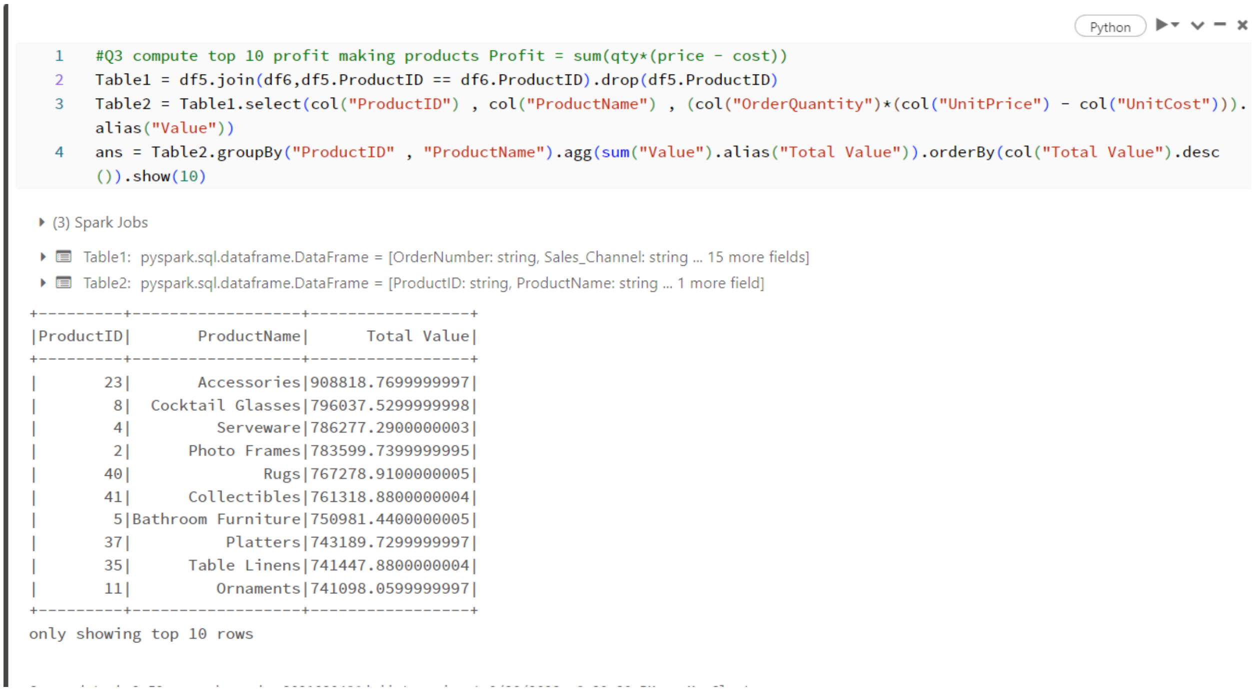
▸ Table1: pyspark.sql.dataframe.DataFrame = [OrderNumber: string, Sales_Channel: string ... 15 more fields]

▸ Table2: pyspark.sql.dataframe.DataFrame = [ProductID: string, ProductName: string ... 1 more field]

```
+---------+------------------+------------------+
|ProductID|       ProductName|       Total Value|
+---------+------------------+------------------+
|       23|       Accessories| 2358788.599999999|
|       40|              Rugs|         2130841.2|
|        4|         Serveware|2071546.1999999993|
|       37|          Platters|         2052886.7|
|       41|      Collectibles|         2049958.8|
|        5|Bathroom Furniture|2011333.3000000012|
|        2|      Photo Frames|         2005638.3|
|       35|      Table Linens| 1981973.900000001|
|        8|  Cocktail Glasses|1976895.2999999996|
|       17|Furniture Cushions|1925111.0000000002|
+---------+------------------+------------------+
only showing top 10 rows
```

Q3)Compute top-10 profit making products. Profit = sum(qty*(price-cost))

Code And Output



```python
#Q3 compute top 10 profit making products Profit = sum(qty*(price - cost))
Table1 = df5.join(df6,df5.ProductID == df6.ProductID).drop(df5.ProductID)
Table2 = Table1.select(col("ProductID") , col("ProductName") , (col("OrderQuantity")*(col("UnitPrice") - col("UnitCost"))).
alias("Value"))
ans = Table2.groupBy("ProductID" , "ProductName").agg(sum("Value").alias("Total Value")).orderBy(col("Total Value").desc
()).show(10)
```

(3) Spark Jobs

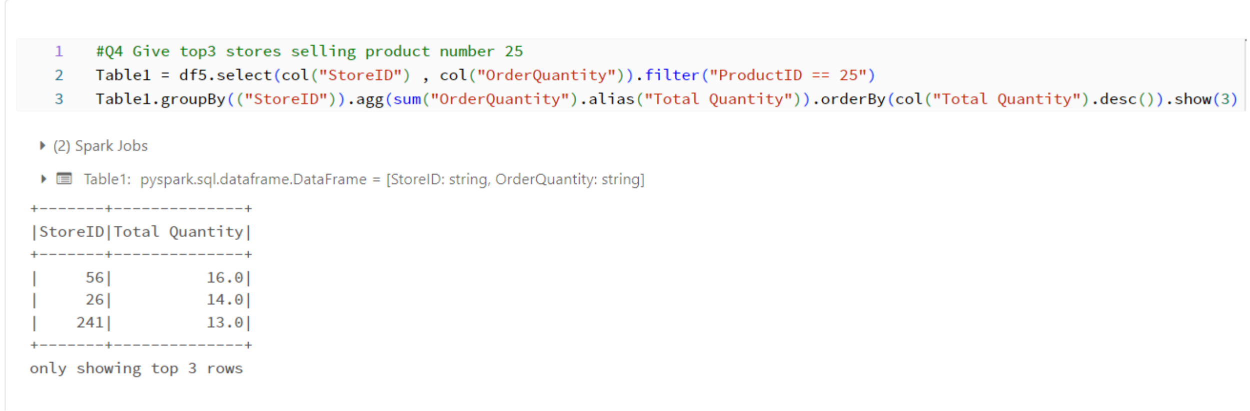
Table1: pyspark.sql.dataframe.DataFrame = [OrderNumber: string, Sales_Channel: string ... 15 more fields]

Table2: pyspark.sql.dataframe.DataFrame = [ProductID: string, ProductName: string ... 1 more field]

```
+---------+------------------+-----------------+
|ProductID|       ProductName|      Total Value|
+---------+------------------+-----------------+
|       23|       Accessories|908818.7699999997|
|        8|  Cocktail Glasses|796037.5299999998|
|        4|         Serveware|786277.2900000003|
|        2|      Photo Frames|783599.7399999995|
|       40|              Rugs|767278.9100000005|
|       41|      Collectibles|761318.8800000004|
|        5|Bathroom Furniture|750981.4400000005|
|       37|          Platters|743189.7299999997|
|       35|      Table Linens|741447.8800000004|
|       11|         Ornaments|741098.0599999997|
+---------+------------------+-----------------+
only showing top 10 rows
```

Q4)Give top-3 stores selling product product number 25
Solution:

Code And Output



```python
#Q4 Give top3 stores selling product number 25
Table1 = df5.select(col("StoreID") , col("OrderQuantity")).filter("ProductID == 25")
Table1.groupBy(("StoreID")).agg(sum("OrderQuantity").alias("Total Quantity")).orderBy(col("Total Quantity").desc()).show(3)
```

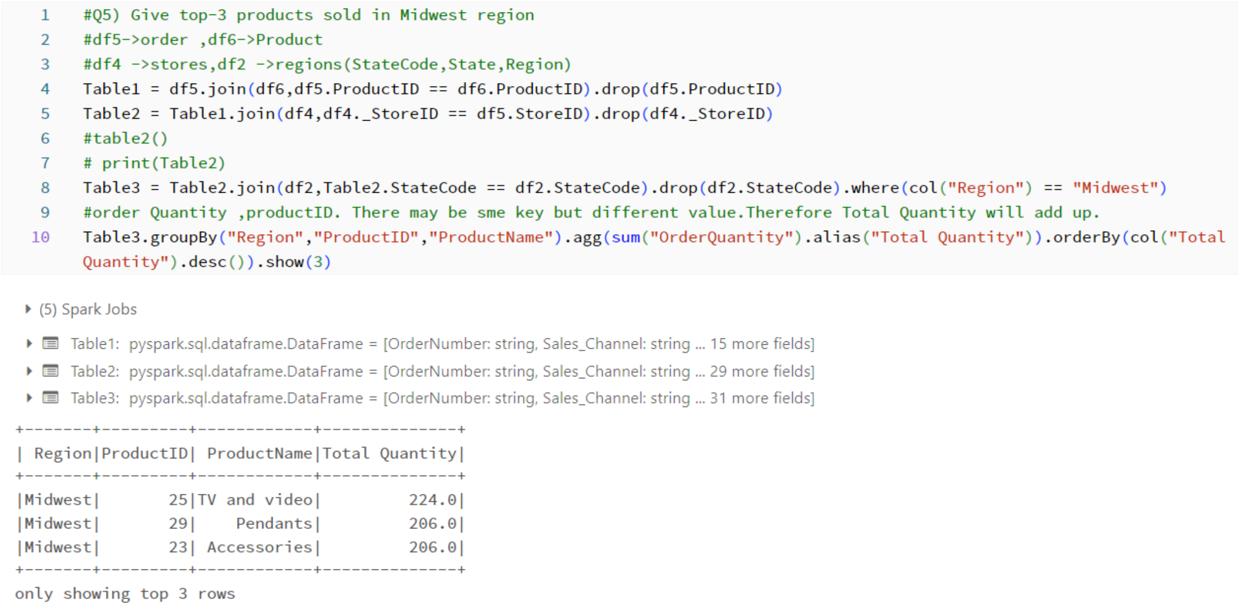
(2) Spark Jobs

Table1: pyspark.sql.dataframe.DataFrame = [StoreID: string, OrderQuantity: string]

```
+-------+--------------+
|StoreID|Total Quantity|
+-------+--------------+
|     56|          16.0|
|     26|          14.0|
|    241|          13.0|
+-------+--------------+
only showing top 3 rows
```

Q5). Give top-3 products sold in midwest region
Solution:

CodeAnd Output



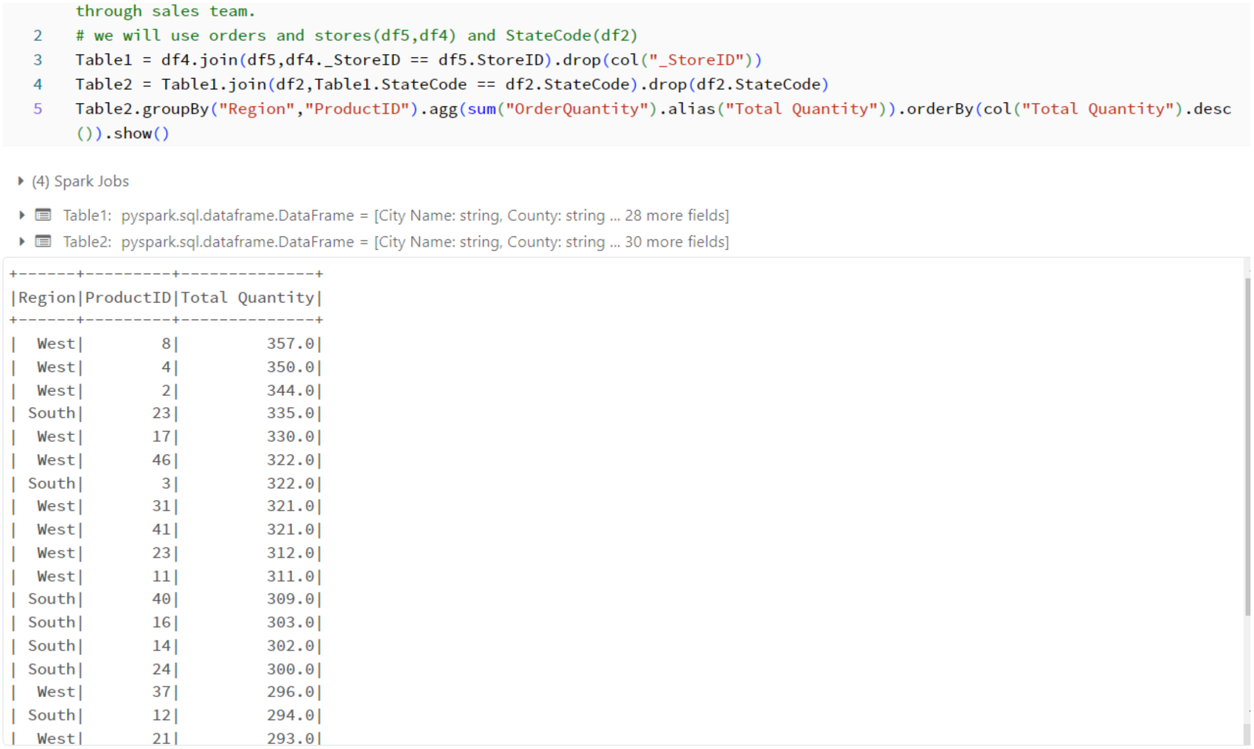
```
#Q5) Give top-3 products sold in Midwest region
#df5->order ,df6->Product
#df4 ->stores,df2 ->regions(StateCode,State,Region)
Table1 = df5.join(df6,df5.ProductID == df6.ProductID).drop(df5.ProductID)
Table2 = Table1.join(df4,df4._StoreID == df5.StoreID).drop(df4._StoreID)
#table2()
# print(Table2)
Table3 = Table2.join(df2,Table2.StateCode == df2.StateCode).drop(df2.StateCode).where(col("Region") == "Midwest")
#order Quantity ,productID. There may be sme key but different value.Therefore Total Quantity will add up.
Table3.groupBy("Region","ProductID","ProductName").agg(sum("OrderQuantity").alias("Total Quantity")).orderBy(col("Total Quantity").desc()).show(3)
```

```
(5) Spark Jobs
Table1:  pyspark.sql.dataframe.DataFrame = [OrderNumber: string, Sales_Channel: string ... 15 more fields]
Table2:  pyspark.sql.dataframe.DataFrame = [OrderNumber: string, Sales_Channel: string ... 29 more fields]
Table3:  pyspark.sql.dataframe.DataFrame = [OrderNumber: string, Sales_Channel: string ... 31 more fields]
+-------+---------+------------+--------------+
| Region|ProductID| ProductName|Total Quantity|
+-------+---------+------------+--------------+
|Midwest|       25|TV and video|         224.0|
|Midwest|       29|    Pendants|         206.0|
|Midwest|       23| Accessories|         206.0|
+-------+---------+------------+--------------+
only showing top 3 rows
```

Q6)Give region wise quantity sold for each product. Compute: Region, Product ID, Sum(Qty). Region is related to a order through sales team.

Solution:

Code And Output



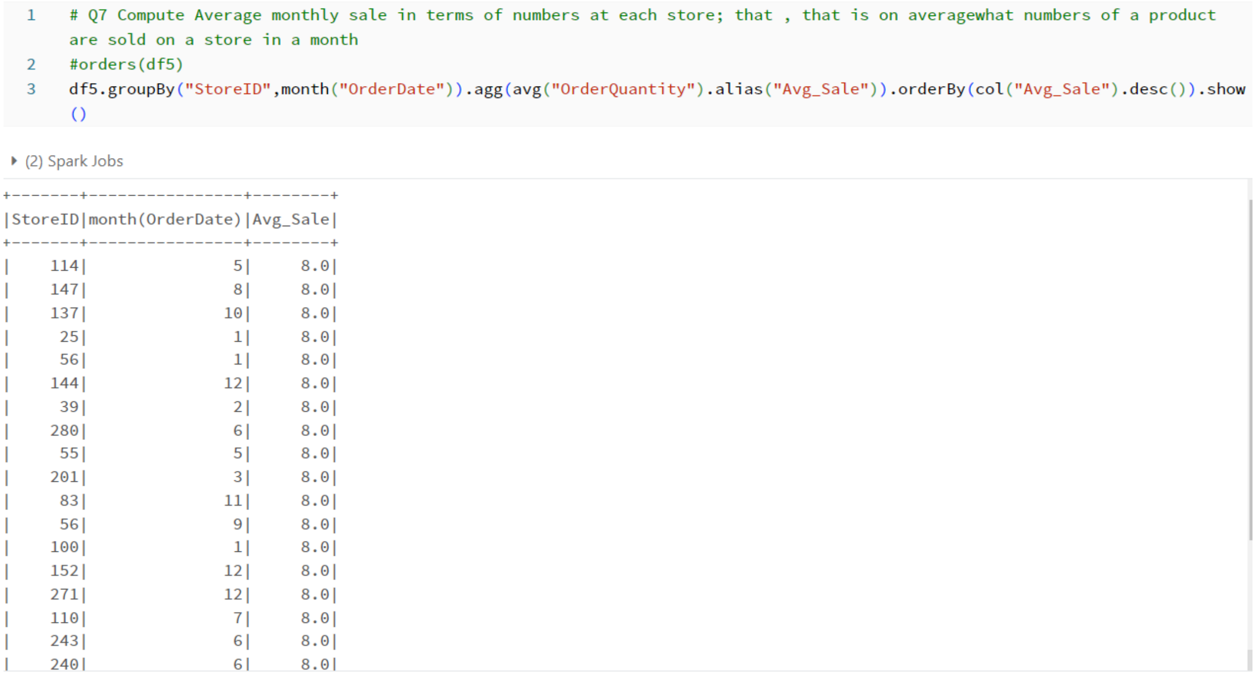
```
through sales team.
# we will use orders and stores(df5,df4) and StateCode(df2)
Table1 = df4.join(df5,df4._StoreID == df5.StoreID).drop(col("_StoreID"))
Table2 = Table1.join(df2,Table1.StateCode == df2.StateCode).drop(df2.StateCode)
Table2.groupBy("Region","ProductID").agg(sum("OrderQuantity").alias("Total Quantity")).orderBy(col("Total Quantity").desc
()).show()
```

```
▸ (4) Spark Jobs
▸ Table1:  pyspark.sql.dataframe.DataFrame = [City Name: string, County: string ... 28 more fields]
▸ Table2:  pyspark.sql.dataframe.DataFrame = [City Name: string, County: string ... 30 more fields]
```

```
+------+---------+--------------+
|Region|ProductID|Total Quantity|
+------+---------+--------------+
|  West|        8|         357.0|
|  West|        4|         350.0|
|  West|        2|         344.0|
| South|       23|         335.0|
|  West|       17|         330.0|
|  West|       46|         322.0|
| South|        3|         322.0|
|  West|       31|         321.0|
|  West|       41|         321.0|
|  West|       23|         312.0|
|  West|       11|         311.0|
| South|       40|         309.0|
| South|       16|         303.0|
| South|       14|         302.0|
| South|       24|         300.0|
|  West|       37|         296.0|
| South|       12|         294.0|
|  West|       21|         293.0|
```

Q7) Compute Average monthly sale in terms of numbers at each store; that , that is on average what numbers of a product are sold on a store in a month.
Solution:

Code And Output



```
# Q7 Compute Average monthly sale in terms of numbers at each store; that , that is on averagewhat numbers of a product
are sold on a store in a month
#orders(df5)
df5.groupBy("StoreID",month("OrderDate")).agg(avg("OrderQuantity").alias("Avg_Sale")).orderBy(col("Avg_Sale").desc()).show
()
```

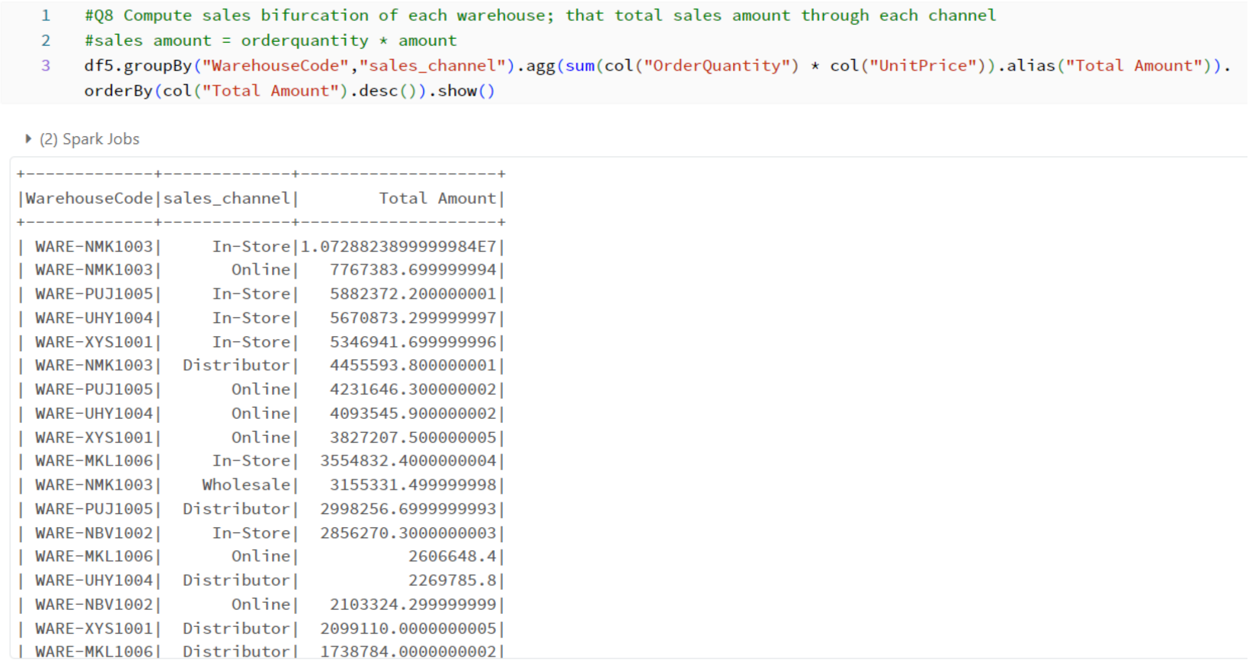
```
▸ (2) Spark Jobs
```

```
+-------+----------------+--------+
|StoreID|month(OrderDate)|Avg_Sale|
+-------+----------------+--------+
|    114|               5|     8.0|
|    147|               8|     8.0|
|    137|              10|     8.0|
|     25|               1|     8.0|
|     56|               1|     8.0|
|    144|              12|     8.0|
|     39|               2|     8.0|
|    280|               6|     8.0|
|     55|               5|     8.0|
|    201|               3|     8.0|
|     83|              11|     8.0|
|     56|               9|     8.0|
|    100|               1|     8.0|
|    152|              12|     8.0|
|    271|              12|     8.0|
|    110|               7|     8.0|
|    243|               6|     8.0|
|    240|               6|     8.0|
```

Q8) Compute sales bifurcation of each warehouse; that total sales amount through each channel

Solution:

Code And Output



```
#Q8 Compute sales bifurcation of each warehouse; that total sales amount through each channel
#sales amount = orderquantity * amount
df5.groupBy("WarehouseCode","sales_channel").agg(sum(col("OrderQuantity") * col("UnitPrice")).alias("Total Amount")).
orderBy(col("Total Amount").desc()).show()
```

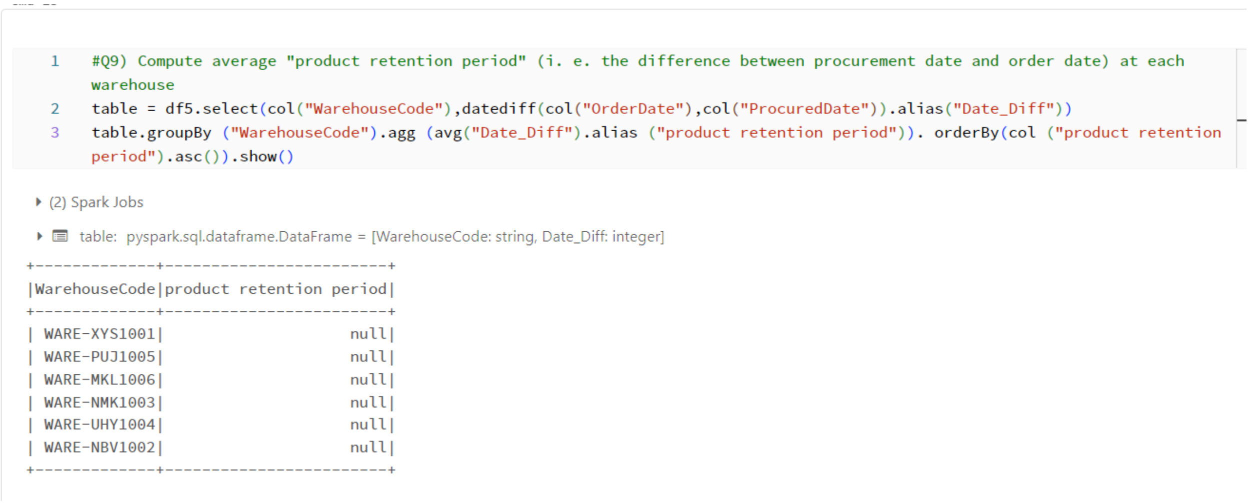
▸ (2) Spark Jobs

```
+-------------+-------------+--------------------+
|WarehouseCode|sales_channel|        Total Amount|
+-------------+-------------+--------------------+
| WARE-NMK1003|     In-Store|1.0728823899999984E7|
| WARE-NMK1003|       Online|   7767383.699999994|
| WARE-PUJ1005|     In-Store|   5882372.200000001|
| WARE-UHY1004|     In-Store|   5670873.299999997|
| WARE-XYS1001|     In-Store|   5346941.699999996|
| WARE-NMK1003|  Distributor|   4455593.800000001|
| WARE-PUJ1005|       Online|   4231646.300000002|
| WARE-UHY1004|       Online|   4093545.900000002|
| WARE-XYS1001|       Online|   3827207.500000005|
| WARE-MKL1006|     In-Store|  3554832.4000000004|
| WARE-NMK1003|    Wholesale|   3155331.499999998|
| WARE-PUJ1005|  Distributor|  2998256.6999999993|
| WARE-NBV1002|     In-Store|  2856270.3000000003|
| WARE-MKL1006|       Online|           2606648.4|
| WARE-UHY1004|  Distributor|           2269785.8|
| WARE-NBV1002|       Online|   2103324.299999999|
| WARE-XYS1001|  Distributor|  2099110.0000000005|
| WARE-MKL1006|  Distributor|  1738784.0000000002|
```

Q9) Compute average "product retention period" (i. e. the difference between procurement date and order date) at each warehouse

Solution:

Code And Output



```
#Q9) Compute average "product retention period" (i. e. the difference between procurement date and order date) at each
warehouse
table = df5.select(col("WarehouseCode"),datediff(col("OrderDate"),col("ProcuredDate")).alias("Date_Diff"))
table.groupBy ("WarehouseCode").agg (avg("Date_Diff").alias ("product retention period")). orderBy(col ("product retention
period").asc()).show()
```

▸ (2) Spark Jobs

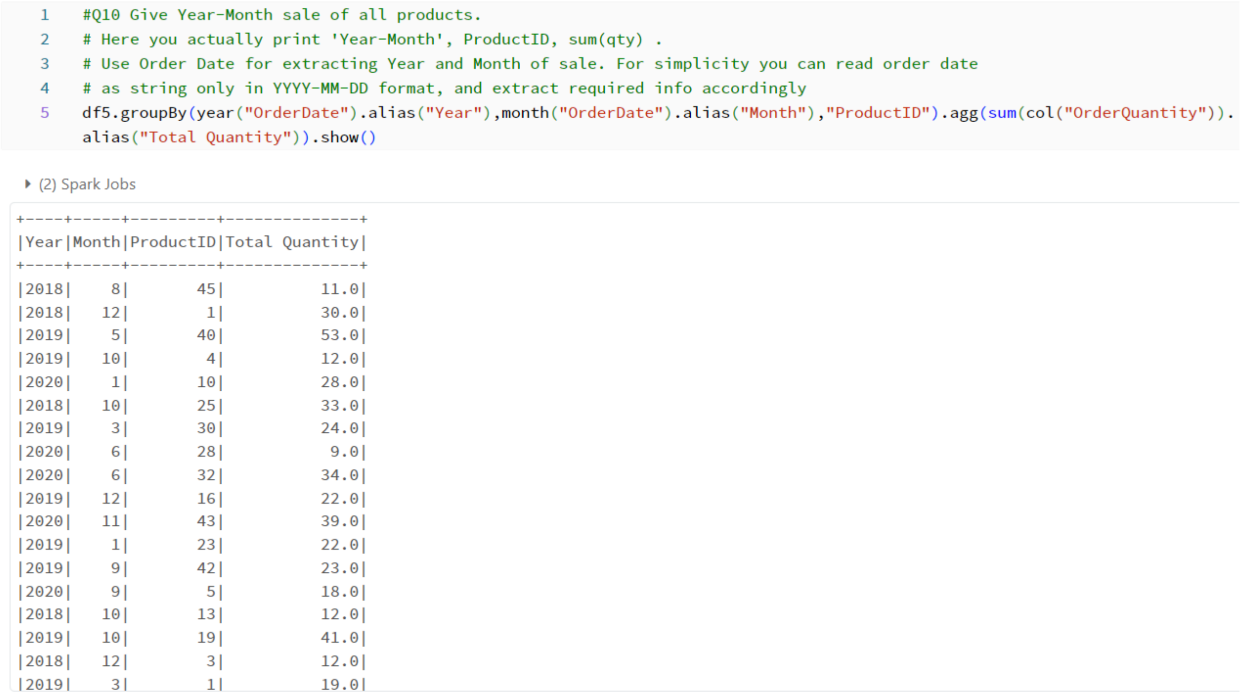
▸ table: pyspark.sql.dataframe.DataFrame = [WarehouseCode: string, Date_Diff: integer]

```
+-------------+------------------------+
|WarehouseCode|product retention period|
+-------------+------------------------+
| WARE-XYS1001|                    null|
| WARE-PUJ1005|                    null|
| WARE-MKL1006|                    null|
| WARE-NMK1003|                    null|
| WARE-UHY1004|                    null|
| WARE-NBV1002|                    null|
+-------------+------------------------+
```

Q10)Give Year-Month sale of all products. Here you actually print 'Year-Month', ProductID, sum(qty) . Use Order Date for extracting Year and Month of sale. For simplicity you can read order date as string only in YYYY-MM-DD format, and extract required info accordingly.

Solution:

Code And Output



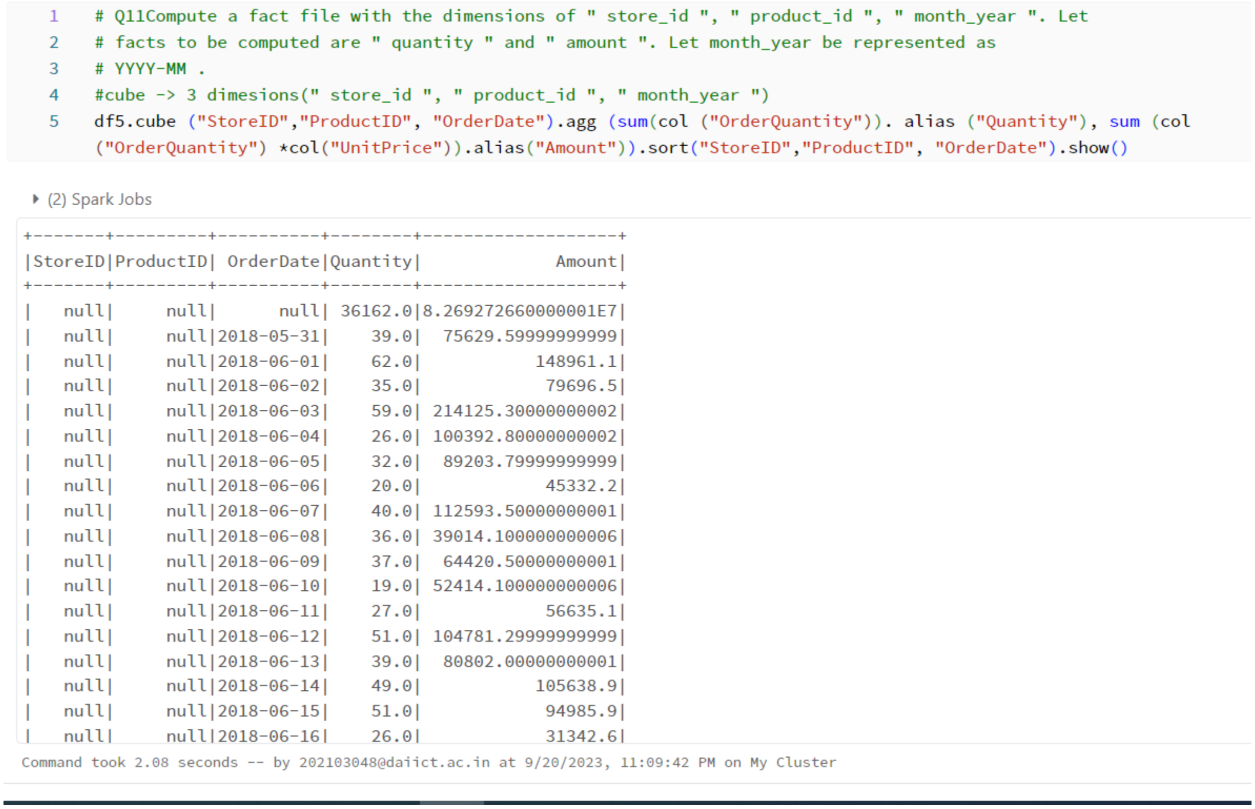
```
#Q10 Give Year-Month sale of all products.
# Here you actually print 'Year-Month', ProductID, sum(qty) .
# Use Order Date for extracting Year and Month of sale. For simplicity you can read order date
# as string only in YYYY-MM-DD format, and extract required info accordingly
df5.groupBy(year("OrderDate").alias("Year"),month("OrderDate").alias("Month"),"ProductID").agg(sum(col("OrderQuantity")).
alias("Total Quantity")).show()
```

```
+----+-----+---------+--------------+
|Year|Month|ProductID|Total Quantity|
+----+-----+---------+--------------+
|2018|    8|       45|          11.0|
|2018|   12|        1|          30.0|
|2019|    5|       40|          53.0|
|2019|   10|        4|          12.0|
|2020|    1|       10|          28.0|
|2018|   10|       25|          33.0|
|2019|    3|       30|          24.0|
|2020|    6|       28|           9.0|
|2020|    6|       32|          34.0|
|2019|   12|       16|          22.0|
|2020|   11|       43|          39.0|
|2019|    1|       23|          22.0|
|2019|    9|       42|          23.0|
|2020|    9|        5|          18.0|
|2018|   10|       13|          12.0|
|2019|   10|       19|          41.0|
|2018|   12|        3|          12.0|
|2019|    3|        1|          19.0|
```

Q11) Compute a fact file with the dimensions of " store_id ", " product_id ", " month_year ". Let facts to be computed are " quantity " and " amount ". Let month_year be represented as YYYY-MM.

Solution:

Code And Solution



```
# Q11Compute a fact file with the dimensions of " store_id ", " product_id ", " month_year ". Let
# facts to be computed are " quantity " and " amount ". Let month_year be represented as
# YYYY-MM .
#cube -> 3 dimesions(" store_id ", " product_id ", " month_year ")
df5.cube ("StoreID","ProductID", "OrderDate").agg (sum(col ("OrderQuantity")). alias ("Quantity"), sum (col
("OrderQuantity") *col("UnitPrice")).alias("Amount")).sort("StoreID","ProductID", "OrderDate").show()
```

```
+-------+---------+----------+--------+-------------------+
|StoreID|ProductID| OrderDate|Quantity|             Amount|
+-------+---------+----------+--------+-------------------+
|   null|     null|      null| 36162.0|8.269272660000001E7|
|   null|     null|2018-05-31|    39.0|  75629.59999999999|
|   null|     null|2018-06-01|    62.0|           148961.1|
|   null|     null|2018-06-02|    35.0|            79696.5|
|   null|     null|2018-06-03|    59.0| 214125.30000000002|
|   null|     null|2018-06-04|    26.0| 100392.80000000002|
|   null|     null|2018-06-05|    32.0|  89203.79999999999|
|   null|     null|2018-06-06|    20.0|            45332.2|
|   null|     null|2018-06-07|    40.0| 112593.50000000001|
|   null|     null|2018-06-08|    36.0| 39014.100000000006|
|   null|     null|2018-06-09|    37.0|  64420.50000000001|
|   null|     null|2018-06-10|    19.0| 52414.100000000006|
|   null|     null|2018-06-11|    27.0|            56635.1|
|   null|     null|2018-06-12|    51.0| 104781.29999999999|
|   null|     null|2018-06-13|    39.0|  80802.00000000001|
|   null|     null|2018-06-14|    49.0|           105638.9|
|   null|     null|2018-06-15|    51.0|            94985.9|
|   null|     null|2018-06-16|    26.0|            31342.6|
```

```
Command took 2.08 seconds -- by 202103048@daiict.ac.in at 9/20/2023, 11:09:42 PM on My Cluster
```