

**Retail Sales Analysis**

|  |
| --- |
| E. Divya Sree 2320625  A. N. V. Rohit 2320408  Sanmith Chetan N 2320907  Santanu Adhikary 2320846  Shashank K Hulmani 2320613  Vivechan Gowda G.C 2320755 |

**Contents**

1.Introduction --------------------------------------------------------------------- 03

a) Pyspark

2.Business Requirements -------------------------------------------------------- 04

3.Overview of the project ------------------------------------------------------- 05

4.Implementation -----------------------------------------------------------------06

5.Data Analysis & Visualisation --------------------------------------------------34

6.Conclusion ----------------------------------------------------------------------45

**Introduction**

Data has become an essential part of our daily lives in today’s digital age. From searching for a product on [e-commerce](https://www.analyticsvidhya.com/blog/2022/06/e-commerce-customer-churn-prediction/) platforms to placing an order and receiving it at home, we are constantly generating and consuming data. Today’s data-driven world generates data from various sectors like retail, automobile, finance, technology, aviation, food, media, etc. These data sets are in different forms and massive in quantity.

Extracting valuable insights from these data sets requires using modern tools and technologies to handle the data’s scale and complexity. With the right tools, organizations can gain a deeper understanding of their operations and make data-driven decisions to improve their performance.

In this Project, we will work on a case study, and the data we will use is from the retail industry. The tool we use for this case study is PySpark and Databricks. To work on this case study, I am using the [Databrick](https://www.analyticsvidhya.com/blog/2022/12/case-study-restaurants-insights-using-pyspark-databricks/)s Community Edition.

**Retail Data :**

Retail data is information that a retail shop owner might collect to better their firm. This information gives merchants information about their customers, sales patterns, and inventories across the retail industry.



Retail data plays a crucial role in the decision-making process of retailers. By collecting and analyzing various forms of data, retailers can gain valuable insights into their [business performance](https://www.analyticsvidhya.com/blog/2022/11/data-lineage-case-studies-of-data-driven-businesses/) and make informed decisions to improve their bottom line. By using the retail data, we will see how a retail company can improve its operations and increase its profits with the help of data analysis. We are going to find the hidden insight from this data.

**Pyspark :**

PySpark is a Python API to support Python with Apache Spark. PySpark provides **Py4j library,** with the help of this library, Python can be easily integrated with Apache Spark. PySpark plays an essential role when it needs to work with a vast dataset or analyze them. This feature of PySpark makes it a very demanding tool among data engineers.



A large amount of data is generated offline and online. It is necessary to extract valuable information from the raw data.We require a more efficient tool to perform different types of operations on the big data. There are various tools to perform the multiple tasks on the huge dataset but these tools are not so appealing anymore. It is needed some scalable and flexible tools to crack big data and gain benefit from it.

**Business Requirements**

**Description:**   
Build a retail sales analytics platform to analyze sales data, customer behavior, and product trends.

**Features:**

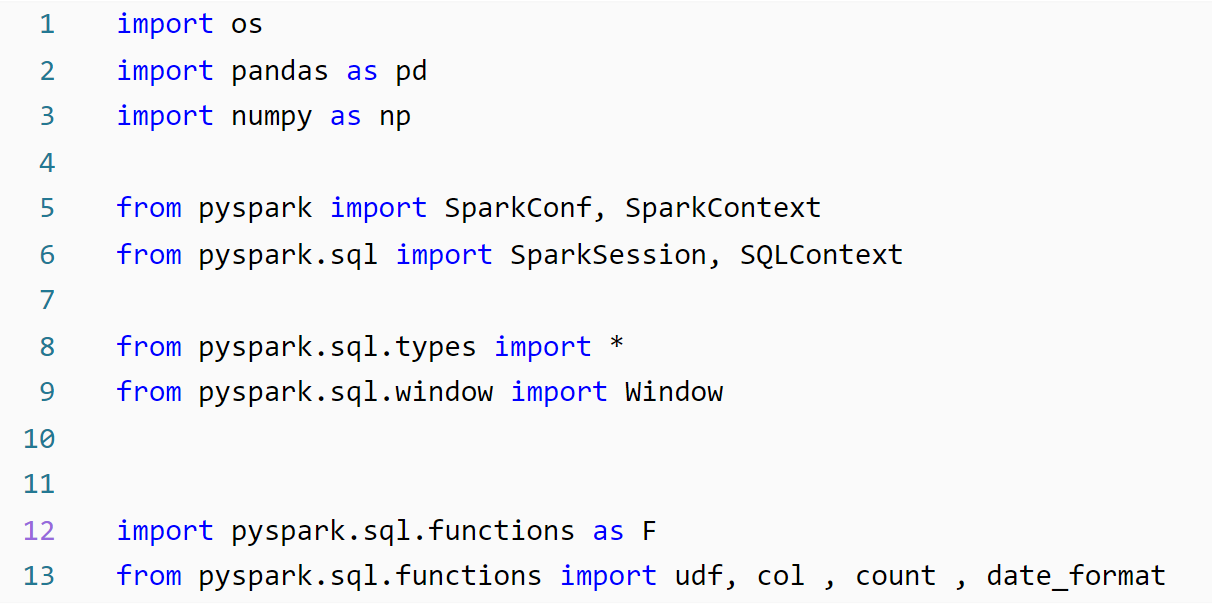
**Data Ingestion**: Ingest sales data from various retail channels, POS systems, or databases.  
**Data Transformation**: Clean, transform, and aggregate sales data to calculate metrics like total sales, average order value, and customer lifetime value.  
**Customer Segmentation**: Segment customers based on purchase history, demographics, or behavior.  
**Trend Analysis**: Identify product trends, seasonal patterns, and sales trends over time.

**Challenges:**

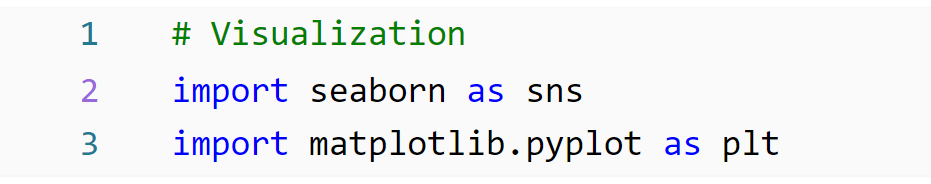
- Handling and joining multiple data sources.  
- Efficiently aggregating and summarizing large volumes of sales data.  
- Performing complex analytical queries to derive meaningful insights.

**Implementation**

We are taking some random data and we will create a dataframe from the provided dataset. We have six datasets with customer details, product order details, product details, and product category information.



**For Visualisation :**



We are going to create six data frames. Which contains the following information:-

1.**Customer Dataframe:** This dataframe contains information related to the customer. It has nine columns which are as follows:-

* **customer\_id**: This column contains the id of the customer. Ex:- 1, 2, 3, etc.
* **customer\_fname**: This column has the customer’s first name details.
* **customer\_lname**: This column has the customer’s last name details.
* **customer\_email**: This column includes the customer’s email info.
* **customer\_password**: This column has customer password information. It’s encrypted.
* **customer\_street**: This has customer address-related info, which is street in this case.
* **customer\_city**: This has city-related information.
* **customer\_state**: The state info of the customer.
* **customer\_zipcode**: The zip code of the customer location.

Now we will create the schema for the customer dataframe.

# define the schema for customer dataset

customers\_1\_schema=StructType(fields=[StructField("Customer\_id",IntegerType(),nullable=False),

                        StructField("First\_Name",StringType(),nullable=True),

                        StructField("Last\_Name",StringType(),nullable= True),

                        StructField("Customer\_email",StringType(),nullable= True),

                        StructField("Customer\_password",StringType(),nullable= True),

                        StructField("Street",StringType(),nullable= True),

                        StructField("City",StringType(),nullable= True),

                        StructField("State",StringType(),nullable= True),

                        StructField("Zipcode",IntegerType(),nullable= True),

                               ])

We will make the dataframe using the above schema.

# Reading the data through a CSV file.

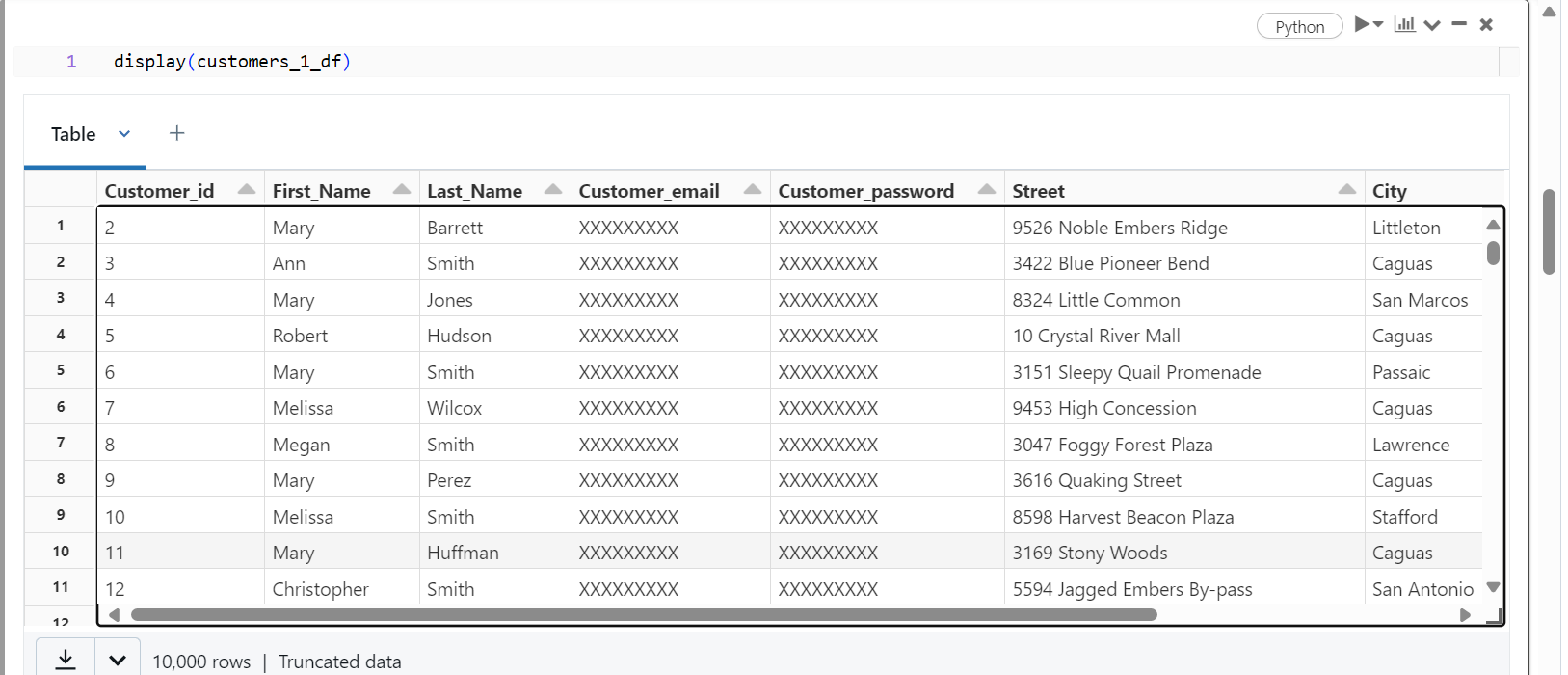
customers\_1\_df = spark.read.csv\

("dbfs:/FileStore/tables/customers\_1.csv",header=True,schema=customers\_1\_schema)

To show the data frame the command is

display(customers\_1\_df)

The customer dataframe will look like this:



Now we will create a Product dataframe.

2**. Product dataframe**: The product information contained in this dataframe. The six columns within it are as follows:-

* **product\_id:**This column contains the product ids.
* **product\_category\_id:**This column help in finding the category of the product.
* **product\_name**:- This column includes product names that we have in store.
* **product\_description**: It consists of the product details.
* **product\_price**: The product price is in this column.
* **product\_image**: Product image URLs are present in this column.

First, we will create the schema for this dataframe. Which will look like this:-

products\_schema = StructType(fields=[StructField('product\_id',IntegerType(), nullable=False),

    StructField('product\_category\_id', IntegerType(), nullable=False),

    StructField('product\_name',StringType(), nullable=False),

    StructField('product\_description',StringType(), nullable=False),

    StructField('product\_price',FloatType(), nullable=False),

    StructField('product\_image',StringType(), nullable=False)

    ])

We will make the dataframe using the above schema.

# Reading the data through a CSV file.

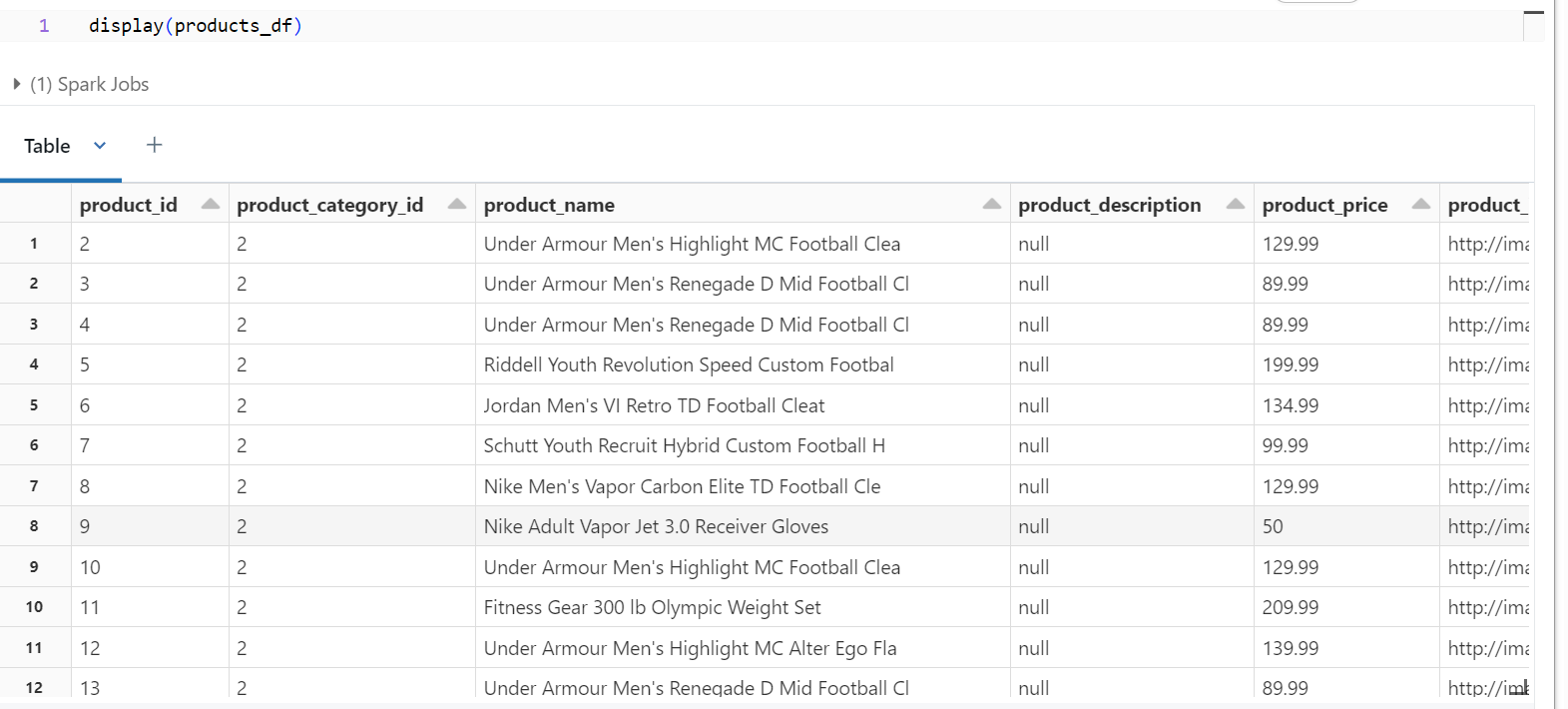
products\_df = spark.read.csv\

("dbfs:/FileStore/tables/products\_2.csv",header=True,schema=products\_schema)

To show the data frame the command is

display(products\_df)

The product dataframe will look like this:-

****

**3. Categories DataFrame:**This dataframe has a list of product categories. It has three columns product\_id, product\_category\_id, and category\_name.

* **category\_name:**The categories to which a product may belong are in this column. The product name**“Under Armour Men’s Highlight MC Football Clean”** with product\_id and category\_id two will belong to **the “Soccer”**category.

First, we will create the schema for this dataframe. Which will look like this:-

categories\_schema = StructType(fields=[StructField('category\_id',IntegerType(), nullable=True),

    StructField('category\_department\_id', IntegerType(), nullable=True),

    StructField('category\_name',StringType(), nullable=True)])

We will make the dataframe using the above schema.

# Reading the data through a CSV file.

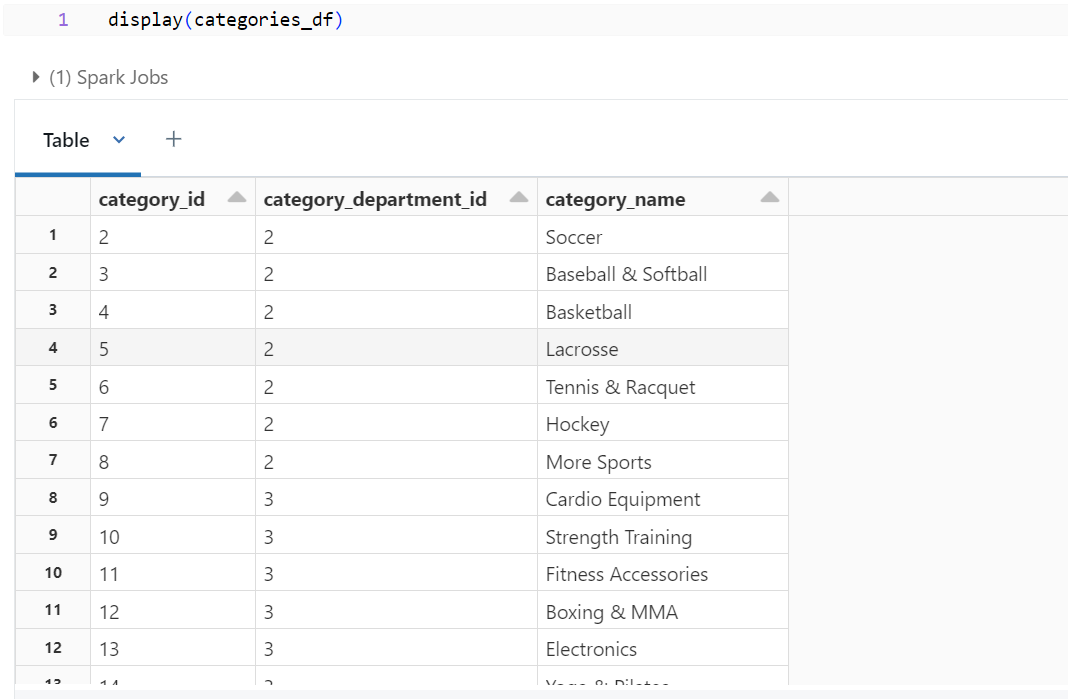
categories\_df = spark.read.csv\

("dbfs:/FileStore/tables/categories\_1.csv",header=True,schema=categories\_schema)

To show the data frame the command is

display(categories\_df)

The product dataframe will look like this:-



**4.** **Orders Dataframe:** In this dataframe, we have details related to item orders and their payment status. It has four columns which are as follows:-

* **order\_id:**It has the Ids of the ordered item.
* **order\_date**: The date and time values are included in this column.
* **order\_customer\_id**: The customer order Ids is contained in this column.
* **order\_status**: The payment status details are in this column.

Let’s create a schema and ‘**orders\_df**‘ dataframe using this dataset.

# define the schema for orders

orders\_schema = StructType(fields=[

    StructField('order\_id',IntegerType(), nullable=True),

    StructField('order\_date',StringType(), nullable=True),

    StructField('order\_customer\_id',IntegerType(), nullable=True),

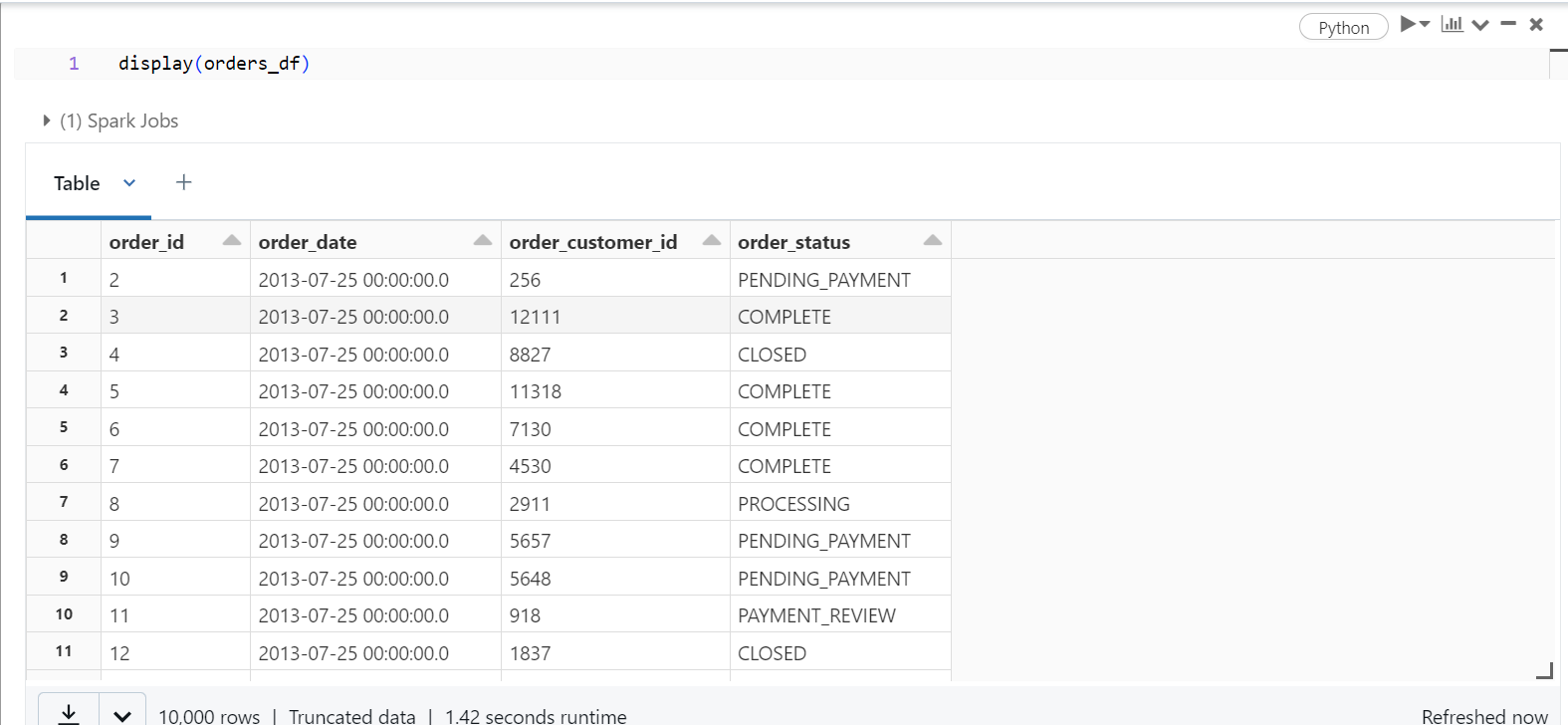
    StructField('order\_status',StringType(), nullable=True)])

# Reading the data through a CSV file.

orders\_df = spark.read.csv\

("dbfs:/FileStore/tables/orders\_1.csv",header=True,schema=orders\_schema)

The ‘**orders\_df’** will look like this:-



**5. Departments Dataframe**: In this dataframe, we have department details. It has two columns which are as follows:-

* **department\_id**: It has the ID’s information.
* **department\_name**: It has a list of department names. Ex:- Footwear, Apparel, Golf, Outdoors, etc.

Now we will create a schema and dataframe for the department dataset.

departments\_schema = StructType(fields=[

    StructField('department\_id',   IntegerType(), nullable=True),

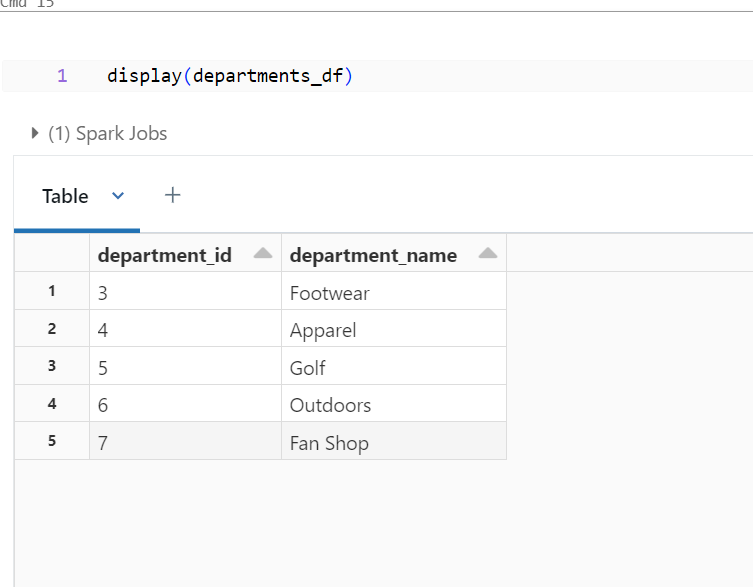
    StructField('department\_name', StringType(), nullable=True)])

# Reading the data through a CSV file.

departments\_df = spark.read.csv\

("dbfs:/FileStore/tables/departments\_1.csv",header=True,schema=departments\_schema)

The sample data will look like this:-



**6. Order Items Dataframe:** A collection of information about items ordered on an e-commerce platform or retail store is present in the order items dataframe. It comprises several columns providing details about each order item. The columns in the dataframe are as follows:-

* “**order\_item\_id**“: This column contains a unique identifier for each order item.
* “**order\_item\_order\_id**“: This column contains the unique identifier of the order that the ordered item belongs.
* **“order\_item\_product\_id”**: The unique identifier of the ordered products is stored in this column.
* **“order\_item\_quantity”:**The ordered product quantity is recorded in this column.
* **“order\_item\_subtotal**“: This column contains the total cost of the ordered item, calculated by multiplying the quantity by the product price.
* “**order\_item\_product\_price**“: The cost of each product is recorded in this column.

Now we will create the schema and dataframe from the dataset.

# define the schema for order items

order\_items\_schema = StructType(fields=[

    StructField('order\_item\_id',            IntegerType(), nullable=True),

    StructField('order\_item\_order\_id',      IntegerType(), nullable=True),

    StructField('order\_item\_product\_id',    IntegerType(), nullable=True),

    StructField('order\_item\_quantity',      IntegerType(), nullable=True),

    StructField('order\_item\_subtotal',      FloatType(), nullable=True),

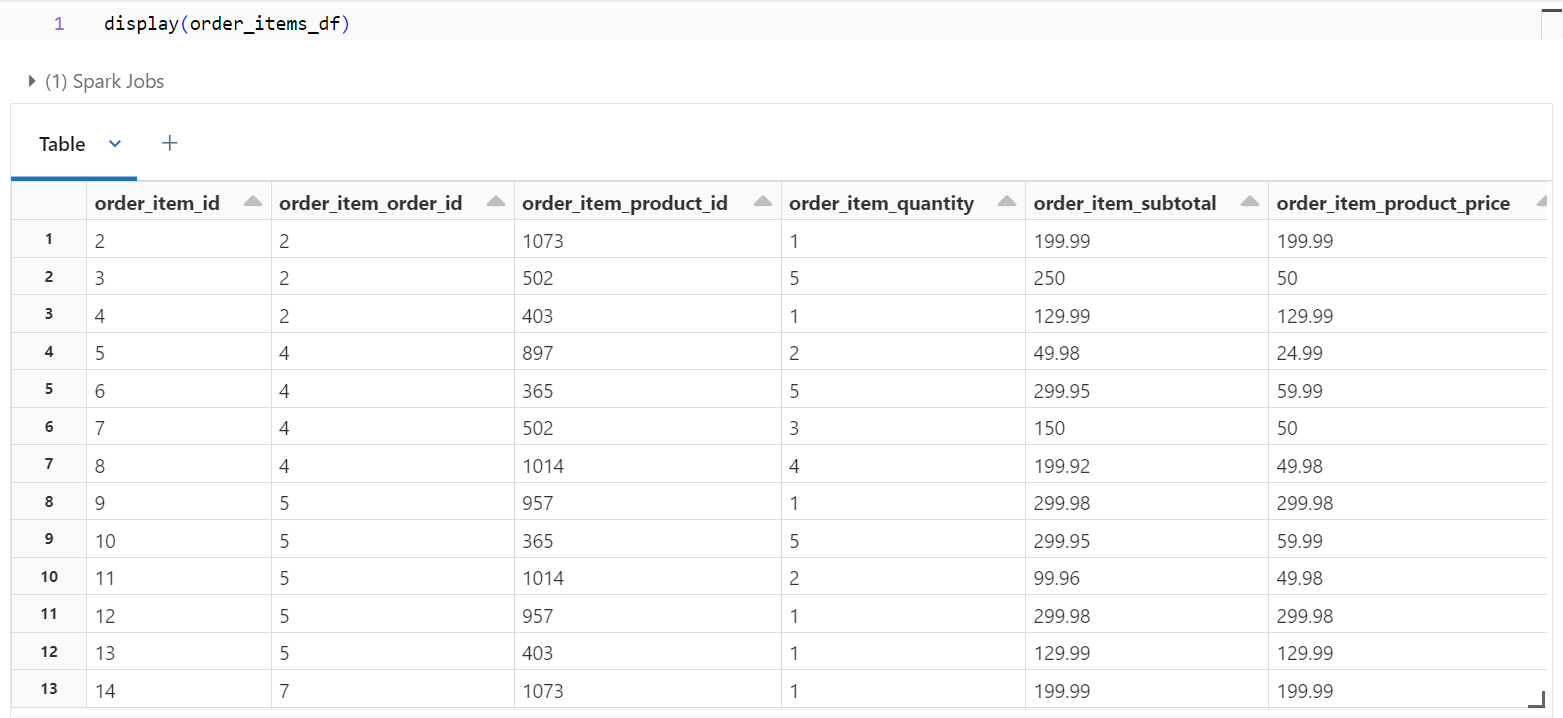
    StructField('order\_item\_product\_price', FloatType(), nullable=True)])

# Reading the data through a CSV file.

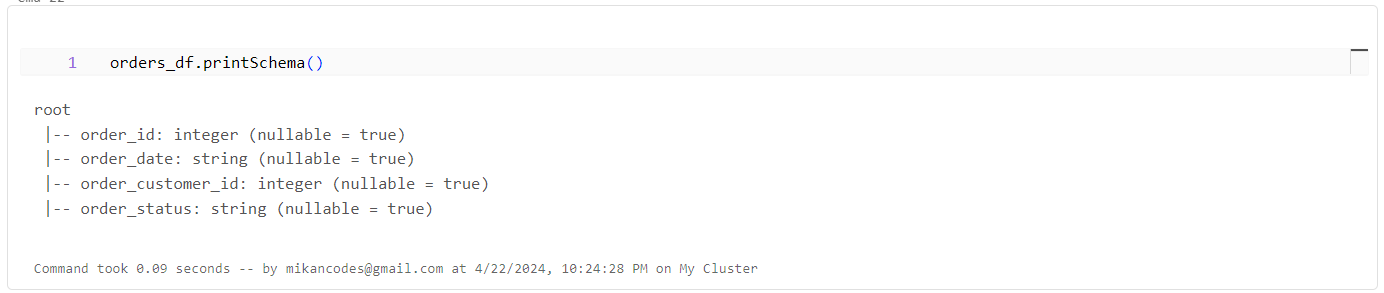
order\_items\_df = spark.read.csv\

("dbfs:/FileStore/tables/order\_items\_1.csv",header=True,schema=order\_items\_schema)

The dataframe will look like this:

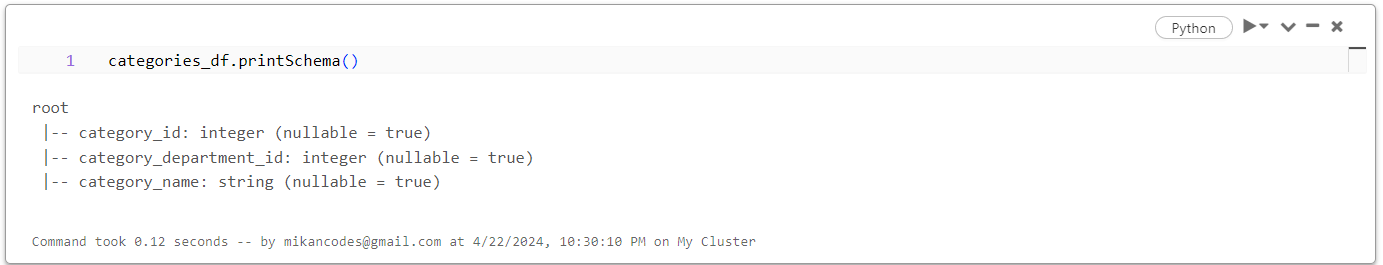


**Schemas :**













**Creating Temporary Views:**

* We can create temporary view for a Data Frame using createTempView or createOrReplaceTempView.
* createOrReplaceTempView will replace existing view, if it already exists.
* While tables in Metastore are permanent, views are temporary.
* Once the application exits, temporary views will be deleted or flushed out.

A white background with green text

Description automatically generated

**Checking Null Values:**

A screenshot of a computer

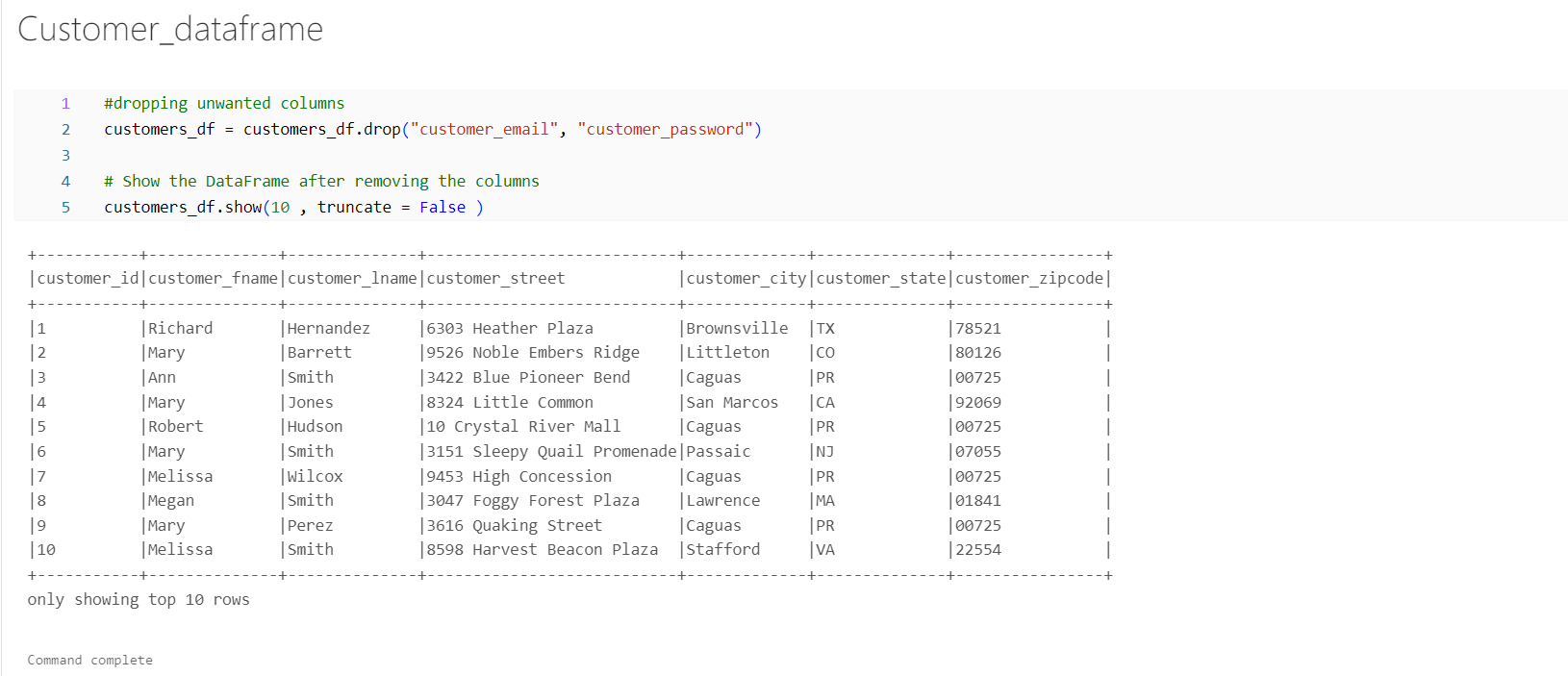
Description automatically generated

**Data Cleaning :**

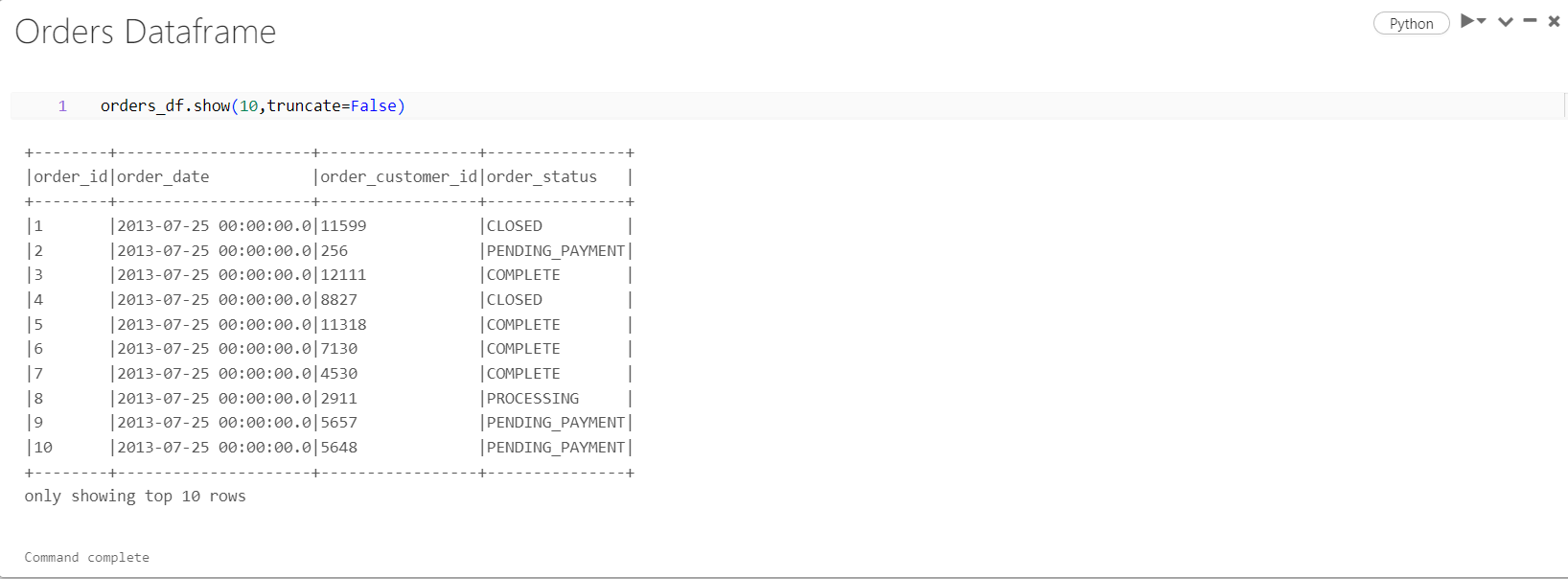
A screenshot of a computer

Description automatically generated

* In customer data frame, customer\_email & customer\_password are unnecessary for us to deal with data analysis and also data in corrupted form. So, we are removing those 2 columns.

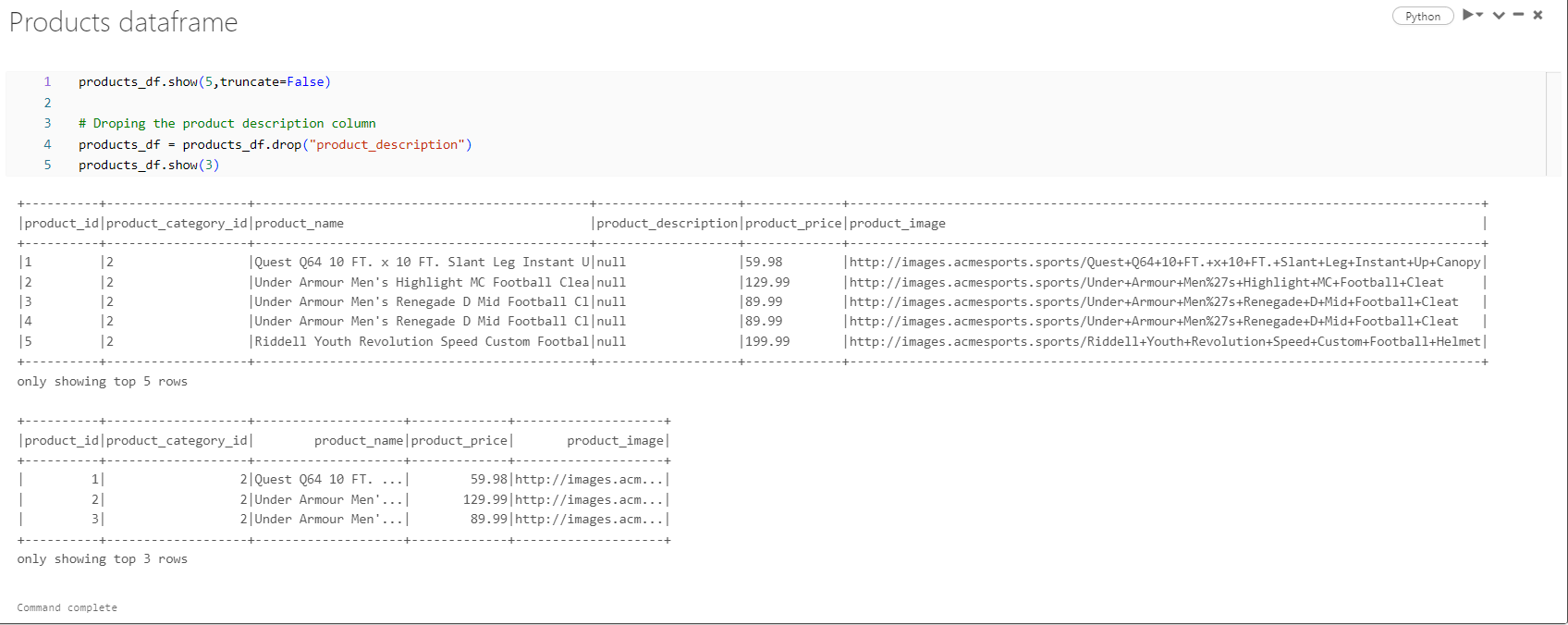


* In orders Data Frame order\_date is in timestamp format we are converting it into date format(yyyy-mm-dd).



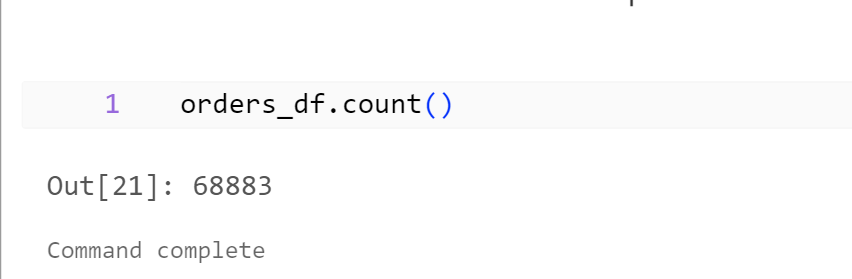


* Products Data Frame – Dropping the product description column



**DATA ANALYSIS & VISUALISATION**

1.Total Number of Orders placed :



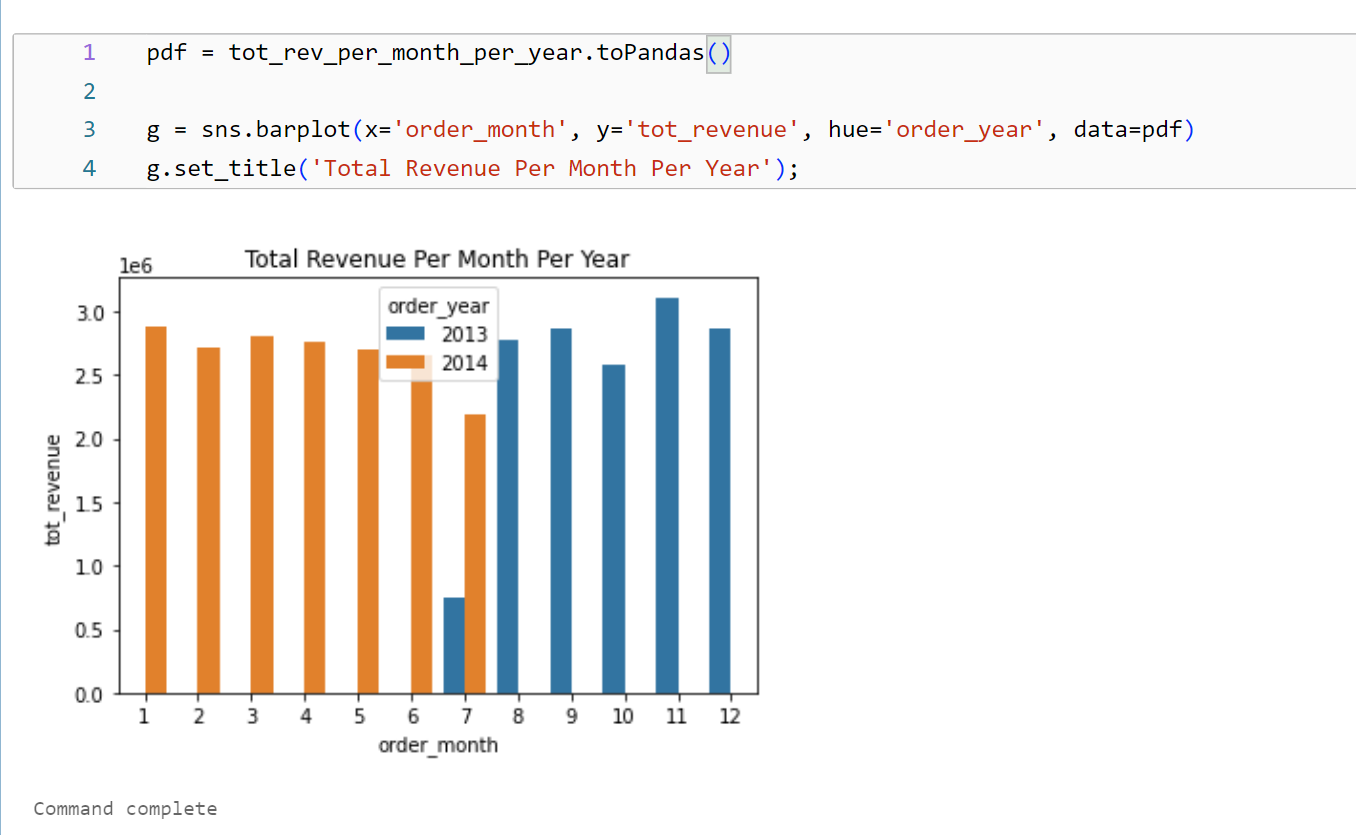
2.Total Revenue for each year :



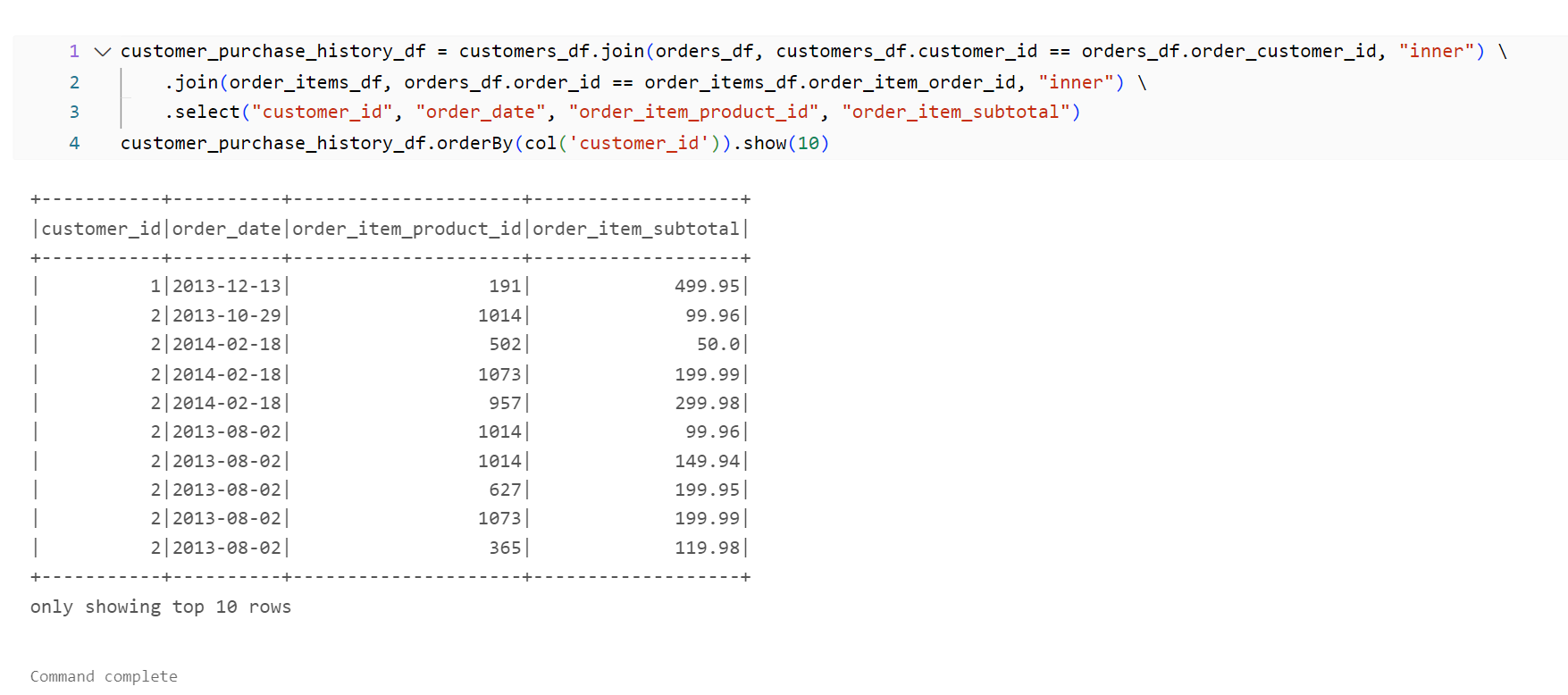
3. Total revenue for each month:



Plot For Total Revenue Per Month-

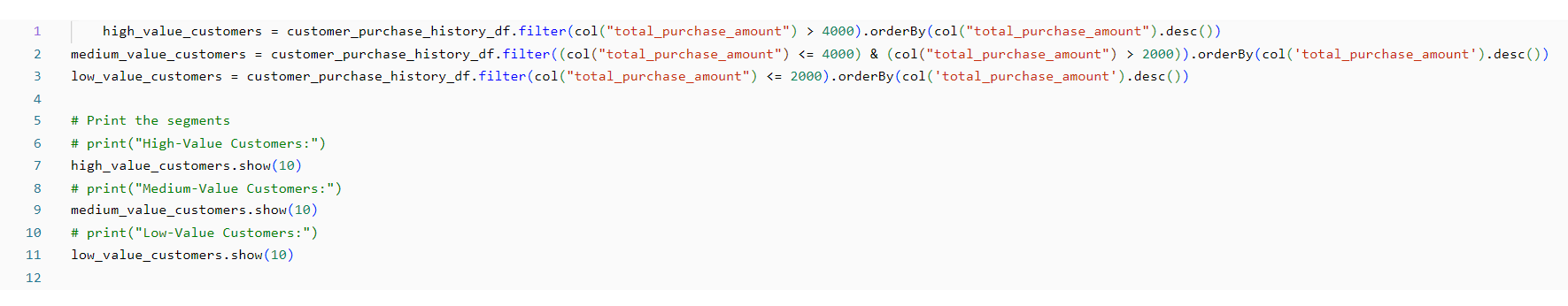


4.Total purchase by each customer :





5. Customer type according to their purchase :



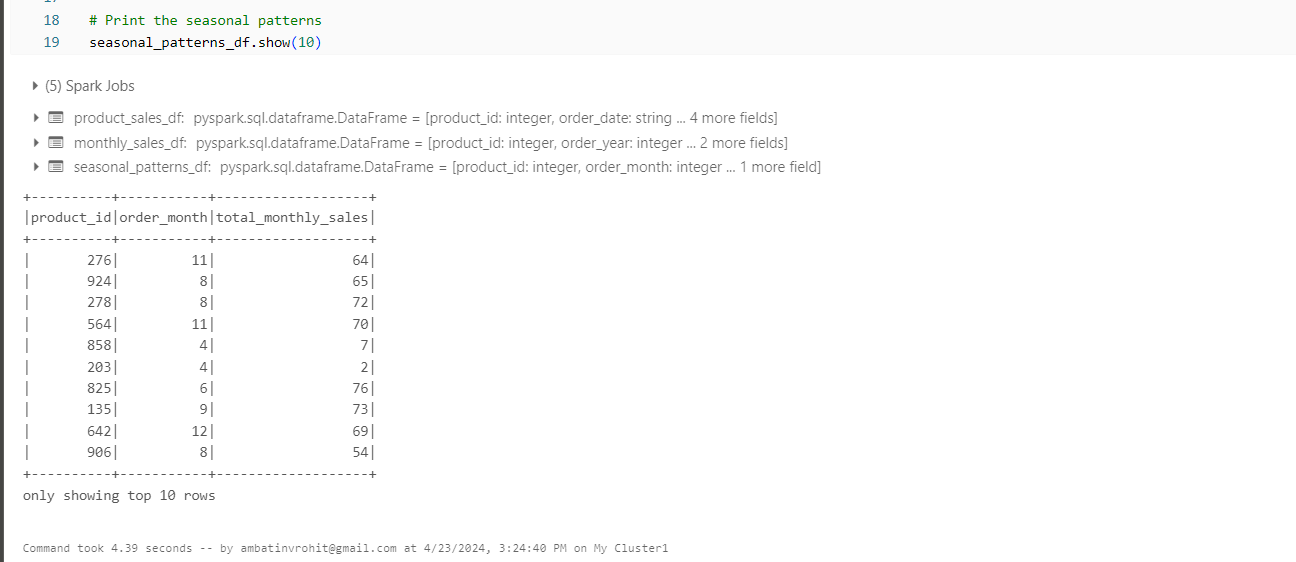






6.Monthly sales product trends :

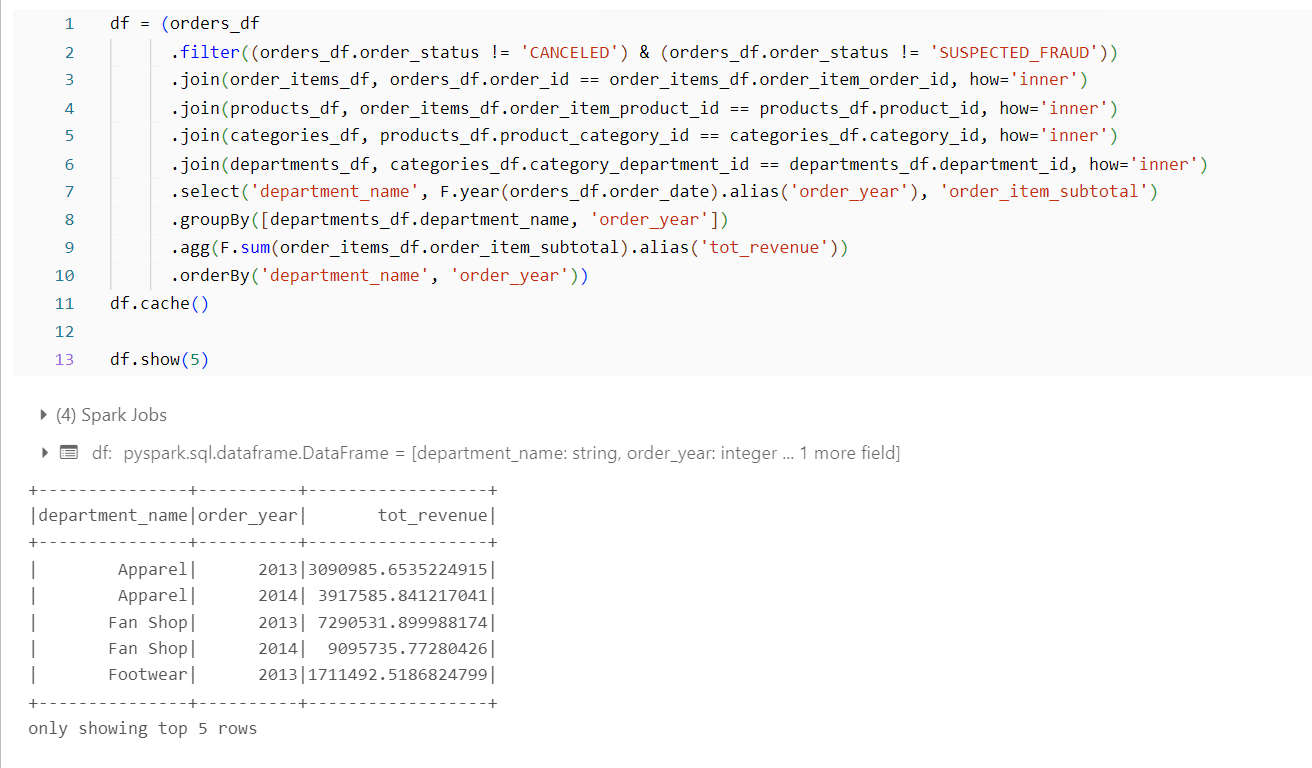




Plot-



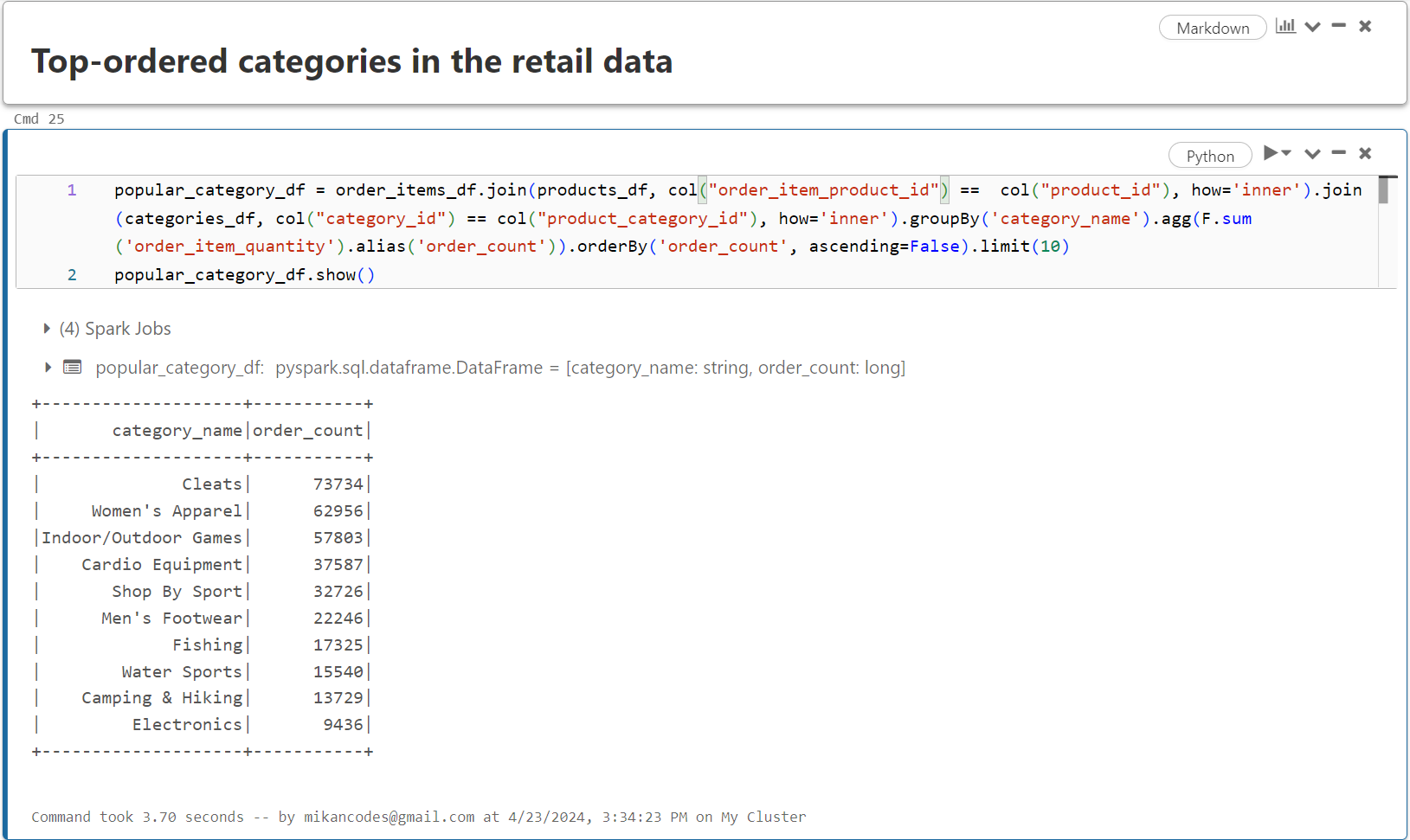
7.Top performing departments :



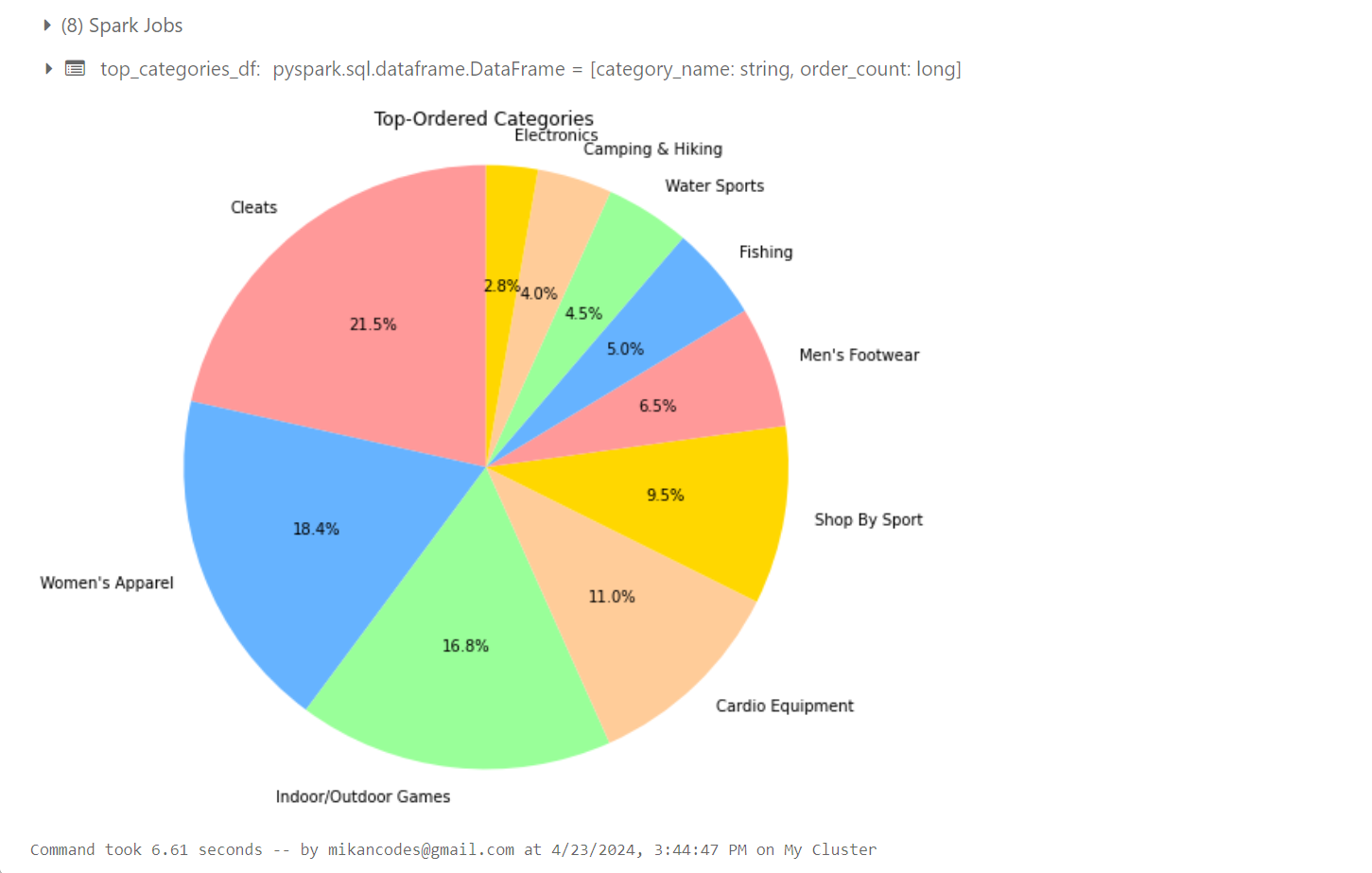
8. Highest priced product :



9.Popular Category :



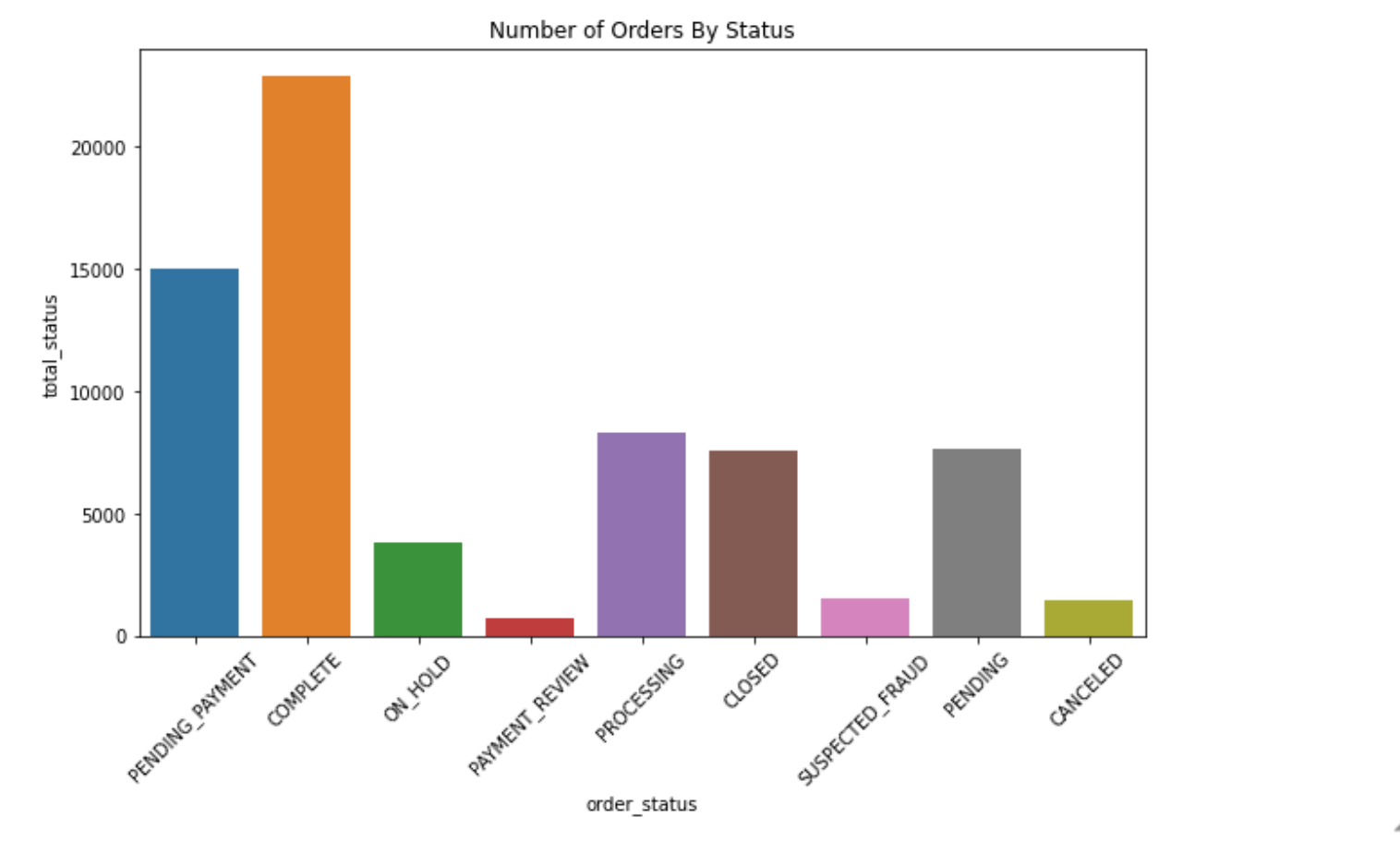




10. Count of Orders based on their Status :







**Conclusion**

We solve a case study utilizing retail data to uncover hidden insights that can increase sales using PySpark. This analysis allows the store manager to understand which products require more attention. The insights gained from the data can assist in tracking individual product performance, identifying top-selling items, and adjusting inventory levels accordingly. Retailers can also use sales data to analyze customer purchasing patterns and make informed decisions about product placement, promotions, and pricing strategies.

1. Working with real-world data to gather valuable information can benefit businesses in various ways.
2. We have also seen the detailed step-by-step solutions to the problem using PySpark functions and analysis of the output at the end of each problem statement.