

# **Data Analysis with PySpark: Exploring Trends and Insights in Retail Sales**

Section Number: 44517-03

Team Name: DataXplorers

Team Members:

1. Venkatalahari Ravipati
2. Chiranjeevi Ankaiah
3. Vikas Allam

## **Abstract**

This project involves analyzing sales data using PySpark. The primary objectives are to clean and preprocess the data, gain insights into sales performance by region and product, analyze sales trends over time, visualize key findings, and evaluate profitability for strategic decision-making.

# 1. Implementation Steps

## 1.1. Goal 1: Data Cleaning and Preparation

1. **Load Data:** The dataset is loaded into a PySpark DataFrame using the `spark.read.csv` function.

The screenshot shows a Jupyter Notebook titled 'Untitled4.ipynb' running on a local host. The code in the cell is as follows:

```
[6]: from pyspark.sql import SparkSession
spark = SparkSession.builder.appName("retailstore").getOrCreate()
df = spark.read.option("header", "true").csv("retail_store_inventory.csv")
df.show(5)
```

The output of the code is a table showing the first 5 rows of the 'retail\_store\_inventory.csv' file. The table has 14 columns: Date, Store ID, Product ID, Category, Region, Inventory Level, Units Sold, Units Ordered, Demand Forecast, Price, Discount, Weather Condition, Holiday/Promotion, Competitor Pricing, and Seasonality. The output is truncated to show only the top 5 rows.

Date	Store ID	Product ID	Category	Region	Inventory Level	Units Sold	Units Ordered	Demand Forecast	Price	Discount	Weather Condition	Holiday/Promotion	Competitor Pricing	Seasonality
2022-01-01	S001	P0001	Groceries	North	231	127	55							
2022-01-01	S001	P0002	Toys	South	204	150	66							
2022-01-01	S001	P0003	Toys	West	102	65	51							
2022-01-01	S001	P0004	Toys	North	469	61	164							

## 2. Handle Missing Values: Rows with missing values are either dropped or filled with default values.

The screenshot shows a Jupyter Notebook titled 'Untitled4.ipynb' running on a local host. The notebook contains the following code:

```
[7]: df_cleaned = df.dropna()
df_filled = df.fillna({"Units Sold": 0, "Units Ordered": 0, "Price": 0.0})
df_filled.show(5)
```

The output of the code is a preview of the first 5 rows of the cleaned dataset. The preview shows a table with the following columns: Date, Store ID, Product ID, Category, Region, Inventory Level, Units Sold, Units Ordered, and Demand. The data is as follows:

Date	Store ID	Product ID	Category	Region	Inventory Level	Units Sold	Units Ordered	Demand
2022-01-01	S001	P0001	Groceries	North	231	127	55	135.47
2022-01-01	S001	P0002	Rainy	South	0	29.69	Autumn	33.5
2022-01-01	S001	P0002	Sunny	South	204	150	Autumn	63.01
2022-01-01	S001	P0003	Sunny	West	102	65	51	27.99
2022-01-01	S001	P0004	Sunny	North	1	31.32	Summer	32.72

The notebook interface also shows a file explorer on the left with various files and folders, and a status bar at the bottom indicating the current mode and file name.

### 3. Standardize Column Types: Numeric columns are converted to the appropriate data types.

The screenshot shows a Jupyter Notebook titled 'Untitled4.ipynb' running on a Python 3 (ipykernel) environment. The code cell contains the following PySpark SQL functions to cast columns to integer and float types:

```
[8]: from pyspark.sql.functions import col
df_cleaned = df_cleaned.withColumn("Units Sold", col("Units Sold").cast("int")) \
                        .withColumn("Price", col("Price").cast("float"))
df_cleaned.show(5)
```

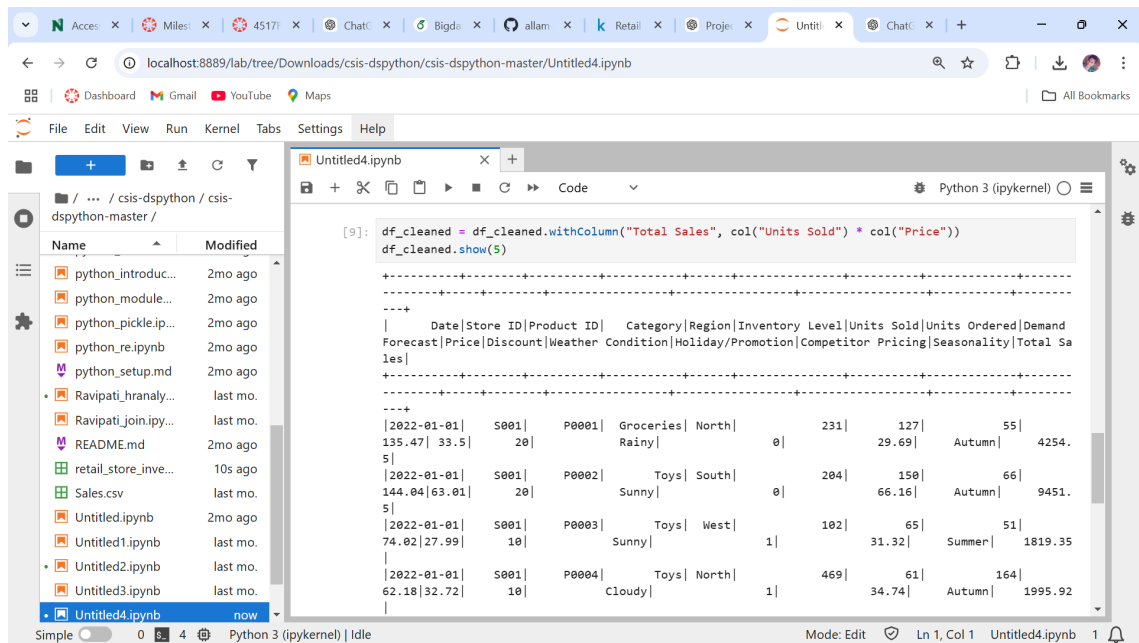
The output displays a table with the following columns: Date, Store ID, Product ID, Category, Region, Inventory Level, Units Sold, Units Ordered, Demand Forecast, Price, Discount, Weather Condition, Holiday/Promotion, Competitor Pricing, and Seasonality. The table shows the top 5 rows of data.

Date	Store ID	Product ID	Category	Region	Inventory Level	Units Sold	Units Ordered	Demand Forecast	Price	Discount	Weather Condition	Holiday/Promotion	Competitor Pricing	Seasonality
2022-01-01	S001	P0001	Groceries	North	231	127	55	135.47	33.5	20	Rainy	0	29.69	Autumn
2022-01-01	S001	P0002	Toys	South	204	150	66	144.04	63.01	20	Sunny	0	66.16	Autumn
2022-01-01	S001	P0003	Toys	West	102	65	51	74.02	27.99	10	Sunny	1	31.32	Summer
2022-01-01	S001	P0004	Toys	North	469	61	164	62.18	32.72	10	Cloudy	1	34.74	Autumn
2022-01-01	S001	P0005	Electronics	East	166	14	135	9.26	73.64	0	Sunny	0	68.95	Summer

only showing top 5 rows

## 1.2. Goal 2: Sales Insights by Region

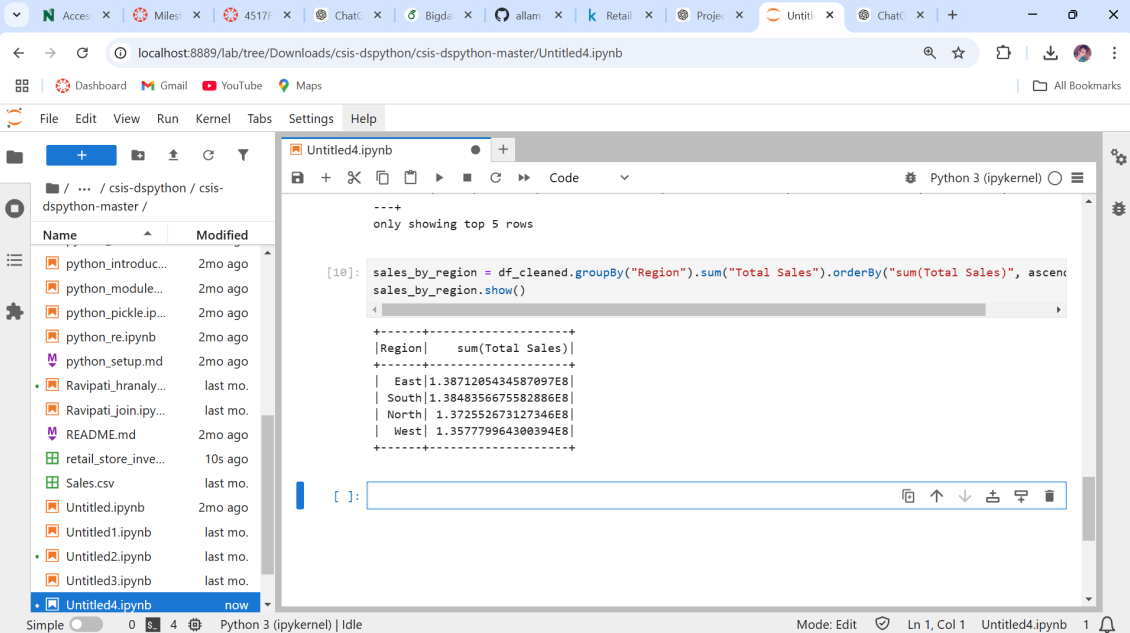
### 1. Calculate total sales by multiplying Units Sold with Price.



```
[9]: df_cleaned = df_cleaned.withColumn("Total Sales", col("Units Sold") * col("Price"))
df_cleaned.show(5)
```

Date	Store ID	Product ID	Category	Region	Inventory Level	Units Sold	Units Ordered	Demand Forecast	Price	Discount	Weather Condition
2022-01-01	S001	P0001	Groceries	North	231	127	55	135.47	33.5	20	Rainy
2022-01-01	S001	P0002	Toys	South	204	150	66	144.04	63.01	20	Sunny
2022-01-01	S001	P0003	Toys	West	102	65	51	74.02	27.99	10	Sunny
2022-01-01	S001	P0004	Toys	North	469	61	164	62.18	32.72	10	Cloudy

## 2. Group by Region and calculate total sales.



The screenshot shows a Jupyter Notebook interface with a file explorer on the left and a code cell in the center. The file explorer lists files in the directory `/.../csis-dspython/csis-dspython-master/`. The code cell contains the following code:

```
only showing top 5 rows

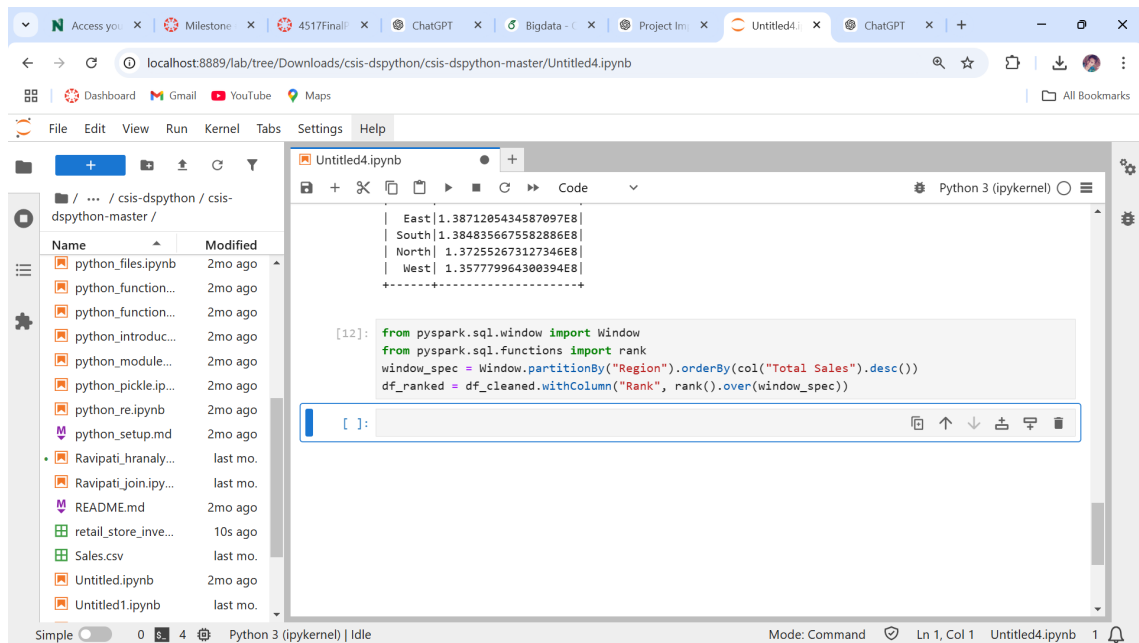
[10]: sales_by_region = df_cleaned.groupby("Region").sum("Total Sales").orderBy("sum(Total Sales)", ascending=False)
      sales_by_region.show()
```

The output of the code is a table showing the sum of total sales for each region, ordered by the sum of total sales in descending order:

Region	sum(Total Sales)
East	1.3871205434587097E8
South	1.3848356675582886E8
North	1.372552673127346E8
West	1.357779964300394E8

### 1.3. Goal 3: Top Products by Region

1. Use a window function to rank products by sales within each region.



The screenshot shows a Jupyter Notebook interface with a file explorer on the left and a code editor on the right. The file explorer displays a list of files and folders, including 'python\_files.ipynb', 'python\_function...', 'python\_introduc...', 'python\_module...', 'python\_pickle.ip...', 'python\_re.ipynb', 'python\_setup.md', 'Ravipati\_hranaly...', 'Ravipati\_join.ip...', 'README.md', 'retail\_store\_inve...', 'Sales.csv', 'Untitled.ipynb', and 'Untitled1.ipynb'. The code editor shows a table of product sales by region and a PySpark window function query to rank products by total sales within each region.

Region	Product ID	Total Sales
East	1.3871205434587097E8	
South	1.3848356675582886E8	
North	1.372552673127346E8	
West	1.357779964308394E8	

```
[12]: from pyspark.sql.window import Window
      from pyspark.sql.functions import rank
      window_spec = Window.partitionBy("Region").orderBy(col("Total Sales").desc())
      df_ranked = df_cleaned.withColumn("Rank", rank().over(window_spec))
```

## 2. Filter the top 5 products for each region.

The screenshot shows a Jupyter Notebook running on a local host. The notebook contains a PySpark query that uses a window function to rank products by total sales within each region and then filters for the top 5 products. The output of the query is displayed as a table.

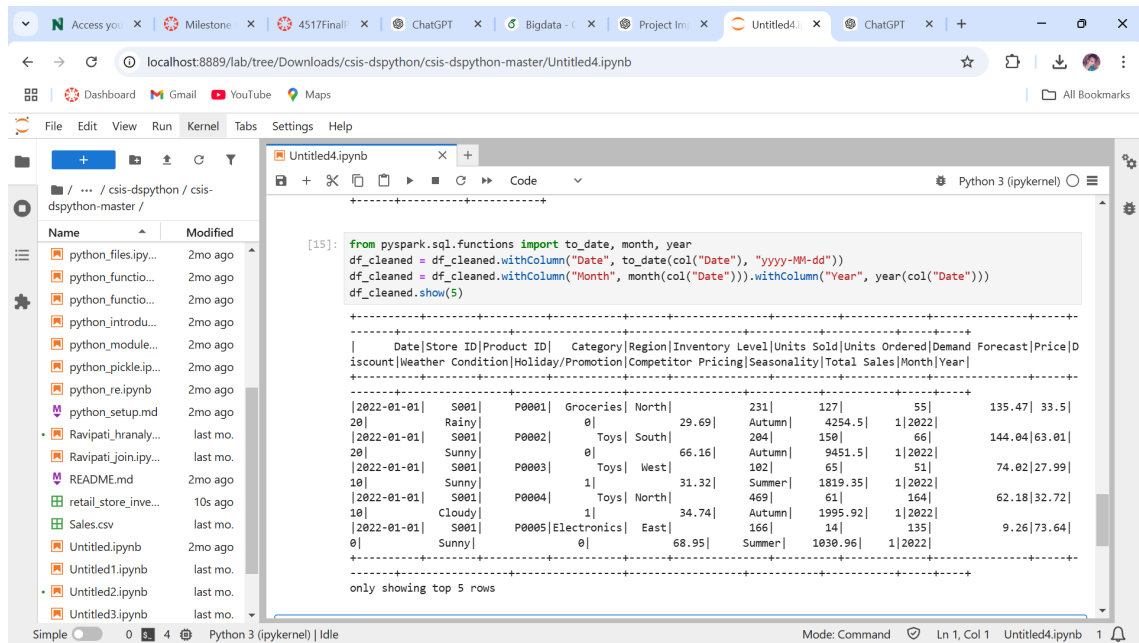
```
[11]: from pyspark.sql.window import Window
      from pyspark.sql.functions import rank
      window_spec = Window.partitionBy("Region").orderBy(col("Total Sales").desc())
      df_ranked = df_cleaned.withColumn("Rank", rank().over(window_spec))
      top_5_products = df_ranked.filter(col("Rank") <= 5).select("Region", "Product ID", "Total Sales")
      top_5_products.show()
```

Region	Product ID	Total Sales
East	P0002	47860.47
East	P0013	45584.12
East	P0010	45361.797
East	P0005	45212.88
East	P0018	45054.902
North	P0009	45575.88
North	P0007	45077.68
North	P0001	45023.473
North	P0010	45018.8
North	P0011	44664.16
South	P0018	47141.6
South	P0010	45421.2
South	P0015	44500.5
South	P0004	43719.94
South	P0017	43612.6
West	P0020	46810.54
West	P0013	46792.13
West	P0004	45929.43
West	P0003	45364.902
West	P0010	44195.2



## 1.4. Goal 4: Sales Trend Analysis

### 1. Extract Month and Year from the Date column.

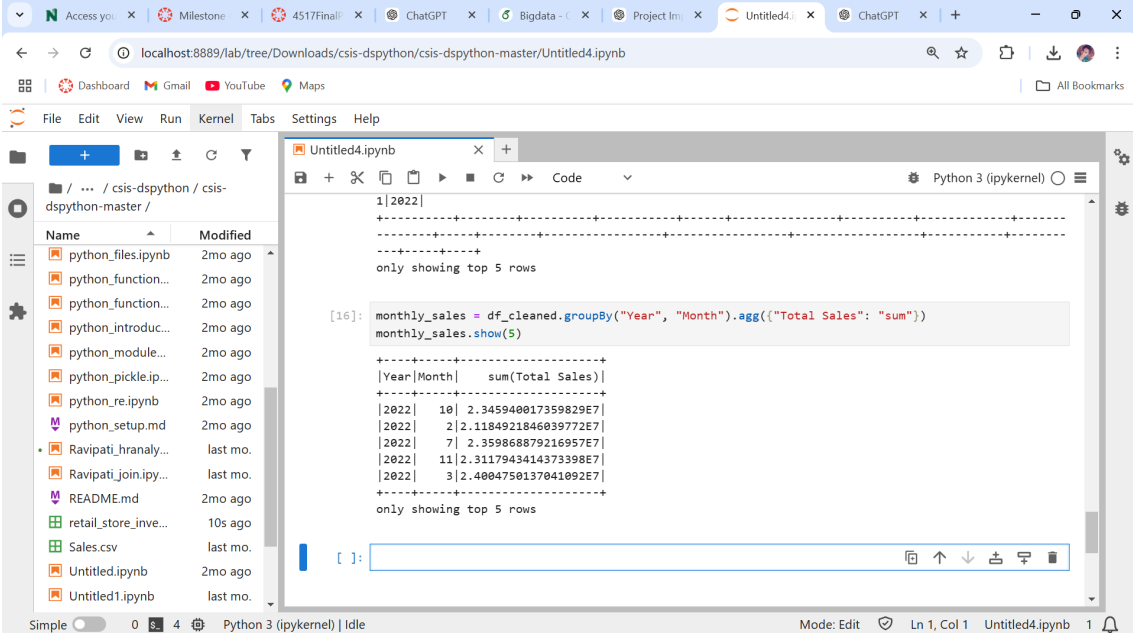


```
[15]: from pyspark.sql.functions import to_date, month, year
df_cleaned = df_cleaned.withColumn("Date", to_date(col("Date"), "yyyy-MM-dd"))
df_cleaned = df_cleaned.withColumn("Month", month(col("Date"))).withColumn("Year", year(col("Date")))
df_cleaned.show(5)
```

Date	Store ID	Product ID	Category	Region	Inventory Level	Units Sold	Units Ordered	Demand Forecast	Price	Discount	Weather Condition	Holiday/Promotion	Competitor Pricing	Seasonality	Total Sales	Month	Year
2022-01-01	S001	P0001	Groceries	North	231	127	55	135.47	33.5								
2022-01-01	S001	P0002	Toys	South	29.69	4254.5	1	144.04	63.01								
2022-01-01	S001	P0003	Toys	West	66.16	9451.5	1	74.02	27.99								
2022-01-01	S001	P0004	Toys	North	31.32	1819.35	1	62.18	32.72								
2022-01-01	S001	P0005	Electronics	East	34.74	1995.92	1	9.26	73.64								

only showing top 5 rows

## 2. Group data by Month and Year and calculate total sales.



The screenshot shows a Jupyter Notebook interface with a file explorer on the left and a code editor on the right. The file explorer shows a directory structure with files like `python_files.ipynb`, `python_function...`, `python_intro...`, `python_module...`, `python_pickle.ip...`, `python_re.ipynb`, `python_setup.md`, `Ravipati_hranaly...`, `Ravipati_join.ipy...`, `README.md`, `retail_store_inve...`, `Sales.csv`, `Untitled.ipynb`, and `Untitled1.ipynb`. The code editor shows the following code:

```
1|2022|
+-----+
only showing top 5 rows

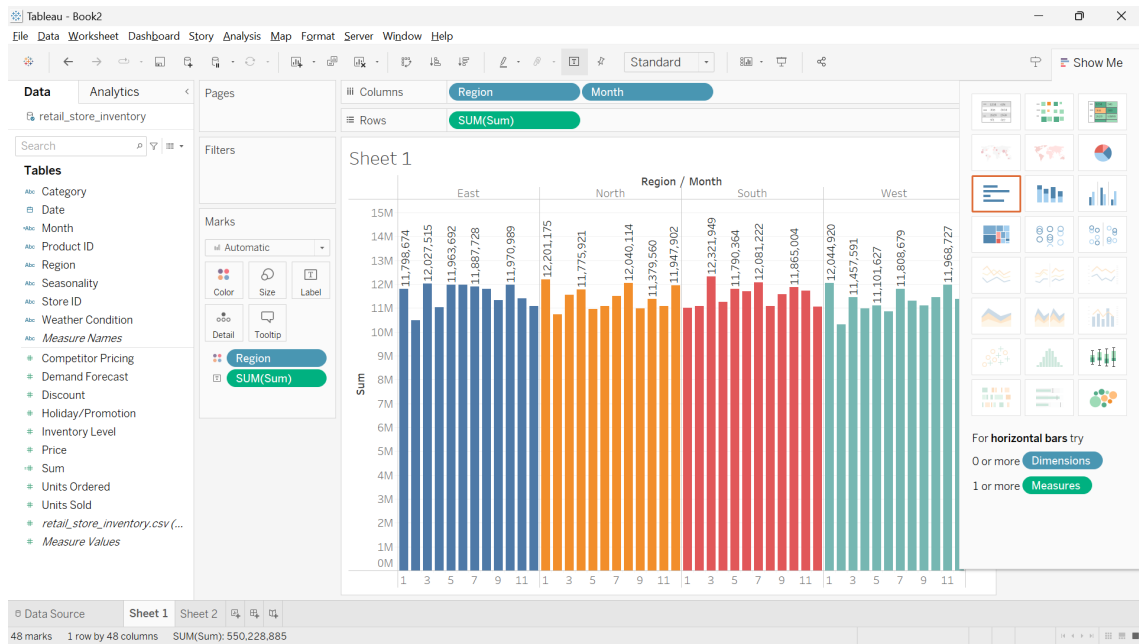
[16]: monthly_sales = df_cleaned.groupby("Year", "Month").agg({"Total Sales": "sum"})
      monthly_sales.show(5)

+-----+
|Year|Month|    sum(Total Sales)|
+-----+
|2022|  10| 2.345940017359829E7|
|2022|   2| 2.1184921846039772E7|
|2022|   7| 2.359868879216957E7|
|2022|  11| 2.3117943414373398E7|
|2022|   3| 2.4004750137041092E7|
+-----+
only showing top 5 rows
```

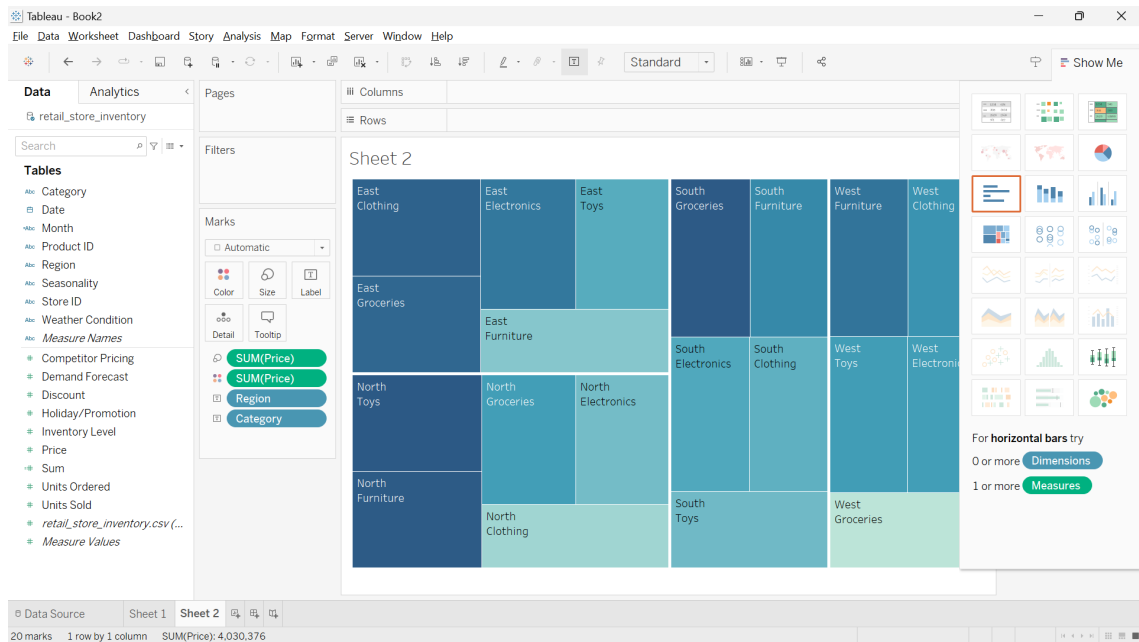
The status bar at the bottom indicates "Simple", "0", "4", "Python 3 (ipykernel) | Idle", "Mode: Edit", "Ln 1, Col 1", "Untitled4.ipynb", and "1".

## 1.5. Goal 5: Visualization of Insights

1. Convert the PySpark DataFrame to Pandas and use Matplotlib for visualizations.
2. Calculating total sales of each region in every month.



### 3. Calculating product price in each region.



## 1.6. Goal 6: Profitability Analysis

### 1. Calculate Profit based on Discounted Price.

localhost:8890/lab/tree/Downloads/csis-dspython/csis-dspython-master/Untitled4.ipynb

File Edit View Run Kernel Tabs Settings Help

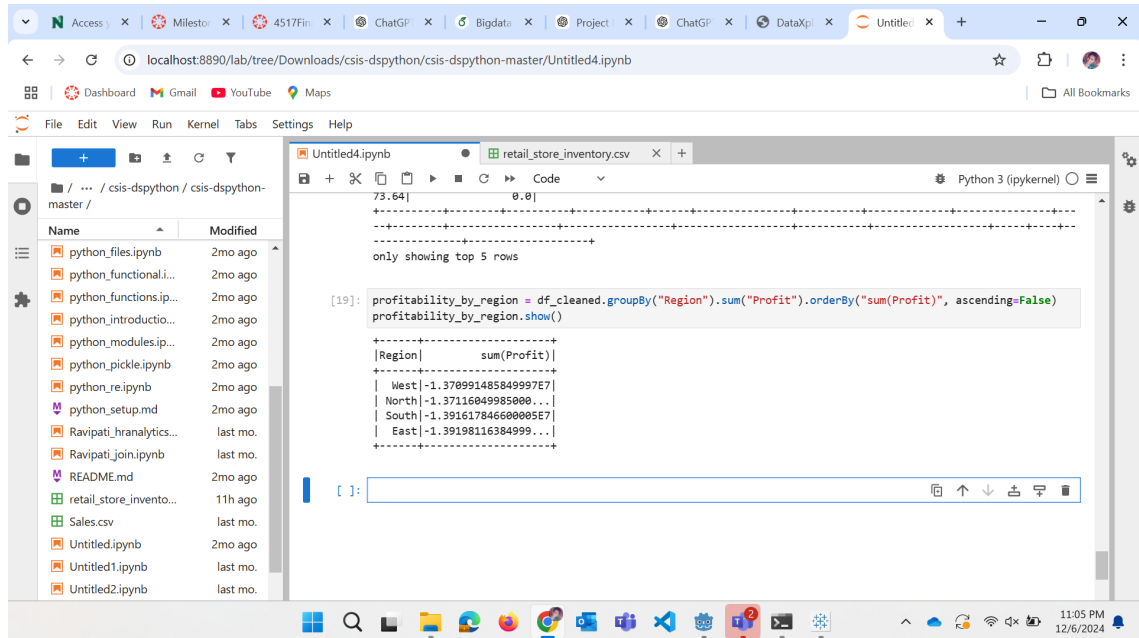
Untitled4.ipynb

```
[18]: df_cleaned = df_cleaned.withColumn("Discounted Price", col("Price") * (1 - col("Discount") / 100))
df_cleaned = df_cleaned.withColumn("Profit", (col("Discounted Price") - col("Price")) * col("Units Sold"))
df_cleaned.show(5)
```

Date	Store ID	Product ID	Category	Region	Inventory Level	Units Sold	Units Ordered	Demand Forecast	Price	Discount	Weather
2022-01-01	S001	P0001	Groceries	North	231	127	55	135.47	33.5	20	
Rainy				Autumn	4254.5	1	2022	26.8	-850.8999999999999		
2022-01-01	S001	P0002	Toys	South	204	150	66	144.04	63.01	20	
Sunny				Autumn	9451.5	1	2022	50.408	-1890.2999999999999		
2022-01-01	S001	P0003	Toys	West	102	65	51	74.02	27.99	10	
Sunny				Summer	1819.35	1	2022	25.191	-181.93499999999997		
2022-01-01	S001	P0004	Toys	North	469	61	164	62.18	32.72	10	
Cloudy				Autumn	1995.9199999999998	1	2022	29.448	-199.59199999999999		
2022-01-01	S001	P0005	Electronics	East	166	14	135	9.26	73.64	0	
Sunny				Summer	1030.96	1	2022	73.64	0.0		

only showing top 5 rows

## 2. Group by Region to calculate total profit.



The screenshot shows a Jupyter Notebook with a file explorer on the left and a code editor on the right. The file explorer lists various files, including 'python\_files.ipynb', 'python\_functional.i...', 'python\_functions.ip...', 'python\_introduction...', 'python\_modules.ip...', 'python\_pickle.ipynb', 'python\_re.ipynb', 'python\_setup.md', 'Ravipati\_hranalytics...', 'Ravipati\_join.ipynb', 'README.md', 'retail\_store\_invento...', 'Sales.csv', 'Untitled.ipynb', 'Untitled1.ipynb', and 'Untitled2.ipynb'. The code editor shows the following code:

```
profitability_by_region = df_cleaned.groupby("Region").sum("Profit").orderby("sum(Profit)", ascending=False)
profitability_by_region.show()
```

The output of the code is displayed below the code cell, showing a table with two columns: 'Region' and 'sum(Profit)'. The table lists the total profit for each region, ordered from highest to lowest profit.

Region	sum(Profit)
West	-1.370991485849997E7
North	-1.37116849985000...
South	-1.39161784660005E7
East	-1.39198116384999...

## 2. Detailed Discussion of Results Achieved for each goal

### 2.1. Goal 1: Data Cleaning and Preparation

The data was successfully cleaned and standardized, ensuring no missing values or inconsistent column types. This improved the dataset's quality and usability.

### 2.2. Goal 2: Sales Insights by Region

Regions were ranked based on total sales, identifying the highest and lowest performers. This insight helps target regions for marketing or operational improvements.

### 2.3. Goal 3: Top Products by Region

The top 5 products for each region were identified, highlighting the most successful products by sales volume in different areas.

### 2.4. Goal 4: Sales Trend Analysis

Monthly sales trends revealed seasonal patterns. For instance, sales increased during specific months, likely due to holidays or promotions.

### **2.5. Goal 5: Visualization of Insights**

The visualizations provided a clear representation of sales trends and regional performance, making it easier for stakeholders to interpret the results.

### **2.6. Goal 6: Profitability Analysis**

Profitability analysis highlighted regions and products with the highest and lowest profit contributions. These insights can guide pricing and marketing strategies.

## **3. Conclusion**

This project demonstrates the power of PySpark for analyzing large datasets. By cleaning the data, generating insights, and visualizing key findings, we provided actionable insights into sales performance and profitability. These results can help organizations optimize their operations and strategies.

## **4. Citations**

1. Dataset Link: <https://www.kaggle.com/datasets/anirudhchauhan/retail-store-inventory-forecasting-dataset>

## **5. GitHub Repository**

The complete project, including source code and visualizations, is available at: <https://github.com/your-repo-url>