**Data Spark: Illuminating Insights for Global Electronics**

**Overview**

This project, titled **Data Spark: Illuminating Insights for Global Electronics**, aims to deliver a comprehensive Exploratory Data Analysis (EDA) report for Global Electronics. The primary goals include ensuring all datasets are clean and integrated, uncovering trends and insights through EDA, creating visualizations to effectively communicate key findings, and providing actionable recommendations.

**Prerequisites**

* Python 3.x
* Required Python libraries: pandas, seaborn, matplotlib, mysql-connector-python
* MySQL server
* Power BI

**Usage**

1. **Load Data**:
   * Ensure all CSV files (Products.csv, Sales.csv, Stores.csv, customers.csv, Exchange\_Rates.csv) are in the same directory as the script.
2. **Preprocess Data**:
   * The script cleans and preprocesses the data by handling missing values, converting data types, and standardizing numerical columns.
3. **Merge Data**:
   * The script merges the cleaned datasets into a single DataFrame for further analysis.
4. **Outlier Detection and Reduction**:
   * The script identifies outliers using box plots and reduces them by standardizing the data and removing points beyond 3 standard deviations.
5. **Database Interaction**:
   * The script creates tables in a MySQL database and inserts the processed data.
   * Ensure the MySQL server is running and accessible.

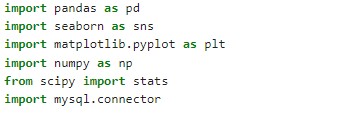
**Setup Instructions**

1. Install Dependencies

Installed the required Python packages using pip:

pip install pandas seaborn matplotlib numpy scipy mysql-connector-python

**Imports:**

****

**Code Explanation:**

**load\_data Function Explanation**

* The **load\_data function** is responsible for loading various datasets from CSV files into pandas Data Frames. These datasets include product information, sales data, store details, customer information, and exchange rates.



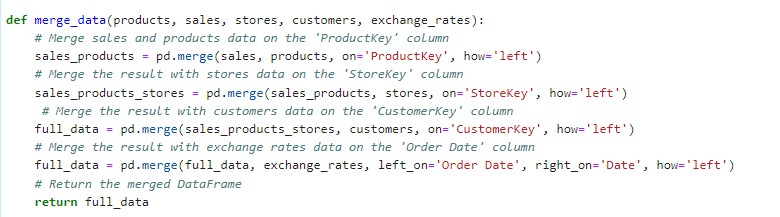
**preprocess\_data Function Explanation**

* The **preprocess\_data** function handles missing values, converts date columns to a consistent format, cleans and converts monetary columns, ensures key columns are of the correct type, and removes any invalid rows. This ensures the data is clean and ready for analysis.



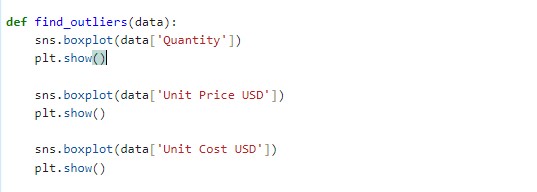
**Merge\_data Function Explanation**

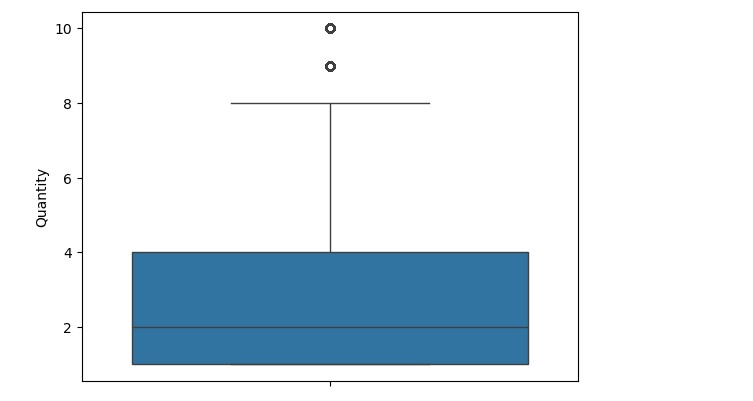
* The **merge\_data** function performs a series of left joins to combine the sales, products, stores, customers, and exchange\_rates Data Frames into a single DataFrame. This merged Data Frame, full\_data, contains comprehensive information from all the source DataFrames, allowing for thorough analysis and insights.

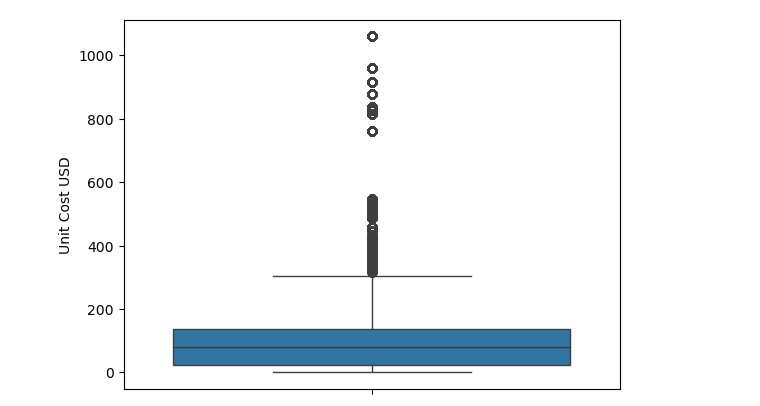


**Find\_outliers Function Explanation**

* The **find\_outliers** function creates and displays box plots for the Quantity, Unit Price USD, and Unit Cost USD columns of the input DataFrame.
* These box plots help visually identify outliers in the data. Visually inspecting these box plots, you can easily identify data points that fall outside the expected range and may be considered outliers.
* This is a crucial step in data analysis, as outliers can significantly affect the results and interpretations of statistical analyses.

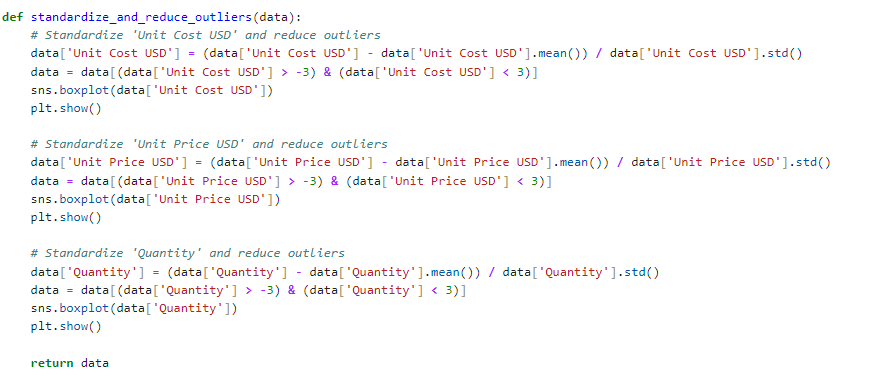




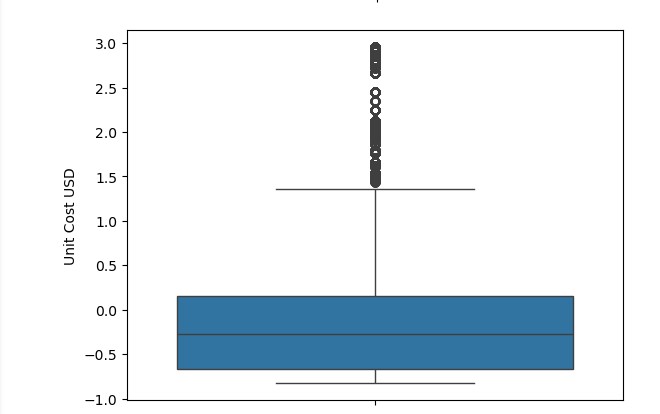


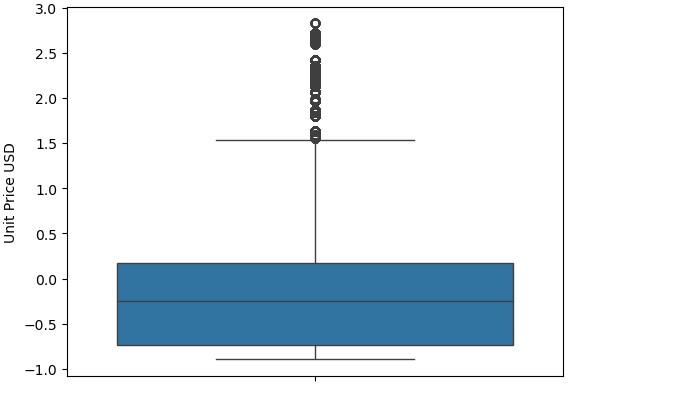
**Standardize\_and\_reduce\_outliers Function Explanation**

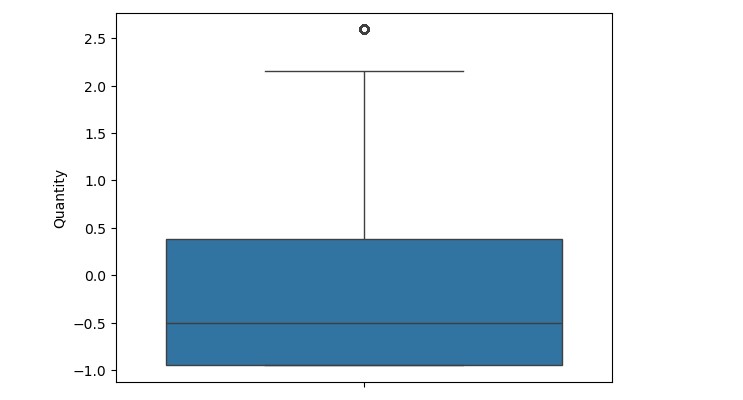
* The standardize\_and\_reduce\_outliers function standardizes the Unit Cost USD, Unit Price USD, and Quantity columns of the DataFrame, removing data points that are more than 3 standard deviations away from the mean.
* This process helps in mitigating the influence of outliers and ensures a more normally distributed dataset. The function also provides visualizations of the standardized data using box plots, aiding in the identification of remaining outliers and understanding the distribution of the data.



Box Plot:

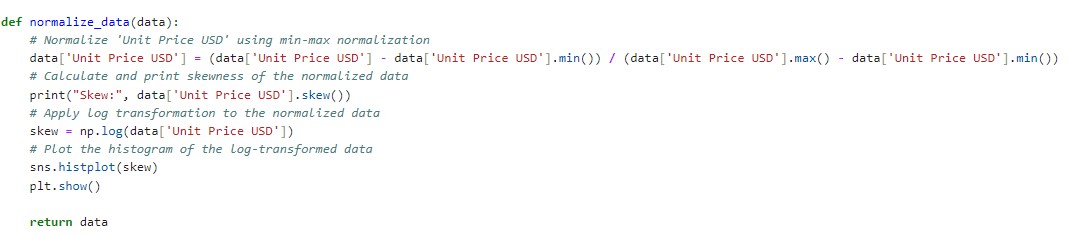




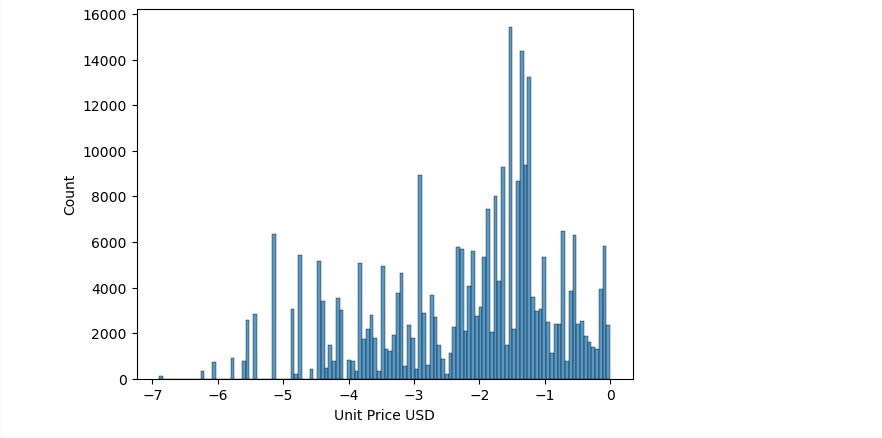


**Normalize\_data Function Explanation:**

* The normalize\_data function normalizes the 'Unit Price USD' column in the dataset, calculates its skewness, and visualizes the data distribution before and after applying a logarithmic transformation.
* This transformation scales the values of 'Unit Price USD' to be between 0 and 1.

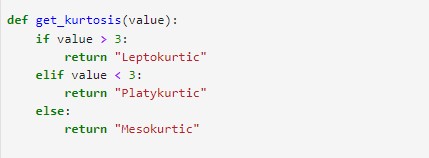


Histplot:



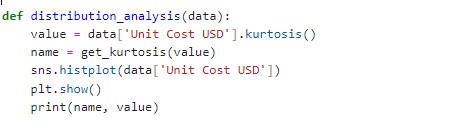
**Get\_kurtosis Function Explanation:**

* The get\_kurtosis function classifies the kurtosis of a distribution based on the given value.
* Kurtosis is a statistical measure that describes the shape of a distribution's tails in relation to its overall shape.

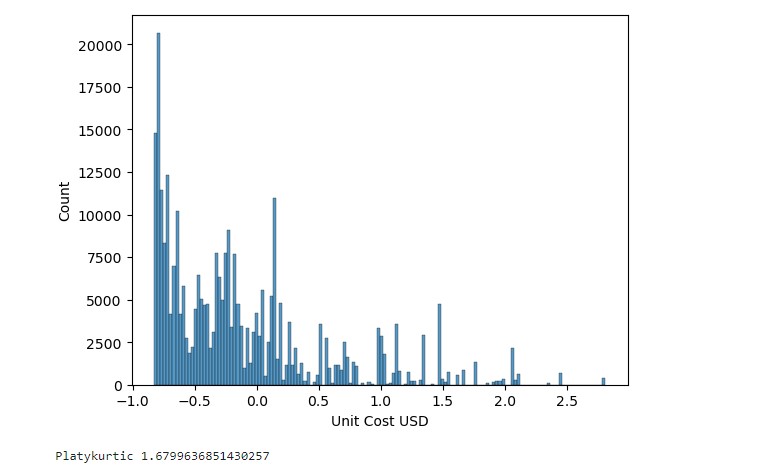


**Distribution\_analysis Function Explanation:**

* The distribution\_analysis function performs a kurtosis analysis on the 'Unit Cost USD' column of a given DataFrame.
* It then classifies the distribution based on the kurtosis value, visualizes the distribution using a histogram, and prints the kurtosis type and value.

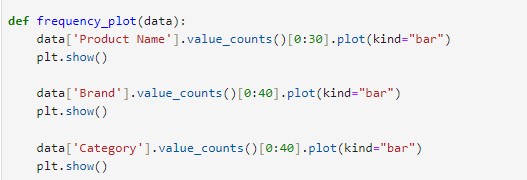


Hist plot:



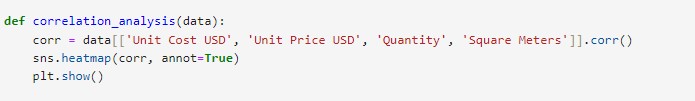
**Frequency\_plot Function Explanation:**

* The frequency\_plot function generates bar plots showing the frequency distribution of product-related attributes from a given Data Frame.
* It visualizes the top 30 products, top 40 brands, and top 40 categories based on their occurrence in the dataset.

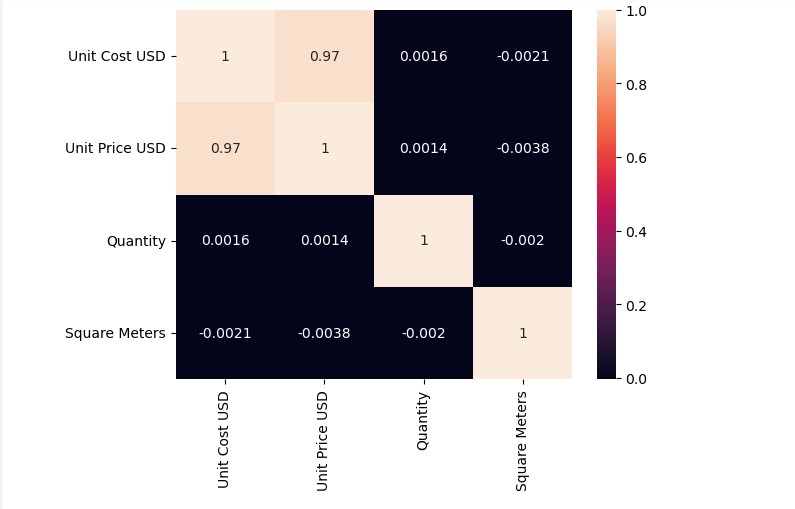


**Correlation\_analysis Function Explanation:**

* The correlation\_analysis function performs a correlation analysis on a subset of numerical columns from a DataFrame and visualizes the results using a heatmap.



Heat map:



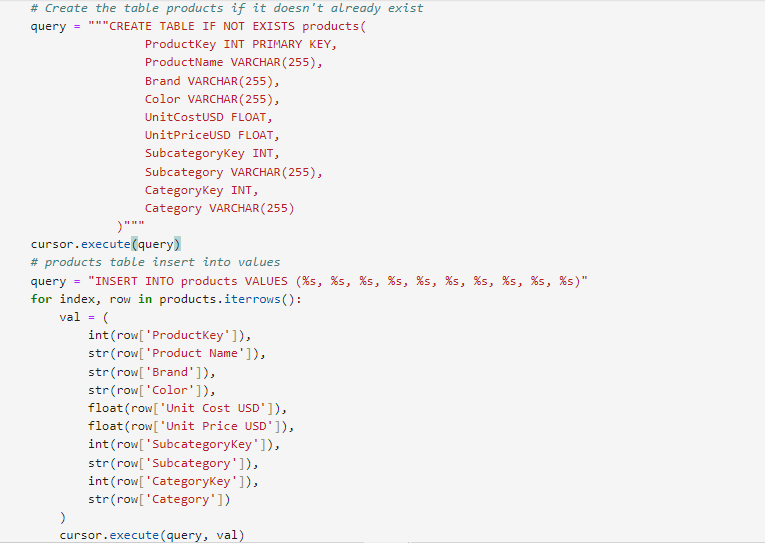
**Database Connection**

* The create\_database function connects to a MySQL database and prepares to interact with it.
* mysql. connector.connect() establishes a connection to a MySQL database.



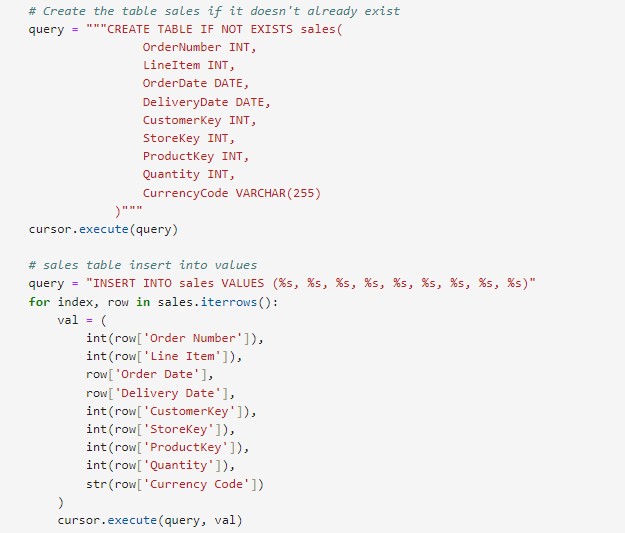
**Create the products Table:**

* create a new table named products if it does not already exist.
* inserts data from a DataFrame into the products table, converted the data to the correct types and handling the insertion in a loop.



**Create the sales Table:**

* creates a new table named sales if it does not already exist.
* Inserts data from a DataFrame into the sales table, ensuring data is converted to the correct types and inserted row by row.



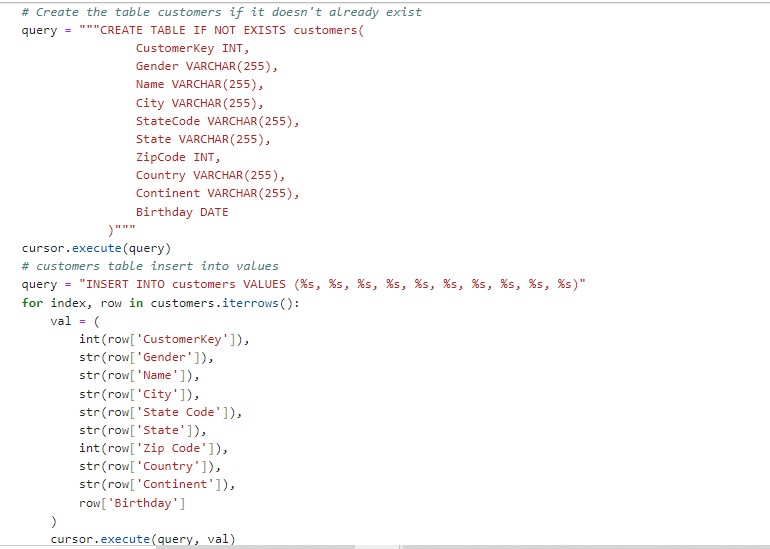
**Create the stores Table:**

* creates a new table named stores if it does not already exist.
* Inserts data from a DataFrame into the stores table, ensuring data is converted to the correct types and inserted row by row.



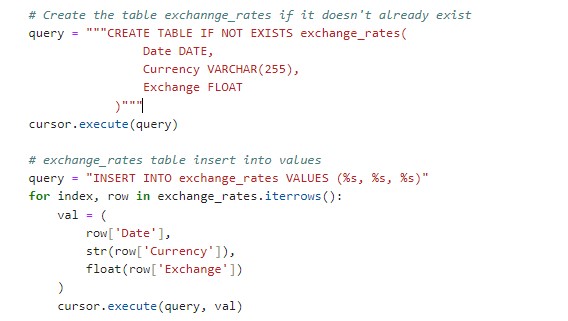
**Create the customers Table:**

* creates a new table named customers if it does not already exist.
* Inserts data from the DataFrame into the customers table, ensuring that each column's data is converted to the correct type and inserted row by row.



**Create the exchannge\_rates Table:**

* creates a new table named exchange\_rates if it does not already exist.
* Inserts data from the DataFrame into the exchange\_rates table, converting each piece of data to the appropriate type and inserting it row by row.

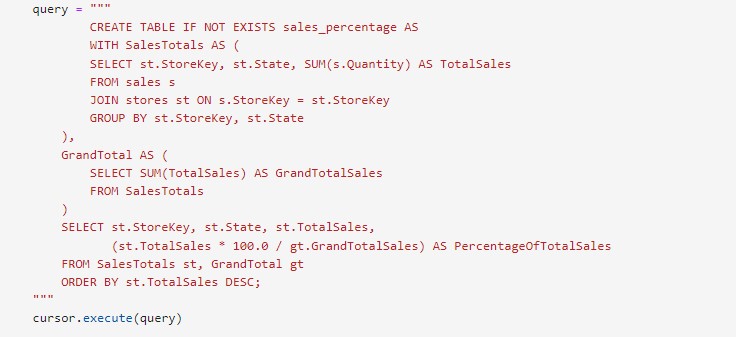


**Visualizations process:**

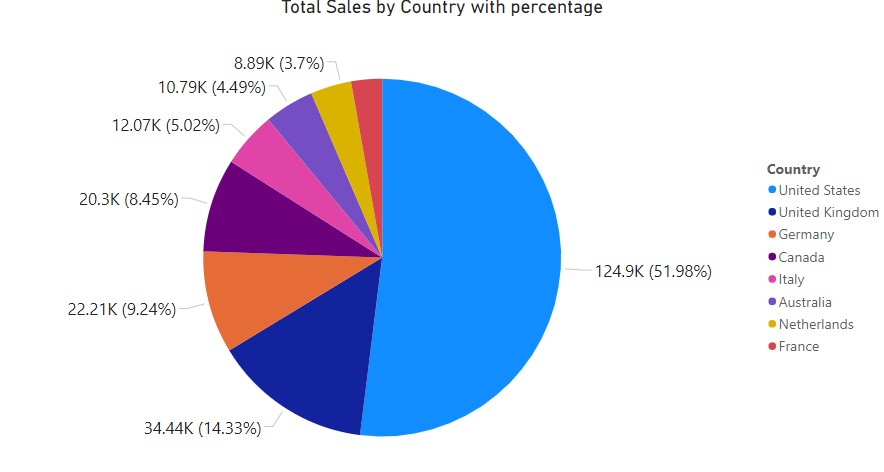
1. **Creating the sales\_percentage Table**:

The function creates a sales\_percentage table that shows the percentage contribution of each store's sales to the total sales.

* Calculating total sales per store.
* Computing the grand total of sales across all stores.
* Determining each store's percentage of the grand total and ordering the results by total sales.



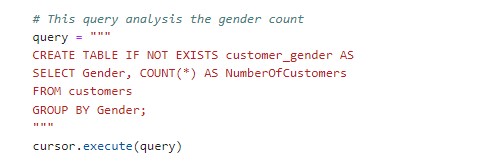
Power BI Visualizations:



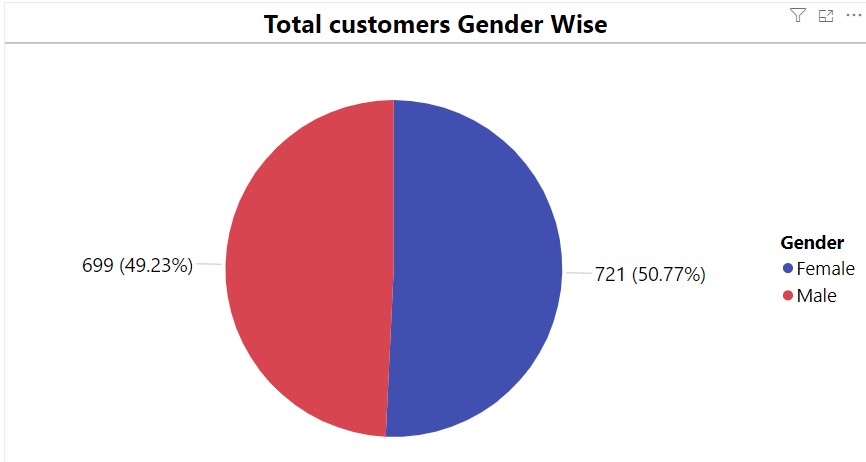
2) **Creating the customer\_gender Table**:

The query creates a table called customer\_gender that contains:

* Gender: The gender of customers.
* NumberOfCustomers: The count of customers for each gender.
* This table contains how many customers fall into each gender category, which can be useful for understanding the gender distribution among the customer base.



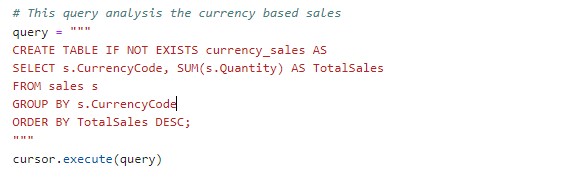
Power BI Visualizations:



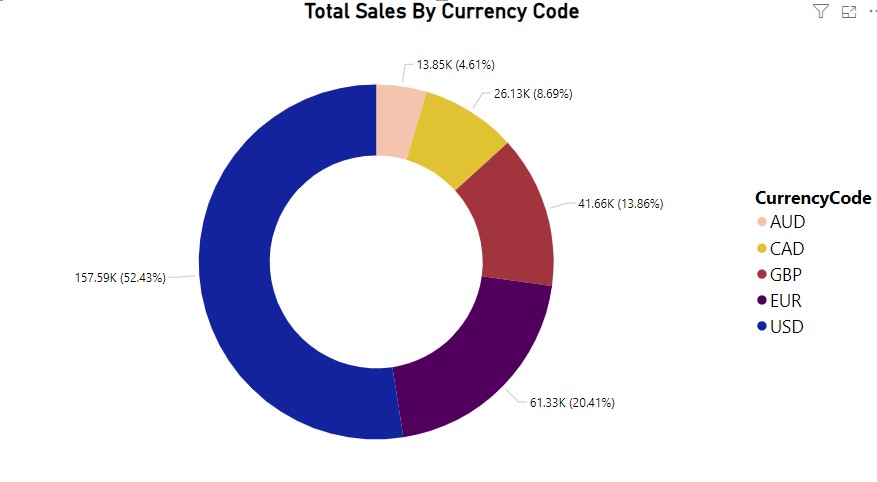
3) **Creating the currency\_sales** **Table**:

The query creates a table called currency\_sales that contains:

* Currency Code: The currency code used in the sales transactions.
* Total Sales: The total quantity of items sold for each currency code.
* This table helps analyse sales performance based on different currencies, providing insights into which currencies are associated with higher sales quantities.



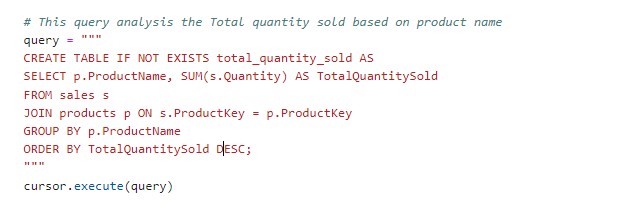
Power BI Visualizations:



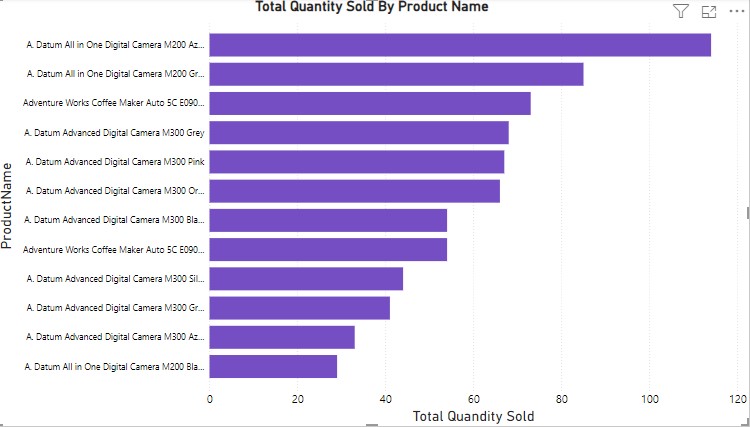
4) **Creating the total\_quantity\_soldTable**:

The query creates a table called total\_quantity\_sold that contains:

* ProductName: The name of each product.
* TotalQuantitySold: The total quantity of each product sold.
* This table provides insight into which products are selling the most, helping in inventory management, sales strategies, and identifying popular products.



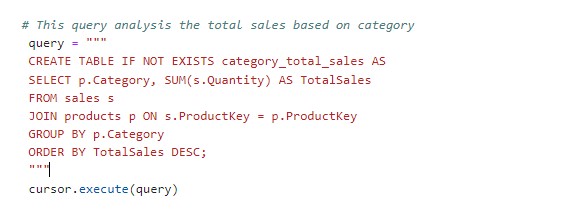
Power BI Visualizations:



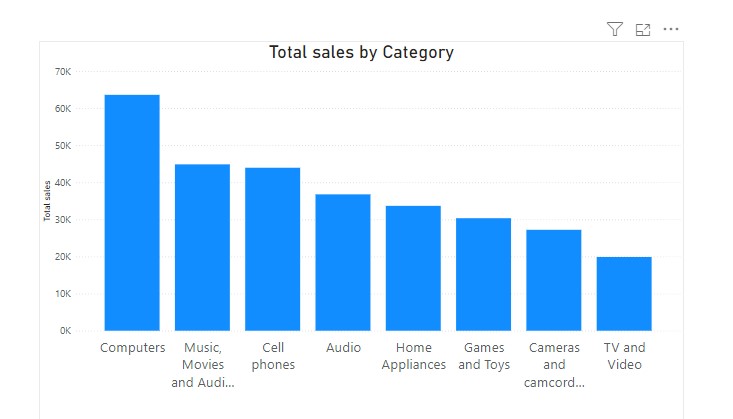
5) **Creating the total\_quantity\_soldTable**:

The query creates a table called category\_total\_sales that contains:

* Category: The category of each product.
* Total Sales: The total quantity of products sold in each category.
* This table provides insight into which product categories are selling the most, helping in category management, marketing strategies, and identifying popular product categories.



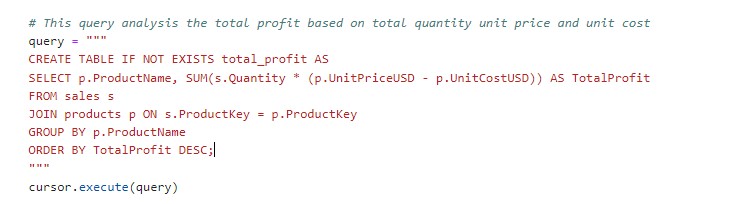
Power BI Visualizations:



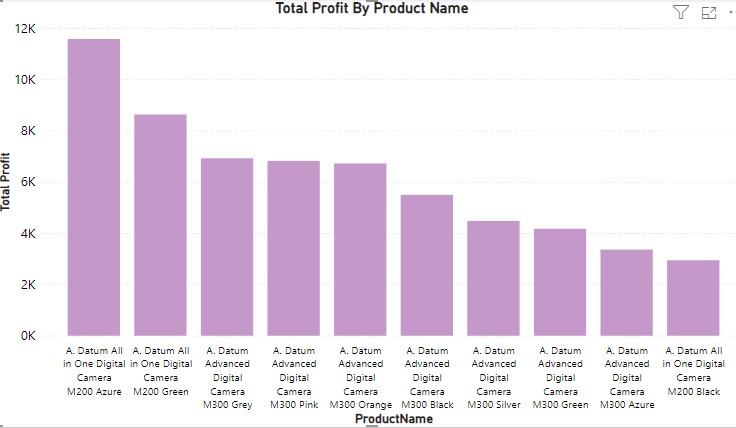
6) **Creating the total\_profit** **Table**:

The query creates a table called total\_profit that contains:

* ProductName: The name of each product.
* TotalProfit: The total profit generated by each product.
* This table provides insight into the profitability of each product, helping to identify which products contribute the most to the overall profit. This information can be used for strategic decision-making in pricing, marketing, and inventory management.



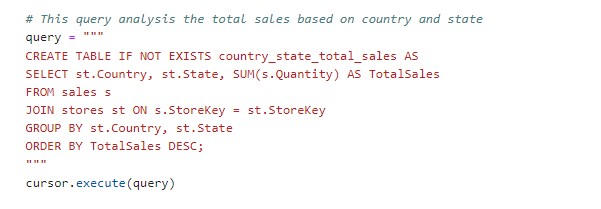
Power BI Visualizations:



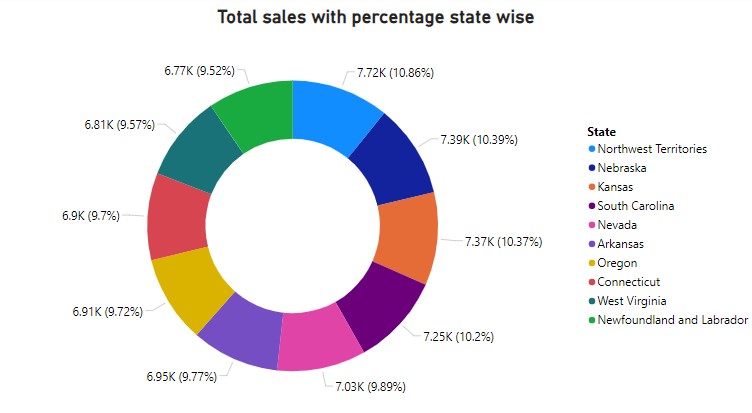
7) **Creating the country\_state\_total\_sales Table**:

The code creates a new table called country\_state\_total\_sales, which includes:

* Country: The country where the sales occurred.
* State: The state where the sales occurred.
* Total Sales: The total sales (quantity sold) for each combination of country and state.
* This table provides valuable insights into the geographical distribution of sales, aiding in strategic decision-making for market expansion, regional marketing strategies, and inventory allocation.



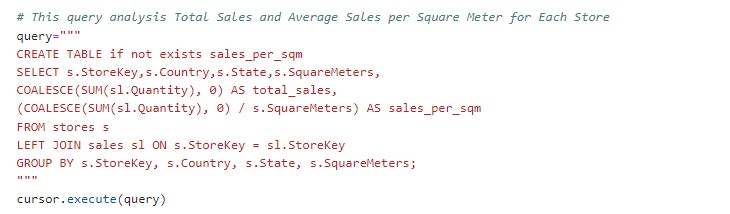
Power BI Visualizations:



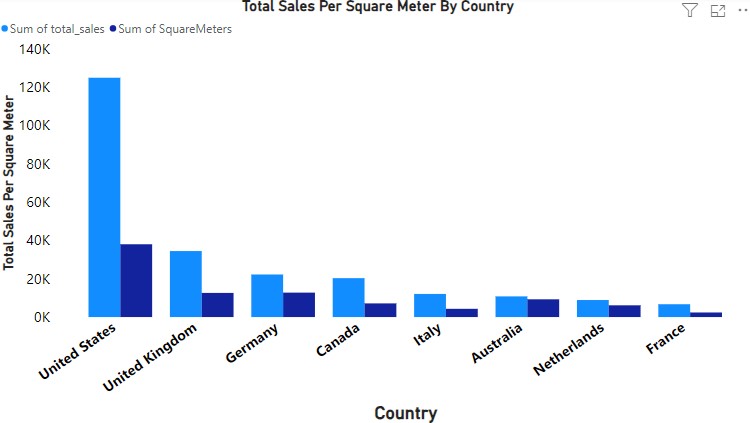
8) **Creating the sales\_per\_sqm Table**:

create a new table called sales\_per\_sqm if it doesn't already exist.

* The query creates a table that provides insights into the sales performance of each store by calculating both the total sales and the sales per square meter for each store.
* It ensures that all stores are included in the analysis, even those with no sales, by using a left join and the COALESCE function. This information can be useful for evaluating the efficiency and productivity of each store based on its size.



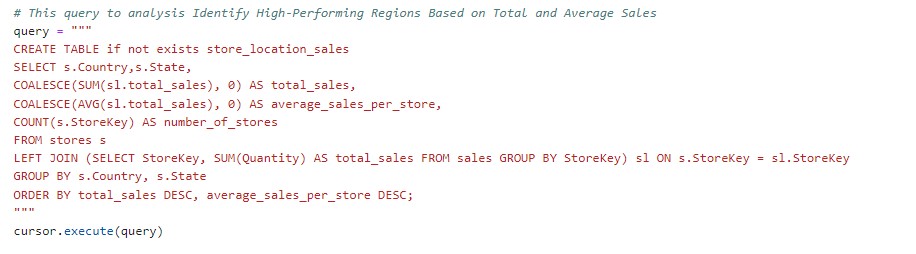
Power BI Visualizations:



9) **Creating the store\_location\_sales Table**:

create a new table called store\_location\_sales if it doesn't already exist.

* The table aims to identify high-performing regions based on total and average sales.
* By grouping the data by country and state, summing the total sales, averaging the sales per store, and counting the number of stores, the query provides insights into which regions perform best in terms of sales.
* The results are ordered to highlight the top-performing regions, making it easier to analyze and compare their performance.



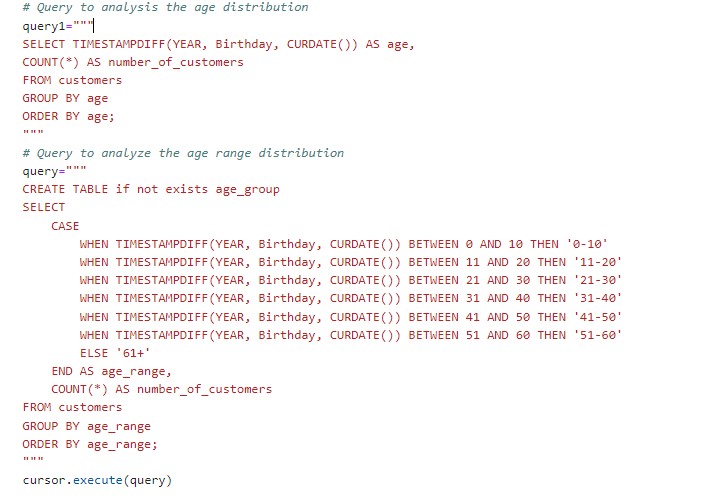
Power BI Visualizations:



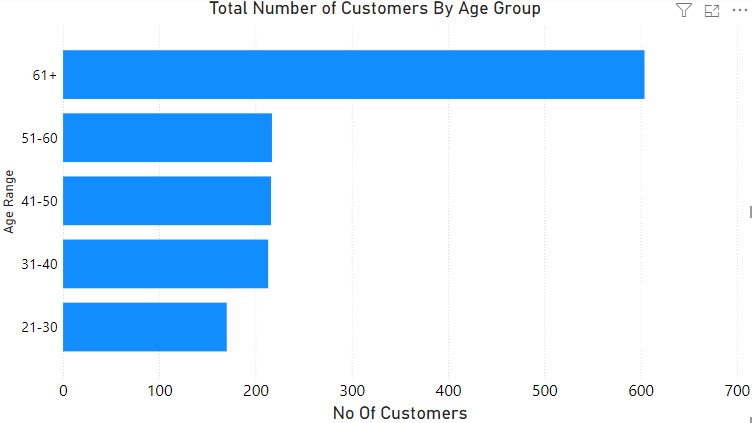
10) **Creating the age\_group** **Table**:

This creates a new table called age\_group if it doesn't already exist.

* To calculate the exact age of customers and count the number of customers for each age.
* To categorize customers into predefined age ranges and count the number of customers in each range.



Power BI Visualizations:



**Main Function:**

The main function manages the entire data processing pipeline, from loading and preprocessing data to analyzing and storing the results in a MySQL database. The process includes:

1. Loading data from various sources.
2. Preprocessing the data.
3. Merging the data into a single DataFrame.
4. Detecting and handling outliers.
5. Normalizing the data.
6. Performing data analysis and visualization.
7. Connecting to a MySQL database.
8. Creating analytics tables in the database based on the processed data.

**Running the Script**

To run the script, executed the following command terminal:

python data\_spark.py

GitHub Link: