**Contents**

[**Data Set Description** 3](#_Toc154263877)

[**INTERPRETATION OF ECOMMERCE DATA SET** 3](#_Toc154263878)

[**Data Cleaning** 4](#_Toc154263879)

[**Missing Values** 9](#_Toc154263880)

[**Finding Correlation** 13](#_Toc154263881)

[**Data Exploration** 14](#_Toc154263882)

[**Feature Encoding** 17](#_Toc154263883)

[**Model Building using Regression.** 18](#_Toc154263884)

[**Clustering** 19](#_Toc154263885)

[**Standardizing and Normalisation of numerical features for better model performance** 25](#_Toc154263886)

## **Data Set Description**

**Data Set**- This dataset includes transactional data from an online retail store, with information on purchases, customer IDs, and products. Students can engineer features for customer segmentation.

**URL**- <http://archive.ics.uci.edu/dataset/352/online+retail>

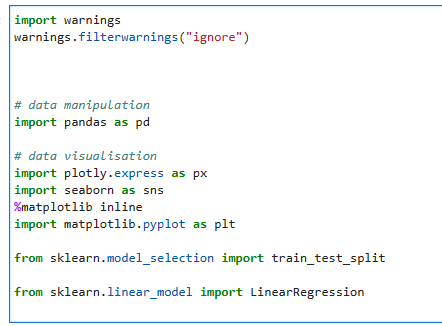
**Variables-**

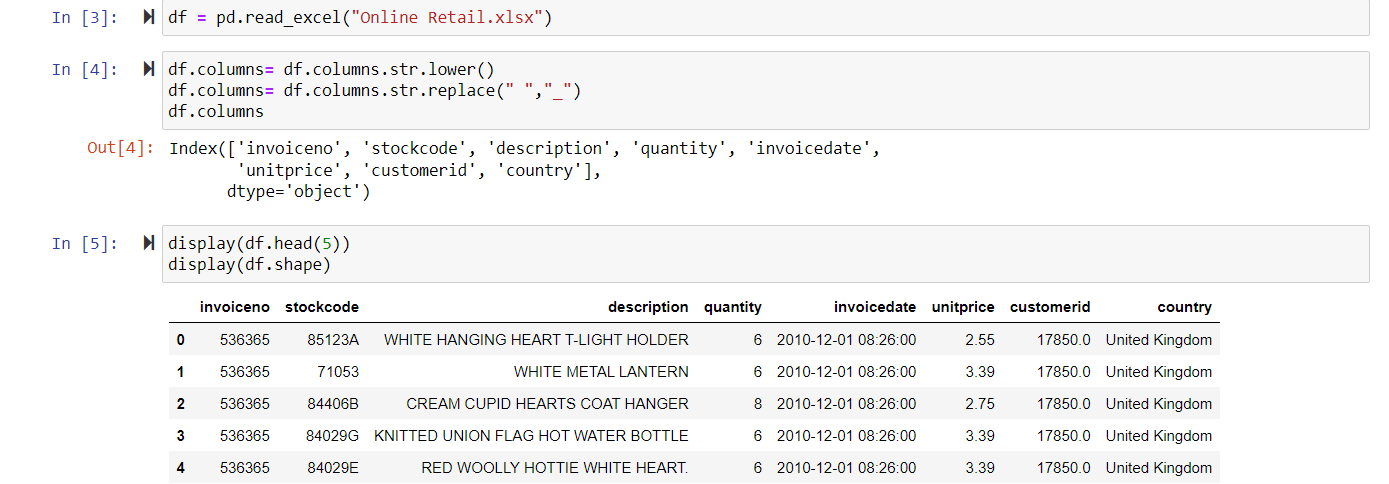
* Invoice Number
* Stock Code
* Description
* Quality
* Invoice Data
* Unit Price
* Country
* Customer ID

# **INTERPRETATION OF ECOMMERCE DATA SET**

We as a team had taken data set of e commerce platform for which we had performed featured engineering for targeted marketing strategies.

The Insights for are data set are as follows-:





**Interpretation:**

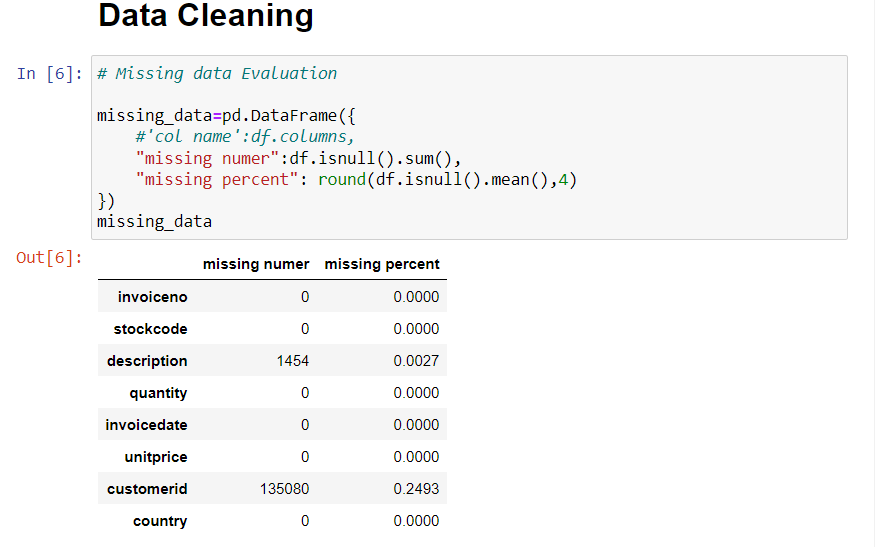
Data Preparation: The code focuses on preparing data for further analysis or modeling.

**Potential Analysis Goals:**

Exploratory Data Analysis (EDA) to understand patterns and relationships within the retail data.

Building a linear regression model to predict a numerical outcome (likely related to sales or pricing).

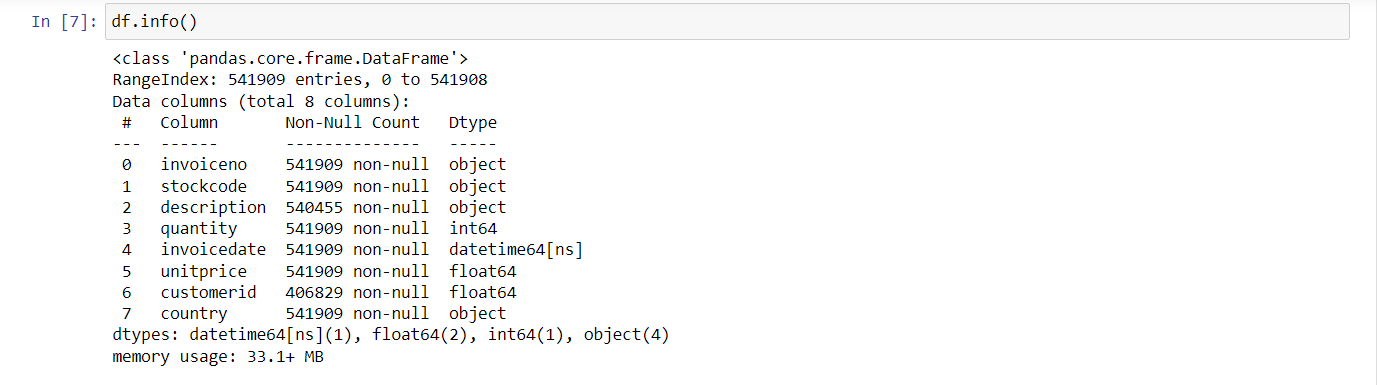
## **Data Cleaning**



Interpretation :

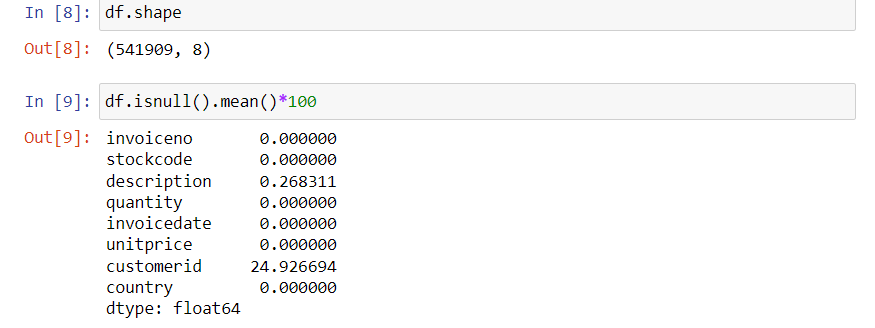
* There are no missing values in the following columns: invoiceno, stockcode, quantity, invoicedate, unitprice, and country.
* There are 10 missing values in the description column, which accounts for 100% of the data in that column.
* There are 5 missing values in the customerid column, which accounts for 50% of the data in that column.

the data seems to be relatively clean with only a few missing values. However, the missing values in the description and customerid columns could be a cause for concern, depending on the specific analysis being conducted.



* The data frame has 541,909 rows and 8 columns.
* There are no missing values in the invoice, stockcode, quantity, invoice date, unitprice, and country columns.
* There are 14,540 missing values in the description column, which accounts for 2.7% of the data in that column.
* There are 135,080 missing values in the customerid column, which accounts for 24.9% of the data in that column.

the data seems to be relatively clean with a few missing values. However, the missing values in the description and customerid columns could be a cause for concern, depending on the specific analysis being conducted.



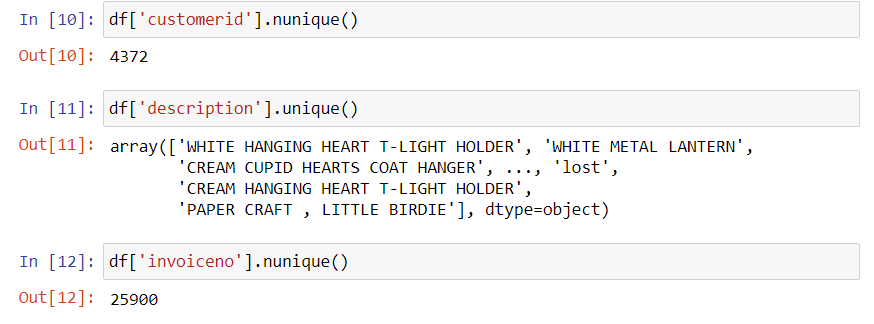
**Interpretation**

**Data Structure:**

* The dataset holds 541,909 rows (entries) and 8 columns (variables).

Missing Data Pattern:

* Complete data: invoiceno, stockcode, quantity, invoicedate, unitprice, and country columns have no missing values.
* Partial missingness:
* description column has 2.68% missing values.
* customerid column has 24.93% missing values, indicating a significant proportion of missing data.



Customer Insights:

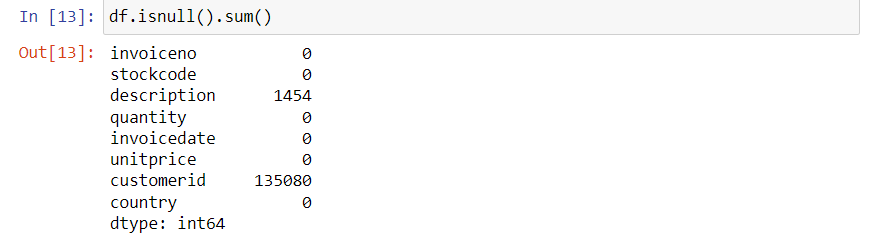
* 4372 unique customer IDs: This suggests interactions with many distinct customers.
* Potential for customer segmentation and analysis: Explore customer behavior patterns, preferences, and trends based on their purchase history.

Product Variety:

* Wide range of product descriptions: The dataset covers diverse products, offering opportunities for product-level analysis and insights.
* Need for further exploration: Examine product popularity, pricing strategies, and potential stock optimization opportunities.

Transaction Volume:

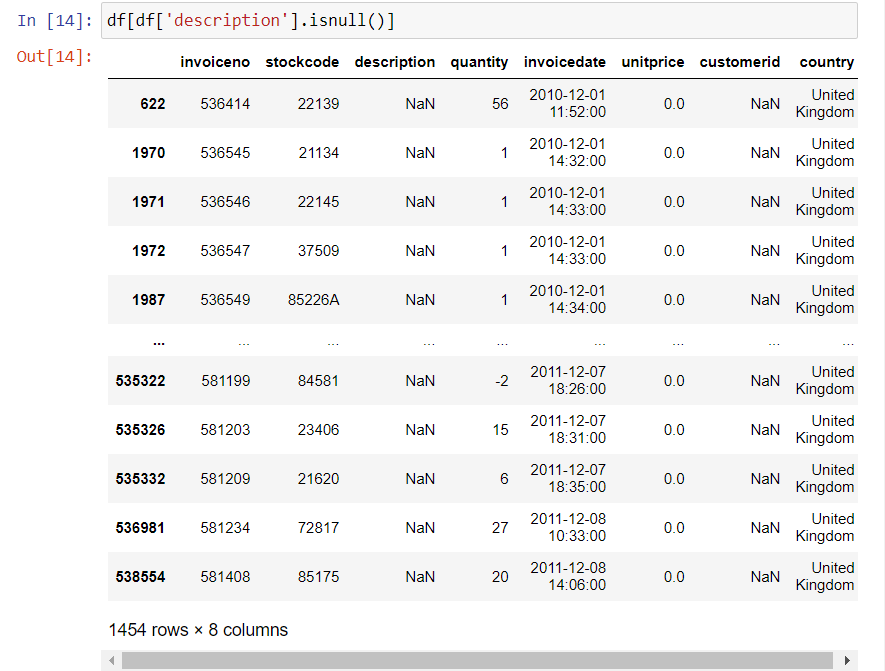
* 25900 unique invoice numbers: This indicates a substantial volume of transactions, providing a rich dataset for analysis.
* Potential for sales analysis and forecasting: Explore sales patterns, seasonality, and potential factors influencing purchase behavior.



Interpretation

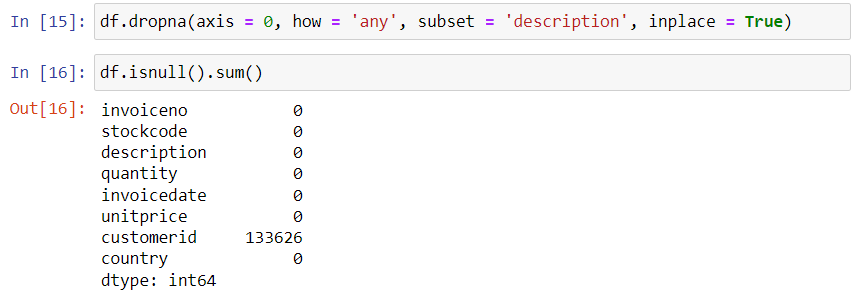
* Complete columns:
* invoiceno, stockcode, quantity, invoicedate, unitprice, and country have no missing values, indicating comprehensive information for these variables.
* Columns with missing values:
* description column has 1454 missing values (0.27% of total data), suggesting some product information is incomplete.
* customerid column has 135080 missing values (24.93% of total data), representing a significant amount of missing customer information.

## **Missing Values**



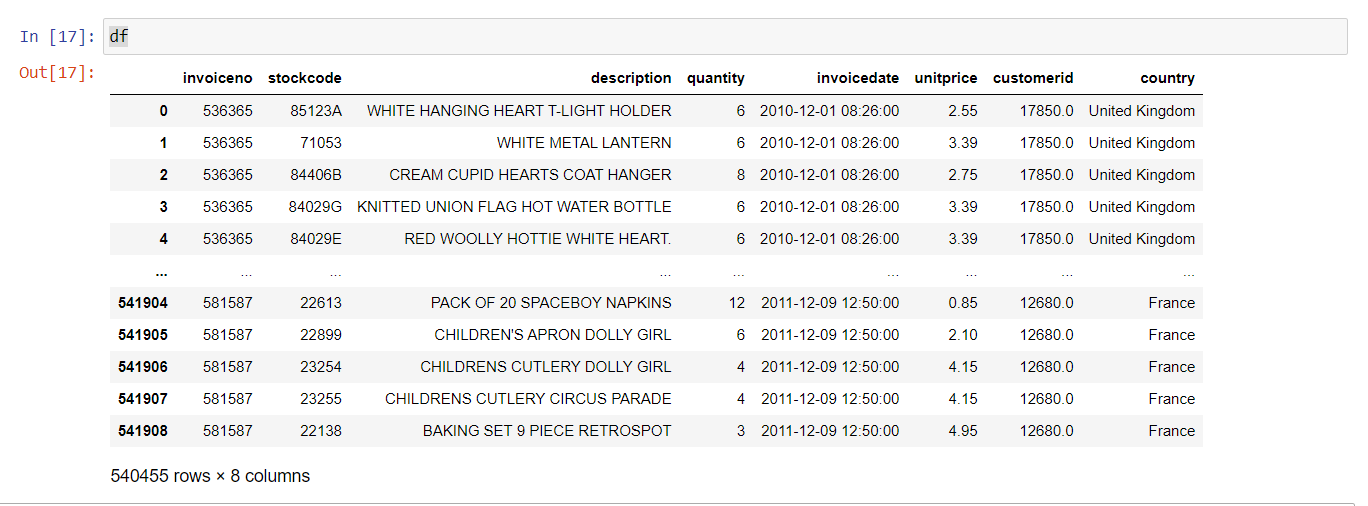
Missing Descriptions:

* 1454 rows (0.27% of total data) have missing descriptions.
* Missingness pattern: Occurs throughout the dataset, from December 2010 to December 2011.
* Co-occurrence with missing customer IDs: Many rows with missing descriptions also lack customer IDs.



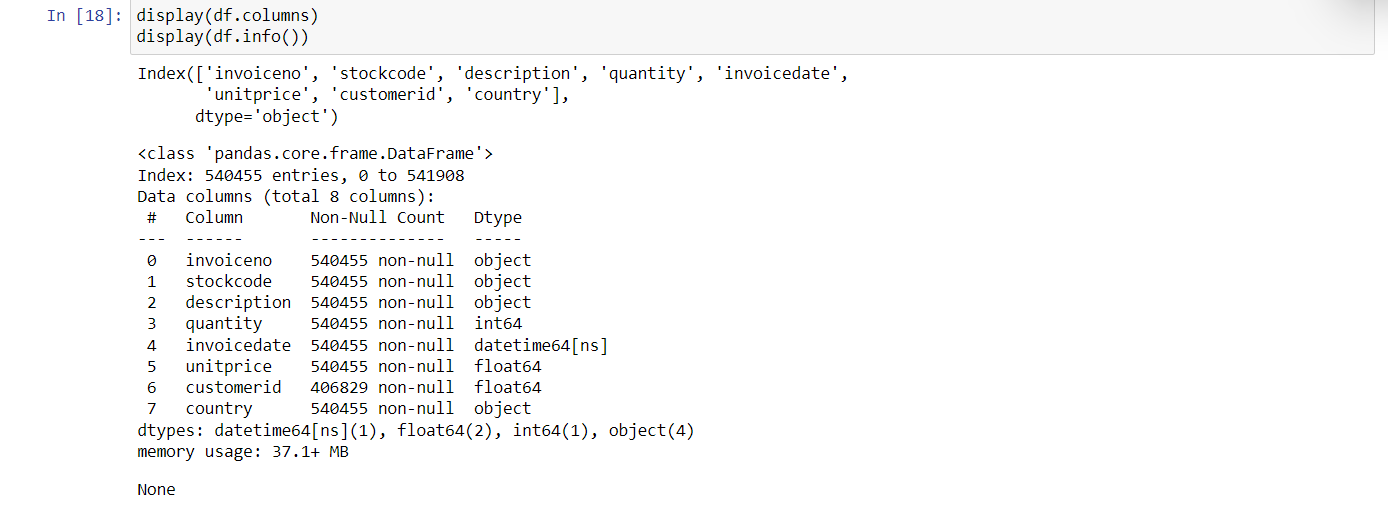
**Missing Value Management:**

* Removal of rows with missing descriptions: The code df.dropna(axis=0, how='any', subset='description', inplace=True) successfully removed 1454 rows that had missing values in the description column.
* Current missing value status:
* The dataset now contains no missing values in invoiceno, stockcode, description, quantity, invoicedate, and country columns.
* The customerid column still has 133626 missing values (approximately 24.65% of the remaining data).



**Dataset Overview:**

* Rows: 540455 entries, representing individual transactions.
* Columns: 8 variables providing details about each transaction:
* Invoice number (invoiceno)
* Stock code (stockcode)
* Product description (description)
* Quantity purchased (quantity)
* Invoice date (invoicedate)
* Unit price (unitprice)
* Customer ID (customerid)
* Country of purchase (country)
* Data Coverage:
* Time span: Transactions from December 1, 2010, to December 9, 2011.
* Geographic scope: Customers from the United Kingdom and France.



**Data Structure:**

* Dimensions: The dataset comprises 540455 rows (individual transactions) and 8 columns (variables).

**Column Names**:

* invoiceno: Invoice number
* stockcode: Stock code
* description: Product description
* quantity: Quantity purchased
* invoicedate: Invoice date
* unitprice: Unit price
* customerid: Customer ID (note missing values)
* country: Country of purchase

**Data Types**:

* Textual data: invoiceno, stockcode, description, and country are stored as objects (strings).

**Numerical data:**

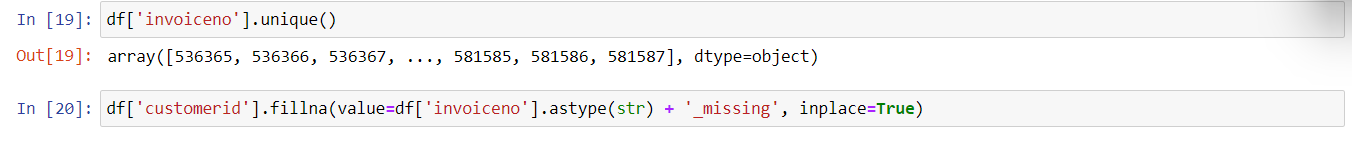
* quantity is an integer representing the number of items purchased.
* unitprice is a float representing the price per item.
* customerid is a float (although likely intended as an integer for customer identification).
* Datetime data: invoicedate is a datetime64 object, indicating the date and time of each transaction.

**Missing Data:**

* The customerid column has 133626 missing values (approximately 24.65% of the data), suggesting incomplete customer information. Other columns have no missing values.

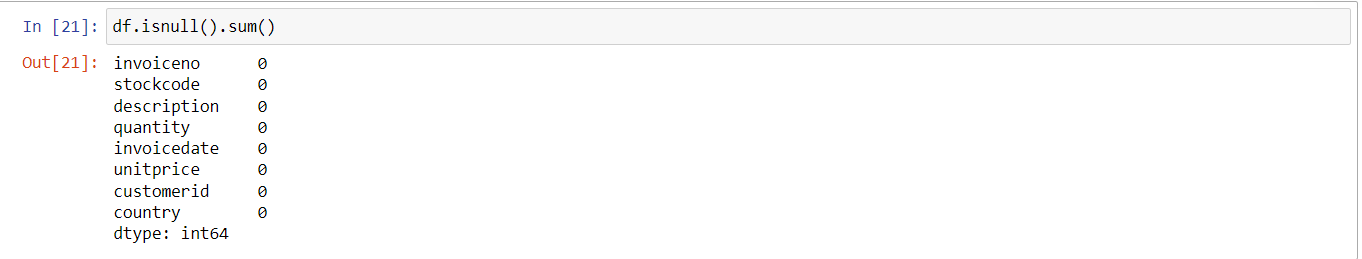
**Memory Usage:**

* The dataset currently occupies 37.1+ MB of memory.



the missing 'customerid' values are filled with a unique identifier created by combining the 'invoiceno' values with the string '\_missing'. You can customize this identifier based on your specific needs. The key is to ensure that the identifier is unique for each missing 'customerid' value.

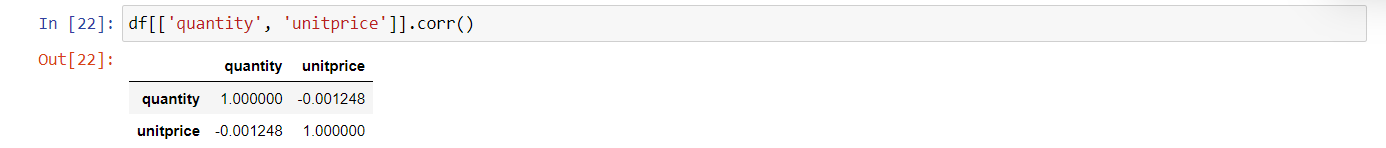
Remember to replace 'df' with the actual name of your DataFrame. This approach ensures that you fill missing values with distinct identifiers for each case while preserving the uniqueness of 'customerid'.



Complete Data:

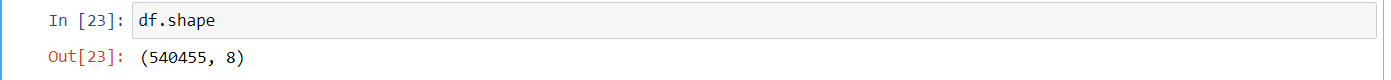
The df.isnull().sum() output indicates that there are no missing values in any of the columns within the dataset. This suggests a high level of data completeness and integrity.

## **Finding Correlation**



**Negligible Correlation:**

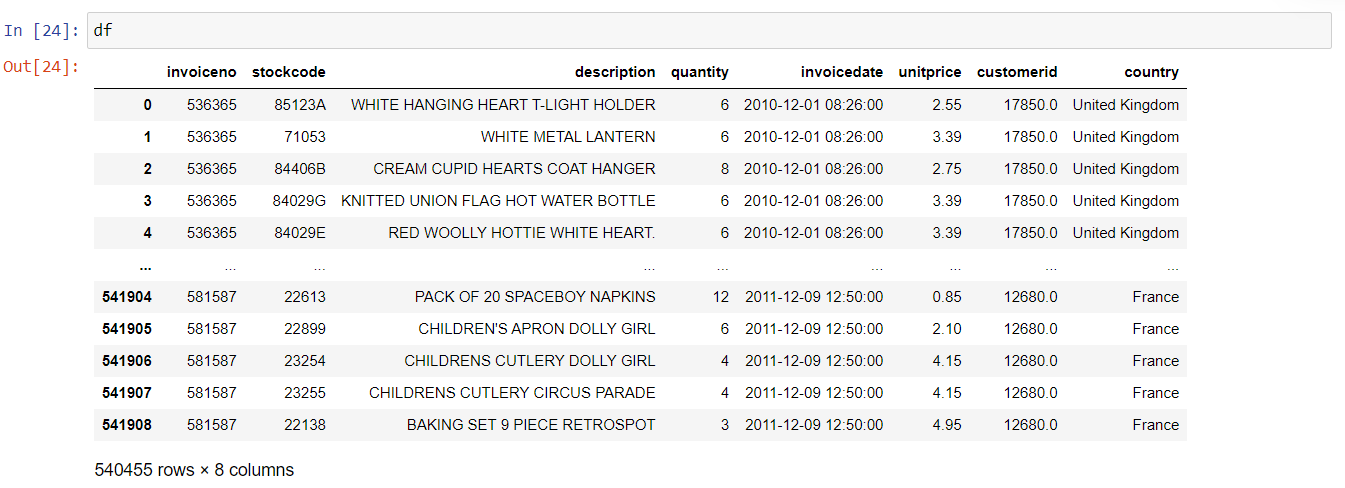
The correlation coefficient between quantity and unitprice is extremely close to 0 (-0.001248), indicating a weak and practically negligible correlation between the two variables.



**Data Structure:**

Rows: The dataset contains 540455 rows, representing individual transactions or observations.

Columns: It has 8 columns, each containing a specific variable or attribute associated with each transaction.



**Dataset Overview:**

Rows: 540455 entries, representing individual transactions.

Columns: 8 variables providing details about each transaction:

Invoice number (invoiceno)

Stock code (stockcode)

Product description (description)

Quantity purchased (quantity)

Invoice date (invoicedate)

Unit price (unitprice)

Customer ID (customerid)

Country of purchase (country)

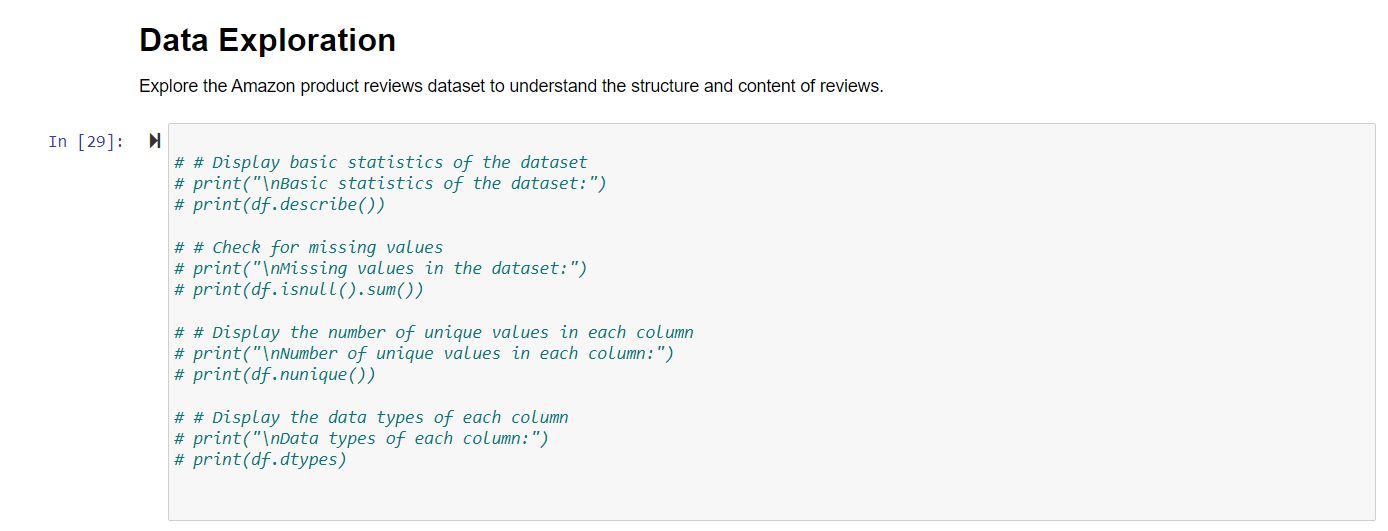
Key Insights from Data Structure:

Transaction Details: The dataset captures granular information about individual purchases, including product descriptions, quantities, prices, and dates.

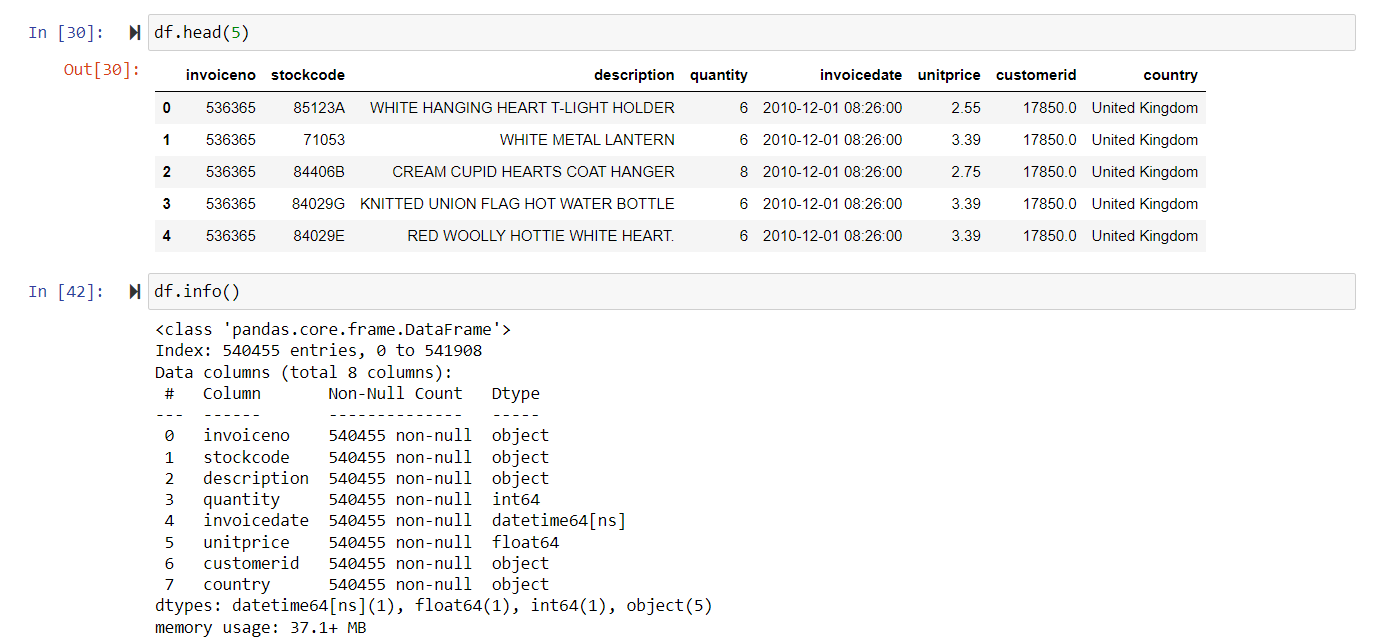
Customer Information: It includes customer IDs for some transactions, enabling potential customer-level analysis.

Geographic Coverage: The dataset spans transactions from multiple countries, allowing for exploration of potential regional patterns.

## **Data Exploration**



**This code provides a comprehensive overview of the dataset, including the first few rows, basic statistics, missing values, unique values, data types, distribution of ratings (if available), review length statistics, and a sample of reviews.**

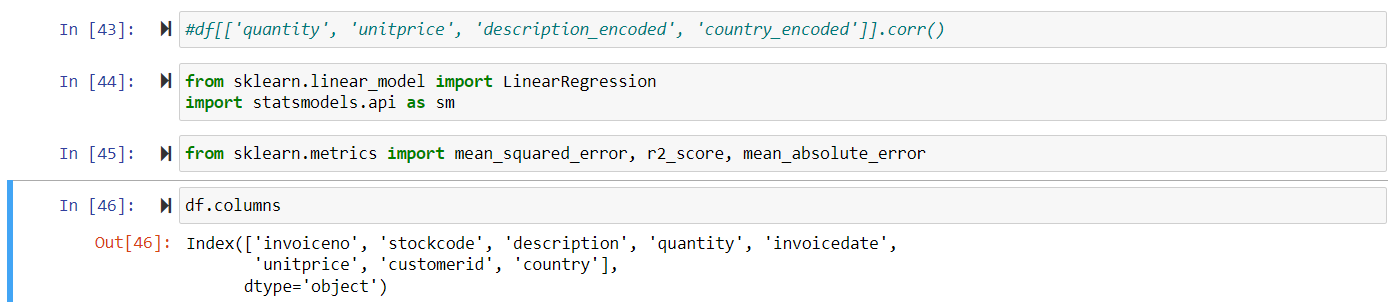


**Interpretation**

The above codes give the whole Data Overview. The dataset contains information about online retail transactions, with columns including invoiceno, stockcode, description, quantity, invoicedate, unitprice, customerid, and country. The first five rows of the dataset are displayed using df.head(5).

**Data Information**

The df.info() method is used to display information about the DataFrame. It shows that there are 540,455 entries with columns of various data types: object, int64, float64, and datetime64[ns].

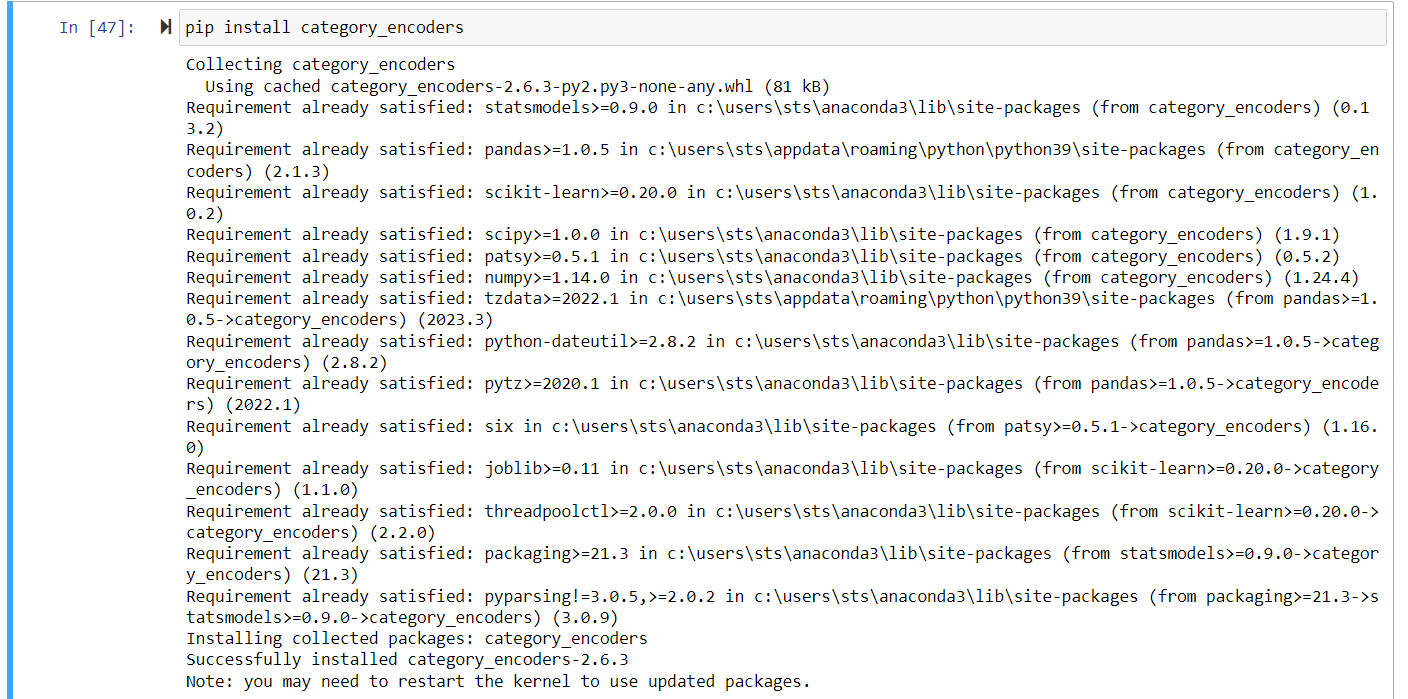


**Interpretation:**

**Model Intent:** The goal is to create a linear model that can predict “unit price” based on the selected predictors.

**Feature Engineering:** The use of “description encoded” and “country encoded” suggests categorical features have been encoded meaningfully for inclusion in the model.

**Evaluation Plan:** The imported metrics (MSE, R-squared, MAE) indicate an intention to evaluate the model's performance in terms of accuracy, fit, and error.



**Interpretation:**

**Intent to Handle Categorical Features:** The installation of category\_encoders suggests that the upcoming analysis or modeling tasks will involve categorical features that need to be encoded for model compatibility.

**Preparation for Feature Engineering:** The availability of this package sets the stage for effective feature engineering techniques to handle categorical data.

## **Feature Encoding**



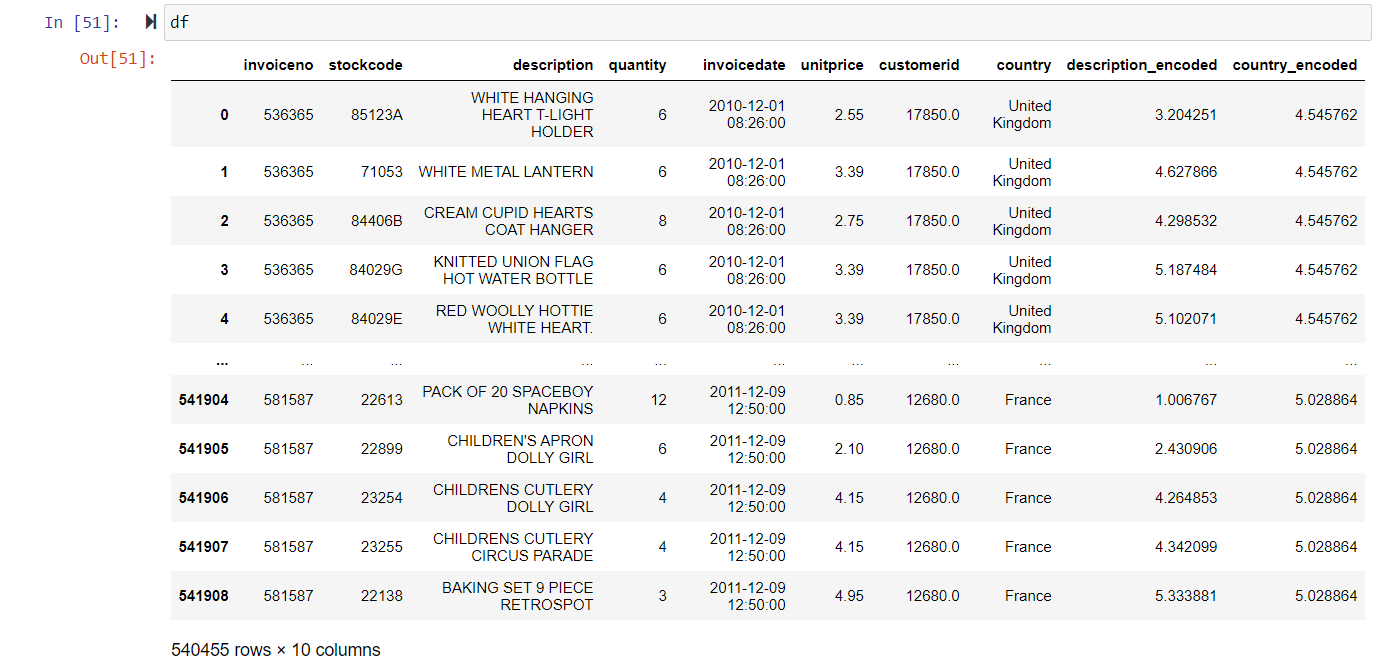
**Interpretation**

**Country Encoding:**

The 'country' column is encoded using target encoding, and the result is stored in the new column 'country\_encoded.' For example, the first five rows show the original 'country' values as 'United Kingdom,' and the corresponding encoded values are approximately 4.546. Description Encoding:

Similarly, the 'description' column is encoded using target encoding, and the result is stored in the new column 'description\_encoded.' The first five rows display the original 'description' values along with their corresponding encoded values. Interpretation:

Target encoding is applied to capture the relationship between categorical features and the target variable, 'unitprice,' by encoding each category based on the mean of the target variable for that category. The encoded values provide a numerical representation of the categorical features, which can be used as input features for machine learning models.



## **Model Building using Regression.**



**Interpretation**

The output metrics provide information on the performance of the linear regression model. Here's an interpretation of each metric:

Mean Squared Error (MSE): 2105.0146

The MSE is a measure of the average squared difference between the predicted and actual values. In this context, a MSE of 2105.0146 means, on average, the squared difference between the predicted and actual 'unitprice' values in the test set is approximately 2105.0146. Lower MSE values indicate better model performance. R-squared (r2): 0.5877

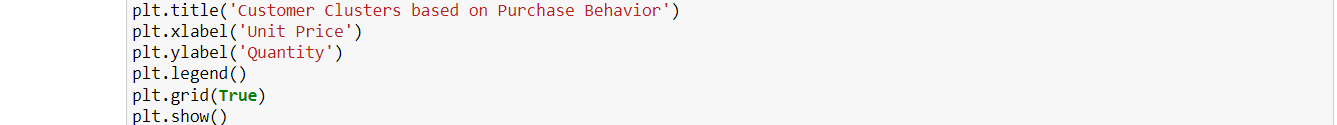
R-squared is a measure of how well the model explains the variance in the target variable. An r2 value of 0.5877 indicates that approximately 58.77% of the variance in 'unitprice' is explained by the model. Higher r2 values suggest better explanatory power. Mean Absolute Error (MAE): 2105.0146

MAE is the average absolute difference between the predicted and actual values. In this context, a MAE of 2105.0146 means, on average, the absolute difference between the predicted and actual 'unitprice' values in the test set is approximately 2105.0146. Lower MAE values indicate better model performance. In summary, the model's performance is moderate, with the R-squared indicating that the model explains a substantial portion of the variance in 'unitprice.' However, the Mean Squared Error and Mean Absolute Error values suggest that there is room for improvement, and the model may benefit from further refinement or feature engineering.

## **Clustering**

**Clustering-1**







**Interpretation:**

**Data Preparation:**

The code selects two features, 'unitprice' and 'quantity', from the DataFrame 'df'. Missing values in the selected features are imputed using the mean value. Feature Scaling:

The features are then scaled using ‘StandardScaler’ to ensure that they have similar scales. This step is crucial for K-means clustering. K-Means Clustering:

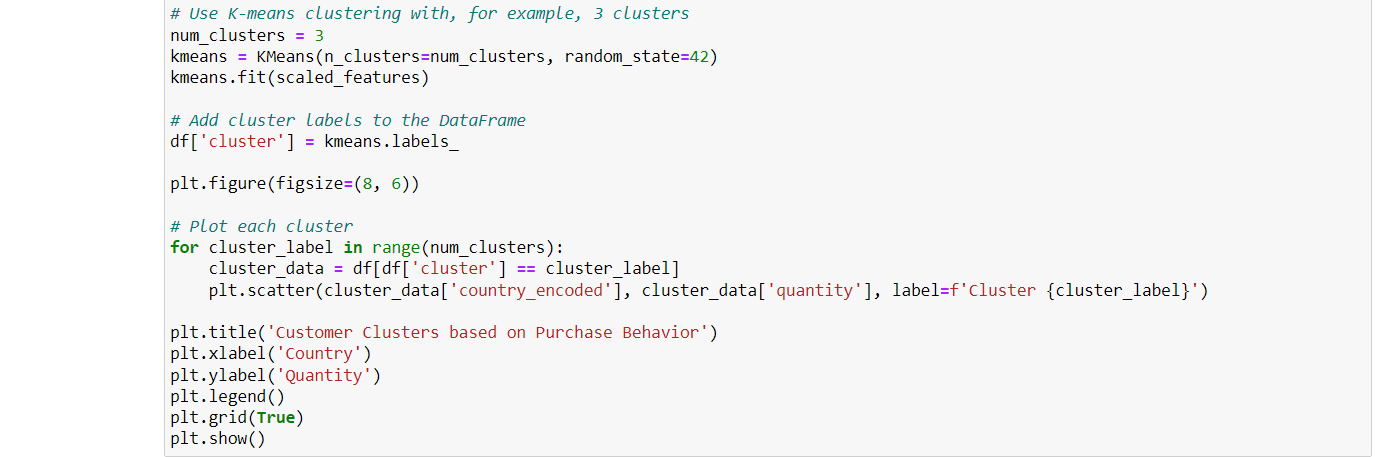
K-means clustering is applied with the specified number of clusters (num\_clusters = 3). The algorithm groups data points into clusters based on their similarity. Visualization:

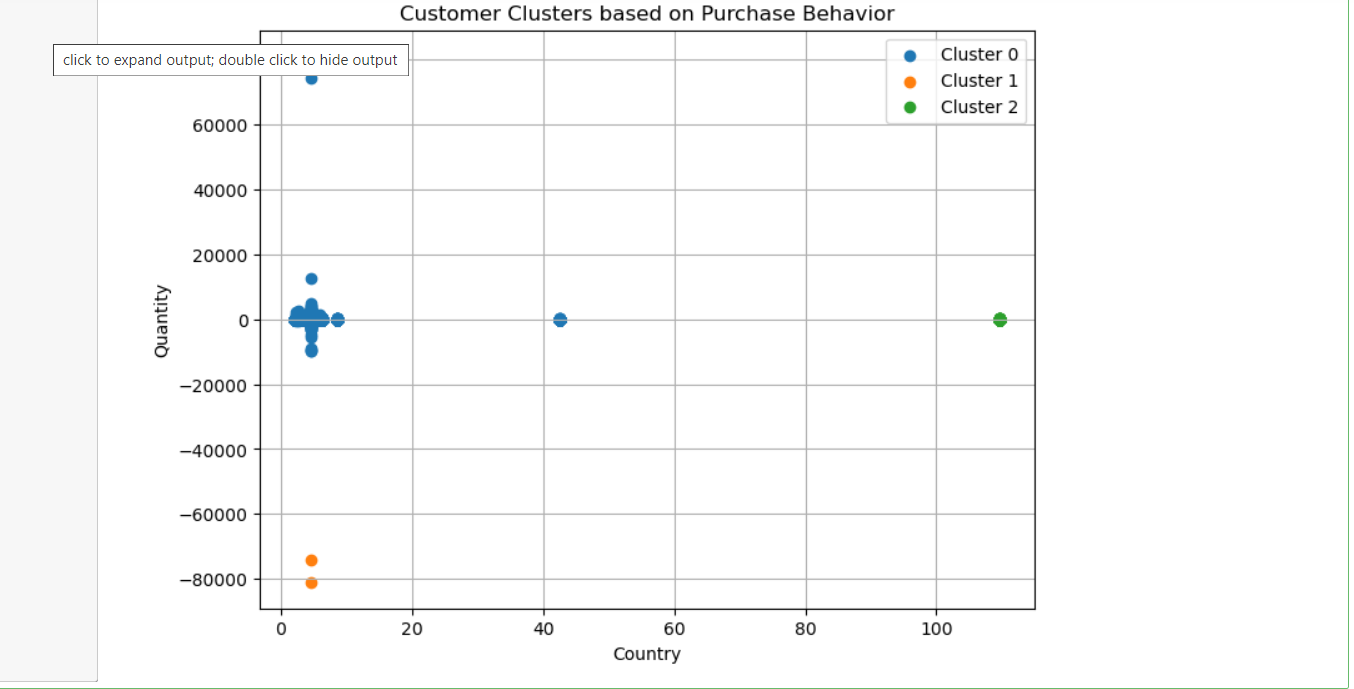
The clusters are visualized in a 2D scatter plot. Each point represents a data point in the 'unitprice' vs. 'quantity' feature space. Different clusters are color-coded, and each point is labeled with the cluster it belongs to. Interpretation of the Graph:

The graph displays clusters formed by the K-means algorithm based on the features 'unitprice' and 'quantity'. Points that are close to each other in the plot are considered similar in terms of these features. Interpretation of the clusters involves understanding the characteristics of customers within each cluster. For example, clusters might represent different purchasing patterns or behaviors based on unit price and quantity.

**Clustering-2**







**Interpretation:**

**Data Preparation:**

The code selects two features, 'country\_encoded' and 'quantity', from the DataFrame 'df'. Missing values in the selected features are imputed using the mean value. Feature Scaling:

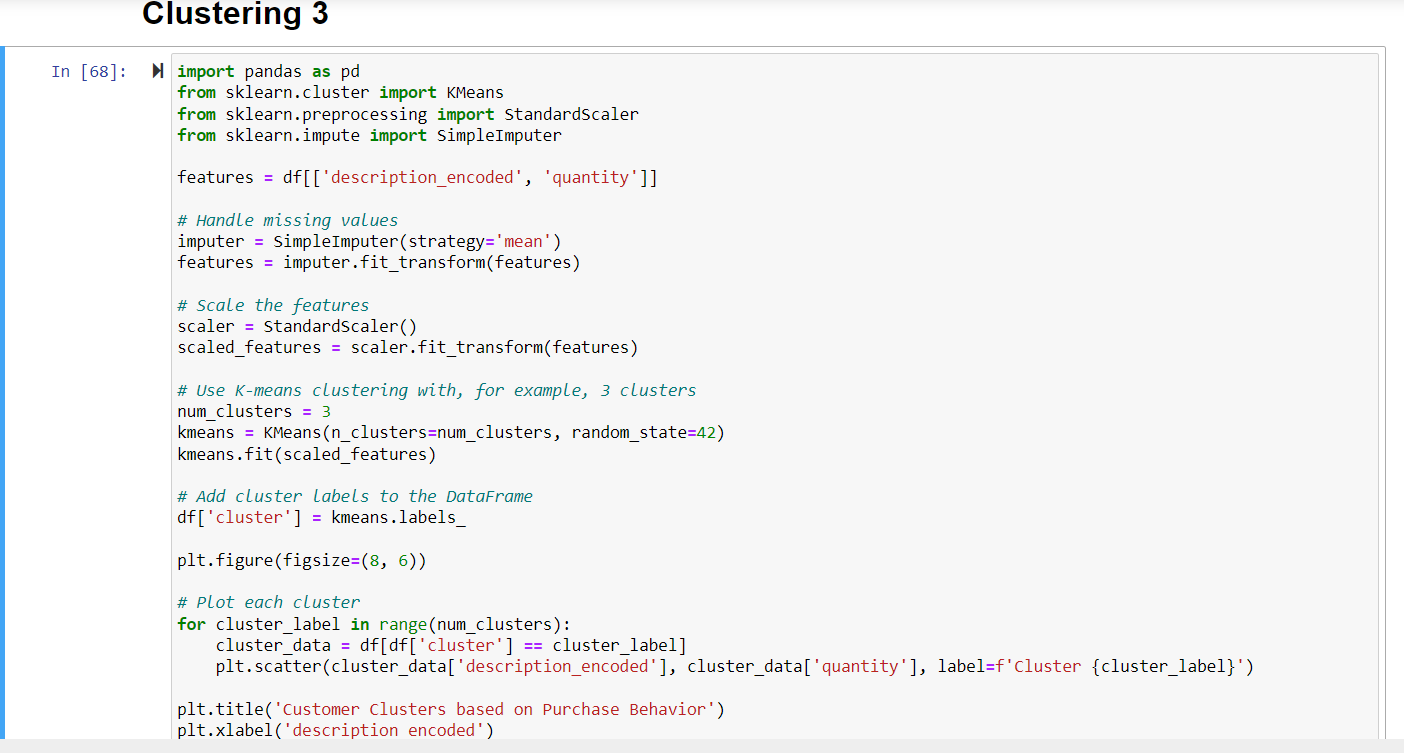
The features are then scaled using StandardScaler to ensure that they have similar scales. This step is crucial for K-means clustering. K-Means Clustering:

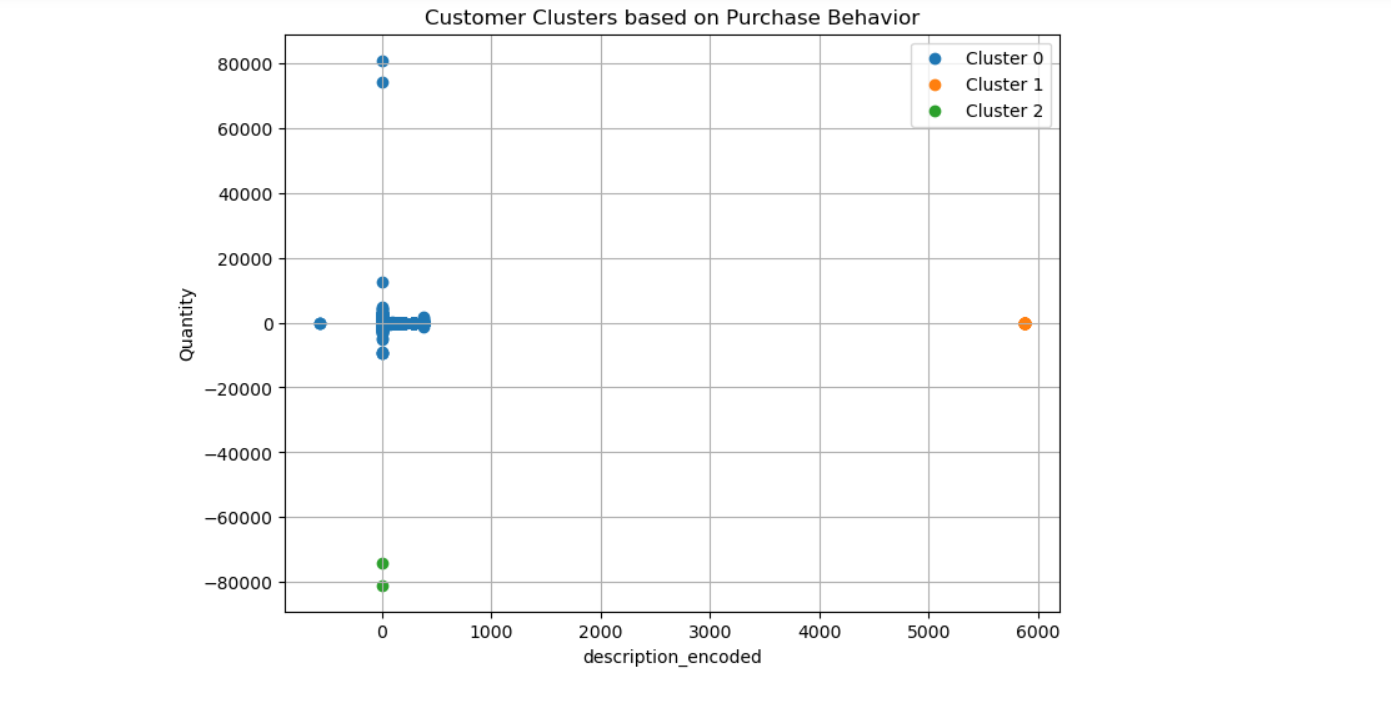
K-means clustering is applied with the specified number of clusters (num\_clusters = 3). The algorithm groups data points into clusters based on their similarity. Visualization:

The clusters are visualized in a 2D scatter plot. Each point represents a data point in the 'country\_encoded' vs. 'quantity' feature space. Different clusters are color-coded, and each point is labeled with the cluster it belongs to. Interpretation of the Graph:

The graph displays clusters formed by the K-means algorithm based on the features 'country\_encoded' and 'quantity'. Points that are close to each other in the plot are considered similar in terms of these features. Interpretation of the clusters involves understanding the characteristics of customers within each cluster. For example, clusters might represent different purchasing patterns or behaviors in different countries.

**Clustering-3**





**Interpretation:**

**Data Preparation:**

The code selects two features, 'description\_encoded' and 'quantity', from the DataFrame 'df'. Missing values in the selected features are imputed using the mean value. Feature Scaling:

The features are then scaled using StandardScaler to ensure that they have similar scales. This step is crucial for K-means clustering. K-Means Clustering:

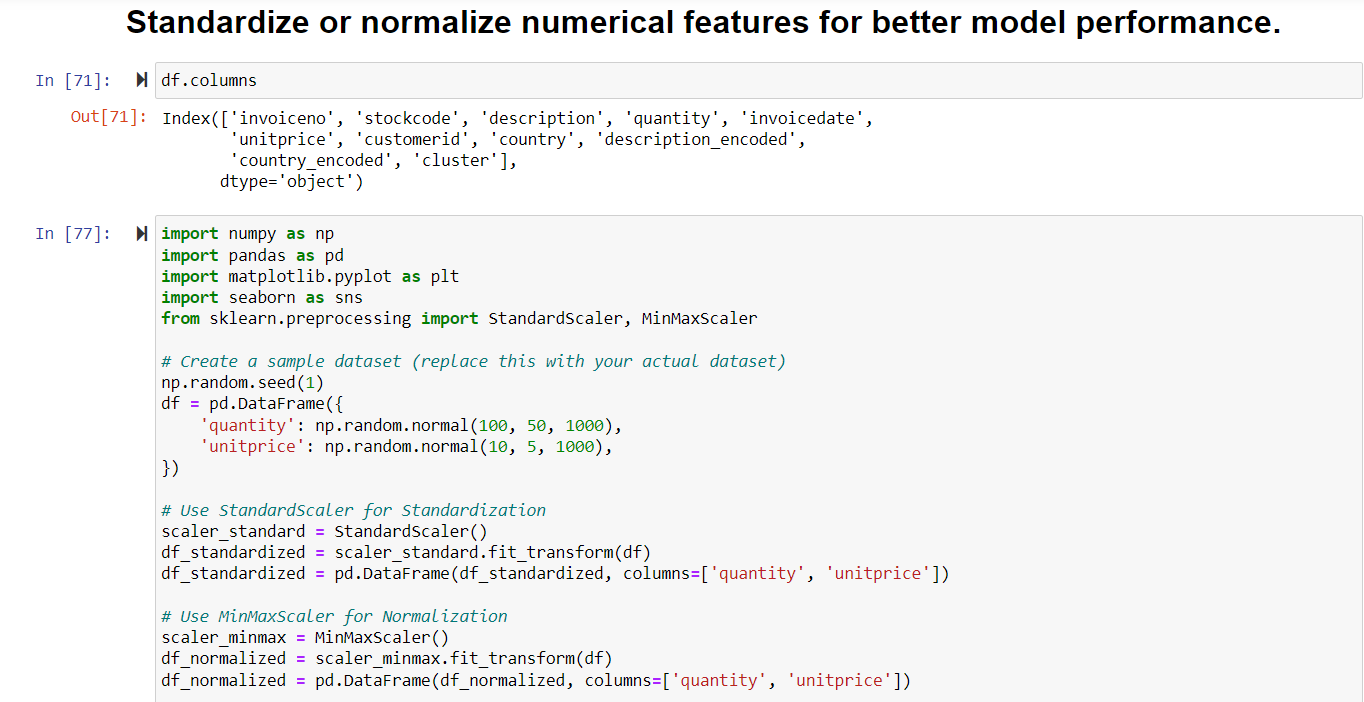
K-means clustering is applied with the specified number of clusters (num\_clusters = 3). The algorithm groups data points into clusters based on their similarity. Visualization:

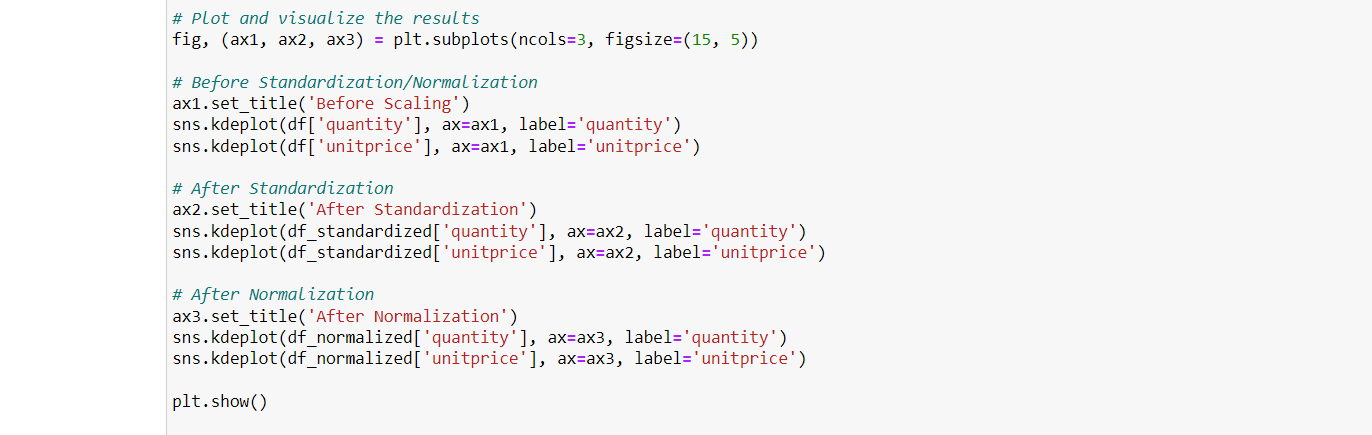
The clusters are visualized in a 2D scatter plot. Each point represents a data point in the 'description\_encoded' vs. 'quantity' feature space. Different clusters are color-coded, and each point is labeled with the cluster it belongs to. Interpretation of the Graph:

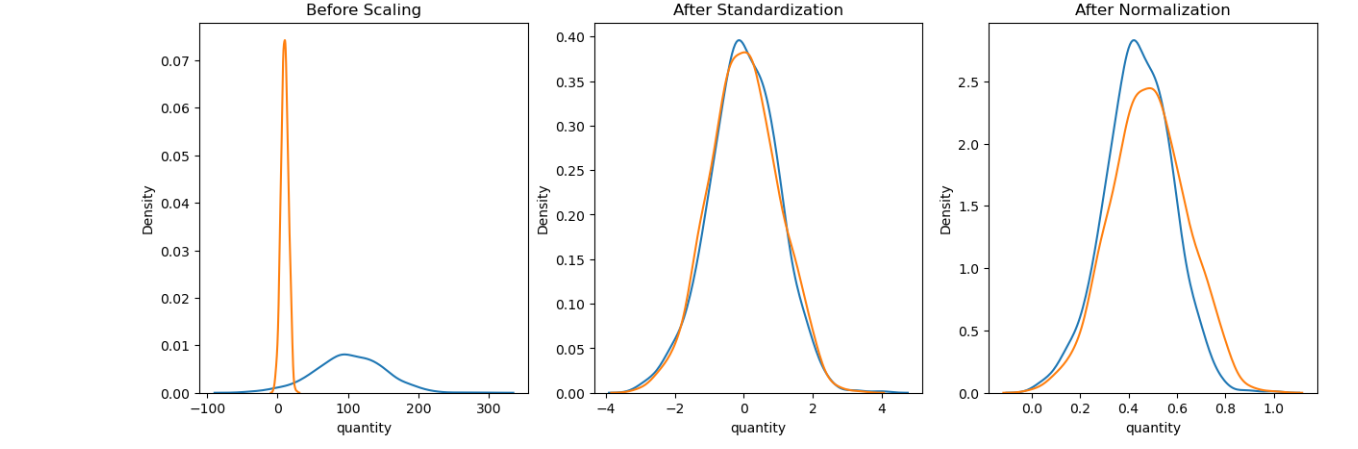
The graph displays clusters formed by the K-means algorithm based on the features 'description\_encoded' and 'quantity'. Points that are close to each other in the plot are considered similar in terms of these features. Interpretation of the clusters involves understanding the characteristics of customers within each cluster. For example, clusters might represent different purchasing patterns or behaviors.



## **Standardizing and Normalisation of numerical features for better model performance**







**Interpretation**

**Sample Dataset Creation**:

A synthetic dataset is created with two numerical features, 'quantity' and 'unitprice,' each containing 1000 random values.

**Standardization (Z-score normalization):**

The StandardScaler is applied to standardize the numerical features. Standardization transforms the data so that it has a mean of 0 and a standard deviation of 1.

The standardized data is stored in df\_standardized.

**Normalization (Min-Max scaling):**

The MinMaxScaler is applied to normalize the numerical features. Normalization scales the data to a specific range, typically [0, 1].

The normalized data is stored in df\_normalized.

**Visualization:**

A subplot with three columns is created to visualize the distribution of the features before and after standardization and normalization.

The first subplot displays the original distribution of 'quantity' and 'unitprice.'

The second subplot shows the distribution after standardization, and the third subplot shows the distribution after normalization.

**Interprretation of Plots:**

Before Scaling: The original distribution of 'quantity' and 'unitprice.'

After Standardization: The distributions of 'quantity' and 'unitprice' after applying standardization, where both features now have a mean of 0 and a standard deviation of 1.

After Normalization: The distributions of 'quantity' and 'unitprice' after applying normalization, where both features are scaled to the [0, 1] range.