Portfolio Project: Online Retail Exploratory Data Analysis with Python

Overview

In this project, you will step into the shoes of an entry-level data analyst at an online retail company, helping interpret real-world data to help make a key business decision.

Case Study

In this project, you will be working with transactional data from an online retail store. The dataset contains information about customer purchases, including product details, quantities, prices, and timestamps. Your task is to explore and analyze this dataset to gain insights into the store's sales trends, customer behavior, and popular products.

By conducting exploratory data analysis, you will identify patterns, outliers, and correlations in the data, allowing you to make data-driven decisions and recommendations to optimize the store's operations and improve customer satisfaction. Through visualizations and statistical analysis, you will uncover key trends, such as the busiest sales months, best-selling products, and the store's most valuable customers. Ultimately, this project aims to provide actionable insights that can drive strategic business decisions and enhance the store's overall performance in the competitive online retail market.

Project Objectives

1. Describe data to answer key questions to uncover insights
2. Gain valuable insights that will help improve online retail performance
3. Provide analytic insights and data-driven recommendations

Dataset

The dataset you will be working with is the "Online Retail" dataset. It contains transactional data of an online retail store from 2010 to 2011. The dataset is available as a .xlsx file named `Online Retail.xlsx`. This data file is already included in the Coursera Jupyter Notebook environment, however if you are working off-platform it can also be downloaded here (https://archive.ics.uci.edu/ml/machine-learning-databases/00352/Online%20Retail.xlsx).

The dataset contains the following columns:

- InvoiceNo: Invoice number of the transaction
- StockCode: Unique code of the product
- Description: Description of the product
- Quantity: Quantity of the product in the transaction
- InvoiceDate: Date and time of the transaction
- UnitPrice: Unit price of the product
- CustomerID: Unique identifier of the customer
- Country: Country where the transaction occurred

Tasks

You may explore this dataset in any way you would like - however if you'd like some help getting started, here are a few ideas:

1. Load the dataset into a Pandas DataFrame and display the first few rows to get an overview of the data.
2. Perform data cleaning by handling missing values, if any, and removing any redundant or unnecessary columns.
3. Explore the basic statistics of the dataset, including measures of central tendency and dispersion.
4. Perform data visualization to gain insights into the dataset. Generate appropriate plots, such as histograms, scatter plots, or bar plots, to visualize different aspects of the data.
5. Analyze the sales trends over time. Identify the busiest months and days of the week in terms of sales.
6. Explore the top-selling products and countries based on the quantity sold.
7. Identify any outliers or anomalies in the dataset and discuss their potential impact on the analysis.
8. Draw conclusions and summarize your findings from the exploratory data analysis.

Task 1: Load the Data

In [1]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [2]:
```
data = pd.read_excel("Online Retail.xlsx")
```

In [3]:
```
print(data.head())
```

```
  InvoiceNo StockCode                          Description  Quantity  \
0    536365    85123A   WHITE HANGING HEART T-LIGHT HOLDER         6
1    536365     71053                  WHITE METAL LANTERN         6
2    536365    84406B       CREAM CUPID HEARTS COAT HANGER         8
3    536365    84029G  KNITTED UNION FLAG HOT WATER BOTTLE         6
4    536365    84029E       RED WOOLLY HOTTIE WHITE HEART.         6

          InvoiceDate  UnitPrice  CustomerID         Country
0 2010-12-01 08:26:00       2.55     17850.0  United Kingdom
1 2010-12-01 08:26:00       3.39     17850.0  United Kingdom
2 2010-12-01 08:26:00       2.75     17850.0  United Kingdom
3 2010-12-01 08:26:00       3.39     17850.0  United Kingdom
4 2010-12-01 08:26:00       3.39     17850.0  United Kingdom
```

In [4]:
```
print(data.tail())
```

```
       InvoiceNo StockCode                       Description  Quantity  \
541904    581587     22613      PACK OF 20 SPACEBOY NAPKINS        12
541905    581587     22899     CHILDREN'S APRON DOLLY GIRL          6
541906    581587     23254    CHILDRENS CUTLERY DOLLY GIRL          4
541907    581587     23255  CHILDRENS CUTLERY CIRCUS PARADE         4
541908    581587     22138    BAKING SET 9 PIECE RETROSPOT          3

               InvoiceDate  UnitPrice  CustomerID Country
541904 2011-12-09 12:50:00       0.85     12680.0  France
541905 2011-12-09 12:50:00       2.10     12680.0  France
541906 2011-12-09 12:50:00       4.15     12680.0  France
541907 2011-12-09 12:50:00       4.15     12680.0  France
541908 2011-12-09 12:50:00       4.95     12680.0  France
```

Task 2: Performing data cleaning

In [5]:
```
# data count
print(data.count())
```

```
InvoiceNo      541909
StockCode      541909
Description    540455
Quantity       541909
InvoiceDate    541909
UnitPrice      541909
CustomerID     406829
Country        541909
dtype: int64
```

In [6]:
```
# detecting missing value
print(data.isnull().sum())
```

```
InvoiceNo          0
StockCode          0
Description     1454
Quantity           0
InvoiceDate        0
UnitPrice          0
CustomerID     135080
Country            0
dtype: int64
```

In [7]:
```
# removing rows with missing values
data_clean = data.dropna()
```

In [8]:
```
# printing data count again
print(data.count())
```

```
InvoiceNo      541909
StockCode      541909
Description    540455
Quantity       541909
InvoiceDate    541909
UnitPrice      541909
CustomerID     406829
Country        541909
dtype: int64
```

In [9]:
```
# Drop column with any missing values
data_clean = data.dropna(axis = 1)
```

In [10]:
```
print(data.count())
```

```
InvoiceNo      541909
StockCode      541909
Description    540455
Quantity       541909
InvoiceDate    541909
UnitPrice      541909
CustomerID     406829
Country        541909
dtype: int64
```

Task 3: Find central tendency and dispersion

In [11]:
```
summary = data.describe()
print(summary)
```

```
            Quantity      UnitPrice     CustomerID
count  541909.000000  541909.000000  406829.000000
mean        9.552250       4.611114   15287.690570
std       218.081158      96.759853    1713.600303
min    -80995.000000  -11062.060000   12346.000000
25%         1.000000       1.250000   13953.000000
50%         3.000000       2.080000   15152.000000
75%        10.000000       4.130000   16791.000000
max     80995.000000   38970.000000   18287.000000
```

Central tendency - Mean, Median and Mode

In [12]:
```
# Columns : Quantity, UnitPrice, InvoiceNo, CustomerID
# Calculate mean for Quantity column
mean_Quantity = data["Quantity"].mean()
print(mean_Quantity)

# Calculating mean for UnitPrice column
mean_UnitPrice = data["UnitPrice"].mean()
print(mean_UnitPrice)
```

```
9.55224954743324
4.611113626083471
```

In [13]:
```
# Columns : Quantity, UnitPrice, InvoiceNo, CustomerID
# Calculate median for Quantity column
median_Quantity = data["Quantity"].median()
print(median_Quantity)

# Calculating median for UnitPrice column
median_UnitPrice = data["UnitPrice"].median()
print(median_UnitPrice)
```

```
3.0
2.08
```

In [14]:
```
# Columns : Quantity, UnitPrice, InvoiceNo, CustomerID
# Calculate mode for Quantity column
mode_Quantity = data["Quantity"].mode().iloc[0]
print(mode_Quantity)

# Calculating mode for UnitPrice column
mode_UnitPrice = data["UnitPrice"].mode().iloc[0]
print(mode_UnitPrice)
```

```
1
1.25
```

Dispersion-Quantity, UnitPrice

In [15]:
```
# Calculate range for Quantity column
range_Quantity = data["Quantity"].max() - data["Quantity"].min()
print(range_Quantity)

# Calculating range for UnitPrice column
range_UnitPrice = data["UnitPrice"].max() - data["UnitPrice"].min()
print(range_UnitPrice)
```

```
161990
50032.06
```

In [16]:
```
# Calculate variance for Quantity column
variance_Quantity = data["Quantity"].var()
print(variance_Quantity)

# Calculating variance for UnitPrice column
variance_UnitPrice = data["UnitPrice"].var()
print(variance_UnitPrice)
```

```
47559.39140913822
9362.469164424467
```

In [17]:
```
# Calculate standard deviation for Quantity column
std_Quantity = data["Quantity"].std()
print(std_Quantity)

# Calculating standard deviation for UnitPrice column
std_UnitPrice = data["UnitPrice"].std()
print(std_UnitPrice)
```

```
218.08115784986612
96.75985306119716
```

In [18]:
```
# Calculating quartiles for Quantity columns
q1 = data["Quantity"].quantile(0.25)
q2 = data["Quantity"].quantile(0.50)
q3 = data["Quantity"].quantile(0.75)

# Calculating quartiles range
iqr = q3 - q1
print(iqr)
```

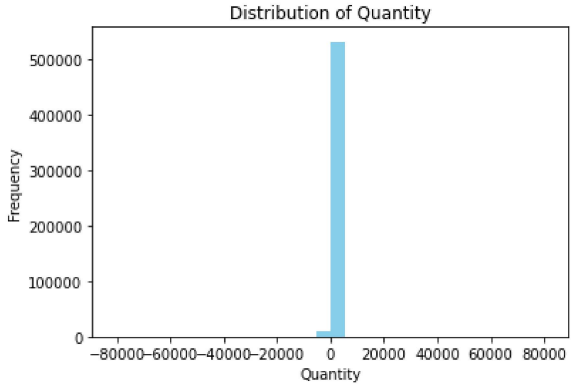
```
9.0
```

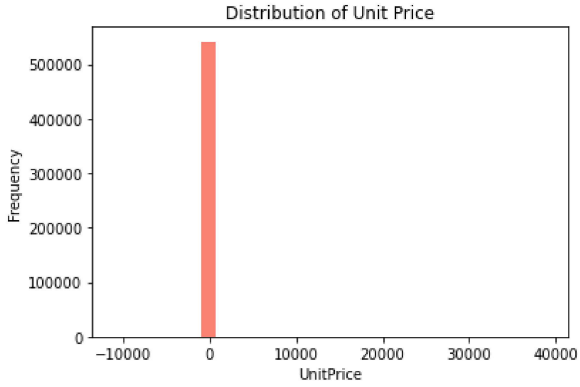
In [19]:
```
# Calculating quartiles for UnitPrice columns
q1 = data["UnitPrice"].quantile(0.25)
q2 = data["UnitPrice"].quantile(0.50)
q3 = data["UnitPrice"].quantile(0.75)

# Calculating quartiles range
iqr = q3 - q1
print(iqr)
```

```
2.88
```

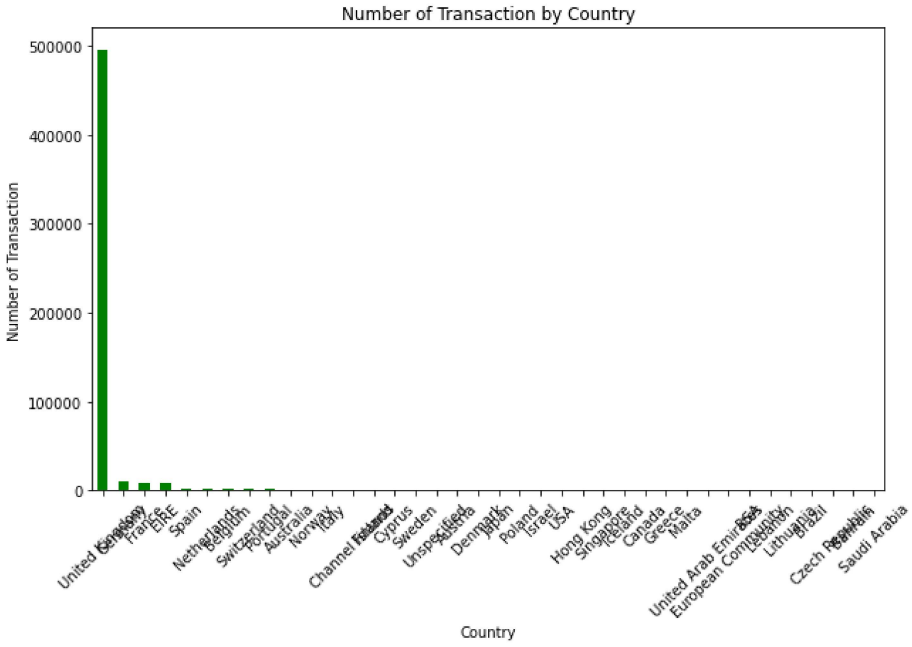
Task 4: Perform data Visualization to gain insight into the dataset

In [20]:
```
# Histogram

# Histogram for Quantity
plt.hist(data['Quantity'], bins=30, color='skyblue')
plt.xlabel('Quantity')
plt.ylabel('Frequency')
plt.title('Distribution of Quantity')
plt.show()

# Histogram for UnitPrice
plt.hist(data['UnitPrice'], bins=30, color='salmon')
plt.xlabel('UnitPrice')
plt.ylabel('Frequency')
plt.title('Distribution of Unit Price')
plt.show()
```

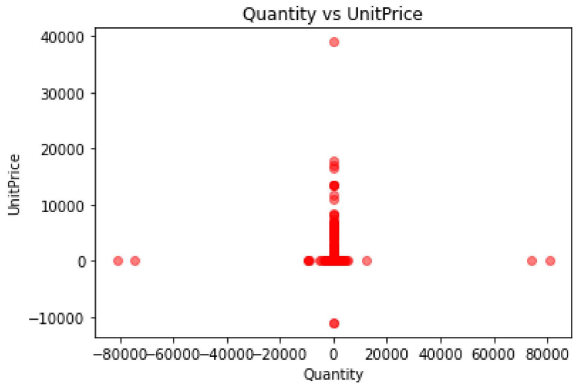




In [21]:
```
# Bar Plot

# Bar plot for Country
country_counts = data['Country'].value_counts()
plt.figure(figsize=(10,6))
country_counts.plot(kind='bar', color='green')
plt.xlabel('Country')
plt.ylabel('Number of Transaction')
plt.title('Number of Transaction by Country')
plt.xticks(rotation=45)
plt.show()
```



In [22]:
```
# Scatter plot

# Scatter plot for Quantity vs UnitPrice
plt.scatter(data['Quantity'], data['UnitPrice'], color='red', alpha=0.5)
plt.xlabel('Quantity')
plt.ylabel('UnitPrice')
plt.title('Quantity vs UnitPrice')
plt.show()
```



Task 5: Identify the busiest months and days of the week in term of sales

In [23]:
```
# Extract month and day of week
data['Month'] = data['InvoiceDate'].dt.month
data['DayofWeek'] = data['InvoiceDate'].dt.dayofweek

# Calculate monthly sales
monthly_sales = data.groupby('Month')['Quantity'].sum()

# Identify the busiest month
busiest_month = monthly_sales.idxmax()

# Calculate sales by day of week
daily_sales = data.groupby('DayofWeek')['Quantity'].sum()

# Identify the busiest day of the week ( 0 = Monday. 6 = Sunday)
busiest_day_of_week = daily_sales.idxmax()

# Print the busiest month and day of the week
print("Busiest Month: ",busiest_month)
print("Busiest Day of the Week: ",busiest_day_of_week)
```

```
Busiest Month:  11
Busiest Day of the Week:  3
```

Task 6: Explore the top-selling products and countries based on the quantity

In [24]:
```
print(data.head())
```

```
  InvoiceNo StockCode                          Description  Quantity  \
0    536365    85123A   WHITE HANGING HEART T-LIGHT HOLDER         6
1    536365     71053                  WHITE METAL LANTERN         6
2    536365    84406B       CREAM CUPID HEARTS COAT HANGER         8
3    536365    84029G  KNITTED UNION FLAG HOT WATER BOTTLE         6
4    536365    84029E       RED WOOLLY HOTTIE WHITE HEART.         6

          InvoiceDate  UnitPrice  CustomerID         Country  Month  DayofWeek
0 2010-12-01 08:26:00       2.55     17850.0  United Kingdom     12          2
1 2010-12-01 08:26:00       3.39     17850.0  United Kingdom     12          2
2 2010-12-01 08:26:00       2.75     17850.0  United Kingdom     12          2
3 2010-12-01 08:26:00       3.39     17850.0  United Kingdom     12          2
4 2010-12-01 08:26:00       3.39     17850.0  United Kingdom     12          2
```

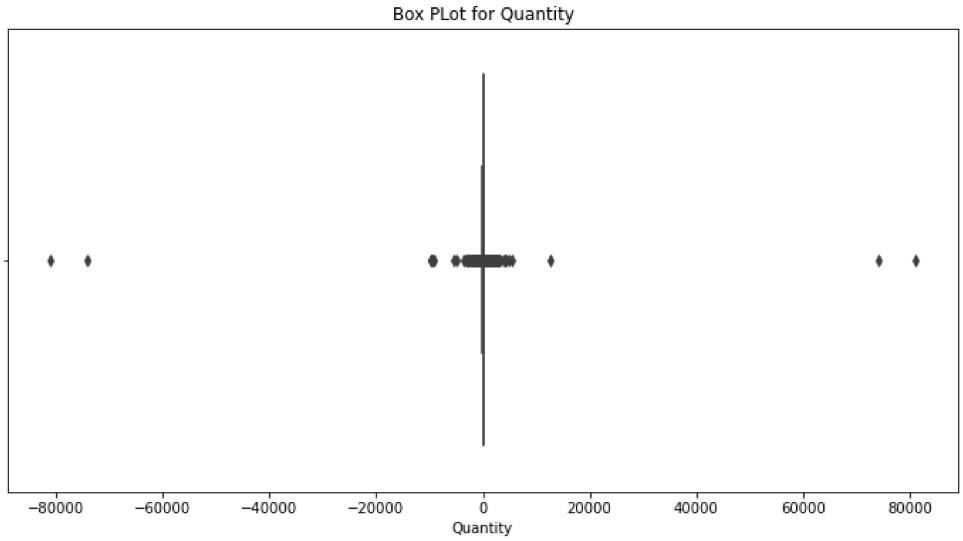
In [26]:
```
# Find the top selling products
top_selling_products = data.groupby('Description')['Quantity'].sum().sort_values(ascending=False)
top_selling_product = top_selling_products.idxmax()

# Print the top-selling products
print('Top Selling Products: ')
print(top_selling_products)
print(f"\nTop Product : {top_selling_product}")
print()

# FInd the top selling countries
top_selling_countries = data.groupby("Description")["Country"].sum().sort_values(ascending=False)

# Print the top-selling countries
print("Top-Sellimg Countries: ")
print(top_selling_countries.head(5))
```

```
Top Selling Products:
Description
WORLD WAR 2 GLIDERS ASSTD DESIGNS    53847
JUMBO BAG RED RETROSPOT              47363
ASSORTED COLOUR BIRD ORNAMENT        36381
POPCORN HOLDER                       36334
PACK OF 72 RETROSPOT CAKE CASES      36039
                                     ...
Damaged                               -7540
Printing smudges/thrown away          -9058
check                                -12030
Unsaleable, destroyed.               -15644
printing smudges/thrown away         -19200
Name: Quantity, Length: 4223, dtype: int64

Top Product : WORLD WAR 2 GLIDERS ASSTD DESIGNS

Top-Sellimg Countries:
Description
HOT WATER BOTTLE KEEP CALM         UnspecifiedUnited KingdomUnited KingdomUnited ...
PINK ROUND COMPACT MIRROR          United KingdomUnited KingdomUnspecifiedUnited ...
VINTAGE GLASS T-LIGHT HOLDER       United KingdomUnited KingdomUnited KingdomUnsp...
PARTY CHARMS 50 PIECES             United KingdomUnited KingdomUnited KingdomUnsp...
ASSORTED SANSKRIT MINI NOTEBOOK    United KingdomUnited KingdomUnited KingdomUnit...
Name: Country, dtype: object
```

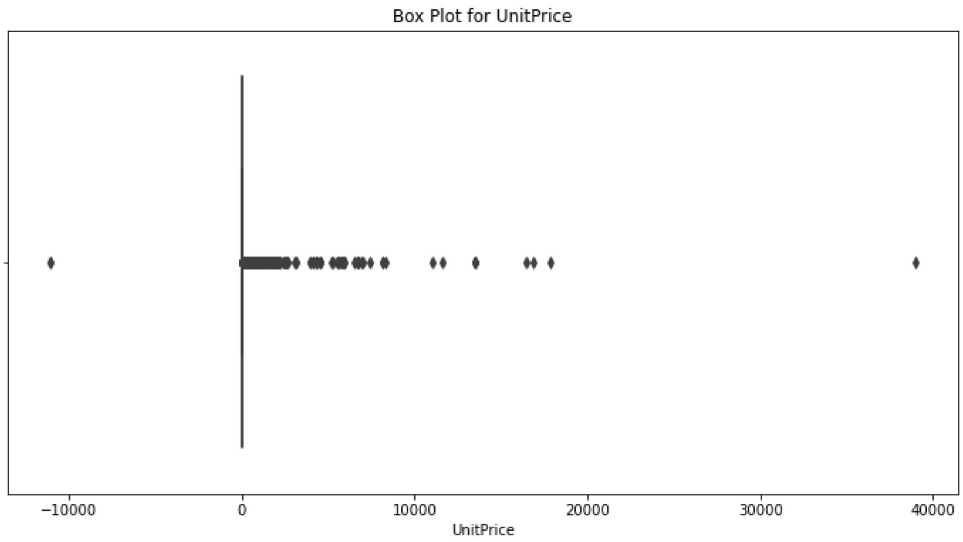
Task 7: Identify any outliers or anomalies int the dataset and discuss their potential impact impact on the analysis.

In [28]:
```
# Create box plots for Quantity and UnitPrice

plt.figure(figsize=(12,6))
x = data['Quantity']
sns.boxplot(x)
plt.title('Box PLot for Quantity')
plt.show()

plt.figure(figsize=(12,6))
y = data['UnitPrice']
sns.boxplot(y)
plt.title('Box Plot for UnitPrice')
plt.show()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a k
eyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments withou
t an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```




```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a k
eyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments withou
t an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



In [30]:
```
# calculate IQR for Quantity
Q1 = data['Quantity'].quantile(0.25)
Q3 = data['Quantity'].quantile(0.75)
IQR = Q3 - Q1

# Define Lower and upper bounds to identify outliers
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 - 1.5 * IQR

# Identity outlier in Quantity
outliers_quantity = data[(data['Quantity'] < lower_bound) | (data['Quantity'] > upper_bound)]

# Calculate IQR for UnitPrice
Q1 = data['UnitPrice'].quantile(0.25)
Q3 = data['UnitPrice'].quantile(0.75)
IQR = Q3 - Q1

# Define Lower and upper bounds to identify outliers
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 - 1.5 * IQR

# Identify outliers in UnitPrice
outliers_unitprice = data[(data['UnitPrice'] < lower_bound) | (data['UnitPrice'] > upper_bound)]
print(f"Outliers in Quantity : {outliers_quantity}")
print(f"\n Outliers in UnitPrice : {outliers_unitprice}")
```

```
Outliers in Quantity :        InvoiceNo StockCode                          Description  Quantity  \
0         536365    85123A   WHITE HANGING HEART T-LIGHT HOLDER         6
1         536365     71053                  WHITE METAL LANTERN         6
2         536365    84406B       CREAM CUPID HEARTS COAT HANGER         8
3         536365    84029G  KNITTED UNION FLAG HOT WATER BOTTLE         6
4         536365    84029E       RED WOOLLY HOTTIE WHITE HEART.         6
...          ...       ...                                  ...       ...
541904    581587     22613          PACK OF 20 SPACEBOY NAPKINS        12
541905    581587     22899         CHILDREN'S APRON DOLLY GIRL          6
541906    581587     23254        CHILDRENS CUTLERY DOLLY GIRL          4
541907    581587     23255      CHILDRENS CUTLERY CIRCUS PARADE         4
541908    581587     22138        BAKING SET 9 PIECE RETROSPOT          3

               InvoiceDate  UnitPrice  CustomerID         Country  Month  \
0      2010-12-01 08:26:00       2.55     17850.0  United Kingdom     12
1      2010-12-01 08:26:00       3.39     17850.0  United Kingdom     12
2      2010-12-01 08:26:00       2.75     17850.0  United Kingdom     12
3      2010-12-01 08:26:00       3.39     17850.0  United Kingdom     12
4      2010-12-01 08:26:00       3.39     17850.0  United Kingdom     12
```

Task 8: Draw conclusions and summarize your findings from exploratory data analysis.

In [33]:
```


alysis makes easy to find the mean, median and modes using the pandas library.")
 data in different types of graphs")
 also be performed")


g values and removed the missing values.")
ime we have come to know about busiest days of week which was in months - November and in Days of the week it was 3 - Wed
```

```
>>>Conclusion<<<
Using Python for Exploratory data analysis makes easy to find the mean, median and modes using the pandas library.
It becomes very easy to visulize the data in different types of graphs
Standard deviation and variation can also be performed


>>>Summarize<<<
FIrst we have checked for any missing values and removed the missing values.
By analyzing the sales trends over time we have come to know about busiest days of week which was in months - November
and in Days of the week it was 3 - Wednesday
```

In [ ]: