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.

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

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

data=pd.read_csv('OnlineRetail.csv')

data.head()
data.drop(['Unnamed: 0'], axis=1, inplace=True)

data.describe()
```

```
            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
```

```
data.isnull().sum()
```

```
InvoiceNo           0
StockCode           0
```

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

```
data.shape
```

```
(540455, 8)
```

Data Cleaning

```
data.CustomerID.fillna(value='mode',inplace=True)
data.isnull().sum()
```

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

```
data.dropna(inplace=True)
```

```
data.isnull().sum()
```

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

```
data.shape
```

```
(540455, 8)
```

```
data.describe()
```

```
            Quantity      UnitPrice
count  540455.000000  540455.000000
mean        9.603129       4.623519
std       218.007598      96.889628
```

```
min    -80995.000000  -11062.060000
25%         1.000000       1.250000
50%         3.000000       2.080000
75%        10.000000       4.130000
max     80995.000000   38970.000000
```

Fomatting Data

```
data['InvoiceDate']=pd.to_datetime(data['InvoiceDate']).dt.date
```

```
data['InvoiceYear']=pd.to_datetime(data['InvoiceDate']).dt.year
```

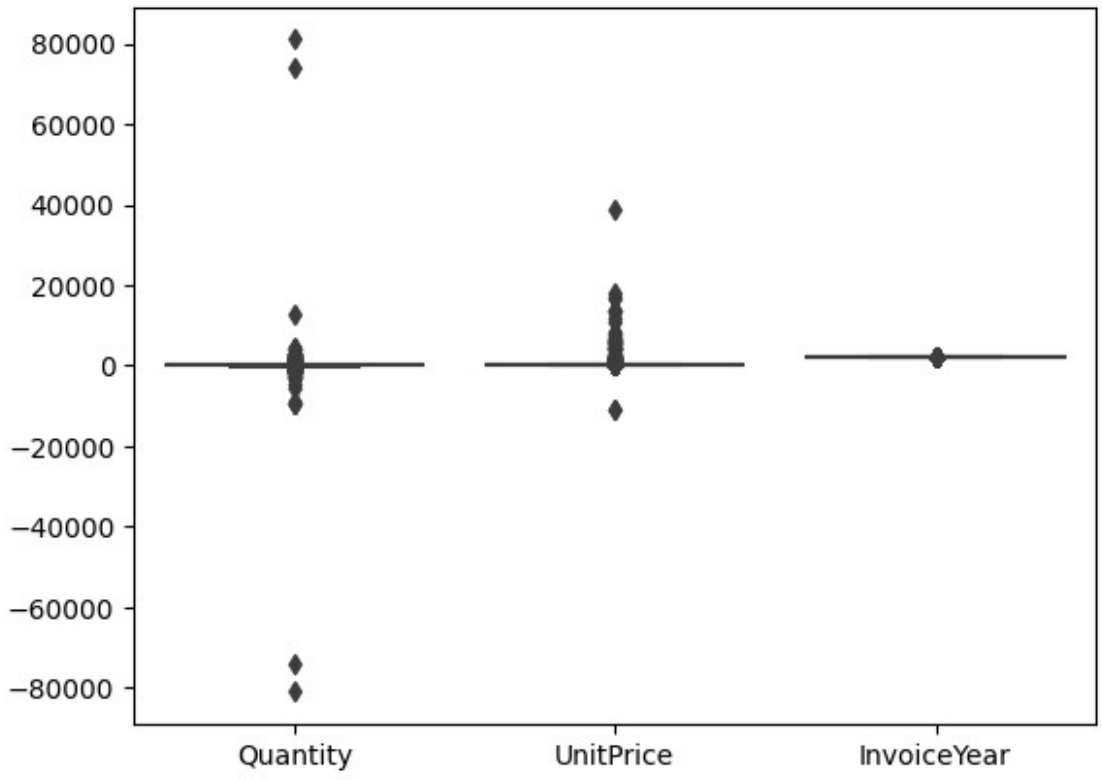
Analyzation

```
data.columns
```

```
Index(['InvoiceNo', 'StockCode', 'Description', 'Quantity',
'InvoiceDate',
       'UnitPrice', 'CustomerID', 'Country', 'InvoiceYear'],
      dtype='object')
```

```
sns.boxplot(data)
```

```
<Axes: >
```

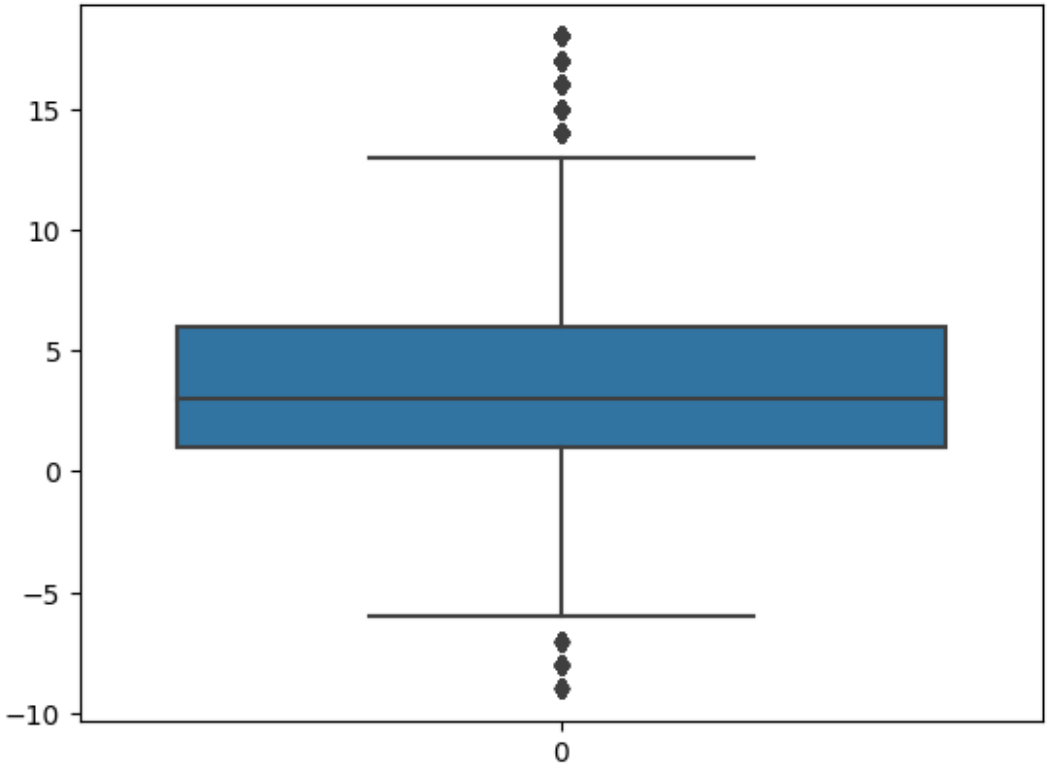


```
                                        Handling Outliers with Z-Score
```

1) Quantity

```
data['zscore_1']=(data.Quantity-data.Quantity.mean())/
data.Quantity.std()
data= data[(data.zscore_1>-3)&(data.zscore_1<3)]
sns.boxplot(data['Quantity'])
```

```
<Axes: >
```



2) Unit price

```
data['zscore_2']=(data.UnitPrice-data.UnitPrice.mean())/
data.UnitPrice.std()
data= data[(data.zscore_2>-3)&(data.zscore_2<3)]
sns.boxplot(data['UnitPrice'])
```

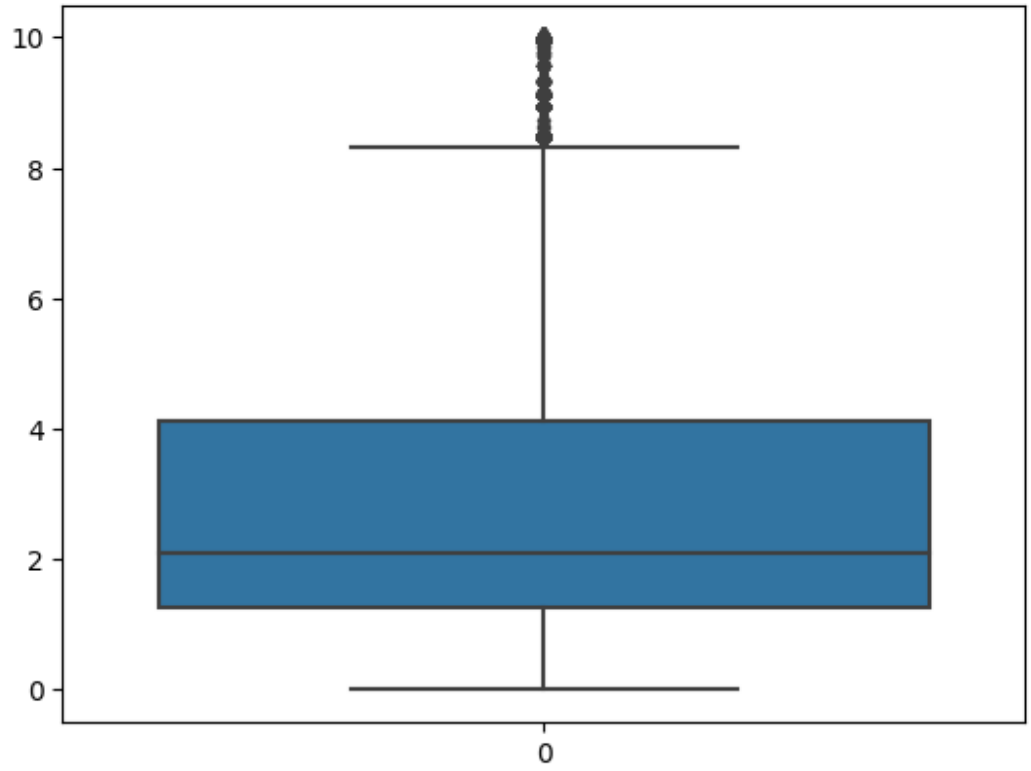
```
C:\Users\aclog\AppData\Local\Temp\ipykernel_2532\2412544767.py:1:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
```

```
returning-a-view-versus-a-copy

data['zscore_2']=(data.UnitPrice-data.UnitPrice.mean())/data.UnitPrice
.std()

<Axes: >
```



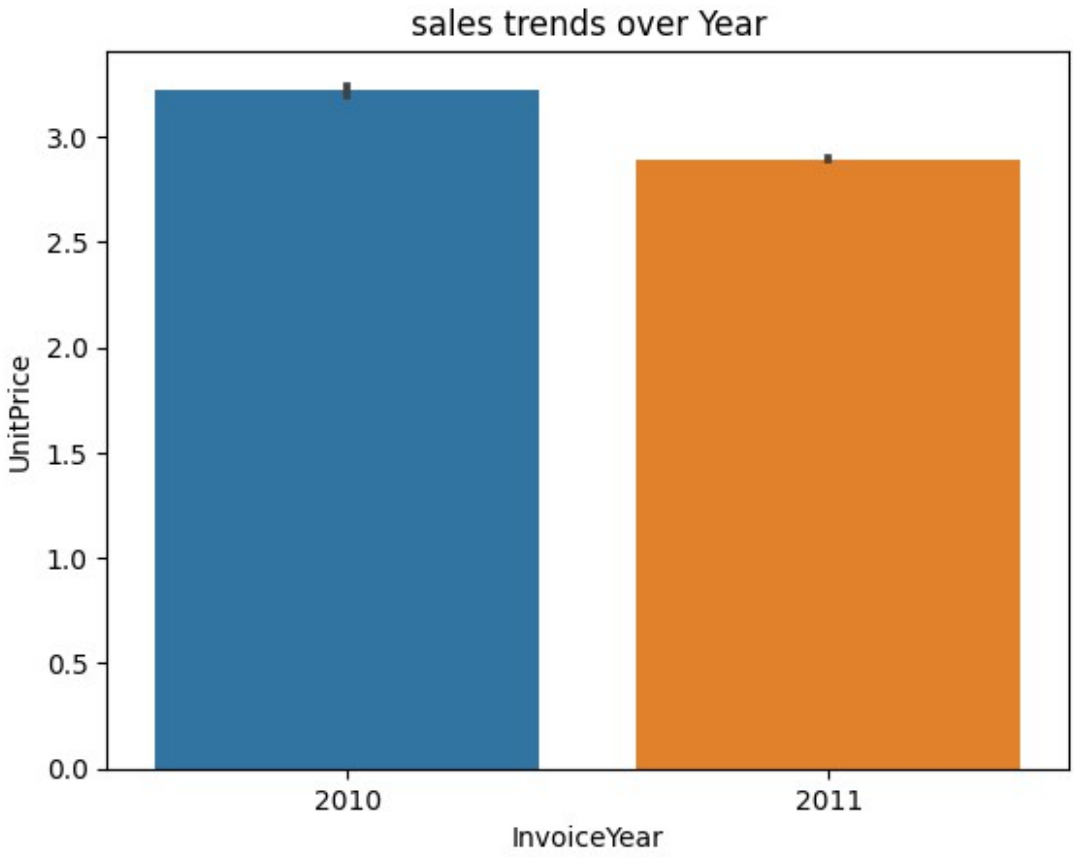

```
data['InvoiceMonth']=pd.to_datetime(data['InvoiceDate']).dt.month
data['InvoiceMonth']

0         12
1         12
2         12
3         12
4         12
         ..
541904    12
541905    12
541906    12
541907    12
541908    12
Name: InvoiceMonth, Length: 451320, dtype: int32
```

1) sales trends over time

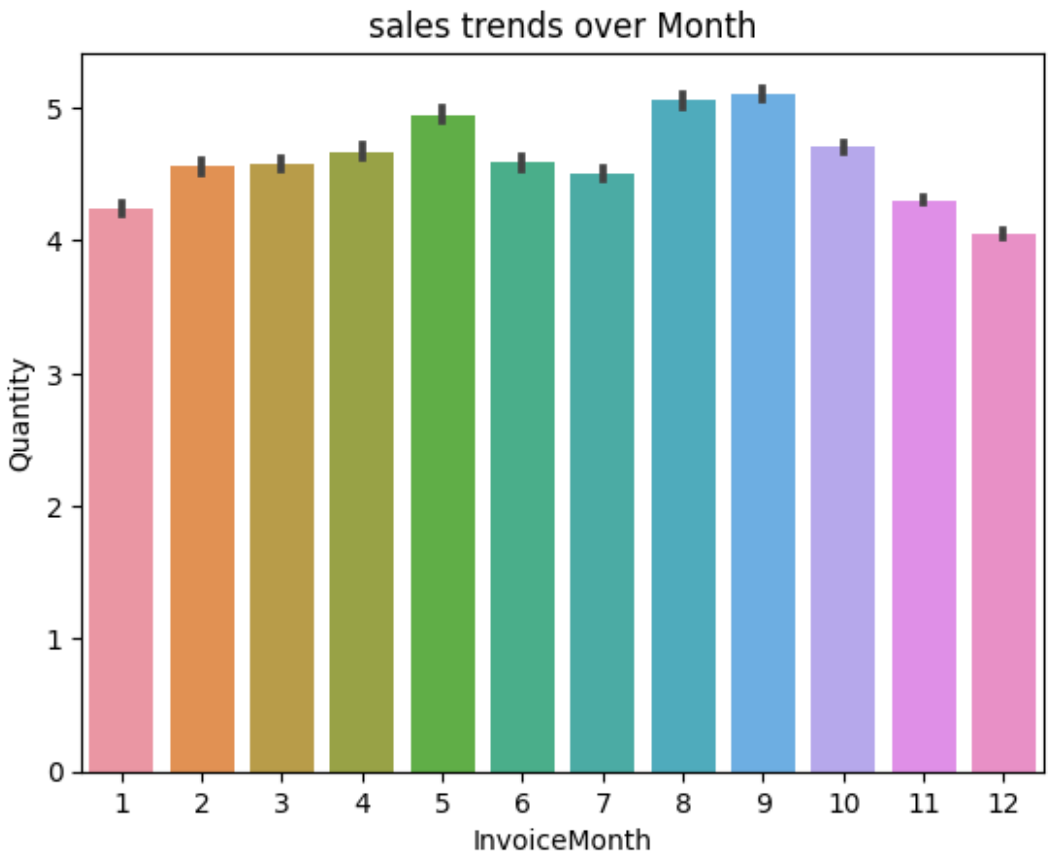
```
sns.barplot(data=data, x='InvoiceYear',y='UnitPrice').set_title('sales trends over Year')
```

```
Text(0.5, 1.0, 'sales trends over Year')
```



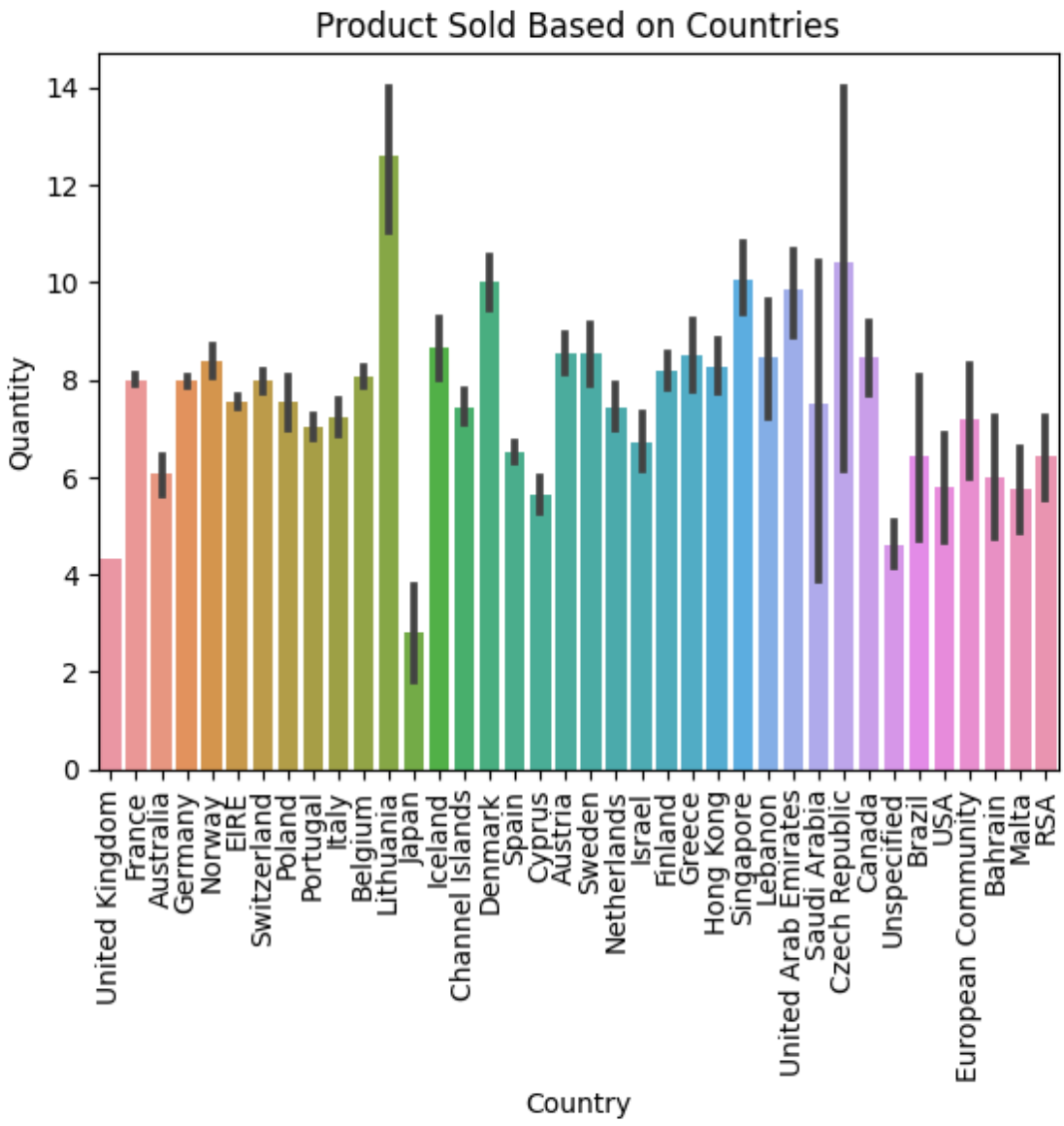

```
sns.barplot(data=data, x='InvoiceMonth',
y='Quantity').set_title('sales trends over Month')
```

```
Text(0.5, 1.0, 'sales trends over Month')
```

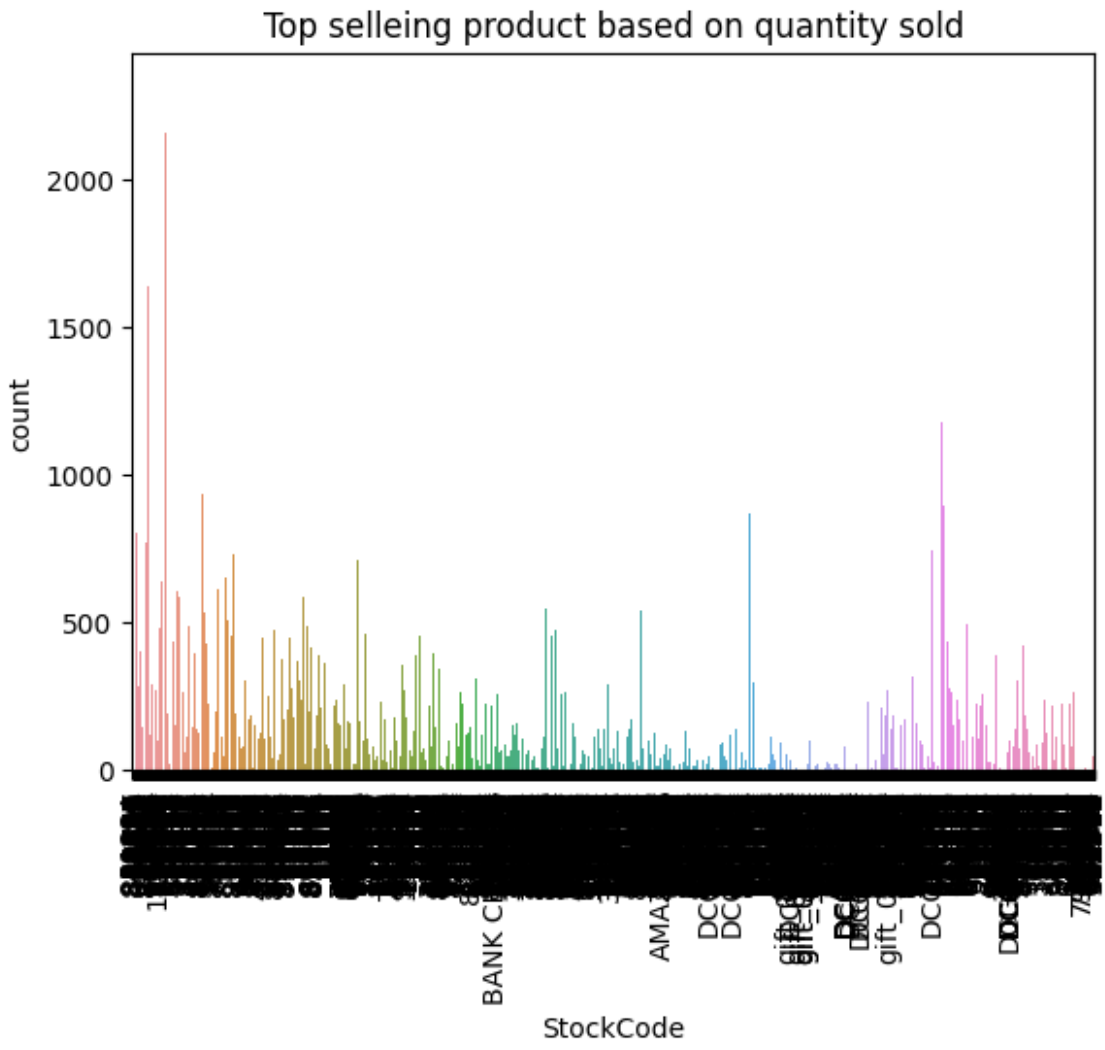


2) Product Sold Based on Countries

```
sns.barplot(data=data, x='Country',y='Quantity').set_title('Product
Sold Based on Countries')
plt.xticks(rotation=90)
plt.show()
```




```
sns.countplot(data=data,x='StockCode').set_title('Top selleing product
based on quantity sold')
plt.xticks(rotation=90)
plt.show()
```


Top selleing product based on quantity sold
2000
1500
1000
500
0
count
StockCode