In [2]:
```
import pandas as pd
import random
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [3]:
```
df = pd.read_excel(r"C:\Users\chuck\Documents\Datasets\Online Retail.xlsx")
print(df)
```

```
       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
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
...                    ...        ...         ...             ...
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

[541909 rows x 8 columns]
```

In [4]:
```
df.head()
```

Out[4]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Cou
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	Un Kingc
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	Un Kingc
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	Un Kingc
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	Un Kingc
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	Un Kingc

In [5]:
```
#Remove null data

df.dropna(inplace=True)
```

In [6]:
```
#Remove duplicate data

df.drop_duplicates(inplace=True)
```

In [7]:
```
# Identify missing data

missing_data = df.isnull().sum()
print('Missing Data: \n', missing_data)
```

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

In [8]:
```
df.fillna(value={'CustomerID': 'Unknown'}, inplace=True)
```

In [9]:
```
#Correct data types

df['InvoiceDate'] = pd.to_datetime(df['InvoiceDate'])
```

In [10]:
```
#Rename columns

df.rename(columns={'InvoiceNo': 'Invoice_Number', 'StockCode': 'Stock_Code', 'Invoi
```

In [11]:
```
df.head()
```

Out[11]:

	Invoice_Number	Stock_Code	Description	Quantity	Invoice_Date	Unit_Price	Customer_
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	1785
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	1785
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	1785
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	1785
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	1785

In [12]:
```
#Remove outliers

upper_limit = df['Unit_Price'].quantile(0.95)
df = df[df['Unit_Price'] < upper_limit]
```

In [13]:
```
#Create new column Total Price of each invoice line

df['Total_Price'] = df['Quantity'] * df['Unit_Price']
df.head()
```

Out[13]:

	Invoice_Number	Stock_Code	Description	Quantity	Invoice_Date	Unit_Price	Customer_
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	1785
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	1785
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	1785
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	1785
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	1785

In [14]:
```
#Find range of years
df['Invoice_Date'] = pd.to_datetime(df['Invoice_Date'], errors='coerce')
df = df.dropna(subset=['Invoice_Date'])
df['Year'] = df['Invoice_Date'].dt.year
start_year = df['Year'].min()
end_year = df['Year'].max()
print(f"The data covers the years from {start_year} to {end_year}.")
```

```
The data covers the years from 2010 to 2011.
```

In [18]:
```
# Calculate the unique numbers in 'Invoice_Number'
unique_invoice_numbers = df['Invoice_Number'].nunique()

# Print the result
print(f'The number of unique invoice numbers is: {unique_invoice_numbers}')
```

```
The number of unique invoice numbers is: 21267
```

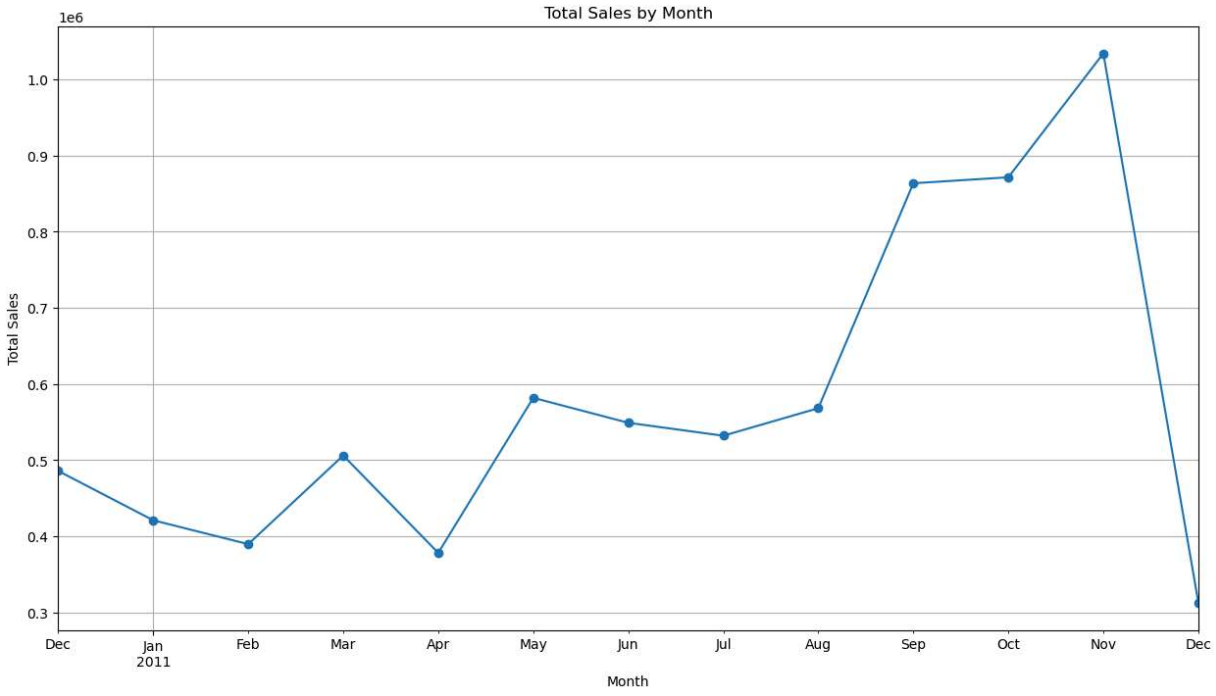
In [19]:
```
# Calculate the unique numbers in 'Stock Code'
unique_stock_code_numbers = df['Stock_Code'].nunique()

# Print the result
print(f'The number of unique Stock Code numbers is: {unique_stock_code_numbers}')
```

```
The number of unique Stock Code numbers is: 3506
```

In [16]:
```
# Calculate sales by month
df['Invoice_Date'] = pd.to_datetime(df['Invoice_Date'], errors='coerce')
df = df.dropna(subset=['Invoice_Date'])
df['YearMonth'] = df['Invoice_Date'].dt.to_period('M')
monthly_sales = df.groupby('YearMonth')['Total_Price'].sum().round(2)
print(monthly_sales)
```

```
YearMonth
2010-12     485667.13
2011-01     420911.86
2011-02     389567.90
2011-03     505915.58
2011-04     378092.60
2011-05     581693.27
2011-06     548913.58
2011-07     531920.66
2011-08     568062.70
2011-09     863461.58
2011-10     871284.49
2011-11    1033994.30
2011-12     313013.37
Freq: M, Name: Total_Price, dtype: float64
```

In [17]:
```
# Plotting the monthly sales
plt.figure(figsize=(15, 8))
monthly_sales.plot(kind='line', marker='o')
plt.title('Total Sales by Month')
plt.xlabel('Month')
plt.ylabel('Total Sales')
plt.grid(True)
plt.show()
```



In [43]:
```
#Total Sales by Country
df['Total_Price'] = df['Total_Price'].round(2)
```

```
country_sales = df.groupby('Country')['Total_Price'].sum().reset_index()
country_sales = country_sales.sort_values(by='Total_Price', ascending=False)
country_sales
```

Out[43]:

	Country	Total_Price
35	United Kingdom	6163088.93
23	Netherlands	270086.24
10	EIRE	213826.32
14	Germany	178670.51
13	France	166715.15
0	Australia	127043.05
32	Switzerland	47091.65
30	Spain	45620.53
31	Sweden	34987.91
19	Japan	34218.89
3	Belgium	31821.41
24	Norway	28525.56
26	Portugal	25083.91
12	Finland	17123.94
9	Denmark	16580.34
6	Channel Islands	16538.44
18	Italy	13864.54
7	Cyprus	9785.71
29	Singapore	8220.29
1	Austria	7824.77
25	Poland	5815.64
17	Israel	5736.55
16	Iceland	4078.95
15	Greece	3665.47
5	Canada	2899.74
36	Unspecified	2330.02
21	Lithuania	1661.06
33	USA	1576.37
22	Malta	1556.67
34	United Arab Emirates	1442.68

	Country	Total_Price
11	European Community	1038.65
20	Lebanon	984.68
4	Brazil	968.40
27	RSA	783.86
8	Czech Republic	667.72
2	Bahrain	443.30
28	Saudi Arabia	131.17

In [45]:
```
# Calculate total sales by customer ID
customer_sales = df.groupby('Customer_ID')['Total_Price'].sum().reset_index()
customer_sales = customer_sales.sort_values(by='Total_Price', ascending=False)
customer_sales
```

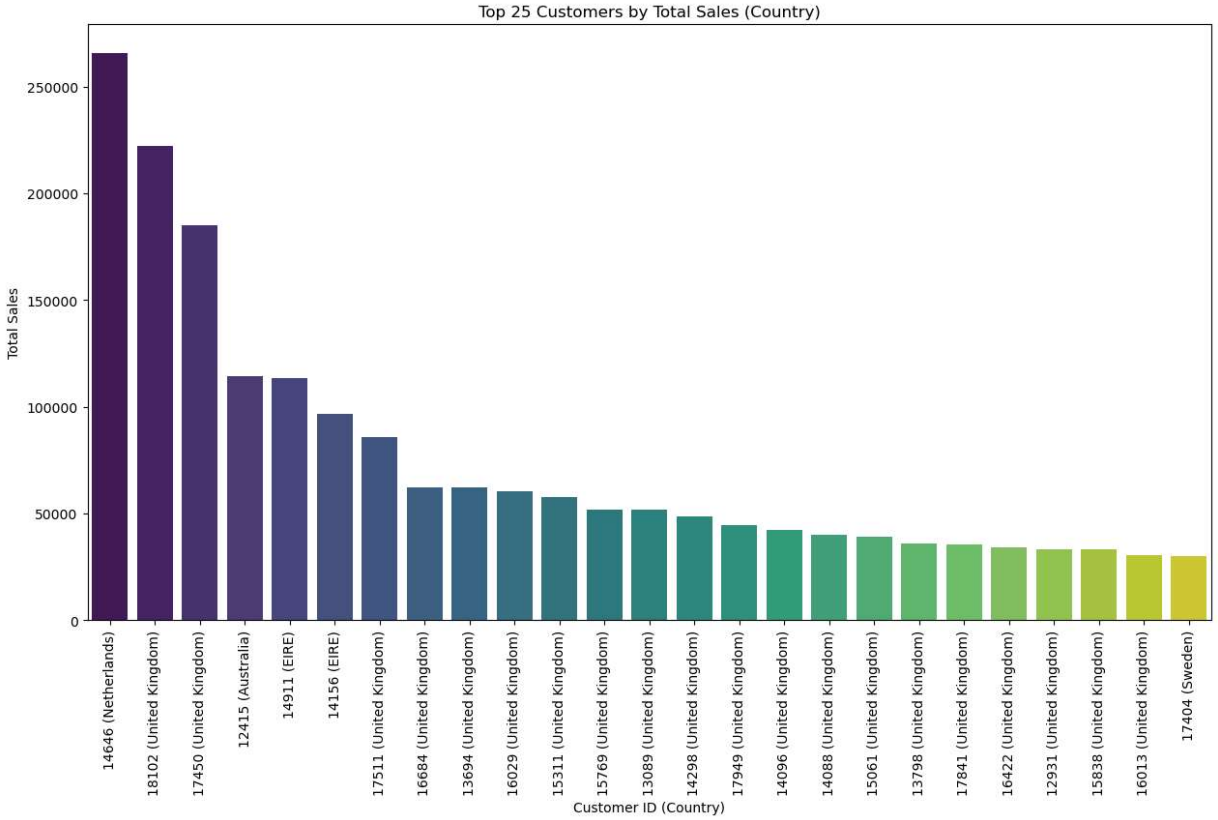
Out[45]:

	Customer_ID	Total_Price
1693	14646.0	265728.62
4203	18102.0	222135.17
3732	17450.0	185026.63
55	12415.0	114243.58
1885	14911.0	113638.49
...	...	...
2874	16252.0	-251.14
3843	17603.0	-414.93
3225	16742.0	-464.90
2562	15823.0	-811.86
1374	14213.0	-1192.20

4340 rows × 2 columns

In [58]:
```
# Calculate total sales by customer ID
customer_sales = df.groupby(['Customer_ID', 'Country'])['Total_Price'].sum().sort_v

# Calculate top 25 customers by total sales
top_25_customers = customer_sales.head(25).reset_index()

# Remove decimal points from Customer_ID
top_25_customers['Customer_ID'] = top_25_customers['Customer_ID'].astype(int)

# Combine Customer_ID and Country for x-axis labels
top_25_customers['Customer_Country'] = top_25_customers['Customer_ID'].astype(str)
```

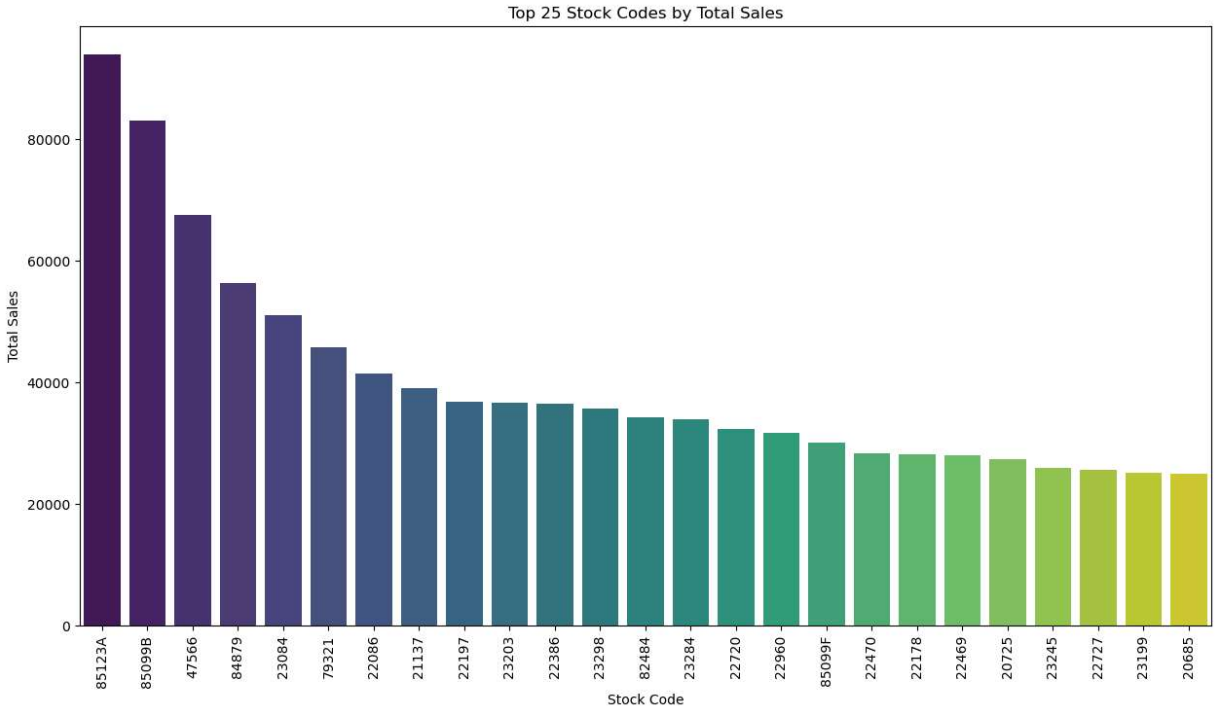
```
# Plotting the top 25 customers
plt.figure(figsize=(15, 8))
sns.barplot(x='Customer_Country', y='Total_Price', data=top_25_customers, palette="
plt.title('Top 25 Customers by Total Sales (Country)')
plt.xlabel('Customer ID (Country)')
plt.ylabel('Total Sales')
plt.xticks(rotation=90)
plt.show()
```



In [26]:
```
# Calculate sales by Stock_Code (top 25)

sales_by_stock_code = df.groupby(['Stock_Code', 'Description'])['Total_Price'].sum(
sales_by_stock_code = sales_by_stock_code.sort_values(by='Total_Price', ascending=F
top_25_stock_code = sales_by_stock_code.head(25)

print(top_25_stock_code)
```

	Stock_Code	Description	Total_Price
3434	85123A	WHITE HANGING HEART T-LIGHT HOLDER	93767.80
3427	85099B	JUMBO BAG RED RETROSPOT	83056.52
2492	47566	PARTY BUNTING	67498.95
2689	84879	ASSORTED COLOUR BIRD ORNAMENT	56331.91
1842	23084	RABBIT NIGHT LIGHT	51042.84
2559	79321	CHILLI LIGHTS	45728.51
905	22086	PAPER CHAIN KIT 50'S CHRISTMAS	41423.78
307	21137	BLACK RECORD COVER FRAME	38990.63
1183	22386	JUMBO BAG PINK POLKADOT	36437.78
2091	23298	SPOTTY BUNTING	35026.74
2569	82484	WOOD BLACK BOARD ANT WHITE FINISH	34210.50
2075	23284	DOORMAT KEEP CALM AND COME IN	33930.37
1492	22720	SET OF 3 CAKE TINS PANTRY DESIGN	32368.14
1710	22960	JAM MAKING SET WITH JARS	31611.72
3429	85099F	JUMBO BAG STRAWBERRY	30120.83
1261	22470	HEART OF WICKER LARGE	28290.15
991	22178	VICTORIAN GLASS HANGING T-LIGHT	28111.71
1260	22469	HEART OF WICKER SMALL	28009.14
101	20725	LUNCH BAG RED RETROSPOT	27050.40
2038	23245	SET OF 3 REGENCY CAKE TINS	25961.35
1498	22727	ALARM CLOCK BAKELIKE RED	25602.70
1972	23199	JUMBO BAG APPLES	25117.05
77	20685	DOORMAT RED RETROSPOT	24872.05
1654	22910	PAPER CHAIN KIT VINTAGE CHRISTMAS	24708.32
1407	22629	SPACEBOY LUNCH BOX	24580.44

In [25]:
```
# Plotting the top 25 Stock_Code
plt.figure(figsize=(15, 8))
sns.barplot(x='Stock_Code', y='Total_Price', data=top_25_stock_code, palette="virid
plt.title('Top 25 Stock Codes by Total Sales')
plt.xlabel('Stock Code')
plt.ylabel('Total Sales')
plt.xticks(rotation=90)
plt.show()
```



In [33]:

```
# Remove decimal from Customer_ID
df['Customer_ID'] = df['Customer_ID'].astype(int)
# Ensure Invoice_Date is in datetime format
df['Invoice_Date'] = pd.to_datetime(df['Invoice_Date'])

# Calculate Recency
current_date = df['Invoice_Date'].max() + pd.DateOffset(1)
df['Recency'] = (current_date - df['Invoice_Date']).dt.days

# Calculate Frequency and Monetary
rfm = df.groupby('Customer_ID').agg({
    'Invoice_Date': lambda x: (current_date - x.max()).days,
    'Invoice_Number': 'nunique',
    'Total_Price': 'sum'
}).reset_index()

# Rename columns
rfm.columns = ['Customer_ID', 'Recency', 'Frequency', 'Monetary']

# Define RFM score function
def rfm_score(x, p, d):
    if x <= p[d][0.25]:
        return 1
    elif x <= p[d][0.50]:
        return 2
    elif x <= p[d][0.75]:
        return 3
    else:
        return 4

# Calculate RFM scores
rfm['R_Score'] = rfm['Recency'].apply(rfm_score, args=(rfm[['Recency']].quantile([0
rfm['F_Score'] = rfm['Frequency'].apply(rfm_score, args=(rfm[['Frequency']].quantil
rfm['M_Score'] = rfm['Monetary'].apply(rfm_score, args=(rfm[['Monetary']].quantile(
```

```
# Combine RFM scores
rfm['RFM_Score'] = rfm['R_Score'].astype(str) + rfm['F_Score'].astype(str) + rfm['M

# Display the RFM table with scores
print(rfm.head())
```

```
   Customer_ID  Recency  Frequency  Monetary  R_Score  F_Score  M_Score  \
0        12346      326          2      0.00        4        2        1
1        12347        2          7   4078.95        1        4        4
2        12348       75          4   1437.24        3        3        3
3        12349       19          1   1287.15        2        1        3
4        12350      310          1    294.40        4        1        2

  RFM_Score
0       421
1       144
2       333
3       213
4       412
```

In [ ]: