E-commerce CRM Analysis

November 19, 2024

1 CRM Analysis

Customer Relationship Management (CRM) analysis involves the systematic examination and interpretation of data related to interactions between a business and its customers. Through CRM analysis, companies evaluate customer behavior, preferences, and feedback to gain valuable insights into their needs and expectations.

1.1 Dataset Description

- **InvoiceNo:** Invoice number that consists 6 digits. If this code starts with letter 'c', it indicates a cancellation.
- **StockCode:** Product code that consists 5 digits.
- **Description:** Product name.
- **Quantity:** The quantities of each product per transaction.
- **InvoiceDate:** This represents the day and time when each transaction was generated.
- **UnitPrice:** Product price per unit.
- **CustomerID:** Customer number that consists 5 digits. Each customer has a unique customer ID.
- **Country:** Name of the country where each customer resides.

1.2 Importing Libraries and Loading Datasets

[ ]:
```
# importing required modules and packages
!pip install squarify

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as stats
import squarify as sq
import warnings
warnings.filterwarnings('ignore')

# Loading dataset
```

```
!gdown 1CmS-dDKvbTCGYlLBfUNGRi5StOP0GOLl

df = pd.read_csv('/content/Ecom_CRM_analysis.csv',encoding='latin1')
```

```
Collecting squarify
  Downloading squarify-0.4.4-py3-none-any.whl.metadata (600 bytes)
Downloading squarify-0.4.4-py3-none-any.whl (4.1 kB)
Installing collected packages: squarify
Successfully installed squarify-0.4.4
Downloading…
From: https://drive.google.com/uc?id=1CmS-dDKvbTCGYlLBfUNGRi5StOP0GOLl
To: /content/Ecom_CRM_analysis.csv
100% 45.6M/45.6M [00:00<00:00, 64.4MB/s]
```

1.3 Preprocessing Dataset

[ ]:
```
df.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  12/1/2010 8:26       2.55     17850.0  United Kingdom
1  12/1/2010 8:26       3.39     17850.0  United Kingdom
2  12/1/2010 8:26       2.75     17850.0  United Kingdom
3  12/1/2010 8:26       3.39     17850.0  United Kingdom
4  12/1/2010 8:26       3.39     17850.0  United Kingdom
```

[ ]:
```
df.shape
```

[ ]: (541909, 8)

[ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 541909 entries, 0 to 541908
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   InvoiceNo    541909 non-null  object
 1   StockCode    541909 non-null  object
 2   Description  540455 non-null  object
 3   Quantity     541909 non-null  int64
```

```
 4   InvoiceDate  541909 non-null  object
 5   UnitPrice    541909 non-null  float64
 6   CustomerID   406829 non-null  float64
 7   Country      541909 non-null  object
dtypes: float64(2), int64(1), object(5)
memory usage: 33.1+ MB
```

[ ]:
```
df[['Quantity','UnitPrice']].describe().T
```

[ ]:
```
              count      mean         std       min   25%   50%    75%  \
Quantity   541909.0  9.552250  218.081158 -80995.00  1.00  3.00  10.00
UnitPrice  541909.0  4.611114   96.759853 -11062.06  1.25  2.08   4.13

               max
Quantity   80995.0
UnitPrice  38970.0
```

[ ]:
```
# changing data types
df['InvoiceDate'] = pd.to_datetime(df['InvoiceDate'])
df['CustomerID'] = df['CustomerID'].astype(str).apply(lambda x: x[:-2])
df['Quantity'] = df['Quantity'].astype('int32')
df['UnitPrice'] = df['UnitPrice'].astype('float32')
```

[ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 541909 entries, 0 to 541908
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   InvoiceNo    541909 non-null  object
 1   StockCode    541909 non-null  object
 2   Description  540455 non-null  object
 3   Quantity     541909 non-null  int32
 4   InvoiceDate  541909 non-null  datetime64[ns]
 5   UnitPrice    541909 non-null  float32
 6   CustomerID   541909 non-null  object
 7   Country      541909 non-null  object
dtypes: datetime64[ns](1), float32(1), int32(1), object(5)
memory usage: 28.9+ MB
```

[ ]:
```
df.nunique()
```

[ ]:
```
InvoiceNo      25900
StockCode       4070
Description     4223
Quantity         722
```

```
InvoiceDate    23260
UnitPrice       1630
CustomerID      4373
Country           38
dtype: int64
```

[ ]:
```
#checking duplicates
df.duplicated().sum()
```

[ ]: 5268

[ ]:
```
#removing duplicates
df.drop_duplicates(inplace=True)
```

[ ]:
```
df.shape
```

[ ]: (536641, 8)

[ ]:
```
#checking nulls
df.isna().sum().sum()
```

[ ]: 1454

[ ]:
```
df.isna().sum()* 100 / len(df)
```

[ ]:
```
InvoiceNo      0.000000
StockCode      0.000000
Description    0.270945
Quantity       0.000000
InvoiceDate    0.000000
UnitPrice      0.000000
CustomerID     0.000000
Country        0.000000
dtype: float64
```

[ ]:
```
# Handimg Nulls
mode_desc = df.groupby('StockCode')['Description'].apply(lambda x: x.mode()[0]␣
 ↪if not x.mode().empty else 'NO DESCRIPTION')
df['Description'] = df['Description'].fillna(df['StockCode'].map(mode_desc))
```

[ ]:
```
df.isna().sum().sum()
```

[ ]: 0

[ ]:
```
df.nunique()
```

[ ]:
```
InvoiceNo      25900
StockCode       4070
Description     4224
Quantity         722
InvoiceDate    23260
UnitPrice       1630
CustomerID      4373
Country           38
dtype: int64
```

[ ]:
```
df = df[df['UnitPrice'] != 0]
```

[ ]:
```
df.shape
```

[ ]:
```
(534131, 8)
```

1.4 Exploratory Data Analysis

[ ]:
```
df['TotalPrice'] = df['Quantity'] * df['UnitPrice']
df.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  TotalPrice
0 2010-12-01 08:26:00       2.55      17850  United Kingdom   15.300000
1 2010-12-01 08:26:00       3.39      17850  United Kingdom   20.340001
2 2010-12-01 08:26:00       2.75      17850  United Kingdom   22.000000
3 2010-12-01 08:26:00       3.39      17850  United Kingdom   20.340001
4 2010-12-01 08:26:00       3.39      17850  United Kingdom   20.340001
```

[ ]:
```
df['TotalPrice'].sum()/1000000
```

[ ]:
```
9.726006907034938
```

1.4.1 Descriptive Statistics

[ ]:
```
df.describe()
```

[ ]:
```
             Quantity                    InvoiceDate      UnitPrice  \
count  534131.000000                         534131  534131.000000
mean        9.916784  2011-07-04 12:02:14.286607360       4.654426
```

```
min    -80995.000000          2010-12-01 08:26:00  -11062.059570
25%         1.000000          2011-03-28 11:36:00       1.250000
50%         3.000000          2011-07-19 15:55:00       2.100000
75%        10.000000          2011-10-18 17:10:00       4.130000
max     80995.000000          2011-12-09 12:50:00   38970.000000
std       216.451709                          NaN      97.414780

          TotalPrice
count  534131.000000
mean       18.209029
min   -168469.593821
25%         3.750000
50%         9.900000
75%        17.570000
max    168469.593821
std       381.547566
```

[ ]:
```python
df.describe(include='object')
```

[ ]:
```
       InvoiceNo StockCode                         Description CustomerID  \
count     534131    534131                              534131     534131
unique     23798      3938                                4042       4372
top       573585    85123A  WHITE HANGING HEART T-LIGHT HOLDER          n
freq        1114      2295                                2353     132567

               Country
count           534131
unique              38
top     United Kingdom
freq            487808
```

- This data is from December 2010 to December 2011.
- There are 4371 customers accross 38 countries.
- *WHITE HANGING HEART T-LIGHT HOLDER* is the most purchased product

1.4.2 Country wise analysis

[ ]:
```python
cust_country = df.groupby('Country')['CustomerID'].nunique().
 ↪sort_values(ascending=False).reset_index()
cust_country
```

[ ]:
```
           Country  CustomerID
0   United Kingdom        3950
1          Germany          95
2           France          88
3            Spain          31
4          Belgium          25
```

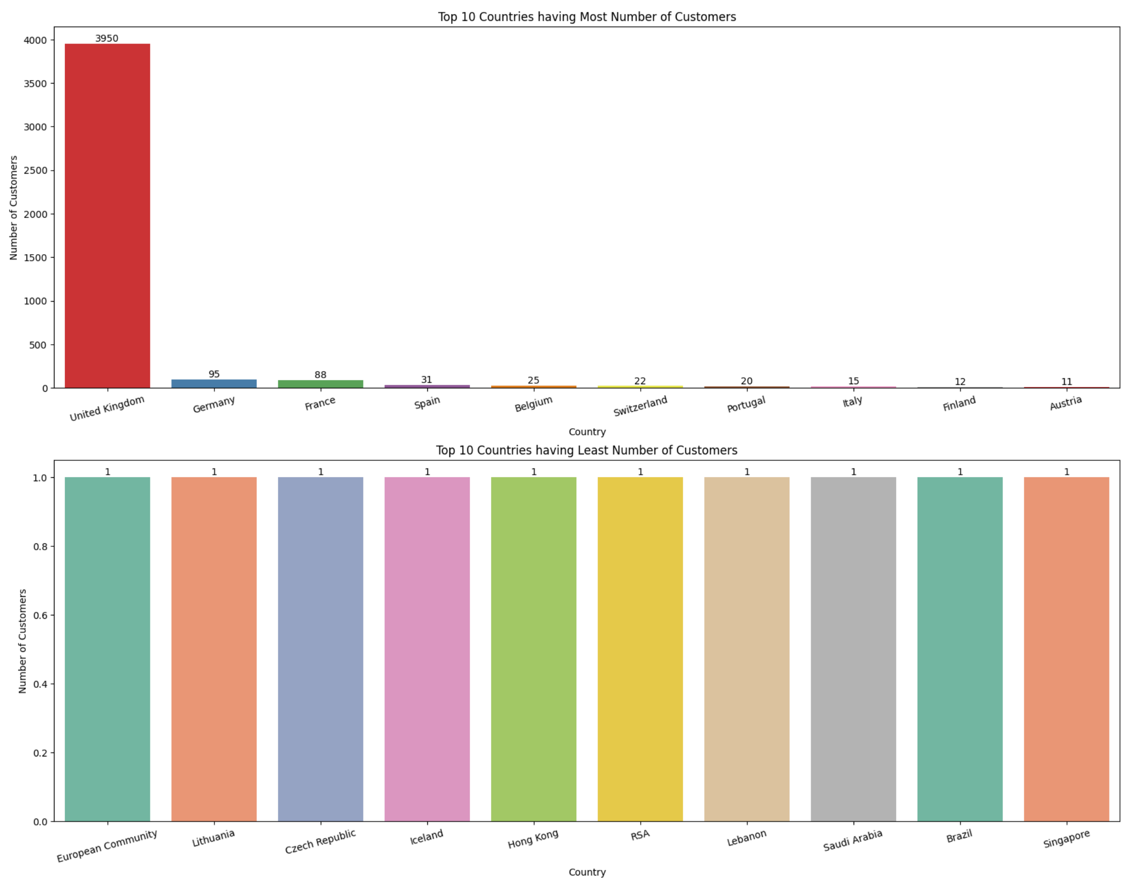
```
5              Switzerland          22
6                 Portugal          20
7                    Italy          15
8                  Finland          12
9                  Austria          11
10                  Norway          10
11             Netherlands           9
12               Australia           9
13                 Denmark           9
14         Channel Islands           9
15                  Cyprus           8
16                  Sweden           8
17                   Japan           8
18                  Poland           6
19             Unspecified           5
20                  Israel           5
21                    EIRE           4
22                     USA           4
23                  Greece           4
24                  Canada           4
25                 Bahrain           3
26                   Malta           2
27    United Arab Emirates           2
28               Singapore           1
29                  Brazil           1
30            Saudi Arabia           1
31                 Lebanon           1
32                     RSA           1
33               Hong Kong           1
34                 Iceland           1
35          Czech Republic           1
36               Lithuania           1
37      European Community           1
```

[ ]:
```
cust_country[cust_country['Country']=='United Kingdom']['CustomerID']/
 ↪cust_country['CustomerID'].sum()
```

[ ]:
```
0    0.900182
Name: CustomerID, dtype: float64
```

[ ]:
```
fig = plt.figure(figsize = (20,15))

plt.subplot(2,1,1)
plt.title('Top 10 Countries having Most Number of Customers')
plt.ylabel('Number of Customers')
plt.xticks(rotation=15)
```

```
g = sns.barplot(data=cust_country.head(10), x='Country', y='CustomerID',␣
 ↪palette='Set1')
for j in g.patches:
    plt.text(x=j.get_x()+j.get_width()/2, y=j.get_height(), s=int(j.
 ↪get_height()), ha='center', va='bottom')

plt.subplot(2,1,2)
plt.title('Top 10 Countries having Least Number of Customers')
plt.ylabel('Number of Customers')
plt.xticks(rotation=15)
g = sns.barplot(data=cust_country.tail(10)[::-1], x='Country', y='CustomerID',␣
 ↪palette='Set2')
for j in g.patches:
    plt.text(x=j.get_x()+j.get_width()/2, y=j.get_height(), s=int(j.
 ↪get_height()), ha='center', va='bottom')

plt.show()
```



- Almost 90% of the customers i.e., 3950 are from United Kingdom that means this store is in United Kingdom and the customers from various countries who came to visit UK also shopped in this.

[ ]:
```
orders_country = df.groupby('Country')['InvoiceNo'].nunique().
    ↪sort_values(ascending=False).reset_index()
orders_country
```

[ ]:
```
                 Country  InvoiceNo
0         United Kingdom      21393
1                Germany        603
2                 France        461
3                   EIRE        360
4                Belgium        119
5                  Spain        105
6            Netherlands        100
7            Switzerland         74
8               Portugal         71
9              Australia         69
10                 Italy         55
11               Finland         48
12                Sweden         46
13                Norway         40
14       Channel Islands         33
15                 Japan         28
16                Poland         24
17               Denmark         21
18                Cyprus         20
19               Austria         19
20             Hong Kong         15
21           Unspecified         13
22                 Malta         10
23             Singapore         10
24                Israel          9
25               Iceland          7
26                   USA          7
27                Greece          6
28                Canada          6
29        Czech Republic          5
30    European Community          5
31             Lithuania          4
32               Bahrain          4
33  United Arab Emirates          3
34          Saudi Arabia          2
35               Lebanon          1
36                   RSA          1
37                Brazil          1
```

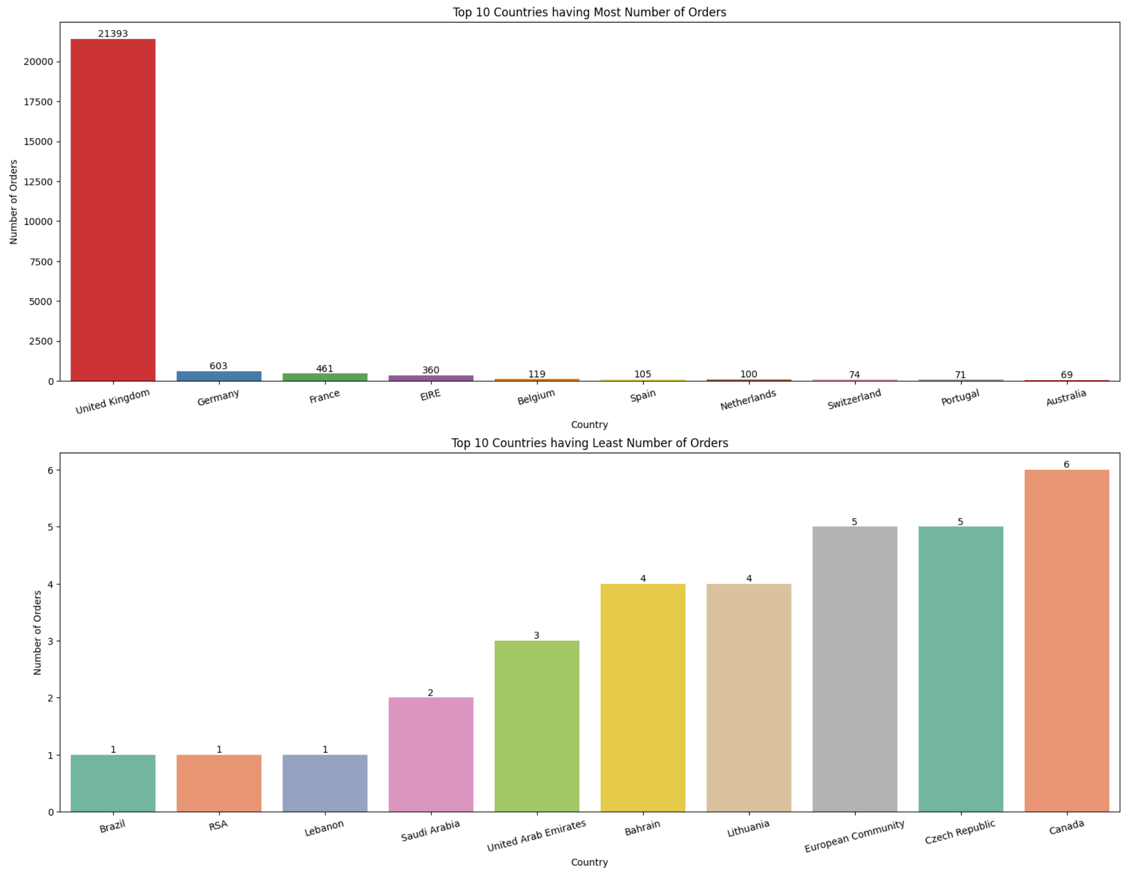
[ ]:
```
orders_country['InvoiceNo'].sum()
```

[ ]: 23798

[ ]:
```
orders_country[orders_country['Country']=='United Kingdom']['InvoiceNo']/
 ↪orders_country['InvoiceNo'].sum()
```

[ ]:
```
0    0.898941
Name: InvoiceNo, dtype: float64
```

[ ]:
```
fig = plt.figure(figsize = (20,15))

plt.subplot(2,1,1)
plt.title('Top 10 Countries having Most Number of Orders')
plt.ylabel('Number of Orders')
plt.xticks(rotation=15)
g = sns.barplot(data=orders_country.head(10), x='Country', y='InvoiceNo',␣
 ↪palette='Set1')
for j in g.patches:
   plt.text(x=j.get_x()+j.get_width()/2, y=j.get_height(), s=int(j.
 ↪get_height()), ha='center', va='bottom')

plt.subplot(2,1,2)
plt.title('Top 10 Countries having Least Number of Orders')
plt.ylabel('Number of Orders')
plt.xticks(rotation=15)
g = sns.barplot(data=orders_country.tail(10)[::-1], x='Country', y='InvoiceNo',␣
 ↪palette='Set2')
for j in g.patches:
   plt.text(x=j.get_x()+j.get_width()/2, y=j.get_height(), s=int(j.
 ↪get_height()), ha='center', va='bottom')

plt.show()
```



- 23798 orders are made in this store and 90% are from UK.

[ ]:
```
sales_country = df.groupby('Country')['TotalPrice'].sum().
    ↪sort_values(ascending=False).reset_index()
sales_country
```

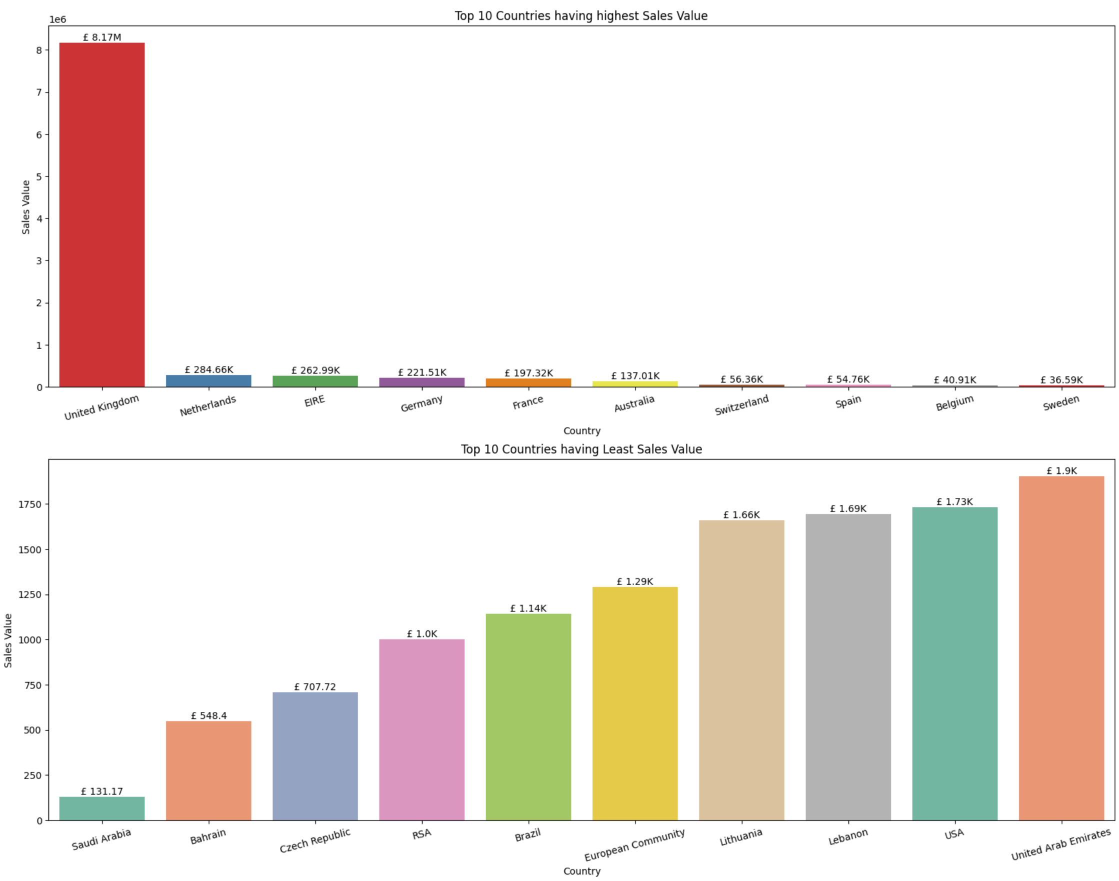
[ ]:
```
           Country   TotalPrice
0   United Kingdom 8.167128e+06
1      Netherlands 2.846615e+05
2             EIRE 2.629934e+05
3          Germany 2.215095e+05
4           France 1.973171e+05
5        Australia 1.370098e+05
6      Switzerland 5.636305e+04
7            Spain 5.475603e+04
8          Belgium 4.091096e+04
9           Sweden 3.658541e+04
10           Japan 3.534062e+04
11          Norway 3.516346e+04
12        Portugal 2.930297e+04
```

```
13                 Finland  2.232674e+04
14         Channel Islands  2.007639e+04
15                 Denmark  1.876814e+04
16                   Italy  1.689051e+04
17                  Cyprus  1.285876e+04
18                 Austria  1.015432e+04
19               Hong Kong  9.908240e+03
20               Singapore  9.120390e+03
21                  Israel  7.901970e+03
22                  Poland  7.213140e+03
23             Unspecified  4.740940e+03
24                  Greece  4.710520e+03
25                 Iceland  4.310000e+03
26                  Canada  3.666380e+03
27                   Malta  2.505470e+03
28    United Arab Emirates  1.902280e+03
29                     USA  1.730920e+03
30                 Lebanon  1.693880e+03
31               Lithuania  1.661060e+03
32      European Community  1.291750e+03
33                  Brazil  1.143600e+03
34                     RSA  1.002310e+03
35          Czech Republic  7.077200e+02
36                 Bahrain  5.484000e+02
37            Saudi Arabia  1.311700e+02
```

[ ]:

```
fig = plt.figure(figsize = (20,15))

plt.subplot(2,1,1)
plt.title('Top 10 Countries having highest Sales Value')
plt.ylabel('Sales Value')
plt.xticks(rotation=15)
g = sns.barplot(data=sales_country.head(10), x='Country', y='TotalPrice',␣
 ↪palette='Set1')
for j in g.patches:
  v,l = j.get_height(), len(str(int(j.get_height())))
  if l < 7:
    v = str((v/1000).round(2))+'K'
  else:
    v = str((v/1000000).round(2))+'M'
  plt.text(x=j.get_x()+j.get_width()/2, y=j.get_height(), s='£ '+v,␣
 ↪ha='center', va='bottom')

plt.subplot(2,1,2)
plt.title('Top 10 Countries having Least Sales Value')
plt.ylabel('Sales Value')
plt.xticks(rotation=15)
```

```python
g = sns.barplot(data=sales_country.tail(10)[::-1], x='Country', y='TotalPrice', palette='Set2')
for j in g.patches:
  v,l = j.get_height(), len(str(int(j.get_height())))
  if l < 4:
    v = str(v.round(2))
  else:
    v = str((v/1000).round(2))+'K'
  plt.text(x=j.get_x()+j.get_width()/2, y=j.get_height(), s='£ '+v, ha='center', va='bottom')

plt.show()
```



[ ]:
```python
(sales_country['TotalPrice'].sum()/1000000).round(2)
```

[ ]: 9.73

[ ]:
```python
sales_country[sales_country['Country']=='United Kingdom']['TotalPrice']/sales_country['TotalPrice'].sum()
```

[ ]:
```
0    0.839721
Name: TotalPrice, dtype: float64
```

- Total Sales of the store is £ 9.73M out of which £ 8.17M i.e., 84% is from UK, followed by Netherlands with £ 284.6K

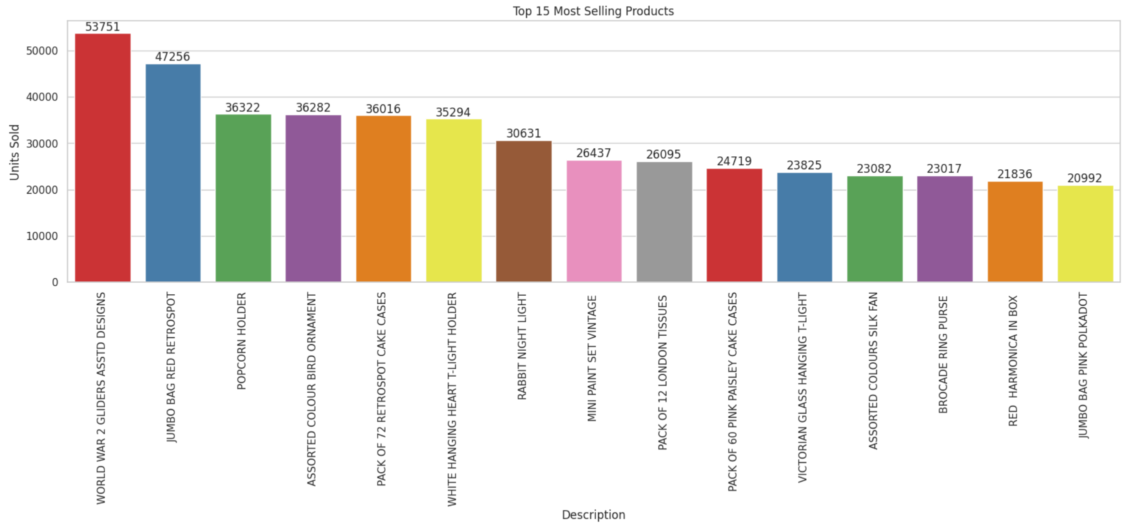
1.4.3 Product Analysis

[ ]:
```
top_products = df.groupby('Description')['Quantity'].sum().
↪sort_values(ascending=False).reset_index()
top_products.head(30)
```

[ ]:
```
                          Description  Quantity
0     WORLD WAR 2 GLIDERS ASSTD DESIGNS     53751
1               JUMBO BAG RED RETROSPOT     47256
2                        POPCORN HOLDER     36322
3         ASSORTED COLOUR BIRD ORNAMENT     36282
4       PACK OF 72 RETROSPOT CAKE CASES     36016
5    WHITE HANGING HEART T-LIGHT HOLDER     35294
6                    RABBIT NIGHT LIGHT     30631
7                MINI PAINT SET VINTAGE     26437
8            PACK OF 12 LONDON TISSUES      26095
9    PACK OF 60 PINK PAISLEY CAKE CASES     24719
10      VICTORIAN GLASS HANGING T-LIGHT     23825
11           ASSORTED COLOURS SILK FAN      23082
12                  BROCADE RING PURSE      23017
13                  RED  HARMONICA IN BOX   21836
14              JUMBO BAG PINK POLKADOT     20992
15                 SMALL POPCORN HOLDER     20105
16      PAPER CHAIN KIT 50'S CHRISTMAS      18876
17              LUNCH BAG RED RETROSPOT     18658
18         60 TEATIME FAIRY CAKE CASES      18015
19                        PARTY BUNTING     18006
20              CHARLOTTE BAG SUKI DESIGN   17974
21                HEART OF WICKER SMALL     17828
22         RED RETROSPOT CHARLOTTE BAG      17538
23                 JUMBO BAG STRAWBERRY     17033
24  COLOUR GLASS T-LIGHT HOLDER HANGING     16332
25   GROW A FLYTRAP OR SUNFLOWER IN TIN     16172
26              JAM MAKING SET PRINTED      16065
27      60 CAKE CASES VINTAGE CHRISTMAS     15720
28         PACK OF 72 SKULL CAKE CASES      15121
29                   VINTAGE SNAP CARDS     14436
```

[ ]:
```
fig = plt.figure(figsize = (20,5))

plt.title('Top 15 Most Selling Products')
```

```
plt.ylabel('Units Sold')
plt.xticks(rotation=90)
g = sns.barplot(data=top_products.head(15), x='Description', y='Quantity',␣
 ↪palette='Set1')
for j in g.patches:
   plt.text(x=j.get_x()+j.get_width()/2, y=j.get_height(), s=int(j.
 ↪get_height()), ha='center', va='bottom')
```



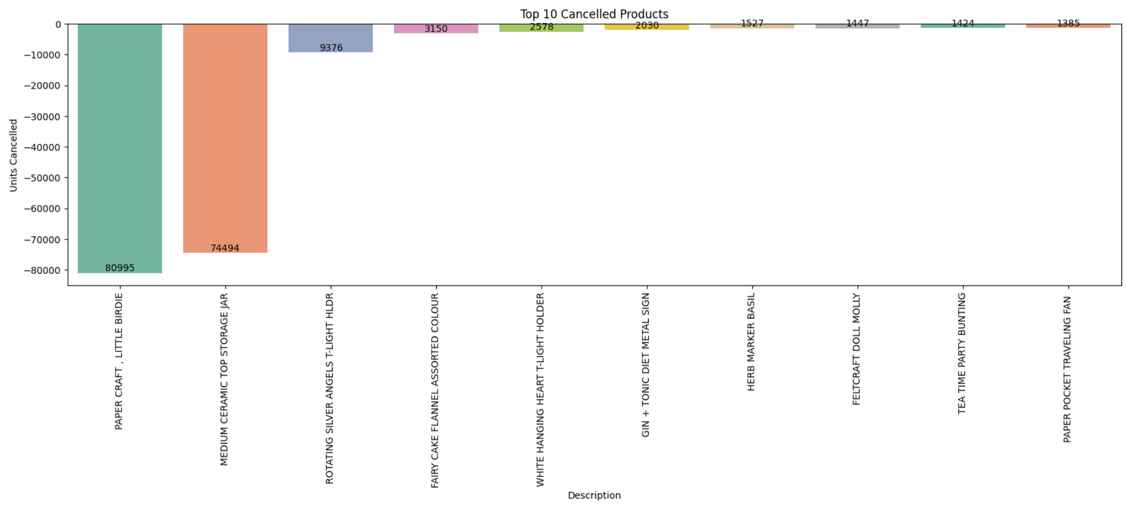
- *WORLD WAR 2 GLIDERS ASSTD DESIGNS* is the most selling product followed by *JUMBO BAG RED RETROSPOT* and *POPCORN HOLDER*

[ ]:
```
cancelled_products = df[df['InvoiceNo'].str.contains('C')].
 ↪groupby('Description')['Quantity'].sum().sort_values().reset_index()
cancelled_products = cancelled_products[~cancelled_products['Description'].
 ↪isin(["Manual", "Discount"])].reset_index(drop=True)
cancelled_products.head(20)
```

[ ]:
```
                             Description  Quantity
0           PAPER CRAFT , LITTLE BIRDIE    -80995
1         MEDIUM CERAMIC TOP STORAGE JAR    -74494
2    ROTATING SILVER ANGELS T-LIGHT HLDR     -9376
3     FAIRY CAKE FLANNEL ASSORTED COLOUR     -3150
4     WHITE HANGING HEART T-LIGHT HOLDER     -2578
5           GIN + TONIC DIET METAL SIGN      -2030
6                     HERB MARKER BASIL      -1527
7                   FELTCRAFT DOLL MOLLY     -1447
8                 TEA TIME PARTY BUNTING     -1424
9           PAPER POCKET TRAVELING FAN       -1385
10      PINK BLUE FELT CRAFT TRINKET BOX     -1321
```

```
11     WORLD WAR 2 GLIDERS ASSTD DESIGNS     -1200
12     COLOUR GLASS. STAR T-LIGHT HOLDER     -1174
13               JUMBO BAG RED RETROSPOT     -1115
14                   HOME SWEET HOME MUG     -1052
15                 PANTRY CHOPPING BOARD      -946
16             PLACE SETTING WHITE HEART      -890
17            FELTCRAFT BUTTERFLY HEARTS      -877
18              REGENCY CAKESTAND 3 TIER      -855
19             ASSORTED COLOURS SILK FAN      -744
```

[ ]:
```
fig = plt.figure(figsize = (20,5))

plt.title('Top 10 Cancelled Products')
plt.ylabel('Units Cancelled')
plt.xticks(rotation=90)
g = sns.barplot(data=cancelled_products.head(10), x='Description',␣
 ↪y='Quantity', palette='Set2')
for j in g.patches:
   plt.text(x=j.get_x()+j.get_width()/2, y=j.get_height(), s=int(j.
 ↪get_height()*-1), ha='center', va='bottom')
```



- *PAPER CRAFT , LITTLE BIRDIE* is the most cancelled product with almost 80K units got cancelled followed by *MEDIUM CERAMIC TOP STORAGE JAR*

1.4.4 Time Series

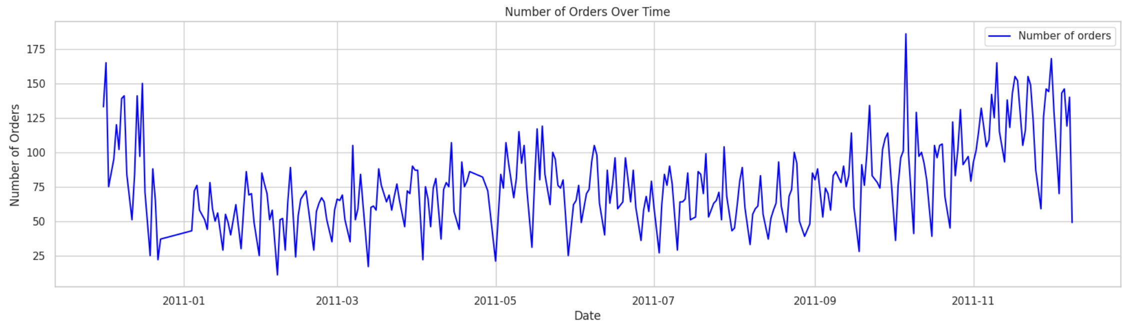
[ ]:
```
df_ts = df[['InvoiceDate','InvoiceNo']].copy()
df_ts['InvoiceDate'] = pd.to_datetime(df_ts['InvoiceDate'].dt.date,␣
 ↪errors='coerce')
df_ts = df_ts.groupby('InvoiceDate')['InvoiceNo'].nunique().reset_index()
```

```
df_ts['InvoiceDay'] = df_ts['InvoiceDate'].dt.day
df_ts['InvoiceMonth'] = df_ts['InvoiceDate'].dt.month_name()
df_ts['WeekdayName'] = df_ts['InvoiceDate'].dt.day_name()
df_ts.head(10)
```

[ ]:

	InvoiceDate	InvoiceNo	InvoiceDay	InvoiceMonth	WeekdayName
0	2010-12-01	133	1	December	Wednesday
1	2010-12-02	165	2	December	Thursday
2	2010-12-03	75	3	December	Friday
3	2010-12-05	95	5	December	Sunday
4	2010-12-06	120	6	December	Monday
5	2010-12-07	102	7	December	Tuesday
6	2010-12-08	139	8	December	Wednesday
7	2010-12-09	141	9	December	Thursday
8	2010-12-10	84	10	December	Friday
9	2010-12-12	51	12	December	Sunday

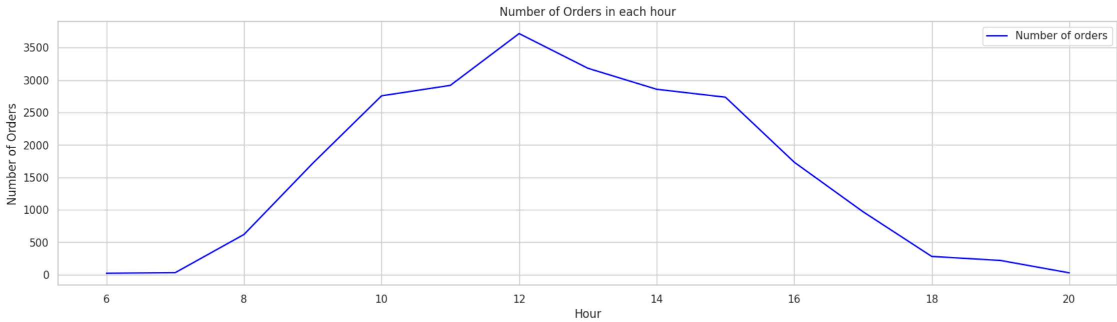
[ ]:
```
fig = plt.figure(figsize = (20,5))
sns.set(style="whitegrid")
sns.lineplot(data=df_ts, x='InvoiceDate', y='InvoiceNo', label='Number of␣
↪orders', color='blue')
plt.title('Number of Orders Over Time')
plt.xlabel('Date')
plt.ylabel('Number of Orders')
plt.show()
```



Purchase Trend: - Peaks in December (holiday shopping season). - Dips in early months after the holiday rush. - Steady fluctuations mid-year (April–August). - Pre-holiday growth (September–November).

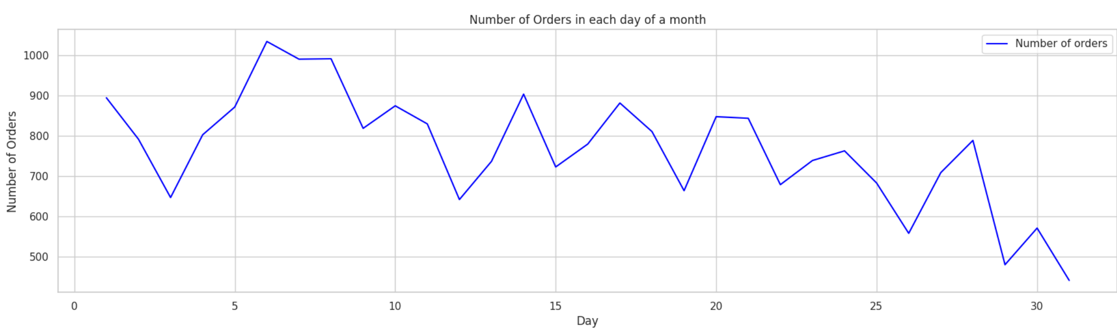
[ ]:
```
hs = df.groupby(df['InvoiceDate'].dt.hour)['InvoiceNo'].nunique().reset_index()

fig = plt.figure(figsize = (20,5))
sns.set(style="whitegrid")
```

```
sns.lineplot(data=hs, x='InvoiceDate', y='InvoiceNo', label='Number of orders',␣
 ↪color='blue')
plt.title('Number of Orders in each hour')
plt.xlabel('Hour')
plt.ylabel('Number of Orders')
plt.show()
```



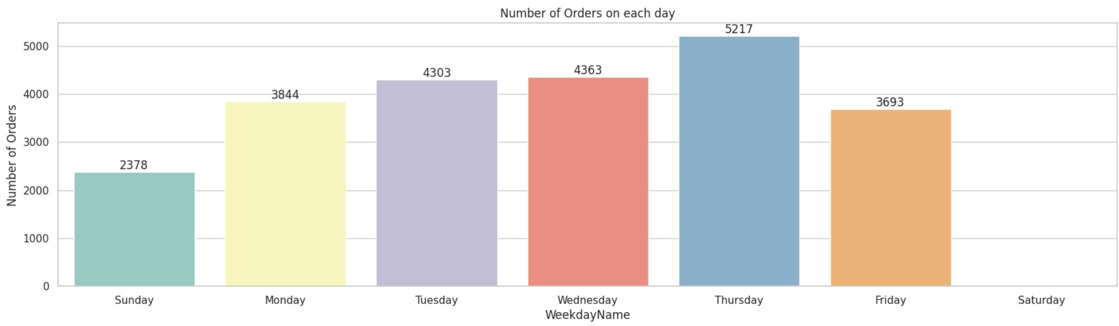
- Most of the sales are happening between 9 to 4 peaks at 12 noon.

```
[ ]: ds = df_ts.groupby('InvoiceDay')['InvoiceNo'].sum().reset_index()
fig = plt.figure(figsize = (20,5))
sns.set(style="whitegrid")
sns.lineplot(data=ds, x='InvoiceDay', y='InvoiceNo', label='Number of orders',␣
 ↪color='blue')
plt.title('Number of Orders in each day of a month')
plt.xlabel('Day')
plt.ylabel('Number of Orders')
plt.show()
```



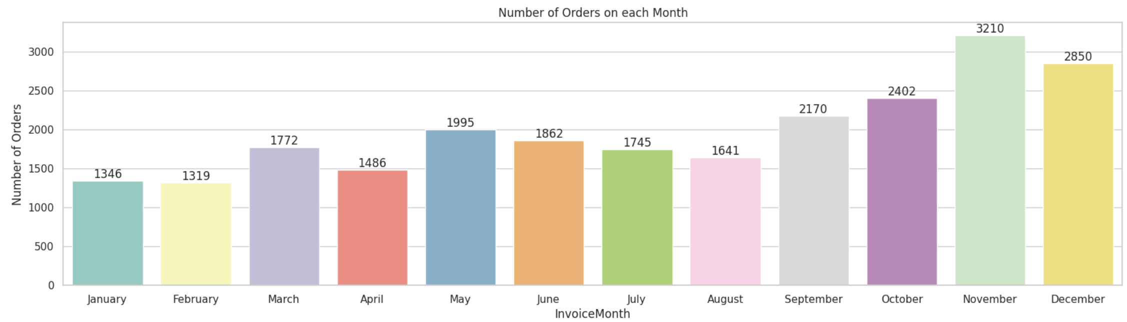
- Sales are high till starting 8 days of the month, decent during middle days and are low during last days.

```python
ws = df_ts.groupby('WeekdayName')['InvoiceNo'].sum().reset_index()
order = ["Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday",␣
↪"Saturday"]
fig = plt.figure(figsize = (20,5))
g = sns.barplot(data=ws, x='WeekdayName', y='InvoiceNo', palette='Set3',␣
↪order=order)
for j in g.patches:
   plt.text(x=j.get_x()+j.get_width()/2, y=j.get_height(), s=int(j.
↪get_height()), ha='center', va='bottom')
plt.title('Number of Orders on each day')
plt.ylabel('Number of Orders')
plt.show()
```



- Most of the sales are happening on Thursday followed by Wednesday and Tuesday.
- There are no sales on Saturdays which means that Saturday is a holiday for the shop.

```python
ms = df_ts.groupby('InvoiceMonth')['InvoiceNo'].sum().reset_index()
order = ["January", "February", "March", "April", "May", "June", "July",␣
↪"August", "September", "October", "November", "December"]

fig = plt.figure(figsize = (20,5))
g = sns.barplot(data=ms, x='InvoiceMonth', y='InvoiceNo', palette='Set3',␣
↪order=order)
for j in g.patches:
   plt.text(x=j.get_x()+j.get_width()/2, y=j.get_height(), s=int(j.
↪get_height()), ha='center', va='bottom')
plt.title('Number of Orders on each Month')
plt.ylabel('Number of Orders')
plt.show()
```



- Most of the sales are happening on November month followed by December and October

1.4.5 Customer Analysis

[ ]:
```
cust_df = df[df['CustomerID']!='n']
```

[ ]:
```
orders_cust = cust_df.groupby('CustomerID')['InvoiceNo'].nunique().
  ↪sort_values(ascending=False).reset_index()
orders_cust
```

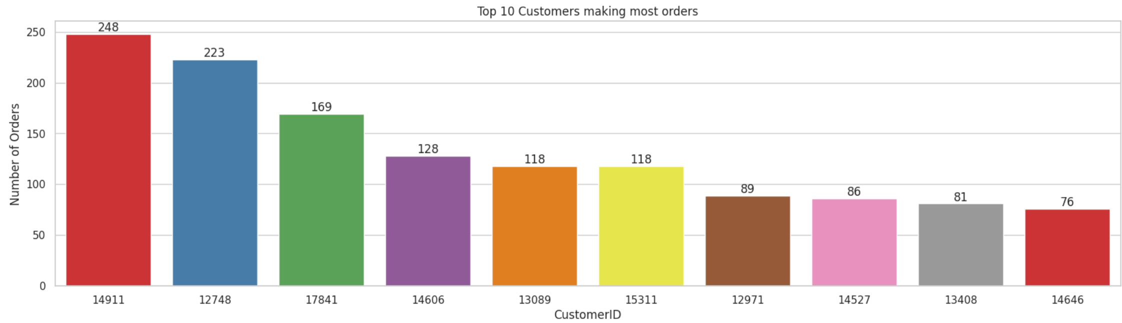
[ ]:

	CustomerID	InvoiceNo
0	14911	248
1	12748	223
2	17841	169
3	14606	128
4	13089	118
…	…	…
4366	13441	1
4367	13449	1
4368	15744	1
4369	14518	1
4370	15076	1

[4371 rows x 2 columns]

[ ]:
```
fig = plt.figure(figsize = (20,5))

plt.title('Top 10 Customers making most orders')
plt.ylabel('Number of Orders')
g = sns.barplot(data=orders_cust.head(10), x='CustomerID', y='InvoiceNo',␣
  ↪palette='Set1')
for j in g.patches:
    plt.text(x=j.get_x()+j.get_width()/2, y=j.get_height(), s=int(j.
  ↪get_height()), ha='center', va='bottom')
```

```
plt.show()
```

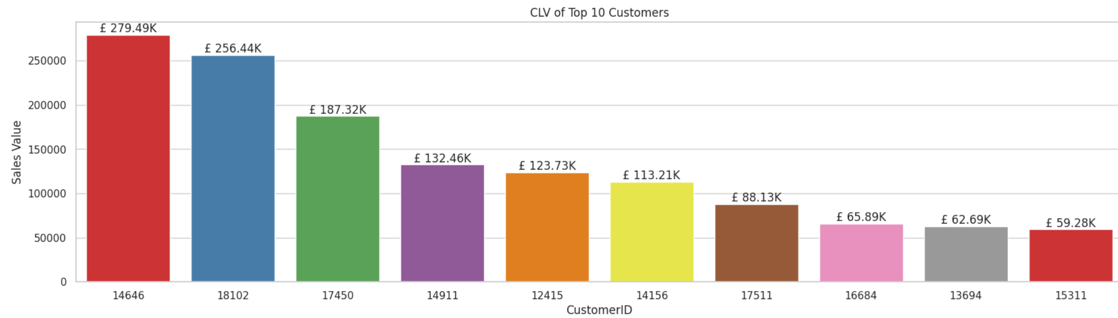


- Customer 14911 made most number of purchases followed by 12748 and 17841

[ ]:
```
sales_cust = cust_df.groupby('CustomerID')['TotalPrice'].sum().
    ↪sort_values(ascending=False).reset_index()
sales_cust
```

[ ]:
```
      CustomerID     TotalPrice
0          14646  279489.019443
1          18102  256438.488361
2          17450  187322.170972
3          14911  132458.729279
4          12415  123725.450123
…            …            …
4366       12503   -1126.000000
4367       17603   -1165.300008
4368       14213   -1192.199991
4369       15369   -1592.489990
4370       17448   -4287.629883

[4371 rows x 2 columns]
```

[ ]:
```
fig = plt.figure(figsize = (20,5))

plt.title('CLV of Top 10 Customers')
plt.ylabel('Sales Value')
g = sns.barplot(data=sales_cust.head(10), x='CustomerID', y='TotalPrice',␣
    ↪palette='Set1')
for j in g.patches:
    plt.text(x=j.get_x()+j.get_width()/2, y=j.get_height(), s='£ '+ str((j.
    ↪get_height()/1000).round(2))+'K', ha='center', va='bottom')

plt.show()
```



- Customer 14646 has high CLV with £ 280K followed by 18102 with £ 256K and 17450 with £ 187K.

Customer Segmentation - RFM Analysis

[ ]:
```
cust_df['InvoiceDate'] = pd.to_datetime(cust_df['InvoiceDate'].dt.date)
today = cust_df['InvoiceDate'].max()
df_rfm = cust_df.groupby('CustomerID').agg({'InvoiceDate': lambda x: (today - x.
 ↪max()).days,
                                            'InvoiceNo': lambda x: len(x),
                                            'TotalPrice': lambda x: x.sum()}).
 ↪reset_index()
df_rfm.columns = ['CustomerID','Recency', 'Frequency', 'Monetary']
df_rfm.head()
```

[ ]:

	CustomerID	Recency	Frequency	Monetary
0	12346	325	2	0.000000
1	12347	2	182	4309.999988
2	12348	75	31	1797.239997
3	12349	18	73	1757.549994
4	12350	310	17	334.399997

[ ]:
```
df_rfm['Recency_Score'] = pd.qcut(df_rfm['Recency'], 5, labels = [5, 4, 3, 2,␣
 ↪1])
df_rfm['Frequency_Score'] = pd.qcut(df_rfm['Frequency'].rank(method = 'first'),␣
 ↪5, labels = [1, 2, 3, 4, 5])
df_rfm['Monetary_Score'] = pd.qcut(df_rfm['Monetary'], 5, labels = [1, 2, 3, 4,␣
 ↪5])
df_rfm.head()
```

[ ]:

	CustomerID	Recency	Frequency	Monetary	Recency_Score	Frequency_Score	\
0	12346	325	2	0.000000	1	1	
1	12347	2	182	4309.999988	5	5	
2	12348	75	31	1797.239997	2	3	
3	12349	18	73	1757.549994	4	4	

```
4      12350      310          17   334.399997             1                2

  Monetary_Score
0              1
1              5
2              4
3              4
4              2
```

[ ]:
```
df_rfm['RFM_Score'] = df_rfm['Recency_Score'].astype(str) +␣
↪df_rfm['Frequency_Score'].astype(str) + df_rfm['Monetary_Score'].astype(str)
df_rfm.head()
```

[ ]:
```
  CustomerID  Recency  Frequency     Monetary Recency_Score Frequency_Score  \
0      12346      325          2     0.000000             1               1
1      12347        2        182  4309.999988             5               5
2      12348       75         31  1797.239997             2               3
3      12349       18         73  1757.549994             4               4
4      12350      310         17   334.399997             1               2

  Monetary_Score RFM_Score
0              1       111
1              5       555
2              4       234
3              4       444
4              2       122
```

[ ]:
```
def segment_customer(row):
    # Champions: Recent, frequent, and high spenders
    if row['RFM_Score'] == '555':
        return 'Champions'
    # Lost Customers: Lowest scores across all dimensions
    elif row['RFM_Score'] == '111':
        return 'Lost Customers'
    # Hibernating: Low recency, frequency, and monetary
    elif row['Recency_Score'] in [1, 2] and row['Frequency_Score'] in [1, 2, 3]:
        return 'Hibernating'
    # About to Sleep: Low engagement with medium recency
    elif row['Recency_Score'] == 3 and row['Frequency_Score'] <= 2:
        return 'About to Sleep'
    # Needs Attention: Medium engagement needing intervention
    elif row['Recency_Score'] in [2, 3] and row['Frequency_Score'] == 3:
        return 'Needs Attention'
    # Can't Lose Them: Loyal customers at risk of churning
    elif row['Recency_Score'] in [1, 2, 3] and row['Frequency_Score'] in [4, 5]:
        return "Can't Lose Them"
    # Loyal Customers: Bread-and-butter customers
```

```
    elif row['Recency_Score'] >= 4 and row['Frequency_Score'] >= 4 and␣
↪row['Monetary_Score'] >= 4:
        return 'Loyal Customers'
    # Recent Users: High recency but not yet fully loyal
    elif row['Recency_Score'] in [4, 5] and row['Frequency_Score'] <= 2:
        return 'New Customers'
    # Potential Loyalists: High recency with moderate frequency or low monetary␣
↪scores
    elif row['Recency_Score'] in [4, 5] and (row['Frequency_Score'] in [3, 4]␣
↪or row['Monetary_Score'] in [1, 2, 3]):
        return 'Potential Loyalists'
    else:
        return 'Unclassified'
```

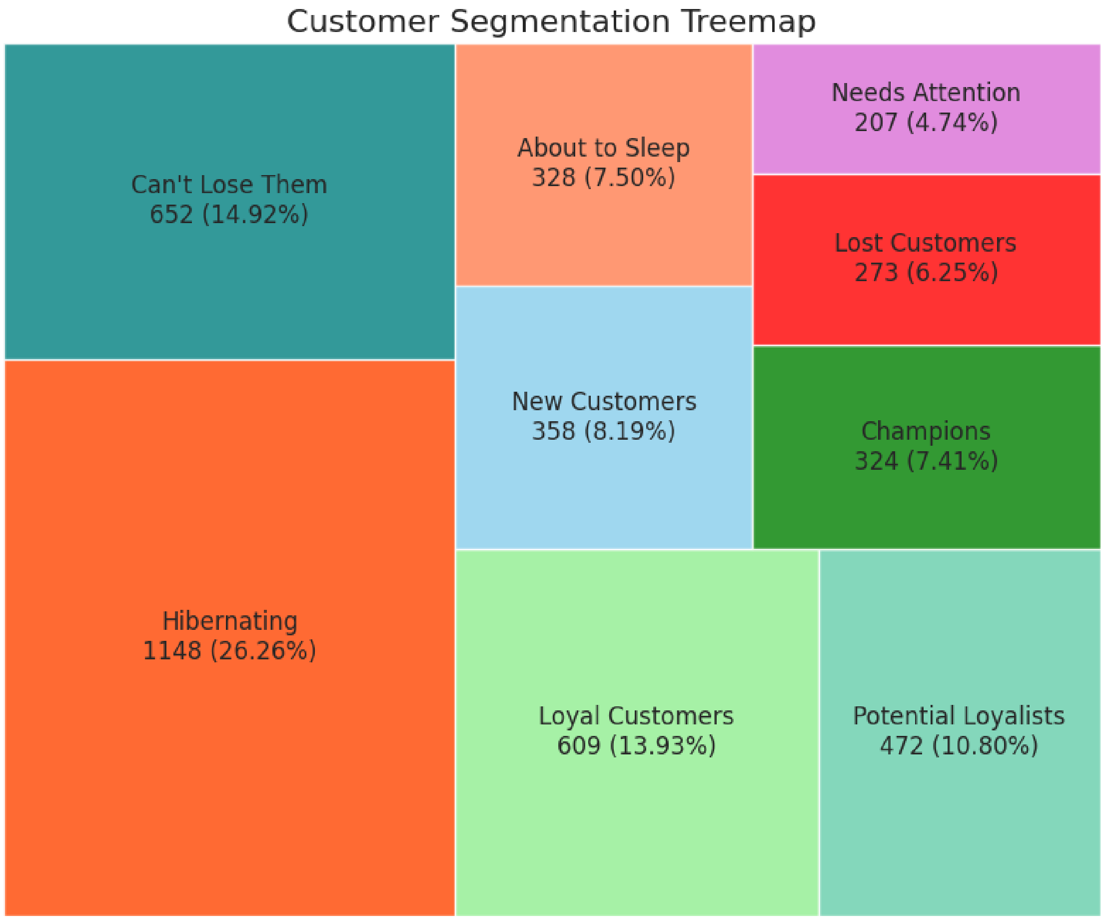
[ ]:
```
df_rfm['Segment'] = df_rfm.apply(segment_customer, axis=1)
df_rfm.head()
```

[ ]:
```
   CustomerID  Recency  Frequency     Monetary Recency_Score Frequency_Score  \
0       12346      325          2     0.000000             1               1
1       12347        2        182  4309.999988             5               5
2       12348       75         31  1797.239997             2               3
3       12349       18         73  1757.549994             4               4
4       12350      310         17   334.399997             1               2

  Monetary_Score RFM_Score          Segment
0              1       111   Lost Customers
1              5       555        Champions
2              4       234      Hibernating
3              4       444  Loyal Customers
4              2       122      Hibernating
```

[ ]:
```
segments = df_rfm['Segment'].value_counts().reset_index()
segments['Percentage'] = (segments['count'] / segments['count'].sum()).round(4)␣
↪* 100
segments
```

[ ]:
```
               Segment  count  Percentage
0          Hibernating   1148       26.26
1       Can't Lose Them   652       14.92
2       Loyal Customers   609       13.93
3   Potential Loyalists   472       10.80
4         New Customers   358        8.19
5        About to Sleep   328        7.50
6             Champions   324        7.41
7        Lost Customers   273        6.25
8       Needs Attention   207        4.74
```

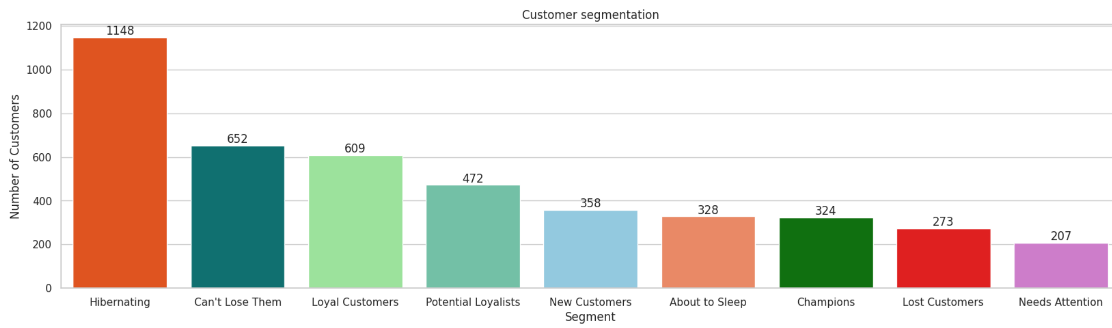
[162]:
```python
# Treemap
plt.figure(figsize=(10, 8))
plt.title('Customer Segmentation Treemap', fontsize=16)
plt.axis('off')

color_map = {
    'Champions': 'green',
    'Loyal Customers': 'lightgreen',
    'Potential Loyalists': 'mediumaquamarine',
    'New Customers': 'skyblue',
    "Can't Lose Them": 'teal',
    'Needs Attention': 'orchid',
    'Lost Customers': 'red',
    'Hibernating': 'orangered',
    'About to Sleep': 'coral'
}

colors = [color_map[row['Segment']] for _, row in segments.iterrows()]
labels = [f"{row['Segment']}\n{row['count']} ({row['Percentage']:.2f}%)" for _,␣
↪row in segments.iterrows()]
sizes = segments['count']

sq.plot(sizes=sizes, label=labels, color=colors, alpha=0.8)

plt.show()
```



1. **Hibernating (26.26%, 1148 customers)**:
   - This is the largest segment, indicating a significant number of customers who haven't purchased recently.
2. **Can't Lose Them (14.92%, 652 customers)**:
   - These are high-value customers at risk of switching to competitors.
3. **Loyal Customers (13.93%, 609 customers)**:
   - These customers consistently purchase and are satisfied with the brand.
4. **Potential Loyalists (10.80%, 472 customers)**:
   - These are customers with the potential to become loyal buyers if nurtured correctly.
5. **New Customers (8.19%, 358 customers)**:
   - Customers who recently made their first purchase.
6. **About to Sleep (7.50%, 328 customers)**:
   - These are at risk of becoming inactive.
7. **Champions (7.41%, 324 customers)**:
   - Your best customers who buy frequently, spend the most, and promote your brand.
8. **Lost Customers (6.25%, 273 customers)**:
   - Customers who are no longer engaged with the brand.
9. **Needs Attention (4.74%, 207 customers)**:

- These customers are showing declining engagement and could churn soon.

[ ]:
```
fig = plt.figure(figsize = (20,5))

plt.title('Customer segmentation')
plt.ylabel('Number of Customers')
g = sns.barplot(data=segments, x='Segment', y='count', palette=color_map)
for j in g.patches:
  plt.text(x=j.get_x()+j.get_width()/2, y=j.get_height(), s=int(j.
↪get_height()), ha='center', va='bottom')

plt.show()
```



Cohort Analysis

[ ]:
```
cust_df['InvoiceDate'] = pd.to_datetime(cust_df['InvoiceDate'])
cust_df['CohortMonth'] = cust_df.groupby('CustomerID')['InvoiceDate'].
↪transform('min').dt.to_period('M')
cust_df['InvoiceMonth'] = cust_df['InvoiceDate'].dt.to_period('M')
cust_df['CohortIndex'] = (cust_df['InvoiceMonth'].dt.to_timestamp() -␣
↪cust_df['CohortMonth'].dt.to_timestamp()).apply(lambda x: x.days // 30)
cust_df.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  TotalPrice CohortMonth  \
0  2010-12-01       2.55      17850  United Kingdom   15.300000     2010-12
1  2010-12-01       3.39      17850  United Kingdom   20.340001     2010-12
2  2010-12-01       2.75      17850  United Kingdom   22.000000     2010-12
3  2010-12-01       3.39      17850  United Kingdom   20.340001     2010-12
```

```
4  2010-12-01       3.39      17850  United Kingdom   20.340001     2010-12

  InvoiceMonth  CohortIndex
0      2010-12            0
1      2010-12            0
2      2010-12            0
3      2010-12            0
4      2010-12            0
```

[ ]:
```
# Cohort Table
cohort_data = cust_df.groupby(['CohortMonth', 'CohortIndex'])['CustomerID'].
 ↪nunique().reset_index()

cohort_table = cohort_data.pivot(index='CohortMonth', columns='CohortIndex',␣
 ↪values='CustomerID')
cohort_table = cohort_table.divide(cohort_table.iloc[:, 0], axis=0)

cohort_table_percentage = cohort_table.style.format("{:.2%}").
 ↪background_gradient(cmap='Blues')
cohort_table_percentage
```

[ ]:
```
<pandas.io.formats.style.Styler at 0x794b729de260>
```

- The first-month retention rate (Month 1) for earlier cohorts like December 2010 (38.19%) and January 2011 (40.62%) is relatively higher compared to later cohorts like June 2011 (20.85%) and August 2011 (25.15%).
- December 2010 cohort shows consistent retention over multiple months, staying around 33%-39%, with a significant spike in Month 11 (50%).
- Later cohorts (e.g., June 2011 onward) exhibit sharper drop-offs in retention after Month 1, with lower long-term retention rates.
- Spikes in retention during specific months (e.g., Month 11 for December 2010 cohort and Month 4 for January 2011 cohort) suggest the influence of seasonality or external events.
- Earlier cohorts have longer retention trends visible, whereas newer cohorts have shorter data ranges due to their recent start dates, limiting the ability to analyze long-term retention.
- Across all cohorts, the most significant drop occurs between Month 0 and Month 1, with retention rates tapering off more gradually in subsequent months.

Average days between purchase

[ ]:
```
adp = cust_df.groupby(['CustomerID']).agg({'InvoiceDate': lambda x: x.
 ↪unique()}).reset_index()
adp.head()
```

[ ]:
```
  CustomerID                                     InvoiceDate
0      12346                           [2011-01-18 00:00:00]
1      12347  [2010-12-07 00:00:00, 2011-01-26 00:00:00, 201…
2      12348  [2010-12-16 00:00:00, 2011-01-25 00:00:00, 201…
```

```
3       12349                                   [2011-11-21 00:00:00]
4       12350                                   [2011-02-02 00:00:00]
```

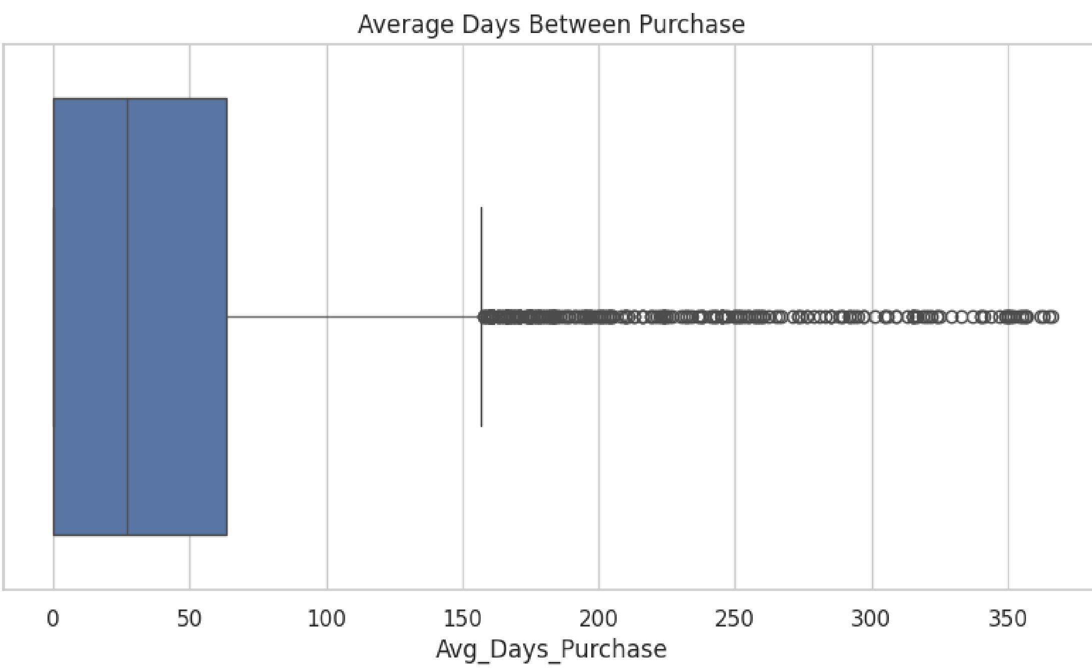
[ ]:
```
def avg_days(x):
  days = []
  for i in range(len(x)-1):
    days.append((x[i+1] - x[i]).days)
  return np.mean(days)
```

[ ]:
```
adp['Avg_Days_Purchase'] = adp['InvoiceDate'].apply(lambda x: avg_days(x))
adp.drop('InvoiceDate', axis=1, inplace=True)
adp.fillna(0, inplace=True)
adp['Avg_Days_Purchase'] = adp['Avg_Days_Purchase'].astype('int32')
adp.sort_values(by='Avg_Days_Purchase',ascending=False,inplace=True)
adp.head()
```

[ ]:

	CustomerID	Avg_Days_Purchase
1666	14594	366
1859	14865	365
634	13173	363
355	12785	362
528	13030	357

[ ]:
```
fig = plt.figure(figsize = (10,5))
sns.boxplot(x=adp['Avg_Days_Purchase'])
plt.title('Average Days Between Purchase')
plt.show()
```



- There are so many outliers in the data of average number of days between purchase after 150.
- 75% of the customers makes the next purchase within 60 days.
- 50% of the customers makes the next purchase within 30 days.

2 Insights

1. **Overview:**
   The dataset spans **December 2010 to December 2011**, capturing **4371 customers** from **38 countries**. The store processed **23,798 orders** and sold **514,503 items**, generating **£9.73M in total sales**.

2. **Country Analysis:**
   - The **United Kingdom** dominates with **3950 customers** (~90%), contributing **£8.17M in sales** (84%) and accounting for 90% of the orders.
   - The next highest sales come from the **Netherlands (£284.6K)**.

3. **Product Analysis:**
   - **WORLD WAR 2 GLIDERS ASSTD DESIGNS** is the **top-selling product**, followed by **JUMBO BAG RED RETROSPOT** and **POPCORN HOLDER**.
   - The most-canceled product is **PAPER CRAFT, LITTLE BIRDIE** (~80K units got cancelled), followed by **MEDIUM CERAMIC TOP STORAGE JAR**.

4. **Purchase Trends:**
   - **December** (holiday shopping season) shows the highest sales, while sales dip in the **early months** and steadily rise again from **September to November**.
   - Most sales occur between **9 AM and 4 PM**, peaking at **12 noon**.
   - Sales are strongest in the **first 8 days of the month**, moderate during the middle, and lowest toward the end.
   - **Thursday** sees the most sales, followed by **Wednesday** and **Tuesday**, while **Saturdays have no sales**, indicating the store is likely closed.
   - By month, **November** leads in sales, followed by **December** and **October**.

5. **Customer Analysis:**
   - The most active customer (ID **14911**) made the highest number of purchases.
   - The highest **Customer Lifetime Value (CLV)** belongs to ID **14646 (£280K)**, followed by IDs **18102 (£256K)** and **17450 (£187K)**.

6. **Customer Segmentation (RFM Analysis):**

- The largest segments include **Hibernating (26.26%)** and **Can't Lose Them (14.92%)**.

- High-value segments: **Champions (7.41%)**, **Loyal Customers (13.93%)**, and **Potential Loyalists (10.8%)**.

- At-risk segments: **Lost Customers (6.25%)**, **About to Sleep (7.5%)**, and **Needs Attention (4.74%)**.

7. **Customer Cohort Analysis:**

    - **First-month retention rates** are higher for earlier cohorts like **December 2010 (38.19%)** and **January 2011 (40.62%)**, compared to newer cohorts like **June 2011 (20.85%)**.

    - The **December 2010 cohort** shows long-term retention (~33%-39%), with a notable spike in **Month 11 (50%)**, possibly influenced by seasonal factors.

    - Retention drops significantly between **Month 0 and Month 1**, with gradual tapering thereafter.

8. **Purchase Intervals:**

    - **75% of customers** repurchase within **60 days**, and **50% repurchase within 30 days**.

    - There are significant outliers with intervals exceeding **150 days**, indicating inconsistent buying behavior among some customers.

3 Recommendations

1. **Reward Your Best Customers (Champions)**:

- Create **exclusive offers** and **VIP loyalty programs** for your **Champions** (7.41%) to maintain their loyalty. Offer them early access to new products, special discounts, or personalized services to ensure they keep coming back.

2. **Encourage Repeat Purchases from Loyal Customers**:

- For **Loyal Customers** (13.93%), offer **personalized discounts**, **birthday rewards**, or **bonus loyalty points** to encourage repeat purchases and strengthen their relationship with your brand.

3. **Convert Potential Loyalists into Regulars**:

- **Potential Loyalists** (10.8%) can be converted with targeted campaigns offering **time-limited discounts** or **exclusive previews**. These offers can help them make another purchase and become regular customers.

4. **Re-engage Lost Customers**:

- **Lost Customers** (6.25%) haven't bought in a while. Use **re-engagement campaigns** with **special offers**, such as free shipping or a discount on their next order, to bring them back.

5. **Prevent Churn for At-Risk Customers**:

- For **About to Sleep** (7.5%) and **Needs Attention** (4.74%) segments, create **urgent and personalized offers** to motivate them to make a purchase before they stop buying altogether.

6. **Revive Hibernating Customers**:

- **Hibernating** customers (26.26%) have been inactive for a while. Send them **"We miss you" offers**, **exclusive discounts**, or special deals tailored to their previous interests to entice them to return.

7. **Utilize Email Campaigns for Different Segments**:

- Send **segment-specific email campaigns** to engage each customer group effectively. For example, **Loyal Customers** might get loyalty bonuses, while **At-Risk** customers get discount reminders to encourage them to return.

8. **Optimize Your Loyalty Program**:

- Design a **tiered loyalty program** that benefits **Loyal Customers** with better rewards, while encouraging **Hibernating** and **At-Risk** customers to return with easy-to-earn incentives.

9. **Expand Market Outside the UK**:

- **Focus on key international markets**: While the UK dominates sales, consider expanding into markets like the **Netherlands** (which already shows good potential), and other **European** or **North American** regions with similar demographics. Tailor marketing campaigns to each region's cultural preferences and seasonal buying behavior.
- **Localized Promotions**: Create region-specific campaigns based on local holidays, trends, and language. For example, you could leverage the **Black Friday** season for **North America** and **Boxing Day** for **Canada**, mirroring successful strategies from the UK holiday season.

10. **Leverage Peak Sales Periods**:

- **December** (holiday season) sees the highest sales. Plan **targeted campaigns** for all segments around this time. For example, offer **exclusive discounts** to **high-value customers** and **special promotions** for **new customers** or **first-time buyers** to maximize sales.

11. **Monitor Purchase Trends and Timing**:

- Since sales peak between **9 AM and 4 PM**, especially around **12 PM**, optimize your store's operations during these hours. Offer **flash sales** or **limited-time promotions** to increase conversion rates during peak hours.

12. **Product Strategy Based on Sales and Cancellations**:

- Focus on promoting **top-selling products** like **World War 2 Gliders** and **Jumbo Bag Red Retrospot**, while addressing issues with **high-cancellation products** like **Paper Craft, Little Birdie**. Ensure better product descriptions, stock levels, and customer support for high-risk items.

13. **Segment-Based Marketing and Retargeting**:

- Use **RFM data** to create **retargeting ads** on social media or through email. For example, show relevant ads to **Loyal Customers** about new products or exclusive deals, while targeting **Hibernating** customers with special offers to bring them back.