ecommerce-crm-analysis

October 13, 2024

[137]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import datetime as dt
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots
import plotly.colors as pc
```

NOTE: Import necessary Pyhton libraries for DATA analysis and DATA Visualisation.

[138]:
```
df=pd.read_csv('/content/drive/MyDrive/Data sets/Ecom_CRM_analysis.
 ↪csv',encoding='latin1')
df
```

[138]:
```
       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         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
…                    …          …         …             …
541904   12/9/2011 12:50       0.85     12680.0          France
541905   12/9/2011 12:50       2.10     12680.0          France
```

```
541906  12/9/2011 12:50      4.15     12680.0          France
541907  12/9/2011 12:50      4.15     12680.0          France
541908  12/9/2011 12:50      4.95     12680.0          France

[541909 rows x 8 columns]
```

NOTE: 1.DATA File is in CSV format 2.Load DATA using pd.read_csv function. 3.variable with name df created to save the data.

Key Insights: 1. There are 541909 rows & 8 Columns in this data set. 2. This Data set is of Online Store based in United Kingdom 3. Data set is of Wholesale customers of Online Store. 4. TOP 5 Competitors: 4.1.Amazon.co.uk 4.2.ebay.co.uk 4.3.etsy.co.uk 4.4.next.co.uk 4.5.Tesco.co.uk

[139]:
```
df.head()
```

[139]:
```
  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
```

NOTE: df.head function gives the top 5 rows & all columns of data set. 2.It gives the top view of data set. 3.It helps to understand the dataset details llike first transaction date& time…

[140]:
```
df.tail()
```

[140]:
```
       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  12/9/2011 12:50       0.85     12680.0  France
541905  12/9/2011 12:50       2.10     12680.0  France
541906  12/9/2011 12:50       4.15     12680.0  France
541907  12/9/2011 12:50       4.15     12680.0  France
541908  12/9/2011 12:50       4.95     12680.0  France
```

NOTE: 1.df.tail give bottom 5 rows of dataset. 2.It helps in understanding the very last transaction details of dat set. 3.It gives the last transaction date& time details.

[141]:
```
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
```

info() 1.functions is used to undrstand the data types of all columns. 2.Data type of InvocieDate need to to bechanged to datetime for further data analysis.

[142]:
```
df.describe()
```

[142]:

	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

Key Insights: The describe() function in pandas is convenient in getting various summary statistics. like mean,min,max,std,count & quartiles.

1 Dataset Description:

The dataset encompasses transactions from 01/12/2010 to 09/12/2011 for a non-store online retail business based and registered in the UK. Specializing in distinctive all-occasion gifts, the company's clientele includes a significant number of **wholesale customers.**

[143]:
```
df.isnull().sum()
```

[143]:
```
InvoiceNo          0
StockCode          0
```

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

Key Insights: 1. NUll values found in columns Description (1454) & CustomerID (135080).

Actionable Insights:

1. Replace Null values for further analysis.

[144]:
```
# replace null values with 0
df.fillna(0,inplace=True)
```

[145]:
```
# Null values removed
df.isnull().sum()
```

[145]:
```
InvoiceNo      0
StockCode      0
Description    0
Quantity       0
InvoiceDate    0
UnitPrice      0
CustomerID     0
Country        0
dtype: int64
```

[146]:
```
# invoice date formate
df['InvoiceDate']=pd.to_datetime(df['InvoiceDate'])
```

Key Insights: 1. dtype of InvoiceDate is chnged from object to pandas datetime function.

[147]:
```
# InvoiceDate Dtype changed to datetime
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  541909 non-null  object
 3   Quantity     541909 non-null  int64
 4   InvoiceDate  541909 non-null  datetime64[ns]
 5   UnitPrice    541909 non-null  float64
```

```
 6   CustomerID   541909 non-null  float64
 7   Country      541909 non-null  object
dtypes: datetime64[ns](1), float64(2), int64(1), object(4)
memory usage: 33.1+ MB
```

[148]:
```
# find duplicates
df.duplicated().sum()
```

[148]:
```
5268
```

Key Insights: 1.Found 5268 duplicate rows

Actonable Insights: 1.Need NOT delete these duplicates 5268 rows, as they are actually coming from inoviceNo & customer ID.

2. As DATA is at Inovice level, hence for multiple items of same InvoiceNO having multiple rows has been generated for same customerID.

[149]:
```
df
```

[149]:
```
       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]
```

[150]:
```
# using dropna() for missing values in customerID.

df.dropna(subset=['CustomerID'],inplace=True)
```

NOTE: dropna function is used to delete the rows having missing values with refrence to customerID.

[151]:
```
# finding outliers

df.describe()
```

[151]:
```
             Quantity                    InvoiceDate      UnitPrice  \
count   541909.000000                         541909  541909.000000
mean         9.552250  2011-07-04 13:34:57.156386048       4.611114
min     -80995.000000            2010-12-01 08:26:00  -11062.060000
25%          1.000000            2011-03-28 11:34:00       1.250000
50%          3.000000            2011-07-19 17:17:00       2.080000
75%         10.000000            2011-10-19 11:27:00       4.130000
max      80995.000000            2011-12-09 12:50:00   38970.000000
std        218.081158                            NaN      96.759853

           CustomerID
count   541909.000000
mean     11476.974671
min          0.000000
25%      12352.000000
50%      14382.000000
75%      16255.000000
max      18287.000000
std       6777.908326
```

Key Insights: 1.describe function is used to get the key average values of columns.(Count,mean,min,max,std & Quartiles) 2.It helps to find outliers in data set.

Actionable Insights: 1. UnitPrice column is having negative values, due to product returns from customers. 2. Negative values to be removed for further analysis.

[152]:
```
# delete negative values from quantity
df=df[df['Quantity']>0]
```

[153]:
```
#delete negative values from UnitPrice
df=df[df['UnitPrice']>0]
```

[154]:
```
df.describe()
```

[154]:
```
             Quantity                    InvoiceDate      UnitPrice  \
count   530104.000000                         530104  530104.000000
mean        10.542037  2011-07-04 20:16:05.225087744       3.907625
```

```
min          1.000000            2010-12-01 08:26:00        0.001000
25%          1.000000            2011-03-28 12:22:00        1.250000
50%          3.000000            2011-07-20 12:58:00        2.080000
75%         10.000000            2011-10-19 12:39:00        4.130000
max      80995.000000            2011-12-09 12:50:00    13541.330000
std        155.524124                            NaN       35.915681

          CustomerID
count  530104.000000
mean    11479.646222
min         0.000000
25%     12352.000000
50%     14388.000000
75%     16265.000000
max     18287.000000
std      6781.976768
```

[155]:
```
# Feature Engineering
# Create new column as Rvenue (quantity * unitprice)
df['Revenue']=df['Quantity']*df['UnitPrice']
```

Key Insights: 1. Ceate Revenue column

Actionable Insights:

1. Revenue column created by multiplying columns (Quantity*UnitPrice).

[156]:
```
df.head()
```

[156]:
```
  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  Revenue
0 2010-12-01 08:26:00       2.55     17850.0  United Kingdom    15.30
1 2010-12-01 08:26:00       3.39     17850.0  United Kingdom    20.34
2 2010-12-01 08:26:00       2.75     17850.0  United Kingdom    22.00
3 2010-12-01 08:26:00       3.39     17850.0  United Kingdom    20.34
4 2010-12-01 08:26:00       3.39     17850.0  United Kingdom    20.34
```

[157]:
```
# Create Refrence date
reference_date=pd.Timestamp(dt.datetime.now().date())
reference_date
```

[157]:
```
Timestamp('2024-10-13 00:00:00')
```

NOTE: 1.I have made assumption & create reference date as DATA looks very old. i.e from 2010-2011.

[158]:
```
# Create refrence date as timedelta + 2 days (i.e, T+2 days, as per industry␣
↪standards)
reference_date=df['InvoiceDate'].max() + dt.timedelta(days=2)
reference_date
```

[158]:
```
Timestamp('2011-12-11 12:50:00')
```

[159]:
```
# Find max number of  quantity sold & stockcode
max_quantity_row = df[df['Quantity'] == df['Quantity'].max()]
print(max_quantity_row)
```

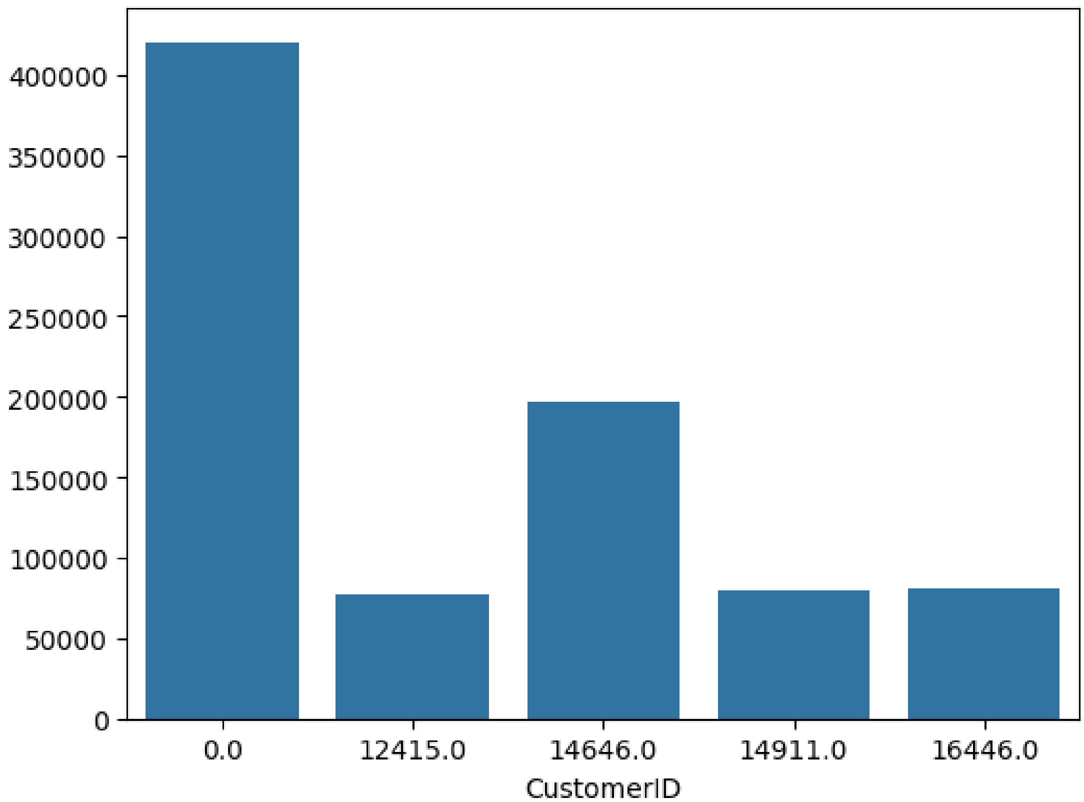
```
       InvoiceNo StockCode                  Description  Quantity  \
540421    581483     23843  PAPER CRAFT , LITTLE BIRDIE     80995

               InvoiceDate  UnitPrice  CustomerID         Country   Revenue
540421 2011-12-09 09:15:00       2.08     16446.0  United Kingdom  168469.6
```

[160]:
```
# groupby customerid
df.groupby('CustomerID')['Quantity'].sum()
sns.barplot(x=df.groupby('CustomerID')['Quantity'].sum().nlargest(5).index,y=df.
↪groupby('CustomerID')['Quantity'].sum().nlargest(5).values)

# Top 5 customerID
```

[160]:
```
<Axes: xlabel='CustomerID'>
```



Key Insights: 1. groupby func is used to group he data at customer level using Quantity sum. 2. TOp 5 customers data is fetched.

[161]:
```
df.groupby('CustomerID')['Quantity'].sum().nlargest(5)
```

[161]:
```
CustomerID
0.0        420564
14646.0    196915
16446.0     80997
14911.0     80265
12415.0     77374
Name: Quantity, dtype: int64
```

Key Insights:

1.Top 5 customers in descending order on the basis of Quantity purchased.

[162]:
```
df.groupby('Country')
```

[162]:
```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x7eceba071d20>
```

[163]:
```
df.groupby('Country')['Quantity'].sum().sort_values(ascending=False)
```

[163]:

```
Country
United Kingdom          4662390
Netherlands              200361
EIRE                     147173
Germany                  119261
France                   112103
Australia                 83901
Sweden                    36083
Switzerland               30629
Spain                     27940
Japan                     26016
Belgium                   23237
Norway                    19336
Portugal                  16258
Finland                   10704
Channel Islands            9491
Denmark                    8235
Italy                      8112
Cyprus                     6361
Singapore                  5241
Austria                    4881
Hong Kong                  4773
Israel                     4409
Poland                     3684
Unspecified                3300
Canada                     2763
USA                        2458
Iceland                    2458
Greece                     1557
United Arab Emirates        982
Malta                       970
Czech Republic              671
Lithuania                   652
European Community          499
Lebanon                     386
Brazil                      356
RSA                         351
Bahrain                     314
Saudi Arabia                 80
Name: Quantity, dtype: int64
```

[164]:

```
# number of countries
df.groupby('Country')['Quantity'].sum().sort_values(ascending=False).count()
```

[164]:

```
38
```

Key Insights: Company sells its products in 38 countries.

2 NOTE: Company name is assumed as Xpreskart.com

Key Insights: 1.The data belongs to xpreskart company located in United Kingdom 2.The xpreskart company sells products in 38 countries including United Kingdom

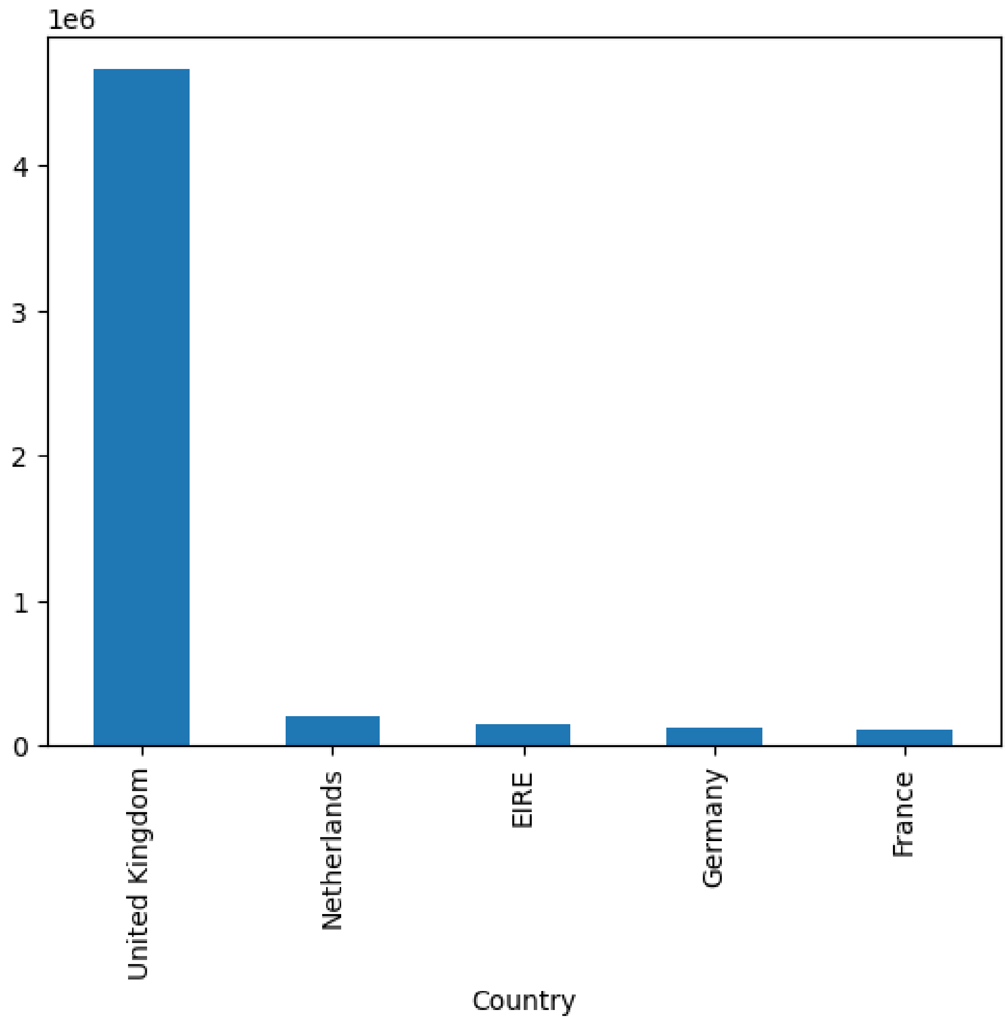
[165]:
```
# TOP 5 Countries

df.groupby('Country')['Quantity'].sum().nlargest(5)
```

[165]:
```
Country
United Kingdom    4662390
Netherlands        200361
EIRE               147173
Germany            119261
France             112103
Name: Quantity, dtype: int64
```

[166]:
```
df.groupby('Country')['Quantity'].sum().nlargest(5).plot(kind='bar')
```

[166]:
```
<Axes: xlabel='Country'>
```



[167]:
```
Country=df.groupby('Country')
```

[168]:
```
Country
```

[168]: <pandas.core.groupby.generic.DataFrameGroupBy object at 0x7eceb96ff670>

[169]:
```
Country.get_group('United Kingdom')
```

[169]:
```
      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
…             …        …                                    …         …
```

```
541889    581585     22466       FAIRY TALE COTTAGE NIGHT LIGHT        12
541890    581586     22061  LARGE CAKE STAND  HANGING STRAWBERY         8
541891    581586     23275      SET OF 3 HANGING OWLS OLLIE BEAK       24
541892    581586     21217         RED RETROSPOT ROUND CAKE TINS       24
541893    581586     20685                 DOORMAT RED RETROSPOT       10

               InvoiceDate  UnitPrice  CustomerID         Country  Revenue
0      2010-12-01 08:26:00       2.55     17850.0  United Kingdom    15.30
1      2010-12-01 08:26:00       3.39     17850.0  United Kingdom    20.34
2      2010-12-01 08:26:00       2.75     17850.0  United Kingdom    22.00
3      2010-12-01 08:26:00       3.39     17850.0  United Kingdom    20.34
4      2010-12-01 08:26:00       3.39     17850.0  United Kingdom    20.34
…                      …        …          …            …        …
541889 2011-12-09 12:31:00       1.95     15804.0  United Kingdom    23.40
541890 2011-12-09 12:49:00       2.95     13113.0  United Kingdom    23.60
541891 2011-12-09 12:49:00       1.25     13113.0  United Kingdom    30.00
541892 2011-12-09 12:49:00       8.95     13113.0  United Kingdom   214.80
541893 2011-12-09 12:49:00       7.08     13113.0  United Kingdom    70.80

[485123 rows x 9 columns]
```

Key Insights: 1.Most of the orders of Xpreskart company are from United Kingdom. 2.Second highest orders are from Netherlands.

Actionable Insights: 1.There is more scope of expansion of markets to european countries. 2.Focus on top 5 countires to increase sales 3.Further analysis like demography of top 5 countires is needed to understand customer behaviour. 4.Formulate marketing strtegies focused on top 10 countries. 5.Growth oppportunities are promising in TOP 5 countries.

[170]:
```python
# In this Dataset, The customers are from 38 Countries
len(Country)
```

[170]:
```
38
```

[171]:
```python
# List of all 38 Countries.
Country.size()
```

[171]:
```
Country
Australia                 1182
Austria                    398
Bahrain                     18
Belgium                   2031
Brazil                      32
Canada                     151
Channel Islands            748
Cyprus                     614
Czech Republic              25
Denmark                    380
```

```
EIRE                      7890
European Community          60
Finland                    685
France                    8407
Germany                   9040
Greece                     145
Hong Kong                  284
Iceland                    182
Israel                     295
Italy                      758
Japan                      321
Lebanon                     45
Lithuania                   35
Malta                      112
Netherlands               2359
Norway                    1071
Poland                     330
Portugal                  1501
RSA                         57
Saudi Arabia                 9
Singapore                  222
Spain                     2484
Sweden                     451
Switzerland               1966
USA                        179
United Arab Emirates        68
United Kingdom          485123
Unspecified                446
dtype: int64
```

[172]:
```python
# unique items(Stockcode)
df.StockCode.nunique()
```

[172]:
```
3922
```

There are 3922 unique items in the xpreskart Online Store (i.e stockCode)

[173]:
```python
df.StockCode.value_counts()
```

[173]:
```
StockCode
85123A      2265
85099B      2112
22423       2017
47566       1706
20725       1595
            …
DCGS0004       1
```

```
84705C        1
20964         1
72803b        1
23843         1
Name: count, Length: 3922, dtype: int64
```

[174]:
```
#TOP 5 products  StockCode,Description and  Quantity sold

df.groupby(['StockCode','Description'])['Quantity'].sum().nlargest(5)
```

[174]:
```
StockCode  Description
23843      PAPER CRAFT , LITTLE BIRDIE           80995
23166      MEDIUM CERAMIC TOP STORAGE JAR        78033
84077      WORLD WAR 2 GLIDERS ASSTD DESIGNS     55047
85099B     JUMBO BAG RED RETROSPOT               48474
85123A     WHITE HANGING HEART T-LIGHT HOLDER    37599
Name: Quantity, dtype: int64
```

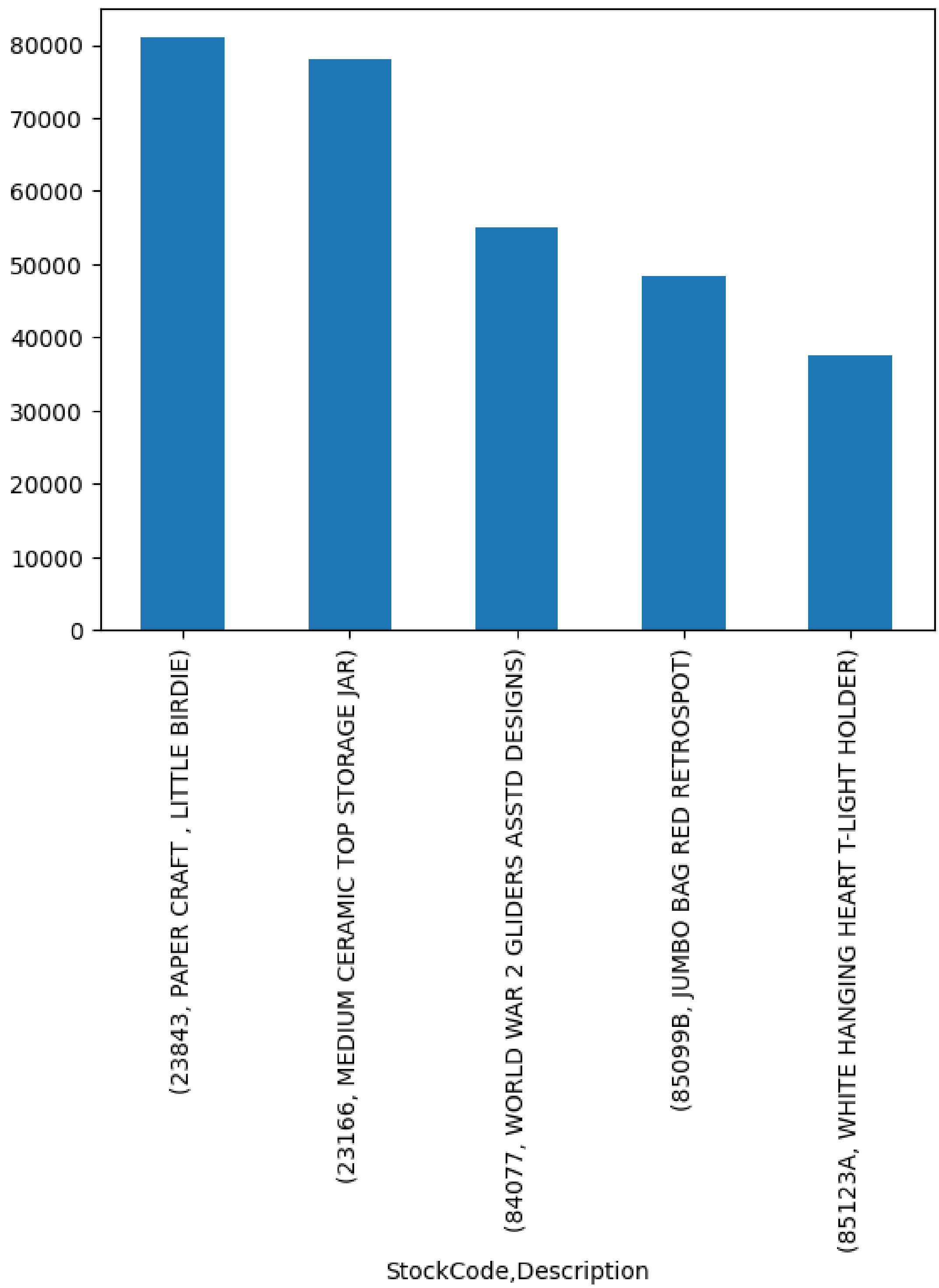
Key Insights: 1.Top 5 product with stockcode are fetched from above query.

Actionable Insights: 1. More analysis is needed to understand the top selling products. 2.Further analysis can be done to understand top 10 selling products in top 10 countries.

[175]:
```
# using bar plot to show data.

df.groupby(['StockCode','Description'])['Quantity'].sum().nlargest(5).
 ↪plot(kind='bar')
```

[175]:
```
<Axes: xlabel='StockCode,Description'>
```



Key Insights: 1.Above graph clearly shows top 5 products with description & stockCode.

Actionable Insights: 1. Furhter analysis is needed to understand the profit contribution of top 5 products in the company revenue. 2.Reveune of top 5 products in top 5 countires to be ascetained for understaing buying pattern & customer behaviuor.

[176]:
```
# correlation between Quantity & Unitprice
df.Quantity.corr(df.UnitPrice)
```

[176]: -0.0037725426616366953

Negative Correlation. i.e, There is NO Correlation between Quantity & UnitPrice

[177]:
```
numerical_df = df.select_dtypes(include=['number'])
sns.heatmap(numerical_df.corr(), annot=True)
```

[177]: <Axes: >

Key Insights:

1.Heat map shows, There is Negative correlation between variables, Unit Price & Quantity (i.e-0.0038).

Actionable Insights: 1. There is a strong need to revise pricing strategies. 2. Optimum price to be fixed to get more sales/Revenue. 3.Promotion of top 10 selling products need to be done.

[178]:
```
# RFM Analysis
# Recency, Frequency, Monetary
df.head()
```

[178]:
```
  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  Revenue
0 2010-12-01 08:26:00       2.55     17850.0  United Kingdom    15.30
1 2010-12-01 08:26:00       3.39     17850.0  United Kingdom    20.34
2 2010-12-01 08:26:00       2.75     17850.0  United Kingdom    22.00
3 2010-12-01 08:26:00       3.39     17850.0  United Kingdom    20.34
4 2010-12-01 08:26:00       3.39     17850.0  United Kingdom    20.34
```

3 Key Insights on RFM:

1.RFM, (Recency, Frequency, Monetary) 2.RFM Analysis is done to understand the customer behaviour 3.RFM analysis helps in customer segmentation based on RFM Score 4.RFM analysis helps is target marketing 5.RFM analysis ensures the right marketing strategies for bucket of customers. 6.RFM analysis gives insights on Recency i.e recent transaction of customer buying from store or Number of days since last purchase , Frequency; How often customer visits the store and Monetary; How much money customer spends for the transactions.

[179]:
```
# RFM

rfm=df.groupby('CustomerID').agg({'InvoiceDate': lambda x:(reference_date - x.
 ↪max()).days,'InvoiceNo':'count','Revenue':'sum'})
rfm
```

[179]:
```
            InvoiceDate  InvoiceNo     Revenue
CustomerID
0.0                   2     132220  1755276.64
12346.0             327          1    77183.60
12347.0               3        182     4310.00
12348.0              76         31     1797.24
12349.0              20         73     1757.55
…                   …          …         …
18280.0             279         10      180.60
18281.0             182          7       80.82
18282.0               9         12      178.05
18283.0               5        756     2094.88
18287.0              44         70     1837.28
```

```
[4339 rows x 3 columns]
```

Key Insights: 1.cretd New dataframe 'rfm' 2.Grouping customers on CustomerID to to get the toal revenue generated from a customer from all transactions.

Actionable Insights: 1.It evident from analysis that, CustomerID 12346 is th top spending customer.

[180]:
```
# Renaming columns for further analysis.

rfm.rename(columns={'InvoiceDate':'Recency','InvoiceNo':'Frequency','Revenue':
 ↪'Monetary_value'},inplace=True)
```

[181]:
```
rfm.head()
```

[181]:
```
            Recency  Frequency  Monetary_value
CustomerID
0.0               2     132220      1755276.64
12346.0         327          1        77183.60
12347.0           3        182         4310.00
12348.0          76         31         1797.24
12349.0          20         73         1757.55
```

Key Insights: 1.Columns have been renamed ('InvoiceDate':'Recency','InvoiceNo':'Frequency','Revenue':'Monetary_value') for RFM analysis.

Actionable Insights: 1. New Columns Recency Frequency & Monetary _value help in RFM analysis.

[182]:
```
# drop CustomerID =0.0
rfm=rfm[rfm.index!=0.0]
```

NOTE: DATA of CustomerID == 0.0 looks suspicious/outlier, may be used for test cases, hence CustomerID 0.0 is dropped for further analysis.

[183]:
```
rfm.head()
```

[183]:
```
            Recency  Frequency  Monetary_value
CustomerID
12346.0         327          1        77183.60
12347.0           3        182         4310.00
12348.0          76         31         1797.24
12349.0          20         73         1757.55
12350.0         311         17          334.40
```

Key Insights: 1. CutomerID 12346 is the most spending customer with Monetary_value 77183.60 but its frequency is low. 2. customerID 12347 is the most frequent customer with frequency value 182 3. CustomerID 12347 is also the most recent customer with recency value 3.

Actionable Insights: 1.Further analysis is recommended to understand the distribution of RFM scores of customers and insights on customer behaviour.

[184]:
```
# Create Quantiles
quantiles=rfm.quantile(q=[0.25,0.5,0.75])
quantiles
```

[184]:

	Recency	Frequency	Monetary_value
0.25	19.0	17.0	307.415
0.50	52.0	41.0	674.485
0.75	143.0	100.0	1661.740

Key Insights: 1. Quartile distribution of RFM values.

Key Insights: 1.

[185]:
```
# Create RFM scores

def r_score(x,p,d):
  if p=='Recency':
    if x<=d[p][0.25]:
      return 4
    elif x<=d[p][0.50]:
      return 3
    elif x<=d[p][0.75]:
      return 2
    else:
      return 1
  else:
    if x<=d[p][0.25]:
      return 1
    elif x<=d[p][0.50]:
      return 2
    elif x<=d[p][0.75]:
      return 3
    else:
      return 4
```

[186]:
```
rfm['R']=rfm['Recency'].apply(r_score,args=('Recency',quantiles))
rfm['F']=rfm['Frequency'].apply(r_score,args=('Frequency',quantiles))
rfm['M']=rfm['Monetary_value'].apply(r_score,args=('Monetary_value',quantiles))

rfm.head()
```

[186]:

	Recency	Frequency	Monetary_value	R	F	M
CustomerID						
12346.0	327	1	77183.60	1	1	4
12347.0	3	182	4310.00	4	4	4
12348.0	76	31	1797.24	2	2	4
12349.0	20	73	1757.55	3	3	4
12350.0	311	17	334.40	1	1	2

4 Key Insights:

5 1.Customers with the lowest recency , highest frequency and monetary_value considered as top customers.

2. Quantile-based discretization function, bins the data based on sample quantiles.

[187]:
```
pd.options.mode.chained_assignment = None  # default='warn'
```

[188]:
```
rfm.head()
```

[188]:
```
             Recency  Frequency  Monetary_value  R  F  M
CustomerID
12346.0          327          1        77183.60  1  1  4
12347.0            3        182         4310.00  4  4  4
12348.0           76         31         1797.24  2  2  4
12349.0           20         73         1757.55  3  3  4
12350.0          311         17          334.40  1  1  2
```

Key Insights:

[189]:
```
rfm['RFM_Segments']=rfm.R.map(str)+rfm.F.map(str)+rfm.M.map(str)
rfm['RFM_Score']=rfm[['R','F','M']].sum(axis=1)
rfm.head()
```

[189]:
```
             Recency  Frequency  Monetary_value  R  F  M RFM_Segments  \
CustomerID
12346.0          327          1        77183.60  1  1  4          114
12347.0            3        182         4310.00  4  4  4          444
12348.0           76         31         1797.24  2  2  4          224
12349.0           20         73         1757.55  3  3  4          334
12350.0          311         17          334.40  1  1  2          112

             RFM_Score
CustomerID
12346.0              6
12347.0             12
12348.0              8
12349.0             10
12350.0              4
```

Key Insights: Create a new column called "RFM_Score" in the "rfm" dataframe. The values in this column are created by concatenating the values in the "R", "F", and "M" columns, which are assumed to be numerical values representing quartiles.

The "(str)" method is used to convert these numerical values to strings before concatenating them. The resulting string values represent the RFM score for each customer, which is a way of segmenting customers based on their recency, frequency, and monetary value.

[190]:
```python
# Crete Labels for Segments.

Label_Segments=['Low_value','Mid_value','High_value']

# creating a Function.

def assign_segments(score):
  if score <5:
    return 'Low_value'
  elif score  <9:
    return 'Mid_value'
  else:
    return 'High_value'

# creating new column RFM_Segements_label & applying the function␣
 ↪assign_segments

rfm['RFM_Segements_label']=rfm['RFM_Score'].apply(assign_segments)

rfm.head()
```

[190]:
```
            Recency  Frequency  Monetary_value  R  F  M RFM_Segments  \
CustomerID
12346.0         327          1        77183.60  1  1  4          114
12347.0           3        182         4310.00  4  4  4          444
12348.0          76         31         1797.24  2  2  4          224
12349.0          20         73         1757.55  3  3  4          334
12350.0         311         17          334.40  1  1  2          112

            RFM_Score RFM_Segements_label
CustomerID
12346.0             6           Mid_value
12347.0            12          High_value
12348.0             8           Mid_value
12349.0            10          High_value
12350.0             4           Low_value
```

Key Insights: 1. Creatd new column as RFM_Segements_label to to assign values High_value,Mid_value,Low_value.

[191]:
```python
segments_count=rfm['RFM_Segements_label'].value_counts().reset_index()
segments_count.columns=['RFM_Segements','Count']
segments_count=segments_count.sort_values('RFM_Segements')
```

[192]:
```python
segments_count
```

[192]:

	RFM_Segements	Count
1	High_value	1682
2	Low_value	767
0	Mid_value	1889

Key Insights:

1.created new dataframe as segments_count 2.New column created as RFM_Segements with corresponding count

Actionable Insights:

1. Bins created High_value,Mid_value& Low_values with counts for firther analysis.

[193]:
```
# create bar plot using plotly library
# create bar plot
fig = px.bar(segments_count,
             x='RFM_Segements',
             y='Count',
             title='Customer Distribution by RFM Segments',
             labels={'RFM_Segements':'RFM Segments','count':'Number of␣
 ↪Customers'},
             color='RFM_Segements',
             color_discrete_sequence=px.colors.qualitative.Prism)

fig.show()
```

Key Insights:

the above bar plot shows the bins created as High_value,Mid_value & Low_value customers with respective counts.

[194]:
```
rfm['RFM_Customer_Segments']=''

rfm.loc[rfm['RFM_Score']>=9,'RFM_Customer_Segments']='VIP_Customers'
rfm.loc[(rfm['RFM_Score']>=6) &␣
 ↪(rfm['RFM_Score']<9),'RFM_Customer_Segments']='Loyal_Customers'
rfm.loc[(rfm['RFM_Score']>=5) &␣
 ↪(rfm['RFM_Score']<6),'RFM_Customer_Segments']='Potential_Loyalists'
rfm.loc[(rfm['RFM_Score']>=4) &␣
 ↪(rfm['RFM_Score']<5),'RFM_Customer_Segments']='Cant_lose'
rfm.loc[(rfm['RFM_Score']>=3) &␣
 ↪(rfm['RFM_Score']<4),'RFM_Customer_Segments']='Lost_Customers'
segments_count = rfm['RFM_Customer_Segments'].value_counts().sort_index()
```

Key Insights: 1. Created Customer RFM segments by respective values. 2.Customer segments based on range of RFM_Score and labelled as; RFM_score >9 == VIP_customers RFM_score >=6 & <9 == Loyal_customers RFM_score >=5 & <6 == Potential_loyalists RFM_score >=4 & <5 == Cant_lose (about to lose) RFM_score >=3 & <4 == 'Lost_customers'

[195]:
```
segments_product_count = rfm.
 ↪groupby(['RFM_Segements_label','RFM_Customer_Segments']).size().
 ↪reset_index(name='Count')
segments_product_count = segments_product_count.sort_values('Count',␣
 ↪ascending=False)
```

[196]:
```
# Create a Tree map

fig_treemap_product_segments=px.treemap(segments_product_count,
                                                              ␣
 ↪path=['RFM_Segements_label','RFM_Customer_Segments'],
                                        values='Count',
                                        color='RFM_Segements_label',
                                        color_discrete_sequence=px.colors.
 ↪qualitative.Prism,
                                        title='Customer RFM Segments by Values')
```

[197]:
```
# Display tree map

fig_treemap_product_segments.show()
```

Key Insights:

1. The above treemap visualisation clearly inidicates the customer segments and their respective RFM scores & buckets they belong to.

Actionable Insights: 1.It is evident from above visualisation that VIP_customers are 1682 with RFm segments belongin to High_value. 2. It is evident from above visualisation that loyal customers are 1373 with RFm segments Mid_value 3.Potential Loyalist count of customers is 516 & belong to Mid_value segment 4.It is evident from above visualisation that cant_lose customers are 382 with RFm segments LOW_value 5.It is evident from above visualisation that lost_customers are 385 with RFm segments belongin to Low_value

[198]:
```
# Create VIP dataFrame

vip_segment=rfm[rfm['RFM_Customer_Segments']=='VIP_Customers']
vip_segment.head()
```

[198]:
```
            Recency  Frequency  Monetary_value  R  F  M RFM_Segments  \
CustomerID
12347.0           3        182         4310.00  4  4  4          444
12349.0          20         73         1757.55  3  3  4          334
12352.0          37         85         2506.04  3  3  4          334
12356.0          24         59         2811.43  3  3  4          334
12357.0          34        131         6207.67  3  4  4          344

            RFM_Score RFM_Segements_label RFM_Customer_Segments
CustomerID
```

```
12347.0              12           High_value          VIP_Customers
12349.0              10           High_value          VIP_Customers
12352.0              10           High_value          VIP_Customers
12356.0              10           High_value          VIP_Customers
12357.0              11           High_value          VIP_Customers
```

Key Insights:

Creted new dataframe vip_segment for furhter analysis

Actionable Insights: 1.new dataframe vip_segement created to deep dive into only VIP_customers analysis. 2.Customer ID 12347,Max RFM_score fro VIP_customer R+F+M i.e (4+4+4=12) 3.TOP 5 High_value VIP_customer can be ascertained from above dataframe.

[199]:
```
# Create box plot

fig=go.Figure()
fig.add_trace(go.Box(y=vip_segment['Recency'],name='Recency'))
fig.add_trace(go.Box(y=vip_segment['Frequency'],name='Frequency'))
fig.add_trace(go.Box(y=vip_segment['Monetary_value'],name='Monetary_value'))
```

Key Insights: Further analysis is needed to understand the Outliers.

** Actionable Insights:** 1. customer with monetary _Value 280k looks like test case. 2.Furhter analysis is need with stake holers to undrstand the authenticiy of outliers data.

[200]:
```
# RFM correlation matrix

correlation_matrix = vip_segment[['R', 'F', 'M']].corr()
correlation_matrix
```

[200]:
```
          R         F         M
R  1.000000 -0.115392 -0.096888
F -0.115392  1.000000  0.278743
M -0.096888  0.278743  1.000000
```

[201]:
```
# Create heat map of correlation

fig_heat_map = go.Figure(data=go.Heatmap(
                   z=correlation_matrix.values,
                   x=correlation_matrix.columns,
                   y=correlation_matrix.columns,
                   colorscale='Viridis',
                   colorbar=dict(title='Correlation')))
fig_heat_map.update_layout(title='Correlation matrix of RFM Heatmap ')
fig_heat_map.show()
```

Key Insights: NO correlation between variable. in other words: Negative correlation (z= -0.1369)observed bewteen Frequency & Recency. AND Monetary & Recency (z= -0.1297).

[202]:
```
# Creat comparison of RFM Segments

pastel_colors = pc.qualitative.Pastel

fig=go.Figure(data=[go.Bar(x=segments_count.index,
                           y=segments_count.values,
                           marker=dict(color=pastel_colors))])

my_color='rgb(158,202,225)'

fig.update_traces(marker_color=[my_color if segment == 'Champions' else␣
 ↪pastel_colors[i]
                                for i, segment in enumerate(segments_count.
 ↪index)],
                  marker_line_color='rgb(8,48,107)',
                  marker_line_width=1.5,opacity=0.5)

# Update the layout

fig.update_layout(title='Customer Comparision by RFM Segments',
                  xaxis_title='RFM Segments',
                  yaxis_title='Number of Customers',
                  showlegend=False)

#Display the figure
fig.show()
```

Key Insights: 1.Anove Visualiation graph dipicts the comparision of all customer segments/Bins.

Actionable Insights: 1. Further analysis is needed to prevent customer segment "Cant_lose", from churn. 2.Target Marketing strategies to be implemented absed on RFM_Customer segements 3.More indepth analysis of customer segementation can reveal new insights on customer behaviuor & buying patterns.

[203]:
```
RFM_segments_Score=rfm.groupby('RFM_Customer_Segments')[['R','F','M']].mean().
 ↪reset_index()
RFM_segments_Score
```

[203]:

	RFM_Customer_Segments	R	F	M
0	Cant_lose	1.468586	1.259162	1.272251
1	Lost_Customers	1.000000	1.000000	1.000000
2	Loyal_Customers	2.442826	2.253460	2.299345
3	Potential_Loyalists	1.720930	1.641473	1.637597
4	VIP_Customers	3.398930	3.567182	3.550535

Key Insights: 1. Mean value of RFM score for RF_customer segemnts can be observed.

mean value of 3.55 is observed for VIP_customers and mean value 1.00 for lost customer .

[204]:

```
# Create bar chart for easy visulisation of RFM Analysis

RFM_segments_Score=rfm.groupby('RFM_Customer_Segments')[['R','F','M']].mean().
↪reset_index()

fig=go.Figure()

# Add Bar chart for Recency Score
fig.add_trace(go.Bar(x=RFM_segments_Score['RFM_Customer_Segments'],
                     y=RFM_segments_Score['R'],
                     name='Recency Score',
                     marker_color='rgb(55, 83, 109)'))

#ADD Bar chart for Frequency Score
fig.add_trace(go.Bar(x=RFM_segments_Score['RFM_Customer_Segments'],
                     y=RFM_segments_Score['F'],
                     name='Frequency Score',
                     marker_color='rgb(26, 118,217)'))

#ADD Bar chart for Monetary Score
fig.add_trace(go.Bar(x=RFM_segments_Score['RFM_Customer_Segments'],
                     y=RFM_segments_Score['M'],
                     name='Monetary Score',
                     marker_color='rgb(32,102,48)'))

# Update layout
fig.update_layout(
    title='Comparision of RFM Score by Customer Segments',
    xaxis_title='Customer RFM Segments',
    yaxis_title='RFM Score',
    barmode='group',
    showlegend=True
)

#Display fig

fig.show()
```

Key Insights:

1. The above data visualiation with the help of bar chart shows the customer RFM segementation with RFM scores respectively.

Actionable Insights: 1. VIP_customers: mean values of Recency Score is 3.39,Frequency is 3.56 & Monetary_Value 3.55 of VIP_customers respectively.

2.Loyal_Customers:mean values of Recency Score is 2.44,Frequency is 2.25 & Monetary_Value 2.29 respectively.

3.Potential_Loyalist:mean values of Recency Score is 1.72,Frequency is 1.64 & Monetary_Value 1.73 respectively.

4.Cant_Lose:mean values of Recency Score is 1.46,Frequency is 1.25 & Monetary_Value 1.27 respectively.

5.Lost_customers:mean values of Recency Score is 1,Frequency is 1 & Monetary_Value 1 respectively.

6 RECOMMENDATIONS:

Further data analysis of all RFM_Customer_Segement can help the Xpreskart company to 1.Reduce the chrun of customers
2.Increase revenue & Profits. 3.Facilitate in formulating marketing strategies 4.Help in target marketing 5.ROI on marketing of RFM customer segments can be ascertained.

[204]:

```

```