Name: VARUN D S

RETAIL PULSE ASSIGNMENT (SECTION-1)

When working with the dataset, we should follow some major steps. Step 1: Ask The Right Questions. Step 2: Data Collection. Step 3: Data Cleaning Step 4: Analyzing The Data Step 5: Interpreting The Results.

Step 1: Ask The Right Questions

Asking the right question,will help us in gaining or extracting the more information or the more insights from the raw daraset.

Questions

1. Based on your understanding of the data, what kind of business is this company in? 2. Analyze the sales performance of this company, and provide your insights regarding the same 3. Based on your analysis of the data, what are potential areas of improvement for the business? 4. What are additional business problems that can be analyzed using this data. Support with explanation. The Answers are provided below with proper analysis. 5. Can customers be segmented into different categories? If yes then perform analysis on the same and also propose categories. If no, then explain why? 6. How would you define a loyal customer? 7. What is the most popular time of year based on this sales data? 8. Is there any seasonality in data? Explain with supportive evidence. 9. Discuss customers lifetime with respect to the given dataset.

In [ ]:
```
# importing the Library
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import os
from pandas import ExcelFile
import datetime as dt
from datetime import date
import warnings
warnings.filterwarnings("ignore")

%matplotlib inline
```

In [6]:
```
df = pd.read_excel(r'C:\Users\VARUN D S\Downloads\RetailPulseAssignmentData.xlsx')
```

In [7]:
```
df.head()
```

Out[7]:

	Invoice	StockCode	Description	Quantity	InvoiceDate	Price	Customer ID	Country
0	489434	85048	15CM CHRISTMAS GLASS BALL 20 LIGHTS	12	2009-12-01 07:45:00	6.95	13085.0	United Kingdom
1	489434	79323P	PINK CHERRY LIGHTS	12	2009-12-01 07:45:00	6.75	13085.0	United Kingdom
2	489434	79323W	WHITE CHERRY LIGHTS	12	2009-12-01 07:45:00	6.75	13085.0	United Kingdom
3	489434	22041	RECORD FRAME 7" SINGLE SIZE	48	2009-12-01 07:45:00	2.10	13085.0	United Kingdom
4	489434	21232	STRAWBERRY CERAMIC TRINKET BOX	24	2009-12-01 07:45:00	1.25	13085.0	United Kingdom

In [8]:
```
ID','Invoice','StockCode','Quantity','Price','Description','InvoiceDate'
```

In [9]:
```
df.head()
```

Out[9]:

	Customer ID	Invoice	StockCode	Quantity	Price	Description	InvoiceDate	Country
0	13085.0	489434	85048	12	6.95	15CM CHRISTMAS GLASS BALL 20 LIGHTS	2009-12-01 07:45:00	United Kingdom
1	13085.0	489434	79323P	12	6.75	PINK CHERRY LIGHTS	2009-12-01 07:45:00	United Kingdom
2	13085.0	489434	79323W	12	6.75	WHITE CHERRY LIGHTS	2009-12-01 07:45:00	United Kingdom
3	13085.0	489434	22041	48	2.10	RECORD FRAME 7" SINGLE SIZE	2009-12-01 07:45:00	United Kingdom
4	13085.0	489434	21232	24	1.25	STRAWBERRY CERAMIC TRINKET BOX	2009-12-01 07:45:00	United Kingdom

In [10]:
```
# Data Preprocessing
df.isnull().sum()
```

Out[10]:
```
Customer ID    107927
Invoice             0
StockCode           0
Quantity            0
Price               0
Description      2928
InvoiceDate         0
Country             0
dtype: int64
```

Question 1. Based on our understanding of the data, what kind of business is this company in?

Insight

Answer - As the name suggest this company is more into online retail across 38 countries like in below

In [11]:
```
df.Country.unique()
```

Out[11]:
```
array(['United Kingdom', 'France', 'USA', 'Belgium', 'Australia', 'EIRE',
       'Germany', 'Portugal', 'Japan', 'Denmark', 'Nigeria',
       'Netherlands', 'Poland', 'Spain', 'Channel Islands', 'Italy',
       'Cyprus', 'Greece', 'Norway', 'Austria', 'Sweden',
       'United Arab Emirates', 'Finland', 'Switzerland', 'Unspecified',
       'Malta', 'Bahrain', 'RSA', 'Bermuda', 'Hong Kong', 'Singapore',
       'Thailand', 'Israel', 'Lithuania', 'West Indies', 'Lebanon',
       'Korea', 'Brazil', 'Canada', 'Iceland'], dtype=object)
```

Question 2. Analyze the sales performance of the company, and provide your insights regarding the same

Analysis

```
TotalAmount = df['Quantity'] * df['Price']
df.insert(loc=5,column='TotalAmt',value=TotalAmount)
```

In [13]:
```
new_df = df[['Customer ID','Invoice','StockCode','Quantity','TotalAmt','InvoiceDate','Cou

new_df2 = df.copy()
```

In [14]:
```
new_df.head()
```

Out[14]:

	Customer ID	Invoice	StockCode	Quantity	TotalAmt	InvoiceDate	Country
0	13085.0	489434	85048	12	83.4	2009-12-01 07:45:00	United Kingdom
1	13085.0	489434	79323P	12	81.0	2009-12-01 07:45:00	United Kingdom
2	13085.0	489434	79323W	12	81.0	2009-12-01 07:45:00	United Kingdom
3	13085.0	489434	22041	48	100.8	2009-12-01 07:45:00	United Kingdom
4	13085.0	489434	21232	24	30.0	2009-12-01 07:45:00	United Kingdom

Exploratory Data Analysis(EDA)

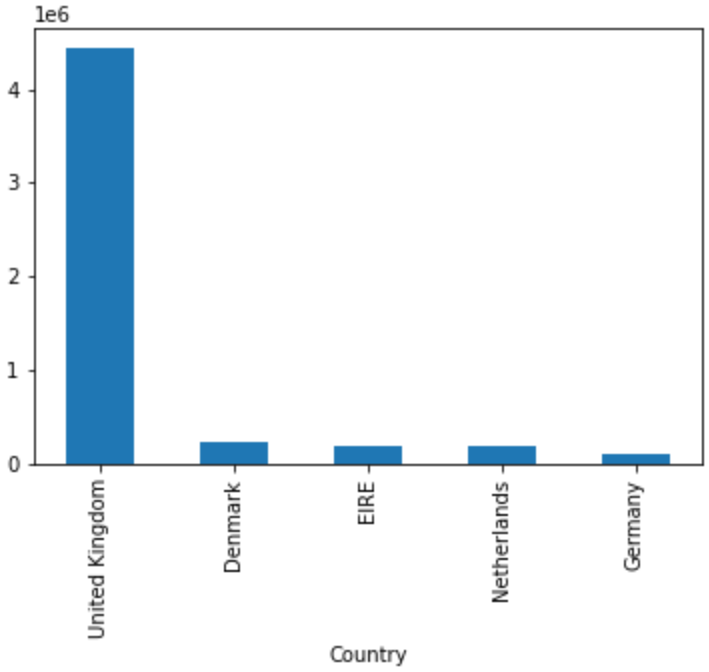
In [15]:
```
# Grouping countries by TotalAmount of sales

country_price = new_df.groupby('Country')['Quantity'].sum().sort_values(ascending = False
country_price
```

Out[15]:
```
Country
United Kingdom          4429046
Denmark                  227030
EIRE                     188704
Netherlands              181823
Germany                  107133
France                    74471
Sweden                    52238
Switzerland               22053
Australia                 20053
Spain                     18332
Belgium                   11980
Portugal                  11878
Channel Islands           10994
Norway                     7863
Italy                      7310
Japan                      6604
Austria                    6479
Greece                     6151
United Arab Emirates       5746
Cyprus                     4371
Finland                    3651
Unspecified                3416
Bermuda                    2798
USA                        2666
Thailand                   2552
Lithuania                  2306
Hong Kong                  2306
Poland                     1991
Singapore                  1753
RSA                        1618
Malta                      1547
Israel                     1132
Bahrain                    1015
                            894
```

```
Iceland                  828
Korea                    598
West Indies              395
Brazil                   189
Lebanon                   71
Nigeria                   56
Name: Quantity, dtype: int64
```

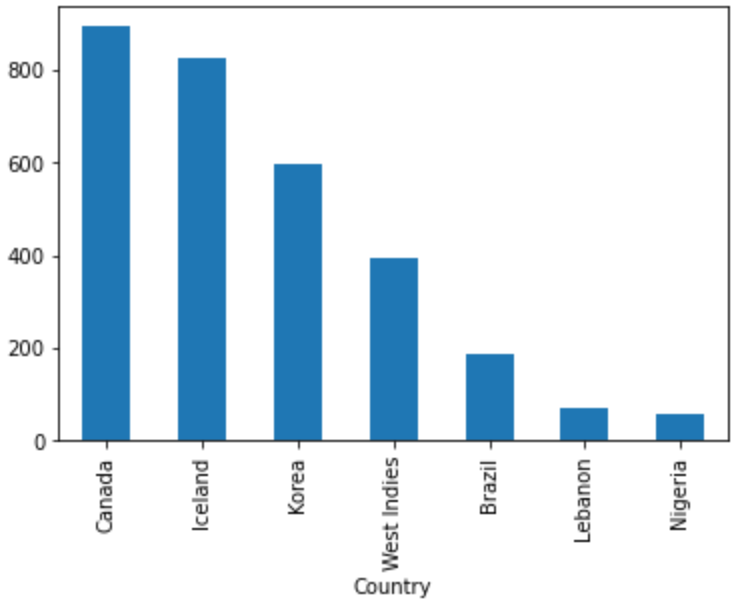
In [16]:
```
# Top 5 Country with high number of purchase
country_price[:5].plot(kind = 'bar')
```

Out[16]:
```
<AxesSubplot:xlabel='Country'>
```



In [17]:
```
# 5 Country with least number of purchase
country_price[33:].plot(kind = 'bar')
```

Out[17]:
```
<AxesSubplot:xlabel='Country'>
```



In [18]:
```
# Adding year feature to the dataset

                'InvoiceDate'].dt.year
```

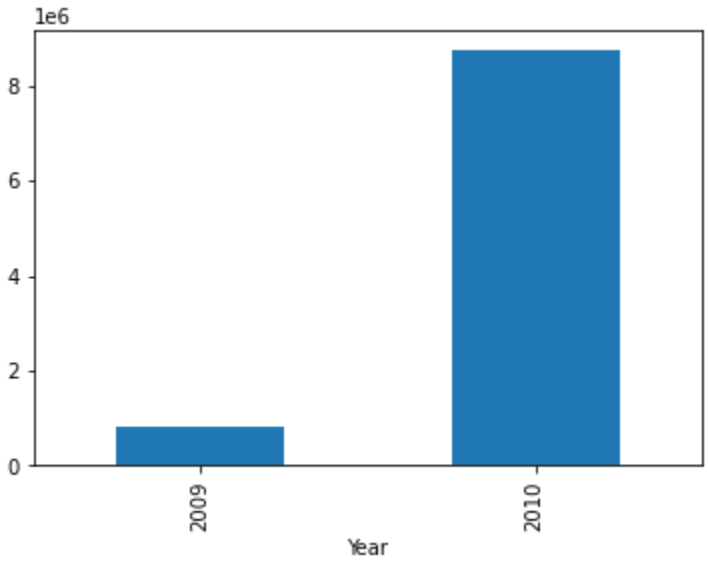
```
new_df['Year'] = timest

new_df.head()
```

Out[18]:

	Customer ID	Invoice	StockCode	Quantity	TotalAmt	InvoiceDate	Country	Year
0	13085.0	489434	85048	12	83.4	2009-12-01 07:45:00	United Kingdom	2009
1	13085.0	489434	79323P	12	81.0	2009-12-01 07:45:00	United Kingdom	2009
2	13085.0	489434	79323W	12	81.0	2009-12-01 07:45:00	United Kingdom	2009
3	13085.0	489434	22041	48	100.8	2009-12-01 07:45:00	United Kingdom	2009
4	13085.0	489434	21232	24	30.0	2009-12-01 07:45:00	United Kingdom	2009

In [19]:
```
# Total sales for different years

new_df.groupby('Year')['TotalAmt'].sum().plot(kind = 'bar')
```

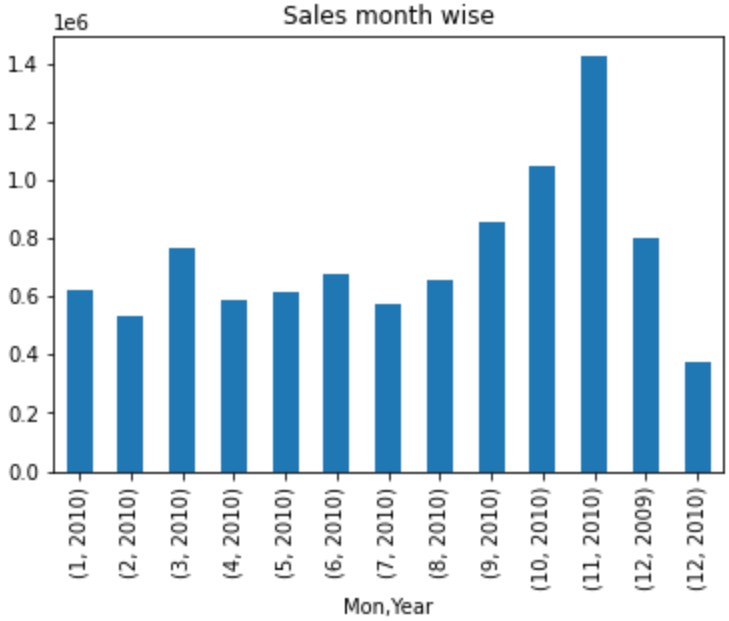
Out[19]:
```
<AxesSubplot:xlabel='Year'>
```



Question 7. What is the most popular time of year based on this sales data?

In [20]:
```
# Sales for different months

new_df['Mon'] = new_df['InvoiceDate'].dt.month
new_df['month'] = new_df['InvoiceDate'].dt.month_name()
new_df.groupby(['Mon','Year'])['TotalAmt'].sum().plot(kind = 'bar', title = 'Sales month
```

Out[20]:
```
<AxesSubplot:title={'center':'Sales month wise'}, xlabel='Mon,Year'>
```



Insight

The most popular time of the year is NOVEMBER 2010. And also, DECEMBER 2010 because even though we have only nine days data still the sales are high

In [ ]:

```
# Checking why dec 2010 has a drop comparing to nov 2010
get_2010 = new_df[(new_df['Year'] == 2010)]
get_dec2010 = get_2010[(new_df['month'] == 'December')]
get_dec2010 = get_dec2010['InvoiceDate'].dt.date.unique()
get_dec2010
```

SEASONALITY: It is the predictable pattern that repeats at a certain frequency within one year, such as weekly, monthly, quarterly,

According to my observstion there is no need to plot the time series graphs beacuse from the above graph (Sales for different months) only we can clearly understand that from september to december there is a consecutive increase in the sales which is from lower to higher we can find the seasonality there.Above graph is the evidence

Insight

Answer : Performance Analysis Sales Performance can be seen with Number of sales every month in the sales for different months above. Number of sales every year in the total sales for different years. We see that in 2009 we have sales only for dec and in 2010 we have sales for all months We can see that from September to Novembor we have very good sales. We could see that DEC 2009 we have more sales and in DEC 2010 we have less sales which is a drop when analyzed further found out that We have only data upto 9th on dec 2010, so we find a sales drop in the month of dec 2010

Question 3. Based on your analysis of the data, what are

potential areas of improvement for the business?

In [22]:
```
new_df.head()
```

Out[22]:

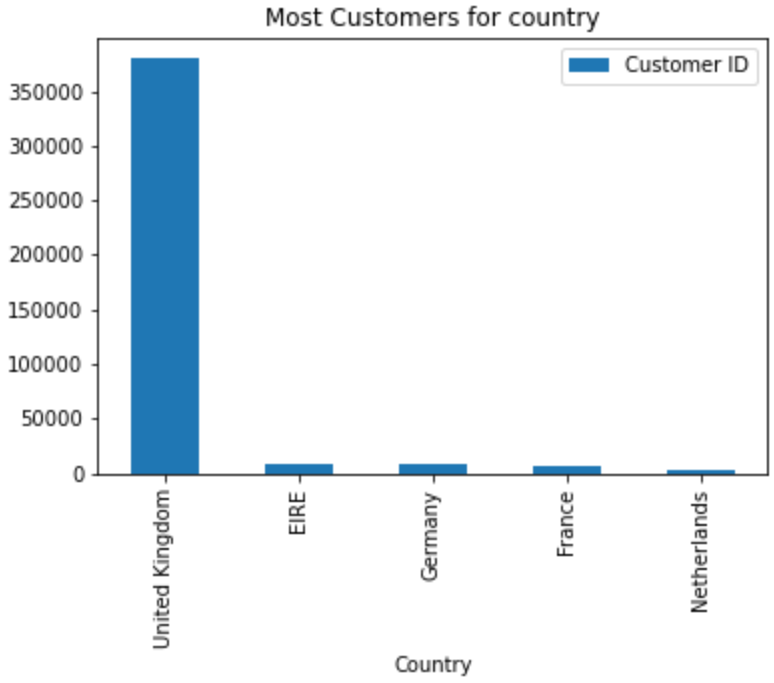
	Customer ID	Invoice	StockCode	Quantity	TotalAmt	InvoiceDate	Country	Year	Mon	month
0	13085.0	489434	85048	12	83.4	2009-12-01 07:45:00	United Kingdom	2009	12	December
1	13085.0	489434	79323P	12	81.0	2009-12-01 07:45:00	United Kingdom	2009	12	December
2	13085.0	489434	79323W	12	81.0	2009-12-01 07:45:00	United Kingdom	2009	12	December
3	13085.0	489434	22041	48	100.8	2009-12-01 07:45:00	United Kingdom	2009	12	December
4	13085.0	489434	21232	24	30.0	2009-12-01 07:45:00	United Kingdom	2009	12	December

In [23]:
```
new_df = new_df.dropna()
new_df.isnull().sum()
```

Out[23]:
```
Customer ID    0
Invoice        0
StockCode      0
Quantity       0
TotalAmt       0
InvoiceDate    0
Country        0
Year           0
Mon            0
month          0
dtype: int64
```

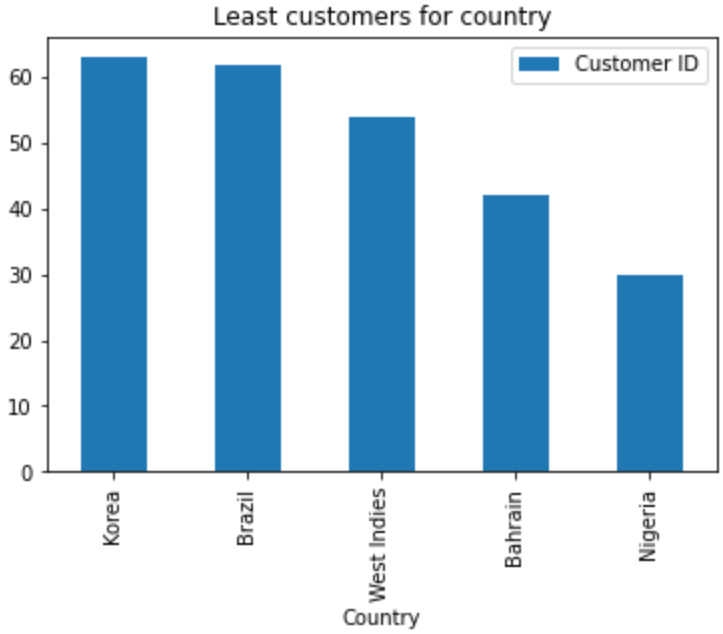
In [24]:
```
#Countries with more number of customers
cus_id = pd.DataFrame(new_df.groupby('Country')['Customer ID'].count().sort_values(ascend
cus_id[:5].plot(kind = 'bar', title = 'Most Customers for country')
```

Out[24]:
```
<AxesSubplot:title={'center':'Most Customers for country'}, xlabel='Country'>
```



In [25]:
```
# Countries with less number of customers
cus_id[-5:].plot(kind = 'bar', title = 'Least customers for country')
```

Out[25]: <AxesSubplot:title={'center':'Least customers for country'}, xlabel='Country'>



Insight

Answer- 1.We see that september to december we have very high sales 2.We can concentrate on improving the sales for the other 8 months 3.We find very less number of customers in Korea,Brazil,West Indies,Bahrain,Nigeria. 4.We have very less sales for Canada Iceland,Korea,West Indies,Brazil,Lebanon,Nigeria. We can concentrate on improving the sales We could improve the business by increasing the customers and sales point of view

In [26]:
```
# Removing the null values since we are checking the data based on customer and descripti
new_df2 = new_df2.dropna()
new_df2.isnull().sum()
new_df2.head()
```

Out[26]:

	Customer ID	Invoice	StockCode	Quantity	Price	TotalAmt	Description	InvoiceDate	Country
0	13085.0	489434	85048	12	6.95	83.4	15CM CHRISTMAS GLASS BALL 20 LIGHTS	2009-12-01 07:45:00	United Kingdom
1	13085.0	489434	79323P	12	6.75	81.0	PINK CHERRY LIGHTS	2009-12-01 07:45:00	United Kingdom
2	13085.0	489434	79323W	12	6.75	81.0	WHITE CHERRY LIGHTS	2009-12-01 07:45:00	United Kingdom
3	13085.0	489434	22041	48	2.10	100.8	RECORD FRAME 7" SINGLE SIZE	2009-12-01 07:45:00	United Kingdom
4	13085.0	489434	21232	24	1.25	30.0	STRAWBERRY CERAMIC TRINKET BOX	2009-12-01 07:45:00	United Kingdom

In [27]:

```
# Sales Average of each product

avg_sales = new_df2.groupby(['StockCode','Description'])['Quantity','TotalAmt'].mean().so
avg_sales
```

Out[27]:

		Quantity	TotalAmt
StockCode	**Description**		
16044	**POP-ART FLUORESCENT PENS**	3096.000000	194.400000
85220	**SMALL FAIRY CAKE FRIDGE MAGNETS**	1389.000000	461.335714
37410	**BLACK AND WHITE PAISLEY FLOWER MUG**	1351.526316	158.530000
37351	**ORANGE FLOWER MUG**	898.500000	95.025000
85218	**S/5 MINI ICE CREAM FRIDGE MAGNETS**	663.000000	206.950000
...	**...**	...	...
21254	**SET OF KITCHEN WALL STICKERS**	-4.000000	-11.800000
35976B	**WHITE SCANDINAVIAN HEART CHRISTMAS**	-5.500000	-6.875000
22003	**VINTAGE BLUE VACUUM FLASK 0.5L**	-10.000000	-67.500000
21701	**SET 6 MINI SUSHI SET FRIDGE MAGNETS**	-12.000000	-20.280000
D	**Discount**	-17.268041	-79.319897

4487 rows × 2 columns

Answers - 1.We can see the demand for each product

Step 2: Data Collection

Data Collection includes the loading the data into the data frame and extracting some information about the data.

Performing Cohort Analysis for extracting the some more insights from the data set

In [28]:

```
#loading the same  dataset for the Cohort Analysis.
```

```
Customers_data= pd.read_excel(r'C:\Users\VARUN D S\Downloads\RetailPulseAssignmentData.xl
```

In [29]:
```
# to see the first 5 rows of the dataset.
Customers_data.head()
```

Out[29]:

	Invoice	StockCode	Description	Quantity	InvoiceDate	Price	Customer ID	Country
0	489434	85048	15CM CHRISTMAS GLASS BALL 20 LIGHTS	12	2009-12-01 07:45:00	6.95	13085.0	United Kingdom
1	489434	79323P	PINK CHERRY LIGHTS	12	2009-12-01 07:45:00	6.75	13085.0	United Kingdom
2	489434	79323W	WHITE CHERRY LIGHTS	12	2009-12-01 07:45:00	6.75	13085.0	United Kingdom
3	489434	22041	RECORD FRAME 7" SINGLE SIZE	48	2009-12-01 07:45:00	2.10	13085.0	United Kingdom
4	489434	21232	STRAWBERRY CERAMIC TRINKET BOX	24	2009-12-01 07:45:00	1.25	13085.0	United Kingdom

In [30]:
```
# to see the total number of rows and columns present in the data set resp.

Customers_data.shape
```

Out[30]:
```
(525461, 8)
```

In [31]:
```
# to see the columns names.

Customers_data.columns
```

Out[31]:
```
Index(['Invoice', 'StockCode', 'Description', 'Quantity', 'InvoiceDate',
       'Price', 'Customer ID', 'Country'],
      dtype='object')
```

Step 3: Data Cleaning

Question 3. Based on your analysis of the data, what are potential areas of improvement for the business?

Data Cleaning includes, identifying and removal of outliers, null values and negetive values

In [32]:
```
#Customer ID distribution by country
country_cust_data=Customers_data[['Country','Customer ID']].drop_duplicates()
country_cust_data.groupby(['Country'])['Customer ID'].aggregate('count').reset_index().so
```

Out[32]:

	Country	Customer ID
37	United Kingdom	4035
13	Germany	68
12	France	47
31	Spain	25
24	Netherlands	23

	Country	Customer ID
28	Portugal	18
3	Belgium	17
32	Sweden	16
0	Australia	15
33	Switzerland	14
7	Channel Islands	12
18	Italy	11
1	Austria	10
9	Denmark	9
11	Finland	8
8	Cyprus	7
35	USA	6
19	Japan	6
10	EIRE	5
38	Unspecified	5
26	Norway	5
14	Greece	4
36	United Arab Emirates	4
27	Poland	2
20	Korea	2
17	Israel	2
2	Bahrain	2
25	Nigeria	1
23	Malta	1
22	Lithuania	1
29	RSA	1
30	Singapore	1
16	Iceland	1
34	Thailand	1
6	Canada	1
5	Brazil	1
39	West Indies	1
21	Lebanon	0
15	Hong Kong	0
4	Bermuda	0

Insight

In 100% of the customer ID present in the country , the countries in the below has maximum and minimum
ITED KINGDOM has the majority of 91.95% . 2.Lebanon, Hong Kong and Bermuda

has the 0% Customer Id. 3.Countries except United Kingdom, there are very negligible number of customers Id so the company can improve their business by providing some, (a)discounts (b)refer and earn option (c)Advertisements about the company in social medias and in news papers. 4.And also they can attract the customers and make their company name by, (a)donating some amount of shares by creating the charity of their own like TATA GROUPS.

In [43]:
```
CCustomers_data=CCustomers_data[['Customer ID','Invoice','StockCode','Quantity','Price','I
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-43-b3a621944982> in <module>
----> 1 CCustomers_data=CCustomers_data[['Customer ID','Invoice','StockCode','Quantity','P
rice','Description','InvoiceDate','Country']]

NameError: name 'CCustomers_data' is not defined
```

In [ ]:
```
#to see the re arranged columns according to our convinience
CCustomers_data.head()
```

In [ ]:
```
##Check for missing values in the dataset
CCustomers_data.isnull().sum(axis=0)
```

In [ ]:
```
#Remove missing values from Customer ID column, can ignore missing values in description
CCustomers_data = CCustomers_data[pd.notnull(Customers_data['Customer ID'])]
```

In [ ]:
```
#Validate if there are any negative values in Quantity column
CCustomers_data.Quantity.min()
```

In [ ]:
```
#Validate if there are any negative values in Price column
CCustomers_data.Price.min()
```

In [ ]:
```
#Filter out records with negative values
CCustomers_data = Customers_data[(Customers_data['Quantity']>0)]
```

In [ ]:
```
#Convert the string date field to datetime
CCustomers_data['InvoiceDate'] = pd.to_datetime(Customers_data['InvoiceDate'])
```

In [ ]:
```
#Add new column depicting total amount
CCustomers_data['TotalAmount'] = Customers_data['Quantity'] * Customers_data['Price']
```

In [ ]:
```
#Check the shape (number of columns and rows) in the dataset after data is cleaned
CCustomers_data.shape
```

In [ ]:
```
CCustomers_data.head()
```

Question 1: Can customers be segmented into different categories?

Answer: Yes

yes then perform analysis on the same and also

propose categories

Step 4: Analyzing the Data

In this step,involves gathering all the information, processing it, exploring the data, and using it to find patterns and other insights.

In [ ]:
```
#Uses the datetime function to gets the month a datetime stamp and strips the time
def get_month(x):
    return dt.datetime(x.year, x.month, 1) #year, month, incremints of day
#Create a new column
CCustomers_data['InvoiceMonth'] = CCustomers_data['InvoiceDate'].apply(get_month)
#Always inspect the data you've just created
CCustomers_data['InvoiceMonth']
#Create a CohortMonth column by grouping data and selecting the earliest instance in the
CCustomers_data['CohortMonth'] = CCustomers_data.groupby('Customer ID')['InvoiceMonth'].t
CCustomers_data['CohortMonth']
CCustomers_data.head()
```

In [ ]:
```
def get_date_int(df, column):
    year = df[column].dt.year
    month = df[column].dt.month
    day = df[column].dt.day
    return year, month, day
invoice_year, invoice_month, _ = get_date_int(CCustomers_data, 'InvoiceMonth')
cohort_year, cohort_month, _ = get_date_int(CCustomers_data, 'CohortMonth')
years_diff = invoice_year - cohort_year
months_diff = invoice_month - cohort_month
CCustomers_data['CohortIndex'] = years_diff * 12 + months_diff
CCustomers_data.head()
```

In [ ]:
```
#Group the data by columns CohortMonth','CohortIndex' then aggreate by column 'Customer I
CCustomers_data = CCustomers_data.groupby(
    ['CohortMonth', 'CohortIndex'])['Customer ID'].apply(pd.Series.nunique).reset_index()
```

In [ ]:
```
#Take the cohort_data and plumb it into a Pivot Table. Setting index, columns and values
cohort_count = CCustomers_data.pivot_table(index = 'CohortMonth',
                                           columns = 'CohortIndex',
                                           values = 'Customer ID')
```

In [ ]:
```
cohort_count
```

In [ ]:
```
cohort_sizes = cohort_count.iloc[:,0]
# Divide all values in the cohort_counts table by cohort_sizes
retention = cohort_count.divide(cohort_sizes, axis=0)
# Check the retention table
retention.round(3) * 100
# Drawing a heatmap
plt.figure(figsize=(10, 8))
plt.title('Retention rates')
sns.heatmap(data = retention,annot = True,fmt = '.0%',vmin = 0.0,vmax = 0.5,cmap = 'BuGn'
plt.show()
```

In [ ]:
```
CCustomers_data.columns
```

In [ ]:
```
Customers_data["TotalSum"] = Customers_data["Price"] * Customers_data["Quantity"]
```

In [ ]:
```
Customers_data.head()
```

In [ ]:
```
from datetime import datetime
from datetime import timedelta
begin = "2021-12-09"
begin= datetime.strptime(begin, "%Y-%m-%d")
print(begin.date())
end = begin + timedelta(days=1)
print(end.date())
```

In [ ]:
```
#Recency = Latest Date - Last Inovice Data, Frequency = count of invoice no. of transacti
#Amount for each customer
import datetime as dt

#Set Latest date 2011-12-10 as last invoice date was 2011-12-09. This is to calculate the
Latest_Date = dt.datetime(2010,12,10)

#Create RFM Modelling scores for each customer
RFMScores = Customers_data.groupby('Customer ID').agg({'InvoiceDate': lambda x: (Latest_D

#Convert Invoice Date into type int
RFMScores['InvoiceDate'] = RFMScores['InvoiceDate'].astype(int)

#Rename column names to Recency, Frequency and Monetary
RFMScores.rename(columns={'InvoiceDate': 'Recency',
                          'Invoice': 'Frequency',
                          'TotalSum': 'Monetary','Country' :'Country'},inplace=True)

RFMScores.reset_index().head()
```

In [ ]:
```
#Descriptive Statistics (Recency)
RFMScores.Recency.describe()
```

In [ ]:
```
#Recency distribution plot
import seaborn as sns
x = RFMScores['Recency']

ax = sns.distplot(x)
```

In [ ]:
```
#Descriptive Statistics (Frequency)
RFMScores.Frequency.describe()
```

In [ ]:
```
#Frequency distribution plot, taking observations which have frequency less than 1000
import seaborn as sns
x = RFMScores.query('Frequency < 1000')['Frequency']

ax = sns.distplot(x)
```

In [ ]:
```
#Descriptive Statistics (Monetary)
RFMScores.Monetary.describe()
```

```
#Monateray distribution plot, taking observations which have monetary value less than 100
import seaborn as sns
x = RFMScores.query('Monetary < 10000')['Monetary']

ax = sns.distplot(x)
```

In [ ]:
```
#Split into four segments using quantiles
quantiles = RFMScores.quantile(q=[0.25,0.5,0.75])
quantiles = quantiles.to_dict()
```

In [ ]:
```
quantiles
```

In [ ]:
```
#Functions to create R, F and M segments
def RScoring(x,p,d):
    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

def FnMScoring(x,p,d):
    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
```

In [ ]:
```
#Calculate Add R, F and M segment value columns in the existing dataset to show R, F and
RFMScores['R'] = RFMScores['Recency'].apply(RScoring, args=('Recency',quantiles,))
RFMScores['F'] = RFMScores['Frequency'].apply(FnMScoring, args=('Frequency',quantiles,))
RFMScores['M'] = RFMScores['Monetary'].apply(FnMScoring, args=('Monetary',quantiles,))
RFMScores.head()
```

In [ ]:
```
#Calculate and Add RFMGroup value column showing combined concatenated score of RFM
RFMScores['RFMGroup'] = RFMScores.R.map(str) + RFMScores.F.map(str) + RFMScores.M.map(str

#Calculate and Add RFMScore value column showing total sum of RFMGroup values
RFMScores['RFMScore'] = RFMScores[['R', 'F', 'M']].sum(axis = 1)
RFMScores.head()
```

In [ ]:
```
#Assign Loyalty Level to each customer
Loyalty_Level = ['Platinum', 'Gold', 'Silver', 'Bronze']
Score_cuts = pd.qcut(RFMScores.RFMScore, q = 4, labels = Loyalty_Level)
RFMScores['RFM_Loyalty_Level'] = Score_cuts.values
RFMScores.reset_index().head()
```

In [ ]:
```
#Validate the data for RFMGroup = 111
RFMScores[RFMScores['RFMGroup']=='111'].sort_values('Monetary', ascending=False).reset_in
```

In [ ]:

```
!pip install chart_studio
import chart_studio as cs
import plotly.offline as po
import plotly.graph_objs as gobj

#Recency Vs Frequency
graph = RFMScores.query("Monetary < 50000 and Frequency < 2000")

plot_data = [
    gobj.Scatter(
        x=graph.query("RFM_Loyalty_Level == 'Bronze'")['Recency'],
        y=graph.query("RFM_Loyalty_Level == 'Bronze'")['Frequency'],
        mode='markers',
        name='Bronze',
        marker= dict(size= 7,
            line= dict(width=1),
            color= 'blue',
            opacity= 0.8
           )
    ),
        gobj.Scatter(
        x=graph.query("RFM_Loyalty_Level == 'Silver'")['Recency'],
        y=graph.query("RFM_Loyalty_Level == 'Silver'")['Frequency'],
        mode='markers',
        name='Silver',
        marker= dict(size= 9,
            line= dict(width=1),
            color= 'green',
            opacity= 0.5
           )
    ),
        gobj.Scatter(
        x=graph.query("RFM_Loyalty_Level == 'Gold'")['Recency'],
        y=graph.query("RFM_Loyalty_Level == 'Gold'")['Frequency'],
        mode='markers',
        name='Gold',
        marker= dict(size= 11,
            line= dict(width=1),
            color= 'red',
            opacity= 0.9
           )
    ),
    gobj.Scatter(
        x=graph.query("RFM_Loyalty_Level == 'Platinum'")['Recency'],
        y=graph.query("RFM_Loyalty_Level == 'Platinum'")['Frequency'],
        mode='markers',
        name='Platinum',
        marker= dict(size= 13,
            line= dict(width=1),
            color= 'black',
            opacity= 0.9
           )
    ),
]

plot_layout = gobj.Layout(
        yaxis= {'title': "Frequency"},
        xaxis= {'title': "Recency"},
        title='Segments'
    )
fig = gobj.Figure(data=plot_data, layout=plot_layout)
po.iplot(fig)

#Frequency Vs Monetary
graph = RFMScores.query("Monetary < 50000 and Frequency < 2000")
```

```
plot_data = [
    gobj.Scatter(
        x=graph.query("RFM_Loyalty_Level == 'Bronze'")['Frequency'],
        y=graph.query("RFM_Loyalty_Level == 'Bronze'")['Monetary'],
        mode='markers',
        name='Bronze',
        marker= dict(size= 7,
            line= dict(width=1),
            color= 'blue',
            opacity= 0.8
           )
    ),
        gobj.Scatter(
        x=graph.query("RFM_Loyalty_Level == 'Silver'")['Frequency'],
        y=graph.query("RFM_Loyalty_Level == 'Silver'")['Monetary'],
        mode='markers',
        name='Silver',
        marker= dict(size= 9,
            line= dict(width=1),
            color= 'green',
            opacity= 0.5
           )
    ),
        gobj.Scatter(
        x=graph.query("RFM_Loyalty_Level == 'Gold'")['Frequency'],
        y=graph.query("RFM_Loyalty_Level == 'Gold'")['Monetary'],
        mode='markers',
        name='Gold',
        marker= dict(size= 11,
            line= dict(width=1),
            color= 'red',
            opacity= 0.9
           )
    ),
    gobj.Scatter(
        x=graph.query("RFM_Loyalty_Level == 'Platinum'")['Frequency'],
        y=graph.query("RFM_Loyalty_Level == 'Platinum'")['Monetary'],
        mode='markers',
        name='Platinum',
        marker= dict(size= 13,
            line= dict(width=1),
            color= 'black',
            opacity= 0.9
           )
    ),
]

plot_layout = gobj.Layout(
        yaxis= {'title': "Monetary"},
        xaxis= {'title': "Frequency"},
        title='Segments'
    )
fig = gobj.Figure(data=plot_data, layout=plot_layout)
po.iplot(fig)

#Recency Vs Monetary
graph = RFMScores.query("Monetary < 50000 and Frequency < 2000")

plot_data = [
    gobj.Scatter(
        x=graph.query("RFM_Loyalty_Level == 'Bronze'")['Recency'],
        y=graph.query("RFM_Loyalty_Level == 'Bronze'")['Monetary'],
        mode='markers',
        name='Bronze',
        marker= dict(size= 7,
            line= dict(width=1),
```

```
            color= 'blue',
            opacity= 0.8
           )
    ),
        gobj.Scatter(
        x=graph.query("RFM_Loyalty_Level == 'Silver'")['Recency'],
        y=graph.query("RFM_Loyalty_Level == 'Silver'")['Monetary'],
        mode='markers',
        name='Silver',
        marker= dict(size= 9,
            line= dict(width=1),
            color= 'green',
            opacity= 0.5
           )
    ),
        gobj.Scatter(
        x=graph.query("RFM_Loyalty_Level == 'Gold'")['Recency'],
        y=graph.query("RFM_Loyalty_Level == 'Gold'")['Monetary'],
        mode='markers',
        name='Gold',
        marker= dict(size= 11,
            line= dict(width=1),
            color= 'red',
            opacity= 0.9
           )
    ),
    gobj.Scatter(
        x=graph.query("RFM_Loyalty_Level == 'Platinum'")['Recency'],
        y=graph.query("RFM_Loyalty_Level == 'Platinum'")['Monetary'],
        mode='markers',
        name='Platinum',
        marker= dict(size= 13,
            line= dict(width=1),
            color= 'black',
            opacity= 0.9
           )
    ),
]

plot_layout = gobj.Layout(
        yaxis= {'title': "Monetary"},
        xaxis= {'title': "Recency"},
        title='Segments'
    )
fig = gobj.Figure(data=plot_data, layout=plot_layout)
po.iplot(fig)
```

In [ ]:

```
#Handle negative and zero values so as to handle infinite numbers during log transformati
def handle_neg_n_zero(num):
    if num <= 0:
        return 1
    else:
        return num
#Apply handle_neg_n_zero function to Recency and Monetary columns
RFMScores['Recency'] = [handle_neg_n_zero(x) for x in RFMScores.Recency]
RFMScores['Monetary'] = [handle_neg_n_zero(x) for x in RFMScores.Monetary]

#Perform Log transformation to bring data into normal or near normal distribution
Log_Tfd_Data = RFMScores[['Recency', 'Frequency', 'Monetary']].apply(np.log, axis = 1).ro
```

In [ ]:

```
#Data distribution after data normalization for Recency
Recency_Plot = Log_Tfd_Data['Recency']
t(Recency_Plot)
```

In [ ]:
```
#Data distribution after data normalization for Frequency
Frequency_Plot = Log_Tfd_Data.query('Frequency < 1000')['Frequency']
ax = sns.distplot(Frequency_Plot)
```

In [ ]:
```
#Data distribution after data normalization for Monetary
Monetary_Plot = Log_Tfd_Data.query('Monetary < 10000')['Monetary']
ax = sns.distplot(Monetary_Plot)
```

In [ ]:
```
from sklearn.preprocessing import StandardScaler

#Bring the data on same scale
scaleobj = StandardScaler()
Scaled_Data = scaleobj.fit_transform(Log_Tfd_Data)

#Transform it back to dataframe
Scaled_Data = pd.DataFrame(Scaled_Data, index = RFMScores.index, columns = Log_Tfd_Data.c
```

In [ ]:
```
from sklearn.cluster import KMeans

sum_of_sq_dist = {}
for k in range(1,15):
    km = KMeans(n_clusters= k, init= 'k-means++', max_iter= 1000)
    km = km.fit(Scaled_Data)
    sum_of_sq_dist[k] = km.inertia_

#Plot the graph for the sum of square distance values and Number of Clusters
sns.pointplot(x = list(sum_of_sq_dist.keys()), y = list(sum_of_sq_dist.values()))
plt.xlabel('Number of Clusters(k)')
plt.ylabel('Sum of Square Distances')
plt.title('Elbow Method For Optimal k')
plt.show()
```

About RFM analysis.

Recency, frequency, monetary value (RFM) is a marketing analysis tool used to identify a firm's best clients based on the nature of their spending habits. An RFM analysis evaluates clients and customers by scoring them in three categories: how recently they've made a purchase, how often they buy, and the size of their purchases. RFM analysis helps firms reasonably predict which customers are likely to purchase their products again, how much revenue comes from new (versus repeat) clients, and how to turn occasional buyers into habitual ones.

In [ ]:
```
#Perform K-Mean Clustering or build the K-Means clustering model
KMean_clust = KMeans(n_clusters= 3, init= 'k-means++', max_iter= 1000)
KMean_clust.fit(Scaled_Data)

#Find the clusters for the observation given in the dataset
RFMScores['Cluster'] = KMean_clust.labels_
RFMScores.head()
```

Recency: How recently a customer has made a purchase Frequency: How often a customer makes a purchase Monetary Value: How much money a customer spends on purchases RFM analysis numerically ranks a customer in each of these three categories, generally on a scale of 1 to 5 (the higher the number, the better the result). The "best" customer would receive a top score in every category.

Insight

From the above table we can observe the insights that, 1.customer ID - 12349 makes a purchase very repeatedly or frequently, and the main observation is he is from ITALY. I would like to highlight this point because, In order to get this insight I didn't drop the other countries name. this customer is one of the main customer for the retail company. Even though we have more number of customers from the United Kingdom but we have very frequently buying customers from the country Italy(Customer ID:12349) and from Iceland(Customer ID:12347) and from UK(Customer ID: 12346) and from Finland(Customer ID:12348).

In [ ]:

```
from matplotlib import pyplot as plt
plt.figure(figsize=(7,7))

##Scatter Plot Frequency Vs Recency
Colors = ["red", "green", "blue"]
RFMScores['Color'] = RFMScores['Cluster'].map(lambda p: Colors[p])
ax = RFMScores.plot(
    kind="scatter",
    x="Recency", y="Frequency",
    figsize=(10,8),
    c = RFMScores['Color']
)
```

Question 3: How would you define a loyal customer?

Answer: Loyal customer means,how willing a customer is to engage with and repeatedly purchase from you versus your competitors. Loyalty is the byproduct of a customers positive experience with you and works to create trust.

In the above table, the column, RFM_Loyalty_Level represents the loyalty level of the customer.

loyal customers are the customers who is having Customer ID, 12349,12347,12346 and 12348. And more importantly top 2 loyal customers are from other countries than UNITED KINGDOM.

In [37]:

```
Customers_data.head()
```

Out[37]:

InvoiceDate	Invoice	StockCode	Description	Quantity	Price	Customer ID	Country
2009-12-01 07:45:00	489434	85048	15CM CHRISTMAS GLASS BALL 20 LIGHTS	12	6.95	13085.0	United Kingdom
2009-12-01 07:45:00	489434	79323P	PINK CHERRY LIGHTS	12	6.75	13085.0	United Kingdom
2009-12-01 07:45:00	489434	79323W	WHITE CHERRY LIGHTS	12	6.75	13085.0	United Kingdom
2009-12-01 07:45:00	489434	22041	RECORD FRAME 7" SINGLE SIZE	48	2.10	13085.0	United Kingdom
2009-12-01 07:45:00	489434	21232	STRAWBERRY CERAMIC TRINKET BOX	24	1.25	13085.0	United Kingdom

In [41]:
```
#let us drop the duplicate values in the customers ID to get original no of customers
Customers_data.drop_duplicates(subset ="Customer ID",
                     keep = False, inplace = True)
display(Customers_data)
```

InvoiceDate	Invoice	StockCode	Description	Quantity	Price	Customer ID	Country
2009-12-01 09:55:00	489444	POST	POSTAGE	1	141.00	12636.0	USA
2009-12-01 10:10:00	489447	POST	POSTAGE	1	130.00	12362.0	Belgium
2009-12-01 12:35:00	C489551	85048	15CM CHRISTMAS GLASS BALL 20 LIGHTS	-1	6.95	17641.0	United Kingdom
2009-12-02 14:09:00	489834	21217	RED SPOTTY ROUND CAKE TINS	24	8.95	14106.0	United Kingdom
2009-12-04 12:31:00	490258	22130	PARTY CONE CHRISTMAS DECORATION	24	0.85	15999.0	United Kingdom
...	...	...	...	...	...	...	...
2010-12-02 10:10:00	536616	21137	BLACK RECORD COVER FRAME	72	3.39	17925.0	United Kingdom
2010-12-02 17:09:00	C536818	84947	ANTIQUE SILVER TEA GLASS ENGRAVED	-1	1.25	16995.0	United Kingdom
2010-12-02 17:51:00	536834	21111	SWISS ROLL TOWEL, CHOCOLATE SPOTS	12	2.95	14576.0	United Kingdom
2010-12-06 12:31:00	537360	84946	ANTIQUE SILVER TEA GLASS ETCHED	72	1.06	18113.0	United Kingdom
2010-12-08 14:53:00	537833	51008	AFGHAN SLIPPER SOCK PAIR	200	2.95	13270.0	United Kingdom

134 rows × 7 columns

In [42]:
```
Customers_data.shape
```

Out[42]: (134, 7)

Insight

The Important observation is there is only 134 customers according to the customer ID.

In [ ]:
```

```