In [1]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import datetime as dt
```

In [2]:

```
pd.set_option("display.max_columns",None);
pd.set_option("display.max_rows",None);
```

In [3]:

```
retail = pd.read_excel("online_retail_II.xlsx", sheet_name="Year 2010-2011")
```

In [4]:

```
df = retail.copy()
```

In [5]:

```
df.head(3)
```

Out[5]:

	Invoice	StockCode	Description	Quantity	InvoiceDate	Price	Customer ID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom

In [6]:

```
df.tail(3)
```

Out[6]:

	Invoice	StockCode	Description	Quantity	InvoiceDate	Price	Customer ID	Country
541907	581587	23255	CHILDRENS CUTLERY CIRCUS PARADE	4	2011-12-09 12:50:00	4.15	12680.0	France
541908	581587	22138	BAKING SET 9 PIECE RETROSPOT	3	2011-12-09 12:50:00	4.95	12680.0	France
541909	581587	POST	POSTAGE	1	2011-12-09 12:50:00	18.00	12680.0	France

In [7]:

```
df["Country"].value_counts()
```

Out[7]:

```
United Kingdom          495478
Germany                   9495
France                    8558
EIRE                      8196
Spain                     2533
Netherlands               2371
Belgium                   2069
Switzerland               2002
Portugal                  1519
Australia                 1259
Norway                    1086
Italy                      803
Channel Islands            758
Finland                    695
Cyprus                     622
Sweden                     462
Unspecified                446
Austria                    401
Denmark                    389
Japan                      358
Poland                     341
Israel                     297
USA                        291
Hong Kong                  288
Singapore                  229
Iceland                    182
Canada                     151
Greece                     146
Malta                      127
United Arab Emirates        68
European Community          61
RSA                         58
Lebanon                     45
Lithuania                   35
Brazil                      32
Czech Republic              30
Bahrain                     19
Saudi Arabia                10
Name: Country, dtype: int64
```

In [8]:

```
#unique country count
df["Country"].nunique()
```

Out[8]:

```
38
```

In [9]:

```
for i in df.columns:
  print(i, df[i].nunique())
```

```
Invoice 25900
StockCode 4070
Description 4223
Quantity 722
InvoiceDate 23260
Price 1630
Customer ID 4372
Country 38
```

In [10]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 541910 entries, 0 to 541909
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   Invoice      541910 non-null  object
 1   StockCode    541910 non-null  object
 2   Description  540456 non-null  object
 3   Quantity     541910 non-null  int64
 4   InvoiceDate  541910 non-null  datetime64[ns]
 5   Price        541910 non-null  float64
 6   Customer ID  406830 non-null  float64
 7   Country      541910 non-null  object
dtypes: datetime64[ns](1), float64(2), int64(1), object(4)
memory usage: 33.1+ MB
```

In [11]:

```
#check null values
df.isnull().sum()
```

Out[11]:

```
Invoice             0
StockCode           0
Description      1454
Quantity            0
InvoiceDate         0
Price               0
Customer ID    135080
Country             0
dtype: int64
```

In [12]:

```
df.shape
```

Out[12]:

```
(541910, 8)
```

In [13]:

```
#ration of total null values to total rows
df.isnull().sum().sum()/df.shape[0]
```

Out[13]:

```
0.2519495857245668
```

In [14]:

```
df.size
```

Out[14]:

```
4335280
```

In [15]:

```
# add new column . Total price column , to see how much customers  paid
df["total-price"] = df["Quantity"]*df["Price"]
```

In [16]:

```
#look new column
df.head()
```

Out[16]:

	Invoice	StockCode	Description	Quantity	InvoiceDate	Price	Customer ID	Country	total-price
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom	15.30
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	20.34
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom	22.00
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	20.34
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	20.34

In [17]:

```
#total payment per customer
df.groupby("Customer ID").agg({"total-price":"sum"}).head()
```

Out[17]:

	total-price
Customer ID	
12346.0	0.00
12347.0	4310.00
12348.0	1797.24
12349.0	1757.55
12350.0	334.40

In [18]:

```
#we see 0 total price value here.
df[df["Customer ID"]==12346.0]
```

Out[18]:

	Invoice	**StockCode**	**Description**	**Quantity**	**InvoiceDate**	**Price**	**Customer ID**	**Country**	
61619	541431	23166	MEDIUM CERAMIC TOP STORAGE JAR	74215	2011-01-18 10:01:00	1.04	12346.0	United Kingdom	77
61624	C541433	23166	MEDIUM CERAMIC TOP STORAGE JAR	-74215	2011-01-18 10:17:00	1.04	12346.0	United Kingdom	-77

In [19]:

```
#In this dataset Invoice IDs which starts with C letter shows that it is a return invoice.
#For this analysis I will delete return invoices. Because it doesnt change that he/she pref
df[df["Invoice"].astype("str").str.get(0)=="C"].head()
```

Out[19]:

	Invoice	StockCode	Description	Quantity	InvoiceDate	Price	Customer ID	Country	tota pric
141	C536379	D	Discount	-1	2010-12-01 09:41:00	27.50	14527.0	United Kingdom	-27.5
154	C536383	35004C	SET OF 3 COLOURED FLYING DUCKS	-1	2010-12-01 09:49:00	4.65	15311.0	United Kingdom	-4.6
235	C536391	22556	PLASTERS IN TIN CIRCUS PARADE	-12	2010-12-01 10:24:00	1.65	17548.0	United Kingdom	-19.8
236	C536391	21984	PACK OF 12 PINK PAISLEY TISSUES	-24	2010-12-01 10:24:00	0.29	17548.0	United Kingdom	-6.9
237	C536391	21983	PACK OF 12 BLUE PAISLEY TISSUES	-24	2010-12-01 10:24:00	0.29	17548.0	United Kingdom	-6.9

In [20]:

```
# Now we choose normal invoices
df[df["Invoice"].astype("str").str.get(0)!="C"].head()
```

Out[20]:

	Invoice	StockCode	Description	Quantity	InvoiceDate	Price	Customer ID	Country	total-price
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom	15.30
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	20.34
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom	22.00
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	20.34
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	20.34

In [21]:

```
#drop missing values
df = df.dropna()
```

In [22]:

```
# Change customer ID values to int.
df["Customer ID"]=df["Customer ID"].astype("int")
```

In [23]:

```
#RECENCY
```

In [24]:

```
df["InvoiceDate"].max()
```

Out[24]:

```
Timestamp('2011-12-09 12:50:00')
```

In [25]:

```
# I choose last Invoicedate as my analysis date
today_date=dt.datetime(2011,12,9)
```

In [26]:

```
 # now use groupby to customerID( bcs there may be many invoice per user) and choose Invoic
 # the last time it came is important to recency
df.groupby("Customer ID").agg({"InvoiceDate": "max"}).head()
```

Out[26]:

	InvoiceDate
Customer ID	
12346	2011-01-18 10:17:00
12347	2011-12-07 15:52:00
12348	2011-09-25 13:13:00
12349	2011-11-21 09:51:00
12350	2011-02-02 16:01:00

In [27]:

```
# Now substract max date values and our analysis date
rec = today_date-df.groupby("Customer ID").agg({"InvoiceDate": "max"})
```

In [28]:

```
rec.head()
```

Out[28]:

	InvoiceDate
Customer ID	
12346	324 days 13:43:00
12347	1 days 08:08:00
12348	74 days 10:47:00
12349	17 days 14:09:00
12350	309 days 07:59:00

In [29]:

```
# clear data time
# List Compheresion
rec["InvoiceDate"]=[i.days for i in rec["InvoiceDate"]]
```

In [30]:

```
#Now we see date time as day
rec.head()
```

Out[30]:

	InvoiceDate
Customer ID	
12346	324
12347	1
12348	74
12349	17
12350	309

In [31]:

```
rec.rename(columns={"InvoiceDate": "Recency"},inplace=True)
```

In [32]:

```
rec.head()
```

Out[32]:

	Recency
Customer ID	
12346	324
12347	1
12348	74
12349	17
12350	309

In [33]:

```
#FREQUENCY
```

In [34]:

```
# how many times have they bought something
df.groupby("Customer ID").agg({"Invoice":"nunique"}).head(10)
```

Out[34]:

	Invoice
Customer ID	
12346	2
12347	7
12348	4
12349	1
12350	1
12352	11
12353	1
12354	1
12355	1
12356	3

In [35]:

```
freq = df.groupby("Customer ID").agg({"Invoice":"nunique"})
```

In [36]:

```
freq = freq.rename(columns={"Invoice":"Frequency"})
```

In [37]:

```
freq.head()
```

Out[37]:

	Frequency
Customer ID	
12346	2
12347	7
12348	4
12349	1
12350	1

In [38]:

```
#MONETARY
```

In [39]:

```
money = df.groupby("Customer ID").agg({"total-price":"sum"}).rename(columns={"total-price":
```

In [40]:

```
money.head()
```

Out[40]:

	Monetary
Customer ID	
12346	0.00
12347	4310.00
12348	1797.24
12349	1757.55
12350	334.40

In [41]:

```
#RFM
```

In [42]:

```
# check shapes
print(rec.shape, freq.shape, money.shape)
```

```
(4372, 1) (4372, 1) (4372, 1)
```

In [43]:

```
# concat rec , freq and money
rfm  = pd.concat([rec,freq,money], axis=1)
```

In [44]:

```
rfm.head()
```

Out[44]:

	Recency	Frequency	Monetary
Customer ID			
12346	324	2	0.00
12347	1	7	4310.00
12348	74	4	1797.24
12349	17	1	1757.55
12350	309	1	334.40

In [45]:

```
# determine range
rfm["Recency-Score"] = pd.qcut(rfm["Recency"],5,labels=[5,4,3,2,1] )
rfm["Frequency-Score"] = pd.qcut(rfm["Frequency"].rank(method="first"),5,labels=[1,2,3,4,5]
rfm["Monetary-Score"] = pd.qcut(rfm["Monetary"],5,labels=[1,2,3,4,5] )
```

In [46]:

```
rfm.head()
```

Out[46]:

	Recency	Frequency	Monetary	Recency-Score	Frequency-Score	Monetary-Score
Customer ID						
12346	324	2	0.00	1	2	1
12347	1	7	4310.00	5	4	5
12348	74	4	1797.24	2	3	4
12349	17	1	1757.55	4	1	4
12350	309	1	334.40	1	1	2

In [47]:

```
rfm["RFM"]= rfm["Recency-Score"].astype("str")+rfm["Monetary-Score"].astype("str")+rfm["Rec
```

In [48]:

```
rfm.head()
```

Out[48]:

Customer ID	Recency	Frequency	Monetary	Recency-Score	Frequency-Score	Monetary-Score	RFM
12346	324	2	0.00	1	2	1	111
12347	1	7	4310.00	5	4	5	555
12348	74	4	1797.24	2	3	4	242
12349	17	1	1757.55	4	1	4	444
12350	309	1	334.40	1	1	2	121

In [49]:

```
# u can see these groups in rfm table
# this func is about recency and frequency

segmnt_map = {r'[1-2][1-2]' : "Hibernating"      ,
              r'[1-2][3-4]' :  "At Risk"         ,
              r'[1-2]5'     :  "Can't Lose"      ,
              r'3[1-2]'     :"About to Sleep"    ,
              r'33'         :"Need Attention"    ,
              r'[3-4][4-5]' :"Loyal Customers"   ,
              r'41'         :"Promising"         ,
              r'51'         :"New Customers"     ,
              r'[4-5][2-3]' :"Potential Loyalist",
              r'5[4-5]'     :"Champions"
}
```

In [50]:

```
rfm["Segment"] = rfm["Recency-Score"].astype("str")+rfm["Frequency-Score"].astype("str")
```

In [51]:

```
rfm["Segment"] = rfm["Segment"].replace(segmnt_map,regex=True)
```

In [52]:

```python
rfm.head()
```

Out[52]:

	Recency	Frequency	Monetary	Recency-Score	Frequency-Score	Monetary-Score	RFM	Segment
Customer ID								
12346	324	2	0.00	1	2	1	111	Hibernating
12347	1	7	4310.00	5	4	5	555	Champions
12348	74	4	1797.24	2	3	4	242	At Risk
12349	17	1	1757.55	4	1	4	444	Promising
12350	309	1	334.40	1	1	2	121	Hibernating

In [53]:

```python
rfm[["Segment","Recency","Frequency","Monetary"]].head()
```

Out[53]:

	Segment	Recency	Frequency	Monetary
Customer ID				
12346	Hibernating	324	2	0.00
12347	Champions	1	7	4310.00
12348	At Risk	74	4	1797.24
12349	Promising	17	1	1757.55
12350	Hibernating	309	1	334.40

In [54]:

```
rfm[["Segment","Recency","Frequency","Monetary"]].groupby("Segment").agg(["min","max","mean
```

Out[54]:

	Recency				Frequency				Monetary		
	min	max	mean	count	min	max	mean	count	min	max	n
Segment											
About to Sleep	31	70	51.046070	369	1	2	1.289973	369	-134.80	6207.67	
At Risk	71	371	156.653400	603	2	7	3.313433	603	-840.76	21535.90	
Can't Lose	71	311	132.088235	68	7	35	9.823529	68	230.70	10217.48	2
Champions	-1	10	4.075188	665	4	248	14.590977	665	151.23	279489.02	6
Hibernating	71	372	216.221273	1053	1	2	1.202279	1053	-4287.63	7829.89	
Loyal Customers	11	70	31.289308	795	4	76	7.991195	795	-1165.30	123725.45	2
Need Attention	31	70	47.726316	190	2	4	2.636842	190	-8.15	3545.69	
New Customers	-1	10	5.238095	42	1	1	1.000000	42	41.99	3861.00	
Potential Loyalist	-1	30	14.571429	490	1	4	2.216327	490	-17.45	12393.70	
Promising	11	30	21.103093	97	1	1	1.000000	97	0.00	1757.55	

In [ ]: