Data Analysis By Gauri Gosavi

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
data = pd.read_csv(r"/content/sample_data/online_retail_II.csv")
```

Retriving Information

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 429785 entries, 0 to 429784
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype  
---  ------       --------------   -----  
 0   Invoice      429785 non-null  object 
 1   StockCode    429785 non-null  object 
 2   Description  427198 non-null  object 
 3   Quantity     429784 non-null  float64
 4   InvoiceDate  429784 non-null  object 
 5   Price        429784 non-null  float64
 6   Customer ID  344769 non-null  float64
 7   Country      429784 non-null  object 
dtypes: float64(3), object(5)
memory usage: 26.2+ MB
```

Missing Values

```
dict(data.isnull().sum()/data.shape[0])
```

```
{'Country': 2.3267447677327034e-06,
 'Customer ID': 0.19781053317356354,
 'Description': 0.006019288714124504,
 'Invoice': 0.0,
 'InvoiceDate': 2.3267447677327034e-06,
 'Price': 2.3267447677327034e-06,
 'Quantity': 2.3267447677327034e-06,
 'StockCode': 0.0}
```

```
df_less_than_50 = data.columns[(data.isnull().sum()/data.shape[0]<0.5)]
```

```
data_mod = data[df_less_than_50]
```

```
data mod.head(10)
```

	Invoice	StockCode	Description	Quantity	InvoiceDate	Price	Customer ID	Coun
0	489434	85048	15CM CHRISTMAS GLASS BALL 20 LIGHTS	12.0	01/12/2009 7:45	6.95	13085.0	Un Kingd
1	489434	79323P	PINK CHERRY LIGHTS	12.0	01/12/2009 7:45	6.75	13085.0	Un Kingd
2	489434	79323W	WHITE CHERRY LIGHTS	12.0	01/12/2009 7:45	6.75	13085.0	Un Kingd
3	489434	22041	RECORD FRAME 7" SINGLE SIZE	48.0	01/12/2009 7:45	2.10	13085.0	Un Kingd
4	489434	21232	STRAWBERRY CERAMIC	24.0	01/12/2009 7:45	1.25	13085.0	Un Kingd

```
data_mod.shape
```

```
(429785, 8)
```

Overall sells Trend

```
data_mod.describe()
```

	Quantity	Price	Customer ID
count	429784.000000	429784.000000	344769.000000
mean	10.738687	4.694958	15332.518631
std	113.806533	154.828060	1673.885570
min	-9600.000000	-53594.360000	12346.000000
25%	1.000000	1.250000	13969.000000
50%	3.000000	2.100000	15271.000000
75%	12.000000	4.210000	16782.000000
max	19152.000000	25111.090000	18287.000000

Column Names

```
data_mod.keys()
```

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

Unique Countries

```
data_mod.Country.unique()
```

```
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', nan], dtype=object)
```

Drop Duplicate Columns

```
Cust_Country = data_mod[['Country', 'Customer ID']].drop_duplicates()
```

Check Unique values for each column

```
def unique_counts(data_mod):
  for i in data_mod.columns:
    count = data_mod[i].nunique()
    print(i, ":" ,count)
unique_counts(data_mod)
```

```
Invoice : 24351
StockCode : 4559
Description : 4555
Quantity : 762
InvoiceDate : 21499
Price : 1470
Customer ID : 4042
Country : 40
```

Adding Column For Total Price

```
data_mod['Total_price'] = data_mod['Quantity'] * data_mod['Price']
```

Calculating Recency

```
data_mod['InvoiceDate'].min()
```

```
Timestamp('2009-01-12 07:45:00')
```

```
data_mod['InvoiceDate'].max()
```

```
Timestamp('2010-12-10 18:33:00')
```

```
import datetime as dt
PRESENT = dt.datetime(2011,12,10)
data_mod['InvoiceDate'] = pd.to_datetime(data_mod['InvoiceDate'])
```

RFM Calculation

```
rfm=data_mod.groupby('Customer ID').agg({'InvoiceDate': lambda x:(PRESENT - x.max()).days,
rfm['InvoiceDate'] = rfm['InvoiceDate'].astype(int)
rfm.rename (columns={'InvoiceDate': 'recency' , 'Invoice': 'frequency' , 'Total_price': 'm
```

```
rfm.head()
```

Customer ID	recency	frequency	monetary
12346.0	527	46	-64.68
12347.0	404	40	611.53
12348.0	438	20	222.16
12349.0	407	107	2646.99
12353.0	408	20	317.76

RFM Interpretation

```
qua = rfm.quantile(q=[0.25,0.50,0.75]) #qua=Quantiles
qua = qua.to_dict()
```

```
seg_rfm = rfm #segmented RFM
```

```
def RScore(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 FMScore(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


seg_rfm['r_quartile'] = seg_rfm['recency'].apply(RScore, args=('recency' , qua,))
seg_rfm['f_quartile'] = seg_rfm['frequency'].apply(RScore, args=('frequency' , qua,))
seg_rfm['m_quartile'] = seg_rfm['monetary'].apply(RScore, args=('monetary' , qua,))
seg_rfm.head()
```

Customer ID	recency	frequency	monetary	r_quartile	f_quartile	m_quartile
12346.0	527	46	-64.68	3	3	1
12347.0	404	40	611.53	1	2	2
12348.0	438	20	222.16	2	2	1
12349.0	407	107	2646.99	1	4	4
12353.0	408	20	317.76	1	2	2

```
seg_rfm['RFMScore'] = seg_rfm.r_quartile.map(str) + seg_rfm.f_quartile.map(str) + seg_rfm.
seg_rfm.head()
```

Customer ID	recency	frequency	monetary	r_quartile	f_quartile	m_quartile	RFMScore
12346.0	527	46	-64.68	3	3	1	331
12347.0	404	40	611.53	1	2	2	122
12348.0	438	20	222.16	2	2	1	221
12349.0	407	107	2646.99	1	4	4	144
12353.0	408	20	317.76	1	2	2	122

Top 10 Customers

```
seg_rfm[seg_rfm['RFMScore']== '122'].sort_values('monetary',ascending=False).head(10)
```

Customer ID	recency	frequency	monetary	r_quartile	f_quartile	m_quartile	RFMSco
12347.0	404	40	611.53	1	2	2	1
17473.0	408	34	607.00	1	2	2	1
17540.0	399	27	599.35	1	2	2	1
13627.0	401	38	593.53	1	2	2	1
12997.0	370	37	566.83	1	2	2	1
13998.0	369	31	566.70	1	2	2	1
15102.0	398	27	554.67	1	2	2	1
15948.0	394	35	553.85	1	2	2	1
14673.0	409	34	553.66	1	2	2	1
12494.0	369	35	546.57	1	2	2	1

✓ 0s completed at 18:00