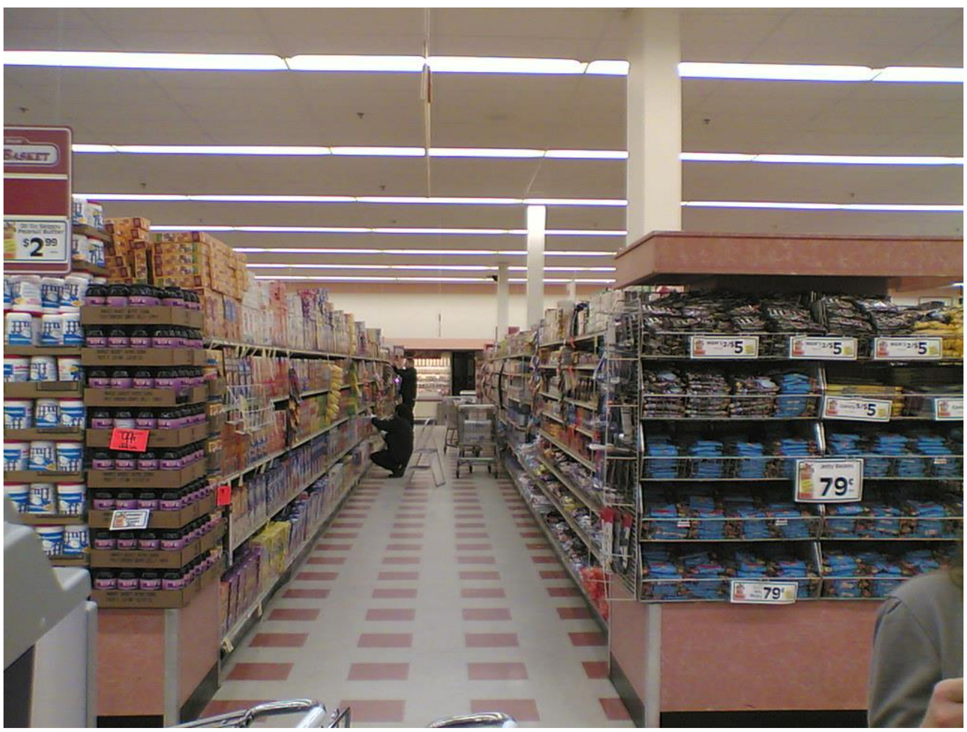
MARKET BASKET INSIGHTS

TEAM MEMBER: N.DHIVYA (922121106010)

PHASE 2 SUBMISSION DOCUMENT

PROJECT: Market basket insights



INTRODUCTION Market basket analysis is a datamining technique used by retailers to increase sales by better understanding customer purchasing patterns. It involves analyzing large data sets, such as purchase history, to reveal product groupings, as well as products that are likely to be purchased together.

DATA SOURCE **https://www.kaggle.com/datasets/aslanahmedov/market-basket-analysis**

PROGRAM

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib import style
style.use("ggplot")
sns.set_palette("bright")
from warnings import filterwarnings
filterwarnings("ignore")
import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```
/kaggle/input/market-basket-analysis/Assignment-1_Data.xlsx
/kaggle/input/market-basket-analysis/Assignment-1_Data.csv
```

```
df = pd.read_csv("../input/market-basket-analysis/Assignment-1_Data.csv",
sep=';')
df.head()
```

Out[3]:

	BillNo	Itemname	Quantity	Date	Price	CustomerID	Country
0	536365	WHITE HANGING HEART T-LIGHT HOLDER	6	01.12.2010 08:26	2,55	17850.0	United Kingdom
1	536365	WHITE METAL LANTERN	6	01.12.2010 08:26	3,39	17850.0	United Kingdom

2	536365	CREAM CUPID HEARTS COAT HANGER	8	01.12.2010 08:26	2,75	17850.0	United Kingdom
3	536365	KNITTED UNION FLAG HOT WATER BOTTLE	6	01.12.2010 08:26	3,39	17850.0	United Kingdom
4	536365	RED WOOLLY HOTTIE WHITE HEART.	6	01.12.2010 08:26	3,39	17850.0	United Kingdom

In [4]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 522064 entries, 0 to 522063
Data columns (total 7 columns):
 #   Column      Non-Null Count   Dtype
---  ------      --------------   -----
 0   BillNo      522064 non-null  object
 1   Itemname    520609 non-null  object
 2   Quantity    522064 non-null  int64
 3   Date        522064 non-null  object
 4   Price       522064 non-null  object
 5   CustomerID  388023 non-null  float64
 6   Country     522064 non-null  object
dtypes: float64(1), int64(1), object(5)
memory usage: 27.9+ MB
```

```
df["Price"] = df["Price"].str.replace(",",".")
df["Price"] = df["Price"].astype("float64")
df.Date.unique()
```

```
array(['01.12.2010 08:26', '01.12.2010 08:28', '01.12.2010 08:34', ...,
       '09.12.2011 12:31', '09.12.2011 12:49', '09.12.2011 12:50'],
      dtype=object)
```

```
today = "2012-01-01"
today = pd.to_datetime(today)
df["Date"] = pd.to_datetime(df["Date"])
```

RFM ANALYSIS

```
rec_table = df.groupby(["CustomerID"]).agg({"Date": lambda x: ((today - x.max()).days)})
rec_table.columns = ["Recency"]
```

In [9]

```
rec_table.head()
```

	Recency
CustomerID	
12346.0	347
12347.0	61
12349.0	40
12350.0	332
12352.0	94

Frequency

```
freq_table = df.drop_duplicates(subset = "BillNo").groupby(["CustomerID"])[["BillNo"]].count()
freq_table.columns = ["Frequency"]
freq_table.head()
```

Out[10]:

	Frequency
CustomerID	
12346.0	1
12347.0	7
12349.0	1
12350.0	1
12352.0	8

Monetary

In [11]:

```
df["Total_Price"] = df["Quantity"] * df["Price"]
```

In [12]:

```
monetary_table = df.groupby(["CustomerID"])[["Total_Price"]].sum()
monetary_table.columns = ["Monetary"]
monetary_table.head()
```

Out[12]:

	Monetary
CustomerID	
12346.0	77183.60
12347.0	4310.00
12349.0	1757.55
12350.0	334.40
12352.0	2506.04

In [13]:

```
rfm_data = pd.concat([rec_table, freq_table, monetary_table], axis = 1)
rfm_data.head()
```

Out[13]:

	Recency	Frequency	Monetary
CustomerID			
12346.0	347	1	77183.60
12347.0	61	7	4310.00
12349.0	40	1	1757.55
12350.0	332	1	334.40
12352.0	94	8	2506.04

In [14]:

```
rfm_data.describe()
```

Out[14]:

	Recency	Frequency	Monetary
count	4297.000000	4297.000000	4297.000000
mean	126.545264	4.227368	1993.140888
std	115.234387	7.091298	8588.143093
min	21.000000	1.000000	0.000000

25%	43.000000	1.000000	306.720000
50%	82.000000	2.000000	668.580000
75%	183.000000	5.000000	1652.580000
max	718.000000	210.000000	280206.020000

Scaling the data for clustering.

CLUSTERING

In [15]:

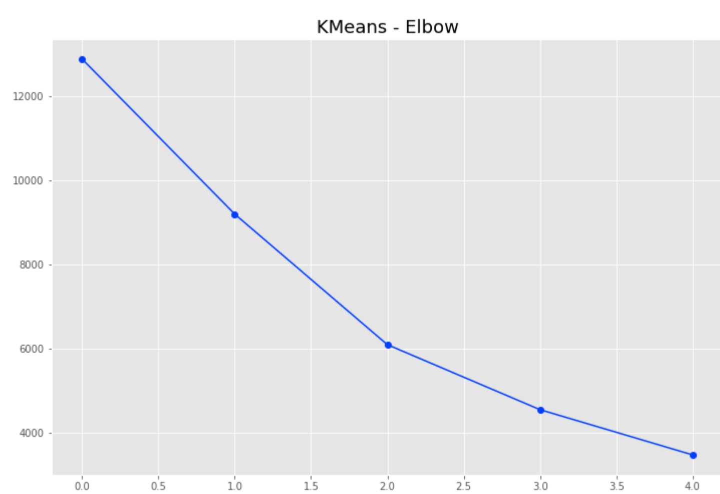
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
rfm_scaled = scaler.fit_transform(rfm_data)
```

Let's determine a cluster number

```
inertia = []

from sklearn.cluster import KMeans
for i in np.arange(1,6):
    kmeans = KMeans(n_clusters = i)
    kmeans.fit(rfm_scaled)
    inertia.append(kmeans.inertia_)

plt.figure(figsize = (12,8))
plt.plot(inertia, marker = "o")
plt.title("KMeans - Elbow", fontsize = 18);
```



Best cluster number is 3. Let's create a clustering model.

```
kmeans = KMeans(n_clusters = 3)
kmeans.fit(rfm_scaled)
rfm_data["Cluster_No"] = (kmeans.labels_ + 1)
```

```
rfm_data.head()
```

	Recency	Frequency	Monetary	Cluster_No
CustomerID				
12346.0	347	1	77183.60	3
12347.0	61	7	4310.00	2
12349.0	40	1	1757.55	2
12350.0	332	1	334.40	1
12352.0	94	8	2506.04	2

Analyzing of Clustering

In [19]:

```
rfm_data.groupby(["Cluster_No"])[["Recency", "Frequency",
"Monetary"]].mean()
```

Out[19]:

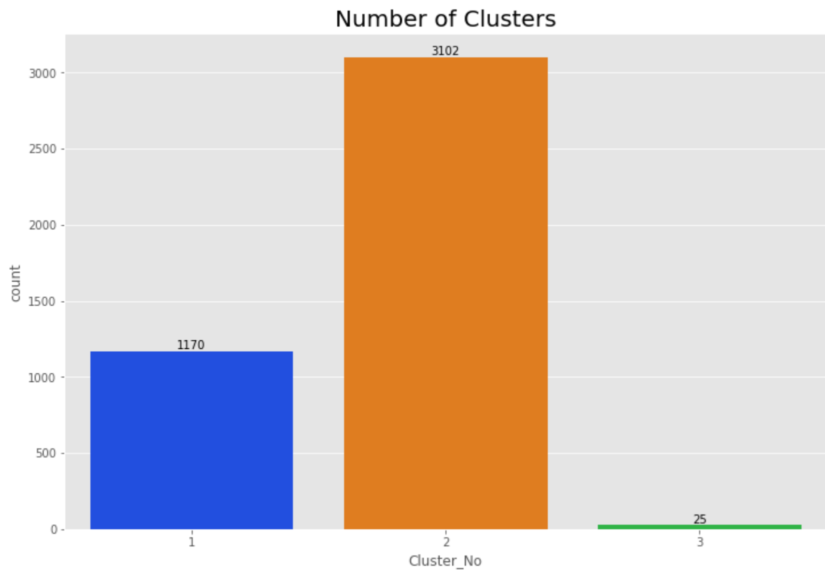
	Recency	Frequency	Monetary
Cluster_No			
1	281.745299	1.545299	495.484189
2	68.634429	4.797872	1913.384218
3	48.760000	58.960000	81979.682000

Hmm. Our model determine **3 clusters** that

- **Cluster 1** --> Customers who haven't been here in a long time. We need to do some discount for them. We can still turn them back.
- **Cluster 2** --> Middle-level customers.
- **Cluster 3** --> Premium customers. We don't want to lose them. They spend a lot of money for us, and their recency is good.

In [20]:

```python
plt.figure(figsize = (12,8))
ax = sns.countplot(rfm_data.Cluster_No)
plt.title("Number of Clusters", fontsize = 20);
for bars in ax.containers:
    ax.bar_label(bars)
```



- **As we can see, only 25 people are premium customers,**
- **3102 people are middle-level customers**
- **1170 people are customers that we can turn back.**

Let's visualize them with scatterplot.

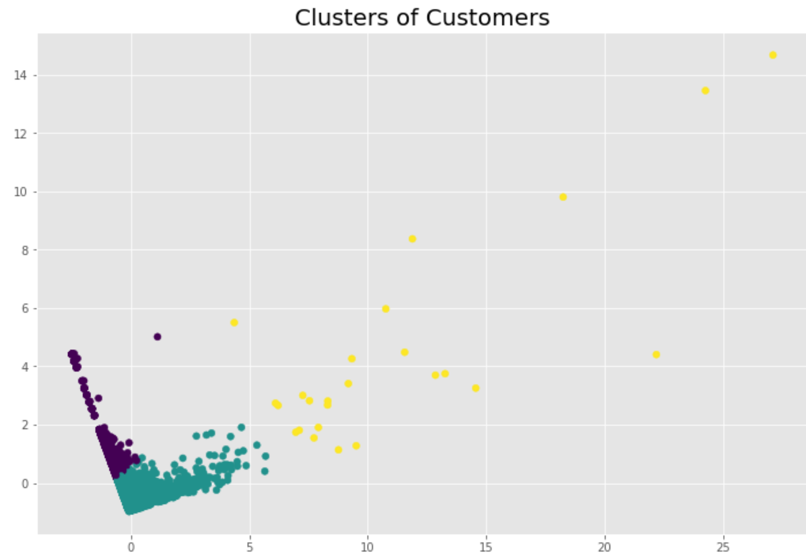
In [21]:

```python
from sklearn.decomposition import PCA

pca = PCA(n_components = 2)
pca = pca.fit_transform(rfm_scaled)
```

In [22]:

```python
plt.figure(figsize = (12,8))
plt.scatter(pca[:,0], pca[:,1], c = kmeans.labels_)
plt.title("Clusters of Customers", fontsize = 20);
```



ASSOCIATION ANALYSIS

Association analysis is the task of finding interesting relationships in large datasets. These interesting relationships can take two forms: frequent item sets or association rules. Frequent item sets are a collection of items that frequently occur together. The second way to view interesting relationships is association rules. Association rules suggest that a strong relationship exists between two items.

Now, we will take a look at which items are related to each other.

In [23]:

```
data_apr = df.groupby(["BillNo", "Itemname"])[["Quantity"]].sum(
).unstack().reset_index().fillna(0).set_index("BillNo")
```

In [24]:

```
data_apr.head()

def num(x):
    if x <= 0:
        return 0
    elif x >=1:
        return 1
basket_new = data_apr.applymap(num)
```

In [26]:

```
basket_new.nunique()
```

Out[26]:

```
         Itemname
Quantity  *Boombox Ipod Classic          2
          *USB Office Mirror Ball        2
          10 COLOUR SPACEBOY PEN         2
          12 COLOURED PARTY BALLOONS     2
          12 DAISY PEGS IN WOOD BOX      2
                                        ..
          wrongly marked carton 22804    1
          wrongly marked. 23343 in box   1
          wrongly sold (22719) barcode   2
          wrongly sold as sets           1
          wrongly sold sets              1
Length: 4185, dtype: int64
```

In [27]:

```
#!pip install mlxtend
```

In [28]:

```
from mlxtend.frequent_patterns import apriori

apr = apriori(basket_new, min_support = 0.02, use_colnames = True)
apr.sort_values(by = "support", ascending = False)
```

```
/opt/conda/lib/python3.7/site-
packages/mlxtend/frequent_patterns/fpcommon.py:115: DeprecationWarning:
DataFrames with non-bool types result in worse computationalperformance
and their support might be discontinued in the future.Please use a
DataFrame with bool type
  DeprecationWarning,
```

Out[28]:

	support	itemsets
263	0.108956	((Quantity, WHITE HANGING HEART T-LIGHT HOLDER))
97	0.102128	((Quantity, JUMBO BAG RED RETROSPOT))
190	0.094211	((Quantity, REGENCY CAKESTAND 3 TIER))
151	0.081940	((Quantity, PARTY BUNTING))
122	0.076249	((Quantity, LUNCH BAG RED RETROSPOT))
...	...	...

274	0.020139	((Quantity, WOODEN UNION JACK BUNTING))
245	0.020139	((Quantity, STRAWBERRY SHOPPER BAG))
219	0.020139	((Quantity, SET OF 60 I LOVE LONDON CAKE CASES))
195	0.020040	((Quantity, RIBBON REEL STRIPES DESIGN))
354	0.020040	((Quantity, WOODEN PICTURE FRAME WHITE FINISH)...

358 rows × 2 columns

In [29]:

```
from mlxtend.frequent_patterns import association_rules
end = association_rules(apr, metric = "lift", min_threshold = 1)
end.sort_values(by = "confidence", ascending = False)
```

Out[29]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
154	((Quantity, ROSES REGENCY TEACUP AND SAUCER), ...	((Quantity, GREEN REGENCY TEACUP AND SAUCER))	0.028204	0.048243	0.025482	0.903509	18.728115	0.024122	9.863659
153	((Quantity, PINK REGENCY TEACUP AND SAUCER), (...	((Quantity, ROSES REGENCY TEACUP AND SAUCER))	0.029936	0.050124	0.025482	0.851240	16.982778	0.023982	6.385280
25	((Quantity, PINK REGENCY TEACUP AND SAUCER))	((Quantity, GREEN REGENCY TEACUP AND SAUCER))	0.036418	0.048243	0.029936	0.822011	17.038810	0.028179	5.347273
160	((Quantity, JUMBO STORAGE BAG SUKI), (Quantity...	((Quantity, JUMBO BAG RED RETROSPOT))	0.025433	0.102128	0.020386	0.801556	7.848573	0.017789	4.524572
139	((Quantity, PINK REGENCY TEACUP AND SAUCER))	((Quantity, ROSES REGENCY TEACUP AND SAUCER))	0.036418	0.050124	0.028204	0.774457	15.450905	0.026378	4.211500
...	...	...	...	...	...	...	...	...	...

70	((Quantity, JUMBO BAG RED RETROSPOT))	((Quantity, JUMBO STORAGE BAG SKULLS))	0.102128	0.034785	0.020435	0.200097	5.752430	0.016883	1.206665
161	((Quantity, JUMBO BAG RED RETROSPOT))	((Quantity, JUMBO STORAGE BAG SUKI), (Quantity...	0.102128	0.025433	0.020386	0.199612	7.848573	0.017789	1.217619
42	((Quantity, JUMBO BAG RED RETROSPOT))	((Quantity, JUMBO BAG ALPHABET))	0.102128	0.043790	0.020336	0.199128	4.547316	0.015864	1.193961
131	((Quantity, WHITE HANGING HEART T-LIGHT HOLDER))	((Quantity, NATURAL SLATE HEART CHALKBOARD))	0.108956	0.060960	0.020336	0.186649	3.061823	0.013695	1.154532
149	((Quantity, WHITE HANGING HEART T-LIGHT HOLDER))	((Quantity, WOODEN PICTURE FRAME WHITE FINISH))	0.108956	0.054033	0.020040	0.183924	3.403936	0.014152	1.159165

```
164 rows × 9 columns
```

CONCLUSION

Look at the confidences, it indicates the possibility that customers buying the X product will buy the Y product. We need to make a decision for them. Maybe in our website, when the customer click on first one, we need to show them the other item.

For example: When our customer clicks on PINK REGENCY TEACUP AND SAUCER, we need to show them GREEN REGENCY TEACUP AND SAUCER and maximize our profit.

In [30]:

```
pd.concat([end["antecedents"], end["consequents"], end["confidence"]], axis = 1
         ).sort_values(by = "confidence", ascending = False)[0:10]
```

Out[30]:

	antecedents	consequents	confidence
154	((Quantity, ROSES REGENCY TEACUP AND SAUCER), ...	((Quantity, GREEN REGENCY TEACUP AND SAUCER))	0.903509
153	((Quantity, PINK REGENCY TEACUP AND SAUCER), (...	((Quantity, ROSES REGENCY TEACUP AND SAUCER))	0.851240

25	((Quantity, PINK REGENCY TEACUP AND SAUCER))	((Quantity, GREEN REGENCY TEACUP AND SAUCER))	0.822011
160	((Quantity, JUMBO STORAGE BAG SUKI), (Quantity...	((Quantity, JUMBO BAG RED RETROSPOT))	0.801556
139	((Quantity, PINK REGENCY TEACUP AND SAUCER))	((Quantity, ROSES REGENCY TEACUP AND SAUCER))	0.774457
29	((Quantity, GREEN REGENCY TEACUP AND SAUCER))	((Quantity, ROSES REGENCY TEACUP AND SAUCER))	0.749744
28	((Quantity, ROSES REGENCY TEACUP AND SAUCER))	((Quantity, GREEN REGENCY TEACUP AND SAUCER))	0.721619
23	((Quantity, GARDENERS KNEELING PAD CUP OF TEA))	((Quantity, GARDENERS KNEELING PAD KEEP CALM))	0.721485
10	((Quantity, CHARLOTTE BAG PINK POLKADOT))	((Quantity, RED RETROSPOT CHARLOTTE BAG))	0.704607
152	((Quantity, ROSES REGENCY TEACUP AND SAUCER), ...	((Quantity, PINK REGENCY TEACUP AND SAUCER))	0.704514