# ONLINE RETAIL STORE

## Load the Data

## Prepare the Data:

1. Create a new df where country = United Kingdom

**retail\_uk = retail\_df[retail\_df['Country']=='United Kingdom']**

1. Remove cancelled Orders

**retail\_uk = retail\_uk[retail\_uk['Quantity']>0]**

1. Drop records where Customer ID is null

**retail\_uk.dropna(subset=['CustomerID'],how='all',inplace=True)**

## RFM ANALYSIS:

* RECENCY (R): Days since last purchase
* FREQUENCY (F): Total number of purchases
* MONETARY VALUE (M): Total money this customer spent.

### RECENCY:

1)last date available in our dataset

**retail\_uk['InvoiceDate'].max()**

2)Get Present Date:

**now = dt.date(2011,12,9)**

3)create a new column called date which contains the date of invoice only

**retail\_uk['date'] = retail\_uk['InvoiceDate'].dt.date**

4)group by customers and check last date of purshace

**recency\_df = retail\_uk.groupby(by='CustomerID', as\_index=False)['date'].max()**

**recency\_df.columns = ['CustomerID','LastPurshaceDate']**

5)calculate recency:

recency\_df['Recency'] = recency\_df['LastPurshaceDate'].apply(lambda x: (now - x).days)

|  | **CustomerID** | **LastPurshaceDate** | **Recency** |
| --- | --- | --- | --- |
| **0** | 12346.0 | 2011-01-18 | 325 |
| **1** | 12747.0 | 2011-12-07 | 2 |
| **2** | 12748.0 | 2011-12-09 | 0 |
| **3** | 12749.0 | 2011-12-06 | 3 |
| **4** | 12820.0 | 2011-12-06 | 3 |

### FREQUENCY:

### Check how many invoices are registered by the same customer.

### **frequency\_df = retail\_uk\_copy.groupby(by=['CustomerID'], as\_index=False)['InvoiceNo'].count()**

### Monetary:

create column total cost

**retail\_uk['TotalCost'] = retail\_uk['Quantity'] \* retail\_uk['UnitPrice']**

**monetary\_df = retail\_uk.groupby(by='CustomerID',as\_index=False).agg({'TotalCost': 'sum'})**

**monetary\_df.columns = ['CustomerID','Monetary']**

**monetary\_df.head()**

| **CustomerID** | **Monetary** |
| --- | --- |
| **0** | 12346.0 | 77183.60 |
| **1** | 12747.0 | 658.89 |
| **2** | 12748.0 | 3739.23 |
| **3** | 12749.0 | 98.35 |
| **4** | 12820.0 | 58.20 |

## Create RFM Table:

#merge recency dataframe with frequency dataframe

**temp\_df = recency\_df.merge(frequency\_df,on='CustomerID')**

|  | **CustomerID** | **Recency** | **Frequency** |
| --- | --- | --- | --- |
| **0** | 12346.0 | 325 | 1 |
| **1** | 12747.0 | 2 | 10 |
| **2** | 12748.0 | 0 | 196 |
| **3** | 12749.0 | 3 | 5 |
| **4** | 12820.0 | 3 | 4 |

#merge with monetary dataframe to get a table with the 3 columns

**rfm\_df = temp\_df.merge(monetary\_df,on='CustomerID')**

|  | **Recency** | **Frequency** | **Monetary** |
| --- | --- | --- | --- |
| **CustomerID** |  |  |  |
| **12346.0** | 325 | 1 | 77183.60 |
| **12747.0** | 2 | 10 | 658.89 |
| **12748.0** | 0 | 196 | 3739.23 |
| **12749.0** | 3 | 5 | 98.35 |
| **12820.0** | 3 | 4 | 58.20 |

### Customer Segmentation with RFM Model:

Pareto’s rule says ****80% of the results come from 20% of the causes****.

Similarly, ****20% customers contribute to 80% of your total revenue****. Let's verify that because that will help us know which customers to focus on when marketing new products.

### **Applying 80-20 rule**

In [23]:

*#get the 80% of the revenue*

pareto\_cutoff **=** rfm\_df['Monetary'].sum() **\*** 0.8

print("The 80% of total revenue is: ",round(pareto\_cutoff,2))

The 80% of total revenue is: 890679.54

**1)Create Customer Rank Table:**

customers\_rank = rfm\_df

# Create a new column that is the rank of the value of coverage in ascending order

customers\_rank['Rank'] = customers\_rank['Monetary'].rank(ascending=0)

|  | **Recency** | **Frequency** | **Monetary** | **Rank** |
| --- | --- | --- | --- | --- |
| **CustomerID** |  |  |  |  |
| **12346.0** | 325 | 1 | 77183.60 | 2.0 |
| **12747.0** | 2 | 10 | 658.89 | 183.0 |
| **12748.0** | 0 | 196 | 3739.23 | 34.0 |
| **12749.0** | 3 | 5 | 98.35 | 1140.0 |
| **12820.0** | 3 | 4 | 58.20 | 1665.5 |

**2)Get the Top Customers:**

customers\_rank.sort\_values('Rank',ascending=True)

**3)Get top 20% of the customers:**

top\_20\_cutoff = 3863 \*20 /100

top\_20\_cutoff

772.6