In [ ]:

```
# This Python 3 environment comes with many helpful analytics libraries insta
lled
# It is defined by the kaggle/python docker image: https://github.com/kaggle/
docker-python
# For example, here's several helpful packages to load in

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the "../input/" directory.

import time, warnings
import datetime as dt

#visualizations
import matplotlib.pyplot as plt
from pandas.plotting import scatter_matrix
%matplotlib inline
import seaborn as sns

warnings.filterwarnings("ignore")
```

Get the Data

In [ ]:

```
#load the dataset
retail_df = pd.read_csv('../input/data.csv',encoding="ISO-8859-1",dtype={'Cus
tomerID': str,'InvoiceID': str})
retail_df.head()
```

Prepare the Data

As customer clusters may vary by geography, I’ ll restrict the data to only United Kingdom customers, which contains most of our customers historical data.

In [ ]:

```
retail_uk = retail_df[retail_df['Country']=='United Kingdom']
#check the shape
retail_uk.shape
```

In [ ]:

```
#remove canceled orders
retail_uk = retail_uk[retail_uk['Quantity']>0]
retail_uk.shape
```

In [ ]:

```
#remove rows where customerID are NA
retail_uk.dropna(subset=['CustomerID'],how='all',inplace=True)
retail_uk.shape
```

In [ ]:

```
#restrict the data to one full year because it's better to use a metric per M
onths or Years in RFM
retail_uk = retail_uk[retail_uk['InvoiceDate']>= "2010-12-09"]
retail_uk.shape
```

In [ ]:

```
print("Summary..")
#exploring the unique values of each attribute
print("Number of transactions: ", retail_uk['InvoiceNo'].nunique())
print("Number of products bought: ",retail_uk['StockCode'].nunique())
print("Number of customers:", retail_uk['CustomerID'].nunique() )
print("Percentage of customers NA: ", round(retail_uk['CustomerID'].isnull().
sum() * 100 / len(retail_df),2),"%" )
```

RFM Analysis

RFM (Recency, Frequency, Monetary) analysis is a customer segmentation technique that uses past purchase behavior to divide customers into groups. RFM helps divide customers into various categories or clusters to identify customers who are more likely to respond to promotions and also for future personalization services.

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

We will create those 3 customer attributes for each customer.

Recency

To calculate recency, we need to choose a date point from which we evaluate **how many days ago was the customer's last purchase**.

In [ ]:

```
#last date available in our dataset
retail_uk['InvoiceDate'].max()
```

The last date we have is 2011-12-09 so we will use it as reference.

In [ ]:

```
now = dt.date(2011,12,9)
print(now)
```

In [ ]:

```
#create a new column called date which contains the date of invoice only
retail_uk['date'] = pd.DatetimeIndex(retail_uk['InvoiceDate']).date
```

In [ ]:

```
retail_uk.head()
```

In [ ]:

```
#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']
recency_df.head()
```

In [ ]:

```
#calculate recency
recency_df['Recency'] = recency_df['LastPurshaceDate'].apply(lambda x: (now -
x).days)
```

In [ ]:

```
recency_df.head()
```

In [ ]:

```
#drop LastPurchaseDate as we don't need it anymore
recency_df.drop('LastPurshaceDate',axis=1,inplace=True)
```

Frequency

Frequency helps us to know how many times a customer purchased from us. To do that we need to check **how many invoices are registered by the same customer**.

In [ ]:

```
# drop duplicates
retail_uk_copy = retail_uk
retail_uk_copy.drop_duplicates(subset=['InvoiceNo', 'CustomerID'], keep="firs
t", inplace=True)
#calculate frequency of purchases
frequency_df = retail_uk_copy.groupby(by=['CustomerID'], as_index=False)['Inv
oiceNo'].count()
frequency_df.columns = ['CustomerID','Frequency']
frequency_df.head()
```

Monetary

Monetary attribute answers the question: **How much money did the customer spent over time?**

To do that, first, we will create a new column total cost to have the total price per invoice.

In [ ]:

```
#create column total cost
retail_uk['TotalCost'] = retail_uk['Quantity'] * retail_uk['UnitPrice']
```

In [ ]:

```
monetary_df = retail_uk.groupby(by='CustomerID',as_index=False).agg({'TotalCo
st': 'sum'})
monetary_df.columns = ['CustomerID','Monetary']
monetary_df.head()
```

Create RFM Table

In [ ]:

```
#merge recency dataframe with frequency dataframe
temp_df = recency_df.merge(frequency_df,on='CustomerID')
temp_df.head()
```

In [ ]:

```
#merge with monetary dataframe to get a table with the 3 columns
rfm_df = temp_df.merge(monetary_df,on='CustomerID')
#use CustomerID as index
rfm_df.set_index('CustomerID',inplace=True)
#check the head
rfm_df.head()
```

RFM Table Correctness verification

In [ ]:

```
retail_uk[retail_uk['CustomerID']=='12820']
```

In [ ]:

```
(now - dt.date(2011,9,26)).days == 74
```

Customer segments with RFM Model

The simplest way to create customers segments from RFM Model is to use **Quartiles**. We assign a score from 1 to 4 to Recency, Frequency and Monetary. Four is the best/highest value, and one is the lowest/worst value. A final RFM score is calculated simply by combining individual RFM score numbers.

Note: Quintiles (score from 1-5) offer better granularity, in case the business needs that but it will be more challenging to create segments since we will have 555 possible combinations. So, we will use quartiles

RFM Quartiles

In [ ]:

```
quantiles = rfm_df.quantile(q=[0.25,0.5,0.75])
quantiles
```

In [ ]:

```
quantiles.to_dict()
```

Creation of RFM Segments

We will create two segmentation classes since, high recency is bad, while high frequency and monetary value is good.

In [ ]:

```
# Arguments (x = value, p = recency, monetary_value, frequency, d = quartiles
dict)
def RScore(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
# Arguments (x = value, p = recency, monetary_value, frequency, k = quartiles
dict)
def FMScore(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
```

In [ ]:

```
#create rfm segmentation table
rfm_segmentation = rfm_df
rfm_segmentation['R_Quartile'] = rfm_segmentation['Recency'].apply(RScore, ar
gs=('Recency',quantiles,))
rfm_segmentation['F_Quartile'] = rfm_segmentation['Frequency'].apply(FMScore,
args=('Frequency',quantiles,))
rfm_segmentation['M_Quartile'] = rfm_segmentation['Monetary'].apply(FMScore,
args=('Monetary',quantiles,))
```

In [ ]:

```
rfm_segmentation.head()
```

Now that we have the score of each customer, we can represent our customer segmentation. First, we need to combine the scores (R_Quartile, F_Quartile,M_Quartile) together.

In [ ]:

```
rfm_segmentation['RFMScore'] = rfm_segmentation.R_Quartile.map(str) \
                            + rfm_segmentation.F_Quartile.map(str) \
                            + rfm_segmentation.M_Quartile.map(str)
rfm_segmentation.head()
```

Best Recency score = 4: most recently purchase. Best Frequency score = 4: most quantity purchase. Best Monetary score = 4: spent the most.

Let's see who are our **Champions** (best customers).

In [ ]:

```
rfm_segmentation[rfm_segmentation['RFMScore']=='444'].sort_values('Monetary',
ascending=False).head(10)
```

We can find here (http://www.blastam.com/blog/rfm-analysis-boosts-sales) a suggestion of key segments and then we can decide which segment to consider for further study.

Note: the suggested link use the opposite valuation: 1 as highest/best score and 4 is the lowest.

How many customers do we have in each segment?

In [ ]:

```
print("Best Customers: ",len(rfm_segmentation[rfm_segmentation['RFMScore']=='444']))
print('Loyal Customers: ',len(rfm_segmentation[rfm_segmentation['F_Quartile']==4]))
print("Big Spenders: ",len(rfm_segmentation[rfm_segmentation['M_Quartile']==4]))
print('Almost Lost: ', len(rfm_segmentation[rfm_segmentation['RFMScore']=='244']))
print('Lost Customers: ',len(rfm_segmentation[rfm_segmentation['RFMScore']=='144']))
print('Lost Cheap Customers: ',len(rfm_segmentation[rfm_segmentation['RFMScore']=='111']))
```

Now that we knew our customers segments we can choose how to target or deal with each segment.

For example:

Best Customers - Champions: Reward them. They can be early adopters to new products. Suggest them "Refer a friend".

At Risk: Send them personalized emails to encourage them to shop.

More ideas about what actions to perform in Ometria (http://54.73.114.30/customer-segmentation#).

In [ ]:

In [ ]: