RFM (Recency Frequency Monetary) Analysis:- RFM is a method used for analyzing customer value. It is commonly used in database marketing and direct marketing and has received particular attention in retail and professional services industries

RFM stands for the three dimensions:

Recency – How recently did the customer purchase? Frequency – How often do they purchase? Monetary Value – How much do they spend?

In [1]:
```
#importing all important package..
import pandas as pd
import numpy as np
import datetime
```

In [2]:
```
#load data into pandas dataframe..
df = pd.read_csv("Dataset.csv")
df.head()
```

Out[2]:

	Invoice	StockCode	Description	Quantity	InvoiceDate	Price	Customer ID	Country
0	489434	85048	15CM CHRISTMAS GLASS BALL 20 LIGHTS	12	1.12.2009 07:45	6,95	13085.0	United Kingdom
1	489434	79323P	PINK CHERRY LIGHTS	12	1.12.2009 07:45	6,75	13085.0	United Kingdom
2	489434	79323W	WHITE CHERRY LIGHTS	12	1.12.2009 07:45	6,75	13085.0	United Kingdom
3	489434	22041	RECORD FRAME 7" SINGLE SIZE	48	1.12.2009 07:45	2,1	13085.0	United Kingdom
4	489434	21232	STRAWBERRY CERAMIC TRINKET BOX	24	1.12.2009 07:45	1,25	13085.0	United Kingdom

Out[2]:

	Invoice	StockCode	Description	Quantity	InvoiceDate	Price	Customer ID	Country
0	489434	85048	15CM CHRISTMAS GLASS BALL 20 LIGHTS	12	1.12.2009 07:45	6,95	13085.0	United Kingdom
1	489434	79323P	PINK CHERRY LIGHTS	12	1.12.2009 07:45	6,75	13085.0	United Kingdom
2	489434	79323W	WHITE CHERRY LIGHTS	12	1.12.2009 07:45	6,75	13085.0	United Kingdom
3	489434	22041	RECORD FRAME 7" SINGLE SIZE	48	1.12.2009 07:45	2,1	13085.0	United Kingdom
4	489434	21232	STRAWBERRY CERAMIC TRINKET BOX	24	1.12.2009 07:45	1,25	13085.0	United Kingdom

In [3]:
```
#information of dataset..
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1048575 entries, 0 to 1048574
Data columns (total 8 columns):
 #   Column       Non-Null Count    Dtype
---  ------       --------------    -----
```

```
 0   Invoice      1048575 non-null  object
 1   StockCode    1048575 non-null  object
 2   Description  1044203 non-null  object
 3   Quantity     1048575 non-null  int64
 4   InvoiceDate  1048575 non-null  object
 5   Price        1048575 non-null  object
 6   Customer ID  811893 non-null   float64
 7   Country      1048575 non-null  object
dtypes: float64(1), int64(1), object(6)
memory usage: 64.0+ MB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1048575 entries, 0 to 1048574
Data columns (total 8 columns):
 #   Column       Non-Null Count    Dtype
---  ------       --------------    -----
 0   Invoice      1048575 non-null  object
 1   StockCode    1048575 non-null  object
 2   Description  1044203 non-null  object
 3   Quantity     1048575 non-null  int64
 4   InvoiceDate  1048575 non-null  object
 5   Price        1048575 non-null  object
 6   Customer ID  811893 non-null   float64
 7   Country      1048575 non-null  object
dtypes: float64(1), int64(1), object(6)
memory usage: 64.0+ MB
```

In [4]:
```
df['Price'] = df['Price'].str.replace(',','.')
df.head()
```

Out[4]:

	Invoice	StockCode	Description	Quantity	InvoiceDate	Price	Customer ID	Country
0	489434	85048	15CM CHRISTMAS GLASS BALL 20 LIGHTS	12	1.12.2009 07:45	6.95	13085.0	United Kingdom
1	489434	79323P	PINK CHERRY LIGHTS	12	1.12.2009 07:45	6.75	13085.0	United Kingdom
2	489434	79323W	WHITE CHERRY LIGHTS	12	1.12.2009 07:45	6.75	13085.0	United Kingdom
3	489434	22041	RECORD FRAME 7" SINGLE SIZE	48	1.12.2009 07:45	2.1	13085.0	United Kingdom
4	489434	21232	STRAWBERRY CERAMIC TRINKET BOX	24	1.12.2009 07:45	1.25	13085.0	United Kingdom

Out[4]:

	Invoice	StockCode	Description	Quantity	InvoiceDate	Price	Customer ID	Country
0	489434	85048	15CM CHRISTMAS GLASS BALL 20 LIGHTS	12	1.12.2009 07:45	6.95	13085.0	United Kingdom
1	489434	79323P	PINK CHERRY LIGHTS	12	1.12.2009 07:45	6.75	13085.0	United Kingdom
2	489434	79323W	WHITE CHERRY LIGHTS	12	1.12.2009 07:45	6.75	13085.0	United Kingdom
3	489434	22041	RECORD FRAME 7" SINGLE SIZE	48	1.12.2009 07:45	2.1	13085.0	United Kingdom
4	489434	21232	STRAWBERRY CERAMIC TRINKET BOX	24	1.12.2009 07:45	1.25	13085.0	United Kingdom

In [5]:
```
# change datatype of price column
```

```
df = df.astype({"Price":"float64"})
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1048575 entries, 0 to 1048574
Data columns (total 8 columns):
 #   Column       Non-Null Count    Dtype  
---  ------       --------------    -----  
 0   Invoice      1048575 non-null  object 
 1   StockCode    1048575 non-null  object 
 2   Description  1044203 non-null  object 
 3   Quantity     1048575 non-null  int64  
 4   InvoiceDate  1048575 non-null  object 
 5   Price        1048575 non-null  float64
 6   Customer ID  811893 non-null   float64
 7   Country      1048575 non-null  object 
dtypes: float64(2), int64(1), object(5)
memory usage: 64.0+ MB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1048575 entries, 0 to 1048574
Data columns (total 8 columns):
 #   Column       Non-Null Count    Dtype  
---  ------       --------------    -----  
 0   Invoice      1048575 non-null  object 
 1   StockCode    1048575 non-null  object 
 2   Description  1044203 non-null  object 
 3   Quantity     1048575 non-null  int64  
 4   InvoiceDate  1048575 non-null  object 
 5   Price        1048575 non-null  float64
 6   Customer ID  811893 non-null   float64
 7   Country      1048575 non-null  object 
dtypes: float64(2), int64(1), object(5)
memory usage: 64.0+ MB
```

In [6]:
```
# change datatype of InvoiceDate column
df['InvoiceDate'] = pd.to_datetime(df['InvoiceDate'], errors='coerce')
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1048575 entries, 0 to 1048574
Data columns (total 8 columns):
 #   Column       Non-Null Count    Dtype         
---  ------       --------------    -----         
 0   Invoice      1048575 non-null  object        
 1   StockCode    1048575 non-null  object        
 2   Description  1044203 non-null  object        
 3   Quantity     1048575 non-null  int64         
 4   InvoiceDate  1048575 non-null  datetime64[ns]
 5   Price        1048575 non-null  float64       
 6   Customer ID  811893 non-null   float64       
 7   Country      1048575 non-null  object        
dtypes: datetime64[ns](1), float64(2), int64(1), object(4)
memory usage: 64.0+ MB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1048575 entries, 0 to 1048574
Data columns (total 8 columns):
 #   Column       Non-Null Count    Dtype         
---  ------       --------------    -----         
 0   Invoice      1048575 non-null  object        
 1   StockCode    1048575 non-null  object        
 2   Description  1044203 non-null  object        
 3   Quantity     1048575 non-null  int64         
 4   InvoiceDate  1048575 non-null  datetime64[ns]
 5   Price        1048575 non-null  float64       
 6   Customer ID  811893 non-null   float64       
 7   Country      1048575 non-null  object        
```

```
dtypes: datetime64[ns](1), float64(2), int64(1), object(4)
memory usage: 64.0+ MB
```

In [7]:
```
# change datatype of Customer ID column
df['Customer ID'] = df['Customer ID'].astype(str).apply(lambda x: x.replace('.0',''))
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1048575 entries, 0 to 1048574
Data columns (total 8 columns):
 #   Column       Non-Null Count    Dtype         
---  ------       --------------    -----         
 0   Invoice      1048575 non-null  object        
 1   StockCode    1048575 non-null  object        
 2   Description  1044203 non-null  object        
 3   Quantity     1048575 non-null  int64         
 4   InvoiceDate  1048575 non-null  datetime64[ns]
 5   Price        1048575 non-null  float64       
 6   Customer ID  1048575 non-null  object        
 7   Country      1048575 non-null  object        
dtypes: datetime64[ns](1), float64(1), int64(1), object(5)
memory usage: 64.0+ MB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1048575 entries, 0 to 1048574
Data columns (total 8 columns):
 #   Column       Non-Null Count    Dtype         
---  ------       --------------    -----         
 0   Invoice      1048575 non-null  object        
 1   StockCode    1048575 non-null  object        
 2   Description  1044203 non-null  object        
 3   Quantity     1048575 non-null  int64         
 4   InvoiceDate  1048575 non-null  datetime64[ns]
 5   Price        1048575 non-null  float64       
 6   Customer ID  1048575 non-null  object        
 7   Country      1048575 non-null  object        
dtypes: datetime64[ns](1), float64(1), int64(1), object(5)
memory usage: 64.0+ MB
```

In [8]:
```
# rename the column name for Customer ID to Customer_ID for future operation
df.rename(columns = {'Customer ID':'Customer_ID'}, inplace = True)
df.head()
```

Out[8]:

	Invoice	StockCode	Description	Quantity	InvoiceDate	Price	Customer_ID	Country
0	489434	85048	15CM CHRISTMAS GLASS BALL 20 LIGHTS	12	2009-01-12 07:45:00	6.95	13085	United Kingdom
1	489434	79323P	PINK CHERRY LIGHTS	12	2009-01-12 07:45:00	6.75	13085	United Kingdom
2	489434	79323W	WHITE CHERRY LIGHTS	12	2009-01-12 07:45:00	6.75	13085	United Kingdom
3	489434	22041	RECORD FRAME 7" SINGLE SIZE	48	2009-01-12 07:45:00	2.10	13085	United Kingdom
4	489434	21232	STRAWBERRY CERAMIC TRINKET BOX	24	2009-01-12 07:45:00	1.25	13085	United Kingdom

Out[8]:

	Invoice	StockCode	Description	Quantity	InvoiceDate	Price	Customer_ID	Country
0	489434	85048	15CM CHRISTMAS GLASS BALL 20 LIGHTS	12	2009-01-12 07:45:00	6.95	13085	United Kingdom
1	489434	79323P	PINK CHERRY LIGHTS	12	2009-01-12 07:45:00	6.75	13085	United Kingdom

2	489434	79323W	WHITE CHERRY LIGHTS	12	2009-01-12 07:45:00	6.75	13085	United Kingdom
3	489434	22041	RECORD FRAME 7" SINGLE SIZE	48	2009-01-12 07:45:00	2.10	13085	United Kingdom
4	489434	21232	STRAWBERRY CERAMIC TRINKET BOX	24	2009-01-12 07:45:00	1.25	13085	United Kingdom

In [9]:
```
# creating new column total revenue as Tot_rev
df["Tot_Rev"]=(df['Price']*df['Quantity'])
df.head()
```

Out[9]:

	Invoice	StockCode	Description	Quantity	InvoiceDate	Price	Customer_ID	Country	Tot_Rev
0	489434	85048	15CM CHRISTMAS GLASS BALL 20 LIGHTS	12	2009-01-12 07:45:00	6.95	13085	United Kingdom	83.4
1	489434	79323P	PINK CHERRY LIGHTS	12	2009-01-12 07:45:00	6.75	13085	United Kingdom	81.0
2	489434	79323W	WHITE CHERRY LIGHTS	12	2009-01-12 07:45:00	6.75	13085	United Kingdom	81.0
3	489434	22041	RECORD FRAME 7" SINGLE SIZE	48	2009-01-12 07:45:00	2.10	13085	United Kingdom	100.8
4	489434	21232	STRAWBERRY CERAMIC TRINKET BOX	24	2009-01-12 07:45:00	1.25	13085	United Kingdom	30.0

Out[9]:

	Invoice	StockCode	Description	Quantity	InvoiceDate	Price	Customer_ID	Country	Tot_Rev
0	489434	85048	15CM CHRISTMAS GLASS BALL 20 LIGHTS	12	2009-01-12 07:45:00	6.95	13085	United Kingdom	83.4
1	489434	79323P	PINK CHERRY LIGHTS	12	2009-01-12 07:45:00	6.75	13085	United Kingdom	81.0
2	489434	79323W	WHITE CHERRY LIGHTS	12	2009-01-12 07:45:00	6.75	13085	United Kingdom	81.0
3	489434	22041	RECORD FRAME 7" SINGLE SIZE	48	2009-01-12 07:45:00	2.10	13085	United Kingdom	100.8
4	489434	21232	STRAWBERRY CERAMIC TRINKET BOX	24	2009-01-12 07:45:00	1.25	13085	United Kingdom	30.0

In [10]:
```
#sorting the Customer_ID
df = df.sort_values(by="Customer_ID",ascending=0)
df.head()
```

Out[10]:

	Invoice	StockCode	Description	Quantity	InvoiceDate	Price	Customer_ID	Country	Tot_Rev
524287	538074	21743	NaN	-80	2010-09-12 14:11:00	0.00	nan	United Kingdom	-0.00
676234	549524	22359	GLASS JAR KINGS CHOICE	1	2011-08-04 15:42:00	5.79	nan	United Kingdom	5.79
676236	549524	22362	GLASS JAR PEACOCK BATH SALTS	1	2011-08-04 15:42:00	5.79	nan	United Kingdom	5.79
676237	549524	22411	JUMBO SHOPPER VINTAGE RED PAISLEY	6	2011-08-04 15:42:00	4.13	nan	United Kingdom	24.78

676238	549524	22424	ENAMEL BREAD BIN CREAM	1	2011-08-04 15:42:00	24.96	nan	United Kingdom	24.96

Out[10]:

	Invoice	StockCode	Description	Quantity	InvoiceDate	Price	Customer_ID	Country	Tot_Rev
524287	538074	21743	NaN	-80	2010-09-12 14:11:00	0.00	nan	United Kingdom	-0.00
676234	549524	22359	GLASS JAR KINGS CHOICE	1	2011-08-04 15:42:00	5.79	nan	United Kingdom	5.79
676236	549524	22362	GLASS JAR PEACOCK BATH SALTS	1	2011-08-04 15:42:00	5.79	nan	United Kingdom	5.79
676237	549524	22411	JUMBO SHOPPER VINTAGE RED PAISLEY	6	2011-08-04 15:42:00	4.13	nan	United Kingdom	24.78
676238	549524	22424	ENAMEL BREAD BIN CREAM	1	2011-08-04 15:42:00	24.96	nan	United Kingdom	24.96

In [11]:
```
#removing Not a Number rows in dataframe
df=df.dropna()
df.head()
```

Out[11]:

	Invoice	StockCode	Description	Quantity	InvoiceDate	Price	Customer_ID	Country	Tot_Rev
676234	549524	22359	GLASS JAR KINGS CHOICE	1	2011-08-04 15:42:00	5.79	nan	United Kingdom	5.79
676236	549524	22362	GLASS JAR PEACOCK BATH SALTS	1	2011-08-04 15:42:00	5.79	nan	United Kingdom	5.79
676237	549524	22411	JUMBO SHOPPER VINTAGE RED PAISLEY	6	2011-08-04 15:42:00	4.13	nan	United Kingdom	24.78
676238	549524	22424	ENAMEL BREAD BIN CREAM	1	2011-08-04 15:42:00	24.96	nan	United Kingdom	24.96
676239	549524	22425	ENAMEL COLANDER CREAM	1	2011-08-04 15:42:00	9.96	nan	United Kingdom	9.96

Out[11]:

	Invoice	StockCode	Description	Quantity	InvoiceDate	Price	Customer_ID	Country	Tot_Rev
676234	549524	22359	GLASS JAR KINGS CHOICE	1	2011-08-04 15:42:00	5.79	nan	United Kingdom	5.79
676236	549524	22362	GLASS JAR PEACOCK BATH SALTS	1	2011-08-04 15:42:00	5.79	nan	United Kingdom	5.79
676237	549524	22411	JUMBO SHOPPER VINTAGE RED PAISLEY	6	2011-08-04 15:42:00	4.13	nan	United Kingdom	24.78
676238	549524	22424	ENAMEL BREAD BIN CREAM	1	2011-08-04 15:42:00	24.96	nan	United Kingdom	24.96
676239	549524	22425	ENAMEL COLANDER CREAM	1	2011-08-04 15:42:00	9.96	nan	United Kingdom	9.96

In [12]:
```
# In customer column their are un-related values need to removed
df.drop(df[df['Customer_ID'] == 'nan'].index, inplace = True)
df.head()
```

Out[12]:

	Invoice	StockCode	Description	Quantity	InvoiceDate	Price	Customer_ID	Country	Tot_Rev
199827	508581	20960	WATERMELON BATH SPONGE	20	2010-05-17 11:55:00	1.25	18287	United Kingdom	25.0
199825	508581	22284	HEN HOUSE DECORATION	12	2010-05-17 11:55:00	1.65	18287	United Kingdom	19.8
724195	554065	22064	PINK DOUGHNUT TRINKET POT	12	2011-05-22 10:39:00	1.65	18287	United Kingdom	19.8
199826	508581	22285	DECORATION HEN ON NEST, HANGING	12	2010-05-17 11:55:00	1.65	18287	United Kingdom	19.8
199832	508581	85041	SET/4 PINK ORCHID CANDLES IN BOWL	12	2010-05-17 11:55:00	1.65	18287	United Kingdom	19.8

Out[12]:

	Invoice	StockCode	Description	Quantity	InvoiceDate	Price	Customer_ID	Country	Tot_Rev
199827	508581	20960	WATERMELON BATH SPONGE	20	2010-05-17 11:55:00	1.25	18287	United Kingdom	25.0
199825	508581	22284	HEN HOUSE DECORATION	12	2010-05-17 11:55:00	1.65	18287	United Kingdom	19.8
724195	554065	22064	PINK DOUGHNUT TRINKET POT	12	2011-05-22 10:39:00	1.65	18287	United Kingdom	19.8
199826	508581	22285	DECORATION HEN ON NEST, HANGING	12	2010-05-17 11:55:00	1.65	18287	United Kingdom	19.8
199832	508581	85041	SET/4 PINK ORCHID CANDLES IN BOWL	12	2010-05-17 11:55:00	1.65	18287	United Kingdom	19.8

In [13]:
```
#create empty data frame
RFM = pd.DataFrame()
print(RFM)
```

```
Empty DataFrame
Columns: []
Index: []
Empty DataFrame
Columns: []
Index: []
```

In [14]:
```
# calculating no of time  purchase made by each customer
RFM['Freq'] = df.groupby("Customer_ID").StockCode.count()
RFM.head()
```

Out[14]:

Customer_ID	Freq
12346	48
12347	242
12348	51
12349	180

Customer_ID	
12350	17

Out[14]:

Customer_ID	Freq
12346	48
12347	242
12348	51
12349	180
12350	17

In [15]:
```
# calculating Total revenue purchased by each customer
RFM['Tot_Rev'] = df.groupby("Customer_ID").Tot_Rev.sum()
RFM.head()
```

Out[15]:

Customer_ID	Freq	Tot_Rev
12346	48	-64.68
12347	242	5408.50
12348	51	2019.40
12349	180	4404.54
12350	17	334.40

Out[15]:

Customer_ID	Freq	Tot_Rev
12346	48	-64.68
12347	242	5408.50
12348	51	2019.40
12349	180	4404.54
12350	17	334.40

In [16]:
```
# calculating Average revenue purchased by each customer
RFM['Avg_Rev'] = RFM['Tot_Rev']/RFM['Freq']
RFM.head()
```

Out[16]:

Customer_ID	Freq	Tot_Rev	Avg_Rev
12346	48	-64.68	-1.347500
12347	242	5408.50	22.349174
12348	51	2019.40	39.596078
12349	180	4404.54	24.469667
12350	17	334.40	19.670588

Out[16]:

Customer_ID	Freq	Tot_Rev	Avg_Rev

Customer_ID			
12346	48	-64.68	-1.347500
12347	242	5408.50	22.349174
12348	51	2019.40	39.596078
12349	180	4404.54	24.469667
12350	17	334.40	19.670588

In [17]:
```
#sorting by Customer_ID
RFM = RFM.sort_values(by="Customer_ID",ascending=0)
RFM.head()
```

Out[17]:

Customer_ID	Freq	Tot_Rev	Avg_Rev
18287	156	4177.89	26.781346
18286	70	1188.43	16.977571
18285	12	427.00	35.583333
18284	29	436.68	15.057931
18283	936	2528.65	2.701549

Out[17]:

Customer_ID	Freq	Tot_Rev	Avg_Rev
18287	156	4177.89	26.781346
18286	70	1188.43	16.977571
18285	12	427.00	35.583333
18284	29	436.68	15.057931
18283	936	2528.65	2.701549

In [18]:
```
RFM = RFM.dropna()
RFM.head()
```

Out[18]:

Customer_ID	Freq	Tot_Rev	Avg_Rev
18287	156	4177.89	26.781346
18286	70	1188.43	16.977571
18285	12	427.00	35.583333
18284	29	436.68	15.057931
18283	936	2528.65	2.701549

Out[18]:

Customer_ID	Freq	Tot_Rev	Avg_Rev
18287	156	4177.89	26.781346
18286	70	1188.43	16.977571

18285	12	427.00	35.583333
18284	29	436.68	15.057931
18283	936	2528.65	2.701549

In [19]:
```
#Creating the column called order_date(which will taken has todays date)
RFM['order_date'] = '1.1.2012 00:00'
```

In [20]:
```
# change datatype of order_date column
RFM['order_date']  = pd.to_datetime(RFM['order_date'] , errors='coerce')
RFM.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 5924 entries, 18287 to 12346
Data columns (total 4 columns):
 #   Column      Non-Null Count  Dtype         
---  ------      --------------  -----         
 0   Freq        5924 non-null   int64         
 1   Tot_Rev     5924 non-null   float64       
 2   Avg_Rev     5924 non-null   float64       
 3   order_date  5924 non-null   datetime64[ns]
dtypes: datetime64[ns](1), float64(2), int64(1)
memory usage: 231.4+ KB
<class 'pandas.core.frame.DataFrame'>
Index: 5924 entries, 18287 to 12346
Data columns (total 4 columns):
 #   Column      Non-Null Count  Dtype         
---  ------      --------------  -----         
 0   Freq        5924 non-null   int64         
 1   Tot_Rev     5924 non-null   float64       
 2   Avg_Rev     5924 non-null   float64       
 3   order_date  5924 non-null   datetime64[ns]
dtypes: datetime64[ns](1), float64(2), int64(1)
memory usage: 231.4+ KB
```

In [21]:
```
# calulating recent date purchansed by each cust
RFM['MAX_InvoiceDate'] = df.groupby("Customer_ID").InvoiceDate.max()
RFM.head()
```

Out[21]:

Customer_ID	Freq	Tot_Rev	Avg_Rev	order_date	MAX_InvoiceDate
18287	156	4177.89	26.781346	2012-01-01	2011-12-10 10:23:00
18286	70	1188.43	16.977571	2012-01-01	2010-08-20 11:57:00
18285	12	427.00	35.583333	2012-01-01	2010-02-17 10:24:00
18284	29	436.68	15.057931	2012-01-01	2010-06-10 12:31:00
18283	936	2528.65	2.701549	2012-01-01	2011-11-30 12:59:00

Out[21]:

Customer_ID	Freq	Tot_Rev	Avg_Rev	order_date	MAX_InvoiceDate
18287	156	4177.89	26.781346	2012-01-01	2011-12-10 10:23:00
18286	70	1188.43	16.977571	2012-01-01	2010-08-20 11:57:00
18285	12	427.00	35.583333	2012-01-01	2010-02-17 10:24:00
18284	29	436.68	15.057931	2012-01-01	2010-06-10 12:31:00

18283	936	2528.65	2.701549	2012-01-01	2011-11-30 12:59:00

In [22]:
```
# calulating duration from last purchased
RFM['duration'] =  RFM['order_date'] - RFM['MAX_InvoiceDate']
RFM.head()
```

Out[22]:

Customer_ID	Freq	Tot_Rev	Avg_Rev	order_date	MAX_InvoiceDate	duration
18287	156	4177.89	26.781346	2012-01-01	2011-12-10 10:23:00	21 days 13:37:00
18286	70	1188.43	16.977571	2012-01-01	2010-08-20 11:57:00	498 days 12:03:00
18285	12	427.00	35.583333	2012-01-01	2010-02-17 10:24:00	682 days 13:36:00
18284	29	436.68	15.057931	2012-01-01	2010-06-10 12:31:00	569 days 11:29:00
18283	936	2528.65	2.701549	2012-01-01	2011-11-30 12:59:00	31 days 11:01:00

Out[22]:

Customer_ID	Freq	Tot_Rev	Avg_Rev	order_date	MAX_InvoiceDate	duration
18287	156	4177.89	26.781346	2012-01-01	2011-12-10 10:23:00	21 days 13:37:00
18286	70	1188.43	16.977571	2012-01-01	2010-08-20 11:57:00	498 days 12:03:00
18285	12	427.00	35.583333	2012-01-01	2010-02-17 10:24:00	682 days 13:36:00
18284	29	436.68	15.057931	2012-01-01	2010-06-10 12:31:00	569 days 11:29:00
18283	936	2528.65	2.701549	2012-01-01	2011-11-30 12:59:00	31 days 11:01:00

In [23]:
```
RFM = RFM.drop(['order_date','MAX_InvoiceDate'], axis=1)
RFM.head()
```

Out[23]:

Customer_ID	Freq	Tot_Rev	Avg_Rev	duration
18287	156	4177.89	26.781346	21 days 13:37:00
18286	70	1188.43	16.977571	498 days 12:03:00
18285	12	427.00	35.583333	682 days 13:36:00
18284	29	436.68	15.057931	569 days 11:29:00
18283	936	2528.65	2.701549	31 days 11:01:00

Out[23]:

Customer_ID	Freq	Tot_Rev	Avg_Rev	duration
18287	156	4177.89	26.781346	21 days 13:37:00
18286	70	1188.43	16.977571	498 days 12:03:00
18285	12	427.00	35.583333	682 days 13:36:00
18284	29	436.68	15.057931	569 days 11:29:00
18283	936	2528.65	2.701549	31 days 11:01:00

In [24]:
```
# this is RFM (Recency Frequency Monetary) table for all customers
RFM = RFM.reset_index()
```

```
RFM.head()
```

Out[24]:

	Customer_ID	Freq	Tot_Rev	Avg_Rev	duration
0	18287	156	4177.89	26.781346	21 days 13:37:00
1	18286	70	1188.43	16.977571	498 days 12:03:00
2	18285	12	427.00	35.583333	682 days 13:36:00
3	18284	29	436.68	15.057931	569 days 11:29:00
4	18283	936	2528.65	2.701549	31 days 11:01:00

Out[24]:

	Customer_ID	Freq	Tot_Rev	Avg_Rev	duration
0	18287	156	4177.89	26.781346	21 days 13:37:00
1	18286	70	1188.43	16.977571	498 days 12:03:00
2	18285	12	427.00	35.583333	682 days 13:36:00
3	18284	29	436.68	15.057931	569 days 11:29:00
4	18283	936	2528.65	2.701549	31 days 11:01:00

In [25]:
```
#creating 10 different buckets for duration (Deciles)
RFM['Recency_Deciles'] = pd.qcut(RFM['duration'],10)
RFM.head()
```

Out[25]:

	Customer_ID	Freq	Tot_Rev	Avg_Rev	duration	Recency_Deciles
0	18287	156	4177.89	26.781346	21 days 13:37:00	(21 days 06:40:59.999999999, 32 days 12:04:36]
1	18286	70	1188.43	16.977571	498 days 12:03:00	(436 days 12:19:24, 557 days 12:04:00]
2	18285	12	427.00	35.583333	682 days 13:36:00	(557 days 12:04:00, 1083 days 14:05:00]
3	18284	29	436.68	15.057931	569 days 11:29:00	(557 days 12:04:00, 1083 days 14:05:00]
4	18283	936	2528.65	2.701549	31 days 11:01:00	(21 days 06:40:59.999999999, 32 days 12:04:36]

Out[25]:

	Customer_ID	Freq	Tot_Rev	Avg_Rev	duration	Recency_Deciles
0	18287	156	4177.89	26.781346	21 days 13:37:00	(21 days 06:40:59.999999999, 32 days 12:04:36]
1	18286	70	1188.43	16.977571	498 days 12:03:00	(436 days 12:19:24, 557 days 12:04:00]
2	18285	12	427.00	35.583333	682 days 13:36:00	(557 days 12:04:00, 1083 days 14:05:00]
3	18284	29	436.68	15.057931	569 days 11:29:00	(557 days 12:04:00, 1083 days 14:05:00]
4	18283	936	2528.65	2.701549	31 days 11:01:00	(21 days 06:40:59.999999999, 32 days 12:04:36]

In [26]:
```
#create empty dataframe called vintage
vin = pd.DataFrame()
print(vin)
```

```
Empty DataFrame
Columns: []
Index: []
Empty DataFrame
Columns: []
Index: []
```

In [27]:
```
# calulating total revenue for each bucket
vin["Tot_rev"] = RFM.groupby('Recency_Deciles').Tot_Rev.sum()
```

```
vin.head(10)
```

Out[27]:

Recency_Deciles	Tot_rev
(21 days 06:40:59.999999999, 32 days 12:04:36]	5711857.734
(32 days 12:04:36, 43 days 13:59:12]	2968252.921
(43 days 13:59:12, 61 days 10:34:24.000000001]	2275605.871
(61 days 10:34:24.000000001, 87 days 18:32:36.000000017]	1533952.931
(87 days 18:32:36.000000017, 135 days 08:26:00]	1262303.344
(135 days 08:26:00, 216 days 09:30:36.000000004]	965468.294
(216 days 09:30:36.000000004, 345 days 09:39:30]	586407.400
(345 days 09:39:30, 436 days 12:19:24]	536943.362
(436 days 12:19:24, 557 days 12:04:00]	398391.840
(557 days 12:04:00, 1083 days 14:05:00]	164913.601

Out[27]:

Recency_Deciles	Tot_rev
(21 days 06:40:59.999999999, 32 days 12:04:36]	5711857.734
(32 days 12:04:36, 43 days 13:59:12]	2968252.921
(43 days 13:59:12, 61 days 10:34:24.000000001]	2275605.871
(61 days 10:34:24.000000001, 87 days 18:32:36.000000017]	1533952.931
(87 days 18:32:36.000000017, 135 days 08:26:00]	1262303.344
(135 days 08:26:00, 216 days 09:30:36.000000004]	965468.294
(216 days 09:30:36.000000004, 345 days 09:39:30]	586407.400
(345 days 09:39:30, 436 days 12:19:24]	536943.362
(436 days 12:19:24, 557 days 12:04:00]	398391.840
(557 days 12:04:00, 1083 days 14:05:00]	164913.601

In [28]:
```
# calculating cumulative revenue
vin["CumSum_Rev"] = vin["Tot_rev"].cumsum()
vin.head(10)
```

Out[28]:

Recency_Deciles	Tot_rev	CumSum_Rev
(21 days 06:40:59.999999999, 32 days 12:04:36]	5711857.734	5.711858e+06
(32 days 12:04:36, 43 days 13:59:12]	2968252.921	8.680111e+06
(43 days 13:59:12, 61 days 10:34:24.000000001]	2275605.871	1.095572e+07
(61 days 10:34:24.000000001, 87 days 18:32:36.000000017]	1533952.931	1.248967e+07
(87 days 18:32:36.000000017, 135 days 08:26:00]	1262303.344	1.375197e+07
(135 days 08:26:00, 216 days 09:30:36.000000004]	965468.294	1.471744e+07
(216 days 09:30:36.000000004, 345 days 09:39:30]	586407.400	1.530385e+07

Recency_Deciles	Tot_rev	CumSum_Rev
(345 days 09:39:30, 436 days 12:19:24]	536943.362	1.584079e+07
(436 days 12:19:24, 557 days 12:04:00]	398391.840	1.623918e+07
(557 days 12:04:00, 1083 days 14:05:00]	164913.601	1.640410e+07

Out[28]:

Recency_Deciles	Tot_rev	CumSum_Rev
(21 days 06:40:59.999999999, 32 days 12:04:36]	5711857.734	5.711858e+06
(32 days 12:04:36, 43 days 13:59:12]	2968252.921	8.680111e+06
(43 days 13:59:12, 61 days 10:34:24.000000001]	2275605.871	1.095572e+07
(61 days 10:34:24.000000001, 87 days 18:32:36.000000017]	1533952.931	1.248967e+07
(87 days 18:32:36.000000017, 135 days 08:26:00]	1262303.344	1.375197e+07
(135 days 08:26:00, 216 days 09:30:36.000000004]	965468.294	1.471744e+07
(216 days 09:30:36.000000004, 345 days 09:39:30]	586407.400	1.530385e+07
(345 days 09:39:30, 436 days 12:19:24]	536943.362	1.584079e+07
(436 days 12:19:24, 557 days 12:04:00]	398391.840	1.623918e+07
(557 days 12:04:00, 1083 days 14:05:00]	164913.601	1.640410e+07

In [29]:
```
# calculating total revenue
vin["Tot_rev_Across_Deciles"] = RFM.Tot_Rev.sum()
vin.head(10)
```

Out[29]:

Recency_Deciles	Tot_rev	CumSum_Rev	Tot_rev_Across_Deciles
(21 days 06:40:59.999999999, 32 days 12:04:36]	5711857.734	5.711858e+06	1.640410e+07
(32 days 12:04:36, 43 days 13:59:12]	2968252.921	8.680111e+06	1.640410e+07
(43 days 13:59:12, 61 days 10:34:24.000000001]	2275605.871	1.095572e+07	1.640410e+07
(61 days 10:34:24.000000001, 87 days 18:32:36.000000017]	1533952.931	1.248967e+07	1.640410e+07
(87 days 18:32:36.000000017, 135 days 08:26:00]	1262303.344	1.375197e+07	1.640410e+07
(135 days 08:26:00, 216 days 09:30:36.000000004]	965468.294	1.471744e+07	1.640410e+07
(216 days 09:30:36.000000004, 345 days 09:39:30]	586407.400	1.530385e+07	1.640410e+07
(345 days 09:39:30, 436 days 12:19:24]	536943.362	1.584079e+07	1.640410e+07
(436 days 12:19:24, 557 days 12:04:00]	398391.840	1.623918e+07	1.640410e+07
(557 days 12:04:00, 1083 days 14:05:00]	164913.601	1.640410e+07	1.640410e+07

Out[29]:

Recency_Deciles	Tot_rev	CumSum_Rev	Tot_rev_Across_Deciles
(21 days 06:40:59.999999999, 32 days 12:04:36]	5711857.734	5.711858e+06	1.640410e+07
(32 days 12:04:36, 43 days 13:59:12]	2968252.921	8.680111e+06	1.640410e+07
(43 days 13:59:12, 61 days 10:34:24.000000001]	2275605.871	1.095572e+07	1.640410e+07
(61 days 10:34:24.000000001, 87 days 18:32:36.000000017]	1533952.931	1.248967e+07	1.640410e+07

(87 days 18:32:36.000000017, 135 days 08:26:00]	1262303.344	1.375197e+07	1.640410e+07
(135 days 08:26:00, 216 days 09:30:36.000000004]	965468.294	1.471744e+07	1.640410e+07
(216 days 09:30:36.000000004, 345 days 09:39:30]	586407.400	1.530385e+07	1.640410e+07
(345 days 09:39:30, 436 days 12:19:24]	536943.362	1.584079e+07	1.640410e+07
(436 days 12:19:24, 557 days 12:04:00]	398391.840	1.623918e+07	1.640410e+07
(557 days 12:04:00, 1083 days 14:05:00]	164913.601	1.640410e+07	1.640410e+07

In [30]:
```
# calculating percentage cumilative revenue with respect to total revenue for all the cu
vin["perc_tot_rev"] = vin["CumSum_Rev"]/vin["Tot_rev_Across_Deciles"]
vin.head(10)
```

Out[30]:

Recency_Deciles	Tot_rev	CumSum_Rev	Tot_rev_Across_Deciles	perc_tot_rev
(21 days 06:40:59.999999999, 32 days 12:04:36]	5711857.734	5.711858e+06	1.640410e+07	0.348197
(32 days 12:04:36, 43 days 13:59:12]	2968252.921	8.680111e+06	1.640410e+07	0.529143
(43 days 13:59:12, 61 days 10:34:24.000000001]	2275605.871	1.095572e+07	1.640410e+07	0.667865
(61 days 10:34:24.000000001, 87 days 18:32:36.000000017]	1533952.931	1.248967e+07	1.640410e+07	0.761375
(87 days 18:32:36.000000017, 135 days 08:26:00]	1262303.344	1.375197e+07	1.640410e+07	0.838325
(135 days 08:26:00, 216 days 09:30:36.000000004]	965468.294	1.471744e+07	1.640410e+07	0.897181
(216 days 09:30:36.000000004, 345 days 09:39:30]	586407.400	1.530385e+07	1.640410e+07	0.932928
(345 days 09:39:30, 436 days 12:19:24]	536943.362	1.584079e+07	1.640410e+07	0.965661
(436 days 12:19:24, 557 days 12:04:00]	398391.840	1.623918e+07	1.640410e+07	0.989947
(557 days 12:04:00, 1083 days 14:05:00]	164913.601	1.640410e+07	1.640410e+07	1.000000

Out[30]:

Recency_Deciles	Tot_rev	CumSum_Rev	Tot_rev_Across_Deciles	perc_tot_rev
(21 days 06:40:59.999999999, 32 days 12:04:36]	5711857.734	5.711858e+06	1.640410e+07	0.348197
(32 days 12:04:36, 43 days 13:59:12]	2968252.921	8.680111e+06	1.640410e+07	0.529143
(43 days 13:59:12, 61 days 10:34:24.000000001]	2275605.871	1.095572e+07	1.640410e+07	0.667865
(61 days 10:34:24.000000001, 87 days 18:32:36.000000017]	1533952.931	1.248967e+07	1.640410e+07	0.761375
(87 days 18:32:36.000000017, 135 days 08:26:00]	1262303.344	1.375197e+07	1.640410e+07	0.838325
(135 days 08:26:00, 216 days 09:30:36.000000004]	965468.294	1.471744e+07	1.640410e+07	0.897181
(216 days 09:30:36.000000004, 345 days 09:39:30]	586407.400	1.530385e+07	1.640410e+07	0.932928
(345 days 09:39:30, 436 days 12:19:24]	536943.362	1.584079e+07	1.640410e+07	0.965661
(436 days 12:19:24, 557 days 12:04:00]	398391.840	1.623918e+07	1.640410e+07	0.989947

(557 days 12:04:00, 1083 days 14:05:00]	164913.601	1.640410e+07	1.640410e+07	1.000000

In [31]:
```
#creating 10 different buckets wr.t frequency column
RFM['Feq_Deciles'] = pd.qcut(RFM['Freq'],10)
RFM.head()
```

Out[31]:

	Customer_ID	Freq	Tot_Rev	Avg_Rev	duration	Recency_Deciles	Feq_Deciles
0	18287	156	4177.89	26.781346	21 days 13:37:00	(21 days 06:40:59.999999999, 32 days 12:04:36]	(114.0, 180.0]
1	18286	70	1188.43	16.977571	498 days 12:03:00	(436 days 12:19:24, 557 days 12:04:00]	(53.0, 76.0]
2	18285	12	427.00	35.583333	682 days 13:36:00	(557 days 12:04:00, 1083 days 14:05:00]	(8.0, 16.0]
3	18284	29	436.68	15.057931	569 days 11:29:00	(557 days 12:04:00, 1083 days 14:05:00]	(25.0, 37.0]
4	18283	936	2528.65	2.701549	31 days 11:01:00	(21 days 06:40:59.999999999, 32 days 12:04:36]	(319.7, 12780.0]

Out[31]:

	Customer_ID	Freq	Tot_Rev	Avg_Rev	duration	Recency_Deciles	Feq_Deciles
0	18287	156	4177.89	26.781346	21 days 13:37:00	(21 days 06:40:59.999999999, 32 days 12:04:36]	(114.0, 180.0]
1	18286	70	1188.43	16.977571	498 days 12:03:00	(436 days 12:19:24, 557 days 12:04:00]	(53.0, 76.0]
2	18285	12	427.00	35.583333	682 days 13:36:00	(557 days 12:04:00, 1083 days 14:05:00]	(8.0, 16.0]
3	18284	29	436.68	15.057931	569 days 11:29:00	(557 days 12:04:00, 1083 days 14:05:00]	(25.0, 37.0]
4	18283	936	2528.65	2.701549	31 days 11:01:00	(21 days 06:40:59.999999999, 32 days 12:04:36]	(319.7, 12780.0]

In [32]:
```
# create empty dataframe
rec = pd.DataFrame()
print(rec)
```

```
Empty DataFrame
Columns: []
Index: []
Empty DataFrame
Columns: []
Index: []
```

In [33]:
```
# calculating total revenue for each bucket
rec["Tot_rev"] = RFM.groupby('Feq_Deciles').Tot_Rev.sum()
rec.head(10)
```

Out[33]:

Feq_Deciles	Tot_rev
(0.999, 8.0]	72193.160
(8.0, 16.0]	242098.130
(16.0, 25.0]	274574.731
(25.0, 37.0]	383583.741

Feq_Deciles	
(37.0, 53.0]	603851.613
(53.0, 76.0]	773353.640
(76.0, 114.0]	1070299.192
(114.0, 180.0]	1630626.023
(180.0, 319.7]	2761936.172
(319.7, 12780.0]	8591580.896

Out[33]:

Feq_Deciles	Tot_rev
(0.999, 8.0]	72193.160
(8.0, 16.0]	242098.130
(16.0, 25.0]	274574.731
(25.0, 37.0]	383583.741
(37.0, 53.0]	603851.613
(53.0, 76.0]	773353.640
(76.0, 114.0]	1070299.192
(114.0, 180.0]	1630626.023
(180.0, 319.7]	2761936.172
(319.7, 12780.0]	8591580.896

In [34]:
```
# calculating cumulative revenue
rec["CumSum_Rev"] = rec["Tot_rev"].cumsum()
rec.head(10)
```

Out[34]:

Feq_Deciles	Tot_rev	CumSum_Rev
(0.999, 8.0]	72193.160	7.219316e+04
(8.0, 16.0]	242098.130	3.142913e+05
(16.0, 25.0]	274574.731	5.888660e+05
(25.0, 37.0]	383583.741	9.724498e+05
(37.0, 53.0]	603851.613	1.576301e+06
(53.0, 76.0]	773353.640	2.349655e+06
(76.0, 114.0]	1070299.192	3.419954e+06
(114.0, 180.0]	1630626.023	5.050580e+06
(180.0, 319.7]	2761936.172	7.812516e+06
(319.7, 12780.0]	8591580.896	1.640410e+07

Out[34]:

Feq_Deciles	Tot_rev	CumSum_Rev
(0.999, 8.0]	72193.160	7.219316e+04
(8.0, 16.0]	242098.130	3.142913e+05

(16.0, 25.0]	274574.731	5.888660e+05
(25.0, 37.0]	383583.741	9.724498e+05
(37.0, 53.0]	603851.613	1.576301e+06
(53.0, 76.0]	773353.640	2.349655e+06
(76.0, 114.0]	1070299.192	3.419954e+06
(114.0, 180.0]	1630626.023	5.050580e+06
(180.0, 319.7]	2761936.172	7.812516e+06
(319.7, 12780.0]	8591580.896	1.640410e+07

In [35]:
```
# calculating total revenue for each bucket
rec["Tot_rev_Across_Deciles"] = RFM.Tot_Rev.sum()
rec.head(10)
```

Out[35]:

Feq_Deciles	Tot_rev	CumSum_Rev	Tot_rev_Across_Deciles
(0.999, 8.0]	72193.160	7.219316e+04	1.640410e+07
(8.0, 16.0]	242098.130	3.142913e+05	1.640410e+07
(16.0, 25.0]	274574.731	5.888660e+05	1.640410e+07
(25.0, 37.0]	383583.741	9.724498e+05	1.640410e+07
(37.0, 53.0]	603851.613	1.576301e+06	1.640410e+07
(53.0, 76.0]	773353.640	2.349655e+06	1.640410e+07
(76.0, 114.0]	1070299.192	3.419954e+06	1.640410e+07
(114.0, 180.0]	1630626.023	5.050580e+06	1.640410e+07
(180.0, 319.7]	2761936.172	7.812516e+06	1.640410e+07
(319.7, 12780.0]	8591580.896	1.640410e+07	1.640410e+07

Out[35]:

Feq_Deciles	Tot_rev	CumSum_Rev	Tot_rev_Across_Deciles
(0.999, 8.0]	72193.160	7.219316e+04	1.640410e+07
(8.0, 16.0]	242098.130	3.142913e+05	1.640410e+07
(16.0, 25.0]	274574.731	5.888660e+05	1.640410e+07
(25.0, 37.0]	383583.741	9.724498e+05	1.640410e+07
(37.0, 53.0]	603851.613	1.576301e+06	1.640410e+07
(53.0, 76.0]	773353.640	2.349655e+06	1.640410e+07
(76.0, 114.0]	1070299.192	3.419954e+06	1.640410e+07
(114.0, 180.0]	1630626.023	5.050580e+06	1.640410e+07
(180.0, 319.7]	2761936.172	7.812516e+06	1.640410e+07
(319.7, 12780.0]	8591580.896	1.640410e+07	1.640410e+07

In [36]:
```
#calculating percentage total revenue with respect to total revenue of all the customers
```

```
rec["perc_tot_rev"] = rec["CumSum_Rev"]/rec["Tot_rev_Across_Deciles"]
vin.head(10)
```

Out[36]:

Recency_Deciles	Tot_rev	CumSum_Rev	Tot_rev_Across_Deciles	perc_tot_rev
(21 days 06:40:59.999999999, 32 days 12:04:36]	5711857.734	5.711858e+06	1.640410e+07	0.348197
(32 days 12:04:36, 43 days 13:59:12]	2968252.921	8.680111e+06	1.640410e+07	0.529143
(43 days 13:59:12, 61 days 10:34:24.000000001]	2275605.871	1.095572e+07	1.640410e+07	0.667865
(61 days 10:34:24.000000001, 87 days 18:32:36.000000017]	1533952.931	1.248967e+07	1.640410e+07	0.761375
(87 days 18:32:36.000000017, 135 days 08:26:00]	1262303.344	1.375197e+07	1.640410e+07	0.838325
(135 days 08:26:00, 216 days 09:30:36.000000004]	965468.294	1.471744e+07	1.640410e+07	0.897181
(216 days 09:30:36.000000004, 345 days 09:39:30]	586407.400	1.530385e+07	1.640410e+07	0.932928
(345 days 09:39:30, 436 days 12:19:24]	536943.362	1.584079e+07	1.640410e+07	0.965661
(436 days 12:19:24, 557 days 12:04:00]	398391.840	1.623918e+07	1.640410e+07	0.989947
(557 days 12:04:00, 1083 days 14:05:00]	164913.601	1.640410e+07	1.640410e+07	1.000000

Out[36]:

Recency_Deciles	Tot_rev	CumSum_Rev	Tot_rev_Across_Deciles	perc_tot_rev
(21 days 06:40:59.999999999, 32 days 12:04:36]	5711857.734	5.711858e+06	1.640410e+07	0.348197
(32 days 12:04:36, 43 days 13:59:12]	2968252.921	8.680111e+06	1.640410e+07	0.529143
(43 days 13:59:12, 61 days 10:34:24.000000001]	2275605.871	1.095572e+07	1.640410e+07	0.667865
(61 days 10:34:24.000000001, 87 days 18:32:36.000000017]	1533952.931	1.248967e+07	1.640410e+07	0.761375
(87 days 18:32:36.000000017, 135 days 08:26:00]	1262303.344	1.375197e+07	1.640410e+07	0.838325
(135 days 08:26:00, 216 days 09:30:36.000000004]	965468.294	1.471744e+07	1.640410e+07	0.897181
(216 days 09:30:36.000000004, 345 days 09:39:30]	586407.400	1.530385e+07	1.640410e+07	0.932928
(345 days 09:39:30, 436 days 12:19:24]	536943.362	1.584079e+07	1.640410e+07	0.965661
(436 days 12:19:24, 557 days 12:04:00]	398391.840	1.623918e+07	1.640410e+07	0.989947
(557 days 12:04:00, 1083 days 14:05:00]	164913.601	1.640410e+07	1.640410e+07	1.000000

Always open for feedback and suggestions.If it helps Thumbs Up !!!