In [40]:
```
from datetime import datetime, timedelta
```

In [41]:
```
import pandas as pd
```

In [42]:
```
%matplotlib inline
import matplotlib.pyplot as plt
```

In [43]:
```
import numpy as np
```

In [18]:
```
import seaborn as sns
```

In [19]:
```
from __future__ import division
```

In [20]:
```
import plotly.plotly as py
```

In [21]:
```
import plotly.offline as pyoff
```

In [22]:
```
import plotly.graph_objs as go
```

In [23]:
```
datasource=r'C:\Users\Arushi Mathur\Desktop\sales_data.xlsx'
```

In [27]:
```
df=pd.read_excel(datasource)
```

In [44]:
```
df.head()
```

Out[44]:

	transaction id	product id	product description	quantity sold	transaction timestamp	unit price	customer id	transaction country	YearMor
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom	2010
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	2010
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom	2010
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	2010
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	2010

In [45]:
```
# MONTHLY REVENUE
```

In [46]:
```
df['transaction timestamp']=pd.to_datetime(df['transaction timestamp'])
```

In [47]:
```
df['YearMonth'] = df['transaction timestamp'].map(lambda date: 100*date.year +
date.month)
```

In [48]:
```
df['Revenue']=df['quantity sold']*df['unit price']
```

In [49]:
```
df_revenue = df.groupby(['YearMonth'])['Revenue'].sum().reset_index()
```

In [50]:
```
df_revenue
```

Out[50]:

	YearMonth	Revenue
0	201012	748957.020
1	201101	560000.260
2	201102	498062.650
3	201103	683267.080
4	201104	493207.121
5	201105	723333.510
6	201106	691123.120
7	201107	681300.111
8	201108	682680.510
9	201109	1019687.622
10	201110	1070704.670
11	201111	1461756.250
12	201112	433668.010

In [51]:
```
plot_data=[go.Scatter(
x=df_revenue['YearMonth'],
y=df_revenue['Revenue'])]
```

In [52]:
```
plot_layout = go.Layout(
        xaxis={"type": "category"},
        title='Montly Revenue'
    )
```

In [53]:
```
fig1 = go.Figure(data=plot_data, layout=plot_layout)
```

In [54]:
```
pyoff.plot(fig1)
```

Out[54]:
```
'temp-plot.html'
```

In [23]:
```
# TOTAL MONTH GROWTH RATE
```

In [55]:
```
df_revenue['MonthGrowth']=df_revenue['Revenue'].pct_change()
```

In [56]:
```
df_revenue.head()
```

Out[56]:

	YearMonth	Revenue	MonthGrowth
0	201012	748957.020	NaN
1	201101	560000.260	-0.252293
2	201102	498062.650	-0.110603
3	201103	683267.080	0.371850
4	201104	493207.121	-0.278163

In [57]:
```
plot_data = [
    go.Scatter(
        x=df_revenue.query("YearMonth < 201112")['YearMonth'],
        y=df_revenue.query("YearMonth < 201112")['MonthGrowth'],
    )
]
```

In [58]:
```
plot_layout = go.Layout(
        xaxis={"type": "category"},
        title='Total Month Growth Rate'
    )
```

In [59]:
```
fig = go.Figure(data=plot_data, layout=plot_layout)
```

In [60]:
```
pyoff.plot(fig)
```

Out[60]:
```
'temp-plot.html'
```

In [61]:
```
# MONTHLY ACTIVE CUSTOMERS
```

In [62]:
```
df.head()
```

Out[62]:

	transaction id	product id	product description	quantity sold	transaction timestamp	unit price	customer id	transaction country	YearMor
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom	2010
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	2010
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom	2010
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	2010
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	2010

In [63]:
```
dfcus=df.groupby(['YearMonth'])['customer id'].nunique().reset_index()
```

In [64]:
```
dfcus
```

Out[64]:

	YearMonth	customer id
0	201012	948
1	201101	783
2	201102	798
3	201103	1020
4	201104	899
5	201105	1079
6	201106	1051
7	201107	993
8	201108	980
9	201109	1302
10	201110	1425
11	201111	1711
12	201112	686

In [65]:
```
plot_data=[go.Bar(
x=dfcus['YearMonth'],
    y=dfcus['customer id'],
)]
```

In [66]:
```
plot_layout=go.Layout(
xaxis={"type":"category"},
    title='Active Customers'
)
```

In [67]:
```
fig=go.Figure(data=plot_data,layout=plot_layout)
```

In [68]:
```
pyoff.plot(fig)
```

Out[68]:
```
'temp-plot.html'
```

In [69]:
```
# Monthly Order Count
```

In [70]:
```
# counting orders in a particular month
dfordercount=df.groupby(['YearMonth'])['quantity sold'].sum().reset_index()
```

In [71]:
```
dfordercount
```

Out[71]:

	YearMonth	quantity sold
0	201012	342228
1	201101	308966
2	201102	277989
3	201103	351872
4	201104	289098
5	201105	380391
6	201106	341623
7	201107	391116
8	201108	406199
9	201109	549817
10	201110	570532
11	201111	740286
12	201112	226333

In [72]:
```
plot_data=[go.Bar(
    x=dfordercount['YearMonth'],
    y=dfordercount['quantity sold'],
)]
```

In [73]:
```
plot_layout=go.Layout(
xaxis={"type":"category"},
    title='Order Count'
)
```

In [74]:
```
fig=go.Figure(data=plot_data,layout=plot_layout)
```

In [75]:
```
pyoff.plot(fig)
```

Out[75]:
```
'temp-plot.html'
```

In [76]:
```
# Finding Average Revenue Per Order
dfavgrev=df.groupby(['YearMonth'])['Revenue'].mean().reset_index()
```

In [77]:
```
dfavgrev
```

Out[77]:

	YearMonth	Revenue
0	201012	17.630400
1	201101	15.933088
2	201102	17.976058
3	201103	18.593313
4	201104	16.486399
5	201105	19.533716
6	201106	18.742830
7	201107	17.240248
8	201108	19.348161
9	201109	20.301987
10	201110	17.627089
11	201111	17.255802
12	201112	16.989932

In [78]:
```
plot_data=[go.Bar(
x=dfavgrev['YearMonth'],
    y=dfavgrev['Revenue'],
)]
```

In [79]:
```
plot_layout=go.Layout(
xaxis={"type":"category"},
    title='Average Revenue per order'
)
```

In [80]:
```
fig=go.Figure(data=plot_data,layout=plot_layout)
```

In [81]:
```
pyoff.plot(fig)
```
Out[81]:
```
'temp-plot.html'
```

In [82]:
```
# New Customer Ratio
```

In [83]:
```
dfminpurchase=df.groupby(['customer id'])['transaction timestamp'].min().reset
_index()
```

In [84]:
```
dfminpurchase.columns=['customer id','MinPurchaseDate']
```

In [85]:
```
dfminpurchase['MinPurchaseYearMonth']=dfminpurchase['MinPurchaseDate'].map(lam
bda date: 100*date.year + date.month)
```

In [86]:
```
df=pd.merge(df,dfminpurchase,on='customer id')
```

In [87]:
```
df.head()
```
Out[87]:

	transaction id	product id	product description	quantity sold	transaction timestamp	unit price	customer id	transaction country	YearMor
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom	2010
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	2010
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom	2010
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	2010
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	2010

In [88]:
```
# create a column caLLED uSER type and assign Existing , if user's first purch
ase year month before the selected Invoice Year Month
df['UserType']='New'
```

In [89]:
```
df.loc[df['YearMonth']>df['MinPurchaseYearMonth'],'UserType']='Existing'
```

In [90]:
```
df
```

Out[90]:

	transaction id	product id	product description	quantity sold	transaction timestamp	unit price	customer id	transaction country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
5	536365	22752	SET 7 BABUSHKA NESTING BOXES	2	2010-12-01 08:26:00	7.65	17850.0	United Kingdom
6	536365	21730	GLASS STAR FROSTED T-LIGHT HOLDER	6	2010-12-01 08:26:00	4.25	17850.0	United Kingdom
7	536366	22633	HAND WARMER UNION JACK	6	2010-12-01 08:28:00	1.85	17850.0	United Kingdom
8	536366	22632	HAND WARMER RED POLKA DOT	6	2010-12-01 08:28:00	1.85	17850.0	United Kingdom
9	536372	22632	HAND WARMER RED POLKA DOT	6	2010-12-01 09:01:00	1.85	17850.0	United Kingdom
10	536372	22633	HAND WARMER UNION JACK	6	2010-12-01 09:01:00	1.85	17850.0	United Kingdom
11	536373	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 09:02:00	2.55	17850.0	United Kingdom
12	536373	71053	WHITE METAL LANTERN	6	2010-12-01 09:02:00	3.39	17850.0	United Kingdom
13	536373	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 09:02:00	2.75	17850.0	United Kingdom
14	536373	20679	EDWARDIAN PARASOL RED	6	2010-12-01 09:02:00	4.95	17850.0	United Kingdom
15	536373	37370	RETRO COFFEE MUGS ASSORTED	6	2010-12-01 09:02:00	1.06	17850.0	United Kingdom
16	536373	21871	SAVE THE PLANET MUG	6	2010-12-01 09:02:00	1.06	17850.0	United Kingdom

	transaction id	product id	product description	quantity sold	transaction timestamp	unit price	customer id	transaction country
17	536373	21071	VINTAGE BILLBOARD DRINK ME MUG	6	2010-12-01 09:02:00	1.06	17850.0	United Kingdom
18	536373	21068	VINTAGE BILLBOARD LOVE/HATE MUG	6	2010-12-01 09:02:00	1.06	17850.0	United Kingdom
19	536373	82483	WOOD 2 DRAWER CABINET WHITE FINISH	2	2010-12-01 09:02:00	4.95	17850.0	United Kingdom
20	536373	82486	WOOD S/3 CABINET ANT WHITE FINISH	4	2010-12-01 09:02:00	6.95	17850.0	United Kingdom
21	536373	82482	WOODEN PICTURE FRAME WHITE FINISH	6	2010-12-01 09:02:00	2.10	17850.0	United Kingdom
22	536373	82494L	WOODEN FRAME ANTIQUE WHITE	6	2010-12-01 09:02:00	2.55	17850.0	United Kingdom
23	536373	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 09:02:00	3.39	17850.0	United Kingdom
24	536373	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 09:02:00	3.39	17850.0	United Kingdom
25	536373	22752	SET 7 BABUSHKA NESTING BOXES	2	2010-12-01 09:02:00	7.65	17850.0	United Kingdom
26	536373	21730	GLASS STAR FROSTED T-LIGHT HOLDER	6	2010-12-01 09:02:00	4.25	17850.0	United Kingdom
27	536375	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 09:32:00	2.55	17850.0	United Kingdom
28	536375	71053	WHITE METAL LANTERN	6	2010-12-01 09:32:00	3.39	17850.0	United Kingdom
29	536375	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 09:32:00	2.75	17850.0	United Kingdom
...	...	...	...	...	...	...	...	...
406799	581578	84997D	CHILDRENS CUTLERY POLKADOT PINK	8	2011-12-09 12:16:00	4.15	12713.0	Germany

	transaction id	product id	product description	quantity sold	transaction timestamp	unit price	customer id	transaction country
406800	581578	84997B	CHILDRENS CUTLERY RETROSPOT RED	8	2011-12-09 12:16:00	4.15	12713.0	Germany
406801	581578	84997C	CHILDRENS CUTLERY POLKADOT BLUE	8	2011-12-09 12:16:00	4.15	12713.0	Germany
406802	581578	22555	PLASTERS IN TIN STRONGMAN	12	2011-12-09 12:16:00	1.65	12713.0	Germany
406803	581578	21914	BLUE HARMONICA IN BOX	12	2011-12-09 12:16:00	1.25	12713.0	Germany
406804	581578	22549	PICTURE DOMINOES	24	2011-12-09 12:16:00	1.45	12713.0	Germany
406805	581578	21918	SET 12 KIDS COLOUR CHALK STICKS	24	2011-12-09 12:16:00	0.42	12713.0	Germany
406806	581578	22992	REVOLVER WOODEN RULER	12	2011-12-09 12:16:00	1.95	12713.0	Germany
406807	581578	22991	GIRAFFE WOODEN RULER	12	2011-12-09 12:16:00	1.95	12713.0	Germany
406808	581578	23229	VINTAGE DONKEY TAIL GAME	6	2011-12-09 12:16:00	3.75	12713.0	Germany
406809	581578	22622	BOX OF VINTAGE ALPHABET BLOCKS	6	2011-12-09 12:16:00	11.95	12713.0	Germany
406810	581578	21506	FANCY FONT BIRTHDAY CARD,	12	2011-12-09 12:16:00	0.42	12713.0	Germany
406811	581578	21507	ELEPHANT BIRTHDAY CARD	12	2011-12-09 12:16:00	0.42	12713.0	Germany
406812	581578	23037	CANDLE HOLDER SILVER MADELINE	12	2011-12-09 12:16:00	1.65	12713.0	Germany
406813	581578	23550	WRAP ALPHABET POSTER	25	2011-12-09 12:16:00	0.42	12713.0	Germany
406814	581578	22711	WRAP CIRCUS PARADE	25	2011-12-09 12:16:00	0.42	12713.0	Germany
406815	581578	21497	FANCY FONTS BIRTHDAY WRAP	25	2011-12-09 12:16:00	0.42	12713.0	Germany
406816	581578	22704	WRAP RED APPLES	25	2011-12-09 12:16:00	0.42	12713.0	Germany

	transaction id	product id	product description	quantity sold	transaction timestamp	unit price	customer id	transaction country
406817	581578	22585	PACK OF 6 BIRDY GIFT TAGS	12	2011-12-09 12:16:00	1.25	12713.0	Germany
406818	581578	23205	CHARLOTTE BAG VINTAGE ALPHABET	10	2011-12-09 12:16:00	0.85	12713.0	Germany
406819	581578	23201	JUMBO BAG ALPHABET	10	2011-12-09 12:16:00	2.08	12713.0	Germany
406820	581578	23515	EMBROIDERED RIBBON REEL DAISY	6	2011-12-09 12:16:00	2.08	12713.0	Germany
406821	581578	22081	RIBBON REEL FLORA + FAUNA	10	2011-12-09 12:16:00	1.65	12713.0	Germany
406822	581578	22080	RIBBON REEL POLKADOTS	10	2011-12-09 12:16:00	1.65	12713.0	Germany
406823	581578	23298	SPOTTY BUNTING	9	2011-12-09 12:16:00	4.95	12713.0	Germany
406824	581578	22993	SET OF 4 PANTRY JELLY MOULDS	12	2011-12-09 12:16:00	1.25	12713.0	Germany
406825	581578	22907	PACK OF 20 NAPKINS PANTRY DESIGN	12	2011-12-09 12:16:00	0.85	12713.0	Germany
406826	581578	22908	PACK OF 20 NAPKINS RED APPLES	12	2011-12-09 12:16:00	0.85	12713.0	Germany
406827	581578	23215	JINGLE BELL HEART ANTIQUE SILVER	12	2011-12-09 12:16:00	2.08	12713.0	Germany
406828	581578	22736	RIBBON REEL MAKING SNOWMEN	10	2011-12-09 12:16:00	1.65	12713.0	Germany

406829 rows × 13 columns

In [91]:
```
# Calculate the Revenue per month for each user type
```

In [92]:
```
dfuserreve=df.groupby(['YearMonth','UserType'])['Revenue'].sum().reset_index()
```

In [93]:
```
dfuserreve=dfuserreve.query("YearMonth!=201012 and YearMonth!=201112")
```

In [94]:
```
plot_data=[
    go.Scatter(
    x=dfuserreve.query("UserType=='Existing'")['YearMonth'],
        y=dfuserreve.query("UserType=='Existing'")['Revenue'],
        name='Existing'
    ),
    go.Scatter(
    x=dfuserreve.query("UserType=='New'")['YearMonth'],
        y=dfuserreve.query("UserType=='New'")['Revenue'],
        name='New'
    )

]
```

In [95]:
```
plot_layout=go.Layout(
xaxis={"type":"category"},
    title='Revenue Per User Type'
)
```

In [96]:
```
fig=go.Figure(data=plot_data,layout=plot_layout)
```

In [97]:
```
pyoff.plot(fig)
```

Out[97]:
```
'temp-plot.html'
```

In [98]:
```
dfuserreve
```

Out[98]:

	YearMonth	UserType	Revenue
1	201101	Existing	271616.520
2	201101	New	203457.860
3	201102	Existing	287024.770
4	201102	New	149521.380
5	201103	Existing	390034.530
6	201103	New	189930.080
7	201104	Existing	306283.600
8	201104	New	119764.251
9	201105	Existing	532392.340
10	201105	New	115858.740
11	201106	Existing	515486.650
12	201106	New	92526.510
13	201107	Existing	508355.610
14	201107	New	65882.871
15	201108	Existing	538709.770
16	201108	New	77658.230
17	201109	Existing	778161.781
18	201109	New	153278.591
19	201110	Existing	819672.900
20	201110	New	154930.690
21	201111	Existing	998176.360
22	201111	New	134231.380

In [99]:
```
# Customer Grouping using RFM
```

In [100]:
```
dfuser=pd.DataFrame(df['customer id'].unique())
```

```
In [101]: dfuser
```

Out[101]:

```
            0
   0  17850.0
   1  13047.0
   2  12583.0
   3  13748.0
   4  15100.0
   5  15291.0
   6  14688.0
   7  17809.0
   8  15311.0
   9  14527.0
  10  16098.0
  11  18074.0
  12  17420.0
  13  16029.0
  14  16250.0
  15  12431.0
  16  17511.0
  17  17548.0
  18  13705.0
  19  13747.0
  20  13408.0
  21  13767.0
  22  17924.0
  23  13448.0
  24  15862.0
  25  15513.0
  26  12791.0
  27  16218.0
  28  14045.0
  29  14307.0
 ...      ...
4342  15773.0
4343  17936.0
4344  16535.0
4345  16988.0
```

```
             0
4346   15097.0
4347   18015.0
4348   16597.0
4349   13790.0
4350   14219.0
4351   12367.0
4352   17383.0
4353   12478.0
4354   15992.0
4355   15318.0
4356   17914.0
4357   16528.0
4358   12442.0
4359   16569.0
4360   12650.0
4361   14578.0
4362   16000.0
4363   15195.0
4364   14087.0
4365   14204.0
4366   15471.0
4367   13436.0
4368   15520.0
4369   13298.0
4370   14569.0
4371   12713.0

4372 rows × 1 columns
```

In [102]:
```
dfuser.columns=['customer id']
```

In [103]:
```
dfuser
```

Out[103]:

	customer id
0	17850.0
1	13047.0
2	12583.0
3	13748.0
4	15100.0
5	15291.0
6	14688.0
7	17809.0
8	15311.0
9	14527.0
10	16098.0
11	18074.0
12	17420.0
13	16029.0
14	16250.0
15	12431.0
16	17511.0
17	17548.0
18	13705.0
19	13747.0
20	13408.0
21	13767.0
22	17924.0
23	13448.0
24	15862.0
25	15513.0
26	12791.0
27	16218.0
28	14045.0
29	14307.0
...	...
4342	15773.0
4343	17936.0
4344	16535.0
4345	16988.0

	customer id
4346	15097.0
4347	18015.0
4348	16597.0
4349	13790.0
4350	14219.0
4351	12367.0
4352	17383.0
4353	12478.0
4354	15992.0
4355	15318.0
4356	17914.0
4357	16528.0
4358	12442.0
4359	16569.0
4360	12650.0
4361	14578.0
4362	16000.0
4363	15195.0
4364	14087.0
4365	14204.0
4366	15471.0
4367	13436.0
4368	15520.0
4369	13298.0
4370	14569.0
4371	12713.0

4372 rows × 1 columns

In [104]:
```
dfusermaxpur=df.groupby(['customer id'])['transaction timestamp'].max().reset_
index()
```

```
In [105]: dfusermaxpur
```

Out[105]:

	customer id	transaction timestamp
0	12346.0	2011-01-18 10:17:00
1	12347.0	2011-12-07 15:52:00
2	12348.0	2011-09-25 13:13:00
3	12349.0	2011-11-21 09:51:00
4	12350.0	2011-02-02 16:01:00
5	12352.0	2011-11-03 14:37:00
6	12353.0	2011-05-19 17:47:00
7	12354.0	2011-04-21 13:11:00
8	12355.0	2011-05-09 13:49:00
9	12356.0	2011-11-17 08:40:00
10	12357.0	2011-11-06 16:07:00
11	12358.0	2011-12-08 10:26:00
12	12359.0	2011-12-02 11:21:00
13	12360.0	2011-10-18 15:22:00
14	12361.0	2011-02-25 13:51:00
15	12362.0	2011-12-06 15:40:00
16	12363.0	2011-08-22 10:18:00
17	12364.0	2011-12-02 10:22:00
18	12365.0	2011-02-21 14:04:00
19	12367.0	2011-12-05 16:48:00
20	12370.0	2011-10-19 14:51:00
21	12371.0	2011-10-26 10:16:00
22	12372.0	2011-09-29 12:12:00
23	12373.0	2011-02-01 13:10:00
24	12374.0	2011-11-14 15:37:00
25	12375.0	2011-12-07 11:27:00
26	12377.0	2011-01-28 15:45:00
27	12378.0	2011-08-02 10:34:00
28	12379.0	2011-09-19 10:09:00
29	12380.0	2011-11-18 11:27:00
...	...	...
4342	18245.0	2011-12-02 14:48:00
4343	18246.0	2011-11-16 11:49:00
4344	18248.0	2011-08-18 06:14:00
4345	18249.0	2011-11-22 15:07:00

	customer id	transaction timestamp
4346	18250.0	2011-02-11 13:59:00
4347	18251.0	2011-09-13 15:03:00
4348	18252.0	2011-10-20 12:43:00
4349	18255.0	2011-09-11 13:16:00
4350	18256.0	2010-12-20 08:27:00
4351	18257.0	2011-10-31 14:48:00
4352	18259.0	2011-11-15 12:34:00
4353	18260.0	2011-06-20 12:37:00
4354	18261.0	2011-10-27 15:36:00
4355	18262.0	2011-07-22 16:04:00
4356	18263.0	2011-11-16 16:19:00
4357	18265.0	2011-09-28 14:10:00
4358	18268.0	2011-07-28 19:13:00
4359	18269.0	2010-12-16 15:39:00
4360	18270.0	2011-11-01 13:57:00
4361	18272.0	2011-12-07 12:43:00
4362	18273.0	2011-12-07 13:16:00
4363	18274.0	2011-11-22 10:18:00
4364	18276.0	2011-11-18 17:01:00
4365	18277.0	2011-10-12 15:22:00
4366	18278.0	2011-09-27 11:58:00
4367	18280.0	2011-03-07 09:52:00
4368	18281.0	2011-06-12 10:53:00
4369	18282.0	2011-12-02 11:43:00
4370	18283.0	2011-12-06 12:02:00
4371	18287.0	2011-10-28 09:29:00

4372 rows × 2 columns

```
In [106]: dfusermaxpur.columns=['customer id','MaxPurchaseDate']
In [107]: dfusermaxpur
```

Out[107]:

	customer id	MaxPurchaseDate
0	12346.0	2011-01-18 10:17:00
1	12347.0	2011-12-07 15:52:00
2	12348.0	2011-09-25 13:13:00
3	12349.0	2011-11-21 09:51:00
4	12350.0	2011-02-02 16:01:00
5	12352.0	2011-11-03 14:37:00
6	12353.0	2011-05-19 17:47:00
7	12354.0	2011-04-21 13:11:00
8	12355.0	2011-05-09 13:49:00
9	12356.0	2011-11-17 08:40:00
10	12357.0	2011-11-06 16:07:00
11	12358.0	2011-12-08 10:26:00
12	12359.0	2011-12-02 11:21:00
13	12360.0	2011-10-18 15:22:00
14	12361.0	2011-02-25 13:51:00
15	12362.0	2011-12-06 15:40:00
16	12363.0	2011-08-22 10:18:00
17	12364.0	2011-12-02 10:22:00
18	12365.0	2011-02-21 14:04:00
19	12367.0	2011-12-05 16:48:00
20	12370.0	2011-10-19 14:51:00
21	12371.0	2011-10-26 10:16:00
22	12372.0	2011-09-29 12:12:00
23	12373.0	2011-02-01 13:10:00
24	12374.0	2011-11-14 15:37:00
25	12375.0	2011-12-07 11:27:00
26	12377.0	2011-01-28 15:45:00
27	12378.0	2011-08-02 10:34:00
28	12379.0	2011-09-19 10:09:00
29	12380.0	2011-11-18 11:27:00
...	...	...
4342	18245.0	2011-12-02 14:48:00
4343	18246.0	2011-11-16 11:49:00
4344	18248.0	2011-08-18 06:14:00
4345	18249.0	2011-11-22 15:07:00

	customer id	MaxPurchaseDate
4346	18250.0	2011-02-11 13:59:00
4347	18251.0	2011-09-13 15:03:00
4348	18252.0	2011-10-20 12:43:00
4349	18255.0	2011-09-11 13:16:00
4350	18256.0	2010-12-20 08:27:00
4351	18257.0	2011-10-31 14:48:00
4352	18259.0	2011-11-15 12:34:00
4353	18260.0	2011-06-20 12:37:00
4354	18261.0	2011-10-27 15:36:00
4355	18262.0	2011-07-22 16:04:00
4356	18263.0	2011-11-16 16:19:00
4357	18265.0	2011-09-28 14:10:00
4358	18268.0	2011-07-28 19:13:00
4359	18269.0	2010-12-16 15:39:00
4360	18270.0	2011-11-01 13:57:00
4361	18272.0	2011-12-07 12:43:00
4362	18273.0	2011-12-07 13:16:00
4363	18274.0	2011-11-22 10:18:00
4364	18276.0	2011-11-18 17:01:00
4365	18277.0	2011-10-12 15:22:00
4366	18278.0	2011-09-27 11:58:00
4367	18280.0	2011-03-07 09:52:00
4368	18281.0	2011-06-12 10:53:00
4369	18282.0	2011-12-02 11:43:00
4370	18283.0	2011-12-06 12:02:00
4371	18287.0	2011-10-28 09:29:00

4372 rows × 2 columns

In [108]:
```
dfusermaxpur['Recency'] = (dfusermaxpur['MaxPurchaseDate'].max() - dfusermaxpu
r['MaxPurchaseDate']).dt.days
```

In [109]:
```
dfusermaxpur
```

Out[109]:

```
      customer id      MaxPurchaseDate  Recency
   0      12346.0  2011-01-18 10:17:00      325
   1      12347.0  2011-12-07 15:52:00        1
   2      12348.0  2011-09-25 13:13:00       74
   3      12349.0  2011-11-21 09:51:00       18
   4      12350.0  2011-02-02 16:01:00      309
   5      12352.0  2011-11-03 14:37:00       35
   6      12353.0  2011-05-19 17:47:00      203
   7      12354.0  2011-04-21 13:11:00      231
   8      12355.0  2011-05-09 13:49:00      213
   9      12356.0  2011-11-17 08:40:00       22
  10      12357.0  2011-11-06 16:07:00       32
  11      12358.0  2011-12-08 10:26:00        1
  12      12359.0  2011-12-02 11:21:00        7
  13      12360.0  2011-10-18 15:22:00       51
  14      12361.0  2011-02-25 13:51:00      286
  15      12362.0  2011-12-06 15:40:00        2
  16      12363.0  2011-08-22 10:18:00      109
  17      12364.0  2011-12-02 10:22:00        7
  18      12365.0  2011-02-21 14:04:00      290
  19      12367.0  2011-12-05 16:48:00        3
  20      12370.0  2011-10-19 14:51:00       50
  21      12371.0  2011-10-26 10:16:00       44
  22      12372.0  2011-09-29 12:12:00       71
  23      12373.0  2011-02-01 13:10:00      310
  24      12374.0  2011-11-14 15:37:00       24
  25      12375.0  2011-12-07 11:27:00        2
  26      12377.0  2011-01-28 15:45:00      314
  27      12378.0  2011-08-02 10:34:00      129
  28      12379.0  2011-09-19 10:09:00       81
  29      12380.0  2011-11-18 11:27:00       21
 ...          ...                  ...      ...
4342      18245.0  2011-12-02 14:48:00        6
4343      18246.0  2011-11-16 11:49:00       23
4344      18248.0  2011-08-18 06:14:00      113
4345      18249.0  2011-11-22 15:07:00       16
```

	customer id	MaxPurchaseDate	Recency
4346	18250.0	2011-02-11 13:59:00	300
4347	18251.0	2011-09-13 15:03:00	86
4348	18252.0	2011-10-20 12:43:00	50
4349	18255.0	2011-09-11 13:16:00	88
4350	18256.0	2010-12-20 08:27:00	354
4351	18257.0	2011-10-31 14:48:00	38
4352	18259.0	2011-11-15 12:34:00	24
4353	18260.0	2011-06-20 12:37:00	172
4354	18261.0	2011-10-27 15:36:00	42
4355	18262.0	2011-07-22 16:04:00	139
4356	18263.0	2011-11-16 16:19:00	22
4357	18265.0	2011-09-28 14:10:00	71
4358	18268.0	2011-07-28 19:13:00	133
4359	18269.0	2010-12-16 15:39:00	357
4360	18270.0	2011-11-01 13:57:00	37
4361	18272.0	2011-12-07 12:43:00	2
4362	18273.0	2011-12-07 13:16:00	1
4363	18274.0	2011-11-22 10:18:00	17
4364	18276.0	2011-11-18 17:01:00	20
4365	18277.0	2011-10-12 15:22:00	57
4366	18278.0	2011-09-27 11:58:00	73
4367	18280.0	2011-03-07 09:52:00	277
4368	18281.0	2011-06-12 10:53:00	180
4369	18282.0	2011-12-02 11:43:00	7
4370	18283.0	2011-12-06 12:02:00	3
4371	18287.0	2011-10-28 09:29:00	42

4372 rows × 3 columns

In [110]:
```
dfuser=pd.merge(dfuser,dfusermaxpur[['customer id','Recency']],on='customer i
d')
```

In [111]:
```
dfuser
```

Out[111]:

	customer id	Recency
0	17850.0	301
1	13047.0	31
2	12583.0	2
3	13748.0	95
4	15100.0	329
5	15291.0	25
6	14688.0	7
7	17809.0	15
8	15311.0	0
9	14527.0	2
10	16098.0	87
11	18074.0	373
12	17420.0	49
13	16029.0	38
14	16250.0	260
15	12431.0	35
16	17511.0	2
17	17548.0	217
18	13705.0	7
19	13747.0	373
20	13408.0	1
21	13767.0	1
22	17924.0	0
23	13448.0	16
24	15862.0	7
25	15513.0	30
26	12791.0	373
27	16218.0	29
28	14045.0	108
29	14307.0	88
...	...	...
4342	15773.0	5
4343	17936.0	4
4344	16535.0	4
4345	16988.0	4

	customer id	Recency
4346	15097.0	3
4347	18015.0	3
4348	16597.0	3
4349	13790.0	3
4350	14219.0	3
4351	12367.0	3
4352	17383.0	3
4353	12478.0	3
4354	15992.0	3
4355	15318.0	3
4356	17914.0	3
4357	16528.0	3
4358	12442.0	2
4359	16569.0	2
4360	12650.0	2
4361	14578.0	2
4362	16000.0	2
4363	15195.0	2
4364	14087.0	2
4365	14204.0	1
4366	15471.0	1
4367	13436.0	1
4368	15520.0	1
4369	13298.0	0
4370	14569.0	0
4371	12713.0	0

4372 rows × 2 columns

In [147]:
```
plot_data=[go.Histogram(x=dfuser['Recency'])]
```

In [148]:
```
plot_layout=go.Layout(
title='Recency'
)
```

In [149]:
```
fig=go.Figure(data=plot_data,layout=plot_layout)
```

In [150]:
```
pyoff.plot(fig)
```

Out[150]:
```
'temp-plot.html'
```

In [116]:
```
dfuser.head()
```

Out[116]:

	customer id	Recency
0	17850.0	301
1	13047.0	31
2	12583.0	2
3	13748.0	95
4	15100.0	329

In [117]:
```
dfuser.Recency.describe()
```

Out[117]:
```
count    4372.000000
mean       91.047118
std       100.765435
min         0.000000
25%        16.000000
50%        49.000000
75%       142.000000
max       373.000000
Name: Recency, dtype: float64
```

In [118]:
```
df.Revenue.describe()
```

Out[118]:
```
count    406829.000000
mean         20.401854
std         427.591718
min     -168469.600000
25%           4.200000
50%          11.100000
75%          19.500000
max      168469.600000
Name: Revenue, dtype: float64
```

In [119]:
```
from sklearn.cluster import KMeans
```

In [120]:
```
sse={}
```

In [121]:
```
dfrecency=dfuser[['Recency']]
```

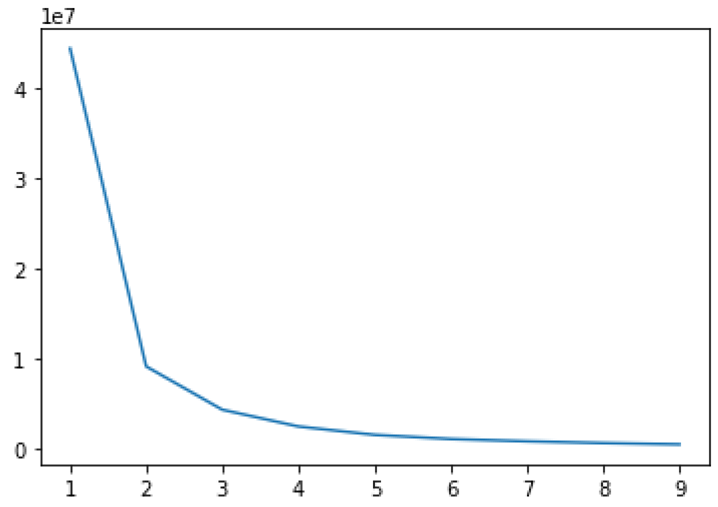
In [122]:
```
for k in range(1, 10):
    kmeans = KMeans(n_clusters=k, max_iter=1000).fit(dfrecency)
    dfrecency["clusters"] = kmeans.labels_
    sse[k] = kmeans.inertia_
```

In [123]:
```
plt.figure()
```

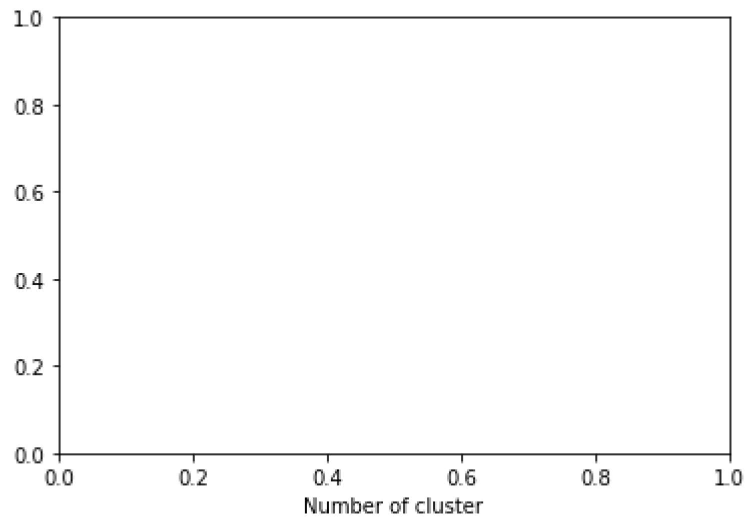
Out[123]:
```
<Figure size 432x288 with 0 Axes>

<Figure size 432x288 with 0 Axes>
```

In [124]:
```
plt.plot(list(sse.keys()), list(sse.values()))
```

Out[124]:
```
[<matplotlib.lines.Line2D at 0x1a7b4819c88>]
```



In [125]:
```
plt.xlabel("Number of cluster")
plt.show()
```



In [126]:
```
#build 4 clusters for recency and add it to dataframe
kmeans = KMeans(n_clusters=4)
kmeans.fit(dfuser[['Recency']])
dfuser['RecencyCluster'] = kmeans.predict(dfuser[['Recency']])
```

In [127]:
```
def order_cluster(cluster_field_name, target_field_name,f,ascending):
    new_cluster_field_name = 'new_' + cluster_field_name
    df_new = f.groupby(cluster_field_name)[target_field_name].mean().reset_ind
ex()
    df_new = df_new.sort_values(by=target_field_name,ascending=ascending).rese
t_index(drop=True)
    df_new['index'] = df_new.index
    df_final = pd.merge(f,df_new[[cluster_field_name,'index']], on=cluster_fie
ld_name)
    df_final = df_final.drop([cluster_field_name],axis=1)
    df_final = df_final.rename(columns={"index":cluster_field_name})
    return df_final
```

In [128]:
```
dfuser = order_cluster('RecencyCluster', 'Recency',dfuser,False)
```

In [129]:
```
dfuser
```

Out[129]:

	customer id	Recency	RecencyCluster
0	17850.0	301	0
1	15100.0	329	0
2	18074.0	373	0
3	16250.0	260	0
4	13747.0	373	0
5	12791.0	373	0
6	17908.0	373	0
7	16583.0	373	0
8	18085.0	329	0
9	17968.0	373	0
10	14729.0	373	0
11	14237.0	372	0
12	15350.0	372	0
13	15922.0	371	0
14	15165.0	372	0
15	17643.0	372	0
16	13093.0	266	0
17	16274.0	372	0
18	14496.0	311	0
19	14142.0	372	0
20	13065.0	372	0
21	18011.0	372	0
22	13715.0	280	0
23	17732.0	372	0
24	12855.0	372	0
25	17855.0	372	0
26	17925.0	372	0
27	13108.0	372	0
28	15070.0	372	0
29	16546.0	290	0
...	...	...	...
4342	15895.0	149	1
4343	15795.0	149	1
4344	16450.0	149	1
4345	13667.0	148	1

	customer id	Recency	RecencyCluster
4346	14400.0	140	1
4347	12405.0	148	1
4348	14847.0	148	1
4349	15256.0	148	1
4350	17948.0	146	1
4351	15004.0	146	1
4352	14722.0	146	1
4353	13967.0	145	1
4354	16054.0	144	1
4355	12833.0	144	1
4356	17984.0	144	1
4357	17448.0	144	1
4358	15369.0	143	1
4359	13154.0	143	1
4360	14117.0	143	1
4361	17065.0	142	1
4362	12521.0	142	1
4363	17866.0	142	1
4364	15802.0	142	1
4365	17962.0	141	1
4366	14259.0	141	1
4367	17694.0	141	1
4368	17660.0	140	1
4369	15623.0	140	1
4370	18262.0	139	1
4371	16305.0	138	1

4372 rows × 3 columns

```
In [130]: dfuser.groupby('RecencyCluster')['Recency'].describe()
```

Out[130]:

	count	mean	std	min	25%	50%	75%	max
RecencyCluster								
0	505.0	308.968317	39.095414	251.0	274.0	305.0	351.0	373.0
1	611.0	191.032733	32.225266	138.0	164.0	189.0	217.0	250.0
2	984.0	83.391260	23.837368	52.0	63.0	77.0	101.0	137.0
3	2272.0	19.036532	14.683479	0.0	7.0	17.0	30.0	51.0

In [131]:
```
#FREQUENCY CLUSTERS
# To create frequency clusters, we need to find total number orders for each c
ustomer.
```

In [132]:
```
#get order counts for each user and create a dataframe with it
dffrequency = df.groupby('customer id')['transaction timestamp'].count().reset
_index()
```

```
In [133]: dffrequency
```

Out[133]:

	customer id	transaction timestamp
0	12346.0	2
1	12347.0	182
2	12348.0	31
3	12349.0	73
4	12350.0	17
5	12352.0	95
6	12353.0	4
7	12354.0	58
8	12355.0	13
9	12356.0	59
10	12357.0	131
11	12358.0	19
12	12359.0	254
13	12360.0	129
14	12361.0	10
15	12362.0	274
16	12363.0	23
17	12364.0	85
18	12365.0	23
19	12367.0	11
20	12370.0	167
21	12371.0	63
22	12372.0	52
23	12373.0	14
24	12374.0	33
25	12375.0	18
26	12377.0	77
27	12378.0	219
28	12379.0	41
29	12380.0	105
...	...	...
4342	18245.0	177
4343	18246.0	4
4344	18248.0	49
4345	18249.0	8

	customer id	transaction timestamp
4346	18250.0	22
4347	18251.0	16
4348	18252.0	98
4349	18255.0	6
4350	18256.0	4
4351	18257.0	123
4352	18259.0	42
4353	18260.0	140
4354	18261.0	21
4355	18262.0	13
4356	18263.0	62
4357	18265.0	46
4358	18268.0	2
4359	18269.0	8
4360	18270.0	13
4361	18272.0	170
4362	18273.0	3
4363	18274.0	22
4364	18276.0	16
4365	18277.0	9
4366	18278.0	9
4367	18280.0	10
4368	18281.0	7
4369	18282.0	13
4370	18283.0	756
4371	18287.0	70

4372 rows × 2 columns

```
In [134]: dffrequency.columns = ['customer id','Frequency']
In [135]: dffrequency
```

Out[135]:

	customer id	Frequency
0	12346.0	2
1	12347.0	182
2	12348.0	31
3	12349.0	73
4	12350.0	17
5	12352.0	95
6	12353.0	4
7	12354.0	58
8	12355.0	13
9	12356.0	59
10	12357.0	131
11	12358.0	19
12	12359.0	254
13	12360.0	129
14	12361.0	10
15	12362.0	274
16	12363.0	23
17	12364.0	85
18	12365.0	23
19	12367.0	11
20	12370.0	167
21	12371.0	63
22	12372.0	52
23	12373.0	14
24	12374.0	33
25	12375.0	18
26	12377.0	77
27	12378.0	219
28	12379.0	41
29	12380.0	105
...	...	...
4342	18245.0	177
4343	18246.0	4
4344	18248.0	49
4345	18249.0	8

	customer id	Frequency
4346	18250.0	22
4347	18251.0	16
4348	18252.0	98
4349	18255.0	6
4350	18256.0	4
4351	18257.0	123
4352	18259.0	42
4353	18260.0	140
4354	18261.0	21
4355	18262.0	13
4356	18263.0	62
4357	18265.0	46
4358	18268.0	2
4359	18269.0	8
4360	18270.0	13
4361	18272.0	170
4362	18273.0	3
4363	18274.0	22
4364	18276.0	16
4365	18277.0	9
4366	18278.0	9
4367	18280.0	10
4368	18281.0	7
4369	18282.0	13
4370	18283.0	756
4371	18287.0	70

4372 rows × 2 columns

```
In [136]: #add this data to our main dataframe
          dfuser = pd.merge(dfuser, dffrequency, on='customer id')
```

In [137]:
```
#plot the histogram
plot_data = [
    go.Histogram(
        x=dfuser.query('Frequency < 1000')['Frequency']
    )
]

plot_layout = go.Layout(
        title='Frequency'
    )
fig = go.Figure(data=plot_data, layout=plot_layout)
pyoff.plot(fig)
```

Out[137]:
```
'temp-plot.html'
```

In [138]:
```
kmeans = KMeans(n_clusters=4)
kmeans.fit(dfuser[['Frequency']])
```

Out[138]:
```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
    n_clusters=4, n_init=10, n_jobs=None, precompute_distances='auto',
    random_state=None, tol=0.0001, verbose=0)
```

In [139]:
```
dfuser['FrequencyCluster'] = kmeans.predict(dfuser[['Frequency']])
```

In [140]:
```
def order_cluster(cluster_field_name, target_field_name,f,ascending):
    new_cluster_field_name = 'new_' + cluster_field_name
    df_new = f.groupby(cluster_field_name)[target_field_name].mean().reset_ind
ex()
    df_new = df_new.sort_values(by=target_field_name,ascending=ascending).rese
t_index(drop=True)
    df_new['index'] = df_new.index
    df_final = pd.merge(f,df_new[[cluster_field_name,'index']], on=cluster_fie
ld_name)
    df_final = df_final.drop([cluster_field_name],axis=1)
    df_final = df_final.rename(columns={"index":cluster_field_name})
    return df_final
```

In [141]:
```
#order the frequency cluster
dfuser = order_cluster('FrequencyCluster', 'Frequency',dfuser,True)
```

In [142]:
```
dfuser
```

Out[142]:

	customer id	Recency	RecencyCluster	Frequency	FrequencyCluster
0	17850.0	301	0	312	1
1	15808.0	305	0	210	1
2	13047.0	31	3	196	1
3	12583.0	2	3	251	1
4	14688.0	7	3	359	1
5	16029.0	38	3	274	1
6	12431.0	35	3	240	1
7	13408.0	1	3	501	1
8	13767.0	1	3	399	1
9	13448.0	16	3	199	1
10	15513.0	30	3	314	1
11	17920.0	3	3	696	1
12	13694.0	3	3	585	1
13	14849.0	21	3	392	1
14	17377.0	23	3	419	1
15	12662.0	0	3	232	1
16	12433.0	0	3	420	1
17	12472.0	30	3	391	1
18	17346.0	3	3	503	1
19	12921.0	3	3	741	1
20	13468.0	1	3	306	1
21	13777.0	0	3	219	1
22	17690.0	29	3	258	1
23	14092.0	7	3	217	1
24	15752.0	39	3	412	1
25	17017.0	2	3	268	1
26	12471.0	1	3	531	1
27	15601.0	10	3	414	1
28	13418.0	11	3	314	1
29	14388.0	8	3	191	1
...	...	...	...	...	...
4342	18262.0	139	1	13	0
4343	16305.0	138	1	22	0
4344	15311.0	0	3	2491	2
4345	14527.0	2	3	1011	2

	customer id	Recency	RecencyCluster	Frequency	FrequencyCluster
4346	17511.0	2	3	1076	2
4347	14606.0	0	3	2782	2
4348	14156.0	9	3	1420	2
4349	13081.0	0	3	1061	2
4350	13089.0	2	3	1857	2
4351	16033.0	5	3	1152	2
4352	16931.0	4	3	898	2
4353	18118.0	10	3	1284	2
4354	15555.0	11	3	925	2
4355	15039.0	9	3	1508	2
4356	14796.0	0	3	1165	2
4357	15005.0	15	3	1160	2
4358	14159.0	19	3	1212	2
4359	14298.0	2	3	1640	2
4360	14769.0	2	3	1094	2
4361	14646.0	1	3	2085	2
4362	15719.0	32	3	938	2
4363	16549.0	9	3	981	2
4364	17811.0	3	3	872	2
4365	13263.0	0	3	1677	2
4366	14056.0	0	3	1128	2
4367	14456.0	4	3	977	2
4368	12748.0	0	3	4642	3
4369	14911.0	0	3	5903	3
4370	17841.0	1	3	7983	3
4371	14096.0	3	3	5128	3

4372 rows × 5 columns

In [143]:
```
#see details of each cluster
dfuser.groupby('FrequencyCluster')['Frequency'].describe()
# high frequency number indicates better customers.
```

Out[143]:

	count	mean	std	min	25%	50%	75%	max
FrequencyCluster								
0	3862.0	49.753755	44.812998	1.0	15.0	34.0	73.0	189.0
1	482.0	329.107884	132.670589	190.0	228.0	286.0	393.5	803.0
2	24.0	1349.750000	508.637759	872.0	1003.5	1156.0	1541.0	2782.0
3	4.0	5914.000000	1473.845537	4642.0	5006.5	5515.5	6423.0	7983.0

In [127]:
```
# On the basis of Revenue
#Let’s see how our customer database looks like when we cluster them based on
 revenue.
```

In [ ]:
```
#calculate revenue for each customer

tx_revenue = tx_uk.groupby('CustomerID').Revenue.sum().reset_index()

#merge it with our main dataframe
tx_user = pd.merge(tx_user, tx_revenue, on='CustomerID')

#plot the histogram
plot_data = [
    go.Histogram(
        x=tx_user.query('Revenue < 10000')['Revenue']
    )
]

plot_layout = go.Layout(
        title='Monetary Value'
    )
fig = go.Figure(data=plot_data, layout=plot_layout)
pyoff.iplot(fig)
```

```
In [129]: df
```

Out[129]:

	transaction id	product id	product description	quantity sold	transaction timestamp	unit price	customer id	transaction country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
5	536365	22752	SET 7 BABUSHKA NESTING BOXES	2	2010-12-01 08:26:00	7.65	17850.0	United Kingdom
6	536365	21730	GLASS STAR FROSTED T-LIGHT HOLDER	6	2010-12-01 08:26:00	4.25	17850.0	United Kingdom
7	536366	22633	HAND WARMER UNION JACK	6	2010-12-01 08:28:00	1.85	17850.0	United Kingdom
8	536366	22632	HAND WARMER RED POLKA DOT	6	2010-12-01 08:28:00	1.85	17850.0	United Kingdom
9	536372	22632	HAND WARMER RED POLKA DOT	6	2010-12-01 09:01:00	1.85	17850.0	United Kingdom
10	536372	22633	HAND WARMER UNION JACK	6	2010-12-01 09:01:00	1.85	17850.0	United Kingdom
11	536373	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 09:02:00	2.55	17850.0	United Kingdom
12	536373	71053	WHITE METAL LANTERN	6	2010-12-01 09:02:00	3.39	17850.0	United Kingdom
13	536373	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 09:02:00	2.75	17850.0	United Kingdom
14	536373	20679	EDWARDIAN PARASOL RED	6	2010-12-01 09:02:00	4.95	17850.0	United Kingdom
15	536373	37370	RETRO COFFEE MUGS ASSORTED	6	2010-12-01 09:02:00	1.06	17850.0	United Kingdom
16	536373	21871	SAVE THE PLANET MUG	6	2010-12-01 09:02:00	1.06	17850.0	United Kingdom

	transaction id	product id	product description	quantity sold	transaction timestamp	unit price	customer id	transaction country
17	536373	21071	VINTAGE BILLBOARD DRINK ME MUG	6	2010-12-01 09:02:00	1.06	17850.0	United Kingdom
18	536373	21068	VINTAGE BILLBOARD LOVE/HATE MUG	6	2010-12-01 09:02:00	1.06	17850.0	United Kingdom
19	536373	82483	WOOD 2 DRAWER CABINET WHITE FINISH	2	2010-12-01 09:02:00	4.95	17850.0	United Kingdom
20	536373	82486	WOOD S/3 CABINET ANT WHITE FINISH	4	2010-12-01 09:02:00	6.95	17850.0	United Kingdom
21	536373	82482	WOODEN PICTURE FRAME WHITE FINISH	6	2010-12-01 09:02:00	2.10	17850.0	United Kingdom
22	536373	82494L	WOODEN FRAME ANTIQUE WHITE	6	2010-12-01 09:02:00	2.55	17850.0	United Kingdom
23	536373	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 09:02:00	3.39	17850.0	United Kingdom
24	536373	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 09:02:00	3.39	17850.0	United Kingdom
25	536373	22752	SET 7 BABUSHKA NESTING BOXES	2	2010-12-01 09:02:00	7.65	17850.0	United Kingdom
26	536373	21730	GLASS STAR FROSTED T-LIGHT HOLDER	6	2010-12-01 09:02:00	4.25	17850.0	United Kingdom
27	536375	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 09:32:00	2.55	17850.0	United Kingdom
28	536375	71053	WHITE METAL LANTERN	6	2010-12-01 09:32:00	3.39	17850.0	United Kingdom
29	536375	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 09:32:00	2.75	17850.0	United Kingdom
...	...	...	...	...	...	...	...	...
406799	581578	84997D	CHILDRENS CUTLERY POLKADOT PINK	8	2011-12-09 12:16:00	4.15	12713.0	Germany

	transaction id	product id	product description	quantity sold	transaction timestamp	unit price	customer id	transaction country
406800	581578	84997B	CHILDRENS CUTLERY RETROSPOT RED	8	2011-12-09 12:16:00	4.15	12713.0	Germany
406801	581578	84997C	CHILDRENS CUTLERY POLKADOT BLUE	8	2011-12-09 12:16:00	4.15	12713.0	Germany
406802	581578	22555	PLASTERS IN TIN STRONGMAN	12	2011-12-09 12:16:00	1.65	12713.0	Germany
406803	581578	21914	BLUE HARMONICA IN BOX	12	2011-12-09 12:16:00	1.25	12713.0	Germany
406804	581578	22549	PICTURE DOMINOES	24	2011-12-09 12:16:00	1.45	12713.0	Germany
406805	581578	21918	SET 12 KIDS COLOUR CHALK STICKS	24	2011-12-09 12:16:00	0.42	12713.0	Germany
406806	581578	22992	REVOLVER WOODEN RULER	12	2011-12-09 12:16:00	1.95	12713.0	Germany
406807	581578	22991	GIRAFFE WOODEN RULER	12	2011-12-09 12:16:00	1.95	12713.0	Germany
406808	581578	23229	VINTAGE DONKEY TAIL GAME	6	2011-12-09 12:16:00	3.75	12713.0	Germany
406809	581578	22622	BOX OF VINTAGE ALPHABET BLOCKS	6	2011-12-09 12:16:00	11.95	12713.0	Germany
406810	581578	21506	FANCY FONT BIRTHDAY CARD,	12	2011-12-09 12:16:00	0.42	12713.0	Germany
406811	581578	21507	ELEPHANT BIRTHDAY CARD	12	2011-12-09 12:16:00	0.42	12713.0	Germany
406812	581578	23037	CANDLE HOLDER SILVER MADELINE	12	2011-12-09 12:16:00	1.65	12713.0	Germany
406813	581578	23550	WRAP ALPHABET POSTER	25	2011-12-09 12:16:00	0.42	12713.0	Germany
406814	581578	22711	WRAP CIRCUS PARADE	25	2011-12-09 12:16:00	0.42	12713.0	Germany
406815	581578	21497	FANCY FONTS BIRTHDAY WRAP	25	2011-12-09 12:16:00	0.42	12713.0	Germany
406816	581578	22704	WRAP RED APPLES	25	2011-12-09 12:16:00	0.42	12713.0	Germany

	transaction id	product id	product description	quantity sold	transaction timestamp	unit price	customer id	transaction country
406817	581578	22585	PACK OF 6 BIRDY GIFT TAGS	12	2011-12-09 12:16:00	1.25	12713.0	Germany
406818	581578	23205	CHARLOTTE BAG VINTAGE ALPHABET	10	2011-12-09 12:16:00	0.85	12713.0	Germany
406819	581578	23201	JUMBO BAG ALPHABET	10	2011-12-09 12:16:00	2.08	12713.0	Germany
406820	581578	23515	EMBROIDERED RIBBON REEL DAISY	6	2011-12-09 12:16:00	2.08	12713.0	Germany
406821	581578	22081	RIBBON REEL FLORA + FAUNA	10	2011-12-09 12:16:00	1.65	12713.0	Germany
406822	581578	22080	RIBBON REEL POLKADOTS	10	2011-12-09 12:16:00	1.65	12713.0	Germany
406823	581578	23298	SPOTTY BUNTING	9	2011-12-09 12:16:00	4.95	12713.0	Germany
406824	581578	22993	SET OF 4 PANTRY JELLY MOULDS	12	2011-12-09 12:16:00	1.25	12713.0	Germany
406825	581578	22907	PACK OF 20 NAPKINS PANTRY DESIGN	12	2011-12-09 12:16:00	0.85	12713.0	Germany
406826	581578	22908	PACK OF 20 NAPKINS RED APPLES	12	2011-12-09 12:16:00	0.85	12713.0	Germany
406827	581578	23215	JINGLE BELL HEART ANTIQUE SILVER	12	2011-12-09 12:16:00	2.08	12713.0	Germany
406828	581578	22736	RIBBON REEL MAKING SNOWMEN	10	2011-12-09 12:16:00	1.65	12713.0	Germany

406829 rows × 13 columns

```
In [133]: dfrevenue = df.groupby('customer id')['Revenue'].sum().reset_index()
In [134]: dfrevenue
```

Out[134]:

	customer id	Revenue
0	12346.0	0.000000e+00
1	12347.0	4.310000e+03
2	12348.0	1.797240e+03
3	12349.0	1.757550e+03
4	12350.0	3.344000e+02
5	12352.0	1.545410e+03
6	12353.0	8.900000e+01
7	12354.0	1.079400e+03
8	12355.0	4.594000e+02
9	12356.0	2.811430e+03
10	12357.0	6.207670e+03
11	12358.0	1.168060e+03
12	12359.0	6.245530e+03
13	12360.0	2.662060e+03
14	12361.0	1.899000e+02
15	12362.0	5.154580e+03
16	12363.0	5.520000e+02
17	12364.0	1.313100e+03
18	12365.0	3.206900e+02
19	12367.0	1.689000e+02
20	12370.0	3.545690e+03
21	12371.0	1.887960e+03
22	12372.0	1.298040e+03
23	12373.0	3.646000e+02
24	12374.0	7.429300e+02
25	12375.0	4.554200e+02
26	12377.0	1.628120e+03
27	12378.0	4.008620e+03
28	12379.0	8.502900e+02
29	12380.0	2.720560e+03
...	...	...
4342	18245.0	2.507560e+03
4343	18246.0	5.961000e+02
4344	18248.0	7.830200e+02
4345	18249.0	9.534000e+01

	customer id	Revenue
4346	18250.0	3.429200e+02
4347	18251.0	4.314720e+03
4348	18252.0	5.266700e+02
4349	18255.0	1.033000e+02
4350	18256.0	-5.010000e+01
4351	18257.0	2.265380e+03
4352	18259.0	2.338600e+03
4353	18260.0	2.595000e+03
4354	18261.0	3.242400e+02
4355	18262.0	1.494800e+02
4356	18263.0	1.211080e+03
4357	18265.0	8.015100e+02
4358	18268.0	0.000000e+00
4359	18269.0	1.389000e+02
4360	18270.0	2.389500e+02
4361	18272.0	3.064780e+03
4362	18273.0	2.040000e+02
4363	18274.0	1.243450e-14
4364	18276.0	3.233600e+02
4365	18277.0	9.763000e+01
4366	18278.0	1.739000e+02
4367	18280.0	1.806000e+02
4368	18281.0	8.082000e+01
4369	18282.0	1.766000e+02
4370	18283.0	2.094880e+03
4371	18287.0	1.837280e+03

4372 rows × 2 columns

```
In [135]: dfuser = pd.merge(dfuser, dfrevenue, on='customer id')
In [136]: dfuser
```

Out[136]:

	customer id	Recency	Frequency	FrequencyCluster	RecencyCluster	Revenue
0	17850.0	301	312	1	0	5288.63
1	15808.0	305	210	1	0	3724.77
2	15100.0	329	6	0	0	635.10
3	18074.0	373	13	0	0	489.60
4	16250.0	260	24	0	0	389.44
5	13747.0	373	1	0	0	79.60
6	12791.0	373	2	0	0	192.60
7	17908.0	373	58	0	0	243.28
8	16583.0	373	14	0	0	233.45
9	18085.0	329	29	0	0	689.95
10	17968.0	373	85	0	0	277.35
11	14729.0	373	71	0	0	313.49
12	14237.0	372	9	0	0	161.00
13	15350.0	372	5	0	0	115.65
14	15922.0	371	12	0	0	363.60
15	15165.0	372	27	0	0	487.75
16	17643.0	372	8	0	0	101.55
17	13093.0	266	170	0	0	7741.47
18	16274.0	372	67	0	0	357.95
19	14496.0	311	19	0	0	538.81
20	14142.0	372	22	0	0	311.81
21	13065.0	372	14	0	0	205.86
22	18011.0	372	28	0	0	102.79
23	13715.0	280	108	0	0	1053.94
24	17732.0	372	18	0	0	303.97
25	12855.0	372	3	0	0	38.10
26	17855.0	372	17	0	0	208.97
27	17925.0	372	1	0	0	244.08
28	13108.0	372	10	0	0	350.06
29	15070.0	372	1	0	0	106.20
...	...	...	...	...	...	...
4342	14722.0	146	29	0	1	187.92
4343	13967.0	145	2	0	1	80.70
4344	16054.0	144	70	0	1	783.90
4345	12833.0	144	24	0	1	417.38

	customer id	Recency	Frequency	FrequencyCluster	RecencyCluster	Revenue
4346	17984.0	144	48	0	1	152.68
4347	17448.0	144	1	0	1	-4287.63
4348	15369.0	143	1	0	1	-1592.49
4349	13154.0	143	1	0	1	-611.86
4350	14117.0	143	3	0	1	90.00
4351	17065.0	142	1	0	1	-112.35
4352	12521.0	142	38	0	1	599.68
4353	17866.0	142	10	0	1	325.70
4354	15802.0	142	3	0	1	-451.42
4355	17962.0	141	35	0	1	102.83
4356	14259.0	141	5	0	1	120.00
4357	17935.0	137	29	0	1	145.79
4358	17694.0	141	15	0	1	283.12
4359	17660.0	140	45	0	1	196.00
4360	15623.0	140	19	0	1	301.03
4361	18262.0	139	13	0	1	149.48
4362	16305.0	138	22	0	1	361.22
4363	12864.0	137	3	0	1	147.12
4364	16178.0	137	8	0	1	197.90
4365	17040.0	137	15	0	1	449.73
4366	14100.0	137	26	0	1	194.90
4367	13296.0	136	5	0	1	87.40
4368	17693.0	135	18	0	1	187.02
4369	15372.0	136	27	0	1	2007.40
4370	13194.0	135	3	0	1	60.70
4371	16447.0	135	36	0	1	259.01

4372 rows × 6 columns

In [137]:
```
#plot the histogram
plot_data = [
    go.Histogram(
        x=dfuser.query('Revenue < 10000')['Revenue']
    )
]
```

In [138]:
```
plot_layout = go.Layout(
        title='Monetary Value'
    )
```

In [139]:
```
fig = go.Figure(data=plot_data, layout=plot_layout)
pyoff.plot(fig)
```

Out[139]:
```
'temp-plot.html'
```

In [141]:
```
#apply clustering
kmeans = KMeans(n_clusters=4)
kmeans.fit(dfuser[['Revenue']])
dfuser['RevenueCluster'] = kmeans.predict(dfuser[['Revenue']])
```

In [142]:
```
#order the cluster numbers
dfuser = order_cluster('RevenueCluster', 'Revenue',dfuser,True)
```

In [143]:
```
#show details of the dataframe
dfuser.groupby('RevenueCluster')['Revenue'].describe()
```

Out[143]:

	count	mean	std	min	25%	50%	7
RevenueCluster							
0	4235.0	1133.431375	1338.199599	-4287.63	284.130	621.23	1450.
1	119.0	14329.592605	7463.000051	7711.38	9144.105	11216.75	16341.
2	15.0	71423.516000	28632.631870	50415.49	52287.280	57385.88	77008.
3	3.0	241136.560000	47874.073443	187482.17	221960.330	256438.49	267963.

In [144]:
```
# Overall Score
```

In [145]:
```
#calculate overall score and use mean() to see details
dfuser['OverallScore'] = dfuser['RecencyCluster'] + dfuser['FrequencyCluster']
+ dfuser['RevenueCluster']
dfuser.groupby('OverallScore')['Recency','Frequency','Revenue'].mean()
```

Out[145]:

	Recency	Frequency	Revenue
OverallScore			
0	308.960239	21.980119	328.607117
1	190.139871	33.217042	549.932783
2	80.241313	46.654440	909.237425
3	20.588818	70.500291	1244.893798
4	13.416667	302.447917	4388.008698
5	7.464286	485.250000	13882.860357
6	6.928571	987.071429	36458.338571
7	2.555556	2432.555556	98056.746667
8	1.333333	4372.000000	156394.183333

In [146]:
```
dfuser['Segment'] = 'Low-Value'
dfuser.loc[dfuser['OverallScore']>2,'Segment'] = 'Mid-Value'
dfuser.loc[dfuser['OverallScore']>4,'Segment'] = 'High-Value'
```

In [148]:
```
dfuser
```

Out[148]:

	customer id	Recency	Frequency	FrequencyCluster	RecencyCluster	Revenue	RevenueClu
0	17850.0	301	312	1	0	5288.63	
1	15808.0	305	210	1	0	3724.77	
2	15100.0	329	6	0	0	635.10	
3	18074.0	373	13	0	0	489.60	
4	16250.0	260	24	0	0	389.44	
5	13747.0	373	1	0	0	79.60	
6	12791.0	373	2	0	0	192.60	
7	17908.0	373	58	0	0	243.28	
8	16583.0	373	14	0	0	233.45	
9	18085.0	329	29	0	0	689.95	
10	17968.0	373	85	0	0	277.35	
11	14729.0	373	71	0	0	313.49	
12	14237.0	372	9	0	0	161.00	
13	15350.0	372	5	0	0	115.65	
14	15922.0	371	12	0	0	363.60	
15	15165.0	372	27	0	0	487.75	
16	17643.0	372	8	0	0	101.55	
17	16274.0	372	67	0	0	357.95	
18	14496.0	311	19	0	0	538.81	
19	14142.0	372	22	0	0	311.81	
20	13065.0	372	14	0	0	205.86	
21	18011.0	372	28	0	0	102.79	
22	13715.0	280	108	0	0	1053.94	

	customer id	Recency	Frequency	FrequencyCluster	RecencyCluster	Revenue	RevenueClu
23	17732.0	372	18	0	0	303.97	
24	12855.0	372	3	0	0	38.10	
25	17855.0	372	17	0	0	208.97	
26	17925.0	372	1	0	0	244.08	
27	13108.0	372	10	0	0	350.06	
28	15070.0	372	1	0	0	106.20	
29	16546.0	290	31	0	0	-95.93	
...	...	...	...	...	...	...	
4342	14796.0	0	1165	2	3	7839.51	
4343	14769.0	2	1094	2	3	10641.60	
4344	14056.0	0	1128	2	3	8124.40	
4345	12748.0	0	4642	3	3	29072.10	
4346	17841.0	1	7983	3	3	40340.78	
4347	12744.0	51	229	1	2	9120.39	
4348	12939.0	63	47	0	2	11581.80	
4349	12435.0	79	36	0	2	7829.89	
4350	16180.0	99	176	0	2	10217.48	
4351	12409.0	78	114	0	2	11056.93	
4352	15749.0	234	15	0	1	21535.90	
4353	12590.0	189	70	0	1	9861.38	
4354	16029.0	38	274	1	3	50992.61	
4355	13694.0	3	585	1	3	62653.10	
4356	15061.0	3	410	1	3	54228.74	

	customer id	Recency	Frequency	FrequencyCluster	RecencyCluster	Revenue	RevenueClu
4357	16684.0	3	281	1	3	65892.08	
4358	12415.0	23	778	1	3	123725.45	
4359	14088.0	9	590	1	3	50415.49	
4360	17949.0	0	79	0	3	52750.84	
4361	15769.0	6	147	0	3	51823.72	
4362	15311.0	0	2491	2	3	59419.34	
4363	17511.0	2	1076	2	3	88125.38	
4364	14156.0	9	1420	2	3	113384.14	
4365	13089.0	2	1857	2	3	57385.88	
4366	14298.0	2	1640	2	3	50862.44	
4367	14911.0	0	5903	3	3	132572.62	
4368	14096.0	3	5128	3	3	57120.91	
4369	17450.0	7	351	1	3	187482.17	
4370	18102.0	0	433	1	3	256438.49	
4371	14646.0	1	2085	2	3	279489.02	

4372 rows × 9 columns

In [149]:
```
# High Value: Improve Retention
#Mid Value: Improve Retention + Increase Frequency
#Low Value: Increase Frequency
```

In [150]:
```
# Revenue v/s Frequency
dfgraph = dfuser.query("Revenue < 50000 and Frequency < 2000")
```

In [151]:
```
plot_data = [
    go.Scatter(
        x=dfgraph.query("Segment == 'Low-Value'")['Frequency'],
        y=dfgraph.query("Segment == 'Low-Value'")['Revenue'],
        mode='markers',
        name='Low',
        marker= dict(size= 7,
            line= dict(width=1),
            color= 'blue',
            opacity= 0.8
           )
    ),
        go.Scatter(
        x=dfgraph.query("Segment == 'Mid-Value'")['Frequency'],
        y=dfgraph.query("Segment == 'Mid-Value'")['Revenue'],
        mode='markers',
        name='Mid',
        marker= dict(size= 9,
            line= dict(width=1),
            color= 'green',
            opacity= 0.5
           )
    ),
        go.Scatter(
        x=dfgraph.query("Segment == 'High-Value'")['Frequency'],
        y=dfgraph.query("Segment == 'High-Value'")['Revenue'],
        mode='markers',
        name='High',
        marker= dict(size= 11,
            line= dict(width=1),
            color= 'red',
            opacity= 0.9
           )
    ),
]
```

In [152]:
```
plot_layout = go.Layout(
        yaxis= {'title': "Revenue"},
        xaxis= {'title': "Frequency"},
        title='Segments'
    )
```

In [153]:
```
fig = go.Figure(data=plot_data, layout=plot_layout)
pyoff.plot(fig)
```

Out[153]:
```
'temp-plot.html'
```

In [154]:
```
#Revenue Recency
dfgraph = dfuser.query("Revenue < 50000 and Frequency < 2000")
```

In [155]:
```
plot_data = [
    go.Scatter(
        x=dfgraph.query("Segment == 'Low-Value'")['Recency'],
        y=dfgraph.query("Segment == 'Low-Value'")['Revenue'],
        mode='markers',
        name='Low',
        marker= dict(size= 7,
            line= dict(width=1),
            color= 'blue',
            opacity= 0.8
           )
    ),
        go.Scatter(
        x=dfgraph.query("Segment == 'Mid-Value'")['Recency'],
        y=dfgraph.query("Segment == 'Mid-Value'")['Revenue'],
        mode='markers',
        name='Mid',
        marker= dict(size= 9,
            line= dict(width=1),
            color= 'green',
            opacity= 0.5
           )
    ),
        go.Scatter(
        x=dfgraph.query("Segment == 'High-Value'")['Recency'],
        y=dfgraph.query("Segment == 'High-Value'")['Revenue'],
        mode='markers',
        name='High',
        marker= dict(size= 11,
            line= dict(width=1),
            color= 'red',
            opacity= 0.9
           )
    ),
]

plot_layout = go.Layout(
        yaxis= {'title': "Revenue"},
        xaxis= {'title': "Recency"},
        title='Segments'
    )
fig = go.Figure(data=plot_data, layout=plot_layout)
pyoff.plot(fig)
```

Out[155]:
```
'temp-plot.html'
```

In [156]:
```
# INITIATION TO INCREASE THE SALES
```

In [157]:
```
df.head()
```

Out[157]:

	transaction id	product id	product description	quantity sold	transaction timestamp	unit price	customer id	transaction country	YearMor
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom	2010
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	2010
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom	2010
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	2010
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	2010

In [158]:
```
import numpy as np
```

In [160]:
```
print(df.loc[:,['unit price']].agg(['mean','median','std']))
```

```
        unit price
mean      3.460471
median    1.950000
std      69.315162
```

In [161]:
```
df['discount']=np.where(df['unit price']>1.9500,1,0)
```

In [164]:
```
df.head(3)
```

Out[164]:

	transaction id	product id	product description	quantity sold	transaction timestamp	unit price	customer id	transaction country	YearMor
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom	2010
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	2010
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom	2010

In [165]:
```
df['buyget1']=np.where(df['unit price']>3.460471,1,0)
```

In [166]:
```
df.head(3)
```

Out[166]:

	transaction id	product id	product description	quantity sold	transaction timestamp	unit price	customer id	transaction country	YearMor
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom	2010
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	2010
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom	2010

In [170]:
```
if df['unit price']==1:
    df['unitnew']=df['unit price']-(0.1*df['unit price'])
else:
    df['unitnew']=df['unit price'].copy()
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-170-e5db8f6d8c24> in <module>
----> 1 if df['unit price']==1:
      2     df['unitnew']=df['unit price']-(0.1*df['unit price'])
      3 else:
      4     df['unitnew']=df['unit price'].copy()
      5 

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\generic.py in __nonzer
o__(self)
   1574         raise ValueError("The truth value of a {0} is ambiguous. "
   1575                          "Use a.empty, a.bool(), a.item(), a.any() or
a.all()."
-> 1576                          .format(self.__class__.__name__))
   1577 
   1578     __bool__ = __nonzero__

ValueError: The truth value of a Series is ambiguous. Use a.empty, a.bool(),
 a.item(), a.any() or a.all().
```

In [ ]: