Retail\_project

By SADAF NASREEN

December 28, 2022

[3]:

**import numpy as np import pandas as pd**

data= pd.read\_excel("Online Retail.xlsx")

[4]:

data.head()

[4]: InvoiceNo StockCode Description Quantity \

0 536365 85123A WHITE HANGING HEART T-LIGHT HOLDER 6

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 536365 | 71053 | WHITE METAL LANTERN | | | 6 |
| 2 | 536365 | 84406B | CREAM CUPID HEARTS COAT HANGER | | | 8 |
| 3 | 536365 | 84029G | KNITTED UNION FLAG HOT WATER BOTTLE | | | 6 |
| 4 | 536365 | 84029E | RED WOOLLY HOTTIE WHITE HEART. | | | 6 |
| InvoiceDate | | | UnitPrice | CustomerID | Country | |
| 0 2010-12-01 08:26:00 | | | 2.55 | 17850.0 | United Kingdom | |
| 1 2010-12-01 08:26:00 | | | 3.39 | 17850.0 | United Kingdom | |
| 2 2010-12-01 08:26:00 | | | 2.75 | 17850.0 | United Kingdom | |
| 3 2010-12-01 08:26:00 | | | 3.39 | 17850.0 | United Kingdom | |
| 4 2010-12-01 08:26:00 | | | 3.39 | 17850.0 | United Kingdom | |

[5]:

data.describe()

|  |  |  |  |
| --- | --- | --- | --- |
| [5]: | Quantity | UnitPrice | CustomerID |
| count | 541909.000000 | 541909.000000 | 406829.000000 |
| mean | 9.552250 | 4.611114 | 15287.690570 |
| std | 218.081158 | 96.759853 | 1713.600303 |
| min | -80995.000000 | -11062.060000 | 12346.000000 |
| 25% | 1.000000 | 1.250000 | 13953.000000 |
| 50% | 3.000000 | 2.080000 | 15152.000000 |
| 75% | 10.000000 | 4.130000 | 16791.000000 |
| max | 80995.000000 | 38970.000000 | 18287.000000 |

[6]:

data.info()

<class 'pandas.core.frame.DataFrame'> RangeIndex: 541909 entries, 0 to 541908 Data columns (total 8 columns):

# Column Non-Null Count Dtype

[7]:

1. InvoiceNo 541909 non-null object
2. StockCode 541909 non-null object
3. Description 540455 non-null object
4. Quantity 541909 non-null int64
5. InvoiceDate 541909 non-null datetime64[ns]
6. UnitPrice 541909 non-null float64
7. CustomerID 406829 non-null float64
8. Country 541909 non-null object

dtypes: datetime64[ns](1), float64(2), int64(1), object(4) memory usage: 33.1+ MB

*# Check for missing data and formulate an apt strategy to treat them.*

typecol = ['Categorical', 'Categorical','Categorical', 'Discrete', 'Date',␣

*‹→*'Continuous' , 'Categorical' , 'Categorical' ]

missingdf = pd.DataFrame({'Columns' : data.columns.to\_list(), 'Type of data' :␣

*‹→*typecol, 'No of missing data' : data.isna().sum()})

**def** highlight\_max(s): is\_max = s

**return** ['background-color: pink' **if** v **else** '' **for** v **in** is\_max] missingdf.style.apply(highlight\_max, subset = ['No of missing data']) missingdf.style.hide\_index()

1. : <pandas.io.formats.style.Styler at 0x7fc4ce8af950> [8]:

data.drop(data[data['CustomerID'].isna()].index,inplace=**True**) data.reset\_index(drop=**True**)

1. : InvoiceNo StockCode Description Quantity \

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 | 536365 | 85123A | WHITE HANGING HEART T-LIGHT HOLDER | | 6 |
| 1 | 536365 | 71053 | WHITE METAL LANTERN | | 6 |
| 2 | 536365 | 84406B | CREAM CUPID HEARTS COAT HANGER | | 8 |
| 3 | 536365 | 84029G | KNITTED UNION FLAG HOT WATER BOTTLE | | 6 |
| 4  …  406824 | 536365  …  581587 | 84029E  …  22613 | RED WOOLLY HOTTIE WHITE HEART.  … …  PACK OF 20 SPACEBOY NAPKINS | | 6  12 |
| 406825 | 581587 | 22899 | CHILDREN'S APRON DOLLY GIRL | | 6 |
| 406826 | 581587 | 23254 | CHILDRENS CUTLERY DOLLY GIRL | | 4 |
| 406827 | 581587 | 23255 | CHILDRENS CUTLERY CIRCUS PARADE | | 4 |
| 406828 | 581587 | 22138 | BAKING SET 9 PIECE RETROSPOT | | 3 |
|  | InvoiceDate | | UnitPrice | CustomerID Country | |
| 0 | 2010-12-01 08:26:00 | | 2.55 | 17850.0 United Kingdom | |
| 1 | 2010-12-01 08:26:00 | | 3.39 | 17850.0 United Kingdom | |
| 2 | 2010-12-01 08:26:00 | | 2.75 | 17850.0 United Kingdom | |
| 3 | 2010-12-01 08:26:00 | | 3.39 | 17850.0 United Kingdom | |
| 4 | 2010-12-01 08:26:00 | | 3.39 | 17850.0 United Kingdom | |

… … … … …

|  |  |  |  |
| --- | --- | --- | --- |
| 406824 2011-12-09 12:50:00 | 0.85 | 12680.0 | France |
| 406825 2011-12-09 12:50:00 | 2.10 | 12680.0 | France |
| 406826 2011-12-09 12:50:00 | 4.15 | 12680.0 | France |
| 406827 2011-12-09 12:50:00 | 4.15 | 12680.0 | France |
| 406828 2011-12-09 12:50:00  [406829 rows x 8 columns] | 4.95 | 12680.0 | France |

[9]:

data.drop(data[data['Description'].isna()].index,inplace=**True**) data.reset\_index(drop=**True**)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [9]: |  | InvoiceNo | StockCode | Description | | | | | Quantity | \ |
|  | 0 | 536365 | 85123A | WHITE HANGING HEART T-LIGHT HOLDER | | | | | 6 |  |
|  | 1 | 536365 | 71053 | WHITE METAL LANTERN | | | | | 6 |  |
|  | 2 | 536365 | 84406B | CREAM CUPID HEARTS COAT HANGER | | | | | 8 |  |
|  | 3 | 536365 | 84029G | KNITTED UNION FLAG HOT WATER BOTTLE | | | | | 6 |  |
|  | 4  …  406824 | 536365  …  581587 | 84029E  …  22613 | RED WOOLLY HOTTIE WHITE HEART.  … …  PACK OF 20 SPACEBOY NAPKINS | | | | | 6  12 |  |
|  | 406825 | 581587 | 22899 | CHILDREN'S APRON DOLLY GIRL | | | | | 6 |  |
|  | 406826 | 581587 | 23254 | CHILDRENS CUTLERY DOLLY GIRL | | | | | 4 |  |
|  | 406827 | 581587 | 23255 | CHILDRENS CUTLERY CIRCUS PARADE | | | | | 4 |  |
|  | 406828 | 581587 | 22138 | BAKING SET 9 PIECE RETROSPOT | | | | | 3 |  |
| InvoiceDate UnitPrice CustomerID Country | | | | | | | | | | |
| 0 | | 2010-12-01 | 08:26:00 |  | 2.55 |  | 17850.0 | United Kingdom | | |
| 1 | | 2010-12-01 | 08:26:00 |  | 3.39 |  | 17850.0 | United Kingdom | | |
| 2 | | 2010-12-01 | 08:26:00 |  | 2.75 |  | 17850.0 | United Kingdom | | |
| 3 | | 2010-12-01 | 08:26:00 |  | 3.39 |  | 17850.0 | United Kingdom | | |
| 4  …  406824 | | 2010-12-01  2011-12-09 | 08:26:00  …  12:50:00 | … | 3.39  0.85 | … | 17850.0  12680.0 | United Kingdom  …  France | | |
| 406825 | | 2011-12-09 | 12:50:00 |  | 2.10 |  | 12680.0 | France | | |
| 406826 | | 2011-12-09 | 12:50:00 |  | 4.15 |  | 12680.0 | France | | |
| 406827 | | 2011-12-09 | 12:50:00 |  | 4.15 |  | 12680.0 | France | | |
| 406828 | | 2011-12-09 | 12:50:00 |  | 4.95 |  | 12680.0 | France | | |

[406829 rows x 8 columns]

[10]:

*# Remove duplicate data records* data.drop\_duplicates(inplace=**True**) data.reset\_index(drop=**True**)

[10]: InvoiceNo StockCode Description Quantity \

* 1. 536365 85123A WHITE HANGING HEART T-LIGHT HOLDER 6
  2. 536365 71053 WHITE METAL LANTERN 6

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 2 | 536365 | 84406B | CREAM CUPID HEARTS COAT HANGER | | | | | 8 |
| 3 | 536365 | 84029G | KNITTED UNION FLAG HOT WATER BOTTLE | | | | | 6 |
| 4  …  401599 | 536365  …  581587 | 84029E  …  22613 | RED WOOLLY HOTTIE WHITE HEART.  … …  PACK OF 20 SPACEBOY NAPKINS | | | | | 6  12 |
| 401600 | 581587 | 22899 | CHILDREN'S APRON DOLLY GIRL | | | | | 6 |
| 401601 | 581587 | 23254 | CHILDRENS CUTLERY DOLLY GIRL | | | | | 4 |
| 401602 | 581587 | 23255 | CHILDRENS CUTLERY CIRCUS PARADE | | | | | 4 |
| 401603 | 581587 | 22138 | BAKING SET 9 PIECE RETROSPOT | | | | | 3 |
| InvoiceDate UnitPrice CustomerID Country | | | | | | | | |
| 0 | 2010-12-01 | 08:26:00 |  | 2.55 |  | 17850.0 | United Kingdom | |
| 1 | 2010-12-01 | 08:26:00 |  | 3.39 |  | 17850.0 | United Kingdom | |
| 2 | 2010-12-01 | 08:26:00 |  | 2.75 |  | 17850.0 | United Kingdom | |
| 3 | 2010-12-01 | 08:26:00 |  | 3.39 |  | 17850.0 | United Kingdom | |
| 4  …  401599 | 2010-12-01  2011-12-09 | 08:26:00  …  12:50:00 | … | 3.39  0.85 | … | 17850.0  12680.0 | United Kingdom  …  France | |
| 401600 | 2011-12-09 | 12:50:00 |  | 2.10 |  | 12680.0 | France | |
| 401601 | 2011-12-09 | 12:50:00 |  | 4.15 |  | 12680.0 | France | |
| 401602 | 2011-12-09 | 12:50:00 |  | 4.15 |  | 12680.0 | France | |
| 401603 | 2011-12-09 | 12:50:00 |  | 4.95 |  | 12680.0 | France | |

[401604 rows x 8 columns]

[11]:

*# Remove the data for last month of 2011 and the values are only for 9 days*

**import datetime def** getmonth(u):

**return** datetime.datetime(u.year,u.month,u.day) print('Number of records before last month transactions') print(len(data))

*# create invoiceMonth* data['invoiceMonth']=data['InvoiceDate'].apply(getmonth) data[data['invoiceMonth']>datetime.datetime(2011,11,30)] data.drop(data[data['invoiceMonth'] > datetime.datetime(2011,11,30)].index,␣

*‹→*inplace = **True**)

data.reset\_index(drop=**True**)

print('Number of records after dropping last month transactions') print(len(data))

[12]:

Number of records before last month transactions 401604

Number of records after dropping last month transactions 384222

data.Country.value\_counts(normalize=**True**).head(10).mul(100).round(2).

*‹→*astype(str)+'%'

1. : United Kingdom 88.73%

Germany 2.38%

France 2.12%

EIRE 1.86%

Spain 0.64%

Netherlands 0.59%

Belgium 0.51%

Switzerland 0.49%

Portugal 0.36%

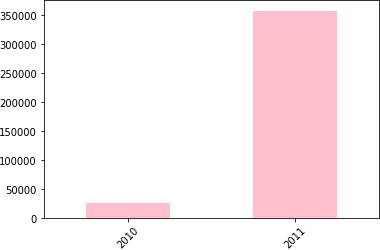
Australia 0.33% Name: Country, dtype: object

[13]:

data.InvoiceDate.dt.year.value\_counts(sort=**False**).

*‹→*plot(kind='bar',rot=45,color='pink')

1. : <AxesSubplot:>



[14]:

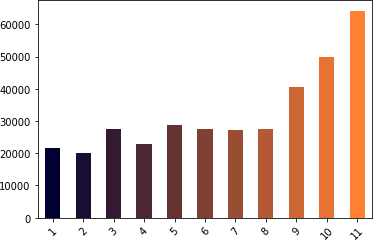
*# Visualize 2011 customer trend on monthly basis*

mycols=[(x/10,x/20.0,0.2) **for** x **in** range(len(data[data.InvoiceDate.dt.

*‹→*year==2011].InvoiceDate.dt.month.value\_counts(sort=**False**)))]

data[data.InvoiceDate.dt.year==2011].InvoiceDate.dt.month.

*‹→*value\_counts(sort=**False**).plot(kind='bar', rot=45, color=mycols);



[15]:

data.head()

1. : InvoiceNo StockCode Description Quantity \

0 536365 85123A WHITE HANGING HEART T-LIGHT HOLDER 6

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 536365 | 71053 | WHITE METAL LANTERN | | | 6 |
| 2 | 536365 | 84406B | CREAM CUPID HEARTS COAT HANGER | | | 8 |
| 3 | 536365 | 84029G | KNITTED UNION FLAG HOT WATER BOTTLE | | | 6 |
| 4 | 536365 | 84029E | RED WOOLLY HOTTIE WHITE HEART. | | | 6 |
| InvoiceDate | | | UnitPrice | CustomerID | Country invoiceMonth | |
| 0 2010-12-01 08:26:00 | | | 2.55 | 17850.0 | United Kingdom 2010-12-01 | |
| 1 2010-12-01 08:26:00 | | | 3.39 | 17850.0 | United Kingdom 2010-12-01 | |
| 2 2010-12-01 08:26:00 | | | 2.75 | 17850.0 | United Kingdom 2010-12-01 | |
| 3 2010-12-01 08:26:00 | | | 3.39 | 17850.0 | United Kingdom 2010-12-01 | |
| 4 2010-12-01 08:26:00 | | | 3.39 | 17850.0 | United Kingdom 2010-12-01 | |

[16]:

data.tail()

1. : InvoiceNo StockCode Description Quantity \ 516379 C579886 22197 POPCORN HOLDER -1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 516380 | C579886 | 23146 | TRIPLE HOOK ANTIQUE IVORY ROSE | -1 |
| 516381 | C579887 | 84946 | ANTIQUE SILVER T-LIGHT GLASS | -1 |
| 516382 | C579887 | 85048 | 15CM CHRISTMAS GLASS BALL 20 LIGHTS | -1 |
| 516383 | C579887 | 23490 | T-LIGHT HOLDER HANGING LOVE BIRD | -3 |

InvoiceDate UnitPrice CustomerID Country invoiceMonth

|  |  |  |
| --- | --- | --- |
| 516379 2011-11-30 17:39:00 | 0.85 | 15676.0 United Kingdom 2011-11-30 |
| 516380 2011-11-30 17:39:00 | 3.29 | 15676.0 United Kingdom 2011-11-30 |
| 516381 2011-11-30 17:42:00 | 1.25 | 16717.0 United Kingdom 2011-11-30 |
| 516382 2011-11-30 17:42:00 | 7.95 | 16717.0 United Kingdom 2011-11-30 |
| 516383 2011-11-30 17:42:00 | 3.75 | 16717.0 United Kingdom 2011-11-30 |

[17]:

*# total price = Quantity \* Unit Price* data['TotPrice']=data.Quantity\*data.UnitPrice *# sort values of total price*

df=data.TotPrice.sort\_values(ascending=**False**).head(10).to\_frame().style.

*‹→*hide\_index()

df

1. : <pandas.io.formats.style.Styler at 0x7fc4d1bd3bd0> [18]:

desc = data.sort\_values(by='TotPrice', ascending=**False**)['Description'].head(10) price = data.sort\_values(by='TotPrice', ascending=**False**)['TotPrice'].head(10)

[19]:

price

[19]: 61619 77183.60

222680 38970.00

173382 8142.75

348325 7144.72

160546 6539.40

52711 6539.40

421601 4992.00

52709 4921.50

160542 4632.00

52772 4522.50

Name: TotPrice, dtype: float64

[20]:

data

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| [20]: | InvoiceNo | StockCode | Description | Quantity | \ |
|  | 0 536365 | 85123A | WHITE HANGING HEART T-LIGHT HOLDER | 6 |  |
|  | 1 536365 | 71053 | WHITE METAL LANTERN | 6 |  |
|  | 2 536365 | 84406B | CREAM CUPID HEARTS COAT HANGER | 8 |  |
|  | 3 536365 | 84029G | KNITTED UNION FLAG HOT WATER BOTTLE | 6 |  |
|  | 4 536365 | 84029E | RED WOOLLY HOTTIE WHITE HEART. | 6 |  |
|  | … … | … | … … |  |  |
|  | 516379 C579886 | 22197 | POPCORN HOLDER | -1 |  |
|  | 516380 C579886 | 23146 | TRIPLE HOOK ANTIQUE IVORY ROSE | -1 |  |
|  | 516381 C579887 | 84946 | ANTIQUE SILVER T-LIGHT GLASS | -1 |  |
|  | 516382 C579887 | 85048 | 15CM CHRISTMAS GLASS BALL 20 LIGHTS | -1 |  |
|  | 516383 C579887 | 23490 | T-LIGHT HOLDER HANGING LOVE BIRD | -3 |  |

InvoiceDate UnitPrice CustomerID Country \

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 2010-12-01 | 08:26:00 |  | 2.55 |  | 17850.0 | United | Kingdom |
| 1 | 2010-12-01 | 08:26:00 |  | 3.39 |  | 17850.0 | United | Kingdom |
| 2 | 2010-12-01 | 08:26:00 |  | 2.75 |  | 17850.0 | United | Kingdom |
| 3 | 2010-12-01 | 08:26:00 |  | 3.39 |  | 17850.0 | United | Kingdom |
| 4 | 2010-12-01 | 08:26:00 |  | 3.39 |  | 17850.0 | United | Kingdom |
| … |  | … | … |  | … |  | … |  |
| 516379 | 2011-11-30 | 17:39:00 |  | 0.85 |  | 15676.0 | United | Kingdom |
| 516380 | 2011-11-30 | 17:39:00 |  | 3.29 |  | 15676.0 | United | Kingdom |
| 516381 | 2011-11-30 | 17:42:00 |  | 1.25 |  | 16717.0 | United | Kingdom |
| 516382 | 2011-11-30 | 17:42:00 |  | 7.95 |  | 16717.0 | United | Kingdom |
| 516383 | 2011-11-30 | 17:42:00 |  | 3.75 |  | 16717.0 | United | Kingdom |

|  |  |  |
| --- | --- | --- |
|  | invoiceMonth | TotPrice |
| 0 | 2010-12-01 | 15.30 |
| 1 | 2010-12-01 | 20.34 |
| 2 | 2010-12-01 | 22.00 |
| 3 | 2010-12-01 | 20.34 |
| 4 | 2010-12-01 | 20.34 |
| … | … | … |
| 516379 | 2011-11-30 | -0.85 |
| 516380 | 2011-11-30 | -3.29 |
| 516381 | 2011-11-30 | -1.25 |
| 516382 | 2011-11-30 | -7.95 |
| 516383 | 2011-11-30 | -11.25 |

[384222 rows x 10 columns]

[21]:

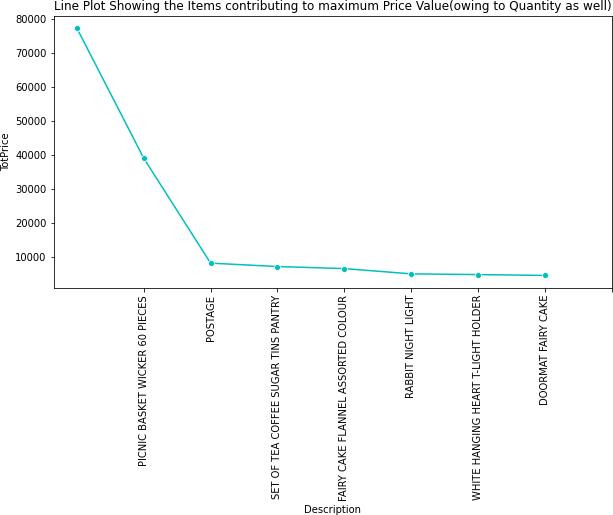
**import seaborn as sns**

**import matplotlib.pyplot as plt**

plt.figure(figsize=(10,5)) sns.lineplot(x=desc,y=price,marker='o',color='c',).set\_title('Line Plot Showing␣

*‹→*the Items contributing to maximum Price Value(owing to Quantity as well)')

plt.xticks(range(1,9),rotation=90) plt.show();



[22]:

print("First bussiness transaction date is **{}**".format(data.InvoiceDate.min())) print("Final bussiness transaction date is **{}**".format(data.InvoiceDate.max())) monthly\_gross=data[data.InvoiceDate.dt.year==2011].groupby(data.InvoiceDate.dt.

*‹→*month).TotPrice.sum()

dat = pd.DataFrame(monthly\_gross) dat

First bussiness transaction date is 2010-12-01 08:26:00 Final bussiness transaction date is 2011-11-30 17:42:00

[22]: TotPrice

InvoiceDate

|  |  |
| --- | --- |
| 1 | 473731.900 |
| 2 | 435534.070 |
| 3 | 578576.210 |
| 4 | 425222.671 |
| 5 | 647011.670 |
| 6 | 606862.520 |

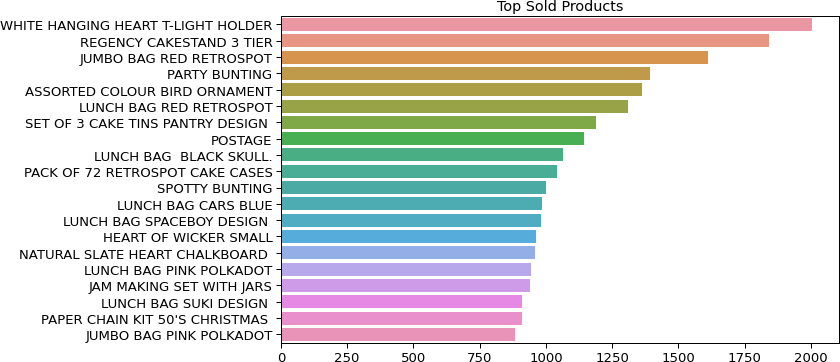
|  |  |
| --- | --- |
| 7 | 573112.321 |
| 8 | 615078.090 |
| 9 | 929356.232 |
| 10 | 973306.380 |
| 11 | 1126815.070 |

[23]:

topproducts = data["Description"].value\_counts()[:20] plt.figure(figsize=(10,6)) sns.set\_context("paper",font\_scale=1.5)

sns.barplot(y = topproducts.index,

x = topproducts.values) plt.title("Top Sold Products") plt.show()



[24]:

*# Analyze outlier by Interquartile range*

**def** outlierDetects(Datacolumns): sorted(Datacolumns) Q1,Q3=np.percentile(Datacolumns,[25,75])

IQR=Q3-Q1

*# cal lower range* lR=Q1-(1.5\*IQR) uR=Q3+(1.5\*IQR)

**return** lR,uR

**def** outlier\_removal(drop\_col=**False**):

**for** col **in** data.columns[[3,5,8]]: lowrange,uprange = outlierDetects(data[col])

**if not** data[(data[col] > uprange) | (data[col] < lowrange)].empty: print("detect outliers for columns **%r**" %**col**)

[25]:

*# Cohort analysis* cohort=data.copy() cohort

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [25]: |  | InvoiceNo | StockCode | Description | | | | | | Quantity | \ |
|  | 0 | 536365 | 85123A | WHITE HANGING HEART T-LIGHT HOLDER | | | | | | 6 |  |
|  | 1 | 536365 | 71053 | WHITE METAL LANTERN | | | | | | 6 |  |
|  | 2 | 536365 | 84406B | CREAM CUPID HEARTS COAT HANGER | | | | | | 8 |  |
|  | 3 | 536365 | 84029G | KNITTED UNION FLAG HOT WATER BOTTLE | | | | | | 6 |  |
|  | 4  …  516379 | 536365  …  C579886 | 84029E  …  22197 | RED WOOLLY HOTTIE WHITE HEART.  … …  POPCORN HOLDER | | | | | | 6  -1 |  |
|  | 516380 | C579886 | 23146 | TRIPLE HOOK ANTIQUE IVORY ROSE | | | | | | -1 |  |
|  | 516381 | C579887 | 84946 | ANTIQUE SILVER T-LIGHT GLASS | | | | | | -1 |  |
|  | 516382 | C579887 | 85048 | 15CM CHRISTMAS GLASS BALL 20 LIGHTS | | | | | | -1 |  |
|  | 516383 | C579887 | 23490 | T-LIGHT HOLDER HANGING LOVE BIRD | | | | | | -3 |  |
| InvoiceDate UnitPrice CustomerID Country \ | | | | | | | | | | | |
| 0 | | 2010-12-01 | 08:26:00 |  | 2.55 |  | 17850.0 | United | Kingdom | | |
| 1 | | 2010-12-01 | 08:26:00 |  | 3.39 |  | 17850.0 | United | Kingdom | | |
| 2 | | 2010-12-01 | 08:26:00 |  | 2.75 |  | 17850.0 | United | Kingdom | | |
| 3 | | 2010-12-01 | 08:26:00 |  | 3.39 |  | 17850.0 | United | Kingdom | | |
| 4  …  516379 | | 2010-12-01  2011-11-30 | 08:26:00  …  17:39:00 | … | 3.39  0.85 | … | 17850.0  15676.0 | United  …  United | Kingdom  Kingdom | | |
| 516380 | | 2011-11-30 | 17:39:00 |  | 3.29 |  | 15676.0 | United | Kingdom | | |
| 516381 | | 2011-11-30 | 17:42:00 |  | 1.25 |  | 16717.0 | United | Kingdom | | |
| 516382 | | 2011-11-30 | 17:42:00 |  | 7.95 |  | 16717.0 | United | Kingdom | | |
| 516383 | | 2011-11-30 | 17:42:00 |  | 3.75 |  | 16717.0 | United | Kingdom | | |

|  |  |  |
| --- | --- | --- |
|  | invoiceMonth | TotPrice |
| 0 | 2010-12-01 | 15.30 |
| 1 | 2010-12-01 | 20.34 |
| 2 | 2010-12-01 | 22.00 |
| 3 | 2010-12-01 | 20.34 |
| 4 | 2010-12-01 | 20.34 |
| … | … | … |
| 516379 | 2011-11-30 | -0.85 |
| 516380 | 2011-11-30 | -3.29 |
| 516381 | 2011-11-30 | -1.25 |
| 516382 | 2011-11-30 | -7.95 |
| 516383 | 2011-11-30 | -11.25 |

[384222 rows x 10 columns]

[26]:

**import datetime def** getmonth(x):

**return** datetime.datetime(x.year,x.month,1)

cohort['invoiceMonth']=cohort['InvoiceDate'].apply(getmonth) grouping=cohort.groupby('CustomerID')['invoiceMonth'] cohort['cohortMon'] = grouping.transform('min')

[27]:

cohort

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [27]: |  | InvoiceNo | StockCode | Description | | | | | | Quantity | \ |
|  | 0 | 536365 | 85123A | WHITE HANGING HEART T-LIGHT HOLDER | | | | | | 6 |  |
|  | 1 | 536365 | 71053 | WHITE METAL LANTERN | | | | | | 6 |  |
|  | 2 | 536365 | 84406B | CREAM CUPID HEARTS COAT HANGER | | | | | | 8 |  |
|  | 3 | 536365 | 84029G | KNITTED UNION FLAG HOT WATER BOTTLE | | | | | | 6 |  |
|  | 4  …  516379 | 536365  …  C579886 | 84029E  …  22197 | RED WOOLLY HOTTIE WHITE HEART.  … …  POPCORN HOLDER | | | | | | 6  -1 |  |
|  | 516380 | C579886 | 23146 | TRIPLE HOOK ANTIQUE IVORY ROSE | | | | | | -1 |  |
|  | 516381 | C579887 | 84946 | ANTIQUE SILVER T-LIGHT GLASS | | | | | | -1 |  |
|  | 516382 | C579887 | 85048 | 15CM CHRISTMAS GLASS BALL 20 LIGHTS | | | | | | -1 |  |
|  | 516383 | C579887 | 23490 | T-LIGHT HOLDER HANGING LOVE BIRD | | | | | | -3 |  |
| InvoiceDate UnitPrice CustomerID Country \ | | | | | | | | | | | |
| 0 | | 2010-12-01 | 08:26:00 |  | 2.55 |  | 17850.0 | United | Kingdom | | |
| 1 | | 2010-12-01 | 08:26:00 |  | 3.39 |  | 17850.0 | United | Kingdom | | |
| 2 | | 2010-12-01 | 08:26:00 |  | 2.75 |  | 17850.0 | United | Kingdom | | |
| 3 | | 2010-12-01 | 08:26:00 |  | 3.39 |  | 17850.0 | United | Kingdom | | |
| 4  …  516379 | | 2010-12-01  2011-11-30 | 08:26:00  …  17:39:00 | … | 3.39  0.85 | … | 17850.0  15676.0 | United  …  United | Kingdom  Kingdom | | |
| 516380 | | 2011-11-30 | 17:39:00 |  | 3.29 |  | 15676.0 | United | Kingdom | | |
| 516381 | | 2011-11-30 | 17:42:00 |  | 1.25 |  | 16717.0 | United | Kingdom | | |
| 516382 | | 2011-11-30 | 17:42:00 |  | 7.95 |  | 16717.0 | United | Kingdom | | |
| 516383 | | 2011-11-30 | 17:42:00 |  | 3.75 |  | 16717.0 | United | Kingdom | | |

|  |  |  |  |
| --- | --- | --- | --- |
|  | invoiceMonth | TotPrice | cohortMon |
| 0 | 2010-12-01 | 15.30 | 2010-12-01 |
| 1 | 2010-12-01 | 20.34 | 2010-12-01 |
| 2 | 2010-12-01 | 22.00 | 2010-12-01 |
| 3 | 2010-12-01 | 20.34 | 2010-12-01 |
| 4 | 2010-12-01 | 20.34 | 2010-12-01 |
| … | … | … | … |
| 516379 | 2011-11-01 | -0.85 | 2011-03-01 |
| 516380 | 2011-11-01 | -3.29 | 2011-03-01 |
| 516381 | 2011-11-01 | -1.25 | 2010-12-01 |
| 516382 | 2011-11-01 | -7.95 | 2010-12-01 |

[28]:

**def** get\_date\_int(dat,column): year = dat[column].dt.year month = dat[column].dt.month **return** year, month

*# get the integer from Invoice Month and Cohort Month* invoice\_year, invoice\_month = get\_date\_int(cohort,'invoiceMonth') cohort\_year, cohort\_month = get\_date\_int(cohort,'cohortMon')

516383 2011-11-01 -11.25 2010-12-01

[384222 rows x 11 columns]

[29]:

print ("Unique terms for Cohort Year is **{}** " .format(cohort\_year.unique())) print ("Unique terms for Cohort Month is **{}** " .format(cohort\_month.unique()))

[30]:

Unique terms for Cohort Year is [2010 2011]

Unique terms for Cohort Month is [12 1 2 3 4 5 6 7 8 9 10 11]

diffYear=invoice\_year-cohort\_year diffMonth=invoice\_month-cohort\_month

*# Extract the difference in months from all previous ones*

cohort['CohortIndex'] = diffYear\*12+diffMonth+1

[31]:

cohort['CohortIndex'].unique() cohort

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| [31]: | InvoiceNo | StockCode | Description | Quantity | \ |
|  | 0 536365 | 85123A | WHITE HANGING HEART T-LIGHT HOLDER | 6 |  |
|  | 1 536365 | 71053 | WHITE METAL LANTERN | 6 |  |
|  | 2 536365 | 84406B | CREAM CUPID HEARTS COAT HANGER | 8 |  |
|  | 3 536365 | 84029G | KNITTED UNION FLAG HOT WATER BOTTLE | 6 |  |
|  | 4 536365 | 84029E | RED WOOLLY HOTTIE WHITE HEART. | 6 |  |
|  | … … | … | … … |  |  |
|  | 516379 C579886 | 22197 | POPCORN HOLDER | -1 |  |
|  | 516380 C579886 | 23146 | TRIPLE HOOK ANTIQUE IVORY ROSE | -1 |  |
|  | 516381 C579887 | 84946 | ANTIQUE SILVER T-LIGHT GLASS | -1 |  |
|  | 516382 C579887 | 85048 | 15CM CHRISTMAS GLASS BALL 20 LIGHTS | -1 |  |
|  | 516383 C579887 | 23490 | T-LIGHT HOLDER HANGING LOVE BIRD | -3 |  |

InvoiceDate UnitPrice CustomerID Country \ 0 2010-12-01 08:26:00 2.55 17850.0 United Kingdom

1 2010-12-01 08:26:00 3.39 17850.0 United Kingdom

2 2010-12-01 08:26:00 2.75 17850.0 United Kingdom

3 2010-12-01 08:26:00 3.39 17850.0 United Kingdom

4 2010-12-01 08:26:00 3.39 17850.0 United Kingdom

… … … … …

516379 2011-11-30 17:39:00 0.85 15676.0 United Kingdom

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 516380 | 2011-11-30 17:39:00 | | 3.29 | 15676.0 United | Kingdom |
| 516381 | 2011-11-30 17:42:00 | | 1.25 | 16717.0 United | Kingdom |
| 516382 | 2011-11-30 17:42:00 | | 7.95 | 16717.0 United | Kingdom |
| 516383 | 2011-11-30 17:42:00 | | 3.75 | 16717.0 United | Kingdom |
|  | invoiceMonth | TotPrice | cohortMon | CohortIndex | |
| 0 | 2010-12-01 | 15.30 | 2010-12-01 | 1 | |
| 1 | 2010-12-01 | 20.34 | 2010-12-01 | 1 | |
| 2 | 2010-12-01 | 22.00 | 2010-12-01 | 1 | |
| 3 | 2010-12-01 | 20.34 | 2010-12-01 | 1 | |
| 4 | 2010-12-01 | 20.34 | 2010-12-01 | 1 | |
| … | … | … | … | … | |
| 516379 | 2011-11-01 | -0.85 | 2011-03-01 | 9 | |
| 516380 | 2011-11-01 | -3.29 | 2011-03-01 | 9 | |
| 516381 | 2011-11-01 | -1.25 | 2010-12-01 | 12 | |
| 516382 | 2011-11-01 | -7.95 | 2010-12-01 | 12 | |
| 516383 | 2011-11-01 | -11.25 | 2010-12-01 | 12 | |

[384222 rows x 12 columns]

[32]:

groups = cohort.groupby(['cohortMon','CohortIndex'])

[33]:

groups

[33]: <pandas.core.groupby.generic.DataFrameGroupBy object at 0x7fc4d1803b50> [34]:

*# retention rate of active customers*

**import pandas as pd**

cohort\_data=groups['CustomerID'].apply(pd.Series.nunique).reset\_index()

[35]:

cohort\_data['CustomerID']

[35]: 0 948

1 362

2 317

3 367

4 341

…

73 89

74 97

75 352

76 93

77 321

Name: CustomerID, Length: 78, dtype: int64

[36]:

cohort\_counts = cohort\_data.pivot(index='cohortMon',␣

*‹→*columns='CohortIndex',values='CustomerID')

[37]:

cohort\_counts

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [37]: | CohortIndex cohortMon | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | \ |
|  | 2010-12-01 | 948.0 | 362.0 | 317.0 | 367.0 | 341.0 | 376.0 | 360.0 | 336.0 | 336.0 |  |
|  | 2011-01-01 | 421.0 | 101.0 | 119.0 | 102.0 | 138.0 | 126.0 | 110.0 | 108.0 | 131.0 |  |
|  | 2011-02-01 | 380.0 | 94.0 | 73.0 | 106.0 | 102.0 | 94.0 | 97.0 | 107.0 | 98.0 |  |
|  | 2011-03-01 | 440.0 | 84.0 | 112.0 | 96.0 | 102.0 | 78.0 | 116.0 | 105.0 | 127.0 |  |
|  | 2011-04-01 | 299.0 | 68.0 | 66.0 | 63.0 | 62.0 | 71.0 | 69.0 | 78.0 | NaN |  |
|  | 2011-05-01 | 279.0 | 66.0 | 48.0 | 48.0 | 60.0 | 68.0 | 74.0 | NaN | NaN |  |
|  | 2011-06-01 | 235.0 | 49.0 | 44.0 | 64.0 | 58.0 | 79.0 | NaN | NaN | NaN |  |
|  | 2011-07-01 | 191.0 | 40.0 | 39.0 | 44.0 | 52.0 | NaN | NaN | NaN | NaN |  |
|  | 2011-08-01 | 167.0 | 42.0 | 42.0 | 42.0 | NaN | NaN | NaN | NaN | NaN |  |
|  | 2011-09-01 | 298.0 | 89.0 | 97.0 | NaN | NaN | NaN | NaN | NaN | NaN |  |
|  | 2011-10-01 | 352.0 | 93.0 | NaN | NaN | NaN | NaN | NaN | NaN | NaN |  |
|  | 2011-11-01 | 321.0 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |  |
|  | CohortIndex cohortMon 2010-12-01 | 10  374.0 | 11  354.0 | 12  474.0 |  |  |  |  |  |  |  |
|  | 2011-01-01 | 146.0 | 155.0 | NaN |  |  |  |  |  |  |  |
|  | 2011-02-01 | 119.0 | NaN | NaN |  |  |  |  |  |  |  |
|  | 2011-03-01 | NaN | NaN | NaN |  |  |  |  |  |  |  |
|  | 2011-04-01 | NaN | NaN | NaN |  |  |  |  |  |  |  |
|  | 2011-05-01 | NaN | NaN | NaN |  |  |  |  |  |  |  |
|  | 2011-06-01 | NaN | NaN | NaN |  |  |  |  |  |  |  |
|  | 2011-07-01 | NaN | NaN | NaN |  |  |  |  |  |  |  |
|  | 2011-08-01 | NaN | NaN | NaN |  |  |  |  |  |  |  |
|  | 2011-09-01 | NaN | NaN | NaN |  |  |  |  |  |  |  |
|  | 2011-10-01 | NaN | NaN | NaN |  |  |  |  |  |  |  |
|  | 2011-11-01 | NaN | NaN | NaN |  |  |  |  |  |  |  |

[38]:

CohortSizes = cohort\_counts.iloc[:,0]

[39]:

CohortSizes

|  |  |  |
| --- | --- | --- |
| [39]: | cohortMon |  |
|  | 2010-12-01 | 948.0 |
|  | 2011-01-01 | 421.0 |
|  | 2011-02-01 | 380.0 |
|  | 2011-03-01 | 440.0 |
|  | 2011-04-01 | 299.0 |
|  | 2011-05-01 | 279.0 |
|  | 2011-06-01 | 235.0 |
|  | 2011-07-01 | 191.0 |
|  | 2011-08-01 | 167.0 |
|  | 2011-09-01 | 298.0 |

2011-10-01 352.0

2011-11-01 321.0

Name: 1, dtype: float64

[40]:

retentionrate = cohort\_counts.divide(CohortSizes, axis=0)\*100

[41]:

retentionrate

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [41]: | CohortIndex cohortMon | 1 | 2 | 3 | 4 | 5 | 6 | \ |
|  | 2010-12-01 | 100.0 | 38.185654 | 33.438819 | 38.713080 | 35.970464 | 39.662447 |  |
|  | 2011-01-01 | 100.0 | 23.990499 | 28.266033 | 24.228029 | 32.779097 | 29.928741 |  |
|  | 2011-02-01 | 100.0 | 24.736842 | 19.210526 | 27.894737 | 26.842105 | 24.736842 |  |
|  | 2011-03-01 | 100.0 | 19.090909 | 25.454545 | 21.818182 | 23.181818 | 17.727273 |  |
|  | 2011-04-01 | 100.0 | 22.742475 | 22.073579 | 21.070234 | 20.735786 | 23.745819 |  |
|  | 2011-05-01 | 100.0 | 23.655914 | 17.204301 | 17.204301 | 21.505376 | 24.372760 |  |
|  | 2011-06-01 | 100.0 | 20.851064 | 18.723404 | 27.234043 | 24.680851 | 33.617021 |  |
|  | 2011-07-01 | 100.0 | 20.942408 | 20.418848 | 23.036649 | 27.225131 | NaN |  |
|  | 2011-08-01 | 100.0 | 25.149701 | 25.149701 | 25.149701 | NaN | NaN |  |
|  | 2011-09-01 | 100.0 | 29.865772 | 32.550336 | NaN | NaN | NaN |  |
|  | 2011-10-01 | 100.0 | 26.420455 | NaN | NaN | NaN | NaN |  |
|  | 2011-11-01 | 100.0 | NaN | NaN | NaN | NaN | NaN |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| CohortIndex | 7 | 8 | 9 | 10 | 11 | 12 |
| cohortMon |  |  |  |  |  |  |
| 2010-12-01 | 37.974684 | 35.443038 | 35.443038 | 39.451477 | 37.341772 | 50.0 |
| 2011-01-01 | 26.128266 | 25.653207 | 31.116390 | 34.679335 | 36.817102 | NaN |
| 2011-02-01 | 25.526316 | 28.157895 | 25.789474 | 31.315789 | NaN | NaN |
| 2011-03-01 | 26.363636 | 23.863636 | 28.863636 | NaN | NaN | NaN |
| 2011-04-01 | 23.076923 | 26.086957 | NaN | NaN | NaN | NaN |
| 2011-05-01 | 26.523297 | NaN | NaN | NaN | NaN | NaN |
| 2011-06-01 | NaN | NaN | NaN | NaN | NaN | NaN |
| 2011-07-01 | NaN | NaN | NaN | NaN | NaN | NaN |
| 2011-08-01 | NaN | NaN | NaN | NaN | NaN | NaN |
| 2011-09-01 | NaN | NaN | NaN | NaN | NaN | NaN |
| 2011-10-01 | NaN | NaN | NaN | NaN | NaN | NaN |
| 2011-11-01 | NaN | NaN | NaN | NaN | NaN | NaN |

[42]:

retentionrate.index = retentionrate.index.date

[43]:

retentionrate

[43]: CohortIndex 1 2 3 4 5 6 \

2010-12-01 100.0 38.185654 33.438819 38.713080 35.970464 39.662447

2011-01-01 100.0 23.990499 28.266033 24.228029 32.779097 29.928741

2011-02-01 100.0 24.736842 19.210526 27.894737 26.842105 24.736842

2011-03-01 100.0 19.090909 25.454545 21.818182 23.181818 17.727273

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 2011-04-01 | 100.0 22.742475 | 22.073579 | 21.070234 | 20.735786 | 23.745819 |
| 2011-05-01 | 100.0 23.655914 | 17.204301 | 17.204301 | 21.505376 | 24.372760 |
| 2011-06-01 | 100.0 20.851064 | 18.723404 | 27.234043 | 24.680851 | 33.617021 |
| 2011-07-01 | 100.0 20.942408 | 20.418848 | 23.036649 | 27.225131 | NaN |
| 2011-08-01 | 100.0 25.149701 | 25.149701 | 25.149701 | NaN | NaN |
| 2011-09-01 | 100.0 29.865772 | 32.550336 | NaN | NaN | NaN |
| 2011-10-01 | 100.0 26.420455 | NaN | NaN | NaN | NaN |
| 2011-11-01 | 100.0 NaN | NaN | NaN | NaN | NaN |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| CohortIndex | 7 | 8 | 9 | 10 | 11 | 12 |
| 2010-12-01 | 37.974684 | 35.443038 | 35.443038 | 39.451477 | 37.341772 | 50.0 |
| 2011-01-01 | 26.128266 | 25.653207 | 31.116390 | 34.679335 | 36.817102 | NaN |
| 2011-02-01 | 25.526316 | 28.157895 | 25.789474 | 31.315789 | NaN | NaN |
| 2011-03-01 | 26.363636 | 23.863636 | 28.863636 | NaN | NaN | NaN |
| 2011-04-01 | 23.076923 | 26.086957 | NaN | NaN | NaN | NaN |
| 2011-05-01 | 26.523297 | NaN | NaN | NaN | NaN | NaN |
| 2011-06-01 | NaN | NaN | NaN | NaN | NaN | NaN |
| 2011-07-01 | NaN | NaN | NaN | NaN | NaN | NaN |
| 2011-08-01 | NaN | NaN | NaN | NaN | NaN | NaN |
| 2011-09-01 | NaN | NaN | NaN | NaN | NaN | NaN |
| 2011-10-01 | NaN | NaN | NaN | NaN | NaN | NaN |
| 2011-11-01 | NaN | NaN | NaN | NaN | NaN | NaN |

[44]:

*# calculate average price metrics and analyze time cohort and others to*␣

*‹→differtiate shopping pattern across different cohorts*

cohortData = groups['UnitPrice'].mean()

[45]:

cohortData

float64

|  |  |  |
| --- | --- | --- |
| [45]: cohortMon | CohortIndex |  |
| 2010-12-01 | 1 | 3.216682 |
|  | 2 | 3.182040 |
|  | 3 | 3.207467 |
|  | 4 | 3.603758 |
|  | 5 | 2.937803  … |
| 2011-09-01 | 2 | 3.584834 |
|  | 3 | 2.957893 |
| 2011-10-01 | 1 | 4.053162 |
|  | 2 | 2.678140 |
| 2011-11-01 1  Name: UnitPrice, Length: | | 2.641554  78, dtype: |

[46]:

cohortData = cohortData.reset\_index()

[47]:

averagePrice = cohortData.pivot(index='cohortMon', columns= 'CohortIndex',␣

*‹→*values= 'UnitPrice')

[48]:

averagePrice

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [48]: | CohortIndex cohortMon | 1 | 2 | 3 | 4 | 5 | 6 | \ |
|  | 2010-12-01 | 3.216682 | 3.182040 | 3.207467 | 3.603758 | 2.937803 | 4.996508 |  |
|  | 2011-01-01 | 3.505492 | 3.653572 | 3.069534 | 8.439024 | 3.157803 | 3.172919 |  |
|  | 2011-02-01 | 3.355968 | 4.469638 | 4.824106 | 3.150045 | 2.987616 | 2.792577 |  |
|  | 2011-03-01 | 3.302802 | 4.990095 | 3.655094 | 3.289768 | 3.616562 | 2.758381 |  |
|  | 2011-04-01 | 3.431172 | 3.958074 | 3.300128 | 2.673439 | 3.028297 | 2.867185 |  |
|  | 2011-05-01 | 4.662054 | 3.243691 | 2.652761 | 3.167391 | 2.667158 | 2.495751 |  |
|  | 2011-06-01 | 10.490030 | 3.205283 | 3.343994 | 2.835952 | 2.553037 | 3.550657 |  |
|  | 2011-07-01 | 4.493676 | 3.480495 | 2.752121 | 2.701985 | 2.403989 | NaN |  |
|  | 2011-08-01 | 3.028246 | 5.425904 | 5.714033 | 7.046410 | NaN | NaN |  |
|  | 2011-09-01 | 3.235116 | 3.584834 | 2.957893 | NaN | NaN | NaN |  |
|  | 2011-10-01 | 4.053162 | 2.678140 | NaN | NaN | NaN | NaN |  |
|  | 2011-11-01 | 2.641554 | NaN | NaN | NaN | NaN | NaN |  |
| CohortIndex cohortMon 2010-12-01 | | 7  3.184572 | 8  3.235695 | 9  3.511560 | 10  3.035982 | 11  3.309705 | 12  2.835557 | |
| 2011-01-01 | | 2.918498 | 2.749649 | 2.641686 | 5.489040 | 2.886220 | NaN | |
| 2011-02-01 | | 2.812985 | 3.214380 | 2.894988 | 2.946092 | NaN | NaN | |
| 2011-03-01 | | 2.843273 | 2.809136 | 2.707846 | NaN | NaN | NaN | |
| 2011-04-01 | | 2.902668 | 2.812492 | NaN | NaN | NaN | NaN | |
| 2011-05-01 | | 2.615408 | NaN | NaN | NaN | NaN | NaN | |
| 2011-06-01 | | NaN | NaN | NaN | NaN | NaN | NaN | |
| 2011-07-01 | | NaN | NaN | NaN | NaN | NaN | NaN | |
| 2011-08-01 | | NaN | NaN | NaN | NaN | NaN | NaN | |
| 2011-09-01 | | NaN | NaN | NaN | NaN | NaN | NaN | |
| 2011-10-01 | | NaN | NaN | NaN | NaN | NaN | NaN | |
| 2011-11-01 | | NaN | NaN | NaN | NaN | NaN | NaN | |

[49]:

month\_list = ["Dec '10", "Jan '11", "Feb '11", "Mar '11", "Apr '11",\

"May '11", "Jun '11", "Jul '11", "Aug '11", "Sep '11", \

"Oct '11", "Nov '11", "Dec '11"]

plt.figure(figsize=(15, 13))

plt.title('average Spending for monthly cohorts')

*# Create the heatmap*

ax = sns.heatmap(data = averagePrice,

annot=**True**, vmin = 0.0,

*# vmax =20,*

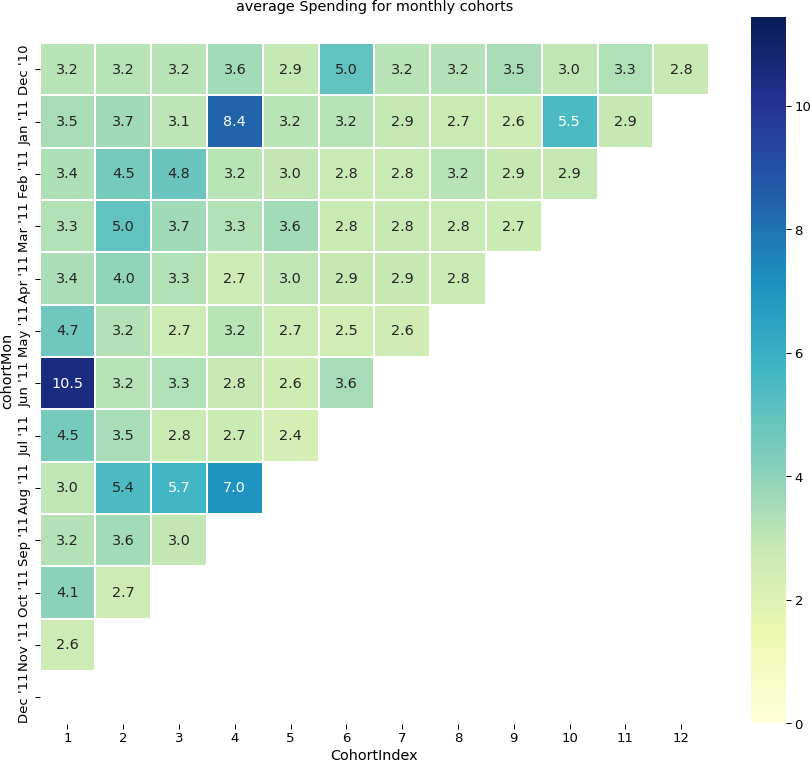
cmap='YlGnBu',

vmax = list(averagePrice.max().sort\_values(ascending = **False**))[1]+3, fmt = '.1f',

linewidth = 0.3,

yticklabels=month\_list) bottom, top = ax.get\_ylim() ax.set\_ylim(bottom + 0.5, top - 0.5) fig = plt.figure()

plt.show();



<Figure size 432x288 with 0 Axes>

[50]:

*# average quantity per cohort and differentiate cohort types across different*␣

*‹→pattern*

cohortData = groups['Quantity'].mean() cohortData=cohortData.reset\_index()

averageQuantity= cohortData.

*‹→*pivot(index='cohortMon',columns='CohortIndex',values='Quantity')

averageQuantity

[50]: CohortIndex 1 2 3 4 5 6 \

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| cohortMon 2010-12-01 | 11.200463 | 14.691852 | 15.108447 | 14.954097 | 13.054649 | 14.416287 |
| 2011-01-01 | 10.127231 | 12.704190 | 12.429557 | 11.032382 | 12.288608 | 15.006101 |
| 2011-02-01 | 10.924450 | 12.251366 | 18.563808 | 12.018144 | 11.167271 | 11.476727 |
| 2011-03-01 | 9.818050 | 9.972109 | 12.249296 | 9.483094 | 13.037510 | 12.369617 |
| 2011-04-01 | 9.803935 | 10.130252 | 9.432453 | 11.622102 | 11.645560 | 8.315994 |
| 2011-05-01 | 10.977360 | 9.138087 | 14.023864 | 11.805435 | 10.973613 | 8.740725 |
| 2011-06-01 | 10.411028 | 13.859783 | 10.509642 | 13.384102 | 10.360800 | 9.901184 |
| 2011-07-01 | 9.804225 | 12.700952 | 7.229385 | 7.929151 | 6.101961 | NaN |
| 2011-08-01 | 9.941459 | 5.983114 | 5.371409 | 5.972992 | NaN | NaN |
| 2011-09-01 | 12.003023 | 5.551129 | 7.657590 | NaN | NaN | NaN |
| 2011-10-01 | 8.553545 | 7.056196 | NaN | NaN | NaN | NaN |
| 2011-11-01 | 8.901297 | NaN | NaN | NaN | NaN | NaN |
| CohortIndex | 7 | 8 | 9 | 10 | 11 | 12 |
| cohortMon  2010-12-01 | 15.306910 | 14.879447 | 16.764934 | 16.809158 | 17.528956 | 13.019471 |
| 2011-01-01 | 14.302480 | 14.519414 | 11.451025 | 9.982762 | 9.256968 | NaN |
| 2011-02-01 | 13.378526 | 12.448602 | 10.381961 | 12.043074 | NaN | NaN |
| 2011-03-01 | 13.221102 | 12.263293 | 10.662973 | NaN | NaN | NaN |
| 2011-04-01 | 9.777895 | 9.480778 | NaN | NaN | NaN | NaN |
| 2011-05-01 | 10.275862 | NaN | NaN | NaN | NaN | NaN |
| 2011-06-01 | NaN | NaN | NaN | NaN | NaN | NaN |
| 2011-07-01 | NaN | NaN | NaN | NaN | NaN | NaN |
| 2011-08-01 | NaN | NaN | NaN | NaN | NaN | NaN |
| 2011-09-01 | NaN | NaN | NaN | NaN | NaN | NaN |
| 2011-10-01 | NaN | NaN | NaN | NaN | NaN | NaN |
| 2011-11-01 | NaN | NaN | NaN | NaN | NaN | NaN |

[51]:

plt.figure(figsize=(15, 13))

plt.title('Average quantity per Monthly Cohorts')

*# Create the heatmap*

ax = sns.heatmap(data = averageQuantity, annot=**True**,

vmin = 0.0, cmap='Pastel2',

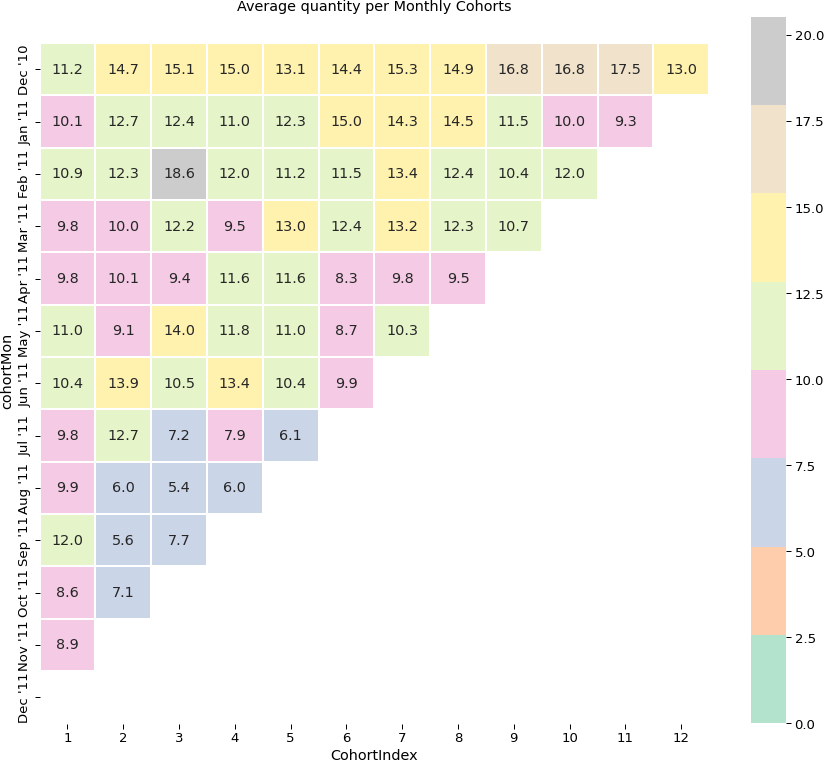
vmax = list(averageQuantity.max().sort\_values(ascending =␣

*‹→***False**))[1]+3,

fmt = '.1f', linewidth = 0.3,

yticklabels=month\_list)

bottom, top = ax.get\_ylim() ax.set\_ylim(bottom + 0.5, top - 0.5) plt.show();



[52]:

*# Date modelling*

*# Build Recency frequency monetary model)* **import datetime as dt** data['InvoiceDate'].max()

[52]: Timestamp('2011-11-30 17:42:00')

[69]:

current\_date = dt.date(2011,11,30) current\_date

[69]: datetime.date(2011, 11, 30)

[54]:

data['purchaseDate'] = data.InvoiceDate.dt.date

[56]:

data

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [56]: |  | InvoiceNo | StockCode | Description | | | | | | Quantity | \ |
|  | 0 | 536365 | 85123A | WHITE HANGING HEART T-LIGHT HOLDER | | | | | | 6 |  |
|  | 1 | 536365 | 71053 | WHITE METAL LANTERN | | | | | | 6 |  |
|  | 2 | 536365 | 84406B | CREAM CUPID HEARTS COAT HANGER | | | | | | 8 |  |
|  | 3 | 536365 | 84029G | KNITTED UNION FLAG HOT WATER BOTTLE | | | | | | 6 |  |
|  | 4  …  516379 | 536365  …  C579886 | 84029E  …  22197 | RED WOOLLY HOTTIE WHITE HEART.  … …  POPCORN HOLDER | | | | | | 6  -1 |  |
|  | 516380 | C579886 | 23146 | TRIPLE HOOK ANTIQUE IVORY ROSE | | | | | | -1 |  |
|  | 516381 | C579887 | 84946 | ANTIQUE SILVER T-LIGHT GLASS | | | | | | -1 |  |
|  | 516382 | C579887 | 85048 | 15CM CHRISTMAS GLASS BALL 20 LIGHTS | | | | | | -1 |  |
|  | 516383 | C579887 | 23490 | T-LIGHT HOLDER HANGING LOVE BIRD | | | | | | -3 |  |
| InvoiceDate UnitPrice CustomerID Country \ | | | | | | | | | | | |
| 0 | | 2010-12-01 | 08:26:00 |  | 2.55 |  | 17850.0 | United | Kingdom | | |
| 1 | | 2010-12-01 | 08:26:00 |  | 3.39 |  | 17850.0 | United | Kingdom | | |
| 2 | | 2010-12-01 | 08:26:00 |  | 2.75 |  | 17850.0 | United | Kingdom | | |
| 3 | | 2010-12-01 | 08:26:00 |  | 3.39 |  | 17850.0 | United | Kingdom | | |
| 4  …  516379 | | 2010-12-01  2011-11-30 | 08:26:00  …  17:39:00 | … | 3.39  0.85 | … | 17850.0  15676.0 | United  …  United | Kingdom  Kingdom | | |
| 516380 | | 2011-11-30 | 17:39:00 |  | 3.29 |  | 15676.0 | United | Kingdom | | |
| 516381 | | 2011-11-30 | 17:42:00 |  | 1.25 |  | 16717.0 | United | Kingdom | | |
| 516382 | | 2011-11-30 | 17:42:00 |  | 7.95 |  | 16717.0 | United | Kingdom | | |
| 516383 | | 2011-11-30 | 17:42:00 |  | 3.75 |  | 16717.0 | United | Kingdom | | |

|  |  |  |  |
| --- | --- | --- | --- |
|  | invoiceMonth | TotPrice | purchaseDate |
| 0 | 2010-12-01 | 15.30 | 2010-12-01 |
| 1 | 2010-12-01 | 20.34 | 2010-12-01 |
| 2 | 2010-12-01 | 22.00 | 2010-12-01 |
| 3 | 2010-12-01 | 20.34 | 2010-12-01 |
| 4 | 2010-12-01 | 20.34 | 2010-12-01 |
| … | … | … | … |
| 516379 | 2011-11-30 | -0.85 | 2011-11-30 |
| 516380 | 2011-11-30 | -3.29 | 2011-11-30 |
| 516381 | 2011-11-30 | -1.25 | 2011-11-30 |
| 516382 | 2011-11-30 | -7.95 | 2011-11-30 |
| 516383 | 2011-11-30 | -11.25 | 2011-11-30 |

[384222 rows x 11 columns]

[60]:

recency = data.groupby('CustomerID')['purchaseDate'].max().reset\_index() recency

|  |  |  |  |
| --- | --- | --- | --- |
| [60]: |  | CustomerID | purchaseDate |
|  | 0 | 12346.0 | 2011-01-18 |
|  | 1 | 12347.0 | 2011-10-31 |
|  | 2 | 12348.0 | 2011-09-25 |
|  | 3 | 12349.0 | 2011-11-21 |
|  | 4 | 12350.0 | 2011-02-02 |
|  | … | … | … |
|  | 4326 | 18280.0 | 2011-03-07 |
|  | 4327 | 18281.0 | 2011-06-12 |
|  | 4328 | 18282.0 | 2011-08-09 |
|  | 4329 | 18283.0 | 2011-11-30 |
|  | 4330 | 18287.0 | 2011-10-28 |

[4331 rows x 2 columns]

[78]:

*# create separate column for this date*

recency= recency.assign(currentDate = current\_date) recency

|  |  |  |  |
| --- | --- | --- | --- |
| [78]: | CustomerID | purchaseDate | currentDate |
|  | 0 12346.0 | 2011-01-18 | 2011-11-30 |
|  | 1 12347.0 | 2011-10-31 | 2011-11-30 |
|  | 2 12348.0 | 2011-09-25 | 2011-11-30 |
|  | 3 12349.0 | 2011-11-21 | 2011-11-30 |
|  | 4 12350.0 | 2011-02-02 | 2011-11-30 |
|  | … … | … | … |
|  | 4326 18280.0 | 2011-03-07 | 2011-11-30 |
|  | 4327 18281.0 | 2011-06-12 | 2011-11-30 |
|  | 4328 18282.0 | 2011-08-09 | 2011-11-30 |
|  | 4329 18283.0 | 2011-11-30 | 2011-11-30 |
|  | 4330 18287.0 | 2011-10-28 | 2011-11-30 |

[4331 rows x 3 columns]

[81]:

*# count no of days since last purchase current-purchase*

recency['RecencyDays'] = recency.purchaseDate.apply(**lambda** x: (current\_date -␣

*‹→*x).days)

[82]:

recency

|  |  |  |  |
| --- | --- | --- | --- |
| [82]: |  | CustomerID purchaseDate currentDate | RecencyDays |
|  | 0 | 12346.0 2011-01-18 2011-11-30 | 316 |
|  | 1 | 12347.0 2011-10-31 2011-11-30 | 30 |
|  | 2 | 12348.0 2011-09-25 2011-11-30 | 66 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3 | 12349.0 | 2011-11-21 2011-11-30 |  | 9 |
| 4 | 12350.0 | 2011-02-02 2011-11-30 |  | 301 |
| … | … | … … | … |  |
| 4326 | 18280.0 | 2011-03-07 2011-11-30 |  | 268 |
| 4327 | 18281.0 | 2011-06-12 2011-11-30 |  | 171 |
| 4328 | 18282.0 | 2011-08-09 2011-11-30 |  | 113 |
| 4329 | 18283.0 | 2011-11-30 2011-11-30 |  | 0 |
| 4330 | 18287.0 | 2011-10-28 2011-11-30 |  | 33 |

[83]:

[84]:

recency

[4331 rows x 4 columns]

recency.drop(['purchaseDate','currentDate'],axis=1,inplace=**True**)

|  |  |  |  |
| --- | --- | --- | --- |
| [84]: |  | CustomerID | RecencyDays |
|  | 0 | 12346.0 | 316 |
|  | 1 | 12347.0 | 30 |
|  | 2 | 12348.0 | 66 |
|  | 3 | 12349.0 | 9 |
|  | 4 | 12350.0 | 301 |
|  | … | … | … |
|  | 4326 | 18280.0 | 268 |
|  | 4327 | 18281.0 | 171 |
|  | 4328 | 18282.0 | 113 |
|  | 4329 | 18283.0 | 0 |
|  | 4330 | 18287.0 | 33 |

[4331 rows x 2 columns]

[91]:

*#frequency number of given period a customer engaged in buying products*

freq=data.groupby('CustomerID').InvoiceNo.nunique().reset\_index().

*‹→*rename(columns={'InvoiceNo':'Frequency'})

[93]:

freq.max()

1. : CustomerID 18287.0

Frequency 238.0

dtype: float64

[94]:

*#Monetary is total amount a customer spent in purchase*

data['Totalcost'] = data.Quantity\*data.UnitPrice data

1. : InvoiceNo StockCode Description Quantity \

0 536365 85123A WHITE HANGING HEART T-LIGHT HOLDER 6

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 536365 | 71053 | WHITE METAL LANTERN | | | | | | 6 |
| 2 | 536365 | 84406B | CREAM CUPID HEARTS COAT HANGER | | | | | | 8 |
| 3 | 536365 | 84029G | KNITTED UNION FLAG HOT WATER BOTTLE | | | | | | 6 |
| 4  …  516379 | 536365  …  C579886 | 84029E  …  22197 | RED WOOLLY HOTTIE WHITE HEART.  … …  POPCORN HOLDER | | | | | | 6  -1 |
| 516380 | C579886 | 23146 | TRIPLE HOOK ANTIQUE IVORY ROSE | | | | | | -1 |
| 516381 | C579887 | 84946 | ANTIQUE SILVER T-LIGHT GLASS | | | | | | -1 |
| 516382 | C579887 | 85048 | 15CM CHRISTMAS GLASS BALL 20 LIGHTS | | | | | | -1 |
| 516383 | C579887 | 23490 | T-LIGHT HOLDER HANGING LOVE BIRD | | | | | | -3 |
| InvoiceDate UnitPrice CustomerID Country \ | | | | | | | | | |
| 0 | 2010-12-01 | 08:26:00 |  | 2.55 |  | 17850.0 | United | Kingdom | |
| 1 | 2010-12-01 | 08:26:00 |  | 3.39 |  | 17850.0 | United | Kingdom | |
| 2 | 2010-12-01 | 08:26:00 |  | 2.75 |  | 17850.0 | United | Kingdom | |
| 3 | 2010-12-01 | 08:26:00 |  | 3.39 |  | 17850.0 | United | Kingdom | |
| 4  …  516379 | 2010-12-01  2011-11-30 | 08:26:00  …  17:39:00 | … | 3.39  0.85 | … | 17850.0  15676.0 | United  …  United | Kingdom  Kingdom | |
| 516380 | 2011-11-30 | 17:39:00 |  | 3.29 |  | 15676.0 | United | Kingdom | |
| 516381 | 2011-11-30 | 17:42:00 |  | 1.25 |  | 16717.0 | United | Kingdom | |
| 516382 | 2011-11-30 | 17:42:00 |  | 7.95 |  | 16717.0 | United | Kingdom | |
| 516383 | 2011-11-30 | 17:42:00 |  | 3.75 |  | 16717.0 | United | Kingdom | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | invoiceMonth | TotPrice | purchaseDate | Totalcost |
| 0 | 2010-12-01 | 15.30 | 2010-12-01 | 15.30 |
| 1 | 2010-12-01 | 20.34 | 2010-12-01 | 20.34 |
| 2 | 2010-12-01 | 22.00 | 2010-12-01 | 22.00 |
| 3 | 2010-12-01 | 20.34 | 2010-12-01 | 20.34 |
| 4 | 2010-12-01 | 20.34 | 2010-12-01 | 20.34 |
| … | … | … | … … |  |
| 516379 | 2011-11-30 | -0.85 | 2011-11-30 | -0.85 |
| 516380 | 2011-11-30 | -3.29 | 2011-11-30 | -3.29 |
| 516381 | 2011-11-30 | -1.25 | 2011-11-30 | -1.25 |
| 516382 | 2011-11-30 | -7.95 | 2011-11-30 | -7.95 |
| 516383 | 2011-11-30 | -11.25 | 2011-11-30 | -11.25 |

[384222 rows x 12 columns]

[96]:

Monetary=data.groupby('CustomerID').Totalcost.sum().reset\_index().

*‹→*rename(columns={'Totalcost':'Monetary'})

Monetary.head()

|  |  |  |  |
| --- | --- | --- | --- |
| [96]: |  | CustomerID | Monetary |
|  | 0 | 12346.0 | 0.00 |
|  | 1 | 12347.0 | 4085.18 |
|  | 2 | 12348.0 | 1797.24 |

[99]:

*# combine monetary frequency and recency to get RFM model*

rf= recency.merge(freq,on='CustomerID') rfm= rf.merge(Monetary,on='CustomerID')

rfm

3 12349.0 1757.55

4 12350.0 334.40

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| [99]: | CustomerID | RecencyDays | Frequency | Monetary |
|  | 0 12346.0 | 316 | 2 | 0.00 |
|  | 1 12347.0 | 30 | 6 | 4085.18 |
|  | 2 12348.0 | 66 | 4 | 1797.24 |
|  | 3 12349.0 | 9 | 1 | 1757.55 |
|  | 4 12350.0 | 301 | 1 | 334.40 |
|  | … … | … | … … |  |
|  | 4326 18280.0 | 268 | 1 | 180.60 |
|  | 4327 18281.0 | 171 | 1 | 80.82 |
|  | 4328 18282.0 | 113 | 2 | 98.76 |
|  | 4329 18283.0 | 0 | 15 | 1837.53 |
|  | 4330 18287.0 | 33 | 3 | 1837.28 |

[4331 rows x 4 columns]

[102]:

rfm.index[1]

[102]: 12347.0

[101]:

rfm.set\_index('CustomerID',inplace=**True**)

[103]:

*#fetch the data record against the customer IDs from rfm table*

data[data.CustomerID == rfm.index[1]]

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| [103]: | InvoiceNo | StockCode | Description | Quantity | \ |
|  | 14938 537626 | 85116 | BLACK CANDELABRA T-LIGHT HOLDER | 12 |  |
|  | 14939 537626 | 22375 | AIRLINE BAG VINTAGE JET SET BROWN | 4 |  |
|  | 14940 537626 | 71477 | COLOUR GLASS. STAR T-LIGHT HOLDER | 12 |  |
|  | 14941 537626 | 22492 | MINI PAINT SET VINTAGE | 36 |  |
|  | 14942 537626 | 22771 | CLEAR DRAWER KNOB ACRYLIC EDWARDIAN | 12 |  |
|  | … … | … | … … |  |  |
|  | 428999 573511 | 22196 | SMALL HEART MEASURING SPOONS | 24 |  |
|  | 429000 573511 | 22195 | LARGE HEART MEASURING SPOONS | 24 |  |
|  | 429001 573511 | 20719 | WOODLAND CHARLOTTE BAG | 10 |  |
|  | 429002 573511 | 23162 | REGENCY TEA STRAINER | 8 |  |
|  | 429003 573511 | 22131 | FOOD CONTAINER SET 3 LOVE HEART | 6 |  |

InvoiceDate UnitPrice CustomerID Country invoiceMonth \

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 14938 | 2010-12-07 | 14:57:00 |  | 2.10 |  | 12347.0 | Iceland |  | 2010-12-07 |
| 14939 | 2010-12-07 | 14:57:00 |  | 4.25 |  | 12347.0 | Iceland |  | 2010-12-07 |
| 14940 | 2010-12-07 | 14:57:00 |  | 3.25 |  | 12347.0 | Iceland |  | 2010-12-07 |
| 14941 | 2010-12-07 | 14:57:00 |  | 0.65 |  | 12347.0 | Iceland |  | 2010-12-07 |
| 14942  …  428999 | 2010-12-07  2011-10-31 | 14:57:00  …  12:25:00 | … | 1.25  0.85 | … | 12347.0  …  12347.0 | Iceland  Iceland | … | 2010-12-07  2011-10-31 |
| 429000 | 2011-10-31 | 12:25:00 |  | 1.65 |  | 12347.0 | Iceland |  | 2011-10-31 |
| 429001 | 2011-10-31 | 12:25:00 |  | 0.85 |  | 12347.0 | Iceland |  | 2011-10-31 |
| 429002 | 2011-10-31 | 12:25:00 |  | 3.75 |  | 12347.0 | Iceland |  | 2011-10-31 |
| 429003 | 2011-10-31 | 12:25:00 |  | 1.95 |  | 12347.0 | Iceland |  | 2011-10-31 |
|  | TotPrice | purchaseDate | Totalcost | | | | | | |
| 14938 | 25.2 | 2010-12-07 | 25.2 | | | | | | |
| 14939 | 17.0 | 2010-12-07 | 17.0 | | | | | | |
| 14940 | 39.0 | 2010-12-07 | 39.0 | | | | | | |
| 14941 | 23.4 | 2010-12-07 | 23.4 | | | | | | |
| 14942 | 15.0 | 2010-12-07 | 15.0 | | | | | | |
| … | … | … | … | | | | | | |
| 428999 | 20.4 | 2011-10-31 | 20.4 | | | | | | |
| 429000 | 39.6 | 2011-10-31 | 39.6 | | | | | | |
| 429001 | 8.5 | 2011-10-31 | 8.5 | | | | | | |
| 429002 | 30.0 | 2011-10-31 | 30.0 | | | | | | |
| 429003 | 11.7 | 2011-10-31 | 11.7 | | | | | | |

[171 rows x 12 columns]

[104]:

rfm.iloc[0,0]

[104]: 316

[105]:

(current\_date - data[data.CustomerID == rfm.index[0]].iloc[0].purchaseDate).days

[105]: 316

[144]:

*# Customer segment with RFM model*

*# simplest way to get best and lowesr values by using quantiles of 25, 50 and*␣

*‹→70 percents*

Quantiles= rfm.quantile(q=[0.25,0.5,0.75])

[145]:

Quantiles=Quantiles.to\_dict()

Quantiles *# convert quantiles to dictionary for pickup cutoffs*

[145]: {'RecencyDays': {0.25: 15.0, 0.5: 48.0, 0.75: 144.0},

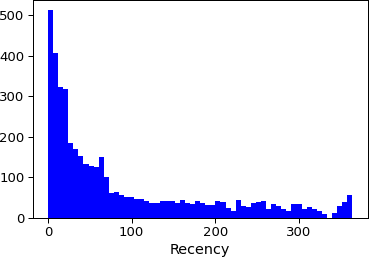
'Frequency': {0.25: 1.0, 0.5: 3.0, 0.75: 5.0},

'Monetary': {0.25: 288.755, 0.5: 628.78, 0.75: 1545.9050000000004}}

[112]:

*# visualize recency monetry and frequency histograms* plt.hist(rfm.RecencyDays,bins= 60,color='b') plt.xlabel('Recency')

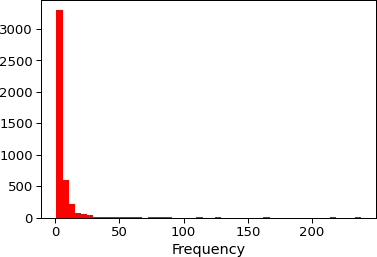
plt.show()



[114]:

plt.hist(rfm.Frequency,bins=50, color='r') plt.xlabel('Frequency')

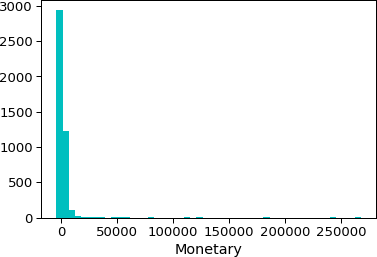
plt.show()



[135]:

plt.hist(rfm.Monetary, bins = 50, color='c') plt.xlabel('Monetary')

plt.show()



[140]:

*# RFM segments*

*#High recency is bad while high frequency and monetary is good*

**def** RScore(x,p,d):

**if** x <= d[p][0.25]:

**return** 4

**elif** x <= d[p][0.50]:

**return** 3

**elif** x <= d[p][0.75]:

**return** 2

**else**:

**return** 1

*#*

**def** MFscore(x,p,d):

**if** x <= d[p][0.25]:

**return** 1

**elif** x <= d[p][0.5]:

**return** 2

**elif** x <= d[p][0.75]:

**return** 3

**else**:

**return** 4

[142]:

rfm\_segment= rfm.copy()

[143]:

rfm\_segment

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| [143]: | CustomerID | RecencyDays | Frequency | Monetary |
|  | 12346.0 | 316 | 2 | 0.00 |
|  | 12347.0 | 30 | 6 | 4085.18 |
|  | 12348.0 | 66 | 4 | 1797.24 |
|  | 12349.0 | 9 | 1 | 1757.55 |
|  | 12350.0 | 301 | 1 | 334.40 |
|  | … | … | … | … |
|  | 18280.0 | 268 | 1 | 180.60 |
|  | 18281.0 | 171 | 1 | 80.82 |
|  | 18282.0 | 113 | 2 | 98.76 |
|  | 18283.0 | 0 | 15 | 1837.53 |
|  | 18287.0 | 33 | 3 | 1837.28 |

[4331 rows x 3 columns]

[146]:

rfm\_segment['R\_Quartile'] = rfm\_segment['RecencyDays'].apply(RScore,␣

*‹→*args=('RecencyDays',Quantiles,))

[149]:

rfm\_segment['F\_Quartile'] = rfm\_segment['Frequency'].apply(MFscore,␣

*‹→*args=('Frequency',Quantiles,))

[150]:

rfm\_segment['M\_Quartile'] = rfm\_segment['Monetary'].apply(MFscore,␣

*‹→*args=('Monetary',Quantiles,))

[151]:

rfm\_segment

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| [151]: | CustomerID | RecencyDays | Frequency | Monetary | R\_Quartile | F\_Quartile | \ |
|  | 12346.0 | 316 | 2 | 0.00 | 1 | 2 |  |
|  | 12347.0 | 30 | 6 | 4085.18 | 3 | 4 |  |
|  | 12348.0 | 66 | 4 | 1797.24 | 2 | 3 |  |
|  | 12349.0 | 9 | 1 | 1757.55 | 4 | 1 |  |
|  | 12350.0  …  18280.0 | 301  …  268 | 1  …  1 | 334.40  …  180.60 | 1  … …  1 | 1  1 |  |
|  | 18281.0 | 171 | 1 | 80.82 | 1 | 1 |  |
|  | 18282.0 | 113 | 2 | 98.76 | 2 | 2 |  |
|  | 18283.0 | 0 | 15 | 1837.53 | 4 | 4 |  |
|  | 18287.0 | 33 | 3 | 1837.28 | 3 | 2 |  |
| CustomerID | | M\_Quartile | | | | | |
| 12346.0 | | 1 | | | | | |
| 12347.0 | | 4 | | | | | |
| 12348.0 | | 4 | | | | | |
| 12349.0 | | 4 | | | | | |
| 12350.0 | | 2 | | | | | |
| … | | … | | | | | |
| 18280.0 | | 1 | | | | | |
| 18281.0 | | 1 | | | | | |
| 18282.0 | | 1 | | | | | |
| 18283.0 | | 4 | | | | | |
| 18287.0 | | 4 | | | | | |

[4331 rows x 6 columns]

[152]:

rfm\_segment['RFMScore'] = rfm\_segment.R\_Quartile.map(str) \

+ rfm\_segment.F\_Quartile.map(str) \

+ rfm\_segment.M\_Quartile.map(str)

[154]:

rfm\_segment.head()

[154]: RecencyDays Frequency Monetary R\_Quartile F\_Quartile \ CustomerID

12346.0 316 2 0.00 1 2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 12347.0 | 30 | 6 | 4085.18 | 3 | 4 |
| 12348.0 | 66 | 4 | 1797.24 | 2 | 3 |
| 12349.0 | 9 | 1 | 1757.55 | 4 | 1 |
| 12350.0 | 301 | 1 | 334.40 | 1 | 1 |
| CustomerID | M\_Quartile | RFMScore | | | |
| 12346.0 | 1 | 121 | | | |
| 12347.0 | 4 | 344 | | | |
| 12348.0 | 4 | 234 | | | |
| 12349.0 | 4 | 414 | | | |
| 12350.0 | 2 | 112 | | | |

[249]:

rfm\_segment.reset\_index(inplace=**True**) *# return back to Customer ID as a column*

[176]: **import itertools**

*# less Recency with high frequency and monetary*

platinum\_customers = ['444', '443']

print ("Platinum Customers : **{}**".format(platinum\_customers))

*# Get the combinations of [1, 2, 3,4] with length 2*

big\_spend= itertools.product([1, 2, 3,4],repeat = 2)

big\_spenders = []

**for** i **in** list(big\_spend): num = (list(i)) num.append(4)

big\_spenders.append( ("".join(map(str,num))))

print ("Big Spenders : **{}**".format(big\_spenders))

*# New Customers – highest spendings*

*# This group consists of those customers in 1-4-1 and 1-4-2.*

high\_spend\_new\_customers = ['413', '314' ,'313','414'] print ("High Spend New Customers : **{}**".

*‹→*format(high\_spend\_new\_customers))

low\_spending\_active\_customer = itertools.product([ 3,4], repeat = 2) low\_spending\_active\_customers = []

**for** i **in** list(low\_spending\_active\_customer): num = (list(i))

num.append(1)

low\_spending\_active\_customers.append( ("".join(map(str,num)))) print ("Low Spending Active Customers : **{}**".

*‹→*format(low\_spending\_active\_customers))

recent\_customer = itertools.product([ 2,3,4], repeat = 2) recent\_customers = []

**for** i **in** list(recent\_customer): num = (list(i)) num.insert(0,4)

recent\_customers.append( ("".join(map(str,num))))

print ("Recent Customers : **{}**".format(recent\_customers))

[177]:

*# less shopping customers*

almost\_lost = ['244', '234', '243', '233']

print ("Good Customers Almost Lost : **{}**".format(almost\_lost))

churned\_best\_customers = ['144', '134' ,'143','133'] print ("Churned Best Customers : **{}**".

*‹→*format(churned\_best\_customers))

*# long time customer shopped with low freq and monetary*

lost\_customers = ['122','111','221','212' ,'211']

print ("Long time slow buyed Customers : **{}**".

*‹→*format(lost\_customers))

Platinum Customers : ['444', '443']

Big Spenders : ['114', '124', '134', '144', '214',

'224', '234', '244', '314', '324', '334', '344', '414', '424', '434', '444']

High Spend New Customers : ['413', '314', '313', '414']

Low Spending Active Customers : ['331', '341', '431', '441']

Recent Customers : ['422', '423', '424', '432', '433',

'434', '442', '443', '444']

Good Customers Almost Lost : ['244', '234', '243', '233']

Churned Best Customers : ['144', '134', '143', '133']

Long time slow buyed Customers : ['122', '111', '221', '212',

'211']

segment\_dict = {

'Platinum Customers':platinum\_customers, 'Big Spenders': big\_spenders,

'High Spend New Customers':high\_spend\_new\_customers,

'Low Spending Active Customers' : low\_spending\_active\_customers , 'Recent Customers': recent\_customers,

'Good Customers Almost Lost':almost\_lost,

'Churned Best Customers': churned\_best\_customers, 'Lost Cheap Customers ': lost\_customers,

}

[178]:

*# allocate key segments to RFM ranking*

**def** find\_key(rank):

**for** r, v **in** segment\_dict.items():

**if** rank **in** v:

**return** r

[179]:

rfm\_segment['Segment']=rfm\_segment.RFMScore.map(find\_key)

[182]:

rfm\_segment.Segment.fillna('Nothing',inplace=**True**)

[183]:

rfm\_segment

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [183]: | CustomerID | RecencyDays | Frequency | | Monetary | R\_Quartile | F\_Quartile | \ |
|  | 12346.0 | 316 | 2 | | 0.00 | 1 | 2 |  |
|  | 12347.0 | 30 | 6 | | 4085.18 | 3 | 4 |  |
|  | 12348.0 | 66 | 4 | | 1797.24 | 2 | 3 |  |
|  | 12349.0 | 9 | 1 | | 1757.55 | 4 | 1 |  |
|  | 12350.0  …  18280.0 | 301  …  268 | 1  …  1 | | 334.40  …  180.60 | 1  … …  1 | 1  1 |  |
|  | 18281.0 | 171 | 1 | | 80.82 | 1 | 1 |  |
|  | 18282.0 | 113 | 2 | | 98.76 | 2 | 2 |  |
|  | 18283.0 | 0 | 15 | | 1837.53 | 4 | 4 |  |
|  | 18287.0 | 33 | 3 | | 1837.28 | 3 | 2 |  |
| CustomerID | | M\_Quartile | RFMScore | |  | Segment | | |
| 12346.0 | | 1 | 121 | |  | Nothing | | |
| 12347.0 | | 4 | 344 | | Big | Spenders | | |
| 12348.0 | | 4 | 234 | | Big | Spenders | | |
| 12349.0 | | 4 | 414 | | Big | Spenders | | |
| 12350.0  … | | 2  … | 112  … | |  | Nothing  … | | |
| 18280.0 | | 1 | 111 | Lost Cheap Customers | | | | |
| 18281.0 | | 1 | 111 | Lost Cheap Customers | | | | |
| 18282.0 | | 1 | 221 | Lost Cheap Customers | | | | |
| 18283.0 | | 4 | 444 | Platinum Customers | | | | |
| 18287.0 | | 4 | 324 | Big Spenders | | | | |

[4331 rows x 8 columns]

[184]:

rfm\_segment.sample(10)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| [184]: | CustomerID | RecencyDays | Frequency | Monetary | R\_Quartile | F\_Quartile | \ |
|  | 15919.0 | 10 | 2 | 525.69 | 4 | 2 |  |
|  | 16478.0 | 16 | 3 | 331.00 | 3 | 2 |  |
|  | 18213.0 | 227 | 1 | 232.48 | 1 | 1 |  |
|  | 13754.0 | 142 | 2 | 282.60 | 2 | 2 |  |
|  | 12830.0 | 28 | 7 | 6748.40 | 3 | 4 |  |
|  | 16393.0 | 2 | 18 | 2656.25 | 4 | 4 |  |
|  | 17309.0 | 86 | 1 | 82.75 | 2 | 1 |  |
|  | 13355.0 | 113 | 1 | 674.70 | 2 | 1 |  |
|  | 17749.0 | 169 | 2 | 674.52 | 1 | 2 |  |
|  | 16806.0 | 94 | 4 | 762.16 | 2 | 3 |  |

CustomerID

M\_Quartile RFMScore Segment

[185]:

rfm\_segment[rfm\_segment.RFMScore=='444'].

*‹→*sort\_values('Monetary',ascending=**False**).head() *# Best Customer*

15919.0 2 422 Recent Customers

16478.0 2 322 Nothing

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 18213.0 | 1 | 111 |  | Lost Cheap Customers |
| 13754.0 | 1 | 221 |  | Lost Cheap Customers |
| 12830.0 | 4 | 344 |  | Big Spenders |
| 16393.0 | 4 | 444 |  | Platinum Customers |
| 17309.0 | 1 | 211 |  | Lost Cheap Customers |
| 13355.0 | 3 | 213 |  | Nothing |
| 17749.0 | 3 | 123 |  | Nothing |
| 16806.0 | 3 | 233 | Good | Customers Almost Lost |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| [185]: | CustomerID | RecencyDays | Frequency | Monetary | R\_Quartile | F\_Quartile | \ |
|  | 14646.0 | 7 | 74 | 267761.00 | 4 | 4 |  |
|  | 18102.0 | 2 | 59 | 244952.95 | 4 | 4 |  |
|  | 17450.0 | 1 | 54 | 185759.77 | 4 | 4 |  |
|  | 14911.0 | 0 | 238 | 125482.36 | 4 | 4 |  |
|  | 12415.0 | 15 | 26 | 123725.45 | 4 | 4 |  |
|  |  | M\_Quartile RFMScore | | Segment | |  |  |
| CustomerID 14646.0 | | 4 | 444 Platinum Customers | | | | |
| 18102.0 | | 4 | 444 Platinum Customers | | | | |
| 17450.0 | | 4 | 444 Platinum Customers | | | | |
| 14911.0 | | 4 | 444 Platinum Customers | | | | |
| 12415.0 | | 4 | 444 Platinum Customers | | | | |

[187]:

rfm\_segment[rfm\_segment.RFMScore=='334'].

*‹→*sort\_values('Frequency',ascending=**False**).head() *# big spenders*

[187]: RecencyDays Frequency Monetary R\_Quartile F\_Quartile \

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| CustomerID 12359.0 | 48 | 5 | | 6274.23 | 3 | 3 |
| 13982.0 | 30 | 5 | | 1861.11 | 3 | 3 |
| 14145.0 | 37 | 5 | | 3148.20 | 3 | 3 |
| 14498.0 | 33 | 5 | | 1957.32 | 3 | 3 |
| 14691.0 | 21 | 5 | | 2114.33 | 3 | 3 |
| CustomerID | M\_Quartile | RFMScore |  | Segment | | |
| 12359.0 | 4 | 334 | Big | Spenders | | |
| 13982.0 | 4 | 334 | Big | Spenders | | |
| 14145.0 | 4 | 334 | Big | Spenders | | |
| 14498.0 | 4 | 334 | Big | Spenders | | |
| 14691.0 | 4 | 334 | Big | Spenders | | |

[189]:

rfm\_segment[rfm\_segment.RFMScore=='111'].

*‹→*sort\_values('Frequency',ascending=**False**).head() *# lost customer*

[189]:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| RecencyDays Frequency Monetary | | | | | | R\_Quartile | F\_Quartile | \ |
| CustomerID 12353.0 | 195 | 1 | | 89.00 | | 1 | 1 | |
| 16446.0 | 196 | 1 | | 2.90 | | 1 | 1 | |
| 16583.0 | 364 | 1 | | 233.45 | | 1 | 1 | |
| 16579.0 | 356 | 1 | | -30.60 | | 1 | 1 | |
| 16565.0 | 355 | 1 | | 173.70 | | 1 | 1 | |
| M\_Quartile RFMScore Segment | | | | | | | | |
| CustomerID 12353.0 | 1 | 111 | Lost | | Cheap Customers | | | |
| 16446.0 | 1 | 111 | Lost | | Cheap Customers | | | |
| 16583.0 | 1 | 111 | Lost | | Cheap Customers | | | |
| 16579.0 | 1 | 111 | Lost | | Cheap Customers | | | |
| 16565.0 | 1 | 111 | Lost | | Cheap Customers | | | |

[190]:

*# convert data into excel*

rfm\_segment.to\_excel('RFMsegment.xlsx')

[ ]:

*#Create clusters using k-means clustering algorithm.*

*#a. Prepare the data for the algorithm. If the data is asymmetrically*␣

*‹→distributed, manage the skewness with appropriate transformation.*␣

*‹→Standardize the data.*

[197]:

*# distribution plots*

fig, axes = plt.subplots(3,1,figsize=(15,15)) sns.distplot(rfm.RecencyDays, color='cyan', ax=axes[0],axlabel='Recency') sns.distplot(rfm.Frequency, color='pink', ax=axes[1],axlabel='Frequency') sns.distplot(rfm.Monetary, color='grey', ax=axes[2],axlabel='Monetary') plt.show();

/usr/local/lib/python3.7/site-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).

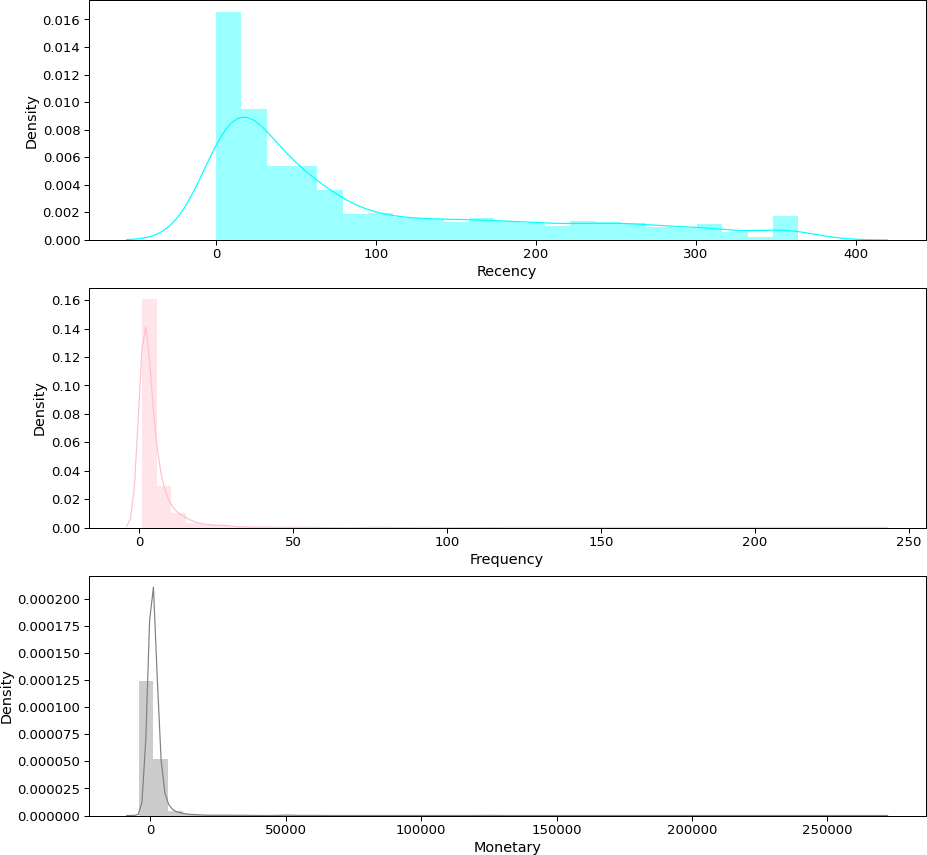
warnings.warn(msg, FutureWarning)

/usr/local/lib/python3.7/site-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).

warnings.warn(msg, FutureWarning)

/usr/local/lib/python3.7/site-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).

warnings.warn(msg, FutureWarning)



[194]:

rfm

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| [194]: | CustomerID | RecencyDays | Frequency | Monetary |
|  | 12346.0 | 316 | 2 | 0.00 |
|  | 12347.0 | 30 | 6 | 4085.18 |
|  | 12348.0 | 66 | 4 | 1797.24 |
|  | 12349.0 | 9 | 1 | 1757.55 |
|  | 12350.0  … | 301  … | 1  … | 334.40  … |
| 18280.0 | | 268 | 1 | 180.60 |
| 18281.0 | | 171 | 1 | 80.82 |
| 18282.0 | | 113 | 2 | 98.76 |
| 18283.0 | | 0 | 15 | 1837.53 |
| 18287.0 | | 33 | 3 | 1837.28 |

[4331 rows x 3 columns]

[196]:

*#By looking the distribution plot we can observe it is highly skewed we need to*␣

*‹→tranform in normal distribution in order to apply kmean algorithm*

[196]: CustomerID

12346.0 2

12347.0 6

12348.0 4

12349.0 1

12350.0 1

..

18280.0 1

18281.0 1

18282.0 2

18283.0 15

18287.0 3

Name: Frequency, Length: 4331, dtype: int64

[198]:

rfm\_Scale=rfm.copy()

[201]:

rfm\_Scale.Monetary=rfm\_Scale.Monetary+abs(rfm\_Scale.Monetary.min())+1 rfm\_Scale.RecencyDays=rfm\_Scale.RecencyDays+abs(rfm\_Scale.RecencyDays.min())+1

[202]:

rfm\_Scale.describe()

|  |  |  |  |
| --- | --- | --- | --- |
| [202]: | RecencyDays | Frequency | Monetary |
| count | 4331.000000 | 4331.000000 | 4331.000000 |
| mean | 91.277303 | 4.910875 | 6123.227551 |
| std | 99.389069 | 9.025901 | 7944.283177 |
| min | 1.000000 | 1.000000 | 3.000000 |
| 25% | 16.000000 | 1.000000 | 4579.385000 |
| 50% | 49.000000 | 3.000000 | 4919.410000 |
| 75% | 145.000000 | 5.000000 | 5836.535000 |
| max | 365.000000 | 238.000000 | 272051.630000 |

[221]:

*# standarize values of Recency, Monetary and Frequency*

**from sklearn.preprocessing import** StandardScaler

*# take logarithmic of data set to deal -ive issues*

data\_log = np.log(rfm\_Scale) scal = StandardScaler()

normal\_dat = scaler.fit\_transform(data\_log)

normal\_dat = pd.DataFrame(data=normal\_dat,index=rfm.index,columns=rfm.columns)

[222]:

*# distribution plots*

fig, axes = plt.subplots(3,1,figsize=(15,15))

sns.distplot(normal\_dat.RecencyDays, color='cyan', ax=axes[0],axlabel='Recency') sns.distplot(normal\_dat.Frequency, color='pink', ax=axes[1],axlabel='Frequency') sns.distplot(normal\_dat.Monetary, color='grey', ax=axes[2],axlabel='Monetary') plt.show();

/usr/local/lib/python3.7/site-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).

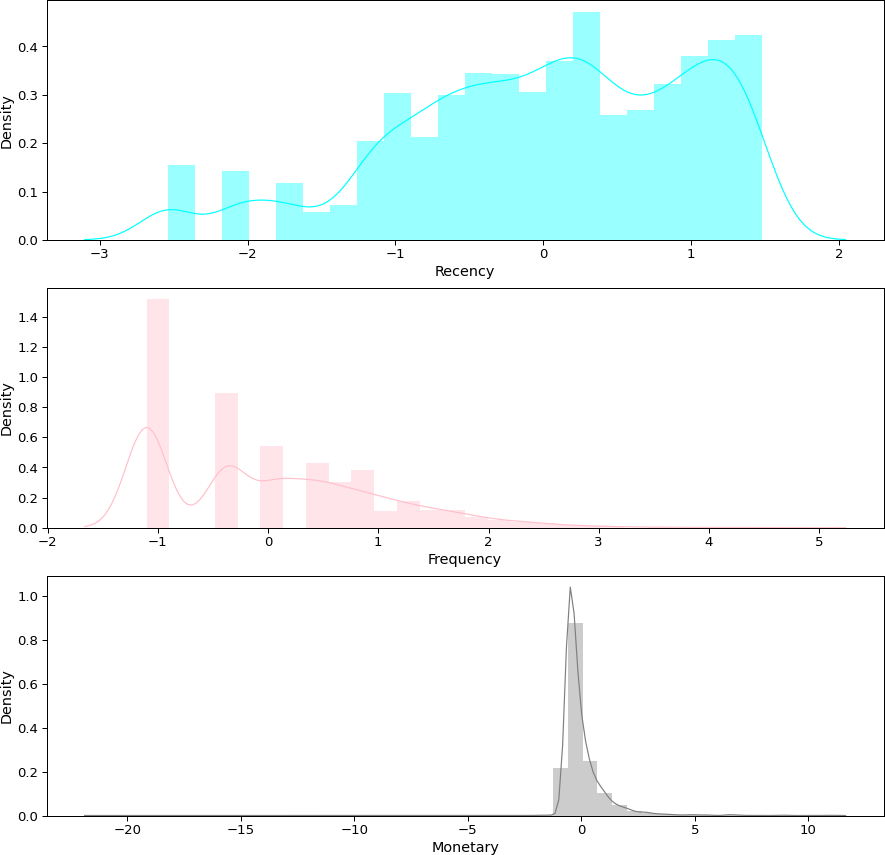
warnings.warn(msg, FutureWarning)

/usr/local/lib/python3.7/site-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).

warnings.warn(msg, FutureWarning)

/usr/local/lib/python3.7/site-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).

warnings.warn(msg, FutureWarning)



[237]:

*# WCSS within clusters sum of squares* **from sklearn.cluster import** KMeans **import matplotlib.pyplot as plt**

wcs = []

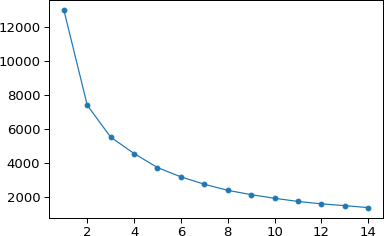
**for** k **in** range(1,15):

kmean = KMeans(n\_clusters=k, init='k-means++') kmean.fit(normal\_dat) wcs.append(kmean.inertia\_)

*# plot elbow*

plt.plot(range(1,15),wcs,marker='o')

[237]: [<matplotlib.lines.Line2D at 0x7fc4d389cf50>]



[239]:

elbow=pd.DataFrame({'Clusters': range(1,15), 'SSE': wcs}) elbow.to\_excel('Elbow\_Data.xlsx')

[242]:

**from sklearn.metrics import** silhouette\_score wcs\_silhouette = []

**for** k **in** range(3,15):

km=KMeans(n\_clusters=k, random\_state=0, init='k-means++').fit(normal\_dat) pred\_values = km.predict(normal\_dat)

silh = silhouette\_score(normal\_dat, pred\_values) wcs\_silhouette.append(silh)

print("Silhouette score for number of clusters **{}**:**{}**".format(k,silh))

Silhouette score for number of clusters 3:0.3949245651816997 Silhouette score for number of clusters 4:0.349423731560158 Silhouette score for number of clusters 5:0.36004906615278526 Silhouette score for number of clusters 6:0.3544184830498085 Silhouette score for number of clusters 7:0.3583202497358669 Silhouette score for number of clusters 8:0.3496330431177284 Silhouette score for number of clusters 9:0.3507643174652312 Silhouette score for number of clusters 10:0.3498839035566435 Silhouette score for number of clusters 11:0.35643305284763627 Silhouette score for number of clusters 12:0.37100949642452535 Silhouette score for number of clusters 13:0.3668379906010333

Silhouette score for number of clusters 14:0.37275408410378397

[244]:

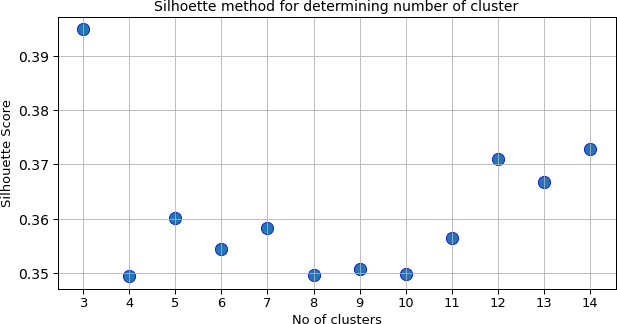
*# plot silhoetter for determining number of cluster*

plt.figure(figsize=(10,5))

plt.title("Silhoette method for determining number of cluster",fontsize=14) plt.scatter(x=range(3,15), y=wcs\_silhouette, s=150, edgecolor='b') plt.grid(**True**)

plt.xlabel("No of clusters", fontsize=13) plt.ylabel("Silhouette Score", fontsize=13) plt.xticks([k **for** k **in** range(3,15)] , fontsize=13) plt.yticks(fontsize=14)

plt.show()



[ ]:

*# The optimum no of cluster is 4*

[246]:

kmeans = KMeans(n\_clusters=4, random\_state=1, init='k-means++') kmeans.fit(normal\_dat)

clus\_labels = kmeans.labels\_

[247]:

clus\_labels.shape

[247]: (4331,)

[250]:

rfm\_segment.shape

[250]: (4331, 9)

[251]:

Cluster\_data= rfm\_segment.assign(cluster=clus\_labels)

[252]:

Cluster\_data

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [252]: |  | CustomerID | RecencyDays | | Frequency | Monetary | R\_Quartile | F\_Quartile | \ |
|  | 0 | 12346.0 | 316 | | 2 | 0.00 | 1 | 2 |  |
|  | 1 | 12347.0 | 30 | | 6 | 4085.18 | 3 | 4 |  |
|  | 2 | 12348.0 | 66 | | 4 | 1797.24 | 2 | 3 |  |
|  | 3 | 12349.0 | 9 | | 1 | 1757.55 | 4 | 1 |  |
|  | 4 | 12350.0 | 301 | | 1 | 334.40 | 1 | 1 |  |
|  | … | … | … | | … … | … | … |  |  |
|  | 4326 | 18280.0 | 268 | | 1 | 180.60 | 1 | 1 |  |
|  | 4327 | 18281.0 | 171 | | 1 | 80.82 | 1 | 1 |  |
|  | 4328 | 18282.0 | 113 | | 2 | 98.76 | 2 | 2 |  |
|  | 4329 | 18283.0 | 0 | | 15 | 1837.53 | 4 | 4 |  |
|  | 4330 | 18287.0 | 33 | | 3 | 1837.28 | 3 | 2 |  |
|  | | M\_Quartile | RFMScore |  | Segment cluster | | | | |
| 0 | | 1 | 121 |  | Nothing 3 | | | | |
| 1 | | 4 | 344 |  | Big Spenders 2 | | | | |
| 2 | | 4 | 234 |  | Big Spenders 2 | | | | |
| 3 | | 4 | 414 |  | Big Spenders 0 | | | | |
| 4  …  4326 | | 2  …  1 | 112  …  111 | Lost | Nothing 3  … …  Cheap Customers 3 | | | | |
| 4327 | | 1 | 111 | Lost | Cheap Customers 3 | | | | |
| 4328 | | 1 | 221 | Lost | Cheap Customers 3 | | | | |
| 4329 | | 4 | 444 | Platinum Customers | | | 2 | | |
| 4330 | | 4 | 324 | Big Spenders | | | 0 | | |

[4331 rows x 10 columns]

[253]:

Cluster\_data.cluster.value\_counts()

|  |  |
| --- | --- |
| [253]: 3 | 1979 |
| 2 | 1103 |
| 0 | 1092 |
| 1 | 157 |

Name: cluster, dtype: int64

[255]:

*# most customer belonges to 3 and 2*

Cluster\_data.sample(10)

print ("Platinum customers relative to cluster

*‹→*format(Cluster\_data[Cluster\_data['Segment']=='Platinum␣

*‹→*Customers']['cluster'].unique()))

: **{}** ".

print ("Big Spenders relative to cluster

: **{}** ".

*‹→*unique()))

print ("High Spend new Customers relative to cluster

*‹→*format(Cluster\_data[Cluster\_data['Segment']=='High Spend New␣

*‹→*Customers']['cluster'].unique()))

*‹→*format(Cluster\_data[Cluster\_data['Segment']=='Big Spenders']['cluster'].

: **{}** ".

print ("Low Spending Active Customers relative to cluster : **{}** ".

*‹→*format(Cluster\_data[Cluster\_data['Segment']=='Low Spending Active␣

*‹→*Customers']['cluster'].unique()))

print ("Recent Customers relative to cluster : **{}** ".

*‹→*format(Cluster\_data[Cluster\_data['Segment']=='Recent Customers']['cluster'].

print ("Good Customers Almost Lost relative to cluster

*‹→*unique()))

: **{}** ".

*‹→*Lost']['cluster'].unique()))

print ("Churned Best Customers relative to cluster

*‹→*format(Cluster\_data[Cluster\_data['Segment']=='Churned Best␣

*‹→*Customers']['cluster'].unique()))

*‹→*format(Cluster\_data[Cluster\_data['Segment']=='Good Customers Almost␣

: **{}** ".

print ("Lost customers relative to cluster

: **{}** ".

*‹→*format(Cluster\_data[Cluster\_data['Segment']=='Lost Cheap Customers␣

*‹→*']['cluster'].unique()))

|  |  |  |
| --- | --- | --- |
| Platinum customers relative to cluster |  | : [2 1] |
| Big Spenders relative to cluster |  | : [2 0 3 1] |
| High Spend new Customers relative to cluster |  | : [0 3] |
| Low Spending Active Customers relative to cluster | : [0 2] |  |
| Recent Customers relative to cluster |  | : [0 2] |
| Good Customers Almost Lost relative to cluster |  | : [3 0 2] |
| Churned Best Customers relative to cluster |  | : [3 2] |

Lost customers relative to cluster : [3 0]

[256]:

Cluster\_data[Cluster\_data.cluster == 3].sample(5)

[256]: CustomerID RecencyDays Frequency Monetary R\_Quartile F\_Quartile \

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1199 | 13976.0 | 197 | 1 357.98 | | 1 | 1 |
| 2494 | 15746.0 | 125 | 2 182.96 | | 2 | 2 |
| 3994 | 17824.0 | 42 | 1 298.40 | | 3 | 1 |
| 671 | 13239.0 | 252 | 1 329.56 | | 1 | 1 |
| 247 | 12652.0 | 313 | 2 914.53 | | 1 | 2 |
|  | M\_Quartile | RFMScore Segment cluster | | | | |
| 1199 | 2 | 112 Nothing 3 | | | | |
| 2494 | 1 | 221 Lost Cheap Customers 3 | | | | |
| 3994 | 2 | 312 | Nothing | 3 | | |
| 671 | 2 | 112 | Nothing | 3 | | |
| 247 | 3 | 123 | Nothing | 3 | | |

[258]:

Cluster\_data[Cluster\_data.cluster == 2].sample(5)

1. : CustomerID RecencyDays Frequency Monetary R\_Quartile F\_Quartile \

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 2755 | 16115.0 | 0 | | 8 | 1660.88 | | 4 | 4 |
| 3334 | 16904.0 | 20 | | 15 | 3304.73 | | 3 | 4 |
| 4299 | 18241.0 | 0 | | 18 | 2058.09 | | 4 | 4 |
| 2179 | 15321.0 | 40 | | 4 | 2663.48 | | 3 | 3 |
| 3028 | 16477.0 | 54 | | 7 | 2494.46 | | 2 | 4 |
|  | M\_Quartile | RFMScore | Segment | | | cluster | | |
| 2755 | 4 | 444 | Platinum Customers | | | 2 | | |
| 3334 | 4 | 344 | Big Spenders | | | 2 | | |
| 4299 | 4 | 444 | Platinum Customers | | | 2 | | |
| 2179 | 4 | 334 | Big Spenders | | | 2 | | |
| 3028 | 4 | 244 | Big Spenders | | | 2 | | |

[259]:

Cluster\_data[Cluster\_data.cluster == 1].sample(5)

1. : CustomerID RecencyDays Frequency Monetary R\_Quartile F\_Quartile \

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 323 | 12748.0 | 1 | | 215 | 27406.56 | | 4 | 4 |
| 1961 | 15039.0 | 0 | | 52 | 19638.59 | | 4 | 4 |
| 793 | 13408.0 | 2 | | 77 | 26097.54 | | 4 | 4 |
| 579 | 13113.0 | 5 | | 39 | 10170.80 | | 4 | 4 |
| 2807 | 16180.0 | 91 | | 10 | 10217.48 | | 2 | 4 |
|  | M\_Quartile | RFMScore |  | Segment | | cluster | | |
| 323 | 4 | 444 | Platinum | Customers | | 1 | | |
| 1961 | 4 | 444 | Platinum | Customers | | 1 | | |
| 793 | 4 | 444 | Platinum | Customers | | 1 | | |
| 579 | 4 | 444 | Platinum | Customers | | 1 | | |

2807 4 244 Big Spenders 1

[ ]:

*# cluter 1 is high monetary with platinum customers*

[266]:

*#Scatter plot for distribution of customer relative to cluster based on RFM*␣

*‹→model*

*# Plotting two dimesional plots of each attributes respectively.*

k = normal\_dat.iloc[:,0:3].values count=k.shape[1]

**for** i **in** range(0,count):

**for** j **in** range(i+1,count): plt.figure(figsize=(15,6));

plt.scatter(k[clus\_labels == 0, i], k[clus\_labels == 0, j], s = 10, c =␣

*‹→*'green', label = 'Cluster0')

plt.scatter(k[clus\_labels == 1, i], k[clus\_labels == 1, j], s = 10, c =␣

*‹→*'blue', label = 'Cluster1')

plt.scatter(k[clus\_labels == 2, i], k[clus\_labels == 2, j], s = 10, c =␣

*‹→*'red', label = 'Cluster2')

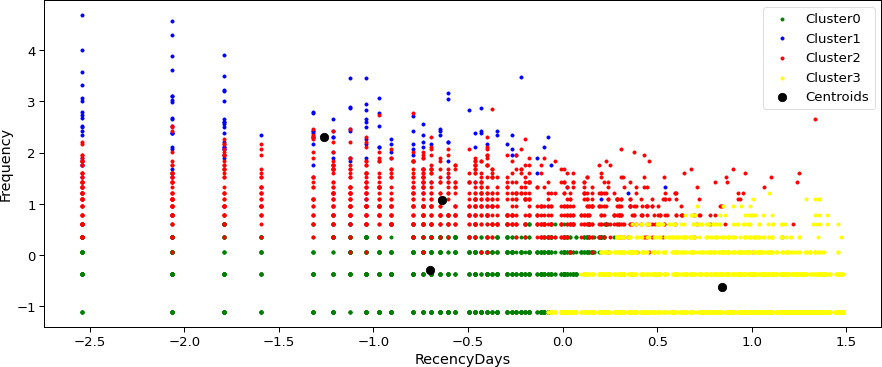
plt.scatter(k[clus\_labels == 3, i], k[clus\_labels == 3, j], s = 10, c =␣

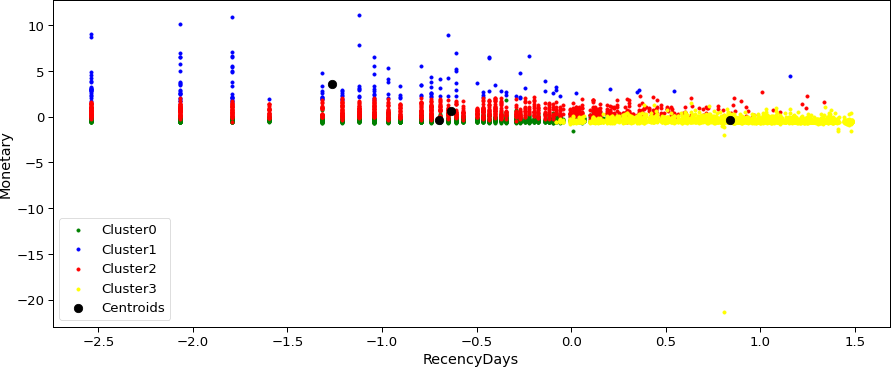
*‹→*'yellow', label = 'Cluster3')

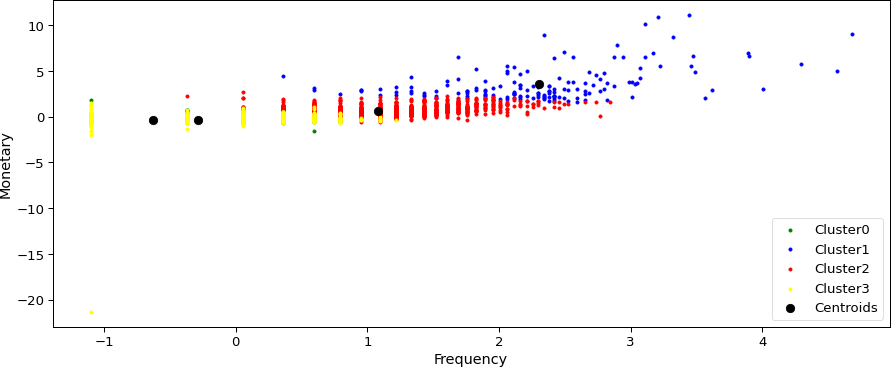
plt.scatter(kmeans.cluster\_centers\_[:,i], kmeans.cluster\_centers\_[:,j],␣

*‹→*s = 70, c = 'black', label = 'Centroids') plt.xlabel(normal\_dat.columns[i]) plt.ylabel(normal\_dat.columns[j]) plt.legend()

plt.show();







[267]:

Cluster\_data.to\_excel('RFMSegment.xlsx')

[268]:

*# Assign cluster labels to RFM records* rfm\_clus= rfm.assign(cluster=clus\_labels) *#average attributes for each cluster*

cluster\_av =rfm\_clus.groupby(['cluster']).mean()

[269]:

cluster\_av

[269]: RecencyDays Frequency Monetary cluster

0 19.816850 2.467033 617.044589

1 11.936306 32.789809 22125.186306

2 28.412511 8.871260 2578.539211

3 169.852451 1.840323 477.708086

[271]:

population\_av= rfm.mean() population\_av

[271]: RecencyDays 90.277303

Frequency 4.910875

Monetary 1832.597551

dtype: float64

[272]:

*# calculate relative importance of attributes by*

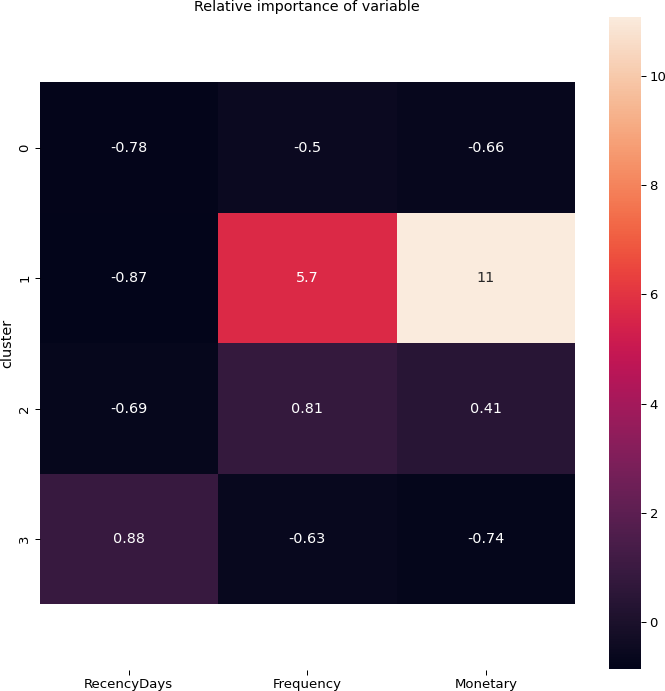
Rel\_imp = cluster\_av/population\_av -1

[274]:

*# Heat map*

plt.figure(figsize=(12,12)) plt.title('Relative importance of variable')

ax = sns.heatmap(Rel\_imp, annot= **True**) bottom, top = ax.get\_ylim() ax.set\_ylim(bottom+0.5, top-0.5) plt.show()



[275]:

data.to\_excel('Tableau.xlsx')

[ ]:

