In [3]:

```
#MaiNguyenAnBinh_K184060777

import pandas as pd
```

In [4]:

```
df = pd.read_csv("D:\Studying\DA\dataCustomerRFM.csv")
```

In [ ]:

```
#Delete StockCode column
df = df.drop(['StockCode'],axis=1)
```

In [ ]:

```
#Delete Description column
df = df.drop(['Description'],axis=1)
```

In [ ]:

```
#Delete Price column
df = df.drop(['Price'],axis=1)
```

In [ ]:

```
#Delete Quantity column
df = df.drop(['Quantity'],axis=1)
```

In [ ]:

```
#Delete Country column
df = df.drop(['Country'],axis=1)
```

In [15]:

```
# Find NULL value
total = df.isnull().sum().sort_values(ascending=False)
percent_1=df.isnull().sum()/df.isnull().count()*100
percent_2 = (round(percent_1,1)).sort_values(ascending=False)
missing_data=pd.concat([total,percent_2],axis=1,keys=['Total','%'])
missing_data.head(5)
```

Out[15]:

	Total	%
Amount	3	0.0
Unnamed: 0	0	0.0
CustomerID	0	0.0
OrderDate	0	0.0
OrderID	0	0.0

In [13]:

```
#Data after deleting a unnecessary column
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303295 entries, 0 to 303294
Data columns (total 5 columns):
 #   Column      Non-Null Count   Dtype  
---  ------      --------------   -----  
 0   Unnamed: 0  303295 non-null  int64  
 1   CustomerID  303295 non-null  object 
 2   OrderDate   303295 non-null  object 
 3   OrderID     303295 non-null  object 
 4   Amount      303292 non-null  float64
dtypes: float64(1), int64(1), object(3)
memory usage: 11.6+ MB
```

In [6]:

```
from datetime import datetime
# 1. Calculate Recency
# 1.1. Find the most recent orderDate.

dfRecentOrder = pd.pivot_table(data = df,
               index = ['CustomerID'],
               values = ['OrderDate'],
               aggfunc = {'OrderDate':max}
              )

dfRecentOrder.columns = ['RecentOrderDate']
df = pd.merge(df, dfRecentOrder.reset_index(), on = ['CustomerID'])
df['RecentOrderDate'] = df['RecentOrderDate'].apply(lambda x: datetime.strptime(x, '%Y-%m-%d %H:%M:%S'))
df['Recency'] = df['RecentOrderDate'].apply(lambda x: (datetime.now() - x).days)
```

In [7]:

```
# Change direction of recency
df['Recency'] = - df['Recency']
```

In [32]:

```
print(df['Recency'])
```

```
0         684
1         666
2         601
3         662
4         633
         ... 
303290    612
303291    612
303292    610
303293    628
303294    589
Name: Recency, Length: 303295, dtype: int64
```

In [8]:

```
# 2. Calculate Frequency
dfFrequency = df.groupby('CustomerID').OrderID.nunique().to_frame()
dfFrequency.columns = ['Frequency']
df = pd.merge(df, dfFrequency.reset_index(), on = 'CustomerID')
```

In [10]:

```
# 3. Calculate Monetary
```

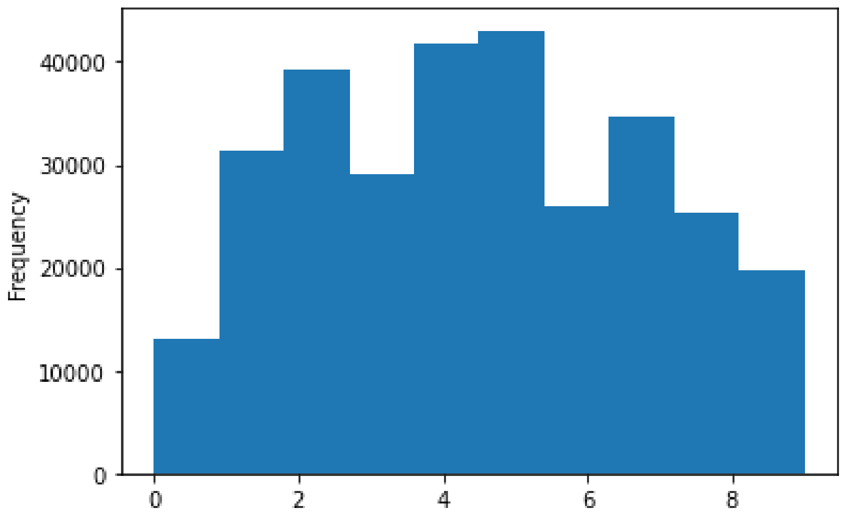
```
dfMonetary = df.groupby('CustomerID').Amount.sum().to_frame()
dfMonetary.columns = ['Monetary']
df = pd.merge(df, dfMonetary.reset_index(), on = 'CustomerID')
```

In [11]:

```
orderFrequencies = df['Frequency'].rank(method='first')
df['rFrequency'] = pd.qcut(orderFrequencies, 10, labels = False)
df[['rRecency', 'rMonetary']] = df[['Recency', 'Monetary']].apply(lambda x: pd.qcut(x, 10, labels = False))
df['rank'] = (df['rFrequency'] + df['rRecency'] + df['rMonetary'])/3
df['FinalRank'] = df['rank'].apply(int)
```

In [12]:

```
import matplotlib.pyplot as plt

df['rank'].plot.hist(bins = 10)
plt.show()
```



In [13]:

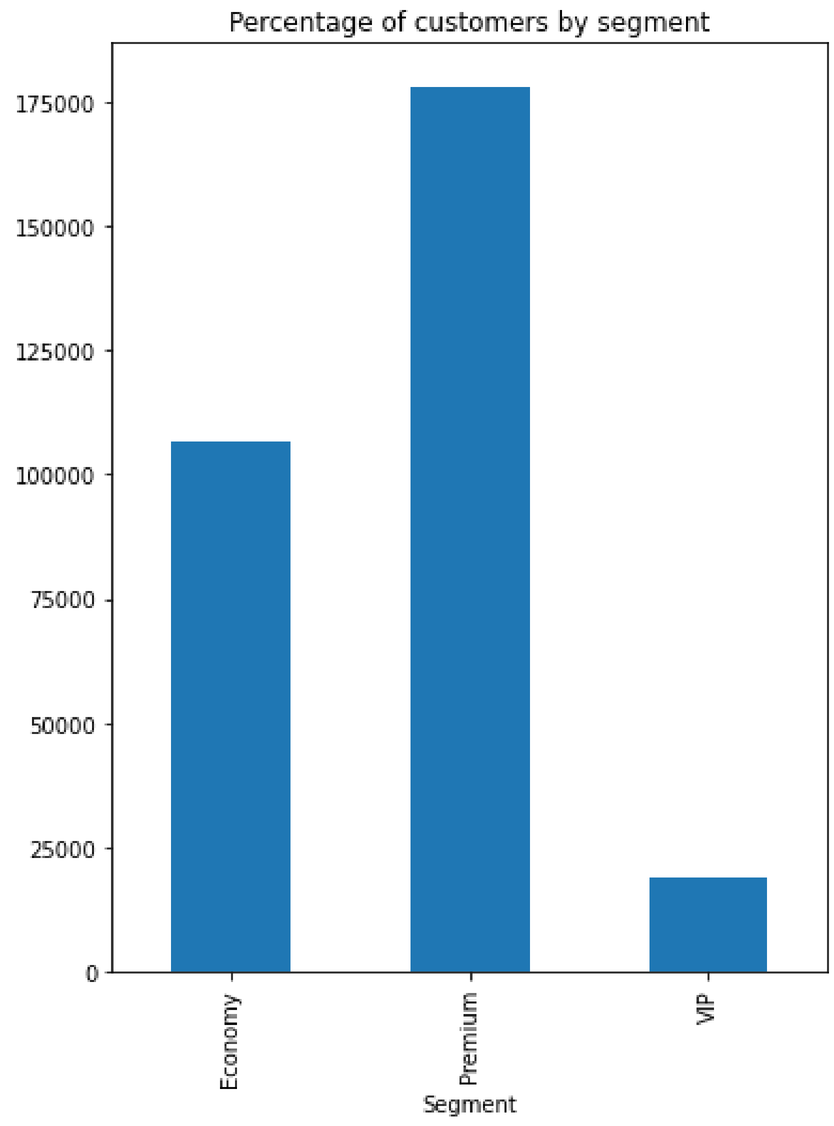
```
df['Segment'] = 'Economy'
df.loc[(df['rank'] < 7) & (df['rank'] >= 4), 'Segment'] = 'Premium'
df.loc[df['rank'] >= 7, 'Segment'] = 'VIP'
```

In [53]:

```
df.groupby('Segment').CustomerID.count().plot.bar(figsize = (6, 8))
plt.title('Percentage of customers by segment')
```

Out[53]:

```
Text(0.5, 1.0, 'Percentage of customers by segment')
```

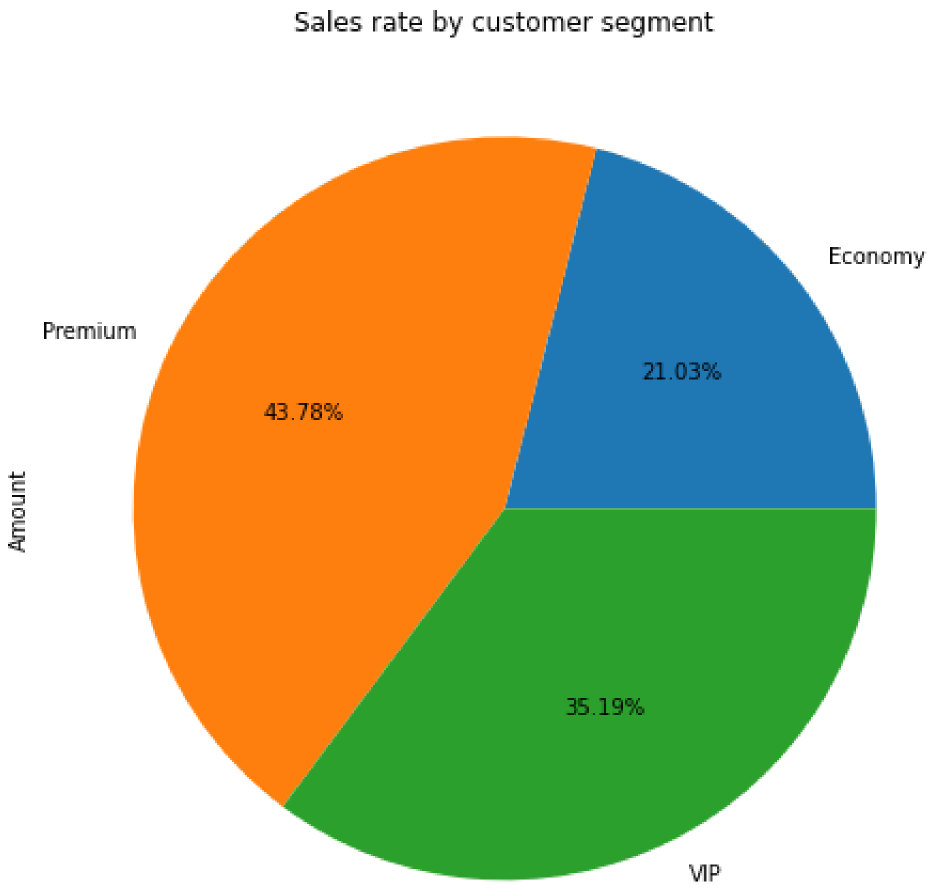


In [15]:

```
df.groupby('Segment').Amount.sum().plot.pie(autopct = '%.2f%%', figsize = (8, 8))
plt.title('Sales rate by customer segment')
```

Out[15]:

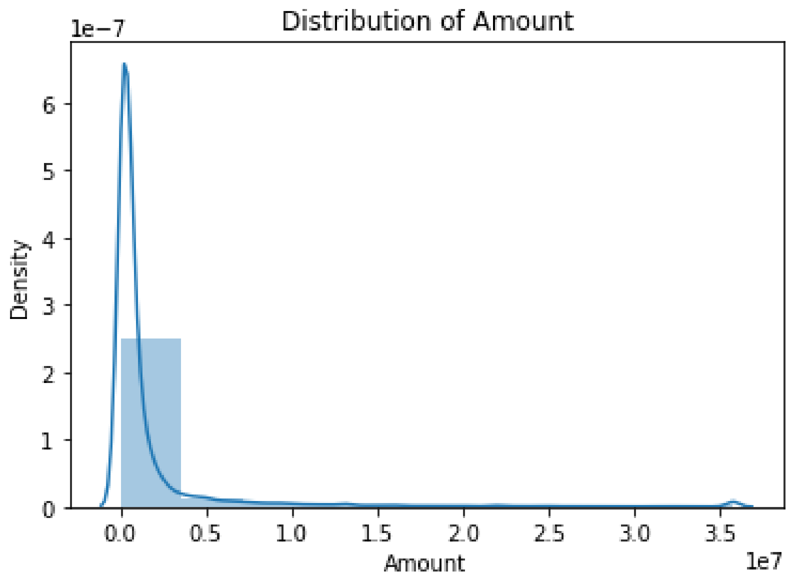
```
Text(0.5, 1.0, 'Sales rate by customer segment')
```



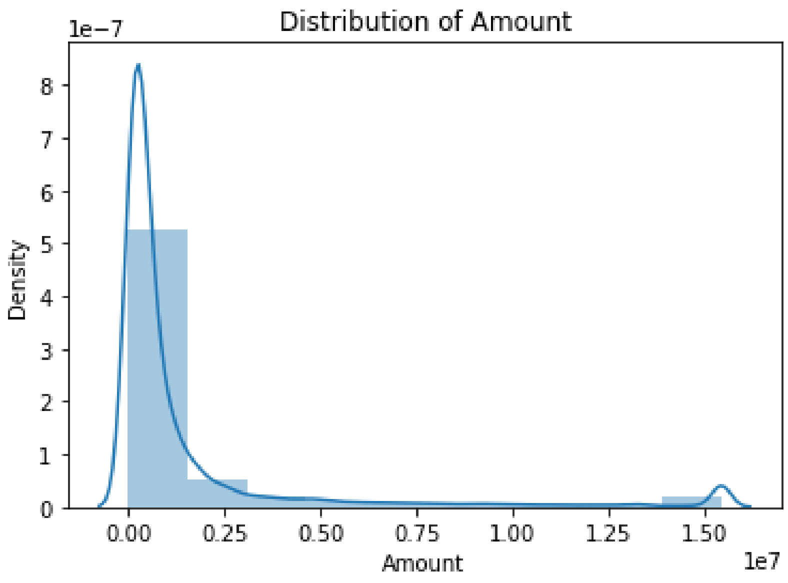
In [23]:

```python
# Thống kê mô tả
print(df.describe())

# Vẽ biểu đồ phân phối các biến
import seaborn as sns
import matplotlib.pyplot as plt

def _plotNumeric(colname, n_bins = 10, hist = True, kde = True):
    sns.distplot(df[colname], hist = hist, kde = kde, bins = n_bins)
    plt.title('Distribution of {}'.format(colname))
    plt.show()

_plotNumeric('Amount', hist = True, kde = True, n_bins = 10)
import numpy as np
def _fillOutlier(colname):
    mu = np.mean(df[colname])
    sigma = np.std(df[colname])
    x_min = max(mu - 3*sigma, 0)
    x_max = mu + 3*sigma
    print('x_min: ', x_min)
    print('x_max: ', x_max)
    out_lower_id = df[df[colname] < x_min].index
    out_upper_id = df[df[colname] > x_max].index
    df[colname].loc[out_lower_id] = x_min
    df[colname].loc[out_upper_id] = x_max

_fillOutlier('Amount')
_plotNumeric('Amount', hist = True, kde = True, n_bins = 10)
```

```
        Unnamed: 0        Amount       Recency     Frequency  \
count  303295.000000  3.032920e+05  303295.000000  303295.000000
mean   151647.000000  1.762908e+06    -613.695030      39.649058
std     87553.869284  4.555974e+06      34.567504     273.940159
min         0.000000  0.000000e+00    -691.000000       1.000000
25%     75823.500000  1.962000e+05    -642.000000       1.000000
50%    151647.000000  4.300000e+05    -606.000000       3.000000
75%    227470.500000  1.098000e+06    -582.000000       7.000000
max    303294.000000  3.573570e+07    -571.000000    2807.000000

        rFrequency      Monetary      rRecency     rMonetary  \
count  303295.000000  3.032950e+05  303295.000000  303295.000000
mean        4.499998  1.532548e+08       4.475685       4.499405
std         2.872293  1.793786e+09       2.873986       2.872703
min         0.000000 -5.500000e+05       0.000000       0.000000
25%         2.000000  6.040000e+05       2.000000       2.000000
50%         4.000000  2.000100e+06       4.000000       4.000000
75%         7.000000  6.529000e+06       7.000000       7.000000
max         9.000000  4.987068e+10       9.000000       9.000000

               rank      FinalRank
count  303295.000000  303295.000000
mean        4.491696       4.162439
std         2.295621       2.319668
min         0.000000       0.000000
25%         2.666667       2.000000
50%         4.333333       4.000000
75%         6.333333       6.000000
max         9.000000       9.000000
C:\Users\Home\anaconda3\lib\site-packages\seaborn\distributions.py:2557: 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)
```



```
x_min:  0
x_max:  15430807.267419249
C:\Users\Home\anaconda3\lib\site-packages\pandas\core\indexing.py:1637: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  self._setitem_single_block(indexer, value, name)
C:\Users\Home\anaconda3\lib\site-packages\seaborn\distributions.py:2557: 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)
```



In [ ]: