Data Science Training

Default of credit card clients Data Set Project

Upload data set

In [18]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
from sklearn.preprocessing import minmax_scale
```

In [2]:
```
df = pd.read_csv('/home/gmelao/Desktop/default-of-credit-card-clients.csv')
df.columns = df.iloc[0]
df.drop(0, inplace = True)
df.set_index('ID', inplace = True)
pd.set_option('display.max_columns', 24)
pd.set_option('display.max_rows', 24)
```

Features and types

In [3]:
```
df.dtypes
```

Out[3]:
```
0
LIMIT_BAL                   object
SEX                         object
EDUCATION                   object
MARRIAGE                    object
AGE                         object
PAY_0                       object
PAY_2                       object
PAY_3                       object
PAY_4                       object
PAY_5                       object
PAY_6                       object
BILL_AMT1                   object
BILL_AMT2                   object
BILL_AMT3                   object
BILL_AMT4                   object
BILL_AMT5                   object
BILL_AMT6                   object
PAY_AMT1                    object
PAY_AMT2                    object
PAY_AMT3                    object
PAY_AMT4                    object
PAY_AMT5                    object
PAY_AMT6                    object
default payment next month  object
dtype: object
```

Useful Info

Gender (1 = male; 2 = female)

Education (1 = graduate school; 2 = university; 3 = high school; 4 = others).

Marital status (1 = married; 2 = single; 3 = others).

The measurement scale for the repayment status is: -1 = pay duly; 1 = payment delay for one month; 2 = payment delay for two months; . . .; 8 = payment delay for eight months; 9 = payment delay for nine months and above.

In [4]:
```
df.head()
```

Out[4]:

ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_0	PAY_2	PAY_3	PAY_4	PAY_5	PAY_6
1	20000	2	2	1	24	2	2	-1	-1	-2	-2
2	120000	2	2	2	26	-1	2	0	0	0	2
3	90000	2	2	2	34	0	0	0	0	0	0
4	50000	2	2	1	37	0	0	0	0	0	0
5	50000	1	2	1	57	-1	0	-1	0	0	0

In [5]:
```
df[['BILL_AMT1', 'PAY_AMT1']].head()
```

Out[5]:

ID	BILL_AMT1	PAY_AMT1
1	3913	0
2	2682	0
3	29239	1518
4	46990	2000
5	8617	2000

In [6]:
```
df[['BILL_AMT2', 'PAY_AMT2']].head()
```

Out[6]:

ID	BILL_AMT2	PAY_AMT2
1	3102	689
2	1725	1000
3	14027	1500
4	48233	2019
5	5670	36681

FIrst data treatment

In [7]:
```
df = df.apply(lambda df: pd.Series(map(float, df)))
```

In [8]:
```
df.dtypes
```

Out[8]:
```
0
LIMIT_BAL                     float64
SEX                           float64
EDUCATION                     float64
MARRIAGE                      float64
AGE                           float64
PAY_0                         float64
PAY_2                         float64
PAY_3                         float64
PAY_4                         float64
PAY_5                         float64
PAY_6                         float64
BILL_AMT1                     float64
BILL_AMT2                     float64
BILL_AMT3                     float64
BILL_AMT4                     float64
BILL_AMT5                     float64
BILL_AMT6                     float64
PAY_AMT1                      float64
PAY_AMT2                      float64
PAY_AMT3                      float64
PAY_AMT4                      float64
PAY_AMT5                      float64
PAY_AMT6                      float64
default payment next month    float64
dtype: object
```

In [9]:
```
df_no_cats = df.drop(['SEX', 'MARRIAGE', 'EDUCATION', 'default payment next month']
df_no_cats.describe()
```

Out[9]:

	LIMIT_BAL	AGE	PAY_0	PAY_2	PAY_3	PAY_4	
count	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	300
mean	167484.322667	35.485500	-0.016700	-0.133767	-0.166200	-0.220667	
std	129747.661567	9.217904	1.123802	1.197186	1.196868	1.169139	
min	10000.000000	21.000000	-2.000000	-2.000000	-2.000000	-2.000000	
25%	50000.000000	28.000000	-1.000000	-1.000000	-1.000000	-1.000000	
50%	140000.000000	34.000000	0.000000	0.000000	0.000000	0.000000	
75%	240000.000000	41.000000	0.000000	0.000000	0.000000	0.000000	
max	1000000.000000	79.000000	8.000000	8.000000	8.000000	8.000000	

In [10]:
```
df_cats = df[['SEX', 'MARRIAGE', 'EDUCATION', 'default payment next month']].astype
df_cats.describe()
```

Out[10]:

	SEX	MARRIAGE	EDUCATION	default payment next month
count	30000.0	30000.0	30000.0	30000.0
unique	2.0	4.0	7.0	2.0
top	2.0	2.0	2.0	0.0
freq	18112.0	15964.0	14030.0	23364.0

Data Quality

Uniqueness

Verify if duplicated values exists

In [13]:
```
df2 = df_no_cats.apply(lambda df: df.duplicated(), axis=1)
df2.sum()
```

Out[13]:
```
0
LIMIT_BAL        0
AGE              0
PAY_0            0
PAY_2        23136
PAY_3        25005
PAY_4        26043
PAY_5        26808
PAY_6        27264
BILL_AMT1      303
BILL_AMT2     3022
BILL_AMT3     3718
BILL_AMT4     4314
BILL_AMT5     4790
BILL_AMT6     5300
PAY_AMT1      8453
PAY_AMT2     10590
PAY_AMT3     11708
PAY_AMT4     13264
PAY_AMT5     14801
PAY_AMT6     11707
dtype: int64
```

In [62]:
```
df.groupby('PAY_6').size()
```

Out[62]:
```
PAY_6
-2.0     4895
-1.0     5740
 0.0    16286
 2.0     2766
 3.0      184
 4.0       49
 5.0       13
 6.0       19
 7.0       46
 8.0        2
dtype: int64
```

Completeness

Show how many null values in the data set

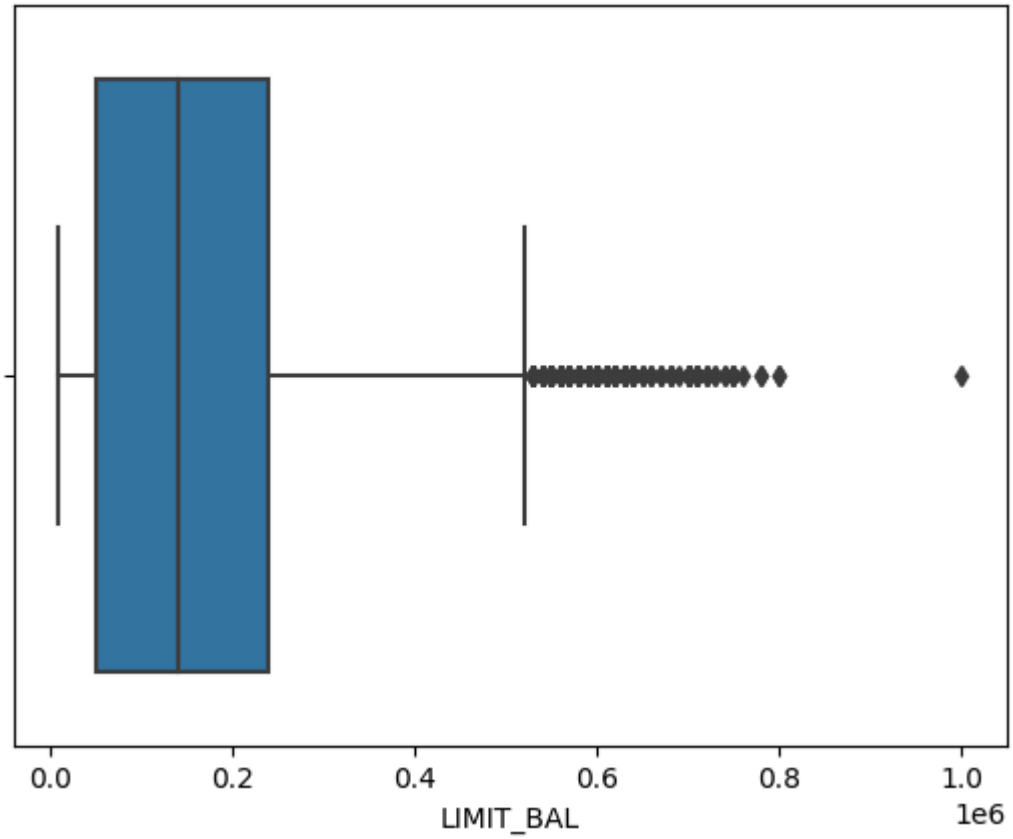
In [14]:
```
df.isna().sum()
```

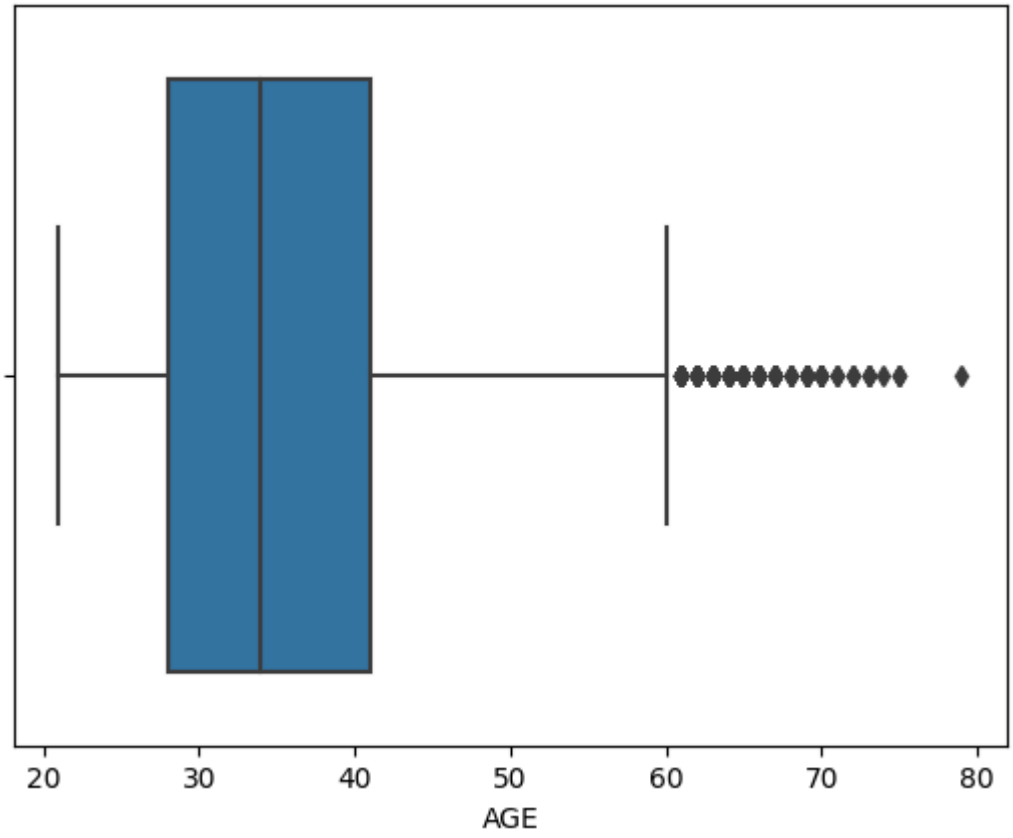
Out[14]:
```
0
LIMIT_BAL                     0
SEX                           0
EDUCATION                     0
MARRIAGE                      0
AGE                           0
PAY_0                         0
PAY_2                         0
PAY_3                         0
PAY_4                         0
PAY_5                         0
PAY_6                         0
BILL_AMT1                     0
BILL_AMT2                     0
BILL_AMT3                     0
BILL_AMT4                     0
BILL_AMT5                     0
BILL_AMT6                     0
PAY_AMT1                      0
PAY_AMT2                      0
PAY_AMT3                      0
PAY_AMT4                      0
PAY_AMT5                      0
PAY_AMT6                      0
default payment next month    0
dtype: int64
```

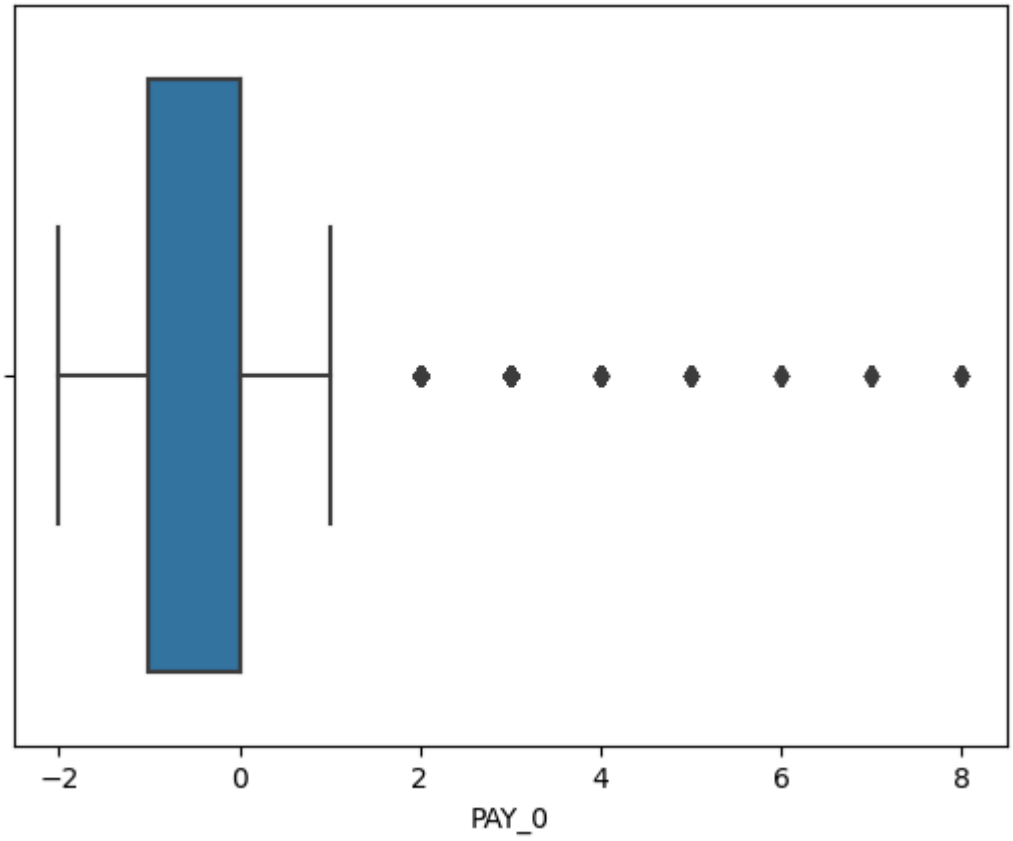
Boxplot

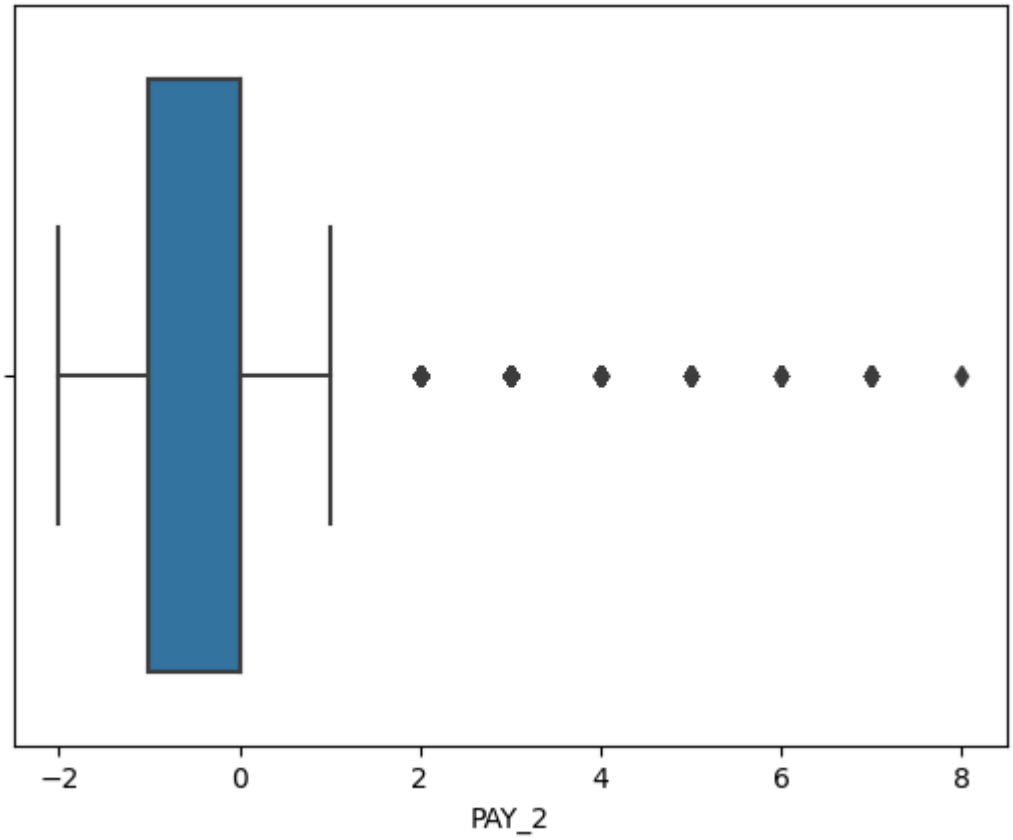
It's a method for graphically demonstrating the variation groups of numerical data through their quartiles. Boxplot graphs are useful to identify dospersion of data, simetry, outliers and positions.

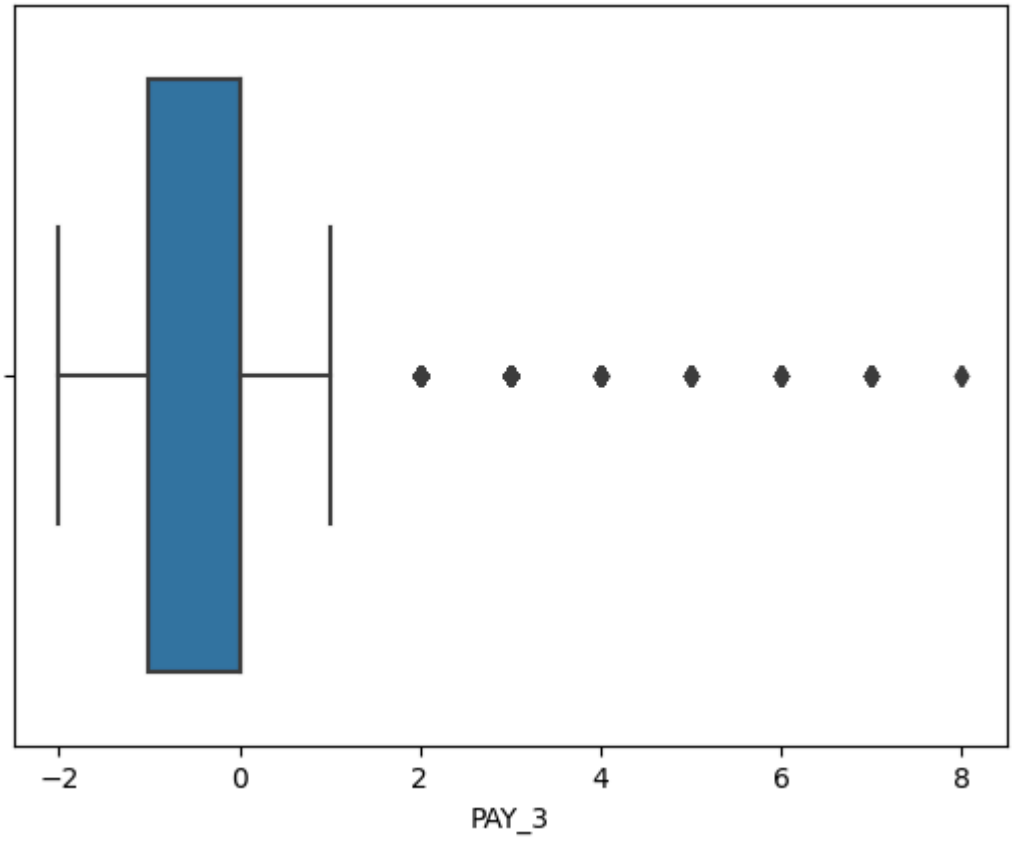
In [69]:
```
for c in df_no_cats.columns:
    sns.boxplot(df_no_cats, x=c)
    plt.show()
    plt.close()
```

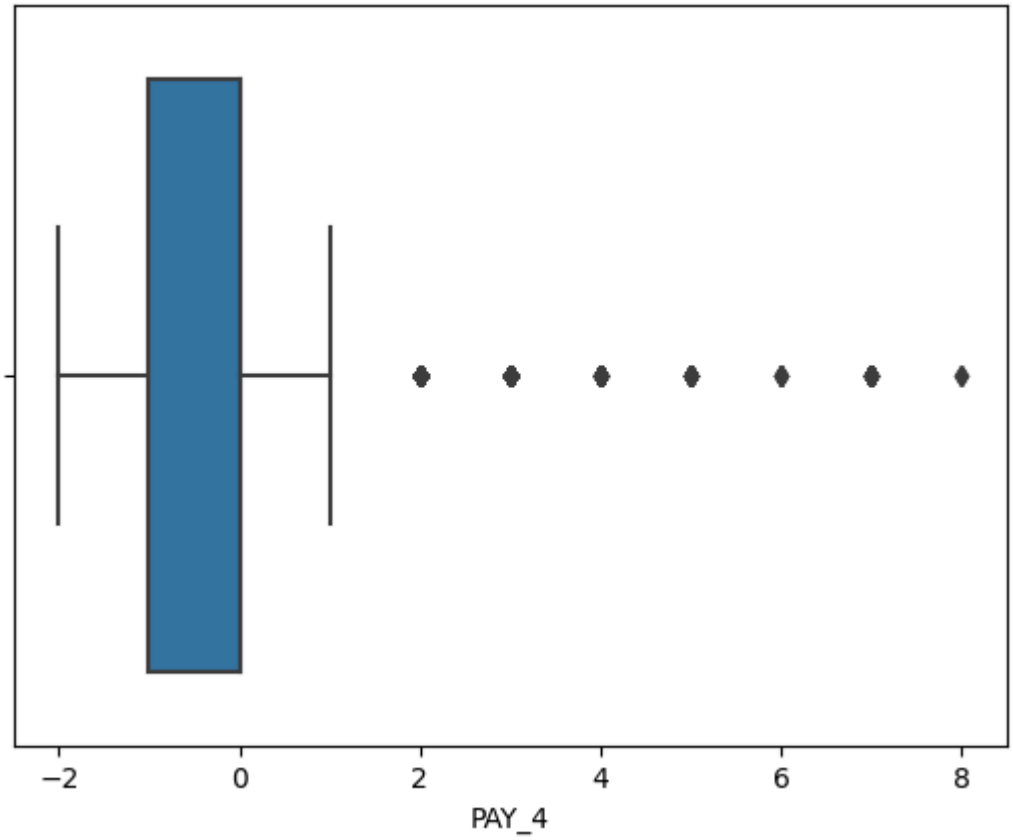

0.0
0.2
0.4
0.6
0.8
1.0
LIMIT_BAL
1e6

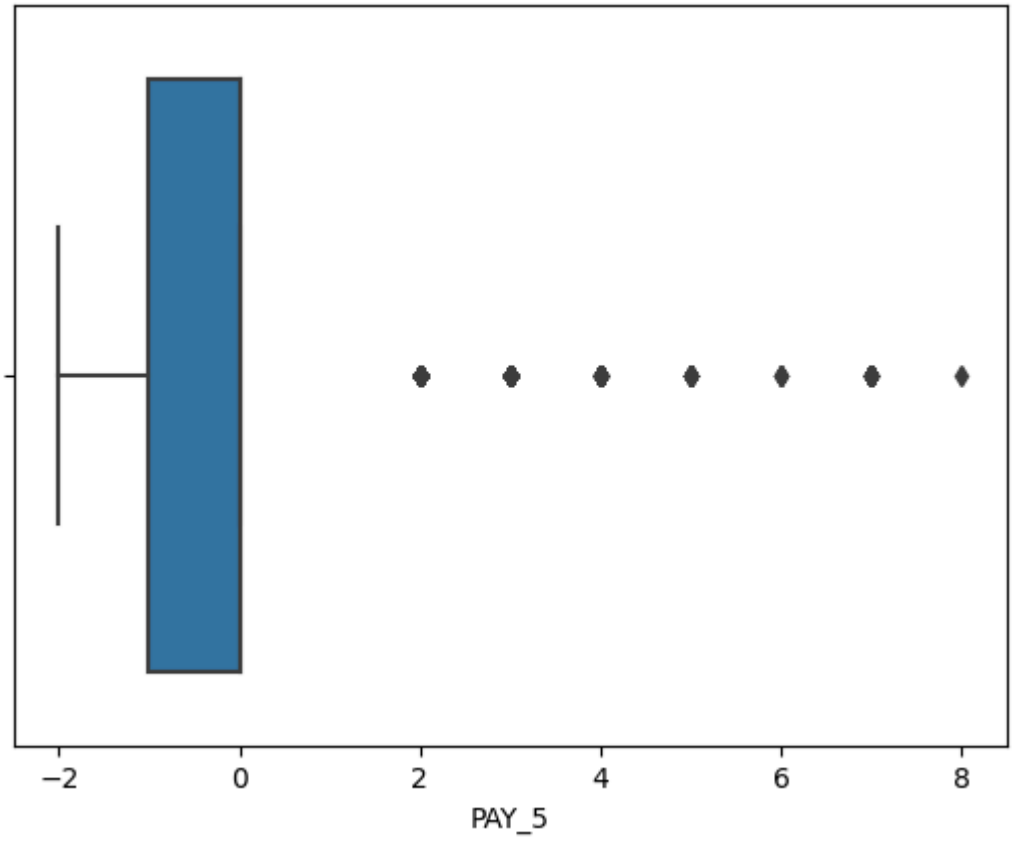

20
30
40
50
60
70
80
AGE

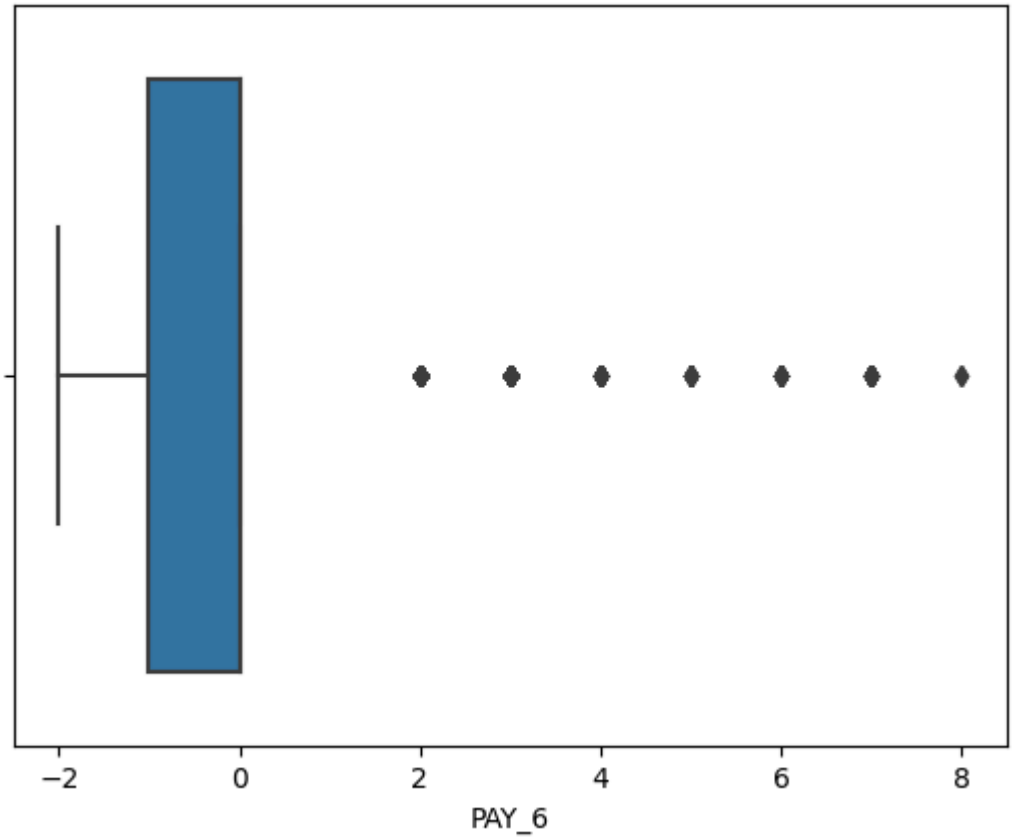

−2
0
2
4
6
8
PAY_0

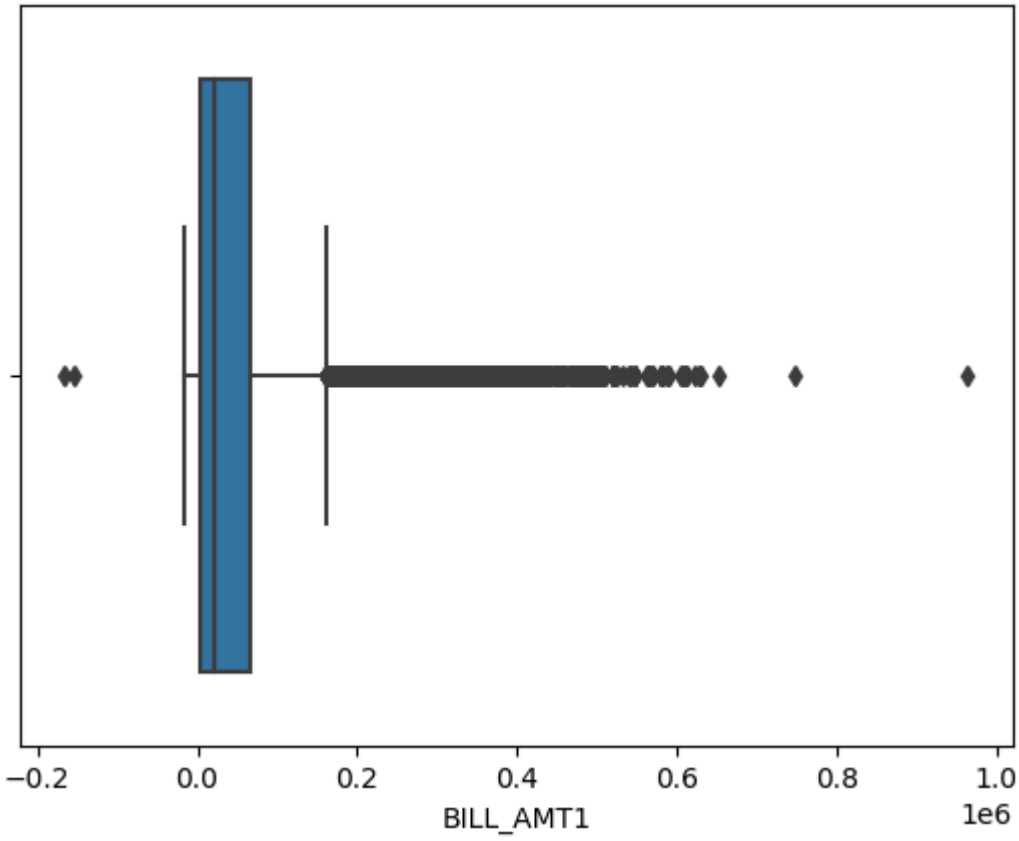

−2
0
2
4
6
8
PAY_2

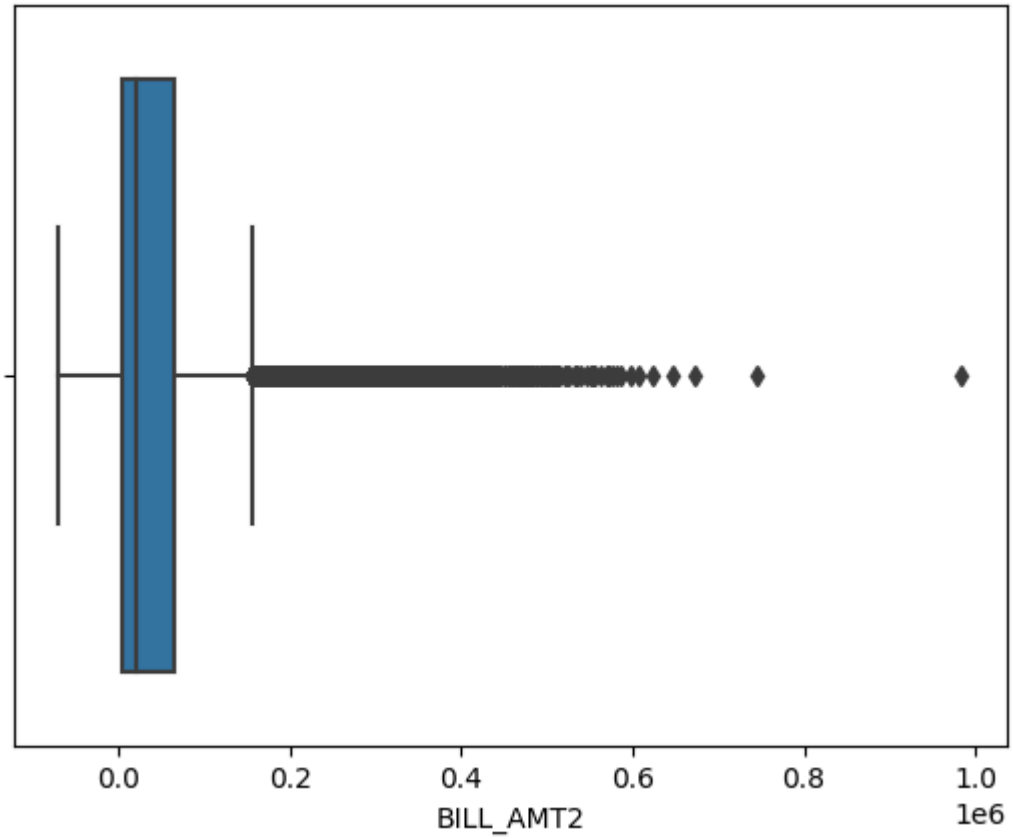

−2
0
2
4
6
8
PAY_3

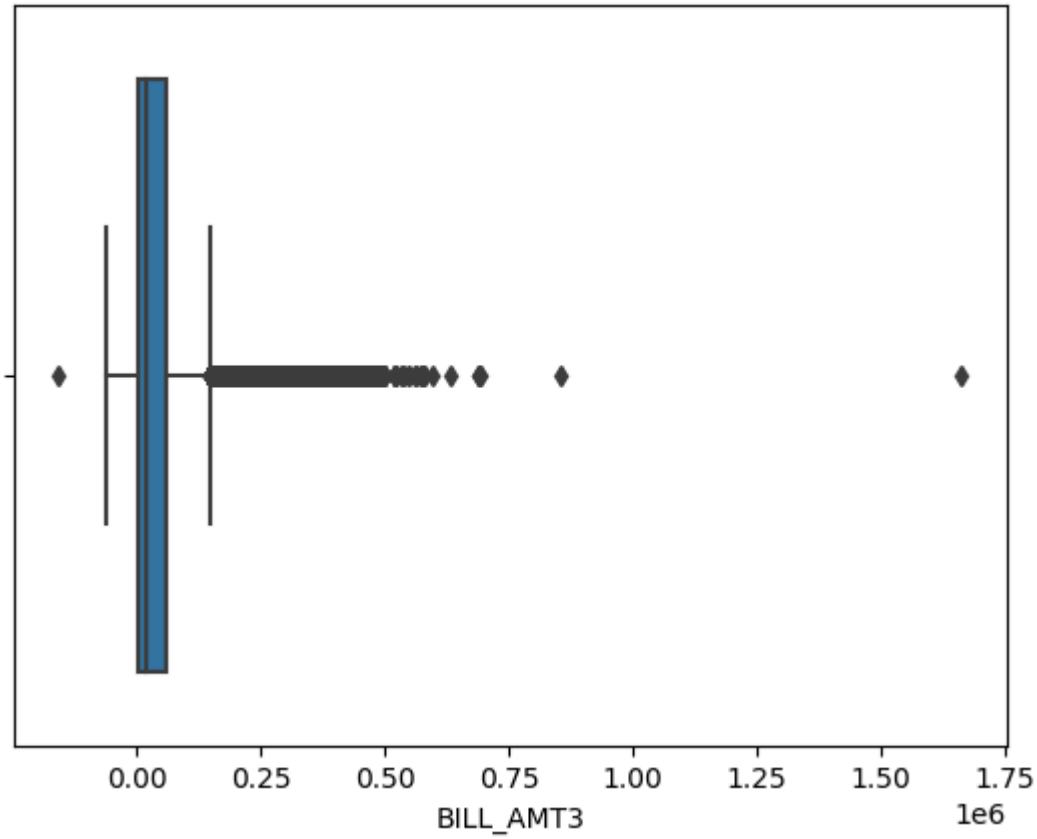

−2
0
2
4
6
8
PAY_4

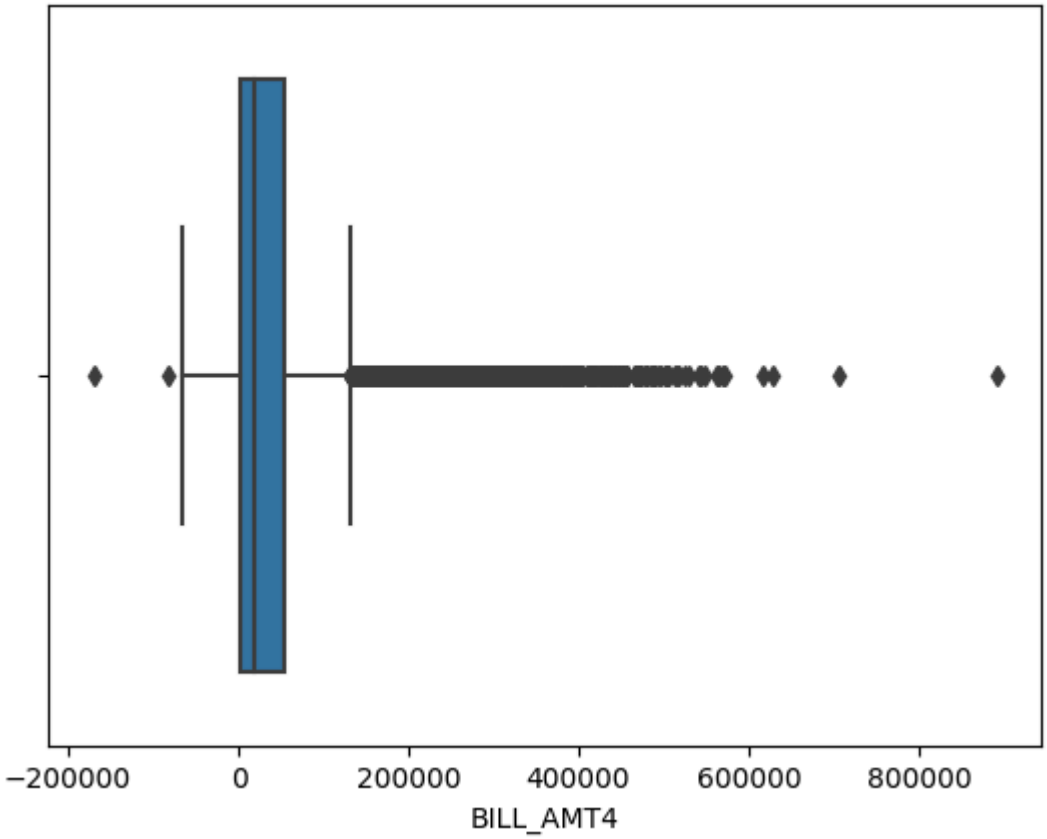

−2
0
2
4
6
8
PAY_5

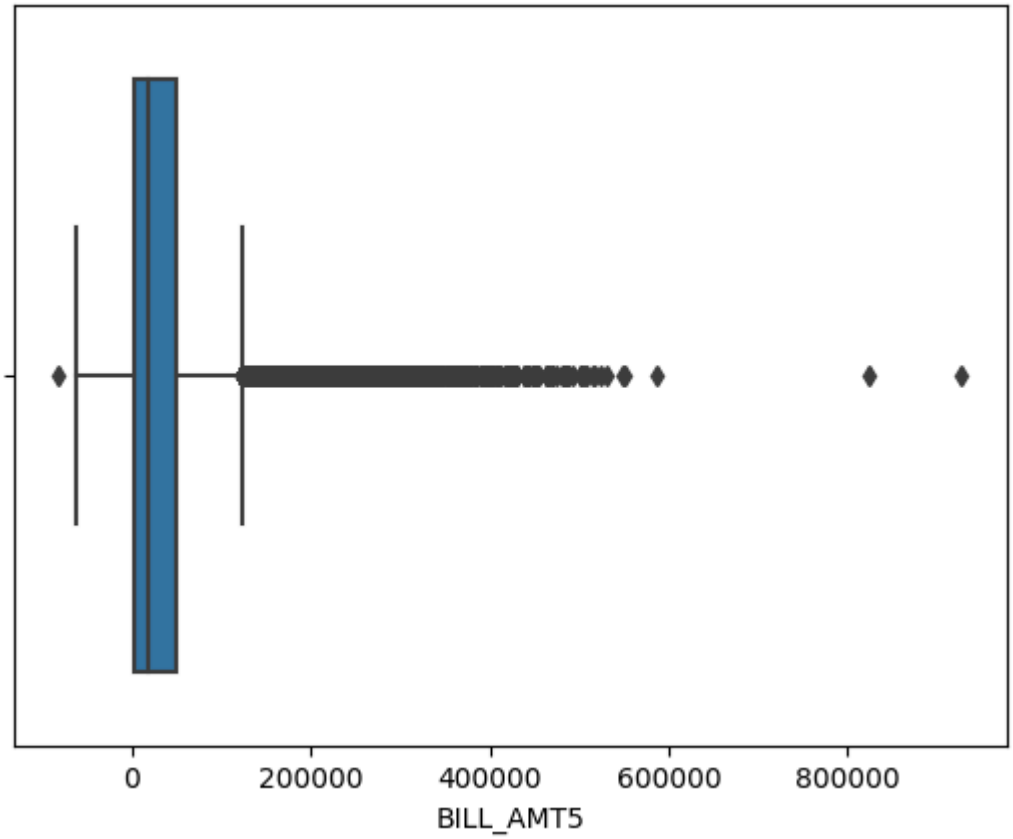

−2
0
2
4
6
8
PAY_6

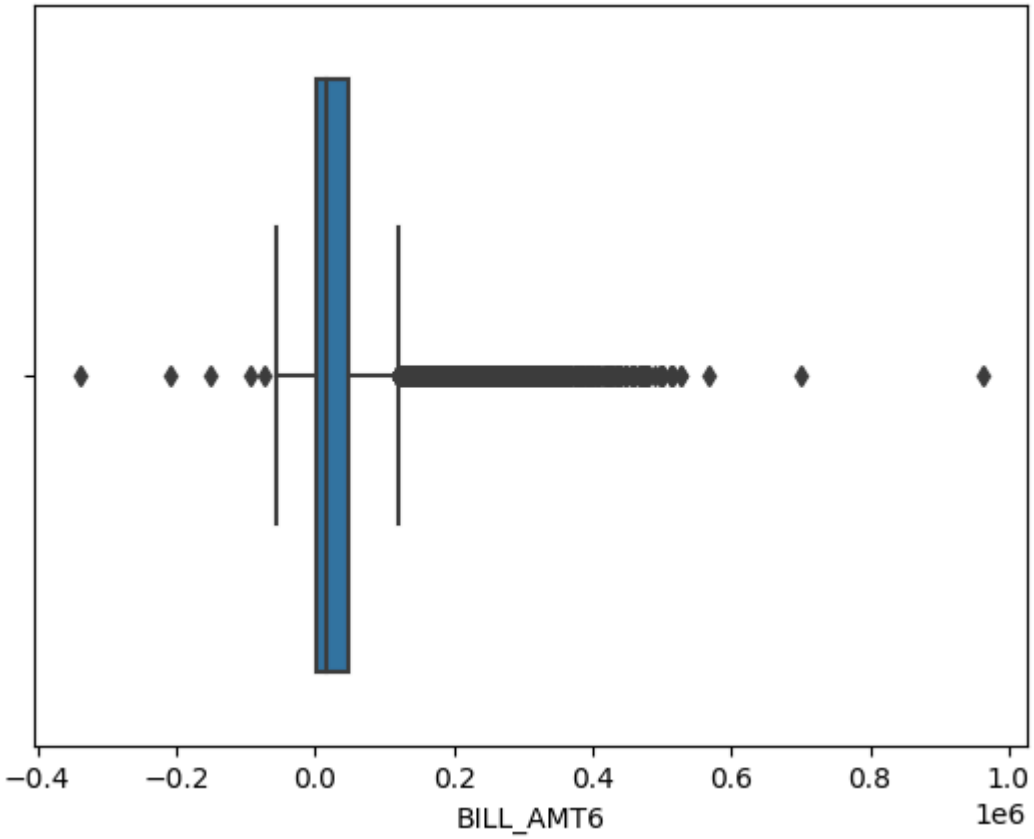

−0.2
0.0
0.2
0.4
0.6
0.8
1.0
1e6
BILL_AMT1

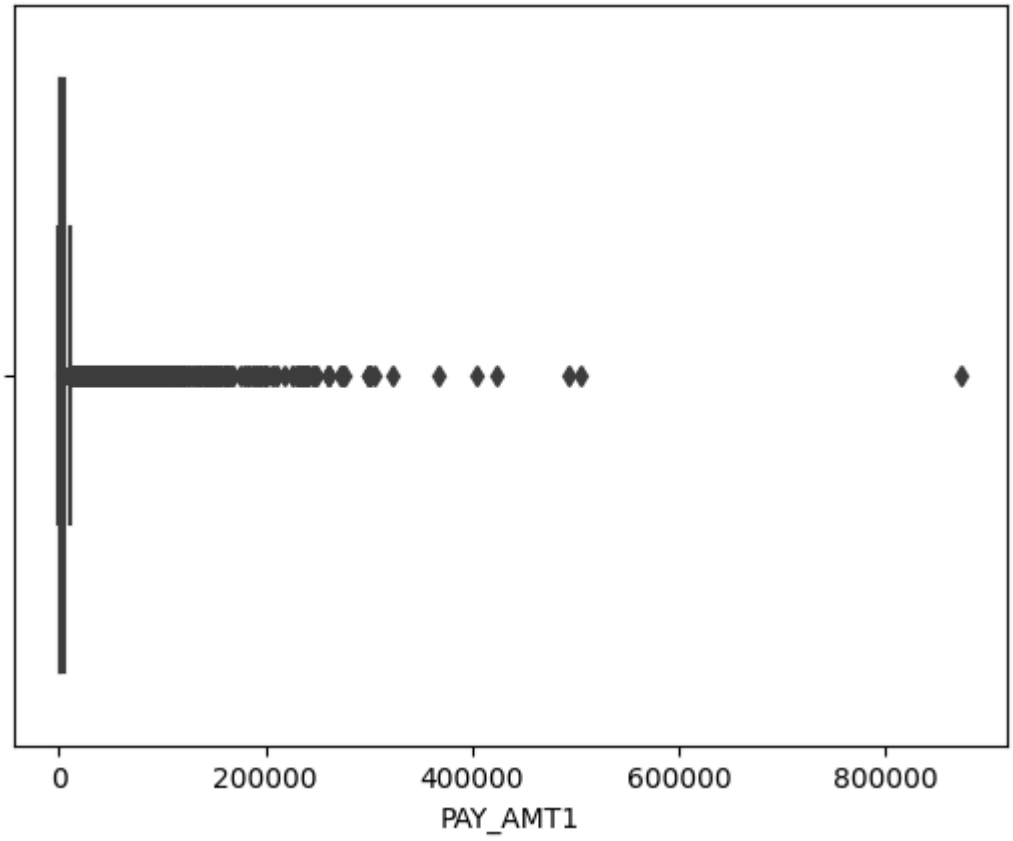

0.0
0.2
0.4
0.6
0.8
1.0
1e6
BILL_AMT2

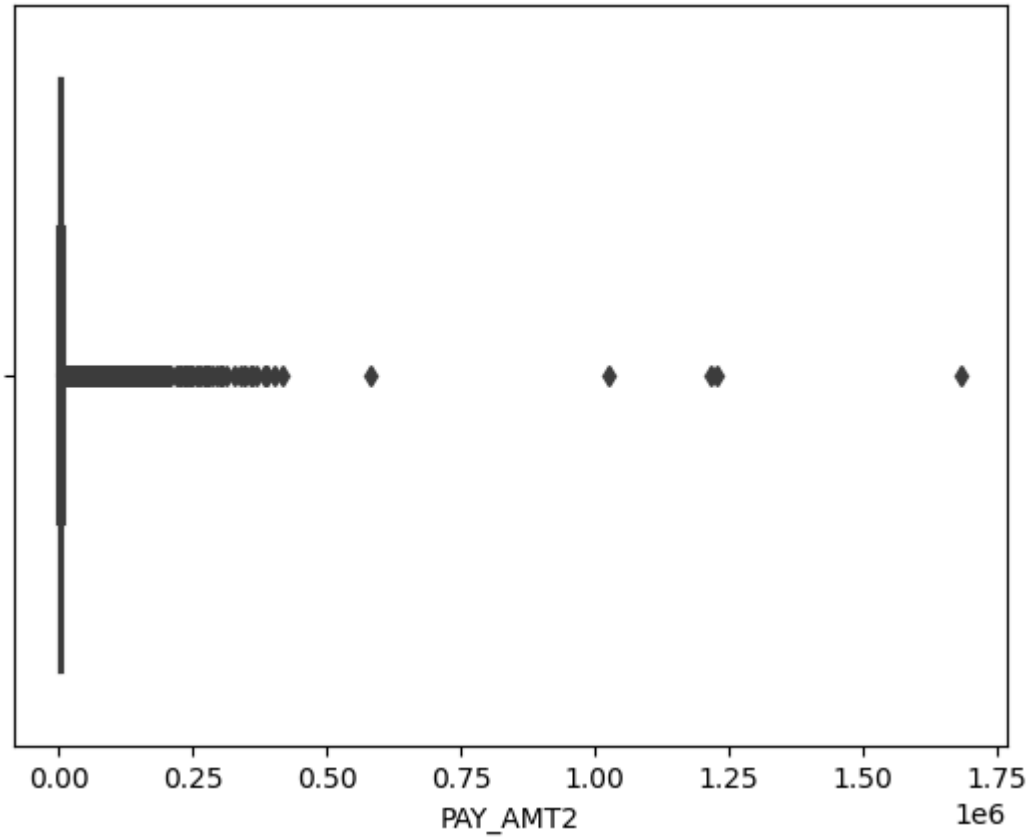

0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
BILL_AMT3
1e6

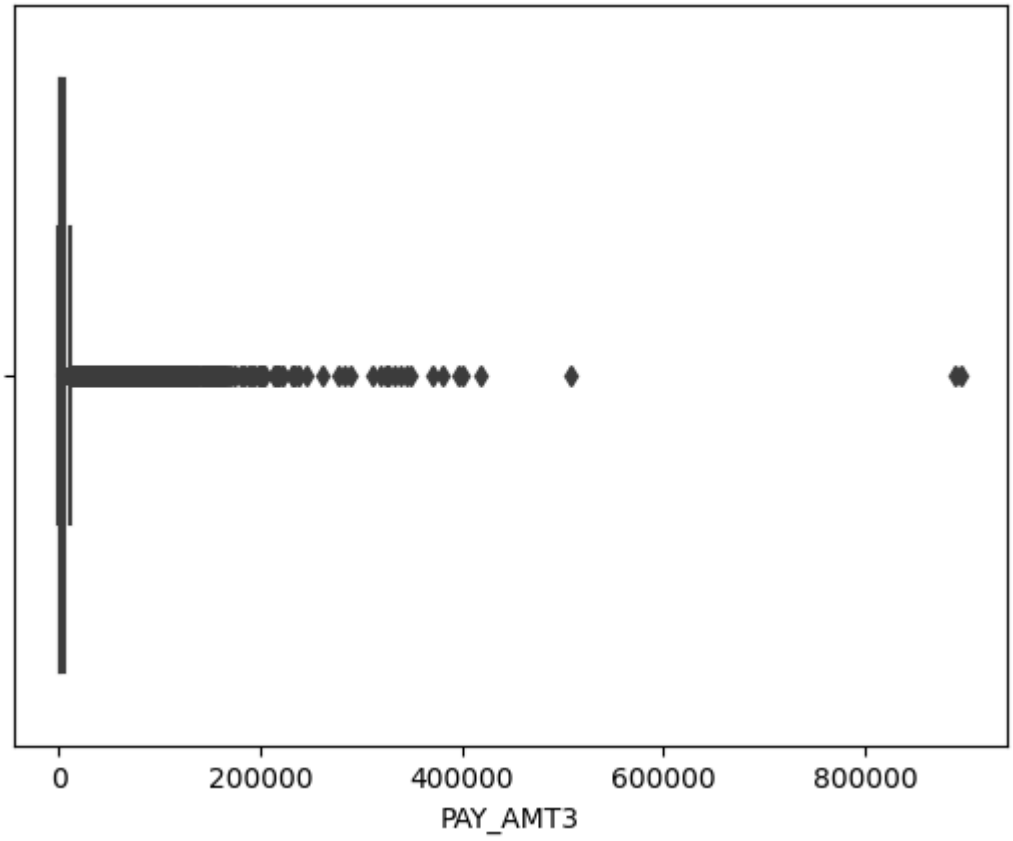

−200000
0
200000
400000
600000
800000
BILL_AMT4

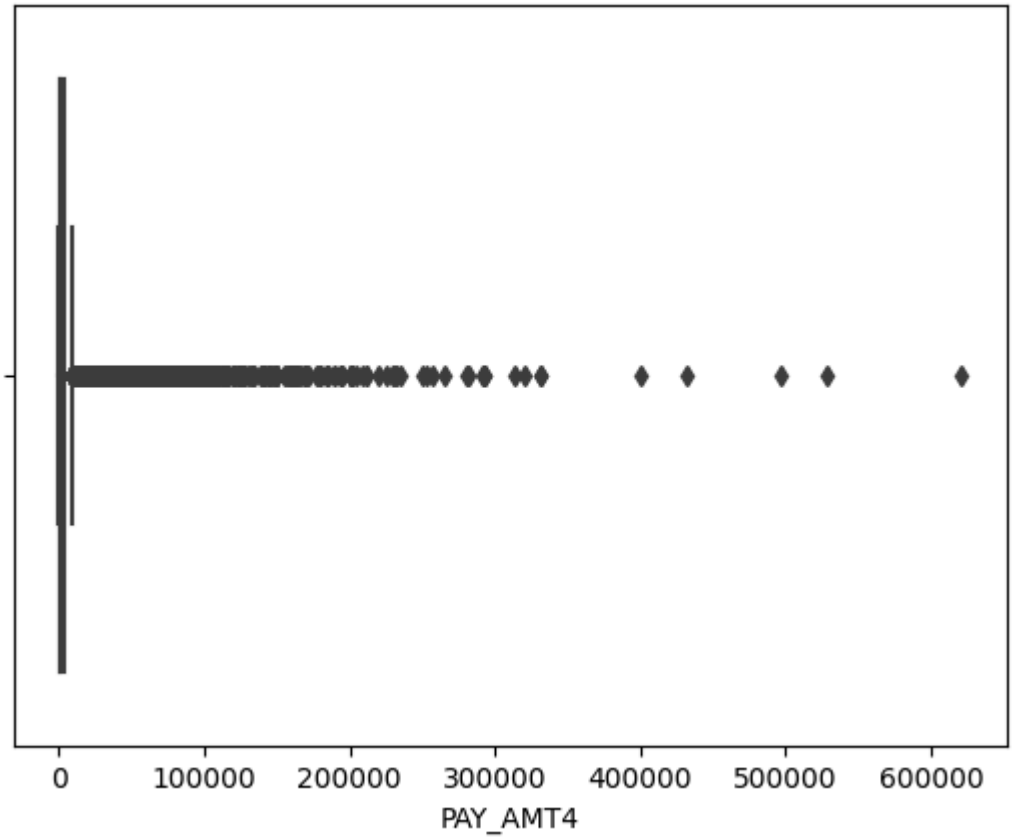

0
200000
400000
600000
800000
BILL_AMT5

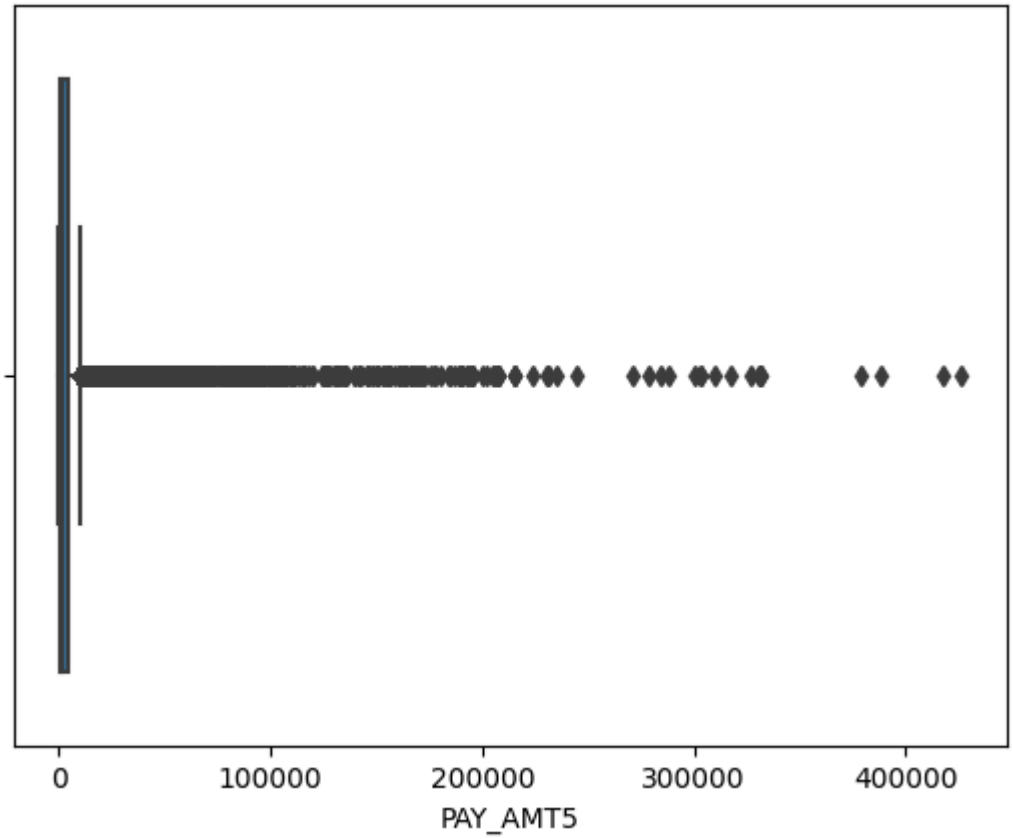

−0.4
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
1e6
BILL_AMT6

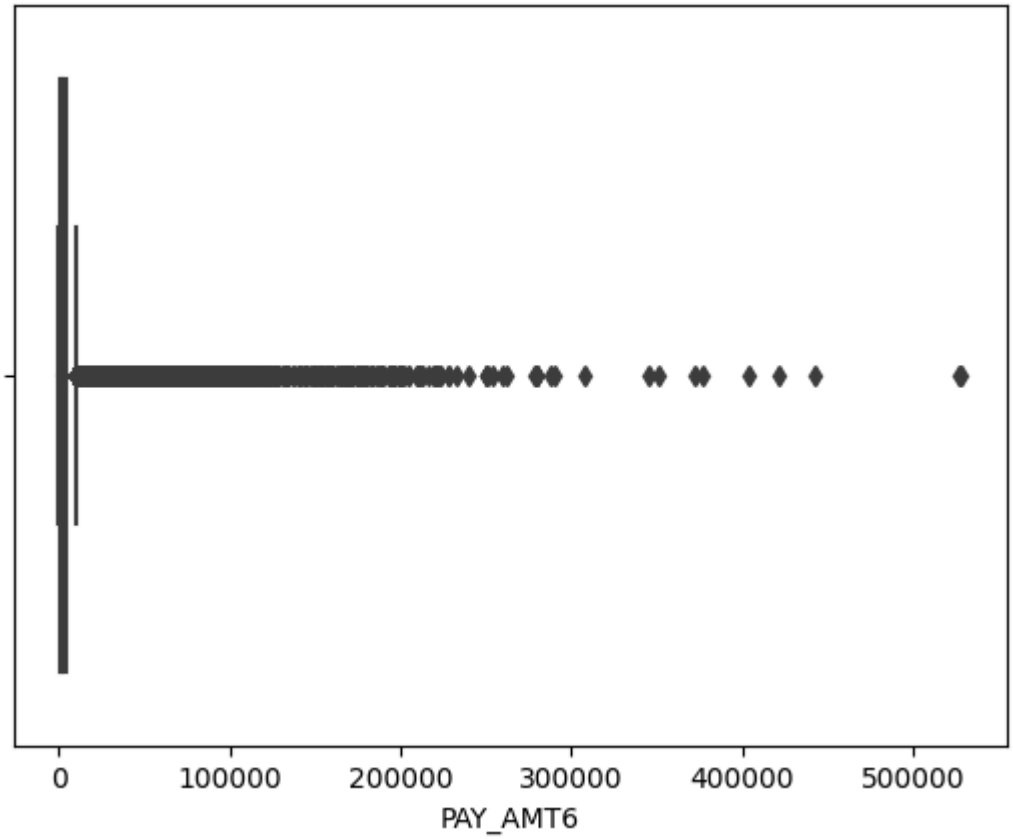

0
200000
400000
600000
800000
PAY_AMT1


0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
1e6
PAY_AMT2

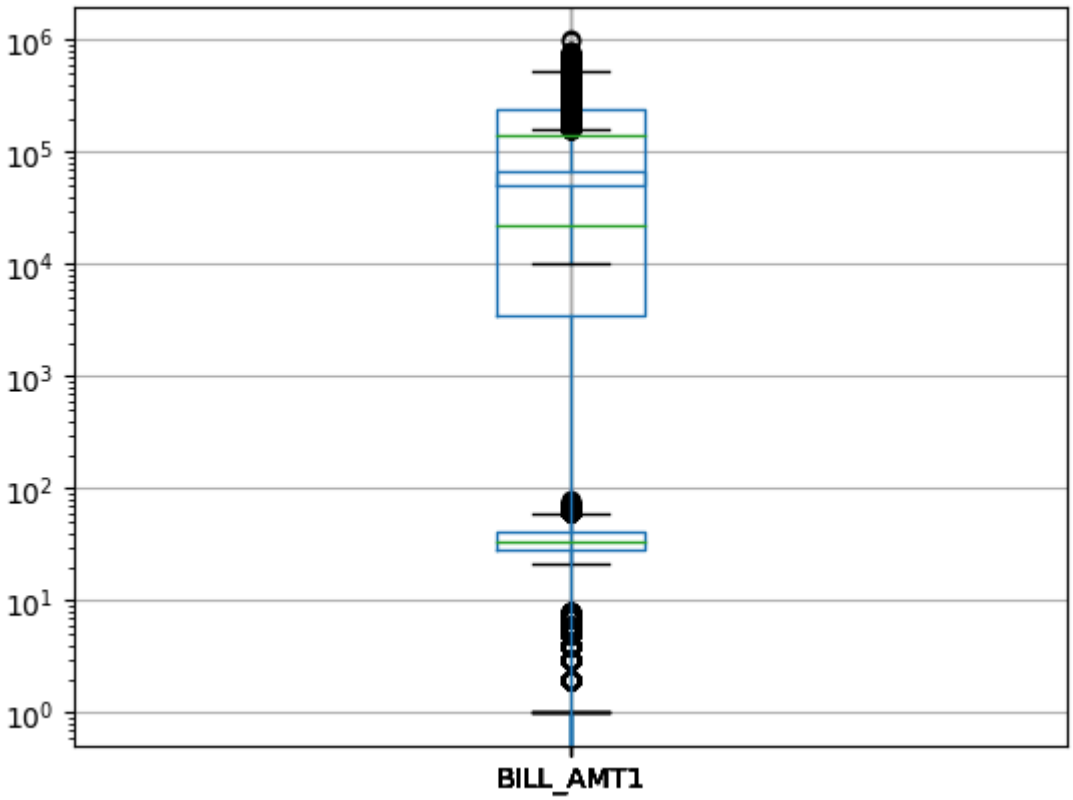

0
200000
400000
600000
800000
PAY_AMT3

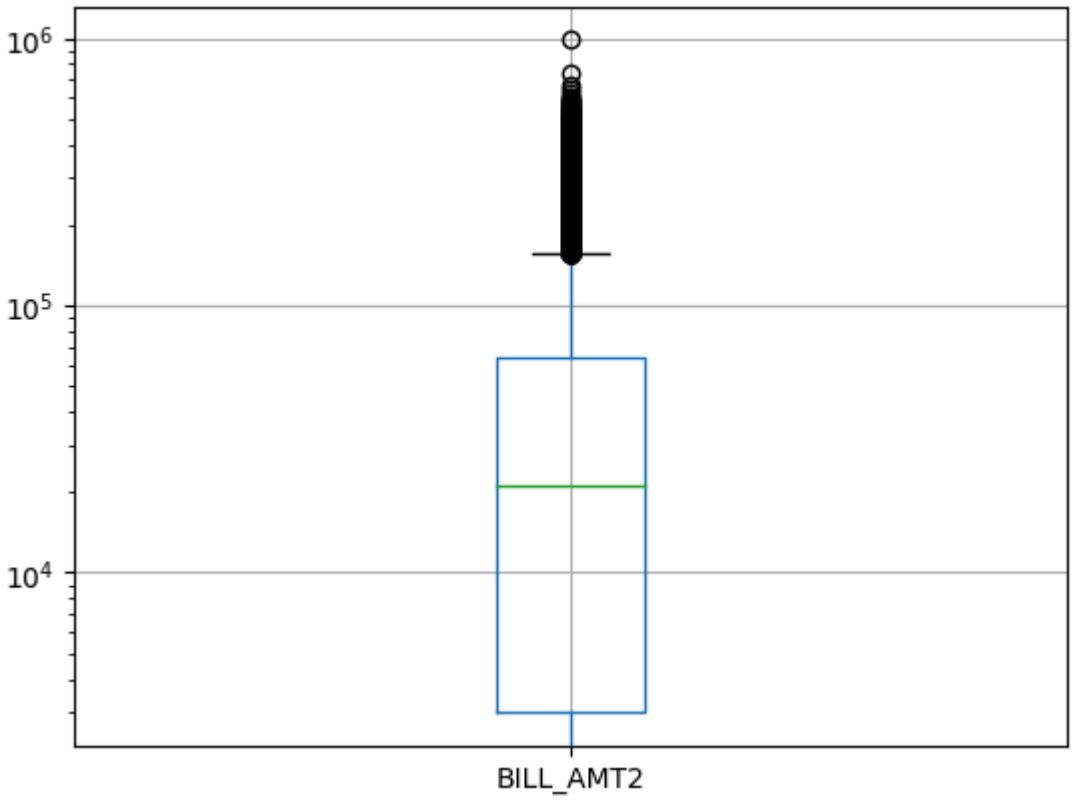

0
100000
200000
300000
400000
500000
600000
PAY_AMT4

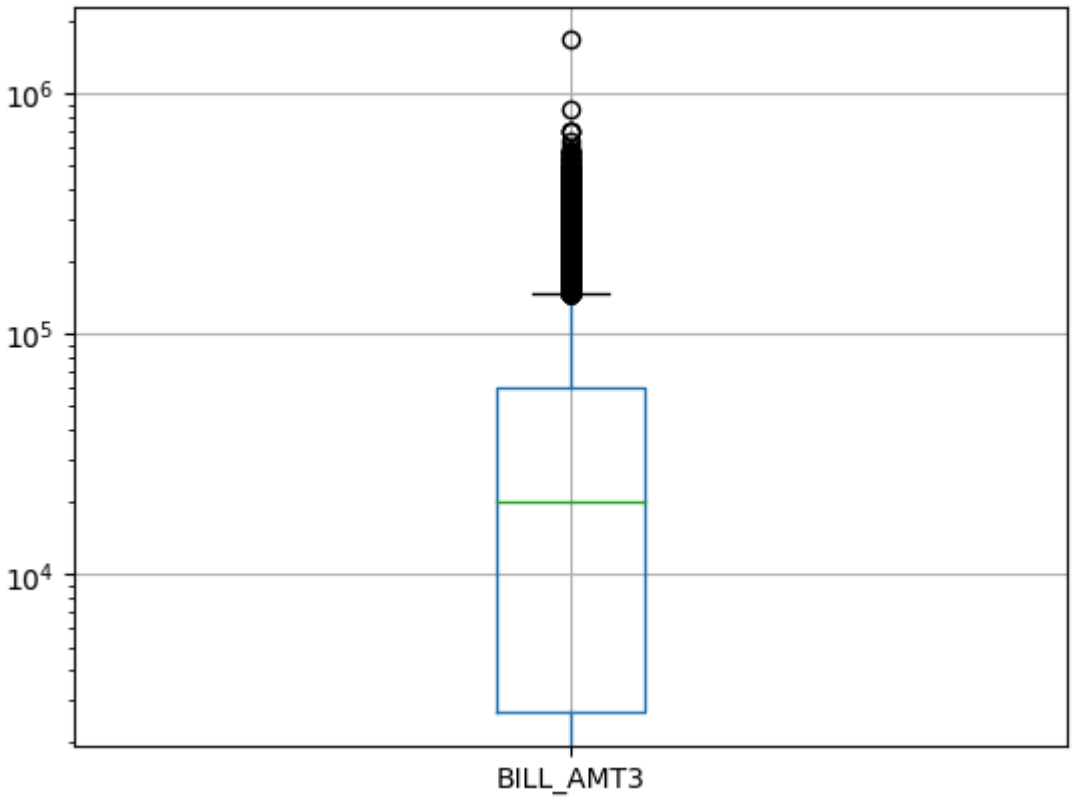


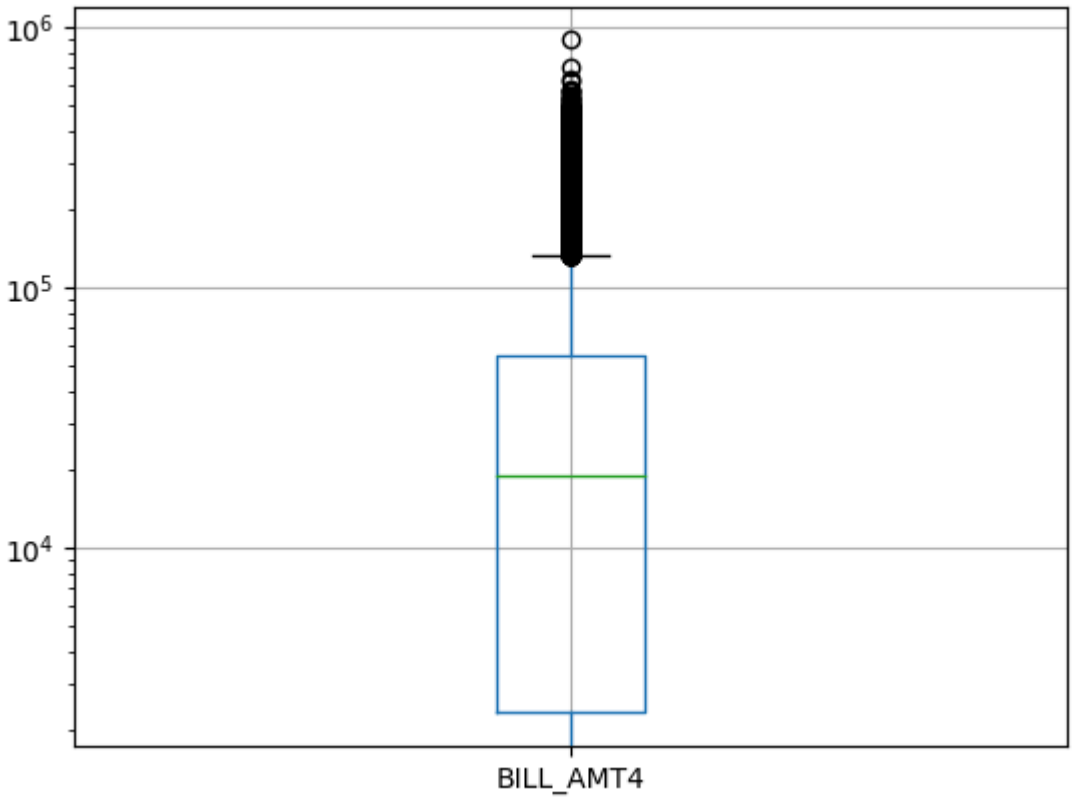


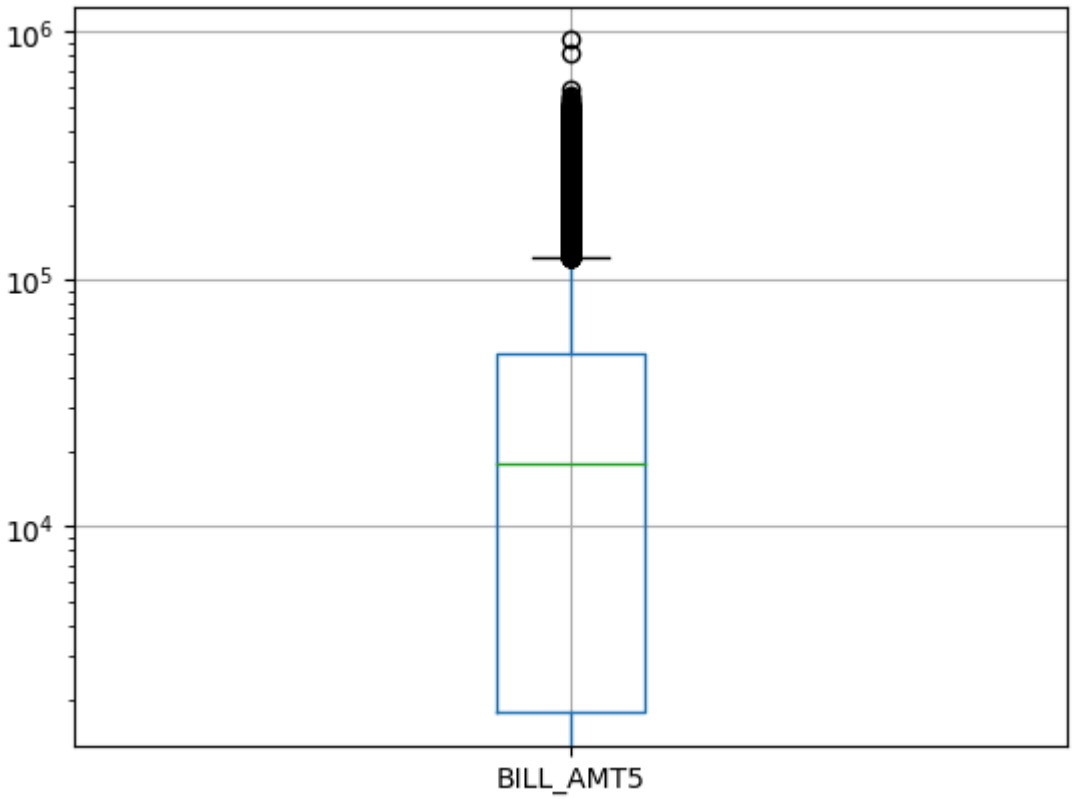
In [70]:
```
for c in df_no_cats.columns:
    ax = df_no_cats.boxplot(c)
    if c.startswith('BILL_AMT') or c.startswith('PAY_AMT'):
        ax.set_yscale("log")
        plt.show()
        plt.close()
```

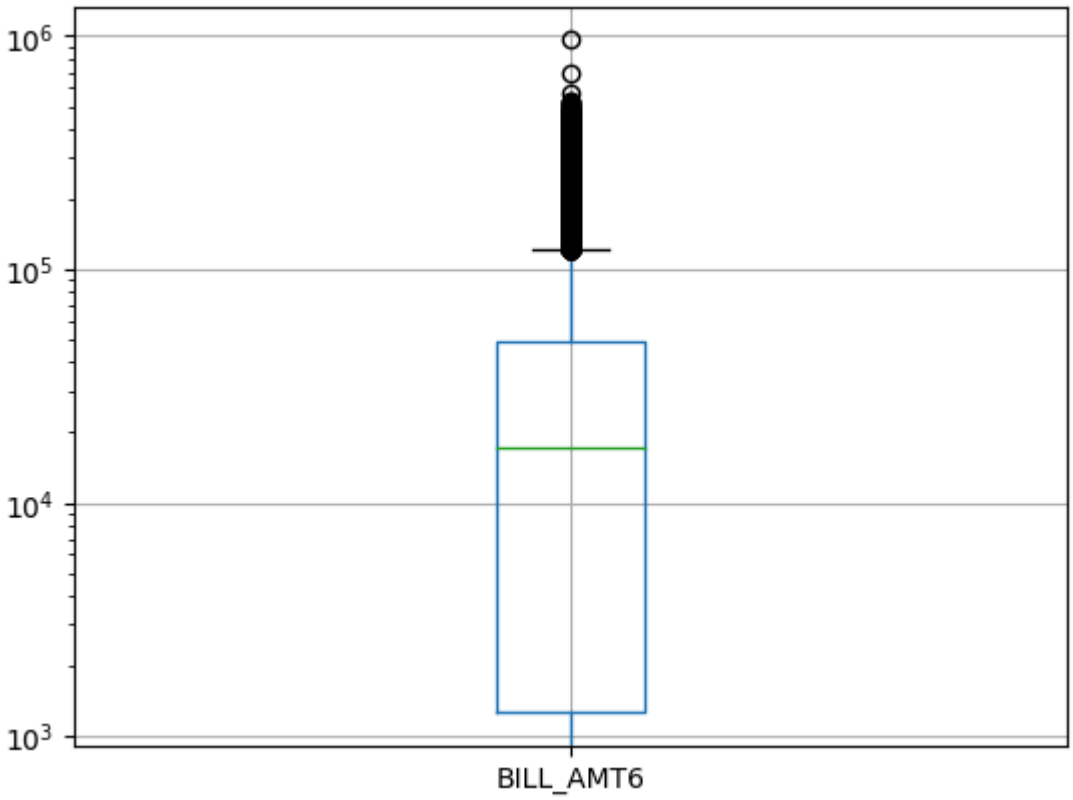

10^6
10^5
10^4
10^3
10^2
10^1
10^0
BILL_AMT1

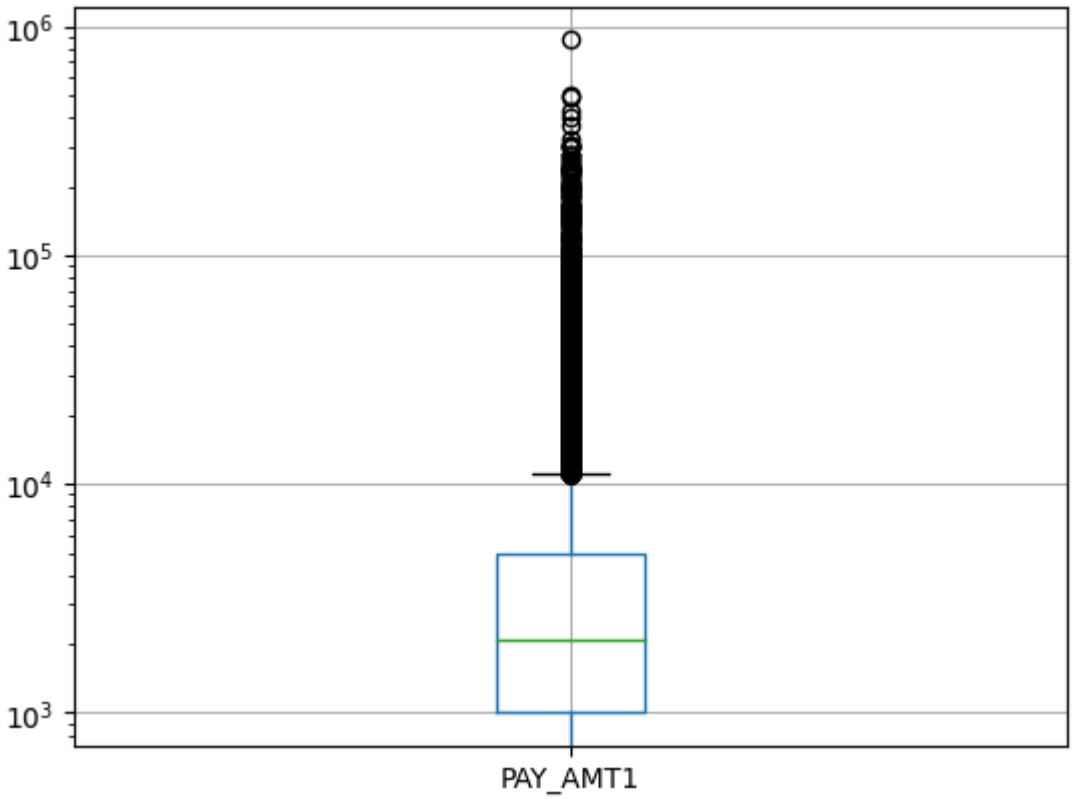

10^6
10^5
10^4
BILL_AMT2

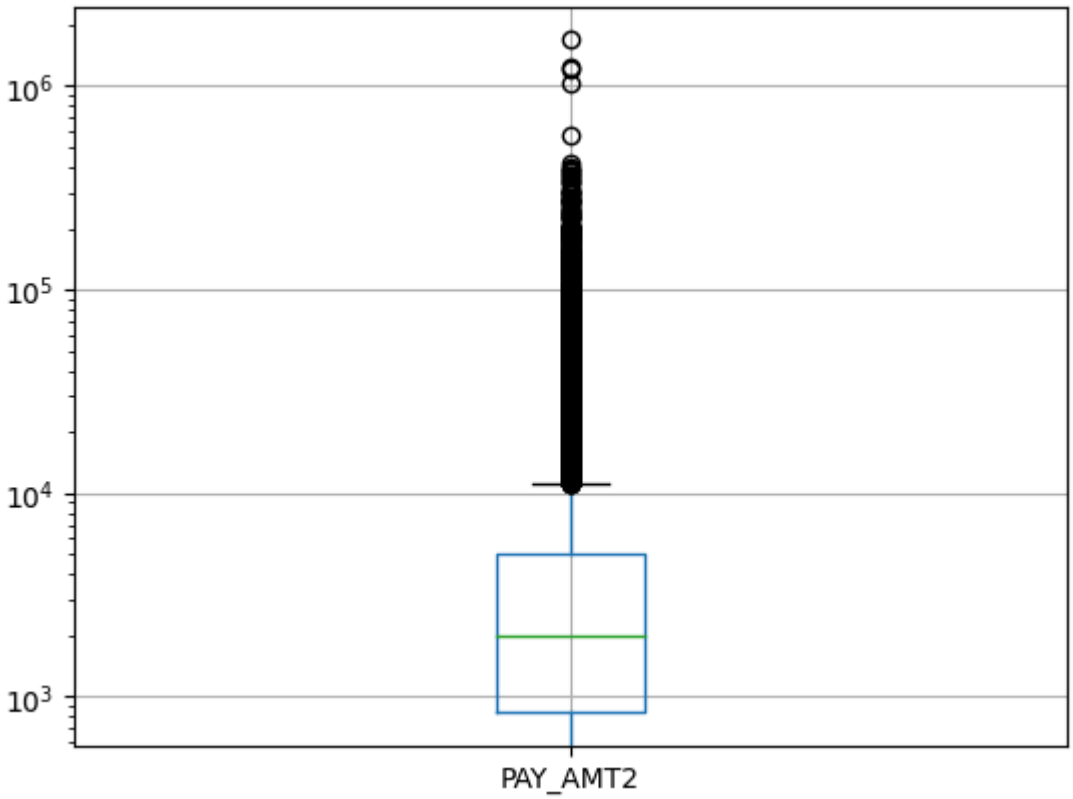

10^6
10^5
10^4
BILL_AMT3

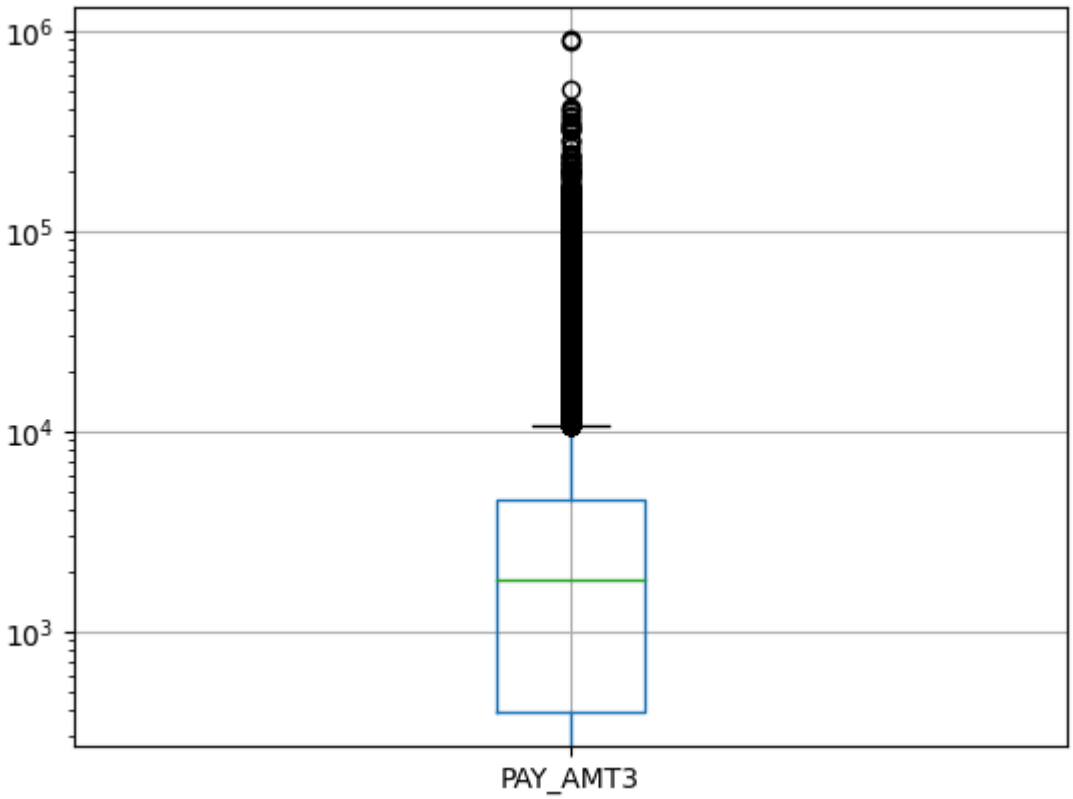

10^6
10^5
10^4
BILL_AMT4

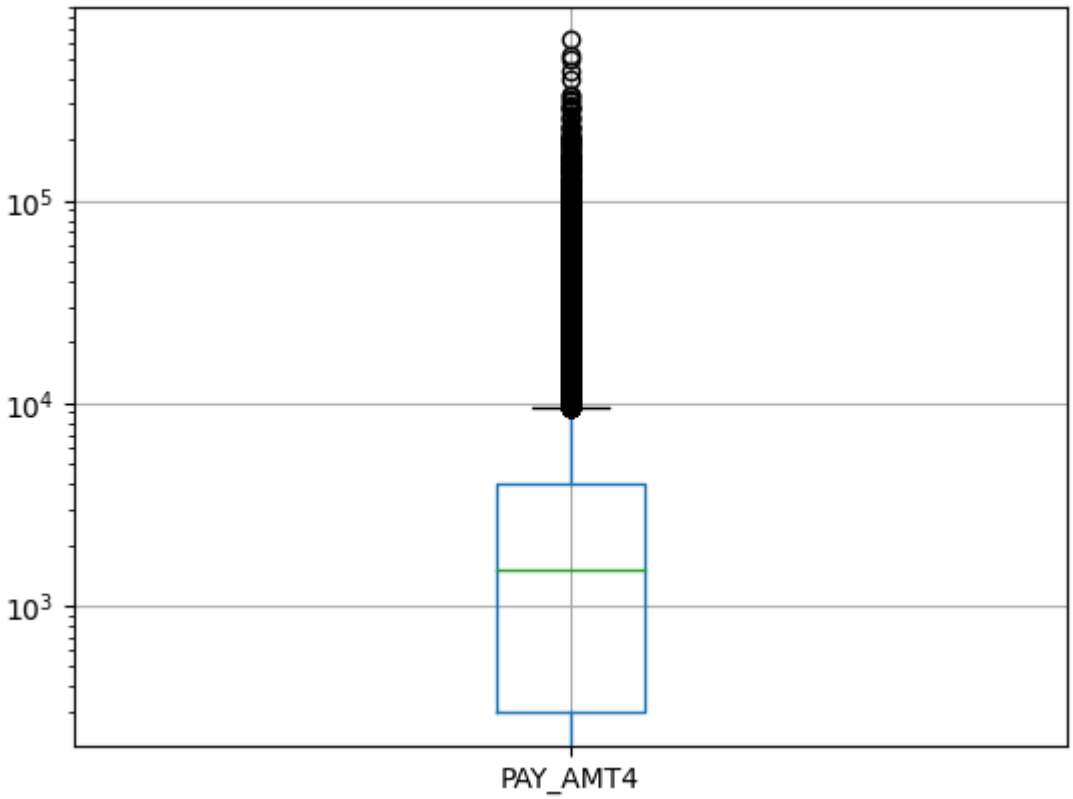

10^6
10^5
10^4
BILL_AMT5

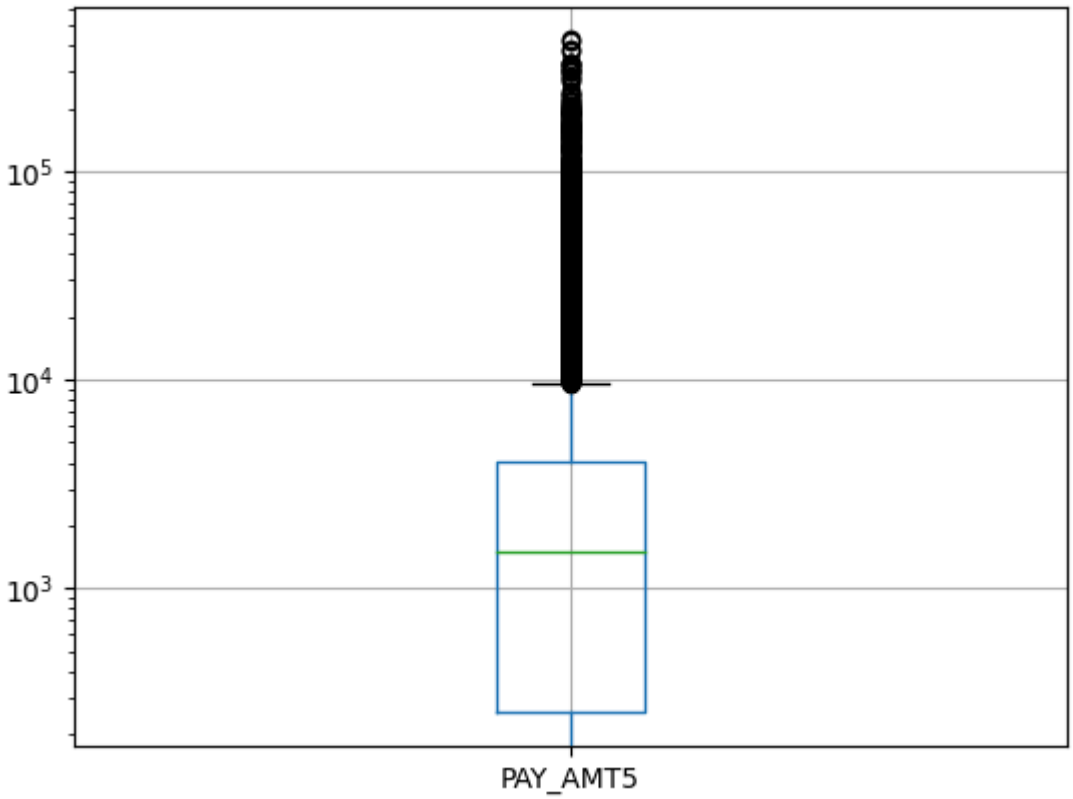

10^6
10^5
10^4
10^3
BILL_AMT6

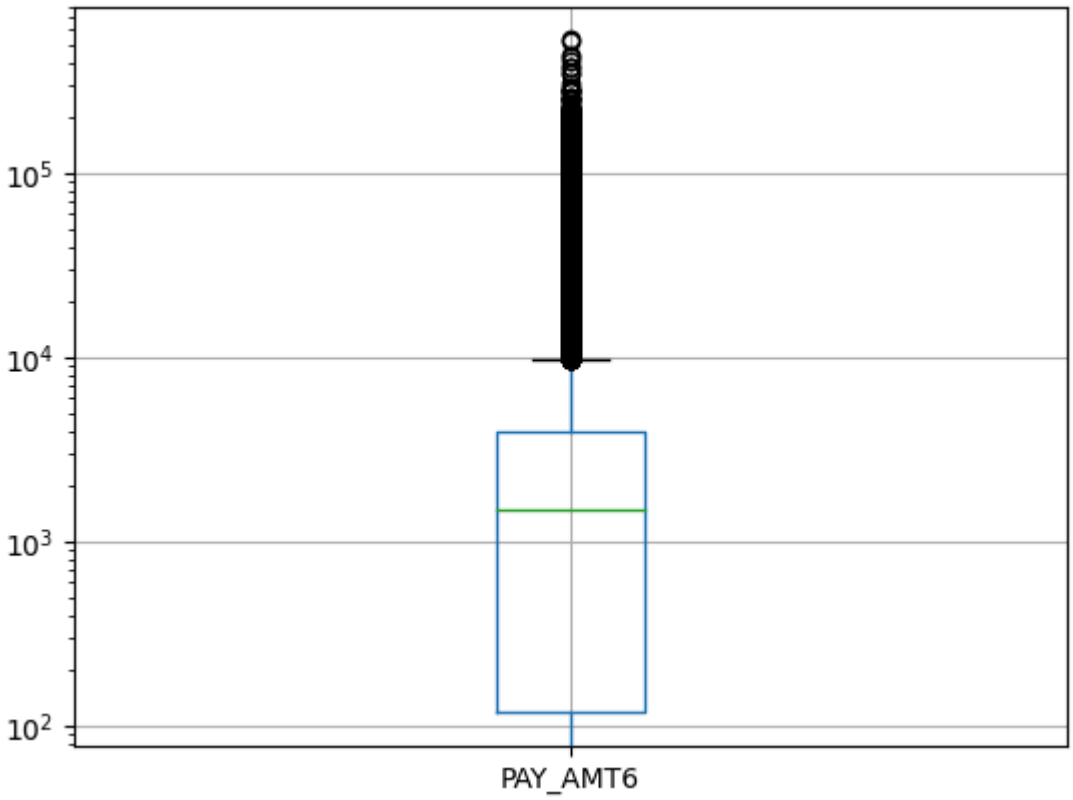

10^6
10^5
10^4
10^3
PAY_AMT1


10^6
10^5
10^4
10^3
PAY_AMT2


10^6
10^5
10^4
10^3
PAY_AMT3


10^5
10^4
10^3
PAY_AMT4



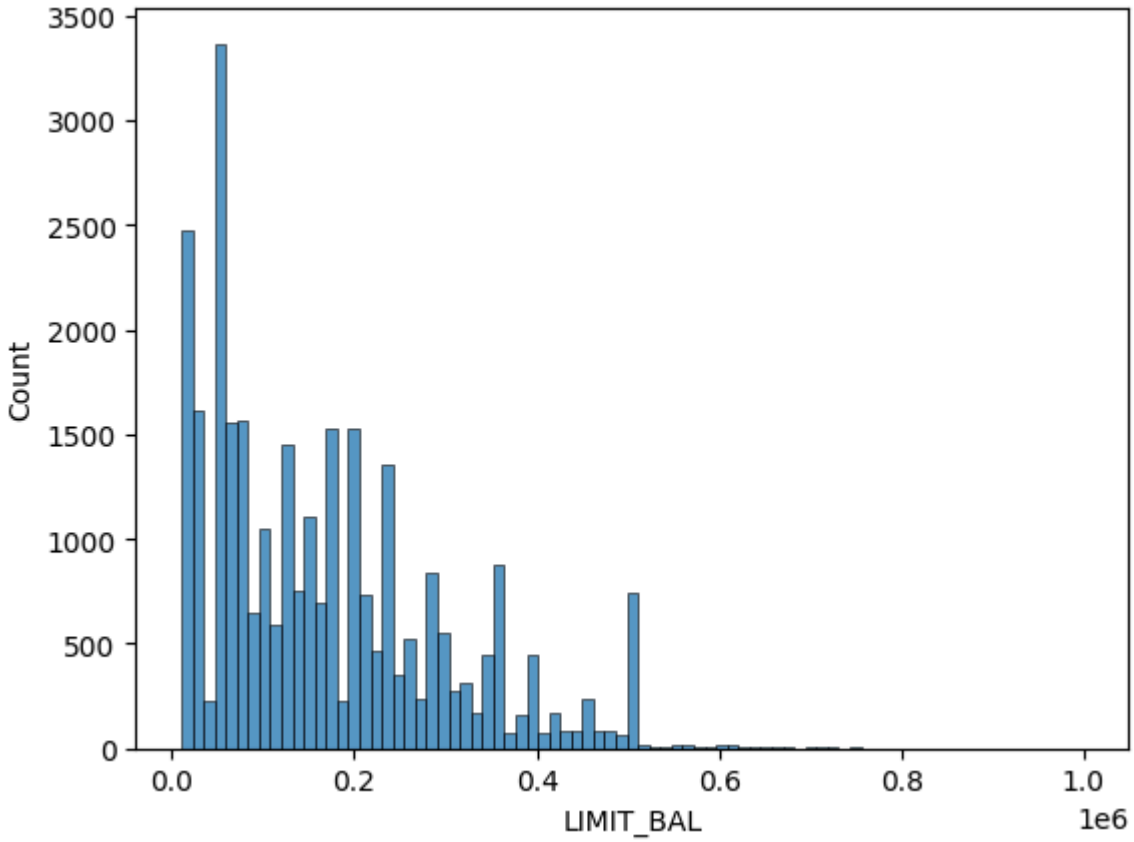


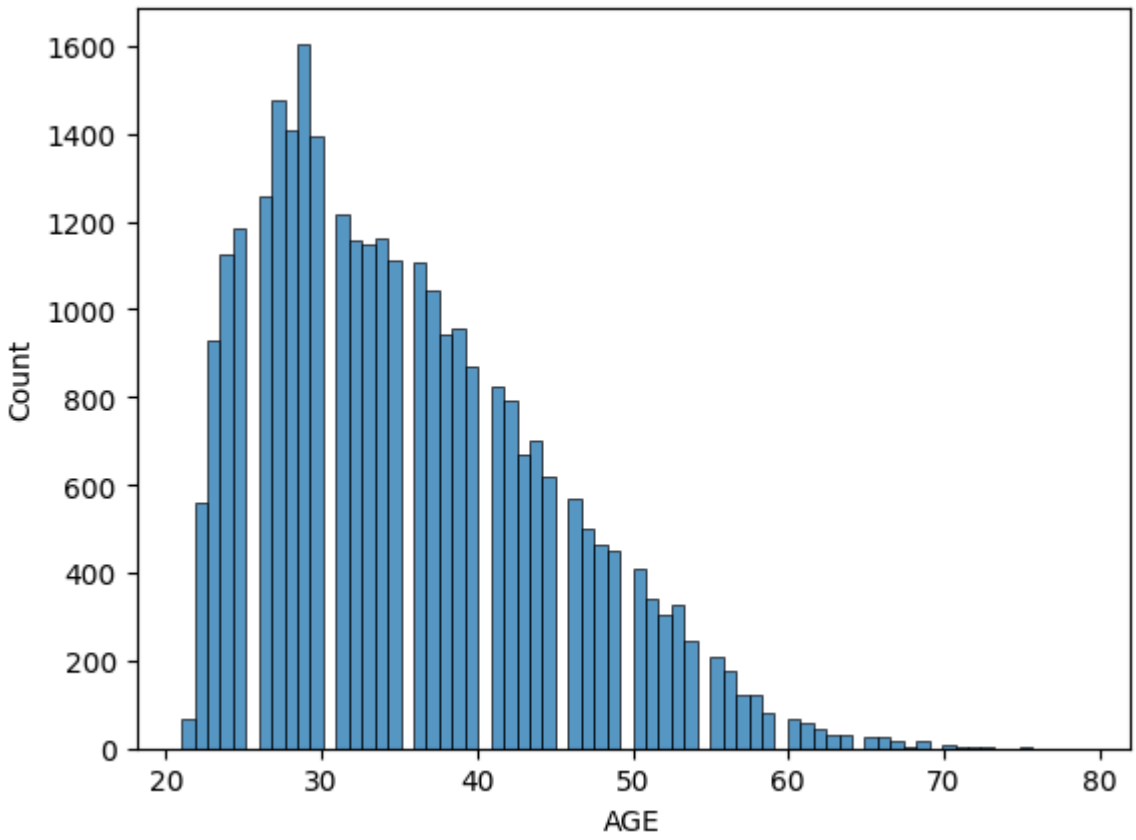
Histograms

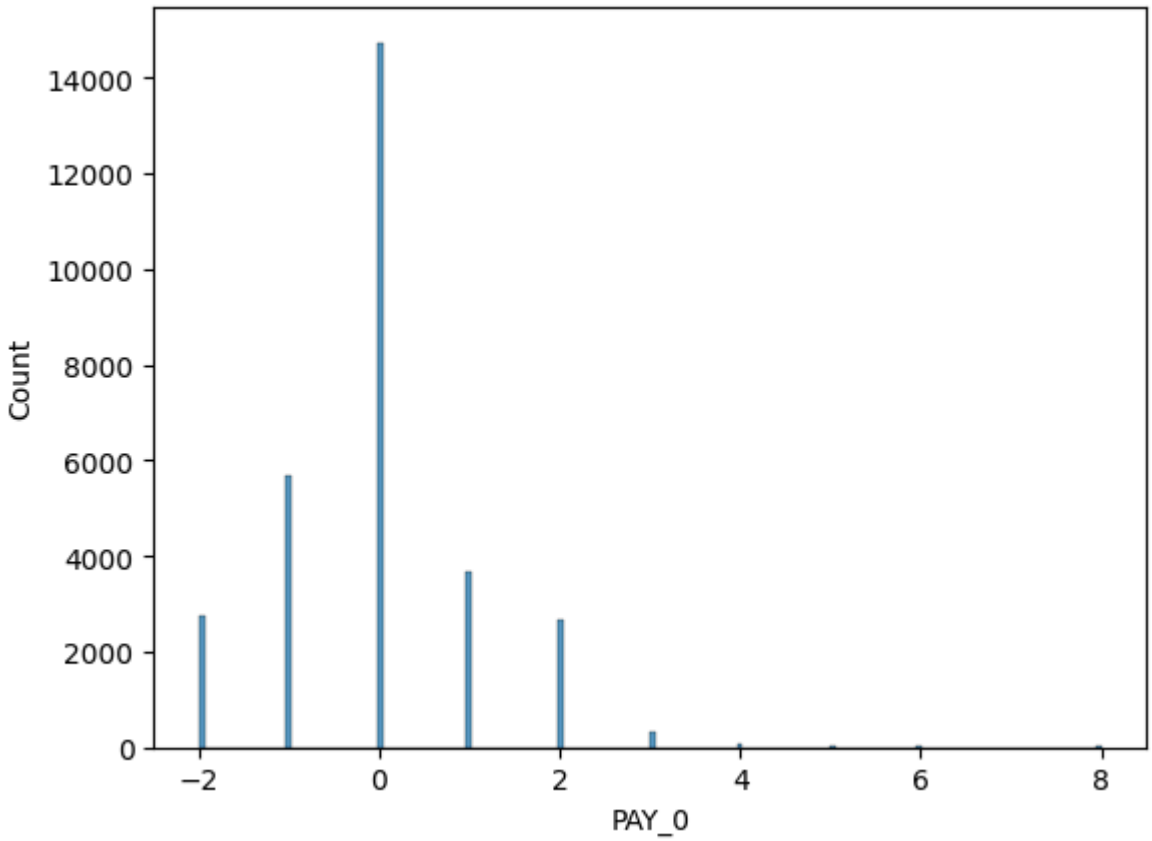
This graph will show the frequency distributions, it will be possible to identify if each feature is a Gaussian distribution or not

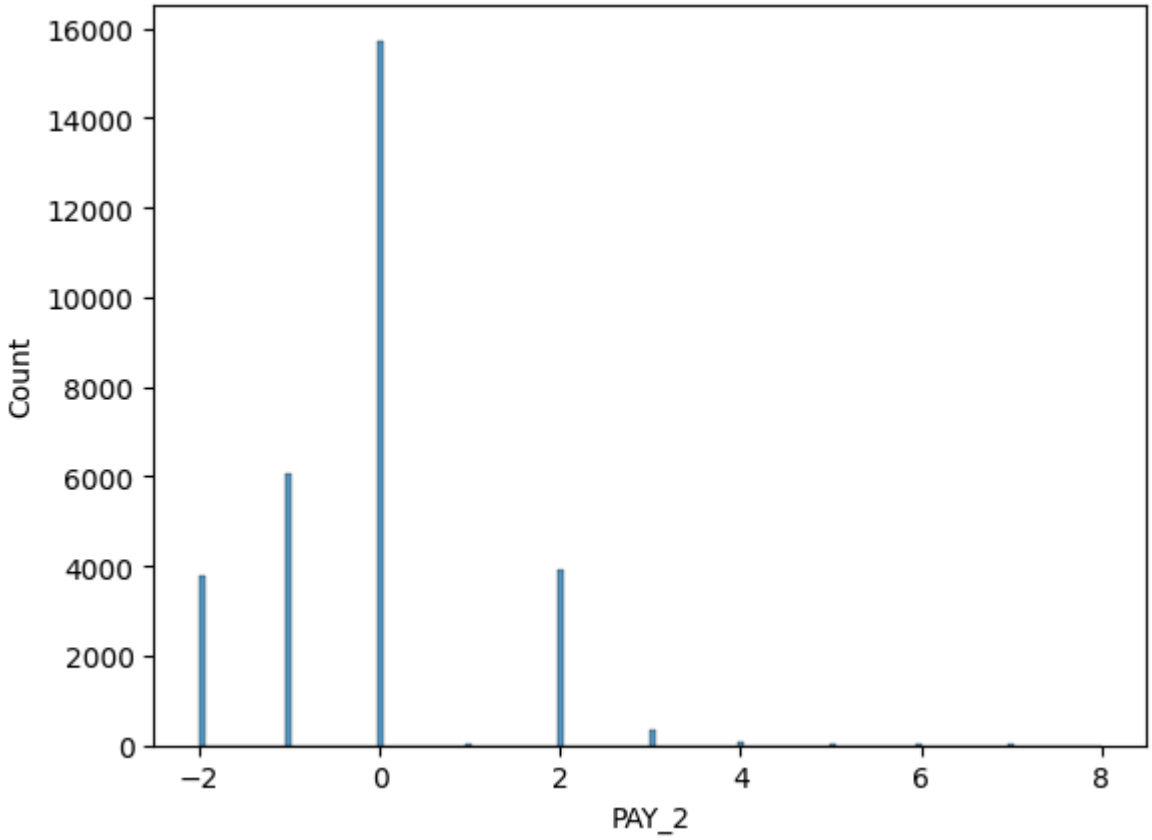
In [67]:
```
for c in df_no_cats.columns:
    ax = sns.histplot(df_no_cats, x=c)
```

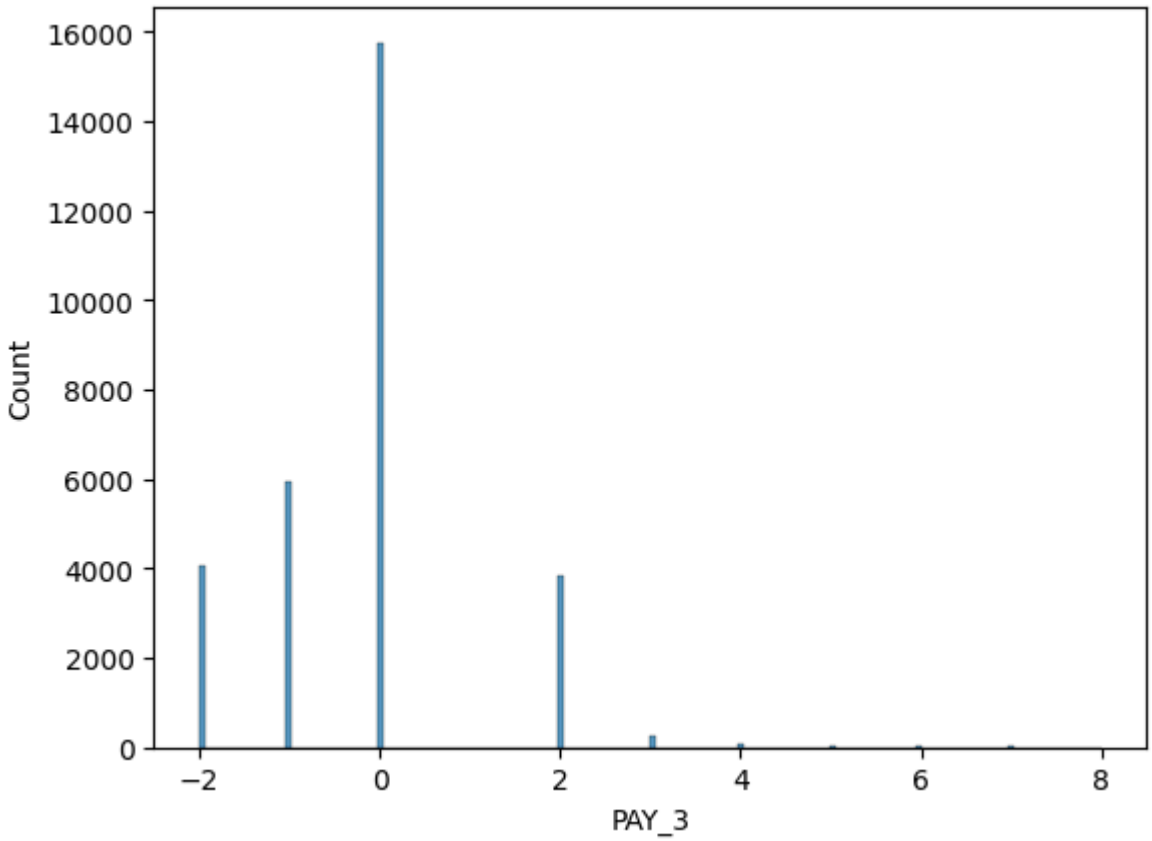
```
    if c.startswith('BILL_AMT') or c.startswith('PAY_AMT'):
        ax.set_yscale("log")
    plt.show()
    plt.close()
```

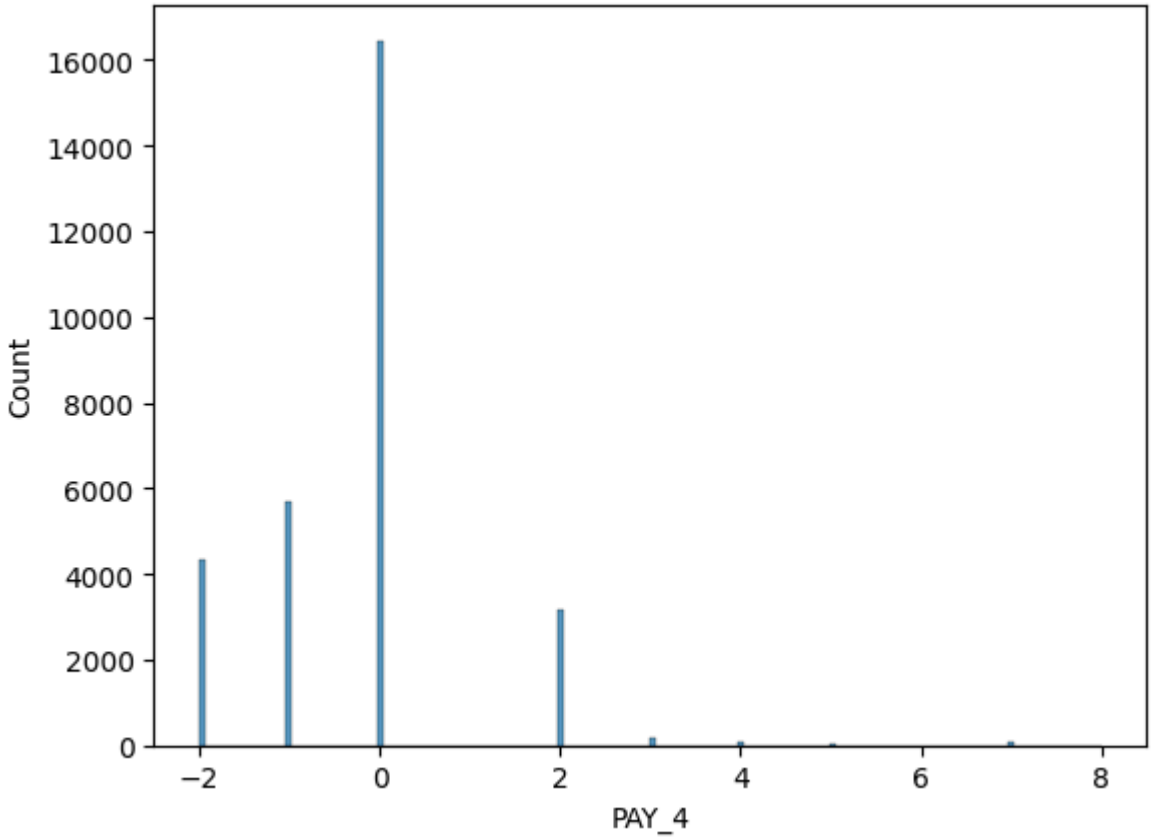


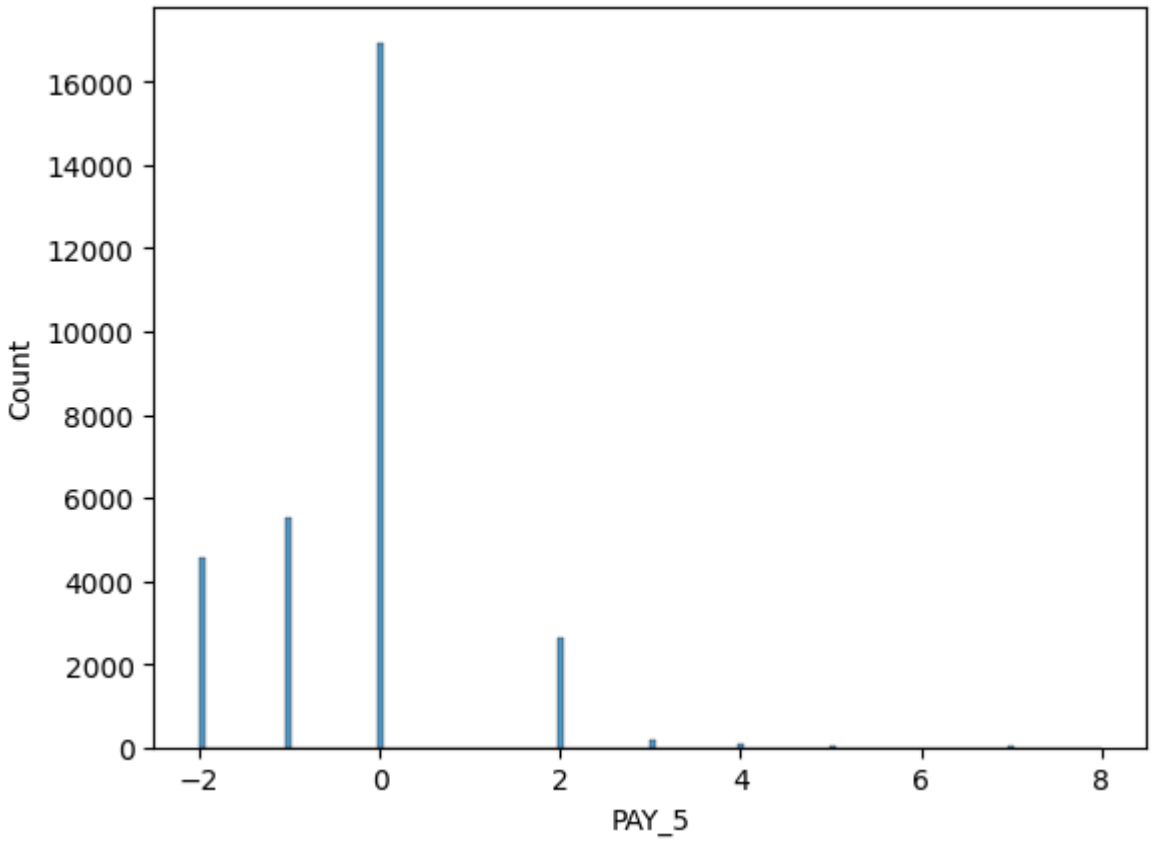


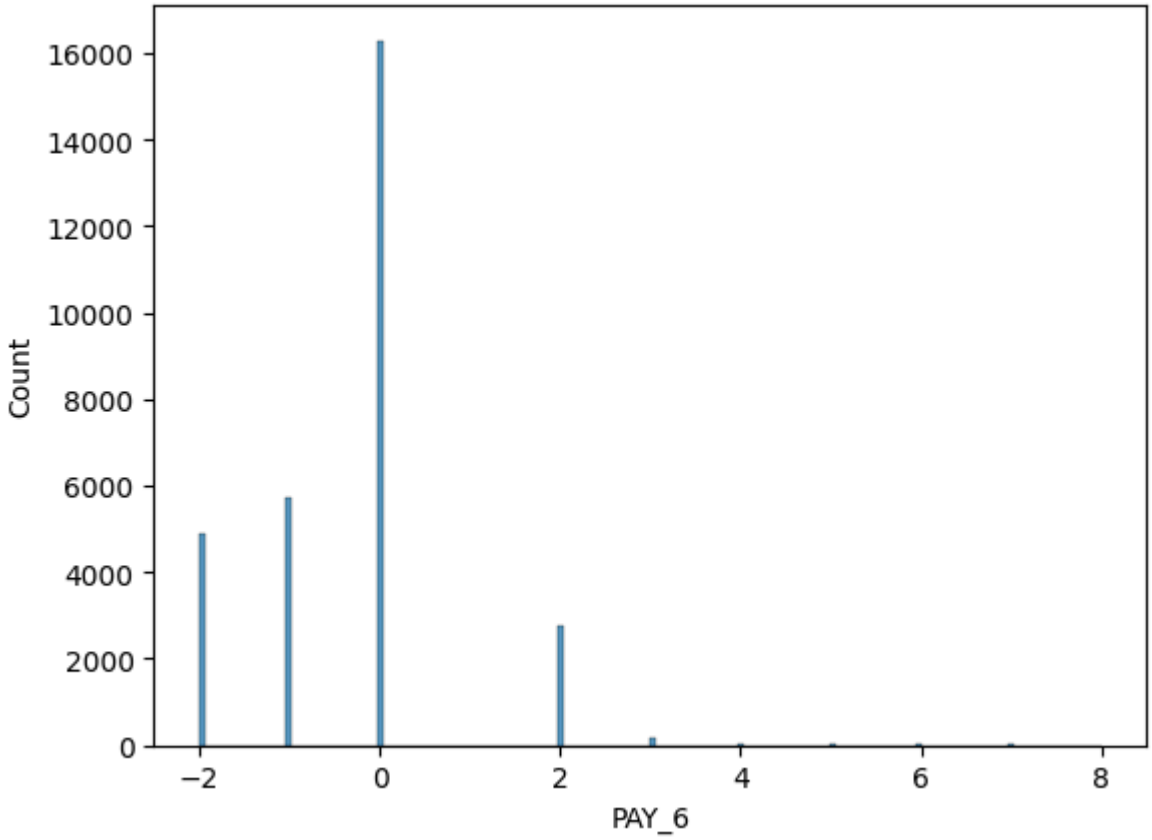

14000
12000
10000
8000
6000
4000
2000
0
Count
−2
0
2
4
6
8
PAY_0

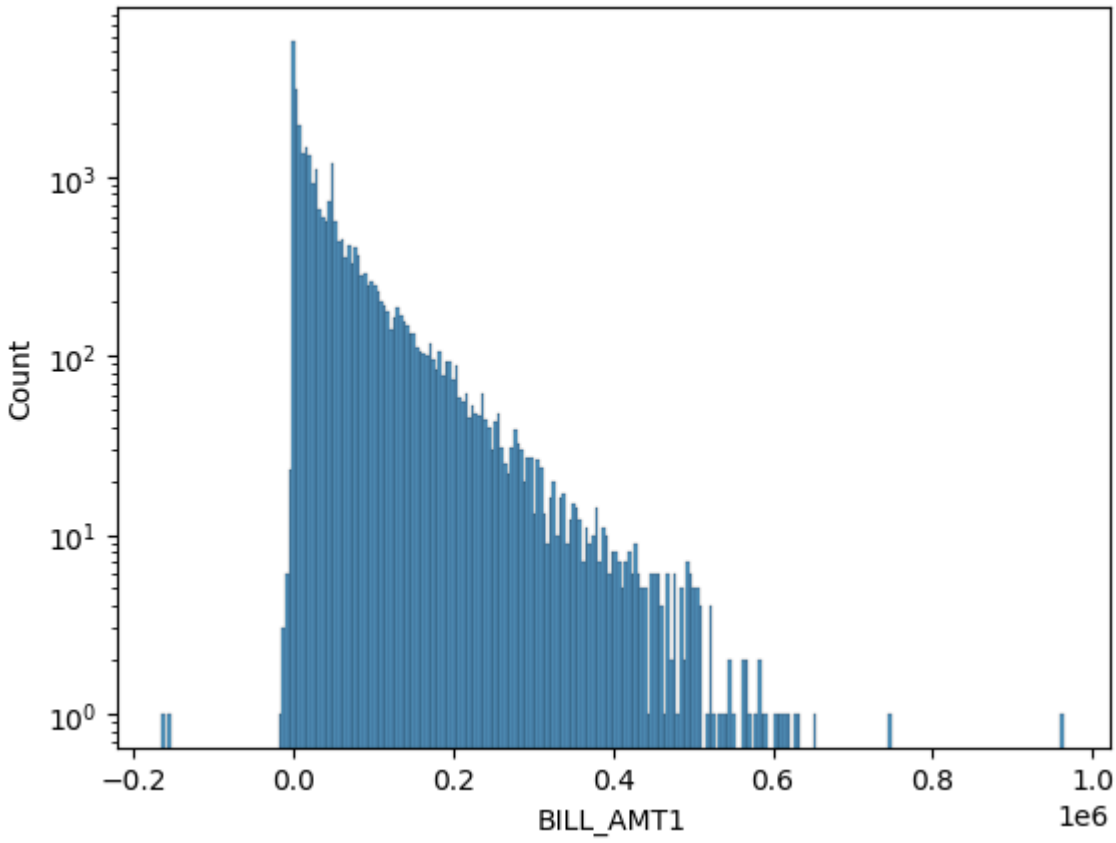

16000
14000
12000
10000
8000
6000
4000
2000
0
Count
−2
0
2
4
6
8
PAY_2

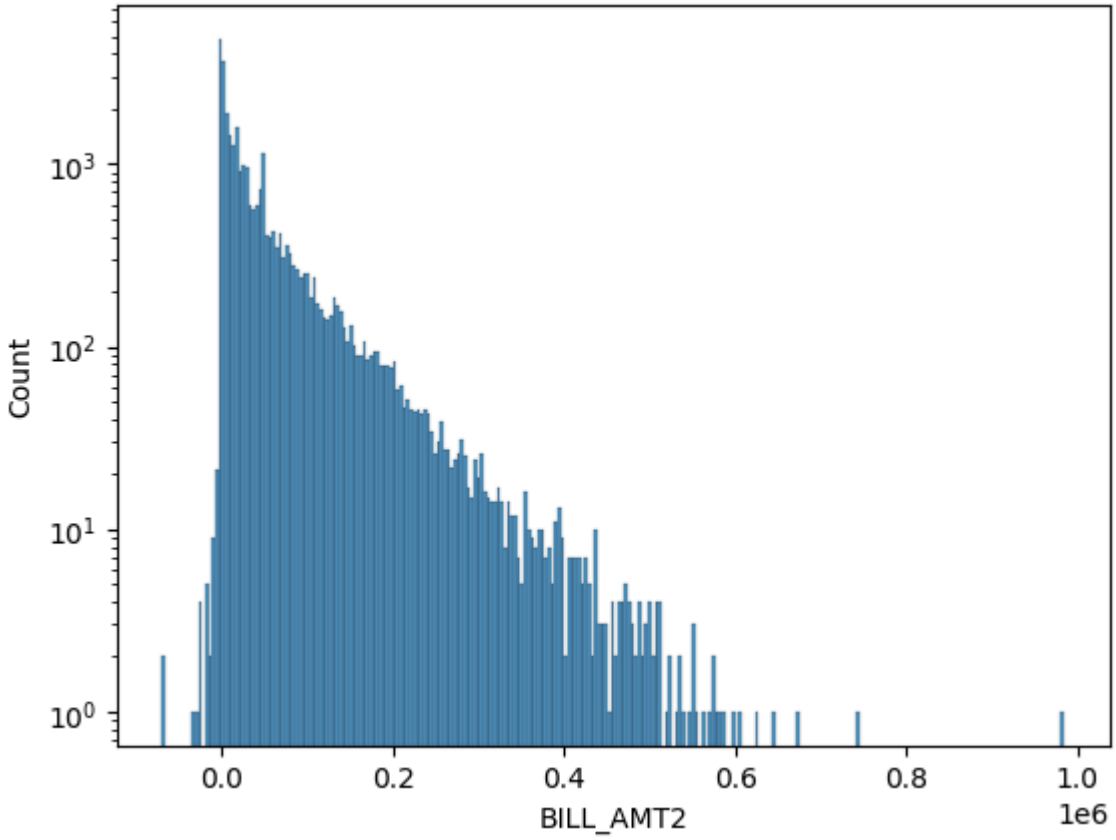

16000
14000
12000
10000
8000
6000
4000
2000
0
Count
−2
0
2
4
6
8
PAY_3

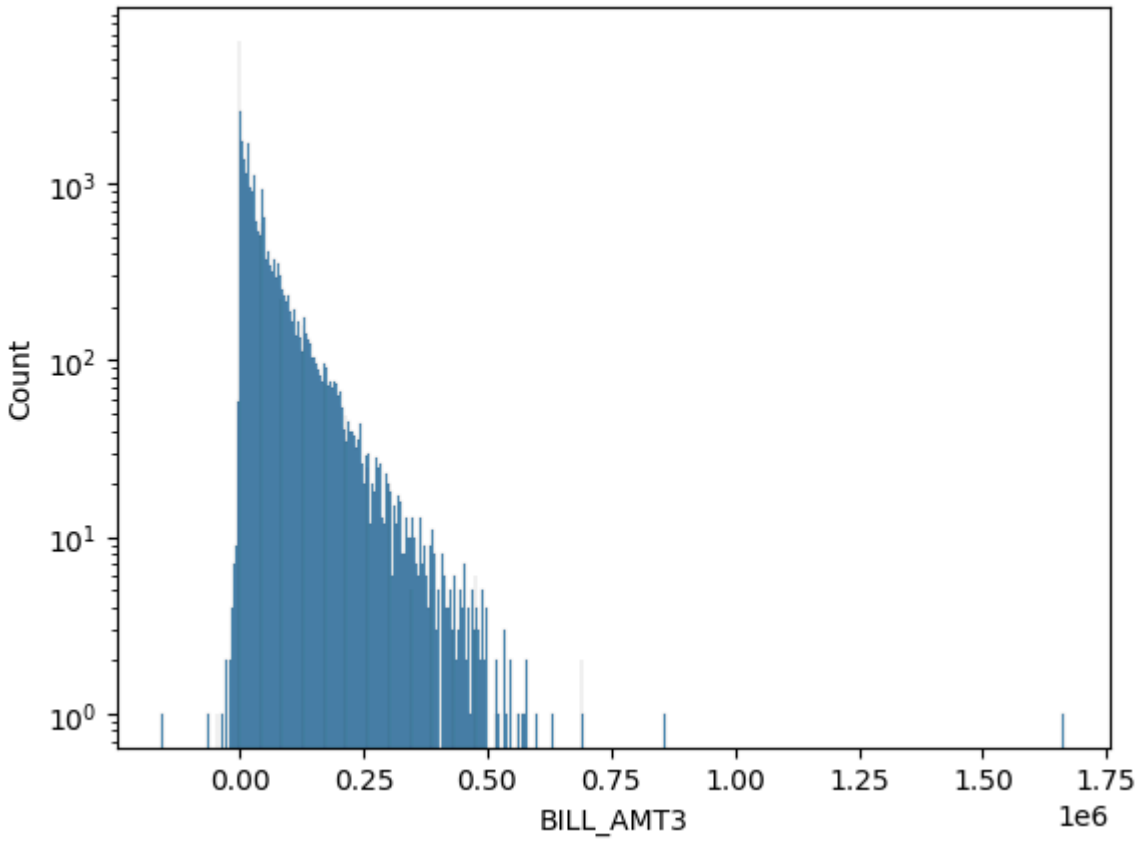

16000
14000
12000
10000
8000
6000
4000
2000
0
Count
−2
0
2
4
6
8
PAY_4

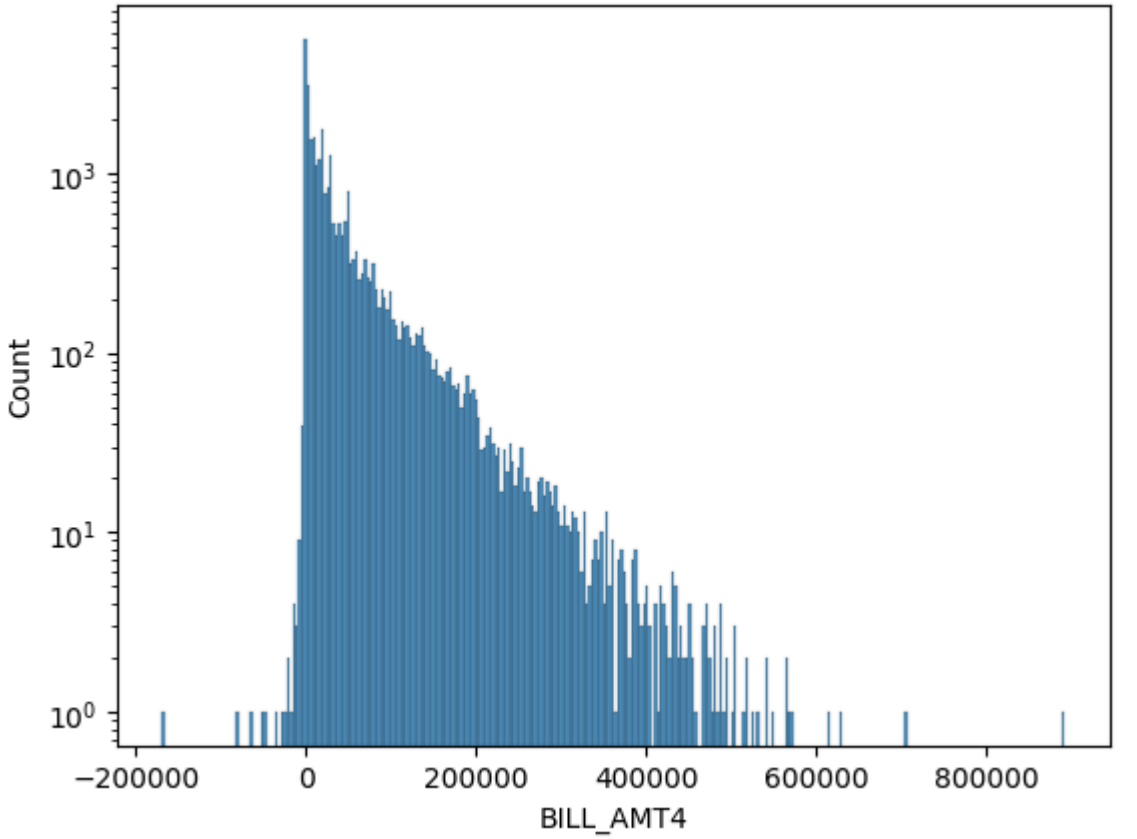

16000
14000
12000
10000
8000
6000
4000
2000
0
Count
−2
0
2
4
6
8
PAY_5

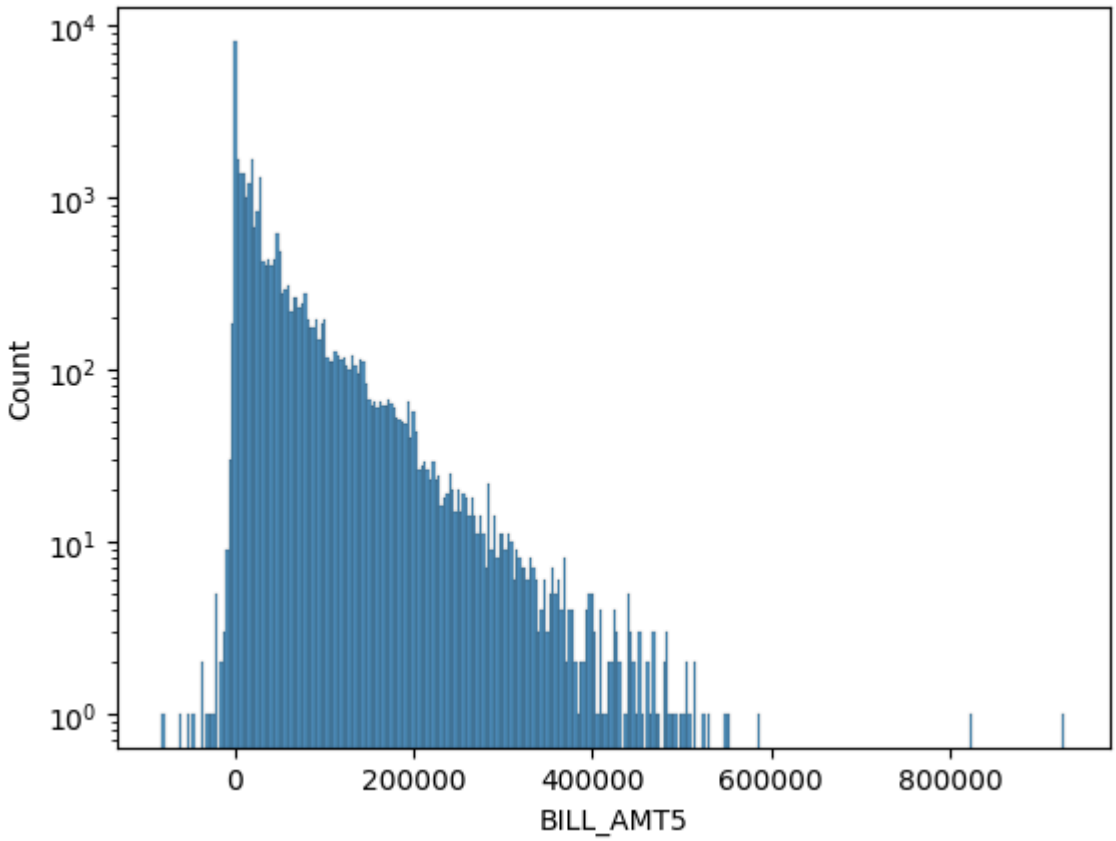

16000
14000
12000
10000
8000
6000
4000
2000
0
Count
−2
0
2
4
6
8
PAY_6

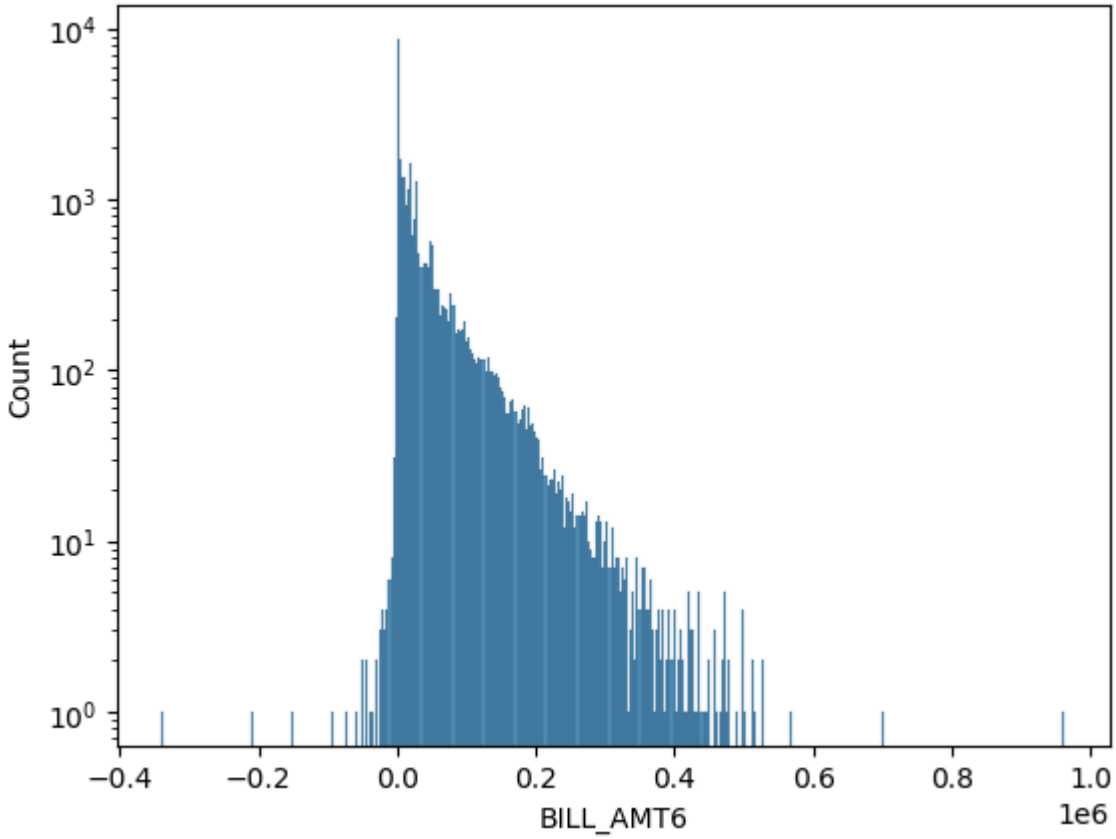

Count
10^3
10^2
10^1
10^0
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
1e6
BILL_AMT1

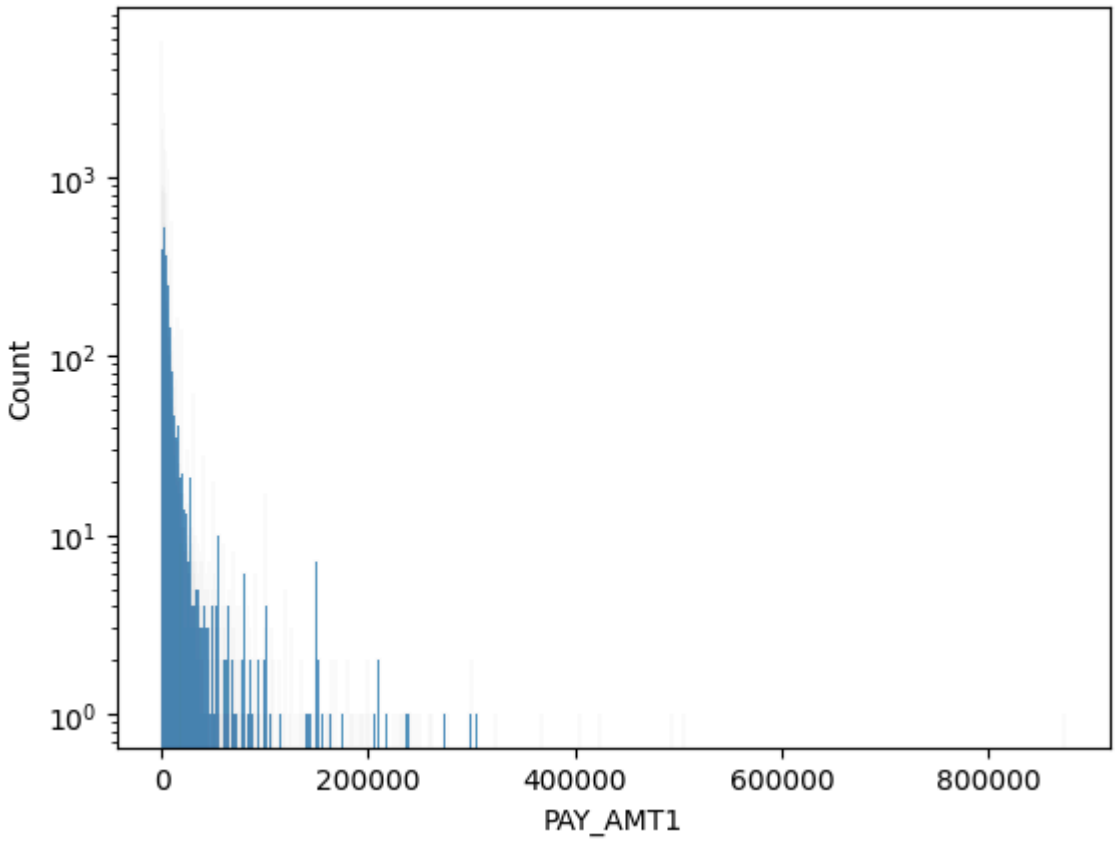

Count
10^3
10^2
10^1
10^0
0.0
0.2
0.4
0.6
0.8
1.0
1e6
BILL_AMT2

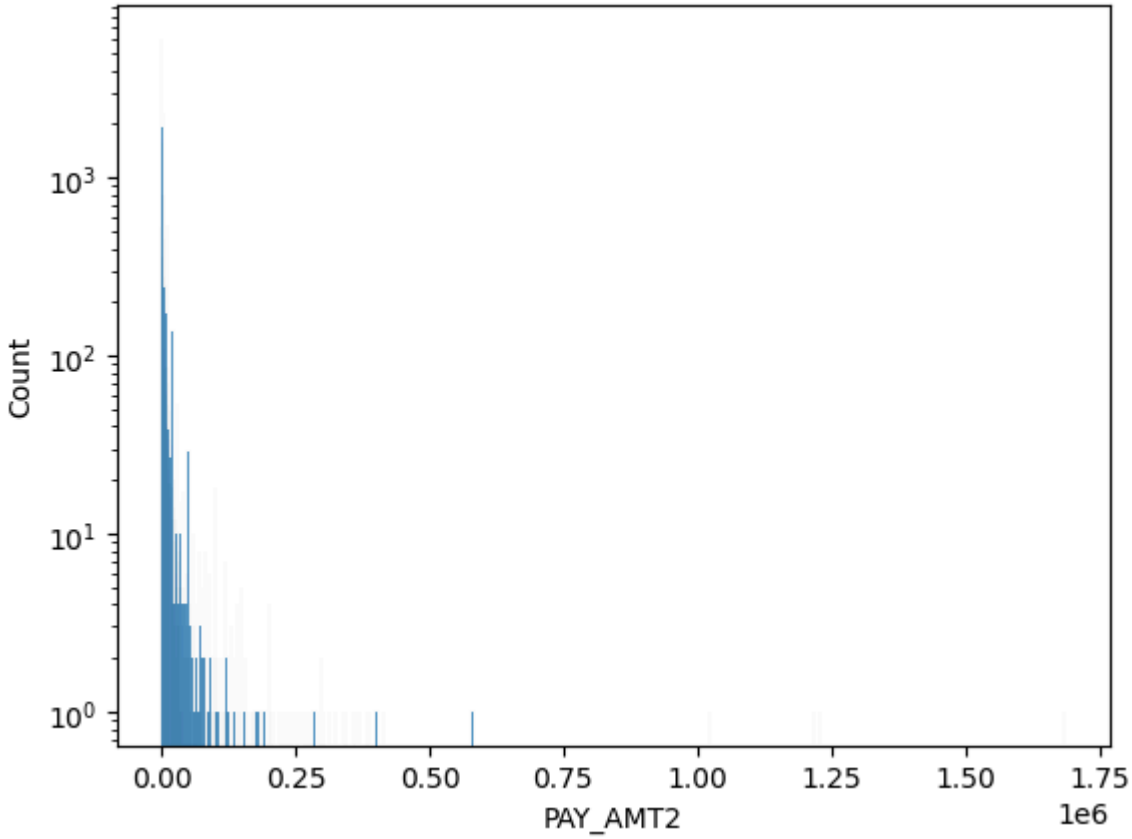

Count
10^3
10^2
10^1
10^0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
1e6
BILL_AMT3

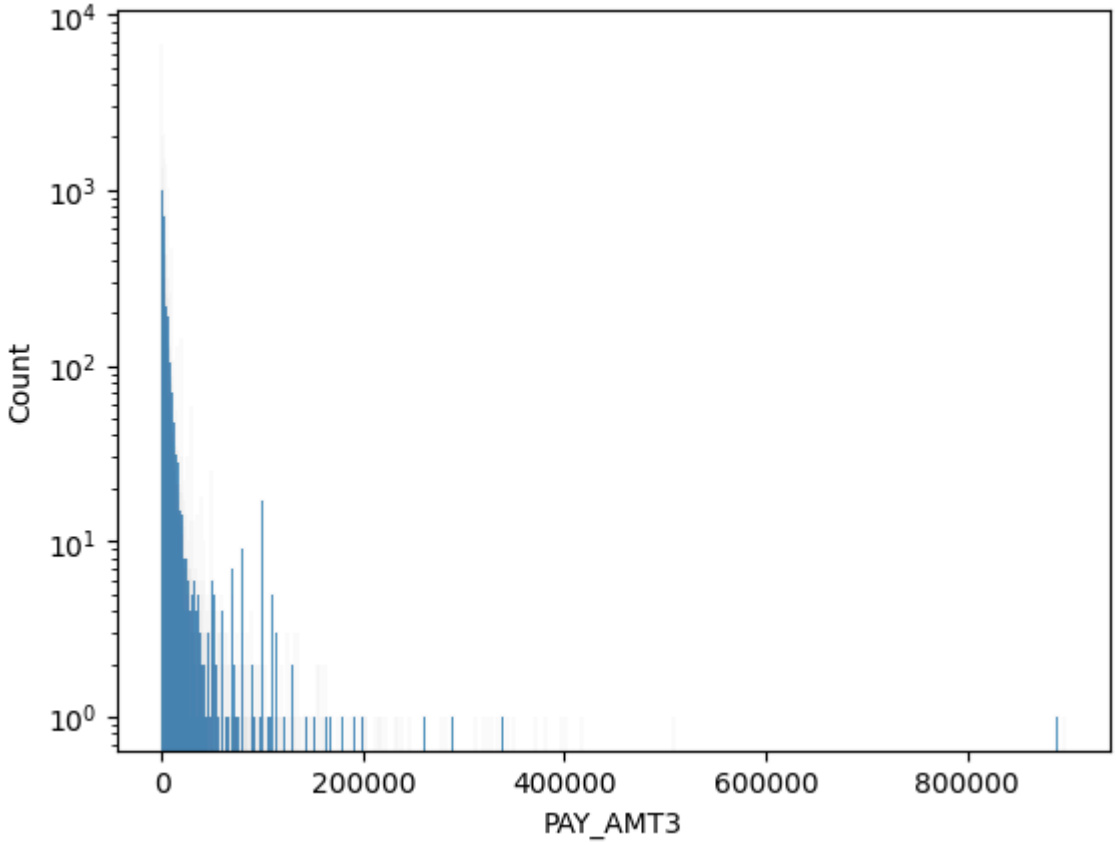

Count
10^3
10^2
10^1
10^0
−200000
0
200000
400000
600000
800000
BILL_AMT4

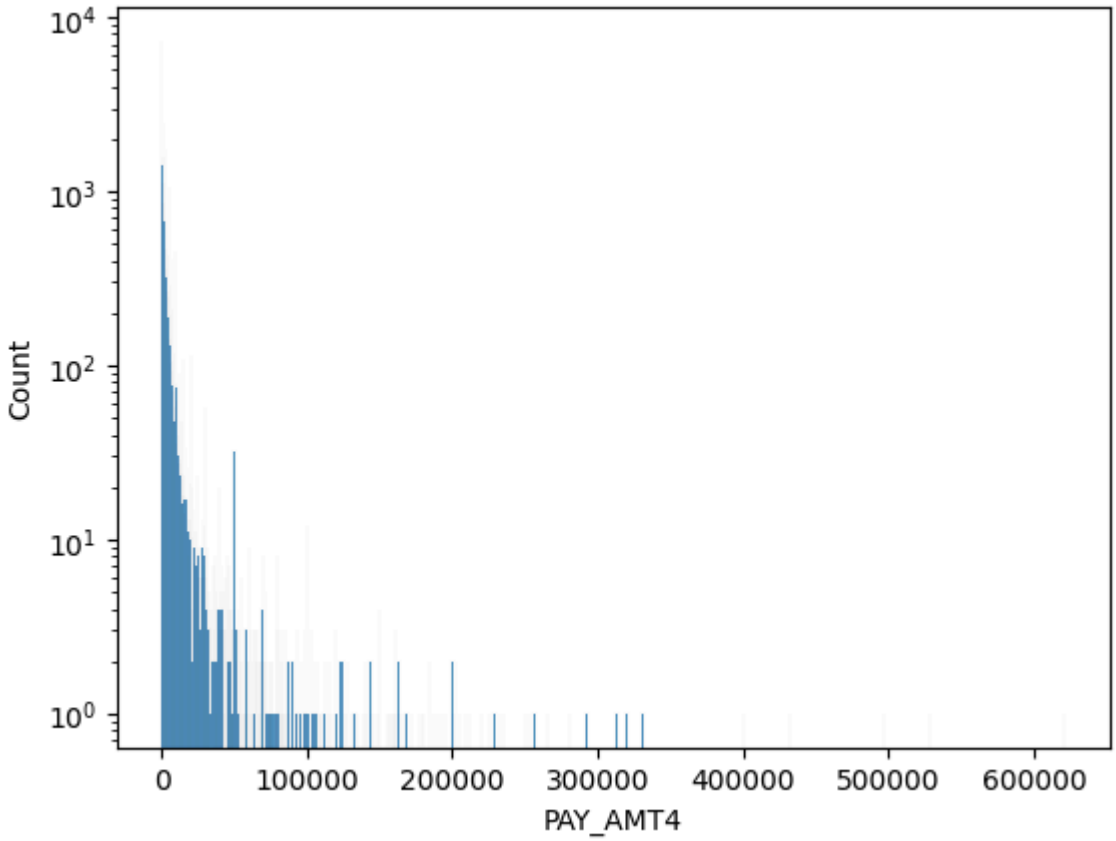

Count
BILL_AMT5

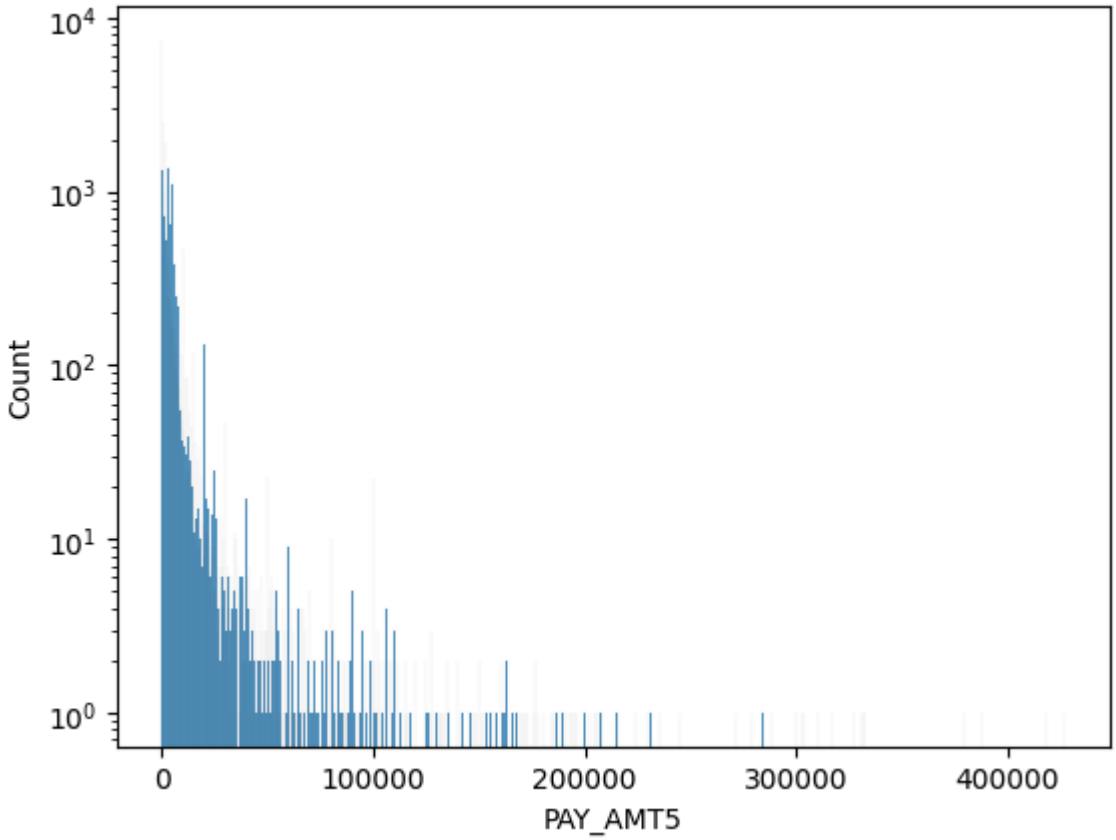

Count
BILL_AMT6
1e6

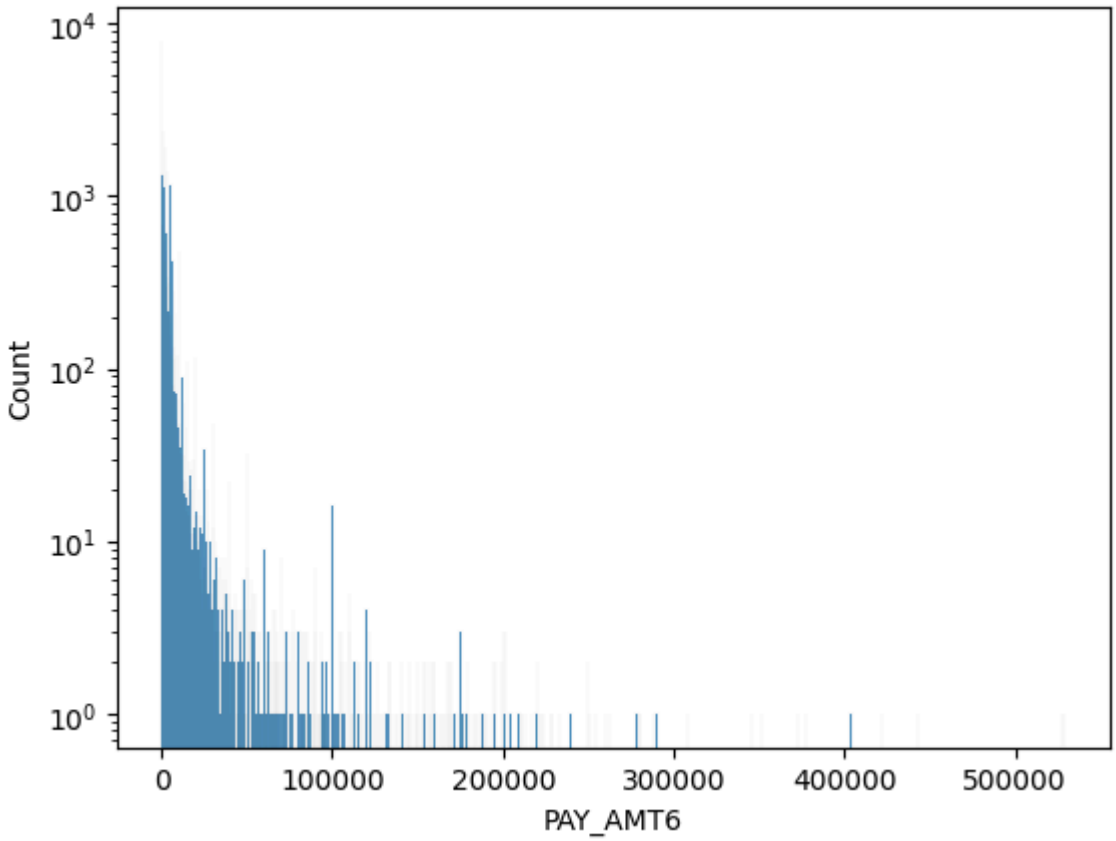

Count
10^3
10^2
10^1
10^0
0
200000
400000
600000
800000
PAY_AMT1


Count
10^3
10^2
10^1
10^0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
1e6
PAY_AMT2


Count
PAY_AMT3
0
200000
400000
600000
800000


Count
PAY_AMT4
0
100000
200000
300000
400000
500000
600000





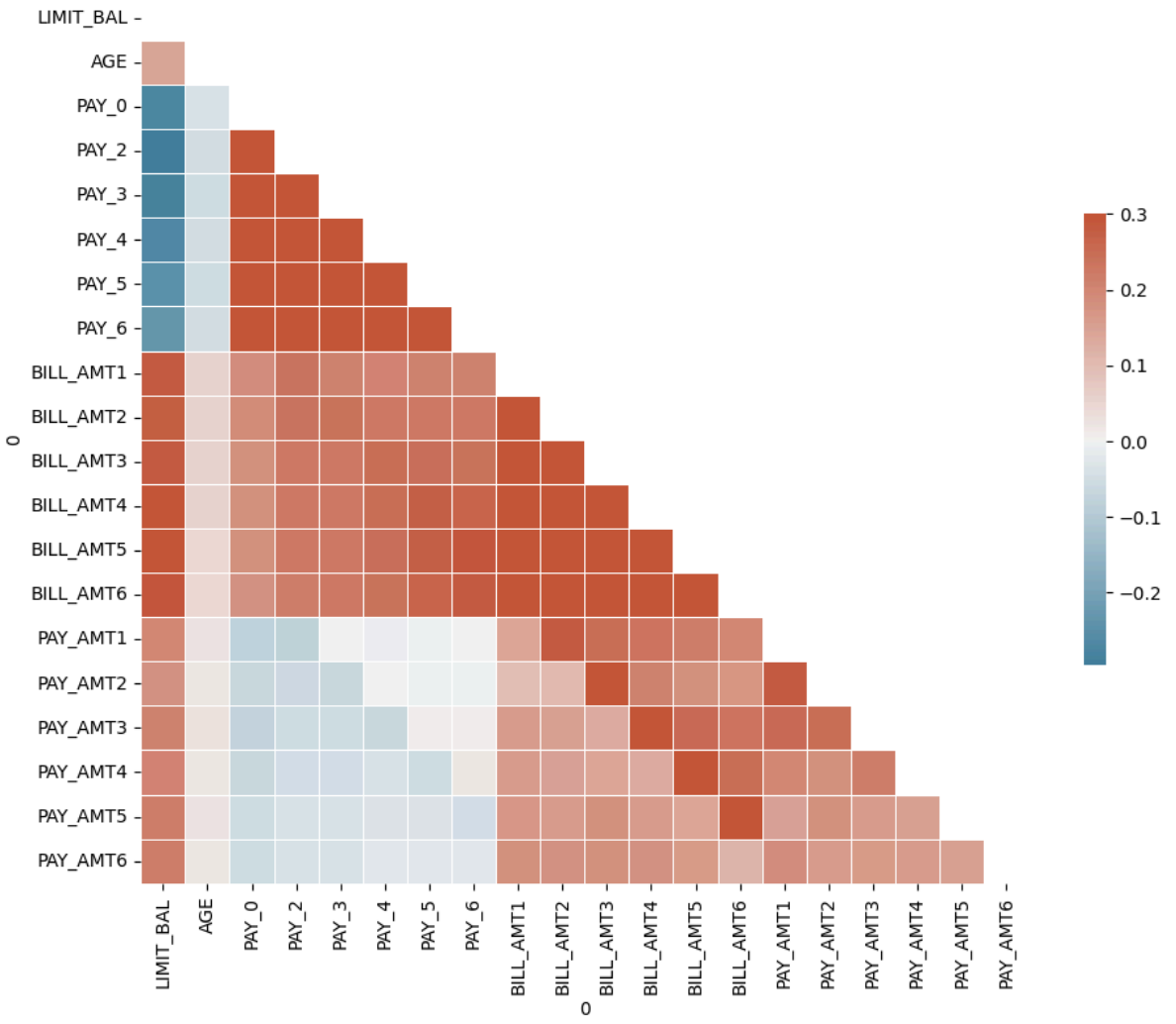
Plotting a diagonal correlation matrix

This diagram will show the correlation of each feature individually

In [32]:
```
corr = df_no_cats.corr()
```

```
mask = np.triu(np.ones_like(corr, dtype=bool))
f, ax = plt.subplots(figsize=(11, 9))
cmap = sns.diverging_palette(230, 20, as_cmap=True)

sns.heatmap(corr, mask=mask, cmap=cmap, vmax=.3, center=0,
            square=True, linewidths=.5, cbar_kws={"shrink": .5})
plt.show()
```



In [ ]:

Dummies Variables and Categorical Data

Dummy variables enable us to use a single regression equation to represent multiple groups

In [38]:
```
pd.get_dummies(df_cats.drop(['default payment next month'], axis=1)).head()
```

Out[38]:

	SEX_1.0	SEX_2.0	MARRIAGE_0.0	MARRIAGE_1.0	MARRIAGE_2.0	MARRIAGE_3.0	EDUCATION_(
0	0	1	0	1	0	0	
1	0	1	0	0	1	0	
2	0	1	0	0	1	0	
3	0	1	0	1	0	0	
4	1	0	0	1	0	0	

In [42]:
```
df2 = pd.concat([df_no_cats, pd.get_dummies(df_cats.drop(['default payment next mor
df2.head()
```

Out[42]:

	LIMIT_BAL	AGE	PAY_0	PAY_2	PAY_3	PAY_4	PAY_5	PAY_6	BILL_AMT1	BILL_AMT2	BILL_AM
0	20000.0	24.0	2.0	2.0	-1.0	-1.0	-2.0	-2.0	3913.0	3102.0	68
1	120000.0	26.0	-1.0	2.0	0.0	0.0	0.0	2.0	2682.0	1725.0	268
2	90000.0	34.0	0.0	0.0	0.0	0.0	0.0	0.0	29239.0	14027.0	1355
3	50000.0	37.0	0.0	0.0	0.0	0.0	0.0	0.0	46990.0	48233.0	4929
4	50000.0	57.0	-1.0	0.0	-1.0	0.0	0.0	0.0	8617.0	5670.0	3583

5 rows × 33 columns

In [43]:
```
df2.describe()
```

Out[43]:

	LIMIT_BAL	AGE	PAY_0	PAY_2	PAY_3	PAY_4	
count	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	300
mean	167484.322667	35.485500	-0.016700	-0.133767	-0.166200	-0.220667	
std	129747.661567	9.217904	1.123802	1.197186	1.196868	1.169139	
min	10000.000000	21.000000	-2.000000	-2.000000	-2.000000	-2.000000	
25%	50000.000000	28.000000	-1.000000	-1.000000	-1.000000	-1.000000	
50%	140000.000000	34.000000	0.000000	0.000000	0.000000	0.000000	
75%	240000.000000	41.000000	0.000000	0.000000	0.000000	0.000000	
max	1000000.000000	79.000000	8.000000	8.000000	8.000000	8.000000	

8 rows × 33 columns

In [ ]:
```

```