Big_Data_Analytics_Project_Outliers_Mar16

March 16, 2025

[39]:
```
import pandas as pd

url = "https://archive.ics.uci.edu/static/public/350/data.csv"
data = pd.read_csv(url, sep= ',')
print(data.head())
```

```
   ID      X1  X2  X3  X4  X5  X6  X7  X8  X9  …    X15    X16    X17   X18  \
0   1   20000   2   2   1  24   2   2  -1  -1  …      0      0      0     0
1   2  120000   2   2   2  26  -1   2   0   0  …   3272   3455   3261     0
2   3   90000   2   2   2  34   0   0   0   0  …  14331  14948  15549  1518
3   4   50000   2   2   1  37   0   0   0   0  …  28314  28959  29547  2000
4   5   50000   1   2   1  57  -1   0  -1   0  …  20940  19146  19131  2000

     X19    X20   X21   X22   X23  Y
0    689      0     0     0     0  1
1   1000   1000  1000     0  2000  1
2   1500   1000  1000  1000  5000  0
3   2019   1200  1100  1069  1000  0
4  36681  10000  9000   689   679  0

[5 rows x 25 columns]
```

[40]:
```
data.rename(columns={'X1': 'LIMIT_BAL', 'X2': 'SEX', 'X3': 'EDUCATION', 'X4':␣
 ↪'MARRIAGE', 'X5': 'AGE', 'X6': 'PAY_0', 'X7': 'PAY_2','X8': 'PAY_3', 'X9':␣
 ↪'PAY_4', 'X10': 'PAY_5', 'X11': 'PAY_6', 'X12': 'BILL_AMT1', 'X13':␣
 ↪'BILL_AMT2', 'X14': 'BILL_AMT3', 'X15': 'BILL_AMT4', 'X16': 'BILL_AMT5',␣
 ↪'X17': 'BILL_AMT6', 'X18': 'PAY_AMT1', 'X19': 'PAY_AMT2', 'X20': 'PAY_AMT3',␣
 ↪'X21': 'PAY_AMT4', 'X22': 'PAY_AMT5', 'X23': 'PAY_AMT6'}, inplace=True)
```

[41]:
```
data.head()
```

[41]:
```
   ID  LIMIT_BAL  SEX  EDUCATION  MARRIAGE  AGE  PAY_0  PAY_2  PAY_3  PAY_4  \
0   1      20000    2          2         1   24      2      2     -1     -1
1   2     120000    2          2         2   26     -1      2      0      0
2   3      90000    2          2         2   34      0      0      0      0
3   4      50000    2          2         1   37      0      0      0      0
4   5      50000    1          2         1   57     -1      0     -1      0
```

```
   …  BILL_AMT4  BILL_AMT5  BILL_AMT6  PAY_AMT1  PAY_AMT2  PAY_AMT3  \
0  …          0          0          0         0       689         0
1  …       3272       3455       3261         0      1000      1000
2  …      14331      14948      15549      1518      1500      1000
3  …      28314      28959      29547      2000      2019      1200
4  …      20940      19146      19131      2000     36681     10000

   PAY_AMT4  PAY_AMT5  PAY_AMT6  Y
0         0         0         0  1
1      1000         0      2000  1
2      1000      1000      5000  0
3      1100      1069      1000  0
4      9000       689       679  0

[5 rows x 25 columns]
```

[42]:
```
# Replacing education values = 0, 5 and 6 with 4, since 0, 5 and 6 are not␣
 ↪defined

fill = (data.EDUCATION == 0) | (data.EDUCATION == 5) | (data.EDUCATION == 6)
data.loc[fill, 'EDUCATION'] = 4

print('EDUCATION ' + str(sorted(data['EDUCATION'].unique())))
```

```
EDUCATION [1, 2, 3, 4]
```

[43]:
```
# Replacing marital status value = 0 to 3, since 0 is not defined

fill = (data.MARRIAGE == 0)
data.loc[fill, 'MARRIAGE'] = 3

print('MARRIAGE ' + str(sorted(data['MARRIAGE'].unique())))
```

```
MARRIAGE [1, 2, 3]
```

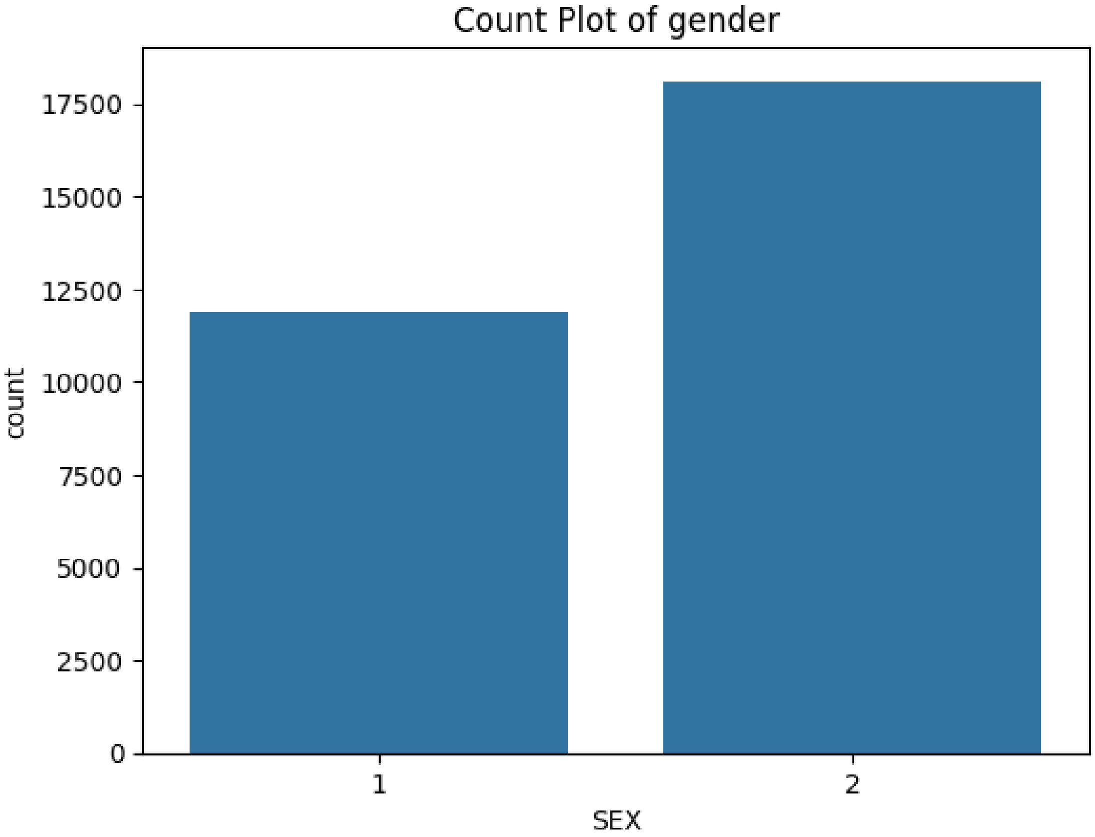
[44]:
```
categorical_variables = ['SEX', 'EDUCATION', 'MARRIAGE', 'PAY_0', 'PAY_2',␣
 ↪'PAY_3', 'PAY_4', 'PAY_5', 'PAY_6']
```

[45]:
```
data['SEX'].value_counts(normalize=True) * 100
```

[45]:
```
SEX
2    60.373333
1    39.626667
Name: proportion, dtype: float64
```

[46]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

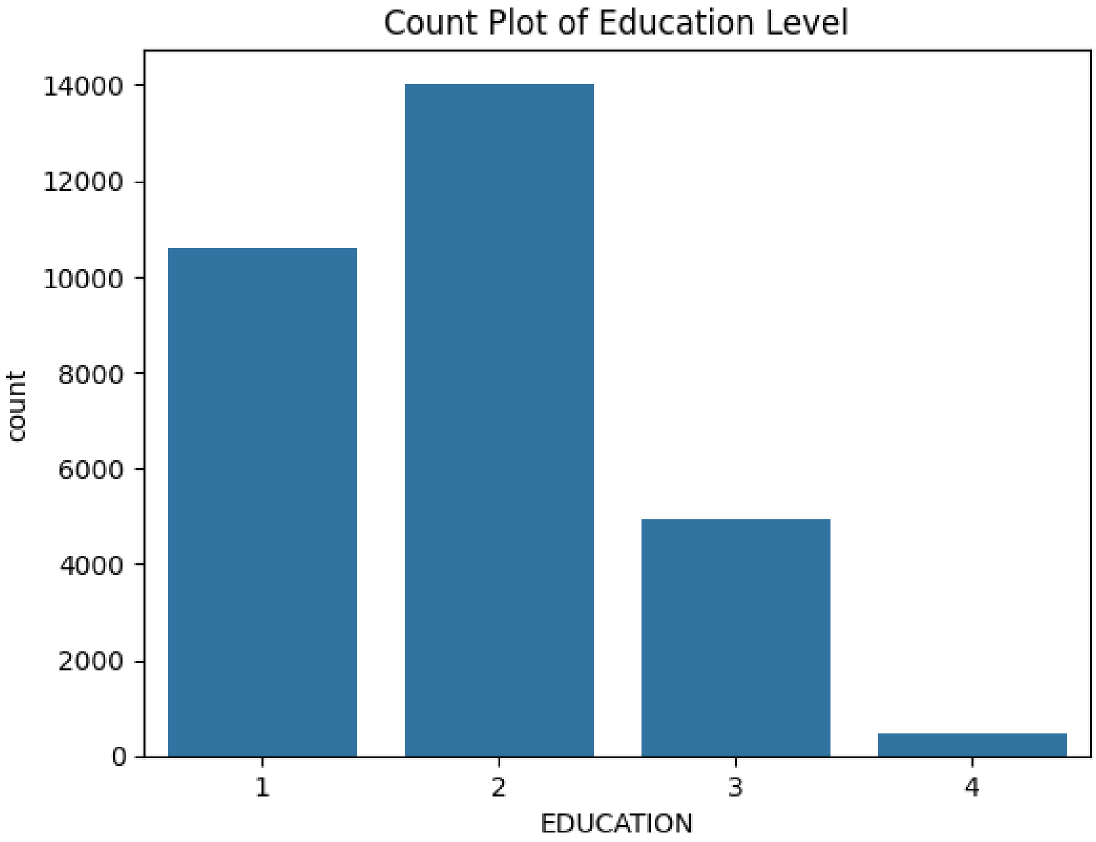
```
# Count plot for Sex
sns.countplot(x='SEX', data=data)
plt.title('Count Plot of gender')
plt.show()
```



```
[47]: data['EDUCATION'].value_counts(normalize=True) * 100
```

```
[47]: EDUCATION
      2    46.766667
      1    35.283333
      3    16.390000
      4     1.560000
      Name: proportion, dtype: float64
```

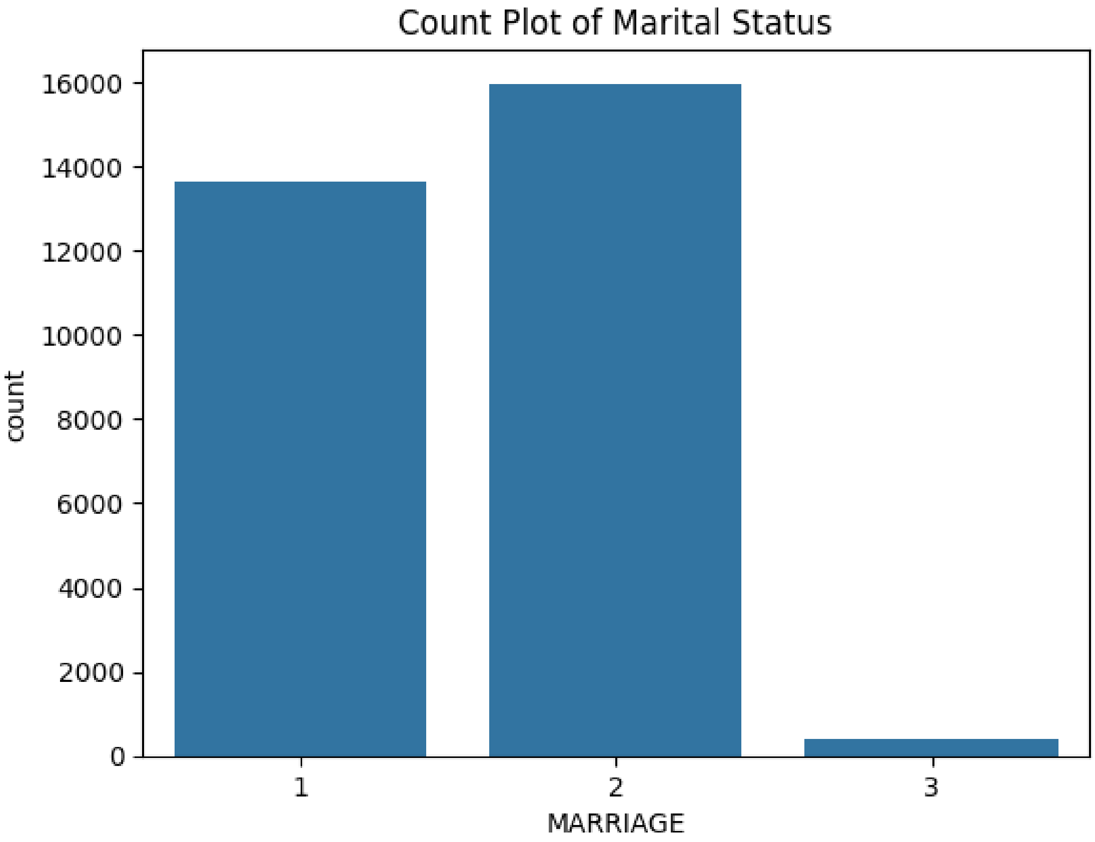
```
[48]: # Count plot for education level
      sns.countplot(x='EDUCATION', data=data)
      plt.title('Count Plot of Education Level')
      plt.show()
```



```
[49]: data['MARRIAGE'].value_counts(normalize=True) * 100
```

```
[49]: MARRIAGE
      2    53.213333
      1    45.530000
      3     1.256667
      Name: proportion, dtype: float64
```

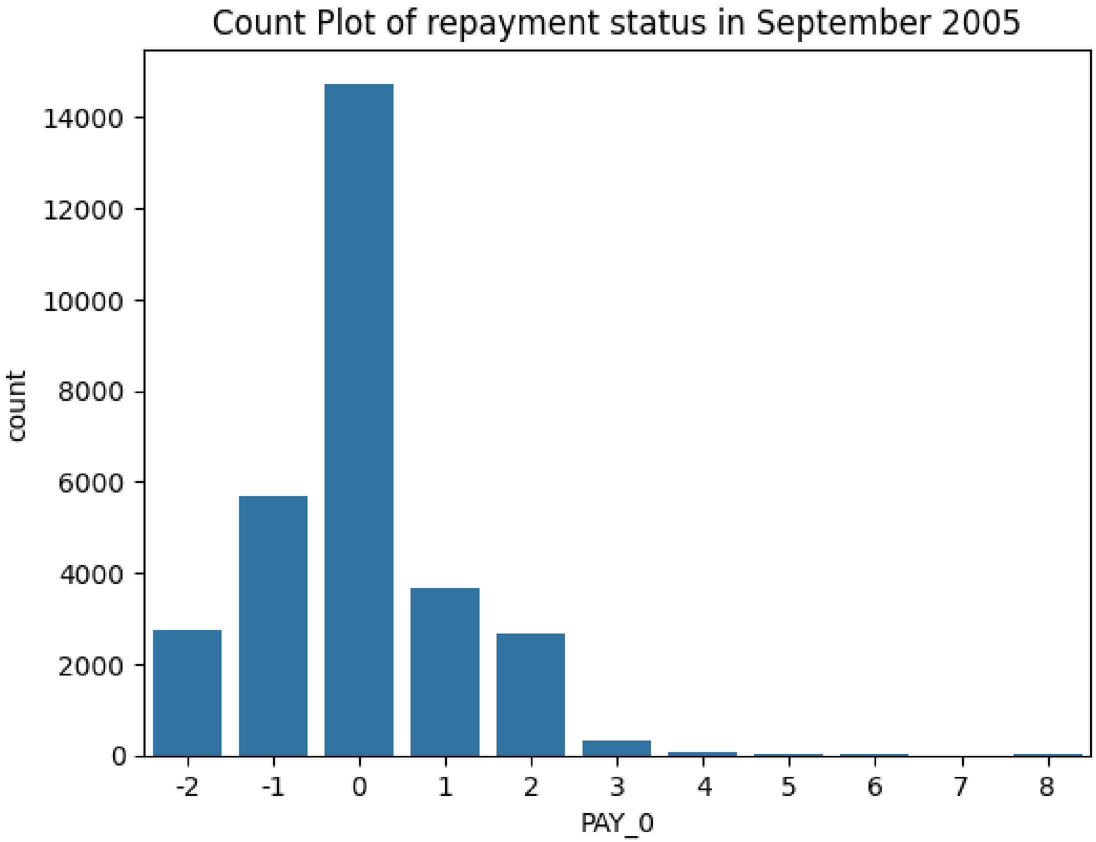
```
[50]: # Count plot for marital status
      sns.countplot(x='MARRIAGE', data=data)
      plt.title('Count Plot of Marital Status')
      plt.show()
```



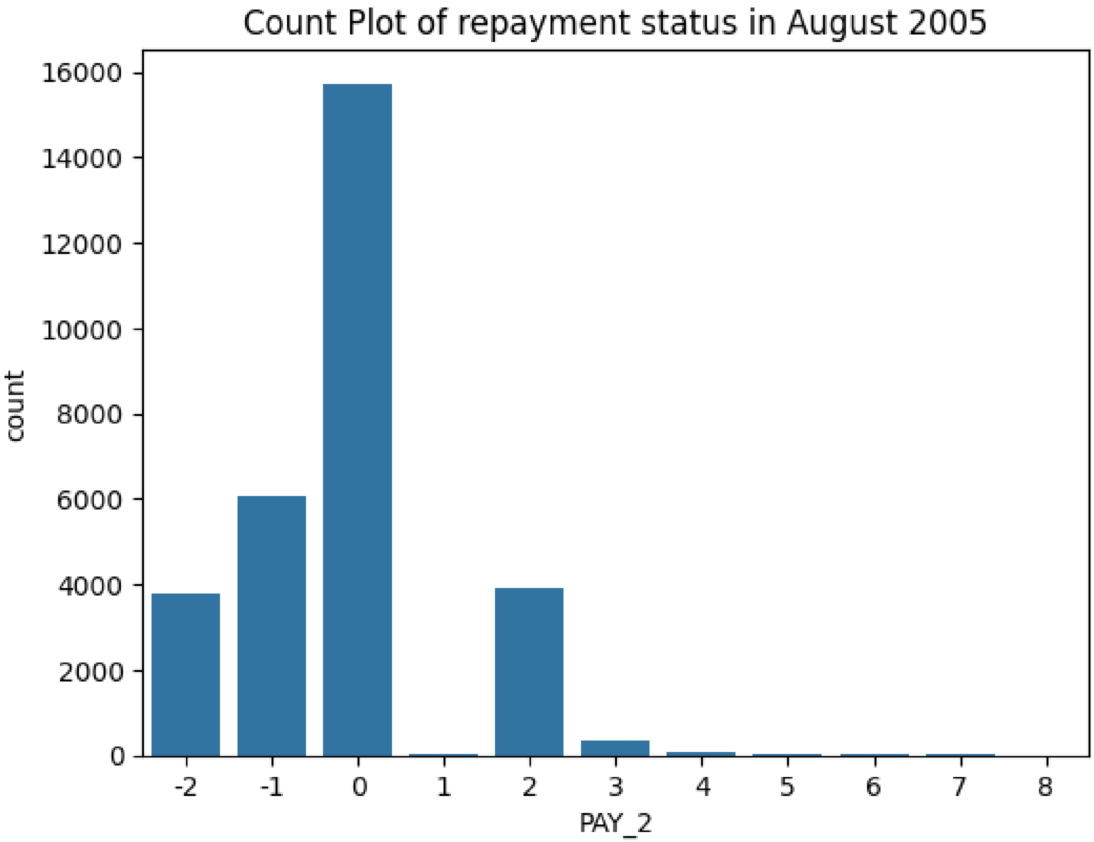
[51]:
```
data['PAY_0'].value_counts(normalize=True) * 100
```

[51]:
```
PAY_0
 0    49.123333
-1    18.953333
 1    12.293333
-2     9.196667
 2     8.890000
 3     1.073333
 4     0.253333
 5     0.086667
 8     0.063333
 6     0.036667
 7     0.030000
Name: proportion, dtype: float64
```

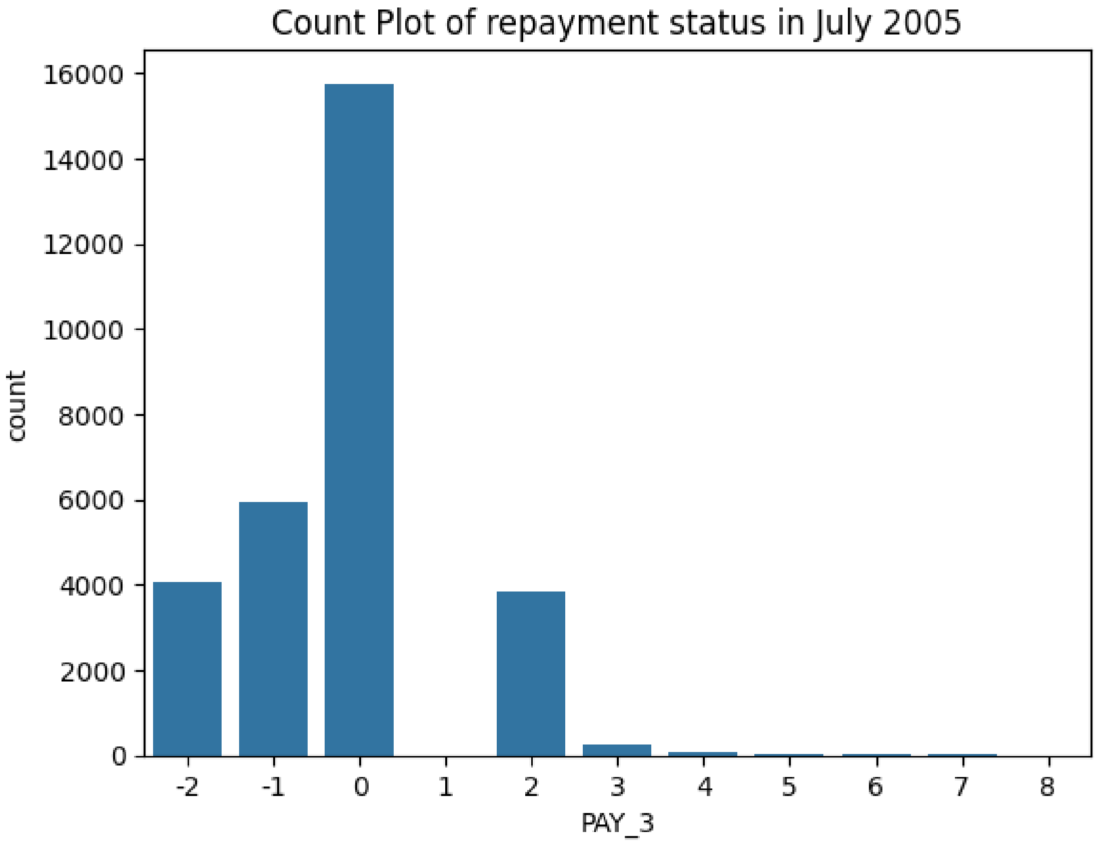
[52]:
```
# Count plot for repayment status in September 2005
sns.countplot(x='PAY_0', data=data)
plt.title('Count Plot of repayment status in September 2005')
plt.show()
```



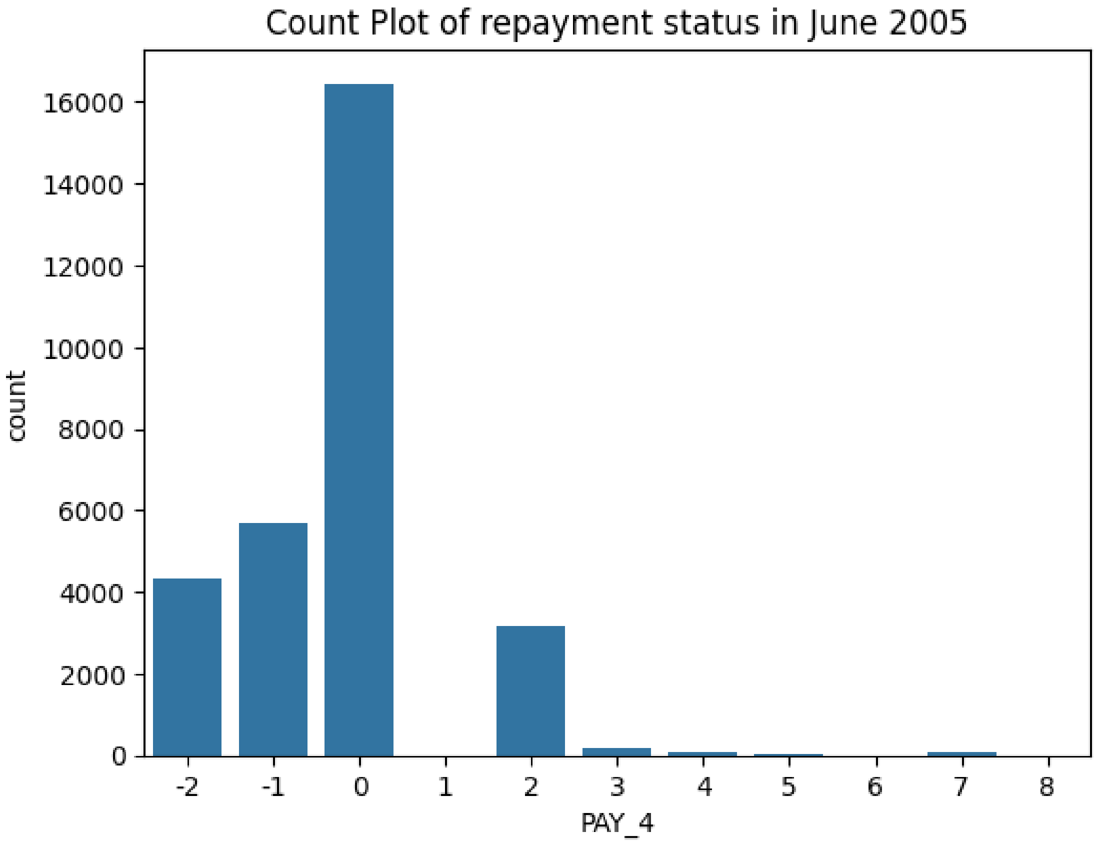
[53]:
```
# Count plot for repayment status in August 2005
sns.countplot(x='PAY_2', data=data)
plt.title('Count Plot of repayment status in August 2005')
plt.show()
```



[54]:
```
# Count plot for repayment status in July 2005
sns.countplot(x='PAY_3', data=data)
plt.title('Count Plot of repayment status in July 2005')
plt.show()
```

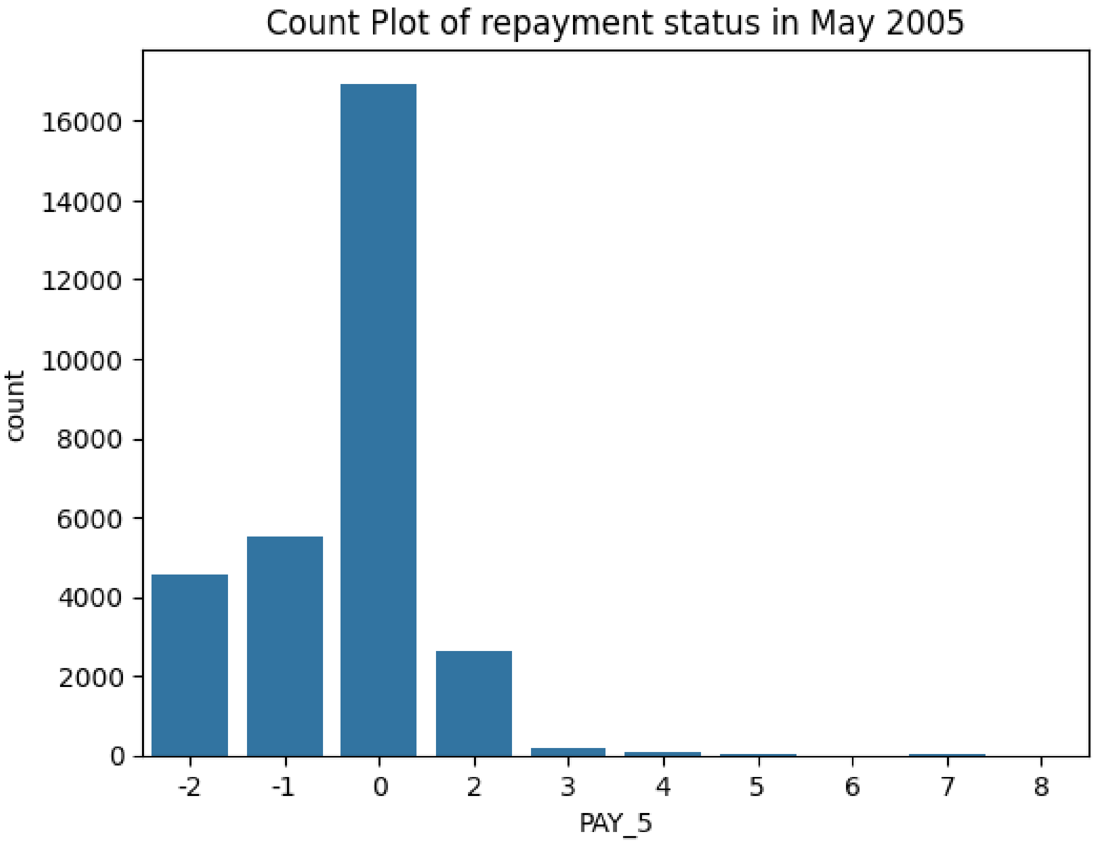


[55]:
```
# Count plot for repayment status in June 2005
sns.countplot(x='PAY_4', data=data)
plt.title('Count Plot of repayment status in June 2005')
plt.show()
```

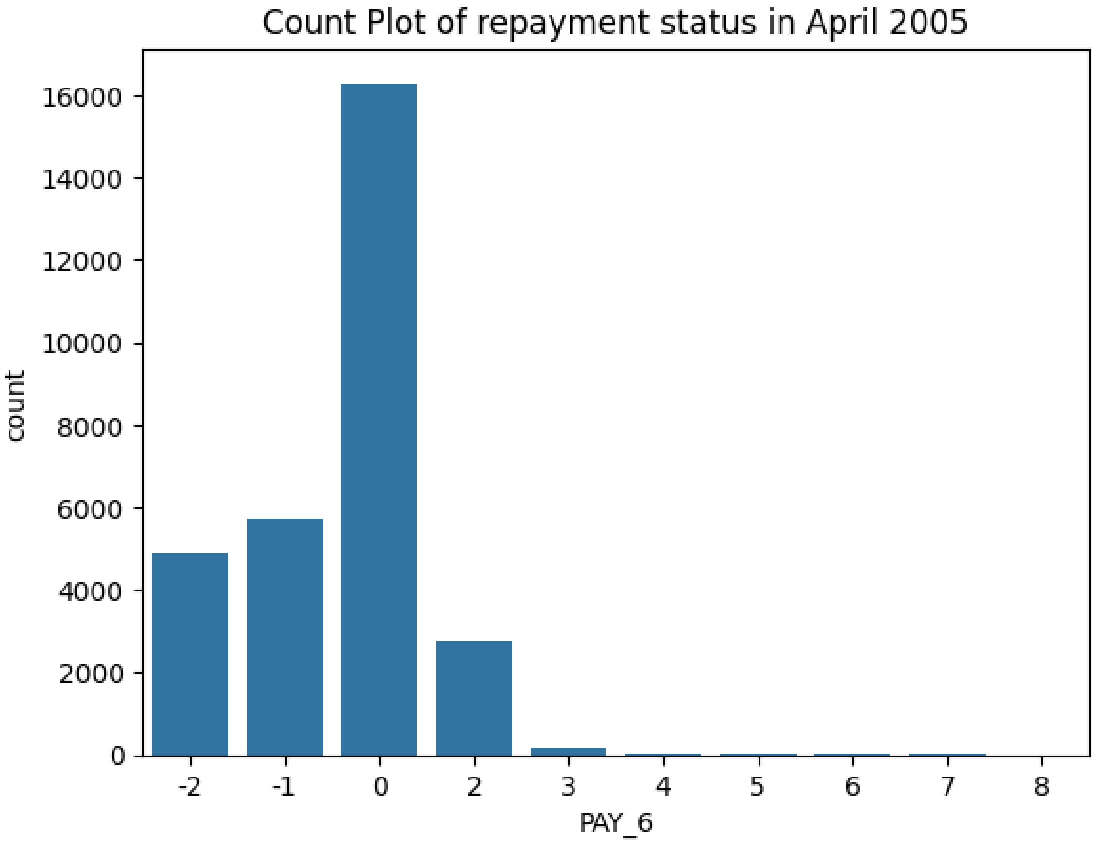


[56]:

```python
# Count plot for repayment status in May 2005
sns.countplot(x='PAY_5', data=data)
plt.title('Count Plot of repayment status in May 2005')
plt.show()
```



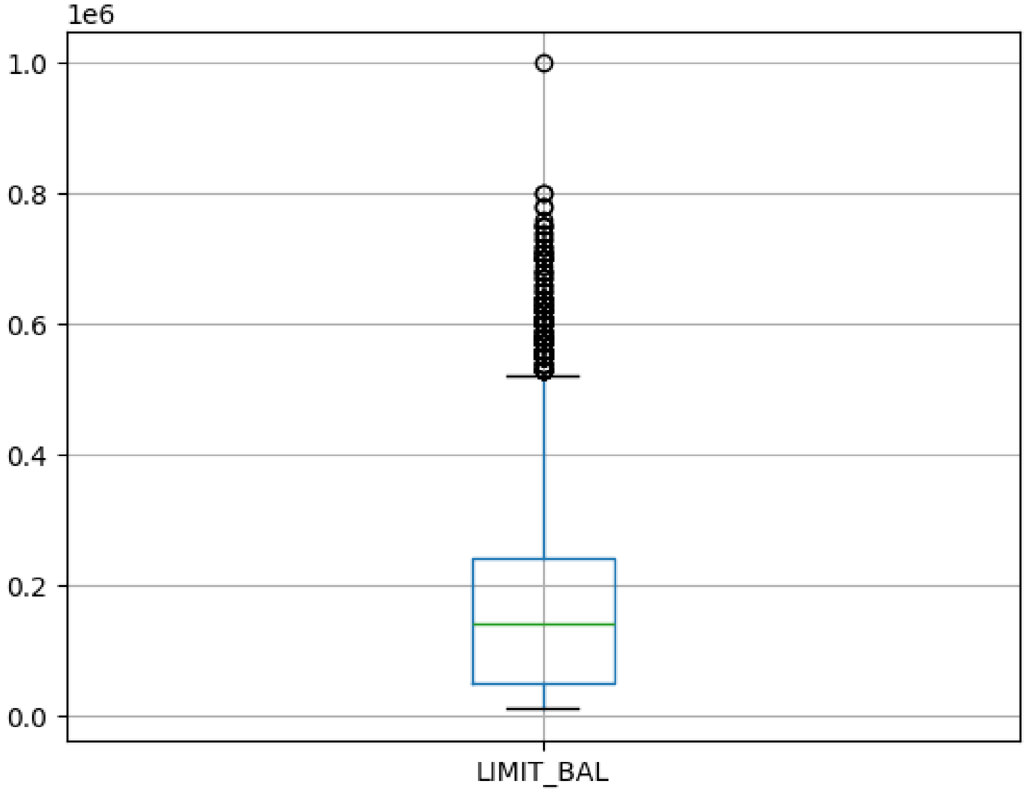
[57]:
```
# Count plot for repayment status in April 2005
sns.countplot(x='PAY_6', data=data)
plt.title('Count Plot of repayment status in April 2005')
plt.show()
```



```
[58]: numeric_variables = ['LIMIT_BAL', 'AGE', 'BILL_AMT1','BILL_AMT2', 'BILL_AMT3',␣
      ↪'BILL_AMT4', 'BILL_AMT5', 'BILL_AMT6', 'PAY_AMT1', 'PAY_AMT2', 'PAY_AMT3',␣
      ↪'PAY_AMT4', 'PAY_AMT5', 'PAY_AMT6']
```

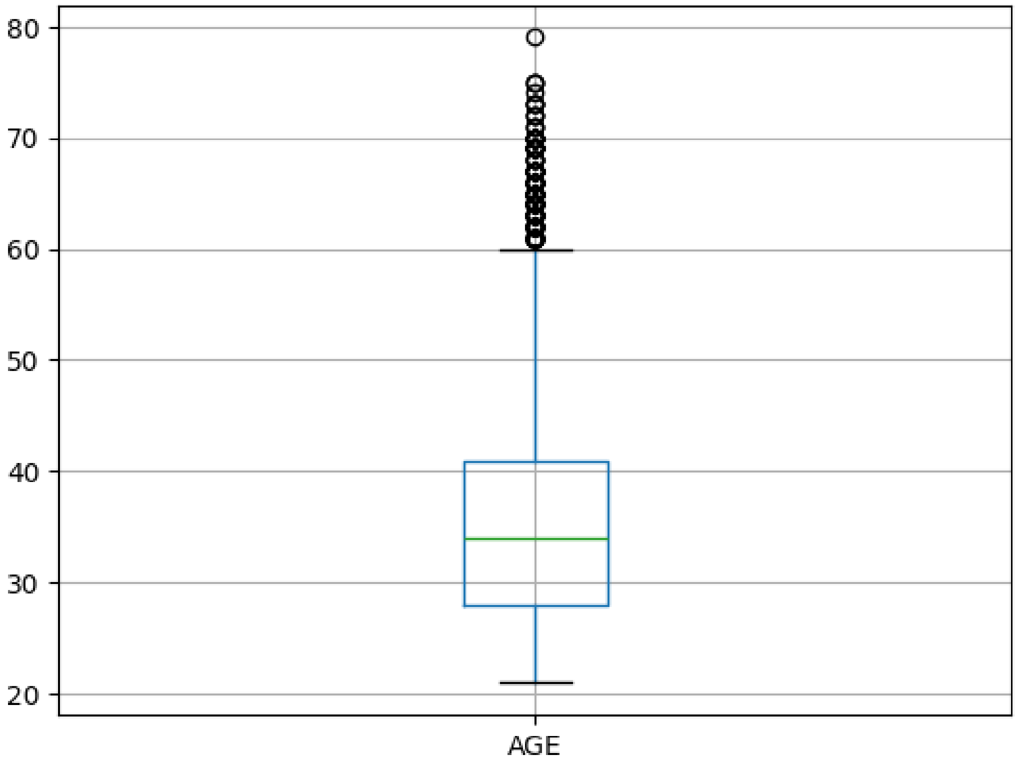
```
[59]: # Limit Balance boxplot before treating outliers:
      data.boxplot(column = 'LIMIT_BAL')
```

```
[59]: <Axes: >
```



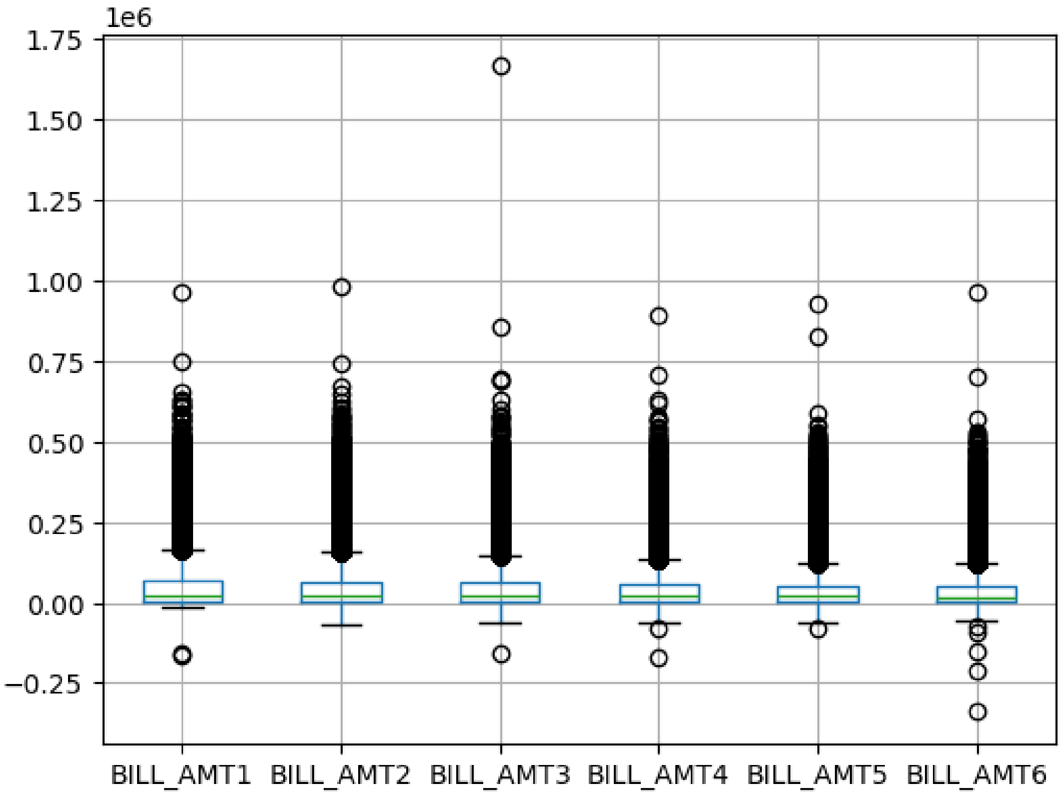
```
[60]: data.boxplot(column = 'AGE')
```

```
[60]: <Axes: >
```



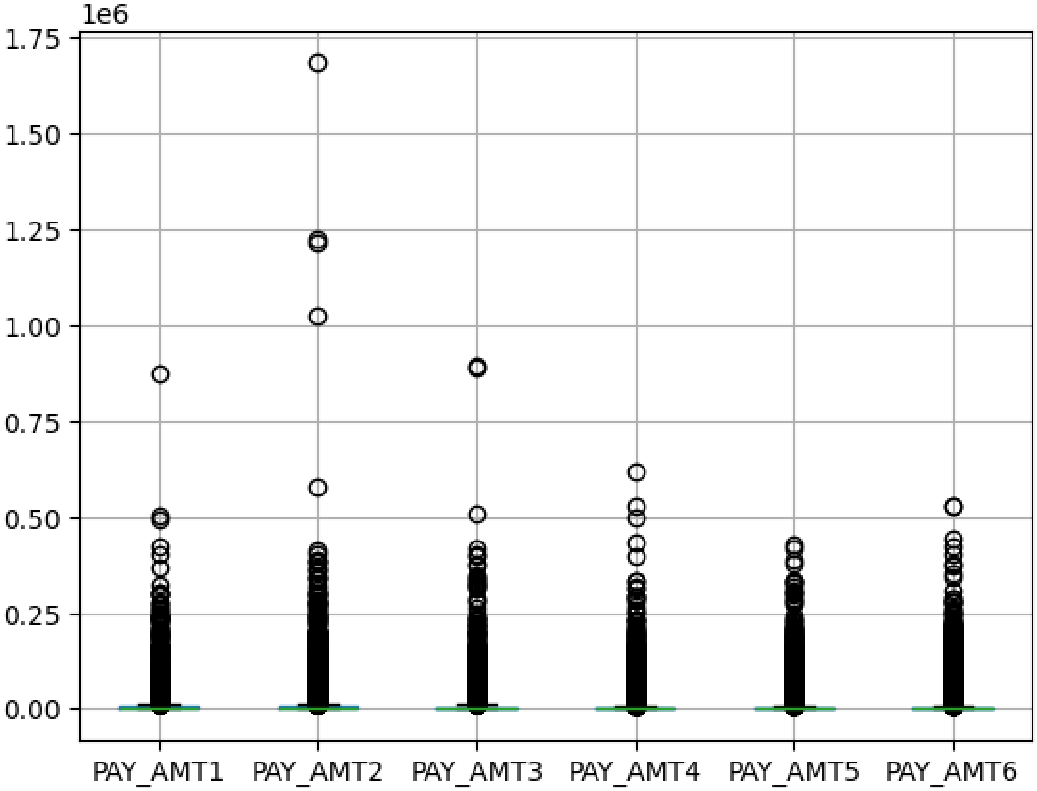
```
[61]: # Bill Amounts boxplot before treating outliers:
      pd.DataFrame(data = data, columns = ['BILL_AMT1', 'BILL_AMT2', 'BILL_AMT3',␣
       ↪'BILL_AMT4', 'BILL_AMT5', 'BILL_AMT6']).boxplot()
```

```
[61]: <Axes: >
```



```
[62]: # Payment Amounts boxplot before treating outliers:
      pd.DataFrame(data = data, columns = ['PAY_AMT1', 'PAY_AMT2', 'PAY_AMT3',␣
       ↪'PAY_AMT4', 'PAY_AMT5', 'PAY_AMT6']).boxplot()
```

```
[62]: <Axes: >
```



[63]:

```
# Getting the summary statistics before treating outliers
print("Summary Statistics of numeric variables before replacing outliers with␣
 ↪median:")
data[numeric_variables].describe().transpose()
```

```
Summary Statistics of numeric variables before replacing outliers with median:
```

[63]:

```
             count           mean            std       min       25%  \
LIMIT_BAL  30000.0  167484.322667  129747.661567   10000.0  50000.00
AGE        30000.0      35.485500       9.217904      21.0     28.00
BILL_AMT1  30000.0   51223.330900   73635.860576 -165580.0   3558.75
BILL_AMT2  30000.0   49179.075167   71173.768783  -69777.0   2984.75
BILL_AMT3  30000.0   47013.154800   69349.387427 -157264.0   2666.25
BILL_AMT4  30000.0   43262.948967   64332.856134 -170000.0   2326.75
BILL_AMT5  30000.0   40311.400967   60797.155770  -81334.0   1763.00
BILL_AMT6  30000.0   38871.760400   59554.107537 -339603.0   1256.00
PAY_AMT1   30000.0    5663.580500   16563.280354       0.0   1000.00
PAY_AMT2   30000.0    5921.163500   23040.870402       0.0    833.00
PAY_AMT3   30000.0    5225.681500   17606.961470       0.0    390.00
PAY_AMT4   30000.0    4826.076867   15666.159744       0.0    296.00
PAY_AMT5   30000.0    4799.387633   15278.305679       0.0    252.50
```

```
PAY_AMT6   30000.0    5215.502567   17777.465775       0.0    117.75

                50%        75%        max
LIMIT_BAL  140000.0  240000.00  1000000.0
AGE            34.0      41.00       79.0
BILL_AMT1   22381.5   67091.00   964511.0
BILL_AMT2   21200.0   64006.25   983931.0
BILL_AMT3   20088.5   60164.75  1664089.0
BILL_AMT4   19052.0   54506.00   891586.0
BILL_AMT5   18104.5   50190.50   927171.0
BILL_AMT6   17071.0   49198.25   961664.0
PAY_AMT1     2100.0    5006.00   873552.0
PAY_AMT2     2009.0    5000.00  1684259.0
PAY_AMT3     1800.0    4505.00   896040.0
PAY_AMT4     1500.0    4013.25   621000.0
PAY_AMT5     1500.0    4031.50   426529.0
PAY_AMT6     1500.0    4000.00   528666.0
```

[64]:

```python
# Defining function to replace outliers with the median
def replace_outliers_with_median(data, column):
    median = data[column].median()
    q1 = data[column].quantile(0.25)
    q3 = data[column].quantile(0.75)
    iqr = q3 - q1
    lower_bound = q1 - 1.5 * iqr
    upper_bound = q3 + 1.5 * iqr

    data[column] = data[column].apply(lambda x: median if x < lower_bound or x > upper_bound else x)
```

[65]:

```python
# Applying the function to the columns with outliers
replace_outliers_with_median(data, 'LIMIT_BAL')
replace_outliers_with_median(data, 'BILL_AMT1')
replace_outliers_with_median(data, 'BILL_AMT2')
replace_outliers_with_median(data, 'BILL_AMT3')
replace_outliers_with_median(data, 'BILL_AMT4')
replace_outliers_with_median(data, 'BILL_AMT5')
replace_outliers_with_median(data, 'BILL_AMT6')
replace_outliers_with_median(data, 'PAY_AMT1')
replace_outliers_with_median(data, 'PAY_AMT2')
replace_outliers_with_median(data, 'PAY_AMT3')
replace_outliers_with_median(data, 'PAY_AMT4')
replace_outliers_with_median(data, 'PAY_AMT5')
replace_outliers_with_median(data, 'PAY_AMT6')
```

[66]:

```python
print("\nSummary Statistics of numeric variables after replacing outliers with median:")
```

```
data[numeric_variables].describe().transpose()
```

```
Summary Statistics of numeric variables after replacing outliers with median:
```

[66]:

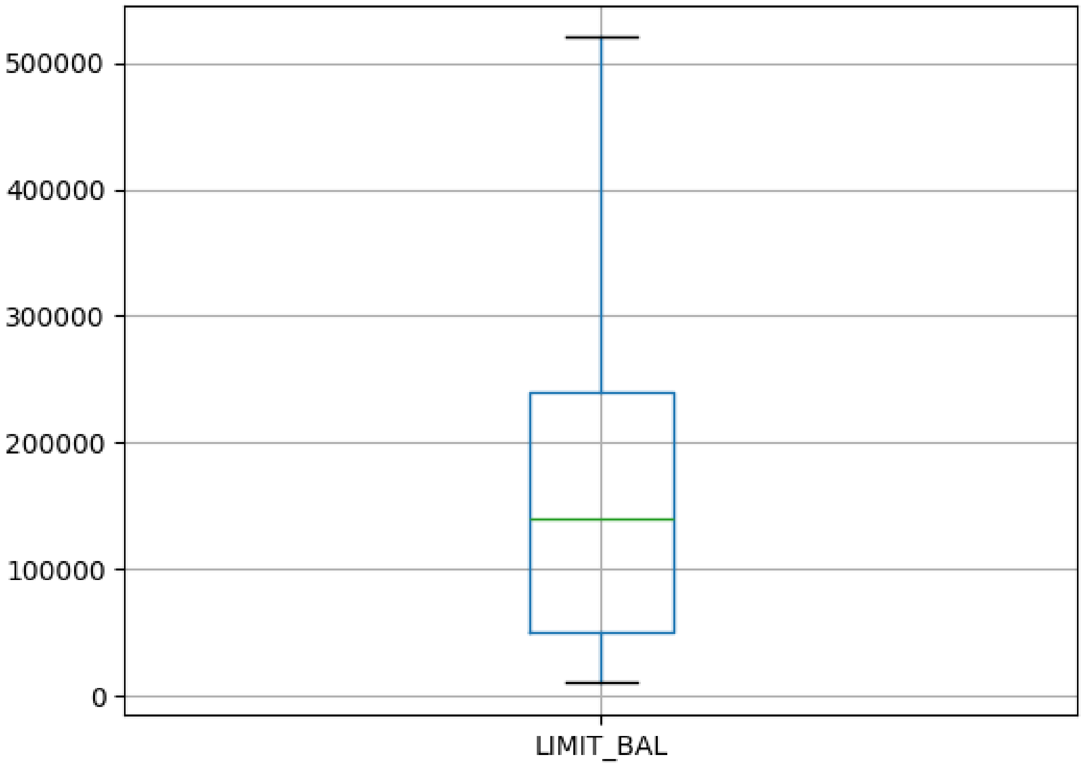
```
             count           mean            std      min       25%       50%  \
LIMIT_BAL  30000.0  164824.322667  125192.989579  10000.0  50000.00  140000.0
AGE        30000.0      35.485500       9.217904     21.0     28.00      34.0
BILL_AMT1  30000.0   33109.792100   37794.502441 -15308.0   3563.00   22381.5
BILL_AMT2  30000.0   31669.887567   36414.965831 -69777.0   2984.75   21198.5
BILL_AMT3  30000.0   29736.798283   34293.746628 -61506.0   2667.75   20088.5
BILL_AMT4  30000.0   26625.608833   30764.323883 -65167.0   2329.00   19052.0
BILL_AMT5  30000.0   24247.883050   28331.916539 -61372.0   1763.75   18104.5
BILL_AMT6  30000.0   23287.670000   27946.193005 -57060.0   1259.75   17071.0
PAY_AMT1   30000.0    2681.008300    2557.378286      0.0   1000.00    2100.0
PAY_AMT2   30000.0    2586.259267    2533.473459      0.0    833.00    2009.0
PAY_AMT3   30000.0    2267.026400    2396.721279      0.0    390.00    1800.0
PAY_AMT4   30000.0    1911.001400    2056.702179      0.0    296.00    1500.0
PAY_AMT5   30000.0    1926.580500    2075.388113      0.0    252.50    1500.0
PAY_AMT6   30000.0    1893.753100    2071.970037      0.0    117.75    1500.0

                 75%       max
LIMIT_BAL  240000.00  520000.0
AGE            41.00      79.0
BILL_AMT1   48707.50  162296.0
BILL_AMT2   47812.25  155508.0
BILL_AMT3   44887.75  146410.0
BILL_AMT4   37803.00  132754.0
BILL_AMT5   32030.50  122830.0
BILL_AMT6   30563.00  121062.0
PAY_AMT1     3706.00   11013.0
PAY_AMT2     3500.00   11249.0
PAY_AMT3     3005.00   10673.0
PAY_AMT4     2816.25    9584.0
PAY_AMT5     2913.50    9700.0
PAY_AMT6     2853.50    9817.0
```

[67]:

```
# Limit Balance boxplot after treating outliers:
data.boxplot(column = 'LIMIT_BAL')
```

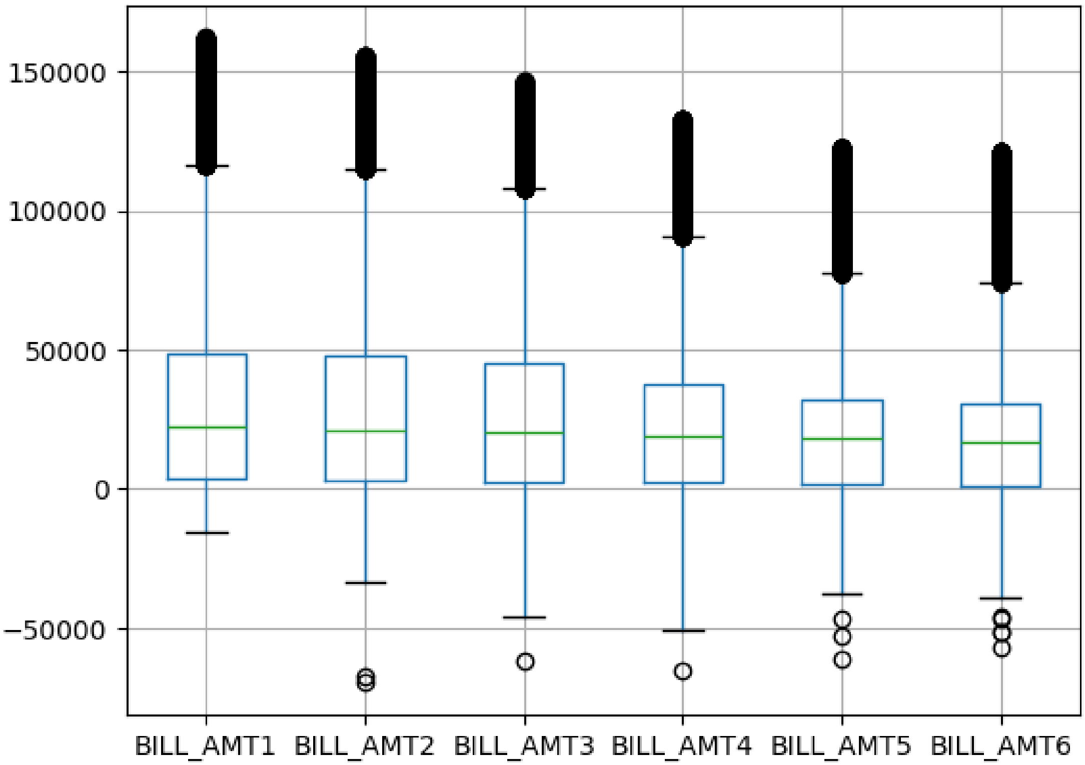
[67]:

```
<Axes: >
```



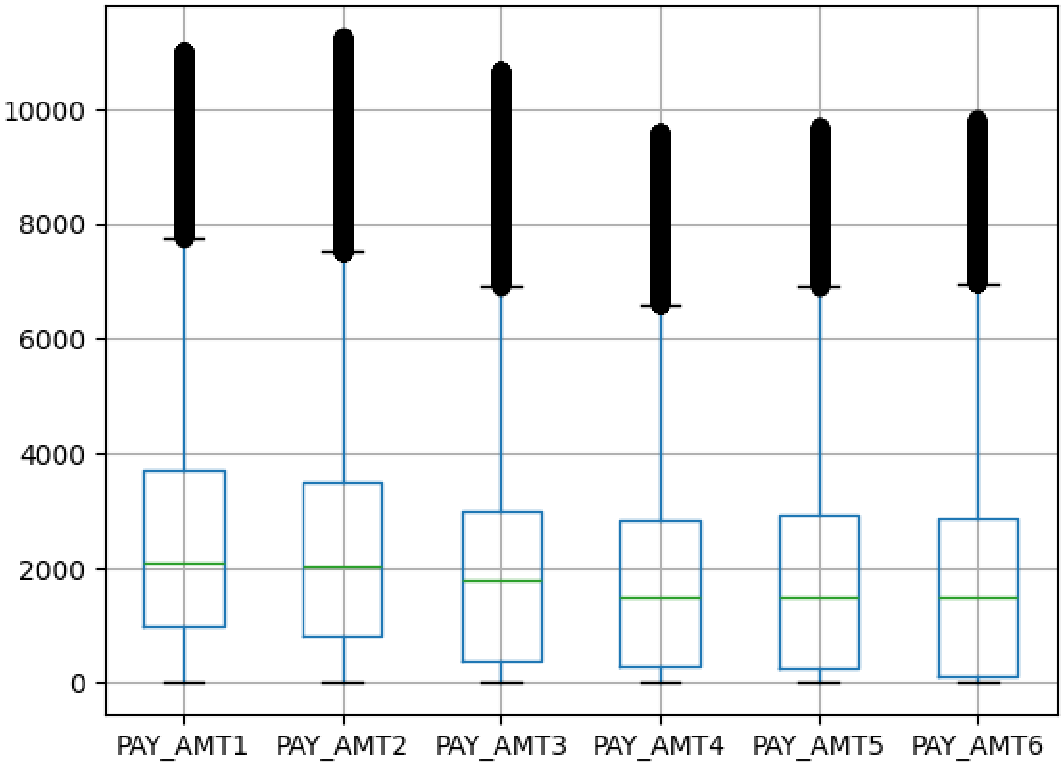
[68]:
```
# Bill Amounts boxplot after treating outliers:
pd.DataFrame(data = data, columns = ['BILL_AMT1', 'BILL_AMT2', 'BILL_AMT3',␣
 ↪'BILL_AMT4', 'BILL_AMT5', 'BILL_AMT6']).boxplot()
```

[68]:
```
<Axes: >
```



```
[69]: # Payment Amounts boxplot after treating outliers:
      pd.DataFrame(data = data, columns = ['PAY_AMT1', 'PAY_AMT2', 'PAY_AMT3',␣
       ↪'PAY_AMT4', 'PAY_AMT5', 'PAY_AMT6']).boxplot()
```

```
[69]: <Axes: >
```


10000
8000
6000
4000
2000
0
PAY_AMT1
PAY_AMT2
PAY_AMT3
PAY_AMT4
PAY_AMT5
PAY_AMT6