In This notebook we have done some data visualization for :

- Checking for missing values
- Data Sanity Checks
- Outlier Capping
- Basic Analysis of the data

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
# sns.set()
from sklearn.model_selection import train_test_split
from collections import Counter
import random
# import logging
```

```
df=pd.read_csv('UCI_Credit_Card.csv')
```

```
df.shape
```

```
(30000, 25)
```

```
df.drop('ID',axis=1,inplace=True)
```

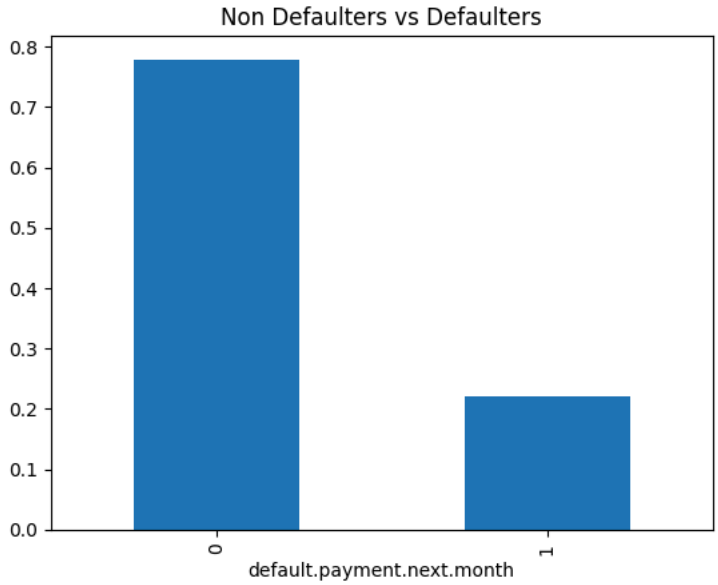
CHECKING FOR NULL VALUES

```
df.isnull().sum(axis=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
```

TARGET ANALYSIS

```
df['default.payment.next.month'].value_counts(normalize=True).plot.bar()
plt.title('Non Defaulters vs Defaulters')
plt.show()
```



Inference

- prob_default=0.22
- Imbalanced data

SEX EDUCATION MARRIAGE

- SEX: Gender (1=male, 2=female)
- EDUCATION: (1=graduate school, 2=university, 3=high school, 4=others, 5=unknown, 6=unknown)
- MARRIAGE: Marital status (1=married, 2=single, 3=others)

```
df[['SEX','EDUCATION','MARRIAGE']].describe()
```

	SEX	EDUCATION	MARRIAGE
count	30000.000000	30000.000000	30000.000000
mean	1.603733	1.853133	1.551867
std	0.489129	0.790349	0.521970
min	1.000000	0.000000	0.000000
25%	1.000000	1.000000	1.000000
50%	2.000000	2.000000	2.000000
75%	2.000000	2.000000	2.000000
max	2.000000	6.000000	3.000000

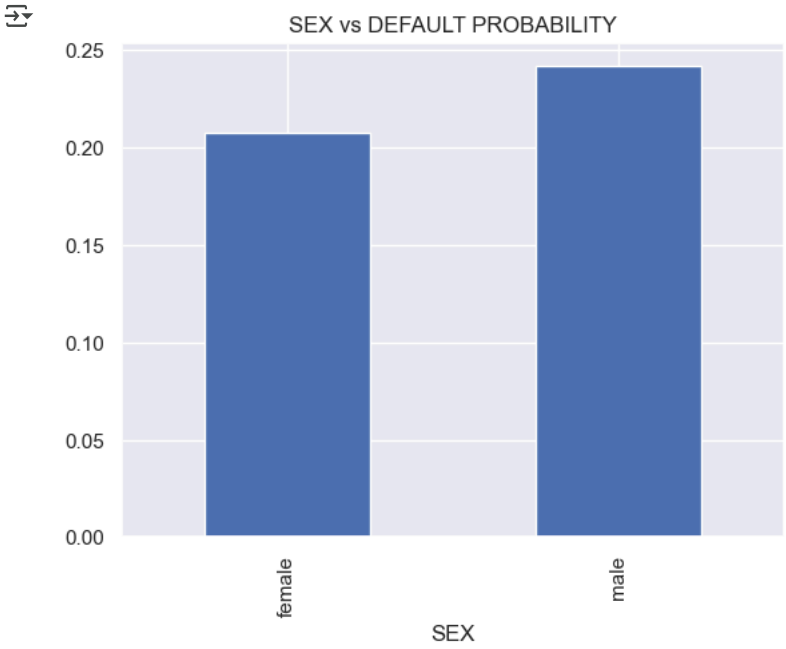
- Education and marriage have a category 0 which is unknown

```
# Replacing 0 of marriage with 3(others)

df['MARRIAGE']=df['MARRIAGE'].replace({0:3})


# Replacing 0,5,6 with 4

df['EDUCATION']=df['EDUCATION'].replace({0:4,5:4,6:4})


df['MARRIAGE']=df['MARRIAGE'].replace({1:'married',2:'single',3:'others'})
df['EDUCATION']=df['EDUCATION'].replace({1:'graduate school',2:'university',3:'high school',4:'others'})
df['SEX']=df['SEX'].replace({1:'male', 2:'female'})


df[['MARRIAGE']].value_counts()
```

```
MARRIAGE
single      15964
married     13659
others        377
Name: count, dtype: int64
```

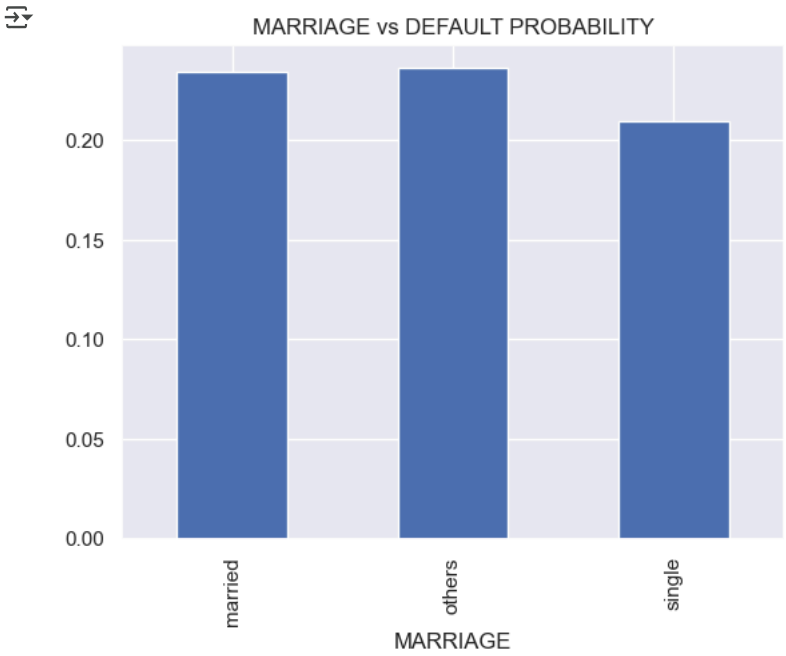
```
def plot_categoric(cat_col):


    plt.title('{0} vs DEFAULT PROBABILITY'.format(cat_col))
    df.groupby(cat_col)['default.payment.next.month'].mean().plot.bar()
    plt.show()
```

```
plot_categoric('SEX')
```

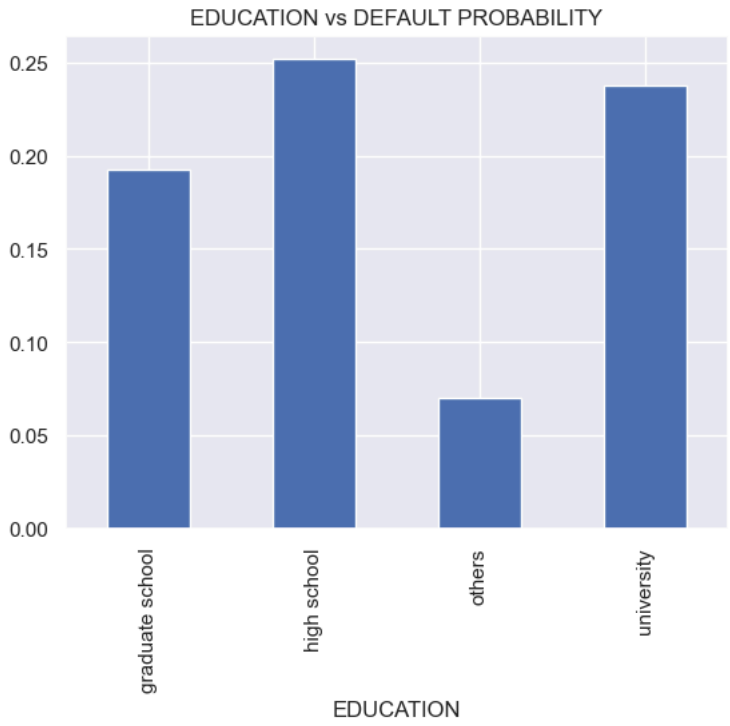


- Men are more likely to default than feamles

```
plot_categoric('MARRIAGE')
```




```
plot_categoric('EDUCATION')
```

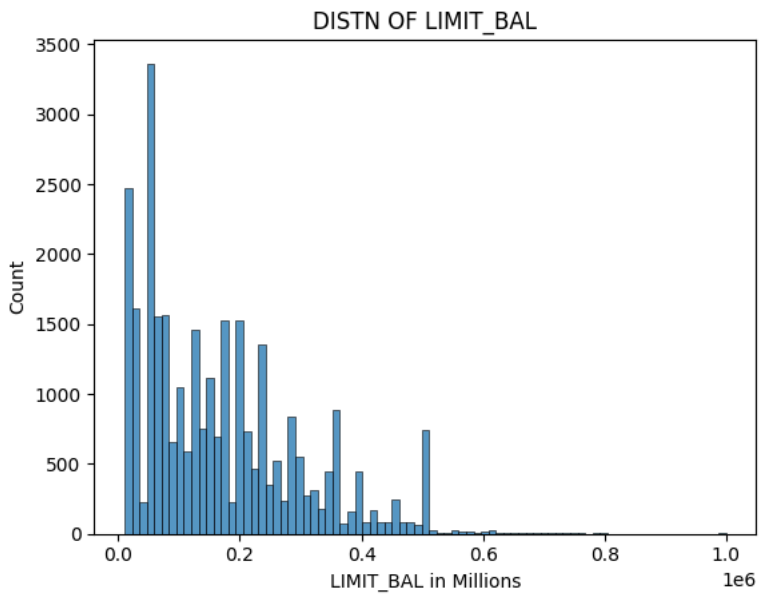


Inference On Categoric Variables

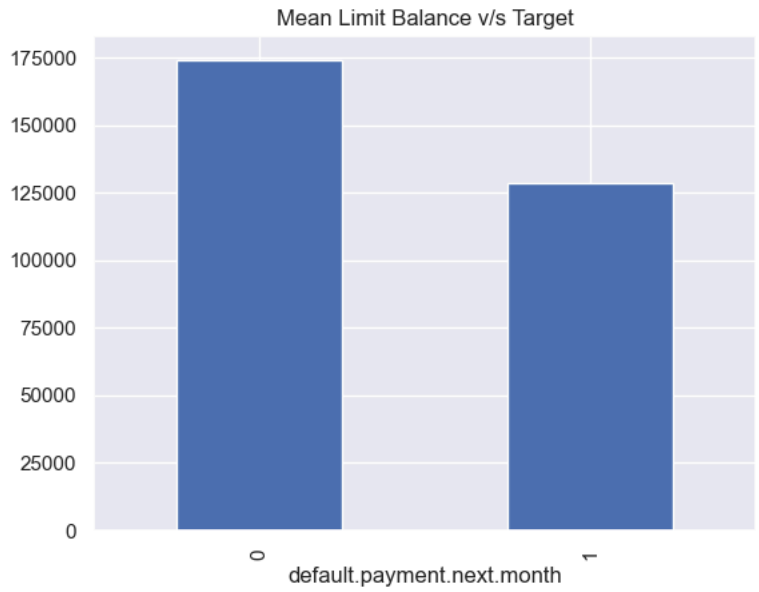
- University and high school people are more likely to default.
- Married People are more likely to default.
- Males are more likely to default

Limit Balance And Influence On Target Var

```
sns.histplot(df['LIMIT_BAL'])
plt.title('DISTN OF LIMIT_BAL')
plt.xlabel('LIMIT_BAL in Millions')
plt.show()
```




```
df.groupby('default.payment.next.month')['LIMIT_BAL'].mean().plot.bar()
plt.title('Mean Limit Balance v/s Target')
plt.show()
```

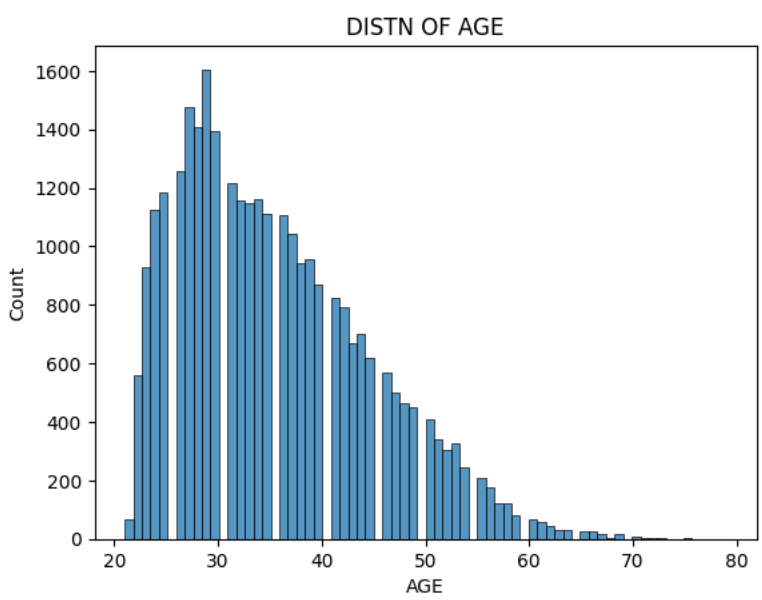


Inference

- Very few have a high limit balance
- Those who do not default have a higher limit balance

AGE And Influence On Target

```
sns.histplot(df['AGE'])
plt.title('DISTN OF AGE')
plt.show()
```

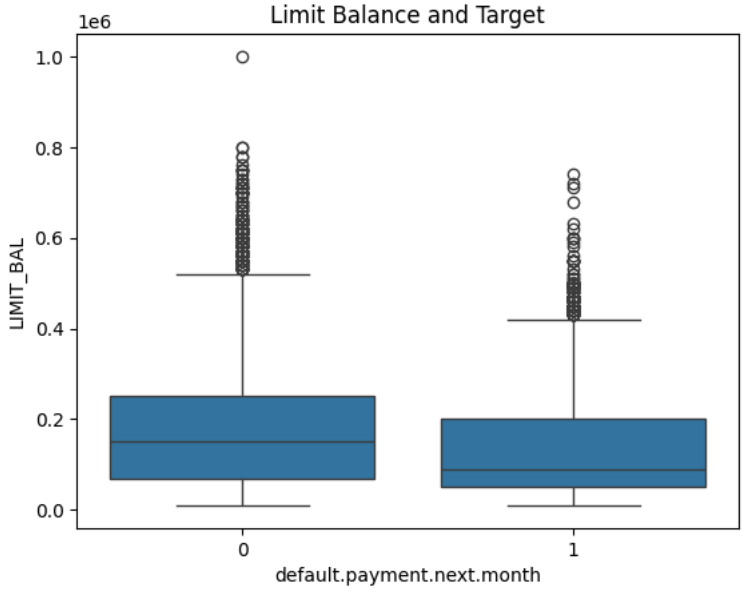


- Most people are in late 20's and early 30's

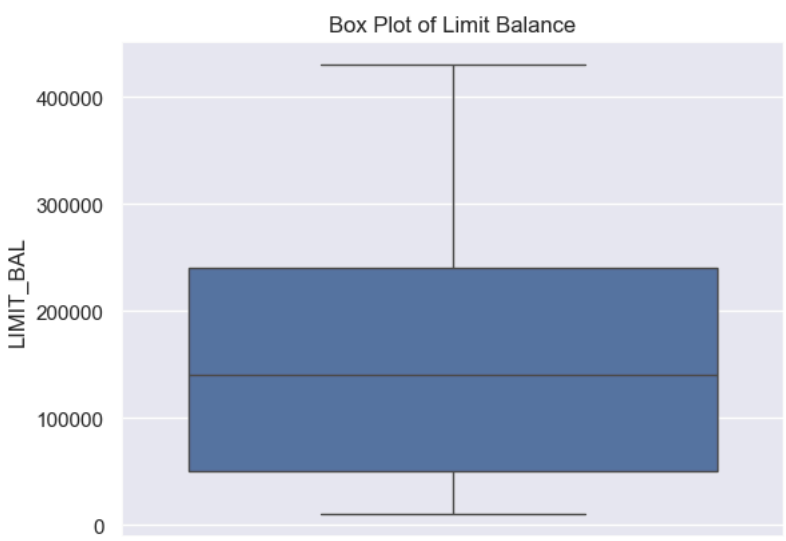
ANALYSING NUMERIC FEATURES FOR OUTLIERS

1.Limit Balance

```
sns.boxplot(x=df['default.payment.next.month'],y=df['LIMIT_BAL'],data=df)
plt.title('Limit Balance and Target')
plt.show()
```




```
sns.boxplot(df['LIMIT_BAL'])
plt.title('Box Plot of Limit Balance')
plt.show()
```



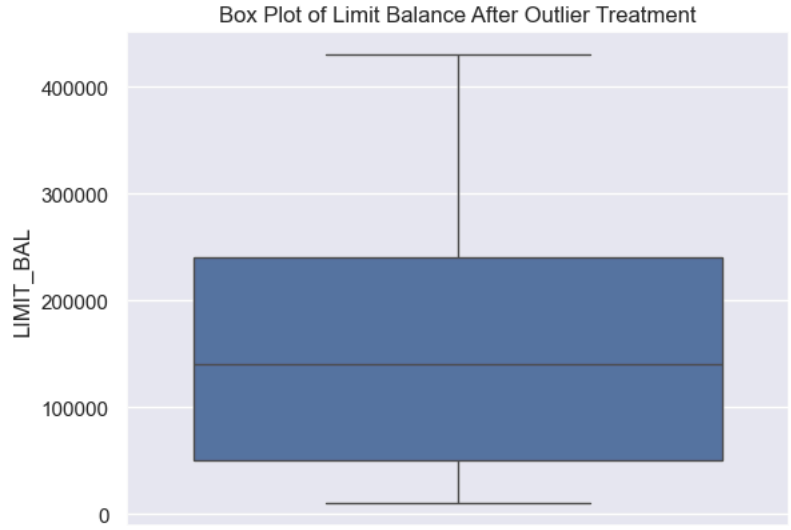

```
df['LIMIT_BAL'].describe()
```

```
count     30000.000000
mean     164061.989333
std      121035.014953
min       10000.000000
25%       50000.000000
50%      140000.000000
75%      240000.000000
max      430000.000000
Name: LIMIT_BAL, dtype: float64
```

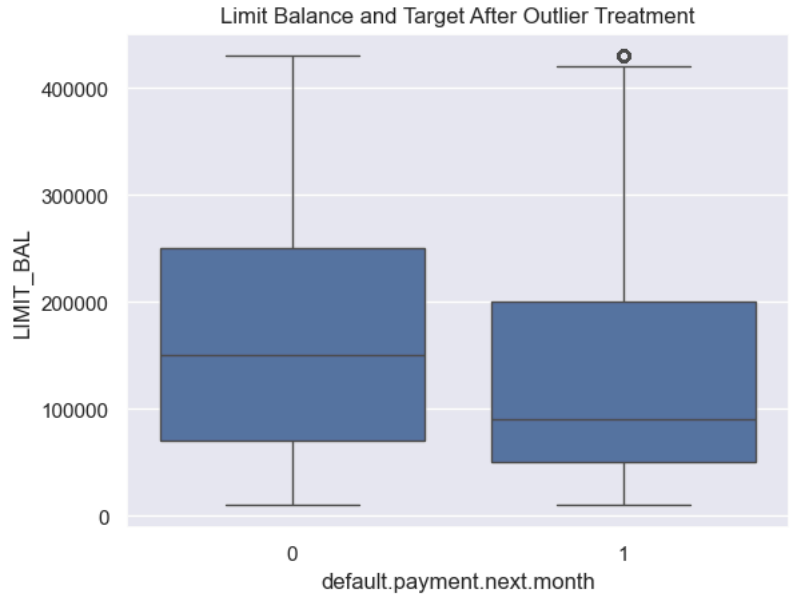
Capping to 95%

```
df.loc[df['LIMIT_BAL']>=df['LIMIT_BAL'].quantile(0.95),'LIMIT_BAL']=df['LIMIT_BAL'].quantile(0.95)
```

```
# Checking the outliers now
sns.boxplot(df['LIMIT_BAL'])
plt.title('Box Plot of Limit Balance After Outlier Treatment')
plt.show()
```




```
sns.boxplot(x=df['default.payment.next.month'],y=df['LIMIT_BAL'],data=df)
plt.title('Limit Balance and Target After Outlier Treatment')
plt.show()
```



2.BILL AMT (1-6)

```
df[['BILL_AMT1','BILL_AMT2','BILL_AMT3','BILL_AMT4','BILL_AMT5','BILL_AMT6']].describe()
```

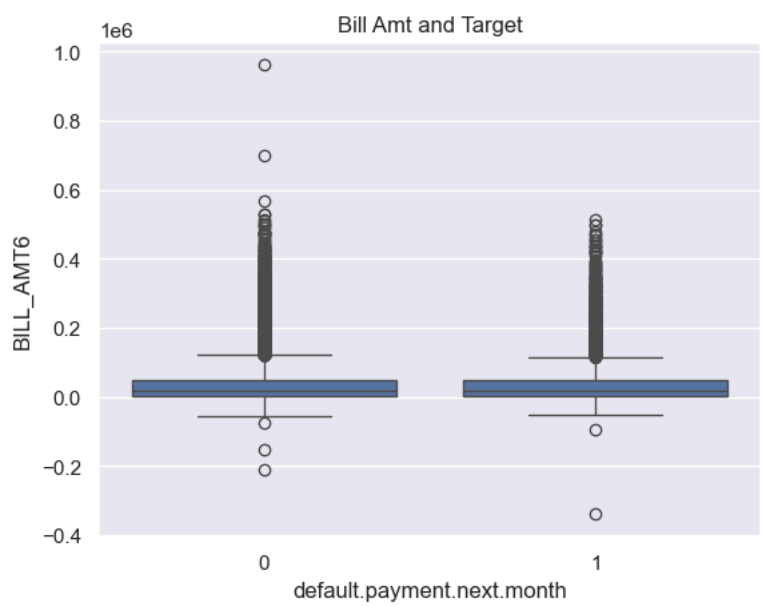
	BILL_AMT1	BILL_AMT2	BILL_AMT3	BILL_AMT4	BILL_AMT5	BILL_AMT6
count	30000.000000	30000.000000	3.000000e+04	30000.000000	30000.000000	30000.000000
mean	51223.330900	49179.075167	4.701315e+04	43262.948967	40311.400967	38871.760400
std	73635.860576	71173.768783	6.934939e+04	64332.856134	60797.155770	59554.107537
min	-165580.000000	-69777.000000	-1.572640e+05	-170000.000000	-81334.000000	-339603.000000
25%	3558.750000	2984.750000	2.666250e+03	2326.750000	1763.000000	1256.000000
50%	22381.500000	21200.000000	2.008850e+04	19052.000000	18104.500000	17071.000000
75%	67091.000000	64006.250000	6.016475e+04	54506.000000	50190.500000	49198.250000
max	964511.000000	983931.000000	1.664089e+06	891586.000000	927171.000000	961664.000000

- Handling negative values

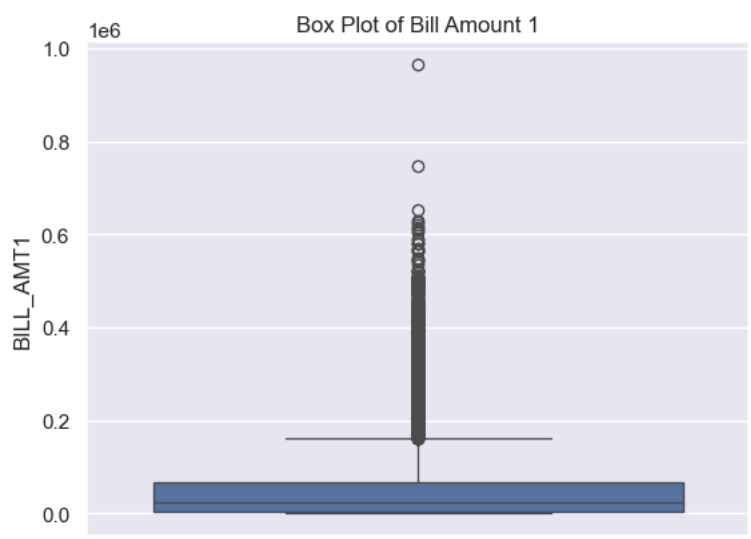
```
for i in range(1,7):
    feature='BILL_AMT'+str(i)+'_POS'
    df[feature]=df['BILL_AMT'+str(i)].apply(lambda x:1 if x>=0 else 0)
```

```
for i in range(1,6):
    feature='BILL_AMT'+str(i)
    df[feature]=df[feature].apply(lambda x:abs(x))
```

```
sns.boxplot(x=df['default.payment.next.month'],y=df['BILL_AMT6'],data=df)
plt.title('Bill Amt and Target')
plt.show()
```




```
sns.boxplot(df['BILL_AMT1'])
plt.title('Box Plot of Bill Amount 1')
plt.show()
```

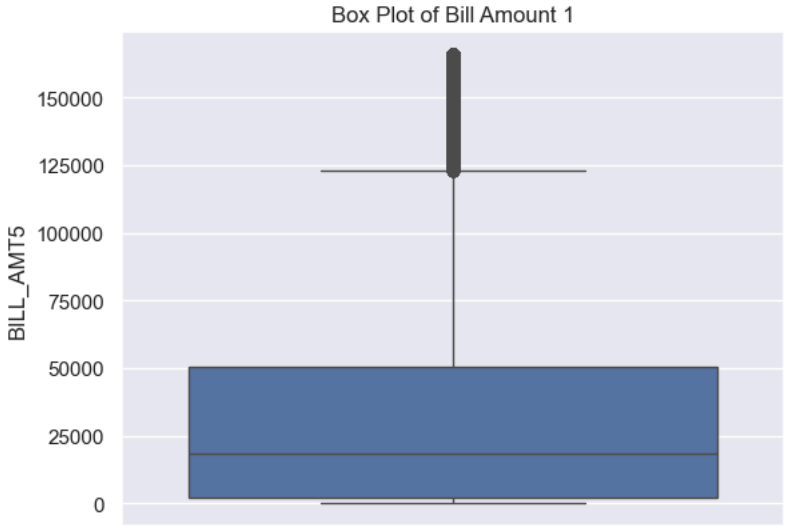


- Bill amount variables do not show much variability with the target
- Capping them at 95%

```
def cap_upper(df,feature,limit):

    percentile_limit=df[feature].quantile(limit)
    df.loc[df[feature]>=percentile_limit,feature]=percentile_limit
    return df[feature]
```

```
for i in range(1,6):
    feature='BILL_AMT'+str(i)
    df[feature]=cap_upper(df,feature,limit=0.95)
```

```
sns.boxplot(df['BILL_AMT5'])
plt.title('Box Plot of Bill Amount 1')
plt.show()
```

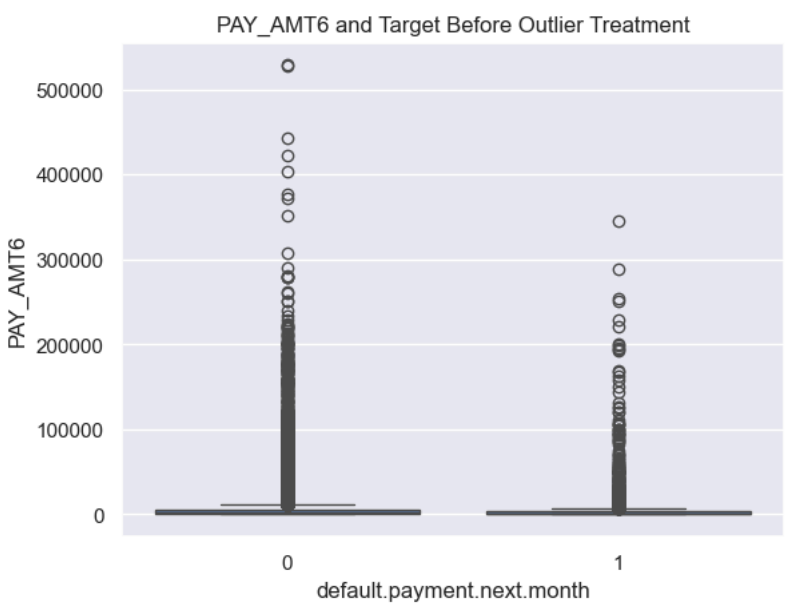


3.PAY_AMT

```
df[['PAY_AMT1','PAY_AMT2','PAY_AMT3','PAY_AMT4','PAY_AMT5','PAY_AMT6']].describe(percentiles=[0.25,0.5,0.75,0.8,0.85,0.9,0.95,0.98])
```

	PAY_AMT1	PAY_AMT2	PAY_AMT3	PAY_AMT4	PAY_AMT5	PAY_AMT6
count	30000.000000	3.000000e+04	30000.00000	30000.000000	30000.000000	30000.000000
mean	5663.580500	5.921163e+03	5225.68150	4826.076867	4799.387633	5215.502567
std	16563.280354	2.304087e+04	17606.96147	15666.159744	15278.305679	17777.465775
min	0.000000	0.000000e+00	0.00000	0.000000	0.000000	0.000000
25%	1000.000000	8.330000e+02	390.00000	296.000000	252.500000	117.750000
50%	2100.000000	2.009000e+03	1800.00000	1500.000000	1500.000000	1500.000000
75%	5006.000000	5.000000e+03	4505.00000	4013.250000	4031.500000	4000.000000
80%	6192.200000	6.000000e+03	5284.00000	5000.000000	5000.000000	5000.000000
85%	8000.000000	7.956150e+03	7000.00000	6300.000000	6271.300000	6200.000000
90%	10300.000000	1.040110e+04	10000.00000	9570.600000	9500.000000	9600.000000
95%	18428.200000	1.900435e+04	17589.40000	16014.950000	16000.000000	17343.800000
98%	40000.000000	4.102788e+04	38661.36000	39634.040000	37259.660000	45010.480000
max	873552.000000	1.684259e+06	896040.00000	621000.000000	426529.000000	528666.000000

```
sns.boxplot(x=df['default.payment.next.month'],y=df['PAY_AMT6'],data=df)
plt.title('PAY_AMT6 and Target Before Outlier Treatment')
plt.show()
```



```
for i in range(1,7):
    feature='PAY_AMT'+str(i)
    df[feature]=cap_upper(df,feature,limit=0.90)


sns.boxplot(x=df[c],y=df['PAY_AMT6'],data=df)
plt.title('PAY_AMT6 and Target After Outlier Treatment')
plt.show()
```