SETUP

* pandas to be able to read data
* matplotlib to plot bar charts and basic histograms
* seaborn to plot box charts easily and then subplots
* sklearn for machine learning implementation

import pandas as pd  
import matplotlib.pyplot as plt   
import seaborn as sns

## DATA INGESTION

df = pd.read\_csv("data/raw/UCI\_Credit\_Card.csv")

## DATA CLEANING

df.head()

|  | ID | LIMIT\_BAL | SEX | EDUCATION | MARRIAGE | AGE | PAY\_0 | PAY\_2 | PAY\_3 | PAY\_4 | ... | BILL\_AMT4 | BILL\_AMT5 | BILL\_AMT6 | PAY\_AMT1 | PAY\_AMT2 | PAY\_AMT3 | PAY\_AMT4 | PAY\_AMT5 | PAY\_AMT6 | default.payment.next.month |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 20000.0 | 2 | 2 | 1 | 24 | 2 | 2 | -1 | -1 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 689.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1 |
| 1 | 2 | 120000.0 | 2 | 2 | 2 | 26 | -1 | 2 | 0 | 0 | ... | 3272.0 | 3455.0 | 3261.0 | 0.0 | 1000.0 | 1000.0 | 1000.0 | 0.0 | 2000.0 | 1 |
| 2 | 3 | 90000.0 | 2 | 2 | 2 | 34 | 0 | 0 | 0 | 0 | ... | 14331.0 | 14948.0 | 15549.0 | 1518.0 | 1500.0 | 1000.0 | 1000.0 | 1000.0 | 5000.0 | 0 |
| 3 | 4 | 50000.0 | 2 | 2 | 1 | 37 | 0 | 0 | 0 | 0 | ... | 28314.0 | 28959.0 | 29547.0 | 2000.0 | 2019.0 | 1200.0 | 1100.0 | 1069.0 | 1000.0 | 0 |
| 4 | 5 | 50000.0 | 1 | 2 | 1 | 57 | -1 | 0 | -1 | 0 | ... | 20940.0 | 19146.0 | 19131.0 | 2000.0 | 36681.0 | 10000.0 | 9000.0 | 689.0 | 679.0 | 0 |

df.tail()

|  | ID | LIMIT\_BAL | SEX | EDUCATION | MARRIAGE | AGE | PAY\_0 | PAY\_2 | PAY\_3 | PAY\_4 | ... | BILL\_AMT4 | BILL\_AMT5 | BILL\_AMT6 | PAY\_AMT1 | PAY\_AMT2 | PAY\_AMT3 | PAY\_AMT4 | PAY\_AMT5 | PAY\_AMT6 | default.payment.next.month |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 29995 | 29996 | 220000.0 | 1 | 3 | 1 | 39 | 0 | 0 | 0 | 0 | ... | 88004.0 | 31237.0 | 15980.0 | 8500.0 | 20000.0 | 5003.0 | 3047.0 | 5000.0 | 1000.0 | 0 |
| 29996 | 29997 | 150000.0 | 1 | 3 | 2 | 43 | -1 | -1 | -1 | -1 | ... | 8979.0 | 5190.0 | 0.0 | 1837.0 | 3526.0 | 8998.0 | 129.0 | 0.0 | 0.0 | 0 |
| 29997 | 29998 | 30000.0 | 1 | 2 | 2 | 37 | 4 | 3 | 2 | -1 | ... | 20878.0 | 20582.0 | 19357.0 | 0.0 | 0.0 | 22000.0 | 4200.0 | 2000.0 | 3100.0 | 1 |
| 29998 | 29999 | 80000.0 | 1 | 3 | 1 | 41 | 1 | -1 | 0 | 0 | ... | 52774.0 | 11855.0 | 48944.0 | 85900.0 | 3409.0 | 1178.0 | 1926.0 | 52964.0 | 1804.0 | 1 |
| 29999 | 30000 | 50000.0 | 1 | 2 | 1 | 46 | 0 | 0 | 0 | 0 | ... | 36535.0 | 32428.0 | 15313.0 | 2078.0 | 1800.0 | 1430.0 | 1000.0 | 1000.0 | 1000.0 | 1 |

df.info()

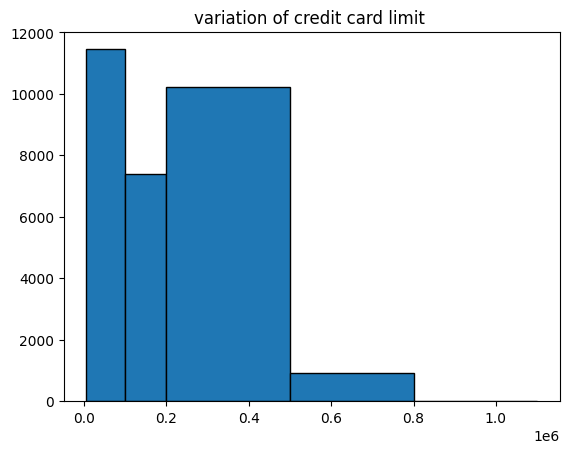
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 30000 entries, 0 to 29999  
Data columns (total 25 columns):  
 # Column Non-Null Count Dtype   
--- ------ -------------- -----   
 0 ID 30000 non-null int64   
 1 LIMIT\_BAL 30000 non-null float64  
 2 SEX 30000 non-null int64   
 3 EDUCATION 30000 non-null int64   
 4 MARRIAGE 30000 non-null int64   
 5 AGE 30000 non-null int64   
 6 PAY\_0 30000 non-null int64   
 7 PAY\_2 30000 non-null int64   
 8 PAY\_3 30000 non-null int64   
 9 PAY\_4 30000 non-null int64   
 10 PAY\_5 30000 non-null int64   
 11 PAY\_6 30000 non-null int64   
 12 BILL\_AMT1 30000 non-null float64  
 13 BILL\_AMT2 30000 non-null float64  
 14 BILL\_AMT3 30000 non-null float64  
 15 BILL\_AMT4 30000 non-null float64  
 16 BILL\_AMT5 30000 non-null float64  
 17 BILL\_AMT6 30000 non-null float64  
 18 PAY\_AMT1 30000 non-null float64  
 19 PAY\_AMT2 30000 non-null float64  
 20 PAY\_AMT3 30000 non-null float64  
 21 PAY\_AMT4 30000 non-null float64  
 22 PAY\_AMT5 30000 non-null float64  
 23 PAY\_AMT6 30000 non-null float64  
 24 default.payment.next.month 30000 non-null int64   
dtypes: float64(13), int64(12)  
memory usage: 5.7 MB

df.describe()

|  | ID | LIMIT\_BAL | SEX | EDUCATION | MARRIAGE | AGE | PAY\_0 | PAY\_2 | PAY\_3 | PAY\_4 | ... | BILL\_AMT4 | BILL\_AMT5 | BILL\_AMT6 | PAY\_AMT1 | PAY\_AMT2 | PAY\_AMT3 | PAY\_AMT4 | PAY\_AMT5 | PAY\_AMT6 | default.payment.next.month |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| count | 30000.000000 | 30000.000000 | 30000.000000 | 30000.000000 | 30000.000000 | 30000.000000 | 30000.000000 | 30000.000000 | 30000.000000 | 30000.000000 | ... | 30000.000000 | 30000.000000 | 30000.000000 | 30000.000000 | 3.000000e+04 | 30000.00000 | 30000.000000 | 30000.000000 | 30000.000000 | 30000.000000 |
| mean | 15000.500000 | 167484.322667 | 1.603733 | 1.853133 | 1.551867 | 35.485500 | -0.016700 | -0.133767 | -0.166200 | -0.220667 | ... | 43262.948967 | 40311.400967 | 38871.760400 | 5663.580500 | 5.921163e+03 | 5225.68150 | 4826.076867 | 4799.387633 | 5215.502567 | 0.221200 |
| std | 8660.398374 | 129747.661567 | 0.489129 | 0.790349 | 0.521970 | 9.217904 | 1.123802 | 1.197186 | 1.196868 | 1.169139 | ... | 64332.856134 | 60797.155770 | 59554.107537 | 16563.280354 | 2.304087e+04 | 17606.96147 | 15666.159744 | 15278.305679 | 17777.465775 | 0.415062 |
| min | 1.000000 | 10000.000000 | 1.000000 | 0.000000 | 0.000000 | 21.000000 | -2.000000 | -2.000000 | -2.000000 | -2.000000 | ... | -170000.000000 | -81334.000000 | -339603.000000 | 0.000000 | 0.000000e+00 | 0.00000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| 25% | 7500.750000 | 50000.000000 | 1.000000 | 1.000000 | 1.000000 | 28.000000 | -1.000000 | -1.000000 | -1.000000 | -1.000000 | ... | 2326.750000 | 1763.000000 | 1256.000000 | 1000.000000 | 8.330000e+02 | 390.00000 | 296.000000 | 252.500000 | 117.750000 | 0.000000 |
| 50% | 15000.500000 | 140000.000000 | 2.000000 | 2.000000 | 2.000000 | 34.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | ... | 19052.000000 | 18104.500000 | 17071.000000 | 2100.000000 | 2.009000e+03 | 1800.00000 | 1500.000000 | 1500.000000 | 1500.000000 | 0.000000 |
| 75% | 22500.250000 | 240000.000000 | 2.000000 | 2.000000 | 2.000000 | 41.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | ... | 54506.000000 | 50190.500000 | 49198.250000 | 5006.000000 | 5.000000e+03 | 4505.00000 | 4013.250000 | 4031.500000 | 4000.000000 | 0.000000 |
| max | 30000.000000 | 1000000.000000 | 2.000000 | 6.000000 | 3.000000 | 79.000000 | 8.000000 | 8.000000 | 8.000000 | 8.000000 | ... | 891586.000000 | 927171.000000 | 961664.000000 | 873552.000000 | 1.684259e+06 | 896040.00000 | 621000.000000 | 426529.000000 | 528666.000000 | 1.000000 |

* Checking variation of credit card limit

values = [df.LIMIT\_BAL]  
bins = [5000, 100000, 200000, 500000, 800000, 1100000]  
plt.hist(values, bins, edgecolor = 'black')  
plt.title("variation of credit card limit ")  
plt.show()



### checking unique values of cols

exclude\_columns = ['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']  
  
for column in df.columns:  
 if column not in exclude\_columns:  
 unique\_values = df[column].unique()  
 print(f"Unique values in {column}: {unique\_values}")

Unique values in ID: [ 1 2 3 ... 29998 29999 30000]  
Unique values in SEX: [2 1]  
Unique values in EDUCATION: [2 1 3 5 4 6 0]  
Unique values in MARRIAGE: [1 2 3 0]  
Unique values in PAY\_0: [ 2 -1 0 -2 1 3 4 8 7 5 6]  
Unique values in PAY\_2: [ 2 0 -1 -2 3 5 7 4 1 6 8]  
Unique values in PAY\_3: [-1 0 2 -2 3 4 6 7 1 5 8]  
Unique values in PAY\_4: [-1 0 -2 2 3 4 5 7 6 1 8]  
Unique values in PAY\_5: [-2 0 -1 2 3 5 4 7 8 6]  
Unique values in PAY\_6: [-2 2 0 -1 3 6 4 7 8 5]  
Unique values in default.payment.next.month: [1 0]

### checking for null

df.isnull().sum()

ID 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

### replacing the numeric entry values with actual

* changing sex given as 1 to M , 2 to F
* have to change it to string type for it to accept this cmd

change = {'1':'M', '2':'F'}  
df['SEX'] = df['SEX'].astype(str).replace(change)

* since 6 and 5 both are unknowns, changing 6 to 5

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

* nothing is given about 0 hence changed it to 3

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

* changed [1 to married] [2 to single] [3 to others]

df['MARRIAGE'] = df['MARRIAGE'].astype(str).replace({'1':'married', '2':'single', '3':'others'})

### Renaming the Attributes

* PAY\_(x) –> PAY\_(month)

df.rename(columns={'PAY\_0':'REPAY\_SEPT','PAY\_2':'REPAY\_AUG', 'PAY\_3':'REPAY\_JUL', 'PAY\_4':'REPAY\_JUN', 'PAY\_5':'REPAY\_MAY', 'PAY\_6':'REPAY\_APR' }, inplace=True)  
df

|  | ID | LIMIT\_BAL | SEX | EDUCATION | MARRIAGE | AGE | REPAY\_SEPT | REPAY\_AUG | REPAY\_JUL | REPAY\_JUN | ... | BILL\_AMT4 | BILL\_AMT5 | BILL\_AMT6 | PAY\_AMT1 | PAY\_AMT2 | PAY\_AMT3 | PAY\_AMT4 | PAY\_AMT5 | PAY\_AMT6 | default.payment.next.month |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 20000.0 | F | 2 | married | 24 | 2 | 2 | -1 | -1 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 689.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1 |
| 1 | 2 | 120000.0 | F | 2 | single | 26 | -1 | 2 | 0 | 0 | ... | 3272.0 | 3455.0 | 3261.0 | 0.0 | 1000.0 | 1000.0 | 1000.0 | 0.0 | 2000.0 | 1 |
| 2 | 3 | 90000.0 | F | 2 | single | 34 | 0 | 0 | 0 | 0 | ... | 14331.0 | 14948.0 | 15549.0 | 1518.0 | 1500.0 | 1000.0 | 1000.0 | 1000.0 | 5000.0 | 0 |
| 3 | 4 | 50000.0 | F | 2 | married | 37 | 0 | 0 | 0 | 0 | ... | 28314.0 | 28959.0 | 29547.0 | 2000.0 | 2019.0 | 1200.0 | 1100.0 | 1069.0 | 1000.0 | 0 |
| 4 | 5 | 50000.0 | M | 2 | married | 57 | -1 | 0 | -1 | 0 | ... | 20940.0 | 19146.0 | 19131.0 | 2000.0 | 36681.0 | 10000.0 | 9000.0 | 689.0 | 679.0 | 0 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 29995 | 29996 | 220000.0 | M | 3 | married | 39 | 0 | 0 | 0 | 0 | ... | 88004.0 | 31237.0 | 15980.0 | 8500.0 | 20000.0 | 5003.0 | 3047.0 | 5000.0 | 1000.0 | 0 |
| 29996 | 29997 | 150000.0 | M | 3 | single | 43 | -1 | -1 | -1 | -1 | ... | 8979.0 | 5190.0 | 0.0 | 1837.0 | 3526.0 | 8998.0 | 129.0 | 0.0 | 0.0 | 0 |
| 29997 | 29998 | 30000.0 | M | 2 | single | 37 | 4 | 3 | 2 | -1 | ... | 20878.0 | 20582.0 | 19357.0 | 0.0 | 0.0 | 22000.0 | 4200.0 | 2000.0 | 3100.0 | 1 |
| 29998 | 29999 | 80000.0 | M | 3 | married | 41 | 1 | -1 | 0 | 0 | ... | 52774.0 | 11855.0 | 48944.0 | 85900.0 | 3409.0 | 1178.0 | 1926.0 | 52964.0 | 1804.0 | 1 |
| 29999 | 30000 | 50000.0 | M | 2 | married | 46 | 0 | 0 | 0 | 0 | ... | 36535.0 | 32428.0 | 15313.0 | 2078.0 | 1800.0 | 1430.0 | 1000.0 | 1000.0 | 1000.0 | 1 |

* BILL\_AMT(x) –> BILL\_(month)

df.rename(columns={'BILL\_AMT1':'BILL\_SEPT', 'BILL\_AMT2':'BILL\_AUG', 'BILL\_AMT3':'BILL\_JUL', 'BILL\_AMT4':'BILL\_JUN', 'BILL\_AMT5':'BILL\_MAY', 'BILL\_AMT6':'BILL\_APR'}, inplace=True)  
df

|  | ID | LIMIT\_BAL | SEX | EDUCATION | MARRIAGE | AGE | REPAY\_SEPT | REPAY\_AUG | REPAY\_JUL | REPAY\_JUN | ... | BILL\_JUN | BILL\_MAY | BILL\_APR | PAY\_AMT1 | PAY\_AMT2 | PAY\_AMT3 | PAY\_AMT4 | PAY\_AMT5 | PAY\_AMT6 | default.payment.next.month |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 20000.0 | F | 2 | married | 24 | 2 | 2 | -1 | -1 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 689.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1 |
| 1 | 2 | 120000.0 | F | 2 | single | 26 | -1 | 2 | 0 | 0 | ... | 3272.0 | 3455.0 | 3261.0 | 0.0 | 1000.0 | 1000.0 | 1000.0 | 0.0 | 2000.0 | 1 |
| 2 | 3 | 90000.0 | F | 2 | single | 34 | 0 | 0 | 0 | 0 | ... | 14331.0 | 14948.0 | 15549.0 | 1518.0 | 1500.0 | 1000.0 | 1000.0 | 1000.0 | 5000.0 | 0 |
| 3 | 4 | 50000.0 | F | 2 | married | 37 | 0 | 0 | 0 | 0 | ... | 28314.0 | 28959.0 | 29547.0 | 2000.0 | 2019.0 | 1200.0 | 1100.0 | 1069.0 | 1000.0 | 0 |
| 4 | 5 | 50000.0 | M | 2 | married | 57 | -1 | 0 | -1 | 0 | ... | 20940.0 | 19146.0 | 19131.0 | 2000.0 | 36681.0 | 10000.0 | 9000.0 | 689.0 | 679.0 | 0 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 29995 | 29996 | 220000.0 | M | 3 | married | 39 | 0 | 0 | 0 | 0 | ... | 88004.0 | 31237.0 | 15980.0 | 8500.0 | 20000.0 | 5003.0 | 3047.0 | 5000.0 | 1000.0 | 0 |
| 29996 | 29997 | 150000.0 | M | 3 | single | 43 | -1 | -1 | -1 | -1 | ... | 8979.0 | 5190.0 | 0.0 | 1837.0 | 3526.0 | 8998.0 | 129.0 | 0.0 | 0.0 | 0 |
| 29997 | 29998 | 30000.0 | M | 2 | single | 37 | 4 | 3 | 2 | -1 | ... | 20878.0 | 20582.0 | 19357.0 | 0.0 | 0.0 | 22000.0 | 4200.0 | 2000.0 | 3100.0 | 1 |
| 29998 | 29999 | 80000.0 | M | 3 | married | 41 | 1 | -1 | 0 | 0 | ... | 52774.0 | 11855.0 | 48944.0 | 85900.0 | 3409.0 | 1178.0 | 1926.0 | 52964.0 | 1804.0 | 1 |
| 29999 | 30000 | 50000.0 | M | 2 | married | 46 | 0 | 0 | 0 | 0 | ... | 36535.0 | 32428.0 | 15313.0 | 2078.0 | 1800.0 | 1430.0 | 1000.0 | 1000.0 | 1000.0 | 1 |

* PAY\_AMT(x) –> PAID\_(month)

df.rename(columns={'PAY\_AMT1':'PAID\_SEPT', 'PAY\_AMT2':'PAID\_AUG', 'PAY\_AMT3':'PAID\_JUL', 'PAY\_AMT4':'PAID\_JUN', 'PAY\_AMT5':'PAID\_MAY', 'PAY\_AMT6':'PAID\_APR'}, inplace=True)  
df

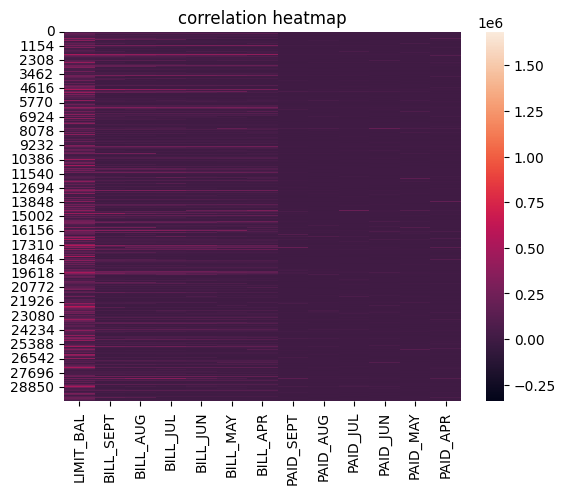
|  | ID | LIMIT\_BAL | SEX | EDUCATION | MARRIAGE | AGE | REPAY\_SEPT | REPAY\_AUG | REPAY\_JUL | REPAY\_JUN | ... | BILL\_JUN | BILL\_MAY | BILL\_APR | PAID\_SEPT | PAID\_AUG | PAID\_JUL | PAID\_JUN | PAID\_MAY | PAID\_APR | default.payment.next.month |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 20000.0 | F | 2 | married | 24 | 2 | 2 | -1 | -1 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 689.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1 |
| 1 | 2 | 120000.0 | F | 2 | single | 26 | -1 | 2 | 0 | 0 | ... | 3272.0 | 3455.0 | 3261.0 | 0.0 | 1000.0 | 1000.0 | 1000.0 | 0.0 | 2000.0 | 1 |
| 2 | 3 | 90000.0 | F | 2 | single | 34 | 0 | 0 | 0 | 0 | ... | 14331.0 | 14948.0 | 15549.0 | 1518.0 | 1500.0 | 1000.0 | 1000.0 | 1000.0 | 5000.0 | 0 |
| 3 | 4 | 50000.0 | F | 2 | married | 37 | 0 | 0 | 0 | 0 | ... | 28314.0 | 28959.0 | 29547.0 | 2000.0 | 2019.0 | 1200.0 | 1100.0 | 1069.0 | 1000.0 | 0 |
| 4 | 5 | 50000.0 | M | 2 | married | 57 | -1 | 0 | -1 | 0 | ... | 20940.0 | 19146.0 | 19131.0 | 2000.0 | 36681.0 | 10000.0 | 9000.0 | 689.0 | 679.0 | 0 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 29995 | 29996 | 220000.0 | M | 3 | married | 39 | 0 | 0 | 0 | 0 | ... | 88004.0 | 31237.0 | 15980.0 | 8500.0 | 20000.0 | 5003.0 | 3047.0 | 5000.0 | 1000.0 | 0 |
| 29996 | 29997 | 150000.0 | M | 3 | single | 43 | -1 | -1 | -1 | -1 | ... | 8979.0 | 5190.0 | 0.0 | 1837.0 | 3526.0 | 8998.0 | 129.0 | 0.0 | 0.0 | 0 |
| 29997 | 29998 | 30000.0 | M | 2 | single | 37 | 4 | 3 | 2 | -1 | ... | 20878.0 | 20582.0 | 19357.0 | 0.0 | 0.0 | 22000.0 | 4200.0 | 2000.0 | 3100.0 | 1 |
| 29998 | 29999 | 80000.0 | M | 3 | married | 41 | 1 | -1 | 0 | 0 | ... | 52774.0 | 11855.0 | 48944.0 | 85900.0 | 3409.0 | 1178.0 | 1926.0 | 52964.0 | 1804.0 | 1 |
| 29999 | 30000 | 50000.0 | M | 2 | married | 46 | 0 | 0 | 0 | 0 | ... | 36535.0 | 32428.0 | 15313.0 | 2078.0 | 1800.0 | 1430.0 | 1000.0 | 1000.0 | 1000.0 | 1 |

### finding correlation

* no inference from correlation

selected\_float\_columns = df.select\_dtypes(include='float')  
correlation\_matrix = selected\_float\_columns.corr()  
sns.heatmap(selected\_float\_columns)  
plt.title("correlation heatmap")

Text(0.5, 1.0, 'correlation heatmap')



## FEATURE ENGINEERING

### sum of repay\_(month)

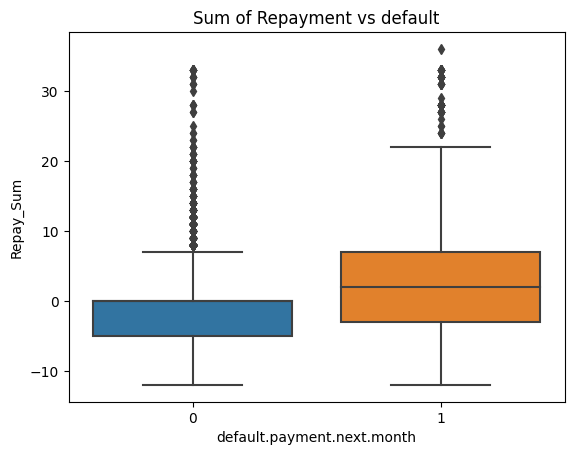
* found rows with highest sum, ie, rows with highest default

cols\_sum = ['REPAY\_SEPT', 'REPAY\_AUG', 'REPAY\_JUL', 'REPAY\_JUN', 'REPAY\_MAY', 'REPAY\_APR']  
df['Repay\_Sum'] = df[cols\_sum].sum(axis=1)

* boxplot to check where the median of sums lie

sns.boxplot(x = df['default.payment.next.month'], y = df.Repay\_Sum)  
plt.title("Sum of Repayment vs default")

Text(0.5, 1.0, 'Sum of Repayment vs default')



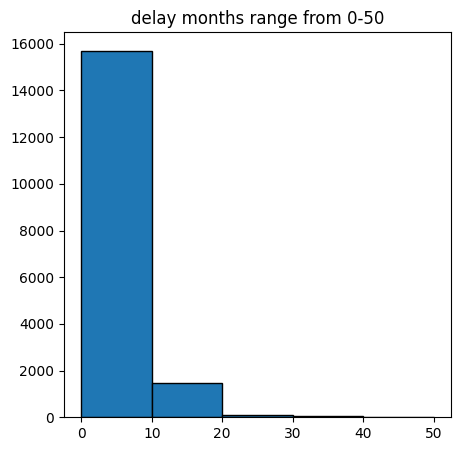
* representation of people default groups

minNo = 30  
filter\_df = df[df['Repay\_Sum']>=minNo]   
print("number of people with more than 30 months of default:", filter\_df.shape[0] )  
  
minNo2 = 20  
filter\_df1 = df[(df['Repay\_Sum']>=minNo2)&(df['Repay\_Sum']<minNo)]   
print("number of people with more than 20 months but less than 30 months of default:", filter\_df1.shape[0])   
  
minNo3 = 10  
filter\_df2 = df[(df['Repay\_Sum']>=minNo3)&(df['Repay\_Sum']<minNo2)]   
print("number of people with more than 10 months but less than 20 months of default:", filter\_df2.shape[0] )  
  
minNo4 = 0  
filter\_df3 = df[(df['Repay\_Sum']>=minNo4)&(df['Repay\_Sum']<minNo3)]   
print("number of people with more than 0 months but less than 10 months of default:", filter\_df3.shape[0])

number of people with more than 30 months of default: 50  
number of people with more than 20 months but less than 30 months of default: 97  
number of people with more than 10 months but less than 20 months of default: 1470  
number of people with more than 0 months but less than 10 months of default: 15708

* graphical representation of people default groups

plt.figure(figsize=(5,5))  
bins = [0,10,20,30,40,50]  
plt.hist(df.Repay\_Sum, bins, edgecolor = 'black')  
plt.title('delay months range from 0-50')  
plt.show()



### mean of repay\_(month)

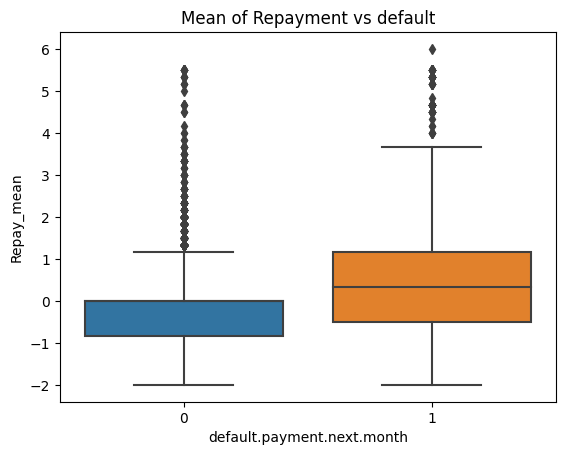
* found highest mean of deafult

cols\_sum = ['REPAY\_SEPT', 'REPAY\_AUG', 'REPAY\_JUL', 'REPAY\_JUN', 'REPAY\_MAY', 'REPAY\_APR']  
df['Repay\_mean'] = df[cols\_sum].mean(axis=1)

* boxplot to check where the median of means lie
* it shows that the median of sums and means is similar

sns.boxplot(x = df['default.payment.next.month'], y = df.Repay\_mean)  
plt.title("Mean of Repayment vs default")

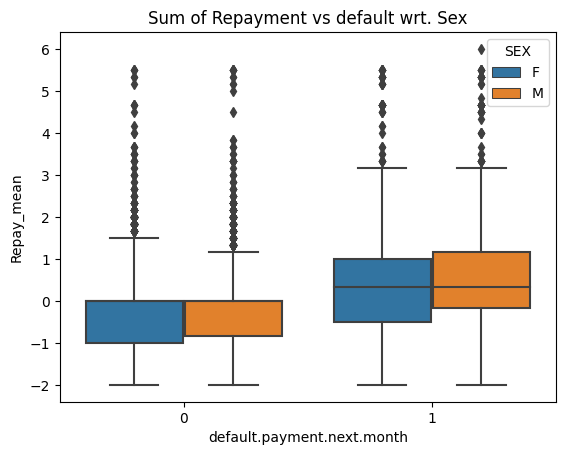
Text(0.5, 1.0, 'Mean of Repayment vs default')



* checking median of deafults with respect to sex and repayment means
* median is similar can’t use

sns.boxplot(x = df['default.payment.next.month'], y = df.Repay\_mean, hue= df.SEX)  
plt.title("Sum of Repayment vs default wrt. Sex")

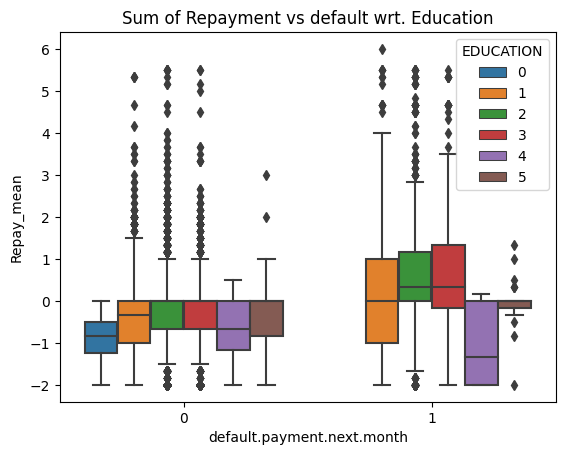
Text(0.5, 1.0, 'Sum of Repayment vs default wrt. Sex')



* checking median of deafults with respect to education and repayment means
* median for university and high school is higher

sns.boxplot(x = df['default.payment.next.month'], y = df.Repay\_mean, hue= df.EDUCATION)  
plt.title("Sum of Repayment vs default wrt. Education")

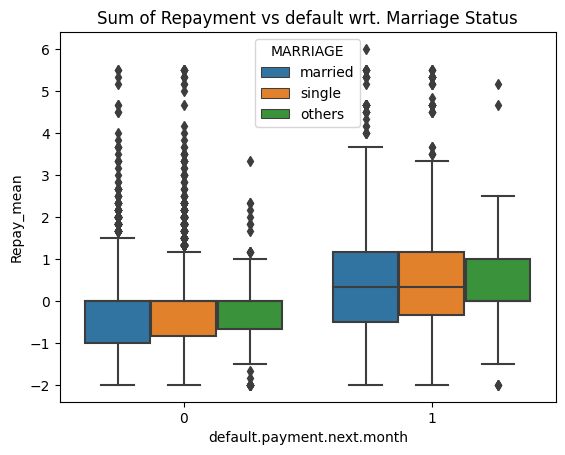
Text(0.5, 1.0, 'Sum of Repayment vs default wrt. Education')



* checking median of deafults with respect to marriage and repayment means
* median is similar so can’t use

sns.boxplot(x = df['default.payment.next.month'], y = df.Repay\_mean, hue= df.MARRIAGE)  
plt.title("Sum of Repayment vs default wrt. Marriage Status")

Text(0.5, 1.0, 'Sum of Repayment vs default wrt. Marriage Status')



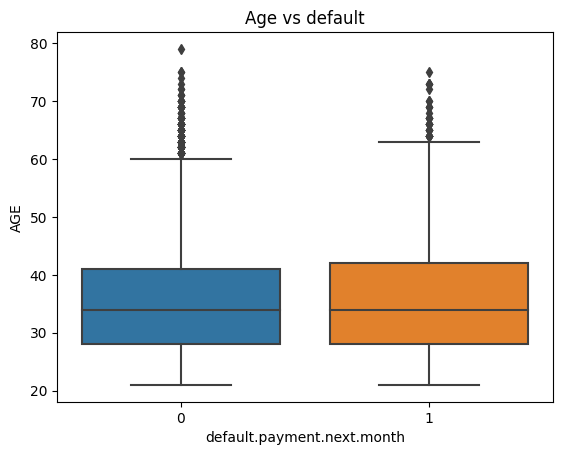
## Prob of default with demographics

### Default based on age

* median is similar so can’t use

sns.boxplot(data=df, x = df['default.payment.next.month'], y = df['AGE'], orient='v')  
plt.title("Age vs default")

Text(0.5, 1.0, 'Age vs default')

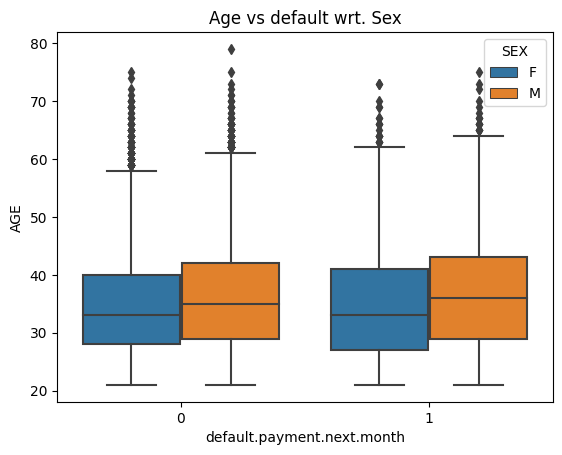


### default based on age and sex

* male defaulters have higher median

sns.boxplot(data=df, x = df['default.payment.next.month'], y = df['AGE'], orient='v', hue = df['SEX'])  
plt.title("Age vs default wrt. Sex")

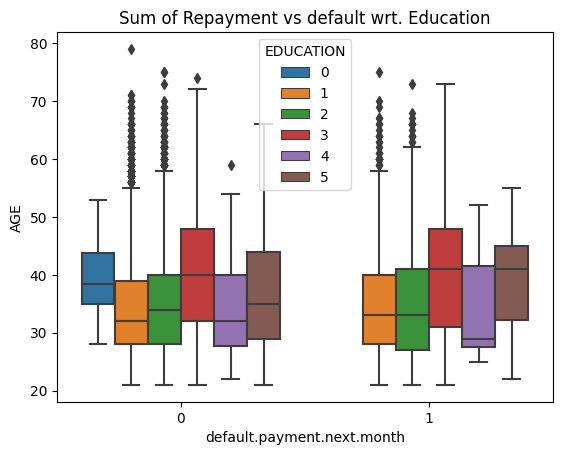
Text(0.5, 1.0, 'Age vs default wrt. Sex')



* median for high school is highest, therefore age of highschool is highest

sns.boxplot(data=df, x = df['default.payment.next.month'], y = df['AGE'], orient='v', hue = df['EDUCATION'])  
plt.title("Sum of Repayment vs default wrt. Education")

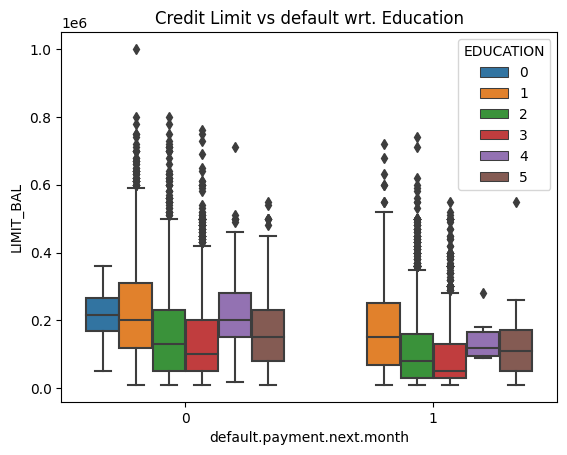
Text(0.5, 1.0, 'Sum of Repayment vs default wrt. Education')



* median credit limit of highschool is minimum

sns.boxplot(data=df, x = df['default.payment.next.month'], y = df['LIMIT\_BAL'], orient='v', hue = df['EDUCATION'])  
plt.title("Credit Limit vs default wrt. Education")

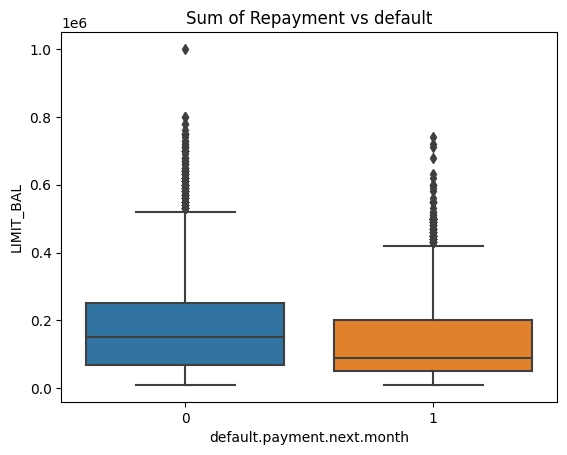
Text(0.5, 1.0, 'Credit Limit vs default wrt. Education')



## credit limit vs default

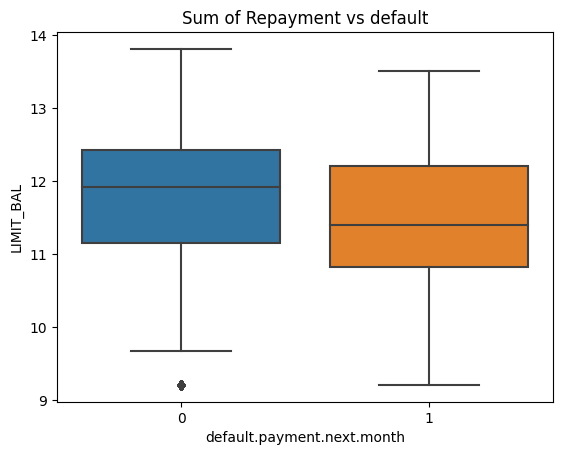
sns.boxplot(x = df['default.payment.next.month'], y = df.LIMIT\_BAL)  
plt.title("Sum of Repayment vs default")

Text(0.5, 1.0, 'Sum of Repayment vs default')



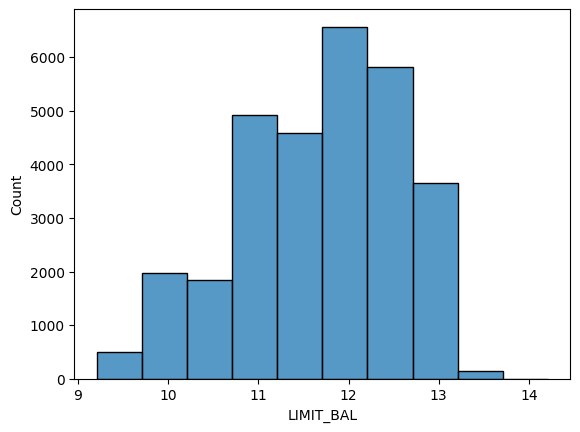
import numpy as np  
sns.boxplot(x = df['default.payment.next.month'], y = np.log1p(df['LIMIT\_BAL']))  
plt.title("Sum of Repayment vs default")

Text(0.5, 1.0, 'Sum of Repayment vs default')



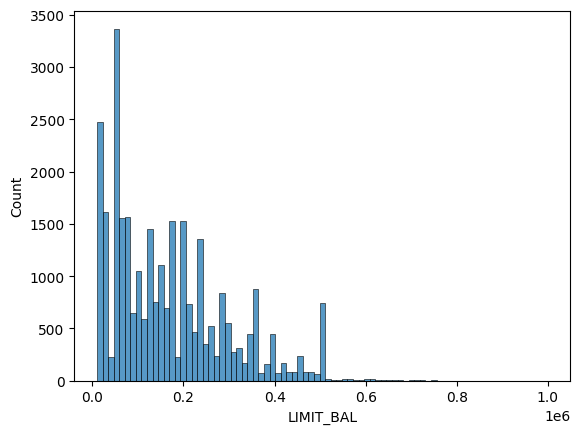
sns.histplot(x=np.log1p(df['LIMIT\_BAL']), binwidth=0.5)

<Axes: xlabel='LIMIT\_BAL', ylabel='Count'>



sns.histplot(x=df.LIMIT\_BAL)

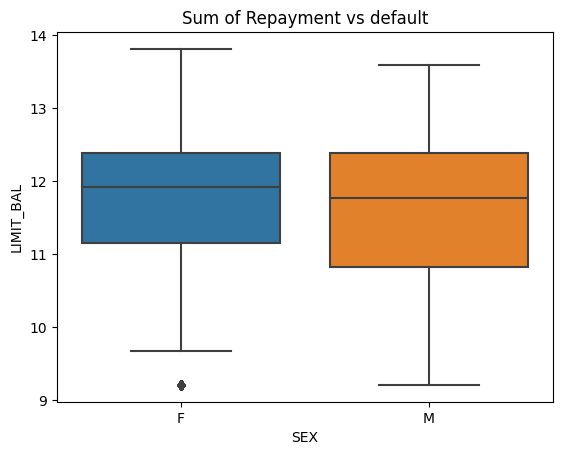
<Axes: xlabel='LIMIT\_BAL', ylabel='Count'>



### Credit limit vs sex

import numpy as np  
sns.boxplot(x = df['SEX'], y = np.log1p(df['LIMIT\_BAL']))  
plt.title("Sum of Repayment vs default")

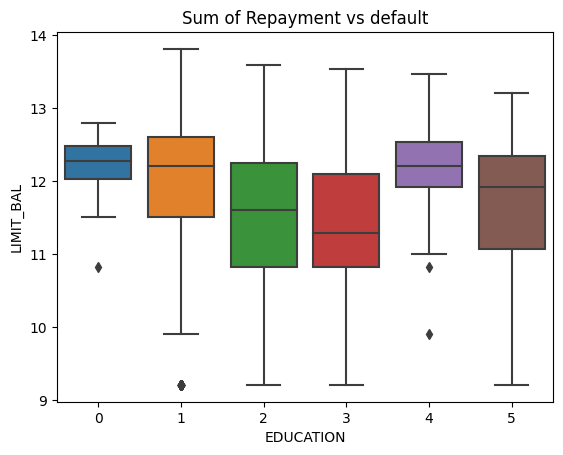
Text(0.5, 1.0, 'Sum of Repayment vs default')



### cREDIT limit vs education

import numpy as np  
sns.boxplot(x = df['EDUCATION'], y = np.log1p(df['LIMIT\_BAL']))  
plt.title("Sum of Repayment vs default")

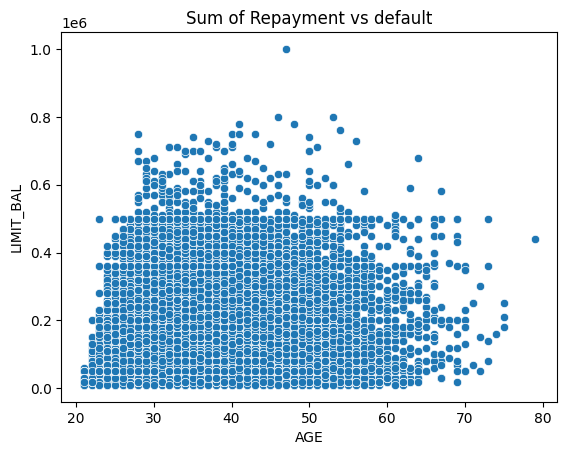
Text(0.5, 1.0, 'Sum of Repayment vs default')



### Credit limit vs age

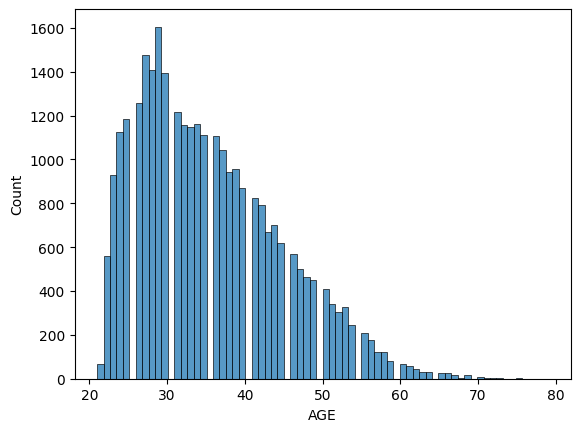
import numpy as np  
sns.scatterplot(x = df['AGE'], y = (df['LIMIT\_BAL']))  
plt.title("Sum of Repayment vs default")

Text(0.5, 1.0, 'Sum of Repayment vs default')



sns.histplot(x=df.AGE)

<Axes: xlabel='AGE', ylabel='Count'>

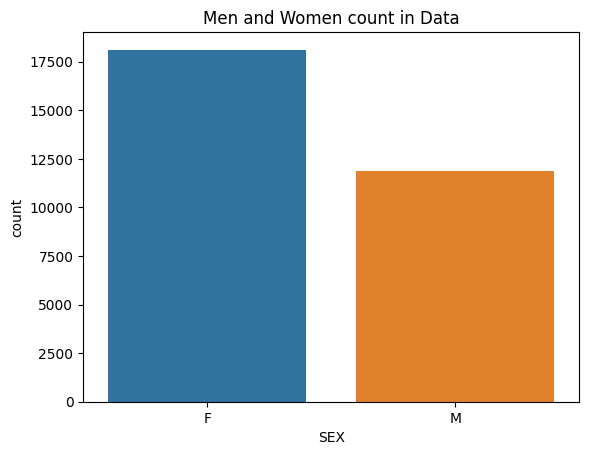


## default based on sex

* number of women in data is more than men

sns.countplot(data=df, x= df['SEX'])  
plt.title("Men and Women count in Data")

Text(0.5, 1.0, 'Men and Women count in Data')



### women

* finding total number of women defaulted
* finding percentage of women defaulting among total women

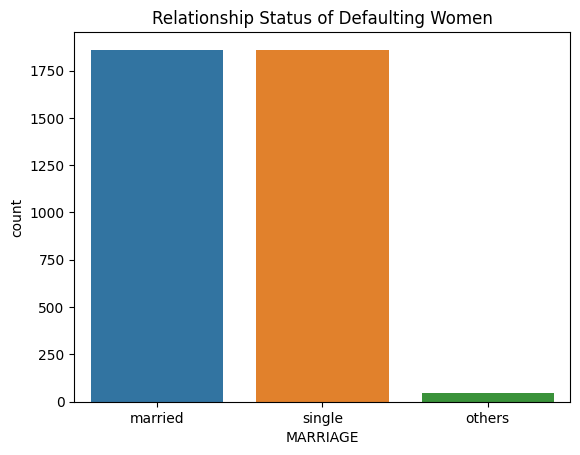
women = df[(df['default.payment.next.month'] == 1) & (df.SEX=='F')]  
print("total no of women defaulters:", women.shape[0])  
  
percent\_F\_def = (women.shape[0]/df[df['SEX']=='F'].shape[0])\*100  
print("percentage of women defaulters:", percent\_F\_def)

total no of women defaulters: 3763  
percentage of women defaulters: 20.776280918727917

* graphical representation of relationship status of defaulting women

sns.countplot(x= women['MARRIAGE'])  
plt.title("Relationship Status of Defaulting Women")

Text(0.5, 1.0, 'Relationship Status of Defaulting Women')



#### married

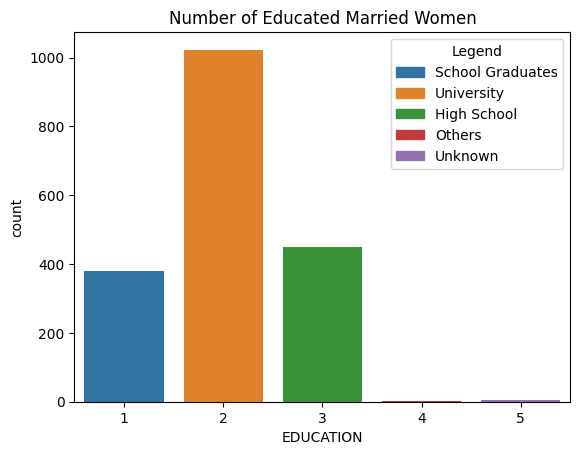
* no of married women among total women defaluters

mwomen = women[women.MARRIAGE=='married']  
mwomen.shape[0]

1860

mwomen\_graph = sns.countplot(x= mwomen['EDUCATION'])  
  
legend\_labels = ['School Graduates', 'University', 'High School', 'Others', 'Unknown']  
  
handles = [plt.Rectangle((0,0),1,1, color=mwomen\_graph.patches[i].get\_facecolor()) for i in range(len(legend\_labels))]  
plt.legend(handles, legend\_labels, title='Legend')  
  
  
plt.title("Number of Educated Married Women")

Text(0.5, 1.0, 'Number of Educated Married Women')



#### single

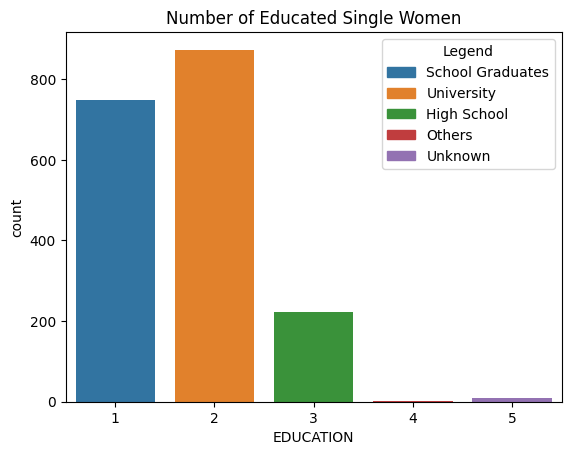
* total single women defaulters

swomen = women[women.MARRIAGE=='single']  
swomen.shape[0]

1856

swomen\_graph = sns.countplot(x= swomen['EDUCATION'])  
  
legend\_labels = ['School Graduates', 'University', 'High School', 'Others', 'Unknown']  
  
handles = [plt.Rectangle((0,0),1,1, color=swomen\_graph.patches[i].get\_facecolor()) for i in range(len(legend\_labels))]  
plt.legend(handles, legend\_labels, title='Legend')  
  
  
plt.title("Number of Educated Single Women")

Text(0.5, 1.0, 'Number of Educated Single Women')



### men

* finding total number of men defaulted
* finding percentage of men defaulting among total men

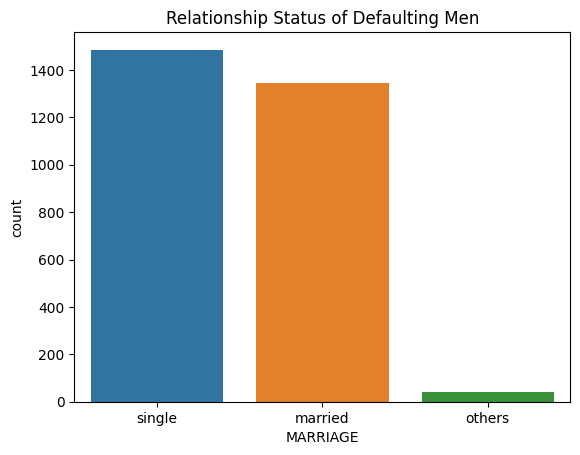
men = df[(df['default.payment.next.month'] == 1) & (df.SEX=='M')]  
print("total no of men defaulters:", men.shape[0])  
  
percent\_M\_def = (men.shape[0]/df[df['SEX']=='M'].shape[0])\*100  
print("percentage of men defaulters:", percent\_M\_def)

total no of men defaulters: 2873  
percentage of men defaulters: 24.16722745625841

* graphical representation of relationship status of defaulting men

sns.countplot(x= men['MARRIAGE'])  
plt.title("Relationship Status of Defaulting Men")

Text(0.5, 1.0, 'Relationship Status of Defaulting Men')



#### married

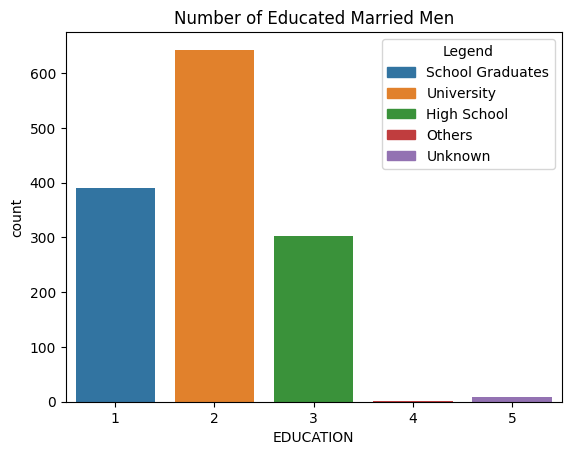
* no of married men among total women defaluters

mmen = men[men.MARRIAGE=='married']  
men.shape[0]

2873

mmen\_graph = sns.countplot(x= mmen['EDUCATION'])  
  
legend\_labels = ['School Graduates', 'University', 'High School', 'Others', 'Unknown']  
  
handles = [plt.Rectangle((0,0),1,1, color=mmen\_graph.patches[i].get\_facecolor()) for i in range(len(legend\_labels))]  
plt.legend(handles, legend\_labels, title='Legend')  
  
  
plt.title("Number of Educated Married Men")

Text(0.5, 1.0, 'Number of Educated Married Men')



#### single

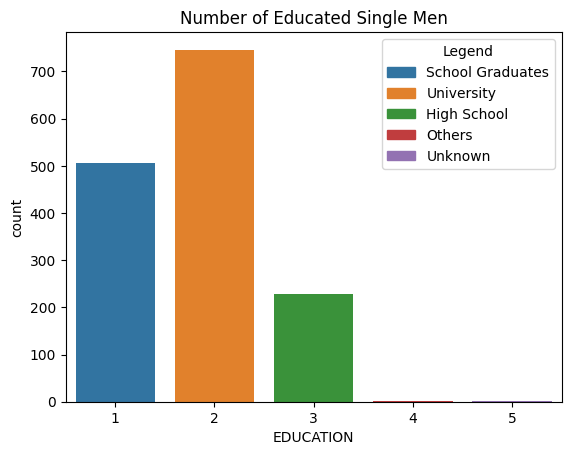
* total single men defaulters

smen = men[men.MARRIAGE=='single']  
smen.shape[0]

1485

smen\_graph = sns.countplot(x= smen['EDUCATION'])  
  
legend\_labels = ['School Graduates', 'University', 'High School', 'Others', 'Unknown']  
  
handles = [plt.Rectangle((0,0),1,1, color=smen\_graph.patches[i].get\_facecolor()) for i in range(len(legend\_labels))]  
plt.legend(handles, legend\_labels, title='Legend')  
  
  
plt.title("Number of Educated Single Men")

Text(0.5, 1.0, 'Number of Educated Single Men')



* combining the graphs

## default based on education

* university defaulters

edu\_uni = df[(df.EDUCATION== 2)]   
print("total university data:", edu\_uni.shape[0])   
  
total\_uni\_def = df[(df.EDUCATION==2)&(df['default.payment.next.month']==1)&(df.MARRIAGE != 'others')].shape[0]  
print("total university defaulters:", total\_uni\_def)  
percentage2 = (total\_uni\_def/edu\_uni.shape[0])\*100  
print("percentage of uni defaulters: ",percentage2)

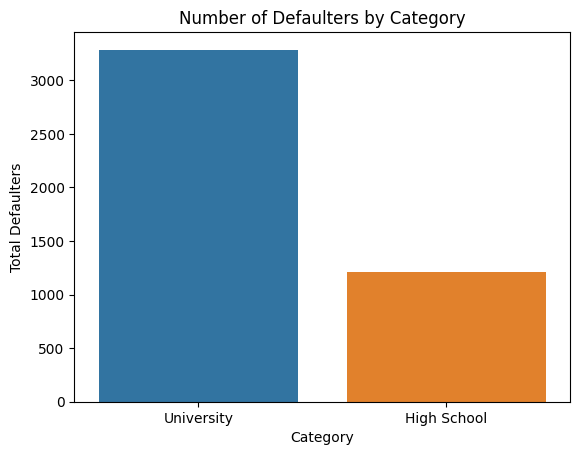
total university data: 14030  
total university defaulters: 3285  
percentage of uni defaulters: 23.414112615823235

* high school defaulters

edu\_high = df[(df.EDUCATION== 3)]   
print("total highschool data:", edu\_high.shape[0])  
  
total\_hschool\_def = df[(df.EDUCATION==3)&(df['default.payment.next.month']==1)&(df.MARRIAGE != 'others')].shape[0]  
print("total highschool defaulters:", total\_hschool\_def)  
percentage3 = (total\_hschool\_def/edu\_high.shape[0])\*100  
print("percentage of highschool defaulters: ",percentage3)

total highschool data: 4917  
total highschool defaulters: 1206  
percentage of highschool defaulters: 24.527150701647347

data = {  
 'Category': ['University', 'High School'],  
 'Total Data': [edu\_uni.shape[0], edu\_high.shape[0]],  
 'Total Defaulters': [total\_uni\_def, total\_hschool\_def],  
 'Percentage of Defaulters': [percentage2, percentage3]  
}  
  
df\_plot = pd.DataFrame(data)  
  
sns.barplot(data=df\_plot, x='Category', y='Total Defaulters')  
  
plt.xlabel('Category')  
plt.ylabel('Total Defaulters')  
plt.title('Number of Defaulters by Category')  
  
plt.show()



## DATA TRANSFORMATION

* transforming wide data to long data for repayment status

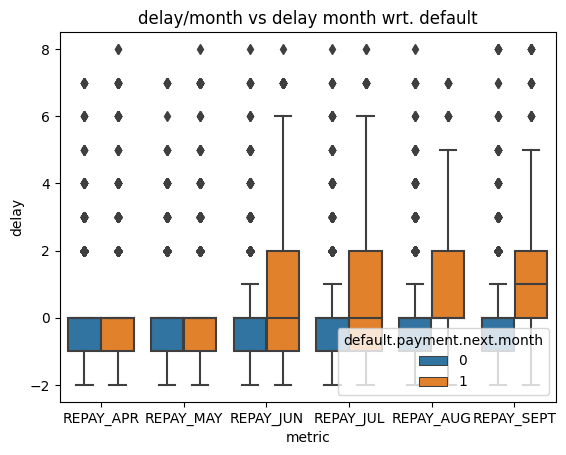
long\_df = df.melt(id\_vars=['default.payment.next.month'], value\_vars=['REPAY\_APR','REPAY\_MAY','REPAY\_JUN','REPAY\_JUL','REPAY\_AUG','REPAY\_SEPT'], var\_name= 'metric', value\_name='delay')  
print(long\_df)

default.payment.next.month metric delay  
0 1 REPAY\_APR -2  
1 1 REPAY\_APR 2  
2 0 REPAY\_APR 0  
3 0 REPAY\_APR 0  
4 0 REPAY\_APR 0  
... ... ... ...  
179995 0 REPAY\_SEPT 0  
179996 0 REPAY\_SEPT -1  
179997 1 REPAY\_SEPT 4  
179998 1 REPAY\_SEPT 1  
179999 1 REPAY\_SEPT 0  
  
[180000 rows x 3 columns]

* repayment in sept is delayed increasing chances of default in next month

sns.boxplot(x = long\_df['metric'], y= long\_df.delay, hue=long\_df['default.payment.next.month'])  
plt.title("delay/month vs delay month wrt. default")

Text(0.5, 1.0, 'delay/month vs delay month wrt. default')



* transforming wide data to long data for bill amount

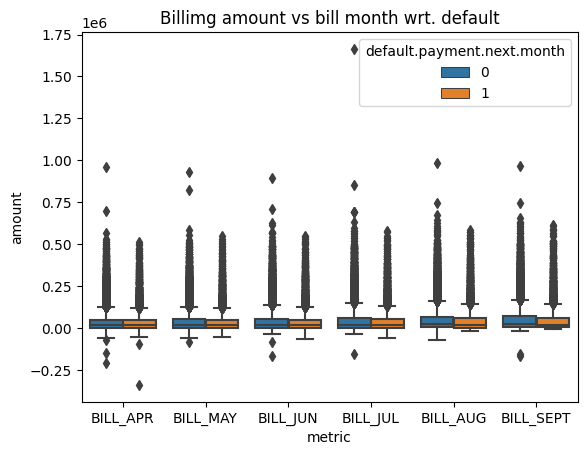
long\_df1 = df.melt(id\_vars=['default.payment.next.month'] , value\_vars=['BILL\_APR','BILL\_MAY','BILL\_JUN','BILL\_JUL','BILL\_AUG','BILL\_SEPT'], var\_name= 'metric', value\_name='amount')  
print(long\_df1)

default.payment.next.month metric amount  
0 1 BILL\_APR 0.0  
1 1 BILL\_APR 3261.0  
2 0 BILL\_APR 15549.0  
3 0 BILL\_APR 29547.0  
4 0 BILL\_APR 19131.0  
... ... ... ...  
179995 0 BILL\_SEPT 188948.0  
179996 0 BILL\_SEPT 1683.0  
179997 1 BILL\_SEPT 3565.0  
179998 1 BILL\_SEPT -1645.0  
179999 1 BILL\_SEPT 47929.0  
  
[180000 rows x 3 columns]

* the data is skewed

sns.boxplot(long\_df1, x='metric', y = 'amount', hue='default.payment.next.month')  
plt.title("Billimg amount vs bill month wrt. default")

Text(0.5, 1.0, 'Billimg amount vs bill month wrt. default')



* transforming wide data to long data for paid amount

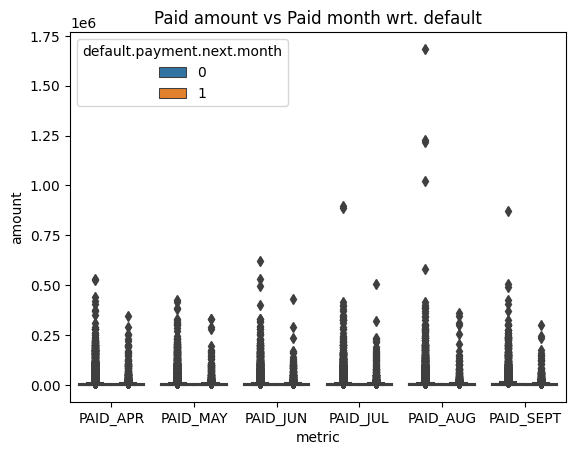
long\_df2 = df.melt(id\_vars=['default.payment.next.month'], value\_vars=['PAID\_APR','PAID\_MAY','PAID\_JUN','PAID\_JUL','PAID\_AUG','PAID\_SEPT'], var\_name= 'metric', value\_name='amount')  
print(long\_df2)

default.payment.next.month metric amount  
0 1 PAID\_APR 0.0  
1 1 PAID\_APR 2000.0  
2 0 PAID\_APR 5000.0  
3 0 PAID\_APR 1000.0  
4 0 PAID\_APR 679.0  
... ... ... ...  
179995 0 PAID\_SEPT 8500.0  
179996 0 PAID\_SEPT 1837.0  
179997 1 PAID\_SEPT 0.0  
179998 1 PAID\_SEPT 85900.0  
179999 1 PAID\_SEPT 2078.0  
  
[180000 rows x 3 columns]

* the data is highly skewed

sns.boxplot(long\_df2, x='metric',y='amount', hue='default.payment.next.month')  
plt.title("Paid amount vs Paid month wrt. default")

Text(0.5, 1.0, 'Paid amount vs Paid month wrt. default')

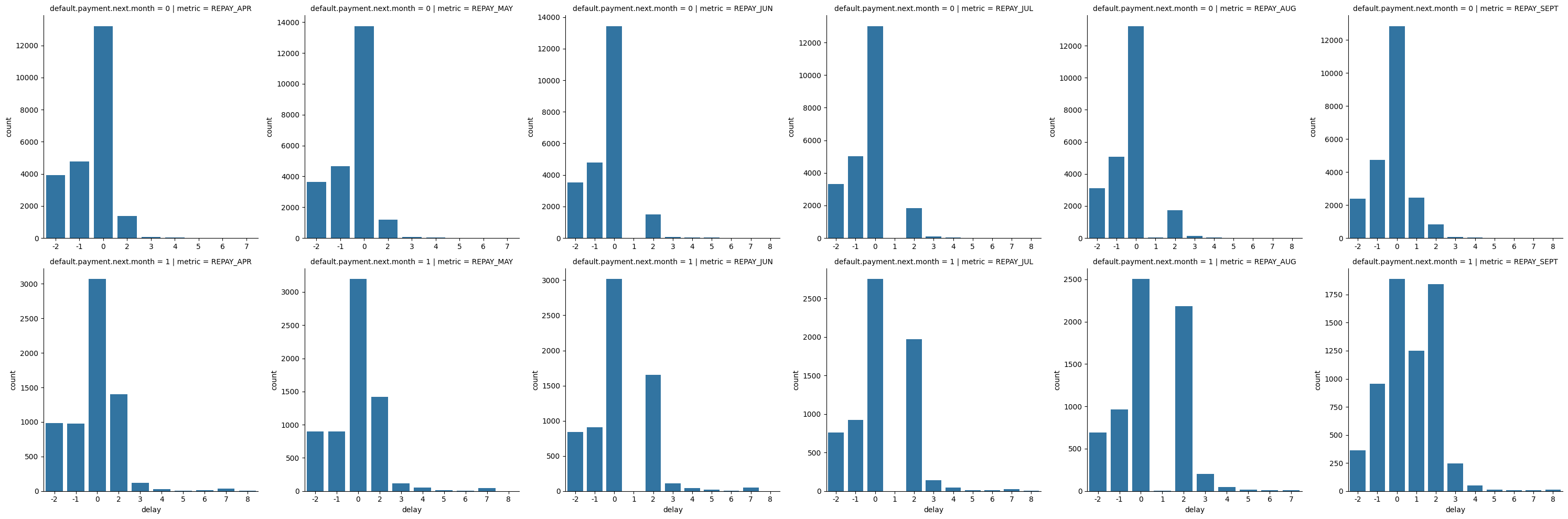


### Creating Subplot

* interpreting the repayment status relation to defaluts further

g = sns.FacetGrid(long\_df, row='default.payment.next.month', col='metric', sharex=False, sharey=False, height=5)  
g.map\_dataframe(sns.countplot, x='delay')

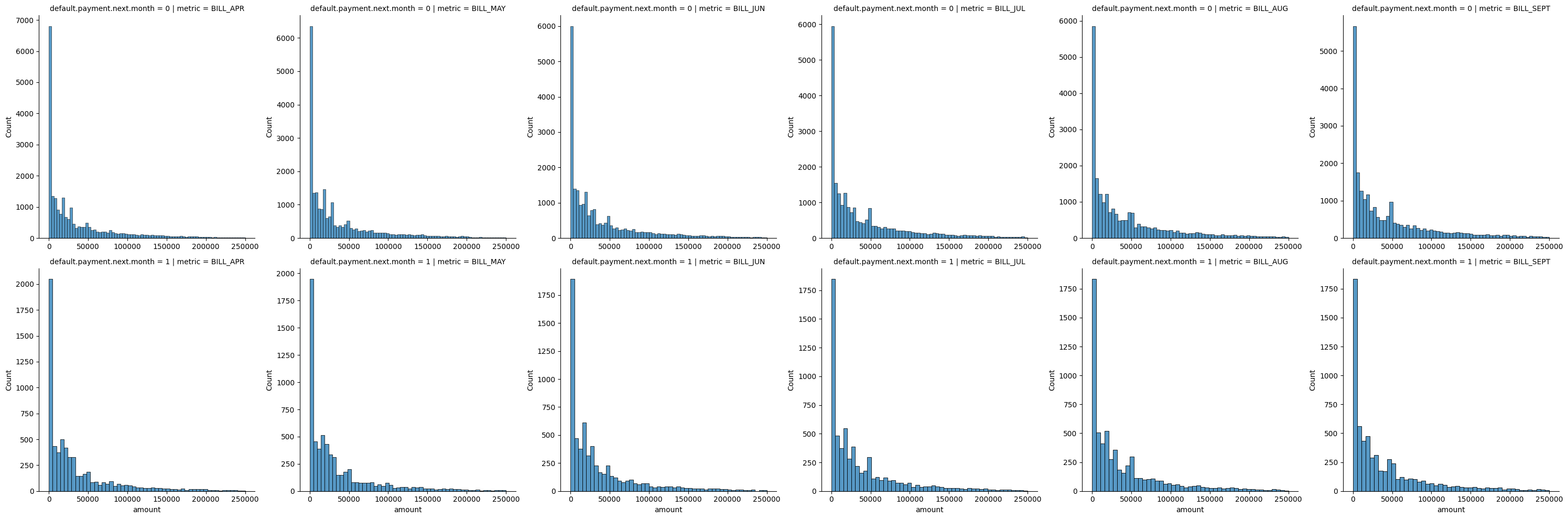
d:\source\repos\OrionDataAnalyticsInternshipJul23\Orion.venv\lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The figure layout has changed to tight  
 self.\_figure.tight\_layout(\*args, \*\*kwargs)



* interpreting the bill amounts relation to defaluts further

g = sns.FacetGrid(long\_df1, row='default.payment.next.month', col='metric', sharex=False, sharey=False, height=5)  
g.map\_dataframe(sns.histplot, x='amount', binrange=(0,250000))

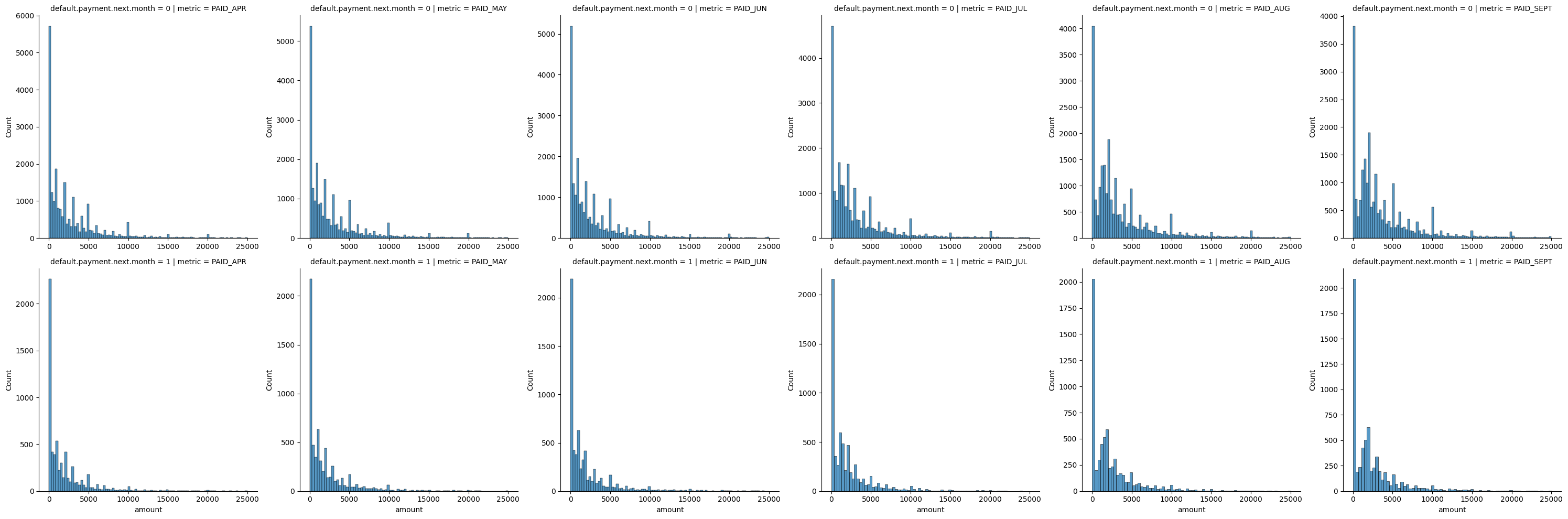
d:\source\repos\OrionDataAnalyticsInternshipJul23\Orion.venv\lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The figure layout has changed to tight  
 self.\_figure.tight\_layout(\*args, \*\*kwargs)



* interpreting the paid amounts relation to defaluts further

g = sns.FacetGrid(long\_df2, row='default.payment.next.month', col='metric', sharex=False, sharey=False, height=5)  
g.map\_dataframe(sns.histplot, x='amount', binrange=(0,25000))

d:\source\repos\OrionDataAnalyticsInternshipJul23\Orion.venv\lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The figure layout has changed to tight  
 self.\_figure.tight\_layout(\*args, \*\*kwargs)



## Extracting Cleaned Data

file\_path = r'D:\source\repos\OrionDataAnalyticsInternshipJul23\data\clean/'  
file\_name = "credit\_default.csv"  
extract\_info = file\_path + file\_name  
df.to\_csv(extract\_info, index=False)