```
# Importing Necessary Libraries
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from sklearn.metrics import classification_report, confusion_matrix, ConfusionMatrixDisplay, roc_auc_score
from sklearn.metrics import classification_report
from sklearn.impute import SimpleImputer
```

```
# Load the Dataset
df = pd.read_csv("/content/default of credit card clients.csv")
```

```
df = df.drop(columns=["Unnamed: 0"], errors="ignore")
df.head()
```

	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	...	X15	X16	X17
0	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_0	PAY_2	PAY_3	PAY_4	PAY_5	...	BILL_AMT4	BILL_AMT5	BILL_AMT6	PA
1	20000	2	2	1	24	2	2	-1	-1	-2	...	0	0	0	
2	120000	2	2	2	26	-1	2	0	0	0	...	3272	3455	3261	
3	90000	2	2	2	34	0	0	0	0	0	...	14331	14948	15549	
4	50000	2	2	1	37	0	0	0	0	0	...	28314	28959	29547	

5 rows × 24 columns

```
# Basic Info
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30001 entries, 0 to 30000
Data columns (total 24 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   X1      30001 non-null  object
 1   X2      30001 non-null  object
 2   X3      30001 non-null  object
 3   X4      30001 non-null  object
 4   X5      30001 non-null  object
 5   X6      30001 non-null  object
 6   X7      30001 non-null  object
 7   X8      30001 non-null  object
 8   X9      30001 non-null  object
 9   X10     30001 non-null  object
 10  X11     30001 non-null  object
 11  X12     30001 non-null  object
 12  X13     30001 non-null  object
 13  X14     30001 non-null  object
 14  X15     30001 non-null  object
 15  X16     30001 non-null  object
 16  X17     30001 non-null  object
 17  X18     30001 non-null  object
 18  X19     30001 non-null  object
 19  X20     30001 non-null  object
 20  X21     30001 non-null  object
 21  X22     30001 non-null  object
 22  X23     30001 non-null  object
 23  Y       30001 non-null  object
dtypes: object(24)
memory usage: 5.5+ MB
```

```
# Shape of dataset
df.shape
```

```
(30001, 24)
```

```
# Data type of columns
df.dtypes
```

	0
X1	object
X2	object
X3	object
X4	object
X5	object
X6	object
X7	object
X8	object
X9	object
X10	object
X11	object
X12	object
X13	object
X14	object
X15	object
X16	object
X17	object
X18	object
X19	object
X20	object
X21	object
X22	object
X23	object
Y	object

dtype: object

```
# Basic statistics
df.describe()
```

	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	...	X15	X16	X17	X18	X19	X20	X
count	30001	30001	30001	30001	30001	30001	30001	30001	30001	30001	...	30001	30001	30001	30001	30001	30001	300
unique	82	3	8	5	57	12	12	12	12	11	...	21549	21011	20605	7944	7900	7519	69
top	50000	2	2	2	29	0	0	0	0	0	...	0	0	0	0	0	0	
freq	3365	18112	14030	15964	1605	14737	15730	15764	16455	16947	...	3195	3506	4020	5249	5396	5968	64

4 rows × 24 columns

```
# unique value in columns
df.nunique()
```

	0
X1	82
X2	3
X3	8
X4	5
X5	57
X6	12
X7	12
X8	12
X9	12
X10	11
X11	11
X12	22724
X13	22347
X14	22027
X15	21549
X16	21011
X17	20605
X18	7944
X19	7900
X20	7519
X21	6938
X22	6898
X23	6940
Y	3

dtype: int64

```
# Checking missing values
df.isnull().sum()
```

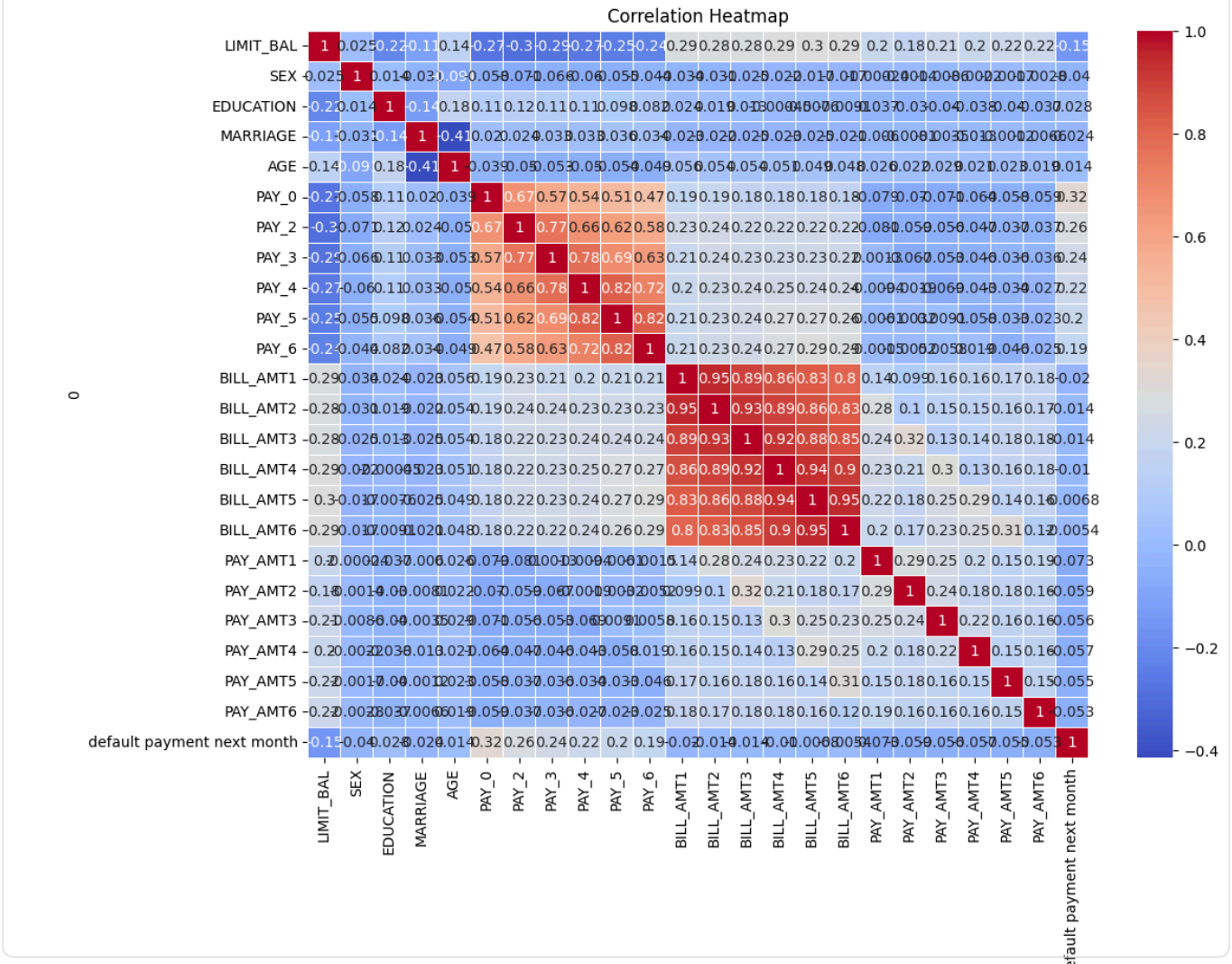
	0
X1	0
X2	0
X3	0
X4	0
X5	0
X6	0
X7	0
X8	0
X9	0
X10	0
X11	0
X12	0
X13	0
X14	0
X15	0
X16	0
X17	0
X18	0
X19	0
X20	0
X21	0
X22	0
X23	0
Y	0

dtype: int64

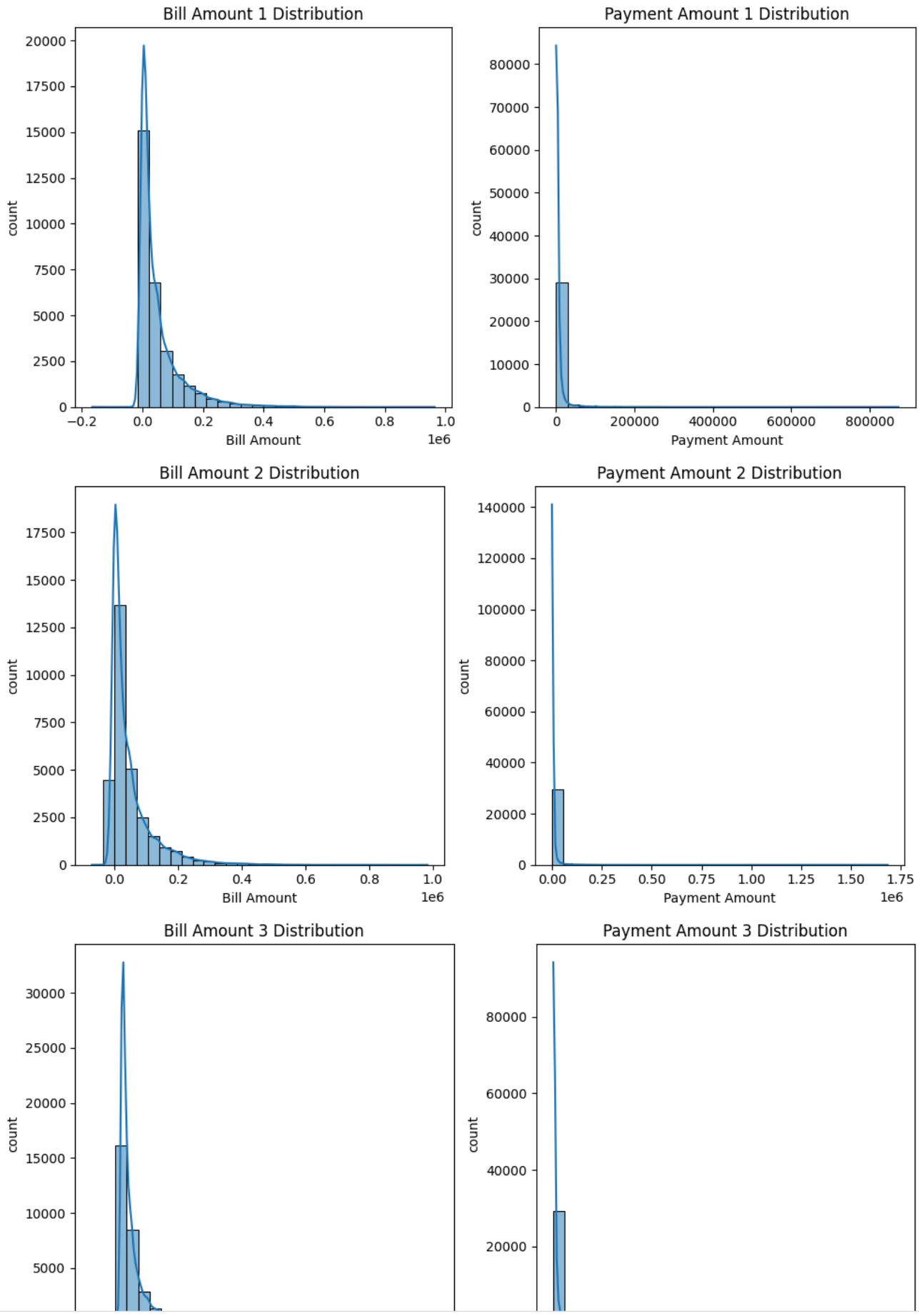
```
# checking duplicate value
df.duplicated().sum()
```

```
np.int64(35)
```

```
# Drop the first row which contains the original header values
# Set the first row as the header
original_header = df.iloc[0]
df_numeric = df.iloc[1:].copy()
df_numeric.columns = original_header


# Convert relevant columns to numeric before calculating correlation
columns_to_numeric = ['LIMIT_BAL', 'SEX', 'EDUCATION', 'MARRIAGE', 'AGE', 'PAY_0', 'PAY_2', 'PAY_3', 'PAY_4',
                      'PAY_5', 'PAY_6', 'BILL_AMT1', 'BILL_AMT2', 'BILL_AMT3', 'BILL_AMT4', 'BILL_AMT5', 'BILL_AMT6',
                      'PAY_AMT1', 'PAY_AMT2', 'PAY_AMT3', 'PAY_AMT4', 'PAY_AMT5', 'PAY_AMT6',
                      'default payment next month']

for col in columns_to_numeric:
    df_numeric[col] = pd.to_numeric(df_numeric[col], errors='coerce')

# Drop non-numeric columns before calculating correlation
correlation_matrix = df_numeric.corr()
plt.figure(figsize=(12, 9))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
plt.title('Correlation Heatmap')
plt.show()
```

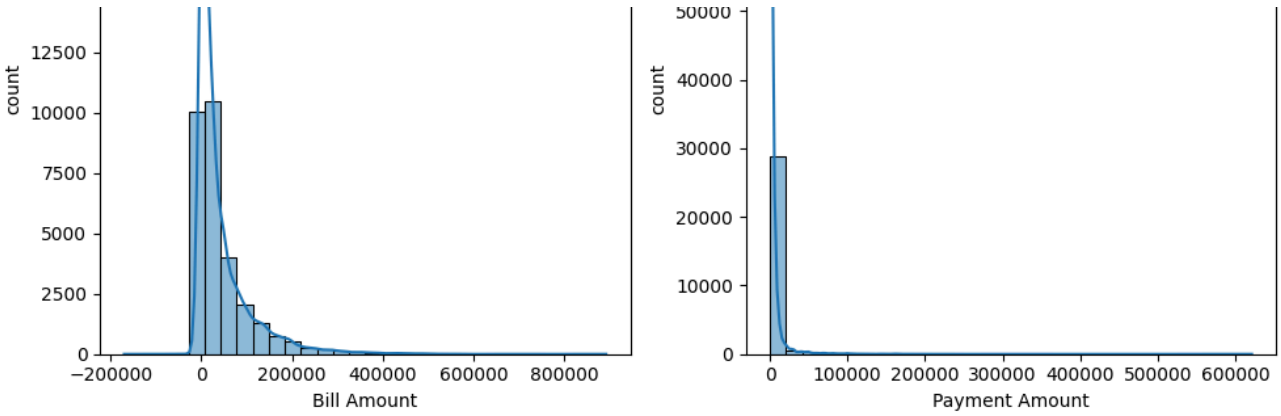


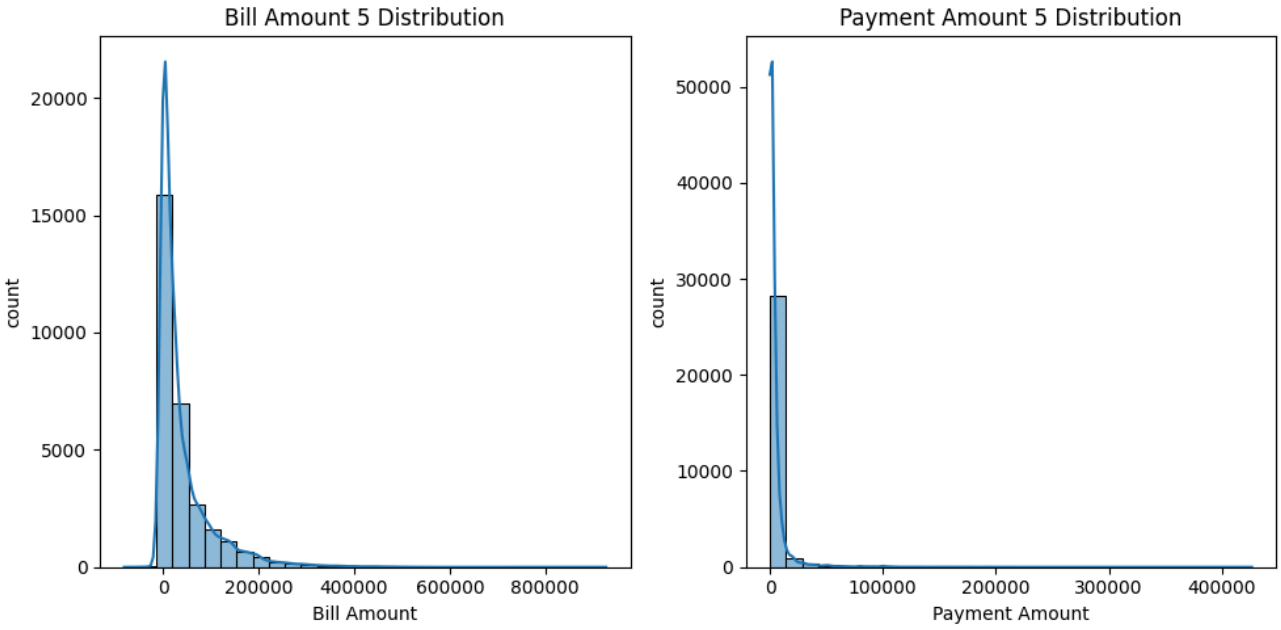

```
# Distribution of Bill and Payment Amounts
for i in range(1,7):
  plt.figure(figsize = (10,5))
  plt.subplot(1,2,1)
  sns.histplot(df_numeric[f'BILL_AMT{i}'], bins =30, kde = True)
  plt.title(f'Bill Amount {i} Distribution')
  plt.xlabel('Bill Amount')
  plt.ylabel('count')

  plt.subplot(1,2,2)
  sns.histplot(df_numeric[f'PAY_AMT{i}'], bins=30, kde=True)
  plt.title(f'Payment Amount {i} Distribution')
  plt.xlabel('Payment Amount')
  plt.ylabel('count')

  plt.tight_layout()
  plt.show()
```

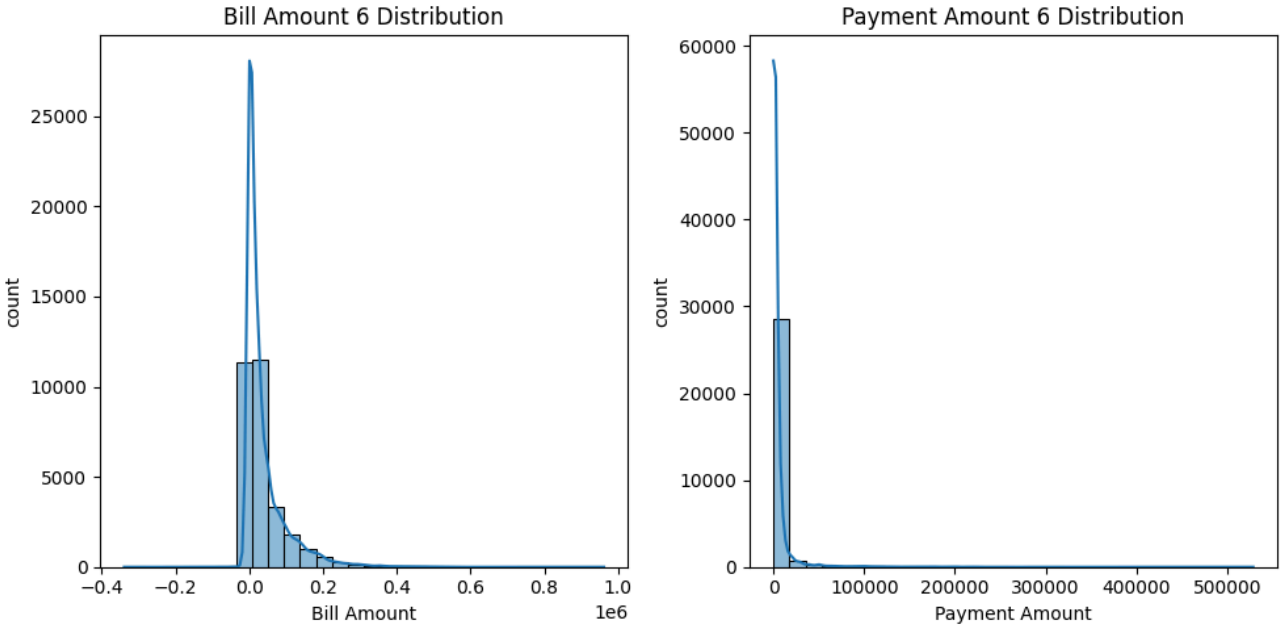


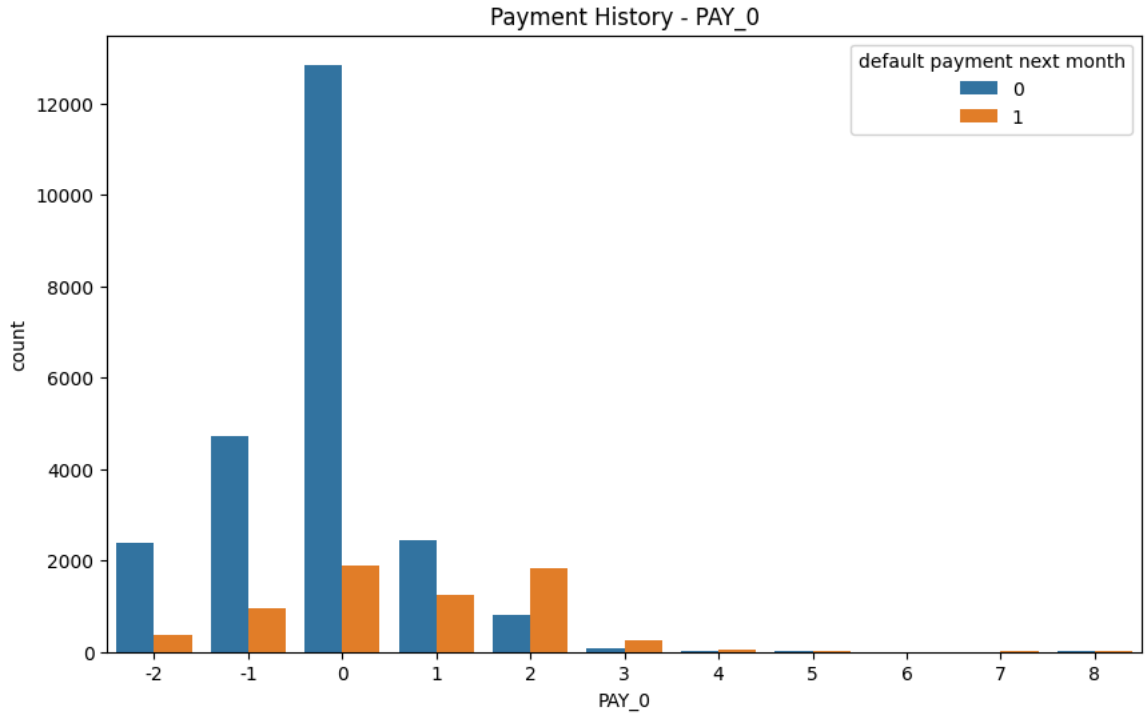

```
# Relationship Between Payment History and Default Status


payment_history_columns = ['PAY_0', 'PAY_2', 'PAY_3', 'PAY_4', 'PAY_5', 'PAY_6']

for col in payment_history_columns:
    plt.figure(figsize=(10, 6))
    sns.countplot( x=col, hue = 'default payment next month', data=df_numeric)
```

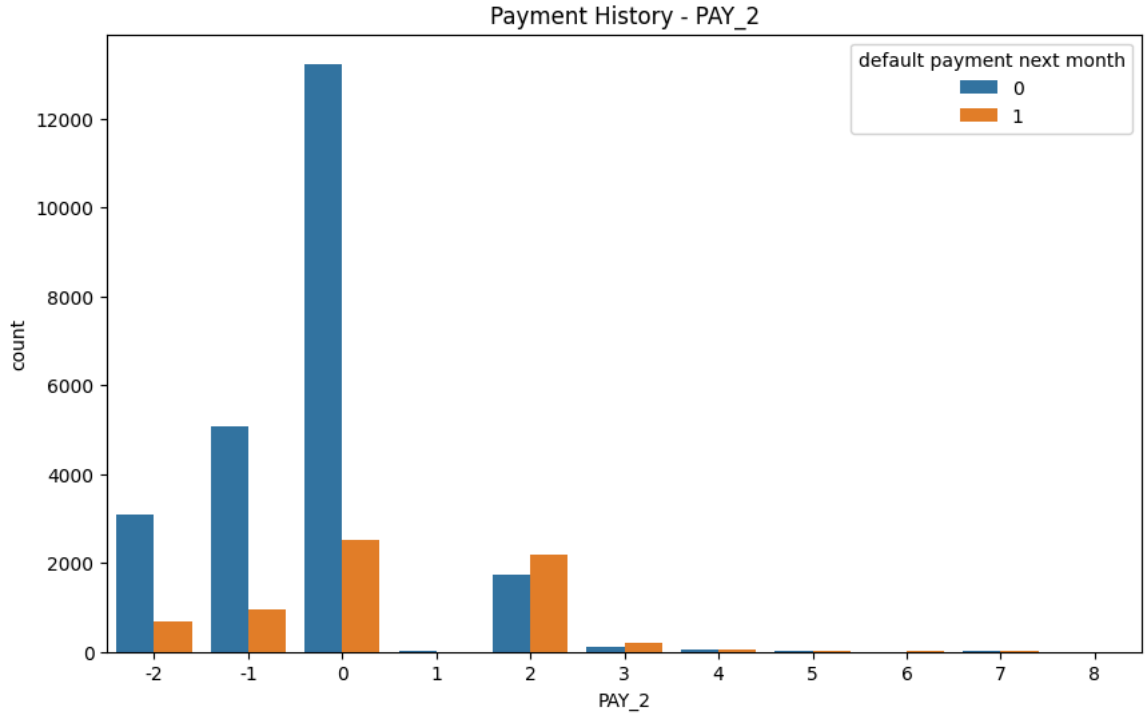
```
plt.title(f'Payment History - {col}')
plt.show()
```

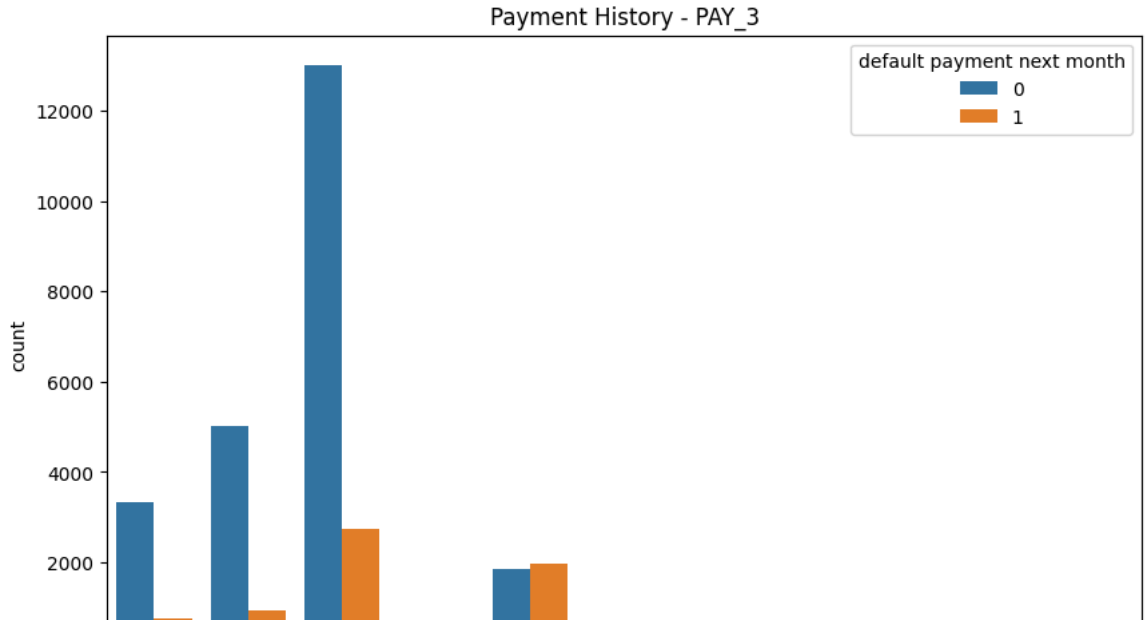


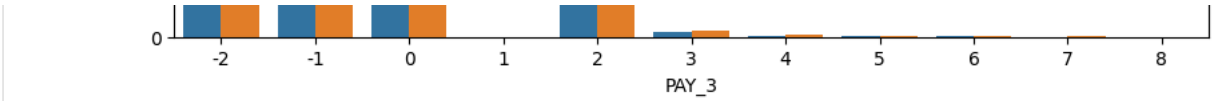


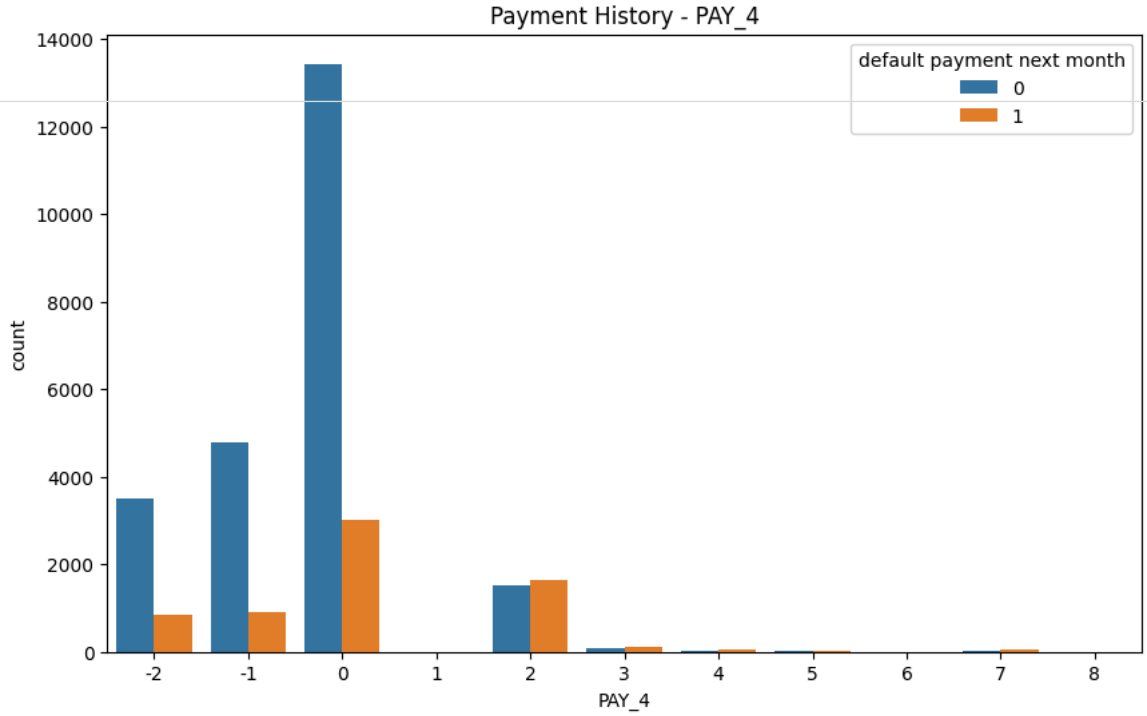


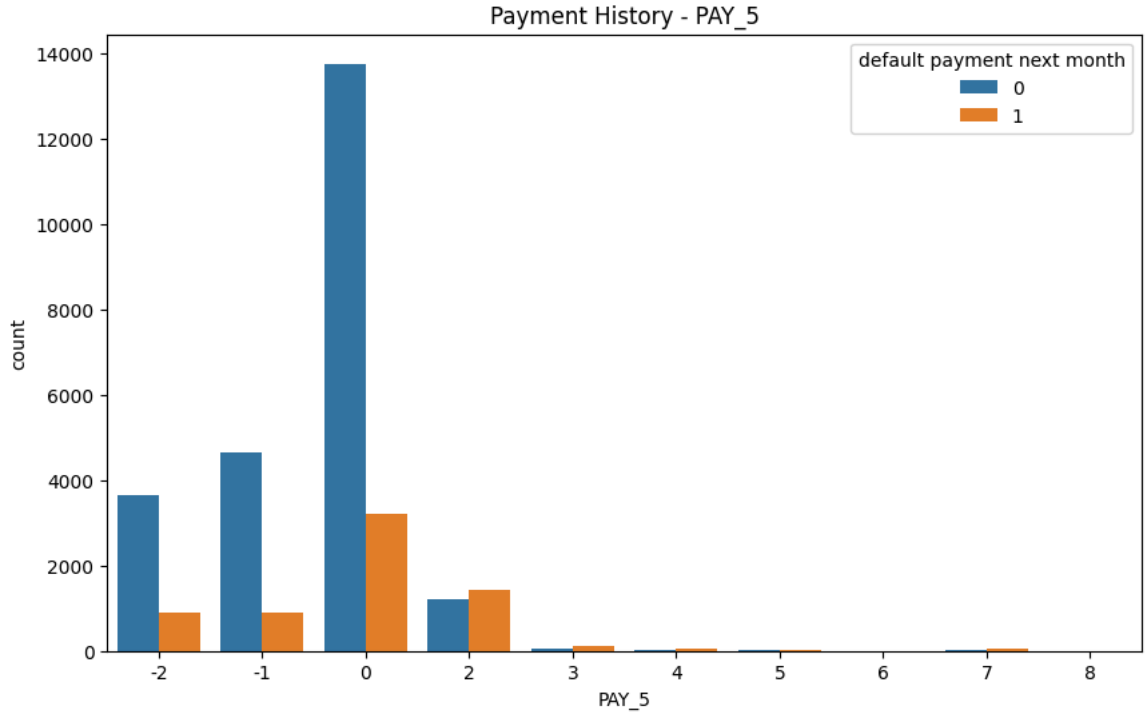

Payment History - PAY_0
default payment next month
0
1
count
PAY_0

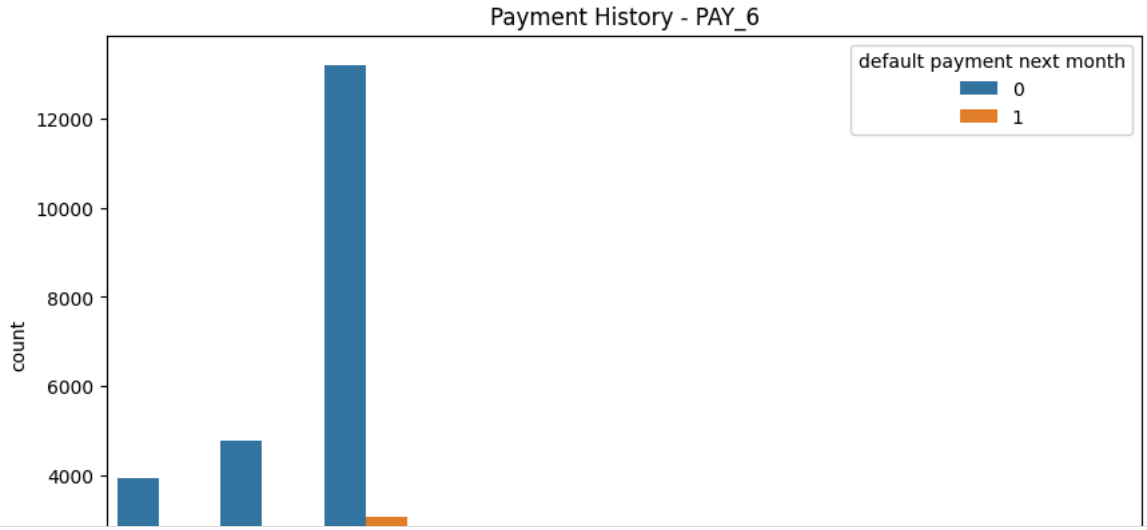

Payment History - PAY_2
default payment next month
0
1
count
PAY_2


Payment History - PAY_3
default payment next month
0
1
count

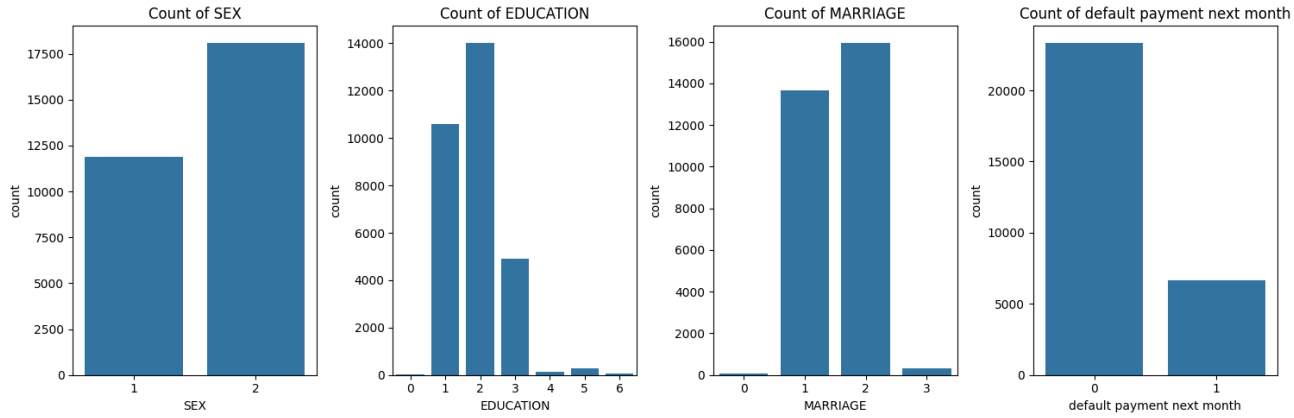

0
-2
-1
0
1
2
3
4
5
6
7
8
PAY_3


Payment History - PAY_4
default payment next month
0
1
count
14000
12000
10000
8000
6000
4000
2000
0
-2
-1
0
1
2
3
4
5
6
7
8
PAY_4

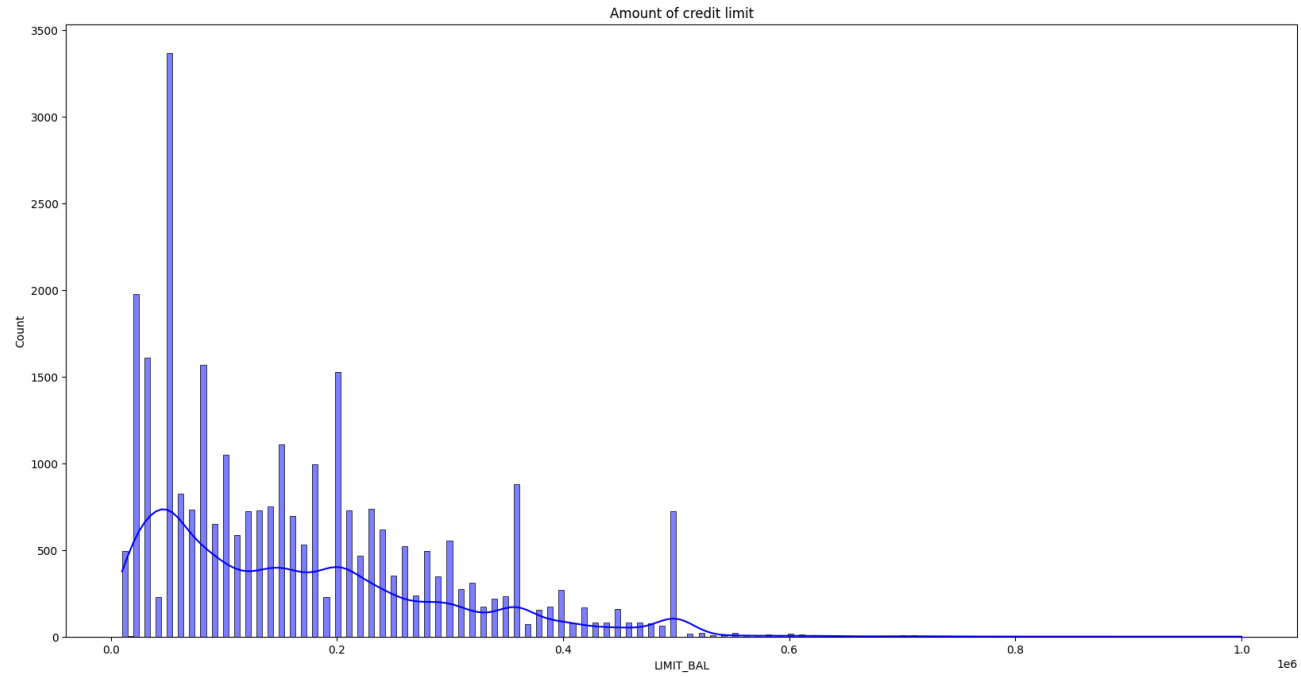

Payment History - PAY_5
default payment next month
0
1
count
14000
12000
10000
8000
6000
4000
2000
0
-2
-1
0
2
3
4
5
6
7
8
PAY_5


Payment History - PAY_6
default payment next month
0
1
count
12000
10000
8000
6000
4000

```
# Visualize the count of categorical features
categorical_cols = ['SEX', 'EDUCATION', 'MARRIAGE', 'default payment next month'] # Selecting categorical columns

plt.figure(figsize=(15, 5))
for i, col in enumerate(categorical_cols):
    plt.subplot(1, 4, i + 1)
    sns.countplot(data=df_numeric, x=col)
    plt.title(f'Count of {col}')
plt.tight_layout()
plt.show()
```




```
# Distribution of Credit Limit (LIMIT_BAL)


plt.figure(figsize = (20, 10))
plt.title('Amount of credit limit')
sns.histplot(df_numeric['LIMIT_BAL'], kde = True, bins = 200, color = 'blue')
plt.show()
```



Skewness and Kurtosis

```
# Select only numeric columns for skewness and kurtosis calculation
numeric_cols = df_numeric.select_dtypes(include=np.number).columns

# Calculate and print skewness for numeric columns
print("Skewness:")
print(df_numeric[numeric_cols].skew())

# Calculate and print kurtosis for numeric columns
print("\nKurtosis:")
print(df_numeric[numeric_cols].kurtosis())
```

```
Skewness:
0
LIMIT_BAL                    0.992867
SEX                         -0.424183
EDUCATION                    0.970972
MARRIAGE                    -0.018742
AGE                          0.732246
PAY_0                        0.731975
PAY_2                        0.790565
PAY_3                        0.840682
PAY_4                        0.999629
PAY_5                        1.008197
PAY_6                        0.948029
BILL_AMT1                    2.663861
BILL_AMT2                    2.705221
BILL_AMT3                    3.087830
BILL_AMT4                    2.821965
BILL_AMT5                    2.876380
BILL_AMT6                    2.846645
PAY_AMT1                    14.668364
PAY_AMT2                    30.453817
PAY_AMT3                    17.216635
PAY_AMT4                    12.904985
PAY_AMT5                    11.127417
PAY_AMT6                    10.640727
default payment next month   1.343504
dtype: float64
```

```
Kurtosis:
0
LIMIT_BAL                      0.536263
SEX                           -1.820190
EDUCATION                      2.078622
MARRIAGE                      -1.363368
AGE                            0.044303
PAY_0                          2.720715
PAY_2                          1.570418
PAY_3                          2.084436
PAY_4                          3.496983
PAY_5                          3.989748
PAY_6                          3.426534
BILL_AMT1                      9.806289
BILL_AMT2                     10.302946
BILL_AMT3                     19.783255
BILL_AMT4                     11.309325
BILL_AMT5                     12.305881
BILL_AMT6                     12.270705
PAY_AMT1                     415.254743
PAY_AMT2                    1641.631911
PAY_AMT3                     564.311229
PAY_AMT4                     277.333768
PAY_AMT5                     180.063940
PAY_AMT6                     167.161430
default payment next month    -0.195010
dtype: float64
```

Outlier Detection using IQR

```
# Select only numeric columns for outlier detection
numeric_cols = df_numeric.select_dtypes(include=np.number).columns

# Calculate IQR and identify outliers
outlier_indices = {}
for col in numeric_cols:
    Q1 = df_numeric[col].quantile(0.25)
    Q3 = df_numeric[col].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    outliers = df_numeric[(df_numeric[col] < lower_bound) | (df_numeric[col] > upper_bound)].index
    outlier_indices[col] = outliers

# Print the number of outliers for each column
print("Number of outliers per column (based on IQR):")
for col, indices in outlier_indices.items():
    print(f"{col}: {len(indices)}")
```

```
Number of outliers per column (based on IQR):
LIMIT_BAL: 167
SEX: 0
EDUCATION: 454
MARRIAGE: 0
AGE: 272
PAY_0: 3130
PAY_2: 4410
PAY_3: 4209
PAY_4: 3508
PAY_5: 2968
PAY_6: 3079
BILL_AMT1: 2400
BILL_AMT2: 2395
BILL_AMT3: 2469
BILL_AMT4: 2622
BILL_AMT5: 2725
BILL_AMT6: 2693
PAY_AMT1: 2745
PAY_AMT2: 2714
PAY_AMT3: 2598
PAY_AMT4: 2994
PAY_AMT5: 2945
PAY_AMT6: 2958
default payment next month: 6636
```

Boxplots to Visualize Outliers

```
# Create boxplots for numeric columns to visualize outliers
numeric_cols = df_numeric.select_dtypes(include=np.number).columns

plt.figure(figsize=(15, 10))
sns.boxplot(data=df_numeric[numeric_cols])
plt.title('Boxplot of Numeric Features (with Outliers)')
```