Big_Data_Analytics_Project_EDA_Mar16

March 16, 2025

[238]:
```
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]
```

[239]:
```
#Renaming columns
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)
```

[240]:
```
data.head()
```

[240]:

	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]
```

[241]:
```
data.tail()
```

[241]:
```
          ID  LIMIT_BAL  SEX  EDUCATION  MARRIAGE  AGE  PAY_0  PAY_2  PAY_3  \
29995  29996     220000    1          3         1   39      0      0      0
29996  29997     150000    1          3         2   43     -1     -1     -1
29997  29998      30000    1          2         2   37      4      3      2
29998  29999      80000    1          3         1   41      1     -1      0
29999  30000      50000    1          2         1   46      0      0      0

       PAY_4  …  BILL_AMT4  BILL_AMT5  BILL_AMT6  PAY_AMT1  PAY_AMT2  \
29995      0  …      88004      31237      15980      8500     20000
29996     -1  …       8979       5190          0      1837      3526
29997     -1  …      20878      20582      19357         0         0
29998      0  …      52774      11855      48944     85900      3409
29999      0  …      36535      32428      15313      2078      1800

       PAY_AMT3  PAY_AMT4  PAY_AMT5  PAY_AMT6  Y
29995      5003      3047      5000      1000  0
29996      8998       129         0         0  0
29997     22000      4200      2000      3100  1
29998      1178      1926     52964      1804  1
29999      1430      1000      1000      1000  1

[5 rows x 25 columns]
```

[242]:
```
#Counting the number of rows:
print("The number of rows is:", len(data.index))
```

```
The number of rows is: 30000
```

[243]:
```
data.shape
```

[243]:
```
(30000, 25)
```

[244]:
```
data.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  int64
 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  int64
 13  BILL_AMT2  30000 non-null  int64
 14  BILL_AMT3  30000 non-null  int64
 15  BILL_AMT4  30000 non-null  int64
 16  BILL_AMT5  30000 non-null  int64
 17  BILL_AMT6  30000 non-null  int64
 18  PAY_AMT1   30000 non-null  int64
 19  PAY_AMT2   30000 non-null  int64
 20  PAY_AMT3   30000 non-null  int64
 21  PAY_AMT4   30000 non-null  int64
 22  PAY_AMT5   30000 non-null  int64
 23  PAY_AMT6   30000 non-null  int64
 24  Y          30000 non-null  int64
dtypes: int64(25)
memory usage: 5.7 MB
```

[245]:
```
#Checking the data types of the attributes:
data.dtypes
```

[245]:
```
ID           int64
LIMIT_BAL    int64
SEX          int64
EDUCATION    int64
MARRIAGE     int64
```

```
AGE          int64
PAY_0        int64
PAY_2        int64
PAY_3        int64
PAY_4        int64
PAY_5        int64
PAY_6        int64
BILL_AMT1    int64
BILL_AMT2    int64
BILL_AMT3    int64
BILL_AMT4    int64
BILL_AMT5    int64
BILL_AMT6    int64
PAY_AMT1     int64
PAY_AMT2     int64
PAY_AMT3     int64
PAY_AMT4     int64
PAY_AMT5     int64
PAY_AMT6     int64
Y            int64
dtype: object
```

[246]:
```
#Checking missing values:
data.isnull().sum()
```

[246]:
```
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
Y            0
dtype: int64
```

[247]:
```
#Identifying duplicate records:
duplicates = data[data.duplicated()]
print(duplicates)
```

```
Empty DataFrame
Columns: [ID, 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, Y]
Index: []

[0 rows x 25 columns]
```

[248]:
```
data.nunique()
```

[248]:
```
ID          30000
LIMIT_BAL      81
SEX             2
EDUCATION       7
MARRIAGE        4
AGE            56
PAY_0          11
PAY_2          11
PAY_3          11
PAY_4          11
PAY_5          10
PAY_6          10
BILL_AMT1   22723
BILL_AMT2   22346
BILL_AMT3   22026
BILL_AMT4   21548
BILL_AMT5   21010
BILL_AMT6   20604
PAY_AMT1     7943
PAY_AMT2     7899
PAY_AMT3     7518
PAY_AMT4     6937
PAY_AMT5     6897
PAY_AMT6     6939
Y               2
dtype: int64
```

[249]:
```
# 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]
```

[250]:
```
# 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]
```

[251]:
```
data['BillAverage'] = data[['BILL_AMT1', 'BILL_AMT2', 'BILL_AMT3', 'BILL_AMT4',␣
↪'BILL_AMT5', 'BILL_AMT6']].mean(axis=1).round()
data.head()
```

[251]:

	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_AMT5	BILL_AMT6	PAY_AMT1	PAY_AMT2	PAY_AMT3	PAY_AMT4	\
0	…	0	0	0	689	0	0	
1	…	3455	3261	0	1000	1000	1000	
2	…	14948	15549	1518	1500	1000	1000	
3	…	28959	29547	2000	2019	1200	1100	
4	…	19146	19131	2000	36681	10000	9000	

	PAY_AMT5	PAY_AMT6	Y	BillAverage
0	0	0	1	1284.0
1	0	2000	1	2846.0
2	1000	5000	0	16942.0
3	1069	1000	0	38556.0
4	689	679	0	18223.0

[5 rows x 26 columns]

[252]:
```
data['PayAverage'] = data[['PAY_AMT1', 'PAY_AMT2', 'PAY_AMT3', 'PAY_AMT4',␣
 ↪'PAY_AMT5', 'PAY_AMT6']].mean(axis=1).round()
data.head()
```

[252]:

	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_AMT6	PAY_AMT1	PAY_AMT2	PAY_AMT3	PAY_AMT4	PAY_AMT5	PAY_AMT6	\
0	…	0	0	689	0	0	0	0	
1	…	3261	0	1000	1000	1000	0	2000	
2	…	15549	1518	1500	1000	1000	1000	5000	
3	…	29547	2000	2019	1200	1100	1069	1000	
4	…	19131	2000	36681	10000	9000	689	679	

	Y	BillAverage	PayAverage
0	1	1284.0	115.0
1	1	2846.0	833.0
2	0	16942.0	1836.0
3	0	38556.0	1398.0
4	0	18223.0	9842.0

[5 rows x 27 columns]

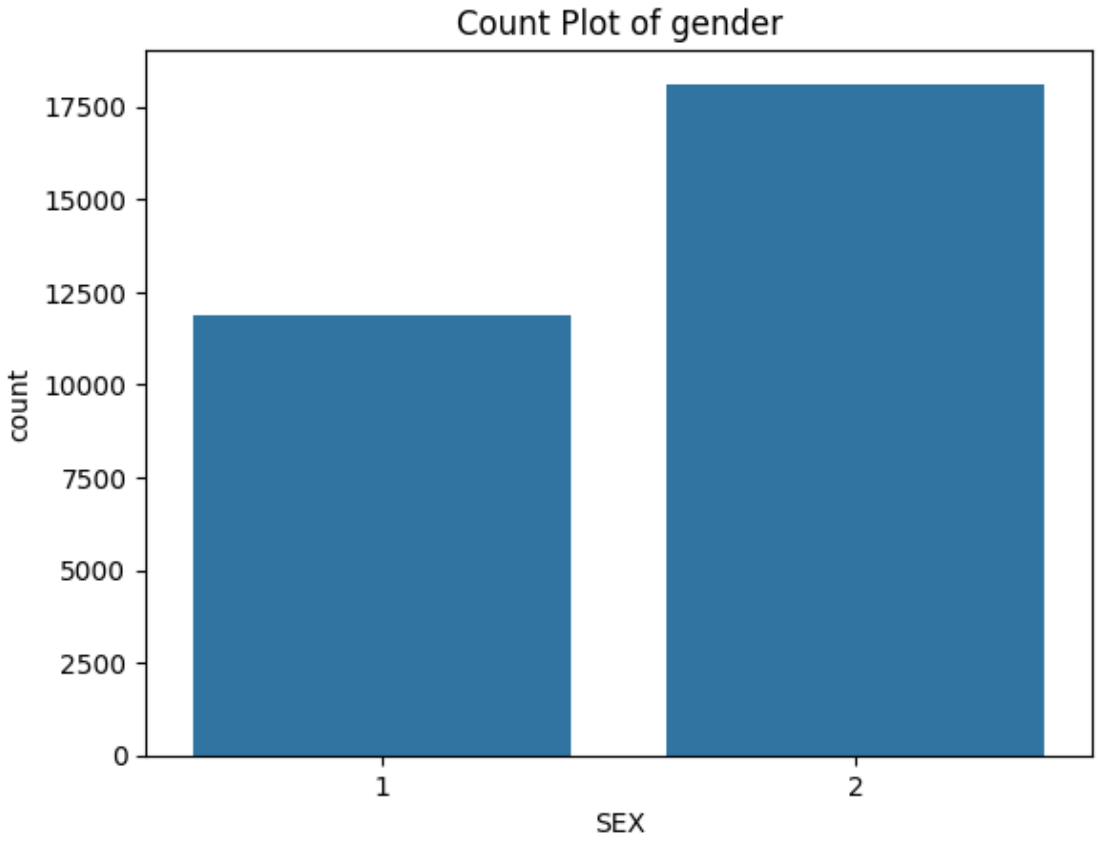
[253]:
```
categorical_variables = ['SEX', 'EDUCATION', 'MARRIAGE', 'PAY_0', 'PAY_2',␣
 ↪'PAY_3', 'PAY_4', 'PAY_5', 'PAY_6']
```

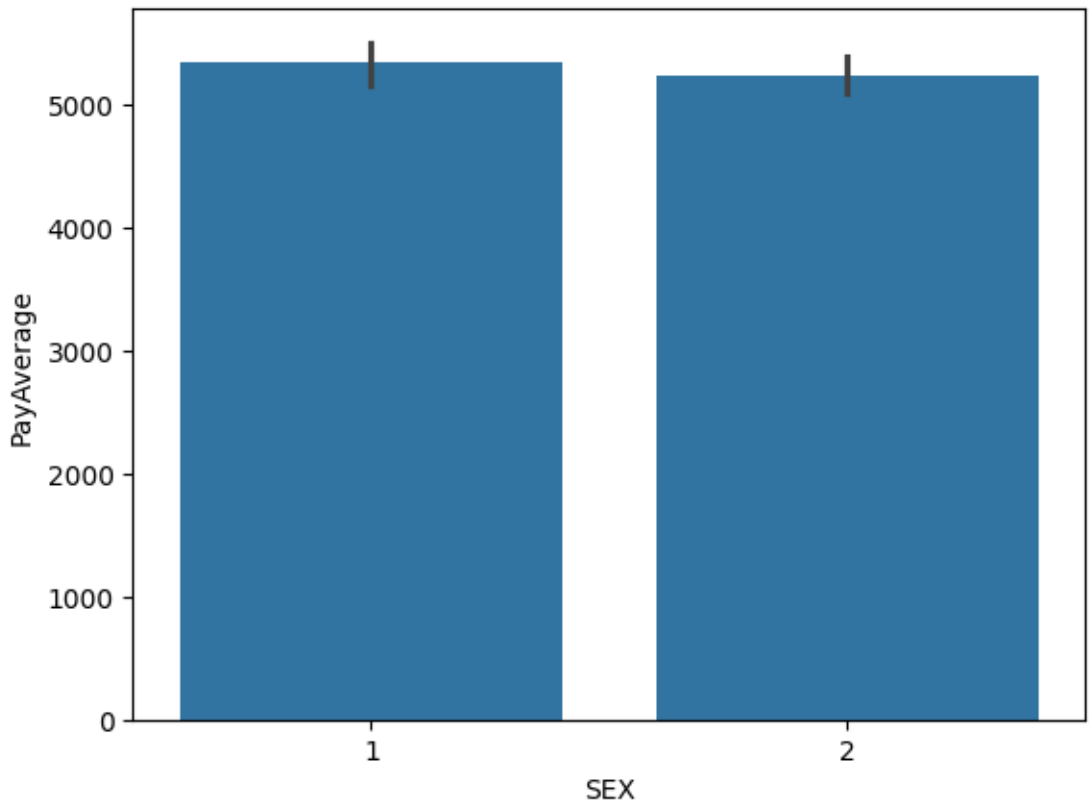
[254]:
```
data['SEX'].value_counts(normalize=True) * 100
```

[254]:
```
SEX
2    60.373333
1    39.626667
Name: proportion, dtype: float64
```

[255]:
```
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()

# Bar plot for Sex and PayAverage
sns.barplot(x='SEX', y='PayAverage', data=data)
```

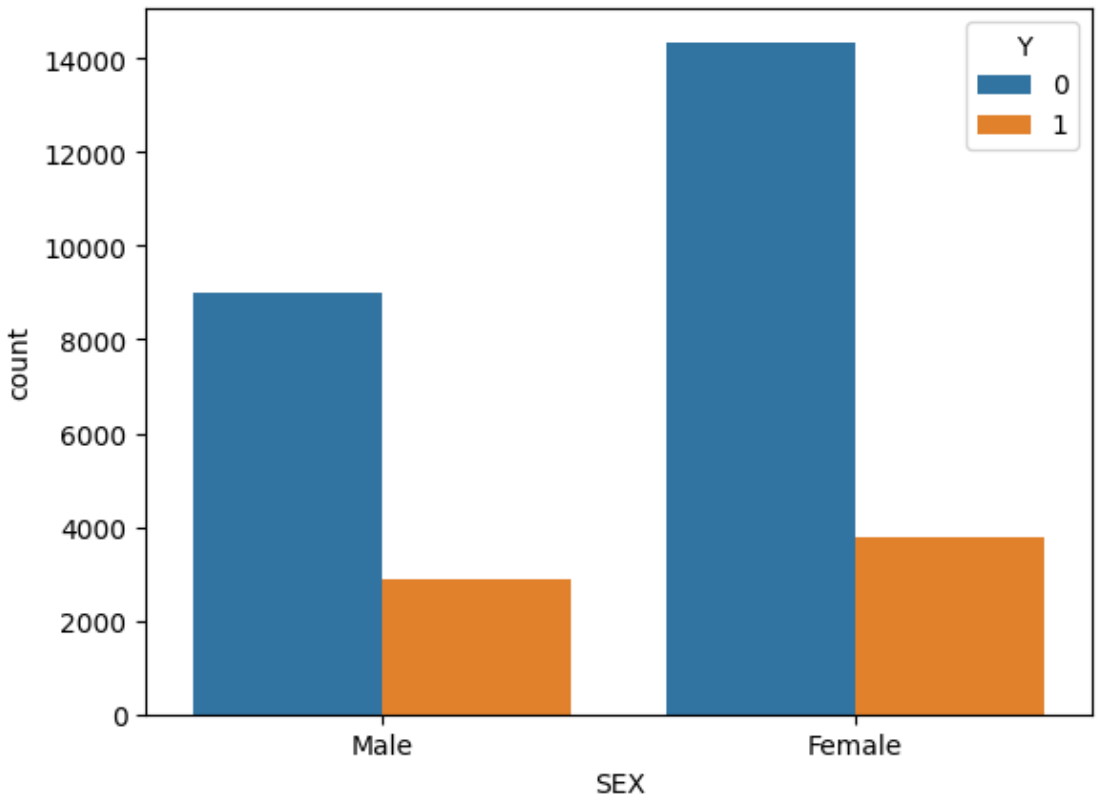
```
plt.show()
```


Count Plot of gender
17500
15000
12500
10000
7500
5000
2500
0
count
1
2
SEX



[256]:
```
sex = sns.countplot(x='SEX', hue='Y', data=data)
sex.set_xticklabels(['Male','Female'])
plt.show()
```

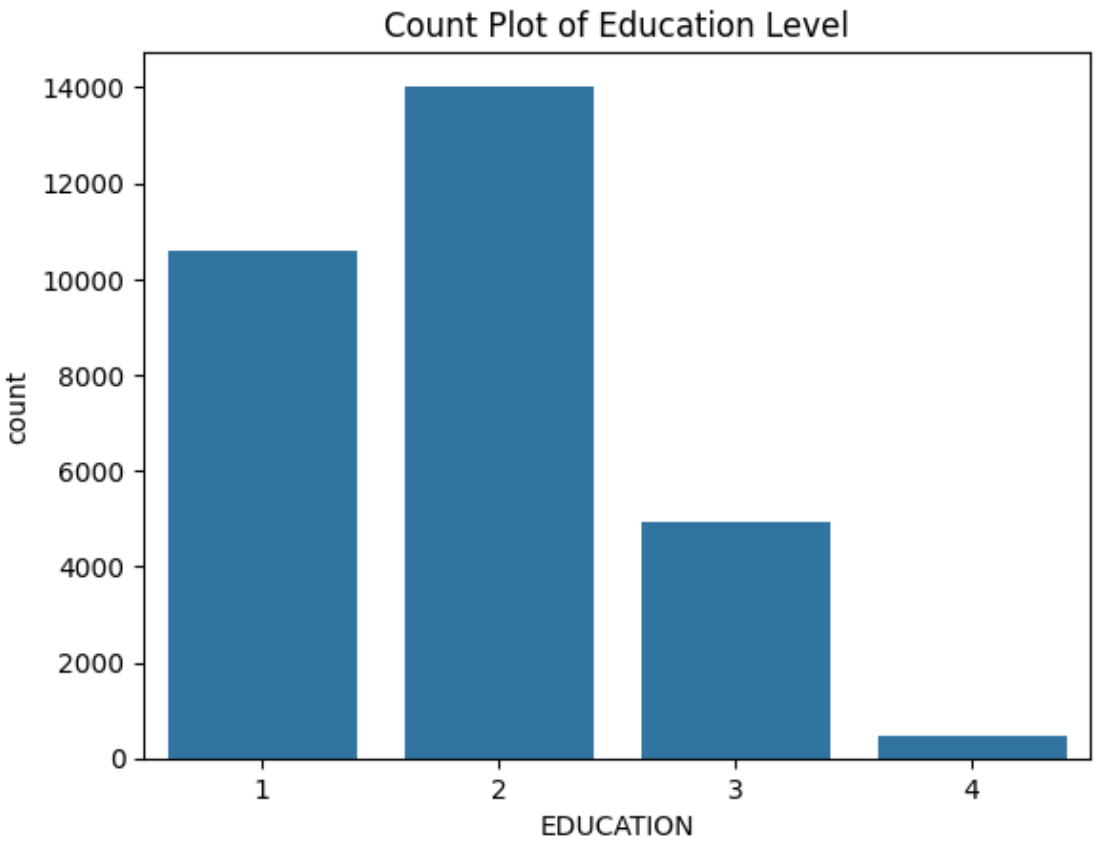
```
<ipython-input-256-19636eb255d9>:2: UserWarning:

set_ticklabels() should only be used with a fixed number of ticks, i.e. after
set_ticks() or using a FixedLocator.
```

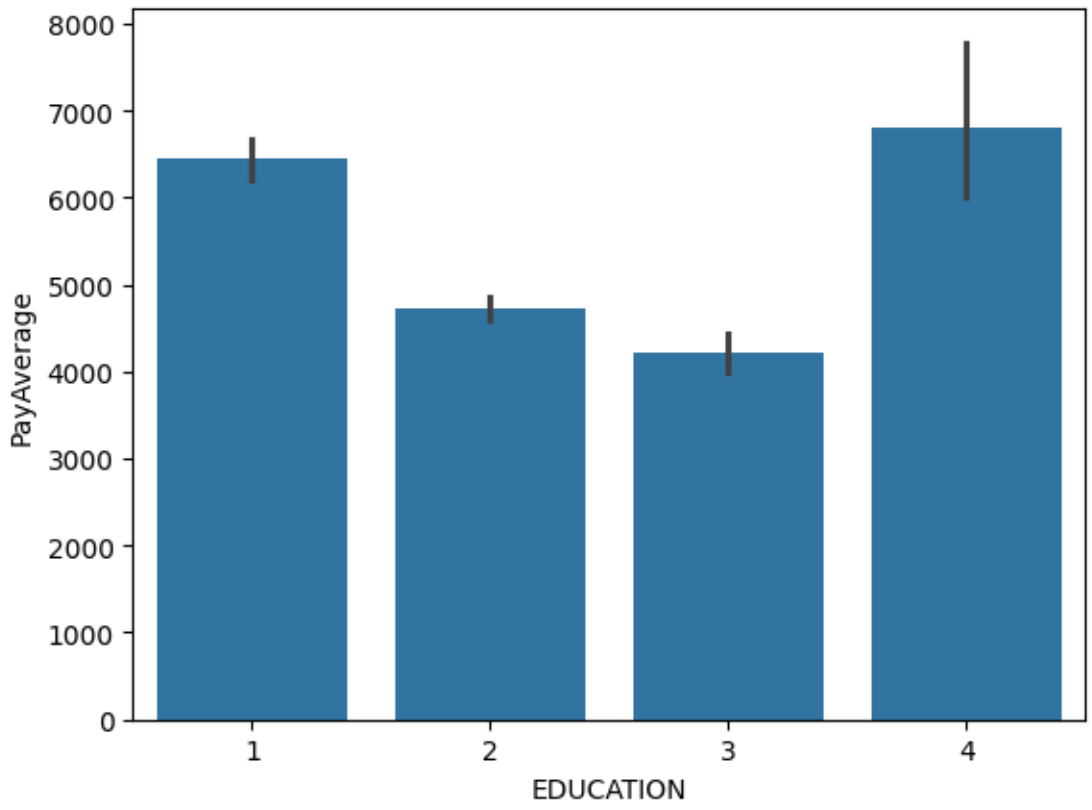


[257]:
```
data['EDUCATION'].value_counts(normalize=True) * 100
```

[257]:
```
EDUCATION
2    46.766667
1    35.283333
3    16.390000
4     1.560000
Name: proportion, dtype: float64
```

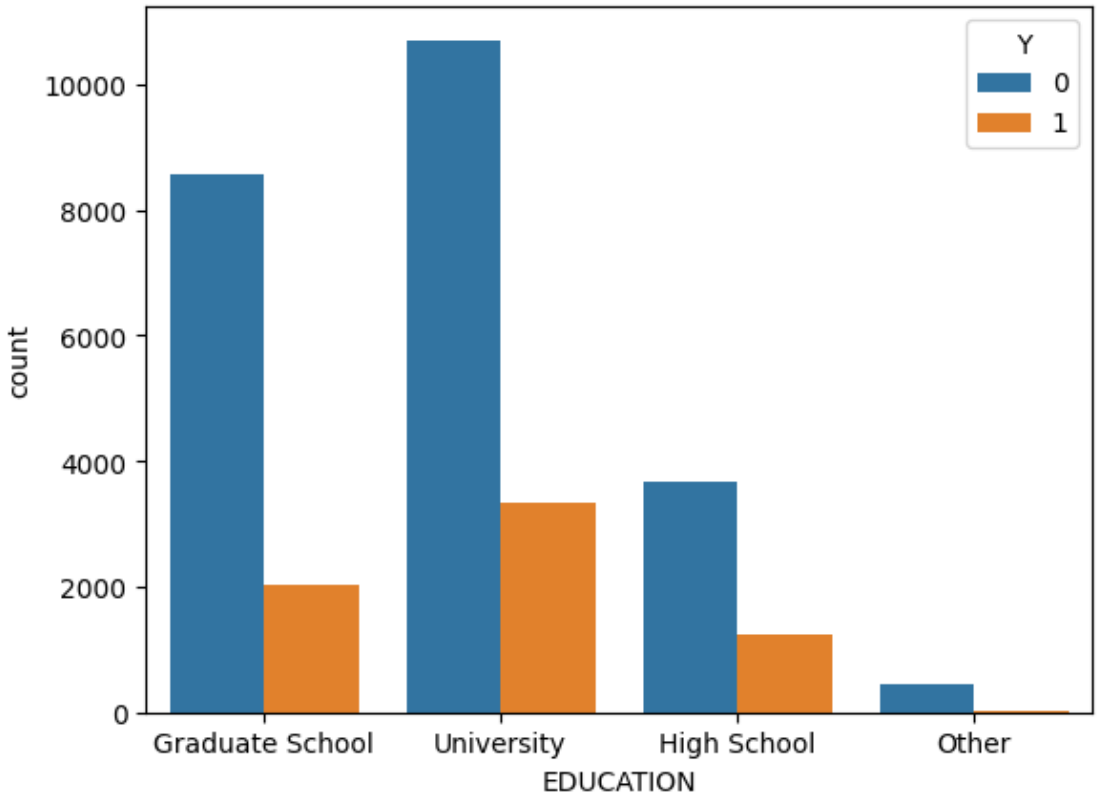
[258]:
```
# Count plot for education level
sns.countplot(x='EDUCATION', data=data)
plt.title('Count Plot of Education Level')
plt.show()

# Bar plot for Education Leval and PayAverage
sns.barplot(x='EDUCATION', y='PayAverage', data=data)
plt.show()
```


Count Plot of Education Level
14000
12000
10000
8000
6000
4000
2000
0
count
1
2
3
4
EDUCATION



[259]:
```
education = sns.countplot(x='EDUCATION', hue='Y', data=data)
education.set_xticklabels(['Graduate School','University','High␣
↪School','Other'])
plt.show()
```

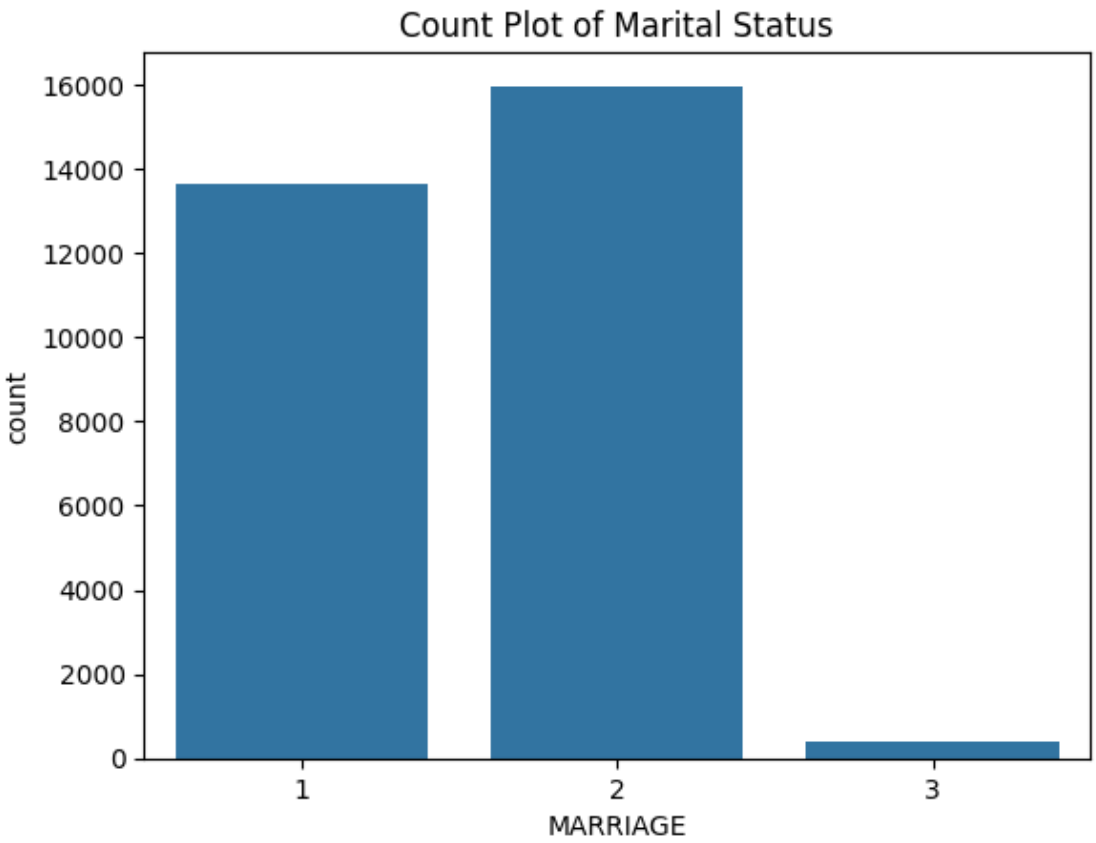
```
<ipython-input-259-34c8c53186c9>:2: UserWarning:

set_ticklabels() should only be used with a fixed number of ticks, i.e. after
set_ticks() or using a FixedLocator.
```

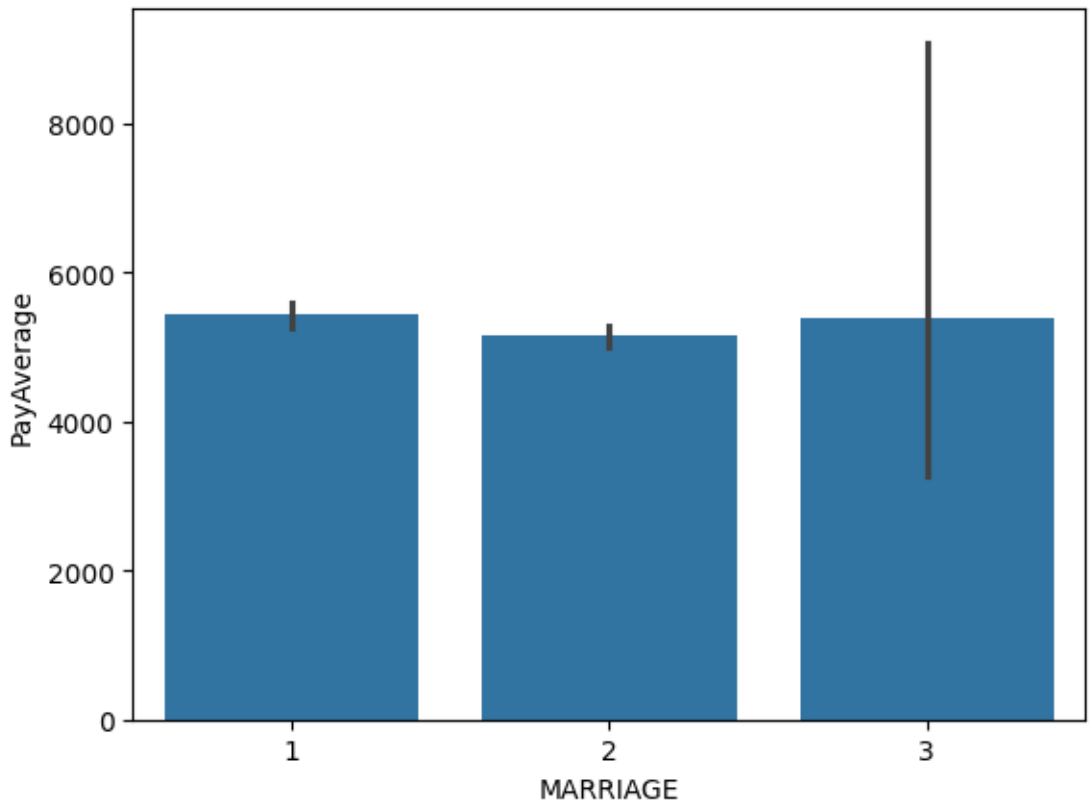


[260]:
```
data['MARRIAGE'].value_counts(normalize=True) * 100
```

[260]:
```
MARRIAGE
2    53.213333
1    45.530000
3     1.256667
Name: proportion, dtype: float64
```

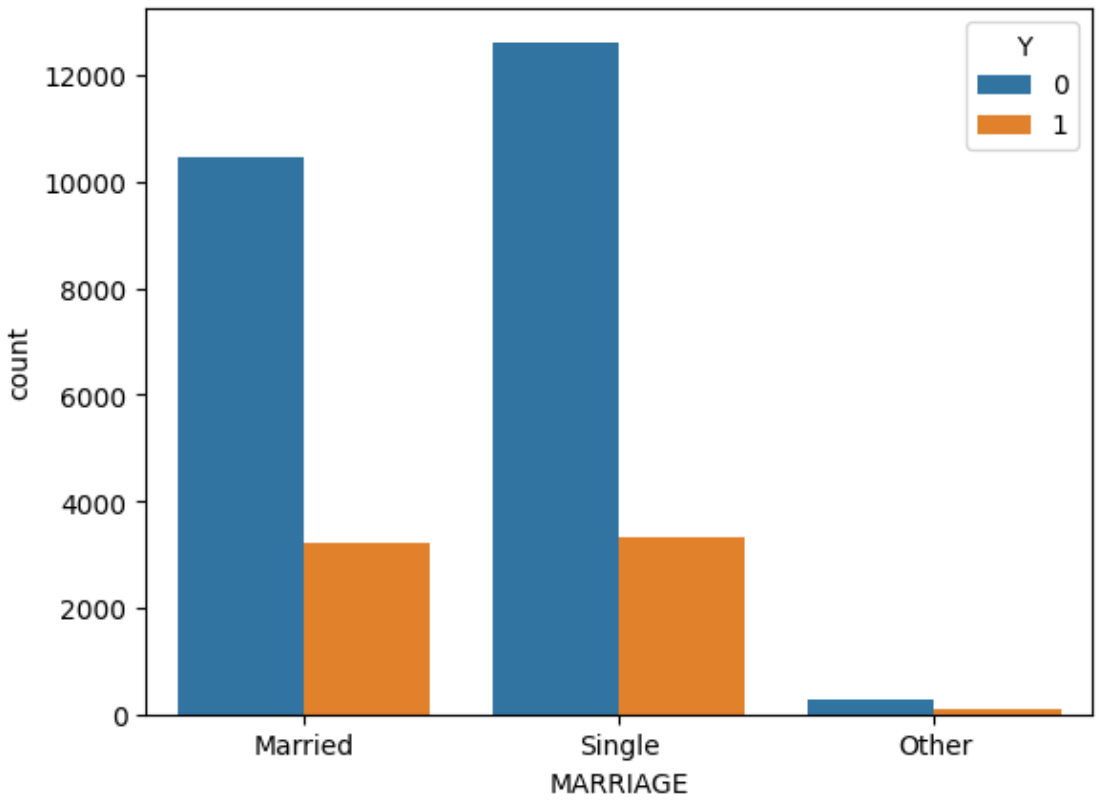
[261]:
```
# Count plot for marital status
sns.countplot(x='MARRIAGE', data=data)
plt.title('Count Plot of Marital Status')
plt.show()

# Bar plot for Education Leval and PayAverage
sns.barplot(x='MARRIAGE', y='PayAverage', data=data)
plt.show()
```


Count Plot of Marital Status
16000
14000
12000
10000
8000
6000
4000
2000
0
count
1
2
3
MARRIAGE



[262]:
```
marriage = sns.countplot(x='MARRIAGE', hue='Y', data=data)
marriage.set_xticklabels(['Married','Single', 'Other'])
plt.show()
```

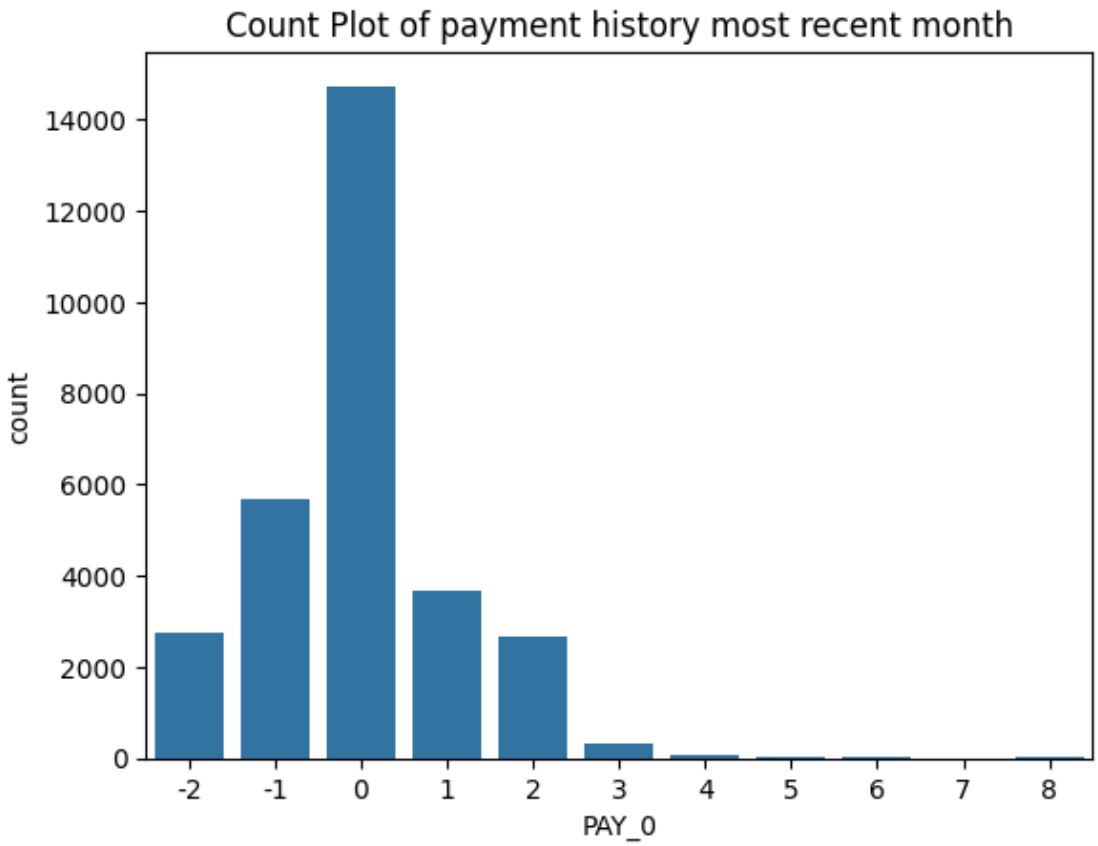
```
<ipython-input-262-b468166dd36f>:2: UserWarning:

set_ticklabels() should only be used with a fixed number of ticks, i.e. after
set_ticks() or using a FixedLocator.
```



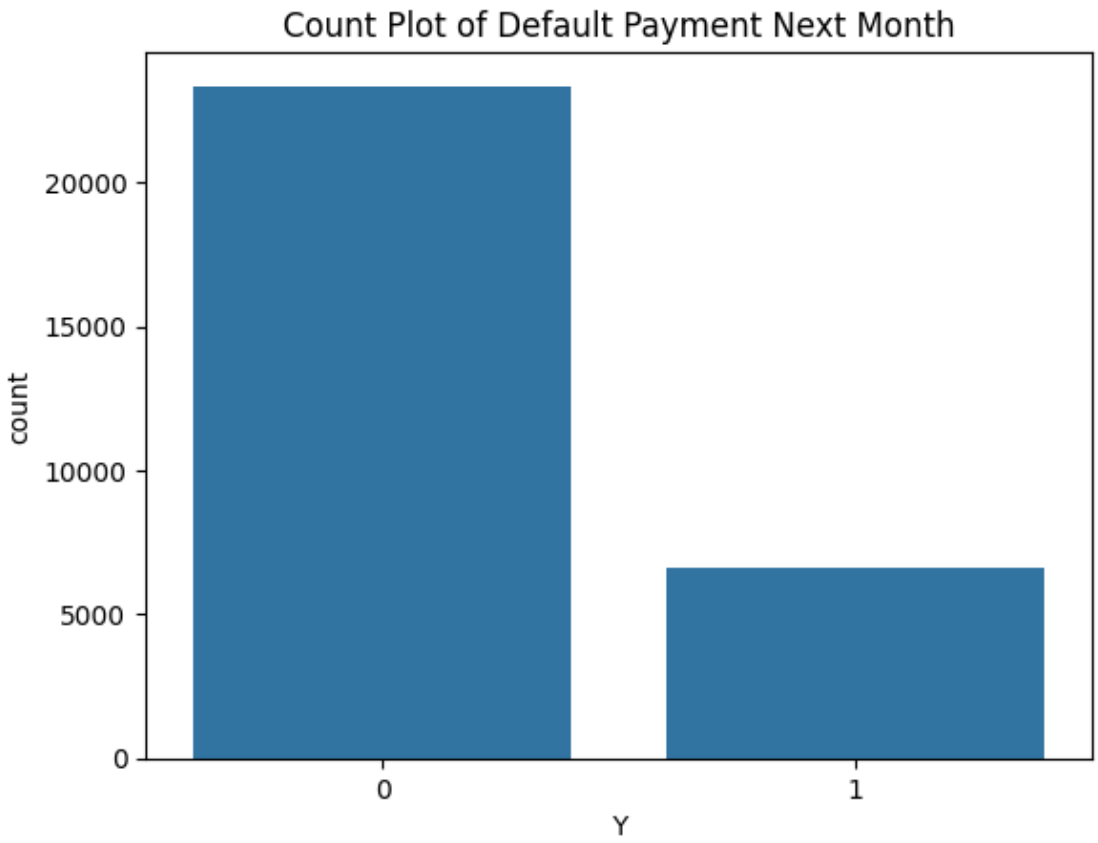
[263]:
```
data['PAY_0'].value_counts(normalize=True) * 100
```

[263]:
```
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
```

[264]:
```
# Count plot for payment history most recent month
sns.countplot(x='PAY_0', data=data)
plt.title('Count Plot of payment history most recent month')
plt.show()
```



[265]:
```
# Count plot for target variable:
sns.countplot(x='Y', data=data)
plt.title('Count Plot of Default Payment Next Month')
plt.show()
```



[266]:
```
data['Y'].value_counts(normalize=True) * 100
# Imbalanced dataset where class '1' (default payment = Yes) significantly␣
 ↪outnumbers
# class '0' (default payment = No).
```

[266]:
```
Y
0    77.88
1    22.12
Name: proportion, dtype: float64
```

[267]:
```
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',␣
 ↪'BillAverage', 'PayAverage']

#Calculating and printing the Pearson correlation matrix
print("Pearson Correlation Matrix of numeric variables:")
pearson_correlation_matrix = data[numeric_variables].corr().round(2)
pearson_correlation_matrix
```

Pearson Correlation Matrix of numeric variables:

[267]:

	LIMIT_BAL	AGE	BILL_AMT1	BILL_AMT2	BILL_AMT3	BILL_AMT4	\
LIMIT_BAL	1.00	0.14	0.29	0.28	0.28	0.29	
AGE	0.14	1.00	0.06	0.05	0.05	0.05	
BILL_AMT1	0.29	0.06	1.00	0.95	0.89	0.86	
BILL_AMT2	0.28	0.05	0.95	1.00	0.93	0.89	
BILL_AMT3	0.28	0.05	0.89	0.93	1.00	0.92	
BILL_AMT4	0.29	0.05	0.86	0.89	0.92	1.00	
BILL_AMT5	0.30	0.05	0.83	0.86	0.88	0.94	
BILL_AMT6	0.29	0.05	0.80	0.83	0.85	0.90	
PAY_AMT1	0.20	0.03	0.14	0.28	0.24	0.23	
PAY_AMT2	0.18	0.02	0.10	0.10	0.32	0.21	
PAY_AMT3	0.21	0.03	0.16	0.15	0.13	0.30	
PAY_AMT4	0.20	0.02	0.16	0.15	0.14	0.13	
PAY_AMT5	0.22	0.02	0.17	0.16	0.18	0.16	
PAY_AMT6	0.22	0.02	0.18	0.17	0.18	0.18	
BillAverage	0.30	0.05	0.94	0.96	0.96	0.96	
PayAverage	0.35	0.04	0.26	0.29	0.36	0.35	

	BILL_AMT5	BILL_AMT6	PAY_AMT1	PAY_AMT2	PAY_AMT3	PAY_AMT4	\
LIMIT_BAL	0.30	0.29	0.20	0.18	0.21	0.20	
AGE	0.05	0.05	0.03	0.02	0.03	0.02	
BILL_AMT1	0.83	0.80	0.14	0.10	0.16	0.16	
BILL_AMT2	0.86	0.83	0.28	0.10	0.15	0.15	
BILL_AMT3	0.88	0.85	0.24	0.32	0.13	0.14	
BILL_AMT4	0.94	0.90	0.23	0.21	0.30	0.13	
BILL_AMT5	1.00	0.95	0.22	0.18	0.25	0.29	
BILL_AMT6	0.95	1.00	0.20	0.17	0.23	0.25	
PAY_AMT1	0.22	0.20	1.00	0.29	0.25	0.20	
PAY_AMT2	0.18	0.17	0.29	1.00	0.24	0.18	
PAY_AMT3	0.25	0.23	0.25	0.24	1.00	0.22	
PAY_AMT4	0.29	0.25	0.20	0.18	0.22	1.00	
PAY_AMT5	0.14	0.31	0.15	0.18	0.16	0.15	
PAY_AMT6	0.16	0.12	0.19	0.16	0.16	0.16	
BillAverage	0.95	0.93	0.23	0.19	0.21	0.19	
PayAverage	0.36	0.36	0.60	0.67	0.59	0.53	

	PAY_AMT5	PAY_AMT6	BillAverage	PayAverage
LIMIT_BAL	0.22	0.22	0.30	0.35
AGE	0.02	0.02	0.05	0.04
BILL_AMT1	0.17	0.18	0.94	0.26
BILL_AMT2	0.16	0.17	0.96	0.29
BILL_AMT3	0.18	0.18	0.96	0.36
BILL_AMT4	0.16	0.18	0.96	0.35
BILL_AMT5	0.14	0.16	0.95	0.36
BILL_AMT6	0.31	0.12	0.93	0.36

PAY_AMT1	0.15	0.19	0.23	0.60
PAY_AMT2	0.18	0.16	0.19	0.67
PAY_AMT3	0.16	0.16	0.21	0.59
PAY_AMT4	0.15	0.16	0.19	0.53
PAY_AMT5	1.00	0.15	0.19	0.49
PAY_AMT6	0.15	1.00	0.18	0.53
BillAverage	0.19	0.18	1.00	0.34
PayAverage	0.49	0.53	0.34	1.00

[268]:
```
# Printing summary statistics of numeric variables
print("\nSummary Statistics of numeric variables:")
data[numeric_variables].describe().transpose()
```

```
Summary Statistics of numeric variables:
```

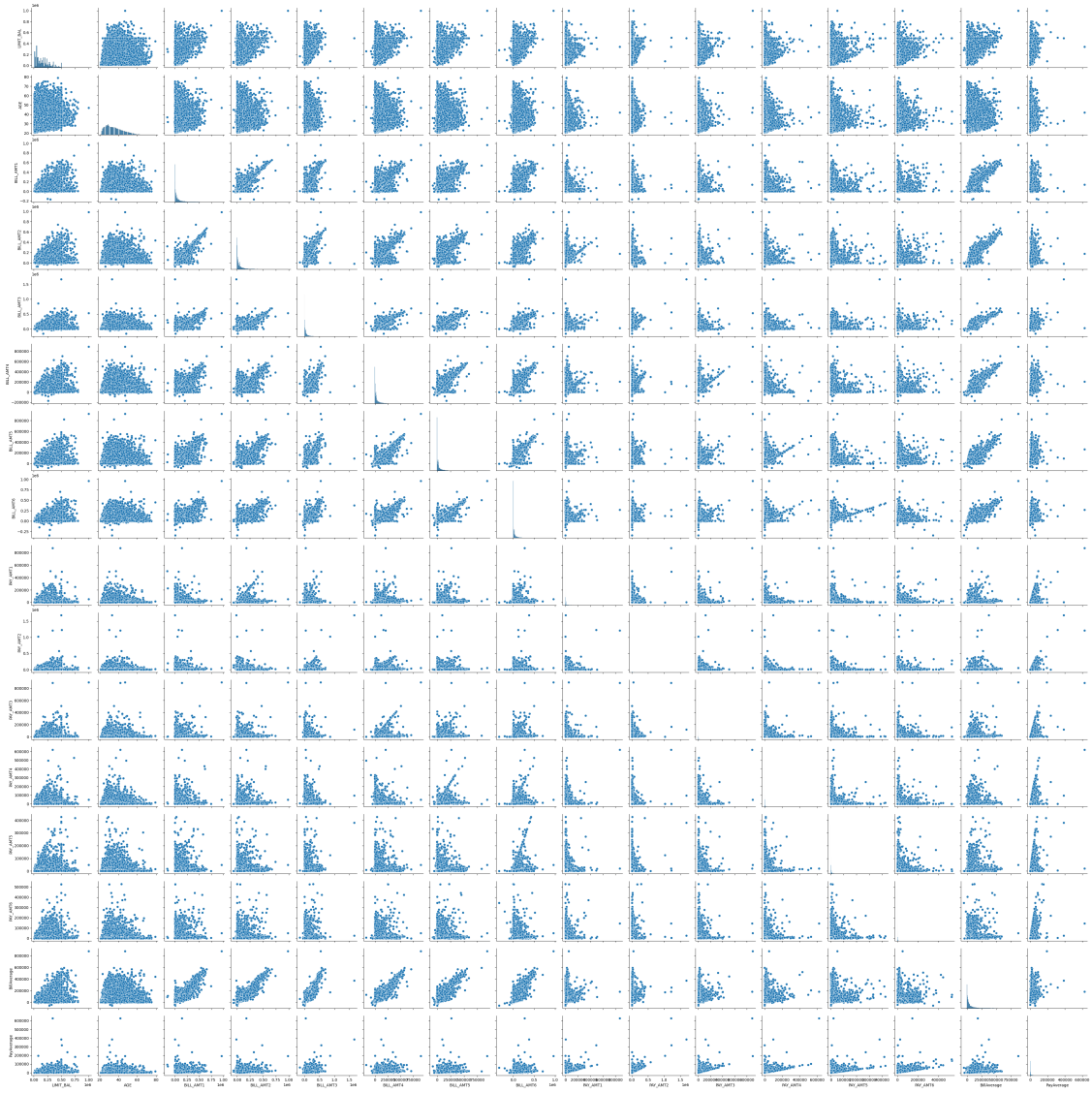
[268]:

	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	
BillAverage	30000.0	44976.943700	63260.722001	-56043.0	4781.75	
PayAverage	30000.0	5275.231633	10137.946665	0.0	1113.00	

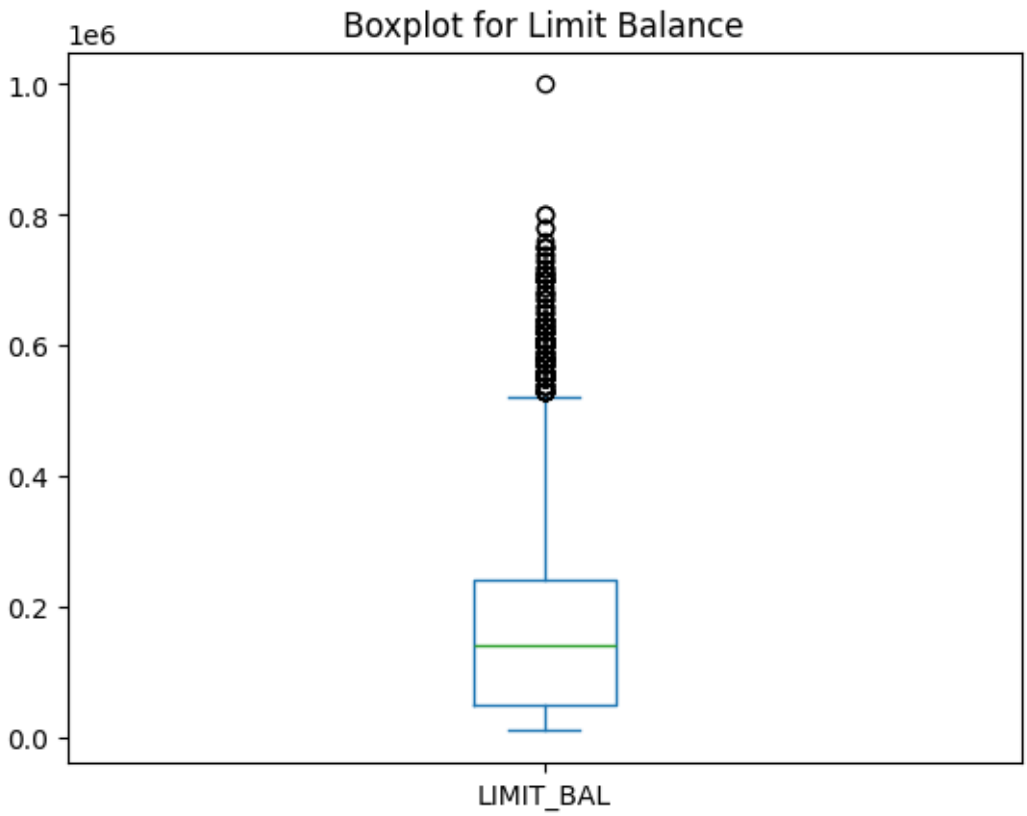
	50%	75%	max
LIMIT_BAL	140000.0	240000.00	1000000.0
AGE	34.0	41.00	79.0
BILL_AMT1	22381.5	67091.00	964511.0
BILL_AMT2	21200.0	64006.25	983931.0
BILL_AMT3	20088.5	60164.75	1664089.0
BILL_AMT4	19052.0	54506.00	891586.0
BILL_AMT5	18104.5	50190.50	927171.0
BILL_AMT6	17071.0	49198.25	961664.0
PAY_AMT1	2100.0	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
BillAverage    21052.0    57104.25    877314.0
PayAverage      2397.5     5584.00    627344.0
```

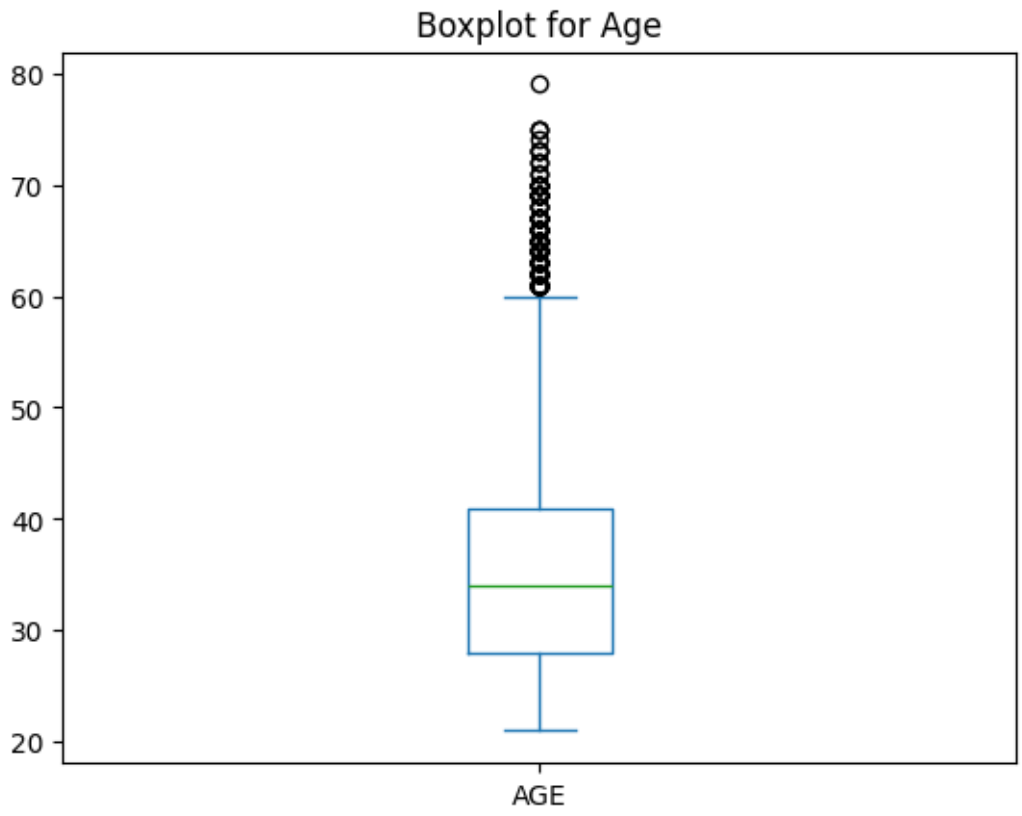
[281]:
```
import matplotlib.pyplot as plt

sns.pairplot(data[numeric_variables])
plt.show()
```



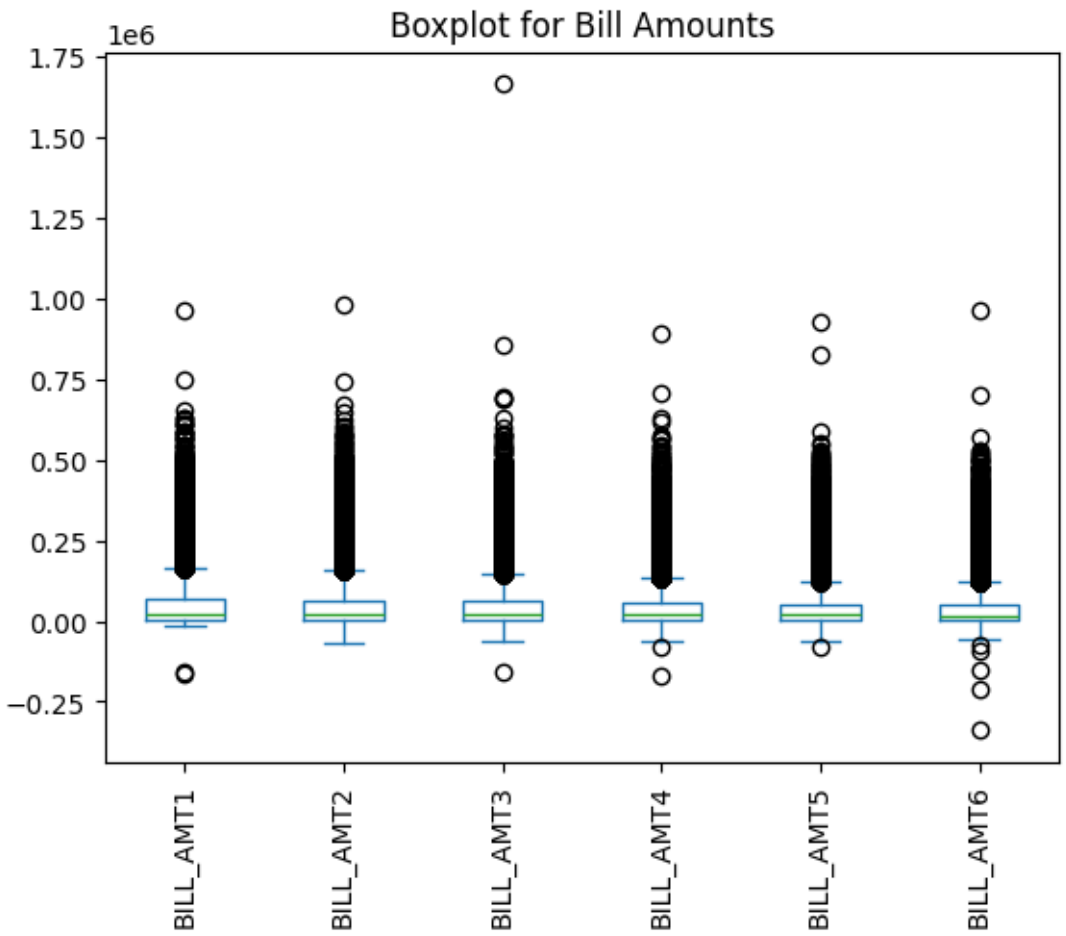
[286]:
```
data['LIMIT_BAL'].plot(kind='box')
plt.title('Boxplot for Limit Balance')
plt.show()
```



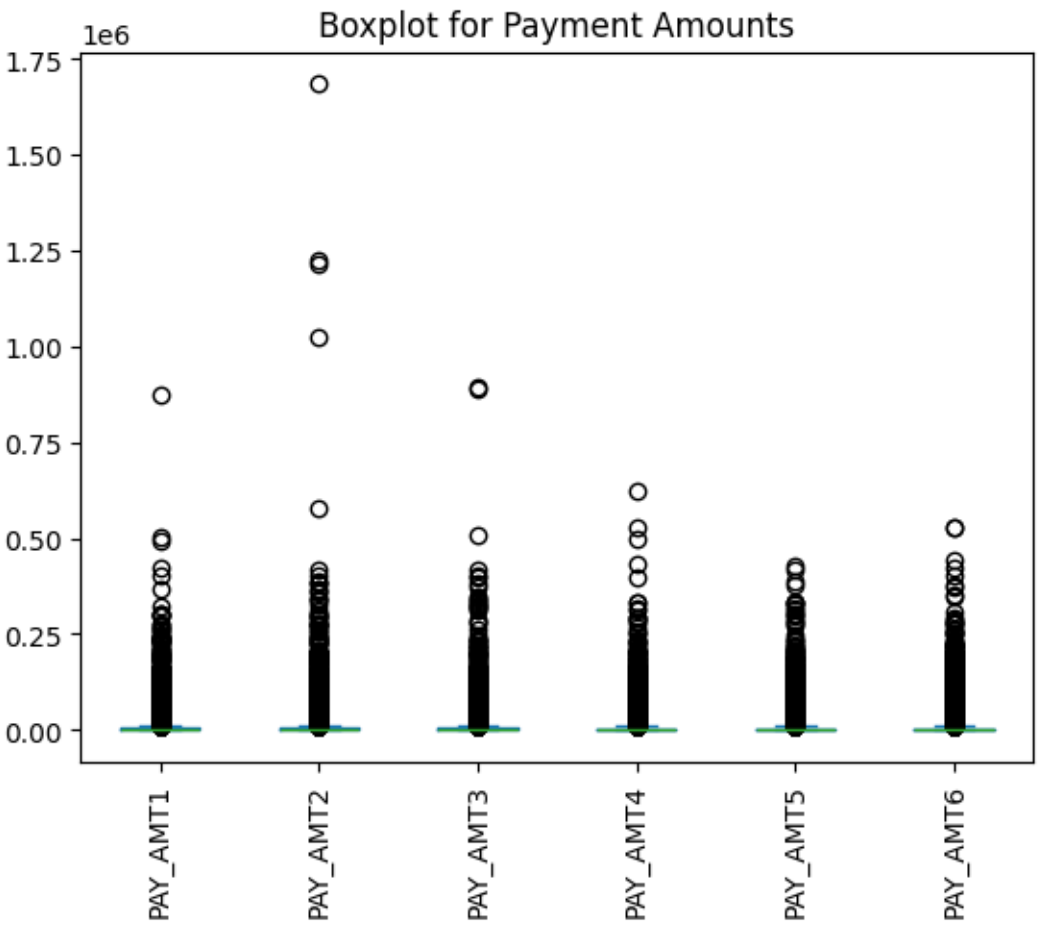
[287]:
```
data['AGE'].plot(kind='box')
plt.title('Boxplot for Age')
plt.show()
```



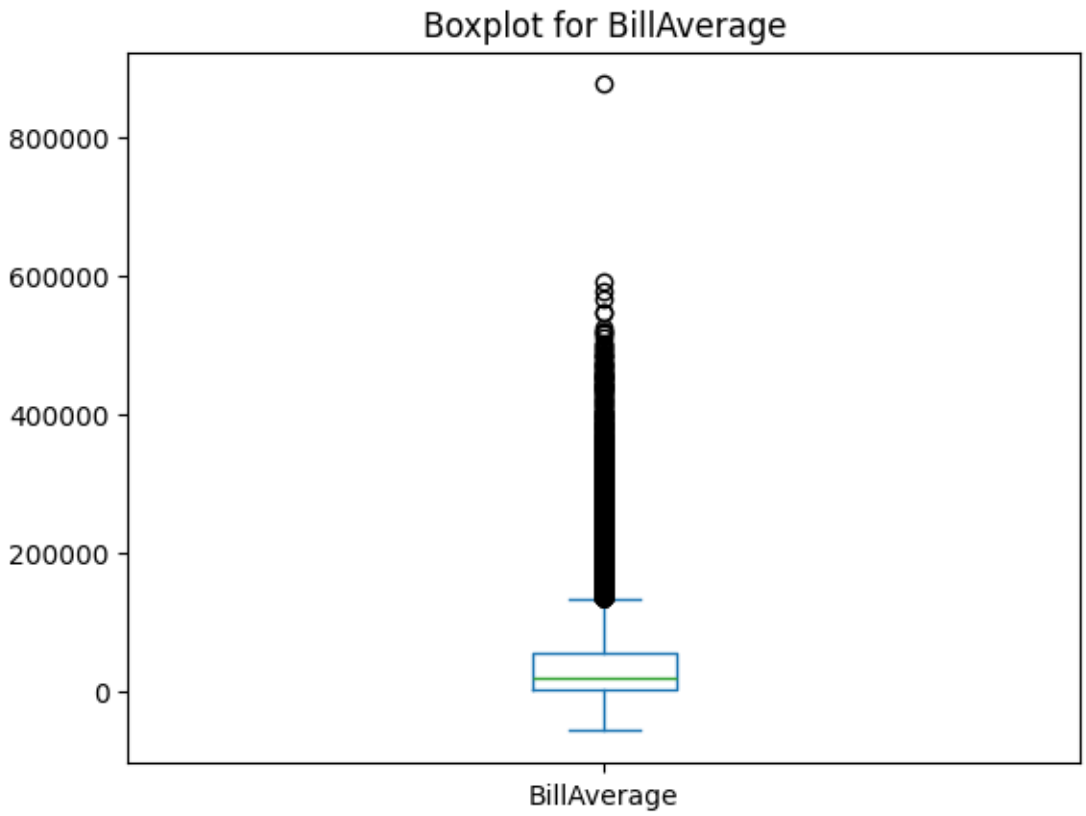
[288]:
```
bill_variables = ['BILL_AMT1','BILL_AMT2', 'BILL_AMT3', 'BILL_AMT4',␣
 ↪'BILL_AMT5', 'BILL_AMT6']
data[bill_variables].plot(kind='box')
plt.title('Boxplot for Bill Amounts')
plt.xticks(rotation=90)
plt.show()
```



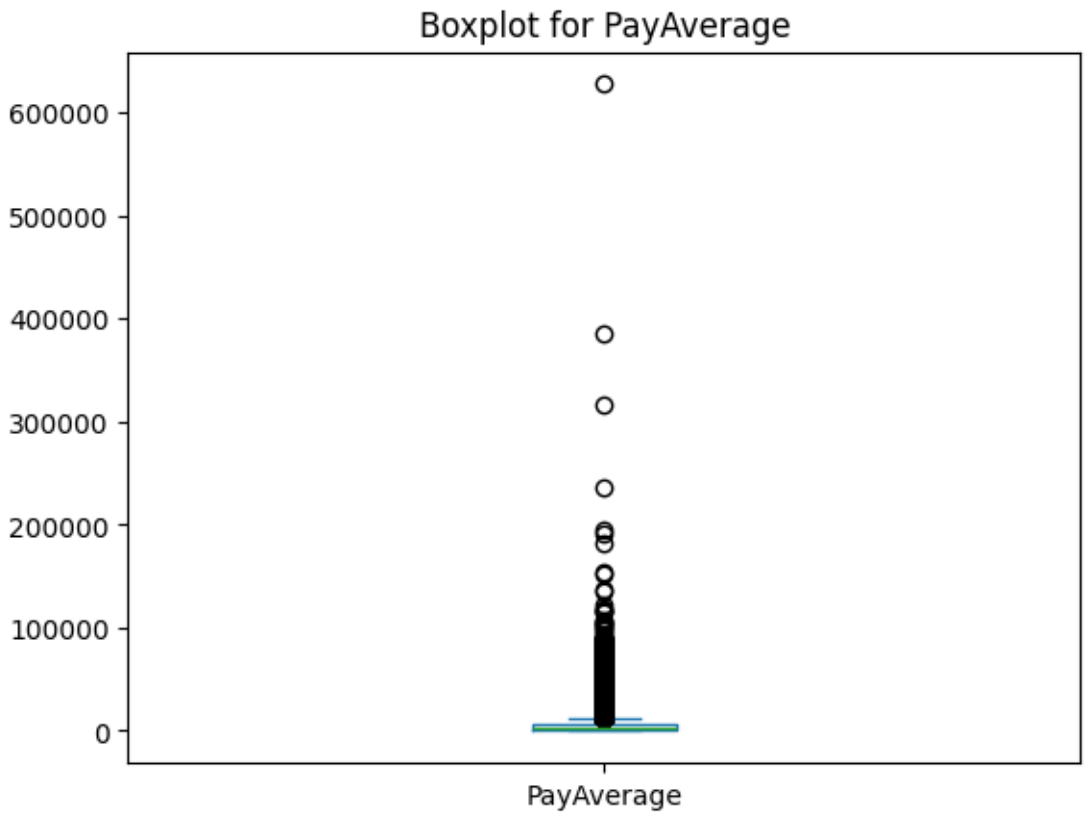
[289]:
```
pay_variables = ['PAY_AMT1','PAY_AMT2', 'PAY_AMT3', 'PAY_AMT4', 'PAY_AMT5',␣
  ↪'PAY_AMT6']
data[pay_variables].plot(kind='box')
plt.title('Boxplot for Payment Amounts')
plt.xticks(rotation=90)
plt.show()
```



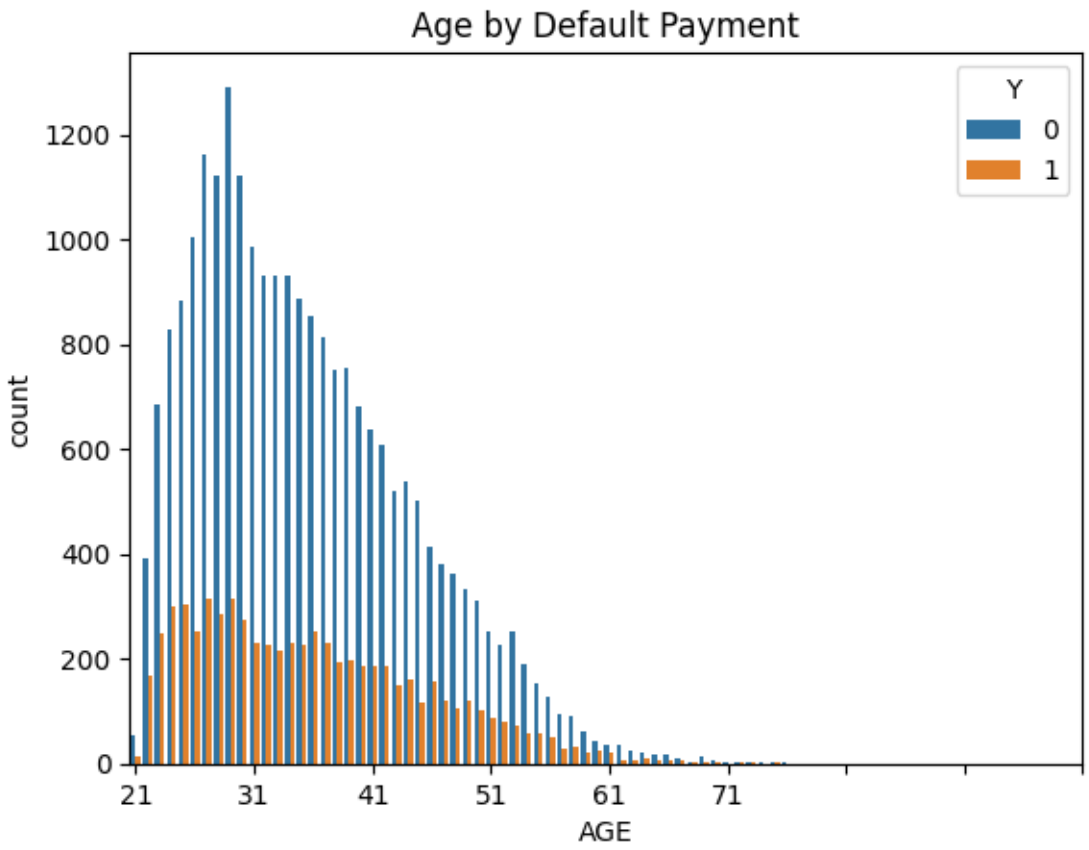
[290]:
```
data['BillAverage'].plot(kind='box')
plt.title('Boxplot for BillAverage')
plt.show()
```



[291]:
```
data['PayAverage'].plot(kind='box')
plt.title('Boxplot for PayAverage')
plt.show()
```

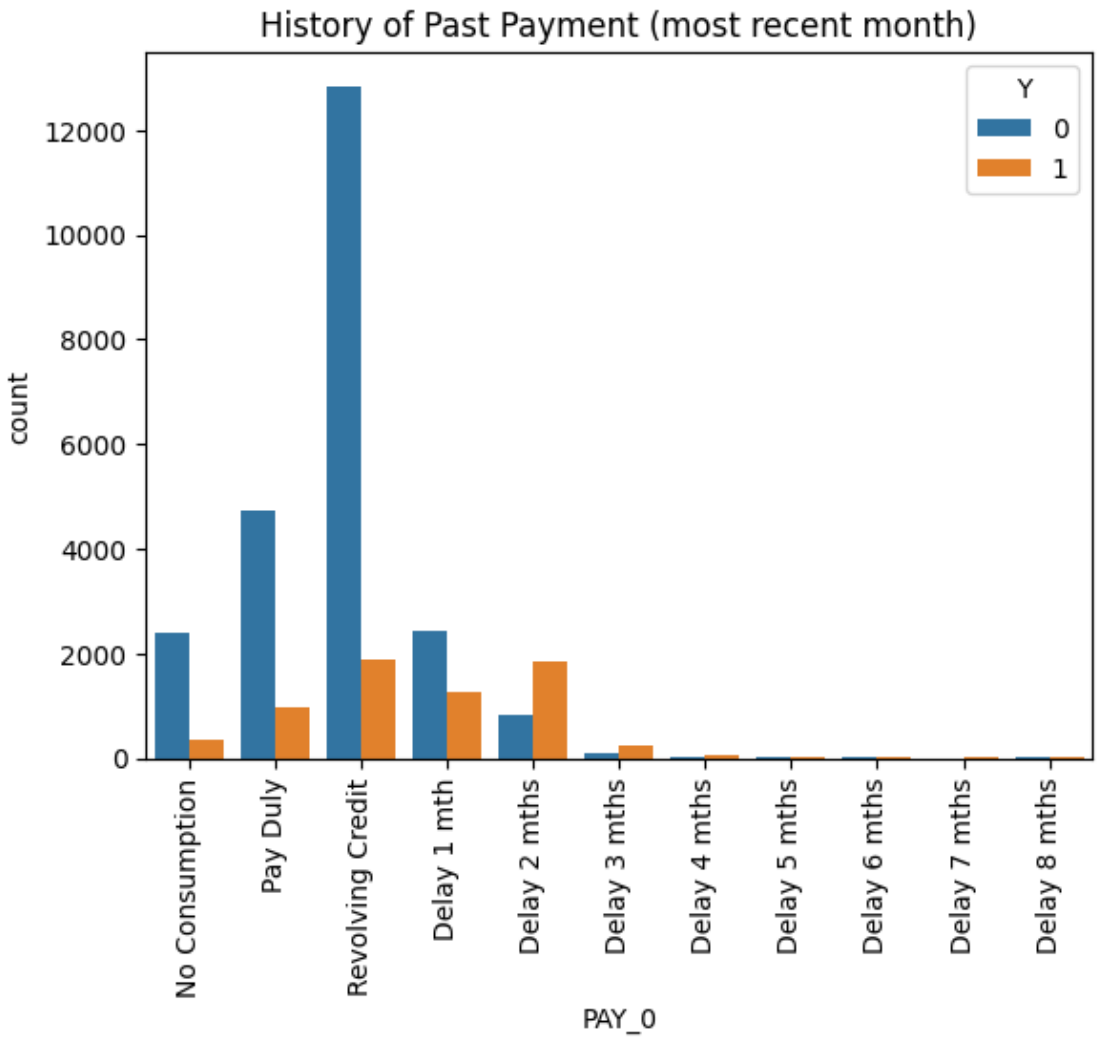


[292]:
```
import numpy as np
age = sns.countplot(x='AGE', hue='Y', data=data)
age.set_title('Age by Default Payment')
plt.xticks(np.arange(0, 90, step=10))
plt.show()
```



[293]:
```
pay0 = sns.countplot(x="PAY_0", hue='Y', data=data)
pay0.set_xticklabels(['No Consumption','Pay Duly','Revolving Credit','Delay 1␣
↪mth','Delay 2 mths','Delay 3 mths','Delay 4 mths','Delay 5 mths','Delay 6␣
↪mths','Delay 7 mths','Delay 8 mths'])
pay0.set_title('History of Past Payment (most recent month)')
plt.xticks(rotation=90)
plt.show()
```

```
<ipython-input-293-ddfd239ade31>:2: UserWarning:

set_ticklabels() should only be used with a fixed number of ticks, i.e. after
set_ticks() or using a FixedLocator.
```



[294]:
```
import plotly.express as px
billaverage = px.histogram(data, x='BillAverage', nbins=100, color='Y',␣
 ↪range_x=[-15000, 350000])
billaverage.show()
```

[295]:
```
payaverage = px.histogram(data, x='PayAverage', nbins=100, color='Y',␣
 ↪range_x=[0, 60000])
payaverage.show()
```

[296]:
```
!apt-get install -y texlive-xetex texlive-fonts-recommended texlive-latex-extra
!apt-get install texlive texlive-latex-extra pandoc
!jupyter nbconvert --to pdf "/content/Big_Data_Analytics_Project_EDA_Mar16.
 ↪ipynb"
```

```
Reading package lists… Done
```

```
Building dependency tree… Done
Reading state information… Done
texlive-fonts-recommended is already the newest version (2021.20220204-1).
texlive-latex-extra is already the newest version (2021.20220204-1).
texlive-xetex is already the newest version (2021.20220204-1).
0 upgraded, 0 newly installed, 0 to remove and 29 not upgraded.
Reading package lists… Done
Building dependency tree… Done
Reading state information… Done
pandoc is already the newest version (2.9.2.1-3ubuntu2).
texlive is already the newest version (2021.20220204-1).
texlive-latex-extra is already the newest version (2021.20220204-1).
0 upgraded, 0 newly installed, 0 to remove and 29 not upgraded.
[NbConvertApp] Converting notebook
/content/Big_Data_Analytics_Project_EDA_Mar16.ipynb to pdf
[NbConvertApp] Support files will be in
Big_Data_Analytics_Project_EDA_Mar16_files/
[NbConvertApp] Making directory ./Big_Data_Analytics_Project_EDA_Mar16_files
[NbConvertApp] Writing 77330 bytes to notebook.tex
[NbConvertApp] Building PDF
[NbConvertApp] Running xelatex 3 times: ['xelatex', 'notebook.tex', '-quiet']
[NbConvertApp] Running bibtex 1 time: ['bibtex', 'notebook']
[NbConvertApp] WARNING | bibtex had problems, most likely because there were no
citations
[NbConvertApp] PDF successfully created
[NbConvertApp] Writing 2643544 bytes to
/content/Big_Data_Analytics_Project_EDA_Mar16.pdf
```