In [62]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
pd.pandas.set_option('display.max_columns',None)
```

In [63]:

```
card = pd.read_excel('default_of_credit_card_clients_0.xlsx')
```

In [64]:

```
card.head()
```

Out[64]:

	ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_0	PAY_2	PAY_3	PAY_4	PAY_
0	1	20000	2	2	1	24	2	2	0	0	
1	2	120000	2	2	2	26	0	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	0	0	0	0	

In [65]:

```
card.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  default payment next month  30000 non-null  int64
dtypes: int64(25)
memory usage: 5.7 MB
```

In [66]:

```
card.isnull()
```

Out[66]:

	ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_0	PAY_2	PAY_3	PAY
0	False	False	False	False	False	False	False	False	False	Fa
1	False	False	False	False	False	False	False	False	False	Fa
2	False	False	False	False	False	False	False	False	False	Fa
3	False	False	False	False	False	False	False	False	False	Fa
4	False	False	False	False	False	False	False	False	False	Fa
...	...	...	...	...	...	...	...	...	...	
29995	False	False	False	False	False	False	False	False	False	Fa
29996	False	False	False	False	False	False	False	False	False	Fa
29997	False	False	False	False	False	False	False	False	False	Fa
29998	False	False	False	False	False	False	False	False	False	Fa
29999	False	False	False	False	False	False	False	False	False	Fa

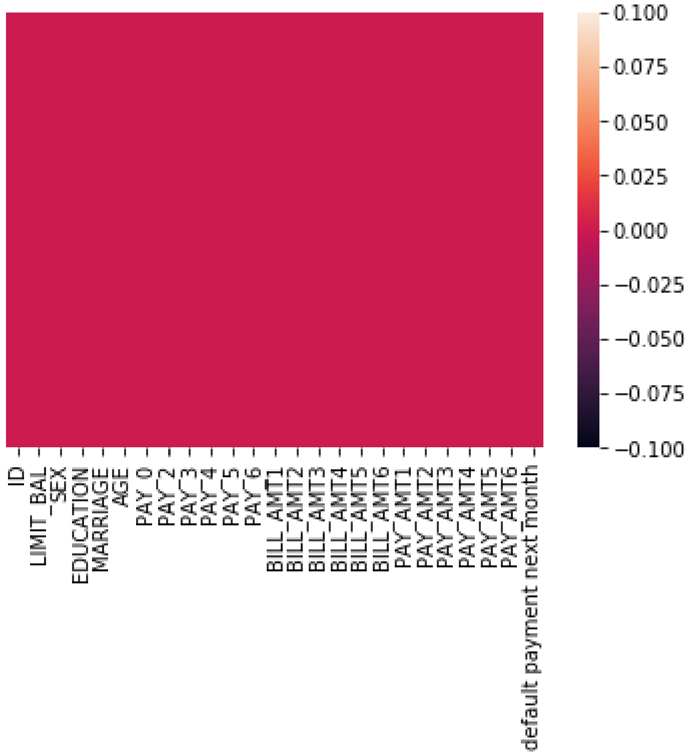
30000 rows × 25 columns

In [67]:

```
sns.heatmap(card.isnull(), yticklabels = False, linecolor = 'yellow')
```

Out[67]:

```
<matplotlib.axes._subplots.AxesSubplot at 0xc134488>
```



In [68]:

```
card.columns
```

Out[68]:

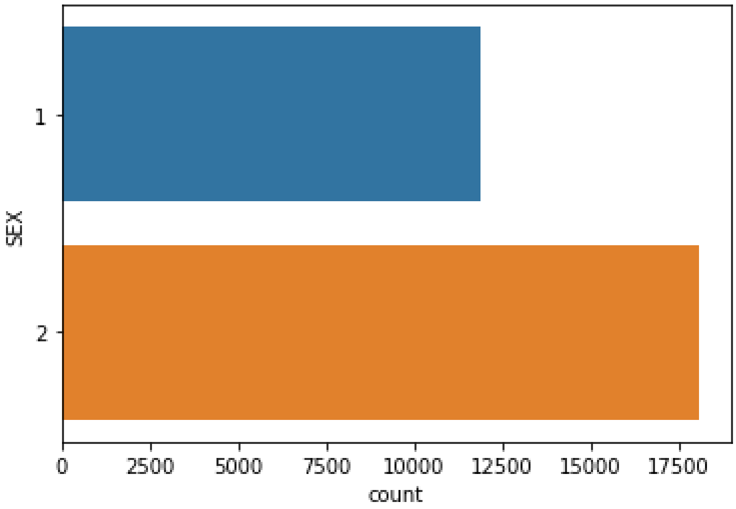
```
Index(['ID', 'LIMIT_BAL', 'SEX', 'EDUCATION', 'MARRIAGE', 'AGE', 'PAY_0',
       'PAY_2', 'PAY_3', 'PAY_4', 'PAY_5', 'PAY_6', 'BILL_AMT1', 'BILL_AMT
2',
       'BILL_AMT3', 'BILL_AMT4', 'BILL_AMT5', 'BILL_AMT6', 'PAY_AMT1',
       'PAY_AMT2', 'PAY_AMT3', 'PAY_AMT4', 'PAY_AMT5', 'PAY_AMT6',
       'default payment next month'],
      dtype='object')
```

In [69]:

```
sns.countplot(y='SEX',data=card)
```

Out[69]:

```
<matplotlib.axes._subplots.AxesSubplot at 0xb2a0448>
```



- 1 indicates male
- 2 indicates female

In [76]:

```
#One-Hot Encoding
sex_dum = pd.get_dummies(card['SEX'])
```

In [77]:

```
sex_dum
```

Out[77]:

	1	2
0	0	1
1	0	1
2	0	1
3	0	1
4	1	0
...	...	...
29995	1	0
29996	1	0
29997	1	0
29998	1	0
29999	1	0

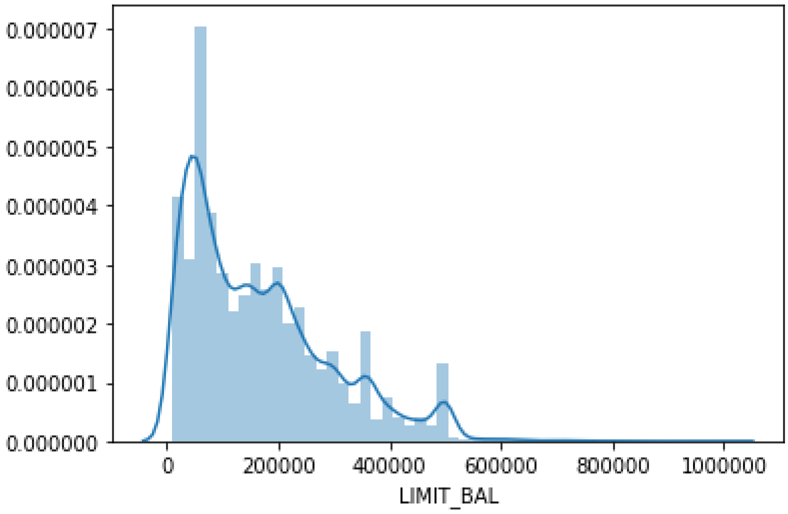
29655 rows × 2 columns

In [72]:

```
sns.distplot(card['LIMIT_BAL'])
```

Out[72]:

```
<matplotlib.axes._subplots.AxesSubplot at 0xb95a548>
```



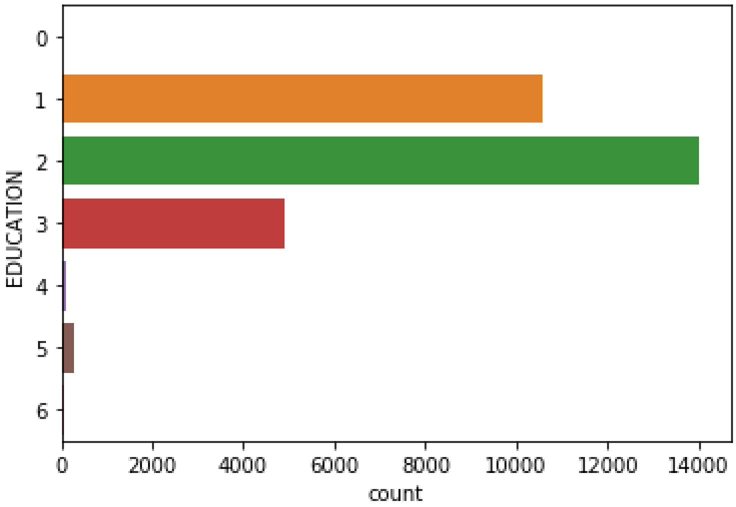
- the graph shows the amount given to customers(NT dollar)

In [73]:

```
sns.countplot(y='EDUCATION',data=card)
```

Out[73]:

```
<matplotlib.axes._subplots.AxesSubplot at 0xd292548>
```



- 1 implies graduate school
- 2 implies university
- 3 implies high school
- 4 implies others
- 5 and 6 are errors in the given dataset

In [74]:

```
# delete all rows with column 'education'
indexNames = card[( card['EDUCATION'] >= 5) & (card['EDUCATION'] <= 6)  ].index
card.drop(indexNames , inplace=True)

indexName = card[( card['EDUCATION'] == 0)].index
card.drop(indexName, inplace=True)
```

In [75]:

```
card['EDUCATION'].unique()
```

Out[75]:

```
array([2, 1, 3, 4], dtype=int64)
```

In [78]:

```
#One-Hot Encoding
edu_dum = pd.get_dummies(card['EDUCATION'])
```

In [79]:

```
edu_dum
```

Out[79]:

	1	2	3	4
0	0	1	0	0
1	0	1	0	0
2	0	1	0	0
3	0	1	0	0
4	0	1	0	0
...	...	...	...	...
29995	0	0	1	0
29996	0	0	1	0
29997	0	1	0	0
29998	0	0	1	0
29999	0	1	0	0

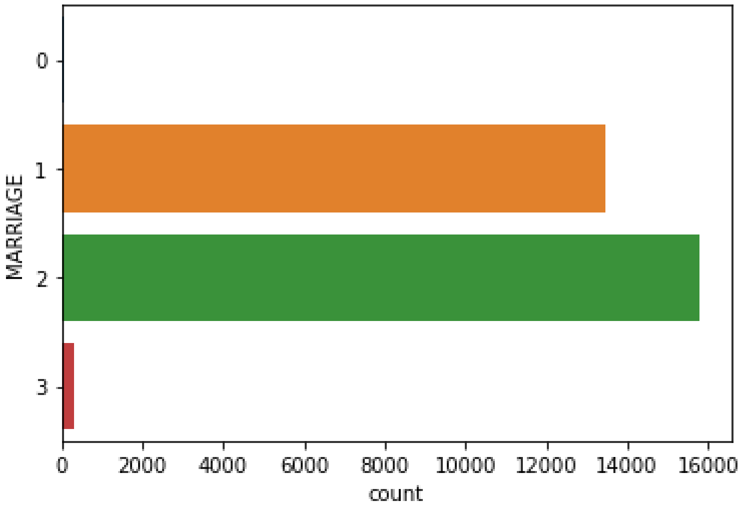
29655 rows × 4 columns

In [80]:

```
sns.countplot(y='MARRIAGE',data=card)
```

Out[80]:

```
<matplotlib.axes._subplots.AxesSubplot at 0xf29a9c8>
```



- 1 implies married
- 2 implies single
- 3 implies others
- 0 is the error from the given dataset

In [81]:

```
card['MARRIAGE'].unique()
```

Out[81]:

```
array([1, 2, 3, 0], dtype=int64)
```

In [82]:

```
index = card[( card['MARRIAGE'] == 0)].index
card.drop(index, inplace=True)
```

In [83]:

```
card['MARRIAGE'].unique()
```

Out[83]:

```
array([1, 2, 3], dtype=int64)
```

In [84]:

```
#One-Hot Encoding
marr_dum = pd.get_dummies(card['MARRIAGE'])
```

In [85]:

```
marr_dum
```

Out[85]:

	1	2	3
0	1	0	0
1	0	1	0
2	0	1	0
3	1	0	0
4	1	0	0
...	...	...	...
29995	1	0	0
29996	0	1	0
29997	0	1	0
29998	1	0	0
29999	1	0	0

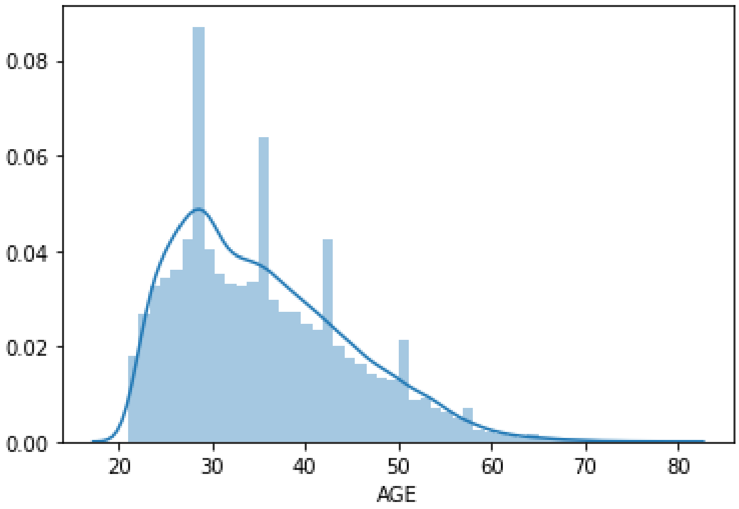
29601 rows × 3 columns

In [86]:

```
sns.distplot(card['AGE'])
```

Out[86]:

```
<matplotlib.axes._subplots.AxesSubplot at 0xddf40c8>
```

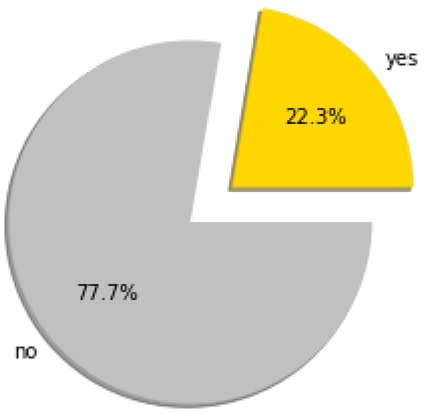


- age group shows from 20 to 70 yrs
- majority of age grp between 20 - 30 yrs from the above graph

In [87]:

```
labels = ['yes','no']
sizes = []
sizes.append(list(card['default payment next month'].value_counts())[1])
sizes.append(list(card['default payment next month'].value_counts())[0])
colors = ['gold','silver']

plt.pie(sizes, explode=(0.3,0), labels=labels, colors=colors, autopct='%1.1f%%', shadow
= True)
plt.title('Percentage of those customers who have to pay default payment next month')
plt.axis('equal')
plt.show()
```

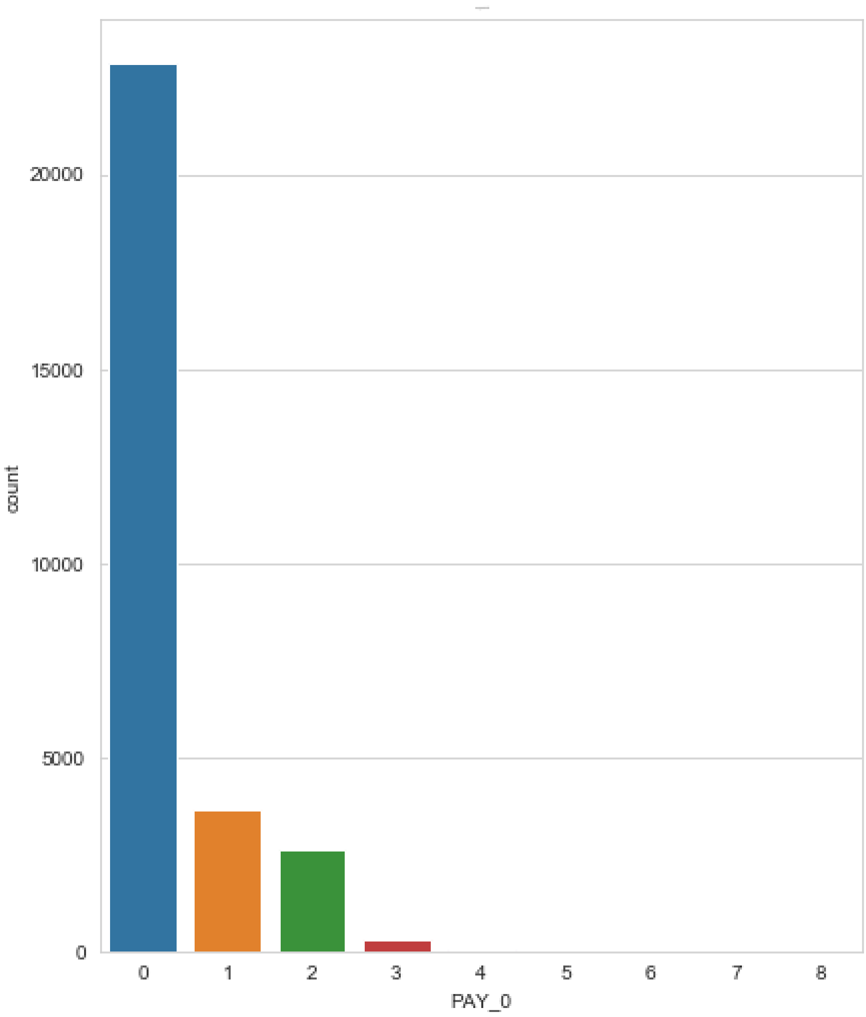
Percentage of those customers who have to pay default payment next month

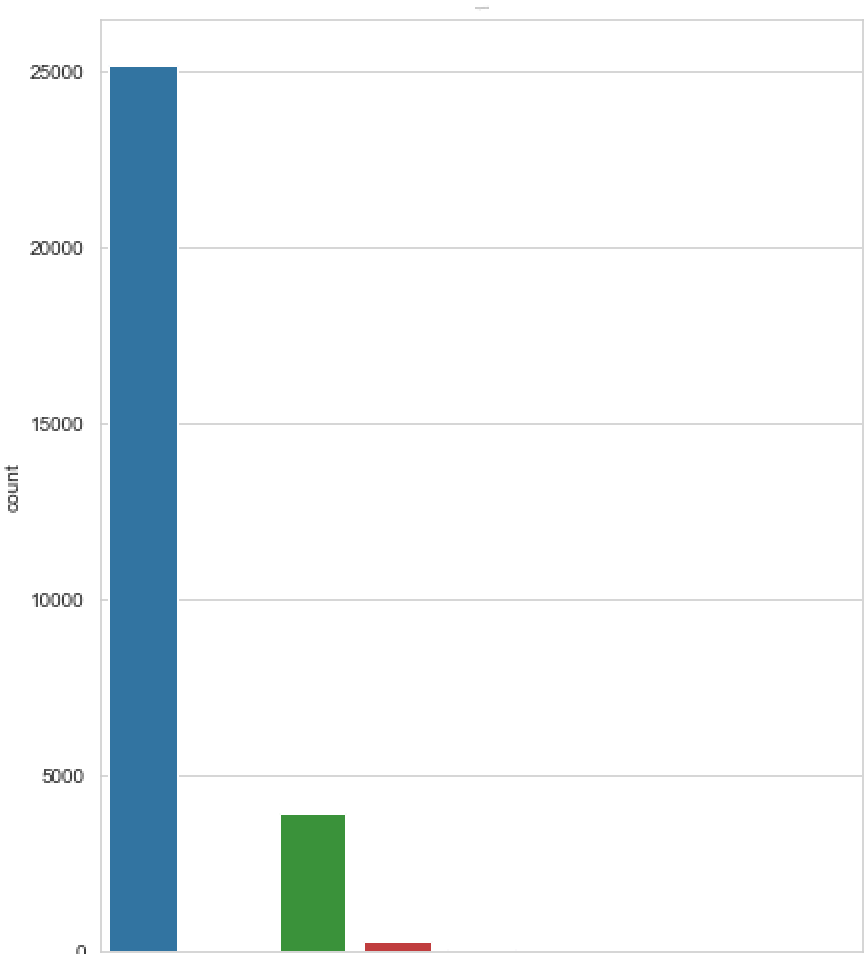


- pie chart shows that 22.1% of customers have to pay default payment next month

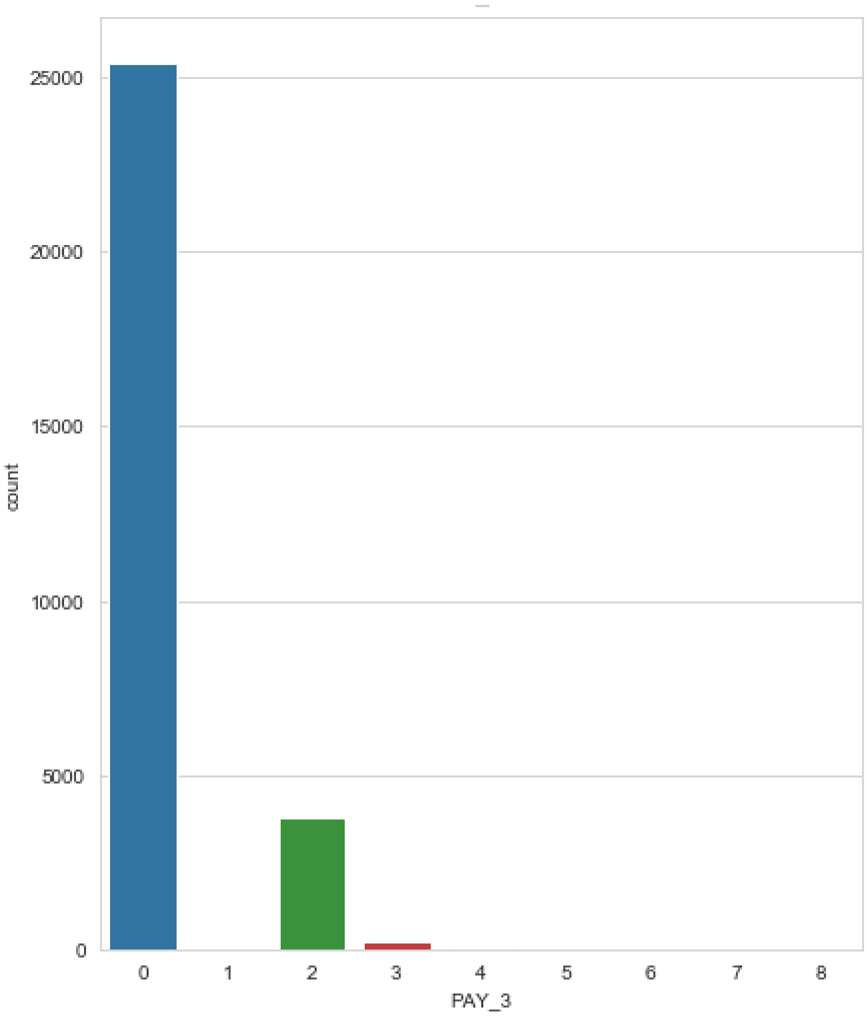
In [88]:

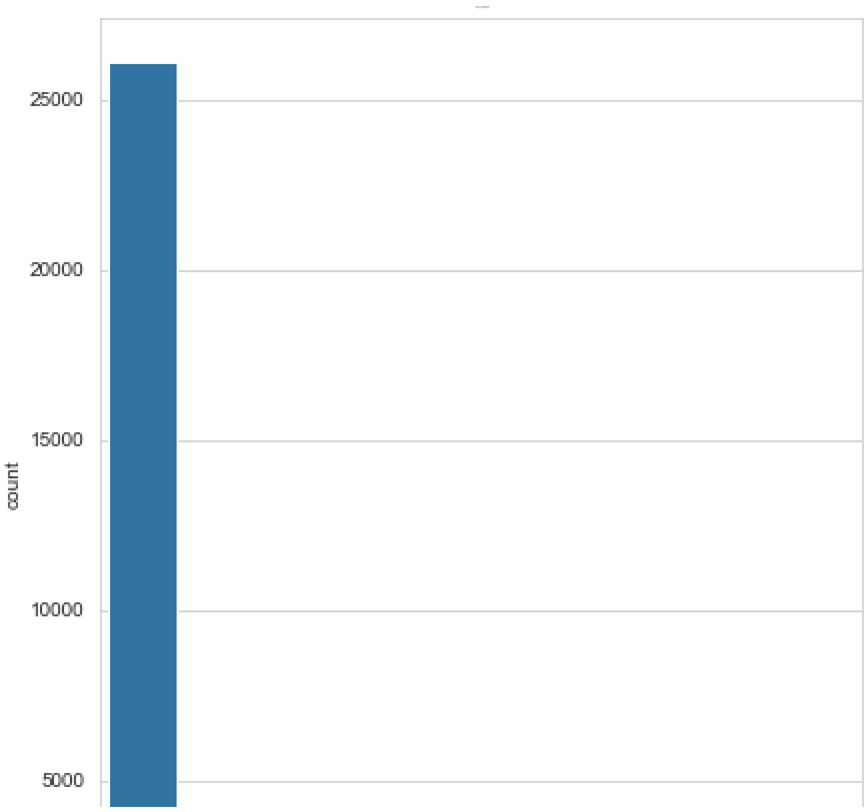
```
his_of_pas_pay = ['PAY_0','PAY_2', 'PAY_3', 'PAY_4', 'PAY_5', 'PAY_6']
plt.figure(figsize=(15,60))

for i,item in enumerate(his_of_pas_pay):
    x_axis = item
    fig_num = 2*i+1
    sns.set_style('whitegrid')
    plt.subplot(6,2,fig_num)
    plt.title(x_axis + 'count plot', fontsize = 0.5)
    sns.countplot(card[x_axis])
```

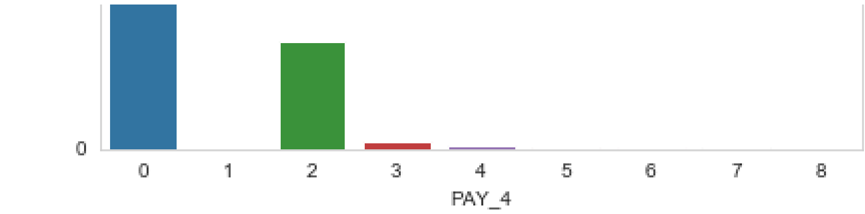

count
20000
15000
10000
5000
0
0
1
2
3
4
5
6
7
8
PAY_0

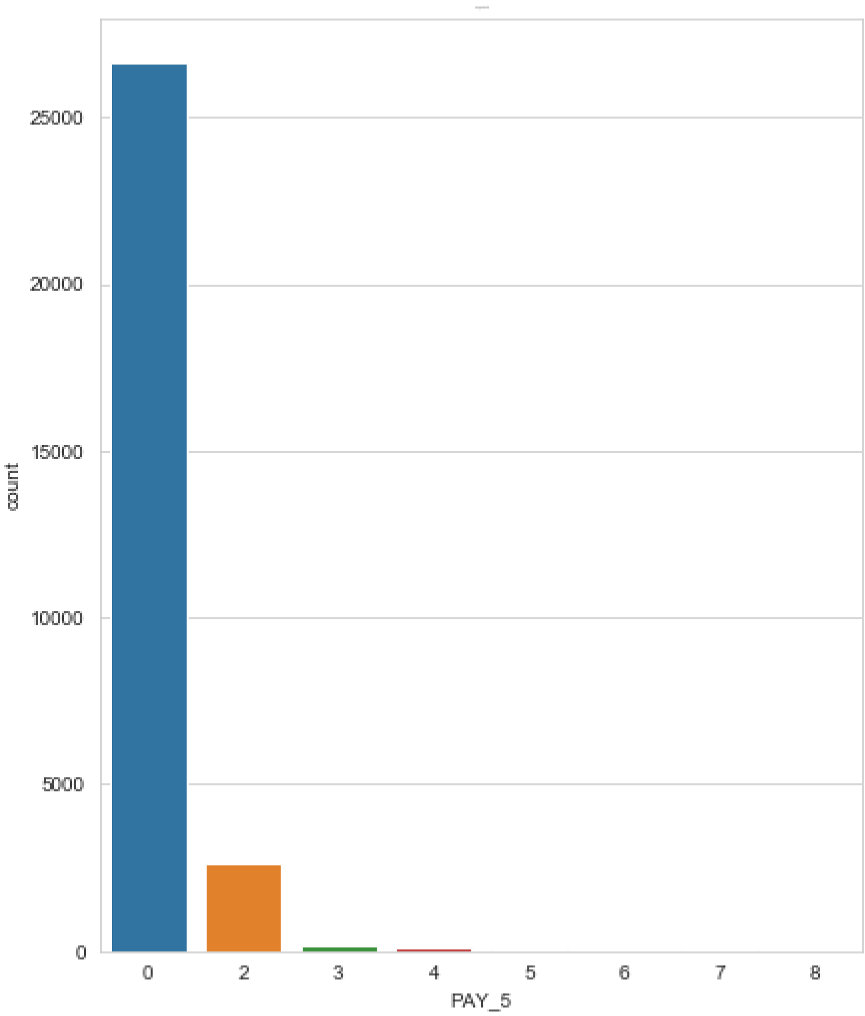

count
25000
20000
15000
10000
5000
0

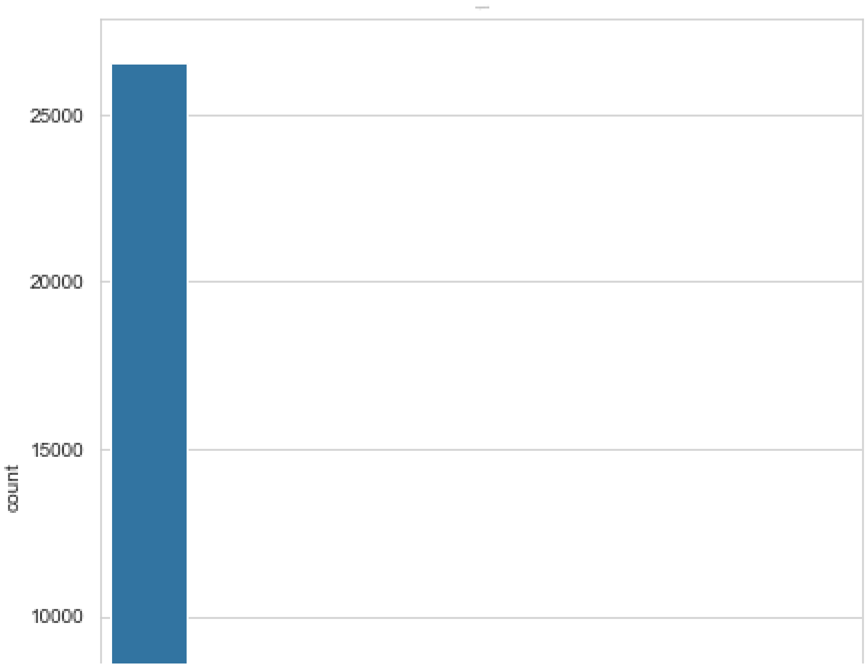
0
0
1
2
3
4
5
6
7
8
PAY_2

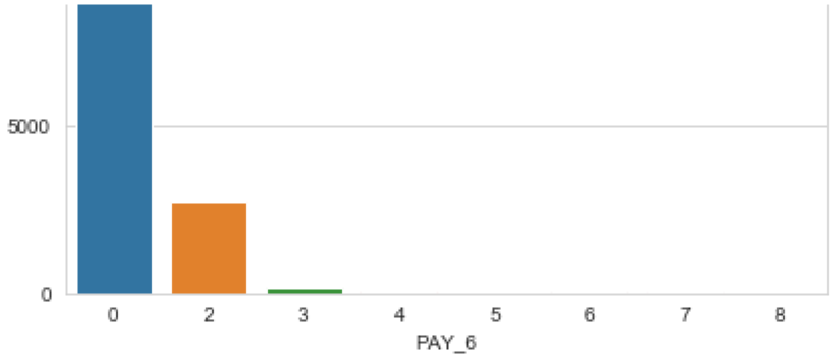

25000
20000
15000
10000
5000
0
count
0
1
2
3
4
5
6
7
8
PAY_3


25000
20000
15000
10000
5000
count


0
0
1
2
3
4
5
6
7
8
PAY_4


count
25000
20000
15000
10000
5000
0
0
2
3
4
5
6
7
8
PAY_5

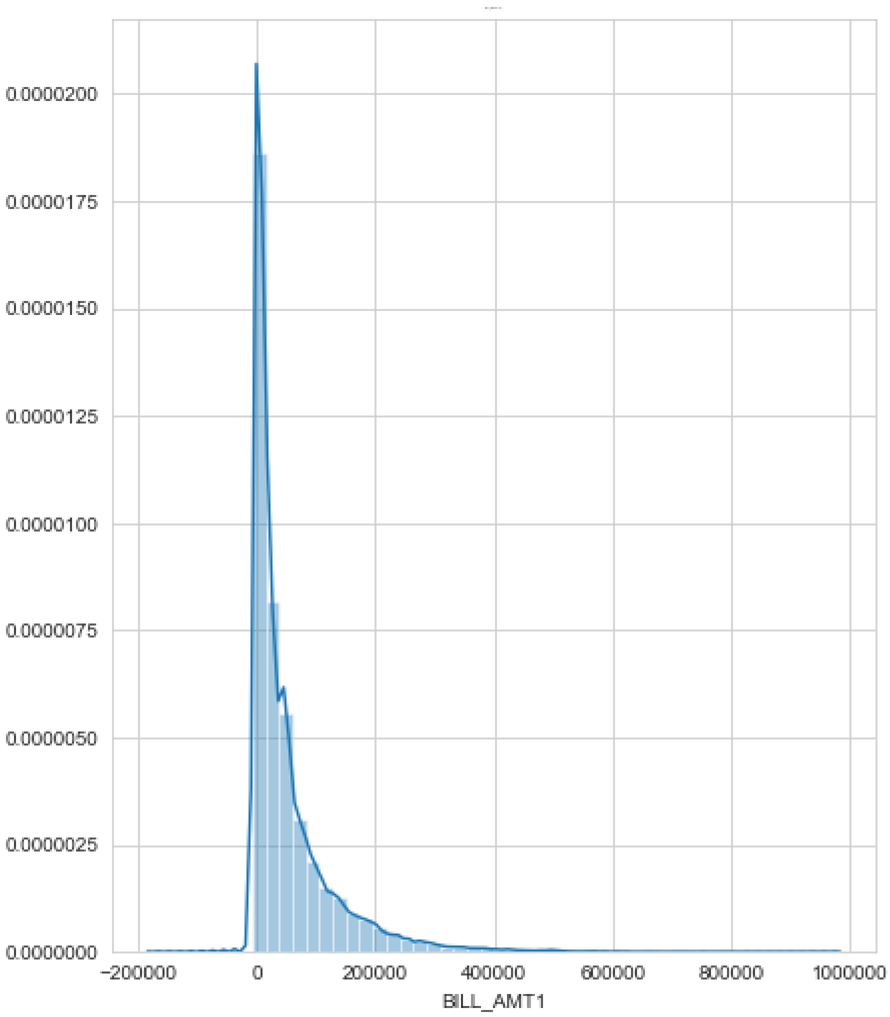

count
25000
20000
15000
10000

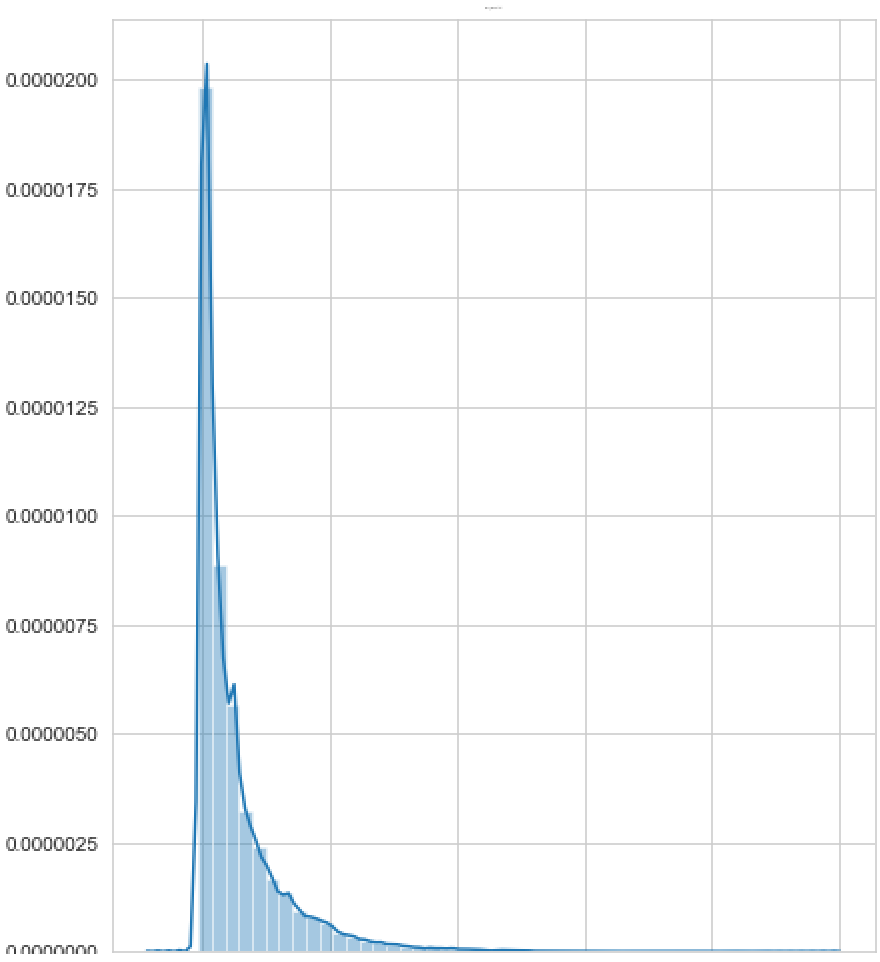


- History of past payment April to sept 2005
- 0 = pay duly
- 1 = payment delay for 1 month
- 2 = payment delay for 2 months and so on till....
- 9 = payment delay for 9 months

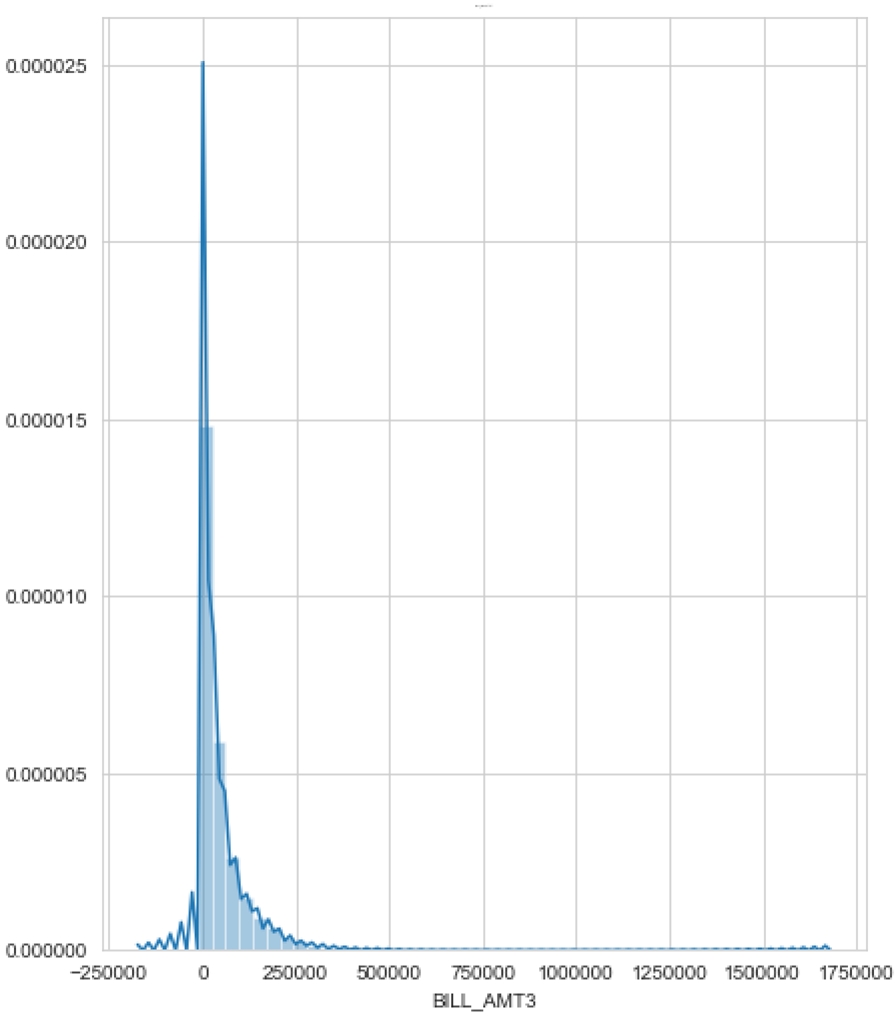
In [89]:

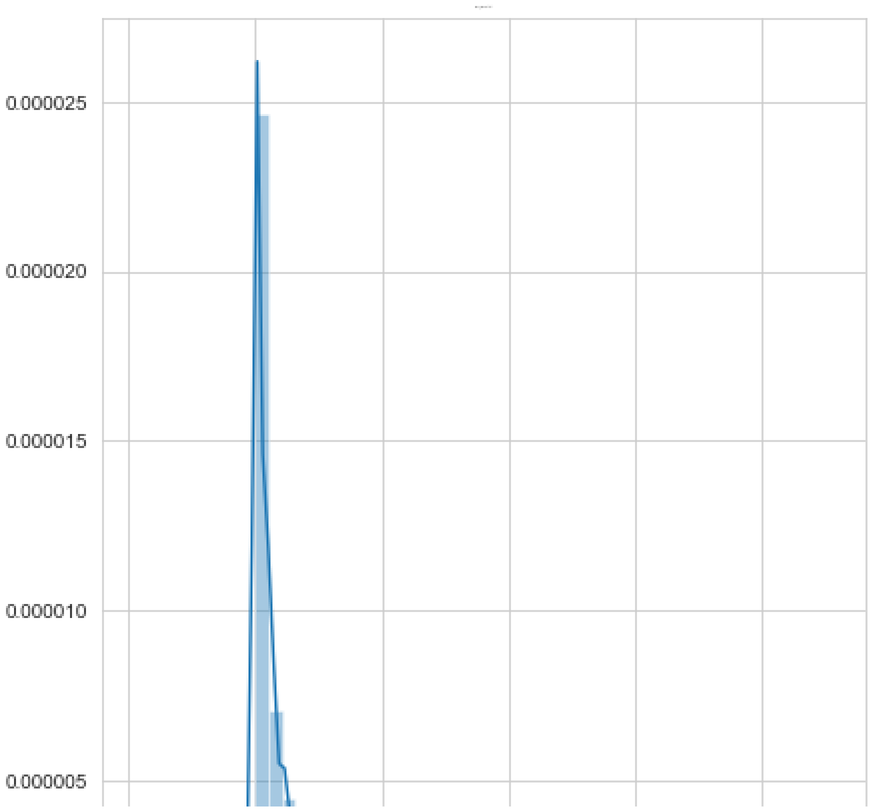
```
amt_of_bill_st = ['BILL_AMT1', 'BILL_AMT2', 'BILL_AMT3', 'BILL_AMT4', 'BILL_AMT5', 'BIL
L_AMT6']
plt.figure(figsize=(15,60))

for j,items in enumerate(amt_of_bill_st):
    y_axis = items
    fig_nums = 2*j+1
    sns.set_style('whitegrid')
    plt.subplot(6,2,fig_nums)
    plt.title(y_axis + 'dist plot', fontsize = 0.5)
    sns.distplot(card[y_axis])
```

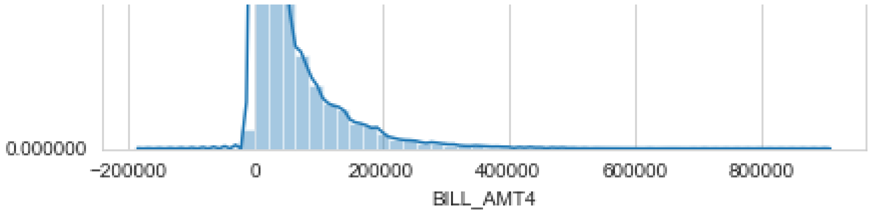

0.0000200
0.0000175
0.0000150
0.0000125
0.0000100
0.0000075
0.0000050
0.0000025
0.0000000
−200000
0
200000
400000
600000
800000
1000000
BILL_AMT1

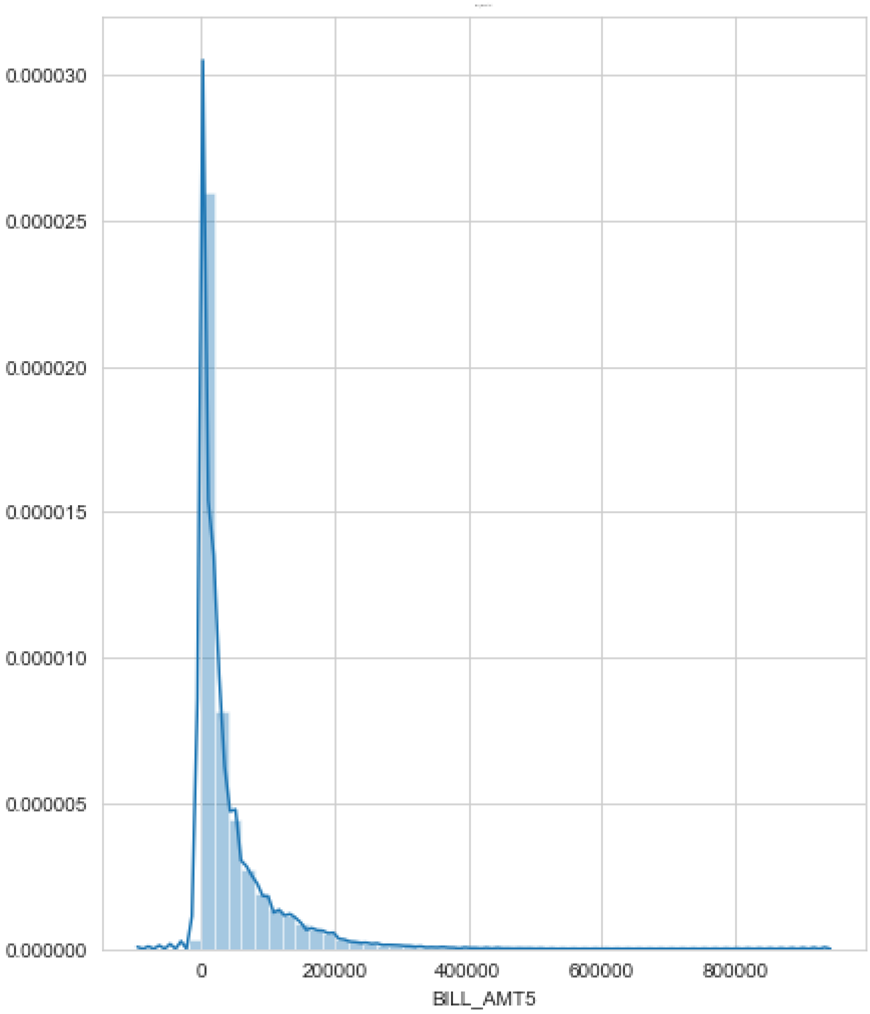

0.0000200
0.0000175
0.0000150
0.0000125
0.0000100
0.0000075
0.0000050
0.0000025
0.0000000

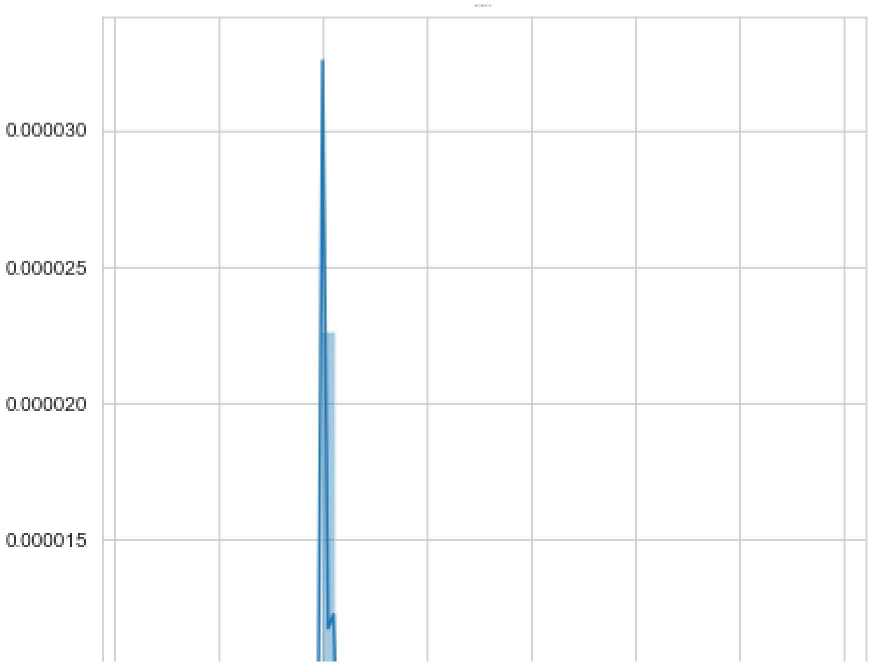
0
200000
400000
600000
800000
1000000
BILL_AMT2

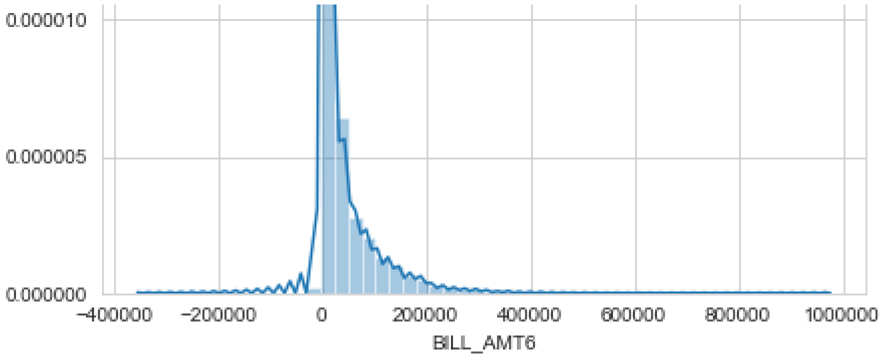

0.000025
0.000020
0.000015
0.000010
0.000005
0.000000
−250000
0
250000
500000
750000
1000000
1250000
1500000
1750000
BILL_AMT3


0.000025
0.000020
0.000015
0.000010
0.000005


0.000000
−200000
0
200000
400000
600000
800000
BILL_AMT4


0.000030
0.000025
0.000020
0.000015
0.000010
0.000005
0.000000
0
200000
400000
600000
800000
BILL_AMT5

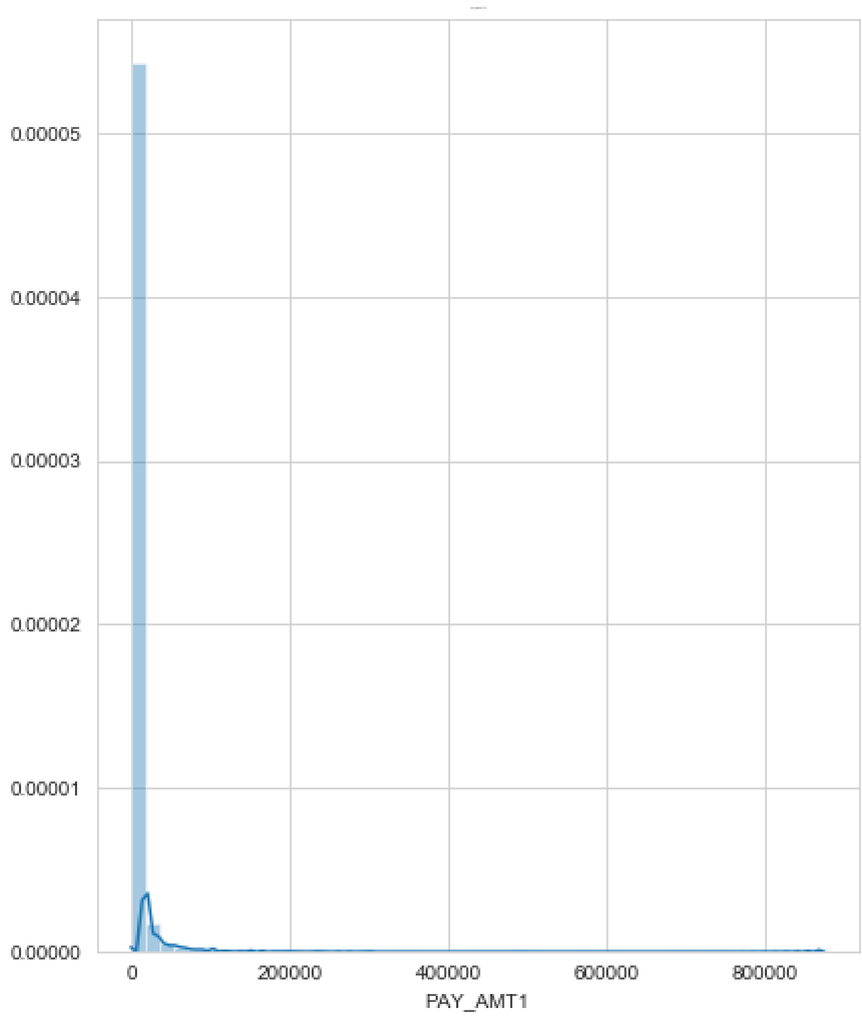

0.000030
0.000025
0.000020
0.000015

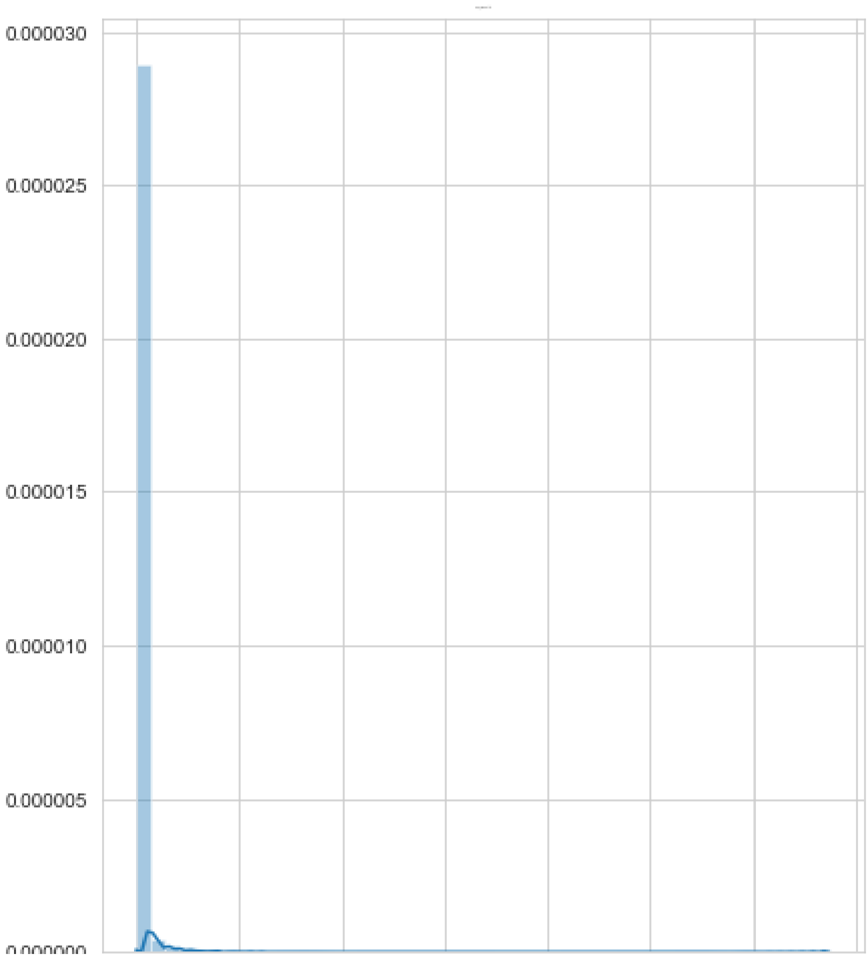


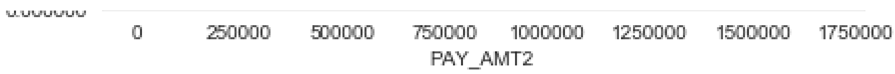
- amt of bill statements.

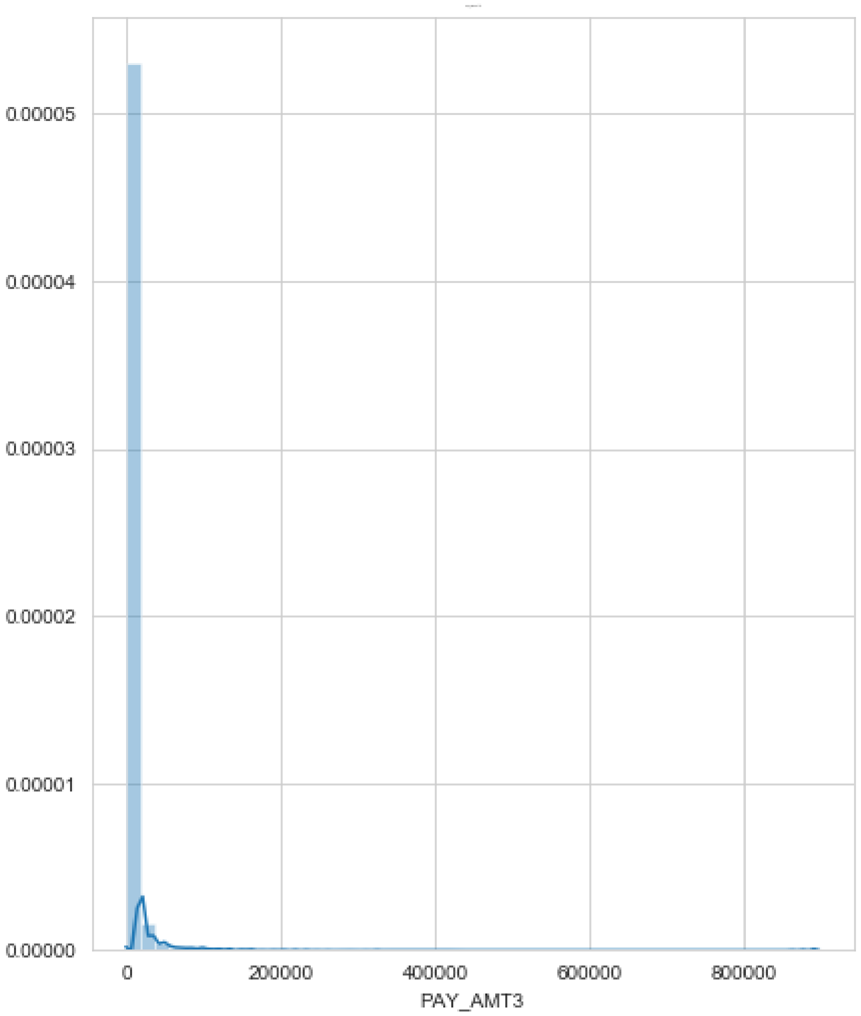
In [90]:

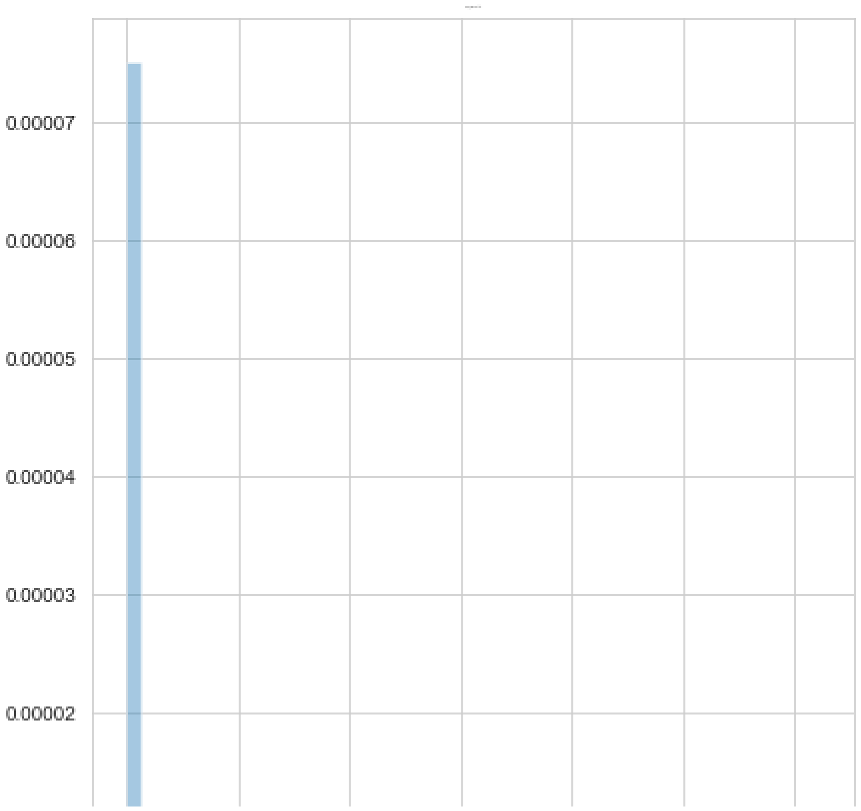
```
paid_amt = ['PAY_AMT1', 'PAY_AMT2', 'PAY_AMT3', 'PAY_AMT4', 'PAY_AMT5', 'PAY_AMT6']
plt.figure(figsize=(15,60))

for k,itemz in enumerate(paid_amt):
    z_axis = itemz
    fig_numz = 2*k+1
    sns.set_style('whitegrid')
    plt.subplot(6,2,fig_numz)
    plt.title(z_axis + 'dist plot', fontsize = 0.5)
    sns.distplot(card[z_axis])
```

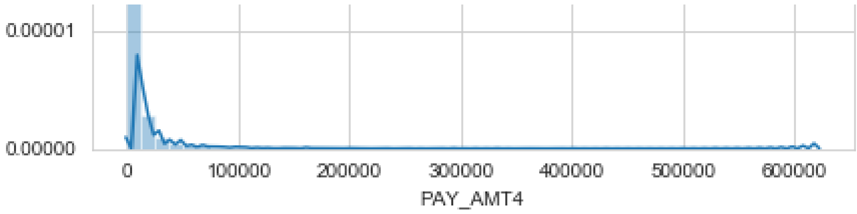

0.00005
0.00004
0.00003
0.00002
0.00001
0.00000
0
200000
400000
600000
800000
PAY_AMT1

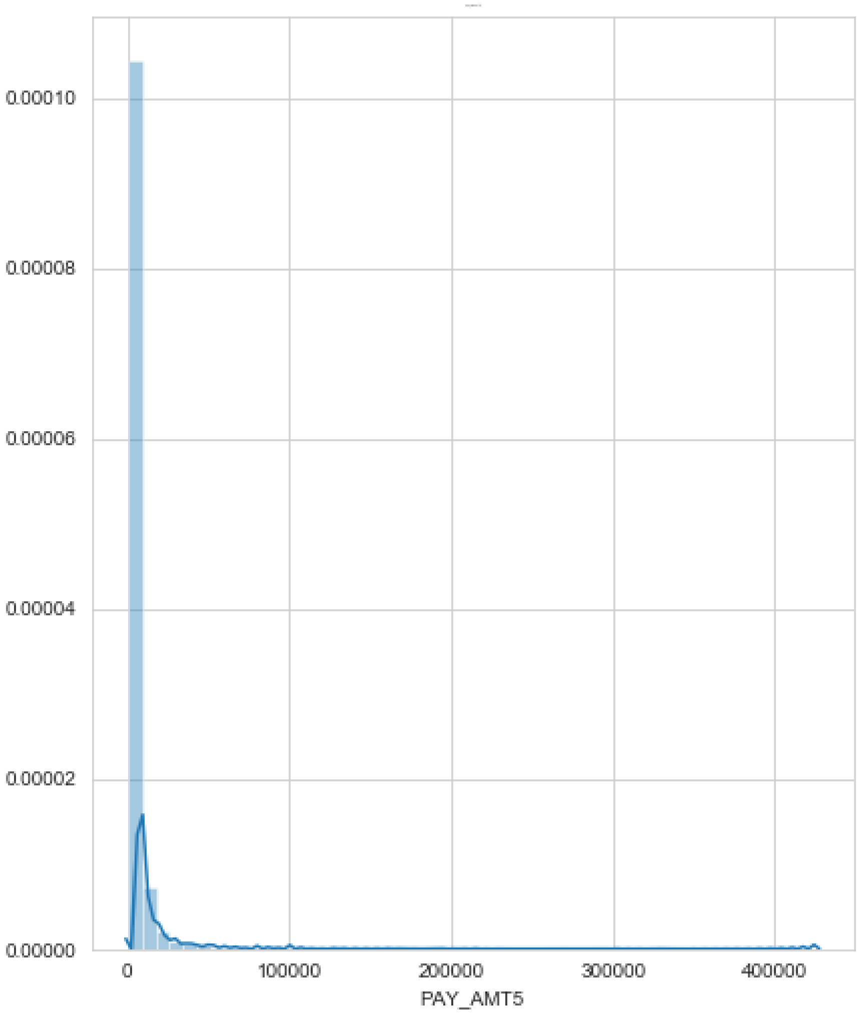

0.000030
0.000025
0.000020
0.000015
0.000010
0.000005
0.000000

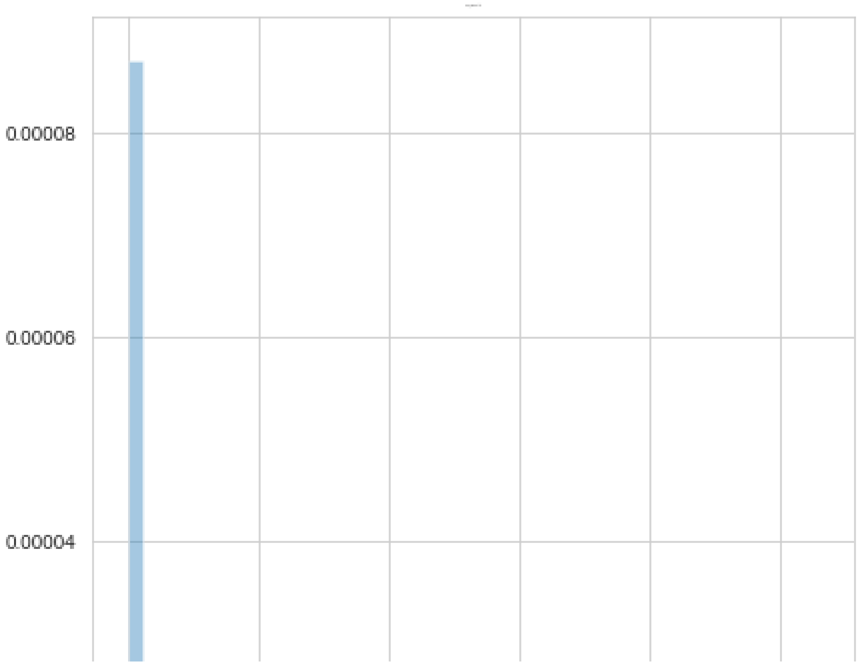

0.000000
0
250000
500000
750000
1000000
1250000
1500000
1750000
PAY_AMT2

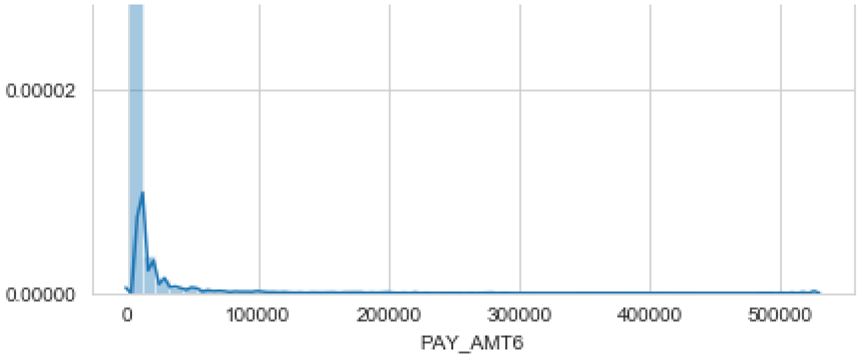

0.00005
0.00004
0.00003
0.00002
0.00001
0.00000
0
200000
400000
600000
800000
PAY_AMT3


0.00007
0.00006
0.00005
0.00004
0.00003
0.00002


0.00001
0.00000
0
100000
200000
300000
400000
500000
600000
PAY_AMT4


0.00010
0.00008
0.00006
0.00004
0.00002
0.00000
0
100000
200000
300000
400000
PAY_AMT5


0.00008
0.00006
0.00004

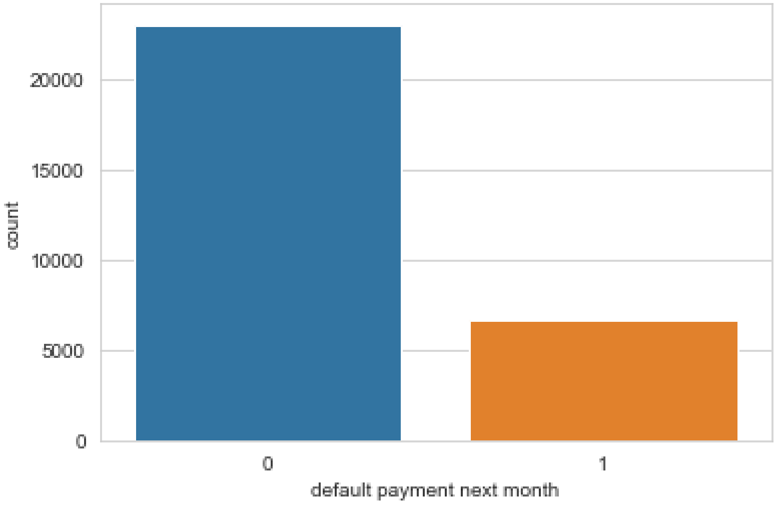


In [91]:

```
sns.countplot(card['default payment next month'])
```

Out[91]:

```
<matplotlib.axes._subplots.AxesSubplot at 0xd3a5308>
```



In [95]:

```
card.to_excel('x_card.xlsx', index=False)
```

In [96]:

```
cardz = pd.read_excel('x_card.xlsx')
```

In [97]:

```
cardz.shape
```

Out[97]:

```
(29601, 25)
```

In [98]:

```
import pandas as pd
from sklearn import metrics
from sklearn.linear_model import LogisticRegression
from sklearn import neighbors
#from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
```

In [99]:

```
cardz.head()
```

Out[99]:

	ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_0	PAY_2	PAY_3	PAY_4	PAY_
0	1	20000	2	2	1	24	2	2	0	0	
1	2	120000	2	2	2	26	0	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	0	0	0	0	

In [100]:

```
del cardz['ID']
```

In [101]:

```
cardz.head()
```

Out[101]:

	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_0	PAY_2	PAY_3	PAY_4	PAY_5
0	20000	2	2	1	24	2	2	0	0	0
1	120000	2	2	2	26	0	2	0	0	0
2	90000	2	2	2	34	0	0	0	0	0
3	50000	2	2	1	37	0	0	0	0	0
4	50000	1	2	1	57	0	0	0	0	0

In [102]:

```
#One-Hot Encoding
se_dum = pd.get_dummies(card['SEX'])
se_dum
```

Out[102]:

	1	2
0	0	1
1	0	1
2	0	1
3	0	1
4	1	0
...	...	...
29995	1	0
29996	1	0
29997	1	0
29998	1	0
29999	1	0

29601 rows × 2 columns

In [103]:

```
#One-Hot Encoding
ed_dum = pd.get_dummies(card['EDUCATION'])
ed_dum
```

Out[103]:

	1	2	3	4
0	0	1	0	0
1	0	1	0	0
2	0	1	0	0
3	0	1	0	0
4	0	1	0	0
...	...	...	...	...
29995	0	0	1	0
29996	0	0	1	0
29997	0	1	0	0
29998	0	0	1	0
29999	0	1	0	0

29601 rows × 4 columns

In [104]:

```
#One-Hot Encoding
marr_dum = pd.get_dummies(card['MARRIAGE'])
marr_dum
```

Out[104]:

	1	2	3
0	1	0	0
1	0	1	0
2	0	1	0
3	1	0	0
4	1	0	0
...	...	...	...
29995	1	0	0
29996	0	1	0
29997	0	1	0
29998	1	0	0
29999	1	0	0

29601 rows × 3 columns

In [105]:

```
cardz.columns
```

Out[105]:

```
Index(['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'],
      dtype='object')
```

In [107]:

```
cardz.corr()['default payment next month']
```

Out[107]:

```
LIMIT_BAL                    -0.154357
SEX                          -0.039815
EDUCATION                     0.049087
MARRIAGE                     -0.026903
AGE                           0.014424
PAY_0                         0.398048
PAY_2                         0.327919
PAY_3                         0.287002
PAY_4                         0.268986
PAY_5                         0.261114
PAY_6                         0.244659
BILL_AMT1                    -0.019303
BILL_AMT2                    -0.013710
BILL_AMT3                    -0.013494
BILL_AMT4                    -0.009474
BILL_AMT5                    -0.006226
BILL_AMT6                    -0.005339
PAY_AMT1                     -0.073881
PAY_AMT2                     -0.058307
PAY_AMT3                     -0.056288
PAY_AMT4                     -0.057012
PAY_AMT5                     -0.056075
PAY_AMT6                     -0.053692
default payment next month    1.000000
Name: default payment next month, dtype: float64
```

In [109]:

```
#sampling
cols = ['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']
X = cardz[cols]
print(X.head())
```

```
   LIMIT_BAL  SEX  EDUCATION  MARRIAGE  AGE  PAY_0  PAY_2  PAY_3  PAY_4  \
0      20000    2          2         1   24      2      2      0      0   
1     120000    2          2         2   26      0      2      0      0   
2      90000    2          2         2   34      0      0      0      0   
3      50000    2          2         1   37      0      0      0      0   
4      50000    1          2         1   57      0      0      0      0   

   PAY_5  PAY_6  BILL_AMT1  BILL_AMT2  BILL_AMT3  BILL_AMT4  BILL_AMT5  \
0      0      0       3913       3102        689          0          0   
1      0      2       2682       1725       2682       3272       3455   
2      0      0      29239      14027      13559      14331      14948   
3      0      0      46990      48233      49291      28314      28959   
4      0      0       8617       5670      35835      20940      19146   

   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  
```

In [110]:

```
colz = ['default payment next month']
y = cardz[colz]
```

In [111]:

```
print(y.head())
```

```
   default payment next month
0                           1
1                           1
2                           0
3                           0
4                           0
```

In [112]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y , test_size = 0.2)
```

In [113]:

```
print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
```

```
(23680, 23)
(23680, 1)
(5921, 23)
(5921, 1)
```

In [114]:

```
#Training
model = LogisticRegression()
#model = neighbors.KNeighborsClassifier()
#model = DecisionTreeClassifier(criterion='entropy',max_depth= 8)
#model = SVC(kernel='linear',  gamma = 10, C= 1)


model.fit(X_train,y_train)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py:76
0: DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  y = column_or_1d(y, warn=True)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.
py:940: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown i
n:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-reg
ression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
```

Out[114]:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=Tru
e,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=
0,
                   warm_start=False)
```

In [115]:

```
#Testing
predicted = model.predict(X_test)
predicted
```

Out[115]:

```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

In [116]:

```
#Evaluation
#Confusion Matrix
print(metrics.confusion_matrix(y_test, predicted))
```

```
[[4605    0]
 [1316    0]]
```

In [117]:

```
#Classification Report
print(metrics.classification_report(y_test, predicted))
```

```
              precision    recall  f1-score   support

           0       0.78      1.00      0.87      4605
           1       0.00      0.00      0.00      1316

    accuracy                           0.78      5921
   macro avg       0.39      0.50      0.44      5921
weighted avg       0.60      0.78      0.68      5921


C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_classificatio
n.py:1272: UndefinedMetricWarning: Precision and F-score are ill-defined a
nd being set to 0.0 in labels with no predicted samples. Use `zero_divisio
n` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [118]:

```
from sklearn.model_selection import cross_val_score
accuracies = cross_val_score(estimator = model, X = X_train,\
     y = y_train, cv = 10)
print("Accuracy Mean {} Accuracy Variance \
     {}".format(accuracies.mean(),accuracies.std()))
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py:76
0: DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  y = column_or_1d(y, warn=True)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.
py:940: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown i
n:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-reg
ression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py:76
0: DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  y = column_or_1d(y, warn=True)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.
py:940: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown i
n:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-reg
ression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py:76
0: DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  y = column_or_1d(y, warn=True)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.
py:940: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown i
n:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-reg
ression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py:76
0: DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  y = column_or_1d(y, warn=True)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.
py:940: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown i
n:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-reg
ression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py:76
0: DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  y = column_or_1d(y, warn=True)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.
py:940: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown i
n:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-reg
ression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py:76
0: DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  y = column_or_1d(y, warn=True)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.
py:940: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown i
n:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-reg
ression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py:76
0: DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  y = column_or_1d(y, warn=True)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.
py:940: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown i
n:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-reg
ression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py:76
0: DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  y = column_or_1d(y, warn=True)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.
py:940: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown i
```

```
n:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-reg
ression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py:76
0: DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  y = column_or_1d(y, warn=True)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.
py:940: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown i
n:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-reg
ression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py:76
0: DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  y = column_or_1d(y, warn=True)


Accuracy Mean 0.7764780405405406 Accuracy Variance      0.0002704022059726
46

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.
py:940: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown i
n:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-reg
ression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
```

In [119]:

```python
#Accuracy Score
from sklearn.metrics import accuracy_score

accuracy_score(y_test,predicted)
```

Out[119]:

```
0.7777402465799697
```

In [120]:

```
from sklearn.model_selection import GridSearchCV
parameters = {'criterion':('gini', 'entropy'), 'max_depth':[8, 10,12]}
dt = DecisionTreeClassifier()
clf = GridSearchCV(dt, parameters)
clf.fit(X_train, y_train)
```

Out[120]:

```
GridSearchCV(cv=None, error_score=nan,
             estimator=DecisionTreeClassifier(ccp_alpha=0.0, class_weight=
None,
                                              criterion='gini', max_depth=
None,
                                              max_features=None,
                                              max_leaf_nodes=None,
                                              min_impurity_decrease=0.0,
                                              min_impurity_split=None,
                                              min_samples_leaf=1,
                                              min_samples_split=2,
                                              min_weight_fraction_leaf=0.
0,
                                              presort='deprecated',
                                              random_state=None,
                                              splitter='best'),
             iid='deprecated', n_jobs=None,
             param_grid={'criterion': ('gini', 'entropy'),
                         'max_depth': [8, 10, 12]},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=Fals
e,
             scoring=None, verbose=0)
```

In [121]:

```
predicted = clf.predict(X_test)
predicted
```

Out[121]:

```
array([1, 0, 0, ..., 0, 1, 0], dtype=int64)
```

In [122]:

```
from sklearn.metrics import accuracy_score

accuracy_score(y_test,predicted)
```

Out[122]:

```
0.8145583516297923
```

In [ ]: