In [1]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```python
apr=pd.read_csv('application_record.csv')
crecord=pd.read_csv('credit_record.csv')
```

In [3]:
```python
apr.head()
```

Out[3]:

	ID	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	NAME_INCOME_T
0	5008804	M	Y	Y	0	427500.0	Wor
1	5008805	M	Y	Y	0	427500.0	Wor
2	5008806	M	Y	Y	0	112500.0	Wor
3	5008808	F	N	Y	0	270000.0	Commercial asso
4	5008809	F	N	Y	0	270000.0	Commercial asso

In [4]:
```python
apr.tail()
```

Out[4]:

	ID	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	NAME_INCO
438552	6840104	M	N	Y	0	135000.0	
438553	6840222	F	N	N	0	103500.0	
438554	6841878	F	N	N	0	54000.0	Commercia
438555	6842765	F	N	Y	0	72000.0	
438556	6842885	F	N	Y	0	121500.0	

In [5]:
```python
crecord.head()
```

Out[5]:

	ID	MONTHS_BALANCE	STATUS
0	5001711	0	X
1	5001711	-1	0
2	5001711	-2	0
3	5001711	-3	0
4	5001712	0	C

In [6]:
```python
crecord.tail()
```

Out[6]:

	ID	MONTHS_BALANCE	STATUS
1048570	5150487	-25	C
1048571	5150487	-26	C
1048572	5150487	-27	C
1048573	5150487	-28	C
1048574	5150487	-29	C

In [7]:
```python
apr.shape
```

Out[7]: (438557, 18)

In [8]:
```python
crecord.shape
```

Out[8]: (1048575, 3)

In [9]:
```python
apr.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 438557 entries, 0 to 438556
Data columns (total 18 columns):
 #   Column               Non-Null Count   Dtype  
---  ------               --------------   -----  
 0   ID                   438557 non-null  int64  
 1   CODE_GENDER          438557 non-null  object 
 2   FLAG_OWN_CAR         438557 non-null  object 
 3   FLAG_OWN_REALTY      438557 non-null  object 
 4   CNT_CHILDREN         438557 non-null  int64  
 5   AMT_INCOME_TOTAL     438557 non-null  float64
 6   NAME_INCOME_TYPE     438557 non-null  object 
 7   NAME_EDUCATION_TYPE  438557 non-null  object 
 8   NAME_FAMILY_STATUS   438557 non-null  object 
 9   NAME_HOUSING_TYPE    438557 non-null  object 
 10  DAYS_BIRTH           438557 non-null  int64  
 11  DAYS_EMPLOYED        438557 non-null  int64  
 12  FLAG_MOBIL           438557 non-null  int64  
 13  FLAG_WORK_PHONE      438557 non-null  int64  
 14  FLAG_PHONE           438557 non-null  int64  
 15  FLAG_EMAIL           438557 non-null  int64  
 16  OCCUPATION_TYPE      304354 non-null  object 
 17  CNT_FAM_MEMBERS      438557 non-null  float64
dtypes: float64(2), int64(8), object(8)
memory usage: 60.2+ MB
```

In [10]:
```python
crecord.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1048575 entries, 0 to 1048574
Data columns (total 3 columns):
 #   Column          Non-Null Count    Dtype 
---  ------          --------------    ----- 
 0   ID              1048575 non-null  int64 
 1   MONTHS_BALANCE  1048575 non-null  int64 
 2   STATUS          1048575 non-null  object
dtypes: int64(2), object(1)
memory usage: 24.0+ MB
```

In [11]:
```python
apr['ID'].nunique()
```

Out[11]: 438510

In [12]:
```python
crecord['ID'].nunique()
```

Out[12]: 45985

In [13]:
```python
apr['CODE_GENDER'].nunique()
```

Out[13]: 2

In [14]:
```python
apr['OCCUPATION_TYPE'].nunique()
```

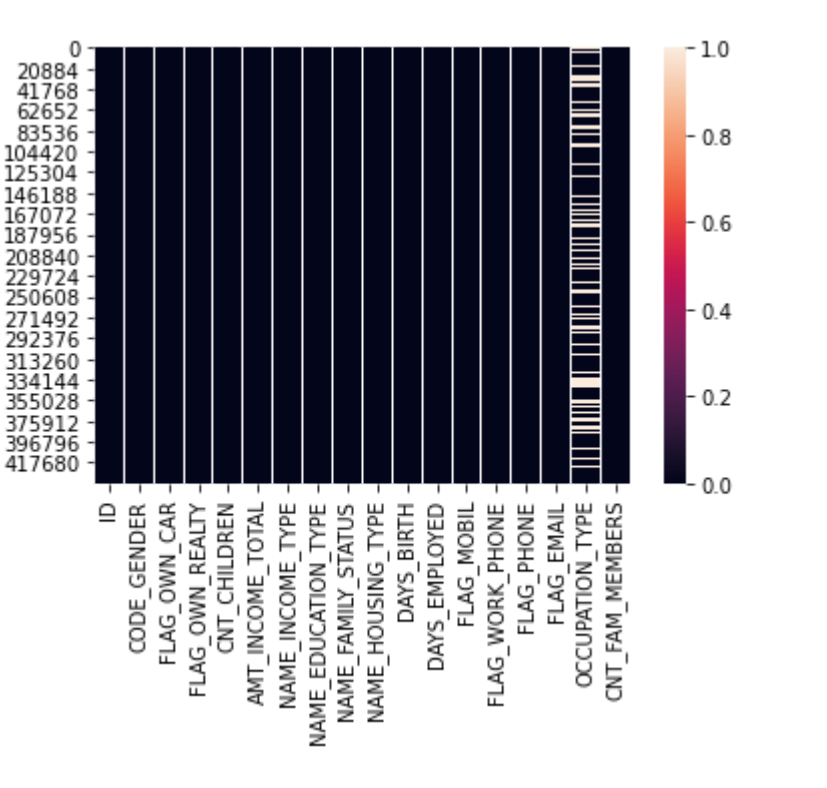
Out[14]: 18

In [15]:
```python
len(set(crecord['ID']).intersection(set(apr['ID'])))
```

Out[15]: 36457

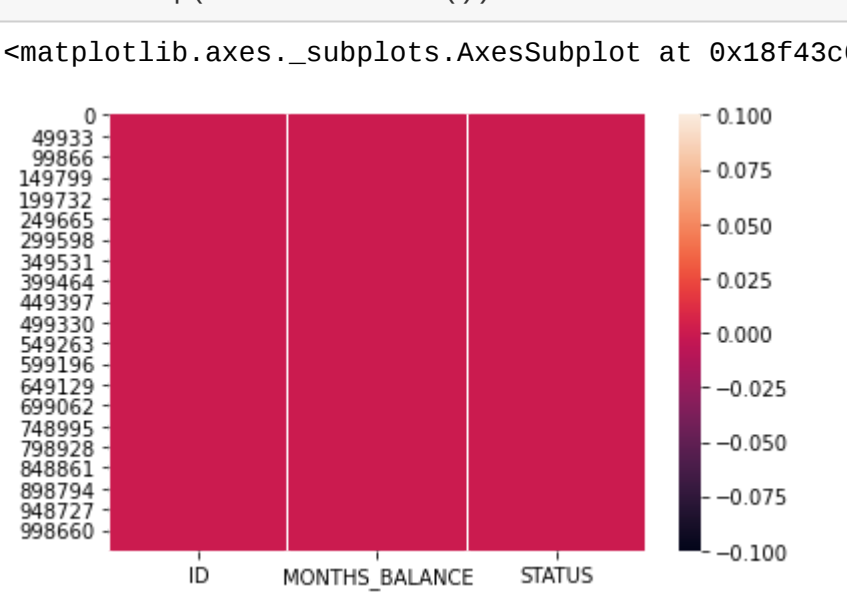
In [16]:
```python
sns.heatmap(apr.isnull())
```

Out[16]: <matplotlib.axes._subplots.AxesSubplot at 0x18f43045430>



In [17]:
```python
sns.heatmap(crecord.isnull())
```

Out[17]: <matplotlib.axes._subplots.AxesSubplot at 0x18f43c66a90>



In [18]:
```python
apr=apr.drop_duplicates('ID', keep='last')
```

In [19]:
```python
apr.drop('OCCUPATION_TYPE',axis=1,inplace=True)
```

In [20]:
```python
op=pd.DataFrame(apr.dtypes=='object').reset_index()
object_type=op[op[0]==True]['index']
object_type
```

Out[20]:
```
1            CODE_GENDER
2           FLAG_OWN_CAR
3        FLAG_OWN_REALTY
6       NAME_INCOME_TYPE
7    NAME_EDUCATION_TYPE
8     NAME_FAMILY_STATUS
9      NAME_HOUSING_TYPE
Name: index, dtype: object
```

In [21]:
```python
num_type=pd.DataFrame(apr.dtypes != 'object').reset_index().rename(columns={0:'yes/no'})
num_type=num_type[num_type['yes/no']==True]['index']
```

In [22]:
```python
a = apr[object_type]['CODE_GENDER'].value_counts()
b = apr[object_type]['FLAG_OWN_CAR'].value_counts()
c = apr[object_type]['FLAG_OWN_REALTY'].value_counts()
d = apr[object_type]['NAME_INCOME_TYPE'].value_counts()
e = apr[object_type]['NAME_EDUCATION_TYPE'].value_counts()
f = apr[object_type]['NAME_FAMILY_STATUS'].value_counts()
g = apr[object_type]['NAME_HOUSING_TYPE'].value_counts()

print( a,"\n",b,'\n', c, '\n', d, '\n', e, '\n', f, '\n', g)
```

```
F    294412
M    144098
Name: CODE_GENDER, dtype: int64
 N    275428
Y    163082
Name: FLAG_OWN_CAR, dtype: int64
 Y    304043
N    134467
Name: FLAG_OWN_REALTY, dtype: int64
 Working                 226087
Commercial associate    100739
Pensioner                75483
State servant            36184
Student                     17
Name: NAME_INCOME_TYPE, dtype: int64
 Secondary / secondary special    301789
Higher education                 117509
Incomplete higher                 14849
Lower secondary                    4051
Academic degree                     312
Name: NAME_EDUCATION_TYPE, dtype: int64
 Married                 299798
Single / not married     55268
Civil marriage           36524
Separated                27249
Widow                    19671
Name: NAME_FAMILY_STATUS, dtype: int64
 House / apartment      393788
With parents            19074
Municipal apartment     14213
Rented apartment         5974
Office apartment         3922
Co-op apartment          1539
Name: NAME_HOUSING_TYPE, dtype: int64
```

In [23]:
```python
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
for x in apr:
    if apr[x].dtypes=='object':
        apr[x]=le.fit_transform(apr[x])
```

In [24]:
```python
apr.head()
```

Out[24]:

	ID	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	NAME_INCOME_T
0	5008804	1	1	1	0	427500.0	
1	5008805	1	1	1	0	427500.0	
2	5008806	1	1	1	0	112500.0	
3	5008808	0	0	1	0	270000.0	
4	5008809	0	0	1	0	270000.0	

In [25]:
```python
apr[num_type].head()
```

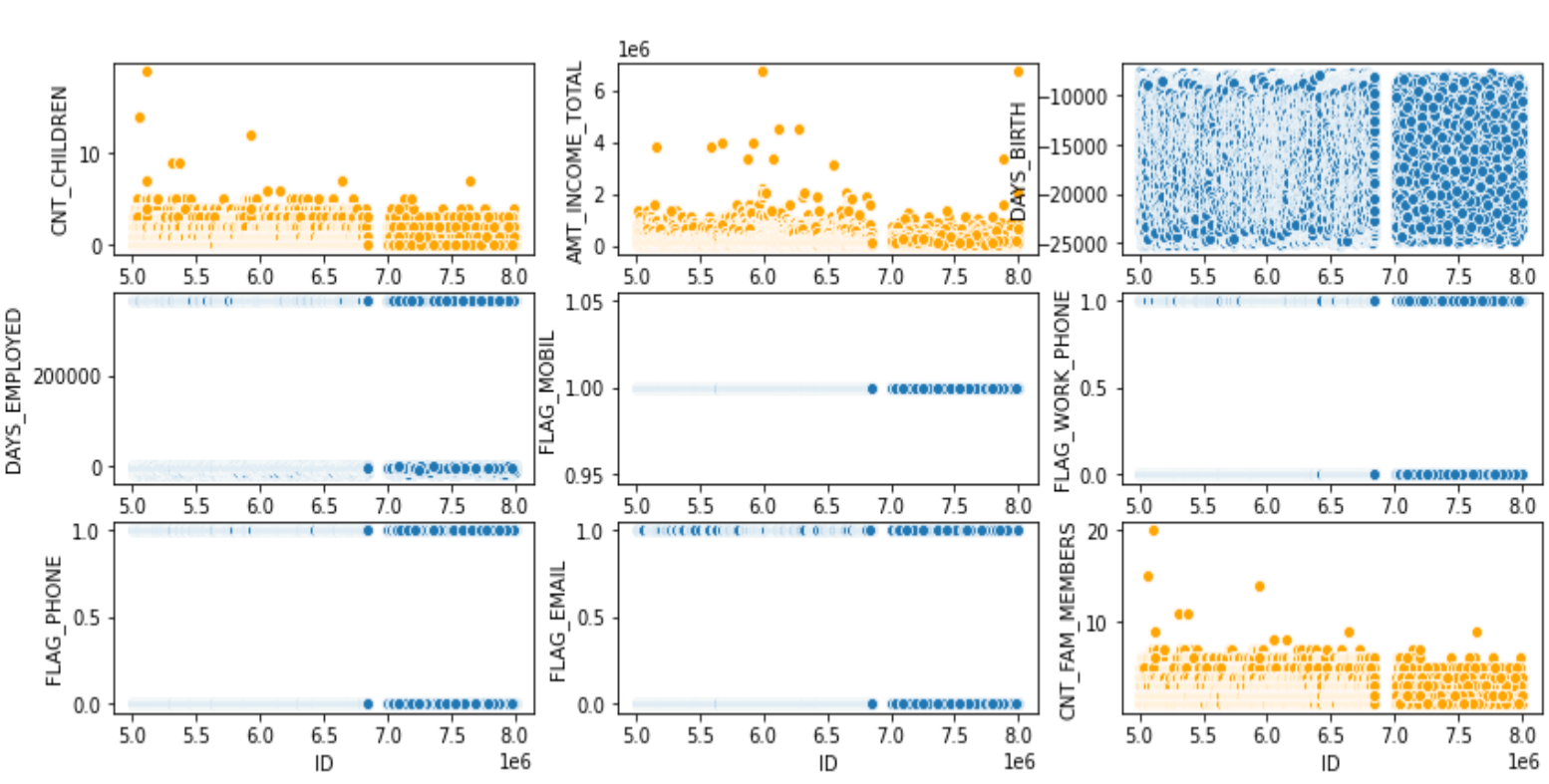
Out[25]:

	ID	CNT_CHILDREN	AMT_INCOME_TOTAL	DAYS_BIRTH	DAYS_EMPLOYED	FLAG_MOBIL	FLAG_WORK_PHONE	FLA
0	5008804	0	427500.0	-12005	-4542	1	1	
1	5008805	0	427500.0	-12005	-4542	1	1	
2	5008806	0	112500.0	-21474	-1134	1	0	
3	5008808	0	270000.0	-19110	-3051	1	0	
4	5008809	0	270000.0	-19110	-3051	1	0	

In [26]:
```python
fig, ax= plt.subplots(nrows= 3, ncols = 3, figsize= (14,6))

sns.scatterplot(x='ID', y='CNT_CHILDREN', data=apr, ax=ax[0][0], color= 'orange')
sns.scatterplot(x='ID', y='AMT_INCOME_TOTAL', data=apr, ax=ax[0][1], color='orange')
sns.scatterplot(x='ID', y='DAYS_BIRTH', data=apr, ax=ax[0][2])
sns.scatterplot(x='ID', y='DAYS_EMPLOYED', data=apr, ax=ax[1][0])
sns.scatterplot(x='ID', y='FLAG_MOBIL', data=apr, ax=ax[1][1])
sns.scatterplot(x='ID', y='FLAG_WORK_PHONE', data=apr, ax=ax[1][2])
sns.scatterplot(x='ID', y='FLAG_PHONE', data=apr, ax=ax[2][0])
sns.scatterplot(x='ID', y='FLAG_EMAIL', data=apr, ax=ax[2][1])
sns.scatterplot(x='ID', y='CNT_FAM_MEMBERS', data=apr, ax=ax[2][2], color= 'orange')
```

Out[26]: <matplotlib.axes._subplots.AxesSubplot at 0x18f4325c160>



In [27]:
```python
q_hi=apr['CNT_CHILDREN'].quantile(0.999)
q_low=apr['CNT_CHILDREN'].quantile(0.001)
apr=apr[(apr['CNT_CHILDREN']>q_low)&(apr['CNT_CHILDREN']<q_hi)]
```

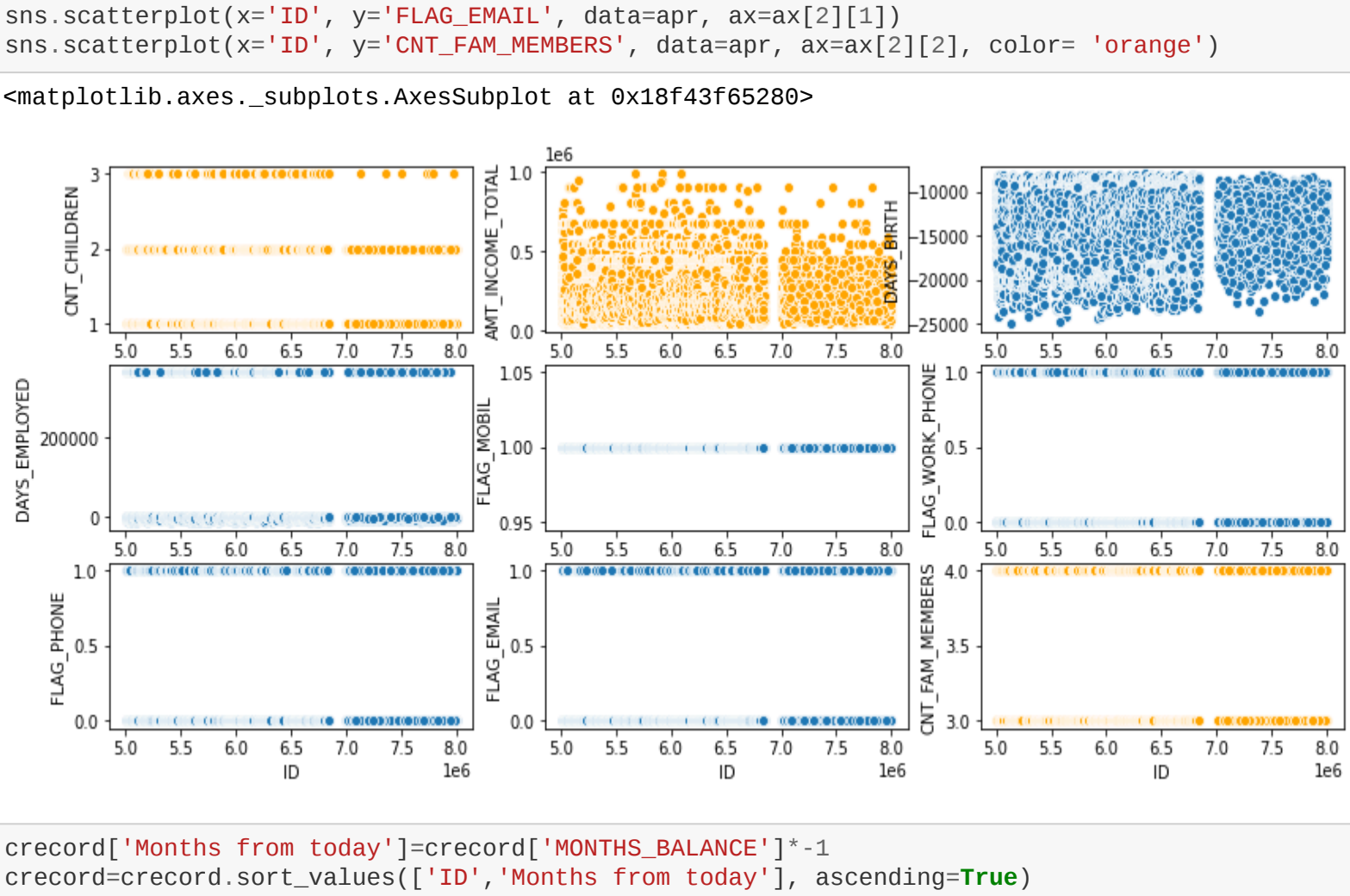
In [28]:
```python
q_hi=apr['AMT_INCOME_TOTAL'].quantile(0.999)
q_low=apr['AMT_INCOME_TOTAL'].quantile(0.001)
apr=apr[(apr['AMT_INCOME_TOTAL']>q_low)&(apr['AMT_INCOME_TOTAL']<q_hi)]
```

In [29]:
```python
q_hi=apr['CNT_FAM_MEMBERS'].quantile(0.999)
q_low=apr['CNT_FAM_MEMBERS'].quantile(0.001)
apr=apr[(apr['CNT_FAM_MEMBERS']>q_low)&(apr['CNT_FAM_MEMBERS']<q_hi)]
```

In [30]:
```python
fig, ax= plt.subplots(nrows= 3, ncols = 3, figsize= (14,6))

sns.scatterplot(x='ID', y='CNT_CHILDREN', data=apr, ax=ax[0][0], color= 'orange')
sns.scatterplot(x='ID', y='AMT_INCOME_TOTAL', data=apr, ax=ax[0][1], color='orange')
sns.scatterplot(x='ID', y='DAYS_BIRTH', data=apr, ax=ax[0][2])
sns.scatterplot(x='ID', y='DAYS_EMPLOYED', data=apr, ax=ax[1][0])
sns.scatterplot(x='ID', y='FLAG_MOBIL', data=apr, ax=ax[1][1])
sns.scatterplot(x='ID', y='FLAG_WORK_PHONE', data=apr, ax=ax[1][2])
sns.scatterplot(x='ID', y='FLAG_PHONE', data=apr, ax=ax[2][0])
sns.scatterplot(x='ID', y='FLAG_EMAIL', data=apr, ax=ax[2][1])
sns.scatterplot(x='ID', y='CNT_FAM_MEMBERS', data=apr, ax=ax[2][2], color= 'orange')
```

Out[30]: <matplotlib.axes._subplots.AxesSubplot at 0x18f43f65280>



In [31]:
```python
crecord['Months from today']=crecord['MONTHS_BALANCE']*-1
crecord=crecord.sort_values(['ID','Months from today'], ascending=True)
crecord.head()
```

Out[31]:

	ID	MONTHS_BALANCE	STATUS	Months from today
0	5001711	0	X	0
1	5001711	-1	0	1
2	5001711	-2	0	2
3	5001711	-3	0	3
4	5001712	0	C	0

In [32]:
```python
crecord['STATUS'].value_counts()
```

Out[32]:
```
C    442031
0    383120
X    209230
1     11090
5      1693
2       868
3       320
4       223
Name: STATUS, dtype: int64
```

In [33]:
```python
crecord['STATUS'].replace({'C': 0, 'X' : 0}, inplace=True)
crecord['STATUS']=crecord['STATUS'].astype('int')
crecord['STATUS']=crecord['STATUS'].apply(lambda x:1 if x >= 2 else 0)
```

In [34]:
```python
crecord['STATUS'].value_counts(normalize=True)
```

Out[34]:
```
0    0.99704
1    0.00296
Name: STATUS, dtype: float64
```

In [35]:
```python
crecordgb=crecord.groupby('ID').agg(max).reset_index()
crecordgb.head()
```

Out[35]:

	ID	MONTHS_BALANCE	STATUS	Months from today
0	5001711	0	0	3
1	5001712	0	0	18
2	5001713	0	0	21
3	5001714	0	0	14
4	5001715	0	0	59

In [36]:
```python
df=apr.join(crecordgb.set_index('ID'), on='ID', how='inner')
df.drop(['Months from today', 'MONTHS_BALANCE'], axis=1, inplace=True)
df.head()
```

Out[36]:

	ID	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	NAME_INCOME_
29	5008838	1	0	1	1	405000.0	
30	5008839	1	0	1	1	405000.0	
31	5008840	1	0	1	1	405000.0	
32	5008841	1	0	1	1	405000.0	
33	5008842	1	0	1	1	405000.0	

In [37]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 9516 entries, 29 to 434805
Data columns (total 18 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   ID                   9516 non-null   int64  
 1   CODE_GENDER          9516 non-null   int32  
 2   FLAG_OWN_CAR         9516 non-null   int32  
 3   FLAG_OWN_REALTY      9516 non-null   int32  
 4   CNT_CHILDREN         9516 non-null   int64  
 5   AMT_INCOME_TOTAL     9516 non-null   float64
 6   NAME_INCOME_TYPE     9516 non-null   int32  
 7   NAME_EDUCATION_TYPE  9516 non-null   int32  
 8   NAME_FAMILY_STATUS   9516 non-null   int32  
 9   NAME_HOUSING_TYPE    9516 non-null   int32  
 10  DAYS_BIRTH           9516 non-null   int64  
 11  DAYS_EMPLOYED        9516 non-null   int64  
 12  FLAG_MOBIL           9516 non-null   int64  
 13  FLAG_WORK_PHONE      9516 non-null   int64  
 14  FLAG_PHONE           9516 non-null   int64  
 15  FLAG_EMAIL           9516 non-null   int64  
 16  CNT_FAM_MEMBERS      9516 non-null   float64
 17  STATUS               9516 non-null   int64  
dtypes: float64(2), int32(7), int64(9)
memory usage: 1.1 MB
```

In [38]:
```python
X=df.iloc[:,1:-1]
y=df.iloc[:,-1]
```

In [39]:
```python
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.3)
```

In [40]:
```python
from sklearn.preprocessing import MinMaxScaler
mms = MinMaxScaler()
X_scaled = pd.DataFrame(mms.fit_transform(X_train), columns=X_train.columns)
X_test_scaled = pd.DataFrame(mms.transform(X_test), columns=X_test.columns)
```

In [42]:
```python
from imblearn import under_sampling, over_sampling
from imblearn.over_sampling import SMOTE
oversample = SMOTE()
X_balanced, y_balanced = oversample.fit_resample(X_scaled, y_train)
X_test_balanced, y_test_balanced = oversample.fit_resample(X_test_scaled, y_test)
```

In [43]:
```python
y_train.value_counts()
```

Out[43]:
```
0    6546
1     115
Name: STATUS, dtype: int64
```

In [44]:
```python
y_balanced.value_counts()
```

Out[44]:
```
1    6546
0    6546
Name: STATUS, dtype: int64
```

In [45]:
```python
y_test.value_counts()
```

Out[45]:
```
0    2819
1      36
Name: STATUS, dtype: int64
```

In [46]:
```python
y_test_balanced.value_counts()
```

Out[46]:
```
1    2819
0    2819
Name: STATUS, dtype: int64
```

In [48]:
```python
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
```

In [49]:
```python
classifiers = {
    "LogisticRegression" : LogisticRegression(),
    "KNeighbors" : KNeighborsClassifier(),
    "SVC" : SVC(),
    "DecisionTree" : DecisionTreeClassifier(),
    "RandomForest" : RandomForestClassifier(),
    "XGBoost" : XGBClassifier()
}
```

In [50]:
```python
train_scores = []
test_scores = []

for key, classifier in classifiers.items():
    classifier.fit(X_balanced, y_balanced)
    train_score = classifier.score(X_balanced, y_balanced)
    train_scores.append(train_score)
    test_score = classifier.score(X_test_balanced, y_test_balanced)
    test_scores.append(test_score)

print(train_scores)
print(test_scores)
```

```
C:\Users\admin\anaconda3\lib\site-packages\xgboost\sklearn.py:888: UserWarning: The use of la
bel encoder in XGBClassifier is deprecated and will be removed in a future release. To remove
this warning, do the following: 1) Pass option use_label_encoder=False when constructing XGBC
lassifier object; and 2) Encode your labels (y) as integers starting with 0, i.e. 0, 1, 2,
..., [num_class - 1].
```

```
[21:17:15] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.3.0/src/learner.
cc:1061: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'bi
nary:logistic' was changed from 'error' to 'logloss'. Explicitly set eval_metric if you'd lik
e to restore the old behavior.
[0.6196914146043385, 0.9793767186067828, 0.9222425908952032, 0.9949587534372135, 0.9949587534
372135, 0.9943476932477849]
[0.5574671869457254, 0.748669741042923, 0.7963816956367507, 0.8769067045051436, 0.85402625044
```

In [51]:
```python
xgb=XGBClassifier()
model=xgb.fit(X_balanced,y_balanced)
prediction=xgb.predict(X_test_balanced)
```

```
C:\Users\admin\anaconda3\lib\site-packages\xgboost\sklearn.py:888: UserWarning: The use of la
bel encoder in XGBClassifier is deprecated and will be removed in a future release. To remove
this warning, do the following: 1) Pass option use_label_encoder=False when constructing XGBC
lassifier object; and 2) Encode your labels (y) as integers starting with 0, i.e. 0, 1, 2,
..., [num_class - 1].
```

```
[21:22:54] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.3.0/src/learner.
cc:1061: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'bi
nary:logistic' was changed from 'error' to 'logloss'. Explicitly set eval_metric if you'd lik
```

In [52]:
```python
from sklearn.metrics import classification_report
print(classification_report(y_test_balanced, prediction))
```

```
              precision    recall  f1-score   support

           0       0.91      0.99      0.95      2819
           1       0.99      0.90      0.94      2819

    accuracy                           0.95      5638
   macro avg       0.95      0.95      0.95      5638
weighted avg       0.95      0.95      0.95      5638
```

In [ ]: