In [145…

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [146…

```python
df = pd.read_csv('C:/Users/Mark Stephen Thomas/Downloads/application_record.csv')
credit_df = pd.read_csv('C:/Users/Mark Stephen Thomas/Downloads/credit_record.csv')
```

In [147…

```python
df.head()
```

Out[147…

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

In [148…

```python
df['ID'].head()
```

Out[148…

```
0    5008804
1    5008805
2    5008806
3    5008808
4    5008809
Name: ID, dtype: int64
```

In [149…

```python
df[df.CODE_GENDER == "M"]
```

Out[149…

	ID	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTA
0	5008804	M	Y	Y	0	427500
1	5008805	M	Y	Y	0	427500
2	5008806	M	Y	Y	0	112500
10	5008815	M	Y	Y	0	270000
11	5112956	M	Y	Y	0	270000
...	...	...	...	...	...	
438541	6837707	M	N	Y	0	202500
438542	6837905	M	Y	Y	1	355050
438543	6837906	M	Y	Y	1	355050
438548	6839936	M	Y	Y	1	135000
438552	6840104	M	N	Y	0	135000

144117 rows × 18 columns

In [150…
```
df[(df.CODE_GENDER == "M") & (df['FLAG_OWN_CAR'] == "Y") & (df['NAME_INCOME_TYPE'] == "V
```

Out[150…

	ID	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTA
2	5008806	M	Y	Y	0	112500
10	5008815	M	Y	Y	0	270000
11	5112956	M	Y	Y	0	270000
12	6153651	M	Y	Y	0	270000
27	5008836	M	Y	Y	3	270000
...	...	...	...	...	...	
438518	6835982	M	Y	Y	0	135000
438519	6836039	M	Y	N	0	157500
438542	6837905	M	Y	Y	1	355050
438543	6837906	M	Y	Y	1	355050
438548	6839936	M	Y	Y	1	135000

41270 rows × 18 columns

In [151…
```
df.dtypes
```

Out[151…
```
ID                        int64
CODE_GENDER              object
FLAG_OWN_CAR             object
FLAG_OWN_REALTY          object
CNT_CHILDREN              int64
AMT_INCOME_TOTAL        float64
NAME_INCOME_TYPE         object
NAME_EDUCATION_TYPE      object
NAME_FAMILY_STATUS       object
NAME_HOUSING_TYPE        object
DAYS_BIRTH                int64
DAYS_EMPLOYED             int64
FLAG_MOBIL                int64
FLAG_WORK_PHONE           int64
FLAG_PHONE                int64
FLAG_EMAIL                int64
OCCUPATION_TYPE          object
CNT_FAM_MEMBERS         float64
dtype: object
```

In [152…
```
df = df.astype({"CNT_FAM_MEMBERS":'int64', "AMT_INCOME_TOTAL":'int64'})
df.dtypes
```

Out[152…
```
ID                       int64
CODE_GENDER             object
FLAG_OWN_CAR            object
FLAG_OWN_REALTY         object
CNT_CHILDREN             int64
AMT_INCOME_TOTAL         int64
```

```
NAME_INCOME_TYPE        object
NAME_EDUCATION_TYPE     object
NAME_FAMILY_STATUS      object
NAME_HOUSING_TYPE       object
DAYS_BIRTH               int64
DAYS_EMPLOYED            int64
FLAG_MOBIL               int64
FLAG_WORK_PHONE          int64
FLAG_PHONE               int64
FLAG_EMAIL               int64
OCCUPATION_TYPE         object
CNT_FAM_MEMBERS          int64
dtype: object
```

DATA CLEANING

In [156…

```
df['NAME_INCOME_TYPE'] = df['NAME_INCOME_TYPE'].astype('category')
df['NAME_EDUCATION_TYPE'] = df['NAME_EDUCATION_TYPE'].astype('category')
df['NAME_FAMILY_STATUS'] = df['NAME_FAMILY_STATUS'].astype('category')
df['NAME_HOUSING_TYPE'] = df['NAME_HOUSING_TYPE'].astype('category')
df['OCCUPATION_TYPE'] = df['OCCUPATION_TYPE'].astype('category')
df['CODE_GENDER']= df['CODE_GENDER'].astype('category')
df['FLAG_OWN_CAR']= df['FLAG_OWN_CAR'].astype('category')
df['FLAG_OWN_REALTY']= df['FLAG_OWN_REALTY'].astype('category')
```

In [157…

```
df.dtypes
```

Out[157…

```
ID                        int64
CODE_GENDER            category
FLAG_OWN_CAR           category
FLAG_OWN_REALTY        category
CNT_CHILDREN              int64
AMT_INCOME_TOTAL          int64
NAME_INCOME_TYPE       category
NAME_EDUCATION_TYPE    category
NAME_FAMILY_STATUS     category
NAME_HOUSING_TYPE      category
DAYS_BIRTH                int64
DAYS_EMPLOYED             int64
FLAG_MOBIL                int64
FLAG_WORK_PHONE           int64
FLAG_PHONE                int64
FLAG_EMAIL                int64
OCCUPATION_TYPE        category
CNT_FAM_MEMBERS           int64
dtype: object
```

In [158…

```
df.head()
```

Out[158…

	ID	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	N/
0	5008804	M	Y	Y	0	427500	
1	5008805	M	Y	Y	0	427500	
2	5008806	M	Y	Y	0	112500	
3	5008808	F	N	Y	0	270000	
4	5008809	F	N	Y	0	270000	

In [159…

```
df.isnull().sum()
```

Out[159…
```
ID                          0
CODE_GENDER                 0
FLAG_OWN_CAR                0
FLAG_OWN_REALTY             0
CNT_CHILDREN                0
AMT_INCOME_TOTAL            0
NAME_INCOME_TYPE            0
NAME_EDUCATION_TYPE         0
NAME_FAMILY_STATUS          0
NAME_HOUSING_TYPE           0
DAYS_BIRTH                  0
DAYS_EMPLOYED               0
FLAG_MOBIL                  0
FLAG_WORK_PHONE             0
FLAG_PHONE                  0
FLAG_EMAIL                  0
OCCUPATION_TYPE        134203
CNT_FAM_MEMBERS             0
dtype: int64
```

In [160…
```
df = df.dropna(how='any',axis=0)
df.isnull().sum()
```

Out[160…
```
ID                     0
CODE_GENDER            0
FLAG_OWN_CAR           0
FLAG_OWN_REALTY        0
CNT_CHILDREN           0
AMT_INCOME_TOTAL       0
NAME_INCOME_TYPE       0
NAME_EDUCATION_TYPE    0
NAME_FAMILY_STATUS     0
NAME_HOUSING_TYPE      0
DAYS_BIRTH             0
DAYS_EMPLOYED          0
FLAG_MOBIL             0
FLAG_WORK_PHONE        0
FLAG_PHONE             0
FLAG_EMAIL             0
OCCUPATION_TYPE        0
CNT_FAM_MEMBERS        0
dtype: int64
```

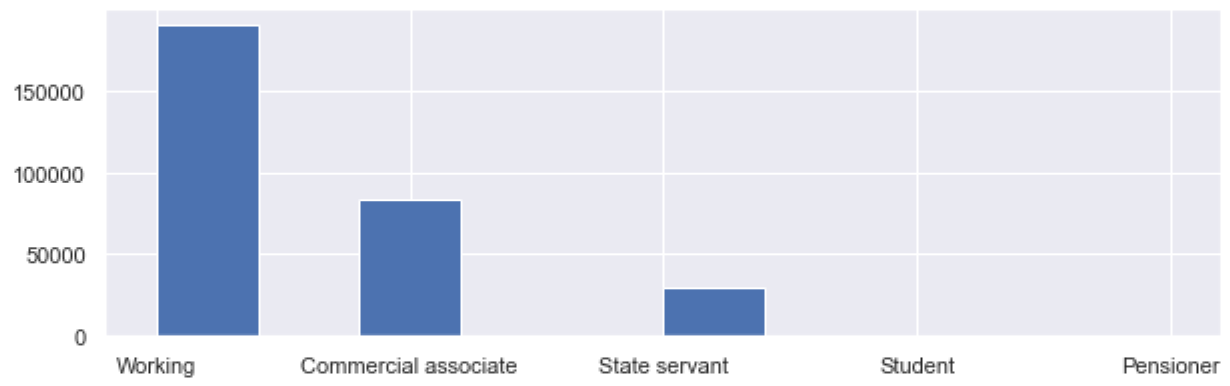
In [161…
```
df.head()
```

Out[161…

	ID	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	N
2	5008806	M	Y	Y	0	112500	
3	5008808	F	N	Y	0	270000	
4	5008809	F	N	Y	0	270000	
5	5008810	F	N	Y	0	270000	
6	5008811	F	N	Y	0	270000	

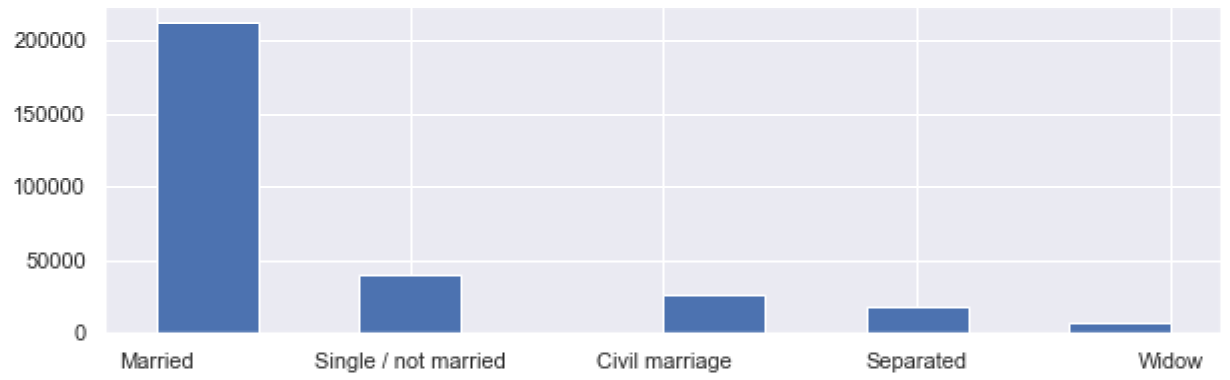
In [162…
```
df['NAME_INCOME_TYPE'].hist()
```

Out[162…
```
<AxesSubplot:>
```



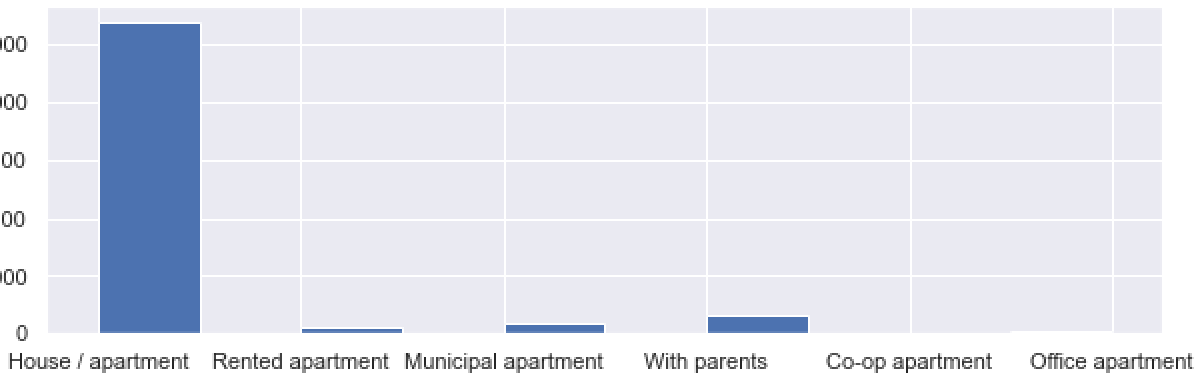
In [163…
```
df['NAME_FAMILY_STATUS'].hist()
```

Out[163…
```
<AxesSubplot:>
```

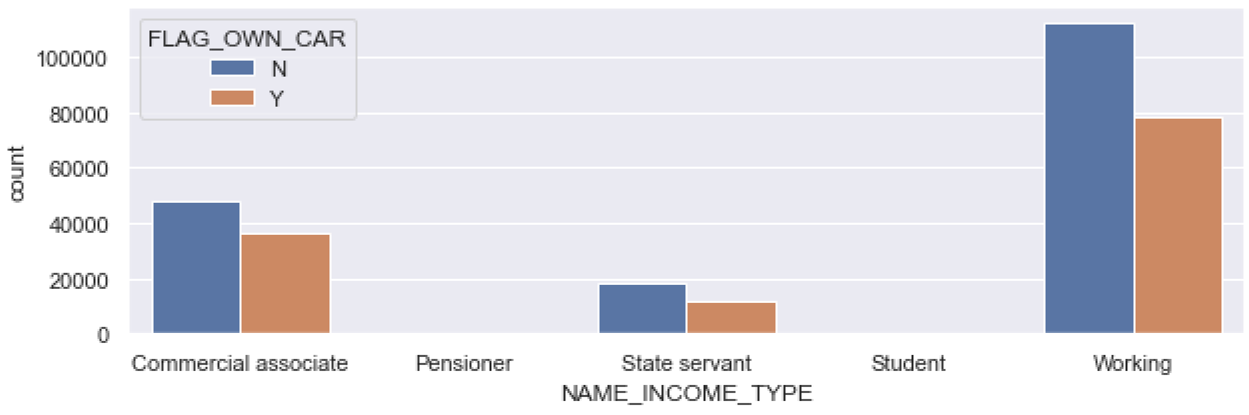


In [164…
```
df['NAME_HOUSING_TYPE'].hist()
```

Out[164…
```
<AxesSubplot:>
```



In [165…
```
from pylab import rcParams
sns.countplot(x='NAME_INCOME_TYPE',hue='FLAG_OWN_CAR',data=df)
rcParams['figure.figsize'] = 25, 10
```

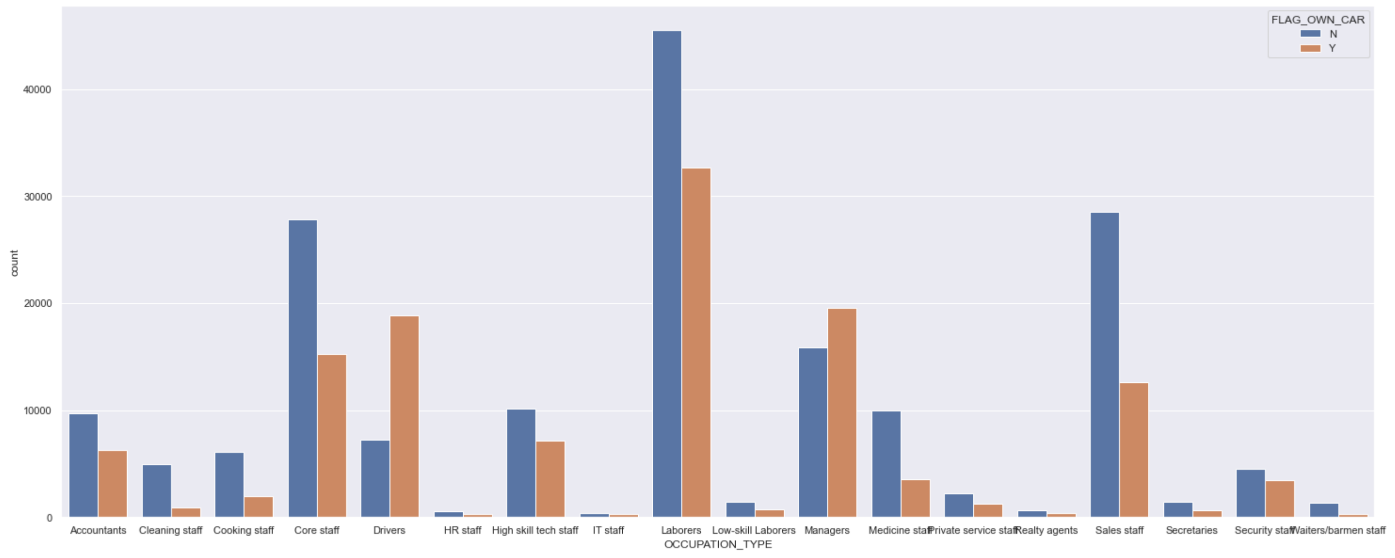


In [166…

```
sns.countplot(x='OCCUPATION_TYPE',hue='FLAG_OWN_CAR',data=df)
```

Out[166…

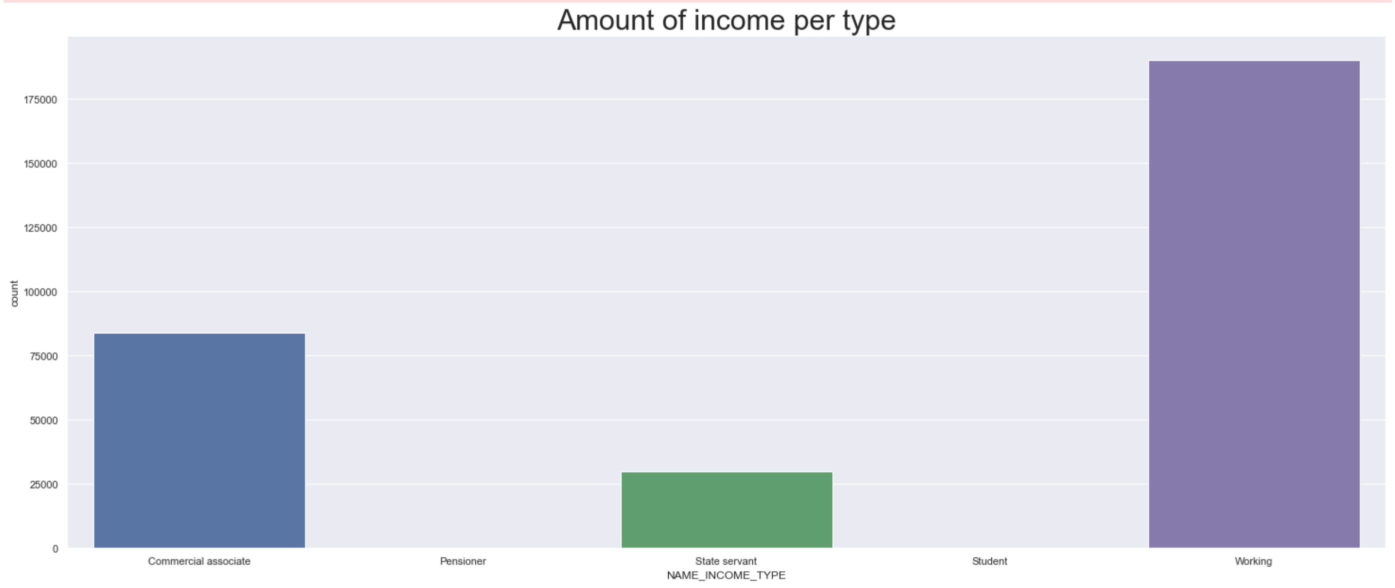
```
<AxesSubplot:xlabel='OCCUPATION_TYPE', ylabel='count'>
```



In [167…

```
p=sns.countplot(df['NAME_INCOME_TYPE'],hue_order=df.groupby('NAME_INCOME_TYPE'))
p.axes.set_title("Amount of income per type",fontsize=30)
plt.show()
```

```
D:\Anaconda\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the followi
ng variable as a keyword arg: x. From version 0.12, the only valid positional argument w
ill be `data`, and passing other arguments without an explicit keyword will result in an
error or misinterpretation.
  warnings.warn(
```



AGE

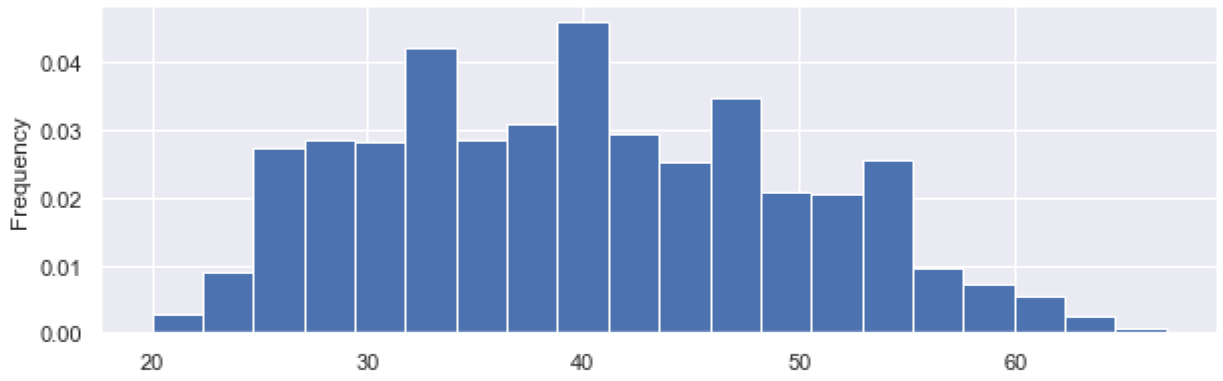
In [168…

```
sns.set(rc={'figure.figsize':(10,3)})
df['Age']=-(df['DAYS_BIRTH'])//365
print(df['Age'].value_counts(bins=10,normalize=True,sort=False))
df['Age'].plot(kind='hist',bins=20,density=True)
plt.show()
```

```
(19.951999999999998, 24.7]    0.027862
(24.7, 29.4]                  0.131097
(29.4, 34.1]                  0.165206
(34.1, 38.8]                  0.139942
(38.8, 43.5]                  0.177543
(43.5, 48.2]                  0.141345
(48.2, 52.9]                  0.097469
(52.9, 57.6]                  0.082361
(57.6, 62.3]                  0.030077
(62.3, 67.0]                  0.007097
Name: Age, dtype: float64
```

```
<ipython-input-168-a35b8cbea5b4>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  df['Age']=-(df['DAYS_BIRTH'])//365
```



Credit Record File

In [169…

```
credit_df.head()
```

Out[169…

	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 [170…

```
credit_df['STATUS'].value_counts()
```

Out[170…

```
C    442031
0    383120
X    209230
1     11090
5      1693
2       868
3       320
```

```
4       223
Name: STATUS, dtype: int64
```

In [171…
```
credit_df['STATUS'].replace(['C', 'X'],0, inplace=True)
```

In [172…
```
credit_df['STATUS'].replace(['2','3','4','5'],1, inplace=True)
```

In [173…
```
credit_df['STATUS'] = credit_df['STATUS'].astype('int')
```

In [174…
```
credit_df.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  int32
dtypes: int32(1), int64(2)
memory usage: 20.0 MB
```

In [175…
```
credit_df['STATUS'].value_counts(normalize=True)*100
```

Out[175…
```
0    98.646353
1     1.353647
Name: STATUS, dtype: float64
```

In [176…
```
credit_df_trans = credit_df.groupby('ID').agg(max).reset_index()
credit_df_trans.drop('MONTHS_BALANCE', axis=1, inplace=True)
credit_df_trans.head()
```

Out[176…

	ID	STATUS
0	5001711	0
1	5001712	0
2	5001713	0
3	5001714	0
4	5001715	0

In [177…
```
credit_df_trans['STATUS'].value_counts(normalize=True)*100
```

Out[177…
```
0    88.365771
1    11.634229
Name: STATUS, dtype: float64
```

Merging records data

In [178…
```
final_df = pd.merge(df, credit_df, on='ID', how='inner')
final_df.head()
```

Out[178…

	ID	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	N
0	5008806	M	Y	Y	0	112500	

1	5008806	M	Y	Y	0	112500
2	5008806	M	Y	Y	0	112500
3	5008806	M	Y	Y	0	112500
4	5008806	M	Y	Y	0	112500

5 rows × 21 columns

In [179…
```
final_df.shape
```
Out[179…
```
(537667, 21)
```
In [180…
```
# dropping 'ID' column as it is having only unique values (not required for ML Model)
final_df.drop('ID', axis=1, inplace=True)
# checking if there are still duplicate rows in Final Dataframe
len(final_df) - len(final_df.drop_duplicates())
```
Out[180…
```
308629
```
In [181…
```
# Dropping duplicate records
final_df = final_df.drop_duplicates()
final_df.reset_index(drop=True ,inplace=True)
```
In [182…
```
final_df.shape
```
Out[182…
```
(229038, 20)
```
In [183…
```
final_df['STATUS'].value_counts(normalize=True)*100
```
Out[183…
```
0    96.672168
1     3.327832
Name: STATUS, dtype: float64
```
In [184…
```
final_df.head()
```
Out[184…

	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	NAME_INCC
0	M	Y	Y	0	112500	
1	M	Y	Y	0	112500	
2	M	Y	Y	0	112500	
3	M	Y	Y	0	112500	
4	M	Y	Y	0	112500	

In [185…
```
cat_columns = final_df.columns[(final_df.dtypes =='category').values].tolist()
```

```
cat_columns
```

Out[185…

```
['CODE_GENDER',
 'FLAG_OWN_CAR',
 'FLAG_OWN_REALTY',
 'NAME_INCOME_TYPE',
 'NAME_EDUCATION_TYPE',
 'NAME_FAMILY_STATUS',
 'NAME_HOUSING_TYPE',
 'OCCUPATION_TYPE']
```

In [186…

```
#Converting all Non-Numerical Columns to Numerical
from sklearn.preprocessing import LabelEncoder

for col in cat_columns:
        globals()['LE_{}'.format(col)] = LabelEncoder()
        final_df[col] = globals()['LE_{}'.format(col)].fit_transform(final_df[col])
final_df.head()
```

Out[186…

	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	NAME_INCO
0	1	1	1	0	112500	
1	1	1	1	0	112500	
2	1	1	1	0	112500	
3	1	1	1	0	112500	
4	1	1	1	0	112500	

In [187…

```
for col in cat_columns:
    print(col , "  : ", globals()['LE_{}'.format(col)].classes_)
```

```
CODE_GENDER   :  ['F' 'M']
FLAG_OWN_CAR   :  ['N' 'Y']
FLAG_OWN_REALTY   :  ['N' 'Y']
NAME_INCOME_TYPE   :  ['Commercial associate' 'Pensioner' 'State servant' 'Student' 'Wor
king']
NAME_EDUCATION_TYPE   :  ['Academic degree' 'Higher education' 'Incomplete higher'
 'Lower secondary' 'Secondary / secondary special']
NAME_FAMILY_STATUS   :  ['Civil marriage' 'Married' 'Separated' 'Single / not married'
'Widow']
NAME_HOUSING_TYPE   :  ['Co-op apartment' 'House / apartment' 'Municipal apartment'
 'Office apartment' 'Rented apartment' 'With parents']
OCCUPATION_TYPE   :  ['Accountants' 'Cleaning staff' 'Cooking staff' 'Core staff' 'Drive
rs'
 'HR staff' 'High skill tech staff' 'IT staff' 'Laborers'
 'Low-skill Laborers' 'Managers' 'Medicine staff' 'Private service staff'
 'Realty agents' 'Sales staff' 'Secretaries' 'Security staff'
 'Waiters/barmen staff']
```

In [188…

```
final_df.corr()
```

Out[188…

	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCO
CODE_GENDER	1.000000	0.349407	-0.043975	0.027138	
FLAG_OWN_CAR	0.349407	1.000000	0.001607	0.067851	
FLAG_OWN_REALTY	-0.043975	0.001607	1.000000	0.015680	
CNT_CHILDREN	0.027138	0.067851	0.015680	1.000000	
AMT_INCOME_TOTAL	0.194587	0.206257	0.034400	-0.009266	
NAME_INCOME_TYPE	0.031145	-0.017421	-0.027685	0.021021	

NAME_EDUCATION_TYPE	0.046703	-0.076087	-0.002119	-0.012616
NAME_FAMILY_STATUS	-0.053884	-0.107828	-0.004062	-0.143306
NAME_HOUSING_TYPE	0.044891	-0.003124	-0.176771	0.000503
DAYS_BIRTH	0.088573	0.066360	-0.112740	0.262863
DAYS_EMPLOYED	0.127286	0.058182	-0.022618	0.069206
FLAG_MOBIL	NaN	NaN	NaN	NaN
FLAG_WORK_PHONE	0.011510	-0.006906	-0.196466	-0.015683
FLAG_PHONE	-0.012262	-0.001275	-0.059274	-0.026336
FLAG_EMAIL	-0.012060	0.011800	0.062776	-0.016856
OCCUPATION_TYPE	-0.034157	-0.047942	0.012515	-0.021263
CNT_FAM_MEMBERS	0.053613	0.113950	0.021224	0.904413
Age	-0.087840	-0.066237	0.112348	-0.262607
MONTHS_BALANCE	0.025500	-0.005145	-0.005468	0.004941
STATUS	0.011427	0.000750	-0.015304	0.004123

In [189…

```
features = final_df.drop(['STATUS'], axis=1)
label = final_df['STATUS']
```

In [190…

```
features.head()
```

Out[190…

	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	NAME_INCC
0	1	1	1	0	112500	
1	1	1	1	0	112500	
2	1	1	1	0	112500	
3	1	1	1	0	112500	
4	1	1	1	0	112500	

In [191…

```
label.head()
```

Out[191…

```
0    0
1    0
2    0
3    0
4    0
Name: STATUS, dtype: int32
```

MACHINE LEARNING MODEL

In [192…

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(features,
                                                    label,
                                                    test_size=0.2,
                                                    random_state = 10)
```

In [193…

```
# Logistic Regression

from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix

log_model = LogisticRegression()
log_model.fit(x_train, y_train)

print('Logistic Model Accuracy : ', log_model.score(x_test, y_test)*100, '%')

prediction = log_model.predict(x_test)
print('\nConfusion matrix :')
print(confusion_matrix(y_test, prediction))

print('\nClassification report:')
print(classification_report(y_test, prediction))
```

```
Logistic Model Accuracy :  96.70581557806497 %

Confusion matrix :
[[44299     0]
 [ 1509     0]]

Classification report:
              precision    recall  f1-score   support

           0       0.97      1.00      0.98     44299
           1       0.00      0.00      0.00      1509

    accuracy                           0.97     45808
   macro avg       0.48      0.50      0.49     45808
weighted avg       0.94      0.97      0.95     45808
```

```
D:\Anaconda\lib\site-packages\sklearn\metrics\_classification.py:1245: UndefinedMetricWa
rning: Precision and F-score are ill-defined and being set to 0.0 in labels with no pred
icted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
D:\Anaconda\lib\site-packages\sklearn\metrics\_classification.py:1245: UndefinedMetricWa
rning: Precision and F-score are ill-defined and being set to 0.0 in labels with no pred
icted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
D:\Anaconda\lib\site-packages\sklearn\metrics\_classification.py:1245: UndefinedMetricWa
rning: Precision and F-score are ill-defined and being set to 0.0 in labels with no pred
icted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

```
In [ ]:
```