applying the necessary libraries

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

import warnings
warnings.filterwarnings("ignore")
```

importing the dataset

```
df = pd.read_csv("credit_train.csv")
```

```
df.head()
```

	Loan ID	Customer ID	Loan Status	Current Loan Amount	Term	Credit Score	Annual Income	Years in current job
0	14dd8831-6af5-400b-83ec-68e61888a048	981165ec-3274-42f5-a3b4-d104041a9ca9	Fully Paid	445412.0	Short Term	709.0	1167493.0	8 years
1	4771cc26-131a-45db-b5aa-537ea4ba5342	2de017a3-2e01-49cb-a581-08169e83be29	Fully Paid	262328.0	Short Term	NaN	NaN	10+ years
2	4eed4e6a-aa2f-4c91-8651-ce984ee8fb26	5efb2b2b-bf11-4dfd-a572-3761a2694725	Fully Paid	99999999.0	Short Term	741.0	2231892.0	8 years
3	77598f7b-32e7-4e3b-a6e5-06ba0d98fe8a	e777faab-98ae-45af-9a86-7ce5b33b1011	Fully Paid	347666.0	Long Term	721.0	806949.0	3 years
4	d4062e70-befa-4995-8643-a0de73938182	81536ad9-5ccf-4eb8-befb-47a4d608658e	Fully Paid	176220.0	Short Term	NaN	NaN	5 years

checking the information of the dataset

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 100514 entries, 0 to 100513
Data columns (total 19 columns):
 #   Column                        Non-Null Count   Dtype  
---  ------                        --------------   -----  
 0   Loan ID                       100000 non-null  object 
 1   Customer ID                   100000 non-null  object 
 2   Loan Status                   100000 non-null  object 
 3   Current Loan Amount           100000 non-null  float64
 4   Term                          100000 non-null  object 
 5   Credit Score                  80846 non-null   float64
 6   Annual Income                 80846 non-null   float64
 7   Years in current job          95778 non-null   object 
 8   Home Ownership                100000 non-null  object 
 9   Purpose                       100000 non-null  object 
 10  Monthly Debt                  100000 non-null  float64
 11  Years of Credit History       100000 non-null  float64
 12  Months since last delinquent  46859 non-null   float64
 13  Number of Open Accounts       100000 non-null  float64
 14  Number of Credit Problems     100000 non-null  float64
 15  Current Credit Balance        100000 non-null  float64
 16  Maximum Open Credit           99998 non-null   float64
 17  Bankruptcies                  99796 non-null   float64
 18  Tax Liens                     99990 non-null   float64
dtypes: float64(12), object(7)
memory usage: 14.6+ MB
```

descriptive data of the dataset

```
df.describe()
```

	Current Loan Amount	Credit Score	Annual Income	Monthly Debt	Years of Credit History	sir del
count	1.000000e+05	80846.000000	8.084600e+04	100000.000000	100000.000000	4685
mean	1.176045e+07	1076.456089	1.378277e+06	18472.412336	18.199141	3
std	3.178394e+07	1475.403791	1.081360e+06	12174.992609	7.015324	2
min	1.080200e+04	585.000000	7.662700e+04	0.000000	3.600000	
25%	1.796520e+05	705.000000	8.488440e+05	10214.162500	13.500000	1
50%	3.122460e+05	724.000000	1.174162e+06	16220.300000	16.900000	3
75%	5.249420e+05	741.000000	1.650663e+06	24012.057500	21.700000	5
max	1.000000e+08	7510.000000	1.655574e+08	435843.280000	70.500000	17

droping the unnecessary columns

```
df.drop(["Loan ID","Customer ID"],axis=1, inplace = True)
```

checking any null values in the dataset

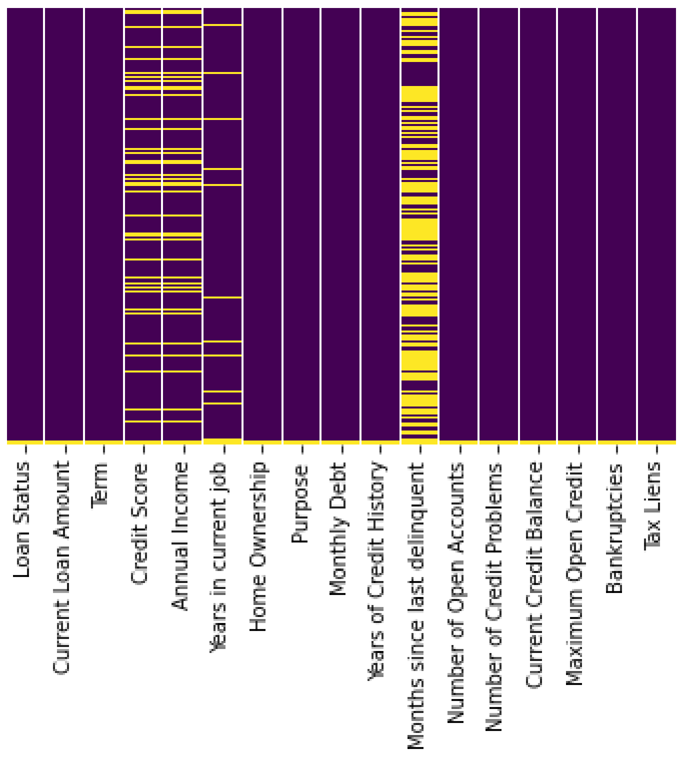
```
df.isna().sum()
```

```
Loan Status                       514
Current Loan Amount               514
Term                              514
Credit Score                    19668
Annual Income                   19668
Years in current job             4736
Home Ownership                    514
Purpose                           514
Monthly Debt                      514
Years of Credit History           514
Months since last delinquent    53655
Number of Open Accounts           514
Number of Credit Problems         514
Current Credit Balance            514
Maximum Open Credit               516
Bankruptcies                      718
Tax Liens                         524
dtype: int64
```

heatmap showing the null values

```
sns.heatmap(df.isnull(), yticklabels=False, cbar=False, cmap="viridis")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7e37a755d0>
```



droping all the columns with null values

```
df.drop(["Months since last delinquent"],axis = 1, inplace = True)
df.drop(["Credit Score"],axis = 1, inplace = True)
df.drop(["Annual Income"],axis = 1, inplace = True)
```

```
df.dropna(inplace = True)
```

```
df.head()
```

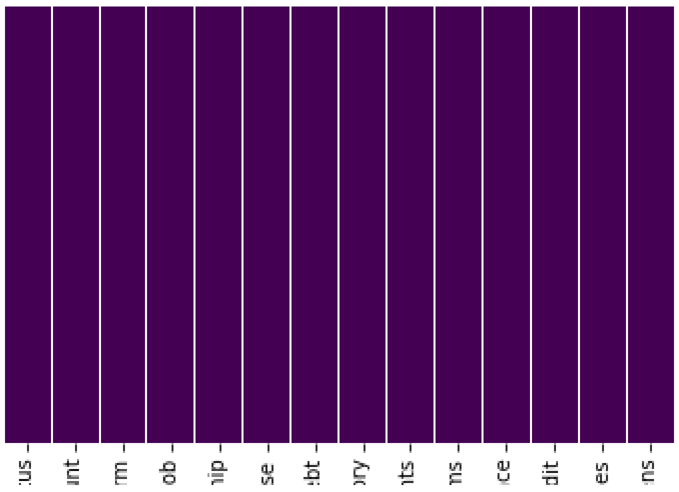
	Loan Status	Current Loan Amount	Term	Years in current job	Home Ownership	Purpose	Monthly Debt	Years of Credit History	A
0	Fully Paid	445412.0	Short Term	8 years	Home Mortgage	Home Improvements	5214.74	17.2	
1	Fully Paid	262328.0	Short Term	10+ years	Home Mortgage	Debt Consolidation	33295.98	21.1	
2	Fully Paid	99999999.0	Short Term	8 years	Own Home	Debt Consolidation	29200.53	14.9	
3	Fully Paid	347666.0	Long Term	3 years	Own Home	Debt Consolidation	8741.90	12.0	
4	Fully Paid	176220.0	Short Term	5 years	Rent	Debt Consolidation	20639.70	6.1	

```
df.isna().sum()
```

```
Loan Status                  0
Current Loan Amount          0
Term                         0
Years in current job         0
Home Ownership               0
Purpose                      0
Monthly Debt                 0
Years of Credit History      0
Number of Open Accounts      0
Number of Credit Problems    0
Current Credit Balance       0
Maximum Open Credit          0
Bankruptcies                 0
Tax Liens                    0
dtype: int64
```

```
sns.heatmap(df.isnull(), yticklabels = False, cbar = False, cmap = "viridis")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7e2f1bb550>
```



```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 95572 entries, 0 to 99998
Data columns (total 14 columns):
 #   Column                     Non-Null Count  Dtype  
---  ------                     --------------  -----  
 0   Loan Status                95572 non-null  object 
 1   Current Loan Amount        95572 non-null  float64
 2   Term                       95572 non-null  object 
 3   Years in current job       95572 non-null  object 
 4   Home Ownership             95572 non-null  object 
 5   Purpose                    95572 non-null  object 
 6   Monthly Debt               95572 non-null  float64
 7   Years of Credit History    95572 non-null  float64
 8   Number of Open Accounts    95572 non-null  float64
 9   Number of Credit Problems  95572 non-null  float64
 10  Current Credit Balance     95572 non-null  float64
 11  Maximum Open Credit        95572 non-null  float64
 12  Bankruptcies               95572 non-null  float64
 13  Tax Liens                  95572 non-null  float64
dtypes: float64(9), object(5)
memory usage: 10.9+ MB
```

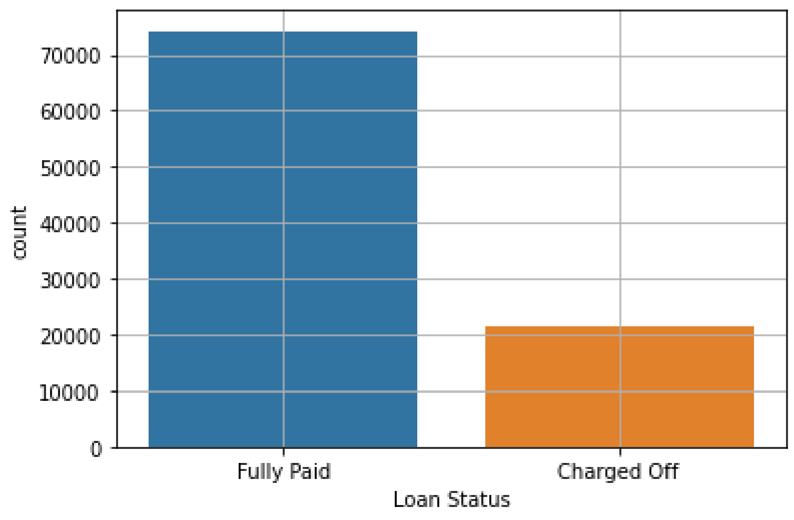
checking the value counts of the column Loan Status

```
df["Loan Status"].value_counts()
```

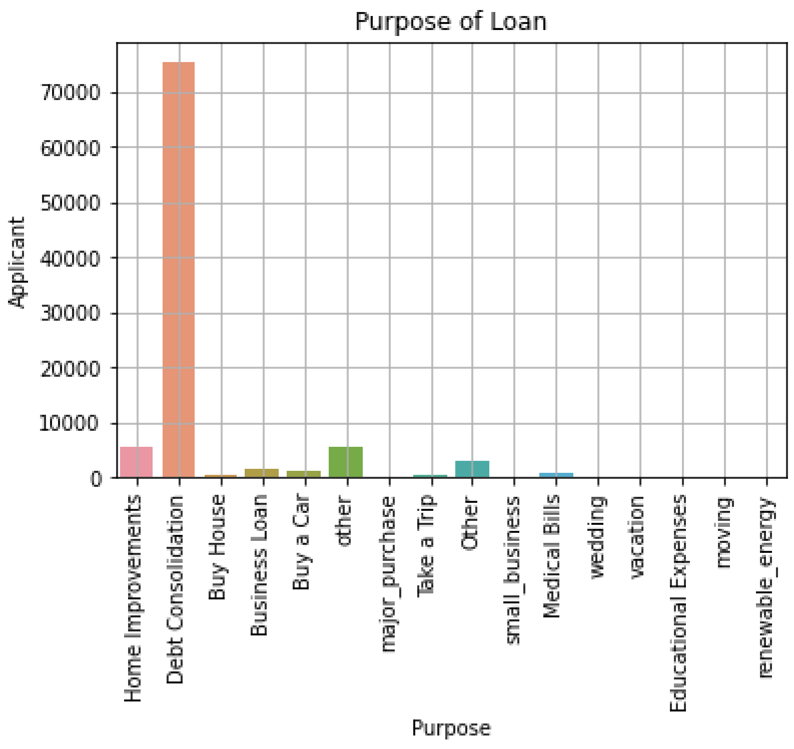
```
Fully Paid     74257
Charged Off    21315
Name: Loan Status, dtype: int64
```

countplot of the Loan Status column

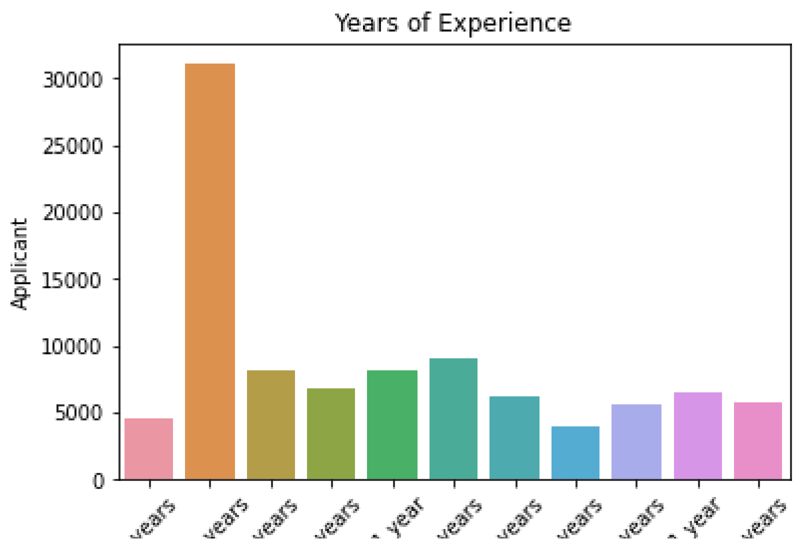
```
sns.countplot(data = df, x= "Loan Status")
plt.grid(True)
```



```
# Purpose of the Loan that was taken
sns.countplot(data = df, x= "Purpose")
plt.title("Purpose of Loan")
plt.xlabel("Purpose")
plt.ylabel("Applicant")
plt.xticks(rotation=90)
plt.grid(True)
```



```
sns.countplot(data = df, x= "Years in current job")
plt.title("Years of Experience")
plt.xlabel("Years")
plt.ylabel("Applicant")
plt.xticks(rotation=45)
plt.show()
```



```
# checking the pairplot for the Dataset
sns.pairplot(df, hue="Loan Status")
```

dividing the dataset into categorical and numerical

```
df_cat = df.select_dtypes(object)
df_num = df.select_dtypes(["int64", "float64"])
```

using label encoder

```
from sklearn.preprocessing import LabelEncoder
```

label encoding the dataset

```
for cols in df_cat:
  le = LabelEncoder()
  df_cat[cols] = le.fit_transform(df_cat[cols])
```

concatinating the dataset

```
df = pd.concat([df_cat,df_num], axis = 1)
```

```
df.head()
```

dividing the dataset into x and y

```
x = df.iloc[:,1:].values
y = df.iloc[:,:1].values
```

```
from imblearn.over_sampling import RandomOverSampler
ros = RandomOverSampler()
x, y = ros.fit_resample(x,y)
```

```
x.shape
```

```
y.shape
```

```
from sklearn.model_selection import train_test_split
```

```
xtrain,xtest,ytrain,ytest = train_test_split(x,y, test_size = 0.3, random_state = 1)
```

using standard scaler to standardise the dataset

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
xtrain = sc.fit_transform(xtrain)
xtest = sc.transform(xtest)
```

```
df.head()
```

```
import tensorflow as tf
```

training the model

```
ann = tf.keras.Sequential()
ann.add(tf.keras.layers.Dense(units=10, activation="relu"))
ann.add(tf.keras.layers.Dense(units=1, activation="sigmoid"))

ann.compile(optimizer="adam", loss="binary_crossentropy", metrics=["accuracy"])

ann.fit(xtrain, ytrain, batch_size=30, epochs=250)
ypred = ann.predict(xtest)
ypred = (ypred>0.5)
```

getting the classification report

```
from sklearn.metrics import classification_report
print(classification_report(ytest,ypred))
```

Executing (5m 7s) > ir > ... > a... > _... > (> dr... > (> _draw_li... > dr... > (> _draw_li... > dr... > (> dr... > ...