In [1]:

```
import pandas as pd
```

In [2]:

```
db = pd.read_csv('credit_train_loan.csv')
```

In [3]:

```
db
```

Out[3]:

	Loan ID	Customer ID	Loan Status	Current Loan Amount	Term	Credit Score	Annual Income	Years in current job	Ow
0	14dd8831-6af5-400b-83ec-68e61888a048	981165ec-3274-42f5-a3b4-d104041a9ca9	Fully Paid	445412.0	Short Term	709.0	1167493.0	8 years	N
1	4771cc26-131a-45db-b5aa-537ea4ba5342	2de017a3-2e01-49cb-a581-08169e83be29	Fully Paid	262328.0	Short Term	NaN	NaN	10+ years	N
2	4eed4e6a-aa2f-4c91-8651-ce984ee8fb26	5efb2b2b-bf11-4dfd-a572-3761a2694725	Fully Paid	99999999.0	Short Term	741.0	2231892.0	8 years	Ov
3	77598f7b-32e7-4e3b-a6e5-06ba0d98fe8a	e777faab-98ae-45af-9a86-7ce5b33b1011	Fully Paid	347666.0	Long Term	721.0	806949.0	3 years	Ov
4	d4062e70-befa-4995-8643-a0de73938182	81536ad9-5ccf-4eb8-befb-47a4d608658e	Fully Paid	176220.0	Short Term	NaN	NaN	5 years	
...	...	...	...	...	...	...	...	...	
100509	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
100510	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
100511	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
100512	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
100513	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

100514 rows × 19 columns

In [4]:

```
Term = db['Term']
```

In [5]:

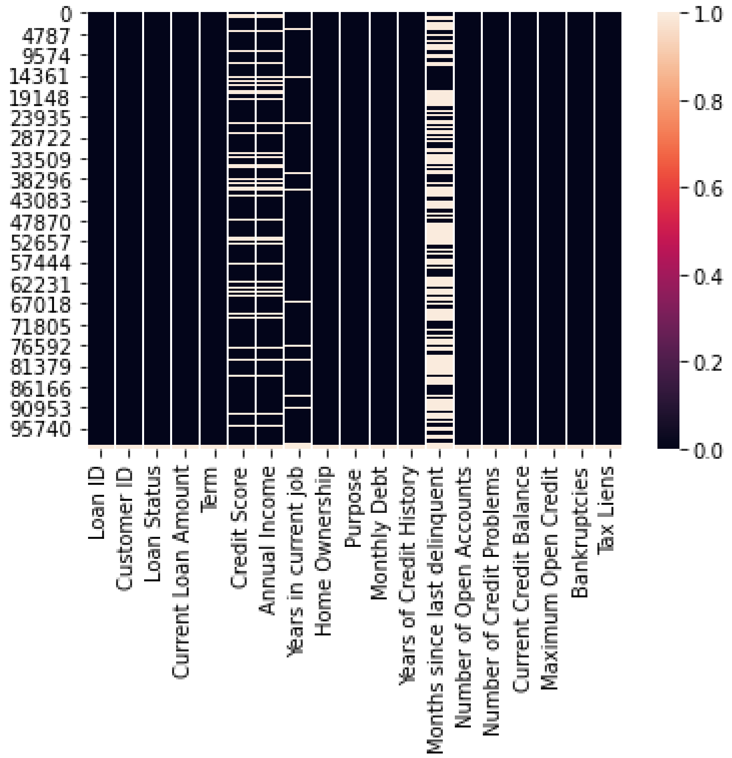
```
import seaborn as sns
```

In [6]:

```
sns.heatmap(db.isnull())
```

Out[6]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1d6c875ec40>
```



In [7]:

```
y = db.iloc[0:99999,2]
```

In [8]:

```
y.shape
```

Out[8]:

```
(99999,)
```

In [22]:

```
X = db.iloc[0:99999,4:19]
```

In [23]:

```
X
```

Out[23]:

	Term	Credit Score	Annual Income	Monthly Debt	Years of Credit History	Number of Credit Problems	Maximum Open Credit	Bankruptcies
0	Short Term	709.0	1167493.0	5214.74	17.2	1.0	416746.0	1.0
1	Short Term	NaN	NaN	33295.98	21.1	0.0	850784.0	0.0
2	Short Term	741.0	2231892.0	29200.53	14.9	1.0	750090.0	0.0
3	Long Term	721.0	806949.0	8741.90	12.0	0.0	386958.0	0.0
4	Short Term	NaN	NaN	20639.70	6.1	0.0	427174.0	0.0
...	...	...	...	...	...	...	...	...
99994	Short Term	719.0	783389.0	3727.61	17.4	0.0	259160.0	0.0
99995	Short Term	725.0	475437.0	2202.86	22.3	0.0	658548.0	0.0
99996	Short Term	732.0	1289416.0	13109.05	9.4	0.0	509234.0	0.0
99997	Short Term	742.0	1150545.0	7315.57	18.8	1.0	537548.0	1.0
99998	Short Term	746.0	1717524.0	9890.07	15.0	0.0	738254.0	0.0

99999 rows × 8 columns

In [11]:

```
X = db.drop('Purpose' , axis=1 , inplace=True)
```

In [12]:

```
X = db.drop('Tax Liens' , axis=1 , inplace=True)
```

In [13]:

```
X = db.drop('Home Ownership' , axis=1 , inplace=True)
```

In [14]:

```
X = db.drop('Number of Open Accounts' , axis=1 , inplace=True)
```

In [15]:

```
X = db.drop('Current Credit Balance' , axis=1 , inplace=True)
```

In [16]:

```
X = db.drop('Years in current job' , axis=1 , inplace=True)
```

In [17]:

```
X = db.drop('Months since last delinquent' , axis=1 , inplace=True)
```

In [18]:

```
#X = db.drop('Term' , axis=1 , inplace=True)
```

In [19]:

```
import numpy as np
```

In [20]:

```
from sklearn.impute import SimpleImputer
```

In [24]:

```
miss = SimpleImputer(missing_values=np.nan , strategy='mean')
```

In [25]:

```
miss = miss.fit(X.iloc[:,1:])
```

In [26]:

```
X.iloc[:,1:] = miss.transform(X.iloc[:,1:])
```

In [27]:

```
#miss = miss.fit(X.iloc[:,1:3])
```

In [28]:

```
#X.iloc[:,1:3] = miss.transform(X.iloc[:,1:3])
```

In [ ]:

```

```

In [29]:

```
X = pd.get_dummies(X)
```

In [30]:

```
X
```

Out[30]:

	Credit Score	Annual Income	Monthly Debt	Years of Credit History	Number of Credit Problems	Maximum Open Credit	Bankruptcies	Term_
0	709.000000	1.167493e+06	5214.74	17.2	1.0	416746.0	1.0	
1	1076.460214	1.378282e+06	33295.98	21.1	0.0	850784.0	0.0	
2	741.000000	2.231892e+06	29200.53	14.9	1.0	750090.0	0.0	
3	721.000000	8.069490e+05	8741.90	12.0	0.0	386958.0	0.0	
4	1076.460214	1.378282e+06	20639.70	6.1	0.0	427174.0	0.0	
...	...	...	...	...	...	...	...	
99994	719.000000	7.833890e+05	3727.61	17.4	0.0	259160.0	0.0	
99995	725.000000	4.754370e+05	2202.86	22.3	0.0	658548.0	0.0	
99996	732.000000	1.289416e+06	13109.05	9.4	0.0	509234.0	0.0	
99997	742.000000	1.150545e+06	7315.57	18.8	1.0	537548.0	1.0	
99998	746.000000	1.717524e+06	9890.07	15.0	0.0	738254.0	0.0	

99999 rows × 9 columns

In [31]:

```
from sklearn.linear_model import LogisticRegression
```

In [32]:

```
model = LogisticRegression()
```

In [33]:

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

In [34]:

```
X_train, X_test, y_train, y_test = train_test_split( X, y, test_size=0.20, random_state=42)
```

In [35]:

```
model.fit(X_train , y_train)
```

Out[35]:

```
LogisticRegression()
```

In [36]:

```
y_pred = model.predict(X_test)
```

In [39]:

```
from sklearn.metrics import confusion_matrix,accuracy_score
```

In [40]:

```
confusion_matrix(y_test , y_pred)
```

Out[40]:

```
array([[  938,  3571],
       [   99, 15392]], dtype=int64)
```

In [41]:

```
# computer method for calculating accuracy
accuracy_score(y_test , y_pred)
```

Out[41]:

```
0.8165
```

In [42]:

```
# human method of calculating accuracy
(938+15392)/(938+3571+99+15392) * 100
```

Out[42]:

```
81.65
```

In [43]:

```
y_test
```

Out[43]:

```
26002     Fully Paid
80420     Fully Paid
19864    Charged Off
81525     Fully Paid
57878     Fully Paid
            ...
99336    Charged Off
29311     Fully Paid
97599     Fully Paid
61294     Fully Paid
84226     Fully Paid
Name: Loan Status, Length: 20000, dtype: object
```

In [44]:

```
y_pred
```

Out[44]:

```
array(['Fully Paid', 'Fully Paid', 'Fully Paid', ..., 'Fully Paid',
       'Fully Paid', 'Fully Paid'], dtype=object)
```

In [ ]:

```

```