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

In [2]:
```
df = pd.read_csv("C:/Users/User/Desktop/PROJECTS/Credit risk/loans_full_sch
```

In [3]:
```
df.head()
```

Out[3]:

	Unnamed: 0	emp_title	emp_length	state	homeownership	annual_income	verified_incom
0	1	global config engineer	3.0	NJ	MORTGAGE	90000.0	Verifie
1	2	warehouse office clerk	10.0	HI	RENT	40000.0	Not Verifie
2	3	assembly	3.0	WI	RENT	40000.0	Source Verifie
3	4	customer service	1.0	PA	RENT	30000.0	Not Verifie
4	5	security supervisor	10.0	CA	RENT	35000.0	Verifie

5 rows × 56 columns

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 56 columns):
 #   Column                            Non-Null Count  Dtype  
---  ------                            --------------  -----  
 0   Unnamed: 0                        10000 non-null  int64  
 1   emp_title                         9167 non-null   object 
 2   emp_length                        9183 non-null   float64
 3   state                             10000 non-null  object 
 4   homeownership                     10000 non-null  object 
 5   annual_income                     10000 non-null  float64
 6   verified_income                   10000 non-null  object 
 7   debt_to_income                    9976 non-null   float64
 8   annual_income_joint               1495 non-null   float64
 9   verification_income_joint         1455 non-null   object 
 10  debt_to_income_joint              1495 non-null   float64
 11  delinq_2y                         10000 non-null  int64  
 12  months_since_last_delinq          4342 non-null   float64
 13  earliest_credit_line              10000 non-null  int64  
 14  inquiries_last_12m                10000 non-null  int64  
 15  total_credit_lines                10000 non-null  int64  
 16  open_credit_lines                 10000 non-null  int64  
 17  total_credit_limit                10000 non-null  int64  
 18  total_credit_utilized             10000 non-null  int64  
 19  num_collections_last_12m          10000 non-null  int64  
 20  num_historical_failed_to_pay      10000 non-null  int64  
 21  months_since_90d_late             2285 non-null   float64
 22  current_accounts_delinq           10000 non-null  int64  
 23  total_collection_amount_ever      10000 non-null  int64  
 24  current_installment_accounts      10000 non-null  int64  
 25  accounts_opened_24m               10000 non-null  int64  
 26  months_since_last_credit_inquiry  8729 non-null   float64
 27  num_satisfactory_accounts         10000 non-null  int64  
 28  num_accounts_120d_past_due        9682 non-null   float64
 29  num_accounts_30d_past_due         10000 non-null  int64  
 30  num_active_debit_accounts         10000 non-null  int64  
 31  total_debit_limit                 10000 non-null  int64  
 32  num_total_cc_accounts             10000 non-null  int64  
 33  num_open_cc_accounts              10000 non-null  int64  
 34  num_cc_carrying_balance           10000 non-null  int64  
 35  num_mort_accounts                 10000 non-null  int64  
 36  account_never_delinq_percent      10000 non-null  float64
 37  tax_liens                         10000 non-null  int64  
 38  public_record_bankrupt            10000 non-null  int64  
 39  loan_purpose                      10000 non-null  object 
 40  application_type                  10000 non-null  object 
 41  loan_amount                       10000 non-null  int64  
 42  term                              10000 non-null  int64  
 43  interest_rate                     10000 non-null  float64
 44  installment                       10000 non-null  float64
 45  grade                             10000 non-null  object 
 46  sub_grade                         10000 non-null  object 
 47  issue_month                       10000 non-null  object 
 48  loan_status                       10000 non-null  object 
 49  initial_listing_status            10000 non-null  object 
 50  disbursement_method               10000 non-null  object 
 51  balance                           10000 non-null  float64
 52  paid_total                        10000 non-null  float64
 53  paid_principal                    10000 non-null  float64
 54  paid_interest                     10000 non-null  float64
 55  paid_late_fees                    10000 non-null  float64
```

```
dtypes: float64(17), int64(26), object(13)
memory usage: 4.3+ MB
```

In [5]:
```
df.isnull()
```

Out[5]:

	Unnamed: 0	emp_title	emp_length	state	homeownership	annual_income	verified_inc
0	False	False	False	False	False	False	F
1	False	False	False	False	False	False	F
2	False	False	False	False	False	False	F
3	False	False	False	False	False	False	F
4	False	False	False	False	False	False	F
...	...	...	...	...	...	...	
9995	False	False	False	False	False	False	F
9996	False	False	False	False	False	False	F
9997	False	False	False	False	False	False	F
9998	False	False	False	False	False	False	F
9999	False	False	False	False	False	False	F

10000 rows × 56 columns

In [6]:
```
df.nunique()
```

Out[6]:
```
Unnamed: 0                          10000
emp_title                            4741
emp_length                             11
state                                  50
homeownership                           3
annual_income                        1463
verified_income                         3
debt_to_income                       3673
annual_income_joint                   596
verification_income_joint               3
debt_to_income_joint                 1189
delinq_2y                              12
months_since_last_delinq               97
earliest_credit_line                   53
inquiries_last_12m                     26
total_credit_lines                     78
open_credit_lines                      45
total_credit_limit                   9119
total_credit_utilized                9497
num_collections_last_12m                4
num_historical_failed_to_pay            9
months_since_90d_late                 106
current_accounts_delinq                 2
total_collection_amount_ever          896
current_installment_accounts           30
accounts_opened_24m                    26
months_since_last_credit_inquiry       25
num_satisfactory_accounts              45
num_accounts_120d_past_due              1
num_accounts_30d_past_due               2
num_active_debit_accounts              25
total_debit_limit                    1222
num_total_cc_accounts                  56
num_open_cc_accounts                   40
num_cc_carrying_balance                30
num_mort_accounts                      15
account_never_delinq_percent          282
tax_liens                               9
public_record_bankrupt                  4
loan_purpose                           12
application_type                        2
loan_amount                           612
term                                    2
interest_rate                          58
installment                          3540
grade                                   7
sub_grade                              32
issue_month                             3
loan_status                             6
initial_listing_status                  2
disbursement_method                     2
balance                              5741
paid_total                           7475
paid_principal                       5765
paid_interest                        7422
paid_late_fees                         29
dtype: int64
```

In [7]:
```
df.dropna(thresh=df.shape[0]*0.5,axis =1, inplace=True)
```

In [8]:
```
df.fillna(df.select_dtypes(include=['number']).mean(), inplace=True)
```

In [9]:
```
df_numeric = df.select_dtypes(include=['number'])
df[df_numeric.columns] = df_numeric.fillna(df_numeric.mean())
```

In [10]:
```
df_categorical = df.select_dtypes(exclude=['number'])
df[df_categorical.columns] = df_categorical.fillna(df_categorical.mode().i
```

In [11]:
```
df_encoded = pd.get_dummies(df, drop_first=True)
```

In [12]:
```
from sklearn.preprocessing import StandardScaler

numeric_columns = df_encoded.select_dtypes(include=['float64','int64']).co

scaler = StandardScaler()
df_encoded[numeric_columns] = scaler.fit_transform(df_encoded[numeric_colu
```

In [13]:
```
print(df_encoded.columns)
```

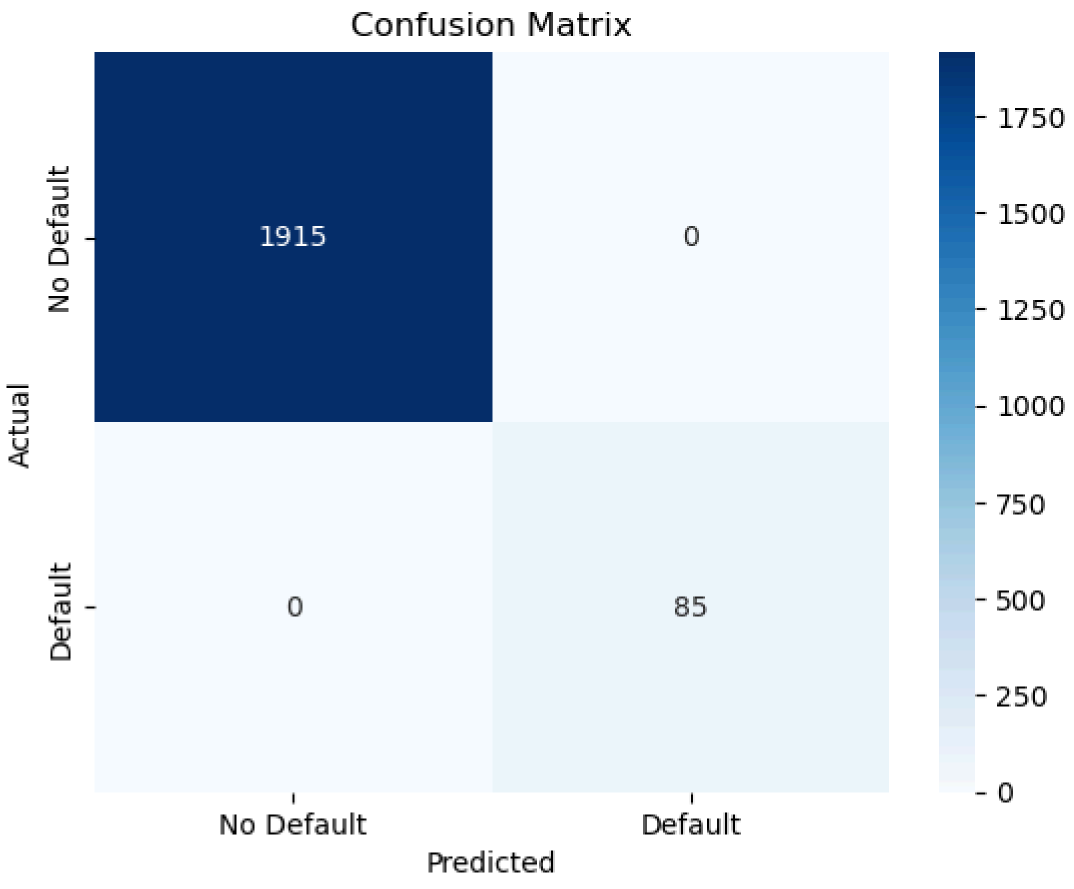
```
Index(['Unnamed: 0', 'emp_length', 'annual_income', 'debt_to_income',
       'delinq_2y', 'earliest_credit_line', 'inquiries_last_12m',
       'total_credit_lines', 'open_credit_lines', 'total_credit_limit',
       ...
       'sub_grade_G4', 'issue_month_Jan-2018', 'issue_month_Mar-2018',
       'loan_status_Current', 'loan_status_Fully Paid',
       'loan_status_In Grace Period', 'loan_status_Late (16-30 days)',
       'loan_status_Late (31-120 days)', 'initial_listing_status_whole',
       'disbursement_method_DirectPay'],
      dtype='object', length=4890)
```

In [14]:
```
from sklearn.model_selection import train_test_split

X = df_encoded.drop('loan_status_Fully Paid', axis =1)
y =df_encoded['loan_status_Fully Paid']

X_train,X_test, y_train, y_test = train_test_split(X,y, test_size = 0.2, ra
```

In [15]:
```
from sklearn.metrics import confusion_matrix, accuracy_score, roc_auc_score
from sklearn.linear_model import LogisticRegression

lr = LogisticRegression()
lr.fit(X_train,y_train)

y_pred = lr.predict(X_test)

accuracy =accuracy_score(y_test,y_pred)
auc_roc = roc_auc_score(y_test, lr.predict_proba(X_test)[:,1])

print(f'Accuracy: {accuracy}')
print(f'AUC-ROC: {auc_roc}')
```

```
Accuracy: 1.0
AUC-ROC: 1.0
```

In [16]:
```
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['No Defaul
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```



In [18]:
```
from sklearn.model_selection import RandomizedSearchCV
from sklearn.ensemble import RandomForestClassifier
param_distributions = {
    'n_estimators': [100, 200, 300],
    'max_depth': [10, 20, None],
    'min_samples_split': [2, 5, 10]
}

rf = RandomForestClassifier(random_state=42)
random_search = RandomizedSearchCV(rf, param_distributions, n_iter=10, cv=
random_search.fit(X_train, y_train)

print(f"Best Parameters: {random_search.best_params_}")
```

```
Best Parameters: {'n_estimators': 100, 'min_samples_split': 10, 'max_dept
h': None}
```

In [ ]: