Case Study

Case study on Lending Club

Uploading Data to Analyze

- import numpy as np
- import pandas as pd
- import seaborn as sns
- import matplotlib.pylab as plt
- df = pd.read_csv("C:\Lending_Class\loan\loan1.csv")
- df

Out[78]:

	id	member_id	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installment	grade	sub_grade	...	num_tl_90g_dpd_24m	num_tl_c
0	1077501	1296599	5000	5000	4975.0	36 months	10.65%	162.87	B	B2	...	NaN	
1	1077430	1314167	2500	2500	2500.0	60 months	15.27%	59.83	C	C4	...	NaN	
2	1077175	1313524	2400	2400	2400.0	36 months	15.96%	84.33	C	C5	...	NaN	
3	1076863	1277178	10000	10000	10000.0	36 months	13.49%	339.31	C	C1	...	NaN	
4	1075358	1311748	3000	3000	3000.0	60 months	12.69%	67.79	B	B5	...	NaN	
...	...	...	...	...	...	...	...	...	...	...	...	...	
39712	92187	92174	2500	2500	1075.0	36 months	8.07%	78.42	A	A4	...	NaN	
39713	90665	90607	8500	8500	875.0	36 months	10.28%	275.38	C	C1	...	NaN	
39714	90395	90390	5000	5000	1325.0	36 months	8.07%	156.84	A	A4	...	NaN	
39715	90376	89243	5000	5000	650.0	36 months	7.43%	155.38	A	A2	...	NaN	
39716	87023	86999	7500	7500	800.0	36 months	13.75%	255.43	E	E2	...	NaN	

Data cleaning

- # delete the rows where the column contains the NAN values
- # By cleaning this table about 45 columns would be removed
- # it make the analysis little bit handy
- # lets use df_1 dataframe for our analysis

- cols_to_ignore = ['emp_length']
- #df_1=df.dropna(axis=1)
- df_1=df.dropna(axis=1,how='all')
- df_1

Out[85]:

	id	member_id	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installment	grade	sub_grade	...	next_pymnt_d	last_credit_pull_d
0	1077501	1296599	5000	5000	4975.0	36 months	10.65%	162.87	B	B2	...	NaN	May-16
1	1077430	1314167	2500	2500	2500.0	60 months	15.27%	59.83	C	C4	...	NaN	Sep-13
2	1077175	1313524	2400	2400	2400.0	36 months	15.96%	84.33	C	C5	...	NaN	May-16
3	1076863	1277178	10000	10000	10000.0	36 months	13.49%	339.31	C	C1	...	NaN	Apr-16
4	1075358	1311748	3000	3000	3000.0	60 months	12.69%	67.79	B	B5	...	Jun-16	May-16
...	...	...	...	...	...	...	...	...	...	...	...	...	...
39712	92187	92174	2500	2500	1075.0	36 months	8.07%	78.42	A	A4	...	NaN	Jun-10
39713	90665	90607	8500	8500	875.0	36 months	10.28%	275.38	C	C1	...	NaN	Jul-10
39714	90395	90390	5000	5000	1325.0	36 months	8.07%	156.84	A	A4	...	NaN	Jun-07
39715	90376	89243	5000	5000	650.0	36 months	7.43%	155.38	A	A2	...	NaN	Jun-07
39716	87023	86999	7500	7500	800.0	36 months	13.75%	255.43	E	E2	...	NaN	Jun-10

39717 rows × 57 columns

Columns Details

- **## Data Sourcing - 2**
- # lets try to understand the columns usage
- # data types of the column

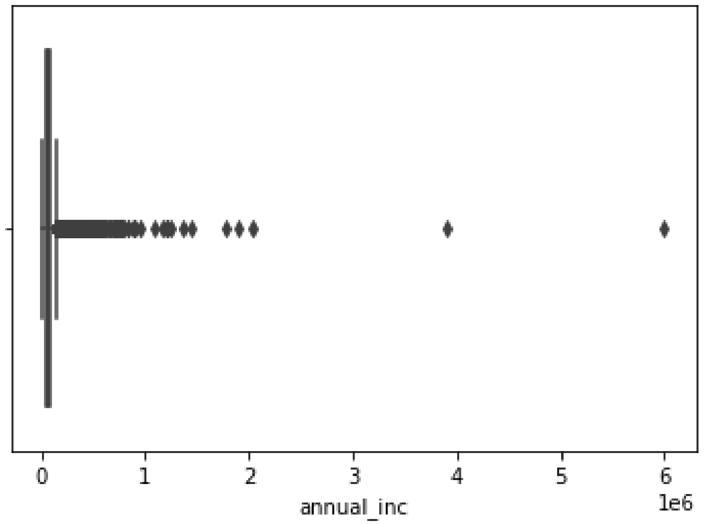
```
In [86]:  1  df_1.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 39717 entries, 0 to 39716
Data columns (total 57 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   id                      39717 non-null  int64
 1   member_id               39717 non-null  int64
 2   loan_amnt               39717 non-null  int64
 3   funded_amnt             39717 non-null  int64
 4   funded_amnt_inv         39717 non-null  float64
 5   term                    39717 non-null  object
 6   int_rate                39717 non-null  object
 7   installment             39717 non-null  float64
 8   grade                   39717 non-null  object
 9   sub_grade               39717 non-null  object
 10  emp_title               37258 non-null  object
 11  emp_length              38642 non-null  object
 12  home_ownership          39717 non-null  object
 13  annual_inc              39717 non-null  float64
 14  verification_status     39717 non-null  object
 15  issue_d                 39717 non-null  object
 16  loan_status             39717 non-null  object
 17  pymnt_plan              39717 non-null  object
```

Finding Outliers

- **#### EDA : Explanatory Data Analysis**
- # from the above annual data, very very few employees earns 60 Lacs
- # this will lay outlayres
- # now lets us find the out layers

```
In [89]:  1  sns.boxplot(x=df_1["annual_inc"])
Out[89]: <AxesSubplot:xlabel='annual_inc'>
```



Removing Outliers

- ## from the above out layers chart,
- # Max number of employees are earning less than 10 Lacs
- # calculating loan eligibililty based on annual income will not give oppertunity to the employess having less annual income
- # compared to the max earner.
- ### Considering the employees who are earing more than 12 lacs as outliers
- # Now our dataframe is df_2

In [90]:

```
df_2=df_1.drop(df_1[df_1['annual_inc']>=1200000].index)
df_2
```

Out[90]:

	id	member_id	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installment	grade	sub_grade	...	next_pymnt_d	last_credit_pull_d
0	1077501	1296599	5000	5000	4975.0	36 months	10.65%	162.87	B	B2	...	NaN	May-16
1	1077430	1314167	2500	2500	2500.0	60 months	15.27%	59.83	C	C4	...	NaN	Sep-13
2	1077175	1313524	2400	2400	2400.0	36 months	15.96%	84.33	C	C5	...	NaN	May-16
3	1076863	1277178	10000	10000	10000.0	36 months	13.49%	339.31	C	C1	...	NaN	Apr-16
4	1075358	1311748	3000	3000	3000.0	60 months	12.69%	67.79	B	B5	...	Jun-16	May-16
...	...	...	...	...	...	...	...	...	...	...	...	...	...
39712	92187	92174	2500	2500	1075.0	36 months	8.07%	78.42	A	A4	...	NaN	Jun-10
39713	90665	90607	8500	8500	875.0	36 months	10.28%	275.38	C	C1	...	NaN	Jul-10
39714	90395	90390	5000	5000	1325.0	36 months	8.07%	156.84	A	A4	...	NaN	Jun-07
39715	90376	89243	5000	5000	650.0	36 months	7.43%	155.38	A	A2	...	NaN	Jun-07
39716	87023	86999	7500	7500	800.0	36 months	13.75%	255.43	E	E2	...	NaN	Jun-10

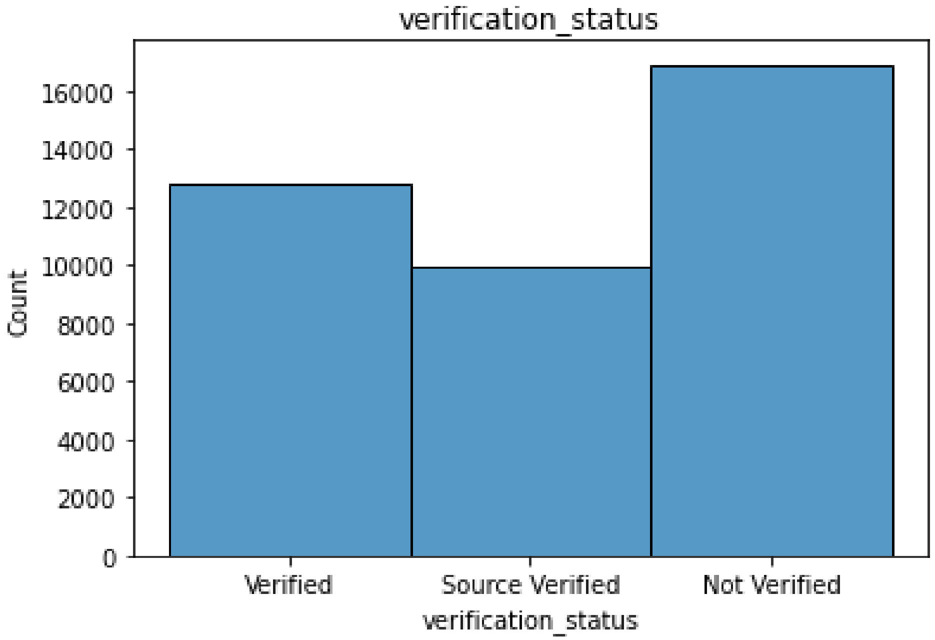
39705 rows × 57 columns

Univariate Analysis with Verification Status

- ### Univariate Analysis
- # Get the chart for Verified users
- # Get the char for loan_status# Get the chart for Verified users
- # Get the char for loan_status

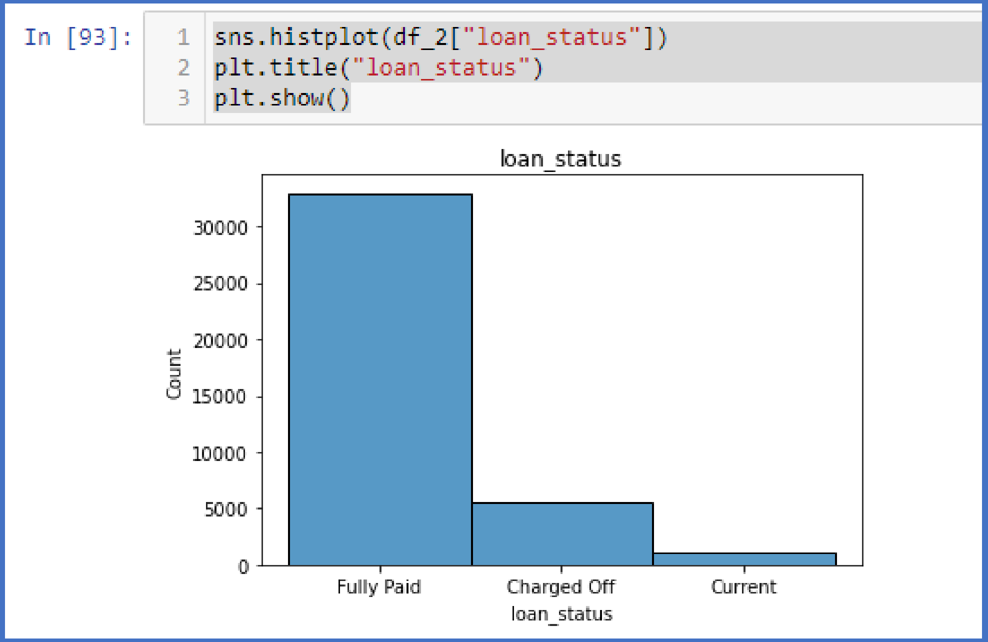
In [92]:

```
sns.histplot(df_2["verification_status"])
plt.title("verification_status")
plt.show()
```



Univariate Analysis on Loan_Status

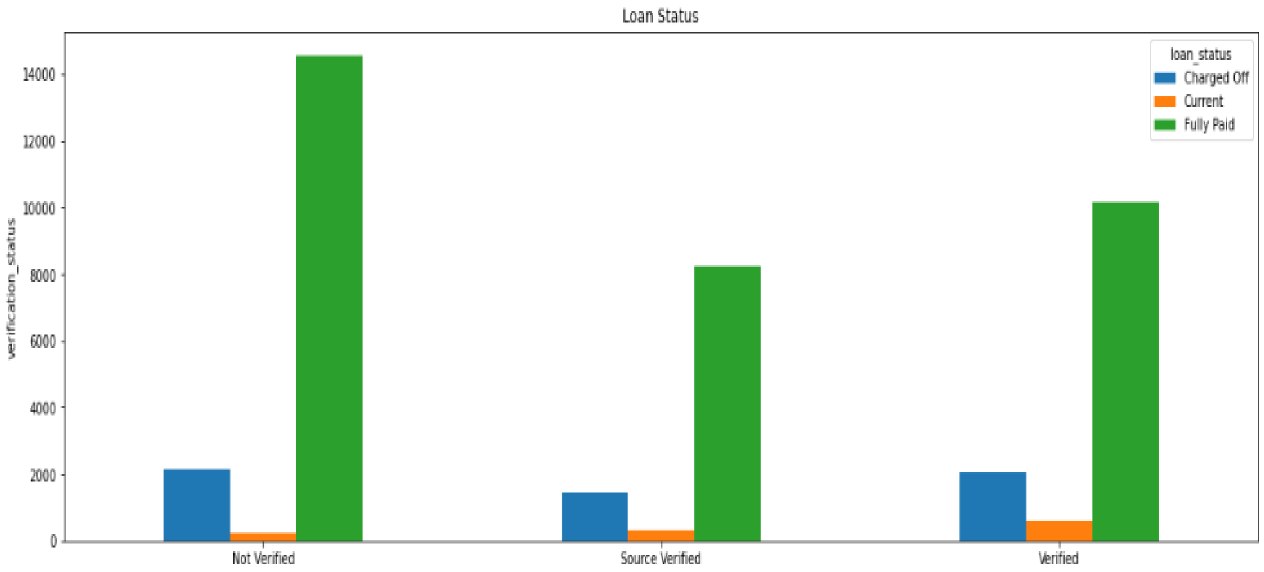
To get the percentage of loan status



Bivariate Analysis

- ## Analysis made between the loan_status with respect to verification_status
- # this is to understand how many verified user are not defaulters
- # We can see max number of applicants are paid their depts promptly irrespective of verification status

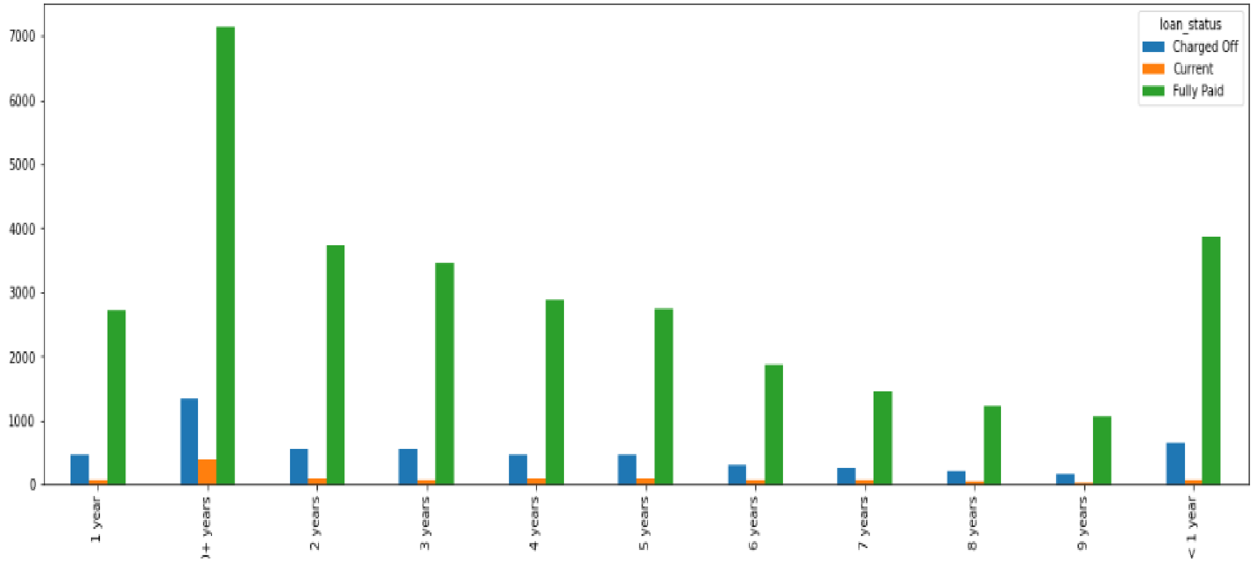
In [70]:
```
pd.crosstab(df_2.verification_status,df_2.loan_status).plot(kind="bar",figsize=(20,6))
plt.title('Loan Status')
plt.xlabel('loan_status')
plt.ylabel('verification_status')
plt.xticks(rotation =0)
plt.savefig('Loan status.png')
plt.show()
```



Bivariate Analysis between loan-status wrt Exp

- # In the below Bar plots with respect to Annual_Income and Loan_status along with candiates work experience
- # We can see higer number of applicants paid fully, at the same time we can notice quitly some high defaulters in all exp levels.
- # in such case we can have trade-off for charged_off canditates

```
In [113]:  1 pd.crosstab(df_2.emp_length,df_2.loan_status).plot(kind="bar",figsize=(20,6))
Out[113]: <AxesSubplot:xlabel='emp_length'>
```

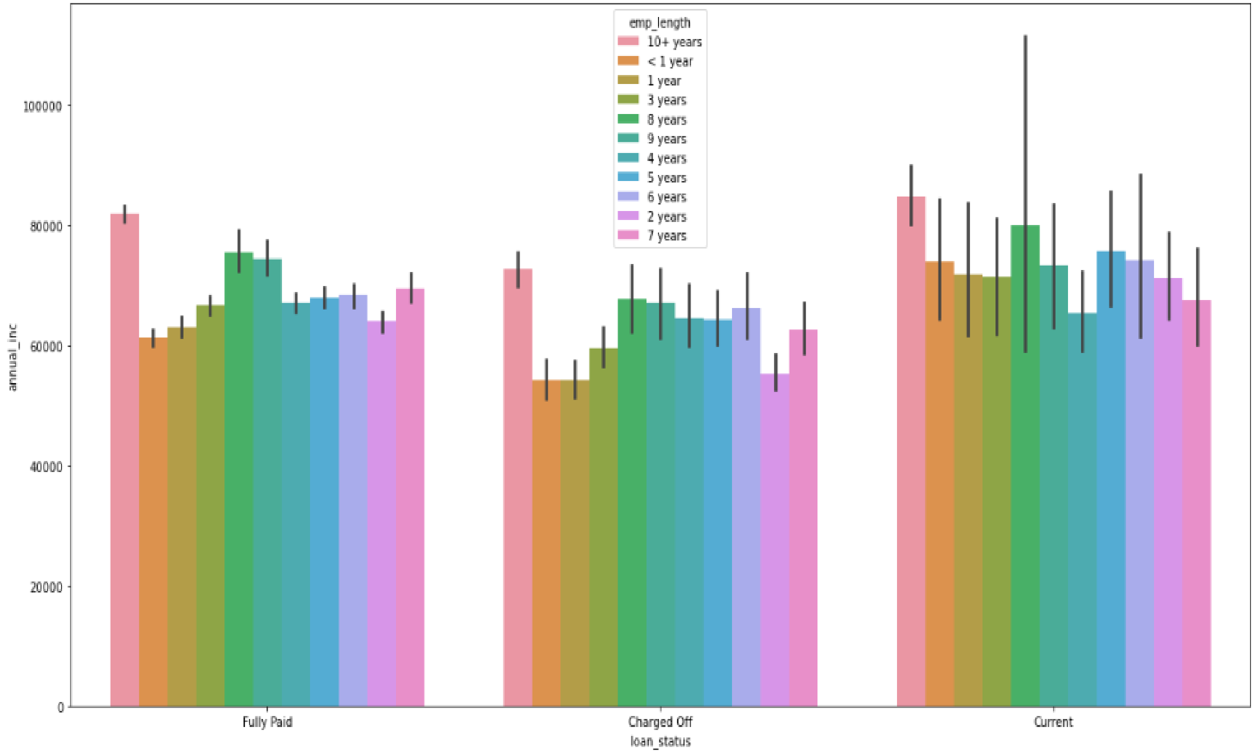


Trade-Off decision

- # The trade-off for applicants having charged_off value can be set,
- # the loan sanction can be considered for the applicants who are haveing >60K as annual income with trade-off condition
- ### Condtions
- # 1. defaulters : missed payment not more than 3 monthts
- # 2. Current loan : 40% of (applicants monthly_income - current_emi)
- # 3. Fully Paid : requested amount can be granted

```
In [112]:
plt.figure(figsize=(20,10))
sns.barplot(x=df_2["loan_status"],y=df_2["annual_inc"], hue=df_2["emp_length"])
#sns.barplot(x=df_2["emp_length"],y=df_2["annual_inc"])

Out[112]: <AxesSubplot:xlabel='loan_status', ylabel='annual_inc'>
```

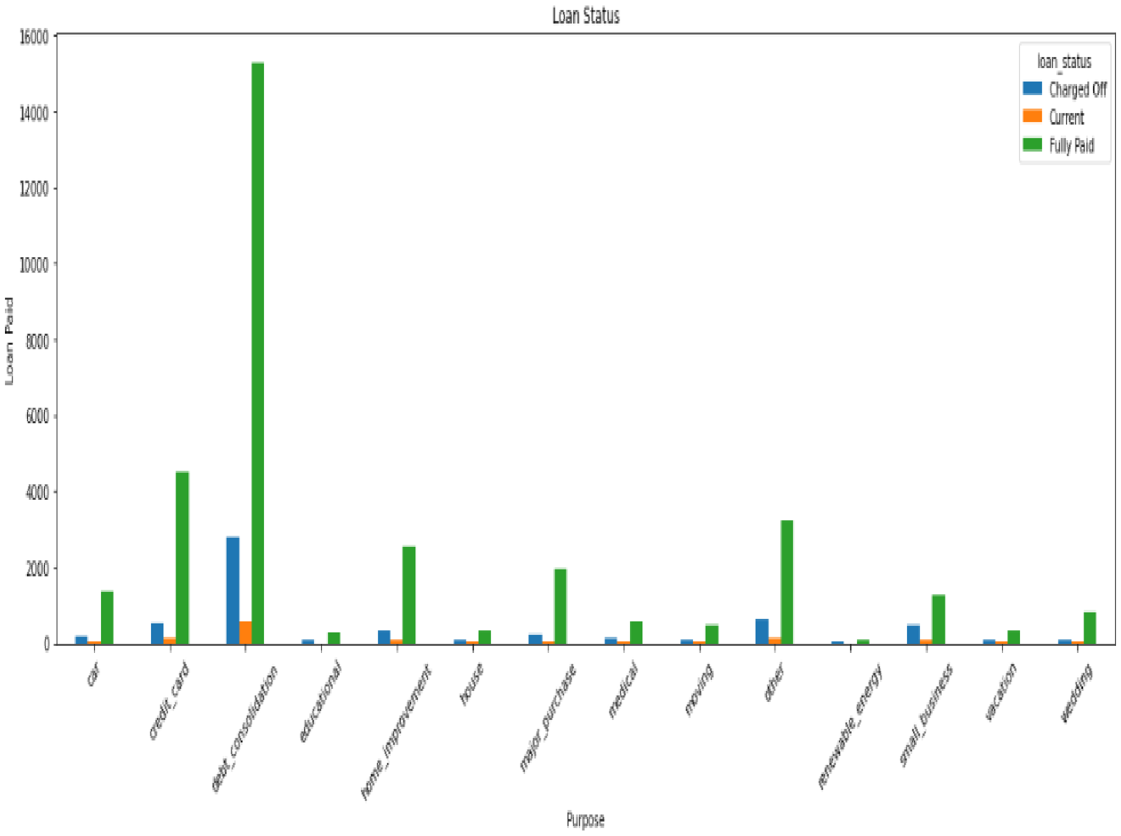


Understanding on Loan Recovery

From the below gragh we can notice loan recovery is high for Balance Transfer applicants

In [116]:

```
pd.crosstab(df_2.purpose,df_2.loan_status).plot(kind="bar",figsize=(20,6))
plt.title('Loan Status')
plt.xlabel('Purpose')
plt.ylabel('Loan Paid')
plt.xticks(rotation =45)
plt.show()
```



Thank You