Candidate Number - 933483

Importing the libraries

In [2]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import statsmodels.api as sm
from scipy import stats
from bokeh.io import output_notebook
output_notebook()

from bokeh.plotting import figure
from bokeh.io import show
```

(https://bokeh.org)
BokehJS 2.2.3 successfully loaded.

1. Load a dataset from a CSV file

In [3]:
```
#LOAD THE DATA SET OF CSV FILE

df = pd.read_csv("loanapp.csv")
df.head(3)
```

Out[3]:

	married	race	loan_decision	occupancy	loan_amount	applicant_income	num_units	num_dependants	self_employed	monthly_income	purchase_price	liquid_assets	mortage_payment_history	consumer_credit
0	True	white	reject	1	128	74	1.0	1.0	False	4583	160.0	52.0	2	
1	False	white	approve	1	128	84	1.0	0.0	False	2666	143.0	37.0	2	
2	True	white	approve	1	66	36	1.0	0.0	True	3000	110.0	19.0	2	

In [4]:
```
df.shape #it is used to display value of rows and columns of the data.
```

Out[4]: (1988, 17)

4. Check if any records in the data have any missing values; handle the missing data as appropriate (interpolate missing values, delete records with missing values, etc).

Doing 4th question first as handling null values will help in better analysis

In [5]:
```
df.isnull().sum()
#isnull function is used to see if there is any null values and adding them.
```

Out[5]:
```
married                   3
race                      0
loan_decision             0
occupancy                 0
loan_amount               0
applicant_income          0
num_units                 4
num_dependants            3
self_employed             0
monthly_income            0
purchase_price            0
liquid_assets             0
mortage_payment_history   0
consumer_credit_history   0
filed_bankruptcy          0
property_type             0
gender                   14
dtype: int64
```

Above results shows that there are null values in many variables

First method for handling null values

In [6]:
```
df1 = df.interpolate(method="linear", axis=0).ffill()
#Linear Interpolation is a method of curve fitting using linear polynomials to construct
#new data points within the range of a discrete set of known data points

#Here we have filled null values with interpolate function.
```

In [7]:
```
df1.isnull().sum()
```

Out[7]:
```
married                   0
race                      0
loan_decision             0
occupancy                 0
loan_amount               0
applicant_income          0
num_units                 0
num_dependants            0
self_employed             0
monthly_income            0
purchase_price            0
liquid_assets             0
mortage_payment_history   0
consumer_credit_history   0
filed_bankruptcy          0
property_type             0
gender                    0
dtype: int64
```

Now in above table we can see there is no null value

Second method for handling null values

In [8]:
```
df2 = df.dropna()
#Dropna function delete the rows and columns having null values from dataset.
```

In [9]:
```
df2.isnull().sum()
```

Out[9]:
```
married                   0
race                      0
loan_decision             0
occupancy                 0
loan_amount               0
applicant_income          0
num_units                 0
num_dependants            0
self_employed             0
monthly_income            0
purchase_price            0
liquid_assets             0
mortage_payment_history   0
consumer_credit_history   0
filed_bankruptcy          0
property_type             0
gender                    0
dtype: int64
```

After performing dropna function in above table we can see there is no null values.

NOW WE WILL USE (df2) DATASET FOR FURTHER ANALYSIS AS IT HAS NO NULL VALUES

2. Display descriptive statistics about the dataset

In [10]:
```
df2.describe().T
#This method is used for calculating descriptive statistics for numerical data set.
```

Out[10]:

	count	mean	std	min	25%	50%	75%	max
occupancy	1969.0	1.031996	0.192576	1.0	1.0	1.0	1.0	3.0
loan_amount	1969.0	143.505333	80.802291	2.0	100.0	126.0	165.0	980.0
applicant_income	1969.0	84.908075	87.439115	0.0	48.0	64.0	88.0	972.0
num_units	1969.0	1.122905	0.438410	1.0	1.0	1.0	1.0	4.0
num_dependants	1969.0	0.770442	1.103450	0.0	0.0	0.0	1.0	8.0
monthly_income	1969.0	5204.838497	5290.834720	0.0	2876.0	3812.0	5600.0	81000.0
purchase_price	1969.0	196.748566	128.587218	25.0	130.0	163.0	225.0	1535.0
liquid_assets	1969.0	4664.390041	67464.766307	0.0	20.0	38.0	83.0	1000000.0
mortage_payment_history	1969.0	1.705942	0.554335	1.0	1.0	2.0	2.0	4.0
consumer_credit_history	1969.0	2.110208	1.662151	1.0	1.0	1.0	2.0	6.0
property_type	1969.0	1.862367	0.535195	1.0	2.0	2.0	2.0	3.0

Here in desciptive table we can see the statistics of our data set like count, mean, std, min, max etc. The most important variable in this table are Loan_amount, Applicant_income, Monthly_income, Num_dependants which are giving important information like:-

Loan_amount(In thousands)-The average loan amount is 143.50 min loan amount is 2 and max is 980.

Applicant_income(in thousands)-The average applicant_income is 84.90 max applicant_income is 972.

Monthly_Income-Average monthly_income is 5204.83 max is 81000.

Num_dependants- The min number of dependents are 1 and max is 8.

In [11]:
```
df2.describe(exclude=[np.number])
#This will include all the categorical variables (strings) and display their summary.
```

Out[11]:

	married	race	loan_decision	self_employed	filed_bankruptcy	gender
count	1969	1969	1969	1969	1969	1969
unique	2	3	2	2	2	2
top	True	white	approve	False	False	male
freq	1298	1666	1727	1712	1834	1601

In [12]:
```
df2. describe(include='all').T
#Using include all we can see descriptive summary of whole table including categorical variables.
```

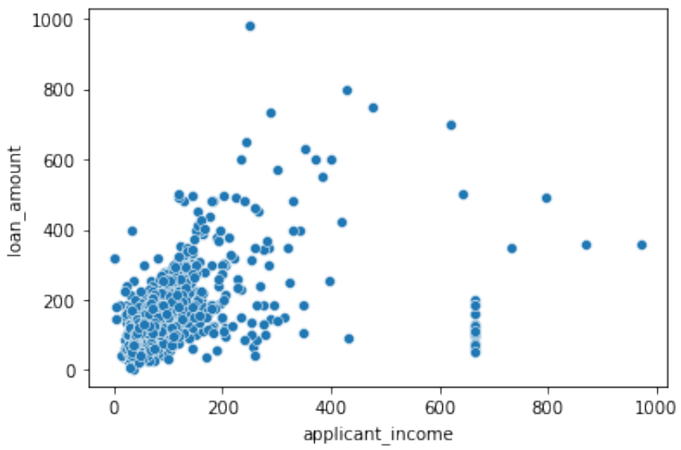
Out[12]:

	count	unique	top	freq	mean	std	min	25%	50%	75%	max
married	1969	2	True	1298	NaN	NaN	NaN	NaN	NaN	NaN	NaN
race	1969	3	white	1666	NaN	NaN	NaN	NaN	NaN	NaN	NaN
loan_decision	1969	2	approve	1727	NaN	NaN	NaN	NaN	NaN	NaN	NaN
occupancy	1969	NaN	NaN	NaN	1.032	0.192576	1	1	1	1	3
loan_amount	1969	NaN	NaN	NaN	143.505	80.8023	2	100	126	165	980
applicant_income	1969	NaN	NaN	NaN	84.9081	87.4391	0	48	64	88	972
num_units	1969	NaN	NaN	NaN	1.12291	0.43841	1	1	1	1	4
num_dependants	1969	NaN	NaN	NaN	0.770442	1.10345	0	0	0	1	8
self_employed	1969	2	False	1712	NaN	NaN	NaN	NaN	NaN	NaN	NaN
monthly_income	1969	NaN	NaN	NaN	5204.84	5290.83	0	2876	3812	5600	81000
purchase_price	1969	NaN	NaN	NaN	196.749	128.587	25	130	163	225	1535
liquid_assets	1969	NaN	NaN	NaN	4664.39	67464.8	0	20	38	83	1e+06
mortage_payment_history	1969	NaN	NaN	NaN	1.70594	0.554335	1	1	2	2	4
consumer_credit_history	1969	NaN	NaN	NaN	2.11021	1.66215	1	1	1	2	6
filed_bankruptcy	1969	2	False	1834	NaN	NaN	NaN	NaN	NaN	NaN	NaN
property_type	1969	NaN	NaN	NaN	1.86237	0.535195	1	2	2	2	3
gender	1969	2	male	1601	NaN	NaN	NaN	NaN	NaN	NaN	NaN

3. Build a graph visualizing the numerical variables of the dataset

In [13]:
```
sns.scatterplot(x='applicant_income', y='loan_amount', data=df2)
#by using seaborn library we are plotting scatterplot
```

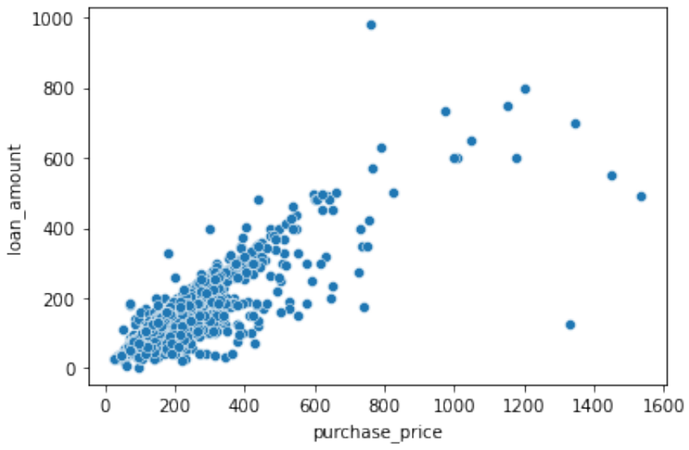
Out[13]: <AxesSubplot:xlabel='applicant_income', ylabel='loan_amount'>



The above graph shows that the loan amount and applicant income is more scattered around 200 to 300

In [14]:
```
sns.scatterplot(x='purchase_price', y='loan_amount', data=df2)
```

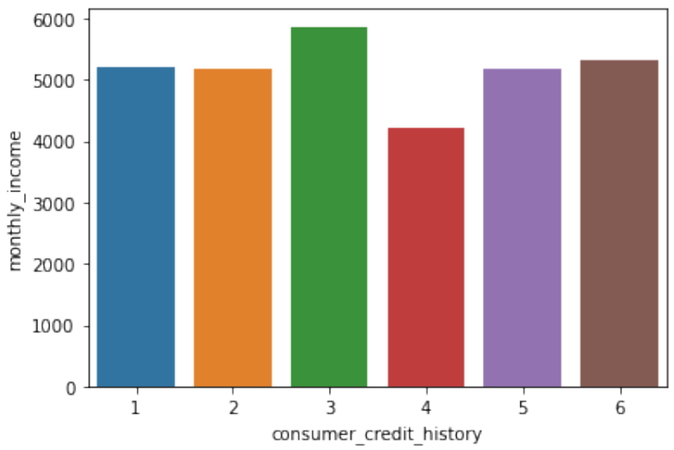
Out[14]: <AxesSubplot:xlabel='purchase_price', ylabel='loan_amount'>



The above graph shows that the loan amount and purchase price increasing with respect to each other. when purchase price is increasing loan amount is also increasing.

In [15]:
```
sns.barplot(x="consumer_credit_history", y="monthly_income", data=df2,ci=None)
#Plotting barplot using seaborn between consumer_credit_history and monthly_income.
```

Out[15]: <AxesSubplot:xlabel='consumer_credit_history', ylabel='monthly_income'>

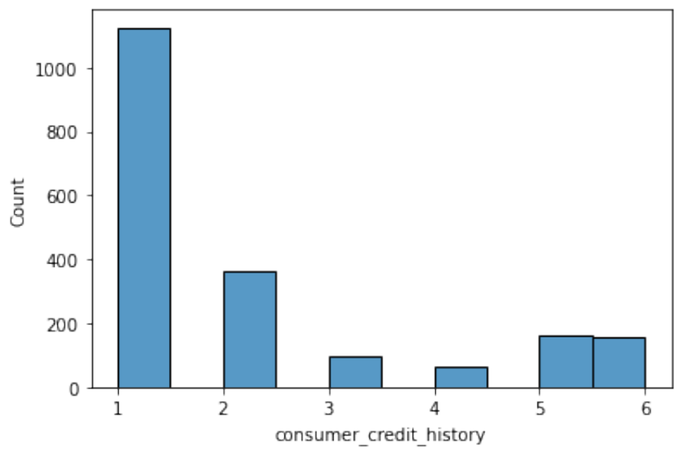


The above graph shows that the people having consumer_credit_history 3 have highest monthly income and people having consumer_credit_history 4 have lowest monthly income

5. Display the distribution of (some of) numerical variables as histograms. Provide verbal comments on the graph.

In [16]:
```
sns.histplot(df2["consumer_credit_history"],bins=10)

#Histplot function is used to plot histogram, here we are plotting frequency of consumer_credit_history .
```

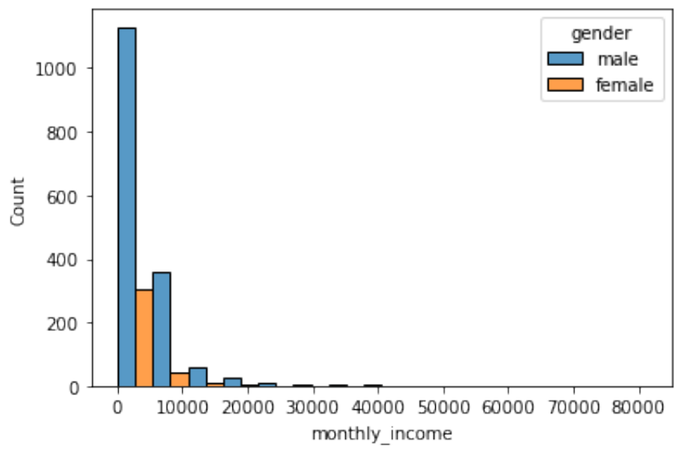
Out[16]: <AxesSubplot:xlabel='consumer_credit_history', ylabel='Count'>



By looking at the graph we can conclude that the frequency of consumer credit history of 1 is more and 3 and 4 are the least.

In [17]:
```
sns.histplot(df2, x="monthly_income", hue="gender", multiple="dodge",bins=15)

# using seaborn for plotting histogram for monthly income and taking hue as gender.
```

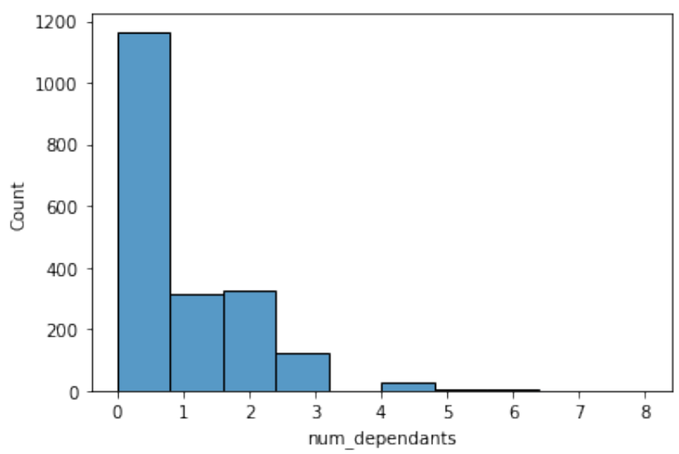
Out[17]: <AxesSubplot:xlabel='monthly_income', ylabel='Count'>



By looking at the graph we can conclude that most of the men and women monthly_income is in range between 2500 to 5000.

In [18]:
```
sns.histplot(df2["num_dependants"],bins=10)

#Histplot function is used to plot histogram, here we are plotting frequency of number of dependents.
```

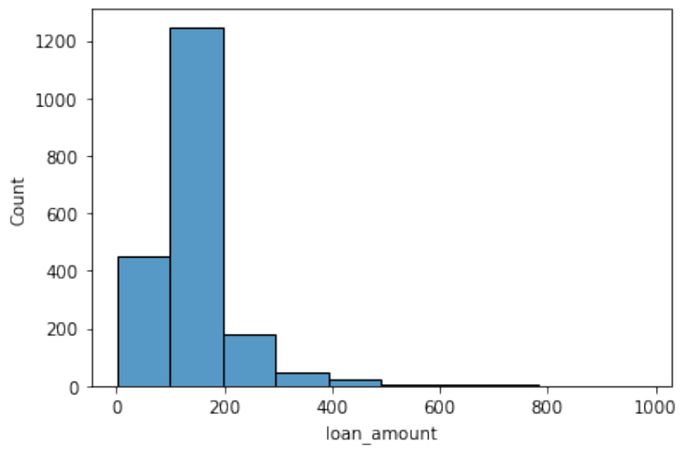
Out[18]: <AxesSubplot:xlabel='num_dependants', ylabel='Count'>



By looking at the graph we can conclude that most of the people dont have any dependents.

In [19]:
```
sns.histplot(df2["loan_amount"],bins=10)

#Histplot function is used to plot histogram, here we are plotting frequency loan amount.
```

Out[19]: <AxesSubplot:xlabel='loan_amount', ylabel='Count'>



The maximum loan amount the people are getting is between 100 to 200 (in thousands).

6. Display unique values of a categorical variable.

In [62]:
```
df_unique = df2.select_dtypes(exclude=['int', 'float'])

# Here from dataframe we are removing int and float values and storing only categorical values in new dataframe

df_unique
```

Out[62]:

	married	race	loan_decision	self_employed	filed_bankruptcy	gender
0	True	white	reject	False	False	male
1	False	white	approve	False	False	male
2	True	white	approve	True	True	male
3	True	white	approve	False	False	male
4	False	white	approve	False	False	male
...	...	...	...	...	...	...
1983	True	white	approve	False	False	male
1984	True	white	approve	False	False	male
1985	True	white	approve	True	False	male
1986	False	white	approve	True	False	male
1987	False	black	reject	True	False	female

1969 rows × 6 columns

In [63]:
```
df_unique.nunique()

#Here by using nunique function we are printing unique values of categorical variable.
```

Out[63]:
```
married            2
race               3
loan_decision      2
self_employed      2
filed_bankruptcy   2
gender             2
dtype: int64
```

In [66]:
```
for col in list(df_unique):#using for loop to iterate in dataframe df_unique
    print(col)
    print(df_[col].unique())#here we are printing unique values in each column
```

```
married
[True False]
race
['white' 'black' 'hispan']
loan_decision
['reject' 'approve']
self_employed
[False  True]
filed_bankruptcy
[False  True]
gender
['male' 'female']
```

7. Build a contingency table of two potentially related categorical variables. Conduct a statistical test of the independence between the variables. Provide verbal comments on the output.

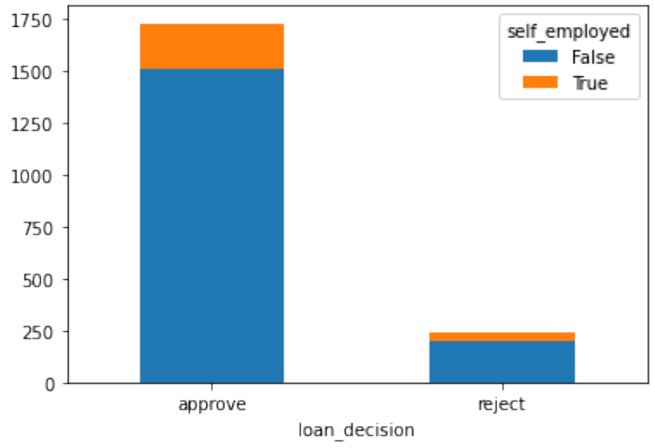
In [67]:
```
cont_table1 = pd.crosstab(df2['loan_decision'], df2["self_employed"])
#cross_tab function is used to create contingency table to seethe realtion between two or more categorical
#variables. It basically gives the tally of counts.
#Here we are making contingency table between loan descision and self_employed

cont_table1 #print the contingency table
```

Out[67]:

self_employed	False	True
loan_decision		
approve	1510	217
reject	202	40

In [68]:
```
cont_table1.plot(kind="bar", stacked=True, rot=0)#plotting a garph on the above contingency table obtained.
```

Out[68]: <AxesSubplot:xlabel='loan_decision'>



The above Graph is depicting the count of loan decision approval or rejection on the criteria whether the person is self employed or not. we can see that the people who are not self employed get more loan approval as compare to self employed

Chi-square Test between categorical variable loan_decision employment_status

Chi-Square test-It is used to determine whether two categorical variables are independent.

-The null hypothesis of the chi-square test is always that the two variables are independent.

-The alternative hypothesis is that they are dependent.

In [69]:
```
chi2, p_val, dof, expected = stats.chi2_contingency(cont_table1)
#chi2_contingency function returns four values: the chi-square value, p-value, degrees of freedom
#and a table with values expected under the independence assumption.

print(f"p-value: {p_val}")
```

```
p-value: 0.10688538967672208
```

The p-value is greater than the usual significance level of 0.05. Therefore, we accept the null hypothesis that there is no dependence between the employment_status and loan decision.

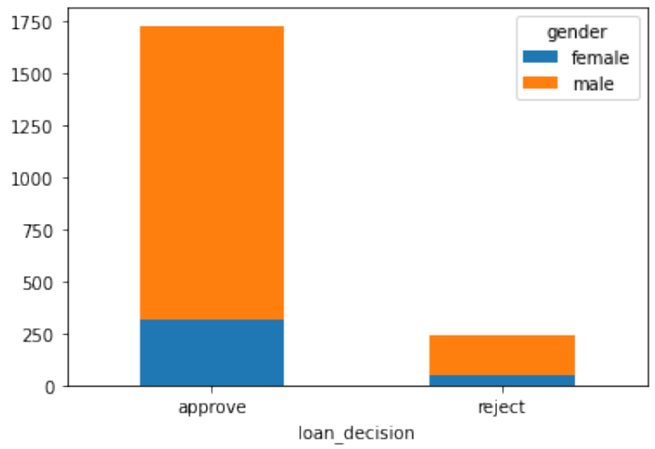
In [70]:
```
cont_table2 = pd.crosstab(df2['loan_decision'], df2['gender'])
#Here we are making contingency table between loan descision and gender.

cont_table2 # printing the contingency table
```

Out[70]:

gender	female	male
loan_decision		
approve	318	1409
reject	50	192

In [71]:
```
cont_table2.plot(kind="bar", stacked=True, rot=0)#plotting a garph on the above contingency table obtained.
```

Out[71]: <AxesSubplot:xlabel='loan_decision'>



The plot suggests that female receive less rejection as compare to the males

Chi-square Test between categorical variable loan_decision and gender

Chi-Square test-It is used to determine whether two categorical variables are independent.

-The null hypothesis of the chi-square test is always that the two variables are independent.

-The alternative hypothesis is that they are dependent.

In [72]:
```
chi2, p_val, dof, expected = stats.chi2_contingency(cont_table2)
print(f"p-value: {p_val}")
```

```
p-value: 0.45204922056492614
```

The p-value is greater than the usual significance level of 0.05. Therefore, we accept the null hypothesis that there is no dependence between the genders and loan decision.

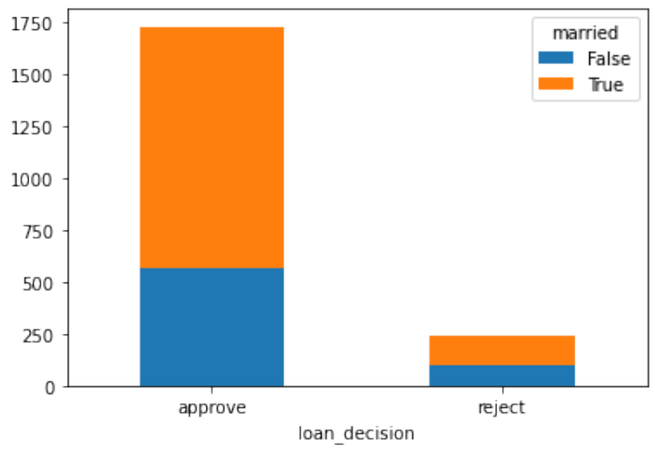
In [73]:
```
cont_table3 = pd.crosstab(df2['loan_decision'], df2['married'])

# print the contingency table
cont_table3
```

Out[73]:

married	False	True
loan_decision		
approve	569	1158
reject	102	140

In [74]:
```
cont_table3.plot(kind="bar", stacked=True, rot=0)
```

Out[74]: <AxesSubplot:xlabel='loan_decision'>



Chi-square Test between categorical variable loan_decision and marital_status

In [75]:
```
chi2, p_val, dof, expected = stats.chi2_contingency(cont_table3)
print(f"p-value: {p_val}")
```

```
p-value: 0.0058521778331842664
```

The p-value is less than the usual significance level of 0.05. Therefore, we reject the null hypothesis that there is dependence between the marital status and loan decision.

8. Retrieve a subset of the data based on two or more criteria and present descriptive statistics on the subset. Provide verbal comments on the output.

In [77]:
```
sub1 = df2[(df2.married == True) & (df2.loan_amount >= 200) & (df2.gender== 'female')]
#In the above code a subset is retrieved on the data based on following criteria like The gender should be female,
# married having loan amount greater than  equal to 200.

sub1.describe().T
```

Out[77]:

	count	mean	std	min	25%	50%	75%	max
occupancy	11.0	1.000000	0.000000	1.0	1.0	1.0	1.0	1.0
loan_amount	11.0	349.000000	127.220281	200.0	259.0	345.0	415.0	600.0
applicant_income	11.0	178.818182	77.800795	43.0	141.0	192.0	246.5	275.0
num_units	11.0	1.272727	0.904534	1.0	1.0	1.0	1.0	4.0
num_dependants	11.0	0.818182	0.750757	0.0	0.0	1.0	1.0	2.0
monthly_income	11.0	13950.727273	6689.289635	3417.0	10750.0	12579.0	17685.5	25750.0
purchase_price	11.0	486.363636	233.606624	145.0	362.0	424.0	577.5	1000.0
liquid_assets	11.0	317.454545	379.932458	34.0	89.0	150.0	401.0	1100.0
mortage_payment_history	11.0	1.181818	0.404520	1.0	1.0	1.0	1.0	2.0
consumer_credit_history	11.0	1.181818	0.404520	1.0	1.0	1.0	1.0	2.0
property_type	11.0	2.000000	0.447214	1.0	2.0	2.0	2.0	3.0

Based on subset criteria the meaningful information we got from above table is:

Females who got loan amount greater than 200 and are married have average monthly income of 13950.72. They have max dependent 2 min is 0, having max liquid asset of 1100 and min is of 34

In [34]:
```
sub2 = df2[(df2.self_employed == True) & (df2.monthly_income >= 20000) & (df2.gender== 'male')]
#In the above code a subset is retrieved on the data based on following criteria like The person should be male,
#self employed and his monthly income should be greater than or equal to 20000

sub2.describe().T
```

Out[34]:

	count	mean	std	min	25%	50%	75%	max
occupancy	12.0	1.166667	0.389249	1.0	1.00	1.0	1.0	2.0
loan_amount	12.0	392.666667	249.817509	120.0	147.50	331.5	575.0	800.0
applicant_income	12.0	338.000000	146.293603	48.0	252.75	316.5	404.5	641.0
num_units	12.0	1.000000	0.000000	1.0	1.00	1.0	1.0	1.0
num_dependants	12.0	1.083333	1.378954	0.0	0.00	0.5	2.0	4.0
monthly_income	12.0	30376.916667	8645.498686	20166.0	24029.25	29000.0	36510.0	46667.0
purchase_price	12.0	649.333333	477.061714	150.0	190.00	570.5	1075.0	1450.0
liquid_assets	12.0	650.416667	753.312136	54.0	233.75	408.5	660.0	2577.0
mortage_payment_history	12.0	1.500000	0.797724	1.0	1.00	1.0	2.0	3.0
consumer_credit_history	12.0	2.000000	1.705606	1.0	1.00	1.0	2.0	6.0
property_type	12.0	1.750000	0.452267	1.0	1.75	2.0	2.0	2.0

By looking at the above descriptive stats table we can conclude that:

1)monthly_income: The avergae monthly income of self employed male with income greater than 20000 is 30376.91

2)loan_amount: the max loan amount for self employed male with income greater than 20000 is 800 and the minimum being 120

3) The count gives the count of male who are self employed and have income greater than 20000. the total count is 12

4) Num_dependents- according to criteria of this subset the max number of dependents are only 1 and min is 0

9. Conduct a statistical test of the significance of the difference between the means of two subsets of the data. Provide verbal comments.

Independent two sample test between white and black people monthly income

The Independent Samples t Test compares the means of two independent groups in order to determine whether there is statistical evidence that the associated population means are significantly different. The Independent Samples t Test is a parametric test. This test is also known as:Two-Sample t Test.

In [78]:
```
df_race = df2[df2['race'] == 'white']['monthly_income']
# calculating mean of white peoples monthly income
df_race.mean()
```

Out[78]: 5334.286914765907

In [79]:
```
df_race2 = df2[df2['race'] == 'black']['monthly_income']
# calculating mean of black peoples monthly income
df_race2.mean()
```

Out[79]: 4715.010256410256

Null hypothesis = the mean of applicant income of white race is equal to the black race

Alternate hypothesis= The mean of applicant income of white race is significantly lower or higher than the mean of black race

Null Hypothesis $H0{:}\mu{=}0$ Alternate Hypothesis $HA{:}\mu{\neq}0$

In [37]:
```
t_val, p_val = stats.ttest_ind(df_race2, df_race)
#The ttest_ind function takes two arguments, two series corresponding to the two samples,
#and returns the t-statistic and the p-value. The function always returns the p-value for a two-tail test.

print(f"t-value: {t_val}, p-value: {p_val}")
```

t-value: -1.5152754063877643, p-value: 0.12987267804013664

The p value is greater than 0.05 therefore we accept the null hypothesis that there is no difference between the means of the two populations that the samples represent.

Independent two sample test between applicant income whose loan decision is rejected and accepted

In [80]:
```
loan = df2[df2['loan_decision'] == 'reject']['applicant_income']
loan.mean()#mean of applicant income whose loan descision is rejected
```

Out[80]: 90.66115702479338

In [81]:
```
loan1 = df2[df2['loan_decision'] == 'approve']['applicant_income']
loan1.mean()#mean of applicant income whose loan descision is rejected
```

Out[81]: 84.10191082802548

Null hypothesis = the mean of applicant income whose loan descision is rejected is equal to the mean of whose loan descision is approved.

Alternate hypothesis= The mean of applicant income whose loan descision is rejected is significantly higher or lower than whose loan descision is approved.

Null Hypothesis $H0{:}\mu{=}0$ Alternate Hypothesis $HA{:}\mu{\neq}0$

In [82]:
```
t_val, p_val = stats.ttest_ind(df_ld, df_ld1)

print(f"t-value: {t_val}, p-value: {p_val}")
```

```
t-value: 1.0929508672464752, p-value: 0.27454920793283455
```

The p value is greater than 0.05 therefore we accept the null hypothesis that there is no difference between the mean applicant income of people whose loan is decision is rejected and whose not.

10. Create pivot tables, i.e., create a table that groups the data by a certain categorical variable and displays summaries for each categorical variable. Provide verbal comments.

In [83]:
```
table1 = pd.pivot_table(df2, index=["gender","married"])
table1.T

#by using the fuction  pivot_table here we are providing gender and married as an index
```

Out[83]:

gender	female		male	
married	False	True	False	True
applicant_income	65.563492	82.060345	77.379475	91.980541
consumer_credit_history	2.154762	2.206897	2.064439	2.107445
filed_bankruptcy	0.071429	0.060345	0.069212	0.068528
liquid_assets	115.680492	85.154569	4851.765172	6017.144644
loan_amount	112.992063	136.810345	129.004773	155.807953
monthly_income	4204.753968	4666.939655	4779.544153	5621.603215
mortage_payment_history	1.793651	1.724138	1.835322	1.639594
num_dependants	0.285714	0.603448	0.181384	1.098985
num_units	1.154762	1.120690	1.171838	1.098985
occupancy	1.027778	1.008621	1.028640	1.036379
property_type	1.615079	1.844828	1.811456	1.934856
purchase_price	154.208849	185.400000	171.166468	216.000124
self_employed	0.099206	0.060345	0.128878	0.144670

Here in table we can see that females and males who are married have higher applicant and monthly income

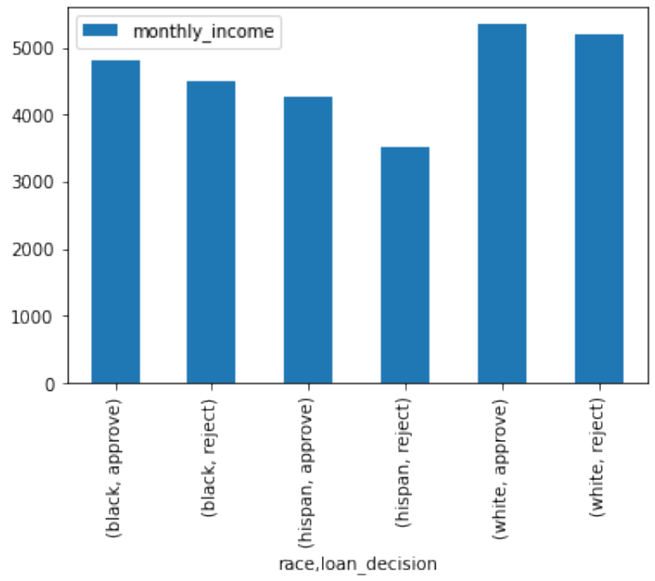
In [86]:
```
table2 = pd.pivot_table(df2,"monthly_income",["race","loan_decision"])
#Here by using pivot_table I am displaying that People in all races with higher monthly income are getting
#their loan approval.
table2
```

Out[86]:

		monthly_income
race	loan_decision	
black	approve	4815.832061
	reject	4508.640625
hispan	approve	4272.780488
	reject	3523.461538
white	approve	5347.772127
	reject	5199.967105

In [85]:
```
table2.plot(kind='bar')
```

Out[85]: <AxesSubplot:xlabel='race,loan_decision'>



From above table and graph We conclude that loan approval or rejection does not depned upon race of people.It depends upon monthly income

In [87]:
```
table3 = pd.pivot_table(df2,"loan_amount",["self_employed"])
table3
```

Out[87]:

self_employed	loan_amount
False	138.962033
True	173.770428

Here by using pivot_table I am displaying that the average loan amount of self employed people is higher than the ones who are not self employed.

In [89]:
```
table4=pd.pivot_table(df2,"monthly_income",["self_employed"],aggfunc=np.max)
table4
```

Out[89]:

self_employed	monthly_income
False	81000
True	46667

Highest income of self employed is 46667. Highest income of those who are not self employed is 81000.

11. Implement a linear regression model and interpret its output.

Here we are doing Multiple Linear Regression to predict the applicant loan amount that he will get if he gets approval for Loan

1st Model

In [46]:
```
model = sm.OLS.from_formula(
    'loan_amount ~ applicant_income +liquid_assets+num_dependants+ property_type+consumer_credit_history + mortage_payment_history+purchase_price+monthly_ir
#Create a Model from a formula and dataframe.
#we are using OLS to estimate this model in order to extend our simple model to include more independent variable


#Here Loan amount is dependent variable and applicant_income, liquid_assets, num_dependents, property_type,
#consumer_credit_history, mortage_payment_history, purchase_price and monthly_income are independent variable
```

In [47]:
```
model.summary() #Create a summary for our model
```

Out[47]:

OLS Regression Results

Dep. Variable:	loan_amount	**R-squared:**	0.709
Model:	OLS	**Adj. R-squared:**	0.708
Method:	Least Squares	**F-statistic:**	598.1
Date:	Tue, 06 Apr 2021	**Prob (F-statistic):**	0.00
Time:	01:43:51	**Log-Likelihood:**	-10225.
No. Observations:	1969	**AIC:**	2.047e+04
Df Residuals:	1960	**BIC:**	2.052e+04
Df Model:	8		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	16.2773	5.115	3.183	0.001	6.247	26.308
applicant_income	0.0497	0.014	3.591	0.000	0.023	0.077
liquid_assets	2.884e-05	1.46e-05	1.975	0.048	2.08e-07	5.75e-05
num_dependants	-1.1764	0.918	-1.281	0.200	-2.977	0.624
property_type	12.0561	1.891	6.375	0.000	8.347	15.765
consumer_credit_history	0.8686	0.599	1.450	0.147	-0.306	2.043
mortage_payment_history	0.0344	1.830	0.019	0.985	-3.554	3.623
purchase_price	0.4762	0.010	46.470	0.000	0.456	0.496
monthly_income	0.0011	0.000	4.203	0.000	0.001	0.002

Omnibus:	746.885	**Durbin-Watson:**	1.992
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	84304.408
Skew:	-0.771	**Prob(JB):**	0.00
Kurtosis:	35.019	**Cond. No.**	3.70e+05

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 3.7e+05. This might indicate that there are
strong multicollinearity or other numerical problems.

Interpretting the first derived model

After performing Multiple Linear Regression we can see in below model that the Significanse value of the following independent variables mortage_payment_history, num_dependants, consumer_credit_history is greater than 0.05 hence we have to remove them from our model to get the best model for prediction of our model

2nd Model

In [48]:
```
model = sm.OLS.from_formula(
    'loan_amount ~ applicant_income +liquid_assets+ property_type+purchase_price+monthly_income', data=df2).fit()
```

In [49]:
```
model.summary() #Create a summary for our model
```

Out[49]:

OLS Regression Results

Dep. Variable:	loan_amount	**R-squared:**	0.709
Model:	OLS	**Adj. R-squared:**	0.708
Method:	Least Squares	**F-statistic:**	955.8
Date:	Tue, 06 Apr 2021	**Prob (F-statistic):**	0.00
Time:	01:43:51	**Log-Likelihood:**	-10226.
No. Observations:	1969	**AIC:**	2.046e+04
Df Residuals:	1963	**BIC:**	2.050e+04
Df Model:	5		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	18.1608	3.769	4.818	0.000	10.769	25.553
applicant_income	0.0505	0.014	3.651	0.000	0.023	0.078
liquid_assets	2.794e-05	1.46e-05	1.915	0.056	-6.79e-07	5.66e-05
property_type	11.7690	1.870	6.294	0.000	8.102	15.436
purchase_price	0.4745	0.010	47.071	0.000	0.455	0.494
monthly_income	0.0011	0.000	4.146	0.000	0.001	0.002

Omnibus:	733.932	**Durbin-Watson:**	1.989
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	81998.675
Skew:	-0.740	**Prob(JB):**	0.00
Kurtosis:	34.580	**Cond. No.**	2.83e+05

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.83e+05. This might indicate that there are
strong multicollinearity or other numerical problems.

Interpretting the Second derived model

After performing Multiple Linear Regression we can see in below model that the Significanse value of independent variables liquid_assets is greater than 0.05 hence we have to remove it from our model to get the best model for predictions.

Final Model

In [50]:
```
model = sm.OLS.from_formula(
    'loan_amount ~ applicant_income + property_type+purchase_price+monthly_income', data=df2).fit()
```

In [51]:
```
model.summary() #Create a summary for our model
```

Out[51]:

OLS Regression Results

Dep. Variable:	loan_amount	**R-squared:**	0.708
Model:	OLS	**Adj. R-squared:**	0.708
Method:	Least Squares	**F-statistic:**	1192.
Date:	Tue, 06 Apr 2021	**Prob (F-statistic):**	0.00
Time:	01:43:51	**Log-Likelihood:**	-10228.
No. Observations:	1969	**AIC:**	2.047e+04
Df Residuals:	1964	**BIC:**	2.049e+04
Df Model:	4		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	18.2039	3.772	4.827	0.000	10.807	25.601
applicant_income	0.0503	0.014	3.634	0.000	0.023	0.077
property_type	11.8422	1.871	6.330	0.000	8.173	15.511
purchase_price	0.4741	0.010	47.009	0.000	0.454	0.494
monthly_income	0.0011	0.000	4.181	0.000	0.001	0.002

Omnibus:	730.425	**Durbin-Watson:**	1.988
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	81244.760
Skew:	-0.732	**Prob(JB):**	0.00
Kurtosis:	34.435	**Cond. No.**	3.11e+04

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 3.11e+04. This might indicate that there are
strong multicollinearity or other numerical problems.

Interpretting the Final derived model

After performing Multiple Linear Regression we can see in below model that the Significanse value of all the independent variables is lesser than 0.05. Hence all the independet variables in our model are significant and creating a impact on our model.

(1) Coefficients on the variables:

The estimated coefficients are specified in the second table. Our model is thus described by the line:

$$\textbf{Loan\_amount} = 18.2039 + 0.0503 * \textbf{applicant\_income} + 11.8422 * \textbf{property\_type} + 0.0011 * \textbf{monthly\_income} + 0.4741 * \textbf{purchase\_price} + e$$

Considering the signs on the coefficients we can state that the Loan Amount is positively affected by the applicant_income, property_type, monthly_income, purchase_price (for example:-the greater the applicant_income, the more likely the loan_amount to be higher).

(2) Significance of the variables:

The p-values on all the coefficients, indicate that the variables are significant, i.e., the factors do have a significant effect on the loan_amount.

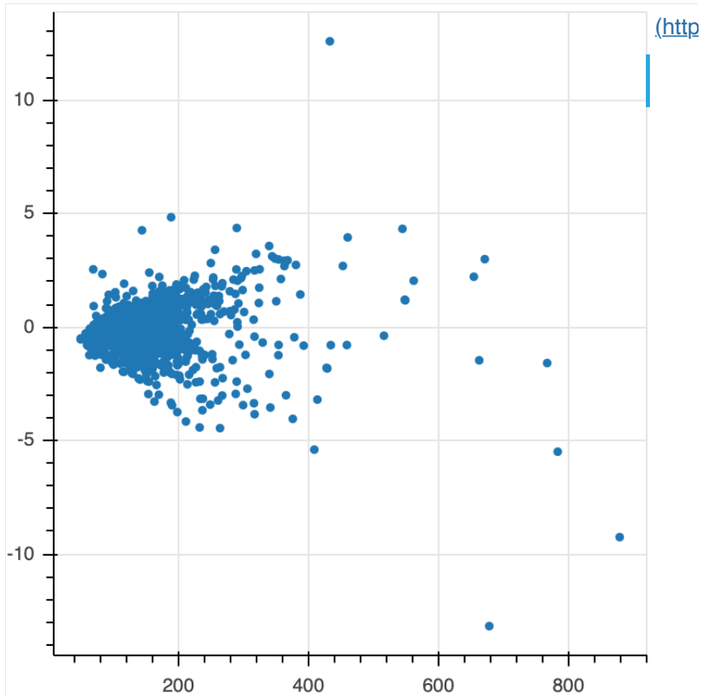
(3) Quality of the model:

The Adjusted R^2 value of 0.708 indicates that 70.8% variation in the LOAN_AMOUNT price that the applicant will get can be explained by the model containing applicant_income, property_type, monthly_income, purchase_price. This is quite high so predictions from the regression equation are fairly reliable. It also means that 29.2% of the variation is still unexplained by the Model.

Checking the assumptions of normality and zero mean of residuals

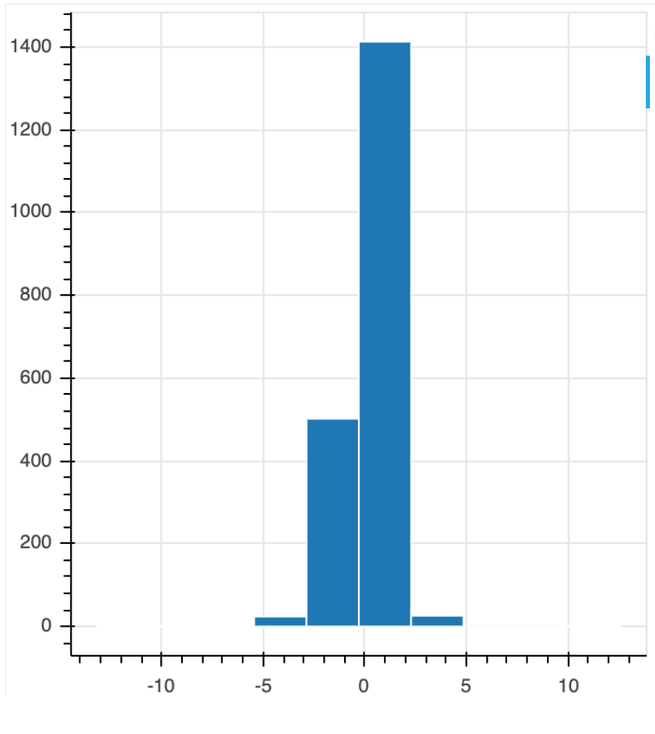
we can plot the standardized residuals and their histogram to confirm that the assumptions of normality of the distribution of residuals and of the zero mean of residuals are valid with this model.

In [52]:
```
fig = figure(height=400, width=400)

# the x axis is the fitted values
# the y axis is the standardized residuals
st_resids = model.get_influence().resid_studentized_internal
fig.circle(model.fittedvalues, st_resids)

show(fig)
```



In [53]:
```
 #create a histogram with 10 bins
hist, edges = np.histogram(st_resids, bins=10)
```

In [54]:
```
fig = figure(height=400, width=400)
fig.quad(top=hist, bottom=0, left=edges[:-1], right=edges[1:], line_color="white")
show(fig)
```



(http

The scatterplot and the histogram suggest the residuals are not equally distributed around 0 and not normally distributed. The results of the Jarque-Bera test on the residuals (the third table of the summary) also indicate that the errors are not distributed normally: the p-value equals 0.00, therefore we reject the null hypothesis of normal distribution.