In [1]: `#SWETHA JENIFER_S_15-02-23`

STEP 1:

In [2]:
```python
import pandas as pd
```

In [3]:
```python
df=pd.read_csv("train_loan.csv")
df
```

						Graduate		
7	LP001014	Male	Yes	3+	Graduate	No	3036
8	LP001018	Male	Yes	2	Graduate	No	4006
9	LP001020	Male	Yes	1	Graduate	No	12841
10	LP001024	Male	Yes	2	Graduate	No	3200
11	LP001027	Male	Yes	2	Graduate	NaN	2500
12	LP001028	Male	Yes	2	Graduate	No	3073
13	LP001029	Male	No	0	Graduate	No	1853
14	LP001030	Male	Yes	2	Graduate	No	1299
15	LP001032	Male	No	0	Graduate	No	4950
16	LP001034	Male	No	1	Not Graduate	No	3596
17	LP001036	Female	No	0	Graduate	No	3510

In [4]:
```python
df.head()
```

Out[4]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	Coapplica
0	LP001002	Male	No	0	Graduate	No	5849	
1	LP001003	Male	Yes	1	Graduate	No	4583	
2	LP001005	Male	Yes	0	Graduate	Yes	3000	
3	LP001006	Male	Yes	0	Not Graduate	No	2583	
4	LP001008	Male	No	0	Graduate	No	6000	

In [5]:
```python
df.shape
```

Out[5]: `(614, 13)`

```
In [26]: df.columns
```

```
Out[26]: Index(['Loan_ID', 'Gender', 'Married', 'Dependents', 'Education',
                'Self_Employed', 'ApplicantIncome', 'CoapplicantIncome', 'LoanAmount',
                'Loan_Amount_Term', 'Credit_History', 'Property_Area'],
               dtype='object')
```

```
In [27]: df.dtypes
```

```
Out[27]: Loan_ID               object
         Gender                object
         Married               object
         Dependents             int64
         Education             object
         Self_Employed         object
         ApplicantIncome        int64
         CoapplicantIncome    float64
         LoanAmount           float64
         Loan_Amount_Term     float64
         Credit_History       float64
         Property_Area         object
         dtype: object
```

```
In [28]: df.info
```

```
Out[28]: <bound method DataFrame.info of        Loan_ID  Gender Married  Dependents
         Education Self_Employed  \
         0     LP001002    Male      No          0       Graduate          No
         1     LP001003    Male     Yes          1       Graduate          No
         2     LP001005    Male     Yes          0       Graduate         Yes
         3     LP001006    Male     Yes          0   Not Graduate          No
         4     LP001008    Male      No          0       Graduate          No
         5     LP001011    Male     Yes          2       Graduate         Yes
         6     LP001013    Male     Yes          0   Not Graduate          No
         7     LP001014    Male     Yes          3       Graduate          No
         8     LP001018    Male     Yes          2       Graduate          No
         9     LP001020    Male     Yes          1       Graduate          No
         10    LP001024    Male     Yes          2       Graduate          No
         11    LP001027    Male     Yes          2       Graduate          No
         12    LP001028    Male     Yes          2       Graduate          No
         13    LP001029    Male      No          0       Graduate          No
         14    LP001030    Male     Yes          2       Graduate          No
         15    LP001032    Male      No          0       Graduate          No
         16    LP001034    Male      No          1   Not Graduate          No
         17    LP001036  Female      N          0
```

In [29]: `df.info()`

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 12 columns):
Loan_ID              614 non-null object
Gender               614 non-null object
Married              614 non-null object
Dependents           614 non-null int64
Education            614 non-null object
Self_Employed        614 non-null object
ApplicantIncome      614 non-null int64
CoapplicantIncome    614 non-null float64
LoanAmount           614 non-null float64
Loan_Amount_Term     614 non-null float64
Credit_History       614 non-null float64
Property_Area        614 non-null object
dtypes: float64(4), int64(2), object(6)
memory usage: 57.6+ KB
```

```
In [30]:  df.Gender.value_counts
```

```
Out[30]:  <bound method IndexOpsMixin.value_counts of 0          Male
          1          Male
          2          Male
          3          Male
          4          Male
          5          Male
          6          Male
          7          Male
          8          Male
          9          Male
          10         Male
          11         Male
          12         Male
          13         Male
          14         Male
          15         Male
          16         Male
          17       Female
          18         Male
          19         Male
          20         Male
          21         Male
          22         Male
          23         Male
          24         Male
          25         Male
          26         Male
          27         Male
          28         Male
          29       Female
                    ...
          584        Male
          585        Male
          586        Male
          587      Female
          588        Male
          589        Male
          590        Male
          591        Male
          592        Male
          593        Male
          594        Male
          595        Male
          596        Male
          597        Male
          598        Male
          599        Male
          600      Female
          601        Male
          602        Male
          603        Male
          604      Female
          605        Male
          606        Male
          607        Male
```

```
608      Male
609    Female
610      Male
611      Male
612      Male
613    Female
Name: Gender, Length: 614, dtype: object>
```

STEP 2:

In [6]:
```python
df["Dependents"].fillna("NO_dep",inplace=True)
```

```
In [7]:  df["Dependents"]
```

```
Out[7]:  0          0
         1          1
         2          0
         3          0
         4          0
         5          2
         6          0
         7          3+
         8          2
         9          1
         10         2
         11         2
         12         2
         13         0
         14         2
         15         0
         16         1
         17         0
         18         0
         19         0
         20         0
         21         1
         22         0
         23         2
         24         1
         25         0
         26         0
         27         2
         28         0
         29         2
                   ...
         584        1
         585        1
         586        0
         587        0
         588        0
         589        2
         590        0
         591        2
         592        3+
         593        0
         594        0
         595        0
         596        2
         597      NO_dep
         598        0
         599        2
         600        3+
         601        0
         602        3+
         603        0
         604        1
         605        0
         606        1
         607        2
```

```
608        0
609        0
610        3+
611        1
612        2
613        0
Name: Dependents, Length: 614, dtype: object
```

In [24]:
```python
dept={"0":0,"1":1,"2":2,"3+":3,"NO_dep":0}
df.Dependents=[dept[item]for item in df.Dependents]
```

In [25]:
```python
df['Dependents'].astype(int)
```

Out[25]:
```
0       0
1       1
2       0
3       0
4       0
5       2
6       0
7       3
8       2
9       1
10      2
11      2
12      2
13      0
14      2
15      0
16      1
17      0
18      0
19      0
20      0
21      1
22      0
23      2
24      1
25      0
26      0
27      2
28      0
29      2
       ..
584     1
585     1
586     0
587     0
588     0
589     2
590     0
591     2
592     3
593     0
594     0
595     0
596     2
597     0
598     0
599     2
600     3
601     0
602     3
603     0
604     1
605     0
606     1
607     2
```

```
608     0
609     0
610     3
611     1
612     2
613     0
Name: Dependents, Length: 614, dtype: int32
```

In [8]:
```python
df['Gender'].fillna(df["Gender"].mode()[0],inplace=True)
df['Married'].fillna(df['Married'].mode()[0],inplace=True)
df['Dependents'].fillna(df['Dependents'].mode()[0],inplace=True)
df['Education'].fillna(df['Education'].mode()[0],inplace=True)
df['Self_Employed'].fillna(df['Self_Employed'].mode()[0],inplace=True)
df['Credit_History'].fillna(df['Credit_History'].mode()[0],inplace=True)
```

In [9]:
```python
df['LoanAmount'].fillna(df['LoanAmount'].mean(),inplace=True)
df['Loan_Amount_Term'].fillna(df['Loan_Amount_Term'].mean(),inplace=True)
```

In [10]:
```python
df.drop(["Loan_ID"],axis=1)
```

Out[10]:

|	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncon
0	Male	No	0	Graduate	No	5849	0
1	Male	Yes	1	Graduate	No	4583	1508
2	Male	Yes	0	Graduate	Yes	3000	0
3	Male	Yes	0	Not Graduate	No	2583	2358
4	Male	No	0	Graduate	No	6000	0
5	Male	Yes	2	Graduate	Yes	5417	4196
6	Male	Yes	0	Not Graduate	No	2333	1516
7	Male	Yes	3+	Graduate	No	3036	2504
8	Male	Yes	2	Graduate	No	4006	1526
9	Male	Yes	1	Graduate	No	12841	10968

STEP 3:

In [11]:
```python
import seaborn as sns
import matplotlib.pyplot as plt
```

In [23]:
```python
sns.countplot(x='Married',data=df)
plt.show()
```
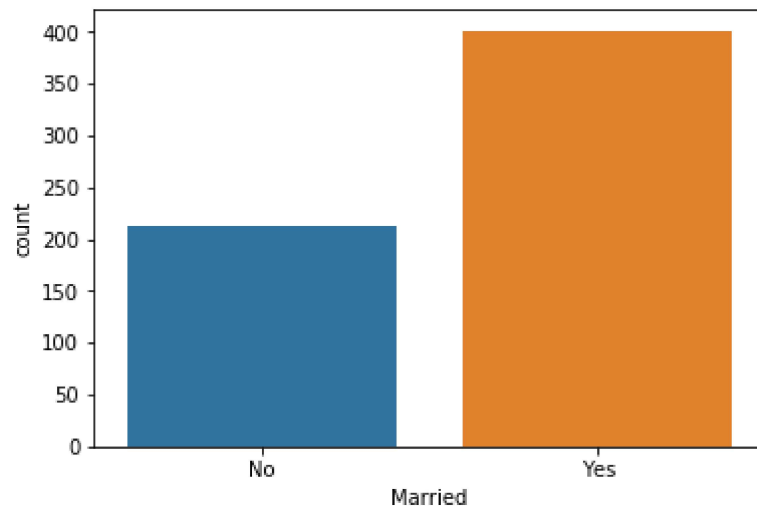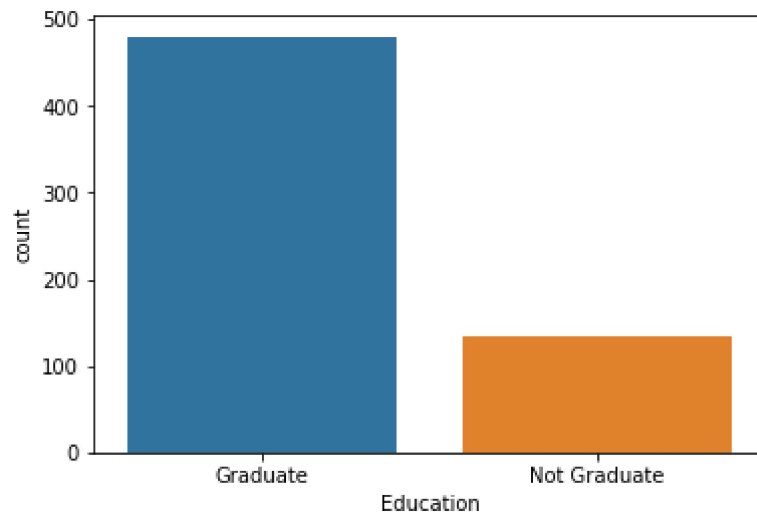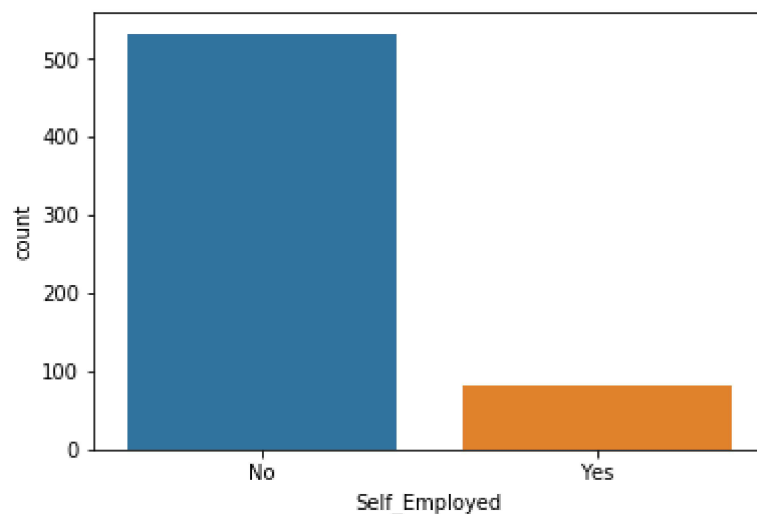

In [14]:
```python
sns.countplot(x='Dependents',data=df)
plt.show()
```
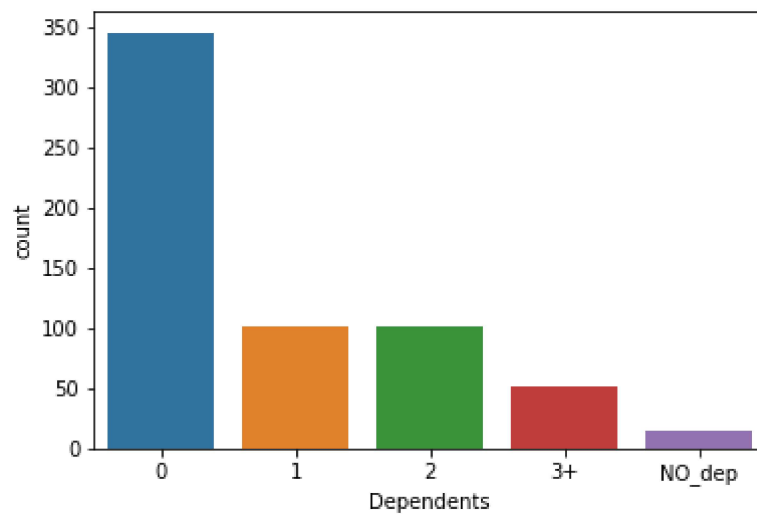
In [22]:
```python
sns.countplot(x='Education',data=df)
plt.show()
```


In [21]:
```python
sns.countplot(x='Self_Employed',data=df)
plt.show()
```


STEP 4:

In [16]:
```python
x=df.drop(['Loan_Status'],axis=1)
```

In [17]:
```python
y=df.pop('Loan_Status')
```

STEP 5:

```
In [19]: x=pd.get_dummies(x)
```

STEP 6:

```
In [20]: from sklearn.model_selection import train_test_split
         x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.30,random_state=42)
```

```
In [34]: from sklearn.preprocessing import StandardScaler
         ss=StandardScaler()
```

```
In [35]: x_train_ss=ss.fit_transform(x_train)
         x_train_ss
```

```
Out[35]: array([[-0.50133384,  0.27865737,  0.40368493, ..., -0.62317695,
                 -0.79056942,  1.40682858],
                [-0.42803179,  0.45103751,  0.09632945, ...,  1.60468065,
                 -0.79056942, -0.71081865],
                [-0.5669725 ,  0.23208844, -0.15191921, ..., -0.62317695,
                  1.26491106, -0.71081865],
                ...,
                [-0.37088951, -0.59751445, -1.38134113, ..., -0.62317695,
                 -0.79056942,  1.40682858],
                [ 0.76362634, -0.59751445, -0.00519051, ..., -0.62317695,
                  1.26491106, -0.71081865],
                [ 1.36387019, -0.59751445, -0.00519051, ..., -0.62317695,
                 -0.79056942,  1.40682858]])
```

```
In [36]: x_test_ss=ss.fit_transform(x_test)
         x_test_ss
```

```
Out[36]: array([[ 0.60310661, -0.4897835 ,  1.00133607, ..., -0.68429085,
                  1.31171195, -0.67579058],
                [-0.1508012 , -0.4897835 , -0.18660311, ..., -0.68429085,
                  1.31171195, -0.67579058],
                [-0.17338842, -0.07075971,  0.15280809, ...,  1.4613669 ,
                 -0.7623625 , -0.67579058],
                ...,
                [ 1.02547189, -0.4897835 ,  0.50434111, ..., -0.68429085,
                 -0.7623625 ,  1.47974835],
                [-0.34587267,  0.20984434, -0.07750665, ...,  1.4613669 ,
                 -0.7623625 , -0.67579058],
                [ 0.03716241, -0.4897835 , -0.48964881, ...,  1.4613669 ,
                 -0.7623625 , -0.67579058]])
```

```
In [37]: from sklearn.svm import LinearSVC
         lvc=LinearSVC()
         lvc.fit(x_train_ss,y_train)
         l_pred=lvc.predict(x_test_ss)
         l_pred
```

```
In [63]: from sklearn.metrics import accuracy_score as acs
         lvc_acc=acs(y_test,l_pred)
         lvc_acc
```

Out[63]: 0.745945945945946

```
In [50]: from sklearn.metrics import confusion_matrix as cm
         mat=cm(y_test,l_pred)
         mat
```

Out[50]: array([[20, 45],
                [2, 118]], dtype=int64)

```
In [52]: from sklearn.metrics import classification_report as cr
         cre=cr(y_test,l_pred)
         print(cre)
```

```
              precision    recall  f1-score   support

           N       0.91      0.31      0.46        65
           Y       0.72      0.98      0.83       120

avg / total        0.79      0.75      0.70       185
```

STEP 7:

```
In [62]: from sklearn.linear_model import LogisticRegression
         lor=LogisticRegression()
         lor.fit(x_train_ss,y_train)
         lr_pred=lor.predict(x_test_ss)

         from sklearn.svm import LinearSVC
         lvc=LinearSVC()
         lvc.fit(x_train_ss,y_train)
         l_pred=lvc.predict(x_test_ss)

         from sklearn.metrics import accuracy_score as acs
         lvc_acc=acs(y_test,l_pred)
         print("linear accuracy score:",lvc_acc)

         lvc_acc=acs(y_test,lr_pred)
         print("logistic regression accuracy score:",lvc_acc)
```

```
linear accuracy score: 0.745945945945946
logistic regression accuracy score: 0.7567567567567568
```