Data Munging, Manipulation, Exploratory analysis using Pandas

```
import pandas as pd
import numpy as np
#Coding for importing csv files in Google colab
from google.colab import files
import io
uploaded = files.upload()
df = pd.read_csv(io.BytesIO(uploaded['loan.csv']))
# Read csv loan.csv into a pandas dataframe
# Take a look at the first few rows
print(df)
```

```
Choose Files loan.csv
• loan.csv(text/csv) - 21589 bytes, last modified: 25/01/2020 - 100% done
Saving loan.csv to loan (2).csv
      Loan_ID Gender Married  ... Loan_Amount_Term Credit_History Property_Ar
0    LP001015   Male     Yes  ...            360.0            1.0         Urb
1    LP001022   Male     Yes  ...            360.0            1.0         Urb
2    LP001031   Male     Yes  ...            360.0            1.0         Urb
3    LP001035   Male     Yes  ...            360.0            NaN         Urb
4    LP001051   Male      No  ...            360.0            1.0         Urb
..        ...    ...     ...  ...              ...            ...           .
362  LP002971   Male     Yes  ...            360.0            1.0         Urb
363  LP002975   Male     Yes  ...            360.0            1.0         Urb
364  LP002980   Male      No  ...            360.0            NaN     Semiurb
365  LP002986   Male     Yes  ...            360.0            1.0         Rur
366  LP002989   Male      No  ...            180.0            1.0         Rur

[367 rows x 12 columns]
```

To view the first 10 rows in the dataset

```
df.head(10)
```

Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantInc
Yes	0	Graduate	No	5720	
Yes	1	Graduate	No	3076	1
Yes	2	Graduate	No	5000	1
Yes	2	Graduate	No	2340	2
No	0	Not Graduate	No	3276	
		Not			

To calculate the statistical calculations for all numerical fields

Graduate

```
df.describe()
```

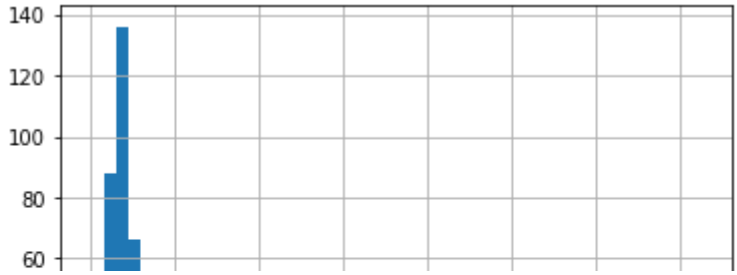
	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credi
count	367.000000	367.000000	362.000000	361.000000	
mean	4805.599455	1569.577657	136.132597	342.537396	
std	4910.685399	2334.232099	61.366652	65.156643	
min	0.000000	0.000000	28.000000	6.000000	
25%	2864.000000	0.000000	100.250000	360.000000	
50%	3786.000000	1025.000000	125.000000	360.000000	
75%	5060.000000	2430.500000	158.000000	360.000000	
max	72529.000000	24000.000000	550.000000	480.000000	

Distribution analysis using EDA

Analysis on Application income alone using histogram

```
df['ApplicantIncome'].hist(bins=50)
```

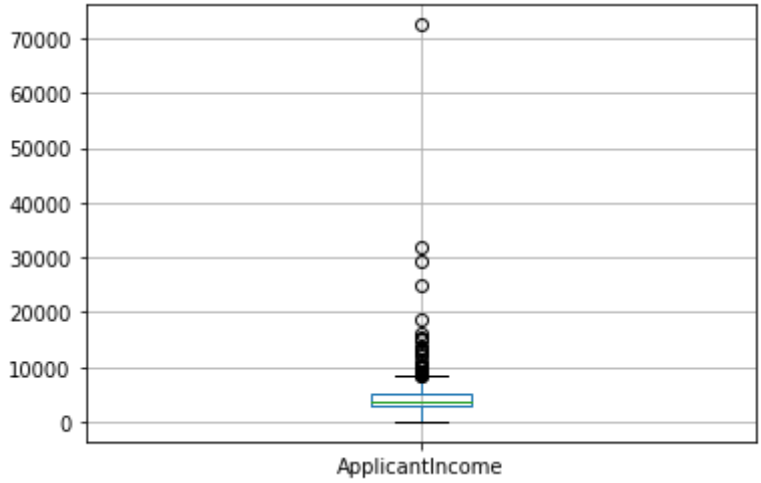
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1c6ccd2ba8>
```



Analysis on Application income alone using boxplot

```
df.boxplot(column='ApplicantIncome')
```

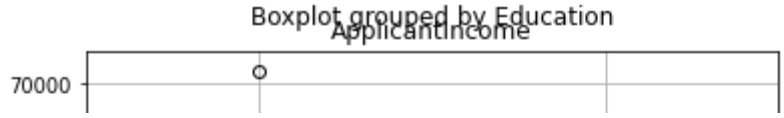
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1c6f2f9978>
```



Analysis on Application income and Education using boxplot

```
df.boxplot(column='ApplicantIncome', by = 'Education')
```

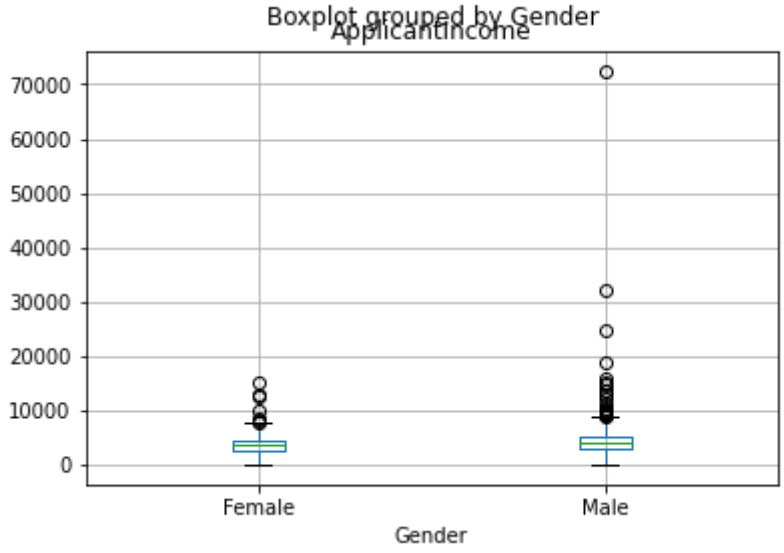
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1c6cbf3eb8>
```



Analysis on Application income and gender using boxplot

```
df.boxplot(column='ApplicantIncome', by = 'Gender')
```

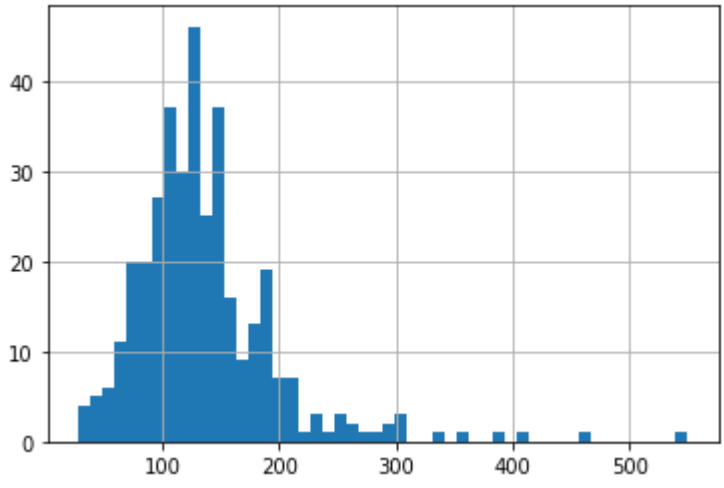
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1c6cd504e0>
```



Analysis on Loan Amount alone using histogram

```
df['LoanAmount'].hist(bins=50)
```

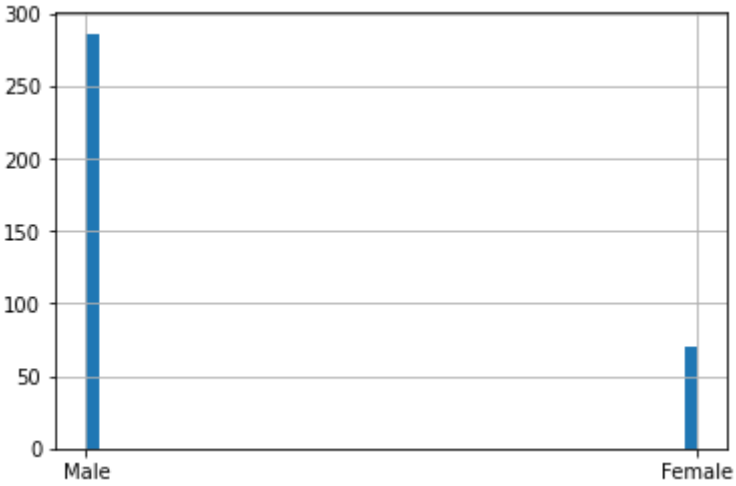
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1c6cb04208>
```



Analysis on Gender alone using histogram

```
df['Gender'].hist(bins=50)
```

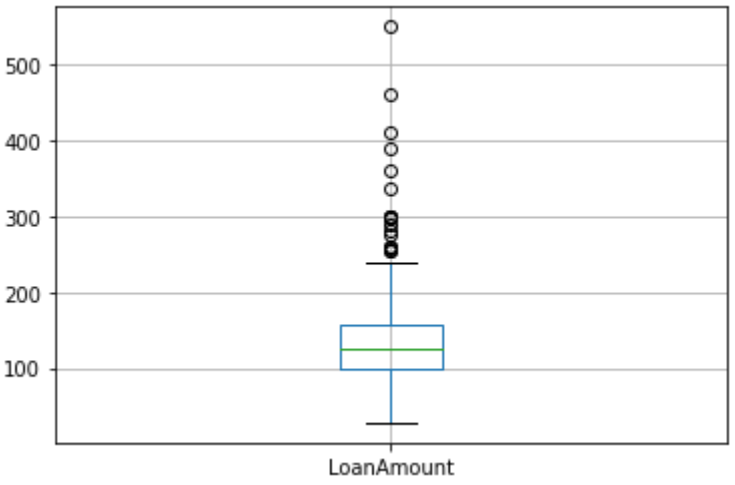
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1c6cad7240>
```



Analysis on Loan Amount alone using boxplot

```
df.boxplot(column='LoanAmount')
```

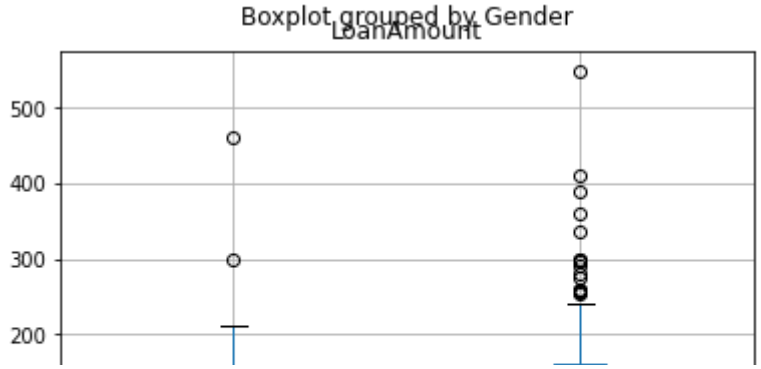
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1c6c8bd5c0>
```



Analysis on Loan Amount and gender using boxplot

```
df.boxplot(column='LoanAmount', by = 'Gender')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1c6ca97080>
```



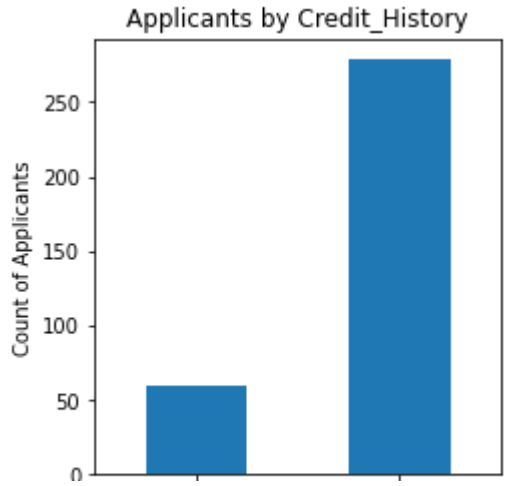
Categorical variable analysis

```
print ('Frequency Table for Credit History:')
temp1=df['Credit_History'].value_counts(ascending=True)
print(temp1)

print ('Frequency Table for Education:')
temp2=df['Education'].value_counts(ascending=True)
print(temp2)
```

```
Frequency Table for Credit History:
0.0     59
1.0    279
Name: Credit_History, dtype: int64
Frequency Table for Education:
Not Graduate     84
Graduate        283
Name: Education, dtype: int64
```

Applicants by Credit_History Analysis

```
import matplotlib.pyplot as plt
fig = plt.figure(figsize=(8,4))

#applicants by credit history
ax1 = fig.add_subplot(121)
ax1.set_xlabel('Credit_History')
ax1.set_ylabel('Count of Applicants')
ax1.set_title("Applicants by Credit_History")
temp1.plot(kind='bar')
```



Applicants by Credit_History Analysis and Applicants by Education Analysis both hand in hand

```
import matplotlib.pyplot as plt
fig = plt.figure(figsize=(8,4))

#applicants by credit history
ax1 = fig.add_subplot(121)
ax1.set_xlabel('Credit_History')
ax1.set_ylabel('Count of Applicants')
ax1.set_title("Applicants by Credit_History")
temp1.plot(kind='bar')
print('')

#applicants by education
ax2 = fig.add_subplot(122)
ax2.set_xlabel('Education')
ax1.set_ylabel('Count of Applicants')
ax1.set_title("Applicants by  Education")
temp2.plot(kind='bar')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1c6c89cb38>
```

Check missing values in the dataset

```
df.apply(lambda x: sum(x.isnull()),axis=0)
```

```
Loan_ID               0
Gender               11
Married               0
Dependents           10
Education             0
Self_Employed        23
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount            5
Loan_Amount_Term      6
Credit_History       29
Property_Area         0
dtype: int64
```

replacing missing loan amount with mean of the loanamount

```
df['LoanAmount'].fillna(df['LoanAmount'].mean(), inplace=True)
```

viewing the data set

```
df
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	Applicant
0	LP001015	Male	Yes	0	Graduate	No	
1	LP001022	Male	Yes	1	Graduate	No	
2	LP001031	Male	Yes	2	Graduate	No	
3	LP001035	Male	Yes	2	Graduate	No	
4	LP001051	Male	No	0	Not Graduate	No	

once again checking empty values

```
df.apply(lambda x: sum(x.isnull()),axis=0)
```

```
Loan_ID               0
Gender               11
Married               0
Dependents           10
Education             0
Self_Employed        23
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount            0
Loan_Amount_Term      6
Credit_History       29
Property_Area         0
dtype: int64
```

checking Self_Employed

```
df['Self_Employed'].value_counts()
```

```
No     307
Yes     37
Name: Self_Employed, dtype: int64
```

As No is dominating, replacing the empty values with No

```
df['Self_Employed'].fillna('No',inplace=True)
```

checking Self_Employed once again

```
df['Self_Employed'].value_counts()
```

```
No     330
Yes     37
Name: Self_Employed, dtype: int64
```

checking Dependents

```
df['Dependents'].value_counts()
```

```
0     200
2      59
1      58
3+     40
Name: Dependents, dtype: int64
```

As 0 is dominating , replace empty values with 0

```
df['Dependents'].fillna('0',inplace=True)
```

once again checking Dependents

```
df['Dependents'].value_counts()
```

```
0     200
2      59
1      58
3+     40
Name: Dependents, dtype: int64
```

once again checking empty values

```
df.apply(lambda x: sum(x.isnull()),axis=0)
```

```
Loan_ID               0
Gender               11
Married               0
Dependents           10
Education             0
Self_Employed         0
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount            5
```

checking Gender

```
df['Gender'].value_counts()
```

```
Male      286
Female     70
Name: Gender, dtype: int64
```

male is dominated with 80% so replace empty values with Male

```
df['Gender'].fillna('Male',inplace=True)
```

once again checking Gender

```
df['Gender'].value_counts()
```

```
Male      297
Female     70
Name: Gender, dtype: int64
```

once again checking empty values

```
df.apply(lambda x: sum(x.isnull()),axis=0)
```

```
Loan_ID               0
Gender                0
Married               0
Dependents           10
Education             0
Self_Employed         0
ApplicantIncome       0
CoapplicantIncome     0
```

checking Loan_Amount_Term

```
dtype: int64
```

```
df['Loan_Amount_Term'].value_counts()
```

```
360.0    311
180.0     22
480.0      8
300.0      7
240.0      4
84.0       3
6.0        1
120.0      1
36.0       1
350.0      1
12.0       1
60.0       1
Name: Loan_Amount_Term, dtype: int64
```

As Loan_Amount_Term=360 is dominating,replace empty values with 360

```
df['Loan_Amount_Term'].fillna(360.0,inplace=True)
```

checking Loan_Amount_Term

```
df['Loan_Amount_Term'].value_counts()
```

```
360.0    317
180.0     22
480.0      8
300.0      7
```

once again checking empty values

```
36.0       1
```

```
df.apply(lambda x: sum(x.isnull()),axis=0)
```

```
Loan_ID               0
Gender                0
Married               0
Dependents           10
Education             0
Self_Employed         0
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount            5
Loan_Amount_Term      0
Credit_History       29
Property_Area         0
dtype: int64
```

checking Credit_History

```
df['Credit_History'].value_counts()
```

```
1.0    279
0.0     59
Name: Credit_History, dtype: int64
```

yes (1.0) is dominating

```
df['Credit_History'].fillna(1.0,inplace=True)
```

once again checking empty values

```
df.apply(lambda x: sum(x.isnull()),axis=0)
```

```
Loan_ID              0
Gender               0
Married              0
Dependents           0
Education            0
Self_Employed        0
ApplicantIncome      0
CoapplicantIncome    0
LoanAmount           0
Loan_Amount_Term     0
```

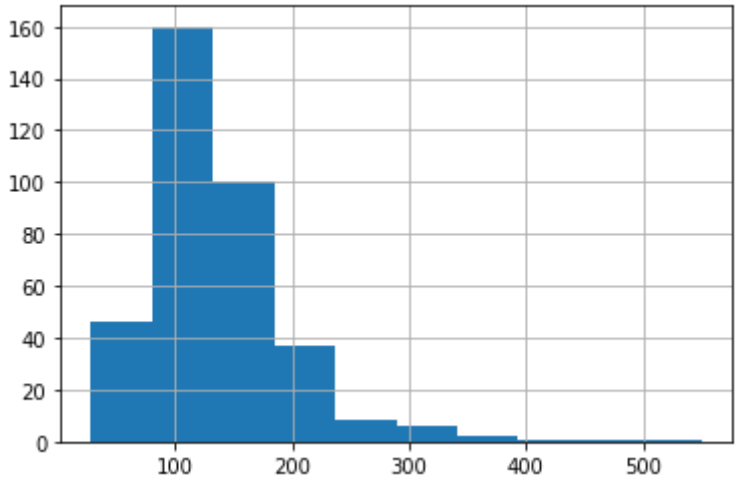
Finally all missing values are clear

Then go to the next phase of normalization

how to treat for extreme values in distribution of LoanAmount and ApplicantIncome

```
df['LoanAmount'].hist(bins=10)
```

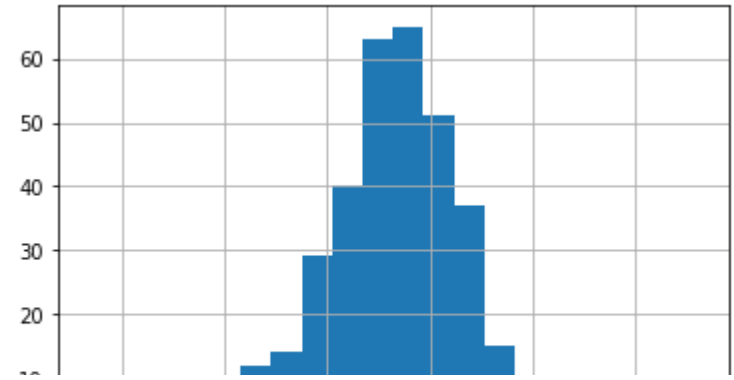
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1c6c725198>
```



creating LoanAmount_log column to treate outliers and extreme values

```
df['LoanAmount_log'] = np.log(df['LoanAmount'])
df['LoanAmount_log'].hist(bins=20)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1c6c76b4e0>
```



The normalized data set with artificial field LoanAmount_log

```
df
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	Applicant
0	LP001015	Male	Yes	0	Graduate	No	
1	LP001022	Male	Yes	1	Graduate	No	
2	LP001031	Male	Yes	2	Graduate	No	
3	LP001035	Male	Yes	2	Graduate	No	
4	LP001051	Male	No	0	Not Graduate	No	
...	...	...	...	...	...	...	
362	LP002971	Male	Yes	3+	Not Graduate	Yes	
363	LP002975	Male	Yes	0	Graduate	No	
364	LP002980	Male	No	0	Graduate	No	
365	LP002986	Male	Yes	0	Graduate	No	
366	LP002989	Male	No	0	Graduate	Yes	

367 rows × 13 columns