In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns


path='train_data.csv'
df = pd.read_csv(path)
df
```

Out[2]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	Coapp
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	
...	...	...	...	...	...	...	...	
609	LP002978	Female	No	0	Graduate	No	2900	
610	LP002979	Male	Yes	3+	Graduate	No	4106	
611	LP002983	Male	Yes	1	Graduate	No	8072	
612	LP002984	Male	Yes	2	Graduate	No	7583	
613	LP002990	Female	No	0	Graduate	Yes	4583	

614 rows × 13 columns

In [4]:
```
# it gives the requried statistical values for the further exploration
df.describe()
```

Out[4]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
count	614.000000	614.000000	592.000000	600.00000	564.000000
mean	5403.459283	1621.245798	146.412162	342.00000	0.842199
std	6109.041673	2926.248369	85.587325	65.12041	0.364878
min	150.000000	0.000000	9.000000	12.00000	0.000000
25%	2877.500000	0.000000	100.000000	360.00000	1.000000
50%	3812.500000	1188.500000	128.000000	360.00000	1.000000
75%	5795.000000	2297.250000	168.000000	360.00000	1.000000
max	81000.000000	41667.000000	700.000000	480.00000	1.000000

In [3]:
```
# Dropping a column Loan ID as it won't serve any purpose here and acts as hind
df.drop(columns='Loan_ID',axis=0,inplace=True)
```

In [169]:
```
df
```

Out[169]:

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncom
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
...	...	...	...	...	...	...	
609	Female	No	0	Graduate	No	2900	0
610	Male	Yes	3+	Graduate	No	4106	0
611	Male	Yes	1	Graduate	No	8072	240
612	Male	Yes	2	Graduate	No	7583	0
613	Female	No	0	Graduate	Yes	4583	0

614 rows × 12 columns

In [50]:
```
# This function gives top few rows in the data
df.head()
```

Out[50]:

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome
0	Male	No	0	Graduate	No	5849	0.0
1	Male	Yes	1	Graduate	No	4583	1508.0
2	Male	Yes	0	Graduate	Yes	3000	0.0
3	Male	Yes	0	Not Graduate	No	2583	2358.0
4	Male	No	0	Graduate	No	6000	0.0

In [52]:
```
# Last 5 rows we will use in tail
df.tail()
```

Out[52]:

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncom
609	Female	No	0	Graduate	No	2900	0
610	Male	Yes	3+	Graduate	No	4106	0
611	Male	Yes	1	Graduate	No	8072	240
612	Male	Yes	2	Graduate	No	7583	0
613	Female	No	0	Graduate	Yes	4583	0

In [53]:
```
# we can get number of rows an columns by using shape
df.shape
```

Out[53]:
```
(614, 12)
```

In [54]:
```
# the above rows and columns can be mentioned as a program
print(f'The number of rows in above prediction dataset is {df.shape[0]}')
print(f'The number of columns in above prediction dataset is {df.shape[1]}')
```

```
The number of rows in above prediction dataset is 614
The number of columns in above prediction dataset is 12
```

In [55]:
```
# we can find out the number of indes by the size of the data function
df.size
```

Out[55]:
```
7368
```

In [56]:
```
# which is nothing but rows multiplied by columns in other words are also calle
614*12
```

Out[56]:
```
7368
```

In [58]:
```
# to know the number of columns which are present in the dataset we can use the
df.columns
```

Out[58]:
```
Index(['Gender', 'Married', 'Dependents', 'Education', 'Self_Employed',
       'ApplicantIncome', 'CoapplicantIncome', 'LoanAmount',
       'Loan_Amount_Term', 'Credit_History', 'Property_Area', 'Loan_Status'],
      dtype='object')
```

In [194]:
```
# to know the rows
df.row
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_13684\510473177.py in ?()
      1 # to know the rows
----> 2 df.row

~\AppData\Local\anaconda3\Lib\site-packages\pandas\core\generic.py in ?(self, name)
   5985             and name not in self._accessors
   5986             and self._info_axis._can_hold_identifiers_and_holds_name(name)
   5987         ):
   5988             return self[name]
-> 5989         return object.__getattribute__(self, name)

AttributeError: 'DataFrame' object has no attribute 'row'
```

In [61]:
```
# to know what is the type of our current dataset  we can use type function ana

type(df)
```

Out[61]:
```
pandas.core.frame.DataFrame
```

In [62]:
```
# to know what are the specific data types in each column we can make use of th

df.dtypes
```

Out[62]:
```
Gender               object
Married              object
Dependents           object
Education            object
Self_Employed        object
ApplicantIncome       int64
CoapplicantIncome   float64
LoanAmount          float64
Loan_Amount_Term    float64
Credit_History      float64
Property_Area        object
Loan_Status          object
dtype: object
```

In [63]:
```
# our first task should be to extract the categorical and numerical columns for
dict(df.dtypes)
```

Out[63]:
```
{'Gender': dtype('O'),
 'Married': dtype('O'),
 'Dependents': dtype('O'),
 'Education': dtype('O'),
 'Self_Employed': dtype('O'),
 'ApplicantIncome': dtype('int64'),
 'CoapplicantIncome': dtype('float64'),
 'LoanAmount': dtype('float64'),
 'Loan_Amount_Term': dtype('float64'),
 'Credit_History': dtype('float64'),
 'Property_Area': dtype('O'),
 'Loan_Status': dtype('O')}
```

In [66]:
```
d=dict(df.dtypes)
for i in d:
    if d[i]=='object':
        print(i)
```

```
Gender
Married
Dependents
Education
Self_Employed
Property_Area
Loan_Status
```

In [68]:
```
d=dict(df.dtypes)
for i in d:
    if d[i]!='object':
        print(i)
```

```
ApplicantIncome
CoapplicantIncome
LoanAmount
Loan_Amount_Term
Credit_History
```

In [70]:
```
cat=[i for i in d if d[i]=='object']
cat
```

Out[70]:
```
['Gender',
 'Married',
 'Dependents',
 'Education',
 'Self_Employed',
 'Property_Area',
 'Loan_Status']
```

In [72]:
```
num=[i for i in d if d[i] != 'object']
num
```

Out[72]:
```
['ApplicantIncome',
 'CoapplicantIncome',
 'LoanAmount',
 'Loan_Amount_Term',
 'Credit_History']
```

In [74]:
```
# the above process where we seperted the rows and columns can also be done by
df.select_dtypes(include='object').columns
```

Out[74]:
```
Index(['Gender', 'Married', 'Dependents', 'Education', 'Self_Employed',
       'Property_Area', 'Loan_Status'],
      dtype='object')
```

In [75]:
```
# to show just numerical data we are supposed to exclude object
df.select_dtypes(exclude='object').columns
```

Out[75]:
```
Index(['ApplicantIncome', 'CoapplicantIncome', 'LoanAmount',
       'Loan_Amount_Term', 'Credit_History'],
      dtype='object')
```

In [77]:
```
df.isnull().head(5)
```

Out[77]:

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome
0	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False

In [78]:
```
df.isnull().sum()
```

Out[78]:
```
Gender               13
Married               3
Dependents           15
Education             0
Self_Employed        32
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount           22
Loan_Amount_Term     14
Credit_History       50
Property_Area         0
Loan_Status           0
dtype: int64
```

In [80]:
```
df.drop_duplicates()
```

Out[80]:

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncom
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
...	...	...	...	...	...	...	
609	Female	No	0	Graduate	No	2900	0
610	Male	Yes	3+	Graduate	No	4106	0
611	Male	Yes	1	Graduate	No	8072	240
612	Male	Yes	2	Graduate	No	7583	0
613	Female	No	0	Graduate	Yes	4583	0

614 rows × 12 columns

In [84]:
```
# in order to consider specific rows and colmns we are supposed to show on the
df.take([0,1,2]).take([8,9],axis=1)
```

Out[84]:

	Loan_Amount_Term	Credit_History
0	360.0	1.0
1	360.0	1.0
2	360.0	1.0

In [85]:
```
# in this we need to consider the rows and columns at this hence we can make us

df.iloc[5:10]
```

Out[85]:

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome
5	Male	Yes	2	Graduate	Yes	5417	4196.0
6	Male	Yes	0	Not Graduate	No	2333	1516.0
7	Male	Yes	3+	Graduate	No	3036	2504.0
8	Male	Yes	2	Graduate	No	4006	1526.0
9	Male	Yes	1	Graduate	No	12841	10968.0

In [90]:
```
df.iloc[2:10,1:9]
```

Out[90]:

	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAm
2	Yes	0	Graduate	Yes	3000	0.0	
3	Yes	0	Not Graduate	No	2583	2358.0	
4	No	0	Graduate	No	6000	0.0	
5	Yes	2	Graduate	Yes	5417	4196.0	
6	Yes	0	Not Graduate	No	2333	1516.0	
7	Yes	3+	Graduate	No	3036	2504.0	
8	Yes	2	Graduate	No	4006	1526.0	
9	Yes	1	Graduate	No	12841	10968.0	

In [170]:
```
# this gives the unique variables in the data set
df['Education'].unique()
```

Out[170]:
```
array(['Graduate', 'Not Graduate'], dtype=object)
```

In [171]:
```
# nunique gives the number f unique data sets available in the column

df['Education'].nunique()
```

Out[171]:
```
2
```

In [172]:
```
df['Property_Area'].unique()
```

Out[172]:
```
array(['Urban', 'Rural', 'Semiurban'], dtype=object)
```

In [173]:
```
df['Property_Area'].nunique()
```

Out[173]:
```
3
```

In [174]:
```
df['Property_Area']=='Urban'
```

Out[174]:
```
0       True
1      False
2       True
3       True
4       True
       ...
609    False
610    False
611     True
612     True
613    False
Name: Property_Area, Length: 614, dtype: bool
```

In [175]:
```
# Frequency table to know how many types of locations are there

len(df['Property_Area'])
count=[]
unique_labels=df['Property_Area'].unique()
for i in unique_labels:
    con=df['Property_Area']==i
    count.append(len(df[con]))
    print(count)
```

```
[202]
[202, 179]
[202, 179, 233]
```

In [176]:
```
Property_Area_Count=pd.DataFrame(zip(unique_labels,count),columns=['Prpty_Area_
Property_Area_Count
```

Out[176]:

	Prpty_Area_Cnt	Count
0	Urban	202
1	Rural	179
2	Semiurban	233

In [177]:
```
# y making use of value cunt function we can easily find the value here
Property_vc=df['Property_Area'].value_counts()
Property_vc
```

Out[177]:
```
Property_Area
Semiurban    233
Urban        202
Rural        179
Name: count, dtype: int64
```

In [178]:
```
# These are the key values of the value count
Property_vc.keys()
```

Out[178]:
```
Index(['Semiurban', 'Urban', 'Rural'], dtype='object', name='Property_Area')
```

In [179]:
```
# These are the values of the values
Property_vc.values
```

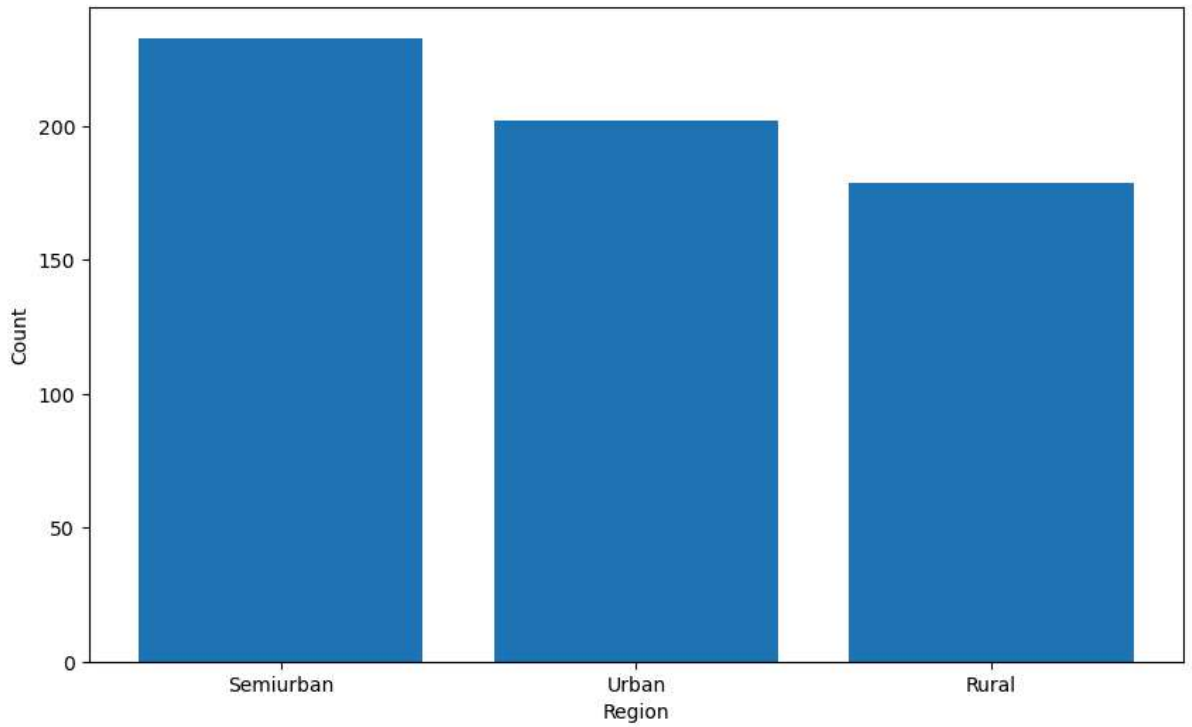
Out[179]:
```
array([233, 202, 179], dtype=int64)
```

In [197]:
```
Property_mc=df['Property_Area'].value_counts()
Property_mc
l1=Property_mc.keys()
l2=Property_mc.values
Property_mc_df=pd.DataFrame(zip(l1,l2),columns=['Property','Counts'])
Property_mc_df
```

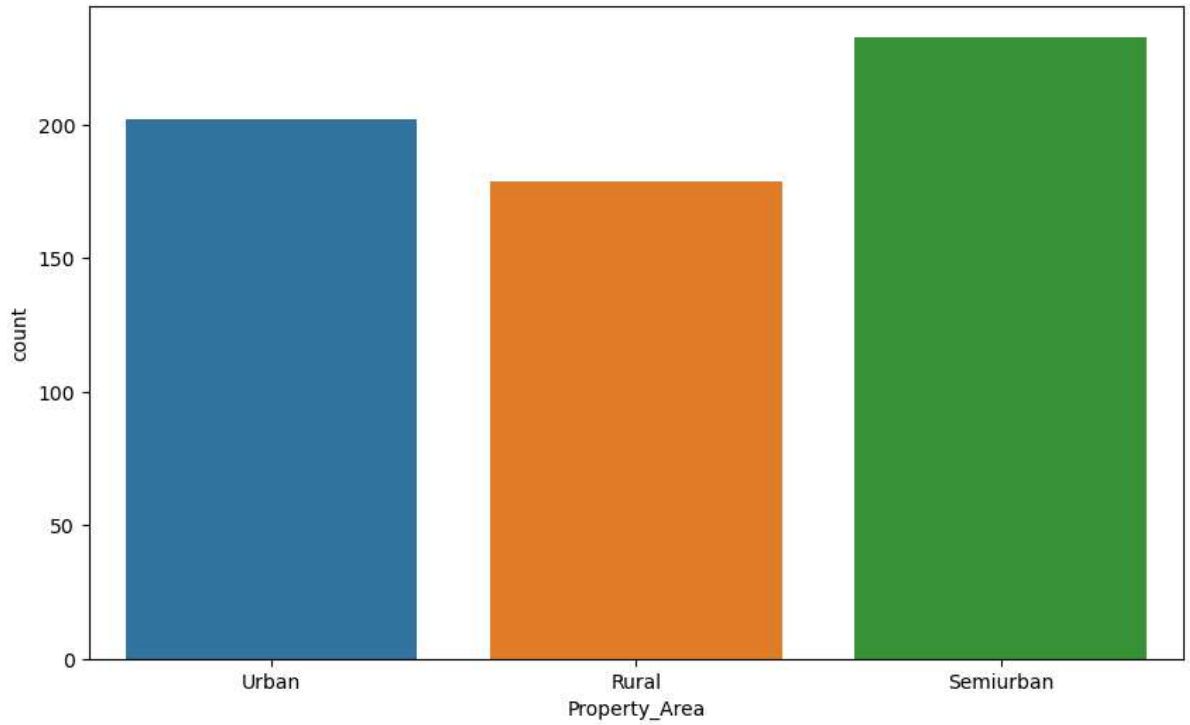
Out[197]:

	Property	Counts
0	Semiurban	233
1	Urban	202
2	Rural	179

In [202]:
```
# Bar graph of the chart
plt.figure(figsize=(10,6))
plt.bar('Property','Counts',data=Property_mc_df)
Property_mc_df
plt.xlabel('Region')
plt.ylabel('Count')
plt.show()
```



In [203]:
```
#  count plot
plt.figure(figsize=(10,6))
sns.countplot(data=df,x='Property_Area')
plt.show()
```



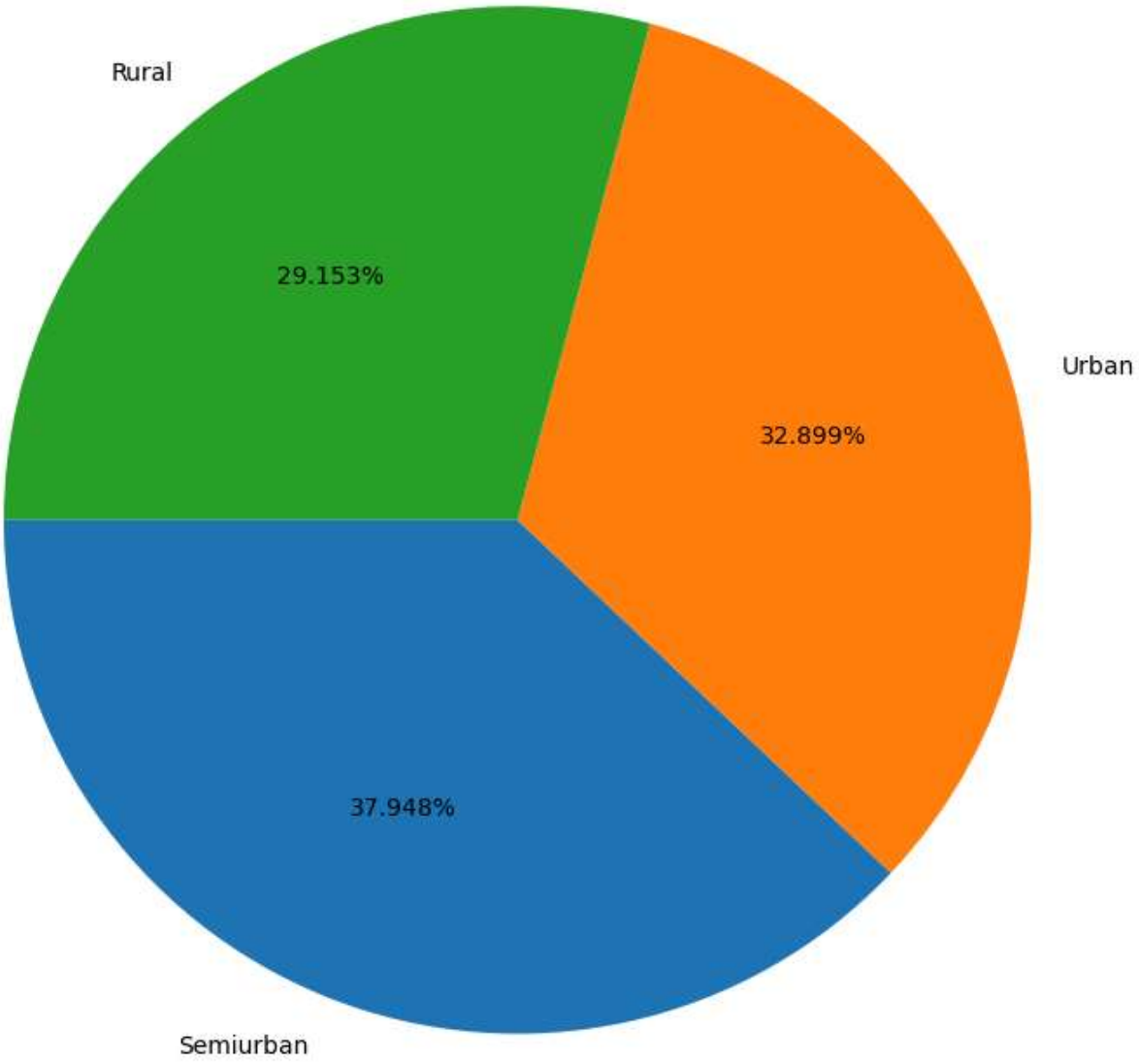
In [204]:
```
# Pie-Chart
df['Property_Area'].value_counts(normalize=True)
```

Out[204]:
```
Property_Area
Semiurban    0.379479
Urban        0.328990
Rural        0.291531
Name: proportion, dtype: float64
```

In [207]:
```
keys=df['Property_Area'].value_counts().keys()
values=df['Property_Area'].value_counts().values
values
```

Out[207]:
```
array([233, 202, 179], dtype=int64)
```

In [213]:
```
plt.pie(values,labels=keys,autopct='%0.3f%%',startangle=180,radius=2)
plt.show()
```



In [216]:
```
df['LoanAmount'].isnull().sum()
```

Out[216]: 22

In [217]:
```
df['ApplicantIncome'].isnull().sum()
```

Out[217]: 0

Count

In [218]:
```
len(df['ApplicantIncome'])
```

Out[218]: 614

In [219]:
```
df['ApplicantIncome'].count()
```

Out[219]: 614

In [221]:
```
df['ApplicantIncome'].mean()
```

Out[221]: 5403.459283387622

In [222]:
```
np.mean(df['ApplicantIncome'])
```

Out[222]: 5403.459283387622

In [223]:
```
df['ApplicantIncome'].median()
```

Out[223]: 3812.5

In [224]:
```
df['ApplicantIncome'].mode()
```

Out[224]:
```
0    2500
Name: ApplicantIncome, dtype: int64
```

In [226]:
```
df['ApplicantIncome'].max()
```

Out[226]: 81000

In [227]:
```
df['ApplicantIncome'].min()
```

Out[227]: 150

In [228]:
```
df['ApplicantIncome'].std()
```

Out[228]: 6109.041673387174

In [236]:
```
Q1=np.percentile(df['ApplicantIncome'],25)
Q1
```

Out[236]: 2877.5

In [237]:
```
Q2=np.percentile(df['ApplicantIncome'],50)
Q2
```

Out[237]: 3812.5

In [238]:
```
Q3=np.percentile(df['ApplicantIncome'],75)
Q3
```

Out[238]: 5795.0

In [241]:
```
IQR=Q3-Q1
IQR
```

Out[241]:
```
2917.5
```

In [242]:
```
LB=Q1-1.5*IQR
UB=Q2+1.5*IQR
```

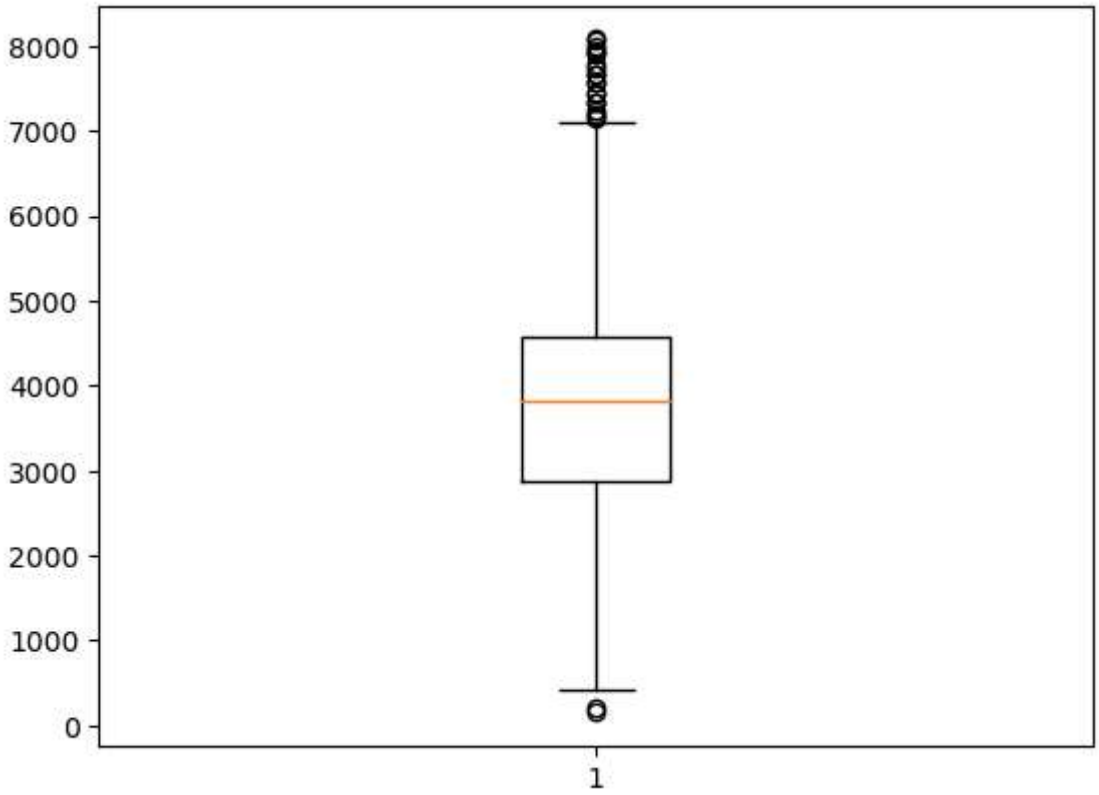
In [250]:
```
C1=df['ApplicantIncome']<LB
C2=df['ApplicantIncome']>UB
con=C1|C2
con

outliers_df=df[con]
outliers_df

non_outliers_df = df[C1&C2]
non_outliers_df

emp=[]
median=df['ApplicantIncome'].median()
for i in df['ApplicantIncome']:
    if i<LB or i>UB:
        emp.append(median)
    else:
        emp.append(i)
df['ApplicantIncome_new']=emp
df
```

Out[250]:

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncom
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
...	...	...	...	...	...	...	
609	Female	No	0	Graduate	No	2900	0
610	Male	Yes	3+	Graduate	No	4106	0
611	Male	Yes	1	Graduate	No	8072	240
612	Male	Yes	2	Graduate	No	7583	0
613	Female	No	0	Graduate	Yes	4583	0

614 rows × 13 columns

In [251]:
```
# box lot:
plt.boxplot(df['ApplicantIncome_new'])
plt.show()
```



In [253]:
```
# Bi-Variate analysis
labels = df['Loan_Status'].unique()
yes_loan_approved=[]
no_loan_approved=[]
for i in labels:
    b1=df['ApplicantIncome_new']==i
    b2=df['Loan_Status']=='Y'
    b3=df['Loan_Status']=='N'
    app_con=b1&b2
    den_con=b1&b3
    yes_loan_approved.append(len(df[app_con]))
    no_loan_approved.append(len(df[den_con]))
yes_loan_approved,no_loan_approved
```

Out[253]: ([0, 0], [0, 0])

In [257]:
```
col1=df['Education']
col2=df['Loan_Status']
res1=pd.crosstab(col1,col2)
res1.head(50)
```

Out[257]:

Loan_Status	N	Y
Education		
Graduate	140	340
Not Graduate	52	82

In [258]:
```
df['Loan_Status'].unique()
```

Out[258]:
```
array(['Y', 'N'], dtype=object)
```

In [268]:
```
# d={'Y':1,'N':0}
# df['Loan_Status']=df['Loan_Status'].map(d)
# df
df['Married'].unique()

# mar={}
```

Out[268]:
```
array(['No', 'Yes', nan], dtype=object)
```

In [271]:
```
path='train_data.csv'
df = pd.read_csv(path)
df
```

Out[271]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	Coapp
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	
...	...	...	...	...	...	...	...	...
609	LP002978	Female	No	0	Graduate	No	2900	
610	LP002979	Male	Yes	3+	Graduate	No	4106	
611	LP002983	Male	Yes	1	Graduate	No	8072	
612	LP002984	Male	Yes	2	Graduate	No	7583	
613	LP002990	Female	No	0	Graduate	Yes	4583	

614 rows × 13 columns

In [272]:
```
# Label encoder
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
df['Loan_Status']=le.fit_transform(df['Loan_Status'])
df
```

Out[272]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	Coapp
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	
...	...	...	...	...	...	...	...	
609	LP002978	Female	No	0	Graduate	No	2900	
610	LP002979	Male	Yes	3+	Graduate	No	4106	
611	LP002983	Male	Yes	1	Graduate	No	8072	
612	LP002984	Male	Yes	2	Graduate	No	7583	
613	LP002990	Female	No	0	Graduate	Yes	4583	

614 rows × 13 columns

In [273]:
```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
df['Married']=le.fit_transform(df['Married'])
df
```

Out[273]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	Coapp
0	LP001002	Male	0	0	Graduate	No	5849	
1	LP001003	Male	1	1	Graduate	No	4583	
2	LP001005	Male	1	0	Graduate	Yes	3000	
3	LP001006	Male	1	0	Not Graduate	No	2583	
4	LP001008	Male	0	0	Graduate	No	6000	
...	...	...	...	...	...	...	...	
609	LP002978	Female	0	0	Graduate	No	2900	
610	LP002979	Male	1	3+	Graduate	No	4106	
611	LP002983	Male	1	1	Graduate	No	8072	
612	LP002984	Male	1	2	Graduate	No	7583	
613	LP002990	Female	0	0	Graduate	Yes	4583	

614 rows × 13 columns

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: