Q.Create a regression model to predict th budget Ans:

Regression: Performing necessary EDA and pre-processing on given data and finding out which Regression model is best Budget. steps are as follow:

In [1]:
```python
#import necesasry libraries:
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns

#read dataset:
df = pd.read_csv('Freelance Platform Projects.csv')

#display first five records of dataset:
df.head()
```

Out[1]:

	Title	Category Name	Experience	Sub Category Name	Currency	Budget	Location	Freelancer Preferred From	Type	Date Posted	Description	Duration	Client Registration Date	Client City	Client Country	Client Currency	Client Job Title
0	Banner images for web desgin websites	Design	Entry ($)	Graphic Design	EUR	60.0	remote	ALL	fixed_price	2023-04-29 18:06:39	We are looking to improve the banner images on...	NaN	2010-11-03	Dublin	Ireland	EUR	PPC Management
1	Make my picture a solid silhouette	Video, Photo & Image	Entry ($)	Image Editing	GBP	20.0	remote	ALL	fixed_price	2023-04-29 17:40:28	Hello \n\nI need a quick designer to make 4 pi...	NaN	2017-02-21	London	United Kingdom	GBP	Office manager
2	Bookkeeper needed	Business	Entry ($)	Finance & Accounting	GBP	12.0	remote	ALL	fixed_price	2023-04-29 17:40:06	Hi - I need a bookkeeper to assist with bookke...	NaN	2023-04-09	London	United Kingdom	GBP	Paralegal
3	Accountant needed	Business	Entry ($)	Tax Consulting & Advising	GBP	14.0	remote	ALL	fixed_price	2023-04-29 17:32:01	Hi - I need an accountant to assist me with un...	NaN	2023-04-09	London	United Kingdom	GBP	Paralegal
4	Guest Post on High DA Website	Digital Marketing	Expert ($$$)	SEO	USD	10000.0	remote	ALL	fixed_price	2023-04-29 17:09:36	Hi, I am currently running a project where I w...	NaN	2016-07-01	Mumbai	India	USD	Guest posts buyer

In [2]:
```python
#display rows and columns of adtaset respectively(rows,columns):
df.shape
```

Out[2]:
```
(12222, 17)
```

In [3]:
```python
#display information of data set:
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12222 entries, 0 to 12221
Data columns (total 17 columns):
 #   Column                     Non-Null Count  Dtype  
---  ------                     --------------  -----  
 0   Title                      12222 non-null  object 
 1   Category Name              12222 non-null  object 
 2   Experience                 12222 non-null  object 
 3   Sub Category Name          12222 non-null  object 
 4   Currency                   12222 non-null  object 
 5   Budget                     12222 non-null  float64
 6   Location                   12222 non-null  object 
 7   Freelancer Preferred From  12222 non-null  object 
 8   Type                       12222 non-null  object 
 9   Date Posted                12222 non-null  object 
 10  Description                12222 non-null  object 
 11  Duration                   1602 non-null   object 
 12  Client Registration Date   12222 non-null  object 
 13  Client City                12222 non-null  object 
 14  Client Country             12222 non-null  object 
 15  Client Currency            12222 non-null  object 
 16  Client Job Title           4588 non-null   object 
dtypes: float64(1), object(16)
memory usage: 1.6+ MB
```

EDA and Preprocessing:

In [4]:
```python
#identifying null values:
df.isna().sum()
```

Out[4]:
```
Title                            0
Category Name                    0
Experience                       0
Sub Category Name                0
Currency                         0
Budget                           0
Location                         0
Freelancer Preferred From        0
Type                             0
Date Posted                      0
Description                      0
Duration                     10620
Client Registration Date         0
Client City                      0
Client Country                   0
Client Currency                  0
Client Job Title              7634
dtype: int64
```

In [5]:
```python
# checking how much % of missing values present in dataset

df.isnull().sum()/ len(df)*100
```

Out[5]:
```
Title                         0.000000
Category Name                 0.000000
Experience                    0.000000
Sub Category Name             0.000000
Currency                      0.000000
Budget                        0.000000
Location                      0.000000
Freelancer Preferred From     0.000000
Type                          0.000000
Date Posted                   0.000000
Description                   0.000000
Duration                     86.892489
Client Registration Date      0.000000
Client City                   0.000000
Client Country                0.000000
Client Currency               0.000000
Client Job Title             62.461136
dtype: float64
```

In [6]:
```python
#Duration column has 86% and Client Job Title has 62% null values

df.dropna(axis=1, inplace=True)
```

In [7]:
```python
#checking the shape after dropping 2 columns

df.shape
```

Out[7]:
```
(12222, 15)
```

In [8]:
```python
#display datatypes of columns:
df.dtypes
```

Out[8]:
```
Title                         object
Category Name                 object
Experience                    object
Sub Category Name             object
Currency                      object
Budget                       float64
Location                      object
Freelancer Preferred From     object
Type                          object
Date Posted                   object
Description                   object
Client Registration Date      object
Client City                   object
Client Country                object
Client Currency               object
dtype: object
```

In [9]:
```python
#changing datatype of columns using labelencoder:
encode = ['Title','Category Name','Sub Category Name','Freelancer Preferred From','Description',
                    'Client City','Client Country','Type','Date Posted','Location','Experience','Client Registration Date','Client Currency']

from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
for column in encode:
    df[column] = le.fit_transform(df[column])
```

In [10]:
```python
#display datatypes of columns:
df.dtypes
```

Out[10]:
```
Title                          int32
Category Name                  int32
Experience                     int32
Sub Category Name              int32
Currency                      object
Budget                       float64
Location                       int32
Freelancer Preferred From      int32
Type                           int32
Date Posted                    int32
Description                    int32
Client Registration Date       int32
Client City                    int32
Client Country                 int32
Client Currency                int32
dtype: object
```

In [11]:
```python
#display first five recorsd:
df.head()
```

Out[11]:

	Title	Category Name	Experience	Sub Category Name	Currency	Budget	Location	Freelancer Preferred From	Type	Date Posted	Description	Client Registration Date	Client City	Client Country	Client Currency
0	969	1	0	42	EUR	60.0	1	1	0	9407	10434	163	489	61	0
1	6377	7	0	45	GBP	20.0	1	1	0	9406	1247	1441	940	129	1
2	1108	0	0	37	GBP	12.0	1	1	0	9405	2179	3144	940	129	1
3	467	0	0	90	GBP	14.0	1	1	0	9404	2181	3144	940	129	1
4	3859	2	1	76	USD	10000.0	1	1	0	9403	3024	1278	1079	58	2

In [12]:
```python
#converting all the currency in usd currency

conversion_rates={'EUR':1.07, 'GBP':1.24,'USD':1}
df['Budget_usd']=df['Currency'].map(conversion_rates)* df['Budget']
```

In [13]:
```python
#display details of dataset
df.describe()
```

Out[13]:

	Title	Category Name	Experience	Sub Category Name	Budget	Location	Freelancer Preferred From	Type	Date Posted	Description	Client Registration Date	Client City	Client Country	Client Currency	Budget_usd
count	12222.000000	12222.000000	12222.000000	12222.000000	12222.000000	12222.000000	12222.000000	12222.000000	12222.000000	12222.000000	12222.000000	12222.000000	12222.000000	12222.000000	12222.000000
mean	5806.995173	3.712813	0.687367	56.839552	229.221486	1.053428	2.186876	0.146048	4765.114629	5958.065538	2054.737932	871.258714	113.476681	1.179021	265.810769
std	3340.786412	2.820344	0.678159	32.627551	1894.327521	0.248736	5.015981	0.353169	2737.791980	3446.774106	986.662067	456.183611	32.275724	0.572128	2280.510207
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	2918.250000	1.000000	0.000000	30.000000	30.000000	1.000000	1.000000	0.000000	2345.250000	2965.250000	1225.250000	487.250000	128.000000	1.000000	37.200000
50%	5842.500000	3.000000	1.000000	52.000000	80.000000	1.000000	1.000000	0.000000	4802.500000	5960.500000	2245.000000	940.000000	129.000000	1.000000	93.000000
75%	8707.750000	6.000000	1.000000	92.000000	150.000000	1.000000	1.000000	0.000000	7170.750000	8938.750000	3073.000000	1139.000000	129.000000	2.000000	186.000000
max	11584.000000	8.000000	2.000000	106.000000	99999.000000	2.000000	41.000000	1.000000	9407.000000	11924.000000	3164.000000	1807.000000	135.000000	2.000000	123998.760000

In [14]:
```python
#dropping column currency and Budget as we have new column Budget_usd
df.drop(columns=['Currency','Budget'], inplace=True)
```

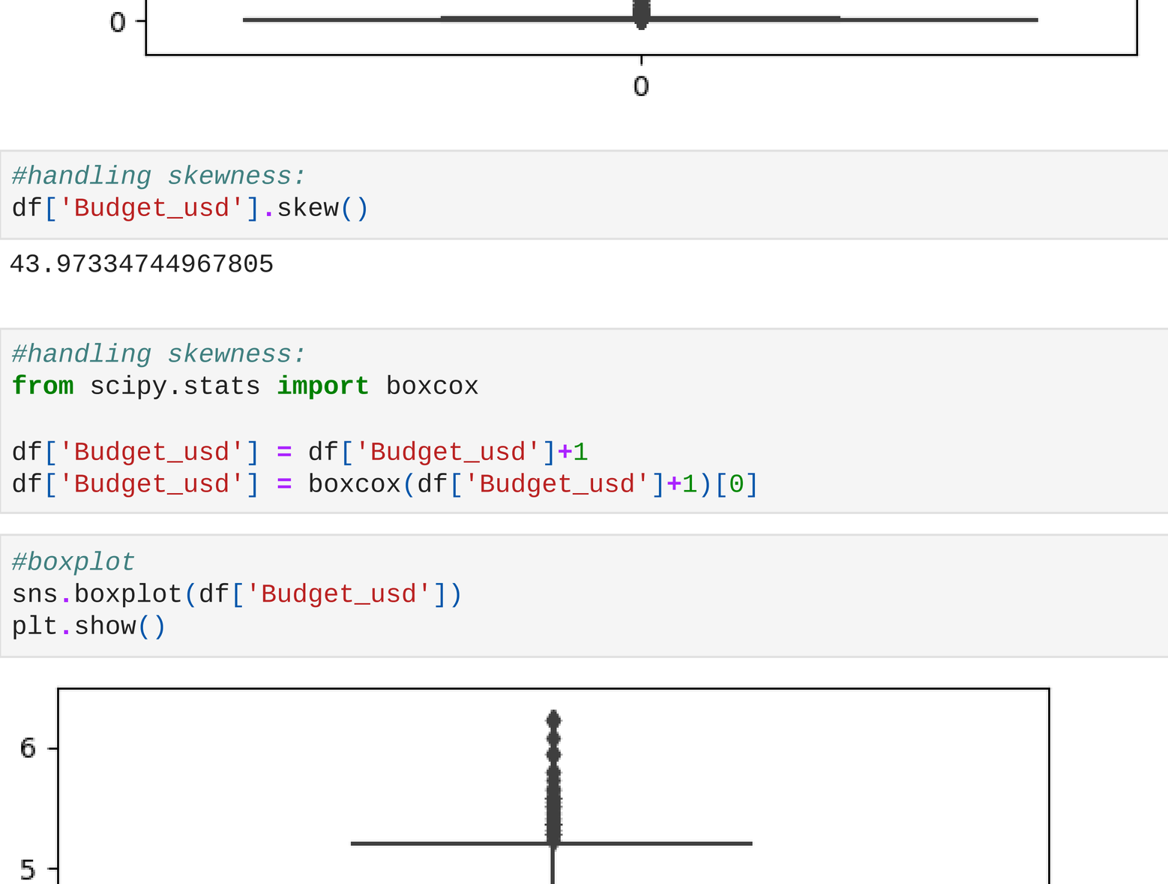
In [15]:
```python
#display first five recorsd:
df.head()
```

Out[15]:

	Title	Category Name	Experience	Sub Category Name	Location	Freelancer Preferred From	Type	Date Posted	Description	Client Registration Date	Client City	Client Country	Client Currency	Budget_usd
0	969	1	0	42	1	1	0	9407	10434	163	489	61	0	64.20
1	6377	7	0	45	1	1	0	9406	1247	1441	940	129	1	24.80
2	1108	0	0	37	1	1	0	9405	2179	3144	940	129	1	14.88
3	467	0	0	90	1	1	0	9404	2181	3144	940	129	1	17.36
4	3859	2	1	76	1	1	0	9403	3024	1278	1079	58	2	10000.00

In [16]:
```python
#identifying outliers and Handling:
import seaborn as sns
import matplotlib.pyplot as plt
sns.boxplot(df['Budget_usd'])
```

Out[16]:
```
<Axes: >
```



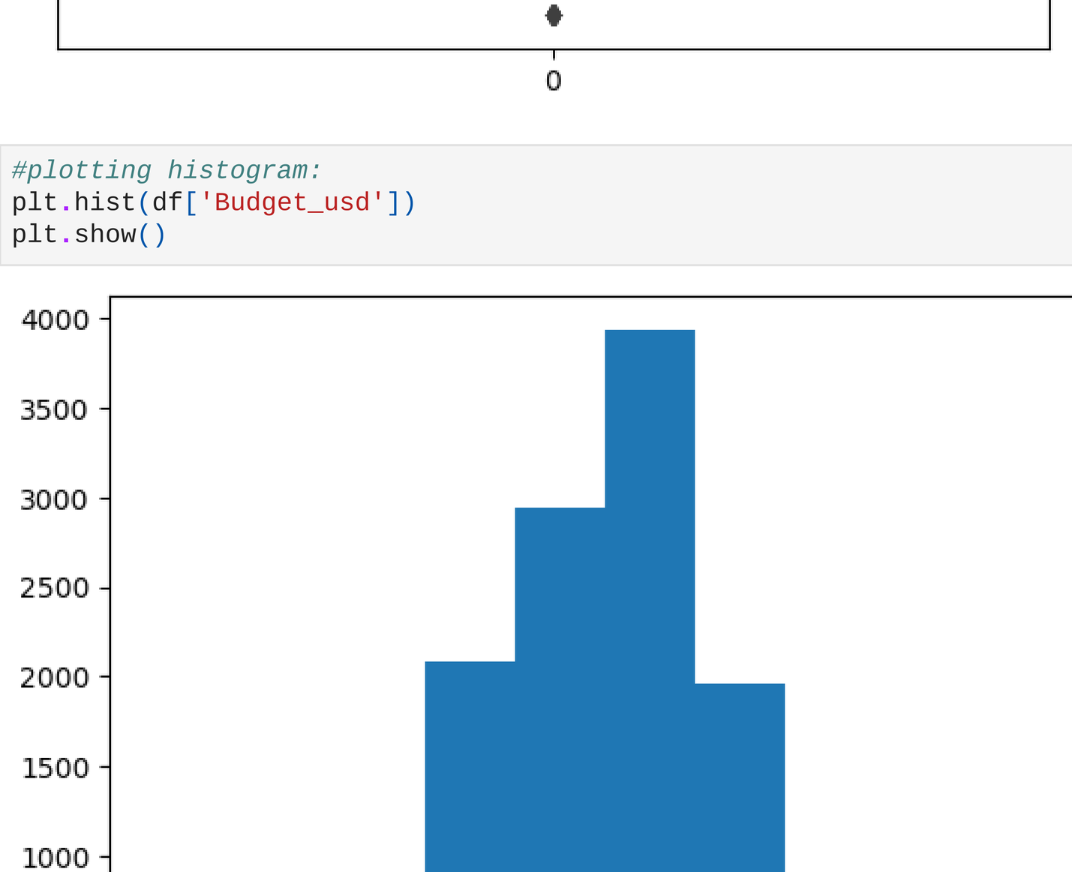
In [17]:
```python
#handling skewness:
df['Budget_usd'].skew()
```

Out[17]:
```
43.97334744967805
```

In [18]:
```python
#handling skewness:
from scipy.stats import boxcox

df['Budget_usd'] = df['Budget_usd']+1
df['Budget_usd'] = boxcox(df['Budget_usd']+1)[0]
```

In [19]:
```python
#boxplot
sns.boxplot(df['Budget_usd'])
plt.show()
```



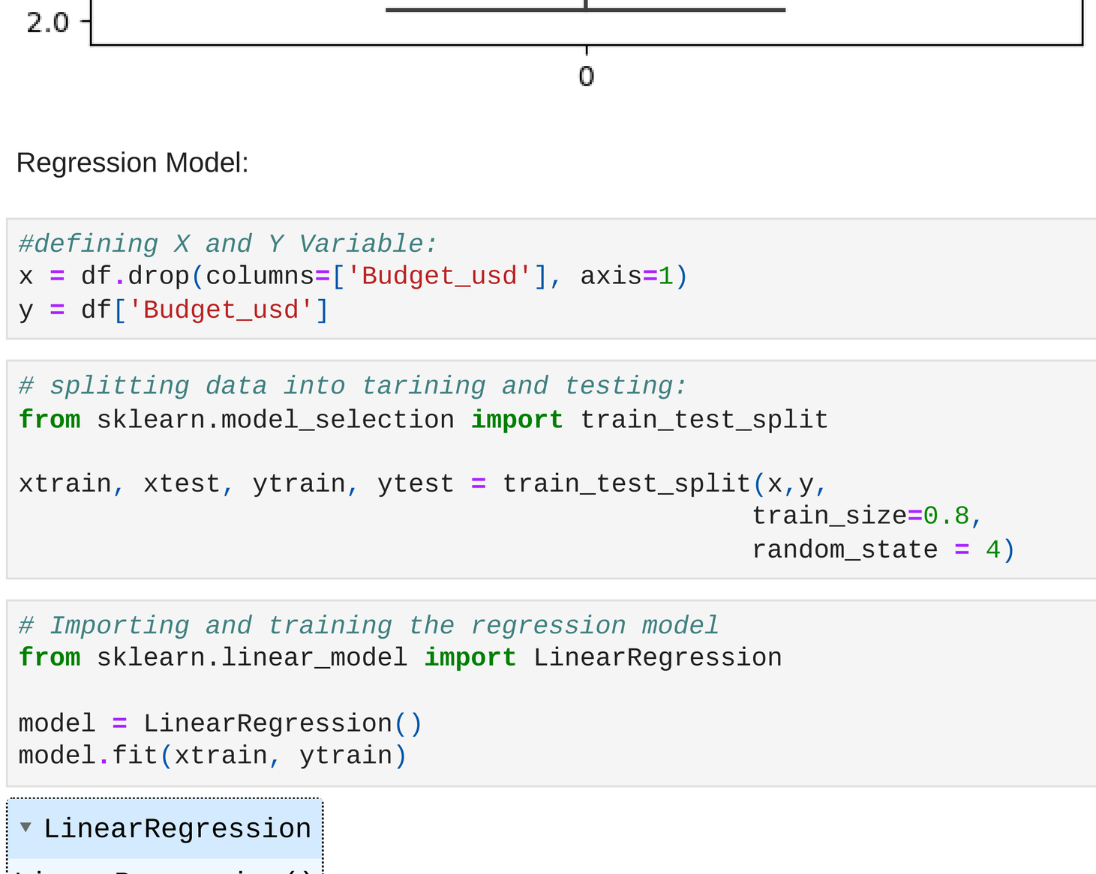
In [20]:
```python
#plotting histogram:
plt.hist(df['Budget_usd'])
plt.show()
```

In [30]:
```python
#handling skewness:
df['Budget_usd'].skew()
```

Out[30]:
```
-0.010823773098500838
```

In [21]:
```python
#Dropping the outliers from Budget_usd
df=df.drop(df[df['Budget_usd'] >5.2].index)
df=df.drop(df[df['Budget_usd']<2].index)
```

In [22]:
```python
sns.boxplot(df['Budget_usd'])
plt.show()
```



Regression Model:

In [23]:
```python
#defining X and Y Variable:
x = df.drop(columns=['Budget_usd'], axis=1)
y = df['Budget_usd']
```

In [24]:
```python
# splitting data into tarining and testing:
from sklearn.model_selection import train_test_split

xtrain, xtest, ytrain, ytest = train_test_split(x,y,
                                                train_size=0.8,
                                                random_state = 4)
```

In [25]:
```python
# Importing and training the regression model
from sklearn.linear_model import LinearRegression

model = LinearRegression()
model.fit(xtrain, ytrain)
```

Out[25]:
```
▾ LinearRegression
LinearRegression()
```

In [26]:
```python
#prediction on training and testing data
trainpred = model.predict(xtrain)
testpred = model.predict(xtest)
testpred
```

Out[26]:
```
array([3.51514839, 4.02452245, 3.70670398, ..., 4.00857193, 3.3746481 ,
       4.13532939])
```

In [27]:
```python
#printing model coefficient and intercept
print(model.coef_)
print(model.intercept_)
print(x.columns)
```

```
[-2.54062457e-06  1.28012312e-02  4.11094063e-01  1.33155619e-03
  8.70380709e-02 -2.15490436e-03 -8.32205869e-01  3.05843126e-07
  3.16451375e-06  9.80444512e-06  3.49409985e-05  7.23125722e-04
  2.29972028e-02]
2.8868118341355347
Index(['Title', 'Category Name', 'Experience', 'Sub Category Name', 'Location',
       'Freelancer Preferred From', 'Type', 'Date Posted', 'Description',
       'Client Registration Date', 'Client City', 'Client Country',
       'Client Currency'],
      dtype='object')
```

In [28]:
```python
#identifying mean square error for traing data
from sklearn.metrics import mean_squared_error

mse_train = mean_squared_error(ytrain, trainpred)
mse_train
```

Out[28]:
```
0.23608038401348141
```

In [29]:
```python
# identifying mean square error for testing data
testpred = model.predict(xtest)
mse_test = mean_squared_error(ytest, testpred)
mse_test
```

Out[29]:
```
0.24858028601851517
```

Observation: From above calculation we can conclude that Polynomial Regression model is suitable for Regression on Budget compared to other models, as it gives lowest mean squared error .