Project name :- Freelance Platform Project

Type:-Supervised Machine Learning (Regression model to predict the budget)

Quck Introduction :- The freelance industry has experienced significant growth in recent years, with online platforms connecting freelancers and clients across various industries. This project aimed to perform an exploratory data analysis on a dataset from a freelance platform to gain insights into the dynamics and trends of the platform. By analyzing the data, we aimed to understand the characteristics of freelancers, client preferences, and job postings on the platform.Simultaneously we will use different algorithms to see prediction on the budget

Import libraries

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Load the data

In [2]:
```
df=pd.read_csv('FreelancePlatformProject.csv')

df.head() #Reading the top 5 data
```

Out[2]:

	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 ($)	Graphic Design	EUR	60	remote	ALL	fixed_price	29-04-2023 18:06	We are looking to improve the banner images on...	NaN	03-11-2010	Dublin	Ireland	EUR	PPC Management
1	Make my picture a solid silhouette	Video, Photo & Image	Entry ($)	Image Editing	GBP	20	remote	ALL	fixed_price	29-04-2023 17:40	Hello \n\nI need a quick designer to make 4 pi...	NaN	21-02-2017	London	United Kingdom	GBP	Office manager
2	Bookkeeper needed	Business	Entry ($)	Finance & Accounting	GBP	12	remote	ALL	fixed_price	29-04-2023 17:40	Hi - I need a bookkeeper to assist with bookke...	NaN	09-04-2023	London	United Kingdom	GBP	Paralegal
3	Accountant needed	Business	Entry ($)	Tax Consulting & Advising	GBP	14	remote	ALL	fixed_price	29-04-2023 17:32	Hi - I need an accountant to assist me with un...	NaN	09-04-2023	London	United Kingdom	GBP	Paralegal
4	Guest Post on High DA Website	Digital Marketing	Expert ($$$)	SEO	USD	10000	remote	ALL	fixed_price	29-04-2023 17:09	Hi, I am currently running a project where I w...	NaN	01-07-2016	Mumbai	India	USD	Guest posts buyer

Understanding the data:

Let's examine the data to get a better understanding of its structure and contents.

In [3]:
```
print('Shape of our dataframe is:',df.shape)
```

```
Shape of our dataframe is: (12202, 17)
```

In [4]:
```
df.info()

#Number of columns are:17
#Budget is the only Numerical columns else are Categorical
#Duration and Client job title columns contains null values
#Budget is our dependent variable and other balance features are independent variables
#Output variable is continuous
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12202 entries, 0 to 12201
Data columns (total 17 columns):
 #   Column                     Non-Null Count  Dtype 
---  ------                     --------------  ----- 
 0   Title                      12202 non-null  object
 1   Category Name              12202 non-null  object
 2   Experience                 12202 non-null  object
 3   Sub Category Name          12202 non-null  object
 4   Currency                   12202 non-null  object
 5   Budget                     12202 non-null  int64 
 6   Location                   12202 non-null  object
 7   Freelancer Preferred From  12202 non-null  object
 8   Type                       12202 non-null  object
 9   Date Posted                12202 non-null  object
 10  Description                12202 non-null  object
 11  Duration                   1602 non-null   object
 12  Client Registration Date   12202 non-null  object
 13  Client City                12202 non-null  object
 14  Client Country             12202 non-null  object
 15  Client Currency            12202 non-null  object
 16  Client Job Title           4581 non-null   object
dtypes: int64(1), object(16)
memory usage: 1.6+ MB
```

Checking Duplicate Values

In [5]:
```
df.duplicated().sum()

#There are one duplicate values present in our dataframe
```

Out[5]:
```
1
```

In [6]:
```
# Drop duplicate rows from the DataFrame

df.drop_duplicates(inplace=True)
```

In [7]:
```
df.duplicated().sum()

#Successfully removed the duplicate values
```

Out[7]:
```
0
```

Handling missing data

In [8]:
```
#Check if there are any missing values

df.isnull().sum()

#Duration column has null values=10599
#Client Job title has null values=7620
```

Out[8]:
```
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                   10599
Client Registration Date       0
Client City                    0
Client Country                 0
Client Currency                0
Client Job Title            7620
dtype: int64
```

In [9]:
```
#Checking how much % of missing values present in dataset

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

Out[9]:
```
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                   86.869929
Client Registration Date    0.000000
Client City                 0.000000
Client Country              0.000000
Client Currency             0.000000
Client Job Title           62.453897
dtype: float64
```

In [10]:
```
#Duration column has 86% null values & Client job title has 62% null values. Dropping both columns

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

In [11]:
```
#Checking the shape after dropping 2 columns

print('the shape of dataset is:-',df.shape)
```

```
the shape of dataset is:- (12201, 15)
```

Data Cleaning

Examining each column whether data is cleaned or not

In [12]:
```
df['Experience'].head()
```

Out[12]:
```
0       Entry ($)
1       Entry ($)
2       Entry ($)
3       Entry ($)
4    Expert ($$$)
Name: Experience, dtype: object
```

In [13]:
```
#Removing ($),($$),($$$) symbol and parenthesis from Experience feature by replace function

df['Experience'] = df['Experience'].str.replace('$','').str.replace('(','').str.replace(')','')
df['Experience'].head()
```

Out[13]:
```
0     Entry
1     Entry
2     Entry
3     Entry
4    Expert
Name: Experience, dtype: object
```

In [14]:
```
#Converting all Budget values in usd currency

conversion_rates = {'EUR': 1.07, 'GBP': 1.24, 'USD': 1}

df['Budget_usd'] = df['Currency'].map(conversion_rates) * df['Budget']
df.head(3)
```

Out[14]:

	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	Budget_usd
0	Banner images for web desgin websites	Design	Entry	Graphic Design	EUR	60	remote	ALL	fixed_price	29-04-2023 18:06	We are looking to improve the banner images on...	03-11-2010	Dublin	Ireland	EUR	64.20
1	Make my picture a solid silhouette	Video, Photo & Image	Entry	Image Editing	GBP	20	remote	ALL	fixed_price	29-04-2023 17:40	Hello \n\nI need a quick designer to make 4 pi...	21-02-2017	London	United Kingdom	GBP	24.80
2	Bookkeeper needed	Business	Entry	Finance & Accounting	GBP	12	remote	ALL	fixed_price	29-04-2023 17:40	Hi - I need a bookkeeper to assist with bookke...	09-04-2023	London	United Kingdom	GBP	14.88

In [15]:
```
#We don't need Budget and Currency column as we have created a new column,so dropping Budget

df.drop(['Budget'],axis=1, inplace=True)
```

In [16]:
```
df['Currency'].value_counts()
```

Out[16]:
```
Currency
GBP    8175
USD    3144
EUR     882
Name: count, dtype: int64
```

In [17]:
```
df["Currency"] = "USD" #Assignning USD to all currency
```

In [18]:
```
#Checking the shape after dropping 2 columns

print('The shape of dataset is:-',df.shape)
```

```
The shape of dataset is:- (12201, 15)
```

In [19]:
```
#Exploring the statistical information

df.describe()
```

Out[19]:

	Budget_usd
count	12201.000000
mean	266.237785
std	2282.447586
min	7.440000
25%	37.200000
50%	93.000000
75%	186.000000
max	123998.760000

Feature Extraction

Lets do some Feature Extraction on columns

In [20]:
```
# #Client Registration Date column having object datatype so converting its datatype and extracting it to day/month/year format

df['Client Registration Date'] = pd.to_datetime(df['Client Registration Date'], format='%d-%m-%Y', errors='coerce')
df['Client Registration date'] = df['Client Registration Date'].dt.day
df['Client Registration Month'] = df['Client Registration Date'].dt.month
df['Client Registration Year'] = df['Client Registration Date'].dt.year
```

In [21]:
```
#Date Posted column having object datatype so converting its datatype and extracting it to day/month/year format

df['Date Posted'] = pd.to_datetime(df['Date Posted'], format='%d-%m-%Y %H:%M')
df['Date Posted in Date'] = df['Date Posted'].dt.day
df['Date Posted in Month'] = df['Date Posted'].dt.month
df['Date Posted in Year'] = df['Date Posted'].dt.year
df['Date Posted in Time'] = df['Date Posted'].dt.time
```

In [22]:
```
#Dropping date posted column as we already have done feature extraction

df.drop(['Date Posted','Client Registration Date'],axis=1, inplace=True)
```

In [23]:
```
#Checking the shape after dropping Date Posted column

print('The shape of dataset is:-',df.shape)
```

```
The shape of dataset is:- (12201, 20)
```

Checking skewness

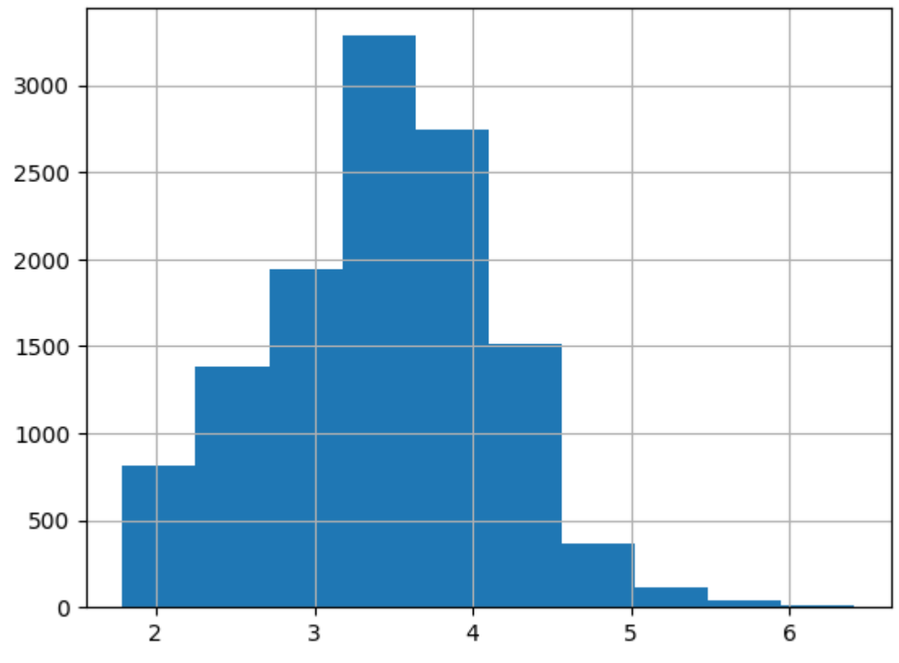
In [24]:
```
df['Budget_usd'].skew()
```

Out[24]: 43.9364193416511

In [25]:
```
#Box cox transformation

from scipy.stats import boxcox
import numpy as np

after_boxcox = boxcox(df['Budget_usd'])
after_boxcox = pd.Series(after_boxcox[0])
df['Budget_usd'] = after_boxcox

after_boxcox.skew() #We successfully reduced the skewness
```

Out[25]: -0.002291121221937113

In [26]:
```
#Checking skewness by visulization, we can see data is normally distributed with bell shape curve

import matplotlib.pyplot as plt
plt.hist(df['Budget_usd'])
plt.grid()

plt.show()
```



In [77]:
```
df.describe()
```

Out[77]:

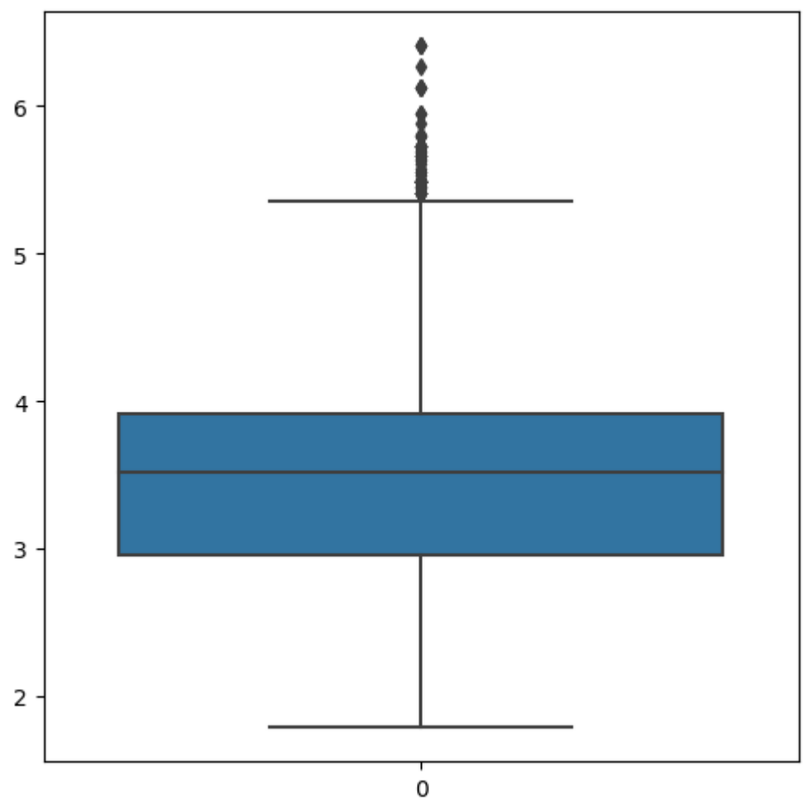
	Sub Category Name	Location	Freelancer Preferred From	Client City	Client Country	Client Currency	Budget_usd	Client Registration date	Client Registration Month	Client Registration Year	...	Category_Digital Marketing	Category_Marketing, Branding & Sales	Category_Music & Audio	Category_S N
count	12077.000000	12077.000000	12077.000000	12077.000000	12077.000000	12077.000000	12077.000000	12077.000000	12077.000000	12077.000000	...	12077.000000	12077.000000	12077.000000	12077.00
mean	0.535481	0.526952	0.053395	0.482149	0.840627	0.589054	0.459207	0.485664	0.401289	0.737088	...	0.065000	0.058293	0.016560	0.03
std	0.307742	0.124432	0.122480	0.252392	0.238944	0.286319	0.195801	0.288494	0.315381	0.250189	...	0.246535	0.234306	0.127623	0.18
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	0.00
25%	0.283019	0.500000	0.024390	0.267629	0.948148	0.500000	0.326561	0.233333	0.090909	0.562500	...	0.000000	0.000000	0.000000	0.00
50%	0.490566	0.500000	0.024390	0.520822	0.955556	0.500000	0.486982	0.500000	0.272727	0.812500	...	0.000000	0.000000	0.000000	0.00
75%	0.867925	0.500000	0.024390	0.630761	0.955556	1.000000	0.597497	0.733333	0.727273	1.000000	...	0.000000	0.000000	0.000000	0.00
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	...	1.000000	1.000000	1.000000	1.00

8 rows × 27 columns

Handling Outliers

In [27]:
```
#Checking the Outliers by boxplot

plt.figure(figsize=(6,6))

sns.boxplot(df['Budget_usd']) #Need to remove outliers present in the dataframe
```

Out[27]: <Axes: >



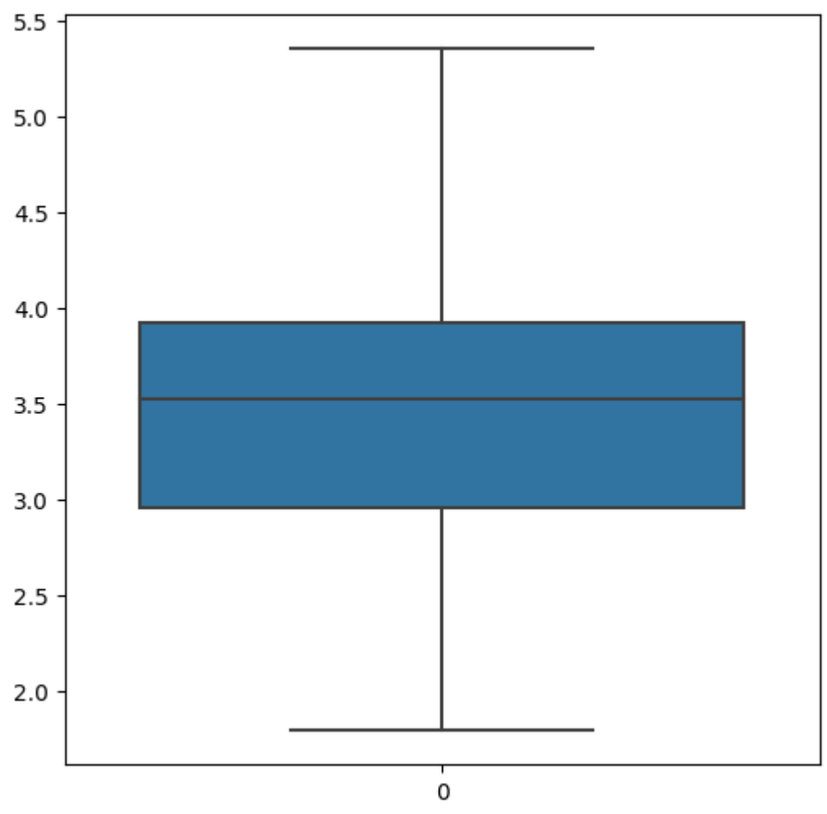
In [28]:
```
#Counting the values that are greater then 5.4

# Convert 'Budget_usd' column to numeric data type
# df['Budget_usd'] = pd.to_numeric(df['Budget_usd'], errors='coerce')

# Count the values greater than 5.4
count = (df['Budget_usd'] > 5.4).sum()
print("Count of values greater than 5.4:", count)
```

```
Count of values greater than 5.4: 63
```

In [29]:
```
#Dropping the outliers from Budget_usd

df= df.drop(df[df['Budget_usd'] > 5.4].index)
```

In [30]:
```
#Checking the Outliers after removing

plt.figure(figsize=(6,6))

sns.boxplot(df['Budget_usd']) #We successfully eliminated all outliers
```

Out[30]: <Axes: >



In [31]:
```
print('The shape of dataset is:-',df.shape)
```

```
The shape of dataset is:- (12138, 20)
```

In [32]:
```
df.dtypes
```

Out[32]:
```
Title                        object
Category Name                object
Experience                   object
Sub Category Name            object
Currency                     object
Location                     object
Freelancer Preferred From    object
Type                         object
Description                  object
Client City                  object
Client Country               object
Client Currency              object
Budget_usd                  float64
Client Registration date      int32
Client Registration Month     int32
Client Registration Year      int32
Date Posted in Date           int32
Date Posted in Month          int32
Date Posted in Year           int32
Date Posted in Time          object
dtype: object
```

Data Analysis

In [33]:
```
df.head()
```

Out[33]:

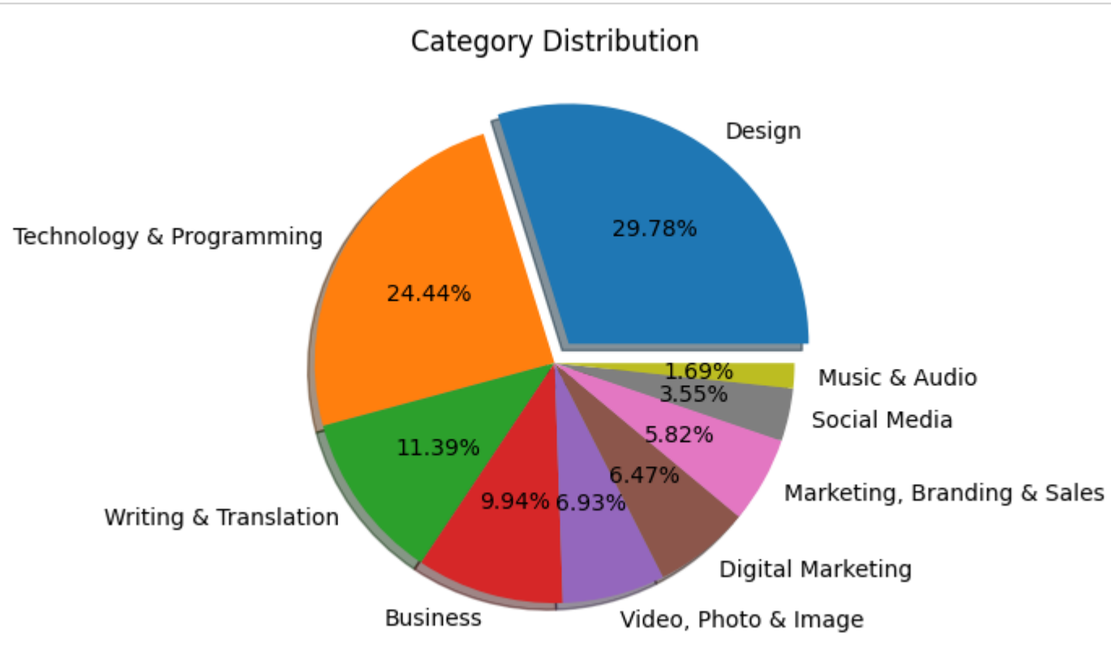
	Title	Category Name	Experience	Sub Category Name	Currency	Location	Freelancer Preferred From	Type	Description	Client City	Client Country	Client Currency	Budget_usd	Client Registration date	Client Registration Month	Client Registration Year	Date Posted in Date	Date Posted in Month	Date Posted in Year	D Pos in Ti
0	Banner images for web desgin websites	Design	Entry	Graphic Design	USD	remote	ALL	fixed_price	We are looking to improve the banner images on...	Dublin	Ireland	EUR	3.301078	3	11	2010	29	4	2023	18:0(
1	Make my picture a solid silhouette	Video, Photo & Image	Entry	Image Editing	USD	remote	ALL	fixed_price	Hello \n\nI need a quick designer to make 4 pi...	London	United Kingdom	GBP	2.680607	21	2	2017	29	4	2023	17:4(
2	Bookkeeper needed	Business	Entry	Finance & Accounting	USD	remote	ALL	fixed_price	Hi - I need a bookkeeper to assist with bookke...	London	United Kingdom	GBP	2.318016	9	4	2023	29	4	2023	17:4(
3	Accountant needed	Business	Entry	Tax Consulting & Advising	USD	remote	ALL	fixed_price	Hi - I need an accountant to assist me with un...	London	United Kingdom	GBP	2.429707	9	4	2023	29	4	2023	17:32
5	Content Database Project for Travel Company	Technology & Programming	Expert	Databases	USD	remote	ALL	fixed_price	Brief\nThe requirements of this brief is to fi...	Dubai	United Arab Emirates	EUR	4.460790	14	9	2013	29	4	2023	17:0(

In [34]:
```
df['Category Name'].value_counts()
```

Out[34]:
```
Category Name
Design                          3615
Technology & Programming        2966
Writing & Translation           1383
Business                        1206
Video, Photo & Image             841
Digital Marketing                785
Marketing, Branding & Sales      706
Social Media                     431
Music & Audio                    205
Name: count, dtype: int64
```

Understanding the category distibution

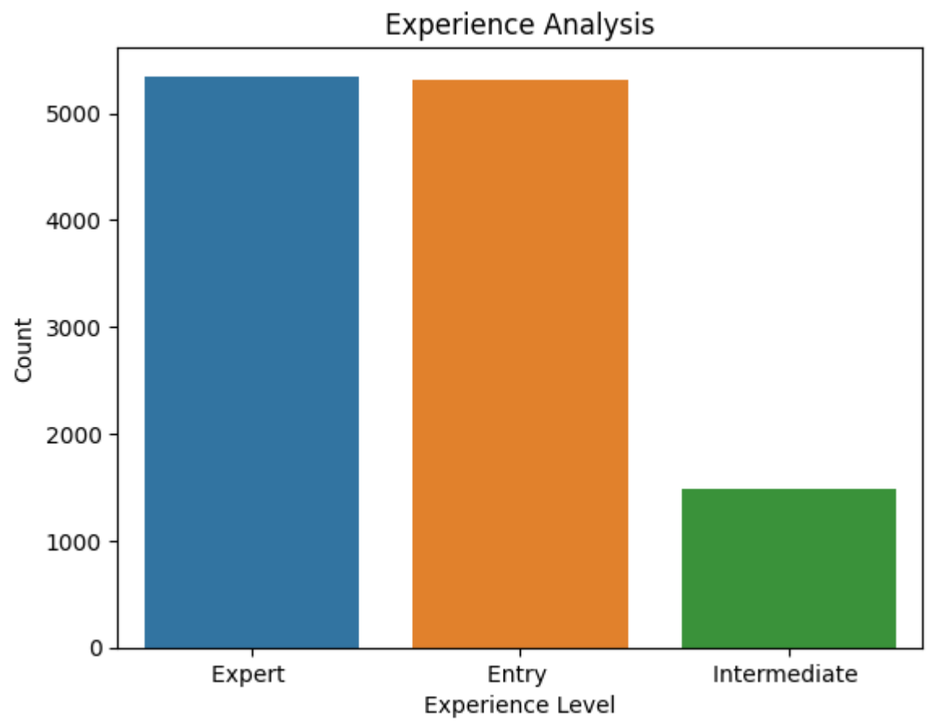
In [35]:
```
category_counts = df['Category Name'].value_counts()

ex=[0.1,0,0,0,0,0,0,0,0]

# Plotting the pie chart
plt.pie(category_counts,explode=ex, labels=category_counts.index, autopct='%0.2f%%', shadow=True)

# Adding a title to the chart
plt.title("Category Distribution")

# Show the plot
plt.show()
```



From above graph we can understand that the Design category has the highest and Music & Audio has the lowest distribution

Understanding the location wise experience

In [36]:
```
experience_counts = df['Experience'].value_counts()

# Plotting the chart
sns.barplot(x = experience_counts.index, y = experience_counts.values)

# Adding a title,label
plt.title('Experience Analysis')
plt.xlabel('Experience Level')
plt.ylabel('Count')

# Show the plot
plt.show()
```

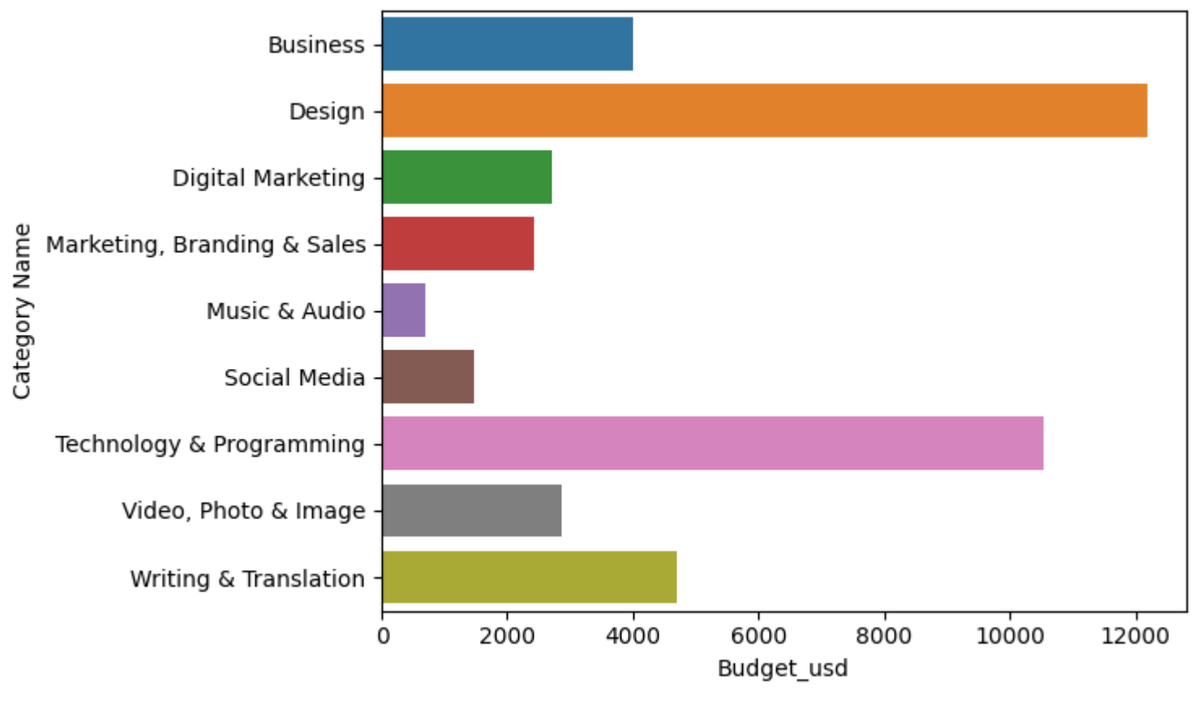


From above graph we can understand that the location wise Expert and entry is higher while intermediate is lower

Undertanding which category has the highest and lowest budget category wise

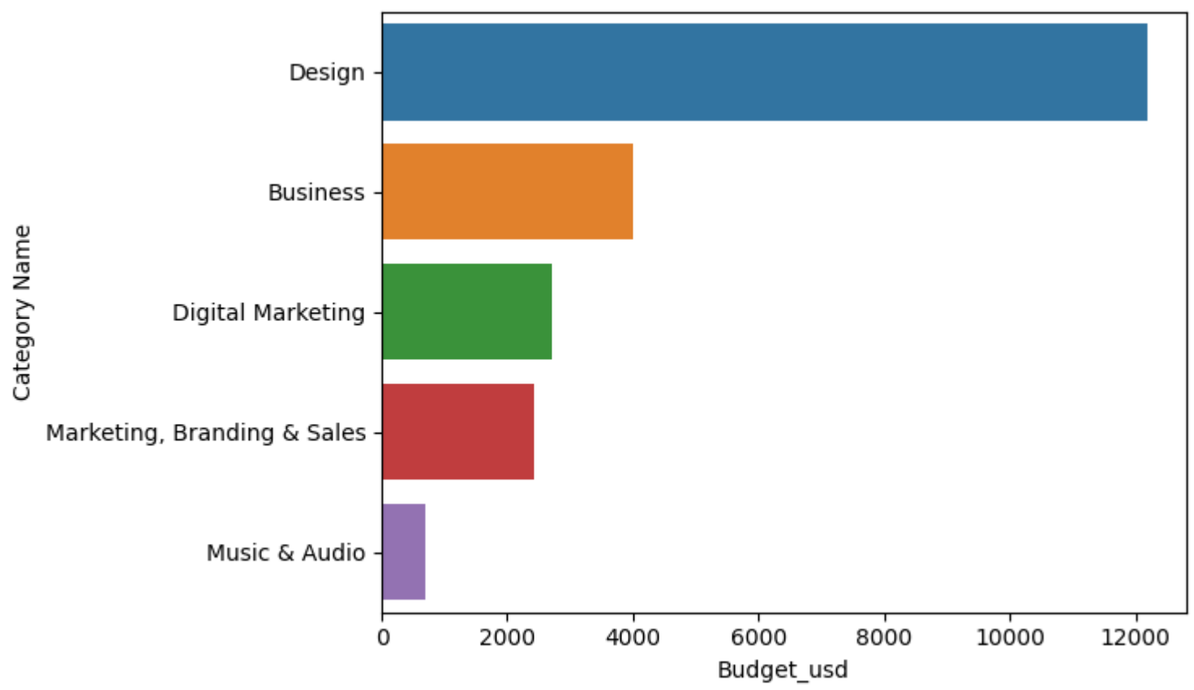
In [37]:
```
grouped = df.groupby('Category Name')['Budget_usd'].sum()

# Plotting the bar plot
sns.barplot(y= grouped.index,x= grouped)
```

Out[37]: <Axes: xlabel='Budget_usd', ylabel='Category Name'>



In [38]:
```
lowest = grouped.head(5).sort_values(ascending = False)

# Plotting the bar plot
sns.barplot(y= lowest.index,x= lowest)
```

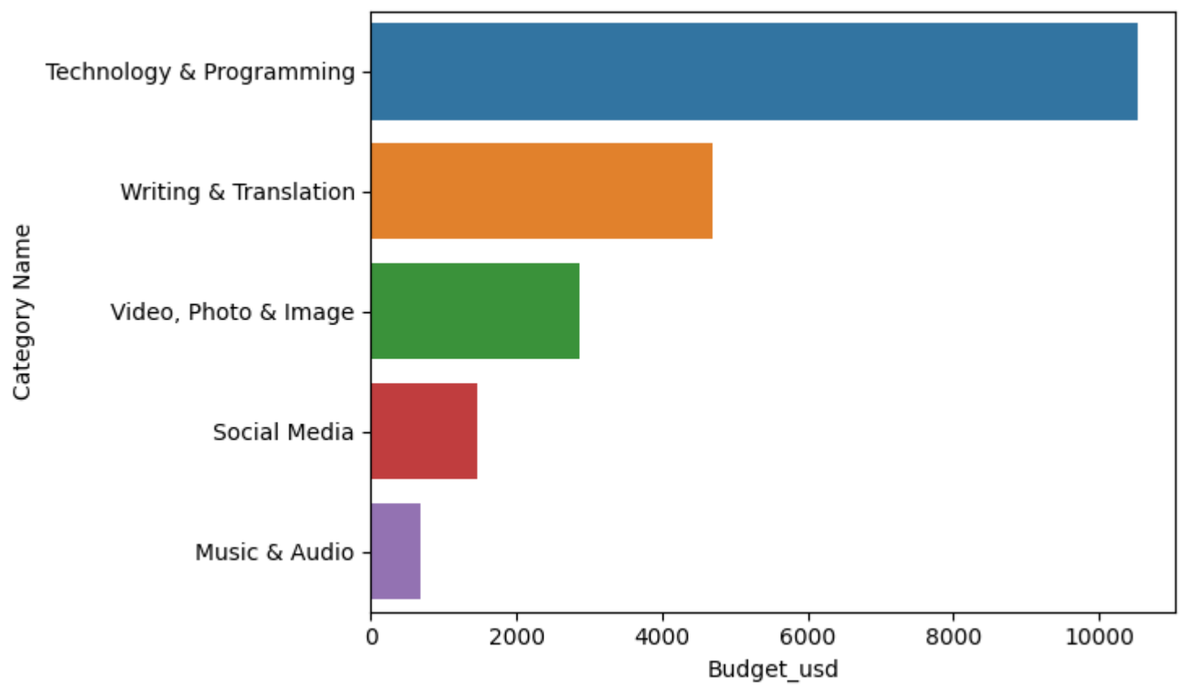
Out[38]: <Axes: xlabel='Budget_usd', ylabel='Category Name'>



From the above graph we can understand the design, business, digital marketing, branding sales, music and audio have the lowest budget

In [39]:
```
highest = grouped.tail(5).sort_values(ascending = False)

# Plotting the bar plot
sns.barplot(y= highest.index, x= highest)
```

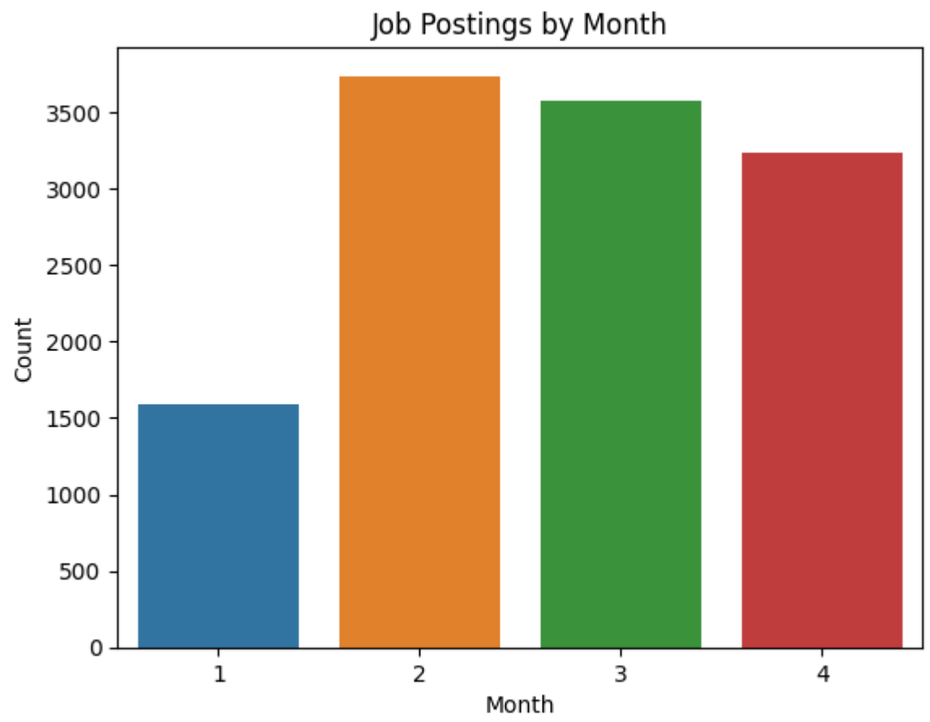
Out[39]: <Axes: xlabel='Budget_usd', ylabel='Category Name'>



From the above graph we can understand that technology & programming, writing & translation, video, photo & Image,Social media have the highest budget

Understanding how much jobs are posted by month wise

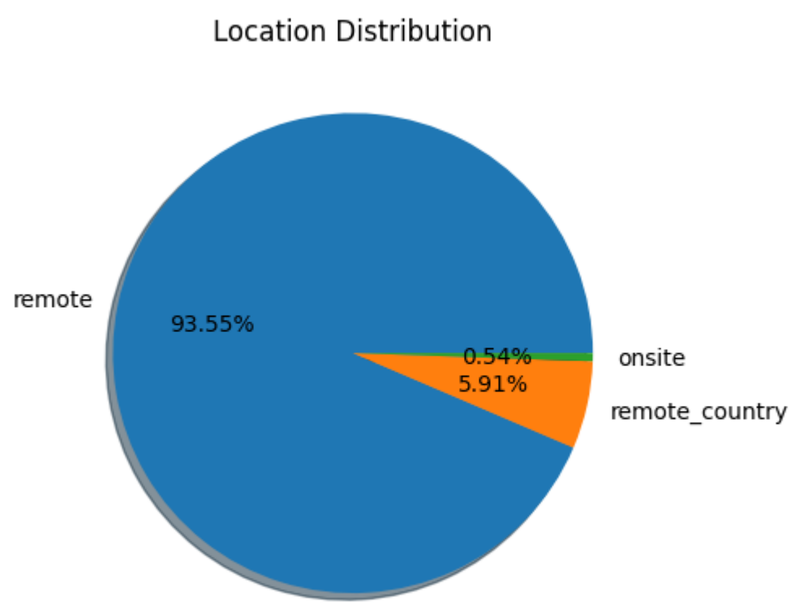
In [40]:
```
date = df['Date Posted in Month'].value_counts()

# Plotting the bar plot
sns.barplot(x = date.index , y = date.values)

#Adding a title,label
plt.title('Job Postings by Month')
plt.xlabel('Month')
plt.ylabel('Count')

# Show the plot
plt.show()
```



From the above graph we can understand in 2nd month job were posted the most

Understanding the Distribution Location wise

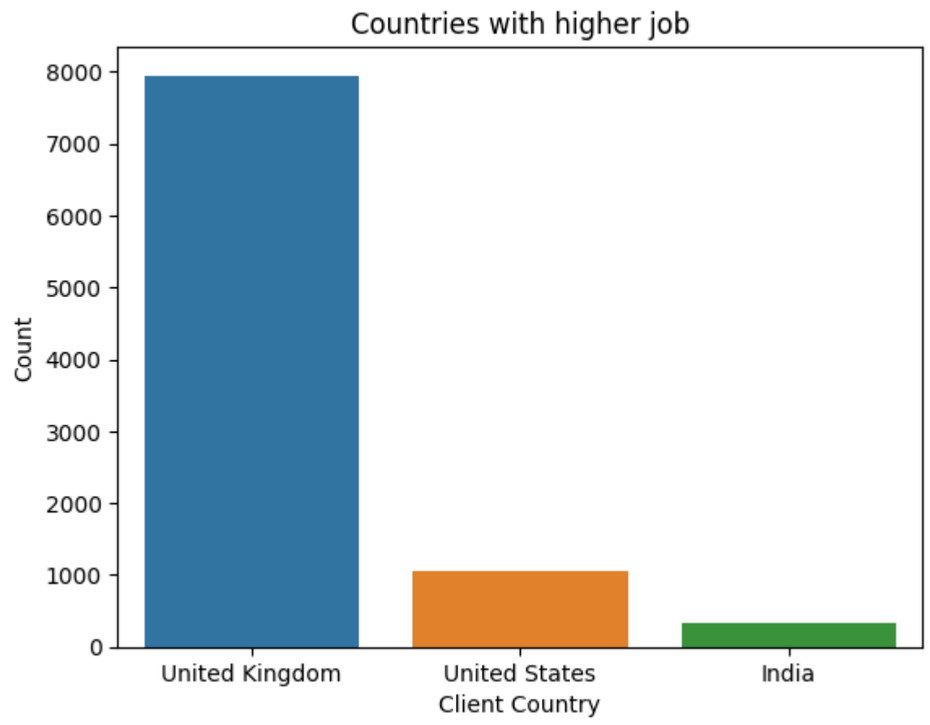
In [41]:
```
location_counts = df['Location'].value_counts()

# Plotting the pie chart
plt.pie(location_counts, labels=location_counts.index, autopct='%0.2f%%', shadow=True)

# Adding a title to the chart
plt.title("Location Distribution")

# Show the plot
plt.show()
```



From the above graph we can understand remote area have the most freelancing jobs

Understanding the top 3 countries which has higher jobs

In [42]:
```
top = df['Client Country'].value_counts().sort_values(ascending = False)
top_new = top.head(3)

# Plotting the bar plot
sns.barplot(x = top_new.index, y = top_new.values)

# Adding the title,label
plt.title('Countries with higher job')
plt.xlabel('Client Country')
plt.ylabel('Count')

# Show the plot
plt.show()
```



From the above graph we can understand that United Kingdom, United States, India are the top 3 countries which has the higher jobs

Understanding the top three countries which has expert, intermediate and entry experience

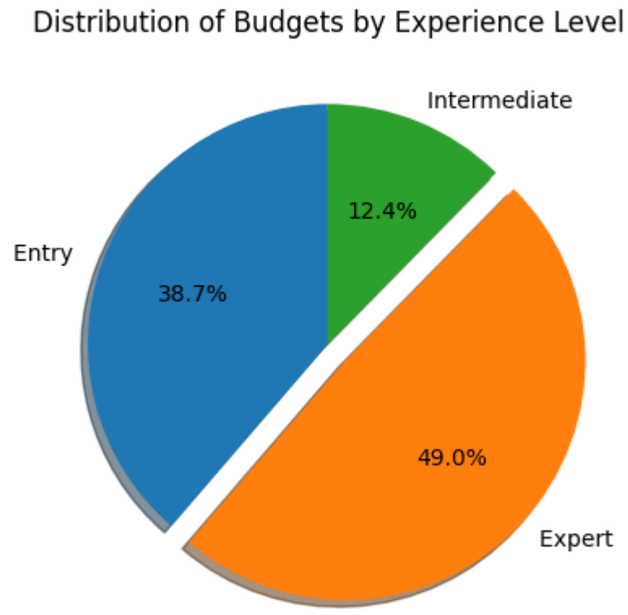
In [43]:
```
entry_df  = df[df['Experience'] == 'Entry ']
expert_df = df[df['Experience'] == 'Expert ']
intermediate_df = df[df['Experience'] == 'Intermediate ']

top_entry = entry_df['Client Country'].value_counts().head(3)
top_expert = expert_df['Client Country'].value_counts().head(3)
top_intermediate = intermediate_df['Client Country'].value_counts().head(3)
```

In [44]:
```
print(top_entry,top_expert,top_intermediate)
```

```
Client Country
United Kingdom    3391
United States      424
India              206
Name: count, dtype: int64 Client Country
United Kingdom    3593
United States      493
India               90
Name: count, dtype: int64 Client Country
United Kingdom    963
United States     140
India              35
Name: count, dtype: int64
```

Understanding the Budget distribution experience wise

In [45]:
```
grouped = df.groupby('Experience').agg({'Budget_usd' : 'sum'})

explode=(0,0.1,0)

# Create the pie chart
plt.pie(grouped['Budget_usd'],explode = explode, labels=grouped.index, autopct='%1.1f%%', startangle=90, shadow=True)

# Set the chart title
plt.title('Distribution of Budgets by Experience Level')

# Show the plot
plt.show()
```

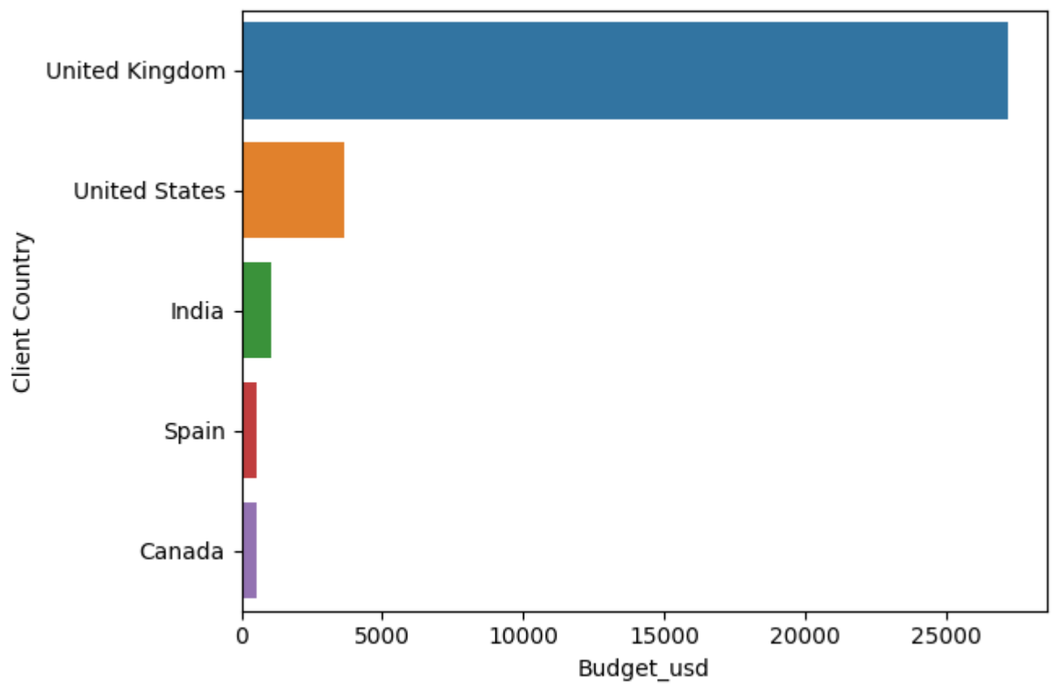


From the above graph we can understand that entry level has the highest budget while intermediate level has the lowest

Understanding the Top 5 highest budget distribution client country wise

In [46]:
```
grouped = df.groupby('Client Country').agg({'Budget_usd' : 'sum'}).sort_values(by = 'Budget_usd', ascending = False).head(5)

# Plotting the bar plot
sns.barplot(y = grouped.index, x = grouped['Budget_usd'])
```

Out[46]: <Axes: xlabel='Budget_usd', ylabel='Client Country'>



From the above graph we can understand the top highest 5 budget distribution client country wise

Categorical data encoding

In [47]:
```
df.dtypes
```

Out[47]:
```
Title                          object
Category Name                  object
Experience                     object
Sub Category Name              object
Currency                       object
Location                       object
Freelancer Preferred From      object
Type                           object
Description                    object
Client City                    object
Client Country                 object
Client Currency                object
Budget_usd                    float64
Client Registration date        int32
Client Registration Month       int32
Client Registration Year        int32
Date Posted in Date             int32
Date Posted in Month            int32
Date Posted in Year             int32
Date Posted in Time            object
dtype: object
```

In [48]:
```
df['Experience'].value_counts()
```

Out[48]:
```
Experience
Expert          5347
Entry           5314
Intermediate    1477
Name: count, dtype: int64
```

In [49]:
```
df['Type'].value_counts()
```

Out[49]:
```
Type
fixed_price    10353
hourly          1785
Name: count, dtype: int64
```

In [50]:
```
df['Category Name'].value_counts()
```

Out[50]:
```
Category Name
Design                         3615
Technology & Programming       2966
Writing & Translation          1383
Business                       1206
Video, Photo & Image            841
Digital Marketing               785
Marketing, Branding & Sales     706
Social Media                    431
Music & Audio                   205
Name: count, dtype: int64
```

In [51]:
```
import pandas as pd

# Apply one-hot encoding to 'Experience', 'Type', 'Category Name' columns
one_hot_encoded = pd.get_dummies(df[['Type', 'Category Name','Experience']], dtype=int, prefix=['Type', 'Category','Experience'])

# Drop the original 'Experience', 'Type', 'Category Name' columns from the DataFrame
df = df.drop(['Type', 'Category Name','Experience'], axis=1)

# Concatenate the original DataFrame and the one-hot encoded DataFrame
df = pd.concat([df, one_hot_encoded], axis=1)

# Print the updated DataFrame
print(df.head())
```

```
                                    Title          Sub Category Name
0        Banner images for web desgin websites             Graphic Design  \
1          Make my picture a solid silhouette               Image Editing
2                        Bookkeeper needed       Finance & Accounting
3                       Accountant needed   Tax Consulting & Advising
5  Content Database Project for Travel Company                  Databases

  Currency Location Freelancer Preferred From
0      USD   remote                       ALL  \
1      USD   remote                       ALL
2      USD   remote                       ALL
3      USD   remote                       ALL
5      USD   remote                       ALL

                                     Description Client City
0  We are looking to improve the banner images on...      Dublin  \
1  Hello \n\nI need a quick designer to make 4 pi...      London
2  Hi - I need a bookkeeper to assist with bookke...      London
3  Hi - I need an accountant to assist me with un...      London
5  Brief\nThe requirements of this brief is to fi...       Dubai

      Client Country Client Currency  Budget_usd  ...
0             Ireland             EUR    3.301078  ...  \
1       United Kingdom             GBP    2.680607  ...
2       United Kingdom             GBP    2.318016  ...
3       United Kingdom             GBP    2.429707  ...
5  United Arab Emirates             EUR    4.460790  ...

   Category_Digital Marketing  Category_Marketing, Branding & Sales
0                           0                                     0  \
1                           0                                     0
2                           0                                     0
3                           0                                     0
5                           0                                     0

   Category_Music & Audio  Category_Social Media
0                       0                     0  \
1                       0                     0
2                       0                     0
3                       0                     0
5                       0                     0

   Category_Technology & Programming  Category_Video, Photo & Image
0                                  0                             0  \
1                                  0                             1
2                                  0                             0
3                                  0                             0
5                                  1                             0

  Category_Writing & Translation  Experience_Entry   Experience_Expert
0                              0                 1                   0  \
1                              0                 1                   0
2                              0                 1                   0
3                              0                 1                   0
5                              0                 0                   1

   Experience_Intermediate
0                        0
1                        0
2                        0
3                        0
5                        0

[5 rows x 31 columns]
```

In [52]:
```
df.columns
```

Out[52]:
```
Index(['Title', 'Sub Category Name', 'Currency', 'Location',
       'Freelancer Preferred From', 'Description', 'Client City',
       'Client Country', 'Client Currency', 'Budget_usd',
       'Client Registration date', 'Client Registration Month',
       'Client Registration Year', 'Date Posted in Date',
       'Date Posted in Month', 'Date Posted in Year', 'Date Posted in Time',
       'Type_fixed_price', 'Type_hourly', 'Category_Business',
       'Category_Design', 'Category_Digital Marketing',
       'Category_Marketing, Branding & Sales', 'Category_Music & Audio',
       'Category_Social Media', 'Category_Technology & Programming',
       'Category_Video, Photo & Image', 'Category_Writing & Translation',
       'Experience_Entry ', 'Experience_Expert ', 'Experience_Intermediate '],
      dtype='object')
```

In [53]:
```
from sklearn.preprocessing import LabelEncoder

# Identify object columns in the DataFrame
object_columns = df.select_dtypes(include=['object']).columns

# Apply label encoding to each object column
le = LabelEncoder()
for column in object_columns:
    df[column] = le.fit_transform(df[column])
```

In [54]:
```
# Checking correlation using heatmap

plt.figure(figsize=(30,15))
sns.heatmap(df.corr(),cmap='PiYG',annot=True);
```

Checking the correlation between the parameters

Darker the colour toward the green shows more the correlation between each other.

In [55]:
```
df.corr()
```

Out[55]:

	Title	Sub Category Name	Currency	Location	Freelancer Preferred From	Description	Client City	Client Country	Client Currency	Budget_usd	...	Category_Digital Marketing	Category_Marketing, Branding & Sales	Category_Music & Audio	Category_Social Media	Category_ & Pr
Title	1.000000	0.132591	NaN	0.017934	0.027072	0.072296	0.011815	0.006451	-0.015450	-0.021606	...	-0.012734	-0.000881	0.005364	0.027058	
Sub Category Name	0.132591	1.000000	NaN	-0.045176	-0.060265	-0.009575	0.011043	-0.022761	0.012042	0.092120	...	0.049096	0.023692	0.055610	0.009840	
Currency	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	
Location	0.017934	-0.045176	NaN	1.000000	0.767232	0.022793	-0.017821	0.052297	-0.041783	-0.020009	...	0.018752	0.046981	-0.018015	0.006965	
Freelancer Preferred From	0.027072	-0.060265	NaN	0.767232	1.000000	0.038218	-0.031136	0.062014	-0.043805	-0.051792	...	0.000011	0.059208	-0.015515	0.004446	
Description	0.072296	-0.009575	NaN	0.022793	0.038218	1.000000	-0.001806	0.017121	-0.007974	-0.001809	...	0.000658	0.023811	-0.022412	-0.004602	
Client City	0.011815	0.011043	NaN	-0.017821	-0.031136	-0.001806	1.000000	-0.022709	0.074847	0.027469	...	-0.000016	0.008289	0.023289	0.006076	
Client Country	0.006451	-0.022761	NaN	0.052297	0.062014	0.017121	-0.022709	1.000000	-0.126571	0.006536	...	-0.009827	0.026956	-0.033258	0.018408	
Client Currency	-0.015450	0.012042	NaN	-0.041783	-0.043805	-0.007974	0.074847	-0.126571	1.000000	0.034350	...	0.012904	-0.010374	0.042954	-0.013876	
Budget_usd	-0.021606	0.092120	NaN	-0.020009	-0.051792	-0.001809	0.027469	0.006536	0.034350	1.000000	...	0.014832	-0.002490	-0.013518	-0.004510	
Client Registration date	0.006780	0.001204	NaN	0.000790	0.007993	-0.007892	0.024297	0.001320	-0.002398	-0.021537	...	0.011198	-0.005599	0.024107	-0.009938	
Client Registration Month	0.001893	0.015428	NaN	0.011353	-0.014911	0.002096	-0.016144	0.014129	0.010360	-0.008550	...	0.034329	0.018512	-0.018604	0.010866	
Client Registration Year	-0.030986	-0.009952	NaN	-0.050867	-0.029698	0.003509	0.034365	-0.110614	0.112682	0.039579	...	-0.017434	0.001688	0.027014	-0.007744	
Date Posted in Date	-0.003846	-0.006886	NaN	0.008177	0.006477	-0.013068	0.018955	-0.016341	-0.012452	-0.017007	...	-0.009451	-0.010104	0.024561	-0.001759	
Date Posted in Month	0.000114	-0.001861	NaN	0.005118	-0.004300	0.005208	-0.026790	0.017002	0.014550	0.003475	...	0.021405	0.008475	-0.036836	-0.001889	
Date Posted in Year	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	
Date Posted in Time	0.000919	-0.006979	NaN	-0.006692	0.014471	0.024634	-0.004704	0.068246	-0.066496	0.005279	...	-0.014459	0.001996	-0.014773	-0.012503	
Type_fixed_price	-0.023586	0.027816	NaN	-0.080831	-0.093147	-0.038270	-0.001221	-0.025372	0.026107	0.395934	...	-0.040253	-0.074716	0.020123	-0.022145	
Type_hourly	0.023586	-0.027816	NaN	0.080831	0.093147	0.038270	0.001221	0.025372	-0.026107	-0.395934	...	0.040253	0.074716	-0.020123	0.022145	
Category_Business	-0.022554	-0.317265	NaN	0.135696	0.147229	0.022988	-0.036931	0.071486	-0.056304	-0.050851	...	-0.087338	-0.082540	-0.043534	-0.063729	
Category_Design	-0.055491	-0.158654	NaN	-0.057971	-0.067875	-0.015621	-0.000297	0.022782	-0.005506	-0.048405	...	-0.171253	-0.161845	-0.085361	-0.124961	
Category_Digital Marketing	-0.012734	0.049096	NaN	0.018752	0.000011	0.000658	-0.000016	-0.009827	0.012904	0.014832	...	1.000000	-0.065346	-0.034465	-0.050454	
Category_Marketing, Branding & Sales	-0.000881	0.023692	NaN	0.046981	0.059208	0.023811	0.008289	0.026956	-0.010374	-0.002490	...	-0.065346	1.000000	-0.032572	-0.047682	
Category_Music & Audio	0.005364	0.055610	NaN	-0.018015	-0.015515	-0.022412	0.023289	-0.033258	0.042954	-0.013518	...	-0.034465	-0.032572	1.000000	-0.025149	
Category_Social Media	0.027058	0.009840	NaN	0.006965	0.004446	-0.004602	0.006076	0.018408	-0.013876	-0.004510	...	-0.050454	-0.047682	-0.025149	1.000000	
Category_Technology & Programming	0.051597	0.314746	NaN	-0.039446	-0.057502	-0.008196	-0.000952	-0.039679	0.006101	0.102033	...	-0.149532	-0.141317	-0.074534	-0.109111	
Category_Video, Photo & Image	0.024568	0.059370	NaN	-0.041956	0.008866	-0.008403	0.014527	-0.013086	0.014774	-0.011352	...	-0.071746	-0.067804	-0.035762	-0.052352	
Category_Writing & Translation	0.004255	-0.029810	NaN	-0.003316	-0.010166	0.012360	0.005798	-0.045456	0.029191	-0.012929	...	-0.094294	-0.089114	-0.047001	-0.068805	
Experience_Entry	0.022497	-0.024441	NaN	-0.040139	-0.015967	-0.021796	-0.027253	-0.029944	0.009750	-0.505025	...	-0.070674	-0.031279	0.042852	-0.031130	
Experience_Expert	-0.000777	0.040091	NaN	0.051621	0.034329	0.020983	0.015716	0.034314	-0.001531	0.486605	...	0.060182	0.019140	-0.042894	0.013574	
Experience_Intermediate	-0.032961	-0.023791	NaN	-0.017476	-0.027901	0.001212	0.017491	-0.006666	-0.012471	0.027470	...	0.015859	0.018402	0.000107	0.026628	

1 rows × 31 columns

In [56]:
```
#Dropping below columns as it's not needed for predictions

df.drop(['Date Posted in Year','Currency'],axis=1,inplace=True)
```

In [57]:
```
df.columns
```

Out[57]:
```
Index(['Title', 'Sub Category Name', 'Location', 'Freelancer Preferred From',
       'Description', 'Client City', 'Client Country', 'Client Currency',
       'Budget_usd', 'Client Registration date', 'Client Registration Month',
       'Client Registration Year', 'Date Posted in Date',
       'Date Posted in Month', 'Date Posted in Time', 'Type_fixed_price',
       'Type_hourly', 'Category_Business', 'Category_Design',
       'Category_Digital Marketing', 'Category_Marketing, Branding & Sales',
       'Category_Music & Audio', 'Category_Social Media',
       'Category_Technology & Programming', 'Category_Video, Photo & Image',
       'Category_Writing & Translation', 'Experience_Entry ',
       'Experience_Expert ', 'Experience_Intermediate '],
      dtype='object')
```

Scaling the Data

In [58]:
```
from sklearn.preprocessing import MinMaxScaler

# Initialize the MinMaxScaler
scaler = MinMaxScaler()

# Specify the column(s) to be scaled
column_to_scale = df.columns

# Apply Min-Max scaling to the selected column(s)
df[column_to_scale] = scaler.fit_transform(df[column_to_scale])

# Print the scaled DataFrame
print(df)
```

```
          Title  Sub Category Name  Location  Freelancer Preferred From
0      0.083065           0.396226       0.5                    0.024390  \
1      0.550613           0.424528       0.5                    0.024390
2      0.095056           0.349057       0.5                    0.024390
3      0.040056           0.849057       0.5                    0.024390
5      0.156399           0.245283       0.5                    0.024390
...         ...                ...       ...                         ...
12197  0.687114           0.198113       0.5                    0.024390
12198  0.759840           0.952830       1.0                    0.390244
12199  0.770614           0.235849       0.5                    0.024390
12200  0.176992           0.952830       0.5                    0.024390
12201  0.120862           0.009434       0.5                    0.024390

       Description  Client City  Client Country  Client Currency  Budget_usd
0         0.875306     0.271516        0.451852              0.0    0.424146  \
1         0.104432     0.520822        0.955556              0.5    0.249945
2         0.182440     0.520822        0.955556              0.5    0.148145
3         0.182609     0.520822        0.955556              0.5    0.179503
5         0.047531     0.270961        0.948148              0.0    0.749742
...            ...          ...             ...              ...         ...
12197     0.306543     0.026097        0.651852              0.5    0.463038
12198     0.731448     0.307607        0.955556              0.5    0.418093
12199     0.549515     0.520822        0.955556              0.5    0.335257
12200     0.347995     0.628540        0.429630              1.0    0.486982
12201     0.264500     0.350916        0.955556              0.5         NaN

       Client Registration date  ...  Category_Digital Marketing
0                      0.066667  ...                         0.0  \
1                      0.666667  ...                         0.0
2                      0.266667  ...                         0.0
3                      0.266667  ...                         0.0
5                      0.433333  ...                         0.0
...                         ...  ...                         ...
12197                  0.166667  ...                         0.0
12198                  0.733333  ...                         0.0
12199                  0.433333  ...                         0.0
12200                  0.666667  ...                         0.0
12201                  0.666667  ...                         0.0

       Category_Marketing, Branding & Sales  Category_Music & Audio
0                                       0.0                    0.0  \
1                                       0.0                    0.0
2                                       0.0                    0.0
3                                       0.0                    0.0
5                                       0.0                    0.0
...                                     ...                    ...
12197                                   0.0                    0.0
12198                                   0.0                    0.0
12199                                   0.0                    0.0
12200                                   0.0                    0.0
12201                                   0.0                    0.0

       Category_Social Media  Category_Technology & Programming
0                        0.0                                0.0  \
1                        0.0                                0.0
2                        0.0                                0.0
3                        0.0                                0.0
5                        0.0                                1.0
...                      ...                                ...
12197                    0.0                                0.0
12198                    0.0                                0.0
12199                    0.0                                1.0
12200                    0.0                                0.0
12201                    0.0                                0.0

       Category_Video, Photo & Image  Category_Writing & Translation
0                                0.0                             0.0  \
1                                1.0                             0.0
2                                0.0                             0.0
3                                0.0                             0.0
5                                0.0                             0.0
...                              ...                             ...
12197                            0.0                             1.0
12198                            0.0                             0.0
12199                            0.0                             0.0
12200                            0.0                             0.0
12201                            0.0                             0.0

       Experience_Entry   Experience_Expert   Experience_Intermediate
0                   1.0                 0.0                       0.0
1                   1.0                 0.0                       0.0
2                   1.0                 0.0                       0.0
3                   1.0                 0.0                       0.0
5                   0.0                 1.0                       0.0
...                 ...                 ...                       ...
12197               1.0                 0.0                       0.0
12198               0.0                 0.0                       1.0
12199               1.0                 0.0                       0.0
12200               0.0                 1.0                       0.0
12201               0.0                 1.0                       0.0

[12138 rows x 29 columns]
```

In [59]:
```
#Dropping below columns which are not required for predictions

columns_to_drop = ['Title', 'Description'] #,'Client Country'
df.drop(columns=columns_to_drop, inplace=True)
```

Checking Duplicated Values before training the model

In [60]:
```
df.duplicated().sum()
```

Out[60]:
```
60
```

In [61]:
```
df.drop_duplicates(inplace=True)
```

In [62]:
```
df.duplicated().sum()
```

Out[62]:
```
0
```

In [63]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 12078 entries, 0 to 12201
Data columns (total 27 columns):
 #   Column                              Non-Null Count  Dtype  
---  ------                              --------------  -----  
 0   Sub Category Name                   12078 non-null  float64
 1   Location                            12078 non-null  float64
 2   Freelancer Preferred From           12078 non-null  float64
 3   Client City                         12078 non-null  float64
 4   Client Country                      12078 non-null  float64
 5   Client Currency                     12078 non-null  float64
 6   Budget_usd                          12077 non-null  float64
 7   Client Registration date            12078 non-null  float64
 8   Client Registration Month           12078 non-null  float64
 9   Client Registration Year            12078 non-null  float64
 10  Date Posted in Date                 12078 non-null  float64
 11  Date Posted in Month                12078 non-null  float64
 12  Date Posted in Time                 12078 non-null  float64
 13  Type_fixed_price                    12078 non-null  float64
 14  Type_hourly                         12078 non-null  float64
 15  Category_Business                   12078 non-null  float64
 16  Category_Design                     12078 non-null  float64
 17  Category_Digital Marketing          12078 non-null  float64
 18  Category_Marketing, Branding & Sales  12078 non-null  float64
 19  Category_Music & Audio              12078 non-null  float64
 20  Category_Social Media               12078 non-null  float64
 21  Category_Technology & Programming   12078 non-null  float64
 22  Category_Video, Photo & Image       12078 non-null  float64
 23  Category_Writing & Translation      12078 non-null  float64
 24  Experience_Entry                    12078 non-null  float64
 25  Experience_Expert                   12078 non-null  float64
 26  Experience_Intermediate             12078 non-null  float64
dtypes: float64(27)
memory usage: 2.6 MB
```

In [64]:
```
df.dropna(axis=0,inplace=True)
```

In [65]:
```
df.shape
```

Out[65]:
```
(12077, 27)
```

Split the Data

In [66]:
```
#split the data into x and y

x = df.drop(columns=['Budget_usd'], axis=1)
y = df['Budget_usd']
```

In [67]:
```
x.head()
```

Out[67]:

	Sub Category Name	Location	Freelancer Preferred From	Client City	Client Country	Client Currency	Client Registration date	Client Registration Month	Client Registration Year	Date Posted in Date	...	Category_Digital Marketing	Category_Marketing, Branding & Sales	Category_Music & Audio	Category_Social Media	Category_Technology & Programming	Categor Photc
0	0.396226	0.5	0.02439	0.271516	0.451852	0.0	0.066667	0.909091	0.1875	0.933333	...	0.0	0.0	0.0	0.0	0.0	
1	0.424528	0.5	0.02439	0.520822	0.955556	0.5	0.666667	0.090909	0.6250	0.933333	...	0.0	0.0	0.0	0.0	0.0	
2	0.349057	0.5	0.02439	0.520822	0.955556	0.5	0.266667	0.272727	1.0000	0.933333	...	0.0	0.0	0.0	0.0	0.0	
3	0.849057	0.5	0.02439	0.520822	0.955556	0.5	0.266667	0.272727	1.0000	0.933333	...	0.0	0.0	0.0	0.0	0.0	
5	0.245283	0.5	0.02439	0.270961	0.948148	0.0	0.433333	0.727273	0.3750	0.933333	...	0.0	0.0	0.0	0.0	1.0	

5 rows × 26 columns

In [68]:
```
y.sample(10)
```

Out[68]:
```
9140     0.676293
348      0.673438
5040     0.533931
9052     0.399656
4702     0.721057
8604     0.769545
3795     0.726107
8605     0.418093
11864    0.673438
1323     0.706129
Name: Budget_usd, dtype: float64
```

In [69]:
```
#split the data into training 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=101)
```

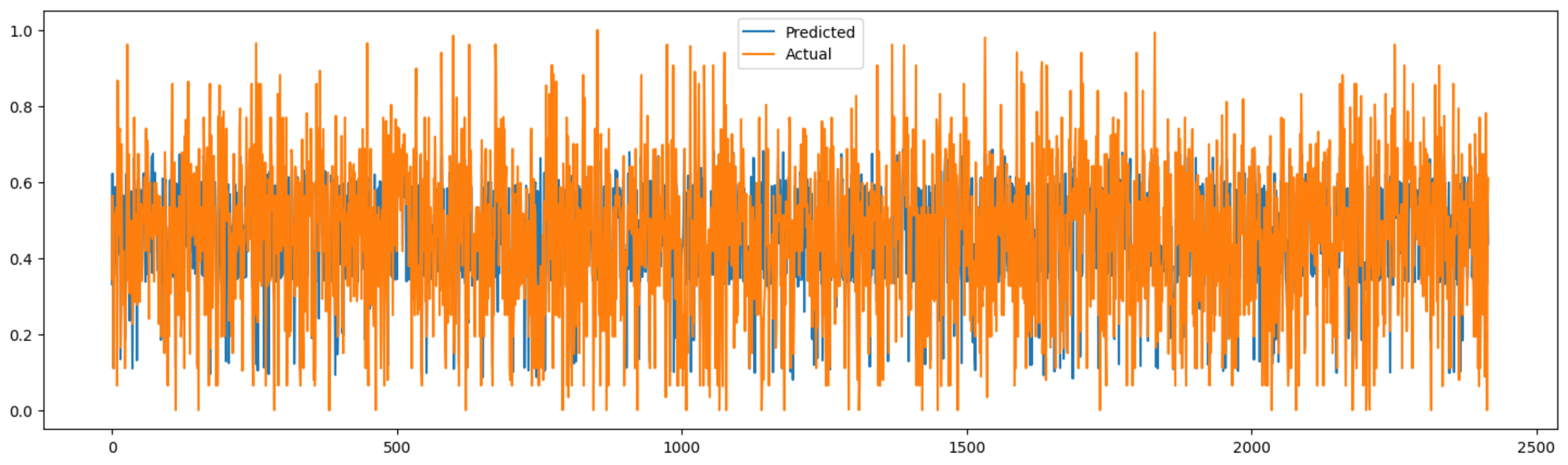
Creating a Regression model to Predict the Budget

Linear Regression

In [70]:
```
#create a model
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.linear_model import LinearRegression

#Train the linear regression model
model = LinearRegression()
model.fit(xtrain, ytrain)

#Making predictions on training and testing
trainpred = model.predict(xtrain)
testpred = model.predict(xtest)

#Evaluating the model
mse_train = mean_squared_error(ytrain, trainpred)
mse_test = mean_squared_error(ytest, testpred)
r2_train = r2_score(ytrain, trainpred)
r2_test = r2_score(ytest, testpred)

print('Mean square error for training data:', mse_train)
print('Mean square error for testing data:', mse_test)
print('R-squared for training data:', r2_train)
print('R-squared for testing data:', r2_test)
```

```
Mean square error for training data: 0.019339265128149485
Mean square error for testing data: 0.021120424237256904
R-squared for training data: 0.4921361136131418
R-squared for testing data: 0.46325444446952047
```

In [71]:
```
# plotting results from above model.

plt.figure(figsize=(18,5))
plt.plot((testpred))
plt.plot(np.array((ytest)))
plt.legend(["Predicted","Actual"])
plt.show()
```



In [72]:
```
from sklearn.linear_model import Lasso

model = Lasso(alpha=8)
model.fit(xtrain, ytrain)

trainpred = model.predict(xtrain)
testpred = model.predict(xtest)


mse_train = mean_squared_error(ytrain, trainpred)
mse_test = mean_squared_error(ytest, testpred)
r2_train = r2_score(ytrain, trainpred)
r2_test = r2_score(ytest, testpred)

print('Mean square error for training data:', mse_train)
print('Mean square error for testing data:', mse_test)
print('R-squared for training data:', r2_train)
print('R-squared for testing data:', r2_test)
```

```
Mean square error for training data: 0.03807962260466394
Mean square error for testing data: 0.03935647651650577
R-squared for training data: 0.0
R-squared for testing data: -0.00018889839865154912
```

In [73]:
```
from sklearn.linear_model import Ridge
from sklearn.metrics import mean_squared_error, r2_score

# Define the Ridge model
model = Ridge(alpha=5)
model.fit(xtrain, ytrain)

# Use the best model to make predictions
trainpred = model.predict(xtrain)
testpred = model.predict(xtest)

# Calculate evaluation metrics
mse_train = mean_squared_error(ytrain, trainpred)
mse_test = mean_squared_error(ytest, testpred)
r2_train = r2_score(ytrain, trainpred)
r2_test = r2_score(ytest, testpred)

# Print the evaluation metrics
print('Mean square error for training data:', mse_train)
print('Mean square error for testing data:', mse_test)
print('R-squared for training data:', r2_train)
print('R-squared for testing data:', r2_test)
```

```
Mean square error for training data: 0.019335261146377847
Mean square error for testing data: 0.021108454498389058
R-squared for training data: 0.4922412612353545
R-squared for testing data: 0.46355863836572275
```

In [74]:
```
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import mean_squared_error, r2_score

# Create polynomial features
degree = 3  # Set the degree of the polynomial
poly_features = PolynomialFeatures(degree=degree)
X_train_poly = poly_features.fit_transform(xtrain)
X_test_poly = poly_features.transform(xtest)

# Train the polynomial regression model
model = LinearRegression()
model.fit(X_train_poly, ytrain)

# Make predictions on training and testing data
trainpred = model.predict(X_train_poly)
testpred = model.predict(X_test_poly)

# Evaluate the model
mse_train = mean_squared_error(ytrain, trainpred)
mse_test = mean_squared_error(ytest, testpred)
r2_train = r2_score(ytrain, trainpred)
r2_test = r2_score(ytest, testpred)

print('Mean square error for training data:', mse_train)
print('Mean square error for testing data:', mse_test)
print('R-squared for training data:', r2_train)
print('R-squared for testing data:', r2_test)
```

```
Mean square error for training data: 0.013137340958815613
Mean square error for testing data: 1.0635995047403071e+18
R-squared for training data: 0.655003383431469
R-squared for testing data: -2.702986931610634e+19
```

Decision Tree

In [75]:
```
from sklearn.tree import DecisionTreeRegressor

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

Out[75]:
```
DecisionTreeRegressor()
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [76]:
```
#create a model
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error, r2_score

#Train the model
model = DecisionTreeRegressor(max_depth=7)
model.fit(xtrain, ytrain)

#Making predictions on training and testing
trainpred = model.predict(xtrain)
testpred = model.predict(xtest)

#Evaluating the model
mse_train = mean_squared_error(ytrain, trainpred)
mse_test = mean_squared_error(ytest, testpred)
r2_train = r2_score(ytrain, trainpred)
r2_test = r2_score(ytest, testpred)

print('Mean square error for training data:', mse_train)
print('Mean square error for testing data:', mse_test)
print('R-squared for training data:', r2_train)
print('R-squared for testing data:', r2_test)
```

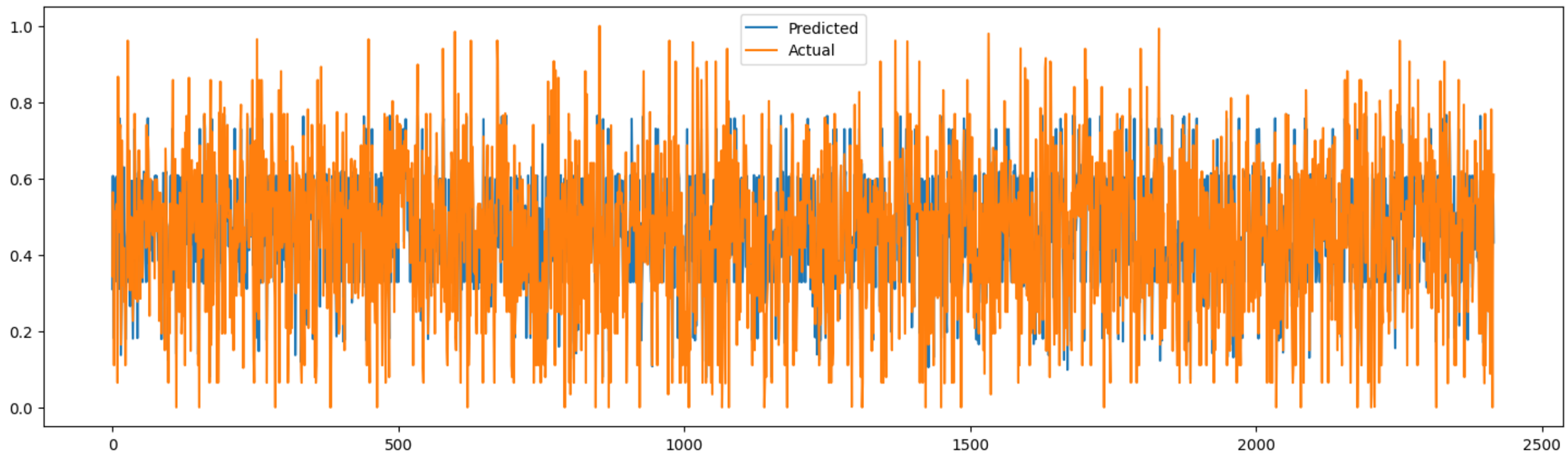
```
Mean square error for training data: 0.015049883994525235
Mean square error for testing data: 0.01759848588776839
R-squared for training data: 0.6047785412484118
R-squared for testing data: 0.5527595005566814
```

Random Forest

In [78]:
```
#create a model
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score

#Train the model
#rf = RandomForestRegressor()
rf = RandomForestRegressor(n_estimators=75, max_depth=6, min_samples_split=6, max_features='auto',bootstrap=True, min_samples_leaf=6)
rf.fit(xtrain, ytrain)

#Making predictions on training and testing
trainpred = rf.predict(xtrain)
testpred = rf.predict(xtest)

#Evaluating the model
mse_train = mean_squared_error(ytrain, trainpred)
mse_test = mean_squared_error(ytest, testpred)
r2_train = r2_score(ytrain, trainpred)
r2_test = r2_score(ytest, testpred)

print('Mean square error for training data:', mse_train)
print('Mean square error for testing data:', mse_test)
print('R-squared for training data:', r2_train)
print('R-squared for testing data:', r2_test)
```

```
C:\Python311\Lib\site-packages\sklearn\ensemble\_forest.py:413: FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the past beha
viour, explicitly set `max_features=1.0` or remove this parameter as it is also the default value for RandomForestRegressors and ExtraTreesRegressors.
  warn(

Mean square error for training data: 0.015505790850176247
Mean square error for testing data: 0.01715896802594977
R-squared for training data: 0.592806078695824
R-squared for testing data: 0.5639292221615748
```

In [79]:
```
# plotting results from above model.

plt.figure(figsize=(18,5))
plt.plot((testpred))
plt.plot(np.array((ytest)))
plt.legend(["Predicted","Actual"])
plt.show()
```



Support Vector Machine

In [80]:
```
from sklearn.svm import SVR
from sklearn.metrics import mean_squared_error, r2_score


model = SVR(kernel ='rbf')
model.fit(xtrain, ytrain)

trainpred = model.predict(xtrain)
testpred = model.predict(xtest)

mse_train = mean_squared_error(ytrain, trainpred)
mse_test = mean_squared_error(ytest, testpred)
r2_train = r2_score(ytrain, trainpred)
r2_test = r2_score(ytest, testpred)

print('Mean square error for training data:', mse_train)
print('Mean square error for testing data:', mse_test)
print('R-squared for training data:', r2_train)
print('R-squared for testing data:', r2_test)
```

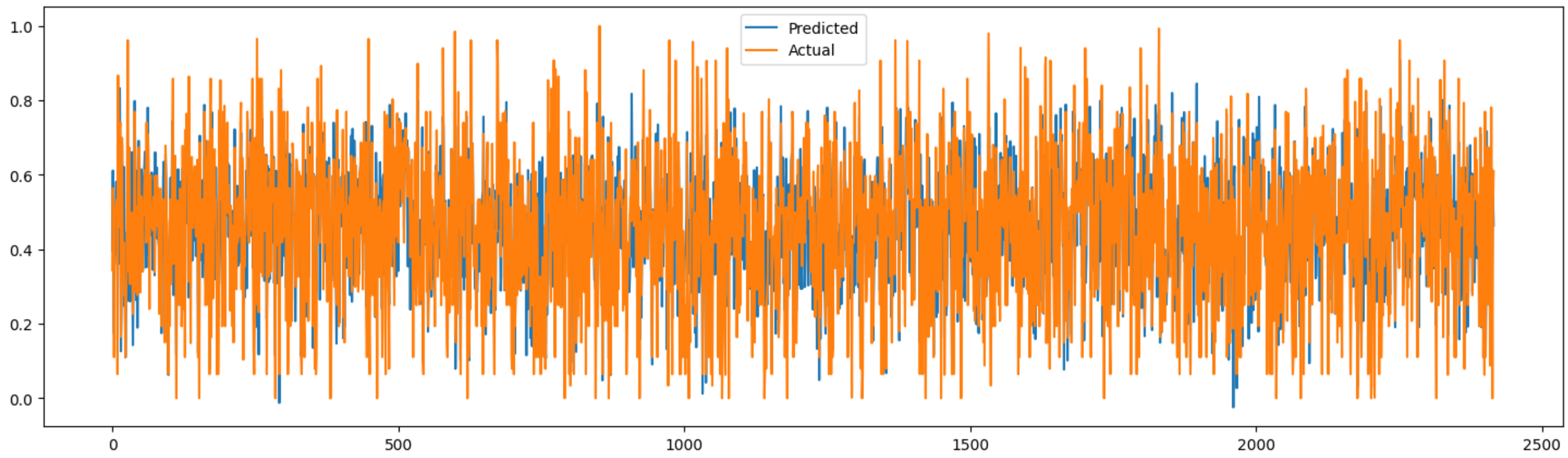
```
Mean square error for training data: 0.014394818308011387
Mean square error for testing data: 0.018247759865555865
R-squared for training data: 0.6219810669486958
R-squared for testing data: 0.5362591254702629
```

Xgboost ¶

In [94]:
```
import xgboost as xgb
from sklearn.metrics import mean_squared_error, r2_score

# Define the XGBoost regressor model
xgb_model = xgb.XGBRegressor()

# Train the model
xgb_model.fit(xtrain, ytrain)

# Make predictions on the test set
trainpred = xgb_model.predict(xtrain)
testpred = xgb_model.predict(xtest)

# Evaluate the model
mse_train = mean_squared_error(ytrain, trainpred)
mse_test = mean_squared_error(ytest, testpred)
r2_train = r2_score(ytrain, trainpred)
r2_test = r2_score(ytest, testpred)

print('Mean square error for training data:', mse_train)
print('Mean square error for testing data:', mse_test)
print('R-squared for training data:', r2_train)
print('R-squared for testing data:', r2_test)
```

```
Mean square error for training data: 0.006190741212273089
Mean square error for testing data: 0.01704348357945881
R-squared for training data: 0.837426403183028
R-squared for testing data: 0.5668640951873533
```

In [95]:
```
# plotting results from above model.

plt.figure(figsize=(18,5))
plt.plot((testpred))
plt.plot(np.array((ytest)))
plt.legend(["Predicted","Actual"])
plt.show()
```



Result:-

Based on the above results, the XGBOOST model performs the best among the models evaluated. It achieves the lowest mean square error (MSE) for the testing data, indicating superior predictive accuracy compared to the other models. Additionally, it demonstrates a relatively high R-squared value for the testing data, further supporting its strong performance.

The XGBOOST model outperforms the Random Forest, Decision Tree, SVM, Ridge, Linear Regression, and Lasso models in terms of MSE for the testing data. Its superior predictive capabilities make it a promising choice for this particular task or dataset.