Freelance Platform Project - Unsupervised Machine Learning

Type:-Kmeans Clustering

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
```

```
<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 and currency

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

In [16]:
```
#Checking the shape after dropping 2 columns

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

```
The shape of dataset is:- (12201, 14)
```

In [17]:
```
#Exploring the statistical information

df.describe()
```

Out[17]:

	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

In [18]:
```
# #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 [19]:
```
#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 [20]:
```
#Dropping date posted column as we already have done feature extraction

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

In [21]:
```
#Checking the shape after dropping Date Posted column

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

```
The shape of dataset is:- (12201, 19)
```

Checking skewness

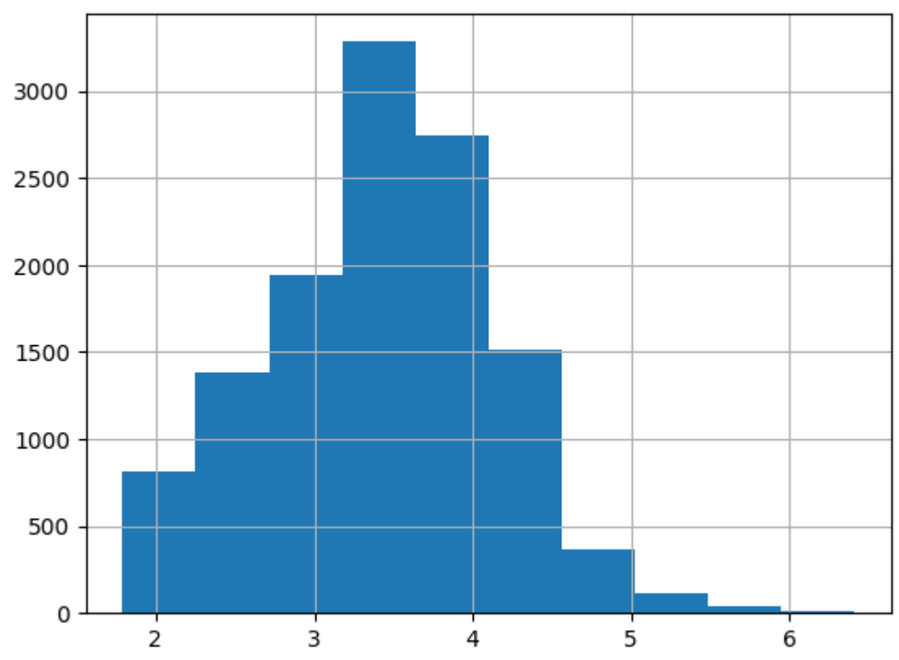
In [22]:
```
df['Budget_usd'].skew()
```

Out[22]:
```
43.9364193416511
```

In [23]:
```
#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[23]:
```
-0.002291121221937113
```

In [24]:
```
#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 [25]:
```
df.info()
```

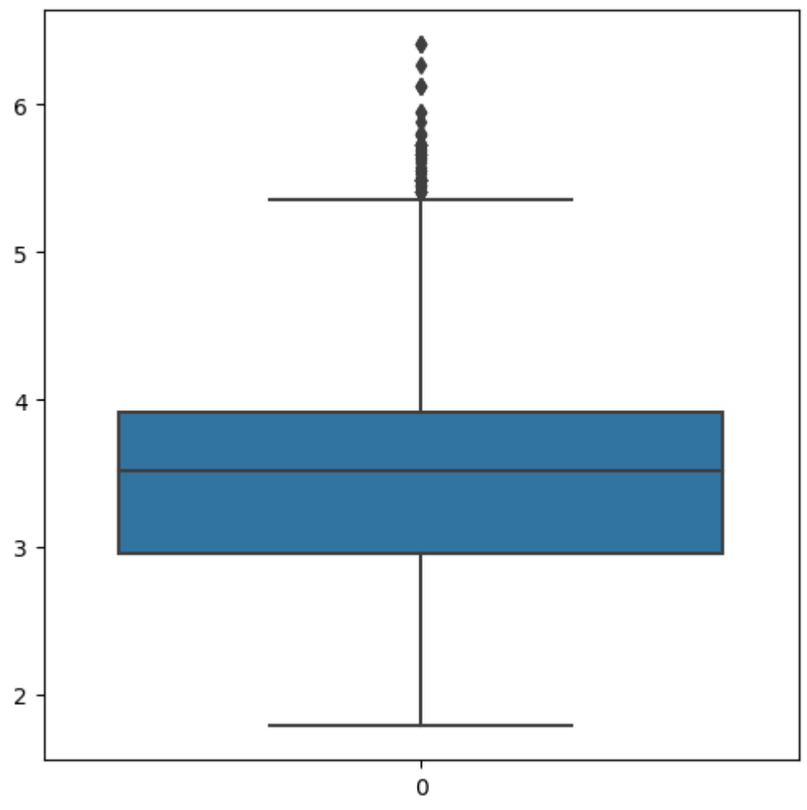
```
<class 'pandas.core.frame.DataFrame'>
Index: 12201 entries, 0 to 12201
Data columns (total 19 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   Title                      12201 non-null  object
 1   Category Name              12201 non-null  object
 2   Experience                 12201 non-null  object
 3   Sub Category Name          12201 non-null  object
 4   Location                   12201 non-null  object
 5   Freelancer Preferred From  12201 non-null  object
 6   Type                       12201 non-null  object
 7   Description                12201 non-null  object
 8   Client City                12201 non-null  object
 9   Client Country             12201 non-null  object
 10  Client Currency            12201 non-null  object
 11  Budget_usd                 12200 non-null  float64
 12  Client Registration date   12201 non-null  int32
 13  Client Registration Month  12201 non-null  int32
 14  Client Registration Year   12201 non-null  int32
 15  Date Posted in Date        12201 non-null  int32
 16  Date Posted in Month       12201 non-null  int32
 17  Date Posted in Year        12201 non-null  int32
 18  Date Posted in Time        12201 non-null  object
dtypes: float64(1), int32(6), object(12)
memory usage: 1.6+ MB
```

In [26]:
```
#Budget_usd column has null value in one row, so dropping

df.dropna(axis = 0, inplace = True)
```

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: >



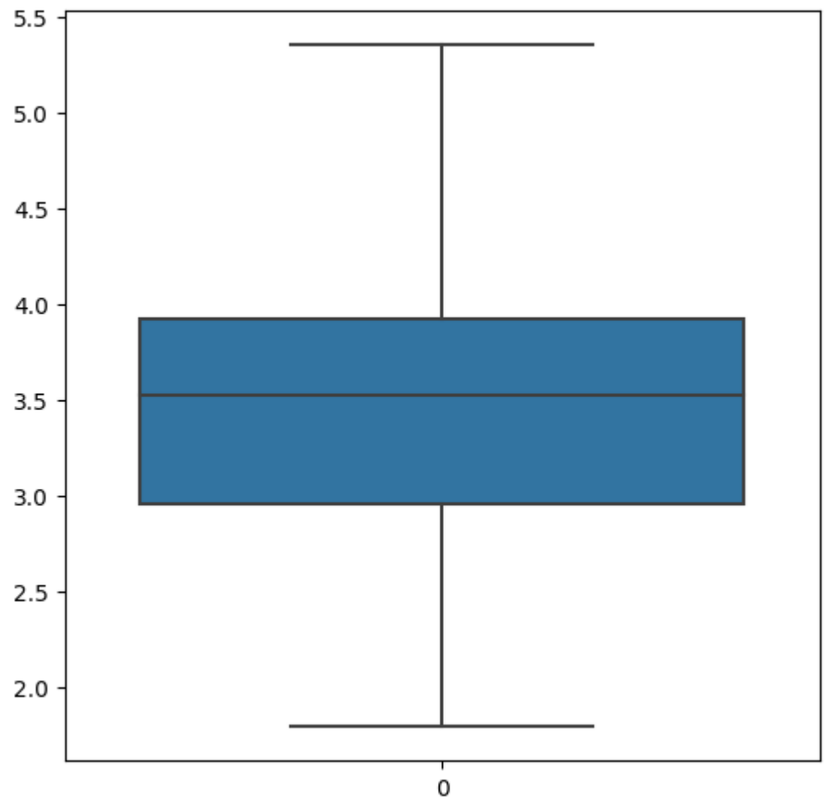
In [28]:
```
#Counting the values that are greater then 5.4

# 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: >



Categorical data encoding

In [31]:
```
import pandas as pd

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

# Drop the original 'Experience', 'Category Name' columns from the DataFrame
df = df.drop(['Category Name','Experience','Location','Freelancer Preferred From','Type'], 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

                                   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  Client Registration date
0             Ireland             EUR    3.301078                         3  \
1       United Kingdom             GBP    2.680607                        21
2       United Kingdom             GBP    2.318016                         9
3       United Kingdom             GBP    2.429707                         9
5  United Arab Emirates             EUR    4.460790                        14

   Client Registration Month  Client Registration Year  ...
0                         11                      2010  ...  \
1                          2                      2017  ...
2                          4                      2023  ...
3                          4                      2023  ...
5                          9                      2013  ...

   Freelancer Preferred From_RU  Freelancer Preferred From_SE
0                             0                             0  \
1                             0                             0
2                             0                             0
3                             0                             0
5                             0                             0

   Freelancer Preferred From_TH Freelancer Preferred From_TR
0                             0                             0  \
1                             0                             0
2                             0                             0
3                             0                             0
5                             0                             0

   Freelancer Preferred From_TW  Freelancer Preferred From_UG
0                             0                             0  \
1                             0                             0
2                             0                             0
3                             0                             0
5                             0                             0

   Freelancer Preferred From_US  Freelancer Preferred From_ZA
0                             0                             0  \
1                             0                             0
2                             0                             0
3                             0                             0
5                             0                             0

   Type_fixed_price  Type_hourly
0                 1            0
1                 1            0
2                 1            0
3                 1            0
5                 1            0

[5 rows x 73 columns]
```

In [32]:
```
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])
```

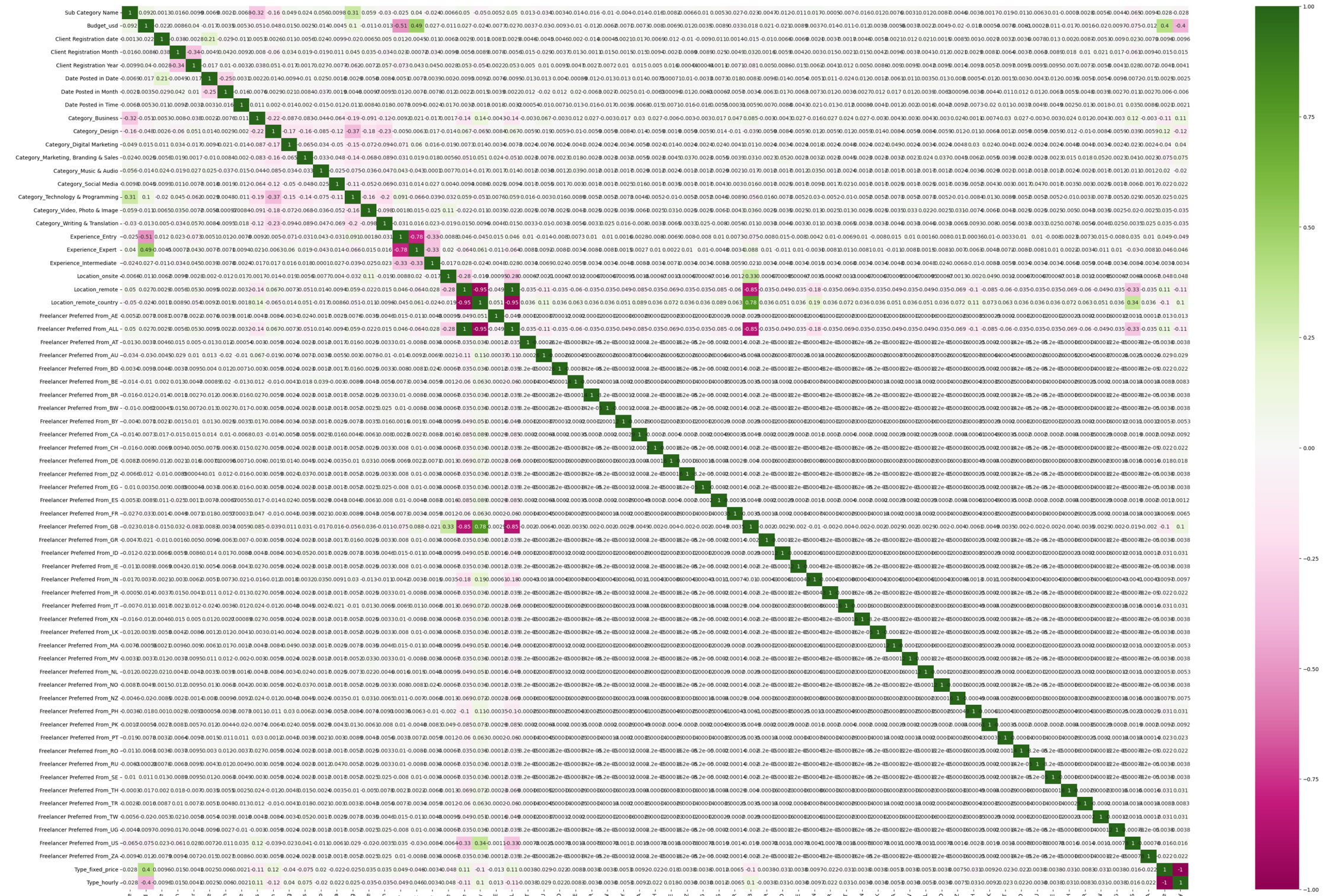
In [33]:
```
df['Experience_Entry '].value_counts()
```

Out[33]:
```
Experience_Entry
0    6823
1    5314
Name: count, dtype: int64
```

Scaling the Data

In [34]:
```
from sklearn.preprocessing import StandardScaler

# Assuming you have a DataFrame named 'df' with the desired column(s) to be scaled

# Initialize the StandardScaler
scaler = StandardScaler()

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

# Apply Standard scaling to the selected column(s)
df[columns_to_scale] = scaler.fit_transform(df[columns_to_scale])
```

In [53]:
```
# Checking correlation using heatmap

plt.figure(figsize=(45,30))
sns.heatmap(df.corr(),cmap='PiYG',annot=True);
```



In [36]:
```
#Droping the specified columns from the DataFrame permanently

columns_to_drop = ['Title', 'Description', 'Client City','Client Country','Client Currency','Date Posted in Year']
df.drop(columns_to_drop, axis=1, inplace=True)
```

In [37]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 12137 entries, 0 to 12200
Data columns (total 67 columns):
 #   Column                                Non-Null Count  Dtype  
---  ------                                --------------  -----  
 0   Sub Category Name                     12137 non-null  float64
 1   Budget_usd                            12137 non-null  float64
 2   Client Registration date              12137 non-null  float64
 3   Client Registration Month             12137 non-null  float64
 4   Client Registration Year              12137 non-null  float64
 5   Date Posted in Date                   12137 non-null  float64
 6   Date Posted in Month                  12137 non-null  float64
 7   Date Posted in Time                   12137 non-null  float64
 8   Category_Business                     12137 non-null  float64
 9   Category_Design                       12137 non-null  float64
 10  Category_Digital Marketing            12137 non-null  float64
 11  Category_Marketing, Branding & Sales  12137 non-null  float64
 12  Category_Music & Audio                12137 non-null  float64
 13  Category_Social Media                 12137 non-null  float64
 14  Category_Technology & Programming     12137 non-null  float64
 15  Category_Video, Photo & Image         12137 non-null  float64
 16  Category_Writing & Translation        12137 non-null  float64
 17  Experience_Entry                      12137 non-null  float64
 18  Experience_Expert                     12137 non-null  float64
 19  Experience_Intermediate               12137 non-null  float64
 20  Location_onsite                       12137 non-null  float64
 21  Location_remote                       12137 non-null  float64
 22  Location_remote_country               12137 non-null  float64
 23  Freelancer Preferred From_AE          12137 non-null  float64
 24  Freelancer Preferred From_ALL         12137 non-null  float64
 25  Freelancer Preferred From_AT          12137 non-null  float64
 26  Freelancer Preferred From_AU          12137 non-null  float64
 27  Freelancer Preferred From_BD          12137 non-null  float64
 28  Freelancer Preferred From_BE          12137 non-null  float64
 29  Freelancer Preferred From_BR          12137 non-null  float64
 30  Freelancer Preferred From_BW          12137 non-null  float64
 31  Freelancer Preferred From_BY          12137 non-null  float64
 32  Freelancer Preferred From_CA          12137 non-null  float64
 33  Freelancer Preferred From_CH          12137 non-null  float64
 34  Freelancer Preferred From_DE          12137 non-null  float64
 35  Freelancer Preferred From_DZ          12137 non-null  float64
 36  Freelancer Preferred From_EG          12137 non-null  float64
 37  Freelancer Preferred From_ES          12137 non-null  float64
 38  Freelancer Preferred From_FR          12137 non-null  float64
 39  Freelancer Preferred From_GB          12137 non-null  float64
 40  Freelancer Preferred From_GR          12137 non-null  float64
 41  Freelancer Preferred From_ID          12137 non-null  float64
 42  Freelancer Preferred From_IE          12137 non-null  float64
 43  Freelancer Preferred From_IN          12137 non-null  float64
 44  Freelancer Preferred From_IR          12137 non-null  float64
 45  Freelancer Preferred From_IT          12137 non-null  float64
 46  Freelancer Preferred From_KN          12137 non-null  float64
 47  Freelancer Preferred From_LK          12137 non-null  float64
 48  Freelancer Preferred From_MA          12137 non-null  float64
 49  Freelancer Preferred From_MV          12137 non-null  float64
 50  Freelancer Preferred From_NL          12137 non-null  float64
 51  Freelancer Preferred From_NO          12137 non-null  float64
 52  Freelancer Preferred From_NZ          12137 non-null  float64
 53  Freelancer Preferred From_PH          12137 non-null  float64
 54  Freelancer Preferred From_PK          12137 non-null  float64
 55  Freelancer Preferred From_PT          12137 non-null  float64
 56  Freelancer Preferred From_RO          12137 non-null  float64
 57  Freelancer Preferred From_RU          12137 non-null  float64
 58  Freelancer Preferred From_SE          12137 non-null  float64
 59  Freelancer Preferred From_TH          12137 non-null  float64
 60  Freelancer Preferred From_TR          12137 non-null  float64
 61  Freelancer Preferred From_TW          12137 non-null  float64
 62  Freelancer Preferred From_UG          12137 non-null  float64
 63  Freelancer Preferred From_US          12137 non-null  float64
 64  Freelancer Preferred From_ZA          12137 non-null  float64
 65  Type_fixed_price                      12137 non-null  float64
 66  Type_hourly                           12137 non-null  float64
dtypes: float64(67)
memory usage: 6.5 MB
```

In [38]:
```
df.columns
```

Out[38]:
```
Index(['Sub Category Name', 'Budget_usd', 'Client Registration date',
       'Client Registration Month', 'Client Registration Year',
       'Date Posted in Date', 'Date Posted in Month', 'Date Posted in Time',
       '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 ',
       'Location_onsite', 'Location_remote', 'Location_remote_country',
       'Freelancer Preferred From_AE', 'Freelancer Preferred From_ALL',
       'Freelancer Preferred From_AT', 'Freelancer Preferred From_AU',
       'Freelancer Preferred From_BD', 'Freelancer Preferred From_BE',
       'Freelancer Preferred From_BR', 'Freelancer Preferred From_BW',
       'Freelancer Preferred From_BY', 'Freelancer Preferred From_CA',
       'Freelancer Preferred From_CH', 'Freelancer Preferred From_DE',
       'Freelancer Preferred From_DZ', 'Freelancer Preferred From_EG',
       'Freelancer Preferred From_ES', 'Freelancer Preferred From_FR',
       'Freelancer Preferred From_GB', 'Freelancer Preferred From_GR',
       'Freelancer Preferred From_ID', 'Freelancer Preferred From_IE',
       'Freelancer Preferred From_IN', 'Freelancer Preferred From_IR',
       'Freelancer Preferred From_IT', 'Freelancer Preferred From_KN',
       'Freelancer Preferred From_LK', 'Freelancer Preferred From_MA',
       'Freelancer Preferred From_MV', 'Freelancer Preferred From_NL',
       'Freelancer Preferred From_NO', 'Freelancer Preferred From_NZ',
       'Freelancer Preferred From_PH', 'Freelancer Preferred From_PK',
       'Freelancer Preferred From_PT', 'Freelancer Preferred From_RO',
       'Freelancer Preferred From_RU', 'Freelancer Preferred From_SE',
       'Freelancer Preferred From_TH', 'Freelancer Preferred From_TR',
       'Freelancer Preferred From_TW', 'Freelancer Preferred From_UG',
       'Freelancer Preferred From_US', 'Freelancer Preferred From_ZA',
       'Type_fixed_price', 'Type_hourly'],
      dtype='object')
```

K-means Clustering: Determining Optimal Number of Clusters

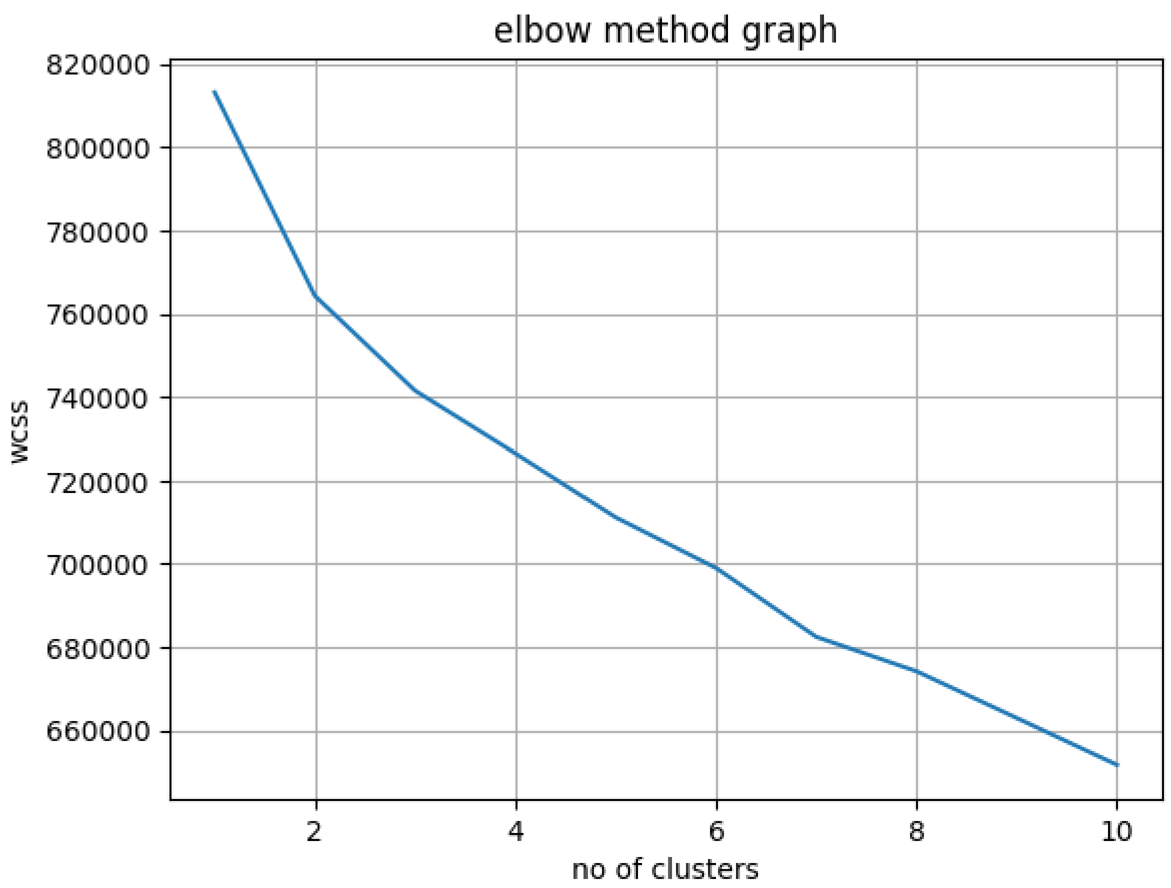
In [39]:
```
from sklearn.cluster import KMeans

wcss_list = []

for i in range(1, 11):
    kmeans = KMeans(n_clusters=i)
    kmeans.fit(df)
    wcss = kmeans.inertia_
    wcss_list.append(wcss)
```

```
C:\Python311\Lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
C:\Python311\Lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
C:\Python311\Lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
C:\Python311\Lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
C:\Python311\Lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
C:\Python311\Lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
C:\Python311\Lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
C:\Python311\Lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
C:\Python311\Lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
C:\Python311\Lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
```

In [40]:
```
wcss_list
```

Out[40]:
```
[813179.0000000003,
 764290.2719905726,
 741511.4773334587,
 726516.575851331,
 711143.5019232366,
 699005.6234971472,
 682449.5249478549,
 674191.4510576604,
 662960.0879043854,
 651706.0519737175]
```

In [41]:
```
#Elbow method to identify number of clusters

import matplotlib.pyplot as plt

plt.plot(range(1,11), wcss_list)
plt.title('elbow method graph')
plt.xlabel('no of clusters')
plt.ylabel('wcss')
plt.grid()
plt.show()
```



In [54]:
```
#Assigning data points to clusters

kmeans = KMeans(n_clusters=7)
kmeans.fit(df)
pred = kmeans.predict(df)
pred[:11]
```

```
C:\Python311\Lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
```

Out[54]:
```
array([3, 3, 3, 3, 0, 3, 3, 3, 3, 5, 5])
```

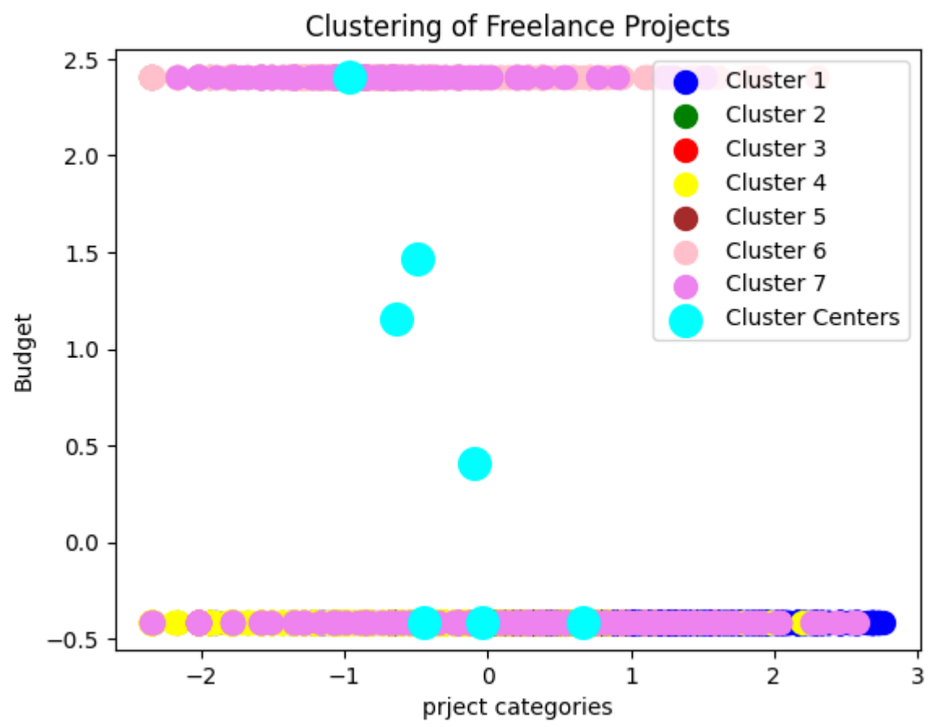
In [55]:
```
pd.Series(pred).value_counts()
```

Out[55]:
```
0    5356
3    4445
5    1553
6     769
1       9
4       3
2       2
Name: count, dtype: int64
```

In [56]:
```
kmeans.cluster_centers_
```

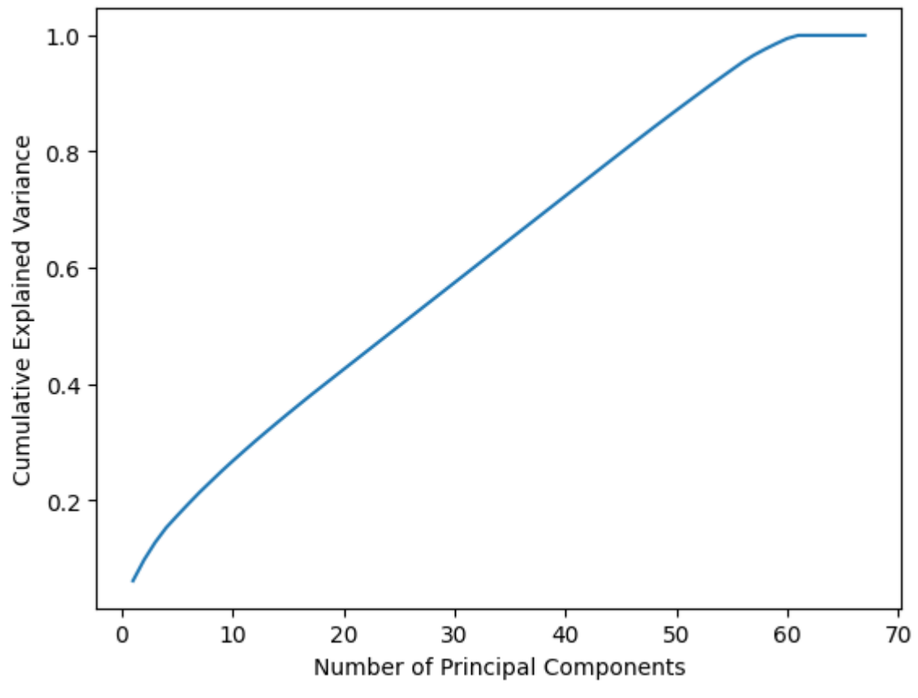
```
array([[-2.55267169e-01,  6.67449018e-03,  1.99958457e-01,
        -5.03439262e-02,  3.99940760e-02, -2.20684378e-01,
         9.07084488e-02, -5.98603872e-02, -1.80057430e-01,
         2.46951746e-01, -1.01756514e-01,  1.09309540e+00,
        -3.80797040e+00,  3.62689487e+00,  1.89781045e-01,
        -3.80797040e+00,  1.34189935e-01,  4.24503266e-01,
         1.34189935e-01,  2.32442939e-01,  1.34189935e-01,
         1.34189935e-01,  1.89781045e-01,  3.28764600e-01,
         1.34189935e-01,  2.68413047e-01,  1.34189935e-01,
         1.34189935e-01,  3.28764600e-01,  2.32442939e-01,
         3.28461283e+00,  1.34189935e-01,  1.89781045e-01,
         1.34189935e-01,  6.98019468e-01,  1.34189935e-01,
         2.68413047e-01,  1.34189935e-01,  1.34189935e-01,
        -1.28379331e-02,  1.34189935e-01,  1.89781045e-01,
         1.34189935e-01,  2.68413047e-01, -2.72412267e-02,
         3.28764600e-01, -1.57238406e-02,  1.34189935e-01,
         1.34189935e-01,  1.34189935e-01,  2.68413047e-01,
         2.32442939e-01,  1.89781045e-01,  1.34189935e-01,
         1.27056021e+00,  1.34189935e-01, -4.10859520e-01,
         4.10859520e-01]])
```

In [58]:
```
plt.scatter(df.iloc[pred==0, 1], df.iloc[pred==0, -1], s=100, c='blue', label='Cluster 1')
plt.scatter(df.iloc[pred==1, 1], df.iloc[pred==1, -1], s=100, c='green', label='Cluster 2')
plt.scatter(df.iloc[pred==2, 1], df.iloc[pred==2, -1], s=100, c='red', label='Cluster 3')
plt.scatter(df.iloc[pred==3, 1], df.iloc[pred==3, -1], s=100, c='yellow', label='Cluster 4')
plt.scatter(df.iloc[pred==4, 1], df.iloc[pred==4, -1], s=100, c='brown', label='Cluster 5')
plt.scatter(df.iloc[pred==5, 1], df.iloc[pred==5, -1], s=100, c='pink', label='Cluster 6')
plt.scatter(df.iloc[pred==6, 1], df.iloc[pred==6, -1], s=100, c='violet', label='Cluster 7')
# plt.scatter(df.iloc[pred==7, 1], df.iloc[pred==7, -1], s=100, c='magenta', label='Cluster 8')

plt.scatter(kmeans.cluster_centers_[:, 1], kmeans.cluster_centers_[:, -1], s=200, c='cyan', label='Cluster Centers')
plt.title('Clustering of Freelance Projects')
plt.xlabel('prject categories')
plt.ylabel('Budget')
plt.legend(loc='upper right')
plt.show()
```



Clustering using PCA and k-means

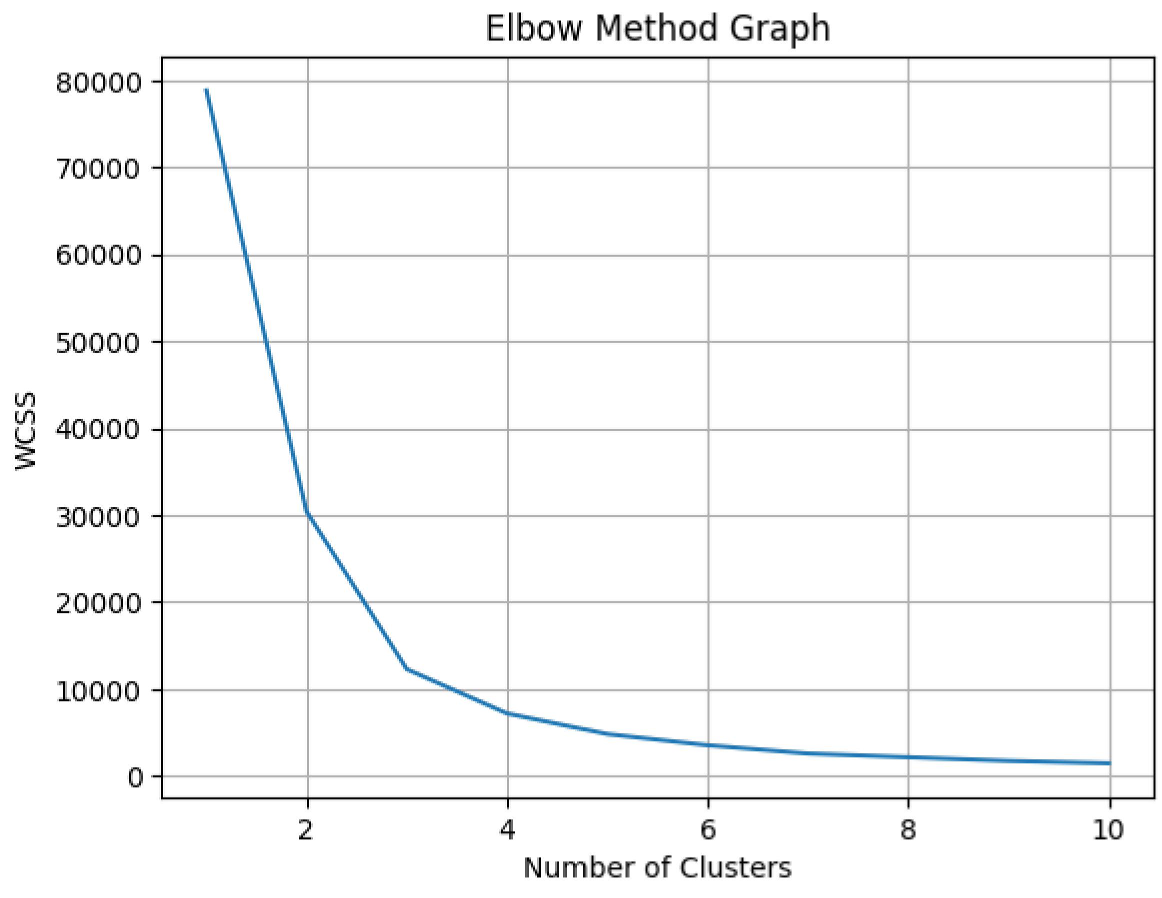
In [46]:
```
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
import numpy as np

# Perform PCA on your data
pca = PCA()
pca.fit(df)

# Obtain the explained variance ratio for each principal component
explained_variance_ratio = pca.explained_variance_ratio_

# Calculate the cumulative explained variance
cumulative_variance = np.cumsum(explained_variance_ratio)

# Plot the cumulative explained variance
plt.plot(range(1, len(cumulative_variance) + 1), cumulative_variance)
plt.xlabel('Number of Principal Components')
plt.ylabel('Cumulative Explained Variance')
plt.show()
```



In [47]:
```
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

# Apply PCA
pca = PCA(n_components=2)
pca_components = pca.fit_transform(df)

wcss_list = []
```

In [48]:
```
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i)
    kmeans.fit(pca_components)
    wcss = kmeans.inertia_
    wcss_list.append(wcss)

plt.plot(range(1, 11), wcss_list)
plt.title('Elbow Method Graph')
plt.xlabel('Number of Clusters')
plt.ylabel('WCSS')
plt.grid()
plt.show()
```

```
C:\Python311\Lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` expli
citly to suppress the warning
  warnings.warn(
C:\Python311\Lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` expli
citly to suppress the warning
  warnings.warn(
C:\Python311\Lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` expli
citly to suppress the warning
  warnings.warn(
C:\Python311\Lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` expli
citly to suppress the warning
  warnings.warn(
C:\Python311\Lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` expli
citly to suppress the warning
  warnings.warn(
C:\Python311\Lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` expli
citly to suppress the warning
  warnings.warn(
C:\Python311\Lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` expli
citly to suppress the warning
  warnings.warn(
C:\Python311\Lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` expli
citly to suppress the warning
  warnings.warn(
C:\Python311\Lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` expli
citly to suppress the warning
  warnings.warn(
C:\Python311\Lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` expli
citly to suppress the warning
  warnings.warn(
```

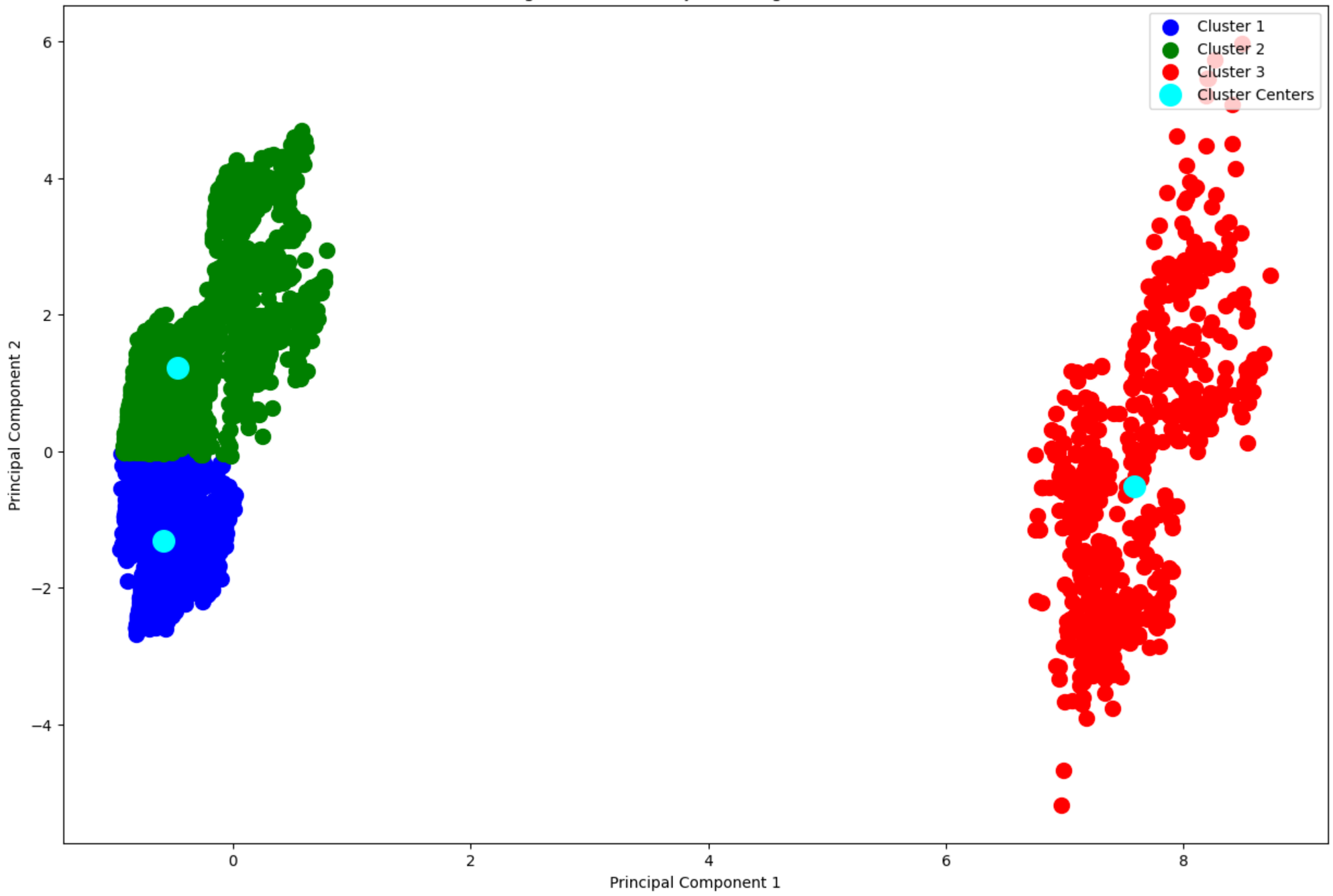


In [49]:
```
wcss_list
```

Out[49]:
```
[78842.4985733451,
 30420.028940825207,
 12286.371127568276,
 7200.2779330864305,
 4821.990510063563,
 3537.4801558811578,
 2585.1819254978705,
 2161.941781556104,
 1744.710690601301,
 1466.961006461028]
```

In [60]:
```
# Perform clustering with the optimal number of clusters using PCA components
kmeans = KMeans(n_clusters=3)
kmeans.fit(pca_components)
pred = kmeans.predict(pca_components)
plt.figure(figsize=(15,10))

# Plot the clusters
plt.scatter(pca_components[pred==0, 0], pca_components[pred==0, 1], s=100, c='blue', label='Cluster 1')
plt.scatter(pca_components[pred==1, 0], pca_components[pred==1, 1], s=100, c='green', label='Cluster 2')
plt.scatter(pca_components[pred==2, 0], pca_components[pred==2, 1], s=100, c='red', label='Cluster 3')
# plt.scatter(pca_components[pred==3, 0], pca_components[pred==3, 1], s=100, c='yellow', label='Cluster 4')

# Add scatter plot for other clusters as well

plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s=200, c='cyan', label='Cluster Centers')
plt.title('Clustering of Freelance Projects using PCA and k-means')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.legend(loc='upper right')
plt.show()
```

```
C:\Python311\Lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` expli
citly to suppress the warning
  warnings.warn(
```



In [ ]: