Analysis of Freelance Jobs on Upwork

Introduction

The landscape of work is undergoing a profound transformation with the advent of digital platforms facilitating freelance opportunities. Upwork, as one of the leading platforms in the gig economy, serves as a focal point for connecting businesses with a diverse pool of freelancers across the globe. This report aims to provide a comprehensive analysis of freelance jobs on Upwork, examining various facets such as demand trends, skill requirements, geographical distribution, and emerging opportunities. By delving into the dynamics of the Upwork ecosystem, this analysis seeks to offer valuable insights into the evolving nature of remote work, the gig economy, and the skills in demand, thereby informing stakeholders, freelancers, businesses, and policymakers alike.

About the Dataset

The dataset was obtained from - https://www.kaggle.com/datasets/ahmedmyalo/upwork-freelance-jobs-60k

Dataset Analysis

Import Libraries

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import squarify
import geopandas as gpd
import statsmodels.api as sm
```

Import dataset

In [2]:
```
df = pd.read_csv("Final_Upwork_Dataset.csv")
```

Display all column of dataset with 5 rows

In [3]:
```
df.head(5)
```

Out[3]:

	Job Title	Job_URL	EX_level_demand	Time_Limitation	Search_Keyword	Posted_from	Descriptio
0	Power bi specialist freelance	https://www.upwork.com/jobs/Power-specialist-f...	Expert	NaN	Data_science	5 minutes ago	Already dat pooled an designed. Nee to refi.
1	Case Study (on-demand delivery startup)	https://www.upwork.com/jobs/Case-Study-demand-...	Intermediate	NaN	Data_science	5 minutes ago	Hi,\n\nWould yo be able to hel me do a case-.
2	File Maker Pro Reports, Charts, Query and Ongo...	https://www.upwork.com/jobs/File-Maker-Pro-Rep...	Intermediate	3 to 6 months, Less than 30 hrs/week	Data_science	9 minutes ago	NITIA PROJECT\n\nSe up Monthl Report mimick.
3	Implementation of EleutherAI/gpt-neox-20b	https://www.upwork.com/jobs/Implementation-Ele...	Expert	3 to 6 months, Less than 30 hrs/week	Data_science	12 minutes ago	As a first step you wi implement th instal.
4	BI and Data Engineer for Upwork Finance System...	https://www.upwork.com/jobs/and-span-Data-span...	Expert	More than 6 months, 30+ hrs/week	Data_science	14 minutes ago	The Upwor Finance System team is lookin for.

5 rows × 41 columns

Display information of every column(non null values count , data type)

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 63950 entries, 0 to 63949
Data columns (total 41 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Job Title             63948 non-null  object
 1   Job_URL               63949 non-null  object
 2   EX_level_demand       56935 non-null  object
 3   Time_Limitation       23034 non-null  object
 4   Search_Keyword        63949 non-null  object
 5   Posted_from           63913 non-null  object
 6   Description           63948 non-null  object
 7   Category1_URL_search  63505 non-null  object
 8   Category_1            63505 non-null  object
 9   highlight             50131 non-null  object
 10  Category2_URL_search  59889 non-null  object
 11  Category_2            59889 non-null  object
 12  Category3_URL_search  54997 non-null  object
 13  Category_3            54997 non-null  object
 14  Category4_URL_search  47543 non-null  object
 15  Category_4            47543 non-null  object
 16  Category5_URL_search  39578 non-null  object
 17  Category_5            39578 non-null  object
 18  Category6_URL_search  31307 non-null  object
 19  Category_6            31307 non-null  object
 20  Category7_URL_search  25192 non-null  object
 21  Category_7            25192 non-null  object
 22  Category8_URL_search  20436 non-null  object
 23  Category_8            20436 non-null  object
 24  Category9_URL_search  16481 non-null  object
 25  Category_9            16481 non-null  object
 26  Applicants_Num        63949 non-null  object
 27  Payment_Situation     63949 non-null  object
 28  Enterprise_Client     53 non-null     object
 29  Freelancers_Num       63950 non-null  int64
 30  Spent($)              62304 non-null  float64
 31  Client_Country        63825 non-null  object
 32  Connects_Num          63949 non-null  float64
 33  New_Connects_Num      63949 non-null  float64
 34  Rating                30593 non-null  float64
 35  Feedback_Num          63949 non-null  float64
 36  Payment_type          63949 non-null  object
 37  Job_Cost              20113 non-null  object
 38  Hourly_Rate           26570 non-null  object
 39  Start_rate            63949 non-null  object
 40  End_rate              25963 non-null  object
dtypes: float64(5), int64(1), object(35)
memory usage: 20.0+ MB
```

Display null values

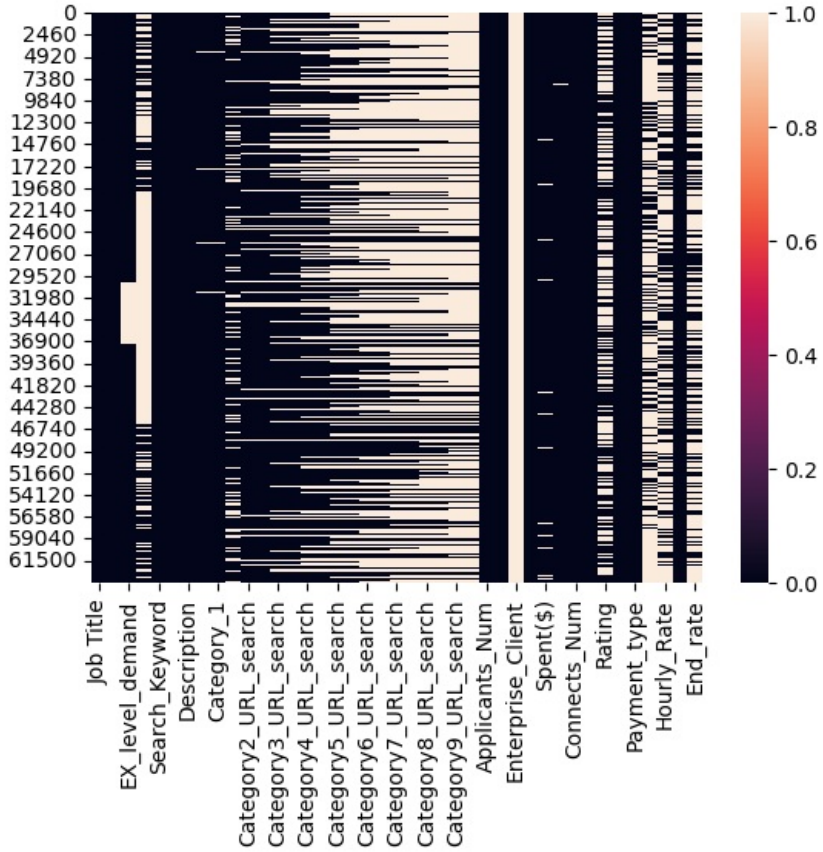
In [5]:
```
df.isnull().sum()
```

Out[5]:
```
Job Title                  2
Job_URL                    1
EX_level_demand         7015
Time_Limitation        40916
Search_Keyword             1
Posted_from               37
Description                2
Category1_URL_search     445
Category_1               445
highlight              13819
Category2_URL_search    4061
Category_2              4061
Category3_URL_search    8953
Category_3              8953
Category4_URL_search   16407
Category_4             16407
Category5_URL_search   24372
Category_5             24372
Category6_URL_search   32643
Category_6             32643
Category7_URL_search   38758
Category_7             38758
Category8_URL_search   43514
Category_8             43514
Category9_URL_search   47469
Category_9             47469
Applicants_Num             1
Payment_Situation          1
Enterprise_Client      63897
Freelancers_Num            0
Spent($)                1646
Client_Country           125
Connects_Num               1
New_Connects_Num           1
Rating                 33357
Feedback_Num               1
Payment_type               1
Job_Cost               43837
Hourly_Rate            37380
Start_rate                 1
End_rate               37987
dtype: int64
```

Display null values in form of Heatmap

In [6]:
```
sns.heatmap(df.isnull())
```

Out[6]:
```
<Axes: >
```



In the dataset, lots of null values are present in every column. Need to handle all null values and

clean the dataset.

Approach used to handlr null values-

1. Sample values from dataset.
2. Replacing null values with a new category (categorical).
3. Mean replacement.
4. Drop column.

Handle null values

In [7]:
```
df=df.drop(columns="Enterprise_Client")  ## Drop Enterprise_Client column
```

In [8]:
```
df["Category1_URL_search"].fillna("No URL" , inplace= True)  ## Replacing null values with a new category (cate
df["Category2_URL_search"].fillna("No URL" , inplace= True)
df["Category3_URL_search"].fillna("No URL" , inplace= True)
df["Category4_URL_search"].fillna("No URL" , inplace= True)
df["Category5_URL_search"].fillna("No URL" , inplace= True)
df["Category6_URL_search"].fillna("No URL" , inplace= True)
df["Category7_URL_search"].fillna("No URL" , inplace= True)
df["Category8_URL_search"].fillna("No URL" , inplace= True)
df["Category9_URL_search"].fillna("No URL" , inplace= True)
```

In [9]:
```
 ##  Replacing null values with a new category (categorical).
df["Job Title"].fillna("No", inplace= True)
df["Job_URL"].fillna("No", inplace= True)
df["Search_Keyword"].fillna("No keyword", inplace= True)
df["Description"].fillna("No description", inplace= True)
df["Applicants_Num"].fillna("0", inplace= True)
df["Payment_Situation"].fillna("Nothing", inplace= True)
df["Connects_Num"].fillna(0, inplace= True)
df["New_Connects_Num"].fillna(0, inplace= True)
df["Feedback_Num"].fillna(0, inplace= True)
df["Start_rate"].fillna(0, inplace= True)
df["Payment_type"].fillna("No payment", inplace= True)
df["Category_1"].fillna("No category" , inplace= True)
df["Category_2"].fillna("No category" , inplace= True)
df["Category_3"].fillna("No category" , inplace= True)
df["Category_4"].fillna("No category" , inplace= True)
df["Category_5"].fillna("No category" , inplace= True)
df["Category_6"].fillna("No category" , inplace= True)
df["Category_7"].fillna("No category" , inplace= True)
df["Category_8"].fillna("No category" , inplace= True)
df["Category_9"].fillna("No category" , inplace= True)
```

In [10]:
```
## Replacing null values with a new category (categorical)
df['EX_level_demand'].fillna('No Exp', inplace=True)
df['Time_Limitation'].fillna('No Limitation', inplace=True)
df['highlight'].fillna('Nothing', inplace=True)
df["Client_Country"].fillna('No', inplace=True)
df["Hourly_Rate"].fillna('$0', inplace=True)
df["End_rate"].fillna('$0', inplace=True)
```

In [11]:
```
df["Job_Cost"]= pd.to_numeric(df["Job_Cost"].str.replace('$',' '), errors='coerce')
## Remove '$' and change data type object to float
```

In [12]:
```
## Replace null values using Sample values from dataset
null_indices= df[df["Job_Cost"].isnull()].index
null_count=len(null_indices)
non_null_values=df["Job_Cost"].dropna()
sample_data=non_null_values.sample(null_count,replace=True,random_state=42)
df.loc[null_indices, 'Job_Cost'] = sample_data.values
```

In [13]:
```
##  change data type object to float
df["Posted_from"]=pd.to_numeric(df["Posted_from"].str.replace("days ago"," "),errors="coerce")
```

In [14]:
```
df["Posted_from"].fillna(0, inplace=True)
```

In [15]:
```
## Replace null values using Sample values from dataset
null_indices= df[df["Spent($)"].isnull()].index
null_count=len(null_indices)
non_null_values=df["Spent($)"].dropna()
sample_data=non_null_values.sample(null_count,replace=True,random_state=42)
df.loc[null_indices, 'Spent($)'] = sample_data.values
```

In [16]:
```
## Replace null values using Mean
```

In [16]:
```
## Replace null values using mean
df["Rating"].mean()
df["Rating"].fillna(df["Rating"].mean(), inplace= True)
```

In [17]:
```
df["Start_rate"]=pd.to_numeric(df["Start_rate"].str.replace("$"," "), errors="coerce")
```

In [18]:
```
df["End_rate"]=pd.to_numeric(df["End_rate"].str.replace("$"," "),errors="coerce")
```

In [19]:
```
df["Start_rate"].fillna(0, inplace=True)
```

In [20]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 63950 entries, 0 to 63949
Data columns (total 40 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   Job Title             63950 non-null  object 
 1   Job_URL               63950 non-null  object 
 2   EX_level_demand       63950 non-null  object 
 3   Time_Limitation       63950 non-null  object 
 4   Search_Keyword        63950 non-null  object 
 5   Posted_from           63950 non-null  float64
 6   Description           63950 non-null  object 
 7   Category1_URL_search  63950 non-null  object 
 8   Category_1            63950 non-null  object 
 9   highlight             63950 non-null  object 
 10  Category2_URL_search  63950 non-null  object 
 11  Category_2            63950 non-null  object 
 12  Category3_URL_search  63950 non-null  object 
 13  Category_3            63950 non-null  object 
 14  Category4_URL_search  63950 non-null  object 
 15  Category_4            63950 non-null  object 
 16  Category5_URL_search  63950 non-null  object 
 17  Category_5            63950 non-null  object 
 18  Category6_URL_search  63950 non-null  object 
 19  Category_6            63950 non-null  object 
 20  Category7_URL_search  63950 non-null  object 
 21  Category_7            63950 non-null  object 
 22  Category8_URL_search  63950 non-null  object 
 23  Category_8            63950 non-null  object 
 24  Category9_URL_search  63950 non-null  object 
 25  Category_9            63950 non-null  object 
 26  Applicants_Num        63950 non-null  object 
 27  Payment_Situation     63950 non-null  object 
 28  Freelancers_Num       63950 non-null  int64  
 29  Spent($)              63950 non-null  float64
 30  Client_Country        63950 non-null  object 
 31  Connects_Num          63950 non-null  float64
 32  New_Connects_Num      63950 non-null  float64
 33  Rating                63950 non-null  float64
 34  Feedback_Num          63950 non-null  float64
 35  Payment_type          63950 non-null  object 
 36  Job_Cost              63950 non-null  float64
 37  Hourly_Rate           63950 non-null  object 
 38  Start_rate            63950 non-null  float64
 39  End_rate              63950 non-null  float64
dtypes: float64(9), int64(1), object(30)
memory usage: 19.5+ MB
```

Check null values after hadle null values . All null values are repalced and dataset is clean.

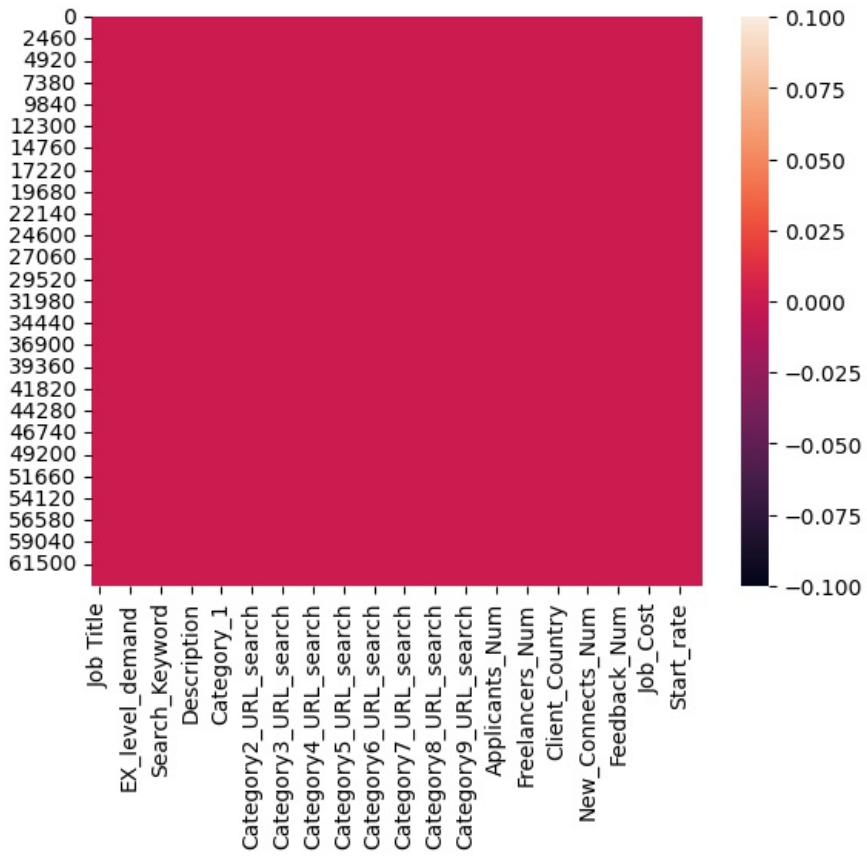
In [21]:
```
df.isnull().sum()
```

Out[21]:
```
Job Title               0
Job_URL                 0
EX_level_demand         0
Time_Limitation         0
Search_Keyword          0
Posted_from             0
Description             0
Category1_URL_search    0
Category_1              0
highlight               0
Category2_URL_search    0
Category_2              0
Category3_URL_search    0
Category_3              0
Category4_URL_search    0
Category_4              0
Category5_URL_search    0
Category_5              0
Category6_URL_search    0
Category_6              0
Category7_URL_search    0
Category_7              0
Category8_URL_search    0
Category_8              0
Category9_URL_search    0
Category_9              0
Applicants_Num          0
Payment_Situation       0
Freelancers_Num         0
Spent($)                0
Client_Country          0
Connects_Num            0
New_Connects_Num        0
Rating                  0
Feedback_Num            0
Payment_type            0
Job_Cost                0
Hourly_Rate             0
Start_rate              0
End_rate                0
dtype: int64
```

Display Heatmap

In [22]:
```
sns.heatmap(df.isnull())
```

Out[22]:
```
<Axes: >
```



Display descriptive statistics of dataset

In [23]:
```
df.describe()
```

Out[23]:

	Posted_from	Freelancers_Num	Spent($)	Connects_Num	New_Connects_Num	Rating	Feedback_Num	Job_Cost	S
count	63950.000000	63950.000000	6.395000e+04	63950.000000	63950.000000	63950.000000	63950.000000	63950.000000	6395
mean	2.656403	1.137654	2.992710e+04	4.506630	9.013260	4.823875	20.996701	181.103597	
std	4.348925	2.391479	3.900837e+05	2.097577	4.195154	0.289385	180.448896	200.625996	1
min	0.000000	0.000000	0.000000e+00	0.000000	0.000000	1.000000	0.000000	5.000000	
25%	0.000000	1.000000	0.000000e+00	2.000000	4.000000	4.823875	0.000000	30.000000	
50%	2.000000	1.000000	7.000000e+01	4.000000	8.000000	4.823875	0.000000	100.000000	
75%	3.000000	1.000000	4.000000e+03	6.000000	12.000000	4.969185	10.000000	250.000000	1
max	23.000000	99.000000	7.000000e+07	8.000000	16.000000	5.000000	12541.000000	999.000000	99

In [24]:
```
float_column=df.select_dtypes(include=['float64']) ## display float datatype column
float_column
```

Out[24]:

	Posted_from	Spent($)	Connects_Num	New_Connects_Num	Rating	Feedback_Num	Job_Cost	Start_rate	End_rate
0	0.0	0.0	6.0	12.0	4.823875	0.0	350.0	0.0	0.0
1	0.0	100.0	4.0	8.0	5.000000	1.0	200.0	0.0	0.0
2	0.0	200.0	6.0	12.0	5.000000	1.0	100.0	40.0	0.0
3	0.0	200000.0	6.0	12.0	4.935536	26.0	50.0	35.0	100.0
4	0.0	0.0	6.0	12.0	4.942242	12512.0	50.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...
63945	3.0	20000.0	4.0	8.0	4.756932	29.0	100.0	0.0	0.0
63946	3.0	70000.0	6.0	12.0	4.666868	42.0	200.0	0.0	0.0
63947	3.0	100.0	6.0	12.0	4.823875	0.0	700.0	0.0	0.0
63948	3.0	0.0	4.0	8.0	4.979751	113.0	900.0	0.0	0.0
63949	0.0	9000.0	0.0	0.0	4.823875	0.0	100.0	0.0	0.0

63950 rows × 9 columns

In [25]:
```
int_column=df.select_dtypes(include=['int64']) ## display integer data type column
int_column
```

Out[25]:

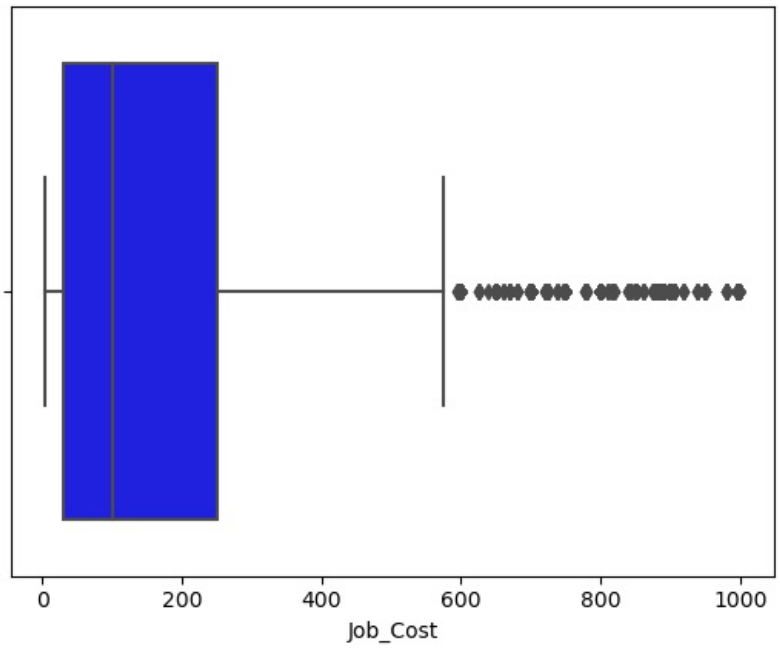
	Freelancers_Num
0	1
1	1
2	1
3	1
4	1
...	...
63945	1
63946	1
63947	1
63948	3
63949	1

63950 rows × 1 columns

Outlier

In [26]:
```
sns.boxplot(data=df,x='Job_Cost',color='blue') ## display outliers using boxplot
```

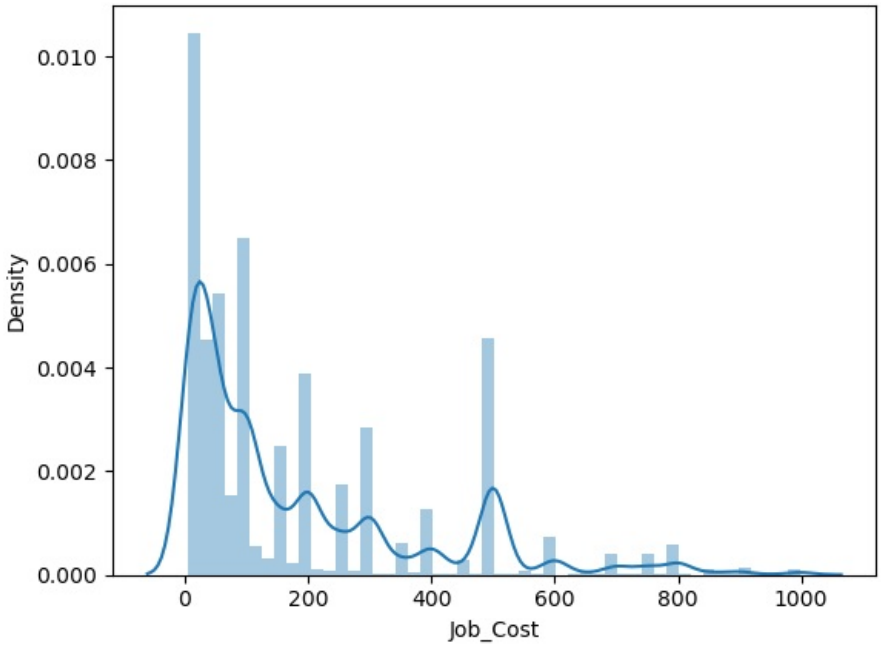
Out[26]:
```
<Axes: xlabel='Job_Cost'>
```



In [27]:
```
sns.distplot(df["Job_Cost"]) ## display distribution plot
```

```
C:\Users\NEW\AppData\Local\Temp\ipykernel_2696\2640447140.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df["Job_Cost"]) ## display distribution plot
```

Out[27]:
```
<Axes: xlabel='Job_Cost', ylabel='Density'>
```



Approach Use-

Remove outlier with IQR (Interquartile Range)

In [28]:
```
std=df["Job_Cost"].std() ## display standard deviation
```

```
std
```

Out[28]:
```
200.6259962195489
```

In [29]:
```
iqr=df["Job_Cost"].quantile(0.75)-df["Job_Cost"].quantile(0.25) ## calculate IQR
iqr
```

Out[29]:
```
220.0
```

In [30]:
```
upperlimit=iqr+1.5*std  ## calculate upperlimit and lower limit
lowerlimit=iqr-1.5*std
```

In [31]:
```
upperlimit
```

Out[31]:
```
520.9389943293234
```

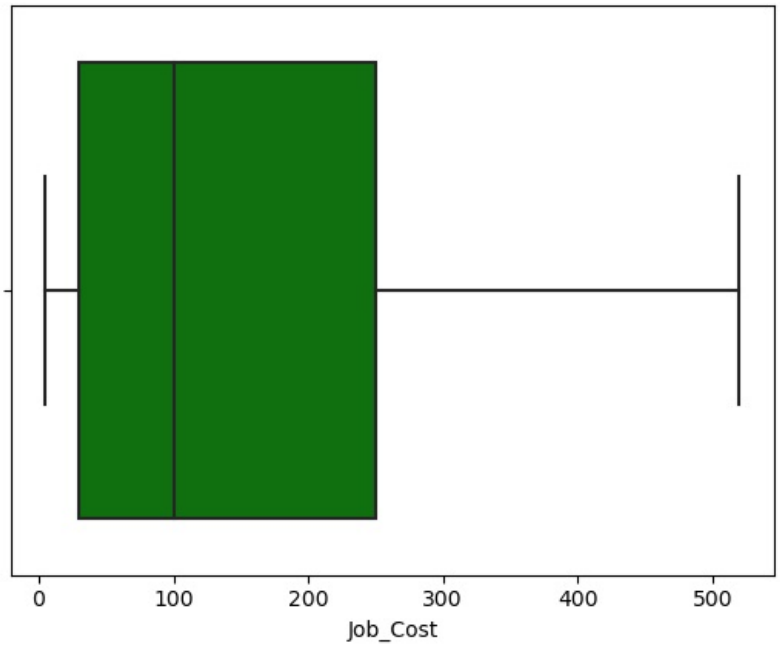
In [32]:
```
lowerlimit
```

Out[32]:
```
-80.93899432932335
```

In [33]:
```
df.loc[df['Job_Cost']>520,"Job_Cost"]=520 ## replace outliers with upperlimit
```

In [34]:
```
sns.boxplot(data=df,x='Job_Cost',color="green") ## boxplot after removing oulier
```

Out[34]:
```
<Axes: xlabel='Job_Cost'>
```

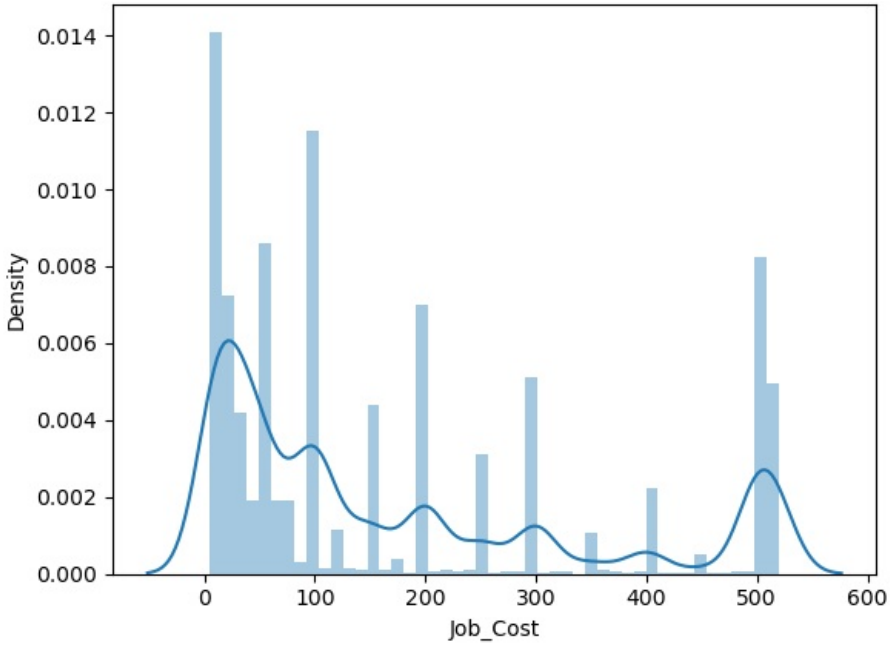


Distribution plot after remove outliers

In [35]:
```
sns.distplot(df['Job_Cost'])
```

```
C:\Users\NEW\AppData\Local\Temp\ipykernel_2696\4288142953.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['Job_Cost'])
```

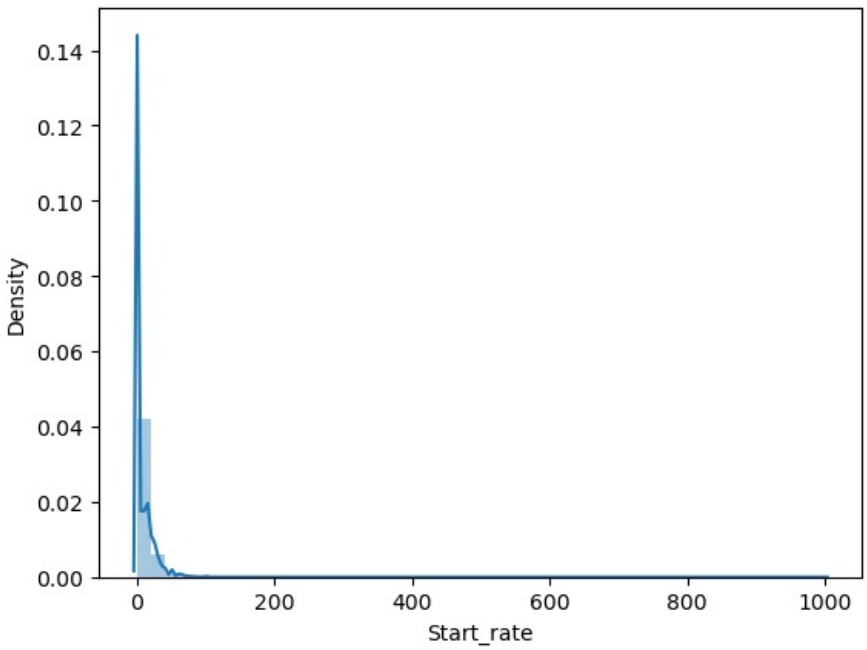
Out[35]:
```
<Axes: xlabel='Job_Cost', ylabel='Density'>
```



In [36]:
```
sns.distplot(df["Start_rate"])
```

```
C:\Users\NEW\AppData\Local\Temp\ipykernel_2696\2057607437.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df["Start_rate"])
```

Out[36]:
```
<Axes: xlabel='Start_rate', ylabel='Density'>
```



In [37]:
```
std=df["Start_rate"].std()
std
```

Out[37]:
```
14.847130543987054
```

In [38]:
```
iqr=df["Start_rate"].quantile(0.75)-df["Start_rate"].quantile(0.25)
iqr
```

Out[38]:
```
13.0
```

In [39]:
```
upperlimit=iqr+1.5*std
upperlimit
```

Out[39]:
```
35.27069581598058
```

In [40]:
```
lowerlimit=iqr-1.5*std
lowerlimit
```

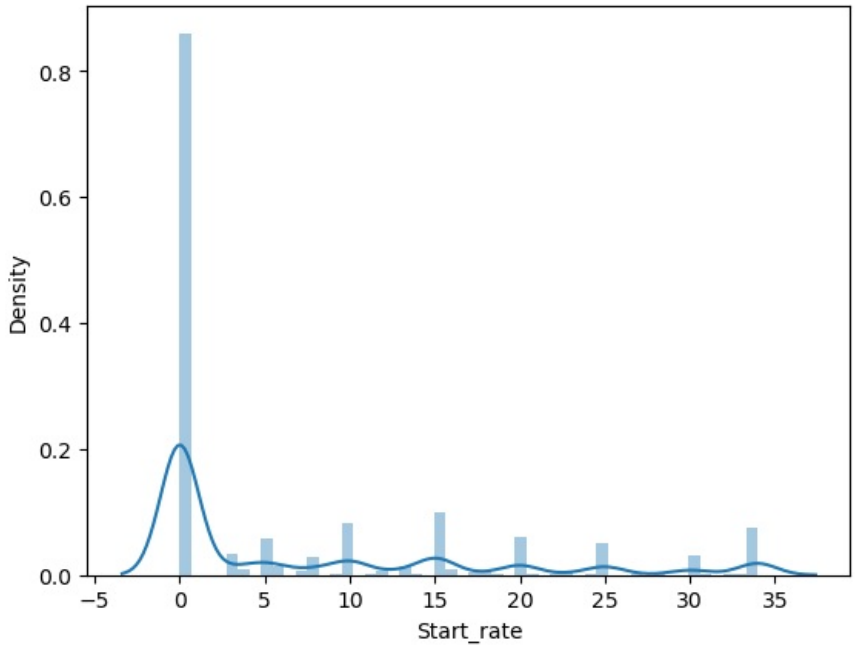
Out[40]:
```
-9.27069581598058
```

In [41]:
```
df.loc[df['Start_rate']>34,"Start_rate"]=34
```

In [42]:
```
sns.distplot(df["Start_rate"])
```

```
C:\Users\NEW\AppData\Local\Temp\ipykernel_2696\2057607437.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df["Start_rate"])
```

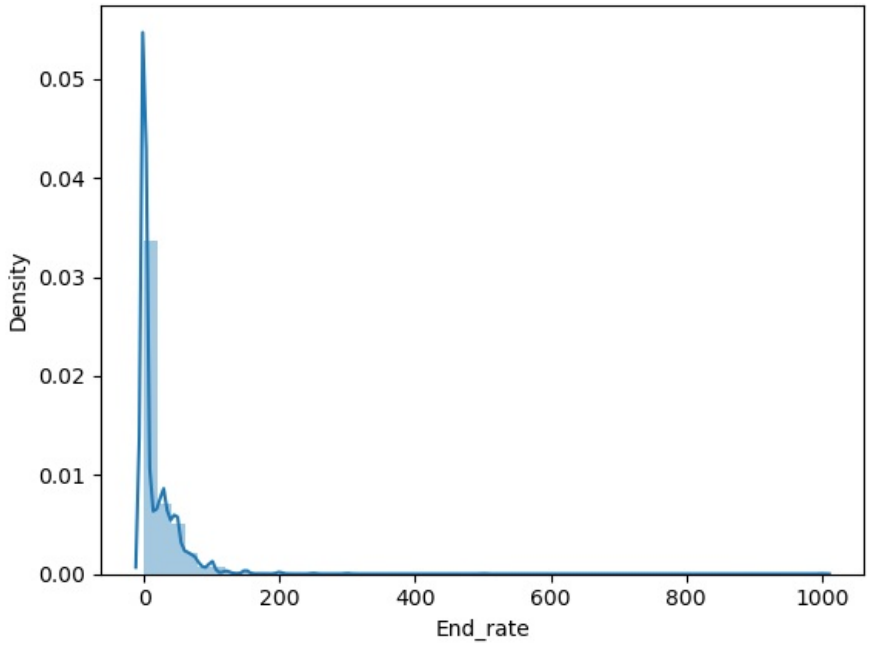
Out[42]:
```
<Axes: xlabel='Start_rate', ylabel='Density'>
```



In [43]:
```
sns.distplot(df["End_rate"])
```

```
C:\Users\NEW\AppData\Local\Temp\ipykernel_2696\1677292270.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df["End_rate"])
```

Out[43]:
```
<Axes: xlabel='End_rate', ylabel='Density'>
```



In [44]:
```
std=df["End_rate"].std()
std
```

Out[44]:
```
36.47022208259511
```

In [45]:
```
iqr=df["End_rate"].quantile(0.75)-df["End_rate"].quantile(0.25)
iqr
```

Out[45]:
```
30.0
```

In [46]:
```
upperlimit=iqr+1.5*std
upperlimit
```

Out[46]:
```
84.70533312389266
```

In [47]:
```
lowerlimit=iqr-1.5*std
lowerlimit
```

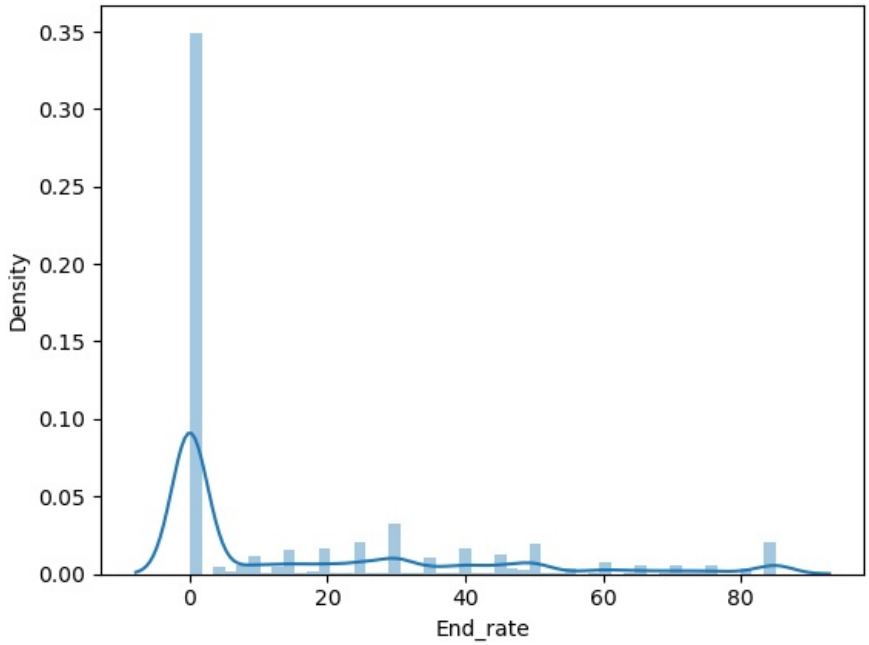
Out[47]:
```
-24.705333123892657
```

In [48]:
```
df.loc[df['End_rate']>85,"End_rate"]=85
```

In [49]:
```
sns.distplot(df["End_rate"])
```

```
C:\Users\NEW\AppData\Local\Temp\ipykernel_2696\1677292270.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df["End_rate"])
```

Out[49]:
```
<Axes: xlabel='End_rate', ylabel='Density'>
```



Business Analysis

Column - Search_Keyword(candidates search for job position)

In [50]:
```
keyword_search=df["Search_Keyword"].value_counts() ## count each related job serach by candidate
keyword_search
```

Out[50]:
```
Search_Keyword
Developer       29220
Marketing       24745
3D               4995
Data_science     4989
No keyword          1
Name: count, dtype: int64
```

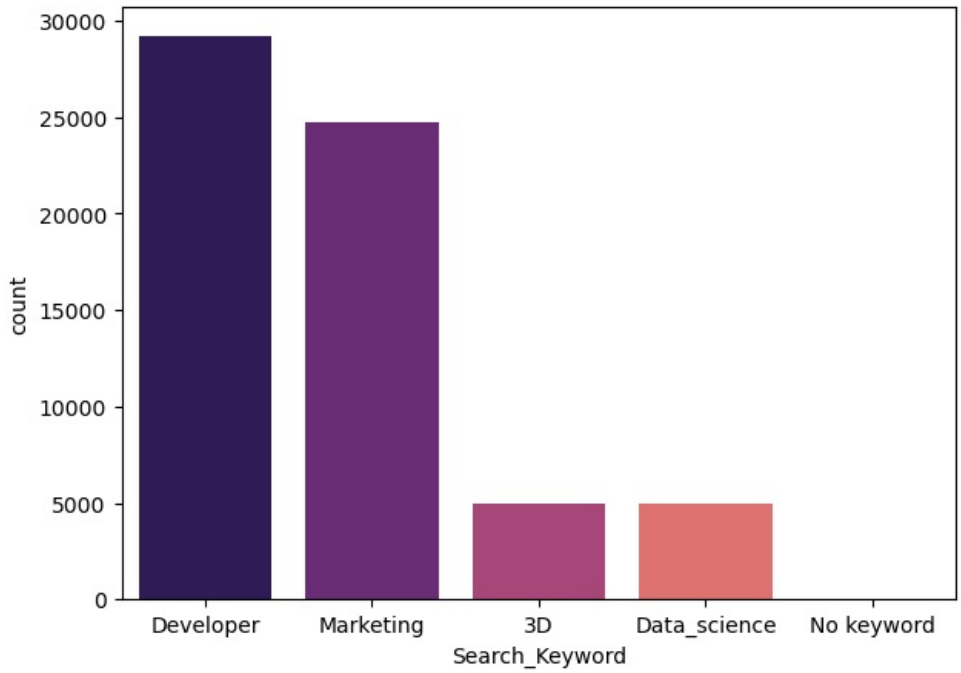
In [51]:
```
most_job_search=keyword_search.idxmax()  ## display most ralated job search by candidate
print("The most job search by candidate: ",most_job_search )
```

```
The most job search by candidate:  Developer
```

Graphical represtation of search keywords

In [52]:
```
fig,ax = plt.subplots(figsize=(7,5))
sns.countplot(df, x = 'Search_Keyword', ax = ax, order = df['Search_Keyword'].value_counts().index, palette="ma
```

Out[52]:
```
<Axes: xlabel='Search_Keyword', ylabel='count'>
```



The data indicates that there is substantial interest in developer-related and marketing-related topics. 3D and Data Science appear to be less popular search topics.

Column- EX_level_demand (Describes the skill tier desired. Helps candidates evaluate if they are a fit for the job. (there's 3 tiers on the site: "Entry level & Intermediate & Expert")

In [53]:
```
Expert_level_count=df["EX_level_demand"].value_counts()
max_expert_level_count= Expert_level_count.idxmax()
print("The most jobs for: ",max_expert_level_count )
```

```
The most jobs for:  Intermediate
```

In [54]:
```
Expert_level_count=df["EX_level_demand"].value_counts()
Expert_level_count
```

Out[54]:
```
EX_level_demand
Intermediate    30391
Expert          20972
No Exp           7015
Entry level      5572
Name: count, dtype: int64
```

In [55]:
```
plt.figure(figsize=(6, 4))
bars = plt.bar(Expert_level_count.index, Expert_level_count.values, color=['blue', 'green', 'red', 'orange'])
plt.xlabel('Expert Level')
plt.ylabel('Count')
plt.title('Count of Expert Levels')
plt.xticks(rotation=45)
plt.show()
```

This graph suggests that there are more jobs for intermediate and expert levels of expertise compared to those with no experience or at the entry level.

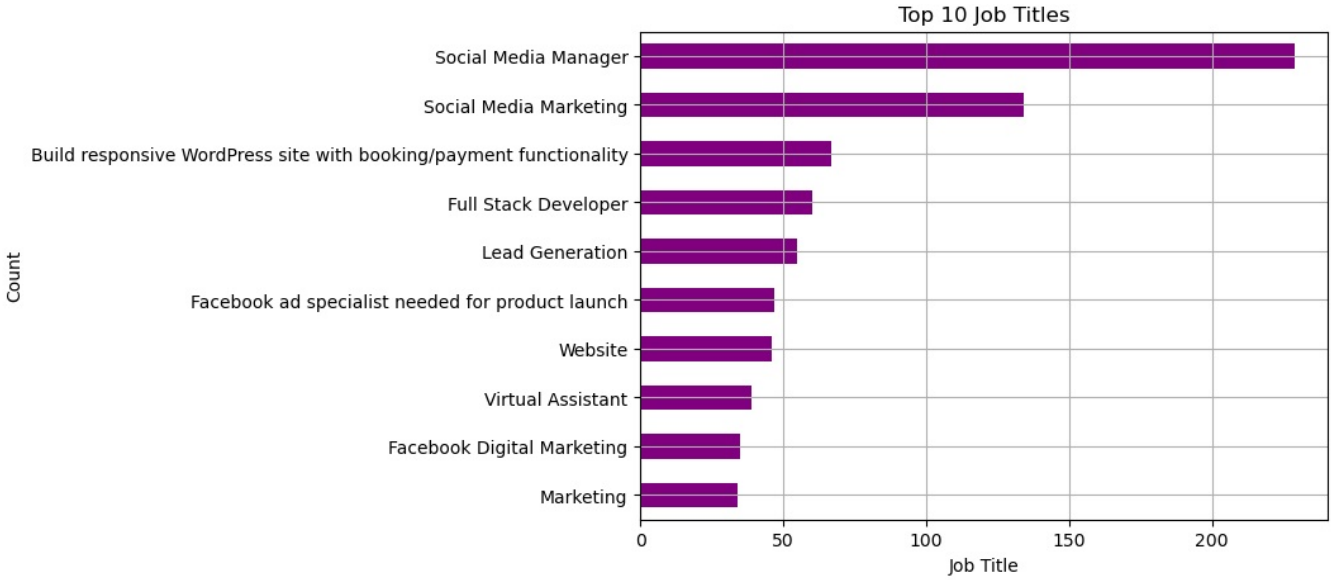
Column- Job Title(Specifies the nature of the job. It aids potential applicants in quickly understanding the role that client needs)

In [56]:
```
df["Job Title"]=df["Job Title"].str.replace("Social media manager","Social Media Manager")
df["Job Title"]=df["Job Title"].str.replace("Social media marketing","Social Media Marketing")
```

In [57]:
```
x=df["Job Title"].value_counts().sort_values(ascending=False)
x          ## display count of total jobs available for each job position and sort descending order
```

Out[57]:
```
Job Title
Social Media Manager                                                    229
Social Media Marketing                                                  134
Build responsive WordPress site with booking/payment functionality       67
Full Stack Developer                                                     60
Lead Generation                                                          55
                                                                       ...
Mail Merge                                                                1
Marketing Portfolio Showcasing Construction Projects                      1
Email Specialist                                                          1
Sales Development Representative needed on the Nordics market (Sweden and Denmark)      1
Creating videos (including UGC) for TikTok - Arabic                       1
Name: count, Length: 54902, dtype: int64
```

In [58]:
```
top_10=x.head(10) ## display top 10 jobs available for each job position
top_10_sorted = top_10.sort_values()
plt.figure(figsize=(7, 5))
top_10_sorted.plot(kind='barh', color='purple')
plt.xlabel('Job Title')
plt.ylabel('Count')
plt.title('Top 10 Job Titles')
plt.grid(True)
plt.show()
```



The bar graph titled “Top 10 Job Titles” provides insights into the most common job titles. Here are the key takeaways:

Social Media Manager: This job title has the highest count, exceeding 200.

Other notable job titles include Social Media Marketing, Full Stack Developer, and Lead Generation.The more jobs are on Social media and website development roles.

Column- highlight(it's an indicator about this job related)

In [59]:
```
top_10_position_SMM_job=df[df["Job Title"]=="Social Media Manager"]["highlight"].value_counts().head(10)
top_10_position_SMM_job
```

Out[59]:
```
highlight
Social Media Marketing             86
marketing                          61
Nothing                            29
Influencer Marketing               13
Social Media Marketing Strategy    11
Marketing                           7
market                              6
Social Video Marketing              3
Digital Marketing                   2
Marketing Analytics                 2
Name: count, dtype: int64
```

Output shows highlighted job of Social Media Manager job. Most Social Media Manager jobs are on marketing.

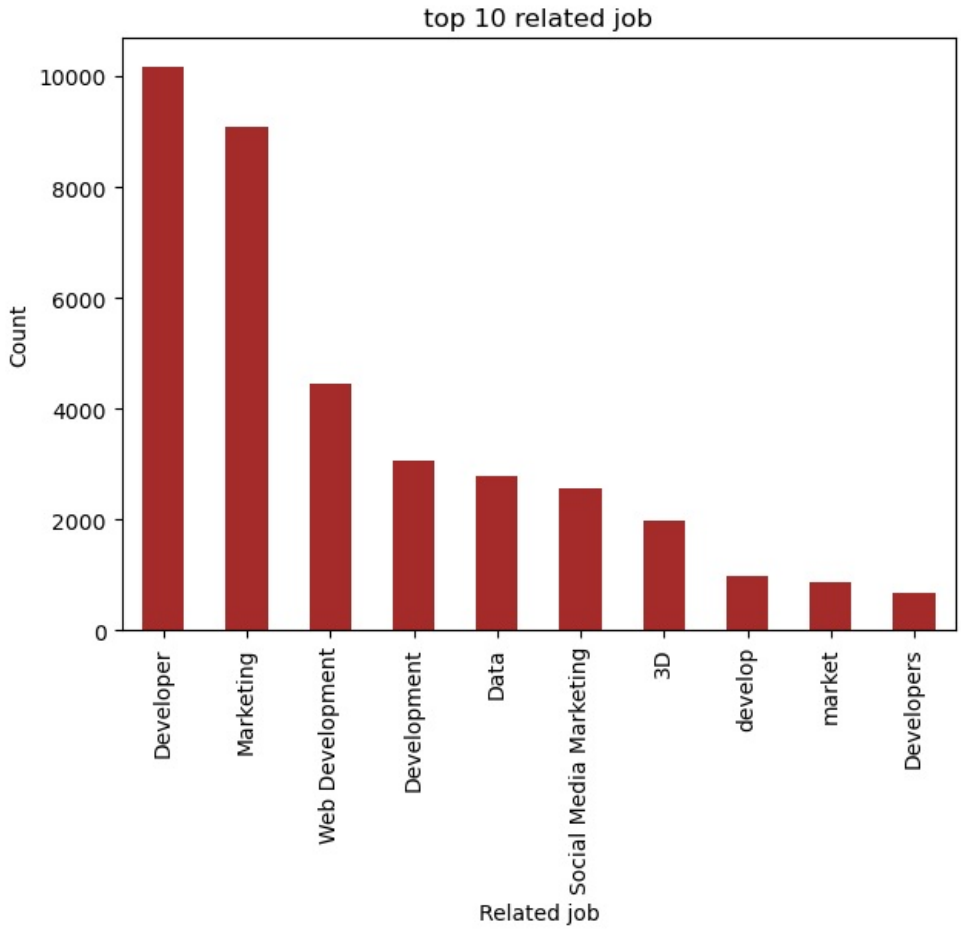
In [60]:
```
df["highlight"]=df["highlight"].str.replace("developer","Developer")
df["highlight"]=df["highlight"].str.replace("data","Data")
df["highlight"]=df["highlight"].str.replace("development","Development")
df["highlight"]=df["highlight"].str.replace("marketing","Marketing")
related_job=df["highlight"].value_counts()
related_job
```

Out[60]:
```
highlight
Nothing                  13819
Developer                10174
Marketing                 9084
Web Development           4456
Development               3063
                         ...
Procedure Development        1
DEveloper                    1
Data Management              1
Computer Science             1
MARKETER                     1
Name: count, Length: 324, dtype: int64
```

Output suggest that not particular highlighted jobs are more available.

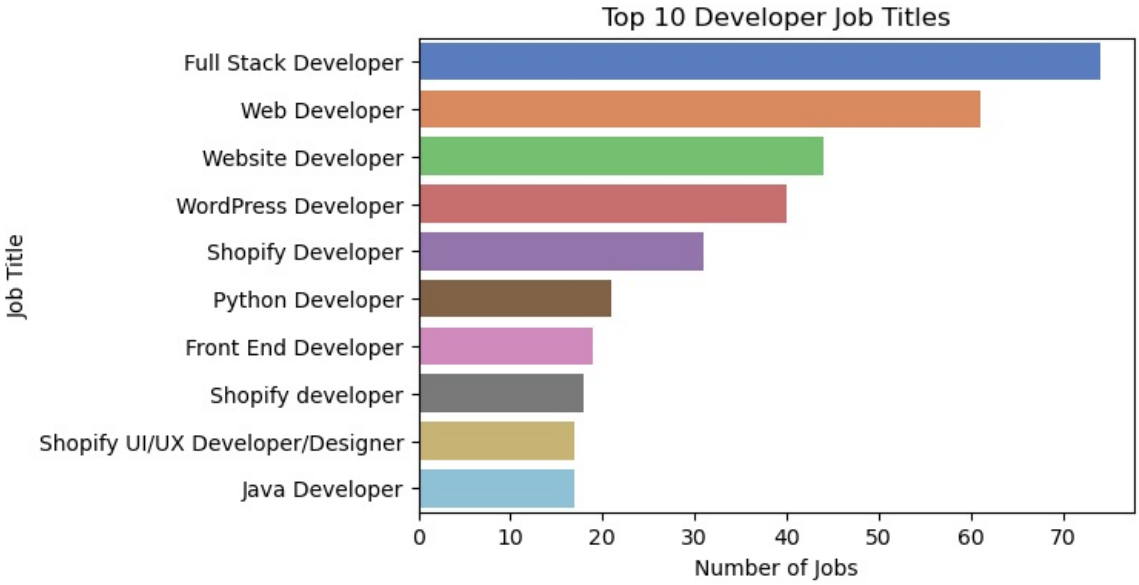
In [61]:
```
top_10_related_job=df[df["highlight"]!="Nothing"]["highlight"].value_counts().head(10)
plt.figure(figsize=(7, 5))
top_10_related_job.plot(kind="bar",color="brown")
plt.xlabel('Related job')
```

```
plt.ylabel('Count')
plt.title('top 10 related job')
plt.show()
```



Graph shows that apart from not particular highlighted jobs, most of the highlighted jos are available for developer only. Other than developer , more no of jobs for marketing.

In [62]:
```
df["Job Title"]=df["Job Title"].str.replace("Full stack developer","Full Stack Developer")
df["Job Title"]=df["Job Title"].str.replace("Website developer","Website Developer")
df["Job Title"]=df["Job Title"].str.replace("Wordpress Developer","WordPress Developer")
df["Job Title"]=df["Job Title"].str.replace("Web developer","Web Developer")
top_10_dev_highlight_job=df[df["highlight"]=="Developer"]["Job Title"].value_counts().head(10)
plt.figure(figsize=(6, 4))
sns.barplot(x=top_10_dev_highlight_job.values, y=top_10_dev_highlight_job.index, palette="muted")
plt.xlabel('Number of Jobs')
plt.ylabel('Job Title')
plt.title('Top 10 Developer Job Titles')
plt.show()
```



Graph shows top 10 no of developer jobs. Most no of developer jobs are for full

stack , web developer , wordPress developer. Other titles include Website Developer, Shopify Developer, Python Developer, Front End Developer, Java Developer, and Shopify UI/UX Developer/Designer.

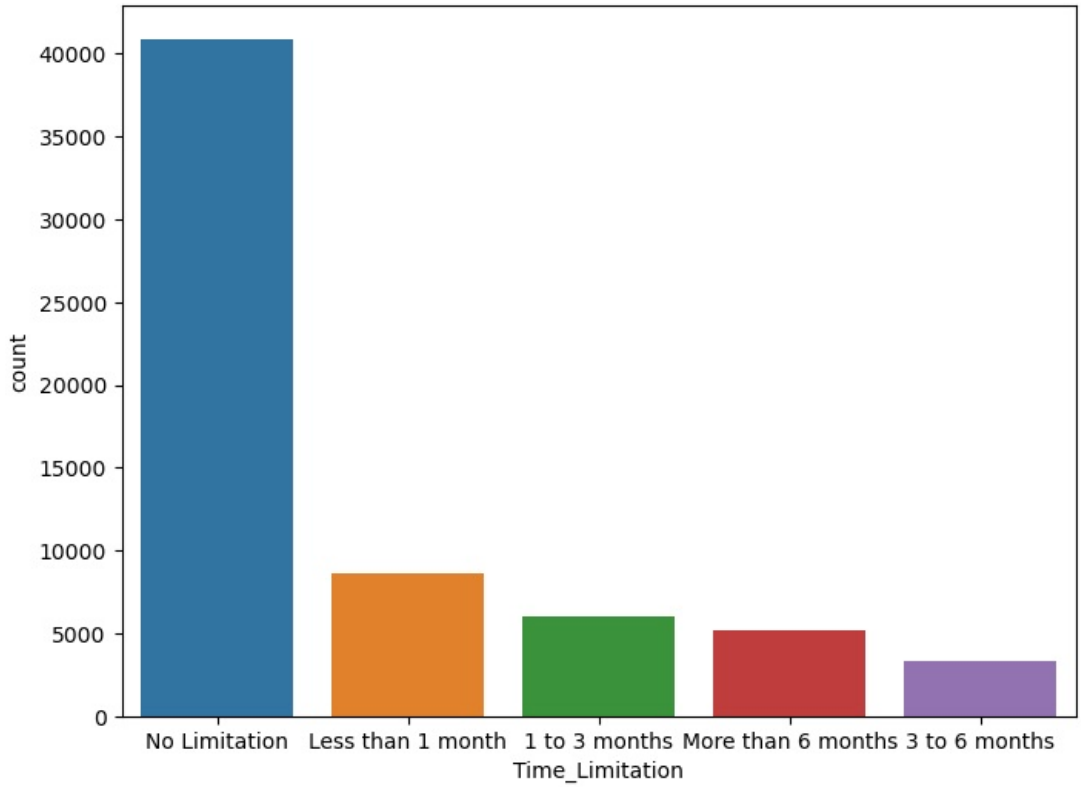
Column- Time_Limitation(Indicates the roughly time frame the client demands for the job, assisting freelancers in managing their schedules)

In [63]:
```
df["Time_Limitation"]=df["Time_Limitation"].str.split(",").str[0] ## clean unnessary data
df["Time_Limitation"]
```

Out[63]:
```
0             No Limitation
1             No Limitation
2             3 to 6 months
3             3 to 6 months
4        More than 6 months
                 ...
63945    More than 6 months
63946    More than 6 months
63947         1 to 3 months
63948         No Limitation
63949         No Limitation
Name: Time_Limitation, Length: 63950, dtype: object
```

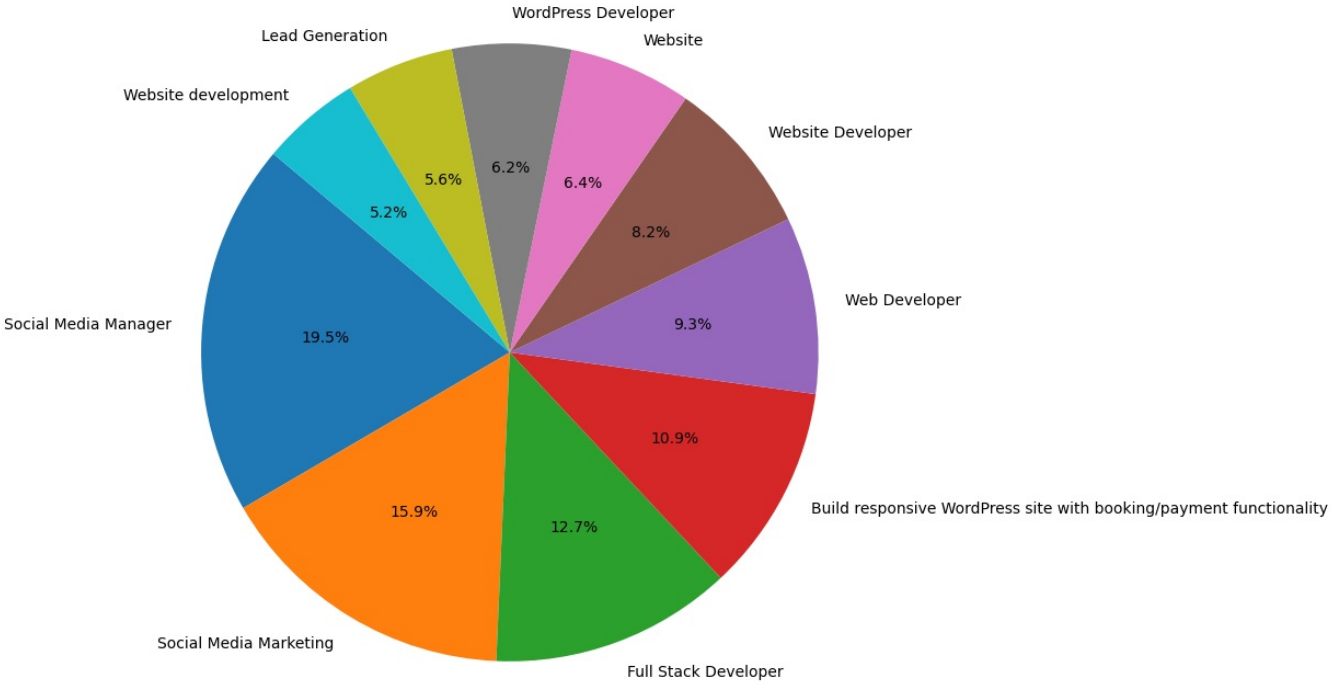
In [64]:
```
job_duration_count=df["Time_Limitation"].value_counts()
fig,ax=plt.subplots(figsize=(8,6))
sns.countplot(df,x="Time_Limitation",ax=ax,order=df["Time_Limitation"].value_counts().index)
```

Out[64]:
```
<Axes: xlabel='Time_Limitation', ylabel='count'>
```



Graph shows the counts of jobs in a particular time duration demand by client. Most of jobs have not particular duration. Clients want freelancers for long duration.

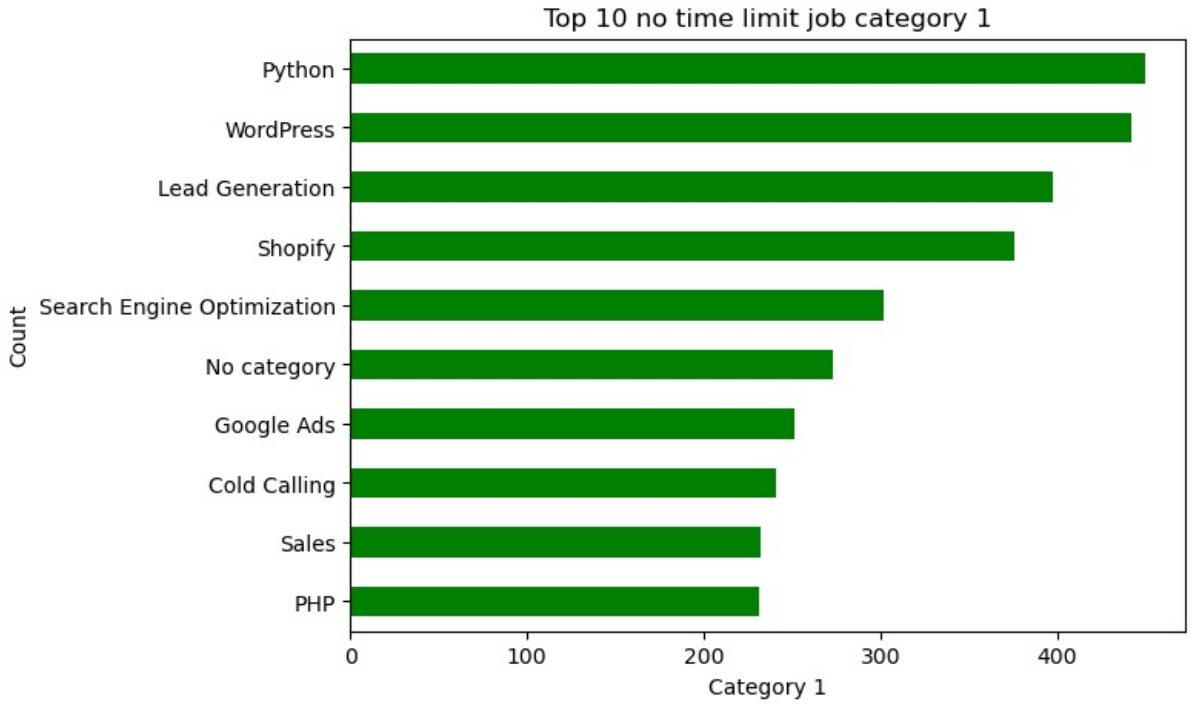
In [65]:
```
top_10_no_duration_job=df[df["Time_Limitation"]=="No Limitation"]["Job Title"].value_counts().head(10)
plt.figure(figsize=(8, 8))
plt.pie(top_10_no_duration_job, labels=top_10_no_duration_job.index, autopct='%1.1f%%', startangle=140)
plt.axis('equal')
plt.show()
```



Pie chart shows tclients need social media manager job role for long duration. After that job role for social media marketing , payment functionality and full stack developer role are available for long duration.

column- Category_1 (This column specify categories of skills that the client wrote as a relevant to the job)

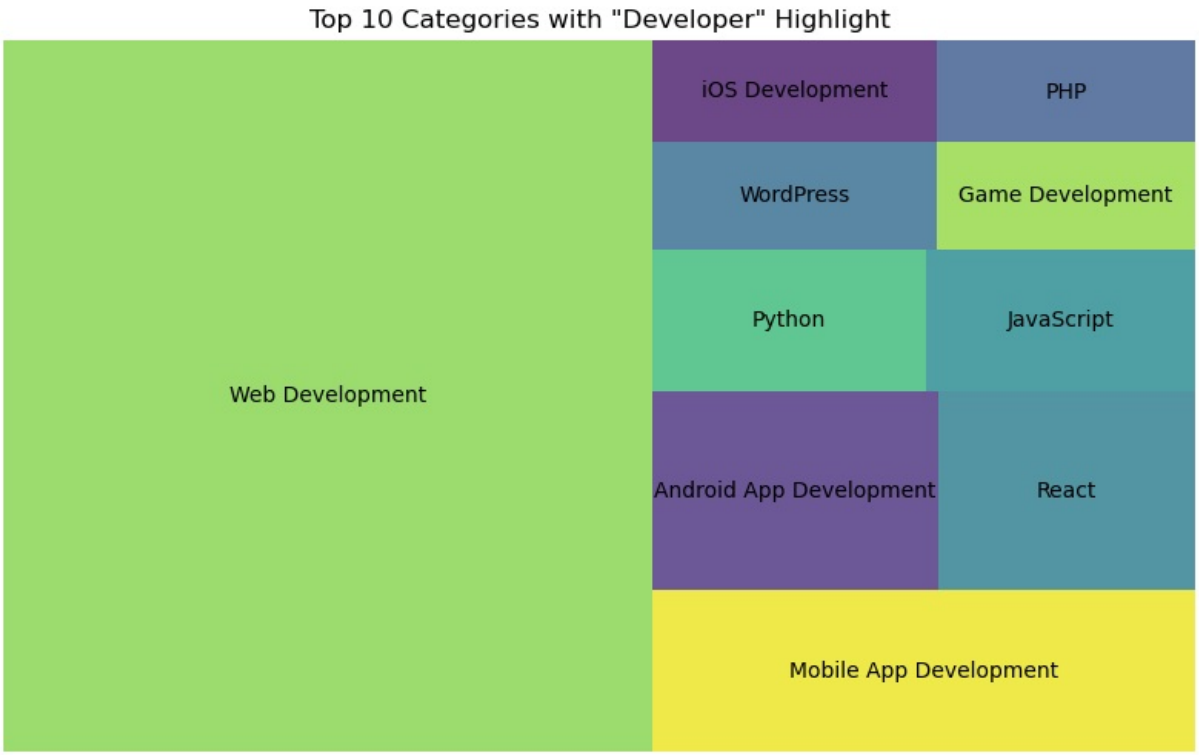
In [66]:
```
category1_with_nothing_highlight=df[df["highlight"]=="Nothing"]["Category_1"].value_counts().head(10)
top_10_sorted = category1_with_nothing_highlight.sort_values()
plt.figure(figsize=(7, 5))
top_10_sorted.plot(kind='barh', color='green')
plt.xlabel('Category 1')
plt.ylabel('Count')
plt.title('Top 10 no time limit job category 1')
plt.show()
```



Bar plot shows that most of skills requires for not highlighted jobs are Python, WordPress. After that skills are required for lead generation, shopify and search engine optimization.

In [67]:
```
category1_with_developer_highlight=df[df["highlight"]=="Developer"]["Category_1"].value_counts().head(10)
plt.figure(figsize=(10, 6))
squarify.plot(sizes=category1_with_developer_highlight.values,
              label=category1_with_developer_highlight.index,
```

```
            alpha=0.8)
plt.axis('off')
plt.title('Top 10 Categories with "Developer" Highlight')
plt.show()
```



Treemap gives the conclusion that most of developer jobs are required skills on web development. Others required skills for developer jobs are app development , React , Python, Javascript.

Column- Applicants_Num (number of the freelancers who applied proposal for that job yet. A high number might deter some from applying)

In [68]:
```
application_num=df["Applicants_Num"].value_counts()
application_num
```

Out[68]:
```
Applicants_Num
20 to 50       17490
5 to 10        12466
Less than 5    11103
10 to 15        9678
15 to 20        7391
50+             5821
0                  1
Name: count, dtype: int64
```

Output shows that most jobs are applied by 20 to 50 freelancers only. Less jobs are applied by 50+ freelancers.

In [69]:
```
most_application_job=df[df["Applicants_Num"]=="20 to 50"]["Category_1"].value_counts().head(10)
most_application_job
```

Out[69]:
```
Category_1
Web Development           2886
Social Media Marketing    1146
3D Modeling                445
Mobile App Development     333
Email Marketing            321
Marketing Strategy         308
3D Design                  300
3D Rendering               296
Python                     271
WordPress                  245
Name: count, dtype: int64
```

From above output we can conclude that each job which required web development skill are applied by 20 to 30 frelancers most. Another jobs which required Social Media Marketing , 3D Modeling ,Mobile App Development also applied more.

In [70]:
```
fifty_plus_application_job=df[df["Applicants_Num"]=="50+"]["Category_1"].value_counts().head(10)
fifty_plus_application_job
```

Out[70]:
```
Category_1
Web Development           1439
Mobile App Development     235
React                      168
Social Media Marketing     164
3D Modeling                104
3D Rendering                99
WordPress                   80
iOS Development             79
JavaScript                  70
PHP                         69
Name: count, dtype: int64
```

From above output we can conclude that each job which required web development skill are applied by 50 plus frelancers most.

Web development , Mobile app development , social media marketing jobs are applied more by freelancers.

In [71]:
```
less_application_job=df[df["Applicants_Num"]=="Less than 5"]["Category_1"].value_counts().tail(10)
less_application_job
```

Out[71]:
```
Category_1
Google Apps Script          1
Budget Management           1
Ember.js                    1
Helpdesk                    1
Web Service                 1
Moodle                      1
Hardware Troubleshooting    1
Game Art                    1
JFrog Artifactory           1
Amazon Vendor Central       1
Name: count, dtype: int64
```

From above result we see that each job which required Google Apps Script ,Budget Management ,Ember.js are applied very less. Demad of these jobs are very low .

Column-Client_Country(Knowledge of client location can influence decisions based on time zones or cultural familiarity)

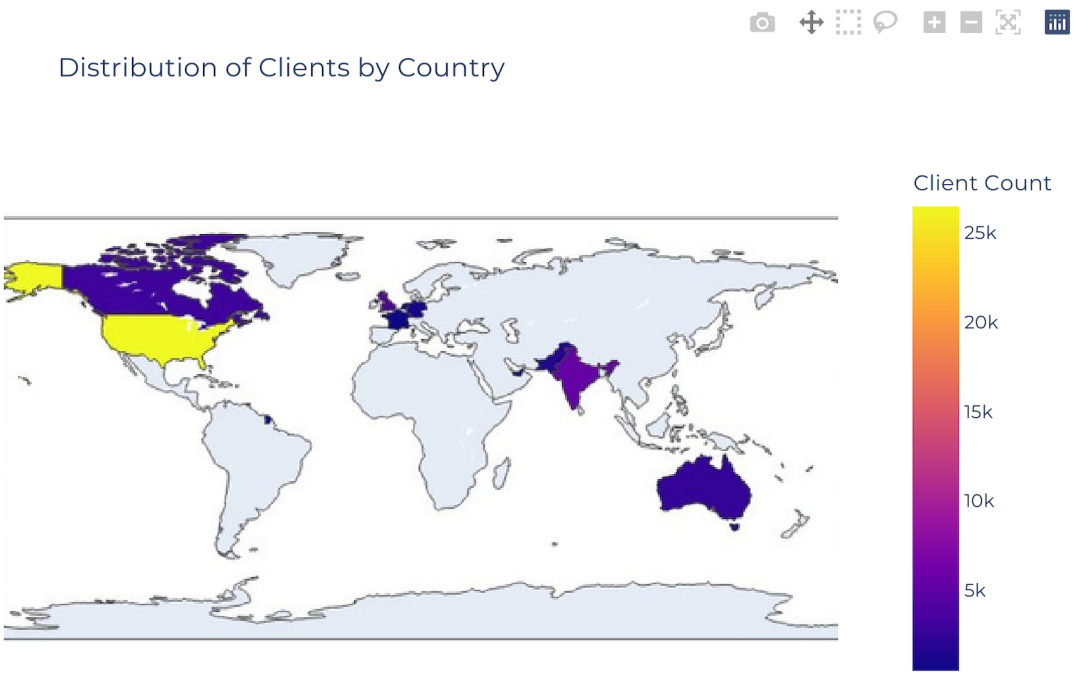
In [72]:
```
job_country_client=df["Client_Country"].value_counts().head(10)
job_country_client
```

Out[72]:
```
Client_Country
United States           26699
India                    5734
United Kingdom           4353
Canada                   3270
Australia                2783
Pakistan                 1841
Germany                  1289
United Arab Emirates     1148
Netherlands               882
France                    764
Name: count, dtype: int64
```

From above result we see that jobs from USA are most. Another more no of jobs are also available from India , UK, Canada, Australia.

In [73]:
```
country_codes = {
    'United States': 'USA',
    'India': 'IND',
    'United Kingdom': 'GBR',
    'Canada': 'CAN',
    'Australia': 'AUS',
    'Pakistan': 'PAK',
    'Germany': 'DEU',
    'United Arab Emirates': 'ARE',
    'Netherlands': 'NLD',
    'France': 'FRA'
}
df['ISO_Code'] = df['Client_Country'].map(country_codes)
job_country_client = df["ISO_Code"].value_counts().reset_index()
job_country_client.columns = ['ISO_Code', 'Client Count']
fig = px.choropleth(job_country_client, locations="ISO_Code",
```

```
            color="Client Count",
            color_continuous_scale=px.colors.sequential.Plasma,
            title="Distribution of Clients by Country")
fig.show()
```



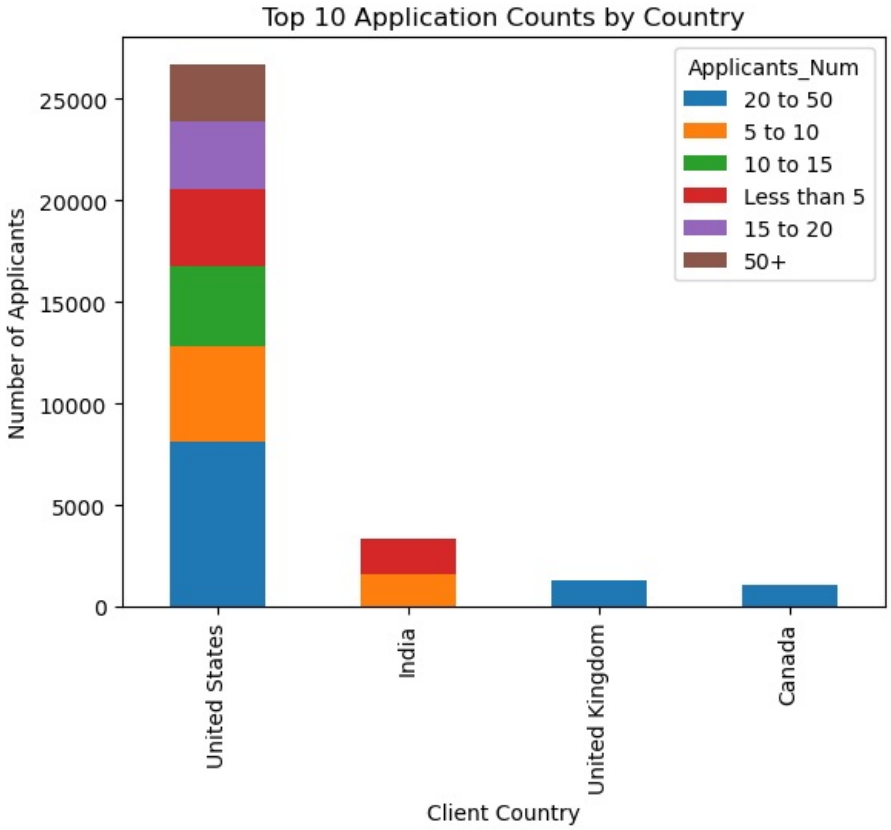
Map visualization of available jobs from different client country.

In [74]:
```
application_countrywise=df.groupby("Client_Country")
["Applicants_Num"].value_counts().sort_values(ascending=Fals application_countrywise
```

Out[74]:
```
Client_Country  Applicants_Num
United States   20 to 50        8130
                5 to 10         4669
                10 to 15        3987
                Less than 5     3807
                15 to 20        3282
                50+             2824
India           Less than 5     1723
                5 to 10         1594
United Kingdom  20 to 50        1288
Canada          20 to 50        1082
Name: count, dtype: int64
```

Output shows that most no of freelancers has applied for US client job.

In [75]:
```
application_country_wise_df = application_countrywise.unstack().fillna(0)
application_country_wise_df.plot(kind='bar', stacked=True)
plt.title('Top 10 Application Counts by Country')
plt.xlabel('Client Country')
plt.ylabel('Number of Applicants')
plt.show()
```



This Stacked bar graph provides a clear comparison of application counts among these four countries. The United States has a significantly higher number of applicants compared to India, the United Kingdom, and Canada.

Column- Freelancers_Num(number of freelancers that the client needs for the job.

In [76]:
```
freelance_num_countrywise=df.groupby("Client_Country")["Freelancers_Num"].sum().sort_values(ascending=False).he
freelance_num_countrywise
```

Out[76]:
```
Client_Country
United States           29812
India                    7437
United Kingdom           4705
Canada                   3658
Australia                2969
Pakistan                 2077
Germany                  1584
United Arab Emirates     1194
Netherlands               944
France                    934
Name: Freelancers_Num, dtype: int64
```

Output shows that most of the requirements are from US only. After that India , UK, Canada have also good requirement.

In [77]:
```
freelance_num_countrywise=df.groupby("Client_Country")["Freelancers_Num"].sum().sort_values(ascending=False).ta
freelance_num_countrywise
```

Out[77]:
```
Client_Country
Niger                          1
Turks and Caicos Islands       1
Brunei Darussalam              1
Antigua and Barbuda            1
Saint Kitts and Nevis          1
Saint Martin (French part)     1
Samoa                          1
Tajikistan                     1
Sierra Leone                   1
Northern Mariana Islands       1
Name: Freelancers_Num, dtype: int64
```

output shows the client country which has very less job

In [78]:
```
freelance_num_job_wise=df.groupby("Job Title")["Freelancers_Num"].sum().sort_values(ascending=False).head(10)
freelance_num_job_wise
```

Out[78]:
```
Job Title
Android Frontend Debug - Java code anti-decompilation & avoid source code leakage    297
Travel Expert Writer                                                                 289
Social Media Manager                                                                 239
Android application upload                                                           208
Umfrage zu Ihrer Plattformarbeit                                                     198
Android App publishing (If you have Google console account apply this job)           173
Need freelancer for host android application                                         152
Doctors Leads Generation                                                             140
Social Media Marketing                                                               134
Need android application upload                                                      130
Name: Freelancers_Num, dtype: int64
```

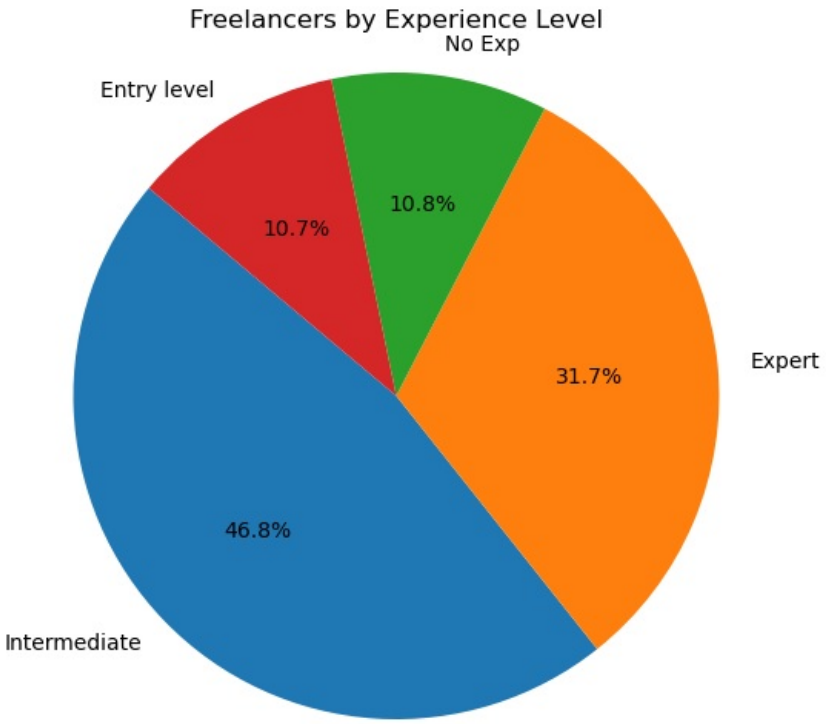
output provides top 10 job roles where requirement for freelancers are high.

In [79]:
```
top_10_freelance_num_category_wise=df.groupby("Category_1")["Freelancers_Num"].sum().sort_values(ascending=Fals
top_10_freelance_num_category_wise
```

Out[79]:
```
Category_1
Web Development            9177
Social Media Marketing     5242
Marketing Strategy         2128
Android App Development    1789
Mobile App Development     1689
Email Marketing            1382
3D Modeling                1259
Market Research            1177
Python                      944
3D Design                   869
Name: Freelancers_Num, dtype: int64
```

Output provides the skills for which most freelancers would be needed. Most freelancers is needed for web development.

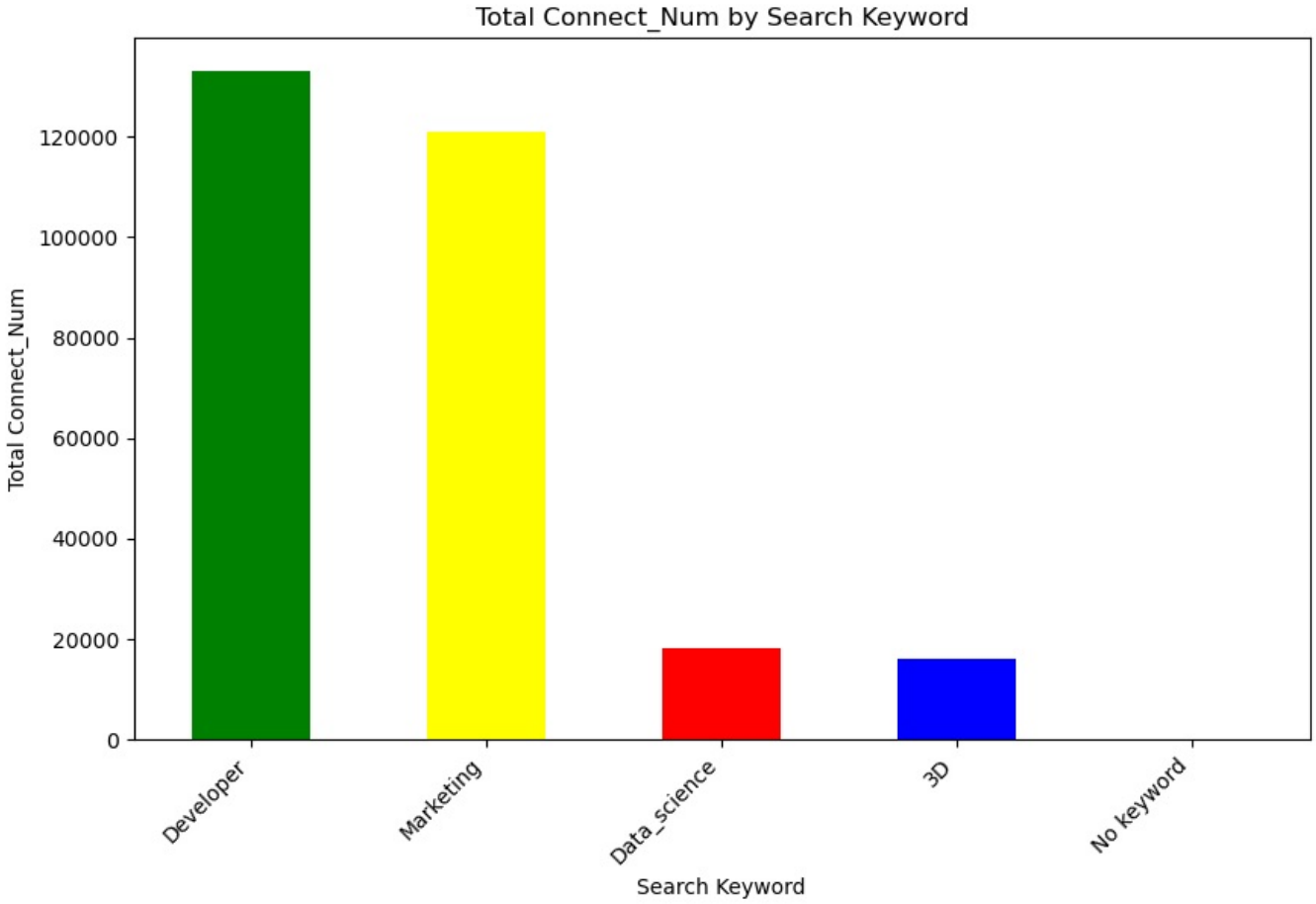
In [80]:
```
freelance_num_exp=df.groupby("EX_level_demand")["Freelancers_Num"].sum().sort_values(ascending=False).head(10)
plt.figure(figsize=(10, 6))
plt.pie(freelance_num_exp, labels=freelance_num_exp.index, autopct='%1.1f%%', startangle=140)
plt.title('Freelancers by Experience Level')
plt.axis('equal')
plt.show()
```



This chart provides a visual representation of the experience levels among freelancers. It shows that a significant portion of freelancers are needed at the Intermediate level, followed by those at the Expert level. Entry Level and No Experience freelancers make up a smaller portion of the total.

Column- Connects_Num(connects are the currency of the site; the freelancers buy every 10 connects for 1.5 . so the more job costs the less freelancers applying for it. (the job costs 1, 2, 4, 6 or 8 connects)

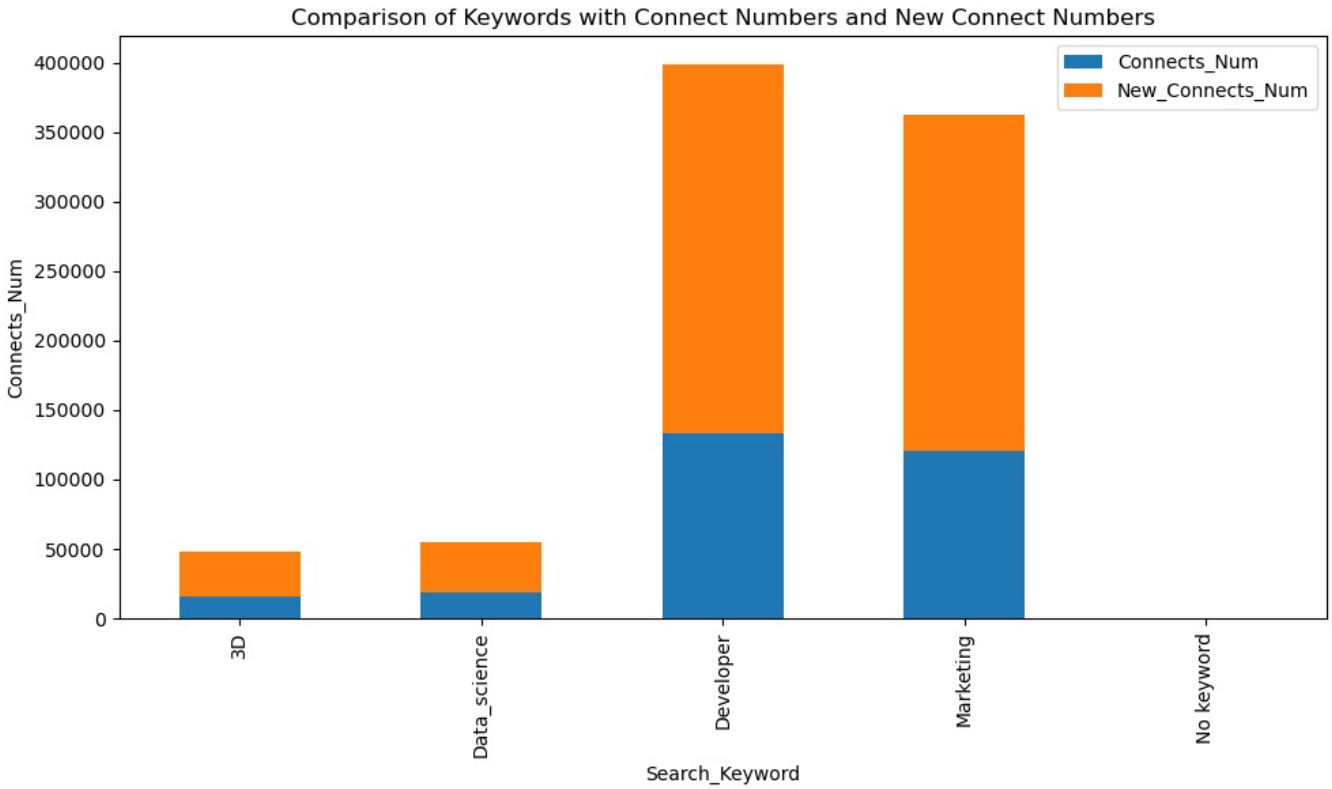
In [81]:
```
connect_num_by_keyword = df.groupby('Search_Keyword')['Connects_Num'].sum().sort_values(ascending=False)
plt.figure(figsize=(10, 6))
connect_num_by_keyword.plot(kind='bar', color=['green','yellow', 'red', 'blue'])
plt.title('Total Connect_Num by Search Keyword')
plt.xlabel('Search Keyword')
plt.ylabel('Total Connect_Num')
plt.xticks(rotation=45, ha='right')
plt.show()
```



The graph suggests that ‘Developer’ and ‘Marketing’ jobs more popular and more available on the platform, leading to more connects being used. In contrast, ‘Data Science’, ‘3D’, and ‘No Keyword’ jobs might have less competition and availability, resulting in fewer connects being used. The cost of a job in connects could also influence these numbers, as jobs that cost more connects might deter some freelancers from applying.

Column-(New_Connects_Num)-currently the site doubled the connects for the jobs, so now it's (2, 4, 8, 12, 16)

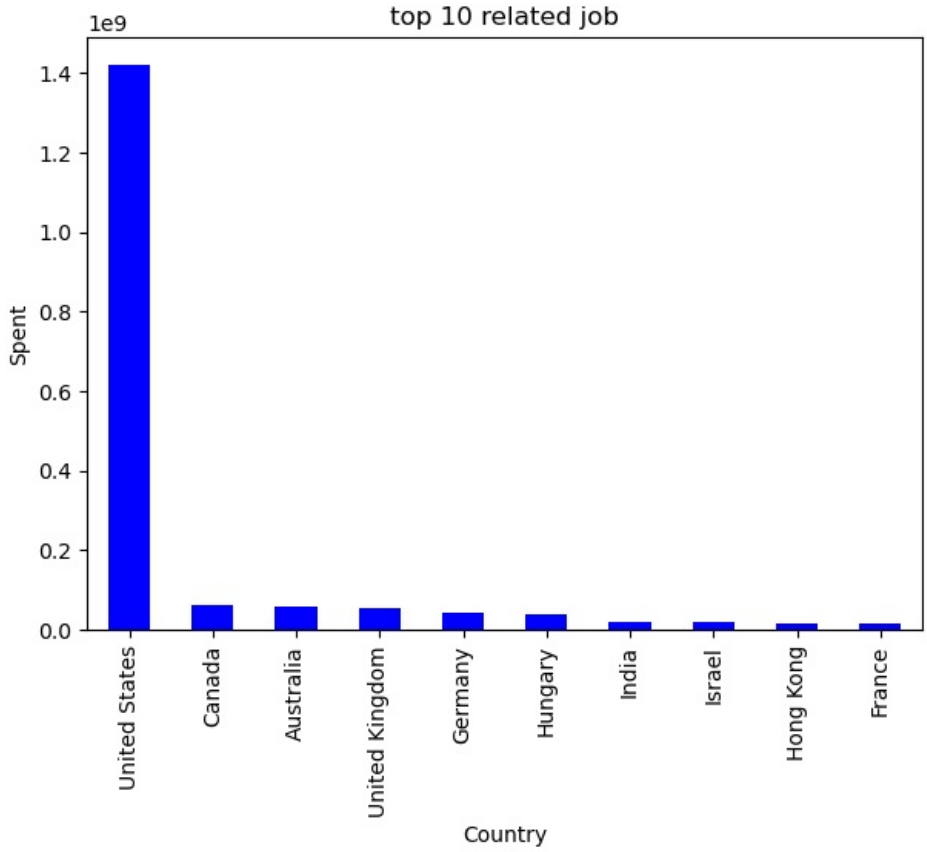
In [82]:
```
data = df.groupby('Search_Keyword')[['Connects_Num', 'New_Connects_Num']].sum()
data.plot(kind='bar', stacked=True, figsize=(10, 6))
plt.xlabel('Search_Keyword')
plt.ylabel('Connects_Num')
plt.title('Comparison of Keywords with Connect Numbers and New Connect Numbers')
plt.legend()
plt.tight_layout()
plt.show()
```



Stacked Bar chart shows the Comparison of Keywords with Connect Numbers and New Connect Numbers". This change affect how users allocate their connects when applying for jobs. For instance, applying for a "Developer" job will now require more connects than before.

Column-Spent(Reflects how much the client spent on the site since he joined. (the more client spends; the more his job is attractive because he's usually serious and generous).

In [83]:
```
spent_country_wise =df.groupby("Client_Country")["Spent($)"].sum().sort_values(ascending=False).head(10)
plt.figure(figsize=(7, 5))
spent_country_wise.plot(kind="bar",color="blue")
plt.xlabel('Country')
plt.ylabel('Spent')
plt.title('top 10 related job')
plt.show()
```



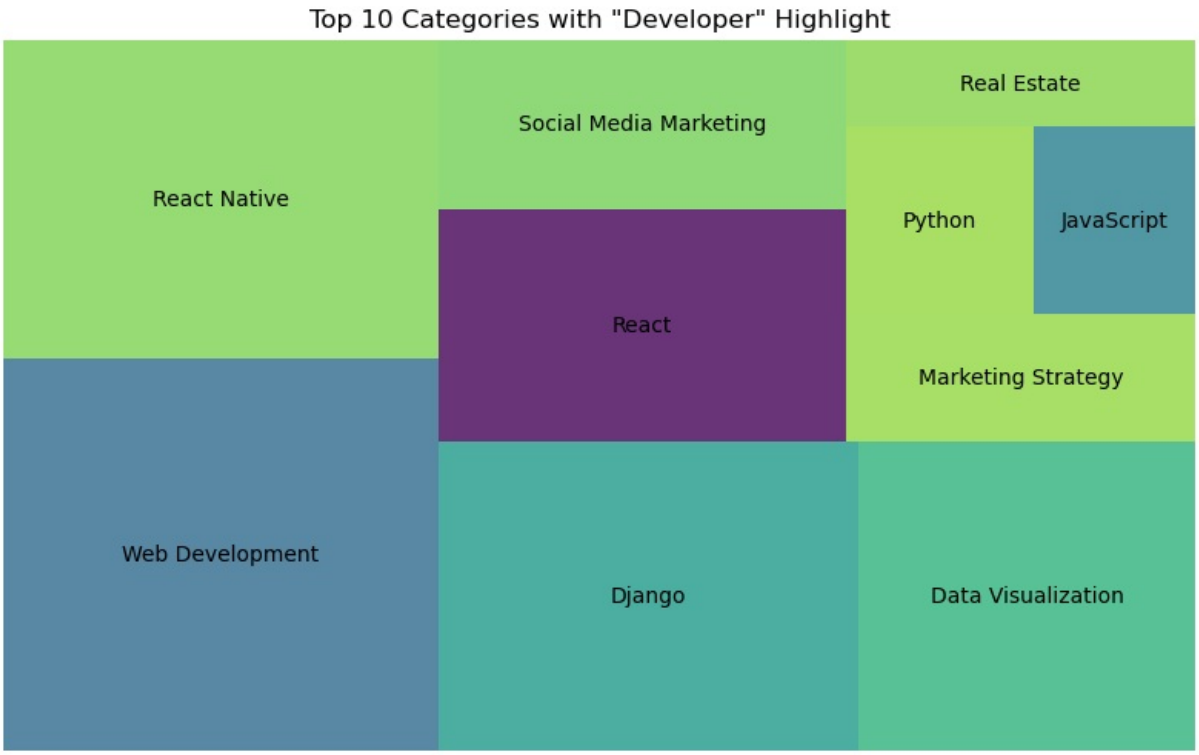
The graph represents the amount spent by clients from different countries on a site. This suggests that clients from the United States are potentially more attractive for the site as they tend to spend more. The more a client spends, the more attractive their job is because they're usually serious and generous.

In [84]:
```
spent_country_wise =df.groupby("Job Title")["Spent($)"].sum().sort_values(ascending=False).head(10)
spent_country_wise
```

Out[84]:
```
Job Title
Expert in Data Analytics                                  70000000.0
Real Estate Sales Agent                                   16000000.0
Social Media Manager and Content Creator                  14440300.0
Shopify UI/UX Developer/Designer                          12700000.0
SC- Utility AM and Metering Improvement                   10000000.0
React Native Developer ( ui-fe-1 | 37655 )                10000000.0
Digital Marketing / Google Ads & Analytics Specialist     10000000.0
Django Developer (ycon-be-4 | 24189)                      10000000.0
Django Developer (mf-be-2 | 36331)                        10000000.0
Django Developer (ui-be-1 |  37655)                       10000000.0
Name: Spent($), dtype: float64
```

This value is used as an indicator of the attractiveness of a job. The idea is that the more a client spends, the more attractive their job postings are likely to be. This is because higher spending is often associated with seriousness and generosity on the part of the client. In other words, clients who spend more are often more committed and willing to invest in getting the job done, which can make their jobs more appealing to potential candidates.

In [85]:
```
filtered_df = df[df['Category_1'] != "No category"]
spent_country_wise =filtered_df.groupby("Category_1")["Spent($)"].sum().sort_values(ascending=False).head(10)
plt.figure(figsize=(10, 6))
squarify.plot(sizes=spent_country_wise.values,
              label=spent_country_wise.index,
              alpha=0.8)
plt.axis('off')
plt.title('Top 10 Categories with "Developer" Highlight')
plt.show()
```



Treemap shows top 10 skills for which clients spent more.

Column-Payment_type(there's 2 types that clients paying for the project "Fixed-price" for the whole project at once & "Hourly" it counts every working hour and pay depends on the price per hour)

In [86]:
```
Payment_type_count=df["Payment_type"].value_counts()
Payment_type_count
```

Out[86]:
```
Payment_type
Hourly         40333
Fixed-price    23616
No payment         1
Name: count, dtype: int64
```

Payment type provides how clients pay for project tio the freelancers. Output shows that clients are more interest to pay hourly than fixed to the freelancers.

In [87]:
```
hourly_jobs_df = df[df['Payment_type'] == 'Hourly']
hourly_jobs_country_wise = hourly_jobs_df.groupby('Client_Country').size().sort_values(ascending=False).head(10
print(hourly_jobs_country_wise)
```

```
Client_Country
United States           19217
United Kingdom           2647
India                    2559
Canada                   2142
Australia                1872
Pakistan                  885
Germany                   799
United Arab Emirates      640
Netherlands               592
France                    480
dtype: int64
```

United states provides most hourly payment jobs to freelancers than Indias , UK.

In [88]:
```
hourly_jobs_df = df[df['Payment_type'] == 'Hourly']
hourly_jobs_country_wise = hourly_jobs_df.groupby('Job Title').size().sort_values(ascending=False).head(10)
print(hourly_jobs_country_wise)
```

```
Job Title
Social Media Manager                                                184
Social Media Marketing                                               91
Full Stack Developer                                                 64
Build responsive WordPress site with booking/payment functionality   42
Web Developer                                                        40
Lead Generation                                                      36
Facebook Digital Marketing                                           35
Facebook ad specialist needed for product launch                     35
Virtual Assistant                                                    35
WordPress Developer                                                  28
dtype: int64
```

Output shows top 10 hourly payment jobs

In [89]:
```
fixed_jobs_df = df[df['Payment_type'] == 'Fixed-price']
fixed_jobs_country_wise =fixed_jobs_df.groupby('Client_Country').size().sort_values(ascending=False).head(10)
print(fixed_jobs_country_wise)
```

```
Client_Country
United States           7482
India                   3175
United Kingdom          1706
Canada                  1128
Pakistan                 956
Australia                911
United Arab Emirates     508
Germany                  490
Romania                  333
Netherlands              290
dtype: int64
```

United states also provides most fixed payment jobs to freelancers than Indias , UK.

In [90]:
```
fixed_jobs_df = df[df['Payment_type'] == 'Fixed-price']
fixed_jobs_country_wise = fixed_jobs_df.groupby('Job Title').size().sort_values(ascending=False).head(10)
print(fixed_jobs_country_wise)
```

```
Job Title
Social Media Manager                                                  45
Social Media Marketing                                                43
Website Developer                                                     26
Build responsive WordPress site with booking/payment functionality    25
Facebook Ads For Multiple Local Businesses                            24
Web Developer                                                         23
I need app publishers                                                 22
Closer for Marketing Agency                                           22
Cold calling                                                          21
Website                                                               20
dtype: int64
```

Output shows top 10 fixed payment jobs

Column- Job_Cost(it's for "Fixed-price" projects; The projected budget can help freelancers gauge the project's scale and value).

In [91]:
```
job_cost_category=df.groupby("Category_1")["Job_Cost"].sum().sort_values(ascending=False).head(10)
job_cost_category
```

Out[91]:
```
Category_1
Web Development           1535671.0
Social Media Marketing     860348.0
Marketing Strategy         310932.0
Mobile App Development     215596.0
3D Modeling                210380.0
Email Marketing            209413.0
Python                     142700.0
3D Design                  139196.0
3D Rendering               129664.0
Market Research            117982.0
Name: Job_Cost, dtype: float64
```

The “Job_Cost” column represents the projected budget for different job categories under fixed-price projects. These values are important as they help freelancers understand the scale and value of the projects in each category. The budget for web development jobs are very high, so that the application no of web development is high .

In [92]:
```
job_cost_title=df.groupby("Job Title")["Job_Cost"].sum().sort_values(ascending=False).head(10)
job_cost_title
```

Out[92]:
```
Job Title
Social Media Manager                                                  39789.0
Social Media Marketing                                                23186.0
Full Stack Developer                                                  12551.0
Web Developer                                                         11865.0
Build responsive WordPress site with booking/payment functionality     9511.0
Facebook ad specialist needed for product launch                       9407.0
Website Developer                                                      9109.0
Marketing                                                              7930.0
WordPress Developer                                                    7889.0
Lead Generation                                                        7502.0
Name: Job_Cost, dtype: float64
```

Output shows the toop 10 high budget job roles.

In [93]:
```
ratings_greater_than_4 = df[df['Rating'] > 4]
count_ratings_greater_than_4 = len(ratings_greater_than_4)
print("Number of ratings greater than 4:", count_ratings_greater_than_4)
```

```
Number of ratings greater than 4: 62727
```

In [94]:
```
job_cost_title=df.groupby("Job Title")["Rating"].value_counts().sort_values(ascending=False).head(10)
job_cost_title
```

Out[94]:
```
Job Title                                                          Rating
Social Media Manager                                               4.823875    146
Social Media Marketing                                             4.823875    106
Build responsive WordPress site with booking/payment functionality 4.823875     64
Full Stack Developer                                               4.823875     57
Facebook ad specialist needed for product launch                   4.823875     45
Web Developer                                                      4.823875     44
Lead Generation                                                    4.823875     39
Website                                                            4.823875     37
Facebook Digital Marketing                                         4.823875     35
Social Media Manager                                               5.000000     30
Name: count, dtype: int64
```

Based on all analysis, it is evident that the demand for developers is

significantly high in the freelance job market. This trend is particularly noticeable on platforms Upwork, where the majority of job postings are targeted towards developers. Following developers, the next most sought-after professionals are those in the marketing field.The job market appears to favor those with an intermediate level of experience, indicating a balance between expertise and cost-effectiveness that employers find attractive. Geographically, the majority of job postings are from clients in the United States. This is likely due to the higher project budgets typically associated with U.S. clients, making these jobs particularly appealing to freelancers.

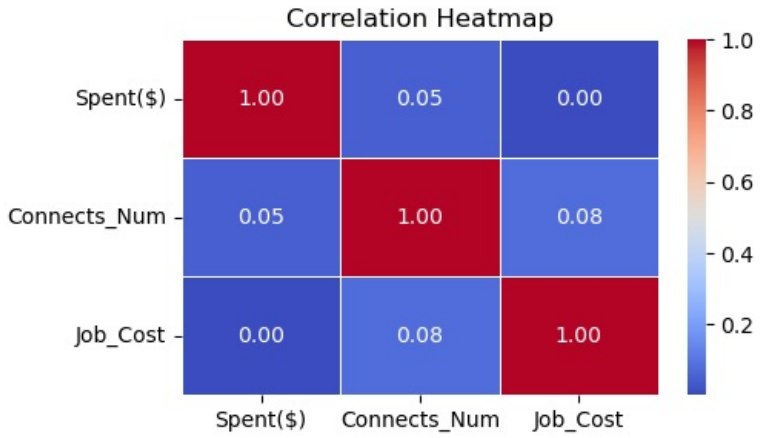
In conclusion, the freelance job market is currently thriving, especially for developers and marketers with intermediate experience levels. The U.S. dominates as the primary source of job opportunities, attracting a large number of freelancers due to the potential for higher earnings. As such, freelancers seeking to maximize their opportunities should consider focusing on developing skills in these high-demand areas and targeting U.S.-based projects.

Statistical Analysis

In [95]:
```
columns_of_interest = ['Spent($)', 'Connects_Num', 'Job_Cost']
selected_columns_df = df[columns_of_interest]
correlation_table = selected_columns_df.corr()
print("Correlation Table:")
print(correlation_table)
```

```
Correlation Table:
              Spent($)  Connects_Num  Job_Cost
Spent($)      1.000000      0.047517  0.000884
Connects_Num  0.047517      1.000000  0.076950
Job_Cost      0.000884      0.076950  1.000000
```

In [96]:
```
plt.figure(figsize=(5, 3))
sns.heatmap(data=correlation_table, annot=True, cmap='coolwarm', fmt=".2f", linewidths=.5)
plt.title('Correlation Heatmap')
plt.show()
```



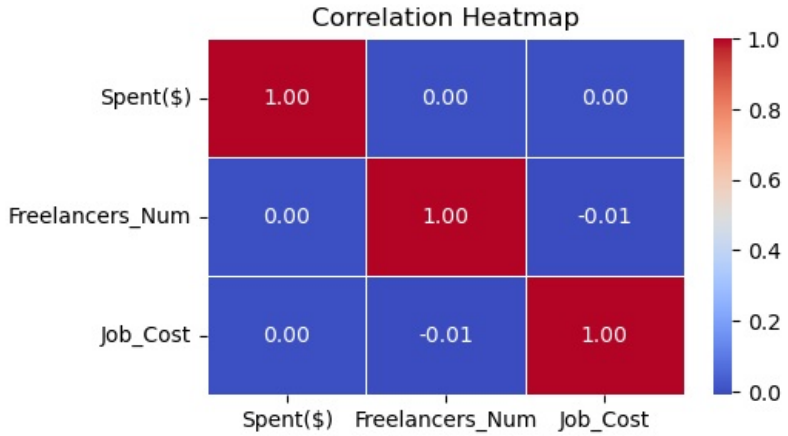
The image you provided is a correlation heatmap that shows the relationships between three variables: Spent, Connects_Num, and Job_Cost. In conclusion, there are weak correlations among the three variables. This means changes in one variable are not strongly associated with changes in the others.

In [97]:
```
columns_of_corr = ['Spent($)', 'Freelancers_Num', 'Job_Cost']
selected_columns_df = df[columns_of_corr]
correlation_table = selected_columns_df.corr()
print("Correlation Table:")
print(correlation_table)
```

```
Correlation Table:
                Spent($)  Freelancers_Num  Job_Cost
Spent($)        1.000000         0.000116  0.000884
Freelancers_Num 0.000116         1.000000 -0.010078
Job_Cost        0.000884        -0.010078  1.000000
```

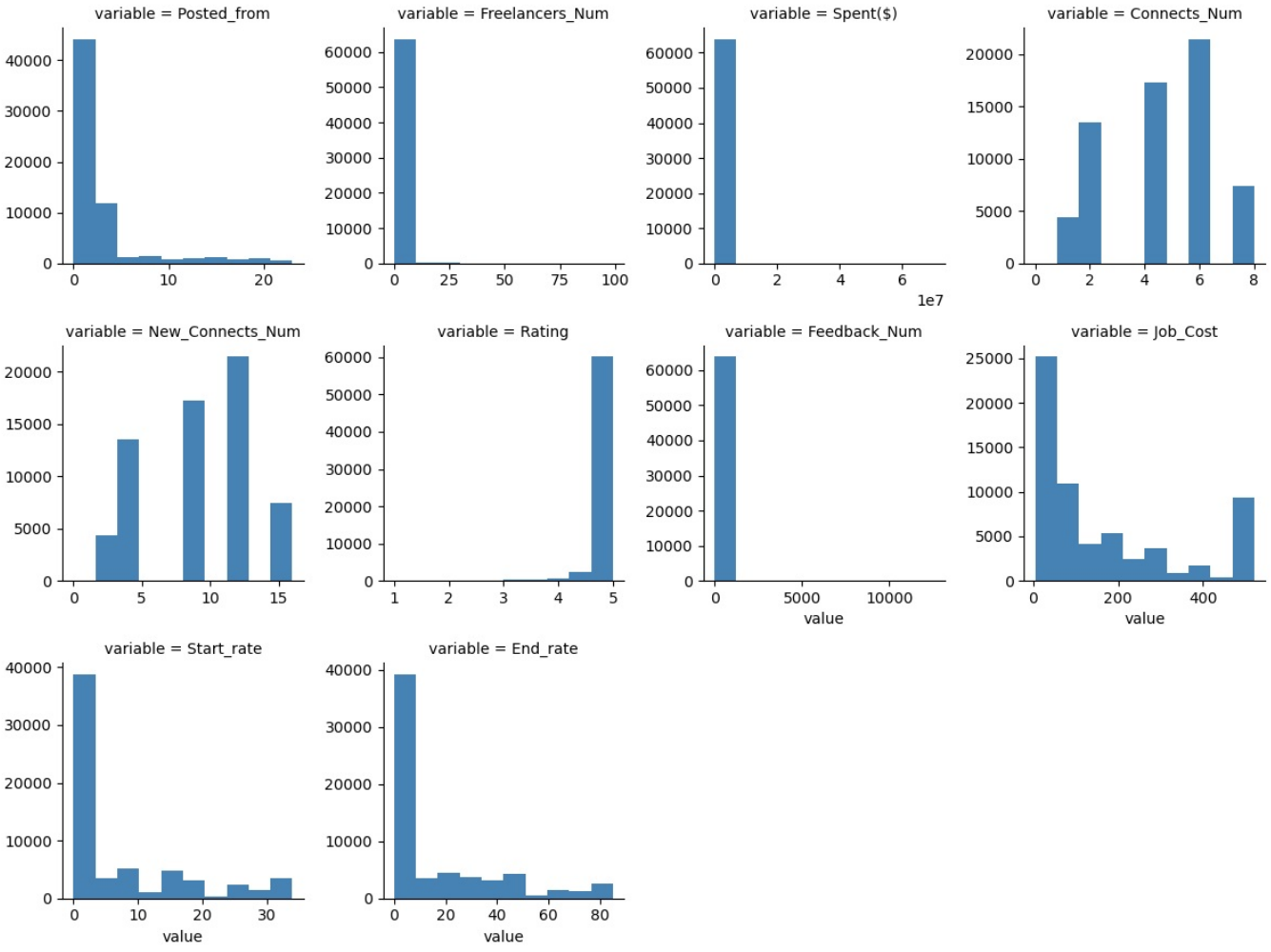
In [98]:
```
plt.figure(figsize=(5, 3))
sns.heatmap(data=correlation_table, annot=True, cmap='coolwarm', fmt=".2f", linewidths=.5)
```

```
plt.title('Correlation Heatmap')
plt.show()
```



In [99]:
```
quan=list(df.loc[:,df.dtypes!='object'].columns.values)
grid=sns.FacetGrid(pd.melt(df,value_vars=quan),col='variable',col_wrap=4,height=3,aspect=1,sharex=False,sharey=
grid.map(plt.hist,'value',color='steelblue')
plt.show()
```

```
C:\Users\NEW\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning:

The figure layout has changed to tight
```



In [100…
```
correlation = df['Start_rate'].corr(df['End_rate'])

print("Correlation between hourly_rate and end_rate:", correlation)
```

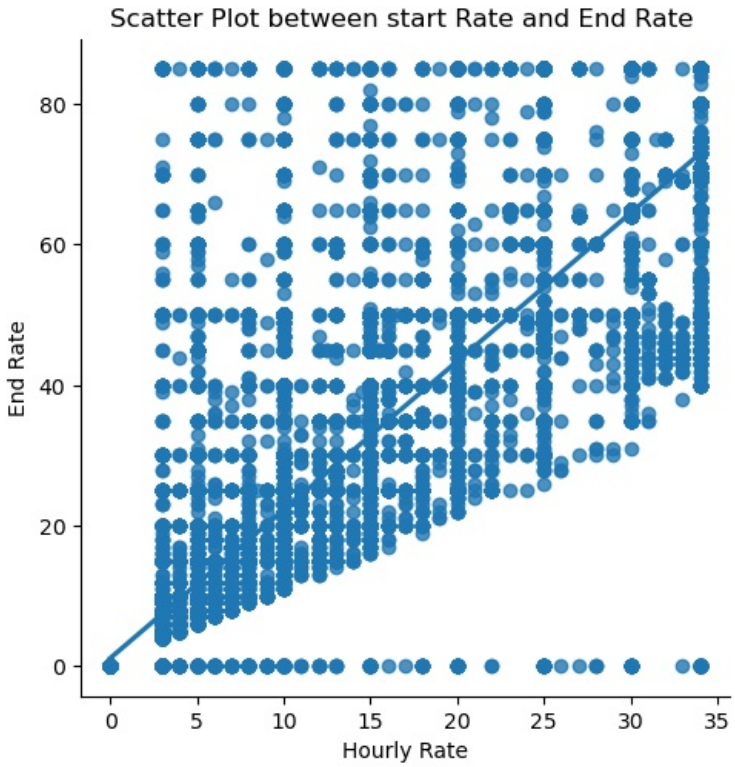
```
Correlation between hourly_rate and end_rate: 0.9146149113218072
```

The calculated correlation value is approximately 0.9146. This indicates a strong positive relationship between ‘Start_rate’ and ‘End_rate’. In other words, when ‘Start_rate’ increases, ‘End_rate’ also tends to increase, and vice versa.

In [101…
```
sns.lmplot(x='Start_rate', y='End_rate', data=df)
plt.title('Scatter Plot between start Rate and End Rate ')
plt.xlabel('Hourly Rate')
```

```
plt.ylabel('End Rate')
plt.show()
```

```
C:\Users\NEW\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning:

The figure layout has changed to tight
```

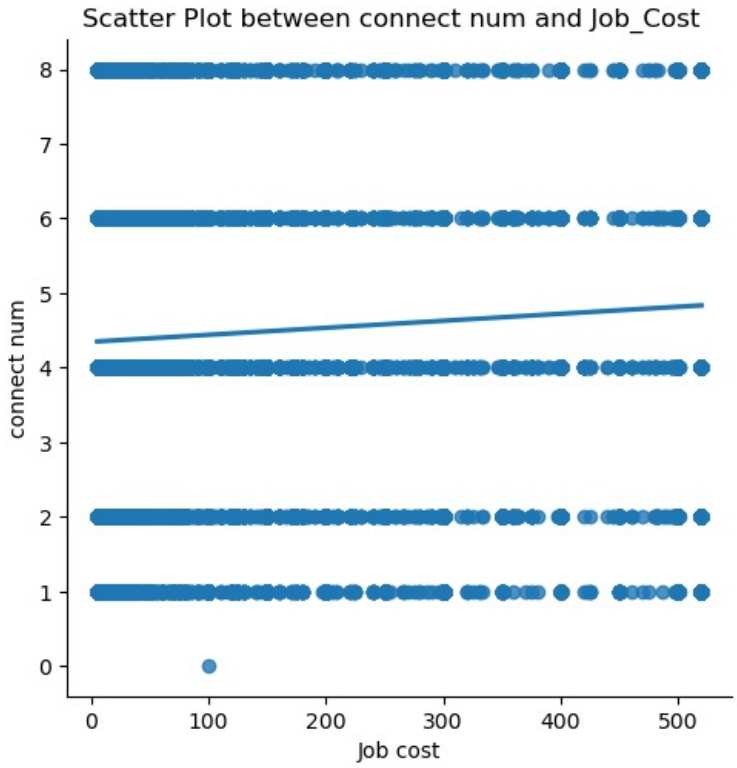


In [102…
```
correlation = df['Connects_Num'].corr(df['Job_Cost'])

print("Correlation between hourly_rate and end_rate:", correlation)
```

```
Correlation between hourly_rate and end_rate: 0.07695028860342412
```

In [103…
```
sns.lmplot(x='Job_Cost', y='Connects_Num', data=df)
plt.title('Scatter Plot between connect num and Job_Cost  ')
plt.xlabel('Job cost')
plt.ylabel('connect num')
plt.show()
```

```
C:\Users\NEW\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning:

The figure layout has changed to tight
```



In [104…
```
import statsmodels.api as sm

multinomial_logit = sm.MNLogit(df['Connects_Num'], sm.add_constant(df['Job_Cost']))
multinomial_result = multinomial_logit.fit()

# Display summary
print(multinomial_result.summary())
```

```
Optimization terminated successfully.
         Current function value: 1.477426
         Iterations 15
                          MNLogit Regression Results
==============================================================================
Dep. Variable:           Connects_Num   No. Observations:                63950
Model:                        MNLogit   Df Residuals:                    63940
Method:                           MLE   Df Model:                            5
Date:                Tue, 02 Apr 2024   Pseudo R-squ.:                0.002817
Time:                        23:25:31   Log-Likelihood:                -94481.
converged:                       True   LL-Null:                       -94748.
Covariance Type:            nonrobust   LLR p-value:                3.827e-113
==============================================================================
Connects_Num=1       coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const              8.0606      1.294      6.230      0.000       5.525      10.596
Job_Cost           0.0026      0.008      0.317      0.751      -0.013       0.019
------------------------------------------------------------------------------
Connects_Num=2       coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const              9.2391      1.294      7.142      0.000       6.704      11.775
Job_Cost           0.0023      0.008      0.275      0.783      -0.014       0.018
------------------------------------------------------------------------------
Connects_Num=4       coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const              9.3135      1.294      7.199      0.000       6.778      11.849
Job_Cost           0.0033      0.008      0.406      0.685      -0.013       0.019
------------------------------------------------------------------------------
Connects_Num=6       coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const              9.4691      1.294      7.320      0.000       6.934      12.005
Job_Cost           0.0037      0.008      0.448      0.654      -0.012       0.020
------------------------------------------------------------------------------
Connects_Num=8       coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const              8.4434      1.294      6.526      0.000       5.908      10.979
Job_Cost           0.0035      0.008      0.425      0.671      -0.013       0.020
==============================================================================
```

The image you've uploaded is a screenshot of the results from a statistical analysis, specifically an MNLogit Regression. Here's a brief interpretation:

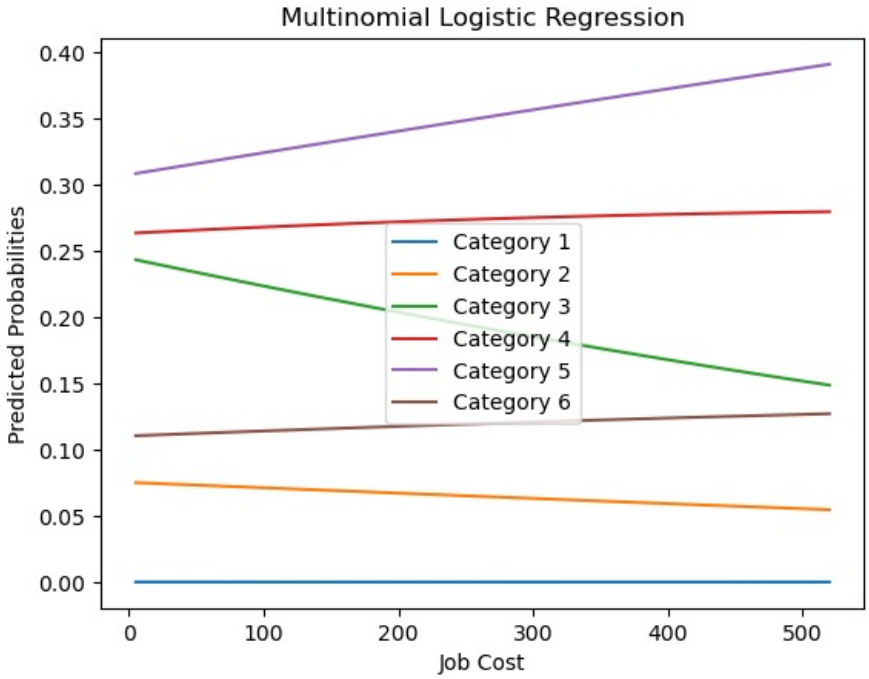
Optimization: The optimization process terminated successfully, which means the model fitting process was completed without errors.

Function Value and Iterations: The current function value and the number of iterations are also provided, which are related to the optimization process.

MNLogit Regression Results: The main part of the image shows the MNLogit Regression Results. The dependent variable is "Connects_Num". The independent variables are "const" and "Job_Cost". The results include various statistics like coefficients, standard errors, z-values, P>|z| values, and confidence intervals for different levels of "Connects_Num".

Models: There are multiple models presented for different levels of the dependent variable "Connects_Num", each with coefficients for constant and job cost.

In [107…
```
X_plot = np.linspace(min(df['Job_Cost']), max(df['Job_Cost']), 100)
predicted_probs = multinomial_result.predict(sm.add_constant(X_plot))
for i in range(predicted_probs.shape[1]):
    plt.plot(X_plot, predicted_probs[:, i], label=f'Category {i+1}')
plt.xlabel('Job Cost')
plt.ylabel('Predicted Probabilities')
plt.title('Multinomial Logistic Regression')
plt.legend()
plt.show()
```



The above plot is for predicted probabilities of different categories using Multinomial Logistic Regression, and the corresponding plot. The plot shows the predicted probabilities for six categories of 'conneect_num' as a function of job cost. Each category is represented by a line of different color. As job cost increases, the predicted probabilities for each category change.

In conclusion, the code is used to visualize how the predicted probabilities of different categories change with respect to job cost in a Multinomial Logistic Regression model. This can be useful in understanding the behavior of the model and how different factors (like job cost) can influence the predicted probabilities of different categories. The plot provides a clear visual representation of this relationship.