LINKEDIN_JOB_POSTINGS_EDA_PROJECT_VIVEK_CHAUHAN

LINKEDIN_JOB_POSTINGS_EDA_PROJECT_VIVEK_CHAUHAN ANALYSIS

The analysis is divided into four main parts:

1. Data understanding

2. Data cleaning (cleaning missing values, removing redundant columns etc.)

3. Data Analysis

4. Recommendations

In [1]:
```
# import the necessary libraries to work with dataset

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

Data Understanding

In [2]:
```
# load the dataset
```

```
data = pd.read_csv("Linkedin_job_posting_data_2023-2024.csv")
data
```

Out[2]:

	job_id	company_name	title	description	max_salary	pay_period	location	company_id	views	m
0	921716	Corcoran Sawyer Smith	Marketing Coordinator	Job descriptionA leading real estate firm in N...	20.0	HOURLY	Princeton, NJ	2774458.0	20.0	
1	1829192	NaN	Mental Health Therapist/Counselor	At Aspen Therapy and Wellness , we are committ...	50.0	HOURLY	Fort Collins, CO	NaN	1.0	
2	10998357	The National Exemplar	Assitant Restaurant Manager	The National Exemplar is accepting application...	65000.0	YEARLY	Cincinnati, OH	64896719.0	8.0	
3	23221523	Abrams Fensterman, LLP	Senior Elder Law / Trusts and Estates Associat...	Senior Associate Attorney - Elder Law / Trusts...	175000.0	YEARLY	New Hyde Park, NY	766262.0	16.0	
4	35982263	NaN	Service Technician	Looking for HVAC service tech with experience ...	80000.0	YEARLY	Burlington, IA	NaN	3.0	
...	...	...	...	...	...	...	...	...	...	
123844	3906267117	Lozano Smith	Title IX/Investigations Attorney	Our Walnut Creek office is currently seeking a...	195000.0	YEARLY	Walnut Creek, CA	56120.0	1.0	
123845	3906267126	Pinterest	Staff Software Engineer, ML Serving Platform	About Pinterest:\n\nMillions of people across ...	NaN	NaN	United States	1124131.0	3.0	
123846	3906267131	EPS Learning	Account Executive, Oregon/Washington	Company Overview\n\nEPS Learning is a leading ...	NaN	NaN	Spokane, WA	90552133.0	3.0	

	job_id	company_name	title	description	max_salary	pay_period	location	company_id	views	m
123847	3906267195	Trelleborg Applied Technologies	Business Development Manager	The Business Development Manager is a 'hunter'...	NaN	NaN	Texas, United States	2793699.0	4.0	
123848	3906267224	Solugenix	Marketing Social Media Specialist	Marketing Social Media Specialist - $70k–75...$	75000.0	YEARLY	San Juan Capistrano, CA	43325.0	2.0	

123849 rows × 31 columns

In [3]:
```
# print the first 5 rows from the dataset

data.head(5)
```

Out[3]:

	job_id	company_name	title	description	max_salary	pay_period	location	company_id	views	med_salary	...	
0	921716	Corcoran Sawyer Smith	Marketing Coordinator	Job descriptionA leading real estate firm in N...	20.0	HOURLY	Princeton, NJ	2774458.0	20.0	NaN	...	Re
1	1829192	NaN	Mental Health Therapist/Counselor	At Aspen Therapy and Wellness , we are committ...	50.0	HOURLY	Fort Collins, CO	NaN	1.0	NaN	...	
2	10998357	The National Exemplar	Assitant Restaurant Manager	The National Exemplar is accepting application...	65000.0	YEARLY	Cincinnati, OH	64896719.0	8.0	NaN	...	
3	23221523	Abrams Fensterman, LLP	Senior Elder Law / Trusts and Estates Associat...	Senior Associate Attorney - Elder Law / Trusts...	175000.0	YEARLY	New Hyde Park, NY	766262.0	16.0	NaN	...	ur
4	35982263	NaN	Service Technician	Looking for HVAC service tech with experience ...	80000.0	YEARLY	Burlington, IA	NaN	3.0	NaN	...	

5 rows × 31 columns

In [4]:

```
# print the last 5 rows from the dataset

data.tail(5)
```

Out[4]:

	job_id	company_name	title	description	max_salary	pay_period	location	company_id	views	m
123844	3906267117	Lozano Smith	Title IX/Investigations Attorney	Our Walnut Creek office is currently seeking a...	195000.0	YEARLY	Walnut Creek, CA	56120.0	1.0	
123845	3906267126	Pinterest	Staff Software Engineer, ML Serving Platform	About Pinterest:\n\nMillions of people across ...	NaN	NaN	United States	1124131.0	3.0	
123846	3906267131	EPS Learning	Account Executive, Oregon/Washington	Company Overview\n\nEPS Learning is a leading ...	NaN	NaN	Spokane, WA	90552133.0	3.0	
123847	3906267195	Trelleborg Applied Technologies	Business Development Manager	The Business Development Manager is a 'hunter'...	NaN	NaN	Texas, United States	2793699.0	4.0	
123848	3906267224	Solugenix	Marketing Social Media Specialist	Marketing Social Media Specialist - $70k$–75...	75000.0	YEARLY	San Juan Capistrano, CA	43325.0	2.0	

5 rows × 31 columns

In [5]:
```
# print the rows and columns in the dataset

data.shape
```

Out[5]: (123849, 31)

In [6]:
```
# print the duplicated rows and columns if any

data.duplicated().sum()
```

Out[6]: 0

In [7]:
```
# print the statistics about the dataset

data.describe().T
```

Out[7]:

	count	mean	std	min	25%	50%	75%	max
job_id	123849.0	3.896402e+09	8.404355e+07	9.217160e+05	3.894587e+09	3.901998e+09	3.904707e+09	3.906267e+09
max_salary	29793.0	9.193942e+04	7.011101e+05	1.000000e+00	4.828000e+01	8.000000e+04	1.400000e+05	1.200000e+08
company_id	122132.0	1.220401e+07	2.554143e+07	1.009000e+03	1.435200e+04	2.269650e+05	8.047188e+06	1.034730e+08
views	122160.0	1.461825e+01	8.590360e+01	1.000000e+00	3.000000e+00	4.000000e+00	8.000000e+00	9.975000e+03
med_salary	6280.0	2.201562e+04	5.225587e+04	0.000000e+00	1.894000e+01	2.550000e+01	2.510500e+03	7.500000e+05
min_salary	29793.0	6.491085e+04	4.959738e+05	1.000000e+00	3.700000e+01	6.000000e+04	1.000000e+05	8.500000e+07
applies	23320.0	1.059198e+01	2.904739e+01	1.000000e+00	1.000000e+00	3.000000e+00	8.000000e+00	9.670000e+02
original_listed_time	123849.0	1.713152e+12	4.848209e+08	1.701811e+12	1.712863e+12	1.713395e+12	1.713478e+12	1.713573e+12
remote_allowed	15246.0	1.000000e+00	0.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00
expiry	123849.0	1.716213e+12	2.321394e+09	1.712903e+12	1.715481e+12	1.716042e+12	1.716088e+12	1.729125e+12
closed_time	1073.0	1.712928e+12	3.622893e+08	1.712346e+12	1.712670e+12	1.712670e+12	1.713283e+12	1.713562e+12
listed_time	123849.0	1.713204e+12	3.989122e+08	1.711317e+12	1.712886e+12	1.713408e+12	1.713484e+12	1.713573e+12
sponsored	123849.0	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
normalized_salary	36073.0	2.053270e+05	5.097627e+06	0.000000e+00	5.200000e+04	8.150000e+04	1.250000e+05	5.356000e+08
zip_code	102977.0	5.040049e+04	3.025223e+04	1.001000e+03	2.411200e+04	4.805900e+04	7.820100e+04	9.990100e+04
fips	96434.0	2.871388e+04	1.601593e+04	1.003000e+03	1.312100e+04	2.918300e+04	4.207700e+04	5.604500e+04

In [8]:
```
# let's check the duplicate values which is present in the dataset by job-id

data.job_id.duplicated().sum()
```

Out[8]: 0

In [9]:
```
# print the datatypes of the columns

data.dtypes
```

Out[9]:
```
job_id                          int64
company_name                   object
title                          object
description                    object
max_salary                    float64
pay_period                     object
location                       object
company_id                    float64
views                         float64
med_salary                    float64
min_salary                    float64
formatted_work_type            object
applies                       float64
original_listed_time          float64
remote_allowed                float64
job_posting_url                object
application_url                object
application_type               object
expiry                        float64
closed_time                   float64
formatted_experience_level     object
skills_desc                    object
listed_time                   float64
posting_domain                 object
sponsored                       int64
work_type                      object
currency                       object
compensation_type              object
normalized_salary             float64
zip_code                      float64
fips                          float64
dtype: object
```

In [10]:
```
# let's count the null values which is present in the dataset

data.isnull().sum().sort_values(ascending = False)
```

Out[10]:
```
closed_time                  122776
skills_desc                  121410
med_salary                   117569
remote_allowed               108603
applies                      100529
max_salary                    94056
min_salary                    94056
currency                      87776
compensation_type             87776
normalized_salary             87776
pay_period                    87776
posting_domain                39968
application_url               36665
formatted_experience_level    29409
fips                          27415
zip_code                      20872
company_name                   1719
company_id                     1717
views                          1689
description                       7
work_type                         0
sponsored                         0
job_id                            0
listed_time                       0
expiry                            0
application_type                  0
original_listed_time              0
formatted_work_type               0
location                          0
title                             0
job_posting_url                   0
dtype: int64
```

Data Cleaning

In [11]:
```
# let's count the missing percentage of the column that contains null values

missing_percentage = (data.isnull().sum()/len(data)) * 100
missing_percentage.sort_values(ascending = False)
```

Out[11]:
```
closed_time                   99.133622
skills_desc                   98.030666
med_salary                    94.929309
remote_allowed                87.689848
applies                       81.170619
max_salary                    75.944093
min_salary                    75.944093
currency                      70.873402
compensation_type             70.873402
normalized_salary             70.873402
pay_period                    70.873402
posting_domain                32.271556
application_url               29.604599
formatted_experience_level    23.745852
fips                          22.135827
zip_code                      16.852780
company_name                   1.387981
company_id                     1.386366
views                          1.363757
description                    0.005652
work_type                      0.000000
sponsored                      0.000000
job_id                         0.000000
listed_time                    0.000000
expiry                         0.000000
application_type               0.000000
original_listed_time           0.000000
formatted_work_type            0.000000
location                       0.000000
title                          0.000000
job_posting_url                0.000000
dtype: float64
```

In [12]:
```
# drop the columns cause 90 % plus missing values containg that columns so don't waist your time in analysis
```

```
cols_to_drop = missing_percentage[missing_percentage >= 90].index
drop_data = data.drop(columns=cols_to_drop)
drop_data
```

Out[12]:

	job_id	company_name	title	description	max_salary	pay_period	location	company_id	views	m
0	921716	Corcoran Sawyer Smith	Marketing Coordinator	Job descriptionA leading real estate firm in N...	20.0	HOURLY	Princeton, NJ	2774458.0	20.0	
1	1829192	NaN	Mental Health Therapist/Counselor	At Aspen Therapy and Wellness , we are committ...	50.0	HOURLY	Fort Collins, CO	NaN	1.0	
2	10998357	The National Exemplar	Assitant Restaurant Manager	The National Exemplar is accepting application...	65000.0	YEARLY	Cincinnati, OH	64896719.0	8.0	
3	23221523	Abrams Fensterman, LLP	Senior Elder Law / Trusts and Estates Associat...	Senior Associate Attorney - Elder Law / Trusts...	175000.0	YEARLY	New Hyde Park, NY	766262.0	16.0	
4	35982263	NaN	Service Technician	Looking for HVAC service tech with experience ...	80000.0	YEARLY	Burlington, IA	NaN	3.0	
...	...	...	...	...	...	...	...	...	...	
123844	3906267117	Lozano Smith	Title IX/Investigations Attorney	Our Walnut Creek office is currently seeking a...	195000.0	YEARLY	Walnut Creek, CA	56120.0	1.0	
123845	3906267126	Pinterest	Staff Software Engineer, ML Serving Platform	About Pinterest:\n\nMillions of people across ...	NaN	NaN	United States	1124131.0	3.0	
123846	3906267131	EPS Learning	Account Executive, Oregon/Washington	Company Overview\n\nEPS Learning is a leading ...	NaN	NaN	Spokane, WA	90552133.0	3.0	
123847	3906267195	Trelleborg Applied Technologies	Business Development Manager	The Business Development Manager is a 'hunter'...	NaN	NaN	Texas, United States	2793699.0	4.0	

	job_id	company_name	title	description	max_salary	pay_period	location	company_id	views	m
123848	3906267224	Solugenix	Marketing Social Media Specialist	Marketing Social Media Specialist - $70k$–75...	75000.0	YEARLY	San Juan Capistrano, CA	43325.0	2.0	

123849 rows × 28 columns

In [13]:

```
# let's drop the unwanted columns so work is easy for analysis
# job-id,compony-id,posting domain,currency is us dollar couse whole dataset containg usa,zip-cods,and fips codes,application_
# sponsored column contains only 0 values that's why drop the whole column

new_data = drop_data.drop(columns = ['job_id','company_id','posting_domain','currency','zip_code','fips','job_posting_url','ap
new_data
```

Out[13]:

	company_name	title	description	max_salary	pay_period	location	views	min_salary	formatted_work_
0	Corcoran Sawyer Smith	Marketing Coordinator	Job descriptionA leading real estate firm in N...	20.0	HOURLY	Princeton, NJ	20.0	17.0	Full-
1	NaN	Mental Health Therapist/Counselor	At Aspen Therapy and Wellness , we are committ...	50.0	HOURLY	Fort Collins, CO	1.0	30.0	Full-
2	The National Exemplar	Assitant Restaurant Manager	The National Exemplar is accepting application...	65000.0	YEARLY	Cincinnati, OH	8.0	45000.0	Full-
3	Abrams Fensterman, LLP	Senior Elder Law / Trusts and Estates Associat...	Senior Associate Attorney - Elder Law / Trusts...	175000.0	YEARLY	New Hyde Park, NY	16.0	140000.0	Full-
4	NaN	Service Technician	Looking for HVAC service tech with experience ...	80000.0	YEARLY	Burlington, IA	3.0	60000.0	Full-
...	...	...	...	...	...	...	...	...	
123844	Lozano Smith	Title IX/Investigations Attorney	Our Walnut Creek office is currently seeking a...	195000.0	YEARLY	Walnut Creek, CA	1.0	120000.0	Full-
123845	Pinterest	Staff Software Engineer, ML Serving Platform	About Pinterest:\n\nMillions of people across ...	NaN	NaN	United States	3.0	NaN	Full-
123846	EPS Learning	Account Executive, Oregon/Washington	Company Overview\n\nEPS Learning is a leading ...	NaN	NaN	Spokane, WA	3.0	NaN	Full-
123847	Trelleborg Applied Technologies	Business Development Manager	The Business Development Manager is a 'hunter'...	NaN	NaN	Texas, United States	4.0	NaN	Full-

	company_name	title	description	max_salary	pay_period	location	views	min_salary	formatted_work_
123848	Solugenix	Marketing Social Media Specialist	Marketing Social Media Specialist - $70k$–75...	75000.0	YEARLY	San Juan Capistrano, CA	2.0	70000.0	Full-

123849 rows × 19 columns

In [14]:

```
# let's check again the missing percentage of the dataset

new_data.isnull().sum().sort_values(ascending = False).head(11)
```

Out[14]:

```
remote_allowed               108603
applies                      100529
min_salary                    94056
max_salary                    94056
compensation_type             87776
normalized_salary             87776
pay_period                    87776
formatted_experience_level    29409
company_name                   1719
views                          1689
description                       7
dtype: int64
```

In [15]:

```
# now this time to replace null values to "Not Mentioned" for accurate analysis

new_data['remote_allowed'] = new_data['remote_allowed'].fillna("Not-Mentioned")

new_data['compensation_type'] = new_data['compensation_type'].fillna("Not-Mentioned")

new_data['pay_period'] = new_data['pay_period'].fillna("Not-Mentioned")

new_data['formatted_experience_level'] = new_data['formatted_experience_level'].fillna("Not-Mentioned")

new_data['company_name'] = new_data['company_name'].fillna("Not-Mentioned")

avg_views = new_data['views'].mean()
```

```
new_data['views'] = new_data['views'].fillna(avg_views)

new_data['description'] = new_data['description'].fillna("Not-Mentioned")
```

In [16]:
```
# check the null values once again

new_data.isnull().sum().sort_values(ascending = False)
```

Out[16]:
```
applies                      100529
max_salary                    94056
min_salary                    94056
normalized_salary             87776
pay_period                        0
location                          0
views                             0
description                       0
formatted_work_type               0
title                             0
original_listed_time              0
remote_allowed                    0
application_type                  0
expiry                            0
formatted_experience_level        0
listed_time                       0
work_type                         0
compensation_type                 0
company_name                      0
dtype: int64
```

In [17]:
```
# let's group by with title and and rest of nan-column and we extract the mean of that nan-column and fill them by title type.

new_data['applies'] = new_data['applies'].fillna(new_data.groupby('title')['applies'].transform('mean'))
new_data['max_salary'] = new_data['max_salary'].fillna(new_data.groupby('title')['max_salary'].transform('mean'))
new_data['min_salary'] = new_data['min_salary'].fillna(new_data.groupby('title')['min_salary'].transform('mean'))
new_data['normalized_salary'] = new_data['normalized_salary'].fillna(new_data.groupby('title')['normalized_salary'].transform(
```

In [18]:
```
# globally fill the columns first if we get nan in perticular column then grouby by function returns error so fix it first

new_data['applies'] = new_data['applies'].fillna(new_data['applies'].mean())
new_data['max_salary'] = new_data['max_salary'].fillna(new_data['max_salary'].mean())
```

```
new_data['min_salary'] = new_data['min_salary'].fillna(new_data['min_salary'].mean())
new_data['normalized_salary'] = new_data['normalized_salary'].fillna(new_data['normalized_salary'].mean())
```

In [19]:
```
# cross check still is there any null values present or not

new_data.isnull().sum().sort_values(ascending = False)
```

Out[19]:
```
company_name                  0
original_listed_time          0
compensation_type             0
work_type                     0
listed_time                   0
formatted_experience_level    0
expiry                        0
application_type              0
remote_allowed                0
applies                       0
title                         0
formatted_work_type           0
min_salary                    0
views                         0
location                      0
pay_period                    0
max_salary                    0
description                   0
normalized_salary             0
dtype: int64
```

In [20]:
```
# let's print once again the datatyps of our cleaned dataset

new_data.dtypes
```

Out[20]:
```
company_name                  object
title                         object
description                   object
max_salary                   float64
pay_period                    object
location                      object
views                        float64
min_salary                   float64
formatted_work_type           object
applies                      float64
original_listed_time         float64
remote_allowed                object
application_type              object
expiry                       float64
formatted_experience_level    object
listed_time                  float64
work_type                     object
compensation_type             object
normalized_salary            float64
dtype: object
```

In [21]:
```
# again print the top 5 rows from the dataset

new_data.head(5)
```

Out[21]:

	company_name	title	description	max_salary	pay_period	location	views	min_salary	formatted_work_type	applies
0	Corcoran Sawyer Smith	Marketing Coordinator	Job descriptionA leading real estate firm in N...	20.0	HOURLY	Princeton, NJ	20.0	17.0	Full-time	2.000000
1	Not-Mentioned	Mental Health Therapist/Counselor	At Aspen Therapy and Wellness , we are committ...	50.0	HOURLY	Fort Collins, CO	1.0	30.0	Full-time	8.596711
2	The National Exemplar	Assitant Restaurant Manager	The National Exemplar is accepting application...	65000.0	YEARLY	Cincinnati, OH	8.0	45000.0	Full-time	8.596711
3	Abrams Fensterman, LLP	Senior Elder Law / Trusts and Estates Associat...	Senior Associate Attorney - Elder Law / Trusts...	175000.0	YEARLY	New Hyde Park, NY	16.0	140000.0	Full-time	8.596711
4	Not-Mentioned	Service Technician	Looking for HVAC service tech with experience ...	80000.0	YEARLY	Burlington, IA	3.0	60000.0	Full-time	8.596711

In [22]:

```
# print last 5 rows from the dataset

new_data.tail(5)
```

Out[22]:

	company_name	title	description	max_salary	pay_period	location	views	min_salary	formatted
123844	Lozano Smith	Title IX/Investigations Attorney	Our Walnut Creek office is currently seeking a...	195000.000000	YEARLY	Walnut Creek, CA	1.0	120000.000000	
123845	Pinterest	Staff Software Engineer, ML Serving Platform	About Pinterest:\n\nMillions of people across ...	86244.695942	Not-Mentioned	United States	3.0	61360.675810	
123846	EPS Learning	Account Executive, Oregon/Washington	Company Overview\n\nEPS Learning is a leading ...	86244.695942	Not-Mentioned	Spokane, WA	3.0	61360.675810	
123847	Trelleborg Applied Technologies	Business Development Manager	The Business Development Manager is a 'hunter'...	116135.216216	Not-Mentioned	Texas, United States	4.0	89918.081081	
123848	Solugenix	Marketing Social Media Specialist	Marketing Social Media Specialist - $70k$–75...	75000.000000	YEARLY	San Juan Capistrano, CA	2.0	70000.000000	

In [23]:

```
# print all the columns and rows in no.s

new_data.shape
```

Out[23]: (123849, 19)

In [24]:

```
new_data.expiry
```

Out[24]:
```
0         1.715990e+12
1         1.715450e+12
2         1.715870e+12
3         1.715488e+12
4         1.716044e+12
              ...
123844    1.716163e+12
123845    1.716164e+12
123846    1.716164e+12
123847    1.716165e+12
123848    1.716165e+12
Name: expiry, Length: 123849, dtype: float64
```

In [25]:
```
# let's convert the miliseconds to actual date

new_data['expiry'] = pd.to_datetime(new_data['expiry'], unit='ms')
```

In [26]:
```
new_data.expiry
```

Out[26]:
```
0        2024-05-17 23:45:08
1        2024-05-11 17:51:27
2        2024-05-16 14:26:54
3        2024-05-12 04:23:32
4        2024-05-18 14:52:23
                 ...
123844   2024-05-20 00:00:23
123845   2024-05-20 00:17:16
123846   2024-05-20 00:18:59
123847   2024-05-20 00:23:52
123848   2024-05-20 00:23:36
Name: expiry, Length: 123849, dtype: datetime64[ns]
```

In [27]:
```
# now again split the year,month,day and day-name and create a new column

new_data['expyear'] = new_data['expiry'].dt.year
new_data['expmonth'] = new_data['expiry'].dt.month
new_data['expday'] = new_data['expiry'].dt.day
new_data['expday-name'] = new_data['expiry'].dt.day_name()
```

In [28]:
```
# print year column

new_data.expyear
```

Out[28]:
```
0         2024
1         2024
2         2024
3         2024
4         2024
          ...
123844    2024
123845    2024
123846    2024
123847    2024
123848    2024
Name: expyear, Length: 123849, dtype: int32
```

In [29]:
```
# print month column

new_data.expmonth
```

Out[29]:
```
0         5
1         5
2         5
3         5
4         5
         ..
123844    5
123845    5
123846    5
123847    5
123848    5
Name: expmonth, Length: 123849, dtype: int32
```

In [30]:
```
# print day column

new_data.expday
```

Out[30]:
```
0         17
1         11
2         16
3         12
4         18
          ..
123844    20
123845    20
123846    20
123847    20
123848    20
Name: expday, Length: 123849, dtype: int32
```

In [31]:
```
# print day-name column

new_data['expday-name']
```

Out[31]:
```
0           Friday
1         Saturday
2         Thursday
3           Sunday
4         Saturday
            ...
123844      Monday
123845      Monday
123846      Monday
123847      Monday
123848      Monday
Name: expday-name, Length: 123849, dtype: object
```

In [32]:
```
# print the original listed column

new_data.original_listed_time
```

Out[32]:
```
0         1.713398e+12
1         1.712858e+12
2         1.713278e+12
3         1.712896e+12
4         1.713452e+12
              ...
123844    1.713571e+12
123845    1.713572e+12
123846    1.713572e+12
123847    1.713573e+12
123848    1.713573e+12
Name: original_listed_time, Length: 123849, dtype: float64
```

In [33]:
```
# let's convert the miliseconds to actual date

new_data['original_listed_time'] = pd.to_datetime(new_data['original_listed_time'], unit='ms')
```

In [34]:
```
# print the original_listed_time column

new_data.original_listed_time
```

Out[34]:
```
0        2024-04-17 23:45:08
1        2024-04-11 17:51:27
2        2024-04-16 14:26:54
3        2024-04-12 04:23:32
4        2024-04-18 14:52:23
                 ...
123844   2024-04-20 00:00:23
123845   2024-04-20 00:05:00
123846   2024-04-20 00:07:02
123847   2024-04-20 00:23:52
123848   2024-04-20 00:23:36
Name: original_listed_time, Length: 123849, dtype: datetime64[ns]
```

In [35]:
```
# now again split the year,month,day and day-name and create a new column

new_data['olistedyear'] = new_data['original_listed_time'].dt.year
new_data['olistedmonth'] = new_data['original_listed_time'].dt.month
new_data['olistedday'] = new_data['original_listed_time'].dt.day
new_data['olistedday-name'] = new_data['original_listed_time'].dt.day_name()
```

In [36]:
```
# print the olistedyear column

new_data.olistedyear
```

Out[36]:
```
0         2024
1         2024
2         2024
3         2024
4         2024
          ...
123844    2024
123845    2024
123846    2024
123847    2024
123848    2024
Name: olistedyear, Length: 123849, dtype: int32
```

In [37]:
```
# print the olistedmonth column

new_data.olistedmonth
```

Out[37]:
```
0         4
1         4
2         4
3         4
4         4
         ..
123844    4
123845    4
123846    4
123847    4
123848    4
Name: olistedmonth, Length: 123849, dtype: int32
```

In [38]:
```
# print the olistedday column

new_data.olistedday
```

Out[38]:
```
0         17
1         11
2         16
3         12
4         18
          ..
123844    20
123845    20
123846    20
123847    20
123848    20
Name: olistedday, Length: 123849, dtype: int32
```

In [39]:
```
# print the olistedday-name column

new_data['olistedday-name']
```

Out[39]:
```
0         Wednesdday
```

In [40]:
```
# let's convert the miliseconds to actual date

new_data['listed_time'] = pd.to_datetime(new_data['listed_time'], unit='ms')
```

In [41]:
```
# print the original_listed_time column

new_data.listed_time
```

Out[41]:
```
0         2024-04-17 23:45:08
1         2024-04-11 17:51:27
2         2024-04-16 14:26:54
3         2024-04-12 04:23:32
4         2024-04-18 14:52:23
                  ...
123844    2024-04-20 00:00:23
123845    2024-04-20 00:17:16
123846    2024-04-20 00:18:59
123847    2024-04-20 00:23:52
123848    2024-04-20 00:23:36
Name: listed_time, Length: 123849, dtype: datetime64[ns]
```

In [42]:
```
# now again split the year,month,day and day-name and create a new column

new_data['listedyear'] = new_data['listed_time'].dt.year
new_data['listedmonth'] = new_data['listed_time'].dt.month
new_data['listedday'] = new_data['listed_time'].dt.day
new_data['listedday-name'] = new_data['listed_time'].dt.day_name()
```

In [43]:
```
# print the listedyear column

new_data.listedyear
```

Out[43]:
```
0         2024
1         2024
2         2024
3         2024
4         2024
          ...
123844    2024
123845    2024
123846    2024
123847    2024
123848    2024
Name: listedyear, Length: 123849, dtype: int32
```

In [44]:
```
# print the listedmonth column

new_data.listedmonth
```

Out[44]:
```
0         4
1         4
2         4
3         4
4         4
         ..
123844    4
123845    4
123846    4
123847    4
123848    4
Name: listedmonth, Length: 123849, dtype: int32
```

In [45]:
```
# print the listedday column

new_data.listedday
```

Out[45]:
```
0         17
1         11
2         16
3         12
4         18
          ..
123844    20
123845    20
123846    20
123847    20
123848    20
Name: listedday, Length: 123849, dtype: int32
```

In [46]:
```
# print the listedday-name column

new_data['listedday-name']
```

Out[46]:
```
0         Wednesday
1          Thursday
2           Tuesday
3            Friday
4          Thursday
            ...
123844     Saturday
123845     Saturday
123846     Saturday
123847     Saturday
123848     Saturday
Name: listedday-name, Length: 123849, dtype: object
```

In [47]:
```
# print all the columns

new_data
```

Out[47]:

	company_name	title	description	max_salary	pay_period	location	views	min_salary	formatted
0	Corcoran Sawyer Smith	Marketing Coordinator	Job descriptionA leading real estate firm in N...	20.000000	HOURLY	Princeton, NJ	20.0	17.000000	
1	Not-Mentioned	Mental Health Therapist/Counselor	At Aspen Therapy and Wellness , we are committ...	50.000000	HOURLY	Fort Collins, CO	1.0	30.000000	
2	The National Exemplar	Assitant Restaurant Manager	The National Exemplar is accepting application...	65000.000000	YEARLY	Cincinnati, OH	8.0	45000.000000	
3	Abrams Fensterman, LLP	Senior Elder Law / Trusts and Estates Associat...	Senior Associate Attorney - Elder Law / Trusts...	175000.000000	YEARLY	New Hyde Park, NY	16.0	140000.000000	
4	Not-Mentioned	Service Technician	Looking for HVAC service tech with experience ...	80000.000000	YEARLY	Burlington, IA	3.0	60000.000000	
...	...	...	...	...	...	...	...	...	
123844	Lozano Smith	Title IX/Investigations Attorney	Our Walnut Creek office is currently seeking a...	195000.000000	YEARLY	Walnut Creek, CA	1.0	120000.000000	
123845	Pinterest	Staff Software Engineer, ML Serving Platform	About Pinterest:\n\nMillions of people across ...	86244.695942	Not-Mentioned	United States	3.0	61360.675810	
123846	EPS Learning	Account Executive, Oregon/Washington	Company Overview\n\nEPS Learning is a leading ...	86244.695942	Not-Mentioned	Spokane, WA	3.0	61360.675810	
123847	Trelleborg Applied Technologies	Business Development Manager	The Business Development	116135.216216	Not-Mentioned	Texas, United States	4.0	89918.081081	

	company_name	title	description	max_salary	pay_period	location	views	min_salary	formatted
			Manager is a 'hunter'...						
123848	Solugenix	Marketing Social Media Specialist	Marketing Social Media Specialist - $70k$–75...	75000.000000	YEARLY	San Juan Capistrano, CA	2.0	70000.000000	

123849 rows × 31 columns

In [48]:
```
# print all the final & clened dataset column names

new_data.columns
```

Out[48]:
```
Index(['company_name', 'title', 'description', 'max_salary', 'pay_period',
       'location', 'views', 'min_salary', 'formatted_work_type', 'applies',
       'original_listed_time', 'remote_allowed', 'application_type', 'expiry',
       'formatted_experience_level', 'listed_time', 'work_type',
       'compensation_type', 'normalized_salary', 'expyear', 'expmonth',
       'expday', 'expday-name', 'olistedyear', 'olistedmonth', 'olistedday',
       'olistedday-name', 'listedyear', 'listedmonth', 'listedday',
       'listedday-name'],
      dtype='object')
```

In [49]:
```
# How many columns present in the new_data dataset

len(new_data.columns)
```

Out[49]:
```
31
```

Word Frequency Count

In [50]:
```
# let's count the words in title column so we get the common word in a job position

from collections import Counter # load the counter library
```

```
from nltk.corpus import stopwords  # Load the stopwords library
import nltk

nltk.download('stopwords')  # download stopwords (only needed once)

stop_words = set(stopwords.words('english'))  # Load English stopwords

all_title_words = [] # create a blank list to store split words in it

for title in new_data['title']:    # Loop for split the words and store it in a blank list
    words = title.split()
    filtered_words = [word for word in words if word.lower() not in stop_words]  # remove stopwords
    all_title_words.extend(filtered_words)  # add filtered words to the list

title_word_freq = Counter(all_title_words)  # count the split words

print(title_word_freq.most_common(10))  # print the 10 most common words
```

```
[nltk_data] Downloading package stopwords to C:\Users\VIVEK
[nltk_data]     CHAUHAN\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[('-', 31473), ('Manager', 15087), ('Engineer', 9063), ('Sales', 8031), ('Senior', 7898), ('Specialist', 6140), ('Associate', 5
961), ('Assistant', 5709), ('Technician', 4911), ('Analyst', 4350)]
```

Top 10 most common words in the job title column is 'Manager','Engineer','Sales','Senior','Specialist','Associate','Assistant','Technician','A

In [51]:

```
# let's count the words in description column so we get the common word in a job description
# we get better idea to create ATS friendly resume for applying job position

from nltk.corpus import stopwords  # import stopwords
import nltk

stop_words = set(stopwords.words('english'))  # Load English stopwords

all_description_words = []  # create a blank list to store split words

for description in new_data['description']:  # Loop for splitting the words
```

```
    words = description.split()
    filtered_words = [word for word in words if word.lower() not in stop_words]  # remove stopwords
    all_description_words.extend(filtered_words)  # add filtered words to the list

description_word_freq = Counter(all_description_words)  # count the split words

print(description_word_freq.most_common(25))  # print the 25
```

```
[('work', 257795), ('experience', 210779), ('team', 167754), ('including', 153531), ('business', 124584), ('customer', 112487), ('&', 110836), ('-', 107962), ('support', 104537), ('years', 103156), ('may', 95263), ('management', 89380), ('position', 8844
7), ('skills', 88330), ('ability', 85159), ('new', 84903), ('working', 83743), ('within', 82518), ('required', 82009), ('care', 81704), ('company', 81671), ('related', 81093), ('sales', 79869), ('and/or', 77765), ('data', 77626)]
```

In the company job discription the common words 'work', 'experience', 'team', 'including', 'business', 'customer', 'support', 'years', 'management', 'position', 'skills', 'ability', 'sales' etc.

In [52]:

```
# let's count the words in location column so we get the common location in a job position

from collections import Counter # load the counter library
from nltk.corpus import stopwords  # import stopwords
import nltk

nltk.download('stopwords')  # download stopwords (only needed once)

stop_words = set(stopwords.words('english'))  # load English stopwords

all_location_words = []  # create a blank list to store split words

for location in new_data['location']:  # loop for splitting the words
    words = location.split()
    filtered_words = [word for word in words if word.lower() not in stop_words]  # remove stopwords
    all_location_words.extend(filtered_words)  # add filtered words to the list

location_word_freq = Counter(all_location_words)  # count the split words

print(location_word_freq.most_common(10))  # print the 10 most common words
```

```
[('United', 12695), ('States', 12695), ('CA', 11484), ('TX', 10271), ('NY', 6044), ('FL', 5907), ('New', 5561), ('NC', 4928),
('Area', 4810), ('IL', 4486)]
[nltk_data] Downloading package stopwords to C:\Users\VIVEK
[nltk_data]     CHAUHAN\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

As you know that this dataset is contains only usa details so common sense locations is 'united states' but the few of top 3 job posting locations is 'CA','TX','NY' & 'FL','NC','IL'.

Data-Analysis

Uni-Variate-Analysis

In [53]:
```
# count the frequent compony names so we get better idea about how many company posted for job postings

compony_names = new_data.company_name.value_counts().sort_values(ascending = False).head(10)
compony_names
```

Out[53]:
```
company_name
Not-Mentioned                                     1719
Liberty Healthcare and Rehabilitation Services    1108
The Job Network                                   1003
J. Galt                                            604
TEKsystems                                         529
Lowe's Companies, Inc.                             527
Ingersoll Rand                                     517
Capital One                                        496
Cogent Communications                              476
Insight Global                                     418
Name: count, dtype: int64
```

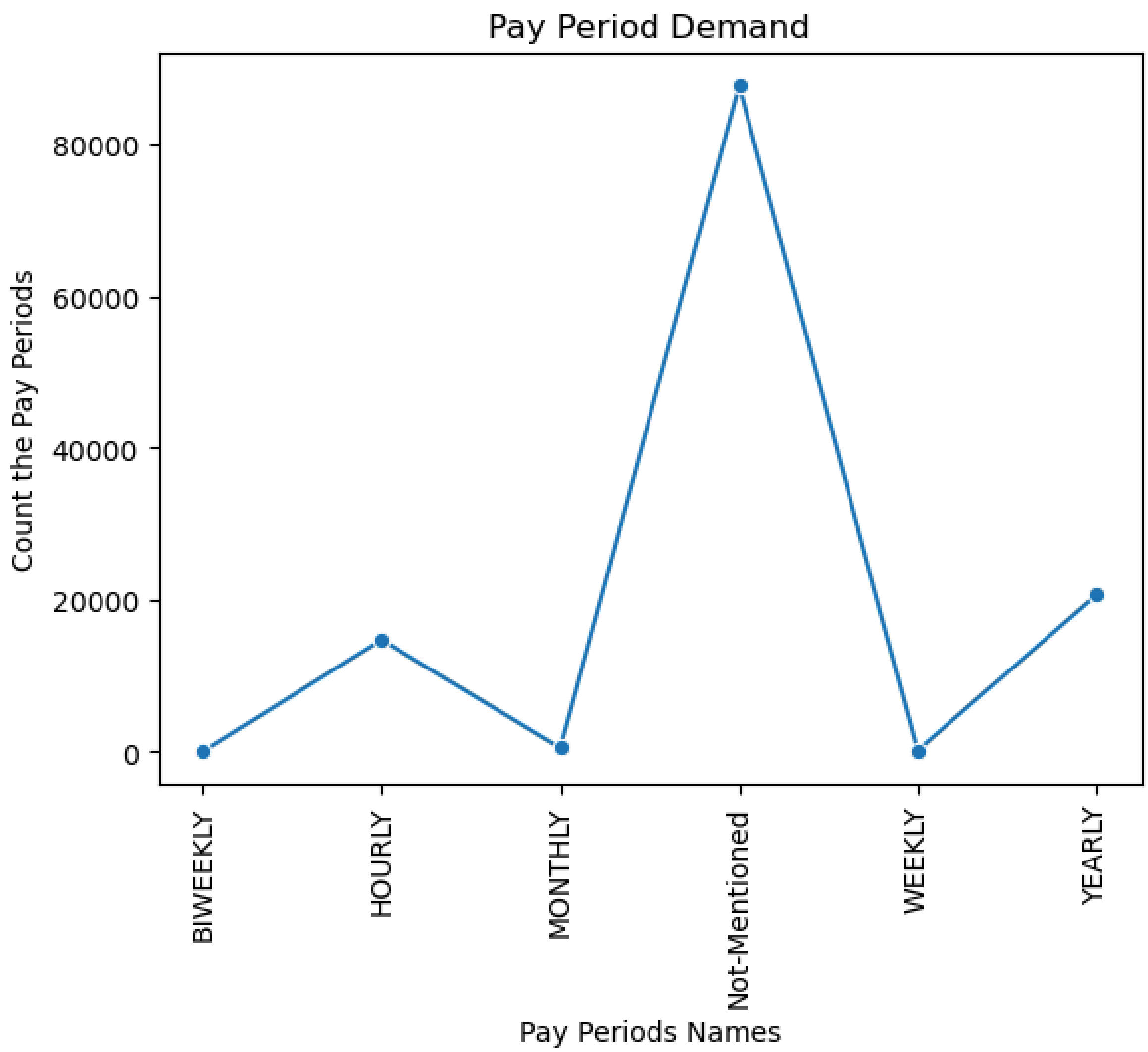
In above compony-counts you can see clearly there is 2 compony who posts frequently and company name is "Liberty Healthcare and Rehabilitation Services","The Job Network ".

In [54]:
```
# count the top 10 frequent compony title so we get better idea about which position is highest for jobs

new_data.title.value_counts().sort_values(ascending = False).head(10)
```

Out[54]:
```
title
Sales Manager                      673
Customer Service Representative    373
Project Manager                    354
Administrative Assistant           254
Senior Accountant                  238
Executive Assistant                228
Salesperson                        211
Registered Nurse                   210
Receptionist                       204
Staff Accountant                   200
Name: count, dtype: int64
```

You can see clearly in the Sales Manager role there is highest jobs openings.

In [55]:
```
pay_period_counts = new_data['pay_period'].value_counts().sort_index()

# Then plot using lineplot
sns.lineplot(x=pay_period_counts.index, y=pay_period_counts.values, marker='o')
plt.title('Pay Period Demand')
plt.xlabel('Pay Periods Names')
plt.ylabel('Count the Pay Periods')
plt.xticks(rotation=90)
plt.show()
```



In the above chart as per the accurate data there is Yearly Pay types jobs is highest on linkedin and rest of the data we have no information so "Not-Mentioned Category" is high if data is proper then game is changed.

In [56]:
```
new_data.dtypes
```

Out[56]:
```
company_name                          object
title                                 object
description                           object
max_salary                           float64
pay_period                            object
location                              object
views                                float64
min_salary                           float64
formatted_work_type                   object
applies                              float64
original_listed_time          datetime64[ns]
remote_allowed                        object
application_type                      object
expiry                        datetime64[ns]
formatted_experience_level            object
listed_time                   datetime64[ns]
work_type                             object
compensation_type                     object
normalized_salary                    float64
expyear                                int32
expmonth                               int32
expday                                 int32
expday-name                           object
olistedyear                            int32
olistedmonth                           int32
olistedday                             int32
olistedday-name                       object
listedyear                             int32
listedmonth                            int32
listedday                              int32
listedday-name                        object
dtype: object
```

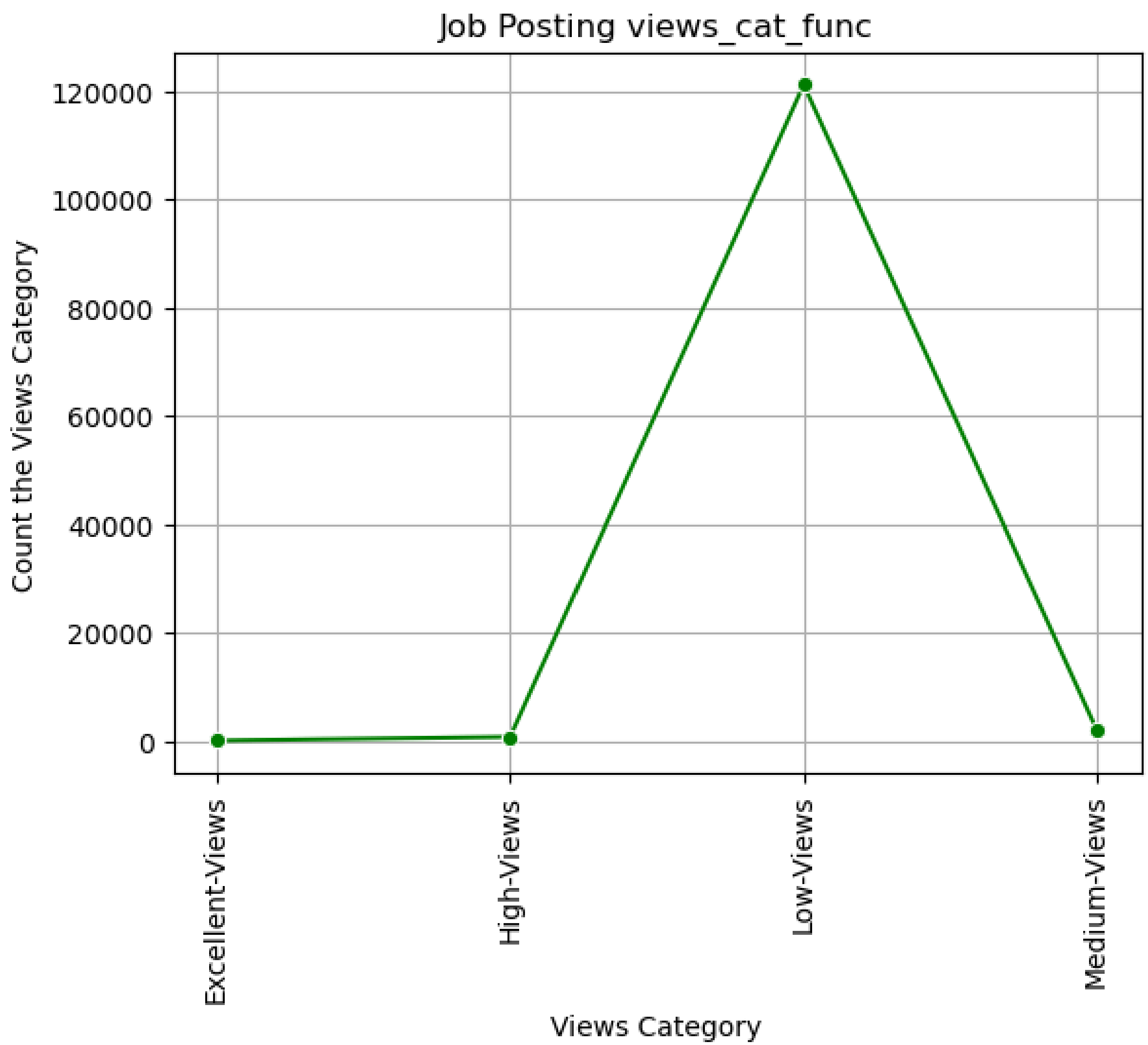
Analyse VIEWS,MIN_SALARY,MAX_SALARY,NORMALIZED_SALARY column there is missing values is replaced by the average values/mean values.

In [57]:
```
# create a category function for views columns

def views_cat_func(x):
    if(x == "Not-Mentioned"):
        return "Not-Mentioned"
    elif(x<100):
        return "Low-Views"
    elif(x>100 and x<=250):
        return "Medium-Views"
    elif(x>250 and 500):
        return "High-Views"
    else:
        return "Excellent-Views"

new_data ['views_cat_func'] = new_data['views'].apply(views_cat_func)
```

In [58]:
```
# print the views category column & count the values

new_data.views_cat_func.value_counts()
```

Out[58]:
```
views_cat_func
Low-Views          121128
Medium-Views         1974
High-Views            718
Excellent-Views        29
Name: count, dtype: int64
```

In [59]:
```
# Calculate counts
views_counts = new_data['views_cat_func'].value_counts().sort_index()

# Line plot
sns.lineplot(x=views_counts.index, y=views_counts.values, marker='o', color='green')
plt.title('Job Posting views_cat_func')
plt.xlabel('Views Category')
plt.ylabel('Count the Views Category')
plt.xticks(rotation=90)
plt.grid(True)
plt.show()
```



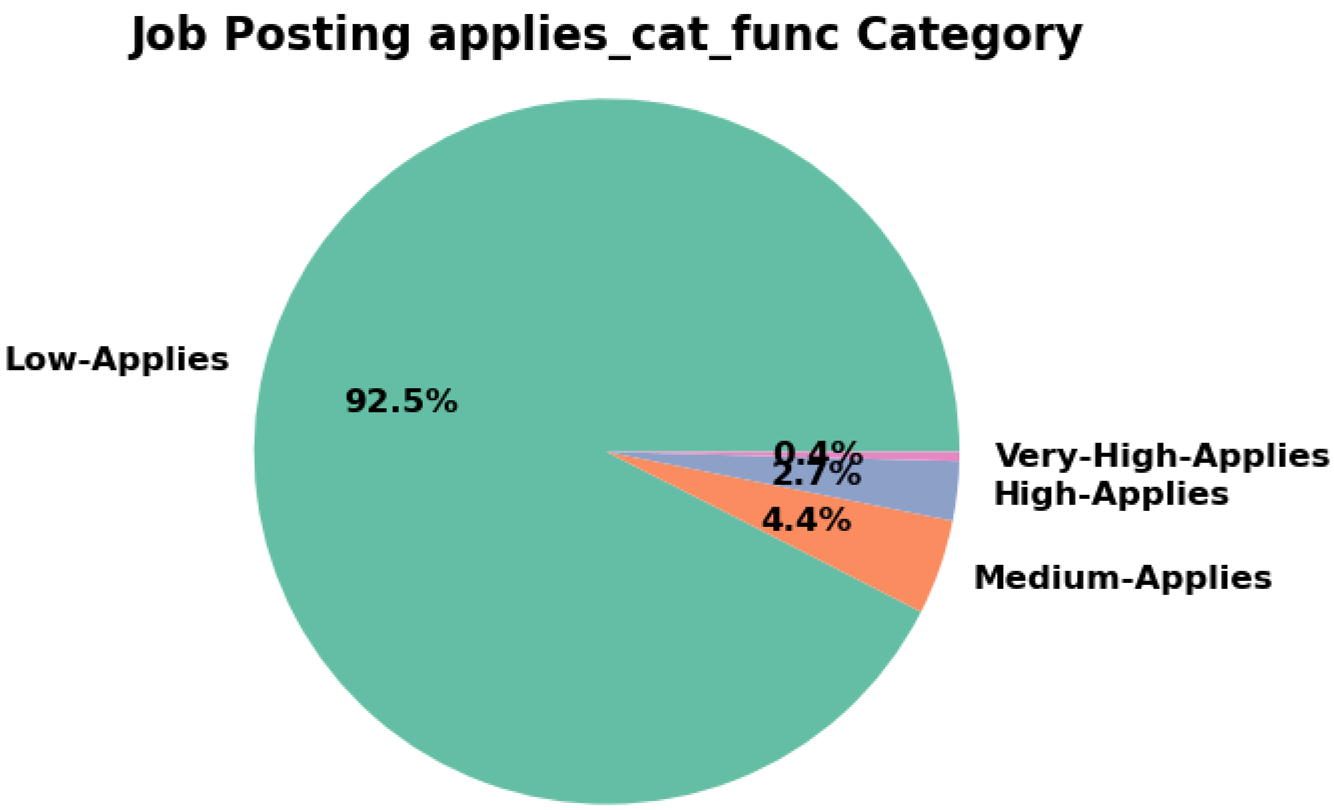
You can see clearly most of the jobs are posted on linkedin platform fall into low views category.

In [60]:
```
# create a category function for applies columns
```

```python
def applies_cat_func(x):
    if(x == "Not-Mentioned"):
        return "Not-Mentioned"
    elif(x<10):
        return "Low-Applies"
    elif(x>10 and x<=25):
        return "Medium-Applies"
    elif(x>25 and 50):
        return "High-Applies"
    else:
        return "Very-High-Applies"

new_data ['applies_cat_func'] = new_data['applies'].apply(applies_cat_func)
```

In [61]:

```python
# Calculate counts
applies_counts = new_data['applies_cat_func'].value_counts()

# Pie chart
plt.figure(figsize=(5,5))  # Optional: size thodi moti rakhi
plt.pie(applies_counts.values, labels=applies_counts.index, autopct='%1.1f%%', colors=sns.color_palette('Set2'),textprops={'fo
plt.title('Job Posting applies_cat_func Category', fontsize=16, fontweight='bold')
plt.axis('equal')  # Circle shape maintain karva
plt.show()
```

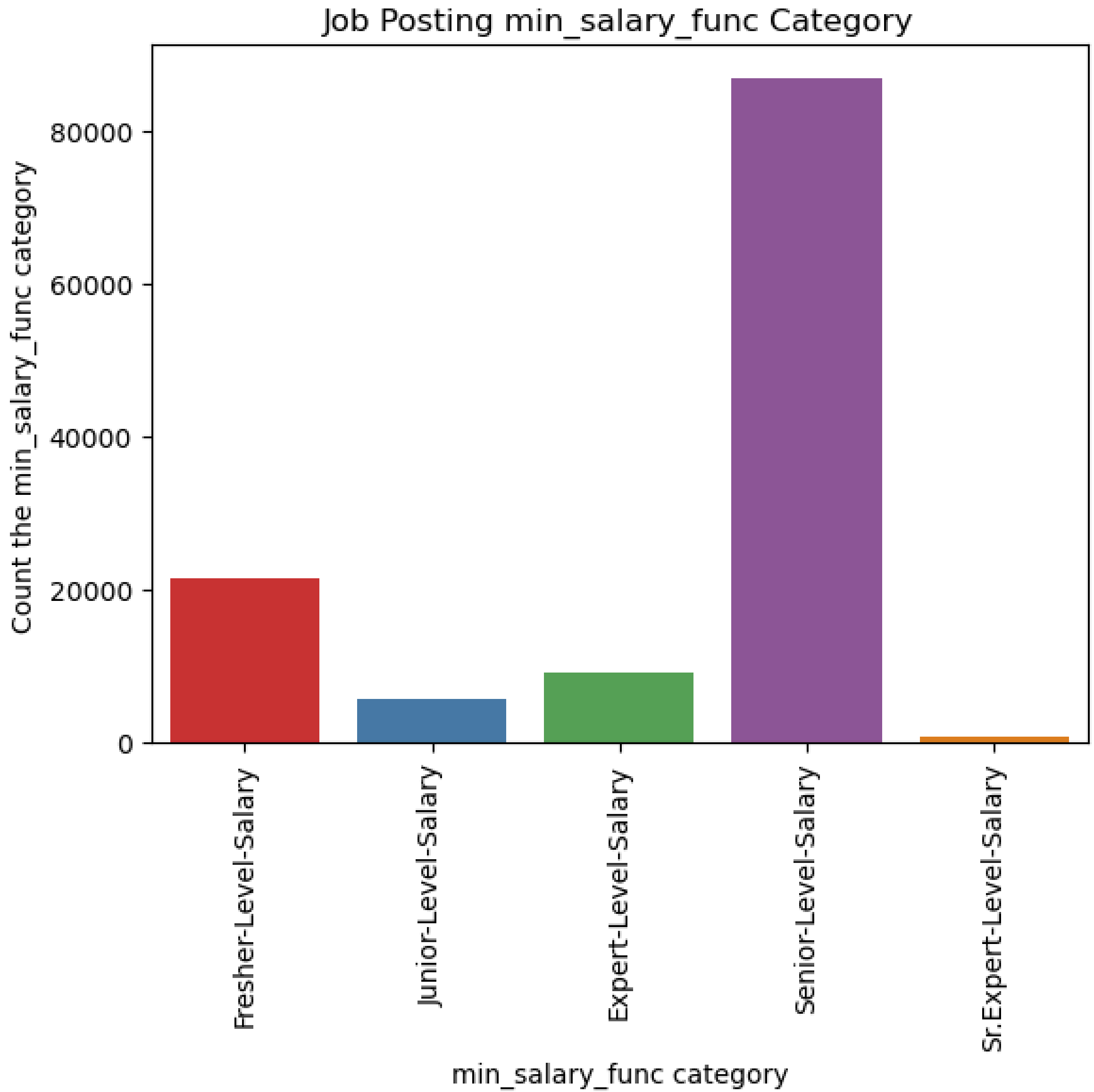


Most of the job posting are fall into Low-Applies category.

In [62]:
```
# let's create the new function for the min_salary_category salary

def min_salary_cat_func(x):
    if(x == "Not-Mentioned"):
        return "Not-Mentioned"
    elif(x<25000):
        return "Fresher-Level-Salary"
    elif(x>25000 and x<=50000):
        return "Junior-Level-Salary"
    elif(x>50000 and x<=100000):
        return "Senior-Level-Salary"
```

```
    elif(x>100000 and x<=200000):
        return "Expert-Level-Salary"
    else:
        return "Sr.Expert-Level-Salary"

new_data['min_salary_func'] = new_data['min_salary'].apply(min_salary_cat_func)
```

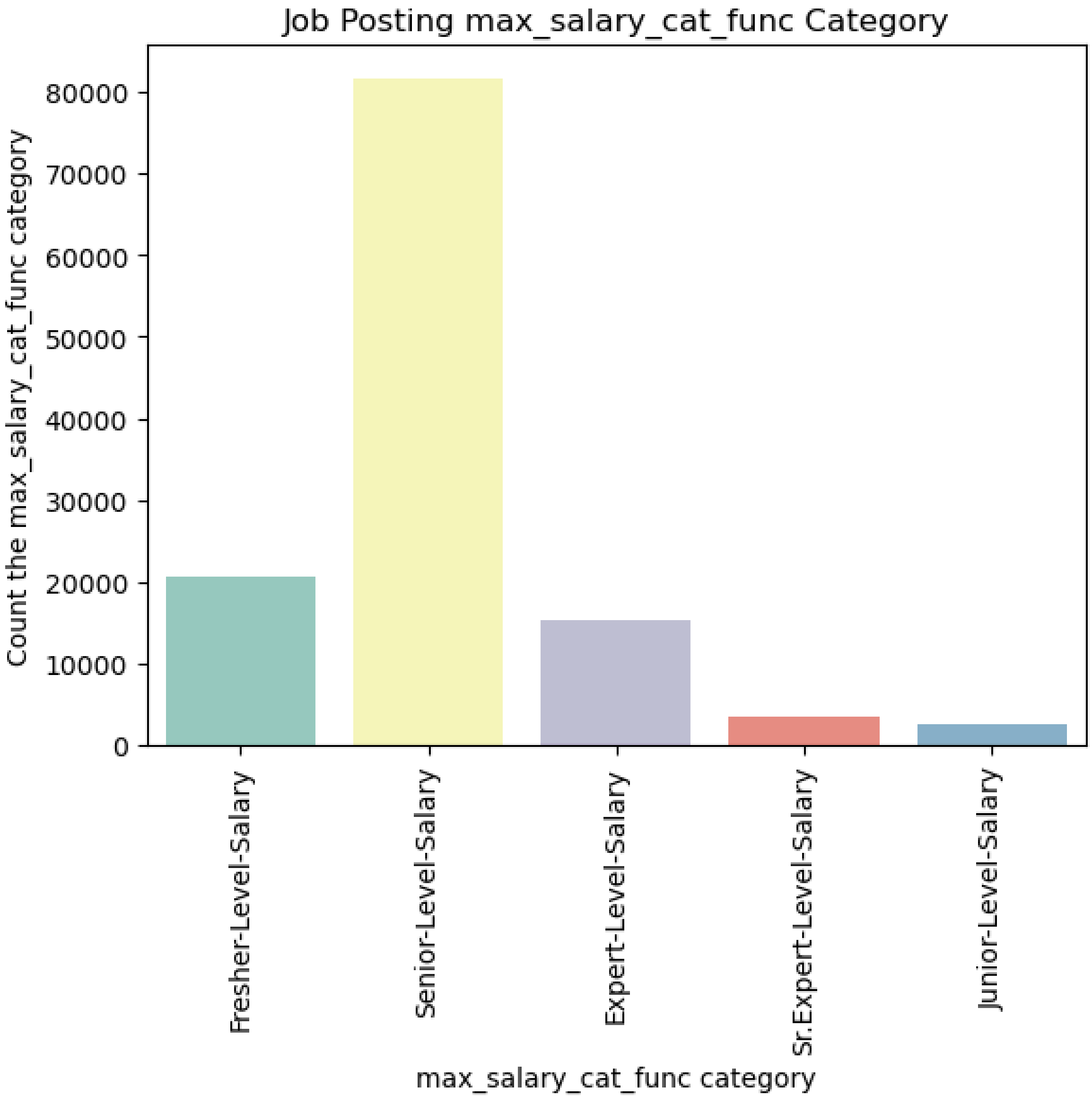
In [63]:
```
sns.countplot(x = 'min_salary_func',data = new_data,palette = 'Set1')
plt.title('Job Posting min_salary_func Category')
plt.xlabel('min_salary_func category')
plt.ylabel('Count the min_salary_func category')
plt.xticks(rotation = 90)
plt.show()
```



As you can see the 'Senior-Level-Salary' is high in the min-salary on linkedin plateform.

In [64]:
```
# let's create the new function for the max_salary_category salary

def max_salary_cat_func(x):
    if(x == "Not-Mentioned"):
        return "Not-Mentioned"
    elif(x<25000):
        return "Fresher-Level-Salary"
    elif(x>25000 and x<=50000):
        return "Junior-Level-Salary"
    elif(x>50000 and x<=100000):
        return "Senior-Level-Salary"
    elif(x>100000 and x<=200000):
        return "Expert-Level-Salary"
    else:
        return "Sr.Expert-Level-Salary"

new_data['max_salary_func'] = new_data['max_salary'].apply(max_salary_cat_func)
```

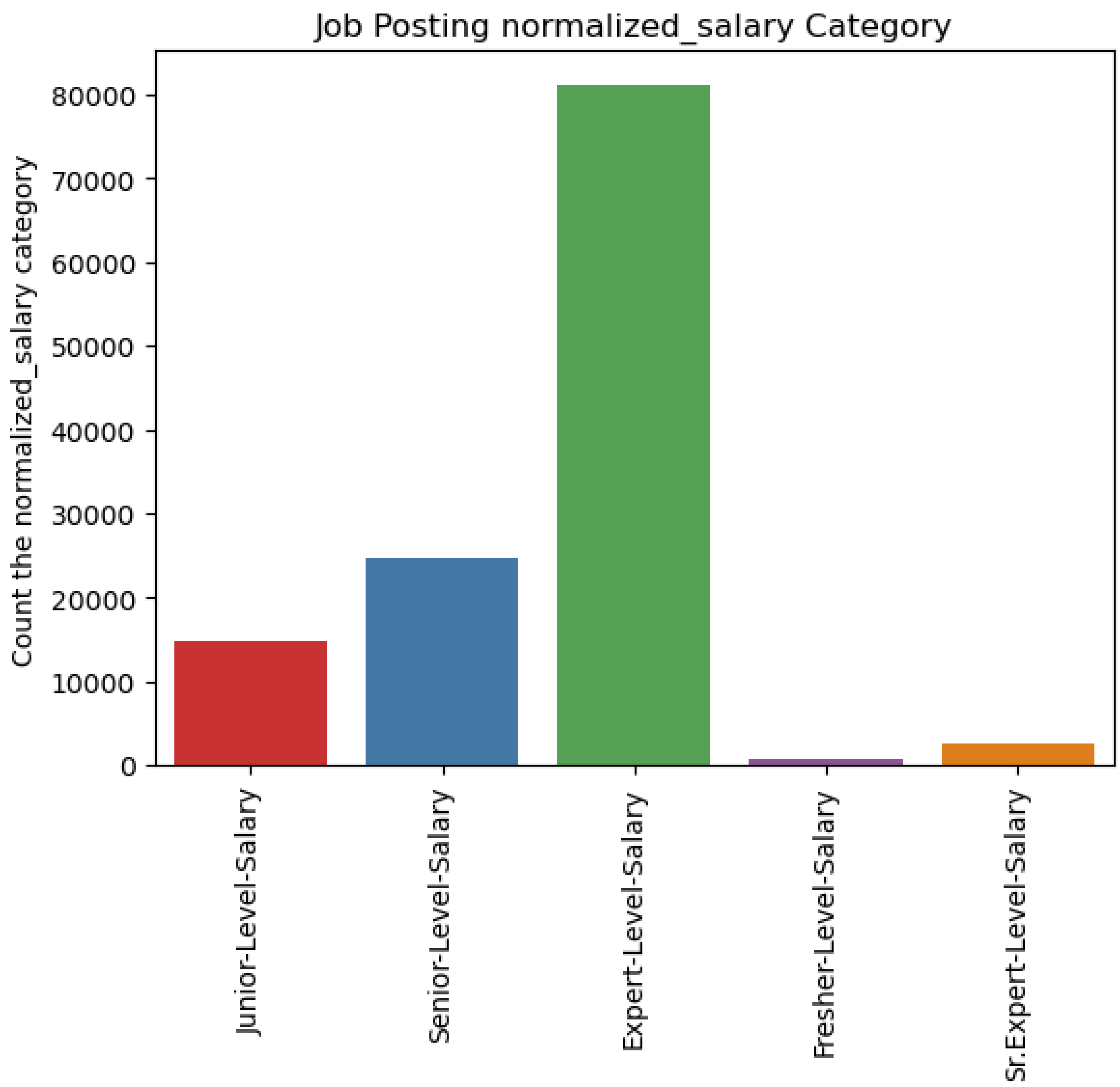
In [65]:
```
sns.countplot(x = 'max_salary_func',data = new_data,palette = 'Set3')
plt.title('Job Posting max_salary_cat_func Category')
plt.xlabel('max_salary_cat_func category')
plt.ylabel('Count the max_salary_cat_func category')
plt.xticks(rotation = 90)
plt.show()
```



Again you can see the winner is 'Senior-Level-Salary' is high in the max_salary criteria on the linkedin platform.

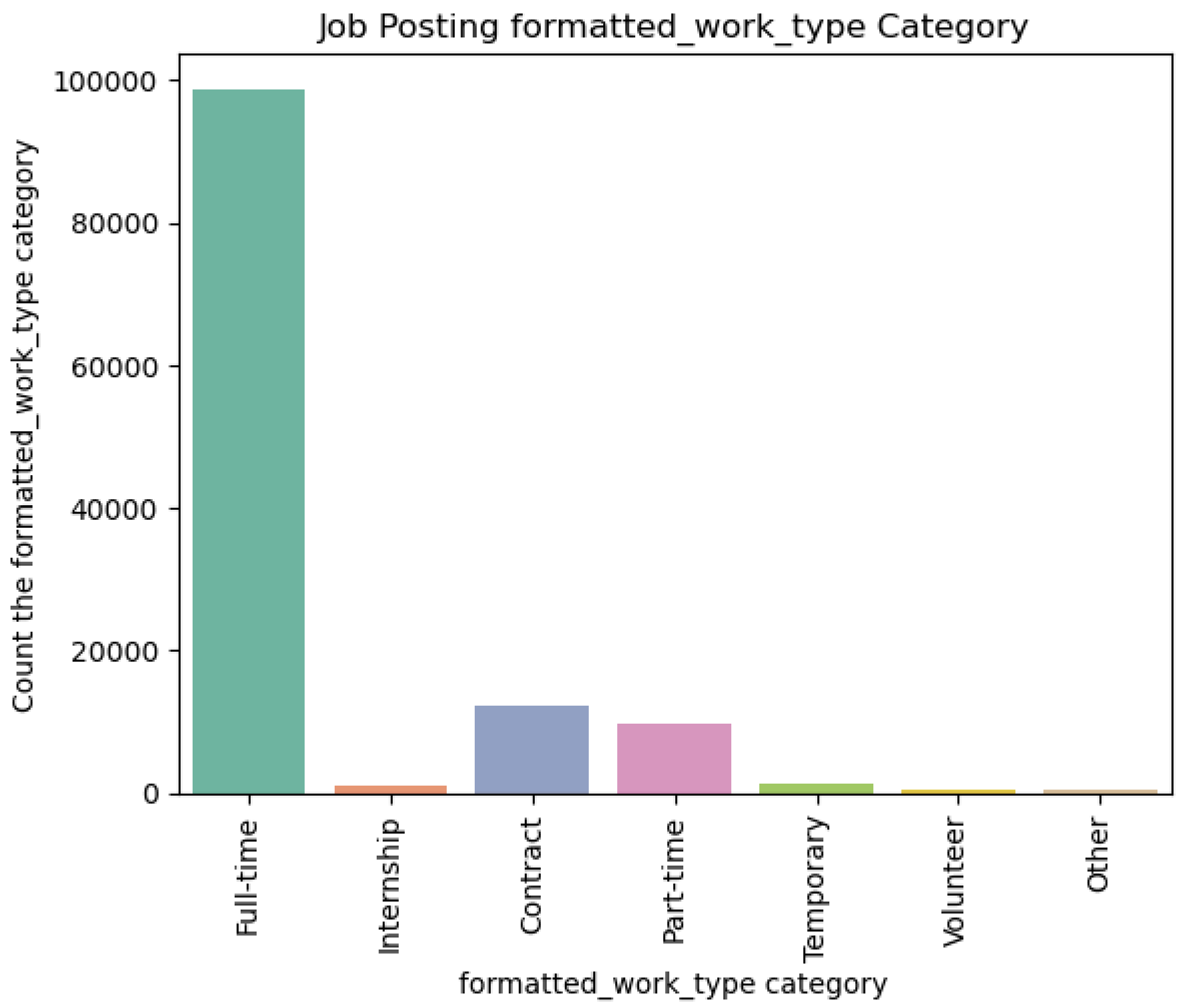
In [66]:
```
# let's create the new function for the normalized_salary salary

def normalized_salary_cat_func(x):
    if(x == "Not-Mentioned"):
        return "Not-Mentioned"
    elif(x<25000):
        return "Fresher-Level-Salary"
    elif(x>25000 and x<=50000):
        return "Junior-Level-Salary"
    elif(x>50000 and x<=100000):
        return "Senior-Level-Salary"
    elif(x>100000 and x<=200000):
        return "Expert-Level-Salary"
    else:
        return "Sr.Expert-Level-Salary"

new_data['normalized_salary_func'] = new_data['normalized_salary'].apply(normalized_salary_cat_func)
```

In [67]:
```
sns.countplot(x = 'normalized_salary_func',data = new_data,palette = 'Set1')
plt.title('Job Posting normalized_salary Category')
plt.xlabel('normalized_salary category')
plt.ylabel('Count the normalized_salary category')
plt.xticks(rotation = 90)
plt.show()
```



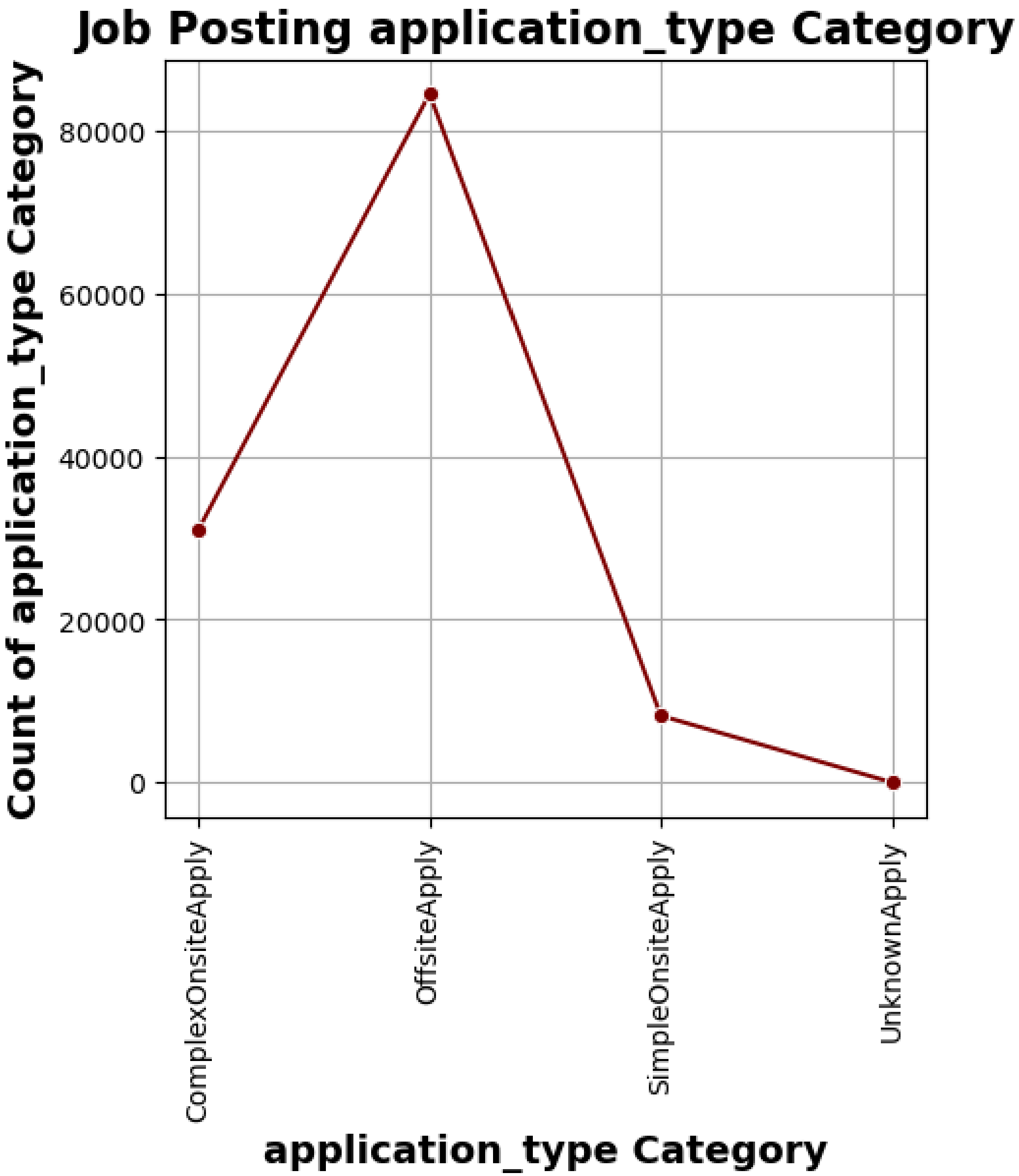
You can see clearly we have not proper data to analyze it but as per this data "Expert-level-Salary" in normalized salary category is most offered by company.

In [68]:
```
sns.countplot(x = 'formatted_work_type',data = new_data,palette = 'Set2')
plt.title('Job Posting formatted_work_type Category')
plt.xlabel('formatted_work_type category')
plt.ylabel('Count the formatted_work_type category')
plt.xticks(rotation = 90)
plt.show()
```



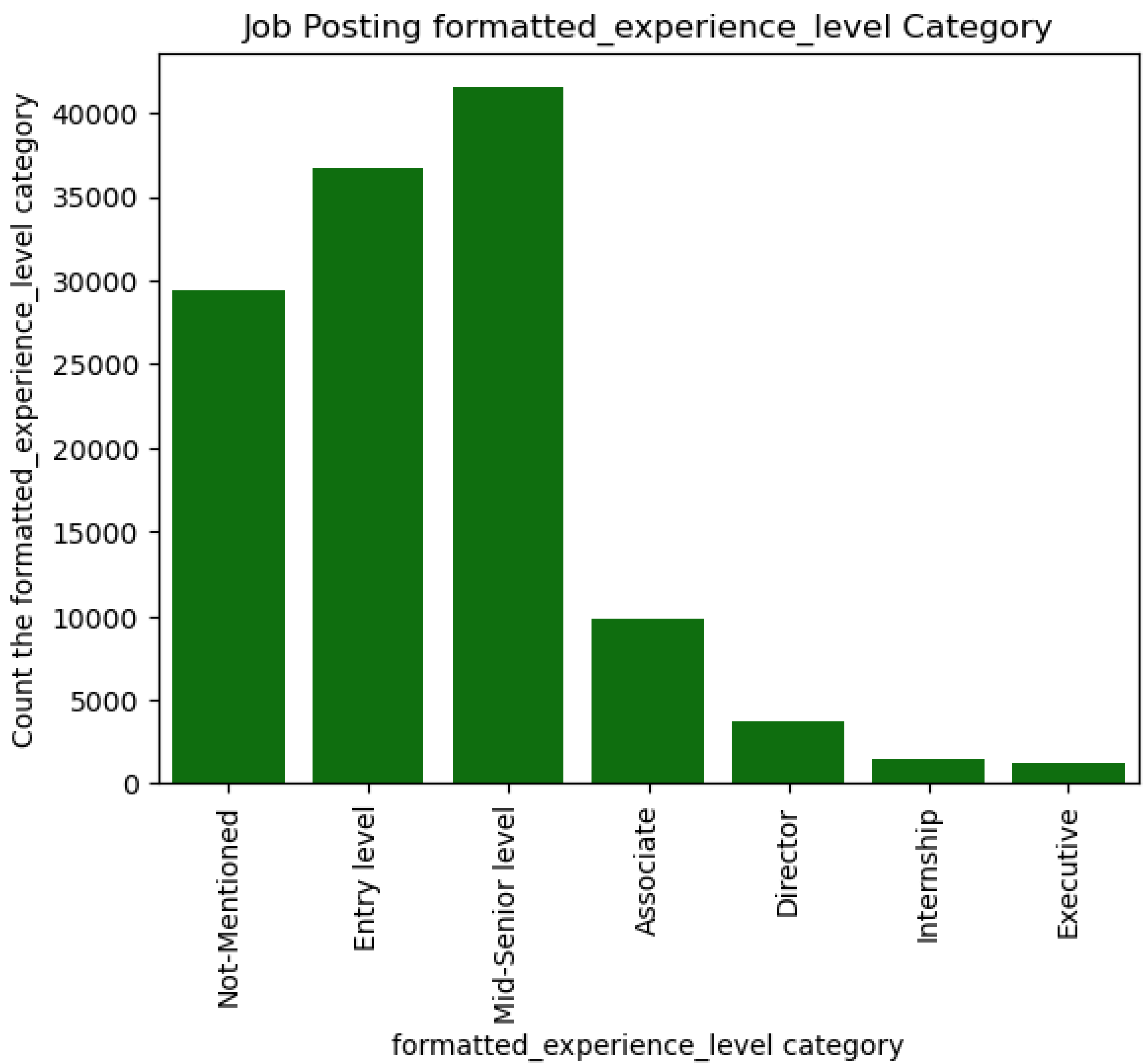
Most of the jobs work type is Full-time.

In [69]:
```
# First, calculate counts manually
application_type_counts = new_data['application_type'].value_counts().sort_index()

# Line plot
plt.figure(figsize=(5,5))
sns.lineplot(x=application_type_counts.index, y=application_type_counts.values, marker='o', color='maroon')

plt.title('Job Posting application_type Category', fontsize=16, fontweight='bold')
plt.xlabel('application_type Category', fontsize=14, fontweight='bold')
plt.ylabel('Count of application_type Category', fontsize=14, fontweight='bold')
plt.xticks(rotation=90)
plt.grid(True)
plt.show()
```


Job Posting application_type Category
Count of application_type Category
0
20000
40000
60000
80000
ComplexOnsiteApply
OffsiteApply
SimpleOnsiteApply
UnknownApply
application_type Category

In the most of the jobs which is posted on linkedin platform application type is "OffsiteApply" is high.

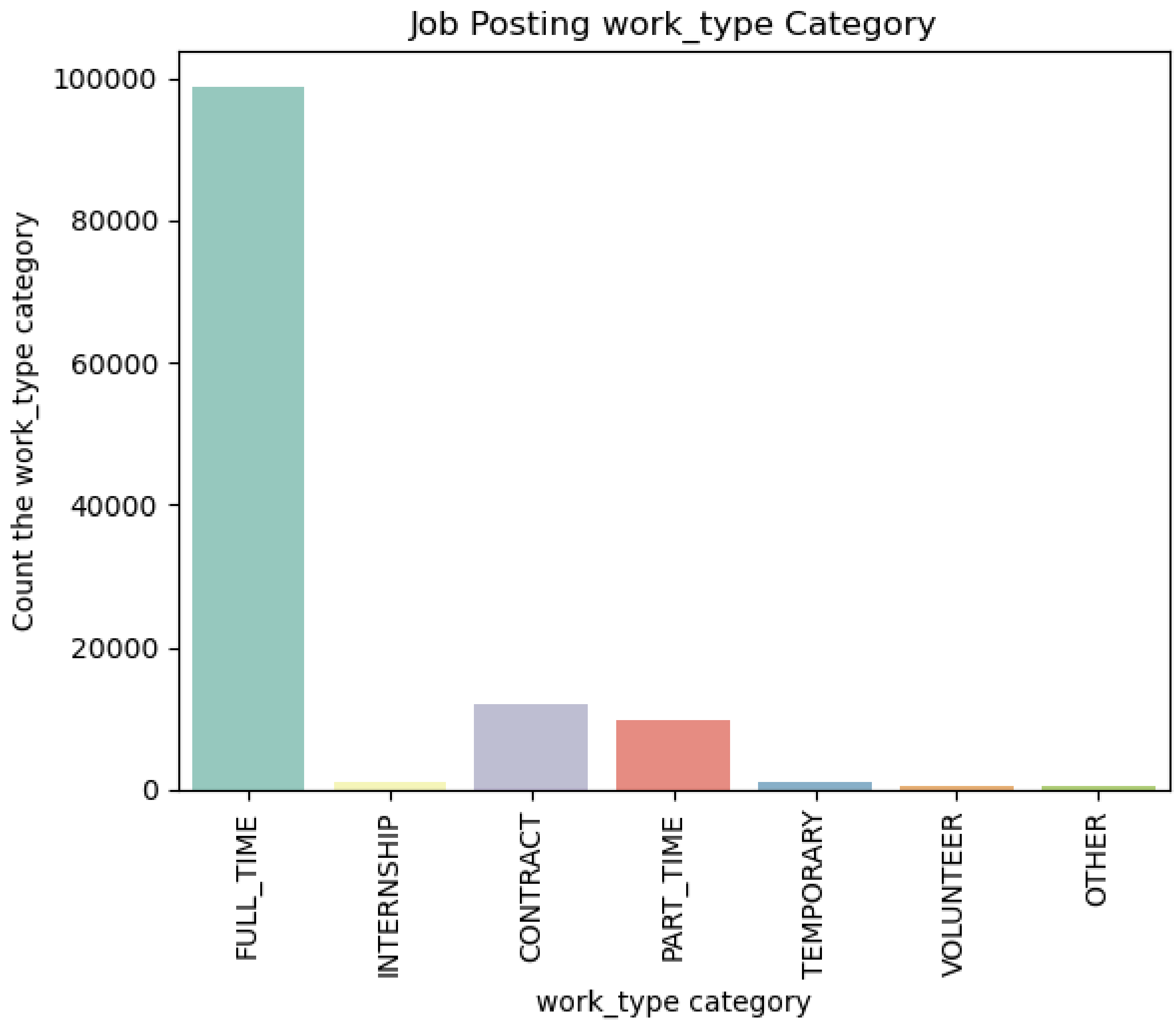
In [70]:
```
sns.countplot(x = 'formatted_experience_level',data = new_data,color = "green")
plt.title('Job Posting formatted_experience_level Category')
plt.xlabel('formatted_experience_level category')
plt.ylabel('Count the formatted_experience_level category')
plt.xticks(rotation = 90)
plt.show()
```



Most of the jobs posting requirement is "Mid-Senior level" & second position is "Entry-Level".

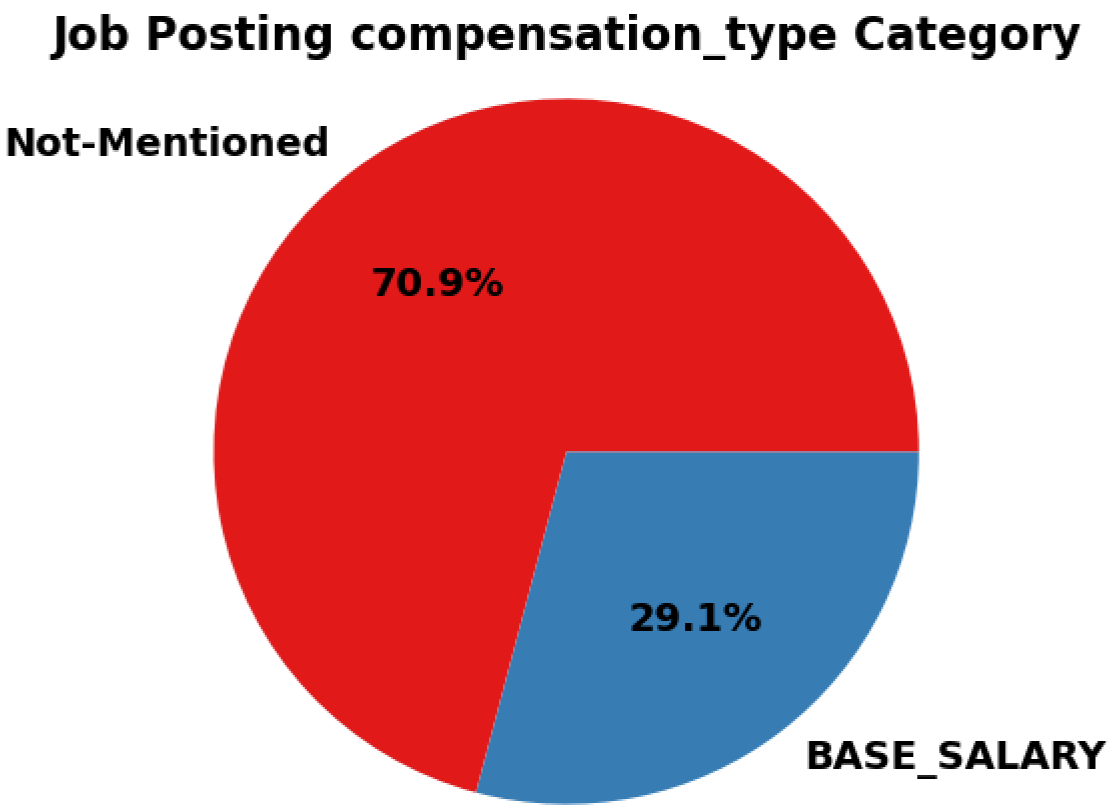
In [71]:
```
sns.countplot(x = 'work_type',data = new_data,palette = "Set3")
plt.title('Job Posting work_type Category')
```

```
plt.xlabel('work_type category')
plt.ylabel('Count the work_type category')
plt.xticks(rotation = 90)
plt.show()
```



In above chart you can see clearly the work type category is "Full-Time".

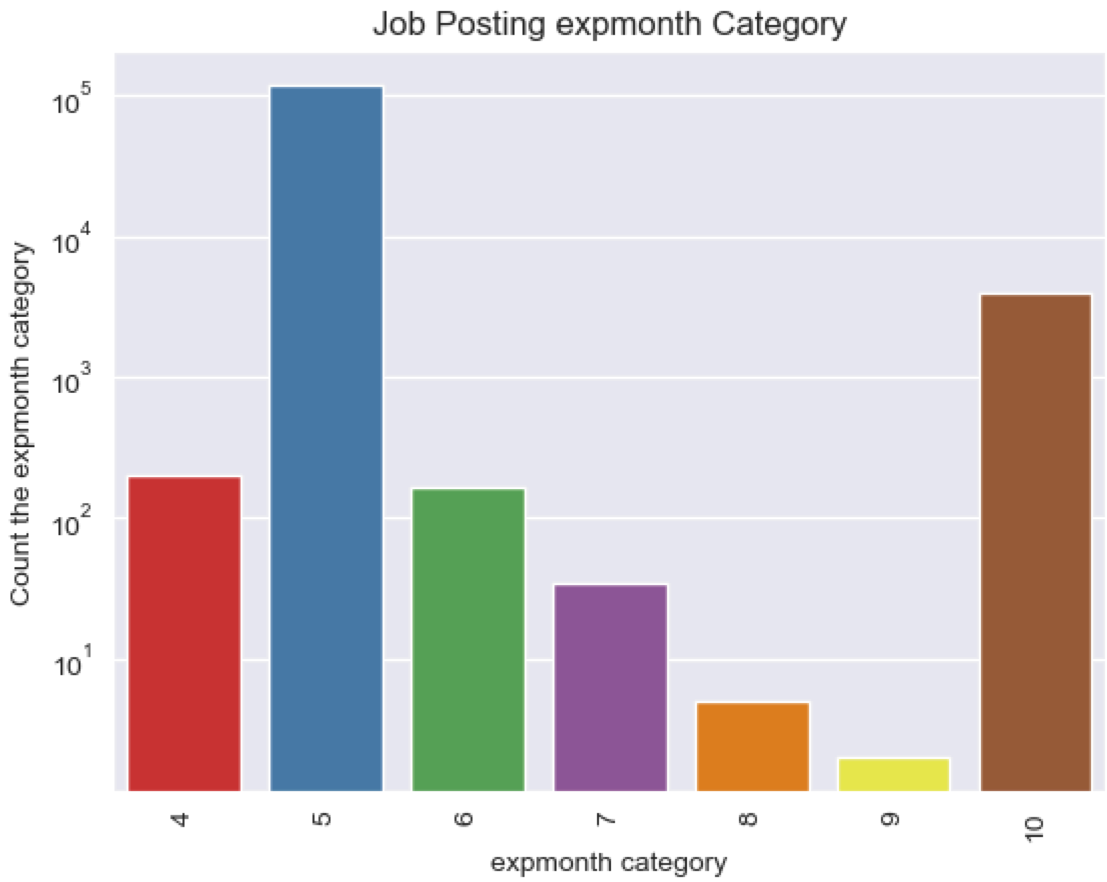
In [72]:
```
# Calculate counts
compensation_counts = new_data['compensation_type'].value_counts()

# Pie chart
plt.figure(figsize=(5,5))
plt.pie(
    compensation_counts.values,
    labels=compensation_counts.index,
    autopct='%1.1f%%',
    colors=sns.color_palette('Set1'),
    textprops={'fontsize': 14, 'fontweight': 'bold'}
)
plt.title('Job Posting compensation_type Category', fontsize=16, fontweight='bold')
plt.axis('equal')  # Circle maintain karva
plt.show()
```



we have not proper or you can say accurate data that's we fill Not-Mentioned but if you consider raw data and analyze it Most of the "Base Salary" Compensetion Type is high.

In [79]:
```
# let's check the job posting expiry month


sns.set_style("darkgrid")
sns.countplot(x = 'expmonth',data = new_data,palette = "Set1")
plt.title('Job Posting expmonth Category')
plt.xlabel('expmonth category')
plt.ylabel('Count the expmonth category')
```

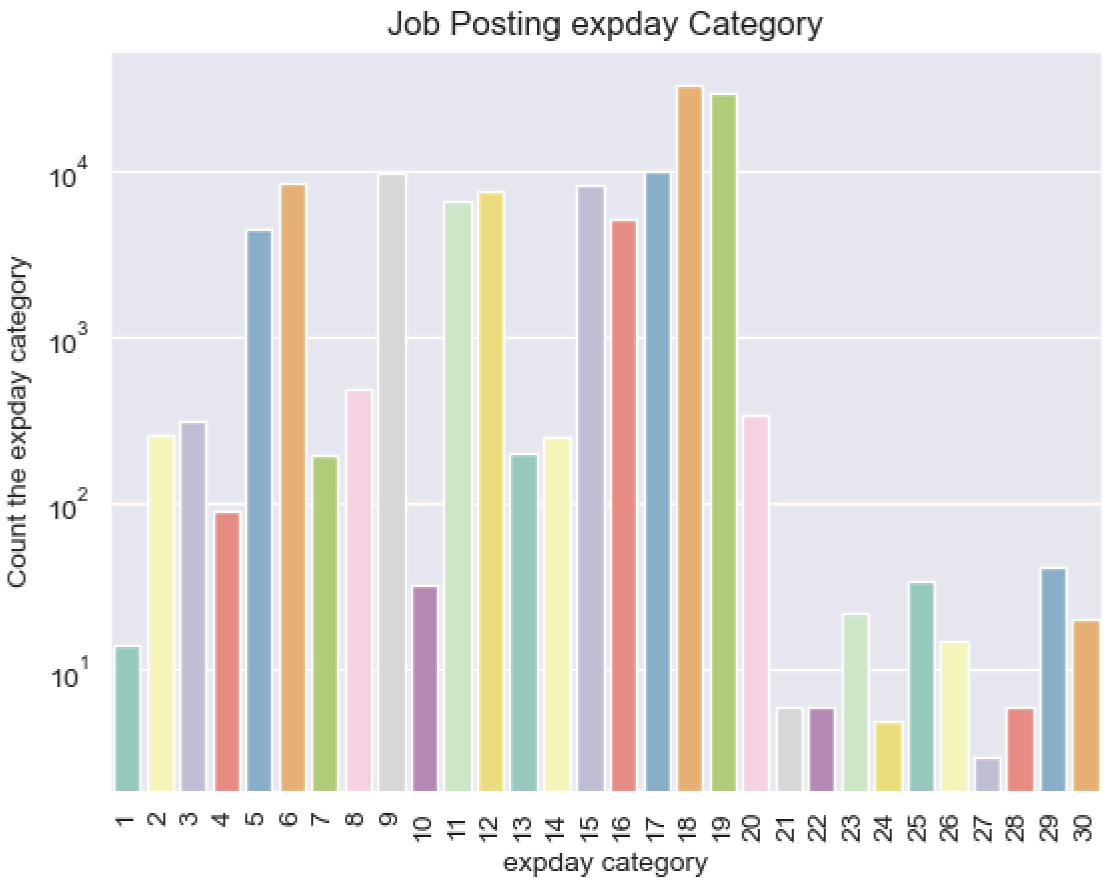
```
plt.xticks(rotation = 90)
plt.yscale("log")
plt.show()
```



Most of the job-posting expiry month is May may be Applicants enjoy summer vacation joke a part.

In [80]:
```
# let's check the job posting expiry day

sns.set_style("darkgrid")
sns.countplot(x = 'expday',data = new_data,palette = "Set3")
```

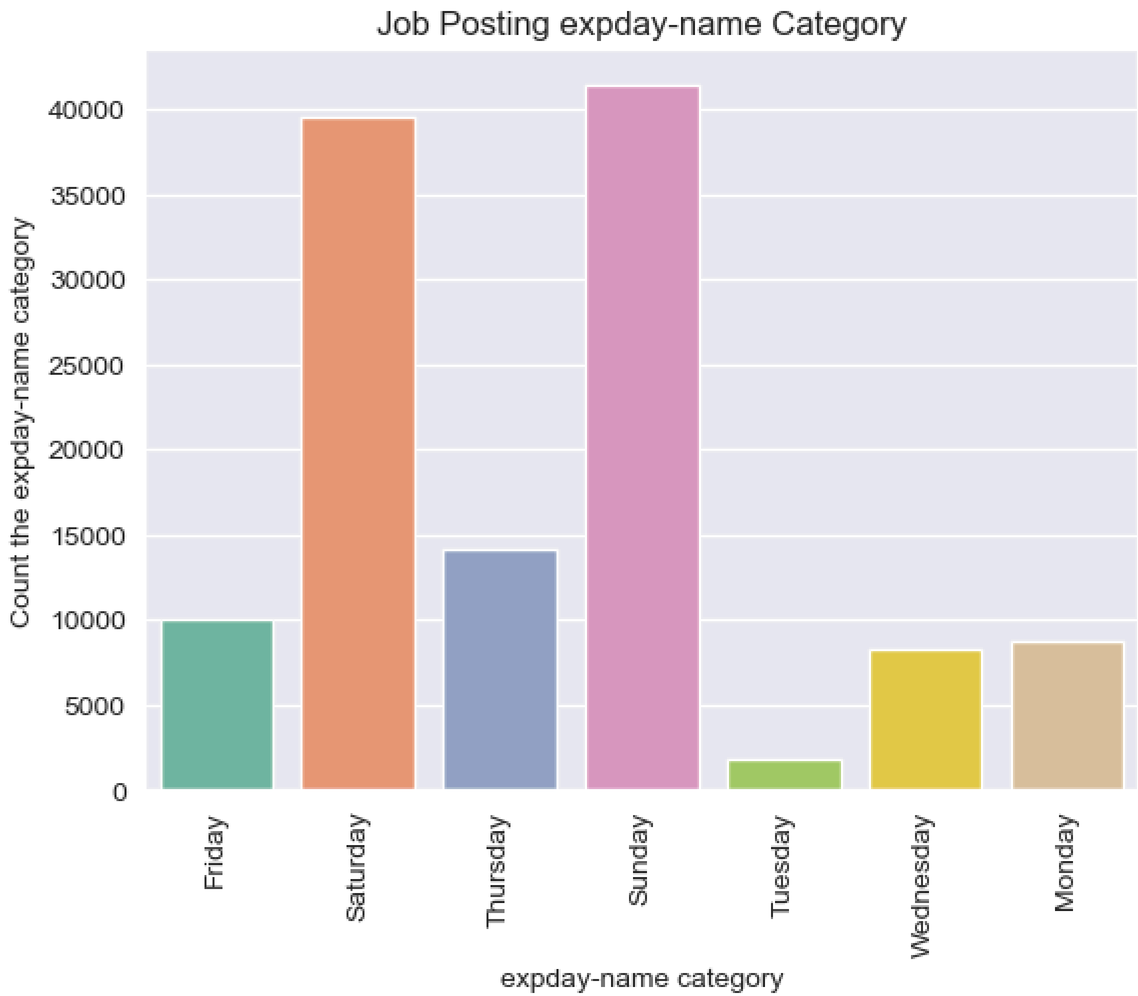
```
plt.title('Job Posting expday Category')
plt.xlabel('expday category')
plt.ylabel('Count the expday category')
plt.xticks(rotation = 90)
plt.yscale("log")
plt.show()
```



Most of the job posting expired in the date of 18th May.

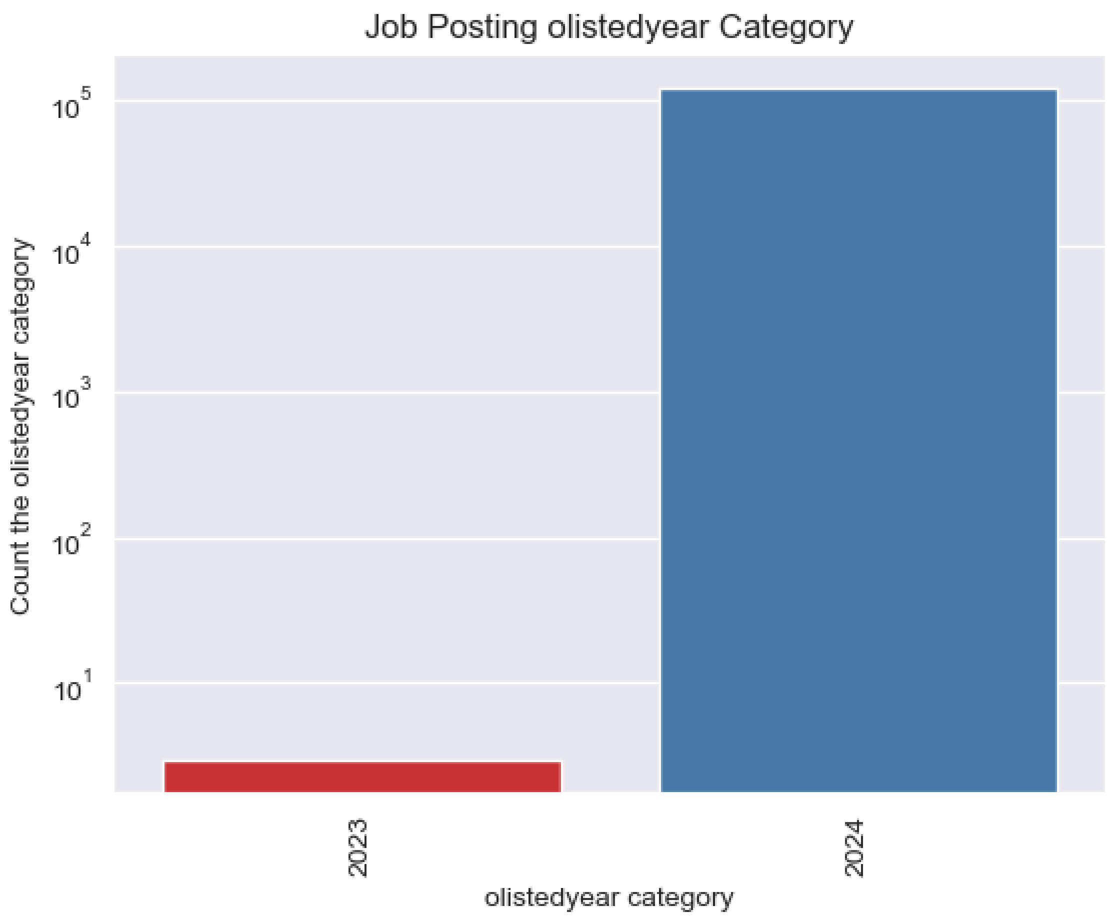
In [98]:
```
# let's check the job posting expiry expday-name

sns.set_style("darkgrid")
```

```
sns.countplot(x = 'expday-name',data = new_data,palette = "Set2")
plt.title('Job Posting expday-name Category')
plt.xlabel('expday-name category')
plt.ylabel('Count the expday-name category')
plt.xticks(rotation = 90)
plt.show()
```

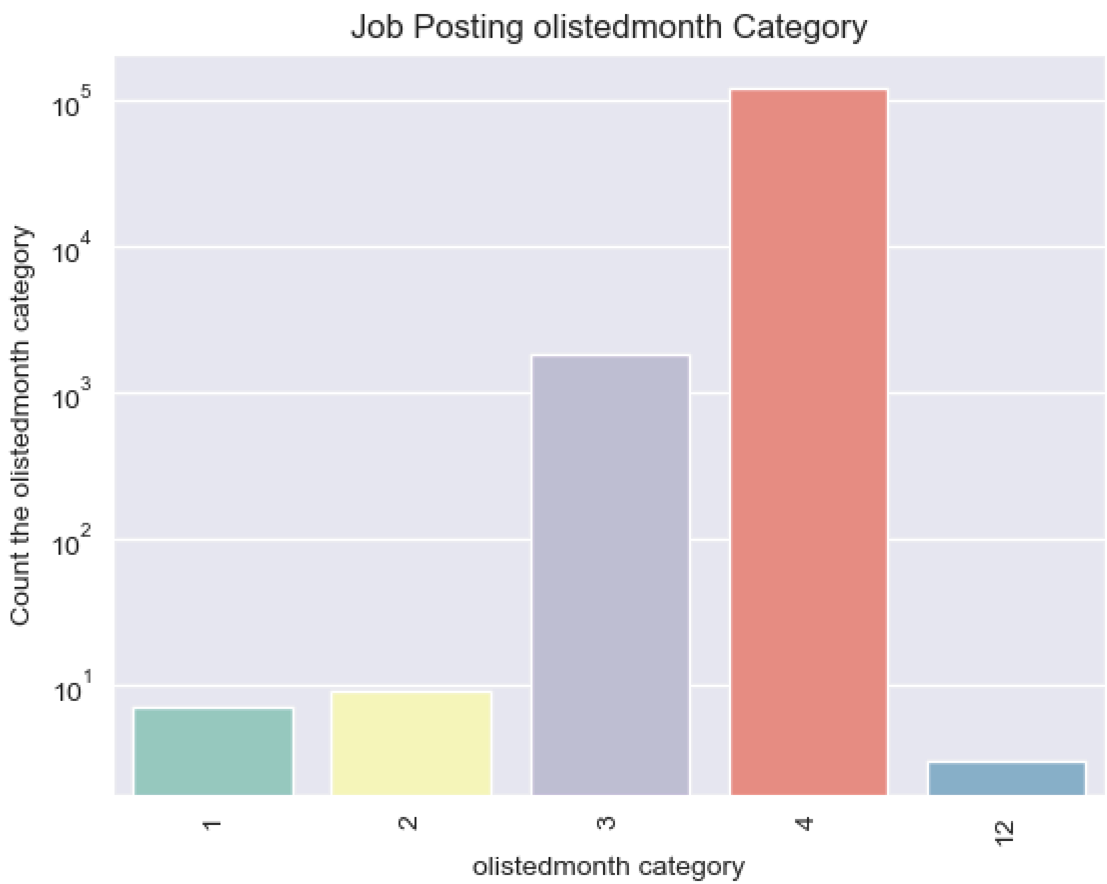


Most of the job posting are expired in the day of Sunday May be Applicants Enjoy Weekend.

In [81]:
```
# let's check the job posting olistedyear

sns.set_style("darkgrid")
sns.countplot(x = 'olistedyear',data = new_data,palette = "Set1")
plt.title('Job Posting olistedyear Category')
plt.xlabel('olistedyear category')
plt.ylabel('Count the olistedyear category')
plt.xticks(rotation = 90)
plt.yscale("log")
plt.show()
```



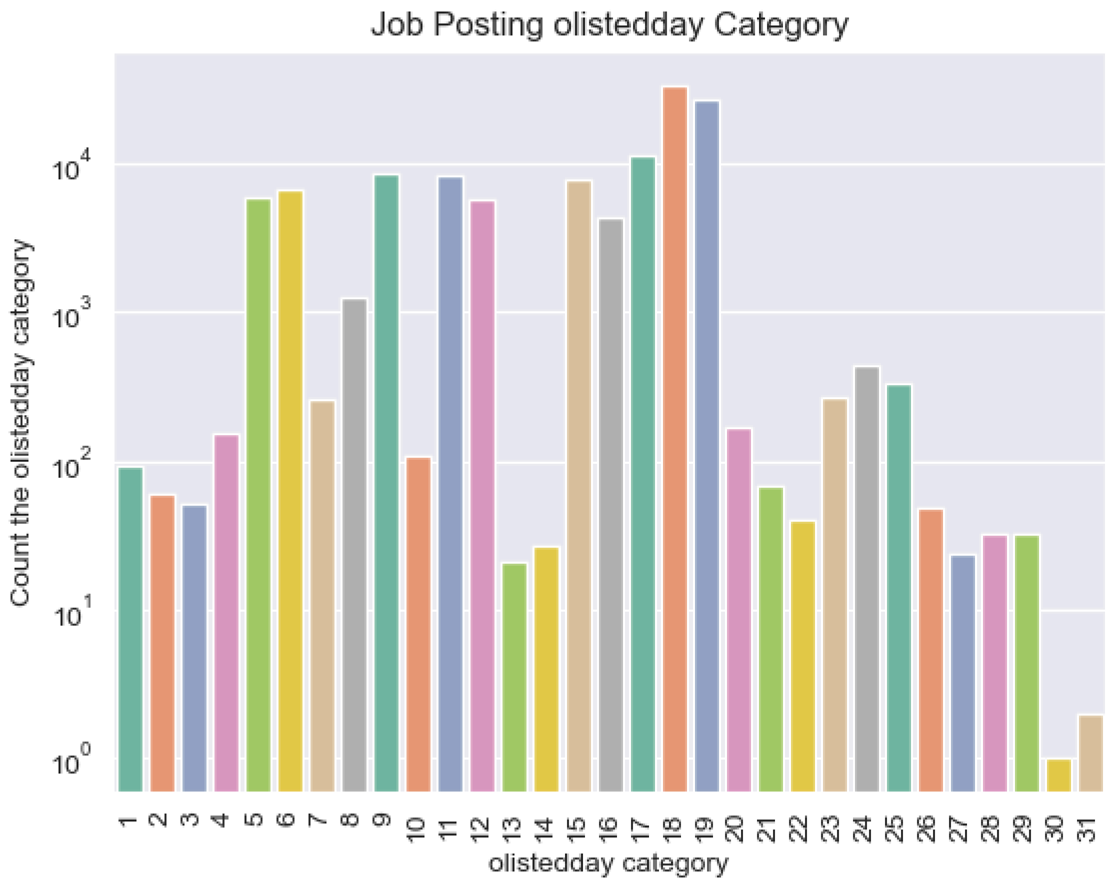
In [82]:
```
# let's check the job posting olistedmonth

sns.set_style("darkgrid")
sns.countplot(x = 'olistedmonth',data = new_data,palette = "Set3")
plt.title('Job Posting olistedmonth Category')
plt.xlabel('olistedmonth category')
plt.ylabel('Count the olistedmonth category')
plt.xticks(rotation = 90)
plt.yscale("log")
plt.show()
```



April Month is frequntly accured in the listedmonth category.

In [83]:
```
# let's check the job posting olistedday

sns.set_style("darkgrid")
sns.countplot(x = 'olistedday',data = new_data,palette = "Set2")
plt.title('Job Posting olistedday Category')
plt.xlabel('olistedday category')
plt.ylabel('Count the olistedday category')
plt.xticks(rotation = 90)
```

```
plt.yscale("log")
plt.show()
```

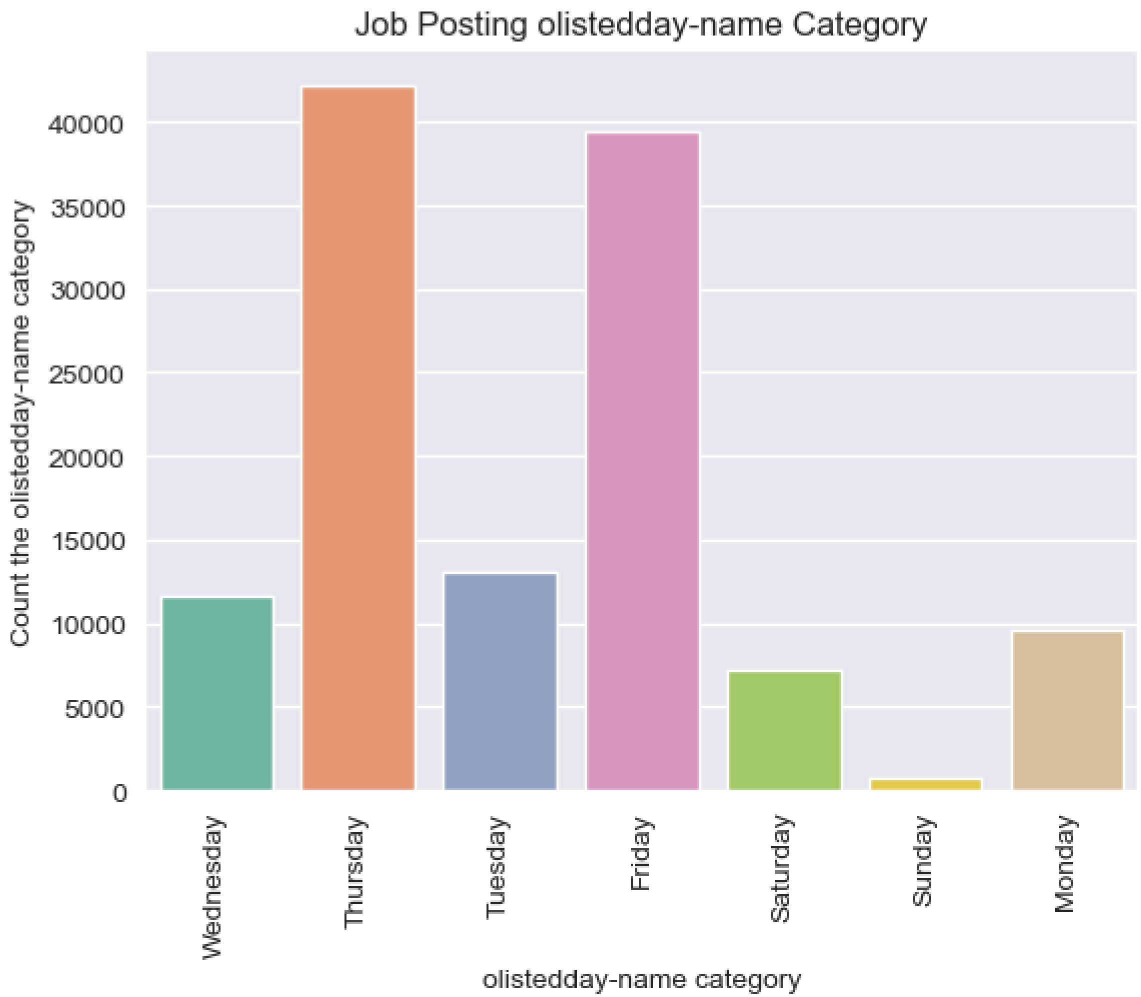


Date 18,19 are more accured in the olistedday category.

In [103…

```
# let's check the job posting olistedday-name

sns.set_style("darkgrid")
sns.countplot(x = 'olistedday-name',data = new_data,palette = "Set2")
plt.title('Job Posting olistedday-name Category')
plt.xlabel('olistedday-name category')
plt.ylabel('Count the olistedday-name category')
```

```
plt.xticks(rotation = 90)
plt.show()
```

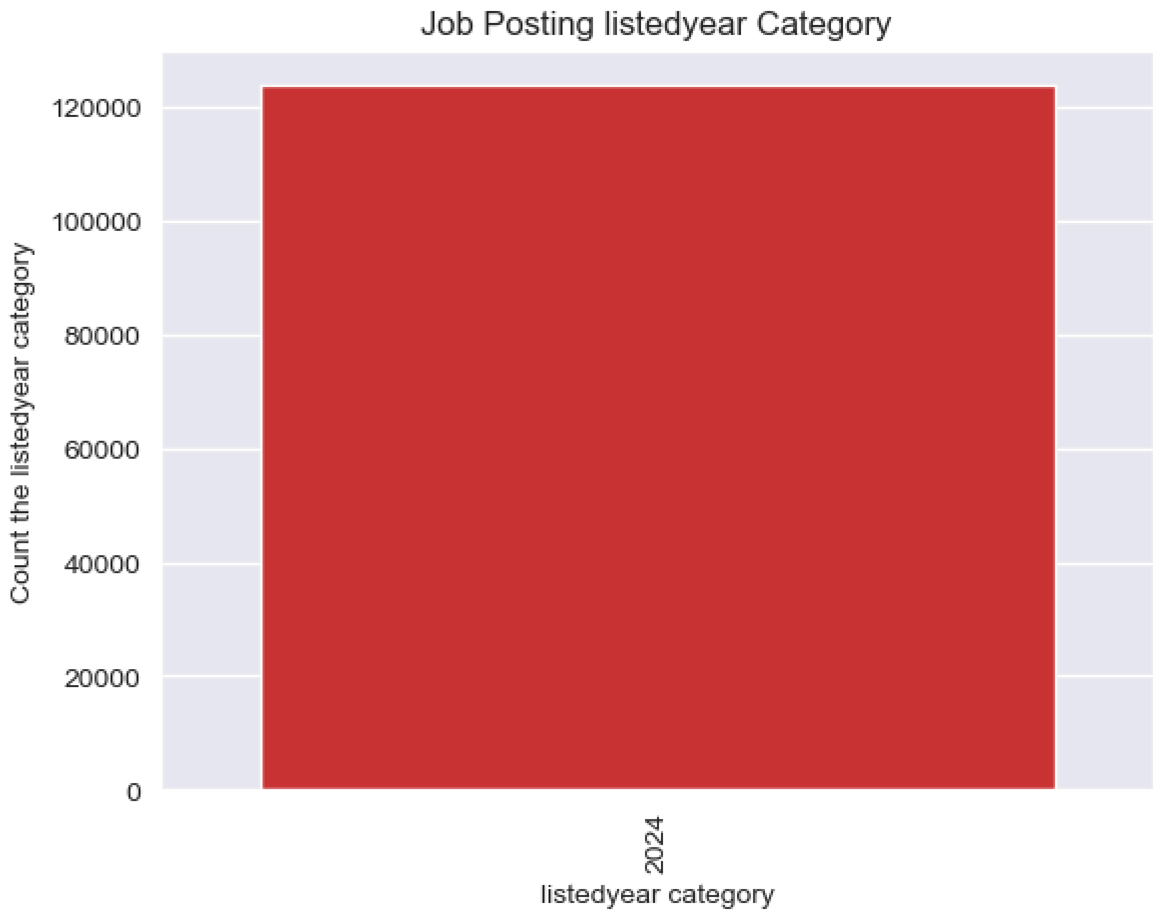


Thirsday is high in listedday-name category.

In [104…

```
# let's check the job posting listedyear

sns.set_style("darkgrid")
```

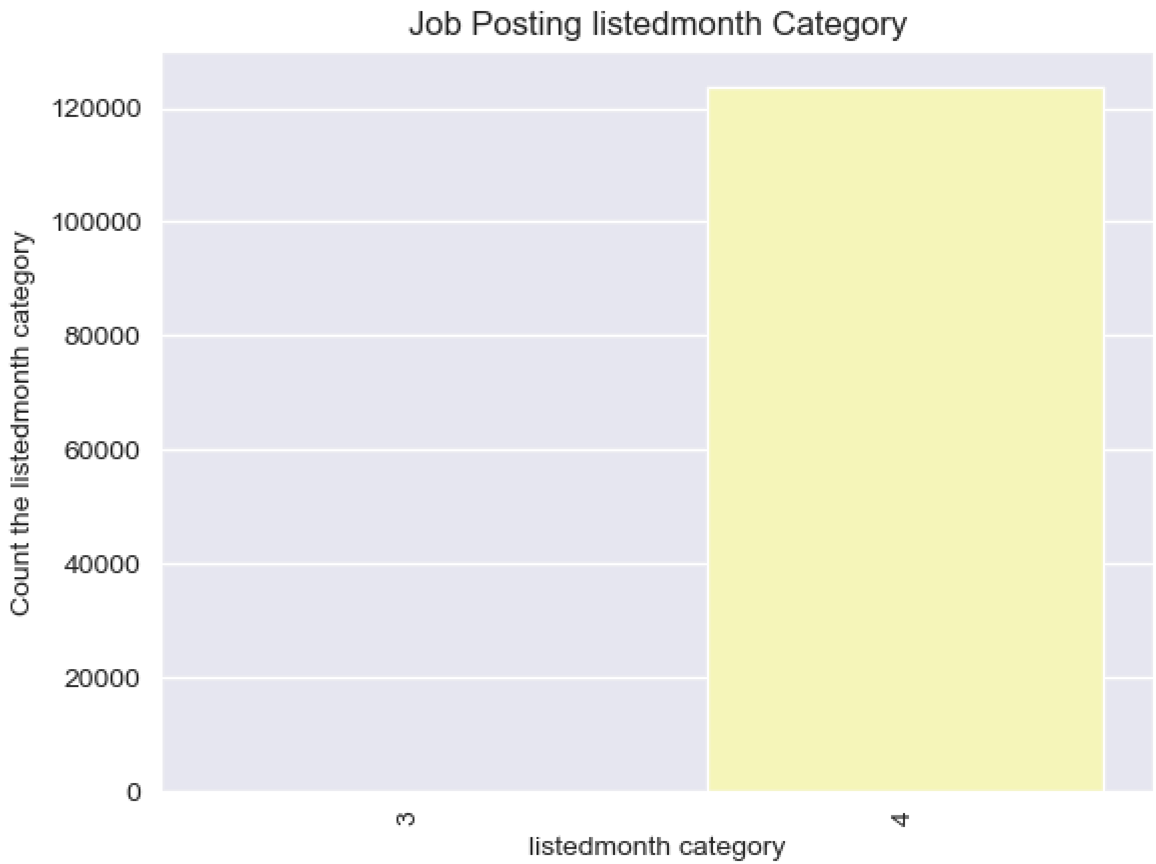
```
sns.countplot(x = 'listedyear',data = new_data,palette = "Set1")
plt.title('Job Posting listedyear Category')
plt.xlabel('listedyear category')
plt.ylabel('Count the listedyear category')
plt.xticks(rotation = 90)
plt.show()
```



In [105…

```
# let's check the job posting listedmonth

sns.set_style("darkgrid")
sns.countplot(x = 'listedmonth',data = new_data,palette = "Set3")
plt.title('Job Posting listedmonth Category')
```

```
plt.xlabel('listedmonth category')
plt.ylabel('Count the listedmonth category')
plt.xticks(rotation = 90)
plt.show()
```



April Month is frequntly accured in the listedmonth category.

In [106…

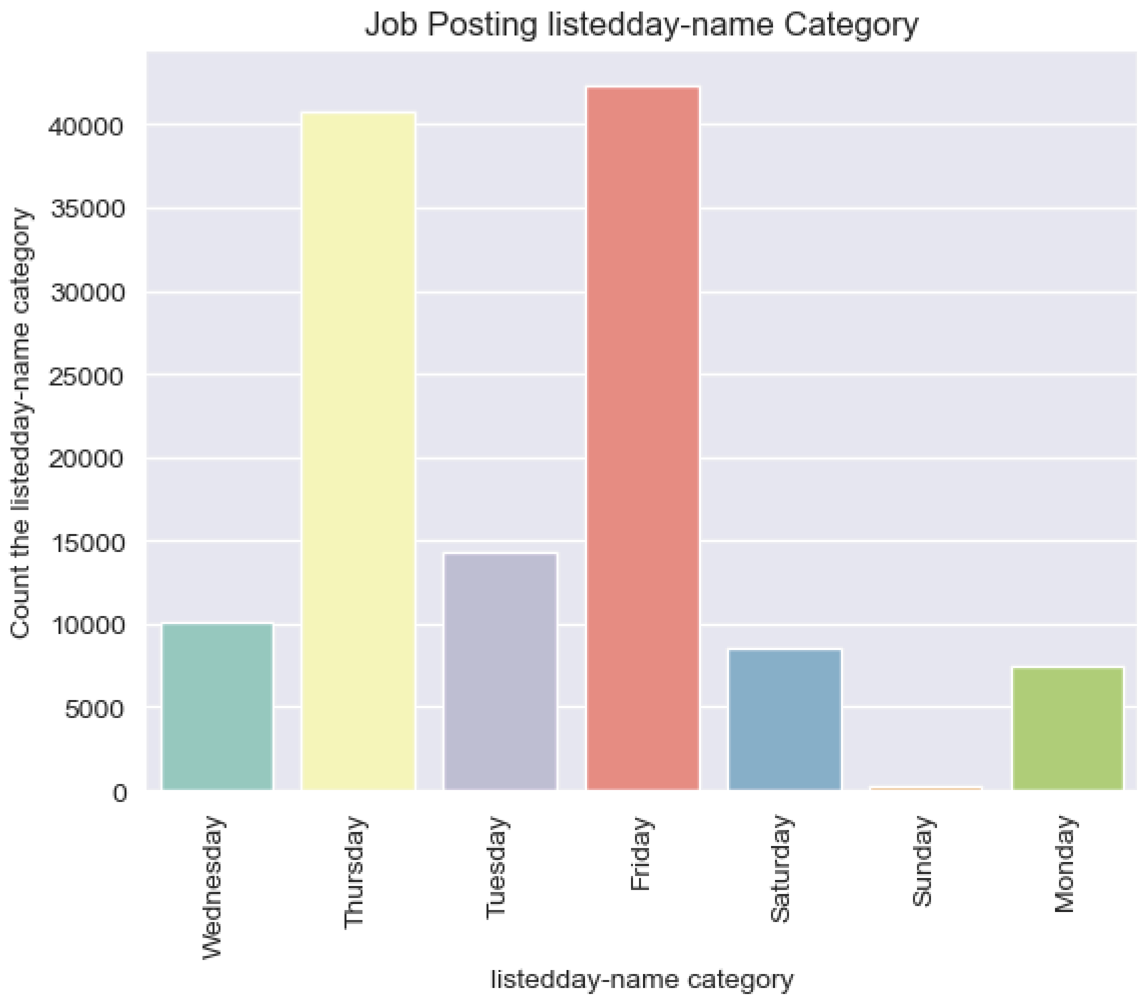
```
# let's check the job posting listedday-name

sns.set_style("darkgrid")
sns.countplot(x = 'listedday',data = new_data,palette = "Set2")
plt.title('Job Posting listedday Category')
```

```
plt.xlabel('listedday category')
plt.ylabel('Count the listedday category')
plt.xticks(rotation = 90)
plt.show()
```

In [107…

```
# let's check the job posting listedday-name

sns.set_style("darkgrid")
sns.countplot(x = 'listedday-name',data = new_data,palette = "Set3")
plt.title('Job Posting listedday-name Category')
plt.xlabel('listedday-name category')
plt.ylabel('Count the listedday-name category')
```

```
plt.xticks(rotation = 90)
plt.show()
```

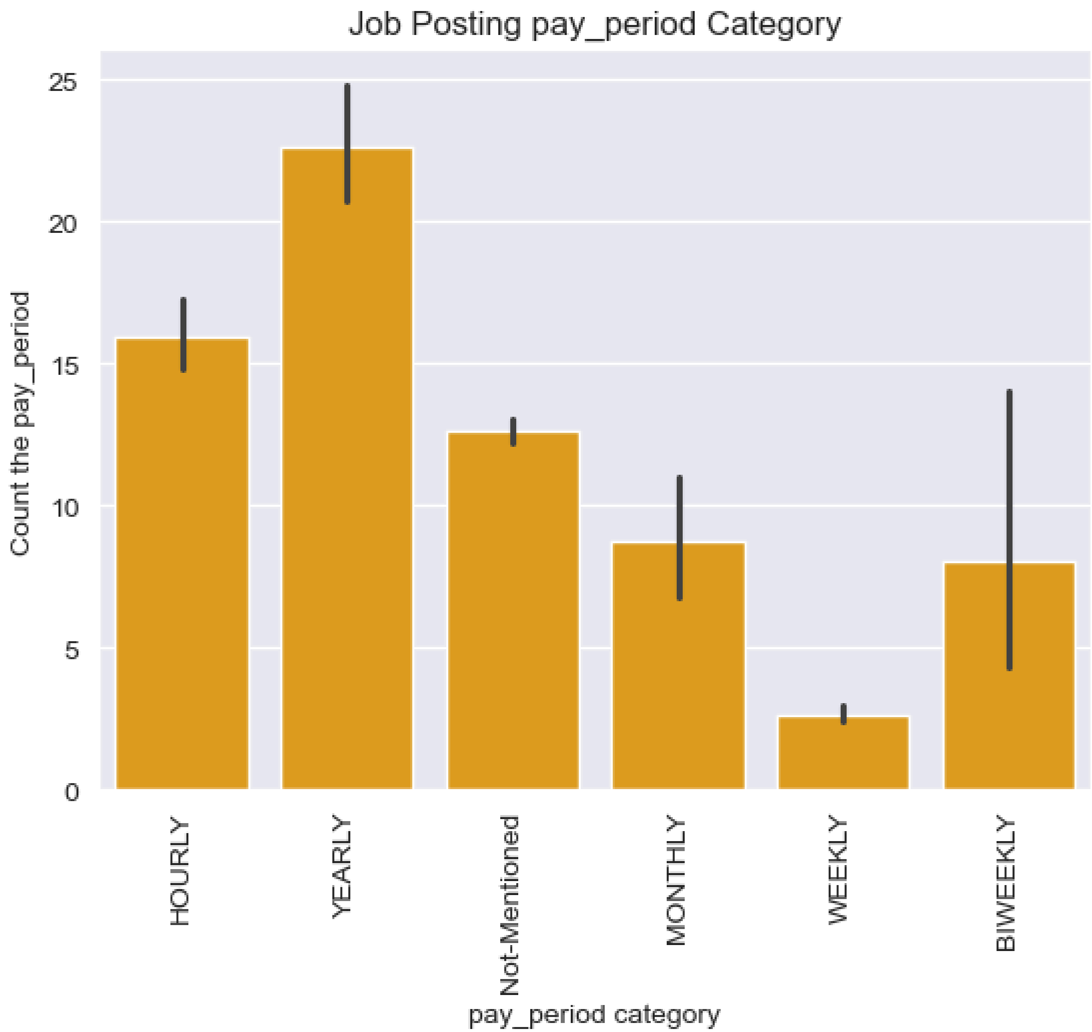


Bi-Variate-Analysis

In [109…

```
# let's check the job posting pay_period wise views

sns.set_style("darkgrid")
```

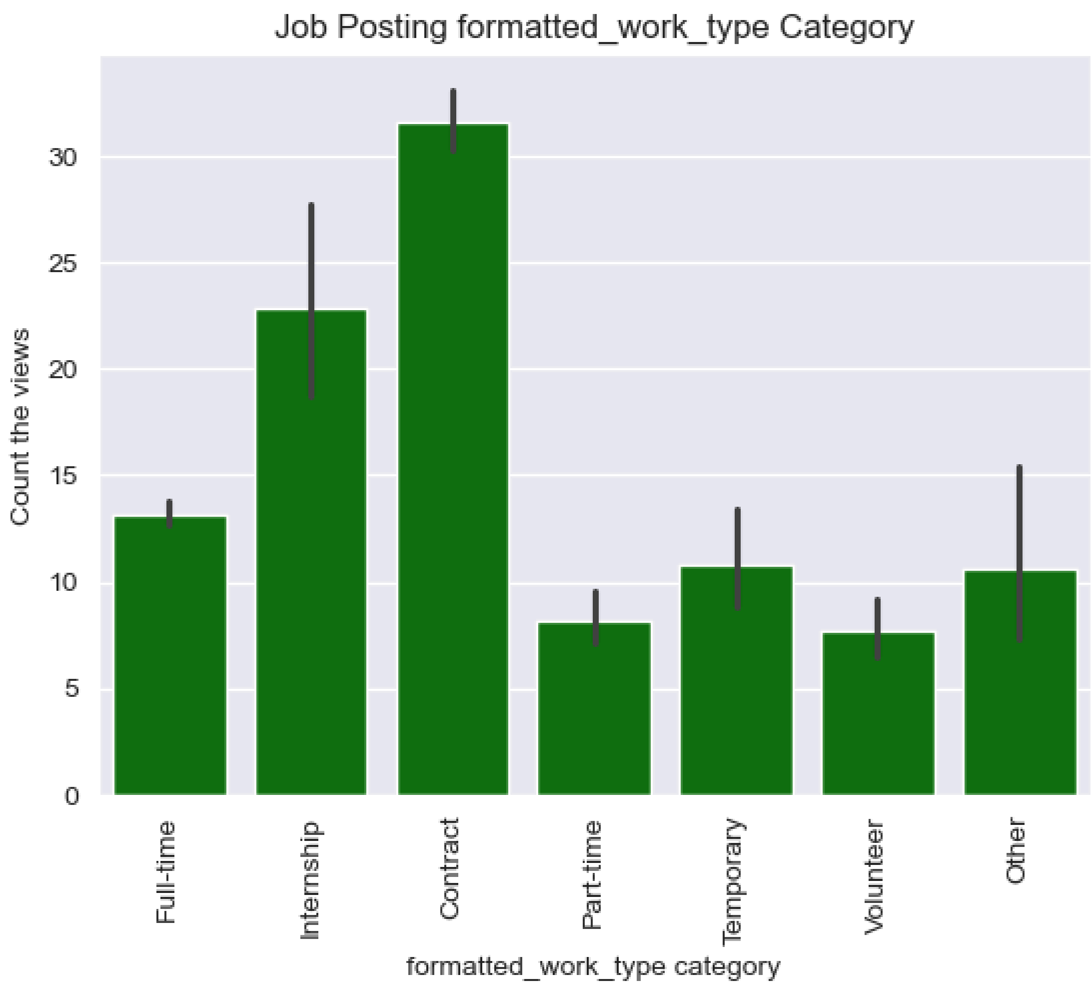
```
sns.barplot(x = 'pay_period',y = 'views',data = new_data,color = "orange")
plt.title('Job Posting pay_period Category')
plt.xlabel('pay_period category')
plt.ylabel('Count the pay_period')
plt.xticks(rotation = 90)
plt.show()
```



Most of the company offered the pay period category type is "Yearly" basis on the linkedin platform.

In [111…

```
# let's check the formatted work type wise views

sns.set_style("darkgrid")
sns.barplot(x = 'formatted_work_type',y = 'views',data = new_data,color = "green")
plt.title('Job Posting formatted_work_type Category')
plt.xlabel('formatted_work_type category')
plt.ylabel('Count the views')
plt.xticks(rotation = 90)
plt.show()
```

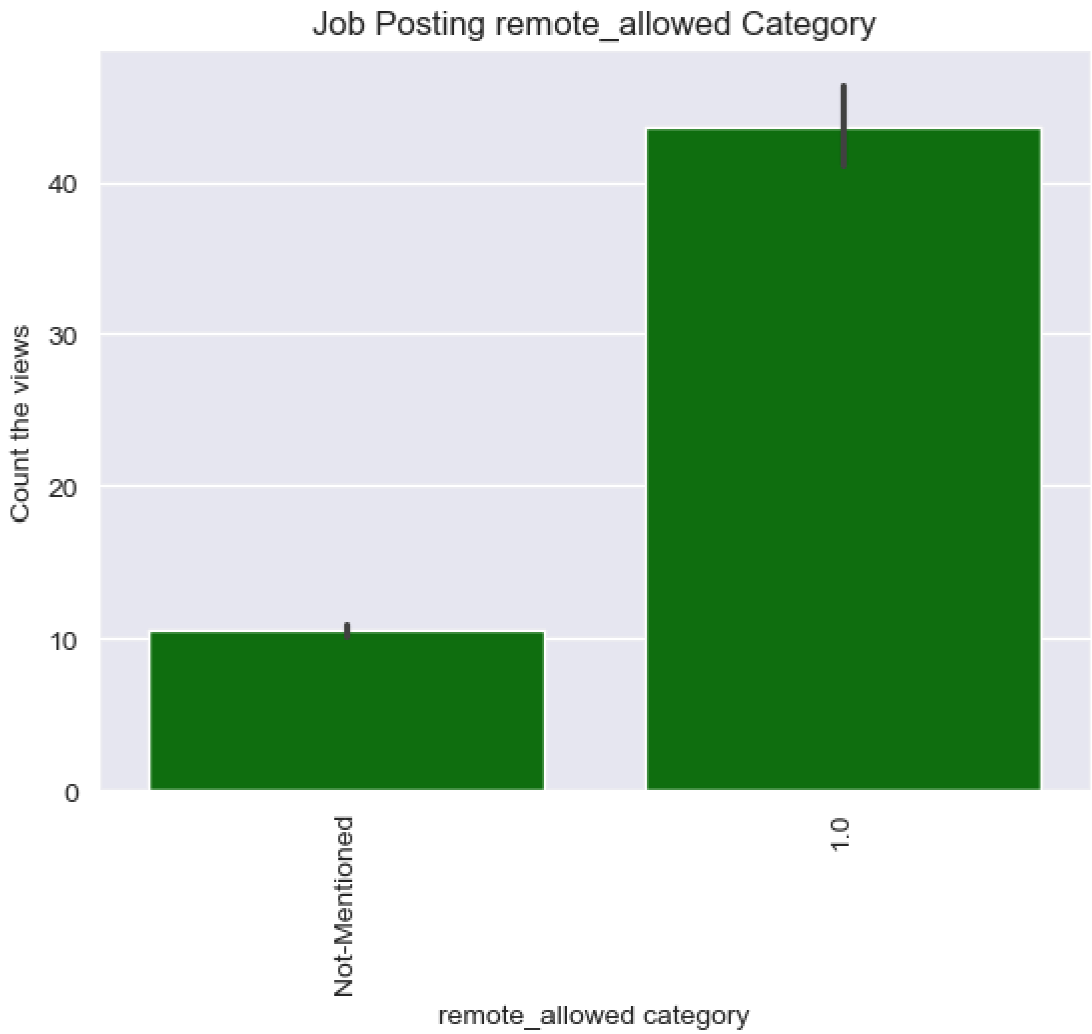


Most of the job posting offered the formatted work type category on 'Contract' bases on linkedin platform.

In [316…

```
# let's check the remote_allowed work type wise views

sns.set_style("darkgrid")
sns.barplot(x = 'remote_allowed',y = 'views',data = new_data,color = "green")
plt.title('Job Posting remote_allowed Category')
```

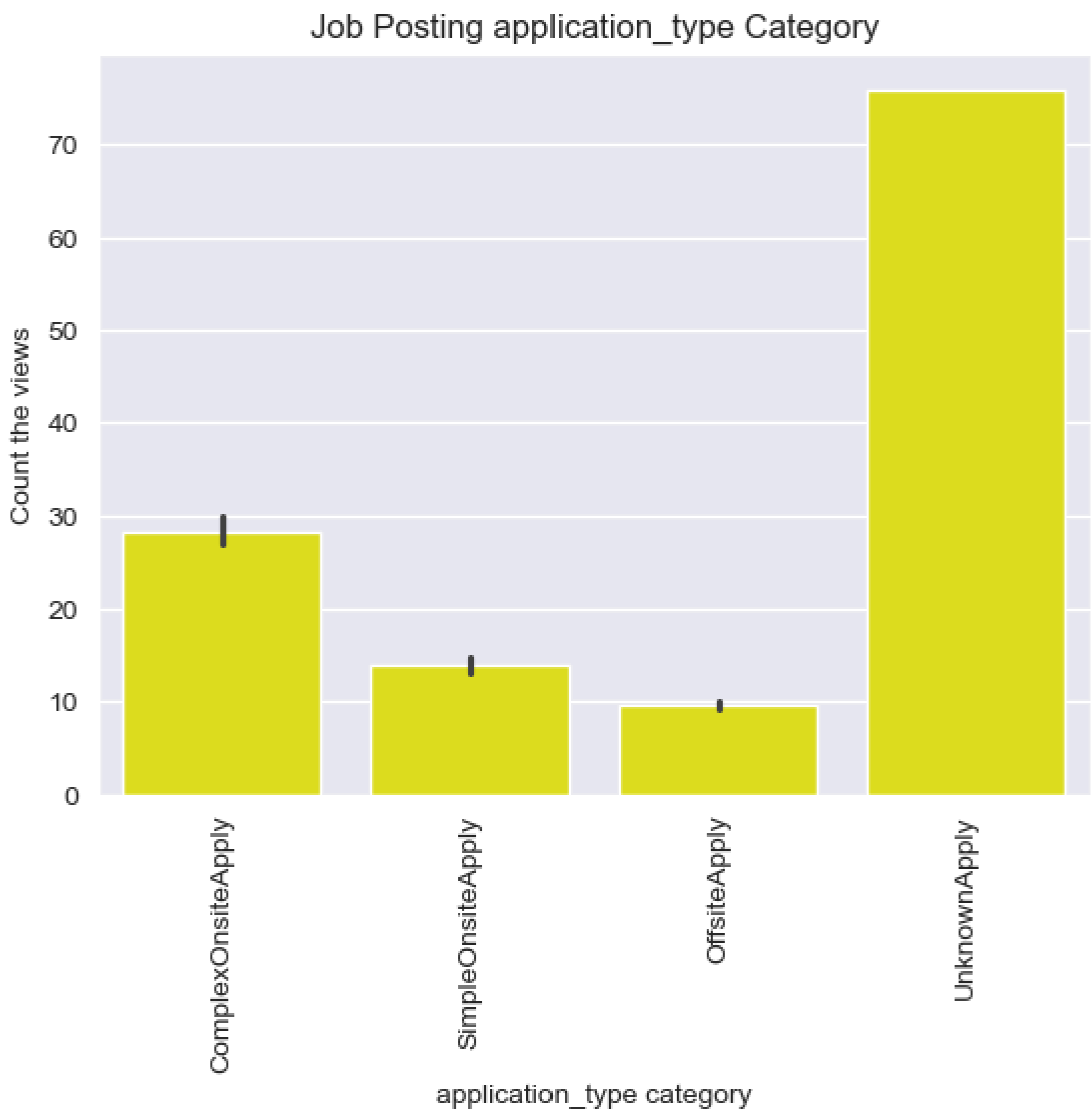
```
plt.xlabel('remote_allowed category')
plt.ylabel('Count the views')
plt.xticks(rotation = 90)
plt.show()
```



Most of the company offered remote allowed getting the highes views.

In [116…

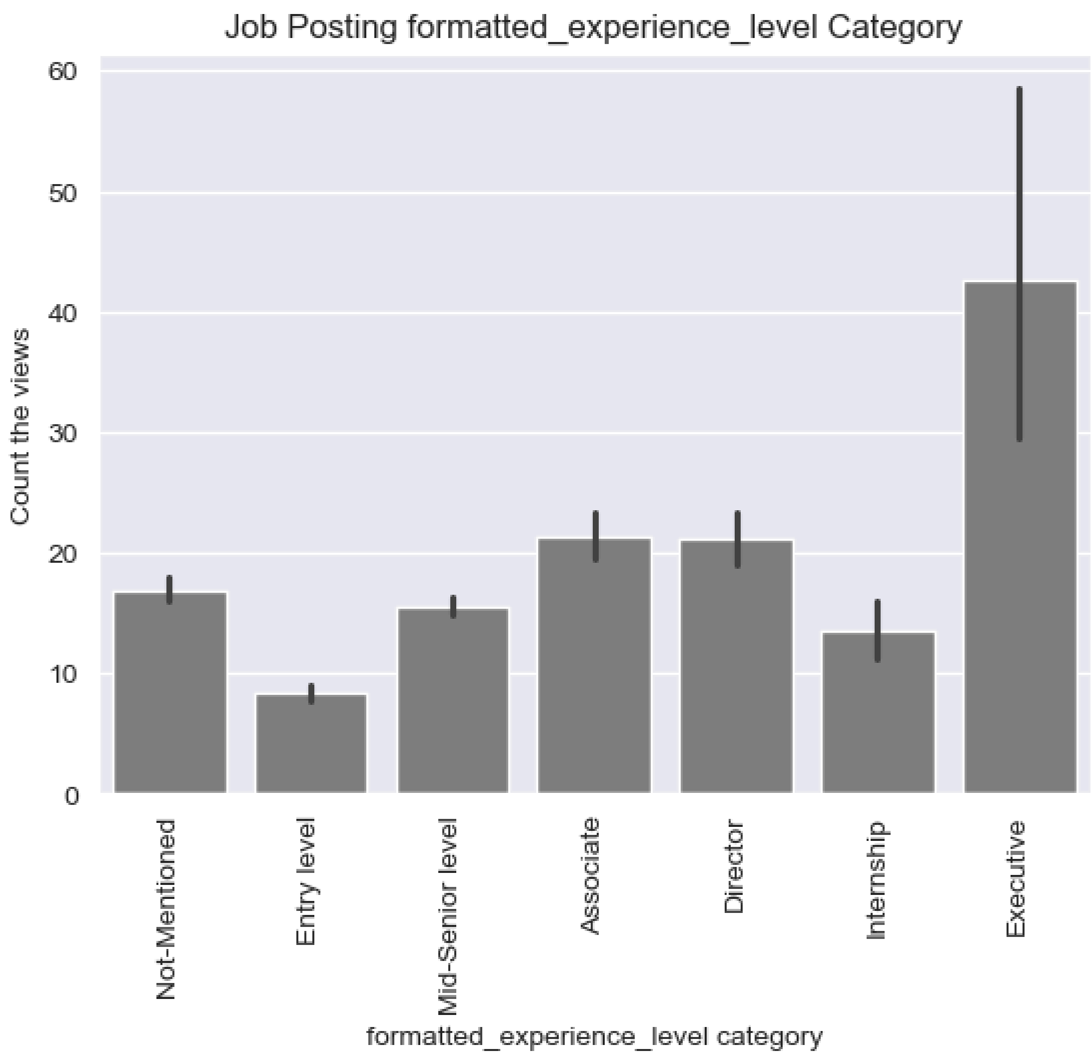
```
# let's check the application_type work type wise views

sns.set_style("darkgrid")
sns.barplot(x = 'application_type',y = 'views',data = new_data,color = "yellow")
plt.title('Job Posting application_type Category')
plt.xlabel('application_type category')
plt.ylabel('Count the views')
plt.xticks(rotation = 90)
plt.show()
```



Those who posted the jobs in 'Unknown Apply' category getting the highest views on linkedin platform.

In [118…

```
# let's check the formatted_experience_level work type wise views

sns.set_style("darkgrid")
sns.barplot(x = 'formatted_experience_level',y = 'views',data = new_data,color = "grey")
plt.title('Job Posting formatted_experience_level Category')
plt.xlabel('formatted_experience_level category')
plt.ylabel('Count the views')
plt.xticks(rotation = 90)
plt.show()
```

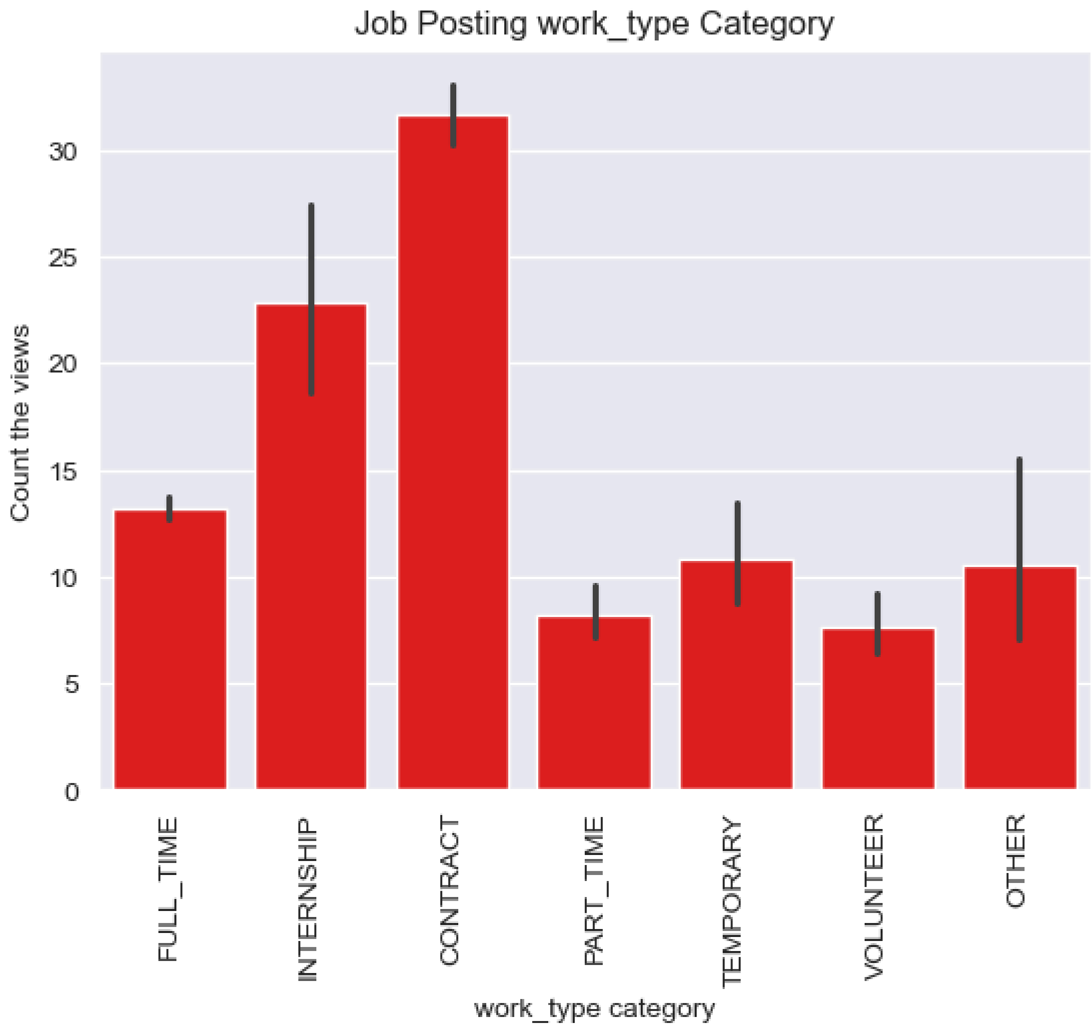


Here the highest variance in the views in 'Executive' formatted experience level but if we want accurate formatted experience level category then the winner is 'Associate' & 'Director' getting the highest views on linkedin platform.

In [121…

```
# let's check the work_type work type wise views
```

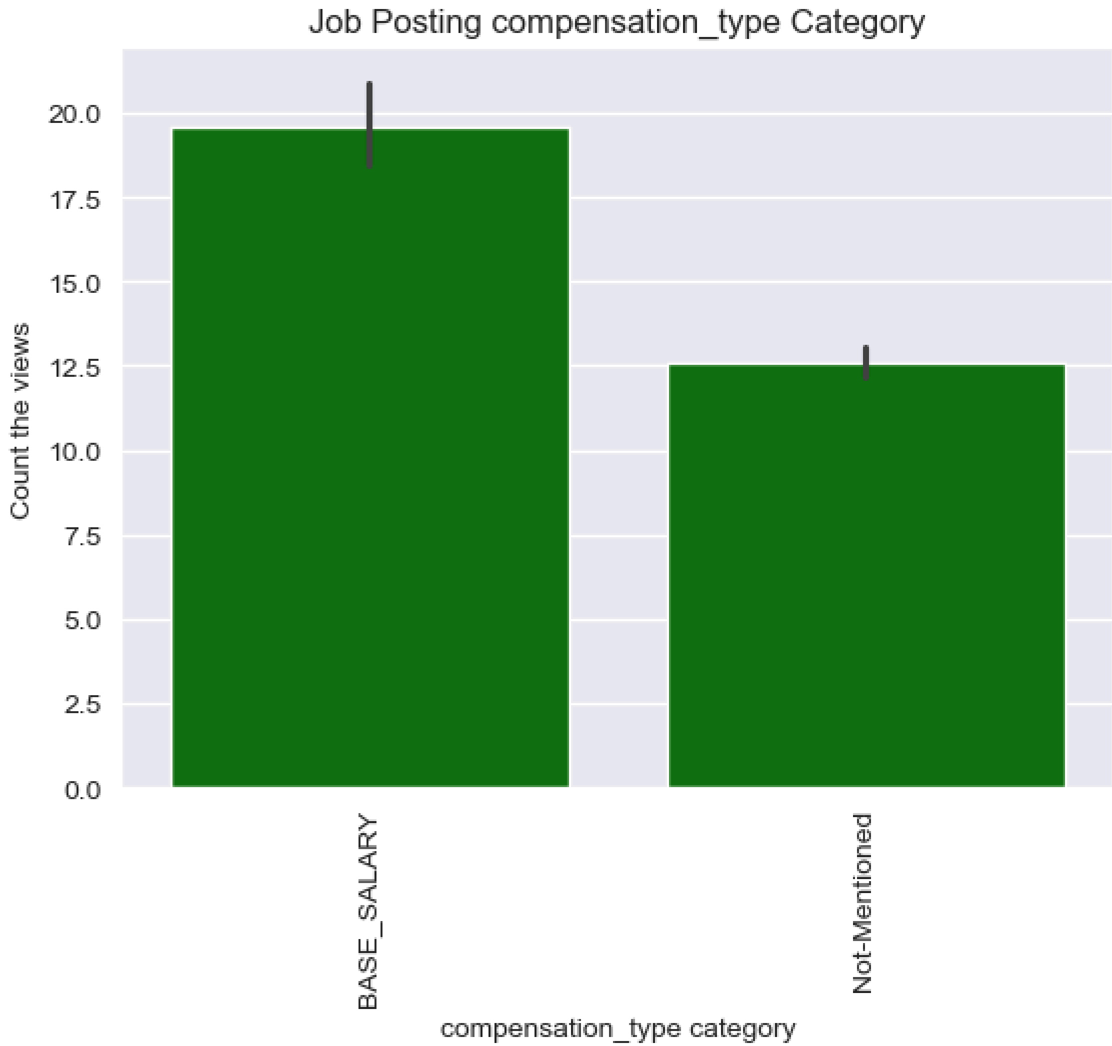
```
sns.set_style("darkgrid")
sns.barplot(x = 'work_type',y = 'views',data = new_data,color = "red")
plt.title('Job Posting work_type Category')
plt.xlabel('work_type category')
plt.ylabel('Count the views')
plt.xticks(rotation = 90)
plt.show()
```



You can see clearly 'Contract' based work type cetegory getting the highest public views on linkedin platform.

In [123…

```
# let's check the compensation_type work type wise views

sns.set_style("darkgrid")
sns.barplot(x = 'compensation_type',y = 'views',data = new_data,color = "green")
plt.title('Job Posting compensation_type Category')
plt.xlabel('compensation_type category')
plt.ylabel('Count the views')
plt.xticks(rotation = 90)
plt.show()
```

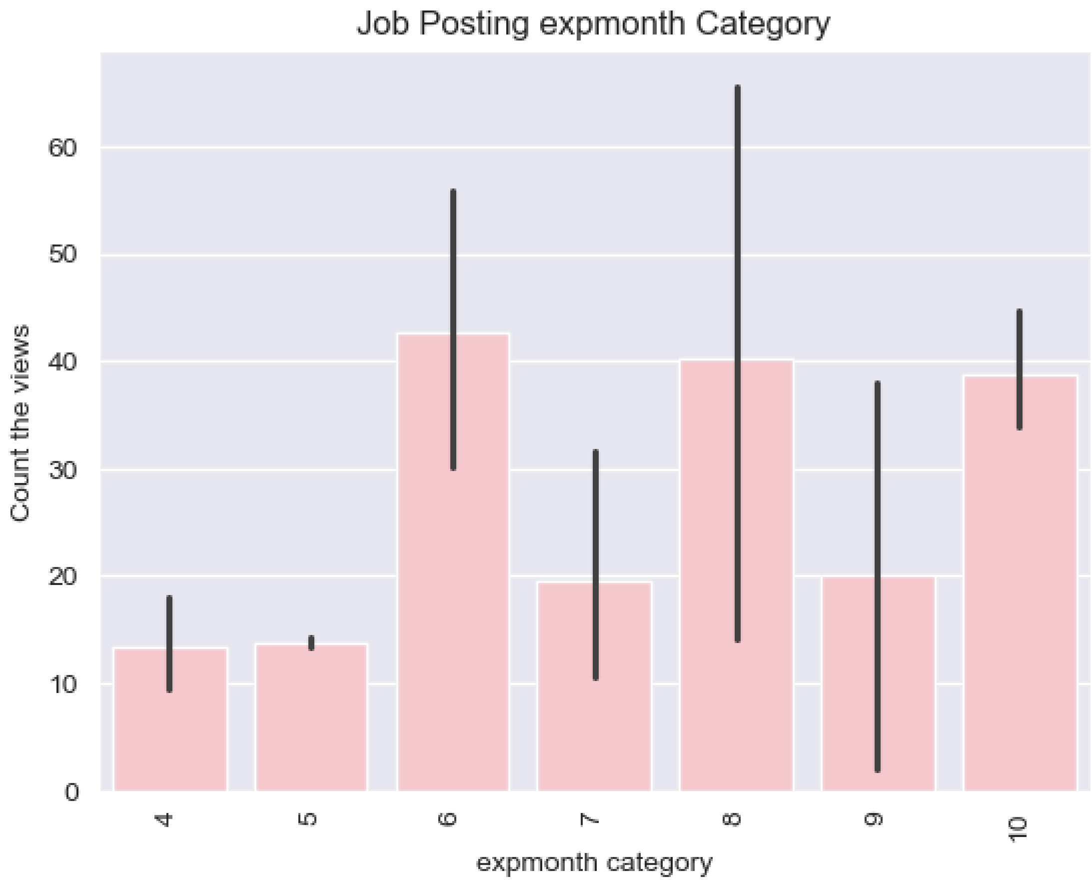


As you know that we have not suffienct or proper data to analyse compensation type column but in the improper data the winner is 'Base Salary' compensation type category getting the highest views on linkedin platform.

In [126…

```
# let's check the expmonth work type wise views
```

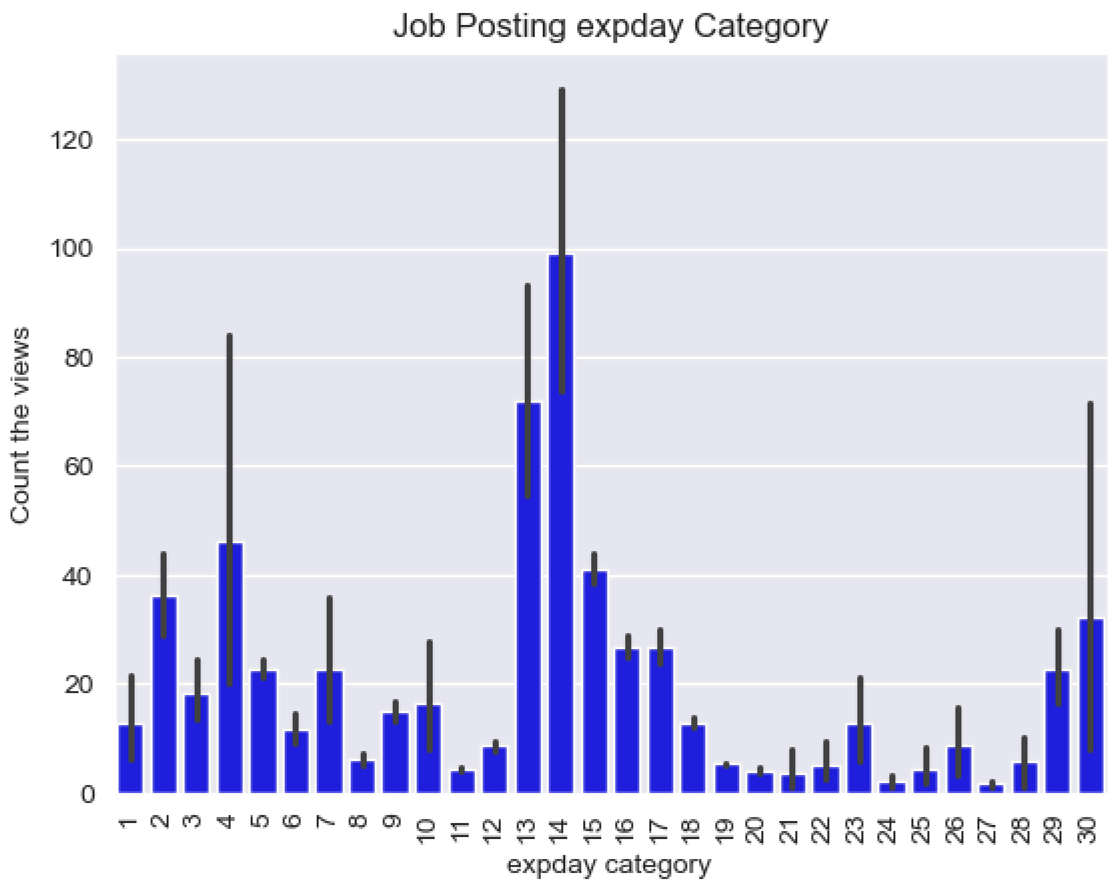
```
sns.set_style("darkgrid")
sns.barplot(x = 'expmonth',y = 'views',data = new_data,color = "pink")
plt.title('Job Posting expmonth Category')
plt.xlabel('expmonth category')
plt.ylabel('Count the views')
plt.xticks(rotation = 90)
plt.show()
```



Job posting expiry date in the month of 'October' getting the highest views may be at last viewers is high.

In [128…

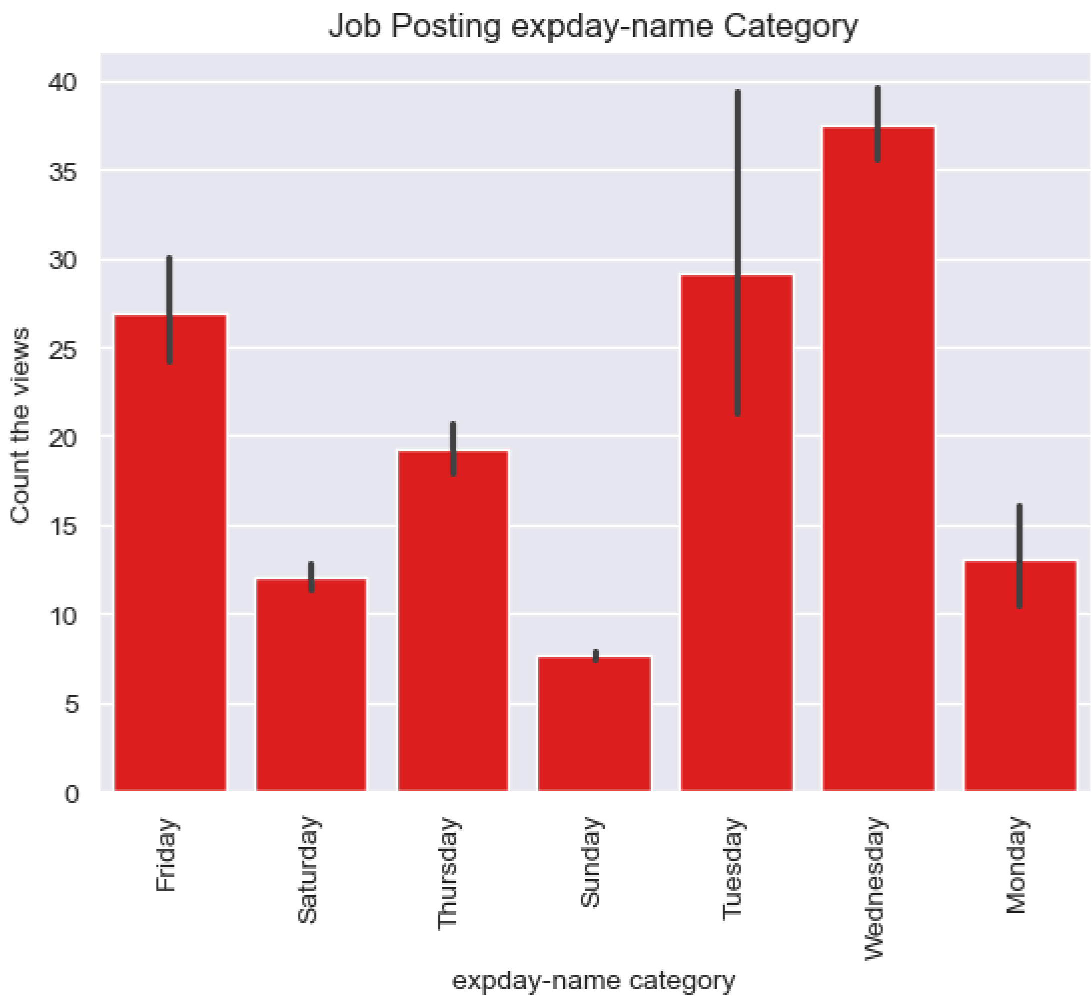
```
# let's check the expday work type wise views

sns.set_style("darkgrid")
sns.barplot(x = 'expday',y = 'views',data = new_data,color = "blue")
plt.title('Job Posting expday Category')
plt.xlabel('expday category')
plt.ylabel('Count the views')
plt.xticks(rotation = 90)
plt.show()
```



In the expiry date of 13th getting the highest views on linkedin platform.

In [130…

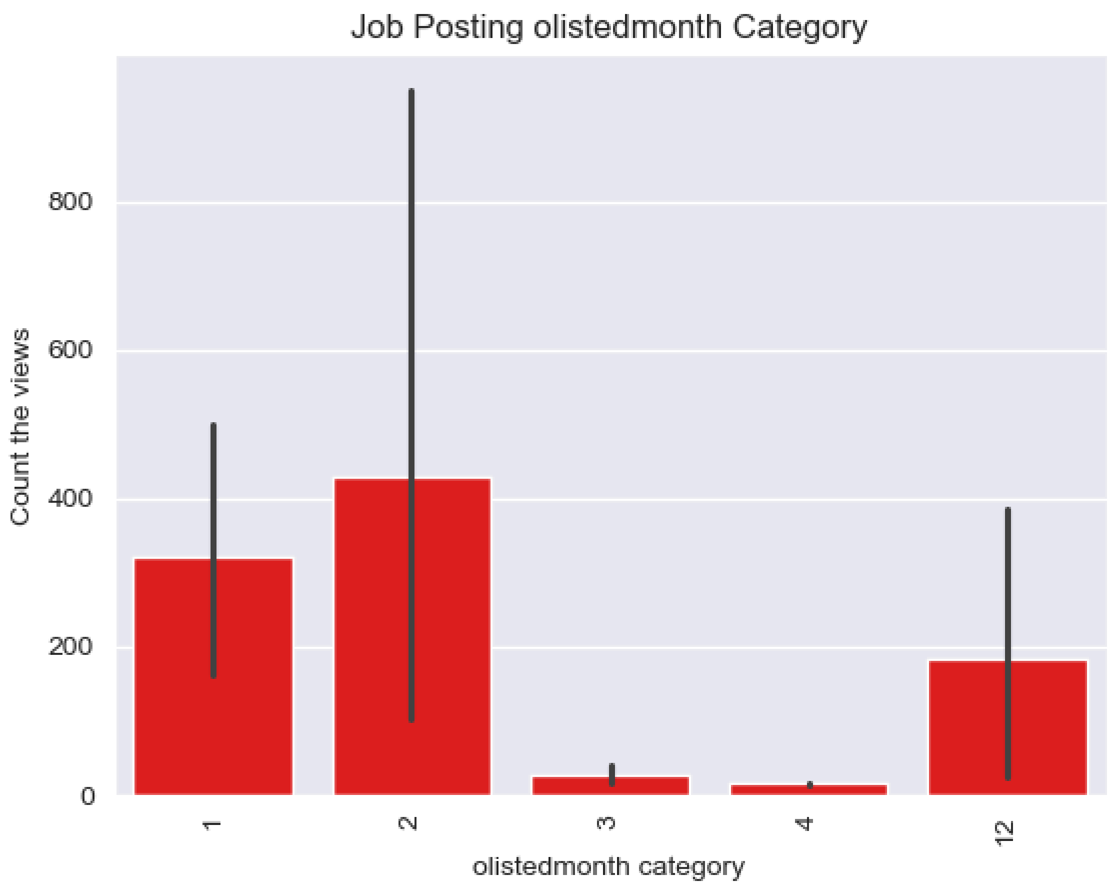
```
# let's check the expday-name work type wise views

sns.set_style("darkgrid")
sns.barplot(x = 'expday-name',y = 'views',data = new_data,color = "red")
plt.title('Job Posting expday-name Category')
plt.xlabel('expday-name category')
plt.ylabel('Count the views')
plt.xticks(rotation = 90)
plt.show()
```



Those job posting expiry date in on the day of 'Wednesday' getting the highest views on linkedin platform.

In [132…

```
# let's check the olistedmonth work type wise views

sns.set_style("darkgrid")
sns.barplot(x = 'olistedmonth',y = 'views',data = new_data,color = "red")
plt.title('Job Posting olistedmonth Category')
plt.xlabel('olistedmonth category')
plt.ylabel('Count the views')
plt.xticks(rotation = 90)
plt.show()
```

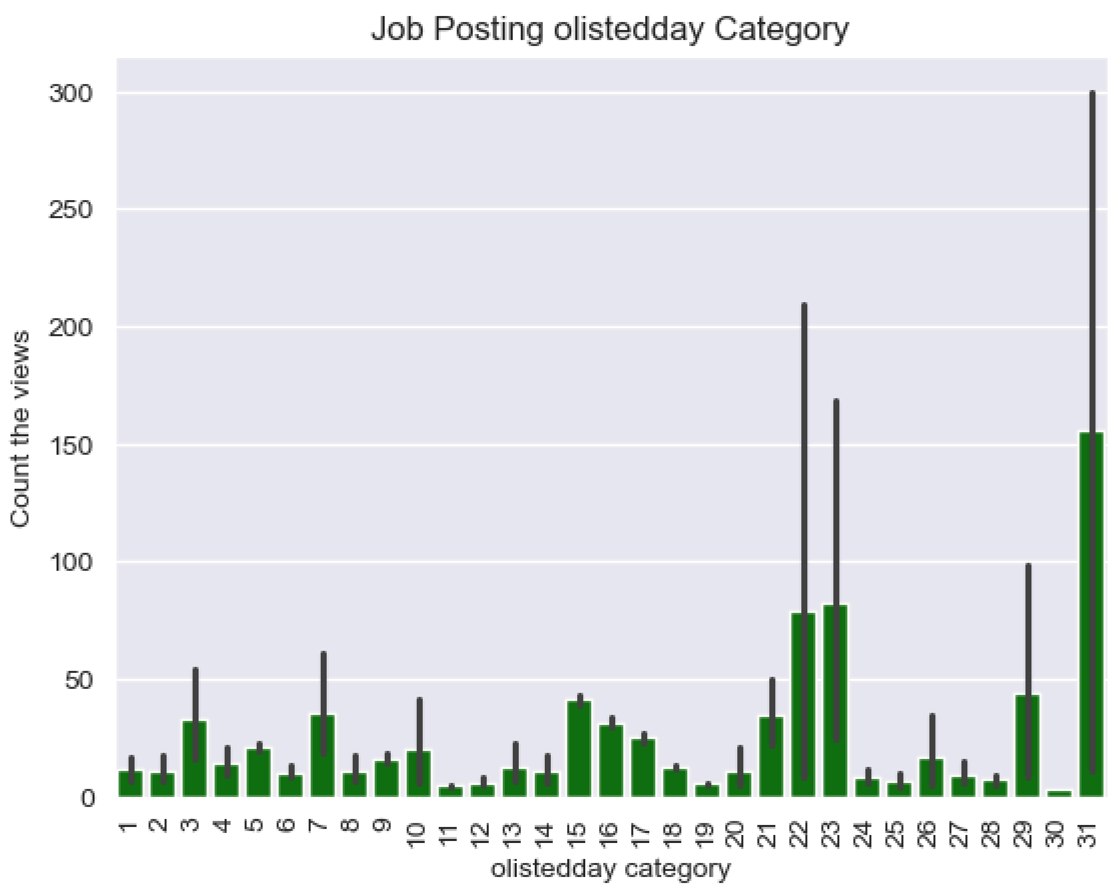


As you can see clearly in the chart in the listed month 1,12 getting the highest views but if you get more variance then go with feb month.

In [350…

```
# let's check the olistedday work type wise views

sns.set_style("darkgrid")
sns.barplot(x = 'olistedday',y = 'views',data = new_data,color = "green")
plt.title('Job Posting olistedday Category')
plt.xlabel('olistedday category')
plt.ylabel('Count the views')
```

```
plt.xticks(rotation = 90)
plt.show()
```

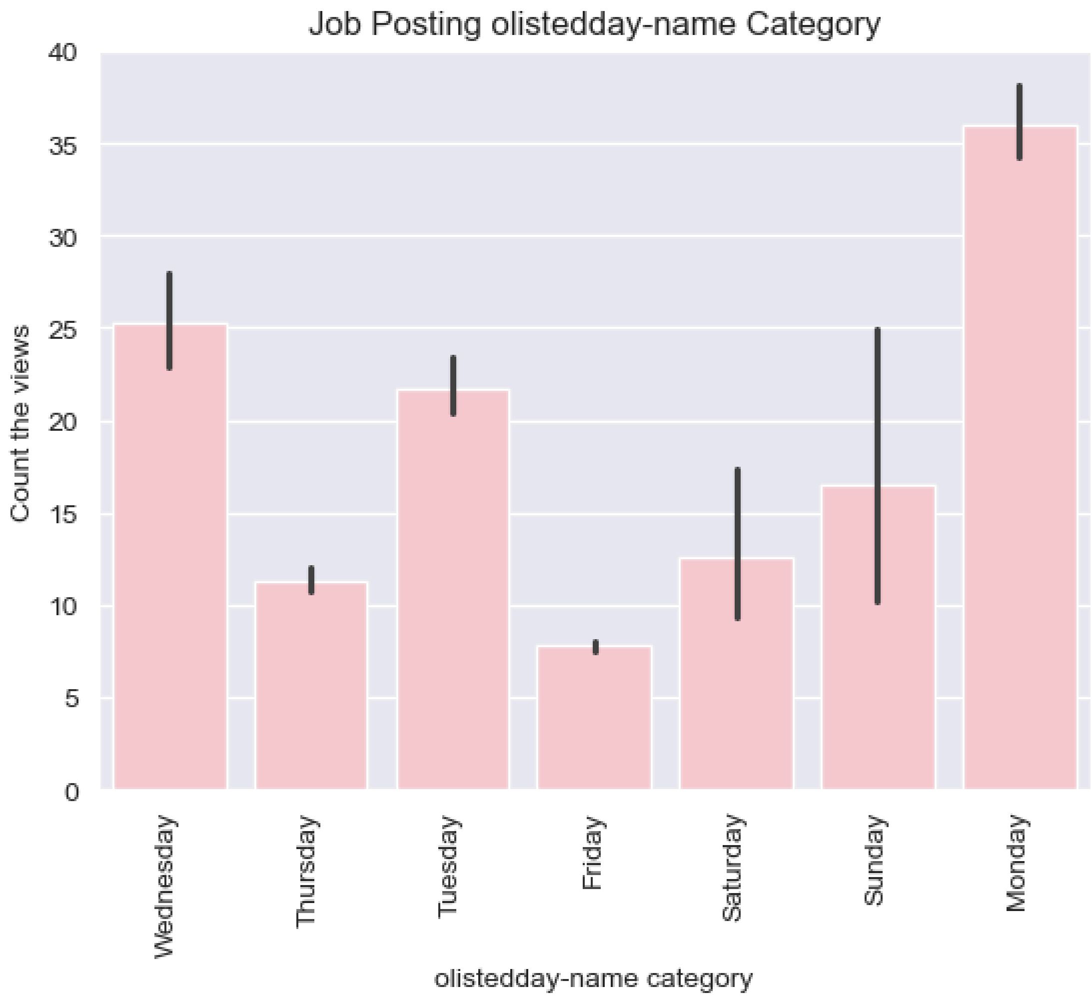


You can see clearly in the listedday category in job postings in the date of 22,23,31 getting the highest variance in views.

In [137…

```
# let's check the olistedday-name work type wise views

sns.set_style("darkgrid")
sns.barplot(x = 'olistedday-name',y = 'views',data = new_data,color = "pink")
plt.title('Job Posting olistedday-name Category')
```

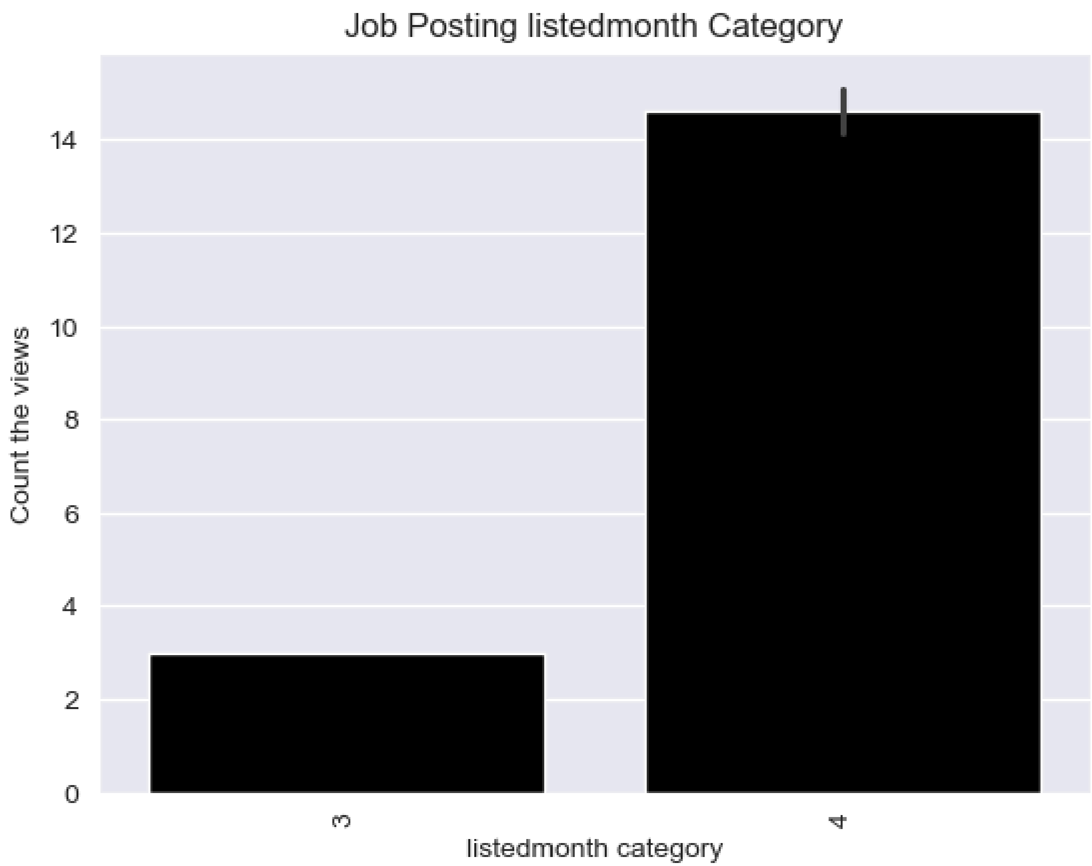
```
plt.xlabel('olistedday-name category')
plt.ylabel('Count the views')
plt.xticks(rotation = 90)
plt.show()
```



Those company's job posting listed on the day of 'Monday' getting the highest views on linkedin platform.

In [139…

```
# let's check the listedmonth work type wise views

sns.set_style("darkgrid")
sns.barplot(x = 'listedmonth',y = 'views',data = new_data,color = "black")
plt.title('Job Posting listedmonth Category')
plt.xlabel('listedmonth category')
plt.ylabel('Count the views')
plt.xticks(rotation = 90)
plt.show()
```



You can see clearly the winner is job posting listed month is April getting the highest public views on linkedin platform.

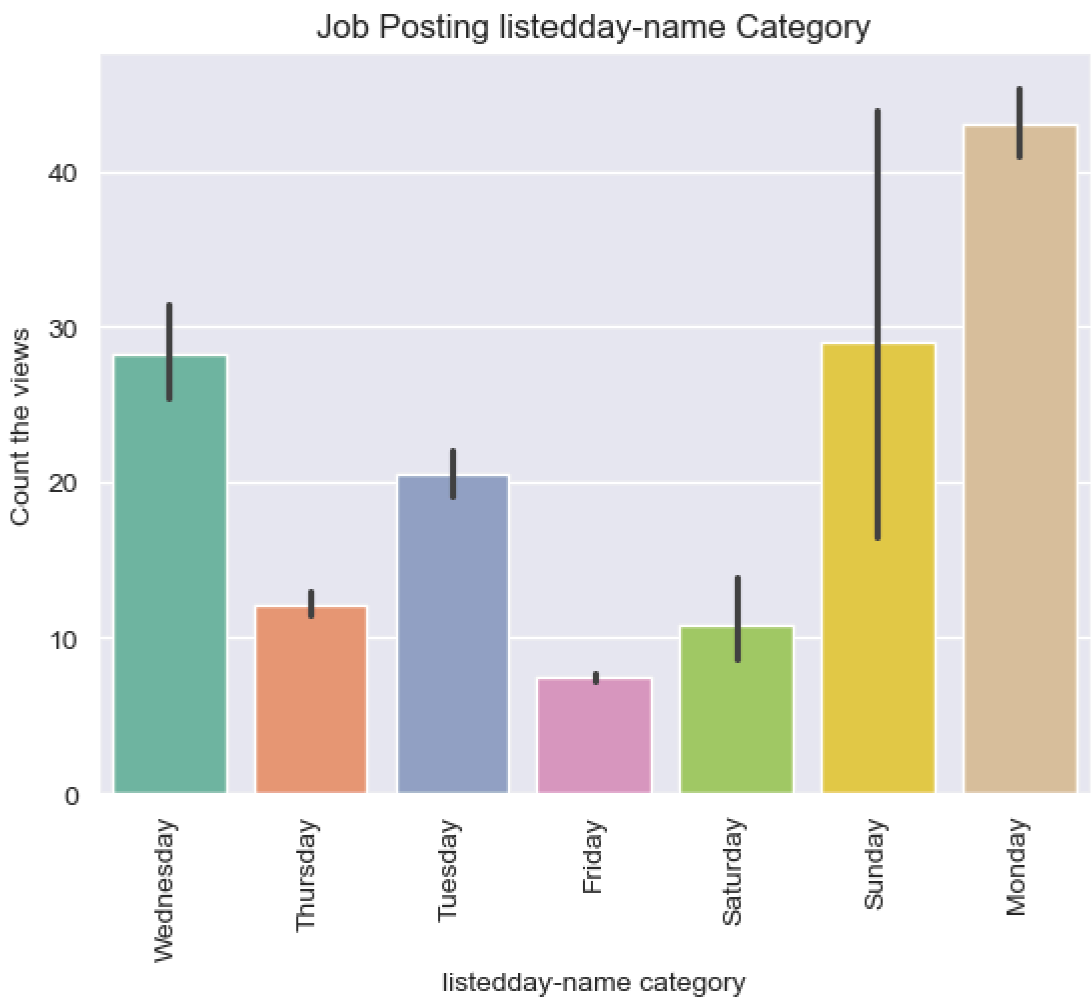
In [141…

```
# let's check the listedday work type wise views

sns.set_style("darkgrid")
sns.barplot(x = 'listedday',y = 'views',data = new_data,palette = "Set1")
plt.title('Job Posting listedday Category')
plt.xlabel('listedday category')
plt.ylabel('Count the views')
plt.xticks(rotation = 90)
plt.show()
```

Those company job posting listed day in the date of 15th getting the highest views on linkedin platform.

In [143…

```
# let's check the listedday-name work type wise views

sns.set_style("darkgrid")
sns.barplot(x = 'listedday-name',y = 'views',data = new_data,palette = "Set2")
plt.title('Job Posting listedday-name Category')
plt.xlabel('listedday-name category')
plt.ylabel('Count the views')
plt.xticks(rotation = 90)
plt.show()
```

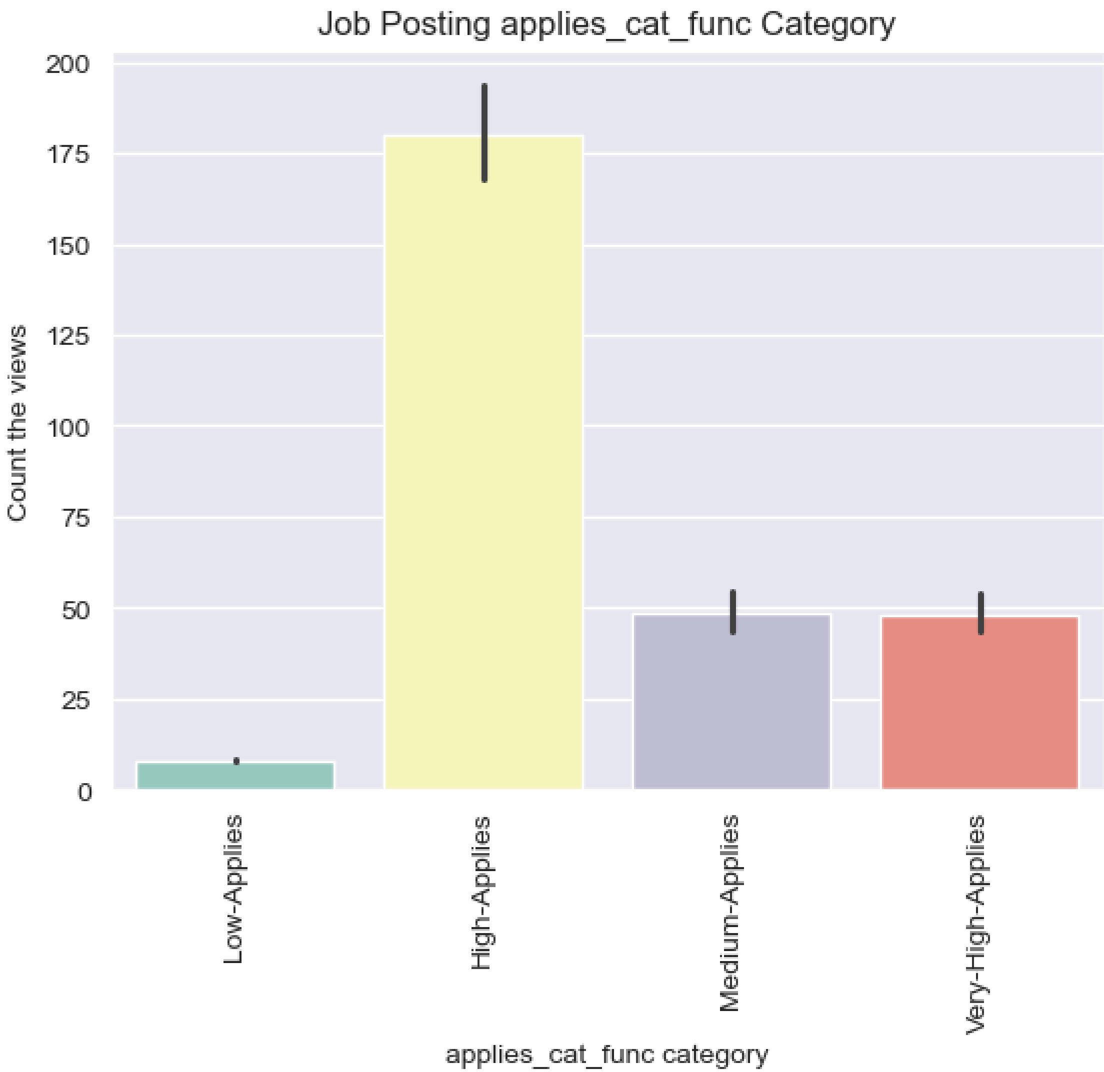


You can see clearly those company's job postings on the day of 'Monday' getting the highest views but "Sunday" get more variance in the views on linkedin platform.

In [146…

```
# let's check the applies_cat_func work type wise views

sns.set_style("darkgrid")
```

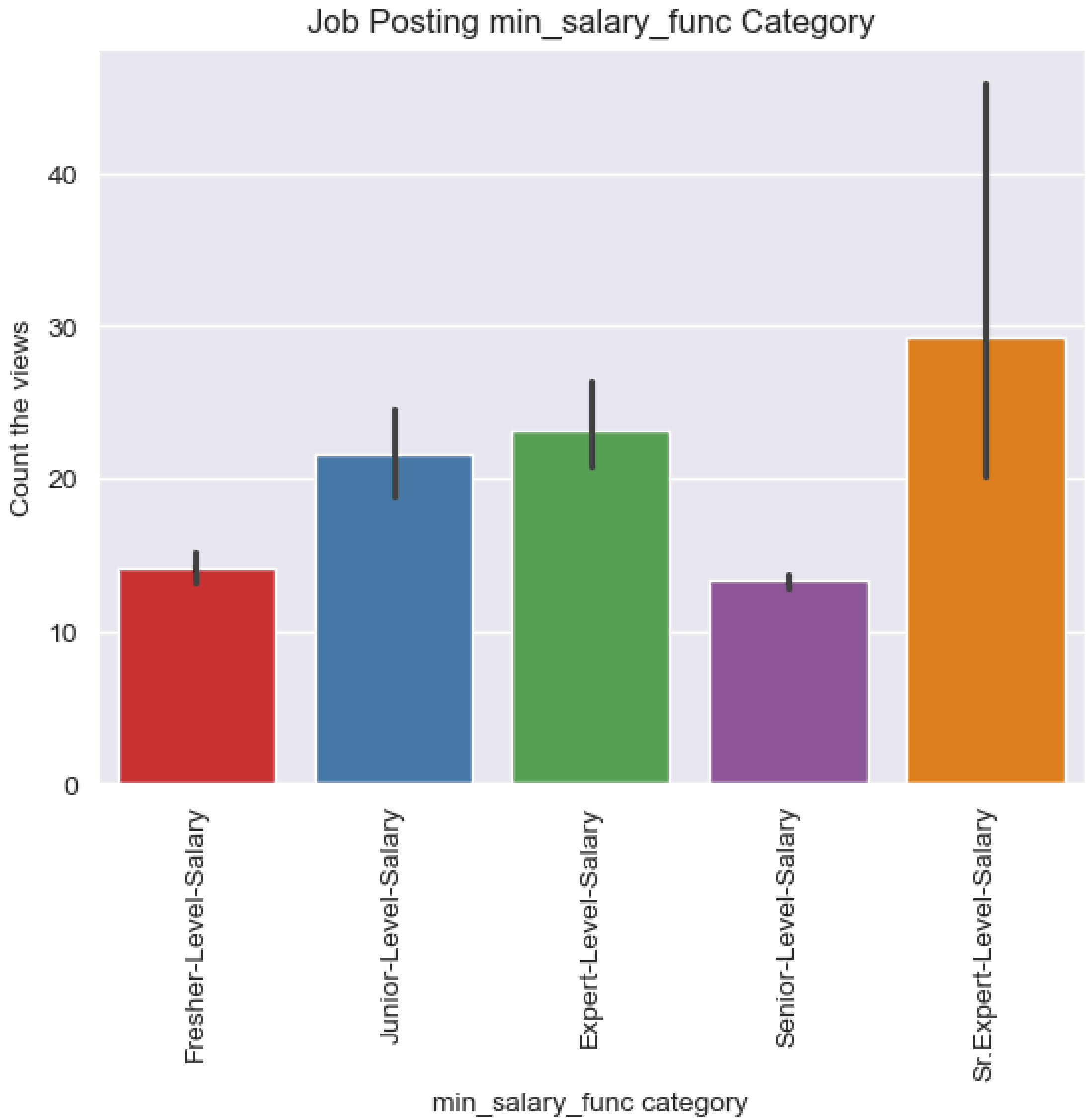
```
sns.barplot(x = 'applies_cat_func',y = 'views',data = new_data,palette = "Set3")
plt.title('Job Posting applies_cat_func Category')
plt.xlabel('applies_cat_func category')
plt.ylabel('Count the views')
plt.xticks(rotation = 90)
plt.show()
```



In the job posting applicants is high getting the highest views on it.

In [148…

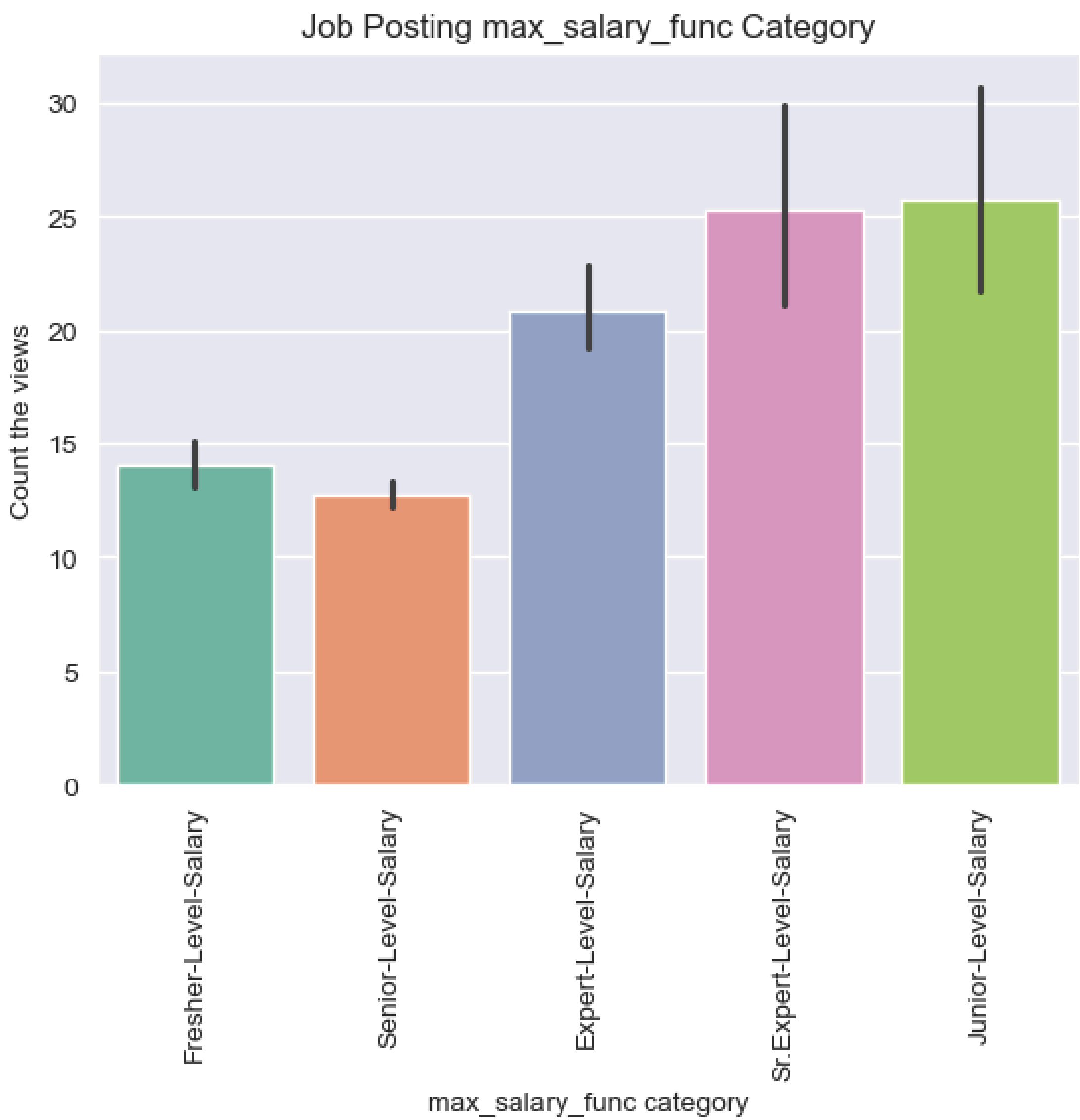
```
# let's check the min_salary_func work type wise views

sns.set_style("darkgrid")
sns.barplot(x = 'min_salary_func',y = 'views',data = new_data,palette = "Set1")
plt.title('Job Posting min_salary_func Category')
plt.xlabel('min_salary_func category')
plt.ylabel('Count the views')
plt.xticks(rotation = 90)
plt.show()
```



Those company offer the minimum salary in the category of 'Expert Level Salary' & 'Sr.Expert Level Salry' getting the highest views on linkedin platform.

In [150…

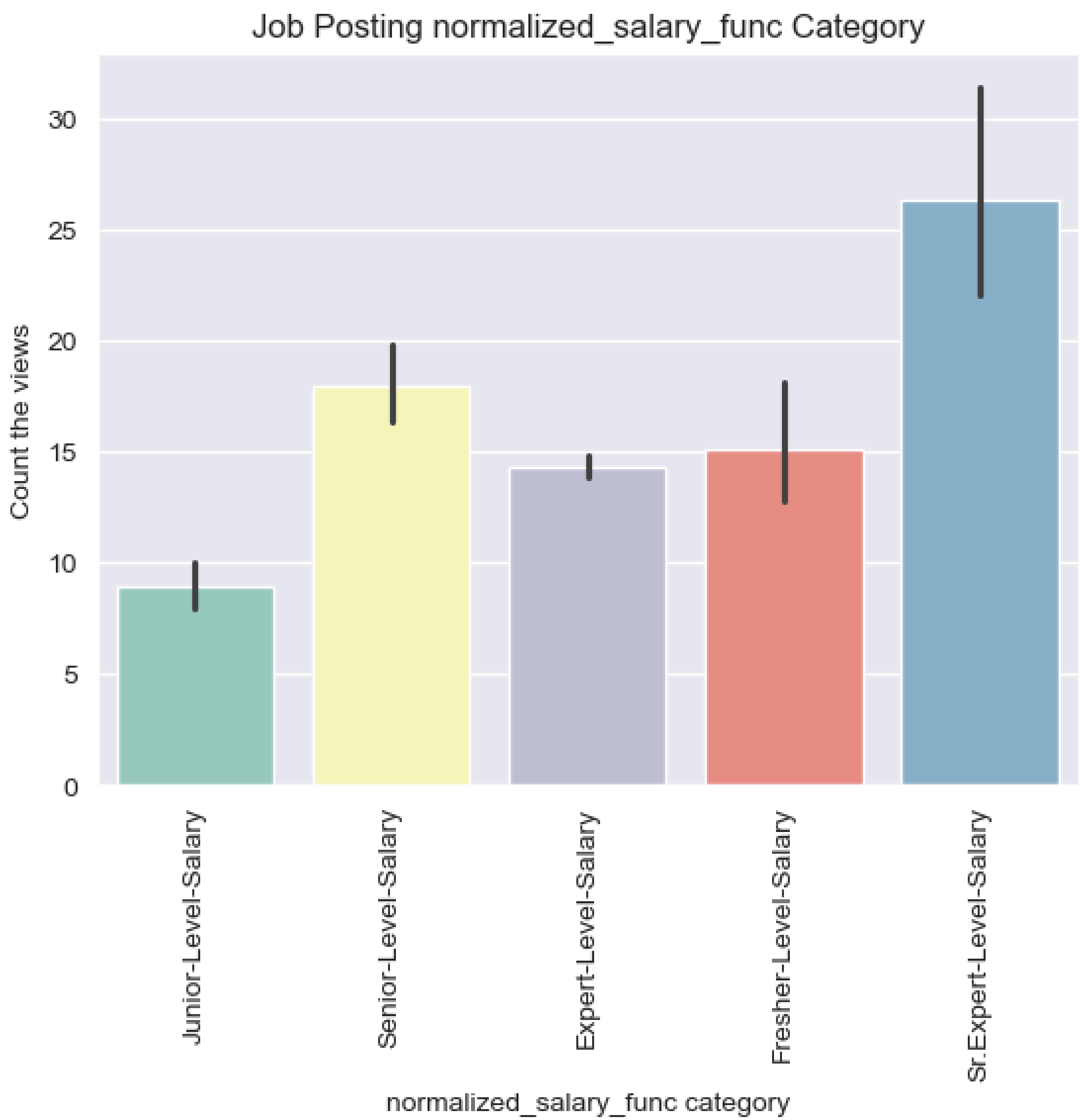
```
# let's check the max_salary_func work type wise views

sns.set_style("darkgrid")
sns.barplot(x = 'max_salary_func',y = 'views',data = new_data,palette = "Set2")
plt.title('Job Posting max_salary_func Category')
plt.xlabel('max_salary_func category')
plt.ylabel('Count the views')
plt.xticks(rotation = 90)
plt.show()
```



Those company offer the maximum salary in the category of 'Sr.Expert Level' & 'Junior Level Salary' getting the highest views.

In [152…

```
# let's check the normalized_salary_func work type wise views

sns.set_style("darkgrid")
sns.barplot(x = 'normalized_salary_func',y = 'views',data = new_data,palette = "Set3")
plt.title('Job Posting normalized_salary_func Category')
plt.xlabel('normalized_salary_func category')
plt.ylabel('Count the views')
plt.xticks(rotation = 90)
plt.show()
```

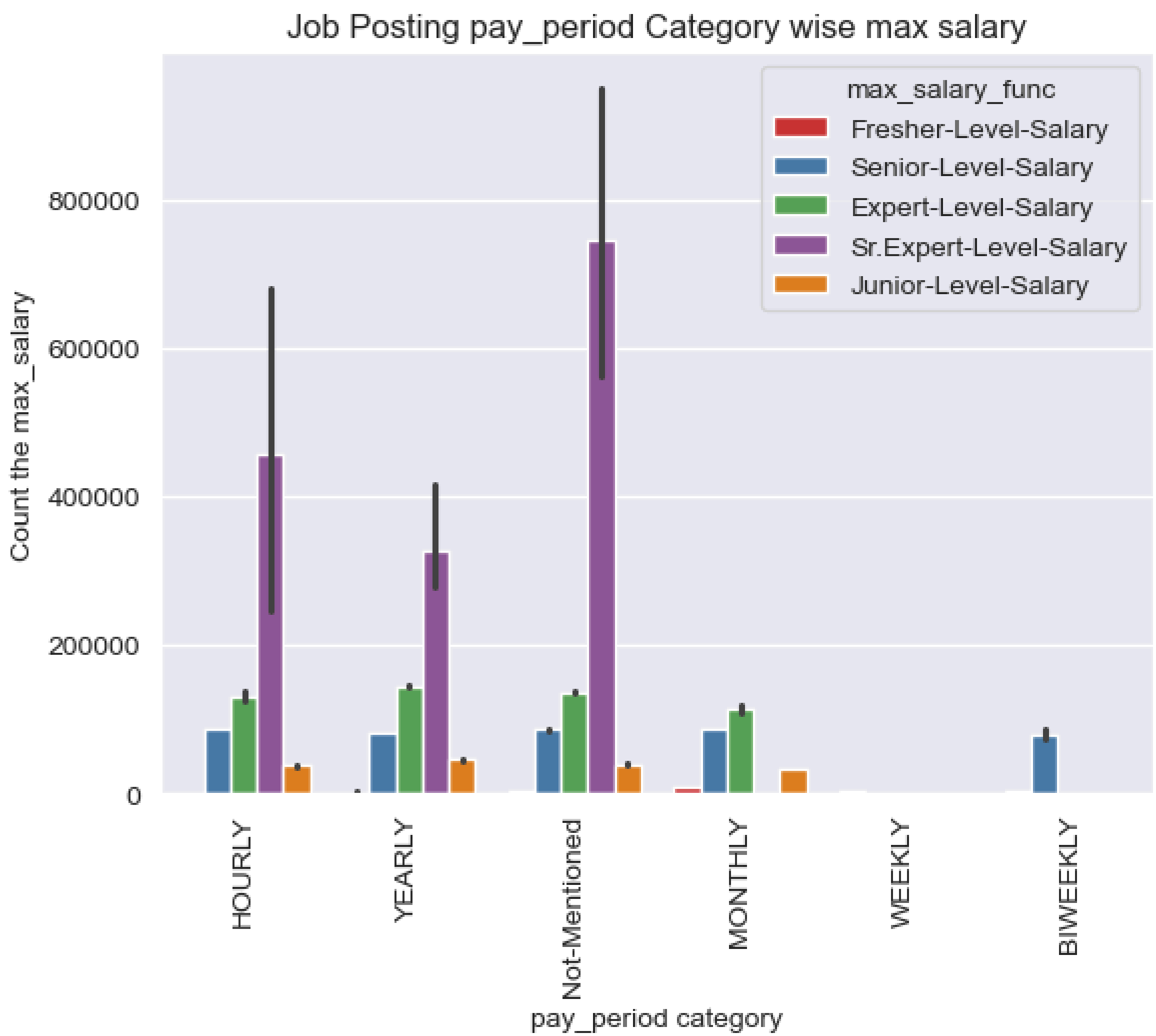


Those company offers the normalized salary in the category of 'Sr.Expert Level' getting the highest views.

Multi-Variate-Analysis

In [156…

```
# let's check the pay_period work type wise max_salary

sns.set_style("darkgrid")
sns.barplot(x = 'pay_period',y = 'max_salary',hue = "max_salary_func",data = new_data,palette = "Set1")
plt.title('Job Posting pay_period Category wise max salary')
plt.xlabel('pay_period category')
plt.ylabel('Count the max_salary')
plt.xticks(rotation = 90)
plt.show()
```



You can see sr.Expert Level Salary category getting the highest maximum salary on the 'Hourly' basis.

In [158…

```
# let's check the pay_period work type wise min_salary

sns.set_style("darkgrid")
```

```
sns.barplot(x = 'pay_period',y = 'min_salary',hue = "min_salary_func",data = new_data,palette = "Set2")
plt.title('Job Posting pay_period Category wise min salary')
plt.xlabel('pay_period category')
plt.ylabel('Count the min_salary')
plt.xticks(rotation = 90)
plt.show()
```

Interesting thing is in the opposite side 'Sr.Expert Level' getting the highest salary in the the minimum salary category but their paytype is not-mentioned.

In [160…

```
# let's check the formatted_work_type work type wise max_salary

sns.set_style("darkgrid")
sns.barplot(x = 'formatted_work_type',y = 'max_salary',hue = "max_salary_func",data = new_data,palette = "Set3")
plt.title('Job Posting formatted_work_type Category wise max salary')
plt.xlabel('formatted_work_type category')
plt.ylabel('Count the max_salary')
plt.xticks(rotation = 90)
plt.show()
```

As you know that 'Sr.Expert Level Salary' getting the highest max salary in all formatted work types.

In [162…

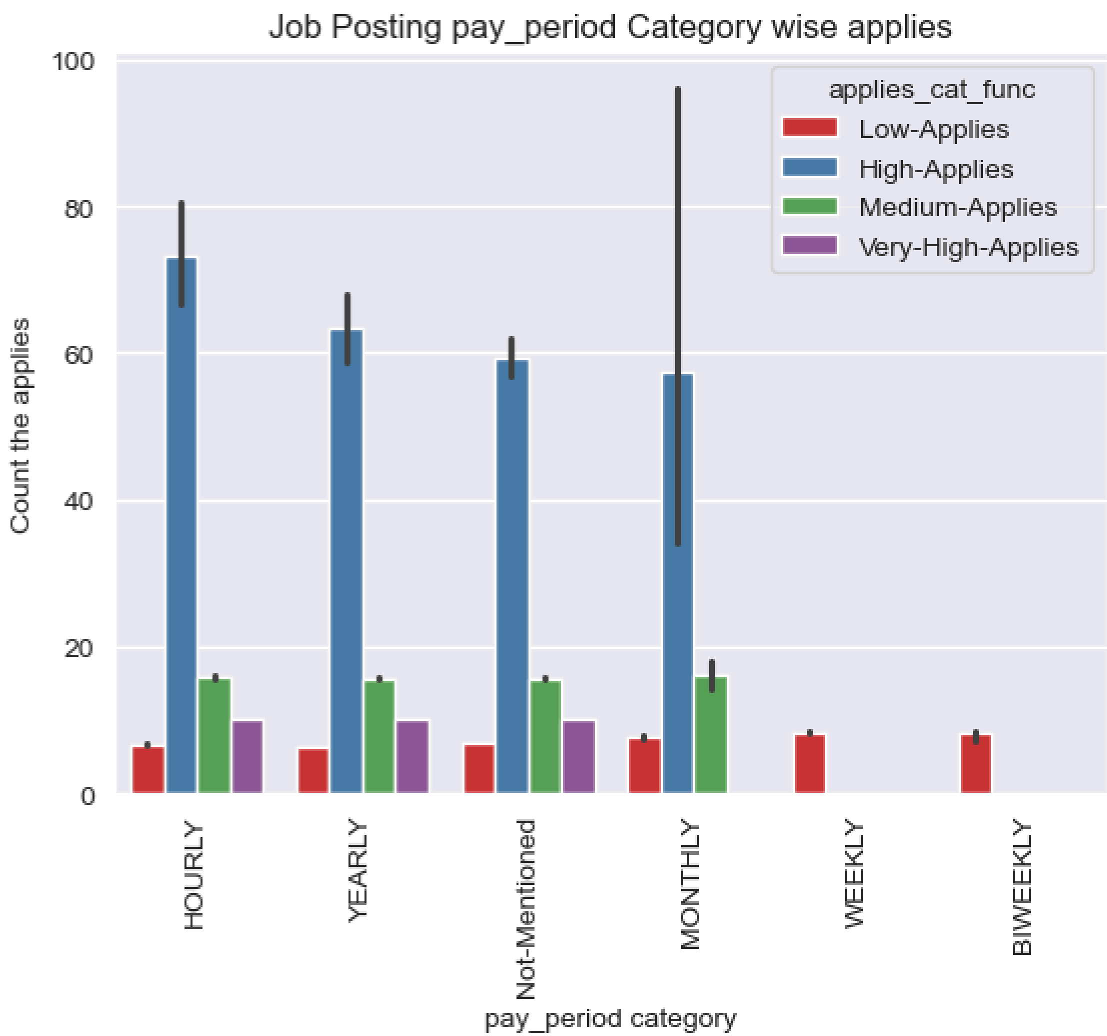
```
# let's check the formatted_work_type work type wise min_salary

sns.set_style("darkgrid")
sns.barplot(x = 'formatted_work_type',y = 'min_salary',hue = "min_salary_func",data = new_data,palette = "Set2")
plt.title('Job Posting formatted_work_type Category wise min salary')
```

```
plt.xlabel('formatted_work_type category')
plt.ylabel('Count the min_salary')
plt.xticks(rotation = 90)
plt.show()
```

Here again the winner is 'Sr.Expert Level' category earns higher minimum salary among all the formatted work type categories.

In [165…

```
# let's check the pay_period work type wise applies

sns.set_style("darkgrid")
sns.barplot(x = 'pay_period',y = 'applies',hue = "applies_cat_func",data = new_data,palette = "Set1")
plt.title('Job Posting pay_period Category wise applies')
plt.xlabel('pay_period category')
plt.ylabel('Count the applies')
plt.xticks(rotation = 90)
plt.show()
```

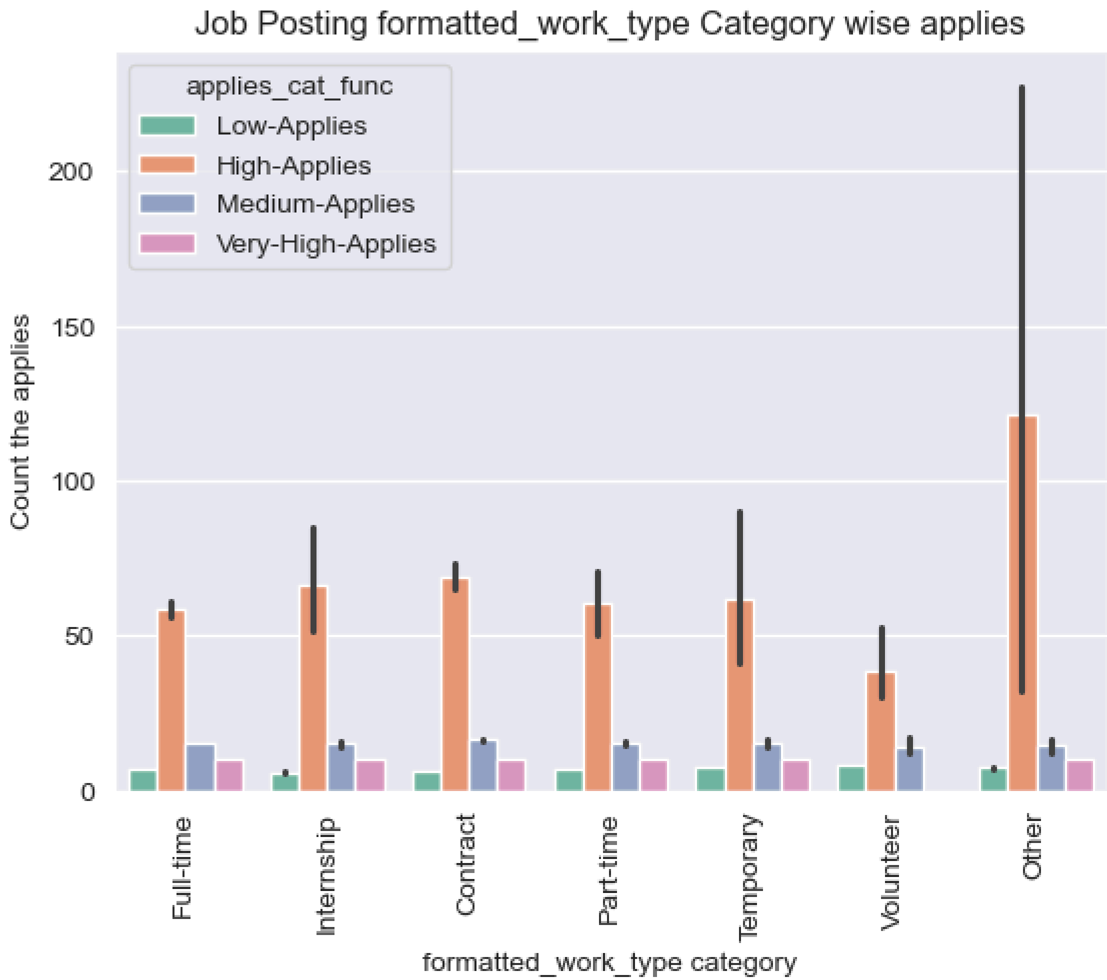


You can seel clearly company offeres salary on the basis of 'Hourly','Yearly','Not-Mentioned','Monthly' pay-period category getting the highest applies & 'Weekly','Biweekly' getting the low applies on linkedin platform.

In [167…

```
# let's check the formatted_work_type work type wise applies
```

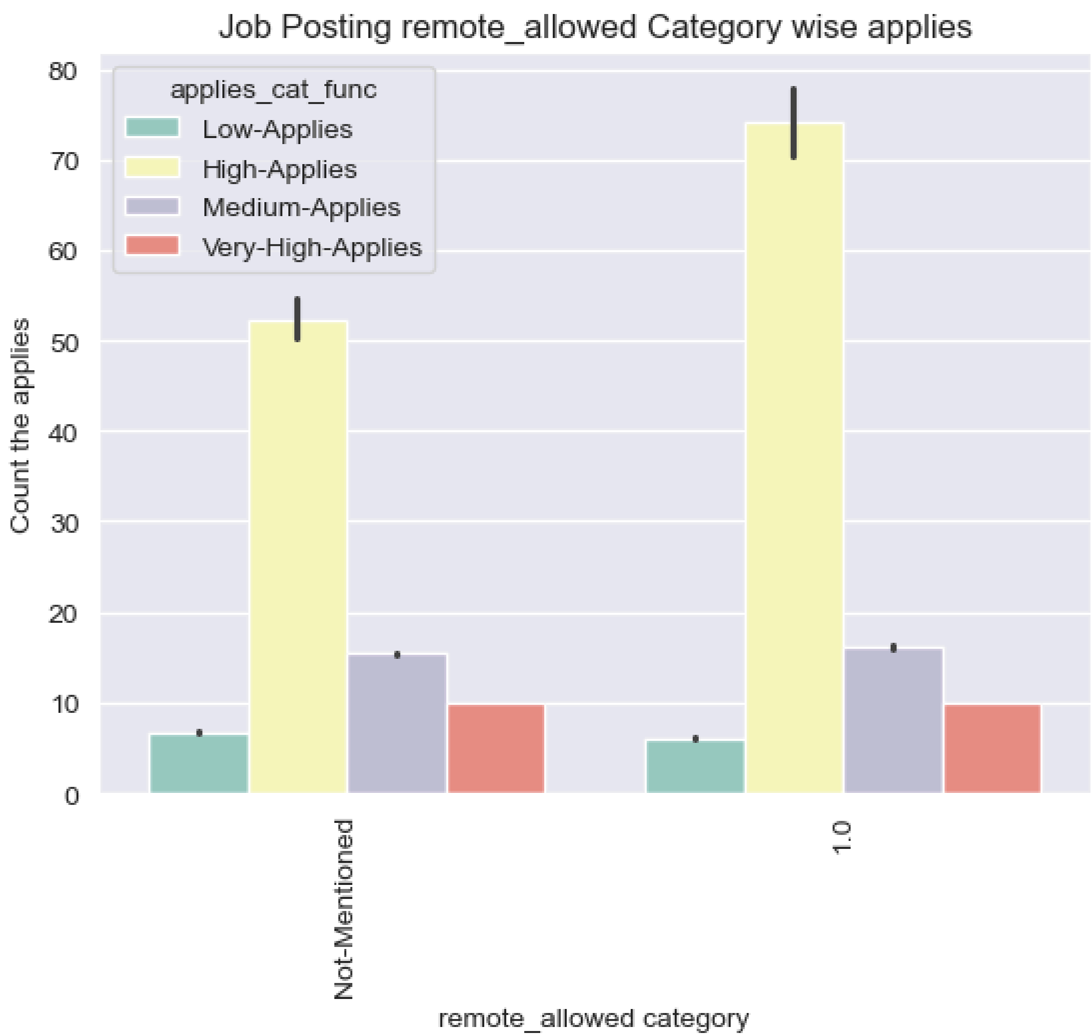
```
sns.set_style("darkgrid")
sns.barplot(x = 'formatted_work_type',y = 'applies',hue = "applies_cat_func",data = new_data,palette = "Set2")
plt.title('Job Posting formatted_work_type Category wise applies')
plt.xlabel('formatted_work_type category')
plt.ylabel('Count the applies')
plt.xticks(rotation = 90)
plt.show()
```



You can see clearly those company offer the other formatted work type category getting the highest applies on it.

In [169…

```
# let's check the remote_allowed work type wise applies

sns.set_style("darkgrid")
sns.barplot(x = 'remote_allowed',y = 'applies',hue = "applies_cat_func",data = new_data,palette = "Set3")
plt.title('Job Posting remote_allowed Category wise applies')
plt.xlabel('remote_allowed category')
plt.ylabel('Count the applies')
plt.xticks(rotation = 90)
plt.show()
```

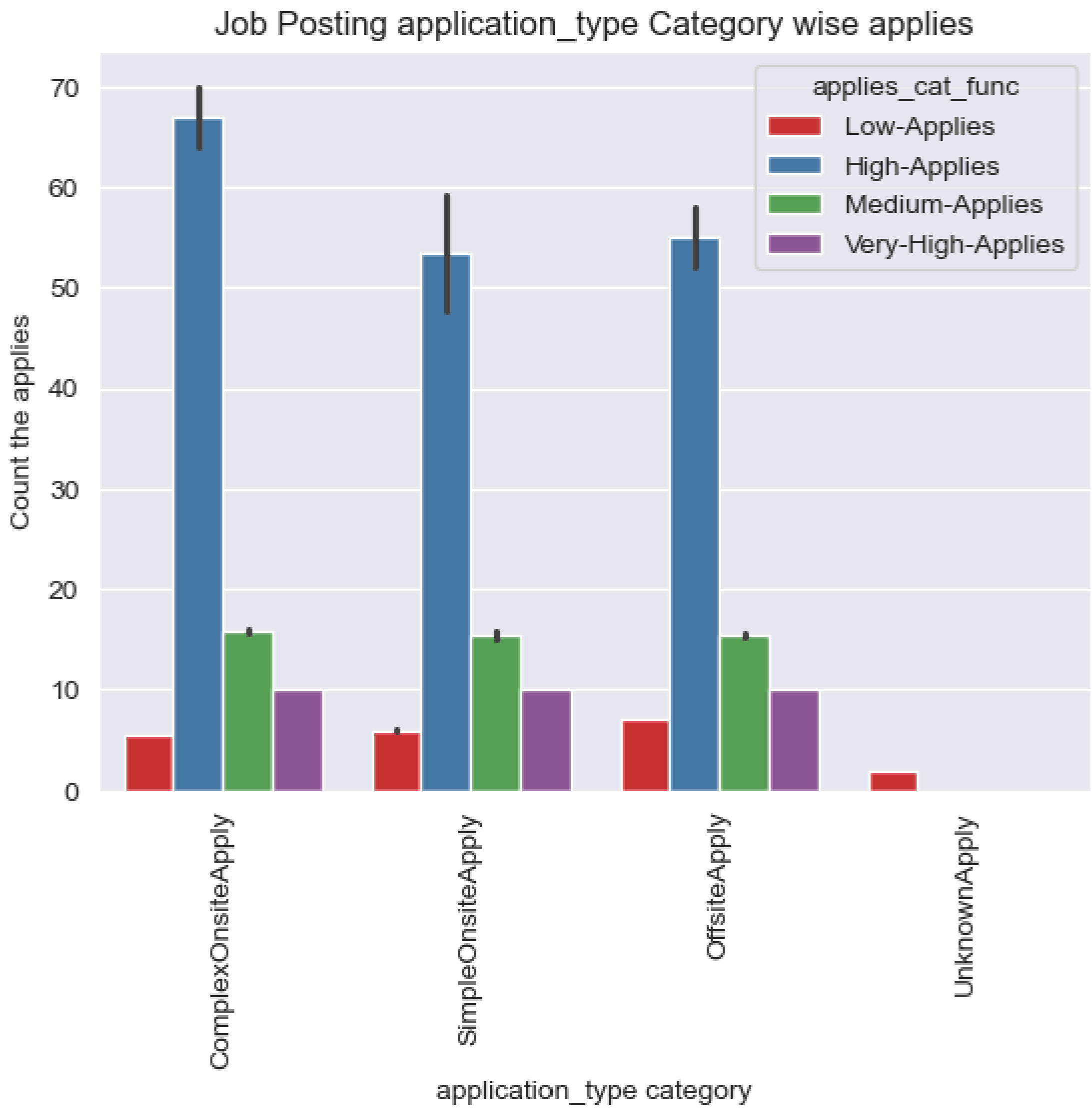


You can see the those company offeres the remotely job allowed getting the highest applies on it.

In [171…

```
# let's check the application_type work type wise applies

sns.set_style("darkgrid")
```

```
sns.barplot(x = 'application_type',y = 'applies',hue = "applies_cat_func",data = new_data,palette = "Set1")
plt.title('Job Posting application_type Category wise applies')
plt.xlabel('application_type category')
plt.ylabel('Count the applies')
plt.xticks(rotation = 90)
plt.show()
```



You can see clearly 'ComplexOnsiteApply' application type category getting the highset applies on it.

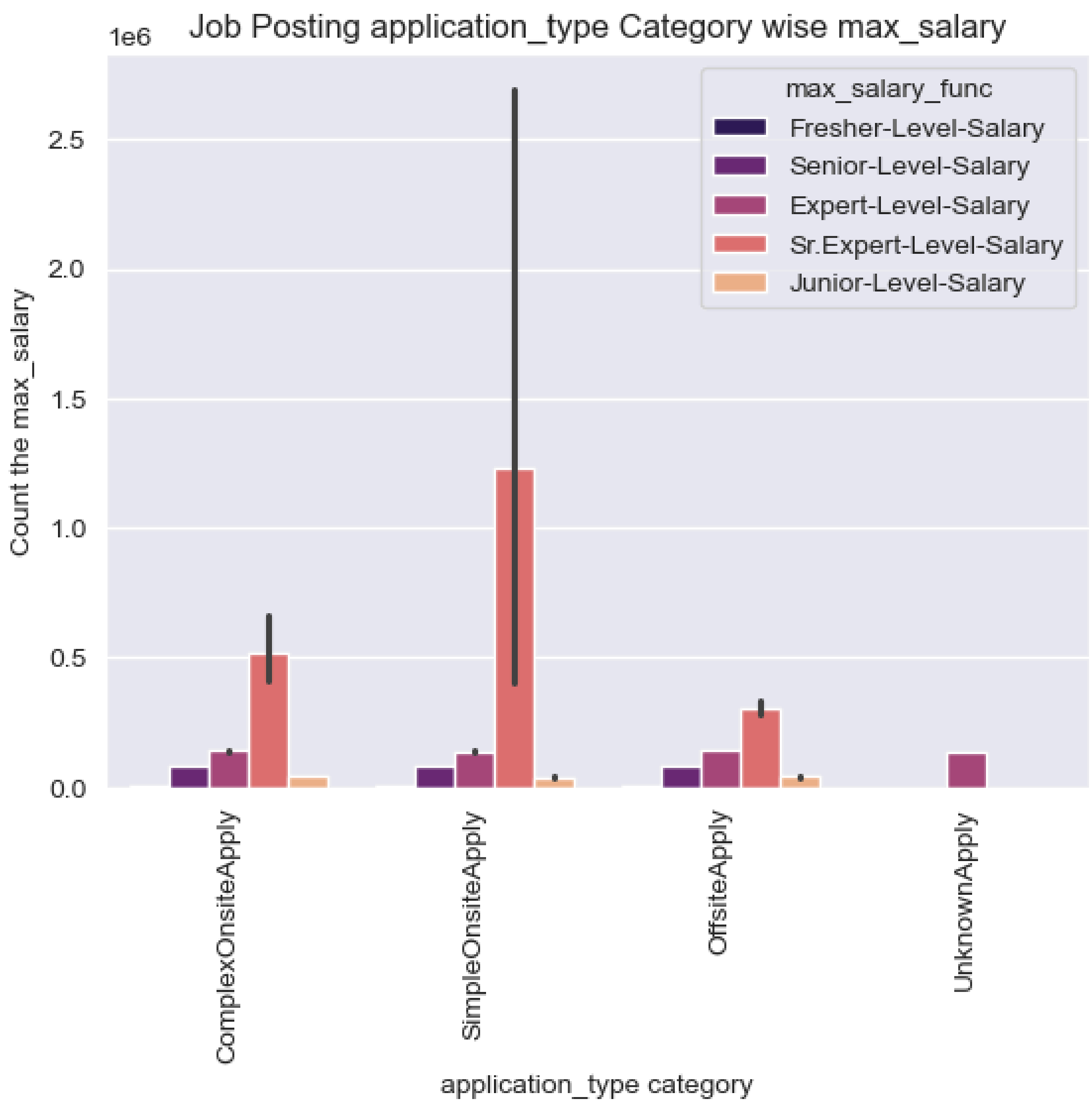
In [173…

```
# let's check the application_type work type wise min_salary

sns.set_style("darkgrid")
sns.barplot(x = 'application_type',y = 'min_salary',hue = "min_salary_func",data = new_data,palette = "rocket")
plt.title('Job Posting application_type Category wise min_salary')
plt.xlabel('application_type category')
plt.ylabel('Count the min_salary')
plt.xticks(rotation = 90)
plt.show()
```

You can see In the application type category 'SimpleOnsiteApply' category and specially 'Sr.Expert Level' getting the highest minimum salary in this category among the all.

In [175…

```
# let's check the application_type work type wise max_salary

sns.set_style("darkgrid")
sns.barplot(x = 'application_type',y = 'max_salary',hue = "max_salary_func",data = new_data,palette = "magma")
plt.title('Job Posting application_type Category wise max_salary')
plt.xlabel('application_type category')
plt.ylabel('Count the max_salary')
plt.xticks(rotation = 90)
plt.show()
```



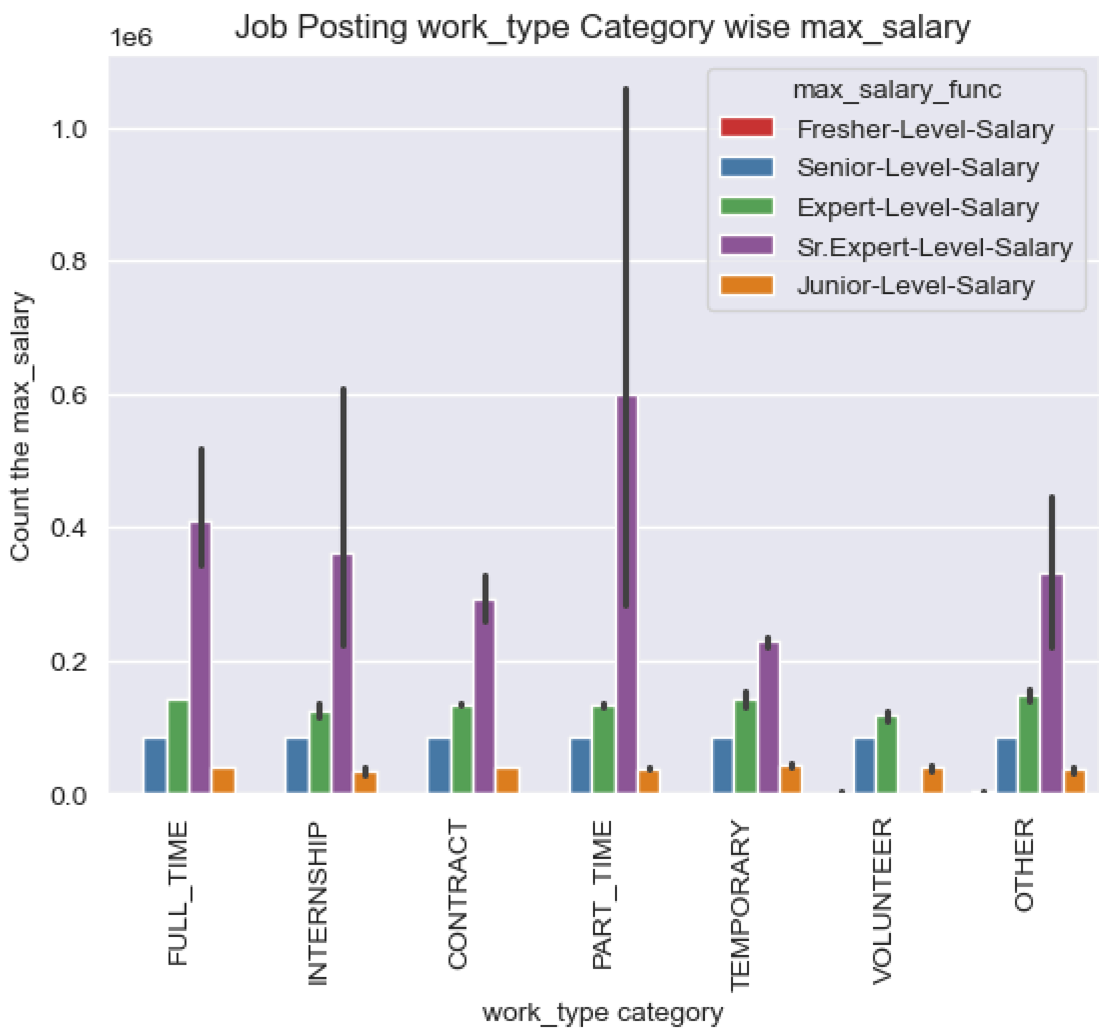
Again the winner is 'SimpleOnsiteApply' category earns more especially for the 'Sr.Expert Level' salary category.

In [178…

```
new_data.work_type
```

Out[178…
```
0         FULL_TIME
1         FULL_TIME
2         FULL_TIME
3         FULL_TIME
4         FULL_TIME
            ...
123844    FULL_TIME
123845    FULL_TIME
123846    FULL_TIME
123847    FULL_TIME
123848    FULL_TIME
Name: work_type, Length: 123849, dtype: object
```

In [179…
```python
# let's check the work_type work type wise max_salary

sns.set_style("darkgrid")
sns.barplot(x = 'work_type',y = 'max_salary',hue = "max_salary_func",data = new_data,palette = "Set1")
plt.title('Job Posting work_type Category wise max_salary')
plt.xlabel('work_type category')
plt.ylabel('Count the max_salary')
plt.xticks(rotation = 90)
plt.show()
```

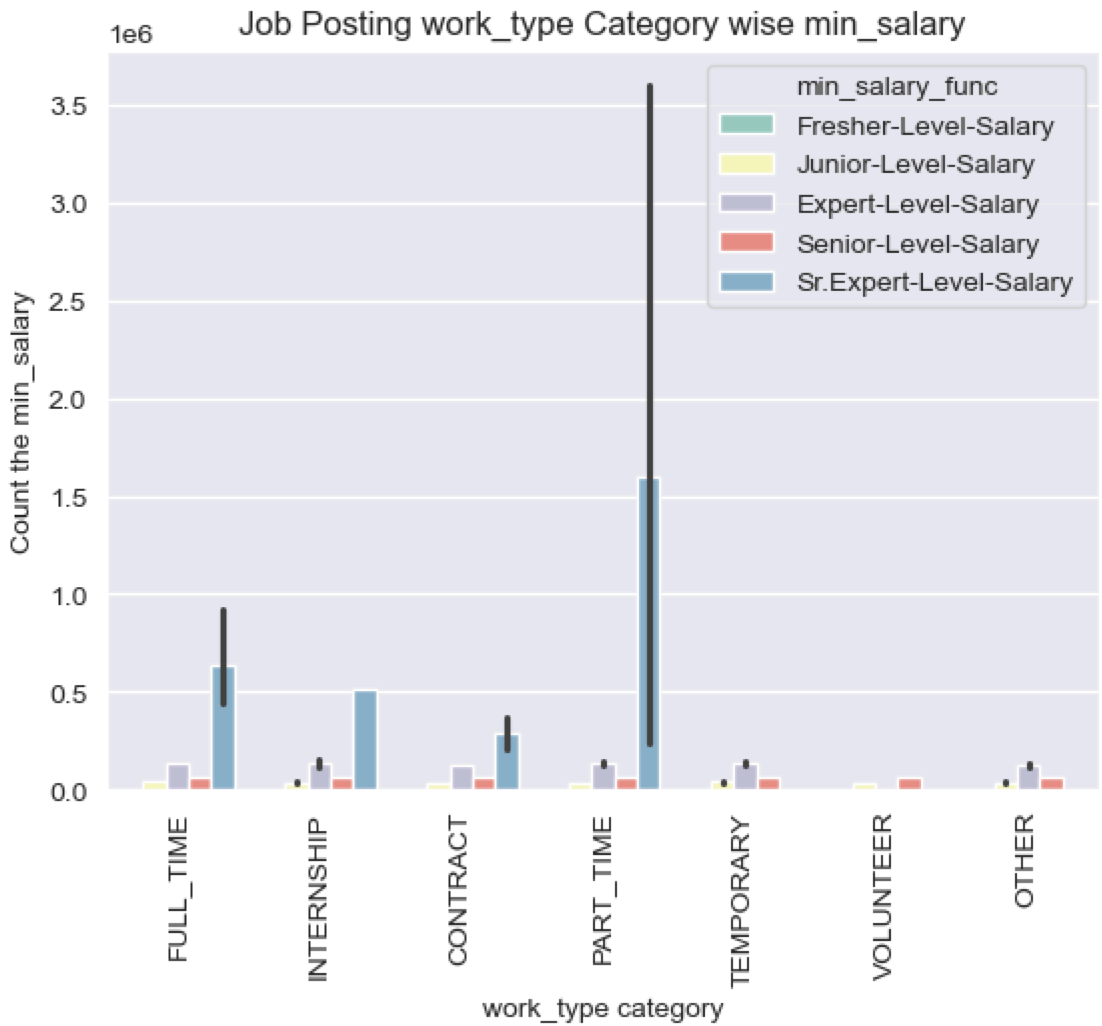


Those Work types is based on the 'Part_Time' category and in the 'Sr.Expert Level Salary' earns more maximum salary rest of all.

In [181…

```
# let's check the work_type work type wise min_salary

sns.set_style("darkgrid")
sns.barplot(x = 'work_type',y = 'min_salary',hue = "min_salary_func",data = new_data,palette = "Set3")
```

```
plt.title('Job Posting work_type Category wise min_salary')
plt.xlabel('work_type category')
plt.ylabel('Count the min_salary')
plt.xticks(rotation = 90)
plt.show()
```



Even the minimum salary is high for 'PART_TIME' jobers but only for 'Senior Expert Level' Salary.

In [183…

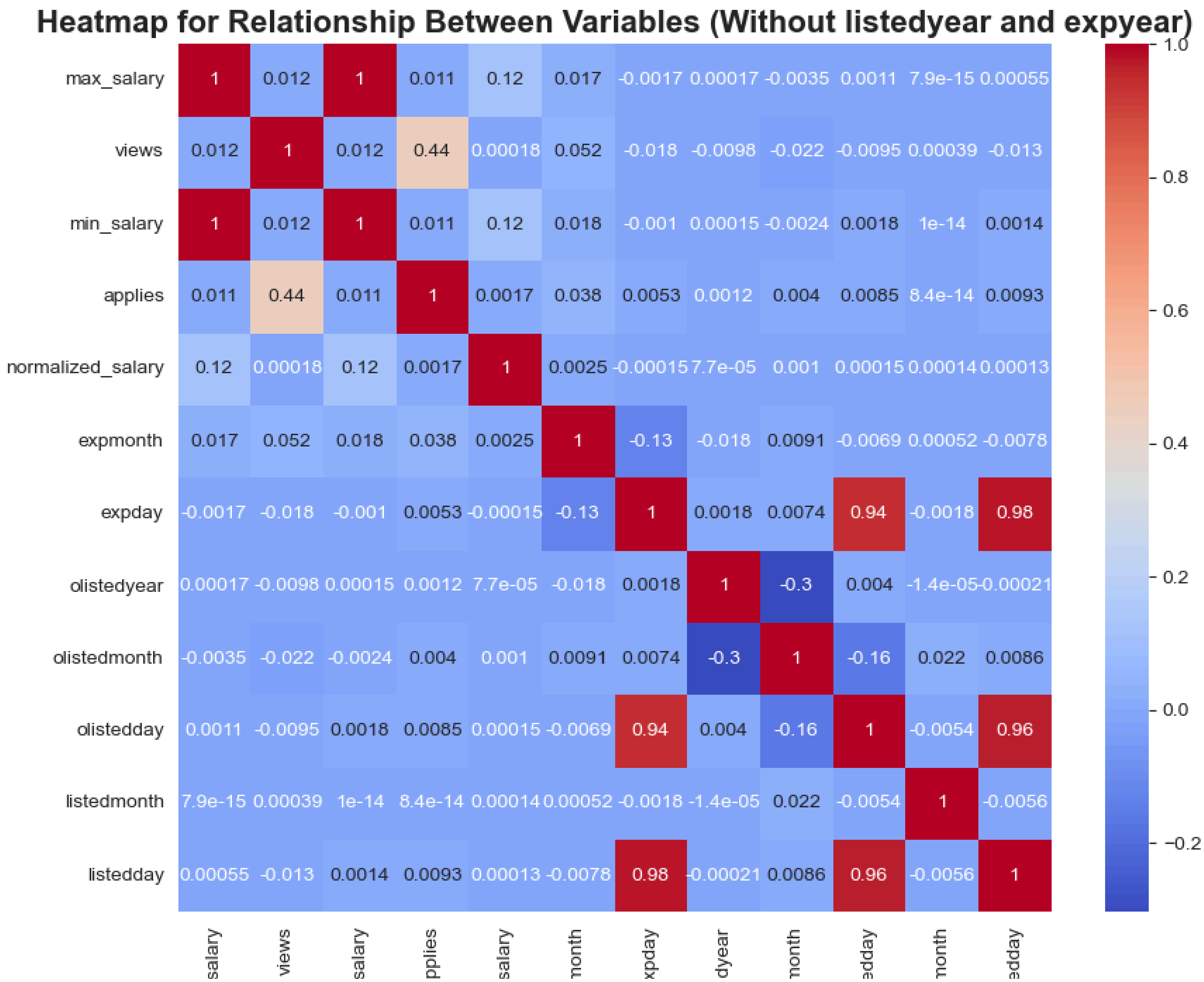
```
# let's check the relationship betweeen variables

relationship_map =  new_data.corr(numeric_only = True)
relationship_map
```

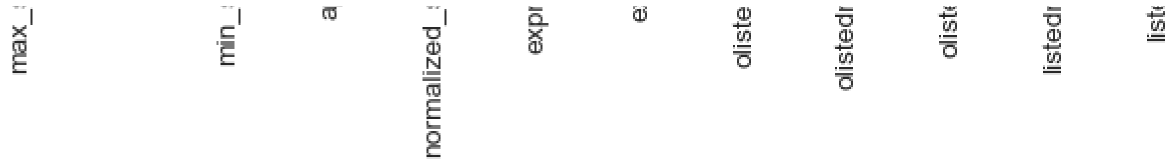
Out[183…

	max_salary	views	min_salary	applies	normalized_salary	expyear	expmonth	expday	olistedyear	ol
max_salary	1.000000e+00	0.011511	9.981402e-01	1.098989e-02	0.119010	NaN	0.017318	-0.001671	0.000175	
views	1.151097e-02	1.000000	1.219343e-02	4.424899e-01	0.000176	NaN	0.051798	-0.018109	-0.009771	
min_salary	9.981402e-01	0.012193	1.000000e+00	1.144690e-02	0.119687	NaN	0.018008	-0.000996	0.000147	
applies	1.098989e-02	0.442490	1.144690e-02	1.000000e+00	0.001738	NaN	0.037808	0.005290	0.001182	
normalized_salary	1.190099e-01	0.000176	1.196874e-01	1.737651e-03	1.000000	NaN	0.002470	-0.000146	0.000077	
expyear	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
expmonth	1.731751e-02	0.051798	1.800759e-02	3.780751e-02	0.002470	NaN	1.000000	-0.128284	-0.017668	
expday	-1.671068e-03	-0.018109	-9.957977e-04	5.290331e-03	-0.000146	NaN	-0.128284	1.000000	0.001824	
olistedyear	1.749863e-04	-0.009771	1.470985e-04	1.182369e-03	0.000077	NaN	-0.017668	0.001824	1.000000	
olistedmonth	-3.458139e-03	-0.022305	-2.352030e-03	3.975013e-03	0.001043	NaN	0.009125	0.007410	-0.304463	
olistedday	1.061332e-03	-0.009472	1.826492e-03	8.538374e-03	0.000146	NaN	-0.006907	0.942541	0.003998	
listedyear	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
listedmonth	7.932863e-15	0.000387	1.032856e-14	8.394510e-14	0.000136	NaN	0.000517	-0.001834	-0.000014	
listedday	5.506094e-04	-0.012646	1.369196e-03	9.341606e-03	0.000132	NaN	-0.007777	0.976837	-0.000214	

In [356…

```
# Remove 'listed_year' and 'expyear' from relationship_map
filtered_relationship_map = relationship_map.drop(['listedyear', 'expyear'], axis=0).drop(['listedyear', 'expyear'], axis=1)

# Now plot the heatmap
plt.figure(figsize=(10,8))
sns.heatmap(filtered_relationship_map, annot=True, cmap='coolwarm')
plt.title('Heatmap for Relationship Between Variables (Without listedyear and expyear)', fontsize=16, fontweight='bold')
plt.show()
```


Heatmap for Relationship Between Variables (Without listedyear and expyear)
max_salary
views
min_salary
applies
normalized_salary
expmonth
expday
olistedyear
olistedmonth
olistedday
listedmonth
listedday
1 0.012 1 0.011 0.12 0.017 -0.0017 0.00017 -0.0035 0.0011 7.9e-15 0.00055
0.012 1 0.012 0.44 0.00018 0.052 -0.018 -0.0098 -0.022 -0.0095 0.00039 -0.013
1 0.012 1 0.011 0.12 0.018 -0.001 0.00015 -0.0024 0.0018 1e-14 0.0014
0.011 0.44 0.011 1 0.0017 0.038 0.0053 0.0012 0.004 0.0085 8.4e-14 0.0093
0.12 0.00018 0.12 0.0017 1 0.0025 -0.00015 7.7e-05 0.001 0.00015 0.00014 0.00013
0.017 0.052 0.018 0.038 0.0025 1 -0.13 -0.018 0.0091 -0.0069 0.00052 -0.0078
-0.0017 -0.018 -0.001 0.0053 -0.00015 -0.13 1 0.0018 0.0074 0.94 -0.0018 0.98
0.00017 -0.0098 0.00015 0.0012 7.7e-05 -0.018 0.0018 1 -0.3 0.004 -1.4e-05 -0.00021
-0.0035 -0.022 -0.0024 0.004 0.001 0.0091 0.0074 -0.3 1 -0.16 0.022 0.0086
0.0011 -0.0095 0.0018 0.0085 0.00015 -0.0069 0.94 0.004 -0.16 1 -0.0054 0.96
7.9e-15 0.00039 1e-14 8.4e-14 0.00014 0.00052 -0.0018 -1.4e-05 0.022 -0.0054 1 -0.0056
0.00055 -0.013 0.0014 0.0093 0.00013 -0.0078 0.98 -0.00021 0.0086 0.96 -0.0056 1
salary
views
salary
pplies
salary
nonth
xpday
dyear
nonth
edday
nonth
edday
1.0
0.8
0.6
0.4
0.2
0.0
−0.2



Common Insights and Recommendations

1) Focus on Sales Manager roles and highlight Senior Expert Level salaries.

2) Prioritize Full-Time and Remote-Allowed jobs to attract maximum applications.

3) Mention Base Salary clearly to improve candidate interest and trust.

4) Post jobs on Mondays, especially around dates like 15th–19th, to get higher visibility.

5) Offer Yearly or Monthly salary pay periods instead of Weekly or Biweekly.

6) Prefer OffsiteApply and ComplexOnsiteApply methods for job applications.

7) Post jobs more in January, February, December, and April for better reach.

8) Senior Expert Level jobs earn the highest minimum and maximum salaries.

9) Jobs with Not-Mentioned Compensation still perform well, but Base Salary detail attracts better candidates.

10) Part-Time jobs with Senior Expert Level offer the highest salaries in that category.

11) Use popular title keywords like Manager, Engineer, Sales to match candidate searches.

12) Offering Remote Jobs significantly increases application rates.