Data Science job postings on Glassdoor - EDA

GOAL

Conduct comprehensive data cleaning and exploratory data analysis (EDA) to enhance the quality and understand the inherent patterns within the dataset, facilitating informed decision-making and future analysis.

PROJECT DURATION

Project duration varies between 2 and 3 days. In order to carry out the project as quickly as possible, it is important that the relevant data is available, complete and clean.

Importing Libraries

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Reading the file

In [2]:

```
df = pd.read_csv('Uncleaned_DS_jobs.csv')
```

In [3]:

```
df.head(3)
```

Out[3]:

	index	Job Title	Salary Estimate	Job Description	Rating	Company Name	Location	Headquarters	Size
0	0	Sr Data Scientist	$137K-$171K (Glassdoor est.)	Description\n\nThe Senior Data Scientist is re...	3.1	Healthfirst\n3.1	New York, NY	New York, NY	1001 to 5000 employees
1	1	Data Scientist	$137K-$171K (Glassdoor est.)	Secure our Nation, Ignite your Future\n\nJoin ...	4.2	ManTech\n4.2	Chantilly, VA	Herndon, VA	5001 to 10000 employees
2	2	Data Scientist	$137K-$171K (Glassdoor est.)	Overview\n\n\nAnalysis Group is one of the lar...	3.8	Analysis Group\n3.8	Boston, MA	Boston, MA	1001 to 5000 employees

In [4]:

```
df.info()
#provides a concise summary of the DataFrame.
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 672 entries, 0 to 671
Data columns (total 15 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   index              672 non-null    int64  
 1   Job Title          672 non-null    object 
 2   Salary Estimate    672 non-null    object 
 3   Job Description    672 non-null    object 
 4   Rating             672 non-null    float64
 5   Company Name       672 non-null    object 
 6   Location           672 non-null    object 
 7   Headquarters       672 non-null    object 
 8   Size               672 non-null    object 
 9   Founded            672 non-null    int64  
 10  Type of ownership  672 non-null    object 
 11  Industry           672 non-null    object 
 12  Sector             672 non-null    object 
 13  Revenue            672 non-null    object 
 14  Competitors        672 non-null    object 
dtypes: float64(1), int64(2), object(12)
memory usage: 78.9+ KB
```

Finding if there is any missing value in the dataset

In [5]:

```
any_null_columns = df.isnull().any()
print(any_null_columns)
```

```
index                    False
Job Title                False
Salary Estimate          False
Job Description          False
Rating                   False
Company Name             False
Location                 False
Headquarters             False
Size                     False
Founded                  False
Type of ownership        False
Industry                 False
Sector                   False
Revenue                  False
Competitors              False
dtype: bool
```

Result: There are not any missing value

Index Column

In [6]:

```
#dropping this column as it used as serial number. (can affect the analysis)
df.drop(columns=['index'], inplace=True)
```

In [7]:

```
df.head(3)
```

Out[7]:

	Job Title	Salary Estimate	Job Description	Rating	Company Name	Location	Headquarters	Size	Foun
0	Sr Data Scientist	$137K-$171K (Glassdoor est.)	Description\n\nThe Senior Data Scientist is re...	3.1	Healthfirst\n3.1	New York, NY	New York, NY	1001 to 5000 employees	1!
1	Data Scientist	$137K-$171K (Glassdoor est.)	Secure our Nation, Ignite your Future\n\nJoin ...	4.2	ManTech\n4.2	Chantilly, VA	Herndon, VA	5001 to 10000 employees	1!
2	Data Scientist	$137K-$171K (Glassdoor est.)	Overview\n\n\nAnalysis Group is one of the lar...	3.8	Analysis Group\n3.8	Boston, MA	Boston, MA	1001 to 5000 employees	1!

Job Title Column

In [8]:

```
df['Job Title'].unique()
```

```
G - SAN ANTONIO OR',
       'AI Ops Data Scientist', 'Intelligence Data Analyst, Senior',
       'Analytics Manager - Data Mart',
       'Data Modeler (Analytical Systems)',
       'Senior Machine Learning Scientist - Bay Area, CA',
       'Report Writer-Data Analyst', 'Staff Data Scientist - Pricing',
       'Equity Data Insights Analyst - Quantitative Analyst',
       'Operations Data Analyst', 'Software Data Engineer',
       'Real World Evidence (RWE) Scientist', 'Computer Scientist 1',
       'Environmental Data Science', 'Staff BI and Data Engineer',
       'Data Scientist - Statistics, Mid-Career',
       'Director of Data Science',
       'Data Engineer, Digital & Comp Pathology',
       'Manager / Lead, Data Science & Analytics',
       'Diversity and Inclusion Data Analyst',
       'Data Scientist Machine Learning', 'Chief Scientist',
       'Development Scientist, Voltaren',
       'Principal Data & Analytics Platform Engineer',
       'Machine Learning Engineer/Scientist',
       'Data Analyst - Unilever Prestige', 'VP, Data Science',
```

Salary Estimate Column

In [9]:

```
df['Salary Estimate'].unique()
```

Out[9]:

```
array(['$137K-$171K (Glassdoor est.)', '$75K-$131K (Glassdoor est.)',
       '$79K-$131K (Glassdoor est.)', '$99K-$132K (Glassdoor est.)',
       '$90K-$109K (Glassdoor est.)', '$101K-$165K (Glassdoor est.)',
       '$56K-$97K (Glassdoor est.)', '$79K-$106K (Glassdoor est.)',
       '$71K-$123K (Glassdoor est.)', '$90K-$124K (Glassdoor est.)',
       '$91K-$150K (Glassdoor est.)', '$141K-$225K (Glassdoor est.)',
       '$145K-$225K(Employer est.)', '$79K-$147K (Glassdoor est.)',
       '$122K-$146K (Glassdoor est.)', '$112K-$116K (Glassdoor est.)',
       '$110K-$163K (Glassdoor est.)', '$124K-$198K (Glassdoor est.)',
       '$79K-$133K (Glassdoor est.)', '$69K-$116K (Glassdoor est.)',
       '$31K-$56K (Glassdoor est.)', '$95K-$119K (Glassdoor est.)',
       '$212K-$331K (Glassdoor est.)', '$66K-$112K (Glassdoor est.)',
       '$128K-$201K (Glassdoor est.)', '$138K-$158K (Glassdoor est.)',
       '$80K-$132K (Glassdoor est.)', '$87K-$141K (Glassdoor est.)',
       '$92K-$155K (Glassdoor est.)', '$105K-$167K (Glassdoor est.)'],
      dtype=object)
```

In [10]:

```python
# function to separate the upper limit and lower limit of the salary
def salary_to_numeric(salary_range):
    salary_range = salary_range.replace('K', '')  # Remove 'K'
    salary_range = salary_range.split('-')  # Split into low and high values
    low_salary = int(salary_range[0][1:]) * 1000  # Convert to numeric value in thousands

    high_salary = salary_range[1].split()[0]
    if high_salary.isdigit():  # Check if the high salary is a valid integer
        high_salary = int(high_salary) * 1000  # Convert to numeric value in thousands
    else:
        high_salary = low_salary  # If the high salary is not a valid integer, use the low salary

    return low_salary, high_salary

# Apply the function to the 'Salary Estimate' column and create two new columns for low and high s
df[['Low_Salary_in_dollar', 'High_Salary_in_dollar']] = df['Salary Estimate'].apply(salary_to_nume
In [11]:
```

```python
df[['Low_Salary_in_dollar', 'High_Salary_in_dollar']]
```

```
Out[11]:
```

	Low_Salary_in_dollar	High_Salary_in_dollar
0	137000	137000
1	137000	137000
2	137000	137000
3	137000	137000
4	137000	137000
...	...	...
667	105000	105000
668	105000	105000
669	105000	105000
670	105000	105000
671	105000	105000

672 rows × 2 columns

Note:- we will be using the upper limit(High_Salary_in_dollar) in this analysis.

Job Description Column

In [12]:

```
#dropping this column
df.drop(columns=['Job Description'], inplace=True)
```

Rating Column

In [13]:

```
df['Rating'].unique()
```

Out[13]:

```
array([ 3.1,  4.2,  3.8,  3.5,  2.9,  3.9,  4.4,  3.6,  4.5,  4.7,  3.7,
        3.4,  4.1,  3.2,  4.3,  2.8,  5. ,  4.8,  3.3,  2.7,  2.2,  2.6,
        4. ,  2.5,  4.9,  2.4, -1. ,  2.3,  4.6,  3. ,  2.1,  2. ])
```

In [15]:

```
# the Scale for rating is 0 to 5

# Replace '-1' with NaN
df['Rating'] = df['Rating'].replace(-1.0, np.nan).round(1)

# Printing the unique values in the 'Rating' column
unique_ratings = df['Rating'].unique()
df['Rating'].unique()
```

Out[15]:

```
array([3.1, 4.2, 3.8, 3.5, 2.9, 3.9, 4.4, 3.6, 4.5, 4.7, 3.7, 3.4, 4.1,
       3.2, 4.3, 2.8, 5. , 4.8, 3.3, 2.7, 2.2, 2.6, 4. , 2.5, 4.9, 2.4,
       nan, 2.3, 4.6, 3. , 2.1, 2. ])
```

Company Name Column

In [16]:

```
df['Company Name'].unique()
```

Out[16]:

```
array(['Healthfirst\n3.1', 'ManTech\n4.2', 'Analysis Group\n3.8',
       'INFICON\n3.5', 'Affinity Solutions\n2.9', 'HG Insights\n4.2',
       'Novartis\n3.9', 'iRobot\n3.5', 'Intuit - Data\n4.4',
       'XSELL Technologies\n3.6', 'Novetta\n4.5', '1904labs\n4.7',
       'PNNL\n3.7', 'Old World Industries\n3.1',
       'Mathematica Policy Research\n3.4',
       'Guzman & Griffin Technologies (GGTI)\n4.4',
       'Upside Business Travel\n4.1', 'Buckman\n3.5',
       'Insight Enterprises, Inc.\n4.2', 'Tower Health\n3.5',
       'Triplebyte\n3.2', 'PulsePoint\n4.3', 'Exponent\n3.5',
       'Guardian Life\n3.5',
       'Spectrum Communications and Consulting\n3.4',
       'Oversight Systems\n4.7', 'LSQ\n4.2',
       'MIT Lincoln Laboratory\n3.8', 'Kingfisher Systems\n4.5',
       'Formation\n2.8', 'Cohere Health\n5.0', 'Acuity Insurance\n4.8',
       'Chef\n3.6', 'Puget Sound Energy\n3.3', 'Sandhills Global\n2.7',
       'A Place for Mom\n2.7', 'Great-Circle Technologies\n2.2',
       'Edmunds.com\n3.4', 'Cambridge Associates, LLC\n3.1',
```

In [17]:

```python
# removing '\n' and number from company name
def remove_newline_and_number(value):
    return value.split('\n')[0]

# Apply the function to each element in the data array
df['Company Name'] = list(map(remove_newline_and_number, df['Company Name']))
df['Company Name']
```

Out[17]:

```
0                Healthfirst
1                    ManTech
2             Analysis Group
3                    INFICON
4         Affinity Solutions
                ...
667                 TRANZACT
668                     JKGT
669               AccessHope
670      ChaTeck Incorporated
671             1-800-Flowers
Name: Company Name, Length: 672, dtype: object
```

Location Column

In [18]:

```python
df['Location'].unique()
```

```
Out[18]:
```

```
array(['New York, NY', 'Chantilly, VA', 'Boston, MA', 'Newton, MA',
       'Santa Barbara, CA', 'Cambridge, MA', 'Bedford, MA',
       'San Diego, CA', 'Chicago, IL', 'Herndon, VA', 'Saint Louis, MO',
       'Richland, WA', 'Northbrook, IL', 'Washington, DC', 'Remote',
       'Memphis, TN', 'Plano, TX', 'West Grove, PA', 'Phoenix, AZ',
       'Appleton, WI', 'Atlanta, GA', 'Orlando, FL', 'Lexington, MA',
       'McLean, VA', 'San Francisco, CA', 'Sheboygan, WI',
       'United States', 'Bothell, WA', 'Lincoln, NE', 'Overland Park, KS',
       'Santa Monica, CA', 'Portsmouth, NH', 'Ewing, NJ',
       'South San Francisco, CA', 'Palo Alto, CA', 'Bellevue, WA',
       'New Orleans, LA', 'Akron, OH', 'Fort Wayne, IN', 'Woburn, MA',
       'Carson, CA', 'Coral Gables, FL', 'Santa Clara, CA',
       'Brisbane, CA', 'Winter Park, FL', 'Redwood City, CA',
       'Peoria, IL', 'Ipswich, MA', 'Carmel, IN', 'Emeryville, CA',
       'Gaithersburg, MD', 'Longmont, CO', 'Austin, TX', 'Yakima, WA',
       'Santa Cruz, CA', 'Springfield, VA', 'Alexandria, VA', 'Utah',
       'Reston, VA', 'Denver, CO', 'New Jersey', 'Aurora, CO',
       'Hill AFB, UT', 'Chandler, AZ', 'Indianapolis, IN',
       'Nashville, TN', 'Timonium, MD', 'Burlingame, CA',
       'Annapolis Junction, MD', 'Bethesda, MD', 'Dayton, OH',
       'Schaumburg, IL', 'Cupertino, CA', 'Lehi, UT', 'Culver City, CA',
       'Lake Oswego, OR', 'San Mateo, CA', 'Holyoke, MA',
       'Woodbridge, NJ', 'Dearborn, MI', 'Maryland Heights, MO',
       'Rockville, MD', 'Carpinteria, CA', 'Columbia, SC',
       'Hauppauge, NY', 'Fort Meade, MD', 'Columbia, MO', 'Vicksburg, MS',
       'Birmingham, AL', 'Blue Bell, PA', 'Cincinnati, OH',
       'Harrisburg, PA', 'Oak Ridge, TN', 'San Carlos, CA', 'Waltham, MA',
       'Fort Worth, TX', 'Smithfield, RI', 'Cedar Rapids, IA',
       'Fort Belvoir, VA', 'Linthicum Heights, MD', 'Maple Plain, MN',
       'Tulsa, OK', 'Baltimore, MD', 'Oklahoma City, OK',
       'Scotts Valley, CA', 'Spartanburg, SC', 'Hartford, CT',
       'Beavercreek, OH', 'Norfolk, VA', 'Charlotte, NC', 'Champaign, IL',
       'Texas', 'Hoboken, NJ', 'Lebanon, IN', 'Oakland, CA',
       'Melbourne, FL', 'Cleveland, OH', 'Norwell, MA', 'San Jose, CA',
       'Piscataway, NJ', 'Danvers, MA', 'Vienna, VA', 'Livermore, CA',
       'Pittsburgh, PA', 'Irvine, CA', 'Oshkosh, WI', 'Menlo Park, CA',
       'Dallas, TX', 'Arlington, VA', 'Monroe, WI', 'Sacramento, CA',
       'Hampton, VA', 'Richmond, VA', 'Monterey, CA', 'Woodlawn, MD',
       'Ann Arbor, MI', 'Concord, CA', 'Durham, NC', 'Kent, OH',
       'Laurel, MD', 'Columbia, MD', 'Falls Church, VA',
       'Thousand Oaks, CA', 'Edison, NJ', 'Adelphi, MD', 'Seattle, WA',
       'Sunnyvale, CA', 'Fremont, CA', 'Hamilton, NJ', 'Huntsville, AL',
       'Merrifield, VA', 'Madison, WI', 'Philadelphia, PA',
       'Winston-Salem, NC', 'Raleigh, NC', 'Burbank, CA', 'San Ramon, CA',
       'Oxnard, CA', 'Kansas City, MO', 'Jersey City, NJ',
       'Manchester, NH', 'Winters, TX', 'Brooklyn, NY', 'Germantown, MD',
       'Omaha, NE', 'Open Fork, VA', 'Ashburn, VA', 'Lombard, IL',
       'Alpharetta, GA', 'Boulder, CO', 'Mountain View, CA',
       'Trumbull, CT', 'Sterling, VA', 'Foster City, CA', 'Frederick, MD',
       'Colorado Springs, CO', 'Southfield, MI', 'San Clemente, CA',
       'The Woodlands, TX', 'Pleasanton, CA', 'Wilmington, DE',
       'Fort Sam Houston, TX', 'Lexington Park, MD',
       'Patuxent, Anne Arundel, MD', 'Fairfax, VA', 'San Antonio, TX',
       'Silver Spring, MD', 'Portland, OR', 'Simi Valley, CA',
       'New Bedford, MA', 'Rancho Cucamonga, CA', 'Collegeville, PA',
       'Minneapolis, MN', 'Gahanna, OH', 'California', 'Wellesley, MA',
       'Washington, VA', 'Orange, CA', 'Bridgeport, WV', 'Oakville, CA',
       'Naperville, IL', 'Houston, TX', 'Redmond, WA', 'West Chester, PA',
       'Quantico, VA', 'Fort Lee, NJ', 'Irwindale, CA'], dtype=object)
```

Headquarters Column

In [20]:

```
#dropping this column
df.drop(columns=['Headquarters'], inplace=True)
```

Size Column

In [21]:

```
df['Size'].unique()
```

Out[21]:

```
array(['1001 to 5000 employees', '5001 to 10000 employees',
       '501 to 1000 employees', '51 to 200 employees', '10000+ employees',
       '201 to 500 employees', '1 to 50 employees', '-1', 'Unknown'],
      dtype=object)
```

In [22]:

```
#Replacing '-1' and 'Unknown' with NaN
df['Size'].replace(['-1', 'Unknown'], np.nan, inplace=True)
df['Size'].unique()
```

Out[22]:

```
array(['1001 to 5000 employees', '5001 to 10000 employees',
       '501 to 1000 employees', '51 to 200 employees', '10000+ employees',
       '201 to 500 employees', '1 to 50 employees', nan], dtype=object)
```

Founded Column

In [23]:

```
df['Founded'].unique()
```

Out[23]:

```
array([1993, 1968, 1981, 2000, 1998, 2010, 1996, 1990, 1983, 2014, 2012,
       2016, 1965, 1973, 1986, 1997, 2015, 1945, 1988, 2017, 2011, 1967,
       1860, 1992, 2003, 1951, 2005, 2019, 1925, 2008, 1999, 1978, 1966,
       1912, 1958, 2013, 1849, 1781, 1926, 2006, 1994, 1863, 1995,   -1,
       1982, 1974, 2001, 1985, 1913, 1971, 1911, 2009, 1959, 2007, 1939,
       2002, 1961, 1963, 1969, 1946, 1957, 1953, 1948, 1850, 1851, 2004,
       1976, 1918, 1954, 1947, 1955, 2018, 1937, 1917, 1935, 1929, 1820,
       1952, 1932, 1894, 1960, 1788, 1830, 1984, 1933, 1880, 1887, 1970,
       1942, 1980, 1989, 1908, 1853, 1875, 1914, 1898, 1956, 1977, 1987,
       1896, 1972, 1949, 1962], dtype=int64)
```

Type of ownership Column

In [24]:

```
df['Type of ownership'].unique()
```

Out[24]:

```
array(['Nonprofit Organization', 'Company - Public',
       'Private Practice / Firm', 'Company - Private', 'Government',
       'Subsidiary or Business Segment', 'Other Organization', '-1',
       'Unknown', 'Hospital', 'Self-employed', 'College / University',
       'Contract'], dtype=object)
```

In [25]:

```
#Replacing '-1' with NaN
df['Type of ownership'].replace(['-1', 'Unknown'], np.nan, inplace=True)
df['Type of ownership'].unique()
```

Out[25]:

```
array(['Nonprofit Organization', 'Company - Public',
       'Private Practice / Firm', 'Company - Private', 'Government',
       'Subsidiary or Business Segment', 'Other Organization', nan,
       'Hospital', 'Self-employed', 'College / University', 'Contract'],
      dtype=object)
```

Industry Column

In [26]:

```
df['Industry'].unique()
```

Out[26]:

```
array(['Insurance Carriers', 'Research & Development', 'Consulting',
       'Electrical & Electronic Manufacturing', 'Advertising & Marketing',
       'Computer Hardware & Software', 'Biotech & Pharmaceuticals',
       'Consumer Electronics & Appliances Stores',
       'Enterprise Software & Network Solutions', 'IT Services', 'Energy',
       'Chemical Manufacturing', 'Federal Agencies', 'Internet',
       'Health Care Services & Hospitals',
       'Investment Banking & Asset Management', 'Aerospace & Defense',
       'Utilities', '-1', 'Express Delivery Services',
       'Staffing & Outsourcing', 'Insurance Agencies & Brokerages',
       'Consumer Products Manufacturing', 'Industrial Manufacturing',
       'Food & Beverage Manufacturing', 'Banks & Credit Unions',
       'Video Games', 'Shipping', 'Telecommunications Services',
       'Lending', 'Cable, Internet & Telephone Providers', 'Real Estate',
       'Venture Capital & Private Equity', 'Miscellaneous Manufacturing',
       'Oil & Gas Services', 'Transportation Equipment Manufacturing',
       'Telecommunications Manufacturing', 'Transportation Management',
       'News Outlet', 'Architectural & Engineering Services',
       'Food & Beverage Stores', 'Other Retail Stores',
       'Hotels, Motels, & Resorts', 'State & Regional Agencies',
       'Financial Transaction Processing', 'Timber Operations',
       'Colleges & Universities', 'Travel Agencies', 'Accounting',
       'Logistics & Supply Chain', 'Farm Support Services',
       'Social Assistance', 'Construction',
       'Department, Clothing, & Shoe Stores', 'Publishing',
       'Health, Beauty, & Fitness', 'Wholesale', 'Rail'], dtype=object)
```

In [27]:

```
#Replacing '-1' with NaN
df['Industry'].replace(['-1'], np.nan, inplace=True)
df['Industry'].unique()
```

```
Out[27]:

array(['Insurance Carriers', 'Research & Development', 'Consulting',
       'Electrical & Electronic Manufacturing', 'Advertising & Marketing',
       'Computer Hardware & Software', 'Biotech & Pharmaceuticals',
       'Consumer Electronics & Appliances Stores',
       'Enterprise Software & Network Solutions', 'IT Services', 'Energy',
       'Chemical Manufacturing', 'Federal Agencies', 'Internet',
       'Health Care Services & Hospitals',
       'Investment Banking & Asset Management', 'Aerospace & Defense',
       'Utilities', nan, 'Express Delivery Services',
       'Staffing & Outsourcing', 'Insurance Agencies & Brokerages',
       'Consumer Products Manufacturing', 'Industrial Manufacturing',
       'Food & Beverage Manufacturing', 'Banks & Credit Unions',
       'Video Games', 'Shipping', 'Telecommunications Services',
       'Lending', 'Cable, Internet & Telephone Providers', 'Real Estate',
       'Venture Capital & Private Equity', 'Miscellaneous Manufacturing',
       'Oil & Gas Services', 'Transportation Equipment Manufacturing',
       'Telecommunications Manufacturing', 'Transportation Management',
       'News Outlet', 'Architectural & Engineering Services',
       'Food & Beverage Stores', 'Other Retail Stores',
       'Hotels, Motels, & Resorts', 'State & Regional Agencies',
       'Financial Transaction Processing', 'Timber Operations',
       'Colleges & Universities', 'Travel Agencies', 'Accounting',
       'Logistics & Supply Chain', 'Farm Support Services',
       'Social Assistance', 'Construction',
       'Department, Clothing, & Shoe Stores', 'Publishing',
       'Health, Beauty, & Fitness', 'Wholesale', 'Rail'], dtype=object)
```

Sector Column

In [28]:

```
df['Sector'].unique()
```

Out[28]:

```
array(['Insurance', 'Business Services', 'Manufacturing',
       'Information Technology', 'Biotech & Pharmaceuticals', 'Retail',
       'Oil, Gas, Energy & Utilities', 'Government', 'Health Care',
       'Finance', 'Aerospace & Defense', '-1',
       'Transportation & Logistics', 'Media', 'Telecommunications',
       'Real Estate', 'Travel & Tourism', 'Agriculture & Forestry',
       'Education', 'Accounting & Legal', 'Non-Profit',
       'Construction, Repair & Maintenance', 'Consumer Services'],
      dtype=object)
```

In [29]:

```
#Replacing '-1' with NaN
df['Sector'].replace(['-1'], np.nan, inplace=True)
df['Sector'].unique()
```

Out[29]:

```
array(['Insurance', 'Business Services', 'Manufacturing',
       'Information Technology', 'Biotech & Pharmaceuticals', 'Retail',
       'Oil, Gas, Energy & Utilities', 'Government', 'Health Care',
       'Finance', 'Aerospace & Defense', nan,
       'Transportation & Logistics', 'Media', 'Telecommunications',
       'Real Estate', 'Travel & Tourism', 'Agriculture & Forestry',
       'Education', 'Accounting & Legal', 'Non-Profit',
       'Construction, Repair & Maintenance', 'Consumer Services'],
      dtype=object)
```

Revenue Column

In [30]:

```
df['Revenue'].unique()
```

Out[30]:

```
array(['Unknown / Non-Applicable', '$1 to $2 billion (USD)',
       '$100 to $500 million (USD)', '$10+ billion (USD)',
       '$2 to $5 billion (USD)', '$500 million to $1 billion (USD)',
       '$5 to $10 billion (USD)', '$10 to $25 million (USD)',
       '$25 to $50 million (USD)', '$50 to $100 million (USD)',
       '$1 to $5 million (USD)', '$5 to $10 million (USD)',
       'Less than $1 million (USD)', '-1'], dtype=object)
```

In [31]:

```
#Replacing '-1' with NaN
df['Revenue'].replace(['-1','Unknown / Non-Applicable'], np.nan, inplace=True)
df['Revenue'].unique()
```

Out[31]:

```
array([nan, '$1 to $2 billion (USD)', '$100 to $500 million (USD)',
       '$10+ billion (USD)', '$2 to $5 billion (USD)',
       '$500 million to $1 billion (USD)', '$5 to $10 billion (USD)',
       '$10 to $25 million (USD)', '$25 to $50 million (USD)',
       '$50 to $100 million (USD)', '$1 to $5 million (USD)',
       '$5 to $10 million (USD)', 'Less than $1 million (USD)'],
      dtype=object)
```

Exploratory Data Analysis(EDA)

In [32]:

```
# Get basic statistics of numerical variables (mean, median, min, max)
df.describe()
```

Out[32]:

	Rating	Founded	Low_Salary_in_dollar	High_Salary_in_dollar
count	622.000000	672.000000	672.000000	672.000000
mean	3.881833	1635.529762	99196.428571	99196.428571
std	0.610805	756.746640	33009.958111	33009.958111
min	2.000000	-1.000000	31000.000000	31000.000000
25%	3.500000	1917.750000	79000.000000	79000.000000
50%	3.800000	1995.000000	91000.000000	91000.000000
75%	4.400000	2009.000000	122000.000000	122000.000000
max	5.000000	2019.000000	212000.000000	212000.000000

Identifying the categorical data types

In [34]:

```python
#Get the data types of each column in the dataset
data_types = df.dtypes

#Identify columns with object or categorical data type
categorical_columns = data_types[data_types == 'object'].index.tolist()

print("Categorical Columns:")
print(categorical_columns)
```

```
Categorical Columns:
['Job Title', 'Salary Estimate', 'Company Name', 'Location', 'Size', 'Type of ownershi
p', 'Industry', 'Sector', 'Revenue', 'Competitors']
```

Finding the value count of categorical columns

In [44]:

```
#Select categorical columns for analysis
categorical_columns = ['Size', 'Type of ownership', 'Industry', 'Sector', 'Revenue', 'Location']

#Get unique values and their counts in each categorical column
for col in categorical_columns:

    value_counts = df[col].value_counts()
    print(f"\nValue counts in '{col}':")
    print(value_counts)

```

```
Value counts in 'Size':
51 to 200 employees        135
1001 to 5000 employees     104
1 to 50 employees           86
201 to 500 employees        85
10000+ employees            80
501 to 1000 employees       77
5001 to 10000 employees     61
Name: Size, dtype: int64

Value counts in 'Type of ownership':
Company - Private                 397
Company - Public                  153
Nonprofit Organization             36
Subsidiary or Business Segment     28
Government                         10
Other Organization                  5
Private Practice / Firm             4
```

OBSERVATION

1. Size of Companies:

- The majority of companies in the dataset have between 51 to 200 employees, with 135 companies falling into this category.
- Companies with 1001 to 5000 employees and 1 to 50 employees are also relatively common, with 104 and 86 companies, respectively.
- The least common size category is companies with 5001 to 10,000 employees, with only 61 companies falling into this group.

2. Type of Ownership:

- The most common type of ownership is "Company - Private," with 397 companies falling into this category.
- "Company - Public" is the second most common type of ownership, with 153 companies.
- Other types of ownership, such as "Nonprofit Organization" and "Subsidiary or Business Segment," are less common.

3. Industry:

- The dataset covers a wide range of industries, with the top three being Biotech & Pharmaceuticals (66 companies), IT Services (61 companies), and Computer Hardware & Software (57 companies).
- Many other industries are represented, including Aerospace & Defense, Enterprise Software & Network Solutions, and Consulting.

4. Sector:

- The most common sector is Information Technology, with 188 companies falling into this category.
- Other significant sectors include Business Services (120 companies), Biotech & Pharmaceuticals (66 companies), and Aerospace & Defense (46 companies).

5. Revenue:

- The dataset includes companies with a diverse range of revenue levels.
- The most common revenue range is 100 to 500 million (USD) with 94 companies.
- Some companies have very high revenue levels, such as 10+ billion (USD), while others have lower revenues, such as Less than 1 million (USD).

6. Location:

- The dataset includes companies from various locations, with San Francisco, CA having the highest representation (69 companies).

- Other notable locations include New York, NY (50 companies), Washington, DC (26 companies), and Boston, MA (24 companies).

Identifying the Numerical data types

In [41]:

```
#Get the data types of each column in the dataset
data_types = df.dtypes

#Identify columns with int64 or float64 data type
numerical_columns = df.select_dtypes(include=['float64', 'int64'])

print("Numerical Columns:")
print(numerical_columns)

```

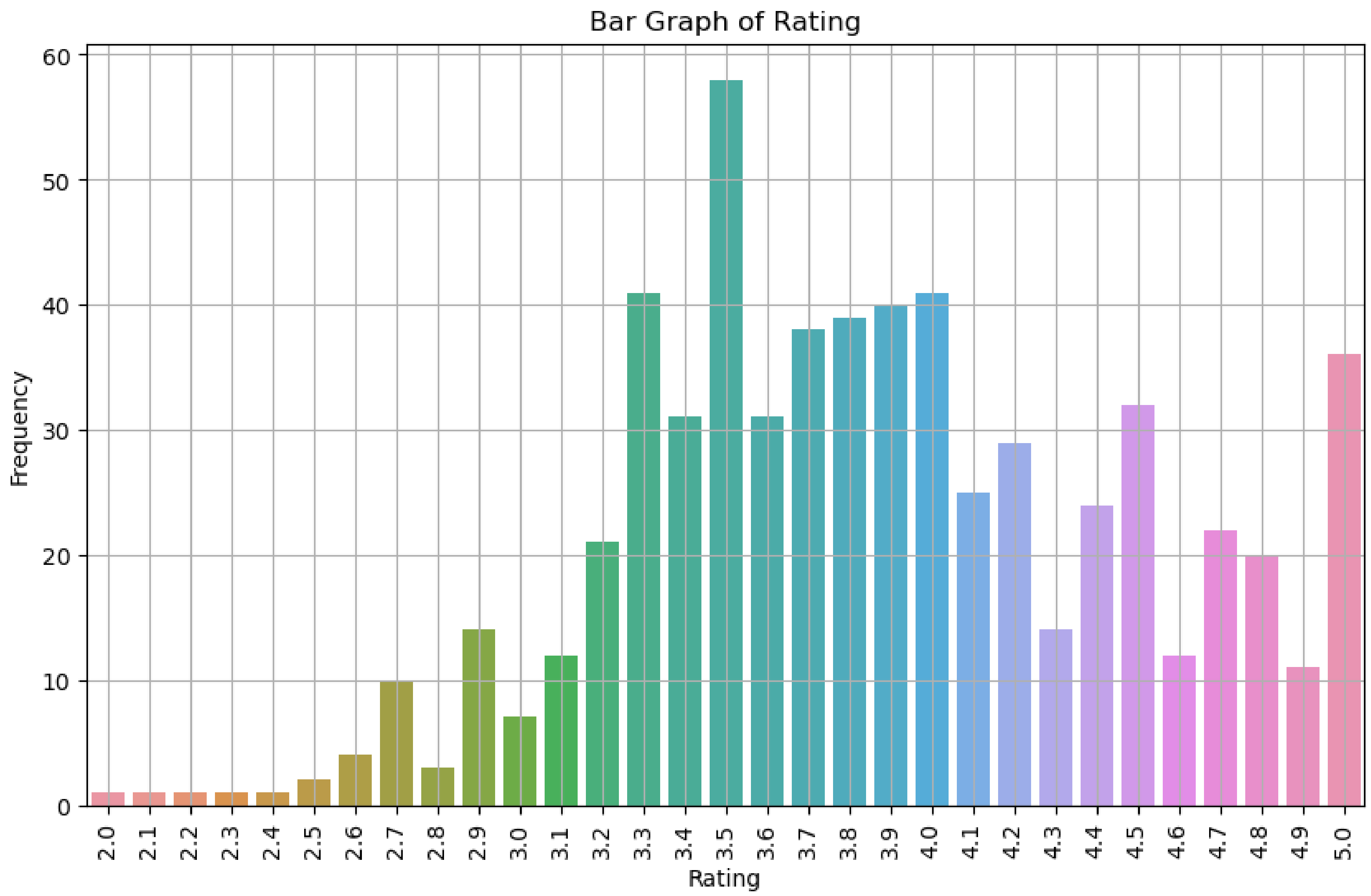
```
Numerical Columns:
     Rating  Founded  Low_Salary_in_dollar  High_Salary_in_dollar
0       3.1     1993                137000                 137000
1       4.2     1968                137000                 137000
2       3.8     1981                137000                 137000
3       3.5     2000                137000                 137000
4       2.9     1998                137000                 137000
..      ...      ...                   ...                    ...
667     3.6     1989                105000                 105000
668     NaN       -1                105000                 105000
669     NaN       -1                105000                 105000
670     5.0       -1                105000                 105000
671     2.7     1976                105000                 105000

[672 rows x 4 columns]
```

Finding skewness and kurtosis for Rating

In [53]:

```python
#Calculate skewness and kurtosis for each numerical column
skewness = df['Rating'].skew()
kurtosis = df['Rating'].kurt()

#Count the frequency of each unique value in the column
value_counts = df['Rating'].value_counts()

#Plot the bar graph
plt.figure(figsize=(10, 6))
sns.barplot(x=value_counts.index, y=value_counts.values)
plt.xlabel('Rating')
plt.ylabel('Frequency')
plt.title(f'Bar Graph of Rating')
plt.xticks(rotation=90)
plt.grid(True)

#Calculate and print skewness and kurtosis
skewness = df['Rating'].skew()
kurtosis = df['Rating'].kurtosis()
print(f"Skewness: {skewness}")
print(f"Kurtosis: {kurtosis}")

#Show the plot
plt.show()


```

```
Skewness: 0.018729142314406803
Kurtosis: -0.443772323456598
```



OBSERVATION

1. Skewness:

- The skewness value is approximately 0.0187, which is very close to 0. This suggests that the distribution of ratings is nearly symmetric, with a slight right (positive) skew. This means that while the majority of ratings are clustered around the mean, there may be some higher ratings that are pulling the distribution slightly to the right.

2. Kurtosis:

- The kurtosis value is approximately -0.44, which is less than 3. This indicates that the distribution of ratings is platykurtic, meaning it has thinner tails and is less peaked than a normal distribution.

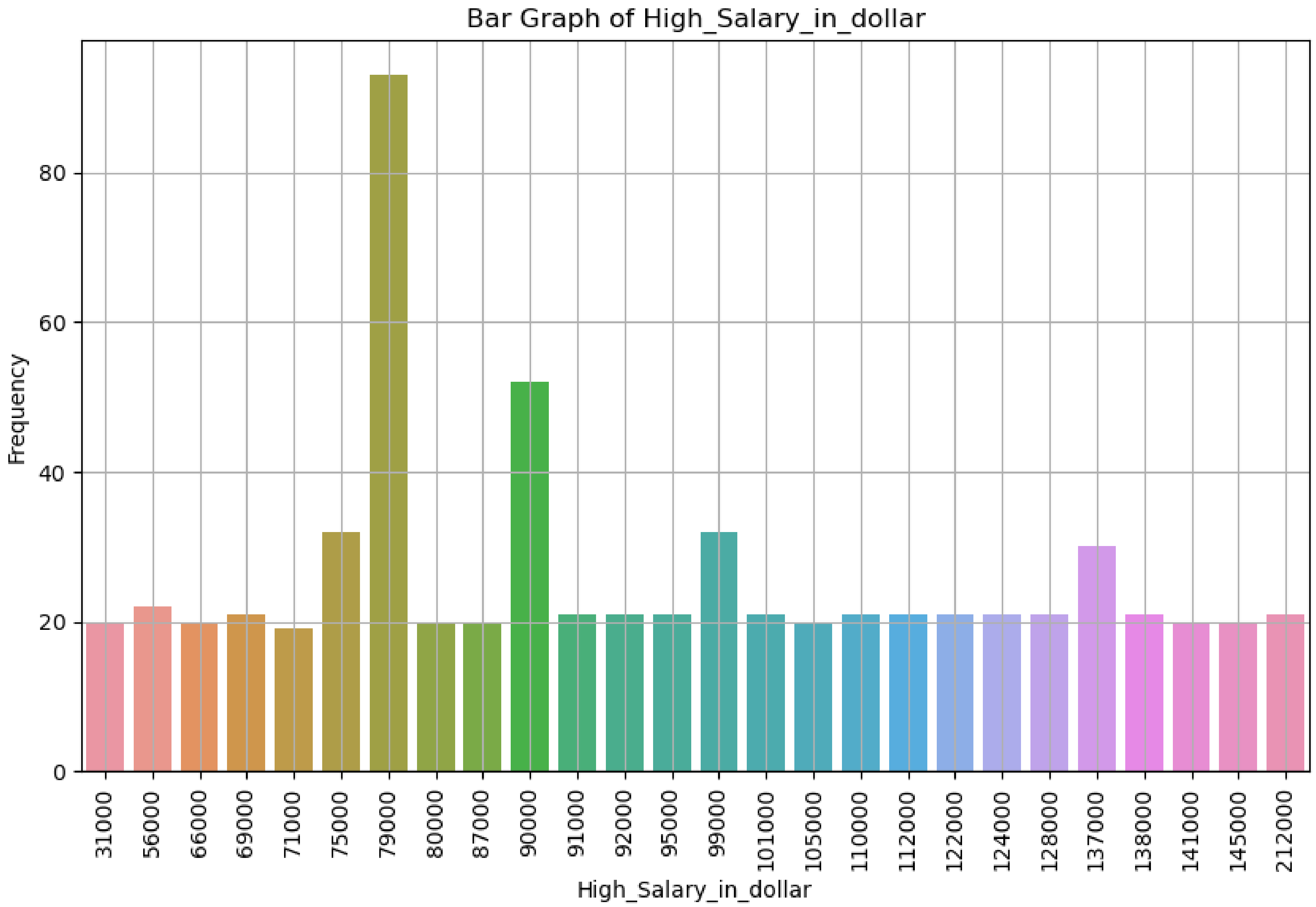
3. '3.5' is the rating by employees mostly

Finding skewness and kurtosis for High_Salary_in_dollar(Upper limit of salary)

In [55]:

```python
#Calculate skewness and kurtosis
skewness = df['High_Salary_in_dollar'].skew()
kurtosis = df['High_Salary_in_dollar'].kurt()

#Count the frequency of each unique value in the column
value_counts = df['High_Salary_in_dollar'].value_counts()

#Plot the bar graph
plt.figure(figsize=(10, 6))
sns.barplot(x=value_counts.index, y=value_counts.values)
plt.xlabel('High_Salary_in_dollar')
plt.ylabel('Frequency')
plt.title(f'Bar Graph of High_Salary_in_dollar')
plt.xticks(rotation=90)
plt.grid(True)

#Calculate and print skewness and kurtosis
skewness = df['High_Salary_in_dollar'].skew()
kurtosis = df['High_Salary_in_dollar'].kurtosis()
print(f"Skewness: {skewness}")
print(f"Kurtosis: {kurtosis}")

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

```
Skewness: 1.0891390200125406
Kurtosis: 2.4071969532297586
```



Bar Graph of High_Salary_in_dollar
Frequency
0
20
40
60
80
31000
56000
66000
69000
71000
75000
79000
80000
87000
90000
91000
92000
95000
99000
101000
105000
110000
112000
122000
124000
128000
137000
138000
141000
145000
212000
High_Salary_in_dollar

1. Skewness:

- Skewness value of approximately 1.09, indicating a moderate right (positive) skew in the distribution of salaries. This suggests that there may be a few companies with relatively high salaries that are causing the distribution to be skewed to the right.

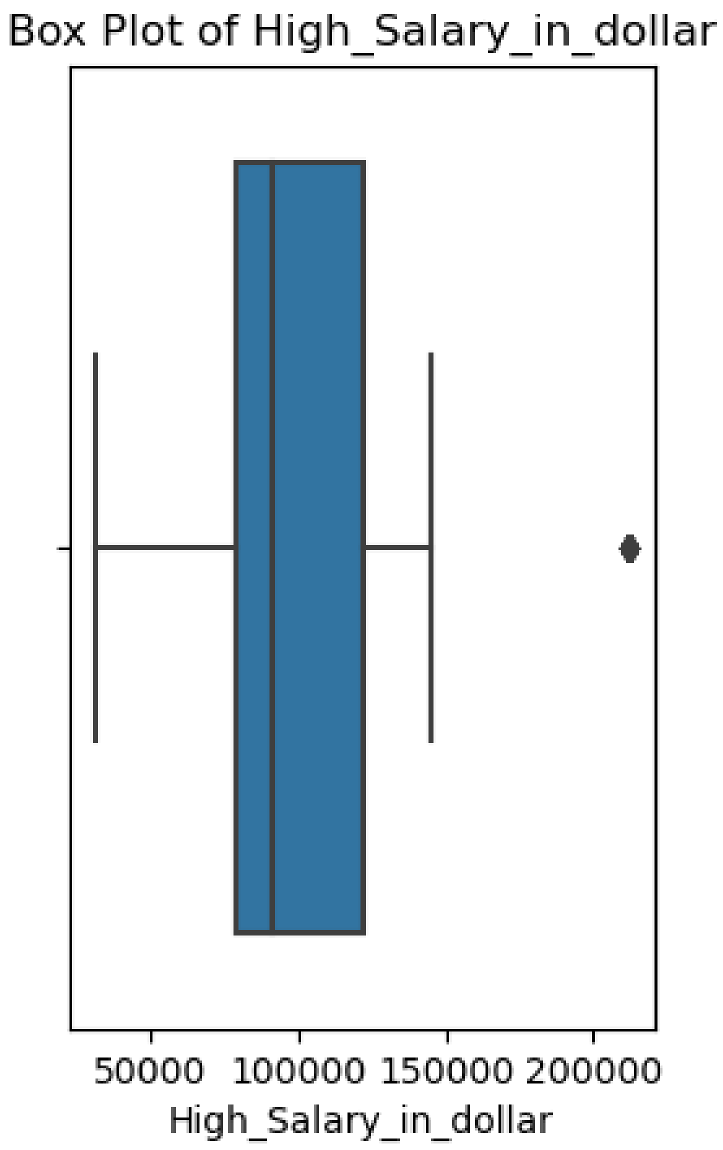
2. Kurtosis:

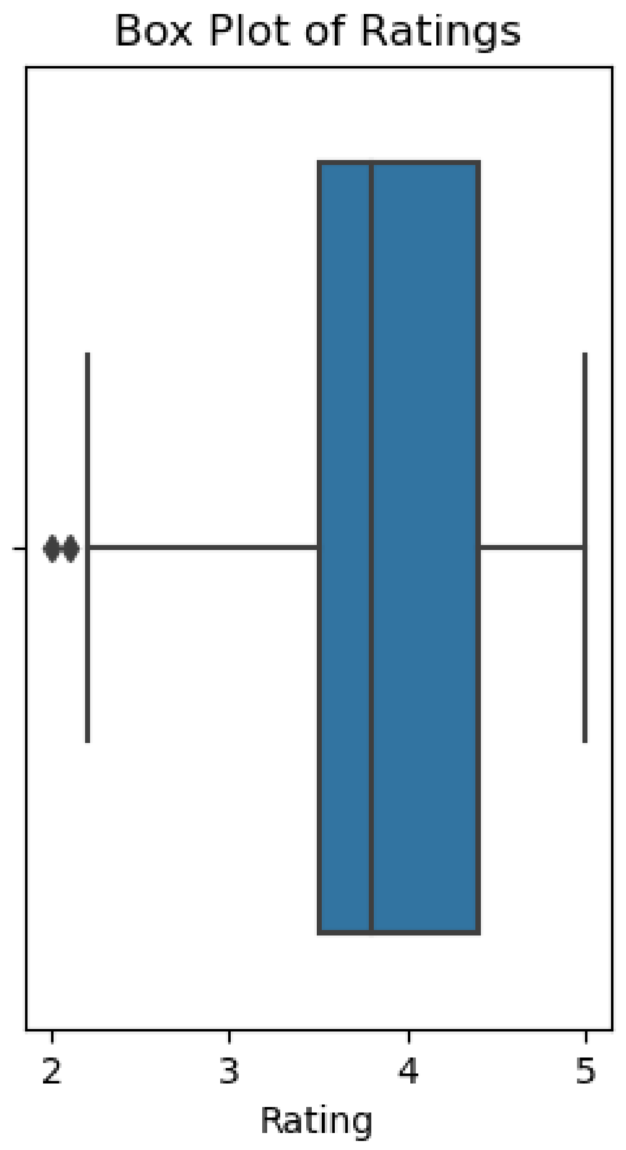
- Kurtosis value of approximately 2.41, which is greater than 3. This suggests that the distributions of salaries are leptokurtic, meaning they have heavier tails and are more peaked than a normal distribution. This indicates that there may be some outliers with very high salaries.

Finding Outliers for High_Salary_in_dollar(Upper limit of salary) and Rating

In [58]:

```
#Visualize the distribution of 'High-Salary-in-dollar' box plot
plt.subplot(1, 2, 2)
sns.boxplot(x=df['High_Salary_in_dollar'])
plt.title('Box Plot of High_Salary_in_dollar')
plt.xlabel('High_Salary_in_dollar')

plt.show()

#Visualize the distribution of 'Rating' using a histogram and box plot
plt.subplot(1, 2, 2)
sns.boxplot(x=df['Rating'])
plt.title('Box Plot of Ratings')
plt.xlabel('Rating')

plt.show()
```

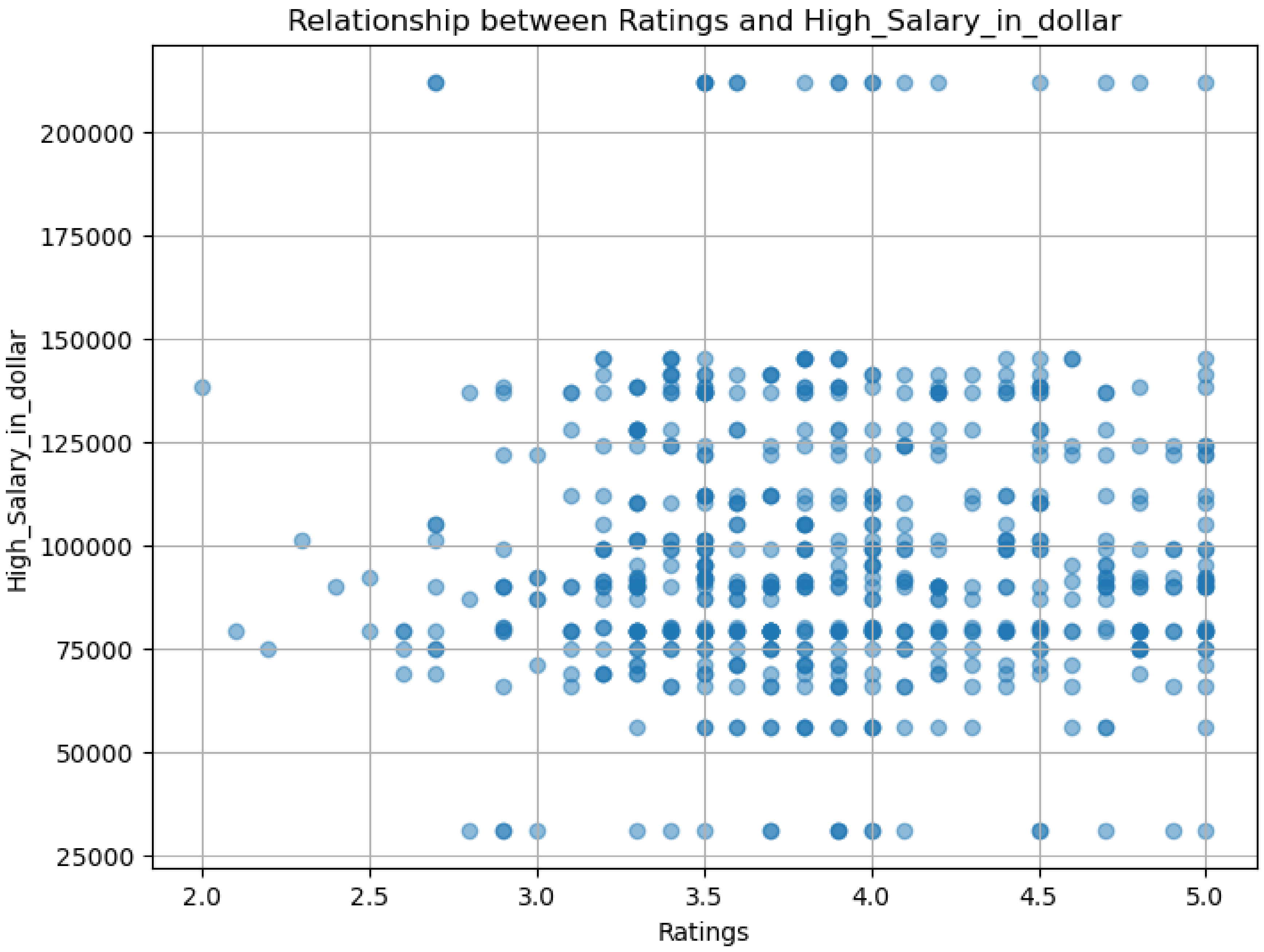

Box Plot of High_Salary_in_dollar
50000
100000
150000
200000
High_Salary_in_dollar


Box Plot of Ratings
2
3
4
5
Rating

Finding relationship between rating and salary paid

In [59]:

```
ratings = df['Rating']
salaries = df['High_Salary_in_dollar']

#Create a scatter plot to visualize the relationship between ratings and salaries
plt.figure(figsize=(8, 6))
plt.scatter(ratings, salaries, alpha=0.5)
plt.title('Relationship between Ratings and High_Salary_in_dollar')
plt.xlabel('Ratings')
plt.ylabel('High_Salary_in_dollar')
plt.grid(True)
plt.show()

#Calculate the correlation coefficient between ratings and salaries
correlation_coefficient = ratings.corr(salaries)
print("Correlation Coefficient:", correlation_coefficient)
```


Relationship between Ratings and High_Salary_in_dollar
High_Salary_in_dollar
Ratings
200000
175000
150000
125000
100000
75000
50000
25000
2.0
2.5
3.0
3.5
4.0
4.5
5.0

```
Correlation Coefficient: 0.009875247968025304
```

OBSERVATION

a correlation coefficient of 0.0099 indicates a very weak positive relationship between the Ratings and High-Salary-in-dollar.

How does the salary vary across different locations?

In [61]:

```
salary_by_location = df.groupby('Location')['High_Salary_in_dollar'].agg(['mean', 'median', 'min',

#Set the option to display all rows and columns
pd.set_option('display.max_rows', None)
pd.set_option('display.max_columns', None)

print("\nSalary Variation by Location:")
print(salary_by_location.to_string())
```

```
Salary Variation by Location:
                                    mean    median     min     max
Location
Adelphi, MD                108500.000000  108500.0  105000  112000
Akron, OH                   75000.000000   75000.0   75000   75000
Alexandria, VA              85000.000000   87000.0   79000   87000
Alpharetta, GA              69000.000000   69000.0   69000   69000
Ann Arbor, MI              134500.000000  134500.0  124000  145000
Annapolis Junction, MD      82200.000000   80000.0   31000  122000
Appleton, WI               137000.000000  137000.0  137000  137000
Arlington, VA               86333.333333   87000.0   31000  141000
Ashburn, VA                 79000.000000   79000.0   79000   79000
Atlanta, GA                 91285.714286   87000.0   31000  145000
Aurora, CO                  99000.000000   99000.0   99000   99000
Austin, TX                 115000.000000  128000.0   79000  138000
Baltimore, MD               84800.000000   87000.0   71000   95000
Beavercreek, OH             71000.000000   71000.0   71000   71000
Bedford, MA                118000.000000  137000.0   79000  138000
```

OBSERVATION

1. Salary Range Variation:

- The salary range varies significantly across different locations.
- For example, salaries in "San Francisco, CA" have a wide range, from 31,000 to 145,000, indicating a high level of income disparity within the city. Conversely, some locations like "Akron, OH" and "Columbia, SC" have a consistent salary of 75,0000.

2. High-Paying Locations:

- Some locations stand out as having higher average and maximum salaries.
- Locations like "Palo Alto, CA," "Mountain View, CA," and "Menlo Park, CA" have high median salaries, suggesting that they are tech hubs with well-paying jobs. Similarly, "Wilmington, DE" has the highest maximum salary of 212,000.

3. Low-Paying Locations:

- Locations such as "Colorado Springs, CO," "San Antonio, TX," and "Tulsa, OK" have relatively low salaries, with minimum salaries of 31,000, 66,000, and 31,000, respectively.

4. Mid-Range Salaries:

- Many locations have median salaries in the 70,000 to 100,000 range. This includes cities like "Chicago, IL," "Dallas, TX," and "Seattle, WA."

5. Outliers:

- Some locations have outliers with significantly higher salaries, such as "Fort Sam Houston, TX" with a maximum salary of 212,000 *and "NewYork,NY"with a maximum salary of* 212,000. These outliers could be due to specific high-paying industries or roles.

6. Regional Differences:

- There are noticeable variations in salaries within the same state or region. For example, in Virginia, you have "McLean, VA" with a median of 101,000 *and "Arlington,VA" with a median of* 87,000. This suggests that factors like proximity to major cities or industries can impact salaries.

7. Uniform Salaries:

- Some locations have uniform salaries with no variation. For instance, "Fort Belvoir, VA," "Chantilly, VA," and "Foster City, CA" all have the same salary for mean, median, minimum, and maximum.

8. Lack of Data:

- Some locations have limited data points, which could affect the accuracy of the statistics. For instance, "California" and "Remote" each have only one data point.

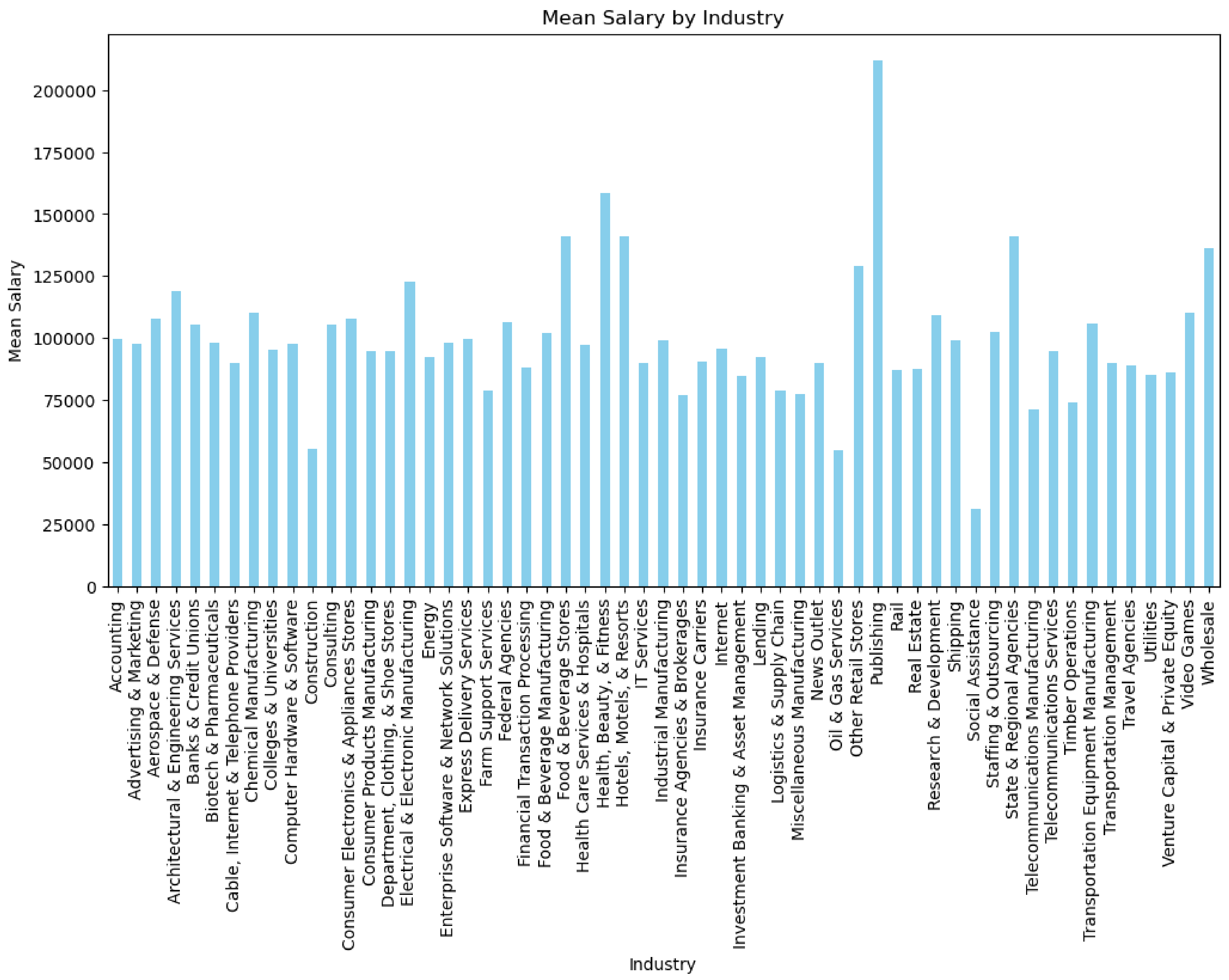
9. Influence of Industry:

- It's clear that the type of industries and job markets in each location play a significant role in determining salary levels. Tech-heavy areas tend to have higher salaries, while other areas may have lower salaries due to different economic factors.

Finding salary with respect to the industry

In [63]:

```
#Group the data by 'Industry' and calculate summary statistics of salary for each industry
salary_by_industry = df.groupby('Industry')['High_Salary_in_dollar'].agg(['mean'])

#Create a bar chart for mean salary by industry
plt.figure(figsize=(12, 6))  # Adjust the figure size as needed
salary_by_industry['mean'].plot(kind='bar', color='skyblue')
plt.xlabel('Industry')
plt.ylabel('Mean Salary')
plt.title('Mean Salary by Industry')
plt.xticks(rotation=90)  # Rotate x-axis labels for better readability
plt.show()

```



Mean Salary by Industry
Mean Salary
0
25000
50000
75000
100000
125000
150000
175000
200000
Industry
Accounting
Advertising & Marketing
Aerospace & Defense
Architectural & Engineering Services
Banks & Credit Unions
Biotech & Pharmaceuticals
Cable, Internet & Telephone Providers
Chemical Manufacturing
Colleges & Universities
Computer Hardware & Software
Construction
Consulting
Consumer Electronics & Appliances Stores
Consumer Products Manufacturing
Department, Clothing, & Shoe Stores
Electrical & Electronic Manufacturing
Energy
Enterprise Software & Network Solutions
Express Delivery Services
Farm Support Services
Federal Agencies
Financial Transaction Processing
Food & Beverage Manufacturing
Food & Beverage Stores
Health Care Services & Hospitals
Health, Beauty, & Fitness
Hotels, Motels, & Resorts
IT Services
Industrial Manufacturing
Insurance Agencies & Brokerages
Insurance Carriers
Internet
Investment Banking & Asset Management
Lending
Logistics & Supply Chain
Miscellaneous Manufacturing
News Outlet
Oil & Gas Services
Other Retail Stores
Publishing
Rail
Real Estate
Research & Development
Shipping
Social Assistance
Staffing & Outsourcing
State & Regional Agencies
Telecommunications Manufacturing
Telecommunications Services
Timber Operations
Transportation Equipment Manufacturing
Transportation Management
Travel Agencies
Utilities
Venture Capital & Private Equity
Video Games
Wholesale

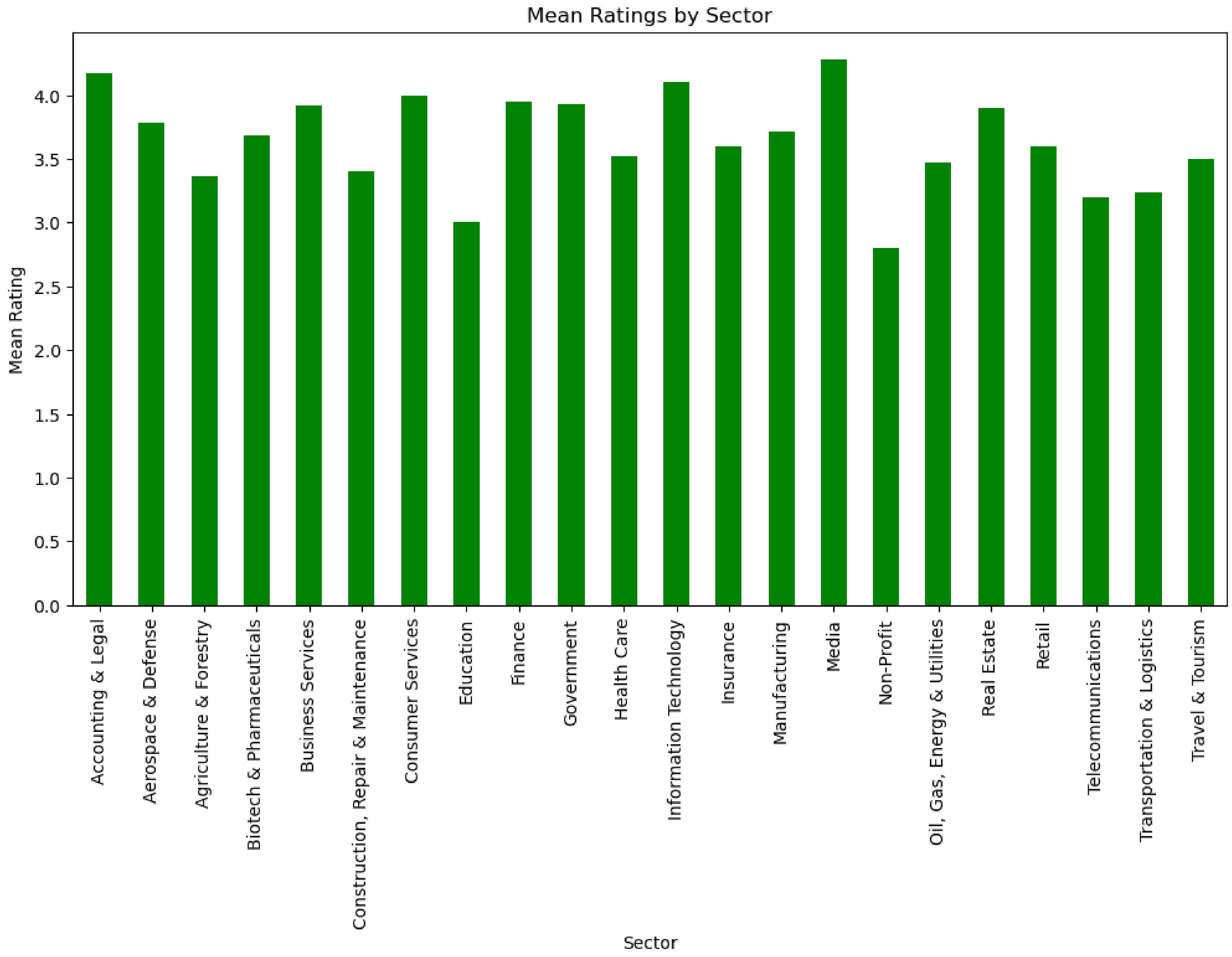
OBSERVATION

Industry plays a crucial role in determining salary levels. Certain industries, such as "Health, Beauty, & Fitness" and "Publishing," have higher mean and median salaries compared to industries like "Social Assistance" and "Oil & Gas Services."

How do ratings vary between different sectors?

In [65]:

```
#Group the data by 'Sector' and calculate summary statistics of ratings for each sector
ratings_by_sector = df.groupby('Sector')['Rating'].agg(['mean', 'median', 'min', 'max'])

#Create a bar chart for mean ratings by sector
plt.figure(figsize=(12, 6))  # Adjust the figure size as needed
ratings_by_sector['mean'].plot(kind='bar', color='green')
plt.xlabel('Sector')
plt.ylabel('Mean Rating')
plt.title('Mean Ratings by Sector')
plt.xticks(rotation=90)  # Rotate x-axis labels for better readability
plt.show()
```


Mean Ratings by Sector
Mean Rating
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Accounting & Legal
Aerospace & Defense
Agriculture & Forestry
Biotech & Pharmaceuticals
Business Services
Construction, Repair & Maintenance
Consumer Services
Education
Finance
Government
Health Care
Information Technology
Insurance
Manufacturing
Media
Non-Profit
Oil, Gas, Energy & Utilities
Real Estate
Retail
Telecommunications
Transportation & Logistics
Travel & Tourism
Sector

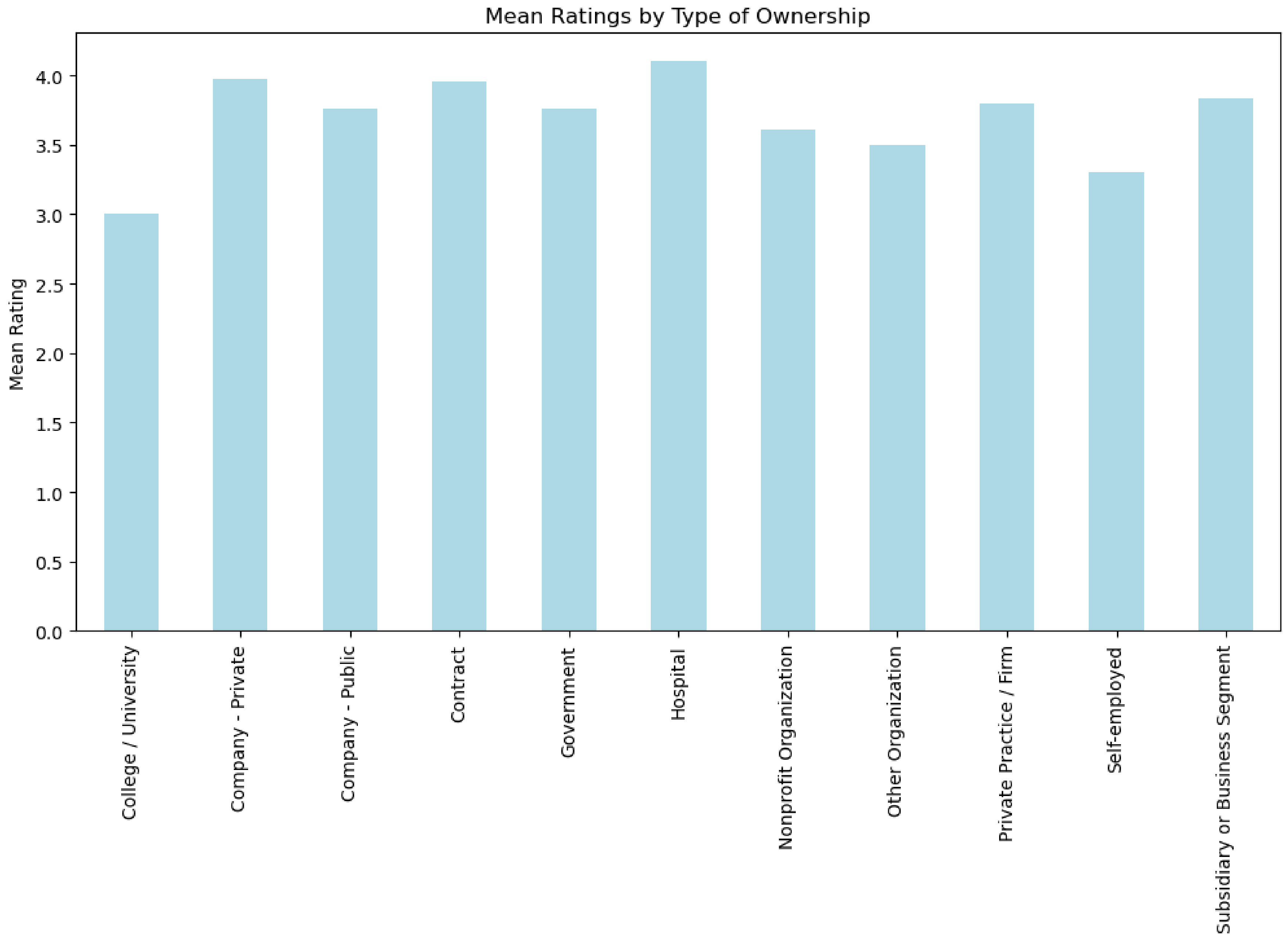
OBSERVATION

- The Media sector tends to have the highest average ratings, indicating generally positive employee sentiment.
- Employees in the Education sector and Non-Profit organizations tend to give lower ratings on average compared to other sectors and ownership types

Finding relationship rating and type of ownership

In [66]:

```
#Group the data by 'Type of ownership' and calculate summary statistics of ratings for each type o
ratings_by_ownership = df.groupby('Type of ownership')['Rating'].agg(['mean', 'median', 'min', 'ma

#Create a bar chart for mean ratings by type of ownership
plt.figure(figsize=(12, 6))  # Adjust the figure size as needed
ratings_by_ownership['mean'].plot(kind='bar', color='lightblue')
plt.xlabel('Type of Ownership')
plt.ylabel('Mean Rating')
plt.title('Mean Ratings by Type of Ownership')
plt.xticks(rotation=90)  # Rotate x-axis labels for better readability
plt.show()
```


Mean Ratings by Type of Ownership
Mean Rating
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
College / University
Company - Private
Company - Public
Contract
Government
Hospital
Nonprofit Organization
Other Organization
Private Practice / Firm
Self-employed
Subsidiary or Business Segment
Type of Ownership

OBSERVATION

- Hospital ownership types having the highest average ratings.

Which companies have the most job postings in the dataset?

In [67]:

```python
#Perform a frequency count of each unique company name
company_job_postings_count = df['Company Name'].value_counts()

#Get the companies with the highest job posting counts
companies_with_most_job_postings = company_job_postings_count.head()

print("Companies with the Most Job Postings:")
print(companies_with_most_job_postings)
```

```
Companies with the Most Job Postings:
Hatch Data Inc          12
Maxar Technologies      12
Tempus Labs             11
AstraZeneca             10
Klaviyo                  8
Name: Company Name, dtype: int64
```

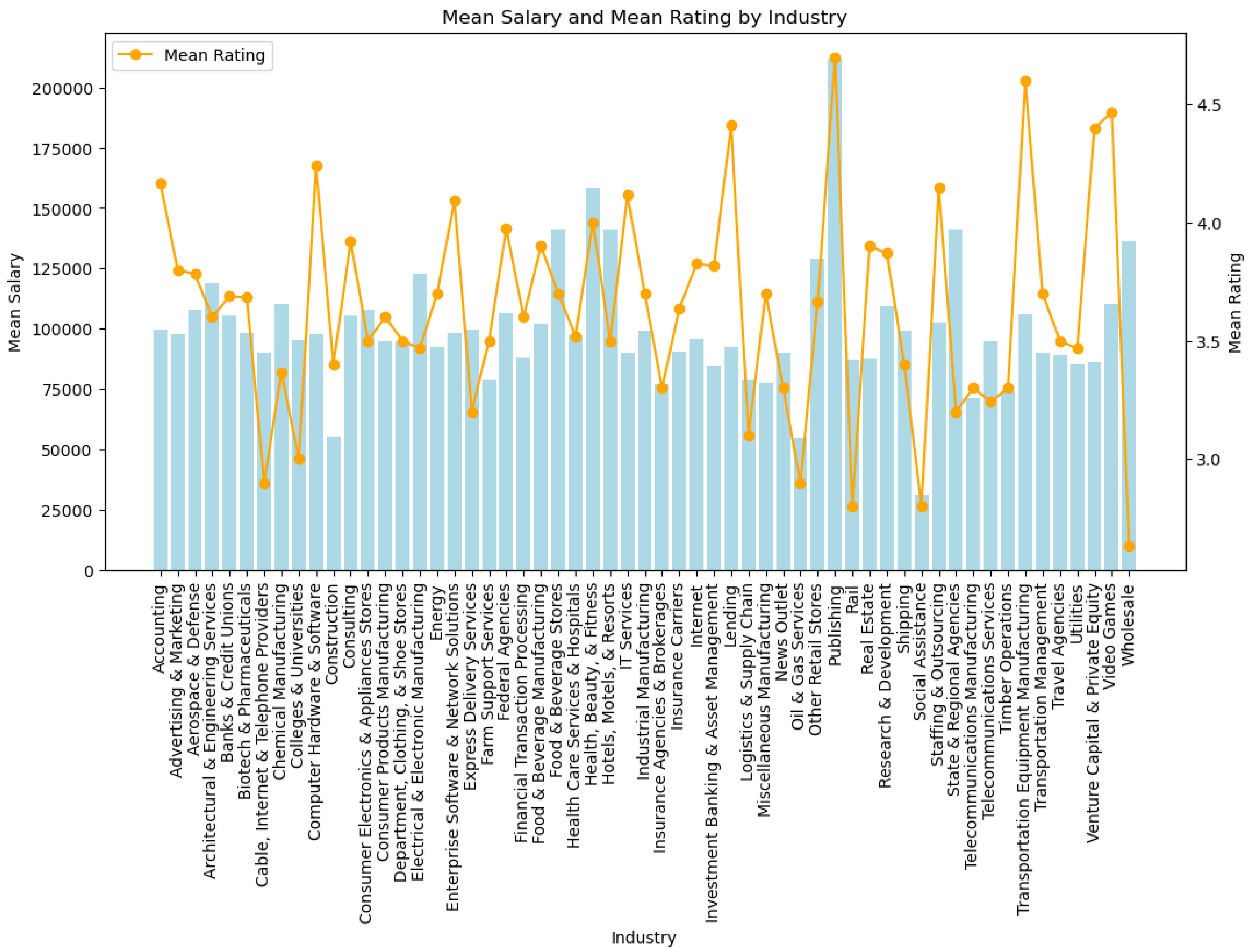
OBSERVATION

The companies listed with the most job postings are actively recruiting and may present various job opportunities for individuals seeking employment. Job seekers interested in these companies should explore the specific job listings to identify

Compare Industry with respect to salary and rating

In [71]:

```python
#Group the data by 'Industry' and calculate metrics within each group
grouped_data = df.groupby('Industry')
metrics_within_groups = grouped_data.agg({
    'High_Salary_in_dollar': 'mean',
    'Rating': 'mean'
    })

#Create a bar chart to visualize the mean salary and mean rating within each industry
plt.figure(figsize=(12, 6))

#Plot mean salary
plt.bar(metrics_within_groups.index, metrics_within_groups['High_Salary_in_dollar'], color='lightb
plt.xlabel('Industry')
plt.ylabel('Mean Salary')
plt.xticks(rotation=90)

#Create a secondary y-axis for mean rating
plt.twinx()
plt.plot(metrics_within_groups.index, metrics_within_groups['Rating'], marker='o', color='orange',
plt.ylabel('Mean Rating')

plt.title('Mean Salary and Mean Rating by Industry')
plt.legend(loc='upper left')
plt.show()
```



Mean Salary and Mean Rating by Industry
Mean Rating
Mean Salary
Mean Rating
Industry
0
25000
50000
75000
100000
125000
150000
175000
200000
3.0
3.5
4.0
4.5
Accounting
Advertising & Marketing
Aerospace & Defense
Architectural & Engineering Services
Banks & Credit Unions
Biotech & Pharmaceuticals
Cable, Internet & Telephone Providers
Chemical Manufacturing
Colleges & Universities
Computer Hardware & Software
Construction
Consulting
Consumer Electronics & Appliances Stores
Consumer Products Manufacturing
Department, Clothing, & Shoe Stores
Electrical & Electronic Manufacturing
Energy
Enterprise Software & Network Solutions
Express Delivery Services
Farm Support Services
Federal Agencies
Financial Transaction Processing
Food & Beverage Manufacturing
Food & Beverage Stores
Health Care Services & Hospitals
Health, Beauty, & Fitness
Hotels, Motels, & Resorts
IT Services
Industrial Manufacturing
Insurance Agencies & Brokerages
Insurance Carriers
Internet
Investment Banking & Asset Management
Lending
Logistics & Supply Chain
Miscellaneous Manufacturing
News Outlet
Oil & Gas Services
Other Retail Stores
Publishing
Rail
Real Estate
Research & Development
Shipping
Social Assistance
Staffing & Outsourcing
State & Regional Agencies
Telecommunications Manufacturing
Telecommunications Services
Timber Operations
Transportation Equipment Manufacturing
Transportation Management
Travel Agencies
Utilities
Venture Capital & Private Equity
Video Games
Wholesale

OBSERVATION

1. Industries that offer the highest average salaries include Publishing, Health, Beauty, & Fitness, Hotels, Motels, & Resorts, Food & Beverage Stores, and State & Regional Agencies.
2. Industries with the highest average ratings from employees include Health, Beauty, & Fitness, Publishing, Video Games, Staffing & Outsourcing, Computer Hardware & Software, and Lending.