

# 733 assignment 1-Analysing the job market trends(year 2021)

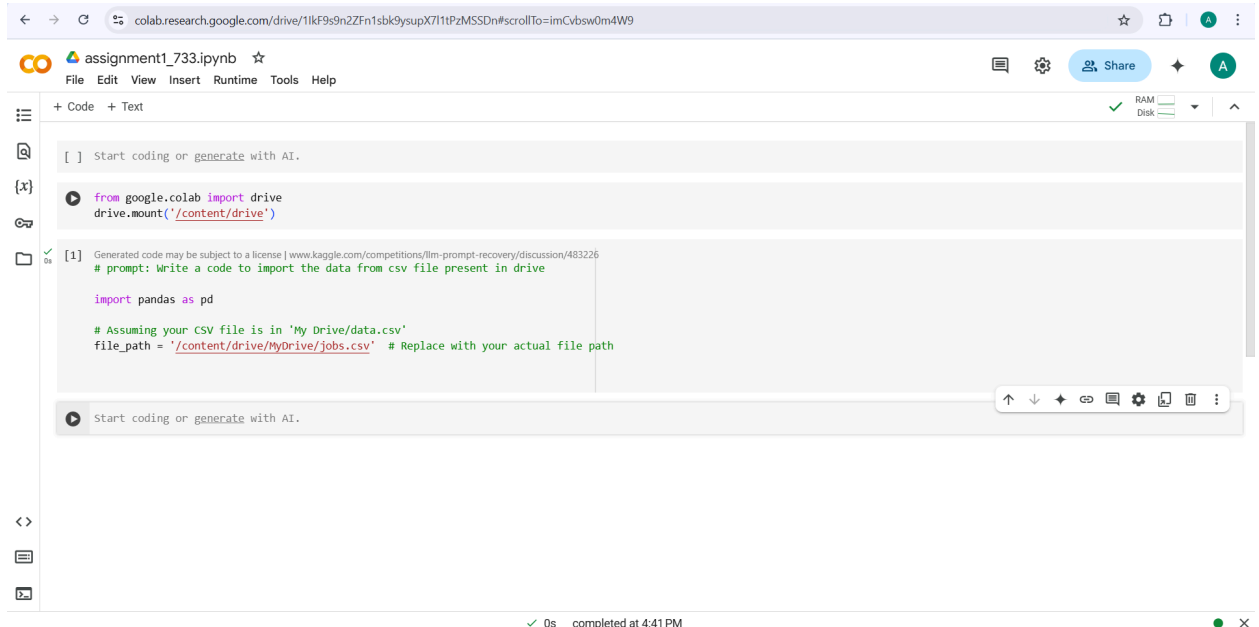
Google collab link- [assignment1\\_733.ipynb](#)

PPT link- [733\\_assignment1](#)

Data set link

<https://github.com/Mlawrence95/LinkedIn-Tech-Job-Data/blob/main/jobs.csv>

## 1) Mounting dataset



The screenshot shows a Google Colab notebook titled 'assignment1\_733.ipynb'. The code cell contains the following Python code:

```
[ ] Start coding or generate with AI.

from google.colab import drive
drive.mount('/content/drive')

[1] Generated code may be subject to a license | www.kaggle.com/competitions/flm-prompt-recovery/discussion/483226
# prompt: Write a code to import the data from csv file present in drive

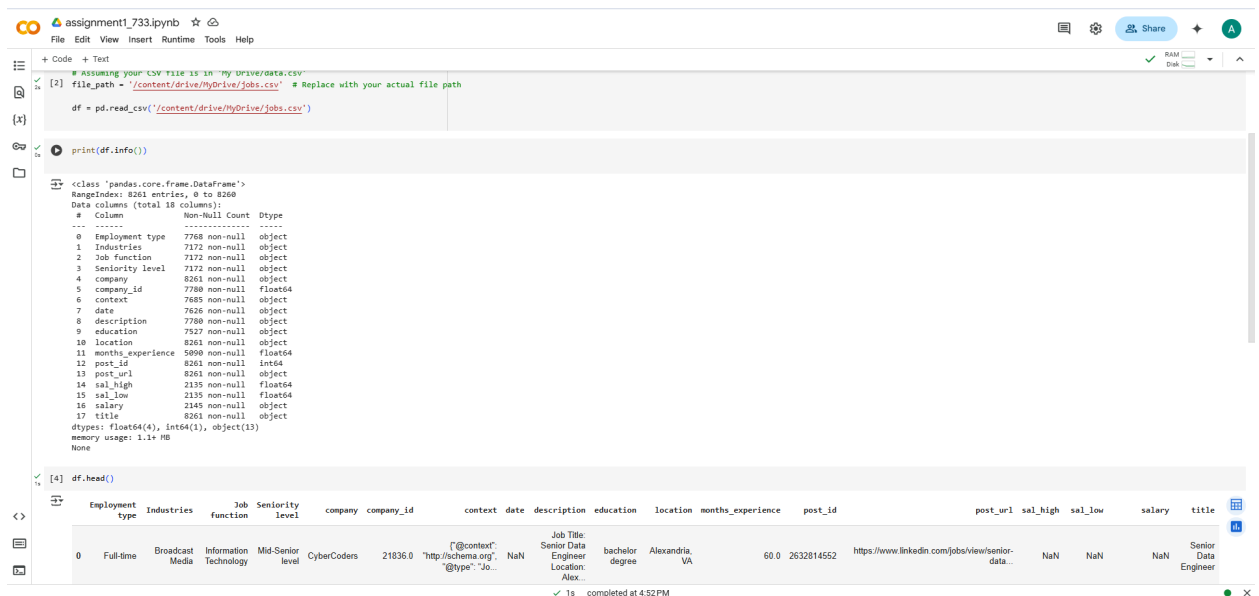
import pandas as pd

# Assuming your CSV file is in 'My Drive/data.csv'
file_path = '/content/drive/MyDrive/jobs.csv' # Replace with your actual file path

Start coding or generate with AI.
```

The status bar at the bottom indicates '0s completed at 4:41 PM'.

## 2) Exploring the Dataset



The screenshot shows the same Google Colab notebook with the following code and output:

```
[2] # Assuming your CSV file is in 'My Drive/data.csv'
file_path = '/content/drive/MyDrive/jobs.csv' # Replace with your actual file path

df = pd.read_csv('/content/drive/MyDrive/jobs.csv')

print(df.info())
```

The output of `df.info()` is:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8261 entries, 0 to 8260
Data columns (total 18 columns):
 #   Column              Non-Null Count  Dtype  
---  --
 0   Employment type      7768 non-null  object  
 1   Industries           7172 non-null  object  
 2   Job function         7172 non-null  object  
 3   Seniority level      7172 non-null  object  
 4   company             8261 non-null  object  
 5   company_id          7780 non-null  float64  
 6   context             7685 non-null  object  
 7   date               7626 non-null  object  
 8   description          7780 non-null  object  
 9   education           7527 non-null  object  
10  location            8261 non-null  object  
11  months_experience    5890 non-null  float64  
12  post_id            8261 non-null  int64   
13  post_url            8261 non-null  object  
14  sal_high           2135 non-null  float64  
15  sal_low            2135 non-null  float64  
16  salary             2145 non-null  object  
17  title              8261 non-null  object  
dtypes: float64(4), int64(1), object(13)
memory usage: 1.1+ MB
None
```

The output of `df.head()` is:

	Employment type	Industries	Job function	Seniority level	company	company_id	context	date	description	education	location	months_experience	post_id	post_url	sal_high	sal_low	salary	title
0	Full-time	Broadcast Media	Information Technology	Mid-Senior level	CyberCoders	21836.0	"@context": "http://schema.org", "@type": "Job..."	NaN	Job Title: Senior Data Engineer Location: Alexandria, VA	bachelor degree	Alexandria, VA	60.0	2632814552	https://www.linkedin.com/jobs/view/senior-data...	NaN	NaN	NaN	Senior Data Engineer

The status bar at the bottom indicates '1s completed at 4:52 PM'.

df.head()

	Employment type	Industries	Job function	Seniority level	company	company_id	context	date	description	education	location
0	Full-time	Broadcast Media	Information Technology	Mid-Senior level	CyberCoders	21836.0	{"@context": "http://schema.org", "@type": "Job..."}	NaN	Job Title: Senior Data Engineer Location: Alex...	bachelor degree	Alex...
1	Full-time	Hospital & Health Care, Medical Devices, and P...	Engineering and Information Technology	Not Applicable	Johnson & Johnson	1207.0	{"@context": "http://schema.org", "@type": "Job..."}	NaN	Ethicon, part of Johnson & Johnson Medical Dev...	bachelor degree	Cl...
2	Full-time	Computer Hardware, Computer Software, and Info...	Engineering and Information Technology	Not Applicable	Microsoft	1035.0	{"@context": "http://schema.org", "@type": "Job..."}	NaN	Microsoft's WCB health team is looking for a S...	bachelor degree	Wash...
3	Full-time	Computer Hardware, Computer Software, and Info...	Engineering and Information Technology	Not Applicable	Microsoft	1035.0	{"@context": "http://schema.org", "@type": "Job..."}	NaN	Microsoft's WCB health team is looking for a S...	bachelor degree	Res...
4	Full-time	Computer Hardware, Computer Software, and Info...	Engineering and Information Technology	Not Applicable	Microsoft	1035.0	{"@context": "http://schema.org", "@type": "Job..."}	NaN	Microsoft's WCB health team is looking for a S...	bachelor degree	Irv...

print(df.describe())

	company_id	months_experience	post_id	sal_high	\
count	7.780000e+03	5090.000000	8.261000e+03	2135.000000	
mean	6.680769e+06	50.457171	2.679644e+09	150923.400937	
std	1.520843e+07	28.439375	6.876523e+07	44080.915030	
min	1.000000e+03	3.000000	1.038733e+08	100.000000	
25%	4.787000e+03	36.000000	2.663260e+09	119000.000000	
50%	1.624790e+05	48.000000	2.682321e+09	151000.000000	
75%	3.334793e+06	60.000000	2.693585e+09	180000.000000	
max	8.011415e+07	180.000000	2.764037e+09	416000.000000	

	sal_low
count	2135.000000
mean	94313.941920
std	27242.371257
min	120.000000
25%	75450.000000
50%	91300.000000
75%	113000.000000
max	312000.000000

```
print(df.isnull().sum())
#check missing value
```

↔	Employment type	493
	Industries	1089
	Job function	1089
	Seniority level	1089
	company	0
	company_id	481
	context	576
	date	635
	description	481
	education	734
	location	0
	months_experience	3171
	post_id	0
	post_url	0
	sal_high	6126
	sal_low	6126
	salary	6116
	title	0
	dtype: int64	

→ So mostly all column has some missing values. We will fill all the numeric values with the median and categorical values with the mode and will remove all the duplicates

```
[ ] # Fill numeric columns with median
numeric_columns = df.select_dtypes(include=[np.number]).columns
df[numeric_columns] = df[numeric_columns].fillna(df[numeric_columns].median())

[ ] # Fill categorical columns with mode
categorical_columns = df.select_dtypes(include=['object']).columns
df[categorical_columns] = df[categorical_columns].fillna(df[categorical_columns].mode().iloc[0])

[ ] #remove duplicates
df.drop_duplicates(inplace=True)
```

### 3) Counting the number of instances in the dataset

There are 8261 rows in the dataset

There are 18 columns in the dataset

```
Name: count, dtype: int64
```

```
[18] num_rows,num_cols = df.shape
      print("Number of rows:",num_rows)
      print("Number of columns:",num_cols)
```

```
↔ Number of rows: 8261
    Number of columns: 18
```

start coding or generate with AI.

### 4) Finding the most popular job-

From the visualization, we can see that Data Scientist has the highest number of jobs compared to other fields.

assignment1\_733.ipynb

File Edit View Insert Runtime Tools Help

Files

- ..
- .config
- drive
- sample\_data
- my\_dataset.csv

```
[15] df.to_csv('my_dataset.csv')
```

```
# Example: Count of job titles
job_title_counts = df['title'].value_counts()
print(job_title_counts.head())
```

title	
Data Scientist	769
Software Engineer	743
Machine Learning Engineer	491
Site Reliability Engineer	441
Senior Software Engineer	123

Name: count, dtype: int64

Start coding or generate with AI.

0s completed at 5:34 PM

Gemini

Okay, here's how you can download the dataset (df) from Google Colab to your local machine:

1. Using `df.to_csv()`:

- This method converts your DataFrame into a CSV (Comma-Separated Values) file, which you can then download.
- Specify a filename for the CSV file (e.g., 'my\_dataset.csv') and use the `index=False` argument to avoid saving the DataFrame's index.

```
df.to_csv('my_dataset.csv', index=False)
```

Use code with caution

- After running this code, a file named 'my\_dataset.csv' will appear in the Colab's file explorer (usually on the left sidebar).
- Right-click the file and select "Download" to save it to your local machine.

Enter a prompt here

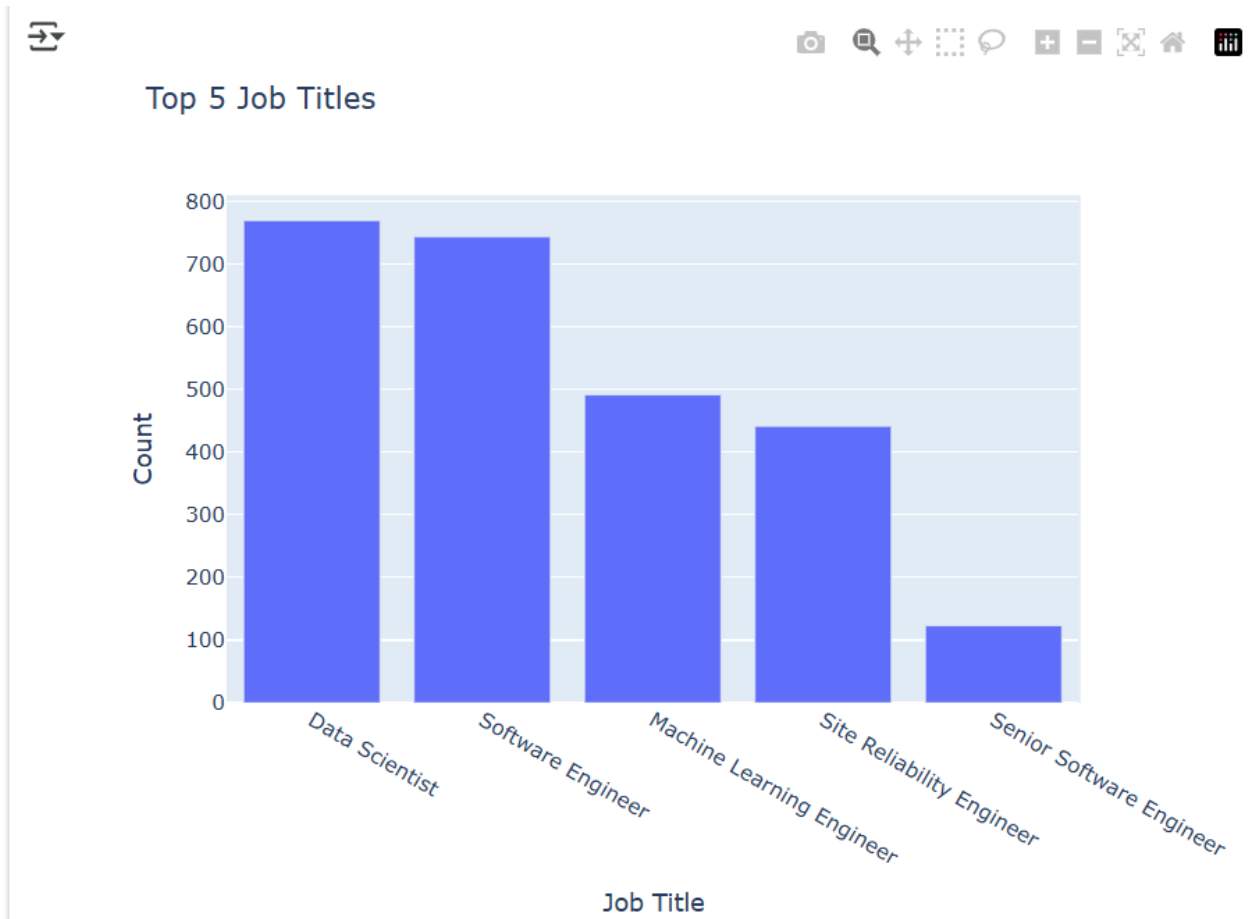
0 / 2000

Responses may display inaccurate or offensive information that doesn't represent Google's views. [Learn more](#)

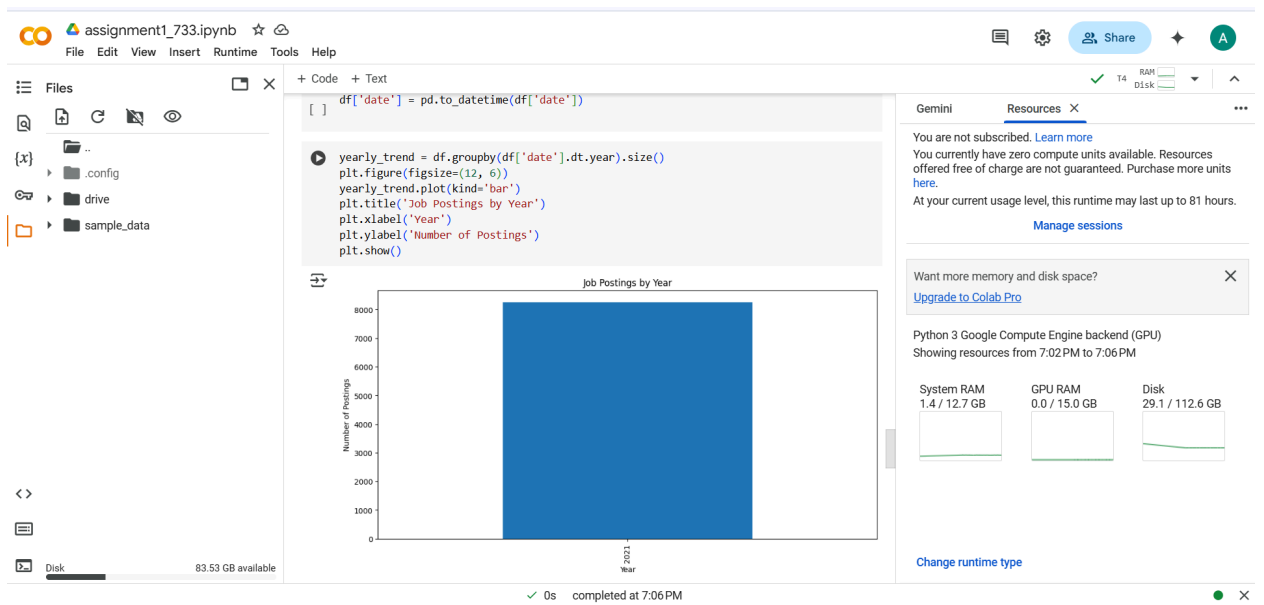
```
import plotly.express as px

# Get the top 5 job titles
job_title_counts = df['title'].value_counts().reset_index().head(5)
job_title_counts.columns = ['Job Title', 'Count'] # Rename columns

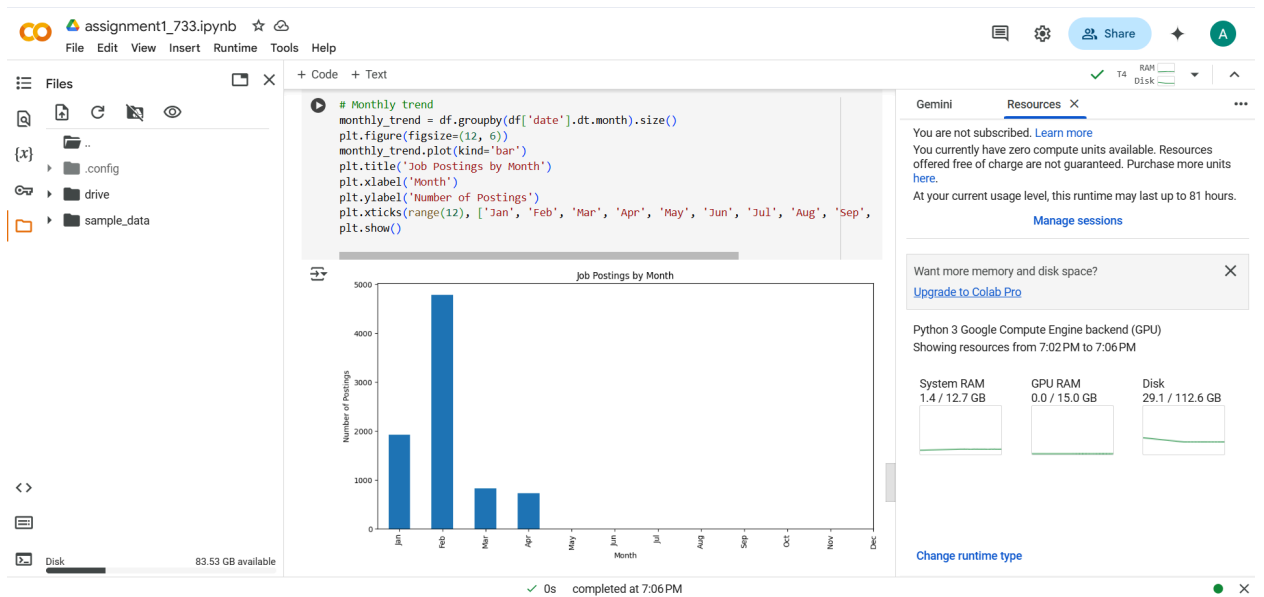
# Create the bar graph for the top 5 titles
fig = px.bar(job_title_counts, x='Job Title', y='Count',
              title='Top 5 Job Titles',
              labels={'Job Title': 'Job Title', 'Count': 'Count'})
fig.show()
```



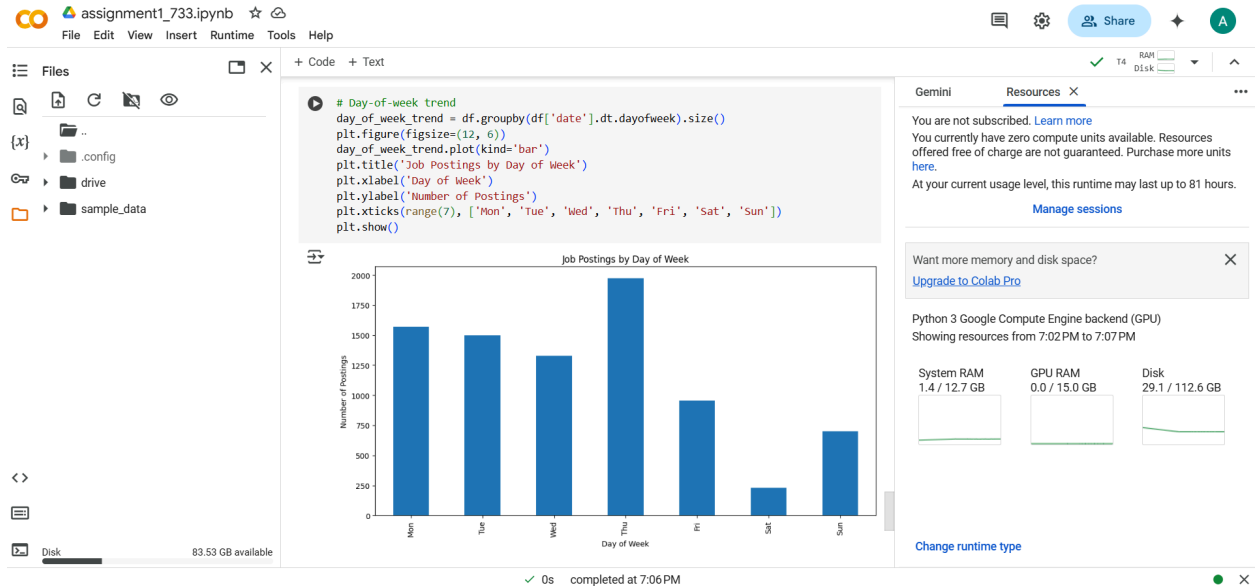
## 5) Job Posting trend yearly



## 6) Job posting trend in distribution of Months

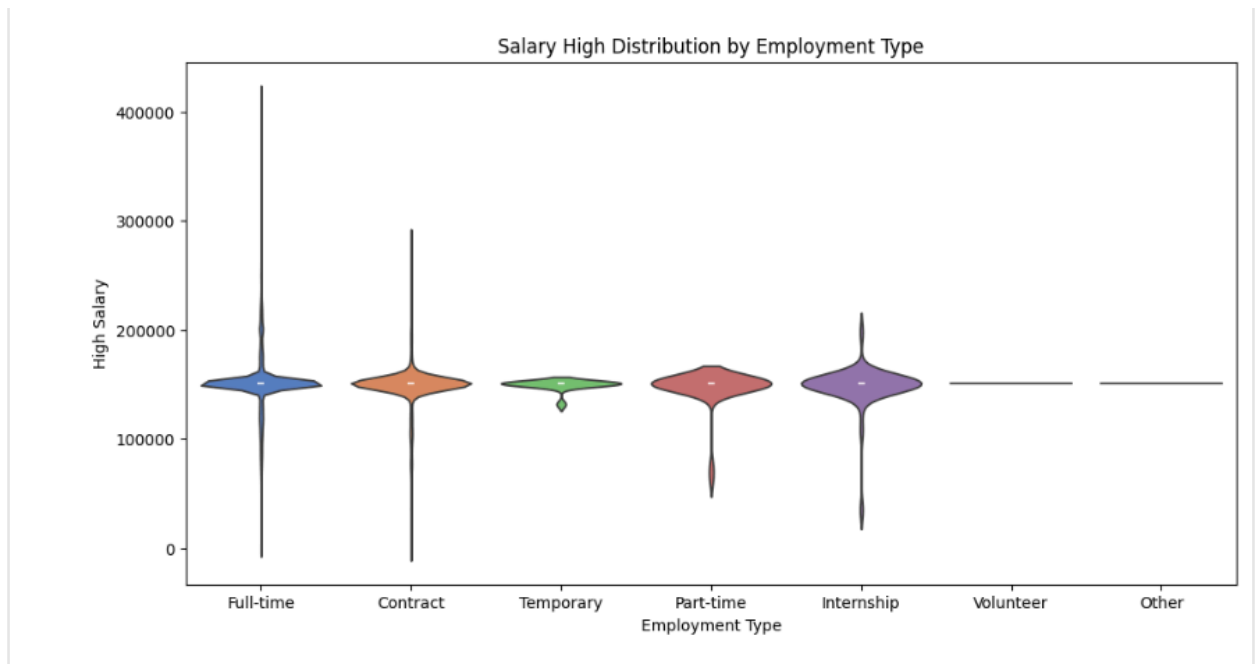


## 7) Job posting trend in distribution of days of the Week



## 8) Violin plots to visualize the distribution of sal\_high and sal\_low for different categories like employment\_type, seniority\_level

### A) High Salary Distribution by Employment Type



## Understanding the Violin Plot Components:

- **Vertical Shape (Width):** Shows the **density** of salary data at different levels.
- **White Dot (Median):** Represents the **median high salary** for each employment type.
- **Thick Bar (IQR):** Displays the **interquartile range (25th–75th percentile)**.
- **Thin Line (Whiskers):** Illustrates the **full range** of data, excluding outliers.

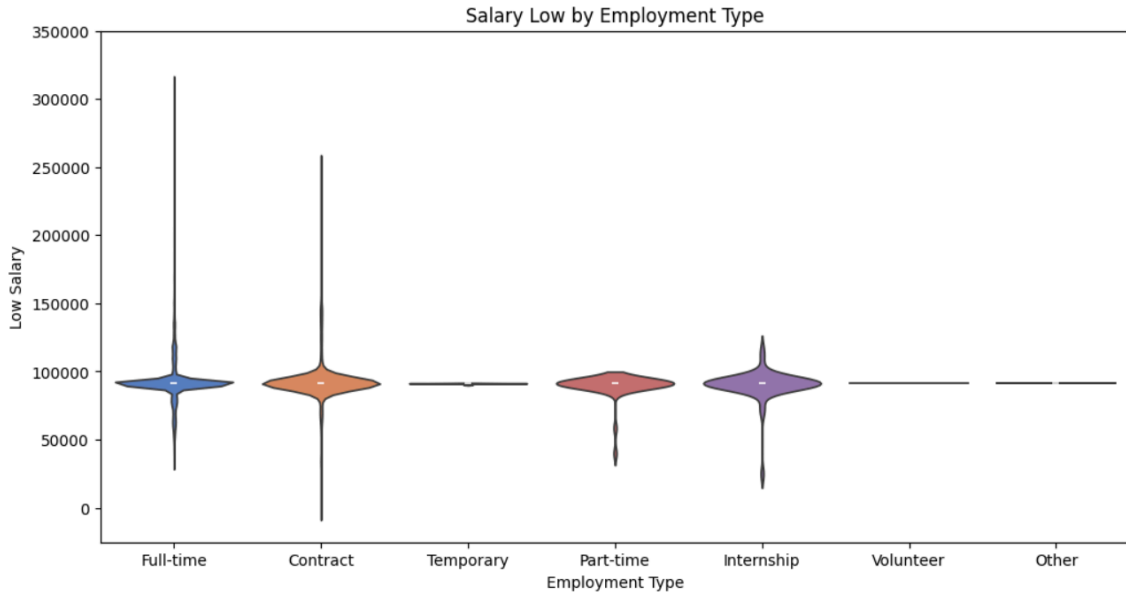
## Key Insights:

- **Full-Time:**
  - **Widest distribution** indicating **diverse salary ranges** depending on industry and role.
  - **Higher median** compared to other employment types, reflecting the stable and lucrative nature of full-time jobs.
- **Internship:**
  - Displays a **long upper tail**, suggesting **some internships offer competitive pay**, potentially in **tech or finance** sectors.
- **Part-Time & Temporary:**
  - More **concentrated salary ranges**, implying **standard hourly wages**.
  - **Lower median salaries**, indicating these positions generally offer less compensation.
- **Volunteer:**
  - **No variability**, suggesting **unpaid roles** across the dataset.



## B) Low Salary Distribution by Employment Type

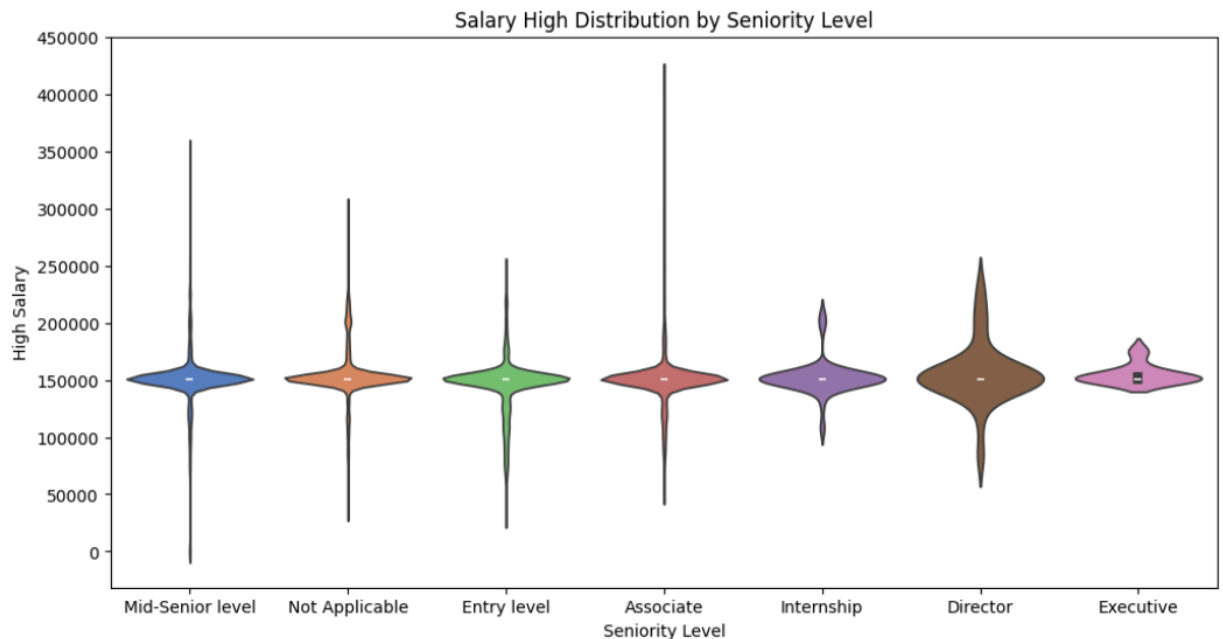




### Insights from the Plot:

- **Full-Time:**
  - **Median low salary:** ~\$90,000.
  - Most salaries are **clustered in a narrow range**, showing **consistency** in base pay for full-time roles.
- **Contract:**
  - **Similar median** to full-time positions but with a **wider spread**—some **low salaries approach zero**, reflecting **part-time contracts** or **unpaid trial periods**.
- **Temporary & Part-Time:**
  - **Compressed distributions** with medians around \$90,000, indicating **uniform low salaries**, likely due to **regulated hourly rates**.
- **Internship:**
  - **Greater spread** than part-time roles.
  - Some salaries close to zero, suggesting **stipend-based** or **unpaid internships**.
- **Volunteer & Other:**
  - **Very narrow distributions**, confirming these roles are either **unpaid** or have **fixed compensation structures**.

### C) High Salary Distribution by Seniority Level



<ipython-input-28-50e15070a5cd>:23: FutureWarning:

### Insights from the Plot:

- **Executive Level:**
  - **Highest median high salary** among all levels.
  - **Narrow distribution**, indicating **less variation** in top executive pay.
- **Director Level:**
  - **Broad salary range with high upper outliers**, indicating some **directors** earn salaries comparable to executives.
- **Mid-Senior & Associate Levels:**
  - **Similar median salaries** but with **wide spreads**, highlighting **overlaps in earnings potential** between these levels.
  - **Greater variability** suggests **industry-specific salary adjustments**.
- **Entry Level:**
  - **Lower median salaries** with a **narrower distribution**, indicating **stable starting pay**.
- **Internship:**
  - **Lowest median high salary** with **occasional outliers**, representing **highly paid internships** in **competitive sectors**.

## D) Low Salary Distribution by Seniority Level



### Insights from the Plot:

- **Executive Level:**
  - **Highest median low salary** across all seniority levels.
  - **Narrow spread**, indicating **stable base salaries** at the executive tier.
- **Director Level:**
  - **Widest spread**, showing **high variability** in base pay.
  - **Outliers** suggest some **directors earn salaries equivalent to executives**.
- **Mid-Senior & Associate Levels:**
  - **Similar median low salaries** with **wide spreads**, indicating **salary overlaps** between these levels.
  - **Associates** display a **broad salary range**, reflecting **inconsistencies** possibly based on **role or company**.
- **Entry Level:**
  - **Consistent median low salary** with a **narrow distribution**, implying **stable starting salaries**.
- **Internship:**
  - **Lowest median low salary** with **notable outliers**, indicating that **some internships offer competitive base salaries**.

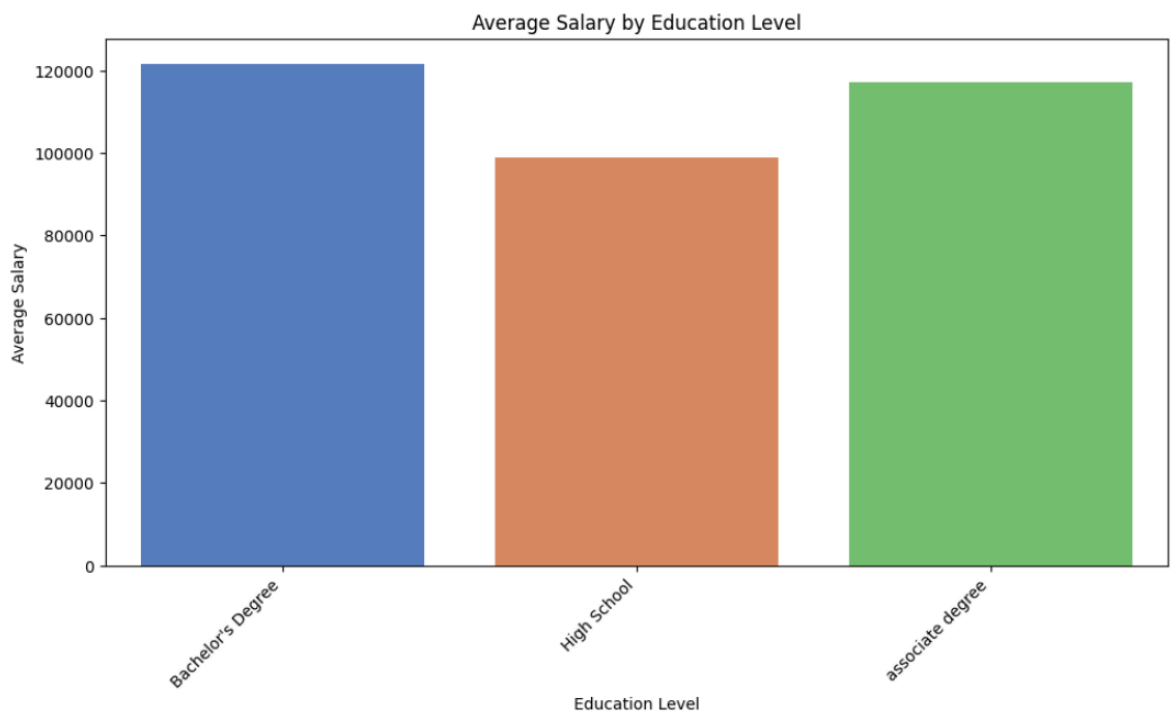
---

### Overall Key Takeaways:

1. **Employment Type Influence:**

- **Full-time roles** dominate in both **high and low salary distributions**, reflecting **stable pay structures**.
  - **Contract roles** show **greater variability**, suggesting a **diverse compensation structure** based on role complexity.
  - **Internships** can be **highly rewarding** in **specialized industries**, evidenced by **upper outliers** in high salary distributions.
2. **Seniority Level Dynamics:**
- **Executives** enjoy the **highest and most stable salaries**, indicating **fixed compensation structures** at top levels.
  - **Directors** experience **greater salary variability**, possibly influenced by **performance-based pay** or **industry factors**.
  - **Entry-level positions** are the **most stable** across both **high and low salary distributions**, reflecting **standardized entry pay**.
  - **Mid-Senior and Associate levels** show **overlaps**, indicating **potential for rapid salary growth** within these categories.
3. **Outlier Significance:**
- **High-salary outliers** in **internship and entry-level positions** suggest **premium roles** in **high-demand industries** like **tech** and **finance**.
  - **Low-salary outliers** in **contract positions** might represent **short-term projects** or **freelance gigs** with **variable compensation**.

## 9) Average salary by education level:



## 10) Word clouds from the description column to see the most frequently mentioned skills and requirements.

```
from wordcloud import WordCloud, STOPWORDS

import matplotlib.pyplot as plt

# Combine all descriptions into a single string
text = ' '.join(df['description'].astype(str).tolist())

# Create a WordCloud object
wordcloud = WordCloud(width=800, height=400, background_color='white',
                        stopwords=STOPWORDS,
                        min_font_size=10).generate(text)

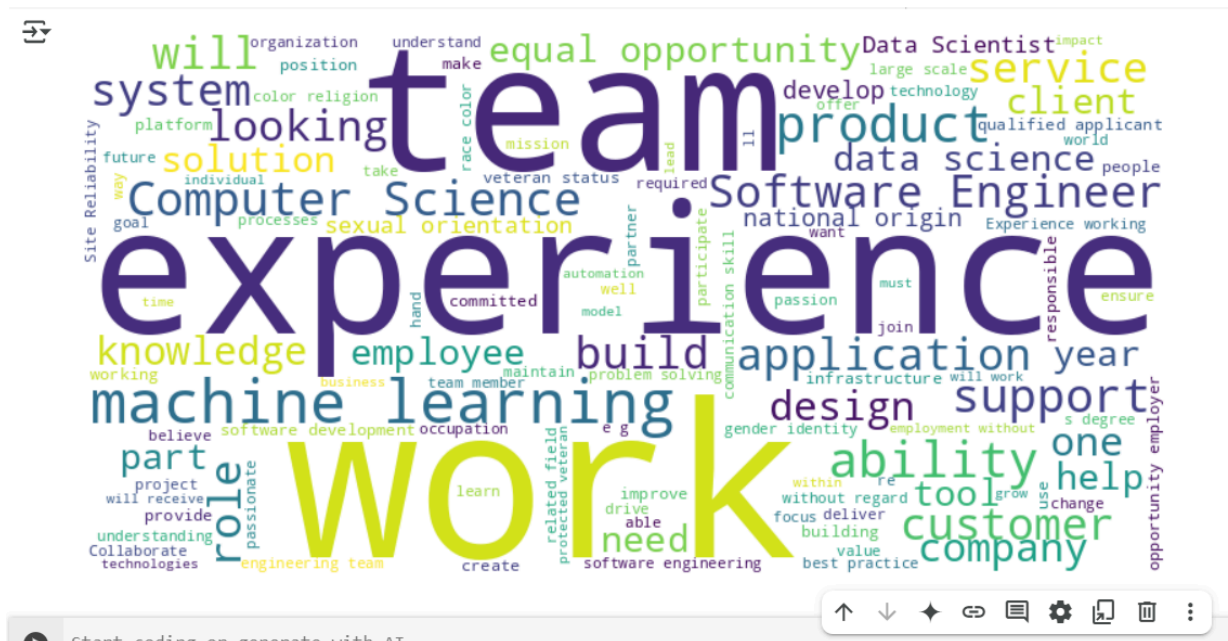
# Plot the WordCloud image
plt.figure(figsize=(8, 8), facecolor=None)

plt.imshow(wordcloud)

plt.axis("off")

plt.tight_layout(pad=0)

plt.show()
```



→ The terms which are most popular in all of the job descriptions are team, experience and work.

## 11) A word cloud from the `title` column to understand the most common job titles.

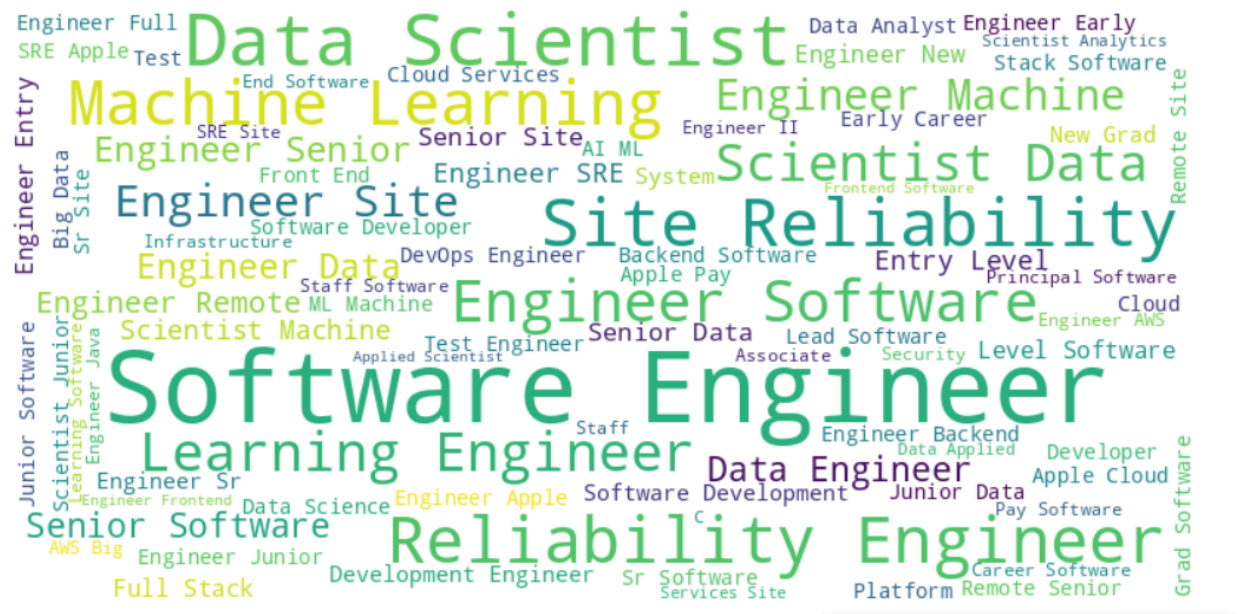
```
from wordcloud import WordCloud, STOPWORDS
import matplotlib.pyplot as plt

# Combine all titles into a single string
text = ' '.join(df['title'].astype(str).tolist())

# Create a WordCloud object
wordcloud = WordCloud(width=800, height=400, background_color='white',
                        stopwords=STOPWORDS,
                        min_font_size=10).generate(text)

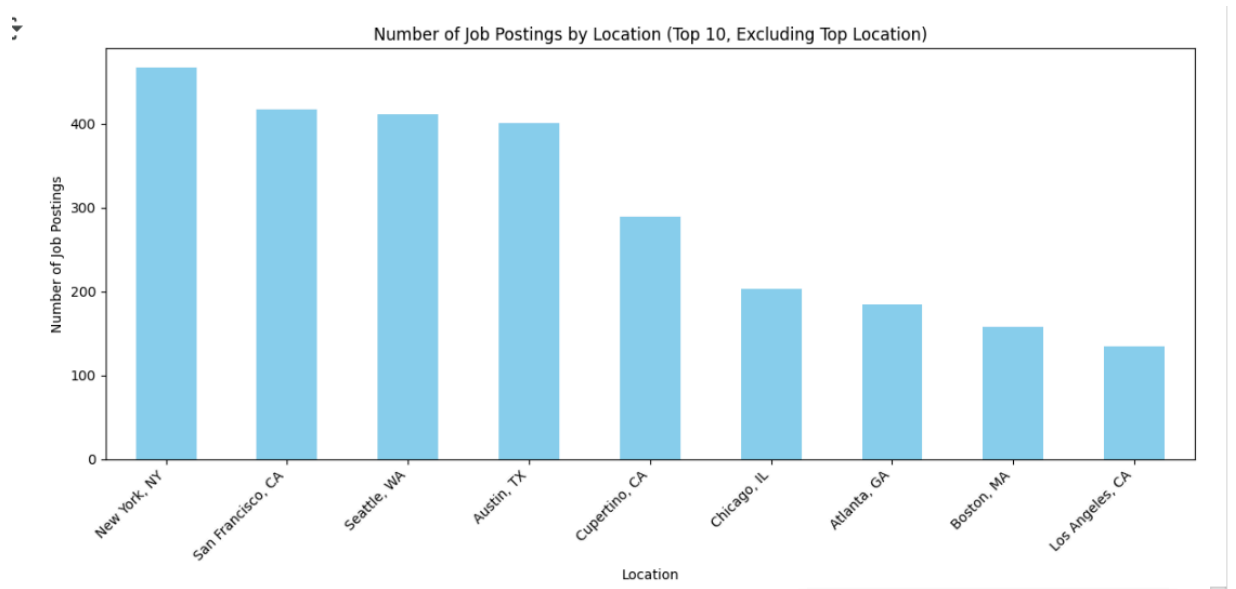
# Plot the WordCloud image
plt.figure(figsize=(8, 8), facecolor=None)
```

```
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad=0)
plt.show()
```

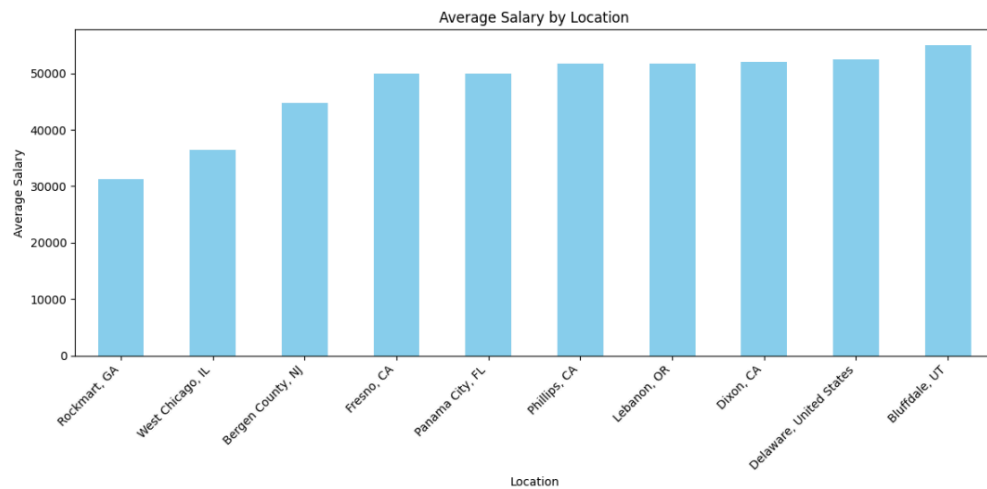


## 12) Location-Based Analysis:

- Bar Chart of Job Postings by Location:
  - Visualize the number of job postings for each location to identify areas with high job demand.

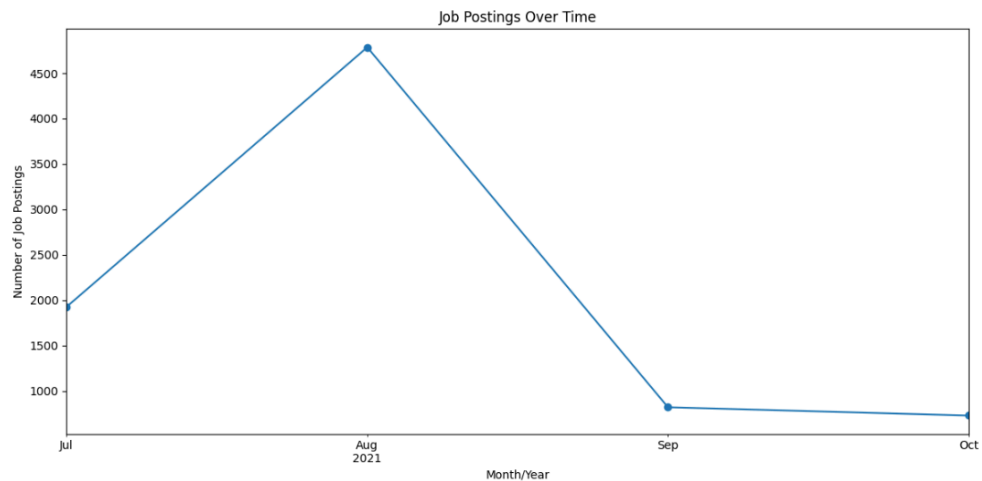


- Location vs. Average Salary:
  - Display this using a bar chart to see which locations offer the highest average salaries.





- Job posting over time



From the graph we can clearly see that the job posting is increased in month of august

### 13) Visualizations Related to Company and Industry:

- Top Companies Hiring:
  - Visualize the top companies with the most job postings.

```
company_counts = df['company'].value_counts().nlargest(10)
```

```
plt.figure(figsize=(12, 6))
```

```
company_counts.plot(kind='bar', color='skyblue')
```

```
plt.title('Top 10 Companies with the Most Job Postings')
```

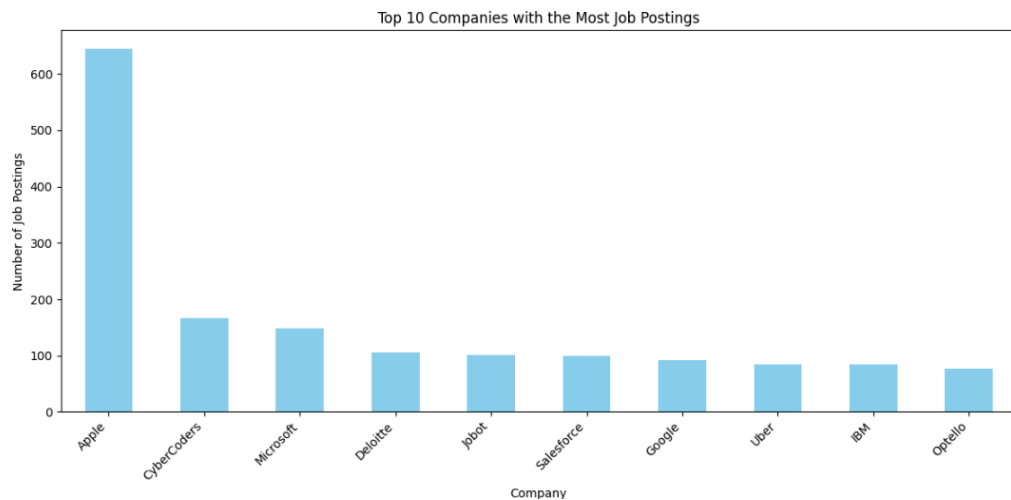
```
plt.xlabel('Company')
```

```
plt.ylabel('Number of Job Postings')

plt.xticks(rotation=45, ha='right')

plt.tight_layout()

plt.show()
```



- Industries by Seniority Level:
  - Use a grouped bar chart to show the distribution of seniority levels within each industry.

```
industry_seniority.plot(kind='barh', figsize=(12, 6)) # barh
for horizontal

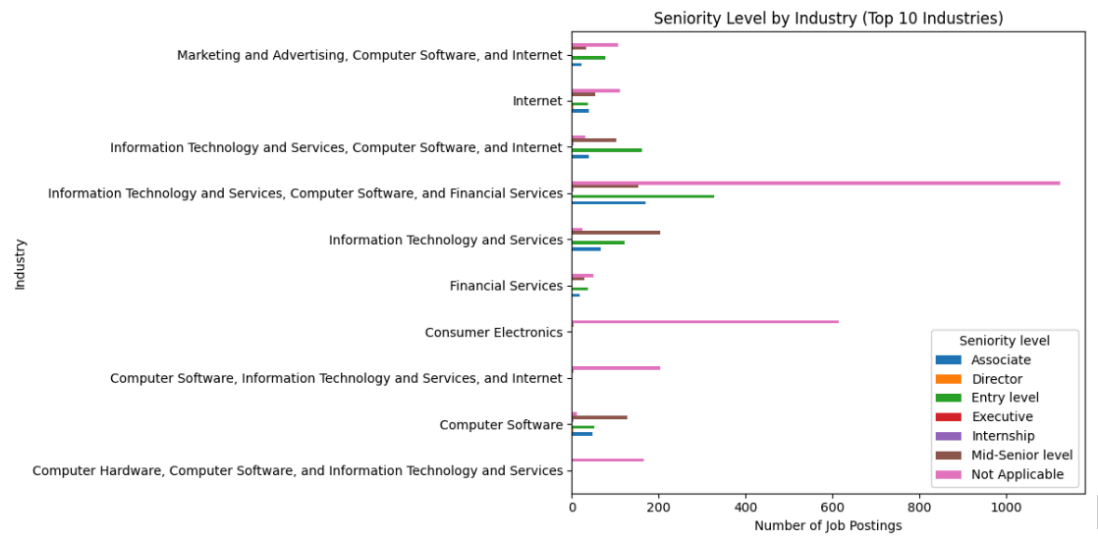
plt.title(f'Seniority Level by Industry (Top {top_n}
Industries)')

plt.ylabel('Industry') # Switch labels for horizontal chart
```

```
plt.xlabel('Number of Job Postings')
```

```
plt.tight_layout()
```

```
plt.show()
```



## 14) Count of unique values in each column

The column `post_id` tells you how many total rows there are, while the rest are indicators of how much variety is in the dataset.

