DSR Assignment 2

Project Title : LinkedIn Insights

160120771016, 160120771019

data-cleaning.R

```r
install.packages("tidyverse")
```

```
## Error in install.packages : Updating loaded packages
```

```r
install.packages("readxl")
```

```
## Error in install.packages : Updating loaded packages
```

```r
install.packages("writexl")
```

```
## Error in install.packages : Updating loaded packages
```

```r
library(tidyverse)
library(readxl)
library(writexl)

# Read the dataset
job_data <- read_excel("C:/Users/Sindhu Madhuri/Desktop/Minor Project Codes/linkedin_job_posts.xlsx")

# Replace empty cells, NULL values, and invalid values with "NA"
job_data_cleaned <- job_data %>%
  mutate(across(where(is.character), ~replace_na(., "NA"))) %>%
  mutate(across(where(is.character), ~if_else(. == "", "NA", .))) %>%
  mutate(across(where(is.logical), ~replace_na(., NA))) %>%
  mutate(across(where(is.numeric), ~replace_na(., NA))) %>%
  mutate(across(where(is.POSIXct), ~replace_na(., NA)))

# Shuffle the dataset
job_data_shuffled <- job_data_cleaned %>%
  sample_n(nrow(job_data_cleaned), replace = FALSE)

# Save the dataset as an Excel file
write_xlsx(job_data, "linkedin_job_posts_cleaned_shuffled.xlsx")

# Change the working directory
setwd("C:/Users/Sindhu Madhuri/Desktop/Minor Project Codes")

# Check the updated working directory
print(getwd())
```

```
## [1] "C:/Users/Sindhu Madhuri/Desktop/Minor Project Codes"
```

```
# Separate city and country in the location attribute
job_data_sep <- job_data %>%
  separate(location, into = c("city", "country"), sep = ",", remove = TRUE, extra = "merge", fill = "ri

# Check the updated dataset
print(job_data_sep)
```

```
## # A tibble: 9,670 x 10
##    job_title        company_name city  country hiring_status date                seniority_level
##    <chr>            <chr>        <chr> <chr>   <chr>         <dttm>              <chr>
##  1 "\r\n          ~ "\r\n      ~ "\r\~ " TX\r~ "\r\n       ~ 2023-03-02 00:00:00 "\r\n         ~
##  2 "\r\n          ~ "\r\n      ~ "\r\~ " TX\r~  <NA>         2023-02-06 00:00:00 "\r\n         ~
##  3 "\r\n          ~ "\r\n      ~ "\r\~ " TX\r~ "\r\n       ~ 2023-02-19 00:00:00 "\r\n         ~
##  4 "\r\n          ~ "\r\n      ~ "\r\~ " TX\r~ "\r\n       ~ 2023-03-07 00:00:00 "\r\n         ~
##  5 "\r\n          ~ "\r\n      ~ "\r\~ " TX\r~ "\r\n       ~ 2023-03-24 00:00:00 "\r\n         ~
##  6 "\r\n          ~ "\r\n      ~ "\r\~ " TX\r~ "\r\n       ~ 2023-03-16 00:00:00 "\r\n         ~
##  7 "\r\n          ~ "\r\n      ~ "\r\~ " TX\r~ "\r\n       ~ 2023-02-11 00:00:00 "\r\n         ~
##  8 "\r\n          ~ "\r\n      ~ "\r\~ " TX\r~  <NA>         2023-03-06 00:00:00 "\r\n         ~
##  9 "\r\n          ~ "\r\n      ~ "\r\~ " TX\r~  <NA>         2023-03-16 00:00:00 "\r\n         ~
## 10 "\r\n          ~ "\r\n      ~ "\r\~ " TX\r~ "\r\n       ~ 2023-03-24 00:00:00 "\r\n         ~
## # i 9,660 more rows
## # i 3 more variables: job_function <chr>, employment_type <chr>, industry <chr>
```

```
# Save the updated dataset as an Excel file
write_xlsx(job_data_sep, "linkedin_job_posts_cleaned_shuffled_sep.xlsx")
```

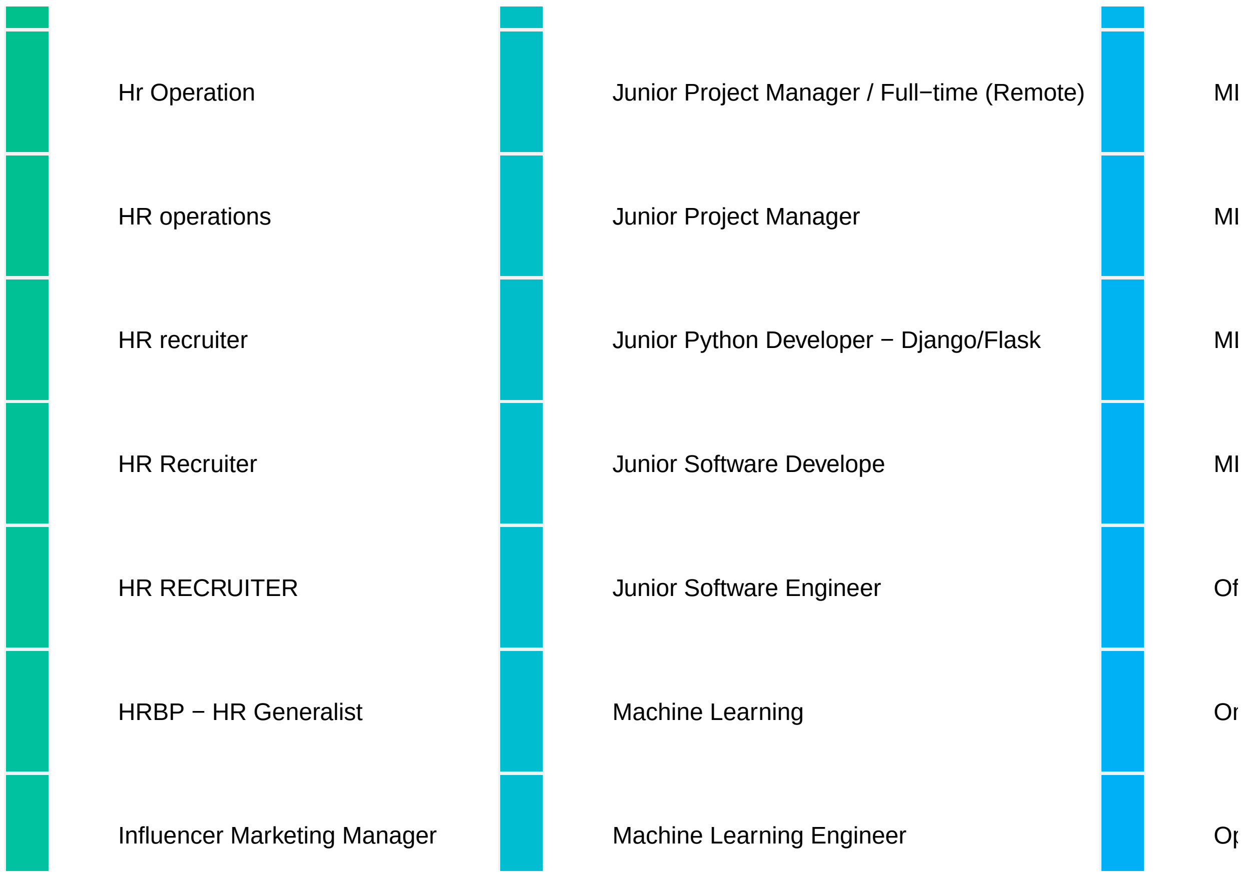
job-title-distribution.R

```r
library(tidyverse)
library(readxl)

# Read the dataset
job_data_sep <- read_excel("linkedin_job_posts_cleaned_shuffled_sep.xlsx")

# Analyze job_title and company_name
job_title_company_analysis <- job_data_sep %>%
  group_by(company_name, job_title) %>%
  summarise(job_count = n()) %>%
  arrange(desc(job_count))
```

```
## `summarise()` has grouped output by 'company_name'. You can override using the `.groups`
## argument.
```

```r
# Filter top 10 companies with the highest number of job postings
top_companies <- job_title_company_analysis %>%
  group_by(company_name) %>%
  summarise(total_job_count = sum(job_count)) %>%
  arrange(desc(total_job_count)) %>%
  head(10) %>%
  pull(company_name)

# Filter job postings for the top 10 companies
top_companies_job_titles <- job_title_company_analysis %>%
  filter(company_name %in% top_companies)

# Create a bar chart
bar_chart <- ggplot(top_companies_job_titles, aes(x = company_name, y = job_count, fill = job_title)) +
  geom_col(position = "dodge") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  labs(title = "Job Titles Distribution for Top 10 Companies",
       x = "Company Name",
       y = "Job Count")

# Show the bar chart
print(bar_chart)
```

```
## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font width unknown
## for character 0xd
## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : conversion failure on
## '"' in 'mbcsToSbcs': dot substituted for <ef>

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : conversion failure on
## '"' in 'mbcsToSbcs': dot substituted for <bb>

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : conversion failure on
## '"' in 'mbcsToSbcs': dot substituted for <bf>

## Warning in grid.Call.graphics(C_text, as.graphicsAnnot(x$label), x$x, x$y, : conversion failure
## on '"' in 'mbcsToSbcs': dot substituted for <ef>

## Warning in grid.Call.graphics(C_text, as.graphicsAnnot(x$label), x$x, x$y, : conversion failure
## on '"' in 'mbcsToSbcs': dot substituted for <bb>

## Warning in grid.Call.graphics(C_text, as.graphicsAnnot(x$label), x$x, x$y, : conversion failure
## on '"' in 'mbcsToSbcs': dot substituted for <bf>
```



job-titles-count.R

```
library(tidyverse)
library(readxl)
library(writexl)

# Read the dataset
job_data_sep <- read_excel("linkedin_job_posts_cleaned_shuffled_sep.xlsx")

# Analyze job_title and company_name
job_title_company_analysis <- job_data_sep %>%
  group_by(company_name, job_title) %>%
  summarise(job_count = n()) %>%
  arrange(desc(job_count))
```

```
## `summarise()` has grouped output by 'company_name'. You can override using the `.groups`
## argument.
```

```
# Check the analysis results
print(job_title_company_analysis)
```

```
## # A tibble: 4,912 x 3
## # Groups:   company_name [3,425]
##    company_name                           job_title                                job_count
##    <chr>                                  <chr>                                        <int>
##  1  <NA>                                  "\r\n          \r\n        Project Man~       139
##  2 "\r\n          Diverse Lynx\r\n"       "\r\n          \r\n        AWS Archite~        90
##  3 "\r\n          Diverse Lynx\r\n"       "\r\n          \r\n        Cloud Archi~        46
##  4 "\r\n          Wipro\r\n"              "\r\n          \r\n        Developer\r~        43
##  5 "\r\n          Random Bit LLC\r\n"     "\r\n          \r\n        AWS Solutio~        30
##  6 "\r\n          Wipro\r\n"              "\r\n          \r\n        Solution Ar~        30
##  7 "\r\n          Carnegie Consulting\r\n" "\r\n          \r\n        Executive A~        28
##  8 "\r\n          Hertility\r\n"          "\r\n          \r\n        Executive A~        27
##  9 "\r\n          Aurecon\r\n"            "\r\n          \r\n        Graduate Ci~        25
## 10 "\r\n          Cheil UK\r\n"           "\r\n          \r\n        Executive A~        25
## # i 4,902 more rows
```

```
# Save the analysis results as an Excel file
write_xlsx(job_title_company_analysis, "job_title_company_analysis.xlsx")
```

sentiment-analysis.R

```r
# Load libraries
library(readxl)
library(tidytext)
library(sentimentr)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(ggplot2)

# Load dataset
data <- read_excel("C:/Users/Sindhu Madhuri/Desktop/Minor Project Codes/linkedin_job_posts_cleaned_shuf

# Extract job titles and company names
titles <- data$job_title
companies <- data$company_name

# Create data frames with one row per word
title_words <- data.frame(word = unlist(strsplit(tolower(titles), "\\W+")))
company_words <- data.frame(word = unlist(strsplit(tolower(companies), "\\W+")))

# Perform sentiment analysis on job titles and company names
title_sentiment <- sentiment(title_words$word)
```

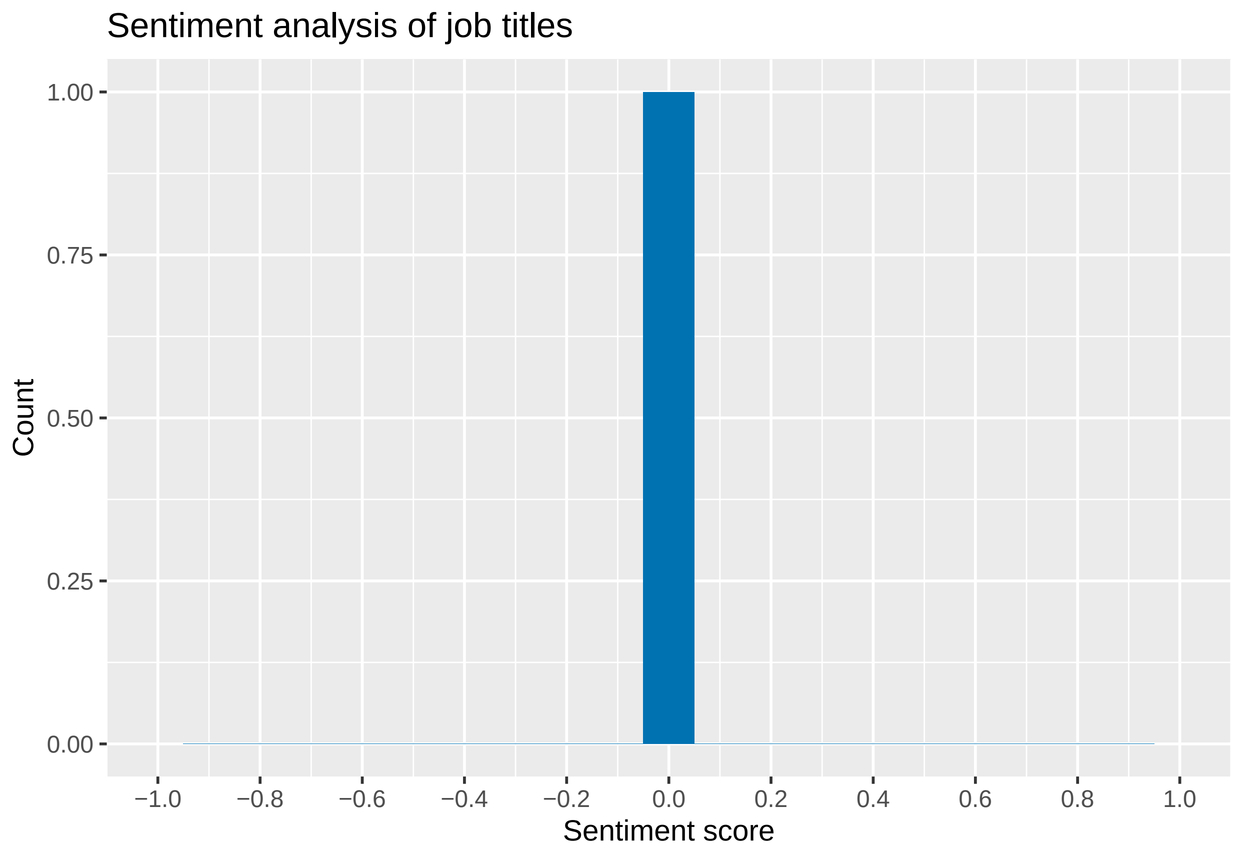
```
## Warning: Each time `sentiment` is run it has to do sentence boundary disambiguation when a
## raw `character` vector is passed to `text.var`. This may be costly of time and
## memory.  It is highly recommended that the user first runs the raw `character`
## vector through the `get_sentences` function.
```

```r
company_sentiment <- sentiment(company_words$word)
```

```
## Warning: Each time `sentiment` is run it has to do sentence boundary disambiguation when a
## raw `character` vector is passed to `text.var`. This may be costly of time and
## memory.  It is highly recommended that the user first runs the raw `character`
## vector through the `get_sentences` function.
```

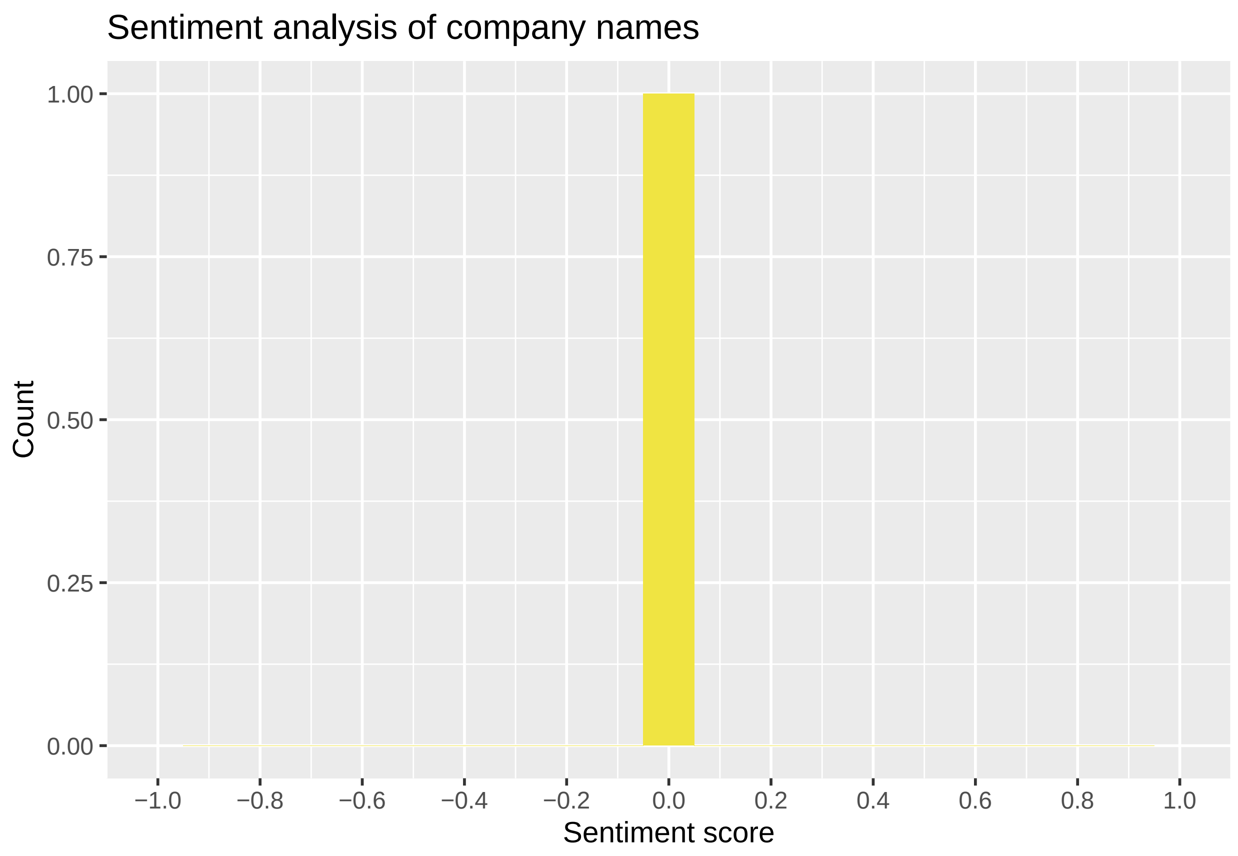
```r
# Aggregate sentiment scores by job title and company name
title_scores <- title_sentiment %>%
  group_by(element_id = sentence_id) %>%
  summarize(sentiment = mean(sentiment))

company_scores <- company_sentiment %>%
  group_by(element_id = sentence_id) %>%
  summarize(sentiment = mean(sentiment))

# Visualize the sentiment scores for job titles
ggplot(title_scores, aes(x = sentiment)) +
  geom_histogram(binwidth = 0.1, fill = "#0072B2") +
  scale_x_continuous(breaks = seq(-1, 1, 0.2), limits = c(-1, 1)) +
  labs(x = "Sentiment score", y = "Count", title = "Sentiment analysis of job titles")
```

```
## Warning: Removed 2 rows containing missing values (`geom_bar()`).
```



```r
# Visualize the sentiment scores for company names
ggplot(company_scores, aes(x = sentiment)) +
  geom_histogram(binwidth = 0.1, fill = "#F0E442") +
  scale_x_continuous(breaks = seq(-1, 1, 0.2), limits = c(-1, 1)) +
  labs(x = "Sentiment score", y = "Count", title = "Sentiment analysis of company names")
```

```
## Warning: Removed 2 rows containing missing values (‘geom_bar()‘).
```



Top 10 Job Titles offered by companies.py

`

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

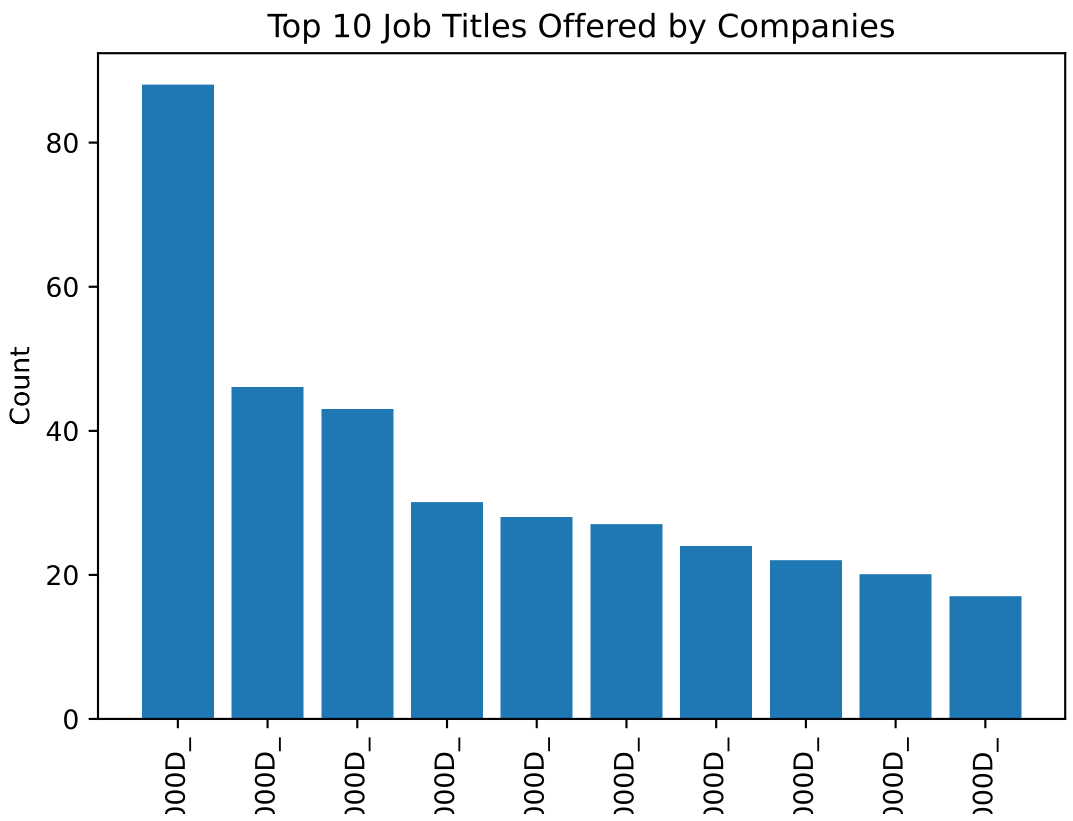
```
#!pip install pandas
#!pip install matplotlib
#!pip install openpyxl
import pandas as pd
import matplotlib.pyplot as plt

# Load the preprocessed dataset
df = pd.read_excel("C:\\Users\\student\\Downloads\\linkedin-job-posts-cleaned-shuffled-sep-preprocessed

# Group the data by company name and job title, and count the number of occurrences
counts = df.groupby(['company_name', 'job_title']).size().reset_index(name='count')

# Sort the data by count in descending order and select the top 10 rows
top_job_titles = counts.sort_values(by=['count'], ascending=False).head(10)

# Plot a bar chart of the top 10 job titles
plt.bar(top_job_titles['job_title'], top_job_titles['count'])
```

```
## <BarContainer object of 10 artists>
```

```
plt.title("Top 10 Job Titles Offered by Companies")
plt.xlabel("Job Title")
plt.ylabel("Count")
plt.xticks(rotation=90)
```

```
## ([0, 1, 2, 3, 4, 5, 6, 7, 8, 9], [Text(0, 0, 'xml:space="preserve">_x000D_ _x000D_ AWS Architect_x00
```

```
plt.show()
```


Top 10 Job Titles Offered by Companies
Count
0
20
40
60
80
)00D_
)00D_
)00D_
)00D_
)00D_
)00D_
)00D_
)00D_
)00D_
)00D_

geographic-analysis.py

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

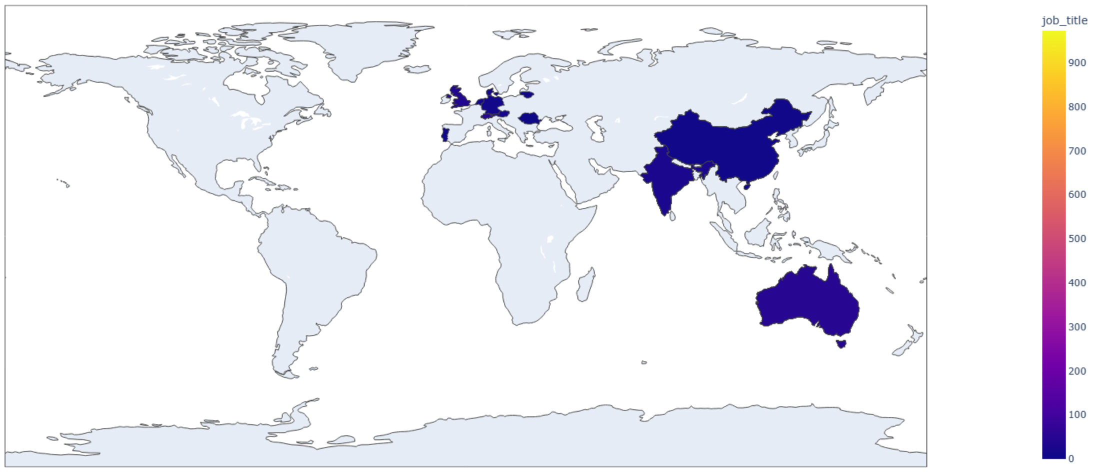
When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
import plotly.express as px
import pandas as pd

# Load the dataset into a pandas DataFrame
df = pd.read_excel("C:\\Users\\student\\Downloads\\linkedin-job-posts-cleaned-shuffled-sep-preprocessed
# Aggregate the job postings by city and country
city_counts = df.groupby('city')['job_title'].count().reset_index()
country_counts = df.groupby('country')['job_title'].count().reset_index()
# Create a choropleth map of cities
fig1 = px.choropleth(city_counts, locations='city', locationmode='country names', color='job_title',
                    hover_name='city', range_color=(0, city_counts['job_title'].max()),
                    title='Job Postings by City')
fig1.show()
# Create a choropleth map of countries
```

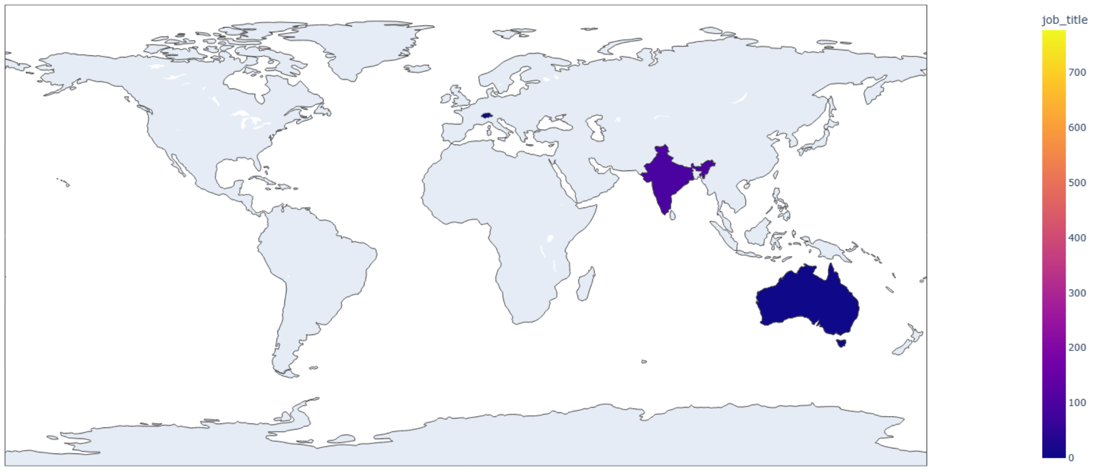
```
fig2 = px.choropleth(country_counts, locations='country', locationmode='country names', color='job_titl
                    hover_name='country', range_color=(0, country_counts['job_title'].max()),
                    title='Job Postings by Country')
fig2.show()
# Aggregate the job postings by country and job title
```

```
country_job_counts = df.groupby(['country', 'job_title'])['job_title'].count().reset_index(name='job_cou
# Create a choropleth map of countries
fig = px.choropleth(country_job_counts, locations='country', locationmode='country names', color='job_c
                    hover_name='job_title', hover_data=['job_count'],
                    range_color=(0, country_job_counts['job_count'].max()),
                    title='Job Postings by Country and Job Title')
fig.show()
```

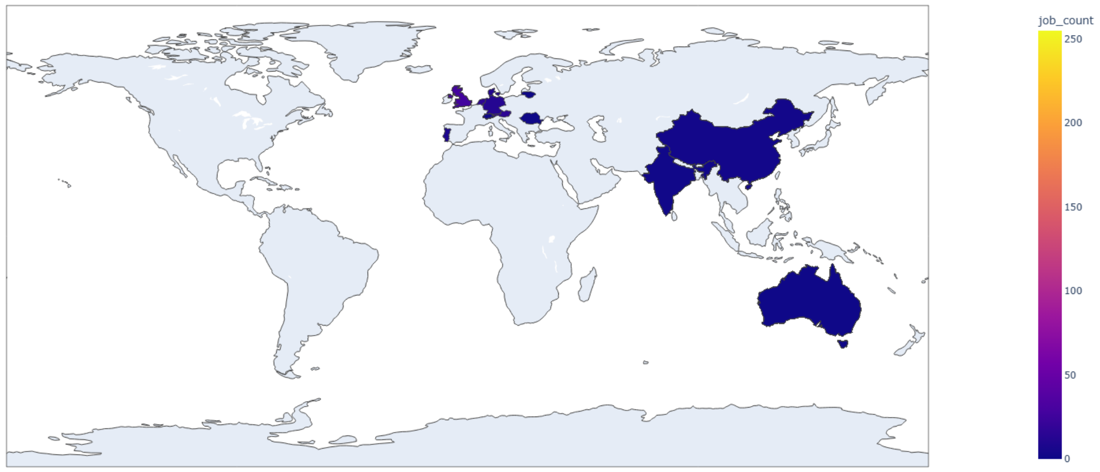
Job Postings by Country



Job Postings by City



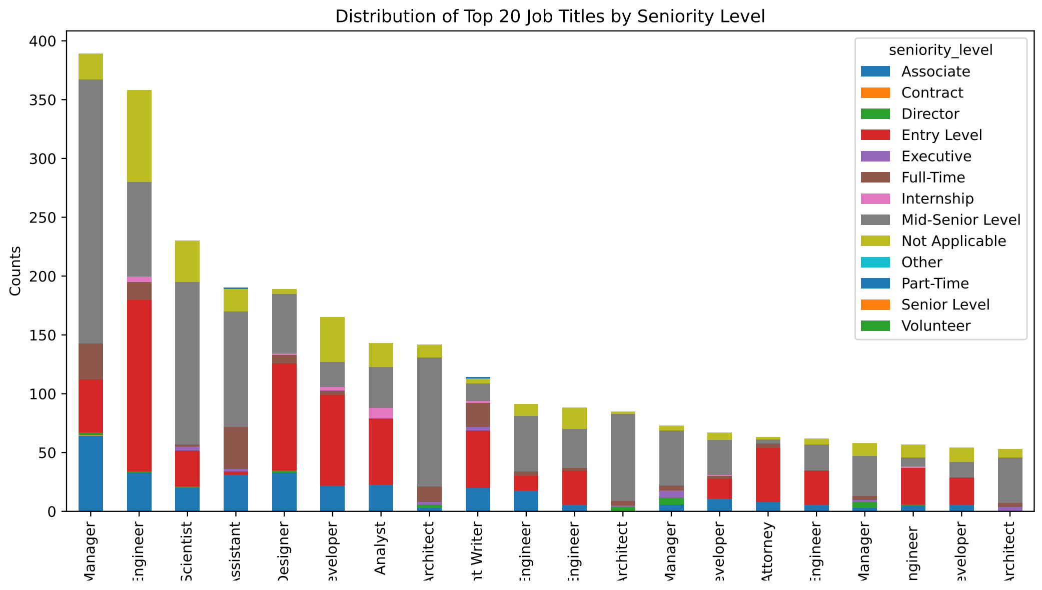
Job Postings by Country and Job Title



Assignment 2

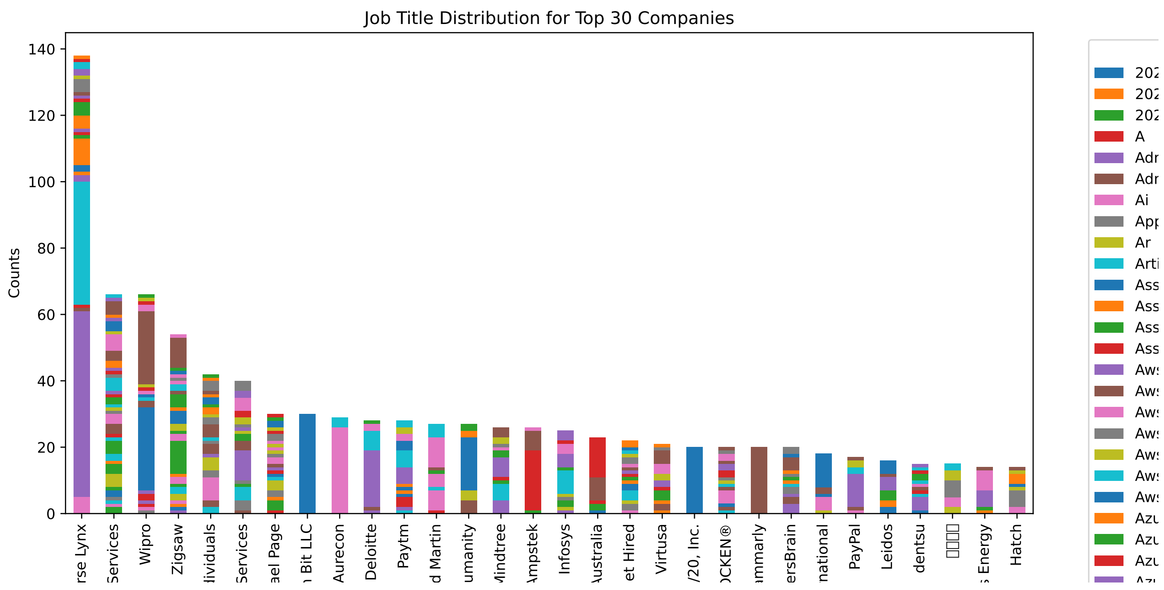
Visualization Tasks

```
import pandas as pd
import matplotlib.pyplot as plt

# Load the dataset
df = pd.read_excel('C:\\Users\\ritwi\\Downloads\\DSR\\cleaned_linkedin_job_posts.xlsx')

# Count the number of job titles by seniority level
title_seniority_counts = df.groupby(['job_title', 'seniority_level']).size().reset_index(name='counts')

# Pivot the data to create a matrix of job titles by seniority level
title_seniority_matrix = title_seniority_counts.pivot(index='job_title', columns='seniority_level', valu

# Sort the matrix by the total count of each job title
title_seniority_matrix['total'] = title_seniority_matrix.sum(axis=1)
title_seniority_matrix = title_seniority_matrix.sort_values('total', ascending=False).drop('total', axis

# Select the top 20 job titles by total count
top_20_titles = title_seniority_matrix.index[:20]
top_20_matrix = title_seniority_matrix.loc[top_20_titles]

# Plot the bar chart
top_20_matrix.plot(kind='bar', stacked=True, figsize=(12,6))
plt.title('Distribution of Top 20 Job Titles by Seniority Level')
plt.xlabel('Job Titles')
plt.ylabel('Counts')
plt.show()
```



```
# Get the top 20 companies by job count
top_companies = df['company_name'].value_counts().head(30).index

# Subset the data to only include the top companies
df = df[df['company_name'].isin(top_companies)]

# Group the data by company name and job title
company_title_counts = df.groupby(['company_name', 'job_title']).size().reset_index(name='counts')

# Pivot the data to create a matrix of job titles by company name
company_title_matrix = company_title_counts.pivot(index='company_name', columns='job_title', values='co

# Sort the matrix by the total count of each company
company_title_matrix['total'] = company_title_matrix.sum(axis=1)
company_title_matrix = company_title_matrix.sort_values('total', ascending=False).drop('total', axis=1)

# Plot the bar chart
company_title_matrix.plot(kind='bar', stacked=True, figsize=(12,6))
plt.title('Job Title Distribution for Top 30 Companies')
plt.xlabel('Companies')
plt.ylabel('Counts')
plt.legend(title='Job Titles', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.show()
```

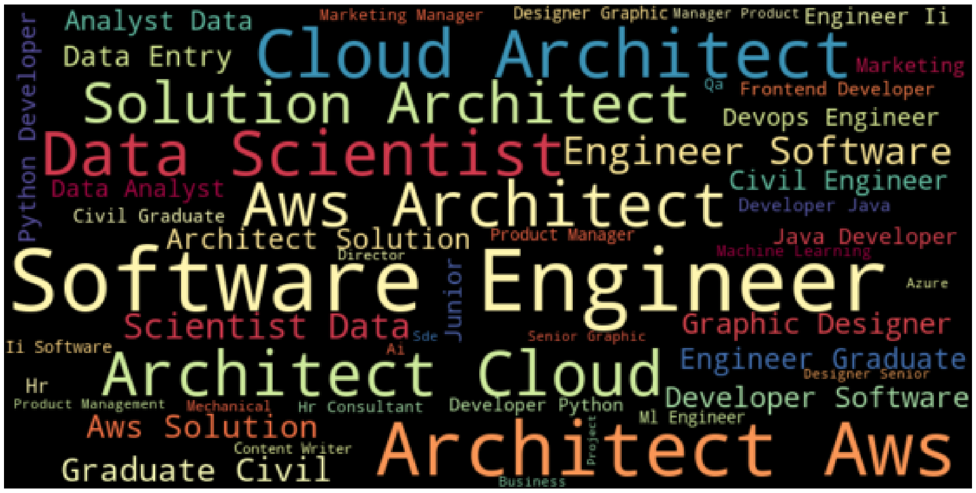
```
## C:\Users\ritwi\Documents\R\win-library\4.1\reticulate\python\rpytools\call.py:10: UserWarning: Glyph
##   value, error = rpycall.call_r_function(f, *args, **kwargs)
## C:\Users\ritwi\Documents\R\win-library\4.1\reticulate\python\rpytools\call.py:10: UserWarning: Glyph
##   value, error = rpycall.call_r_function(f, *args, **kwargs)
## C:\Users\ritwi\Documents\R\win-library\4.1\reticulate\python\rpytools\call.py:10: UserWarning: Glyph
##   value, error = rpycall.call_r_function(f, *args, **kwargs)
## C:\Users\ritwi\Documents\R\win-library\4.1\reticulate\python\rpytools\call.py:10: UserWarning: Glyph
##   value, error = rpycall.call_r_function(f, *args, **kwargs)
## C:\Users\ritwi\Documents\R\win-library\4.1\reticulate\python\rpytools\call.py:10: UserWarning: Glyph
##   value, error = rpycall.call_r_function(f, *args, **kwargs)
## C:\Users\ritwi\Documents\R\win-library\4.1\reticulate\python\rpytools\call.py:10: UserWarning: Glyph
##   value, error = rpycall.call_r_function(f, *args, **kwargs)
## C:\Users\ritwi\Documents\R\win-library\4.1\reticulate\python\rpytools\call.py:10: UserWarning: Glyph
##   value, error = rpycall.call_r_function(f, *args, **kwargs)
## C:\Users\ritwi\Documents\R\win-library\4.1\reticulate\python\rpytools\call.py:10: UserWarning: Glyph
##   value, error = rpycall.call_r_function(f, *args, **kwargs)
## C:\Users\ritwi\Documents\R\win-library\4.1\reticulate\python\rpytools\call.py:10: UserWarning: Glyph
##   value, error = rpycall.call_r_function(f, *args, **kwargs)
## C:\Users\ritwi\Documents\R\win-library\4.1\reticulate\python\rpytools\call.py:10: UserWarning: Glyph
##   value, error = rpycall.call_r_function(f, *args, **kwargs)
## C:\Users\ritwi\Documents\R\win-library\4.1\reticulate\python\rpytools\call.py:10: UserWarning: Glyph
##   value, error = rpycall.call_r_function(f, *args, **kwargs)
## C:\Users\ritwi\Documents\R\win-library\4.1\reticulate\python\rpytools\call.py:10: UserWarning: Glyph
##   value, error = rpycall.call_r_function(f, *args, **kwargs)
## C:\Users\ritwi\Documents\R\win-library\4.1\reticulate\python\rpytools\call.py:10: UserWarning: Glyph
##   value, error = rpycall.call_r_function(f, *args, **kwargs)
## C:\Users\ritwi\Documents\R\win-library\4.1\reticulate\python\rpytools\call.py:10: UserWarning: Glyph
##   value, error = rpycall.call_r_function(f, *args, **kwargs)
## C:\Users\ritwi\Documents\R\win-library\4.1\reticulate\python\rpytools\call.py:10: UserWarning: Glyph
##   value, error = rpycall.call_r_function(f, *args, **kwargs)
## C:\Users\ritwi\Documents\R\win-library\4.1\reticulate\python\rpytools\call.py:10: UserWarning: Glyph
##   value, error = rpycall.call_r_function(f, *args, **kwargs)
## C:\Users\ritwi\Documents\R\win-library\4.1\reticulate\python\rpytools\call.py:10: UserWarning: Glyph
##   value, error = rpycall.call_r_function(f, *args, **kwargs)
```



```python
from wordcloud import WordCloud

# Concatenate all job titles into a single string
job_titles = ' '.join(df['job_title'].astype(str))

# Generate a word cloud
wordcloud = WordCloud(width=800, height=400, background_color='black', colormap='Spectral_r', max_words
# Plot the word cloud
plt.figure(figsize=(10,6))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
```

```
## (-0.5, 799.5, 399.5, -0.5)
```

```
plt.show()
```



```
import pandas as pd
from datetime import datetime

# Convert the date column to datetime format
df['date'] = pd.to_datetime(df['date'])

# Extract the month from the date column and create a new column for it
df['month'] = df['date'].apply(lambda x: datetime.strftime(x, '%B') if not pd.isnull(x) else '')

# Print the first few rows of the updated dataset
print(df.head())
```

```
##             job_title  company_name  ...              industry     month
## 19    Product Manager  Diverse Lynx  ...  Software Development  February
## 23    Product Manager  Diverse Lynx  ...  Software Development     March
## 25    Product Manager  Diverse Lynx  ...  Software Development     March
## 30    Product Manager  Diverse Lynx  ...  Software Development     March
## 31   Business Analyst  Diverse Lynx  ...  Software Development     March
```

```
##
## [5 rows x 10 columns]
```

```
# Convert the date column to datetime format
df['date'] = pd.to_datetime(df['date'])

# Extract the month from the date column and create a new column for it
df['month'] = df['date'].apply(lambda x: datetime.strftime(x, '%B') if not pd.isnull(x) else '')

# Get the top 10 job titles by count
top_job_titles = df['job_title'].value_counts().nlargest(10).index.tolist()

# Filter the data to only include the top 10 job titles
df_top_jobs = df[df['job_title'].isin(top_job_titles)]

# Group the data by job title and month and count the number of occurrences
title_month_counts = df_top_jobs.groupby(['job_title', 'month']).size().reset_index(name='counts')

# Pivot the data to create a matrix with job titles as rows and months as columns
title_month_matrix = title_month_counts.pivot(index='job_title', columns='month', values='counts').fill

# Plot a heatmap of the matrix
plt.figure(figsize=(10,10))
plt.imshow(title_month_matrix, cmap='Reds')
plt.xticks(range(len(title_month_matrix.columns)), title_month_matrix.columns, rotation=90)
```

```
## ([<matplotlib.axis.XTick object at 0x000000004AD00F40>, <matplotlib.axis.XTick object at 0x000000004
```

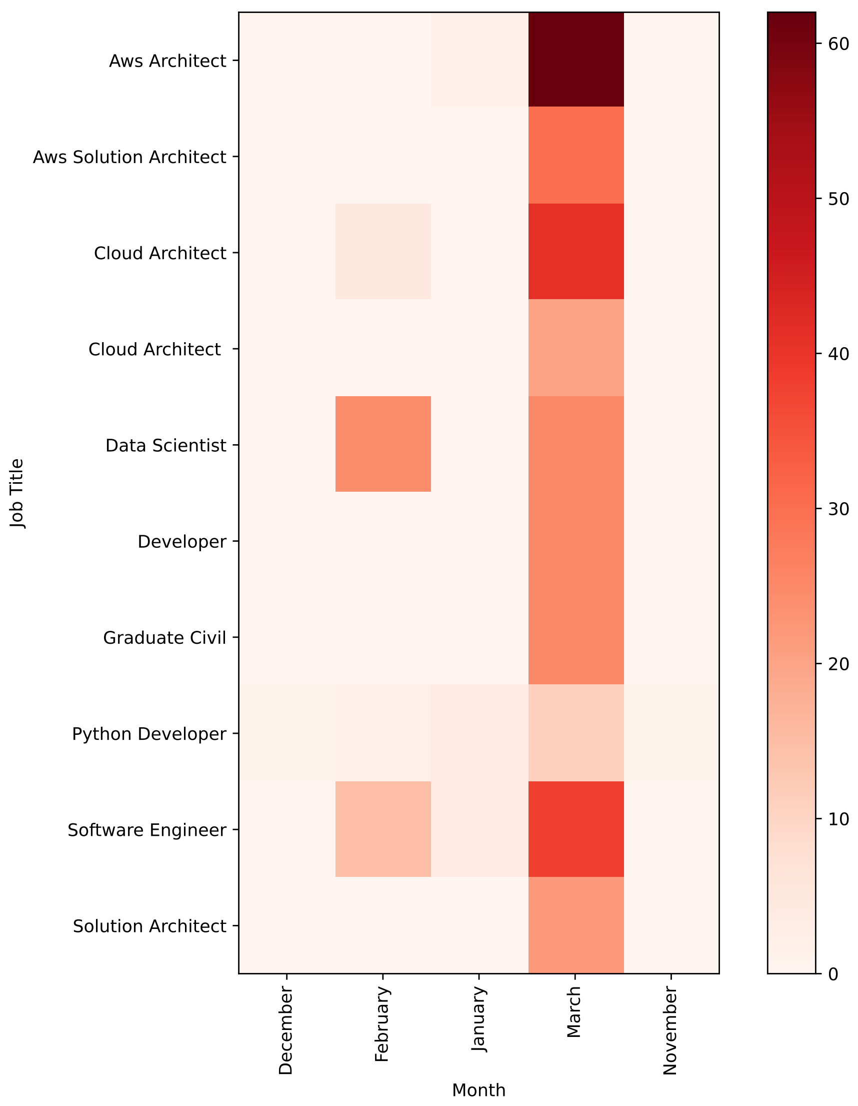
```
plt.yticks(range(len(title_month_matrix.index)), title_month_matrix.index)
```

```
## ([<matplotlib.axis.YTick object at 0x000000004AD00460>, <matplotlib.axis.YTick object at 0x000000004
```

```
plt.xlabel('Month')
plt.ylabel('Job Title')
plt.colorbar()
```

```
## <matplotlib.colorbar.Colorbar object at 0x000000004C90CD30>
```

```
plt.show()
```

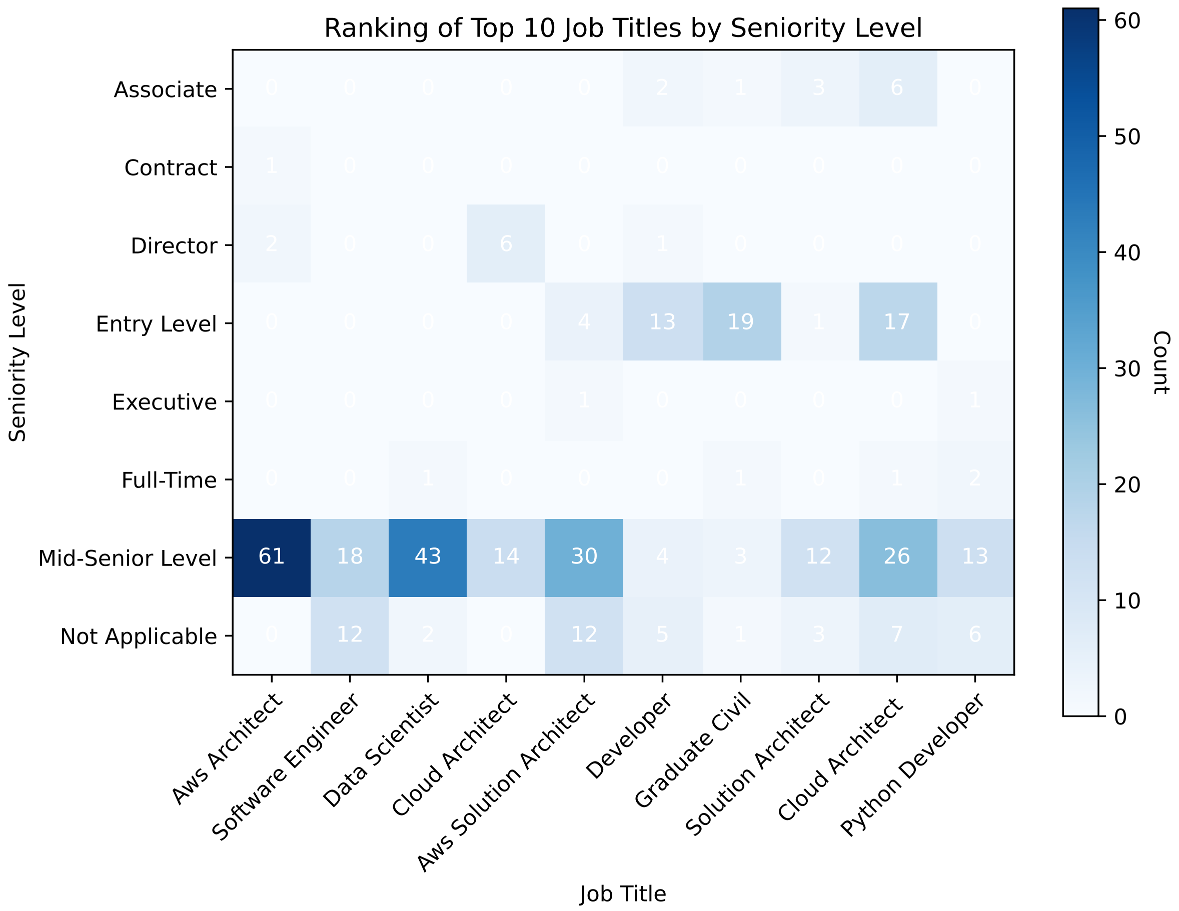


```
import numpy as np
# Compute the top 10 job titles by count
job_counts = df['job_title'].value_counts().nlargest(10).index.tolist()

# Filter the dataset to keep only the top 10 job titles
df_top10 = df[df['job_title'].isin(job_counts)]

# Compute the count of job titles by seniority level
counts = df_top10.groupby(['seniority_level', 'job_title']).size().reset_index(name='count')

# Compute the rank of job titles by seniority level
counts['rank'] = counts.groupby('seniority_level')['count'].rank(method='dense', ascending=False)

# Pivot the data to create a matrix with seniority levels as rows and job titles as columns
```

```python
pivot = counts.pivot(index='seniority_level', columns='job_title', values='count').fillna(0)

# Create a heatmap of the data
fig, ax = plt.subplots(figsize=(8, 6))
im = ax.imshow(pivot, cmap='Blues')

# Add annotations of the counts
for i in range(len(job_counts)):
    for j in range(len(pivot.index)):
        text = ax.text(i, j, int(pivot.iloc[j, i]), ha='center', va='center', color='w')

# Add labels and title
ax.set_xticks(np.arange(len(job_counts)))
ax.set_yticks(np.arange(len(pivot.index)))
ax.set_xticklabels(job_counts)
ax.set_yticklabels(pivot.index)
ax.set_xlabel('Job Title')
ax.set_ylabel('Seniority Level')
ax.set_title('Ranking of Top 10 Job Titles by Seniority Level')

# Rotate the x-axis labels
plt.setp(ax.get_xticklabels(), rotation=45, ha='right', rotation_mode='anchor')

# Add a colorbar
```

```
## [None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, Non
```

```python
cbar = ax.figure.colorbar(im, ax=ax)
cbar.ax.set_ylabel('Count', rotation=-90, va='bottom')

plt.tight_layout()
plt.show()
```



Seniority Level	Aws Architect	Software Engineer	Data Scientist	Cloud Architect	Aws Solution Architect	Developer	Graduate Civil	Solution Architect	Cloud Architect	Python Developer
Associate	0	0	0	0	0	2	1	3	6	0
Contract	1	0	0	0	0	0	0	0	0	0
Director	2	0	0	6	0	1	0	0	0	0
Entry Level	0	0	0	0	4	13	19	1	17	0
Executive	0	0	0	0	1	0	0	0	0	1
Full-Time	0	0	1	0	0	0	1	0	1	2
Mid-Senior Level	61	18	43	14	30	4	3	12	26	13
Not Applicable	0	12	2	0	12	5	1	3	7	6