Abdella Abdella Cambridge Spark Bootcamp Project Description

Project Title: Exploratory Analysis of Science Salaries

Description: This project aims to perform an exploratory analysis of science salaries using data obtained from Kaggle. The dataset was collected on 02/03/2024 and provides insights into the salaries of professionals in the field of data science. The dataset is sourced from Kaggle and can be accessed through the following link: Science Salaries Dataset.

Cambridge Spark, a prominent education institution specializing in data science and machine learning bootcamps, undertook this project to understand the trends and factors influencing salaries in the science domain. The dataset offers a variety of features such as job titles, years of experience, educational qualifications, and salary information. Through this analysis, we aim to uncover patterns, correlations, and insights that can provide valuable information to professionals, job seekers, and employers in the science and data science sectors.

The analysis will involve data cleaning, exploratory data analysis (EDA), visualization, and possibly machine learning modeling to predict salary trends based on various factors. By leveraging Python programming and data analysis libraries such as Pandas, NumPy, Matplotlib, and Seaborn, we will delve into the dataset to extract meaningful insights and present them in a clear and interpretable manner.

This project not only serves as a learning experience for participants of the Cambridge Spark bootcamp but also contributes to the broader understanding of salary dynamics within the science and data science domains. The findings and conclusions drawn from this analysis can potentially inform hiring practices, salary negotiations, and career decisions within the industry.

Data-Science-Salaries-2023

Salaries of Different Data Science Fields in the Data Science Domain

In [43]:
```
# Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker
import plotly.express as px
import seaborn as sns
```

In [44]:
```
# Load the dataset.
df = pd.read_csv("ds_salaries.csv")
```

In [45]:
```
df
```

Out[45]:

	id	work_year	experience_level	employment_type	job_title	salary	salary_currency	salary_in_usd	employee_residence	remote_ratio	company_location	company_size
0	0	2020	MI	FT	Data Scientist	70000	EUR	79833	DE	0	DE	L
1	1	2020	SE	FT	Machine Learning Scientist	260000	USD	260000	JP	0	JP	S
2	2	2020	SE	FT	Big Data Engineer	85000	GBP	109024	GB	50	GB	M
3	3	2020	MI	FT	Product Data Analyst	20000	USD	20000	HN	0	HN	S
4	4	2020	SE	FT	Machine Learning Engineer	150000	USD	150000	US	50	US	L
...	...	...	...	...	...	...	...	...	...	...	...	...
602	602	2022	SE	FT	Data Engineer	154000	USD	154000	US	100	US	M
603	603	2022	SE	FT	Data Engineer	126000	USD	126000	US	100	US	M
604	604	2022	SE	FT	Data Analyst	129000	USD	129000	US	0	US	M
605	605	2022	SE	FT	Data Analyst	150000	USD	150000	US	100	US	M
606	606	2022	MI	FT	AI Scientist	200000	USD	200000	IN	100	US	L

607 rows × 12 columns

In [7]:
```
# Display the statistical summary of the DataFrame.
df.describe()
```

Out[7]:

	id	work_year	salary	salary_in_usd	remote_ratio
count	607.000000	607.000000	6.070000e+02	607.000000	607.00000
mean	303.000000	2021.405272	3.240001e+05	112297.869852	70.92257
std	175.370085	0.692133	1.544357e+06	70957.259411	40.70913
min	0.000000	2020.000000	4.000000e+03	2859.000000	0.00000
25%	151.500000	2021.000000	7.000000e+04	62726.000000	50.00000
50%	303.000000	2022.000000	1.150000e+05	101570.000000	100.00000
75%	454.500000	2022.000000	1.650000e+05	150000.000000	100.00000
max	606.000000	2022.000000	3.040000e+07	600000.000000	100.00000

In [8]:
```
# Display information about the DataFrame.
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 607 entries, 0 to 606
Data columns (total 12 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   id                  607 non-null    int64
 1   work_year           607 non-null    int64
 2   experience_level    607 non-null    object
 3   employment_type     607 non-null    object
 4   job_title           607 non-null    object
 5   salary              607 non-null    int64
 6   salary_currency     607 non-null    object
 7   salary_in_usd       607 non-null    int64
 8   employee_residence  607 non-null    object
 9   remote_ratio        607 non-null    int64
 10  company_location    607 non-null    object
 11  company_size        607 non-null    object
dtypes: int64(5), object(7)
memory usage: 57.0+ KB
```

In [9]:
```
# Display the number of unique values for each column.
df.nunique()
```

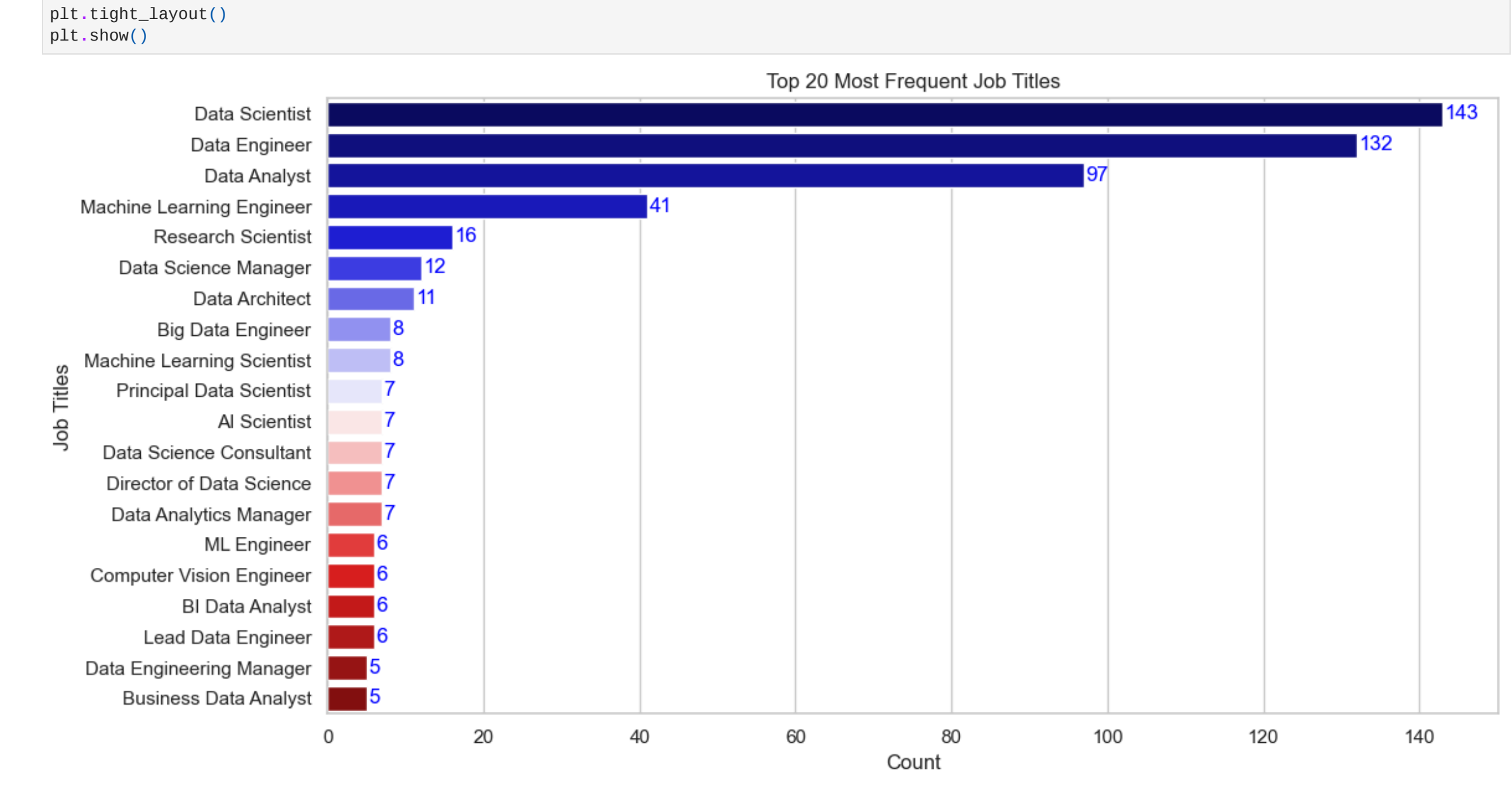
Out[9]:
```
id                    607
work_year               3
experience_level        4
employment_type         4
job_title              50
salary                272
salary_currency        17
salary_in_usd         369
employee_residence     57
remote_ratio            3
company_location       50
company_size            3
dtype: int64
```

In [10]:
```
# Transformation of the codes of the categorical variables

df['experience_level'] = data['experience_level'].replace({'SE': 'Expert', 'MI': 'Intermediate', 'EN': 'Junior', 'EX': 'Director'})

df['employment_type'] = data['employment_type'].replace({'FT': 'Full-time', 'CT': 'Contract', 'FL': 'Freelance', 'PT': 'Part-time'})

def country_name(country_code):
    try:
        return pycountry.countries.get(alpha_2=country_code).name
    except:
        return 'other'

df['company_location'] = data['company_location'].apply(country_name)
df['employee_residence'] = data['employee_residence'].apply(country_name)
```

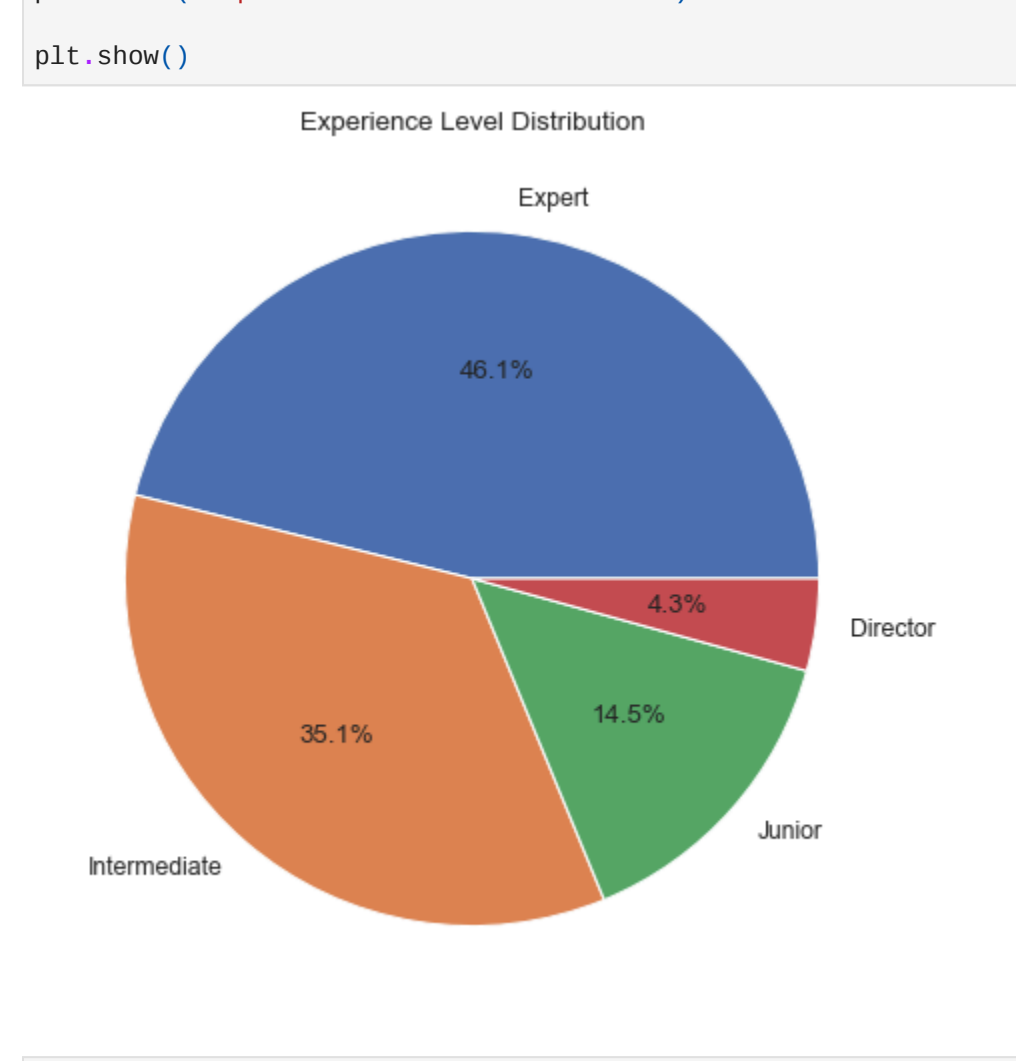
In [14]:
```
# Categorical variables

for column in ['work_year','experience_level','employment_type','company_size','remote_ratio','job_title','company_location']:
    print(df[column].unique())
```

```
[2020 2021 2022]
['Intermediate' 'Expert' 'Junior' 'Director']
['Full-time' 'Contract' 'Part-time' 'Freelance']
['L' 'S' 'M']
[  0  50 100]
['Data Scientist' 'Machine Learning Scientist' 'Big Data Engineer'
 'Product Data Analyst' 'Machine Learning Engineer' 'Data Analyst'
 'Lead Data Scientist' 'Business Data Analyst' 'Lead Data Engineer'
 'Lead Data Analyst' 'Data Engineer' 'Data Science Consultant'
 'BI Data Analyst' 'Director of Data Science' 'Research Scientist'
 'Machine Learning Manager' 'Data Engineering Manager'
 'Machine Learning Infrastructure Engineer' 'ML Engineer' 'AI Scientist'
 'Computer Vision Engineer' 'Principal Data Scientist'
 'Data Science Manager' 'Head of Data' '3D Computer Vision Researcher'
 'Data Analytics Engineer' 'Applied Data Scientist'
 'Marketing Data Analyst' 'Cloud Data Engineer' 'Financial Data Analyst'
 'Computer Vision Software Engineer' 'Director of Data Engineering'
 'Data Science Engineer' 'Principal Data Engineer'
 'Machine Learning Developer' 'Applied Machine Learning Scientist'
 'Data Analytics Manager' 'Head of Data Science' 'Data Specialist'
 'Data Architect' 'Finance Data Analyst' 'Principal Data Analyst'
 'Big Data Architect' 'Staff Data Scientist' 'Analytics Engineer'
 'ETL Developer' 'Head of Machine Learning' 'NLP Engineer'
 'Lead Machine Learning Engineer' 'Data Analytics Lead']
['other']
```

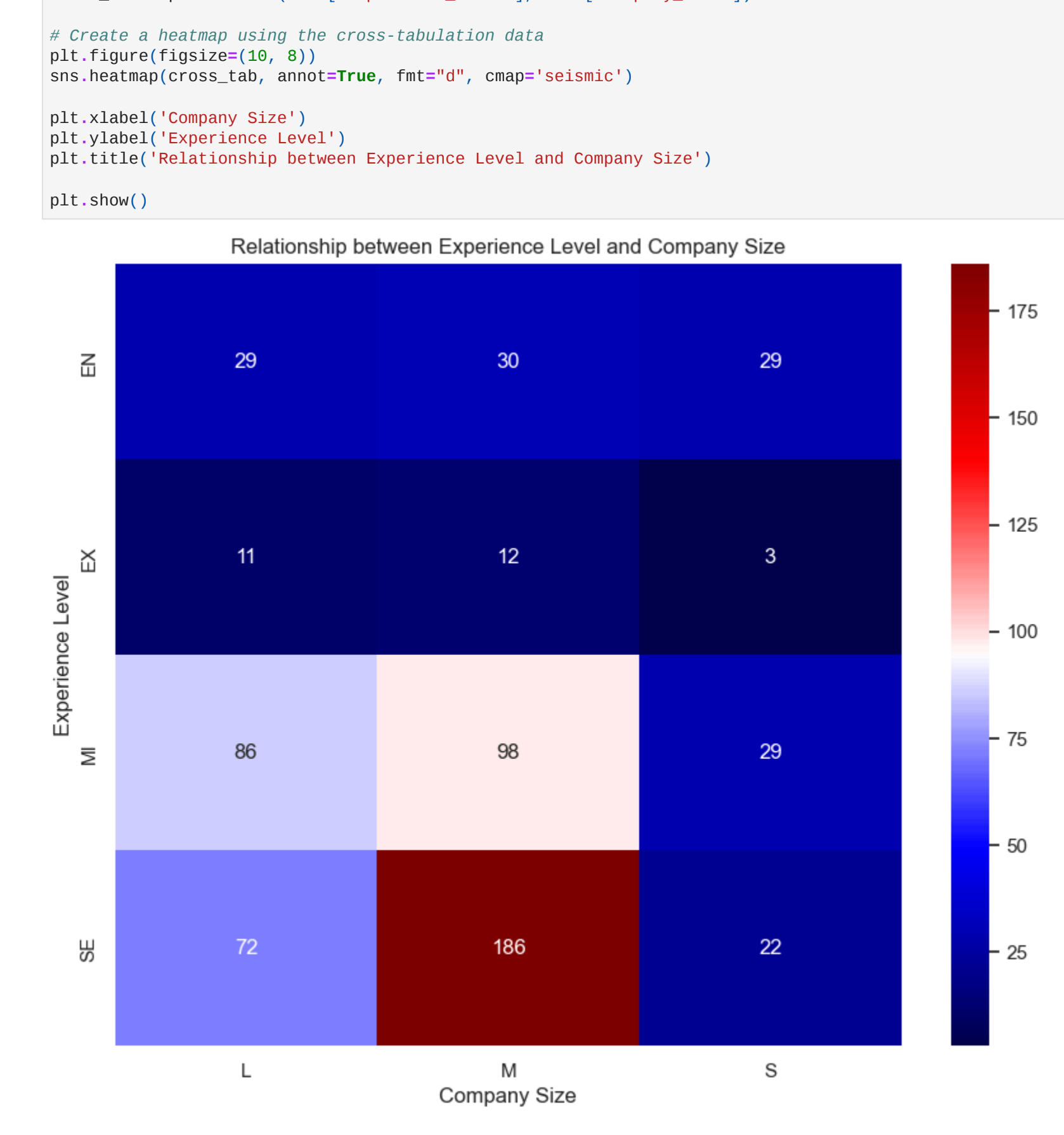
In [28]:
```
# Extract the "job title" column
job_titles = df['job_title']

# Calculate the frequency of each job title
title_counts = job_titles.value_counts()

# Extract the top 20 most frequent job titles
top_20_titles = title_counts.head(20)

# Create a DataFrame for the top 20 titles
top_20_df = pd.DataFrame({'Job Title': top_20_titles.index, 'Count': top_20_titles.values})

# Plotting the count plot
plt.figure(figsize=(12, 6))
sns.set(style="whitegrid")
ax = sns.barplot(data=top_20_df, x='Count', y='Job Title', palette='seismic')
plt.xlabel('Count')
plt.ylabel('Job Titles')
plt.title('Top 20 Most Frequent Job Titles')

# Add count labels to the bars
for i, v in enumerate(top_20_df['Count']):
    ax.text(v + 0.2, i, str(v), color='blue', va='center')

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



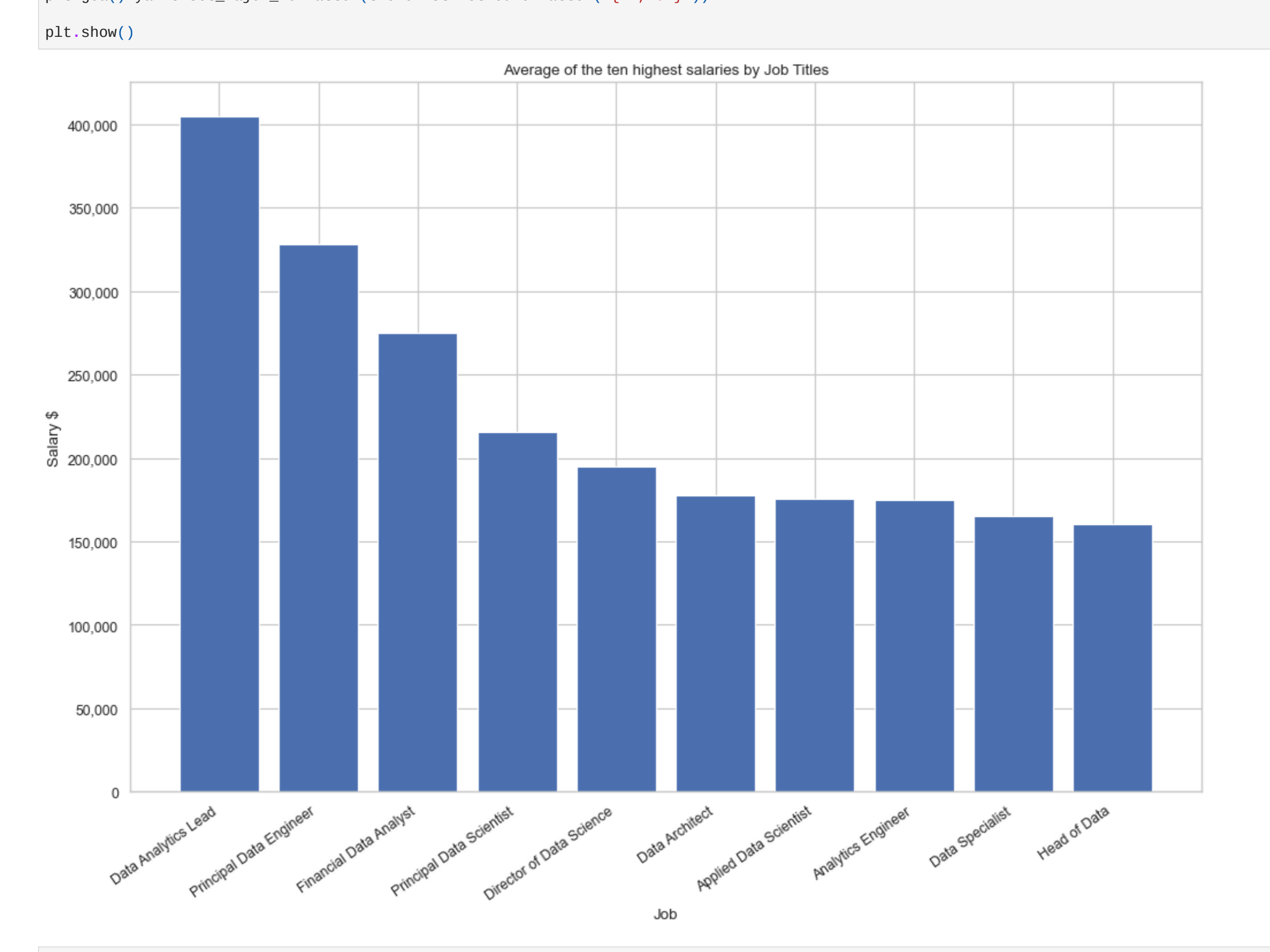
In [26]:
```
# Calculate the number of individuals in each experience level
level_counts = df['experience_level'].value_counts()

# Create a pie chart
plt.figure(figsize=(7,12),dpi=80)
plt.pie(level_counts.values, labels=level_counts.index, autopct='%1.1f%%')
plt.title('Experience Level Distribution')

plt.show()
```



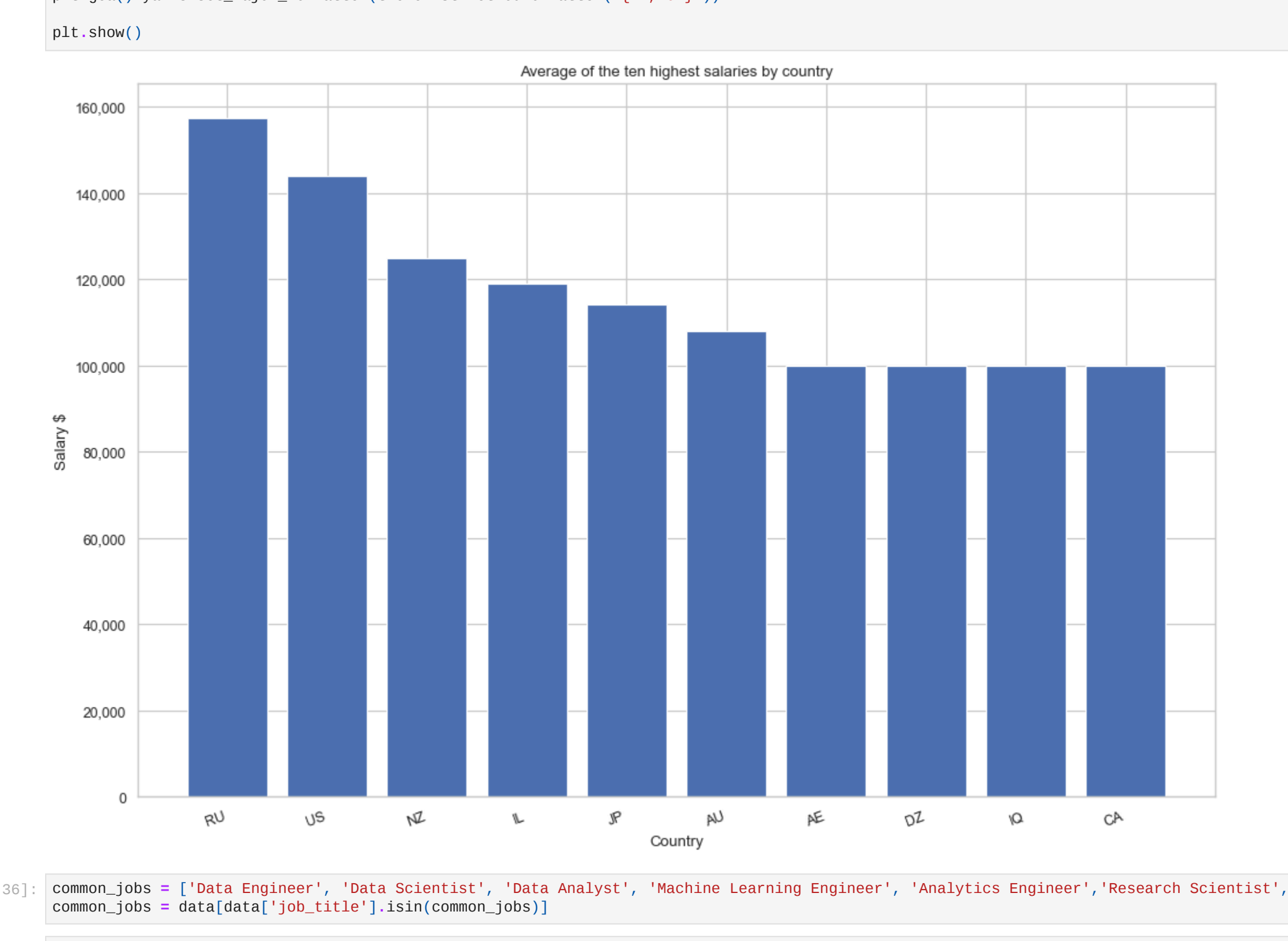
In [27]:
```
# Create a cross-tabulation of the two columns
cross_tab = pd.crosstab(data['experience_level'], data['company_size'])

# Create a heatmap using the cross-tabulation data
plt.figure(figsize=(10, 8))
sns.heatmap(cross_tab, annot=True, fmt="d", cmap='seismic')

plt.xlabel('Company Size')
plt.ylabel('Experience Level')
plt.title('Relationship between Experience Level and Company Size')

plt.show()
```



In [33]:
```
# Create bar chart
average_salary = data.groupby('job_title')['salary_in_usd'].mean().sort_values(ascending=False)
top_ten_salaries = average_salary.head(10)

plt.figure(figsize=(15,10),dpi=80)
plt.bar(top_ten_salaries.index, top_ten_salaries)

# Add labels to the chart
plt.xlabel('Job')
plt.ylabel('Salary $')
plt.title('Average of the ten highest salaries by Job Titles')
plt.xticks(rotation=35, ha='right')
plt.gca().yaxis.set_major_formatter(ticker.StrMethodFormatter('{x:,.0f}'))

plt.show()
```



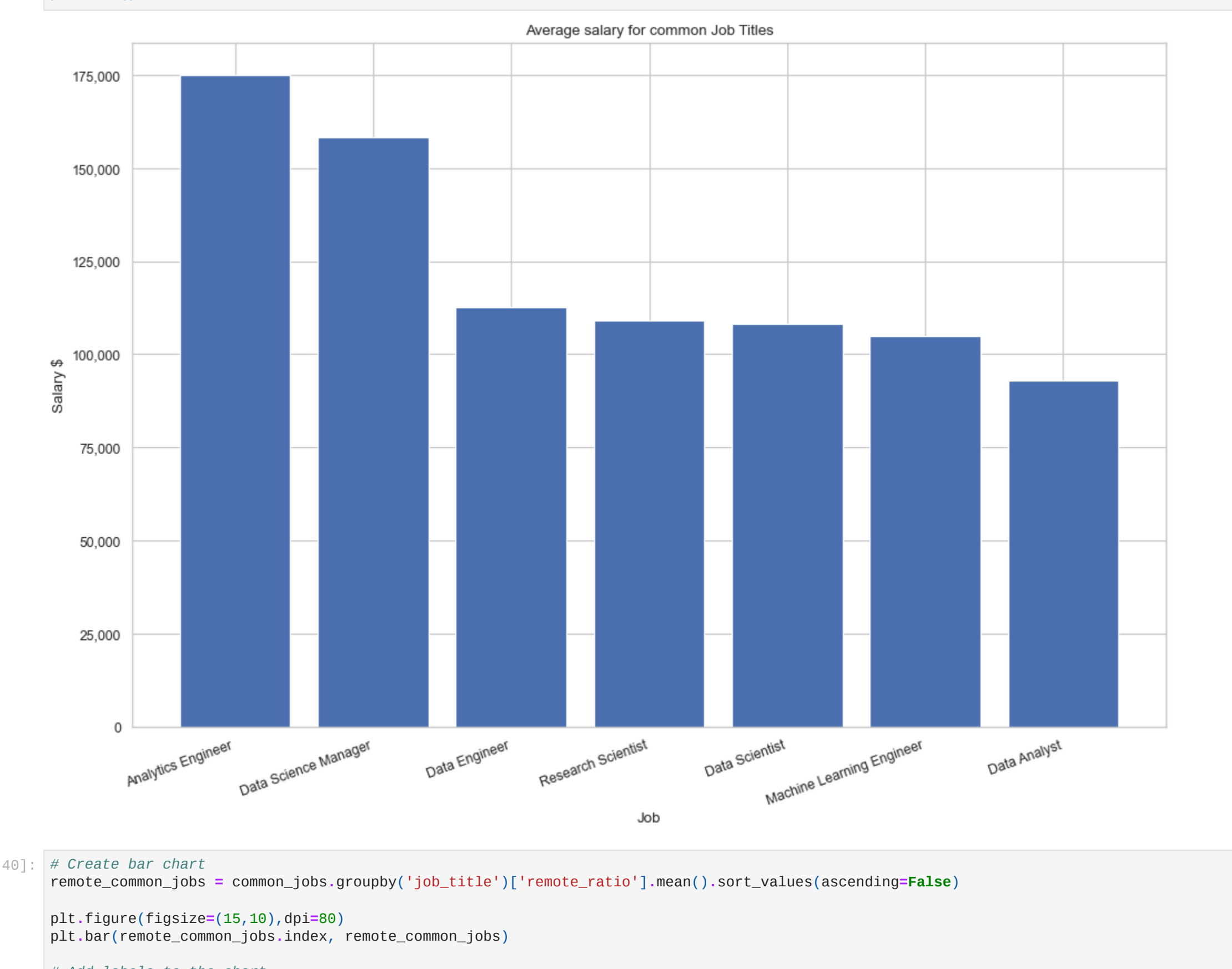
In [39]:
```
# Create bar chart
average_salary = data.groupby('company_location')['salary_in_usd'].mean().sort_values(ascending=False)
top_ten_countries = average_salary.head(10)

plt.figure(figsize=(15,10),dpi=80)
plt.bar(top_ten_countries.index, top_ten_countries)

# Add labels to the chart
plt.xlabel('Country')
plt.ylabel('Salary $')
plt.title('Average of the ten highest salaries by country')
plt.xticks(rotation=20, ha='right')
plt.gca().yaxis.set_major_formatter(ticker.StrMethodFormatter('{x:,.0f}'))

plt.show()
```



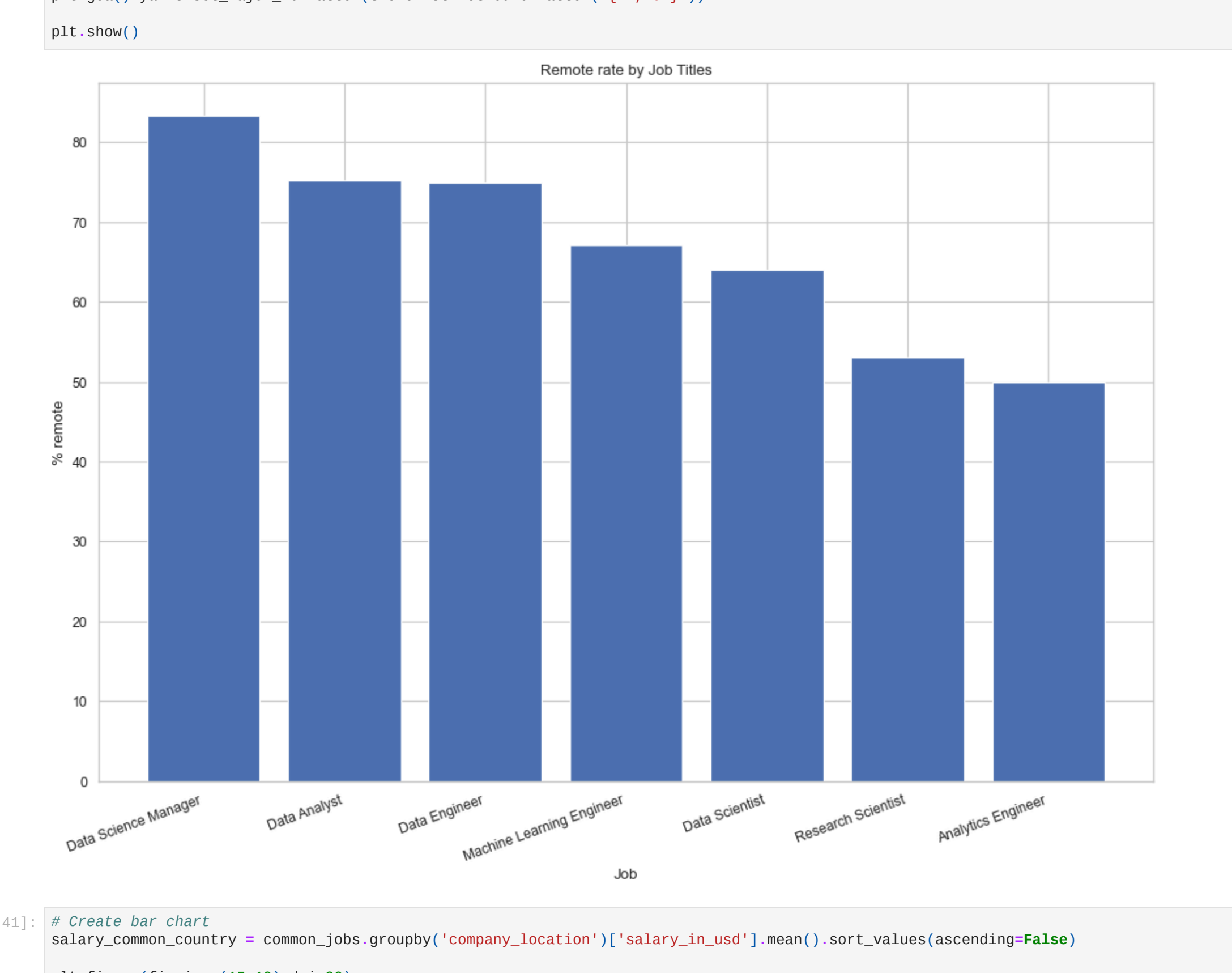
In [36]:
```
common_jobs = ['Data Engineer', 'Data Scientist', 'Data Analyst', 'Machine Learning Engineer', 'Analytics Engineer','Research Scientist', 'Data Scie
common_jobs = data[data['job_title'].isin(common_jobs)]
```

In [37]:
```
salary_common_jobs = common_jobs.groupby('job_title')['salary_in_usd'].mean().sort_values(ascending=False)
remote_common_jobs = common_jobs.groupby('job_title')['remote_ratio'].mean().sort_values(ascending=False)
salary_common_country = common_jobs.groupby('company_location')['salary_in_usd'].mean().sort_values(ascending=False)
```

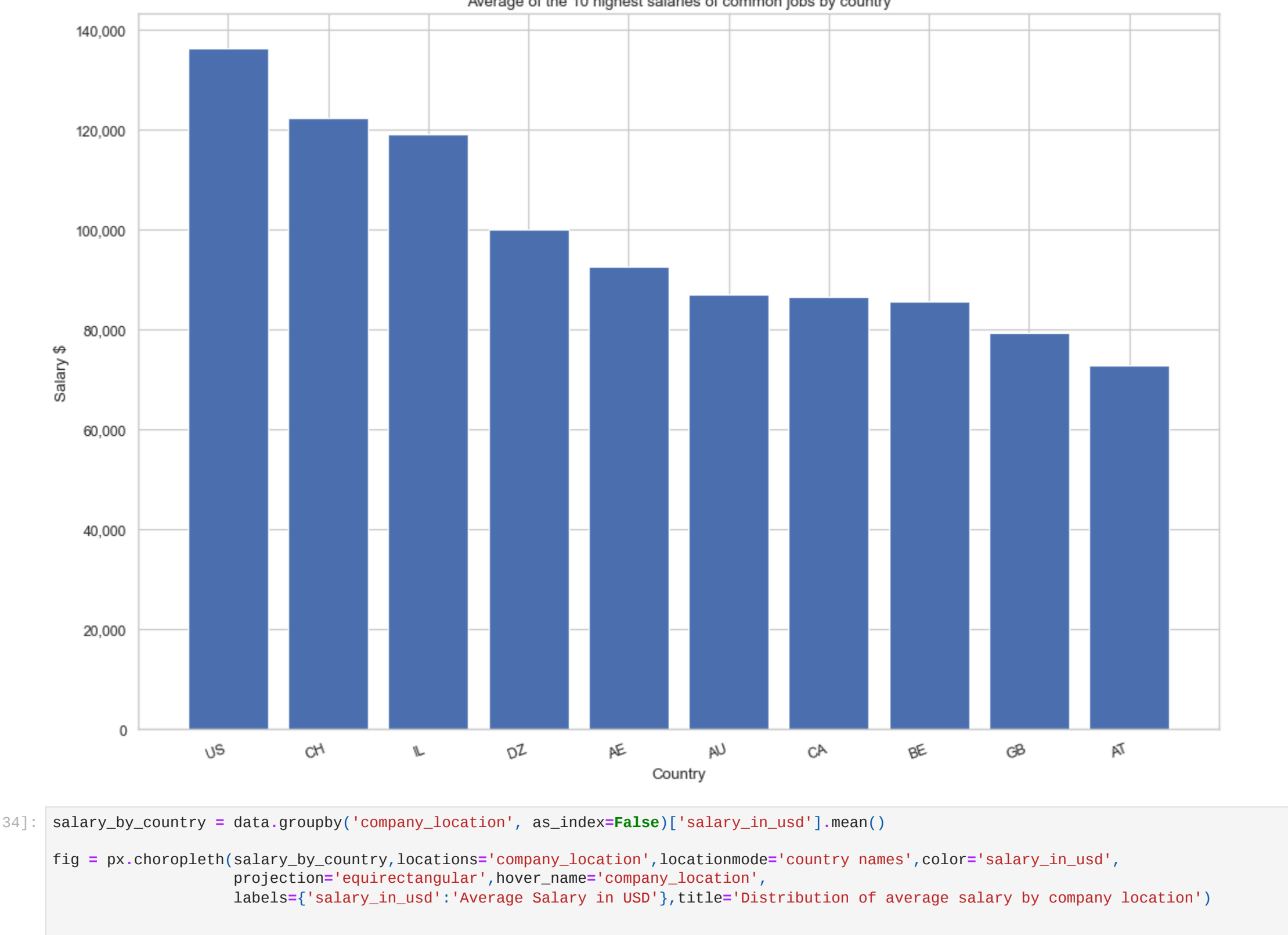
In [38]:
```
plt.figure(figsize=(15,10),dpi=80)
plt.bar(salary_common_jobs.index, salary_common_jobs)

# Add labels to the chart
plt.xlabel('Job')
plt.ylabel('Salary $')
plt.title('Average salary for common Job Titles')
plt.xticks(rotation=20, ha='right')
plt.gca().yaxis.set_major_formatter(ticker.StrMethodFormatter('{x:,.0f}'))

plt.show()
```



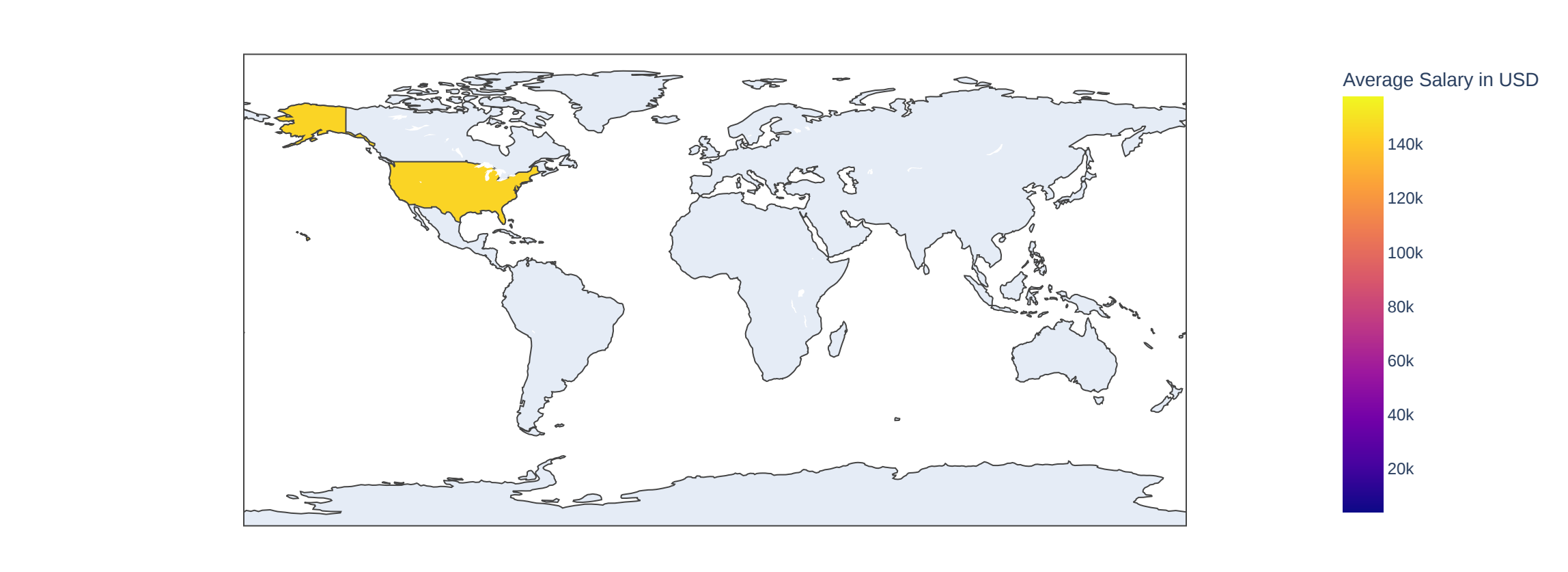
In [40]:
```
# Create bar chart
remote_common_jobs = common_jobs.groupby('job_title')['remote_ratio'].mean().sort_values(ascending=False)

plt.figure(figsize=(15,10),dpi=80)
plt.bar(remote_common_jobs.index, remote_common_jobs)

# Add labels to the chart
plt.xlabel('Job')
plt.ylabel('% remote')
plt.title('Remote rate by Job Titles')
plt.xticks(rotation=20, ha='right')
plt.gca().yaxis.set_major_formatter(ticker.StrMethodFormatter('{x:,.0f}'))

plt.show()
```



In [41]:
```
# Create bar chart
salary_common_country = common_jobs.groupby('company_location')['salary_in_usd'].mean().sort_values(ascending=False)

plt.figure(figsize=(15,10),dpi=80)
plt.bar(salary_common_country.head(10).index, salary_common_country.head(10))

# Add labels to the chart
plt.xlabel('Country')
plt.ylabel('Salary $')
plt.title('Average of the 10 highest salaries of common jobs by country')
plt.xticks(rotation=20, ha='right')
plt.gca().yaxis.set_major_formatter(ticker.StrMethodFormatter('{x:,.0f}'))

plt.show()
```



In [34]:
```
salary_by_country = data.groupby('company_location', as_index=False)['salary_in_usd'].mean()

fig = px.choropleth(salary_by_country,locations='company_location',locationmode='country names',color='salary_in_usd',
                    projection='equirectangular',hover_name='company_location',
                    labels={'salary_in_usd':'Average Salary in USD'},title='Distribution of average salary by company location')

fig.show("notebook")
```



In [ ]: