About the DataSet

- work_year: The year in which the data was recorded. This field indicates the temporal context of the data, important for understanding salary trends over time.
- job_title: The specific title of the job role, like 'Data Scientist', 'Data Engineer', or 'Data Analyst'. This column is crucial for understanding the salary distribution across various specialized roles within the data field.
- job_category: A classification of the job role into broader categories for easier analysis. This might include areas like 'Data Analysis', 'Machine Learning', 'Data Engineering', etc.
- salary_currency: The currency in which the salary is paid, such as USD, EUR, etc. This is important for currency conversion and understanding the actual value of the salary in a global context.
- salary: The annual gross salary of the role in the local currency. This raw salary figure is key for direct regional salary comparisons.

- salary_in_usd: The annual gross salary converted to United States Dollars (USD). This uniform currency conversion aids in global salary comparisons and analyses.
- employee_residence: The country of residence of the employee. This data point can be used to explore geographical salary differences and cost-of-living variations.
- experience_level: Classifies the professional experience level of the employee. Common categories might include 'Entry-level', 'Mid-level', 'Senior', and 'Executive', providing insight into how experience influences salary in data-related roles.
- employment_type: Specifies the type of employment, such as 'Full-time', 'Part-time', 'Contract', etc. This helps in analyzing how different employment arrangements affect salary structures.
- work_setting: The work setting or environment, like 'Remote', 'In-person', or 'Hybrid'. This column reflects the impact of work settings on salary levels in the data industry.
- company_location: The country where the company is located. It helps in analyzing how the location of the company affects salary structures.
- company_size: The size of the employer company, often categorized into small (S), medium (M), and large (L) sizes. This allows for analysis of how company size influences salary.

DataSet link::

https://www.kaggle.com/code/joydeyds/data-science-jobs-analysis/input

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
df = pd.read_csv("jobs_in_data.csv")
```

In [3]:
```
df.head()
```

Out[3]:

	work_year	job_title	job_category	salary_currency	salary	salary_in_usd	employee_
0	2023	Data DevOps Engineer	Data Engineering	EUR	88000	95012	
1	2023	Data Architect	Data Architecture and Modeling	USD	186000	186000	Un
2	2023	Data Architect	Data Architecture and Modeling	USD	81800	81800	Un
3	2023	Data Scientist	Data Science and Research	USD	212000	212000	Un
4	2023	Data Scientist	Data Science and Research	USD	93300	93300	Un

In [4]:
```
df.shape
```

Out[4]:
```
(9355, 12)
```

In [5]:
```
df.size
```

Out[5]:
```
112260
```

In [6]:
```
df.ndim
```

Out[6]:
```
2
```

In [7]:
```
df.isnull()
```

Out[7]:

	work_year	job_title	job_category	salary_currency	salary	salary_in_usd	employe
0	False	False	False	False	False	False	
1	False	False	False	False	False	False	
2	False	False	False	False	False	False	
3	False	False	False	False	False	False	
4	False	False	False	False	False	False	
...	...	...	...	...	...	...	
9350	False	False	False	False	False	False	
9351	False	False	False	False	False	False	
9352	False	False	False	False	False	False	
9353	False	False	False	False	False	False	
9354	False	False	False	False	False	False	

9355 rows × 12 columns

In [8]:
```
df.isnull().sum()
```

Out[8]:
```
work_year            0
job_title            0
job_category         0
salary_currency      0
salary               0
salary_in_usd        0
employee_residence   0
experience_level     0
employment_type      0
work_setting         0
company_location     0
company_size         0
dtype: int64
```

In [9]:
```
df.describe()
```

Out[9]:

	work_year	salary	salary_in_usd
count	9355.000000	9355.000000	9355.000000
mean	2022.760449	149927.981293	150299.495564
std	0.519470	63608.835387	63177.372024
min	2020.000000	14000.000000	15000.000000
25%	2023.000000	105200.000000	105700.000000
50%	2023.000000	143860.000000	143000.000000
75%	2023.000000	187000.000000	186723.000000
max	2023.000000	450000.000000	450000.000000

In [11]:
```
df.describe(include='all')
```

Out[11]:

	work_year	job_title	job_category	salary_currency	salary	salary_in_
count	9355.000000	9355	9355	9355	9355.000000	9355.000
unique	NaN	125	10	11	NaN	N
top	NaN	Data Engineer	Data Science and Research	USD	NaN	N
freq	NaN	2195	3014	8591	NaN	N
mean	2022.760449	NaN	NaN	NaN	149927.981293	150299.495
std	0.519470	NaN	NaN	NaN	63608.835387	63177.372
min	2020.000000	NaN	NaN	NaN	14000.000000	15000.000
25%	2023.000000	NaN	NaN	NaN	105200.000000	105700.000
50%	2023.000000	NaN	NaN	NaN	143860.000000	143000.000
75%	2023.000000	NaN	NaN	NaN	187000.000000	186723.000
max	2023.000000	NaN	NaN	NaN	450000.000000	450000.000

In [12]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9355 entries, 0 to 9354
Data columns (total 12 columns):
 #   Column              Non-Null Count  Dtype 
---  ------              --------------  ----- 
 0   work_year           9355 non-null   int64 
 1   job_title           9355 non-null   object
 2   job_category        9355 non-null   object
 3   salary_currency     9355 non-null   object
 4   salary              9355 non-null   int64 
 5   salary_in_usd       9355 non-null   int64 
 6   employee_residence  9355 non-null   object
 7   experience_level    9355 non-null   object
 8   employment_type     9355 non-null   object
 9   work_setting        9355 non-null   object
 10  company_location    9355 non-null   object
 11  company_size        9355 non-null   object
dtypes: int64(3), object(9)
memory usage: 877.2+ KB
```

In [13]:
```
df.dtypes
```

Out[13]:
```
work_year              int64
job_title             object
job_category          object
salary_currency       object
salary                 int64
salary_in_usd          int64
employee_residence    object
experience_level      object
employment_type       object
work_setting          object
company_location      object
company_size          object
dtype: object
```

In [14]:
```
df.nunique()
```

Out[14]:
```
work_year               4
job_title             125
job_category           10
salary_currency        11
salary               1507
salary_in_usd        1786
employee_residence     83
experience_level        4
employment_type         4
work_setting            3
company_location       70
company_size            3
dtype: int64
```

In [15]:
```
# Get all the Unique Job Categories

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

Out[15]:
```
array(['Data Engineering', 'Data Architecture and Modeling',
       'Data Science and Research', 'Machine Learning and AI',
       'Data Analysis', 'Leadership and Management',
       'BI and Visualization', 'Data Quality and Operations',
       'Data Management and Strategy', 'Cloud and Database'], dtype=object)
```

In [16]:
```
unique_job_categories  = df['job_category'].unique()

for job_category in unique_job_categories :
    print(job_category)
```

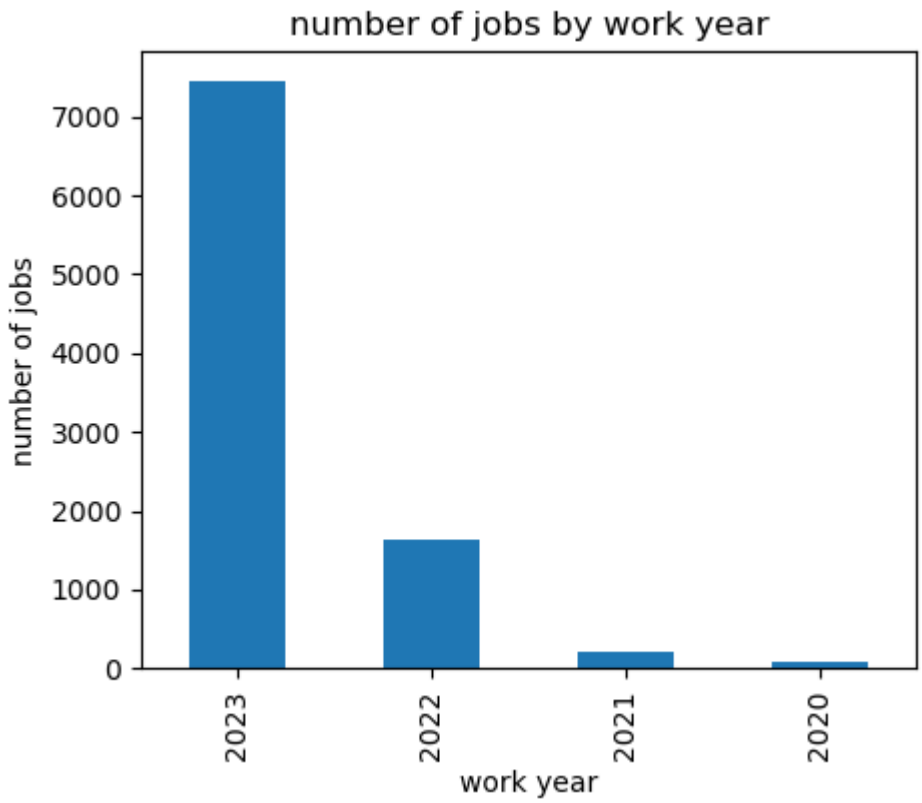
```
Data Engineering
Data Architecture and Modeling
Data Science and Research
Machine Learning and AI
Data Analysis
Leadership and Management
BI and Visualization
Data Quality and Operations
Data Management and Strategy
Cloud and Database
```

In [17]:
```
# Count the number of jobs for each work year

count_by_year = df['work_year'].value_counts()
```

In [18]:
```
for work_year, count in count_by_year.items():
    print(f"Work Year: {work_year}, Number of Jobs: {count}")
```

```
Work Year: 2023, Number of Jobs: 7453
Work Year: 2022, Number of Jobs: 1634
Work Year: 2021, Number of Jobs: 197
Work Year: 2020, Number of Jobs: 71
```

In [22]:
```
# Represent the above result using bar chart

plt.figure(figsize =(5,4))
count_by_year.plot(kind = 'bar')
plt.title('number of jobs by work year')
plt.xlabel('work year')
plt.ylabel('number of jobs')
plt.show()
```



In [23]:
```
# What is the maximum, minimum and average salary?
df.columns
```

Out[23]:
```
Index(['work_year', 'job_title', 'job_category', 'salary_currency', 'salary',
       'salary_in_usd', 'employee_residence', 'experience_level',
       'employment_type', 'work_setting', 'company_location', 'company_size'],
      dtype='object')
```

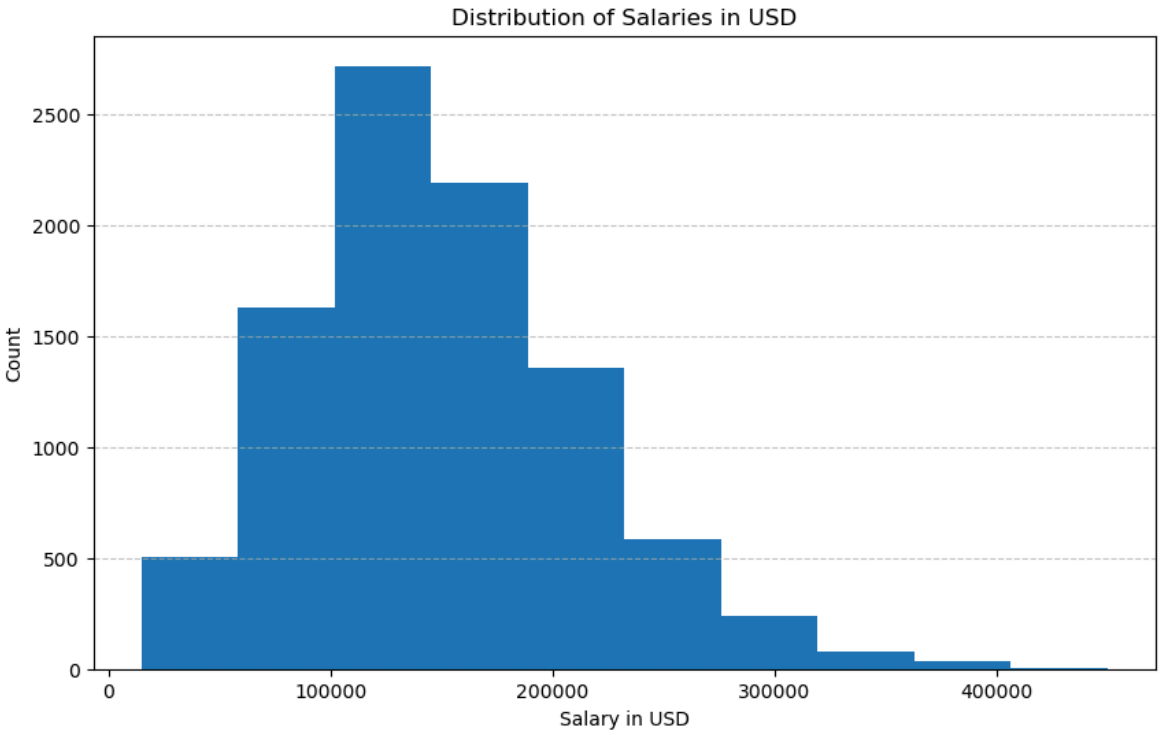
In [24]:
```
salary_range = df['salary_in_usd'].agg(['min','max'])
print("Max Salary:", salary_range['max'])
print("Min Salary:", salary_range['min'])
```

```
Max Salary: 450000
Min Salary: 15000
```

In [25]:
```
mean_salary = df['salary_in_usd'].mean()
print(mean_salary)
```

```
150299.4955638696
```

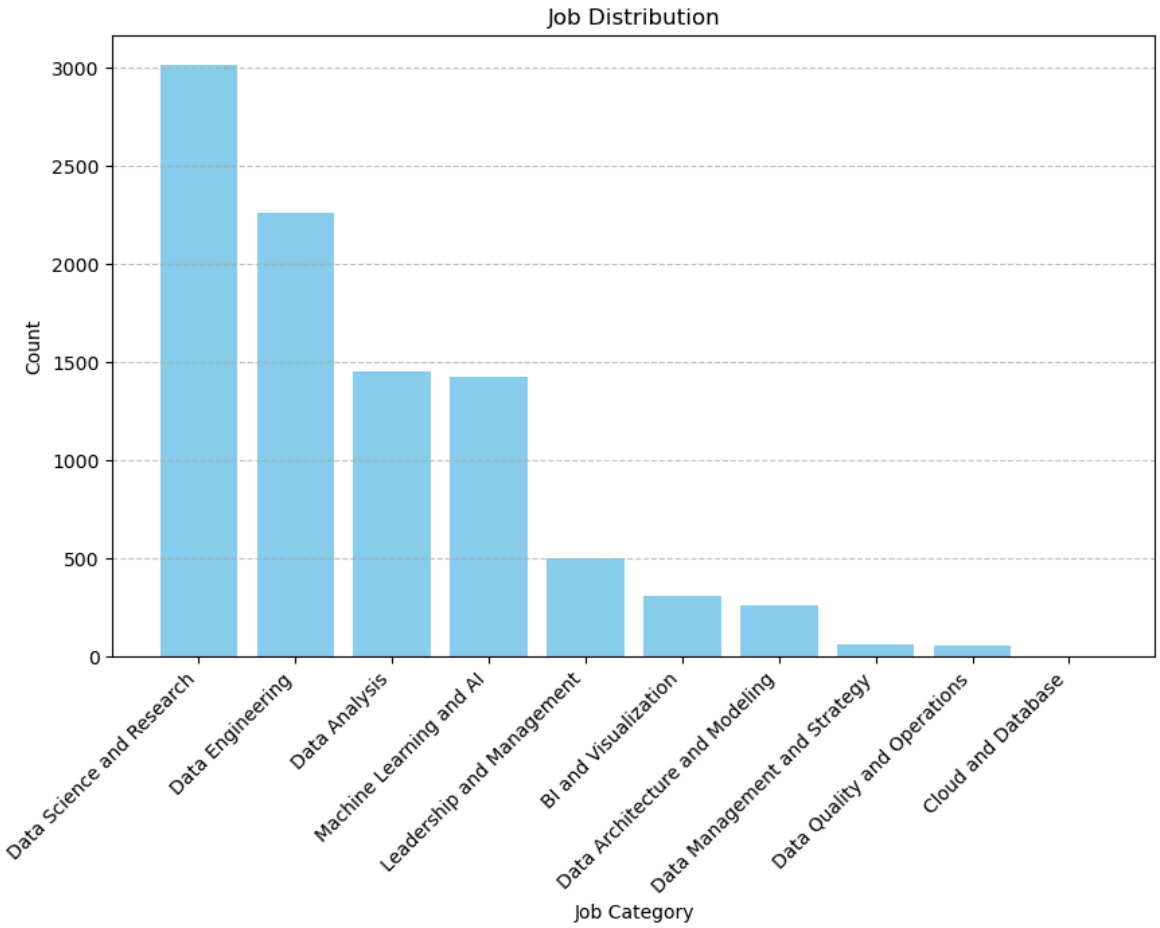
In [26]:
```
# Represt the salary distribution of salaries

plt.figure(figsize=(10, 6))
plt.hist(df['salary_in_usd'])
plt.title('Distribution of Salaries in USD')
plt.xlabel('Salary in USD')
plt.ylabel('Count')
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```



In [27]:
```
unique_job_categories = df['job_category'].value_counts()
print(unique_job_categories)
```

```
job_category
Data Science and Research         3014
Data Engineering                  2260
Data Analysis                     1457
Machine Learning and AI           1428
Leadership and Management          503
BI and Visualization               313
Data Architecture and Modeling     259
Data Management and Strategy        61
Data Quality and Operations         55
Cloud and Database                   5
Name: count, dtype: int64
```

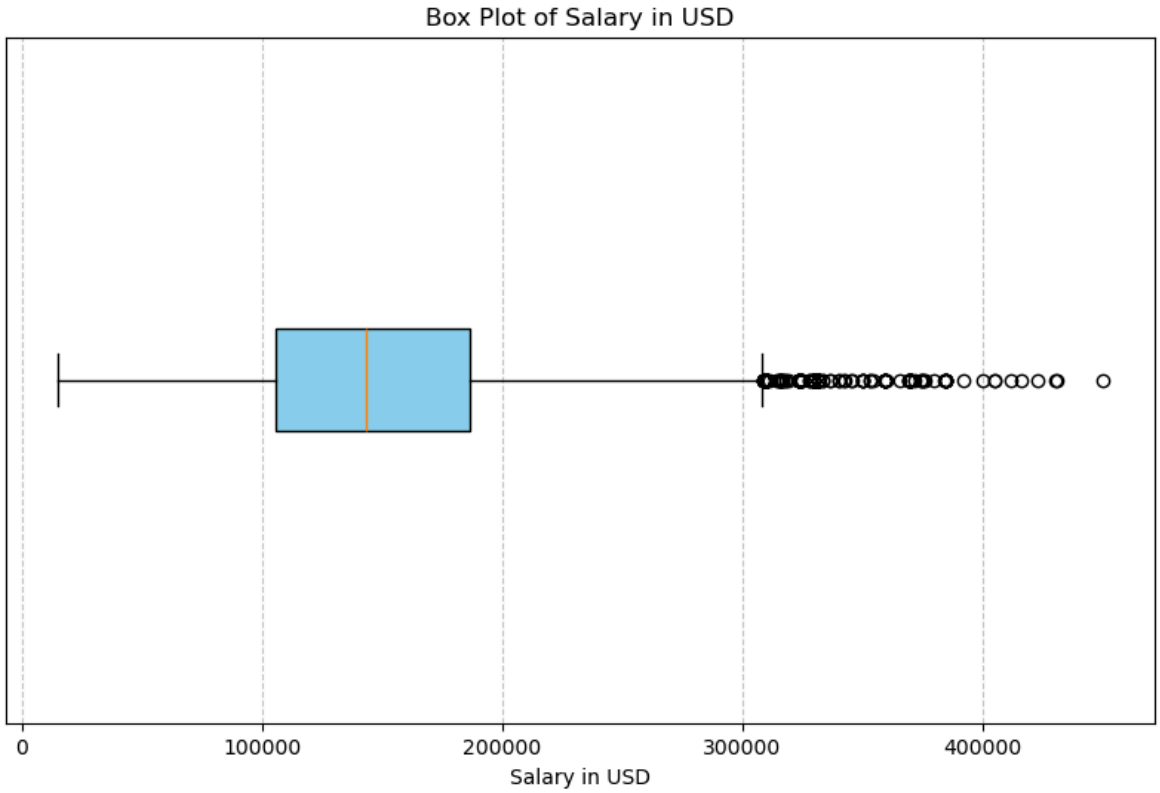
In [28]:
```
plt.figure(figsize=(10, 6))
plt.bar(unique_job_categories.index, unique_job_categories.values, color='skyblu
plt.title('Job Distribution')
plt.xlabel('Job Category')
plt.ylabel('Count')
plt.xticks(rotation=45, ha='right')  # Rotate x-axis labels for better readabili
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```



In [29]:
```
average_salary_by_job_title = df.groupby('job_title')['salary_in_usd'].mean()
print(average_salary_by_job_title)
```

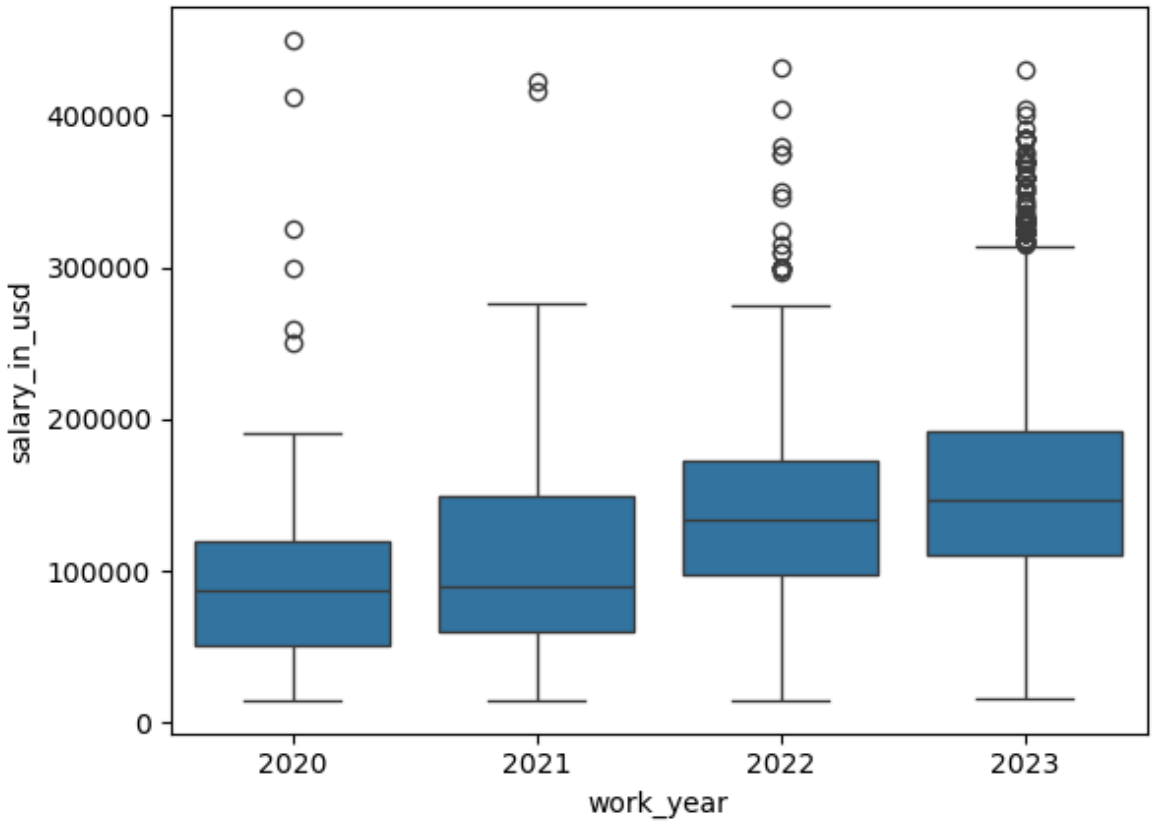
```
job_title
AI Architect                       250328.000000
AI Developer                       141140.888889
AI Engineer                        171663.972222
AI Programmer                       68817.400000
AI Research Engineer                73271.500000
                                          ...
Sales Data Analyst                  60000.000000
Software Data Engineer             111627.666667
Staff Data Analyst                  79917.000000
Staff Data Scientist               134500.000000
Staff Machine Learning Engineer    185000.000000
Name: salary_in_usd, Length: 125, dtype: float64
```

In [30]:
```
plt.figure(figsize=(10, 6))
plt.boxplot(df['salary_in_usd'], vert=False, patch_artist=True, boxprops=dict(fa
plt.title('Box Plot of Salary in USD')
plt.xlabel('Salary in USD')
plt.yticks([])
plt.grid(axis='x', linestyle='--', alpha=0.7)
plt.show()
```



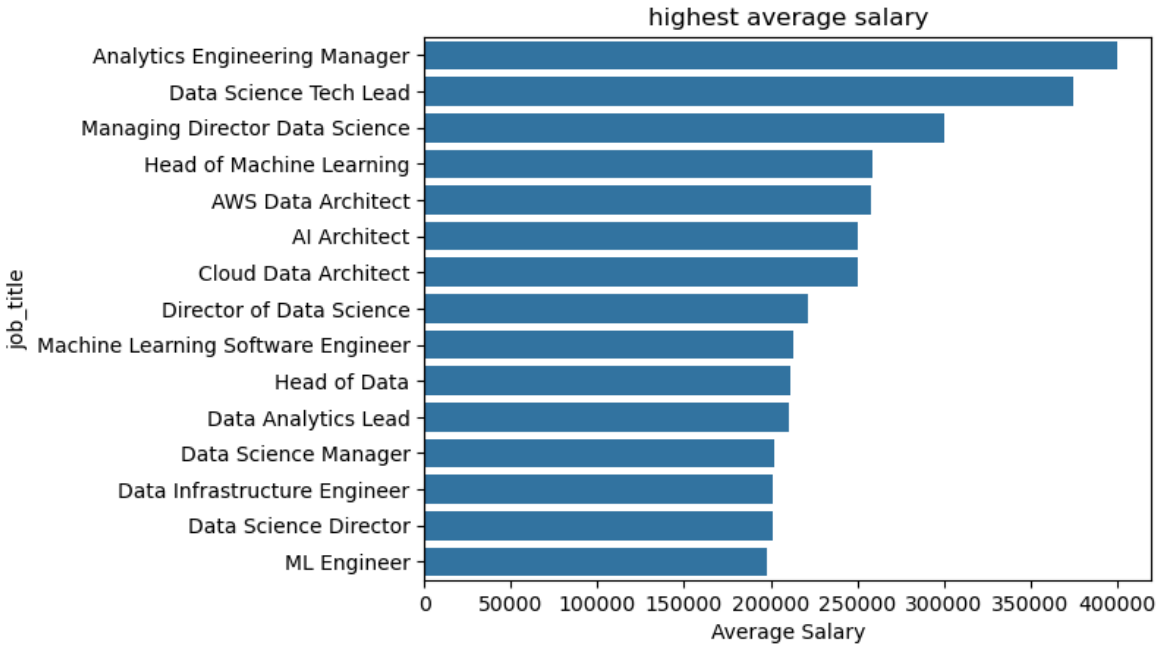
In [31]:
```
sns.boxplot(df, y='salary_in_usd', x='work_year')
```

Out[31]: <Axes: xlabel='work_year', ylabel='salary_in_usd'>



In [33]:
```
sns.barplot(data = df.groupby('job_title')['salary_in_usd'].mean().sort_values(a
            .reset_index(name='Average Salary').head(15),x='Average Salary', y='
plt.title('highest average salary')
```

Out[33]: Text(0.5, 1.0, 'highest average salary')



In [34]:
```
residence = df['employee_residence'].value_counts()
residence
```

Out[34]:
```
employee_residence
United States     8086
United Kingdom     442
Canada             224
Spain              117
Germany             66
                   ...
Serbia               1
New Zealand          1
Hong Kong            1
Luxembourg           1
Malta                1
Name: count, Length: 83, dtype: int64
```

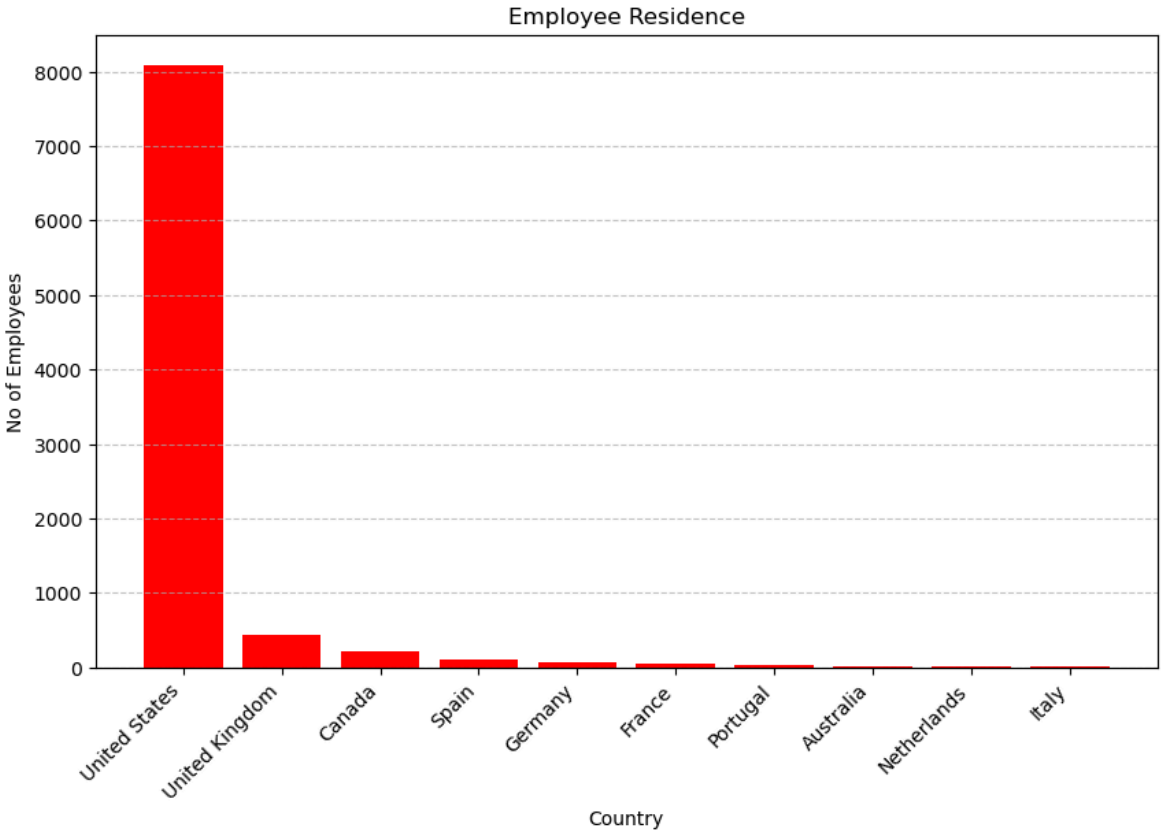
In [35]:
```
top_10_residence = residence.head(10)
```

In [36]:
```
top_10_residence
```

Out[36]:
```
employee_residence
United States     8086
United Kingdom     442
Canada             224
Spain              117
Germany             66
France              54
Portugal            26
Australia           21
Netherlands         21
Italy               20
Name: count, dtype: int64
```

In [41]:
```
plt.figure(figsize=(10, 6))
plt.bar(top_10_residence.index,top_10_residence.values, color='red')
plt.title('Employee Residence')
plt.xlabel('Country')
plt.ylabel('No of Employees')
```

```
plt.xticks(rotation=45, ha='right')  # Rotate x-axis labels for better readabili
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```

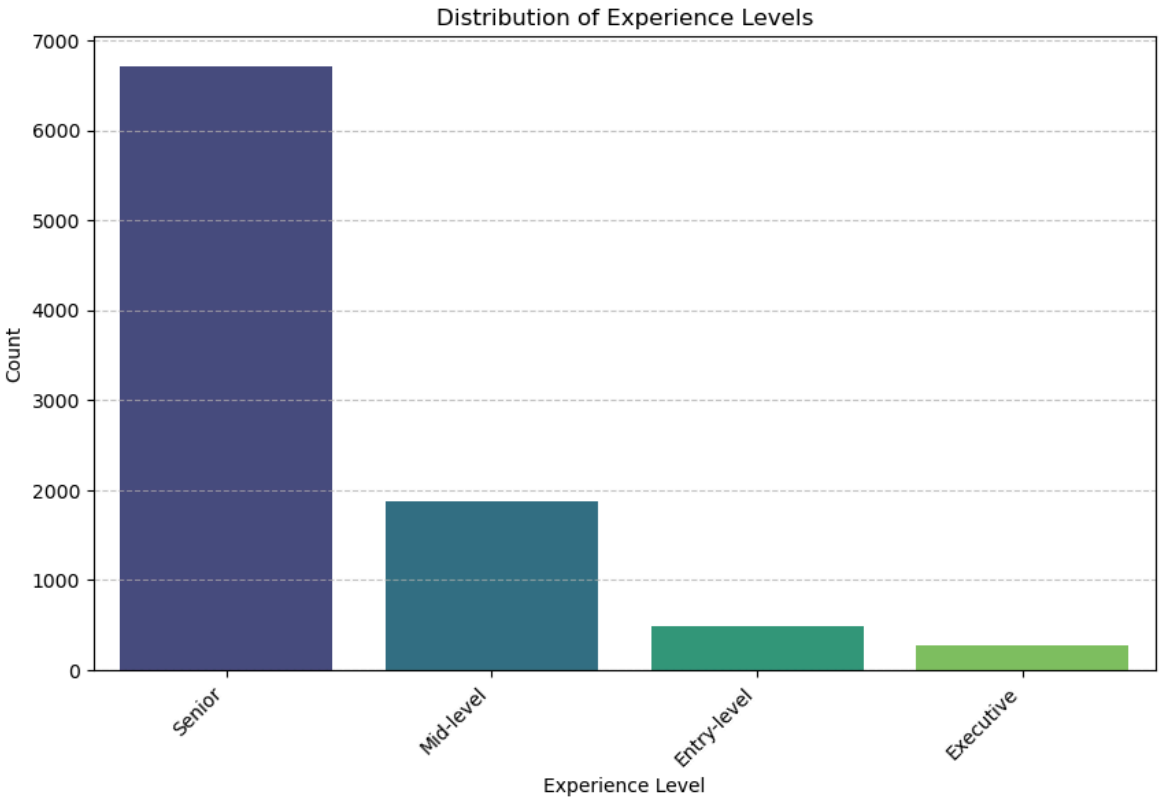


In [38]:
```
experience_level_counts = df['experience_level'].value_counts()
experience_level_counts
```

Out[38]:
```
experience_level
Senior         6709
Mid-level      1869
Entry-level     496
Executive       281
Name: count, dtype: int64
```

In [39]:
```
plt.figure(figsize=(10, 6))
sns.barplot(x=experience_level_counts.index, y=experience_level_counts.values, p
plt.title('Distribution of Experience Levels')
plt.xlabel('Experience Level')
plt.ylabel('Count')
plt.xticks(rotation=45, ha='right')  # Rotate x-axis labels for better readabili
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```

```
C:\Users\sanad\AppData\Local\Temp\ipykernel_5584\1417157126.py:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v
0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effe
ct.

  sns.barplot(x=experience_level_counts.index, y=experience_level_counts.values,
palette='viridis')
```


Distribution of Experience Levels
Count
7000
6000
5000
4000
3000
2000
1000
0
Senior
Mid-level
Entry-level
Executive
Experience Level

In [ ]: