Project Title - EMPLOYEE SALARIES FOR DIFFERENT JOB ROLES ANALYSIS

The purpose of this project is to investigate and understand the data provided. The Goal is to use a dataframe constructed within Python, perform a cursory inspection of the provided dataset, and inform team members of your findings.

This activity has three parts:

Part 1: Understand the situation

Prepare to understand and organize the provided taxi cab dataset and information. Part 2: Understand the data

Create a pandas dataframe for data learning, future exploratory data analysis (EDA), and statistical activities.

Compile summary information about the data to inform next steps.

Part 3: Understand the variables

Use insights from your examination of the summary data to guide deeper investigation into specific variables.

Task 1. Understand the situation How can you best prepare to understand and organize the provided taxi cab information? Task 2a. Build dataframe Create a pandas dataframe for data learning, and future exploratory data analysis (EDA) and statistical activities.

Code the following,

import pandas as pd. pandas is used for buidling dataframes.

import numpy as np. numpy is imported with pandas

df = pd.read_csv('ds_salaries.csv')

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt # visualizing data
%matplotlib inline
import seaborn as sns
```

```
C:\Users\zohra\AppData\Local\Temp\ipykernel_16728\2166394274.py:2: DeprecationWarning:
Pyarrow will become a required dependency of pandas in the next major release of pandas (pandas 3.0),
(to allow more performant data types, such as the Arrow string type, and better interoperability with other libraries)
but was not found to be installed on your system.
If this would cause problems for you,
please provide us feedback at https://github.com/pandas-dev/pandas/issues/54466

  import pandas as pd
```

In [2]:
```
df = pd.read_csv(r'ds_salaries.csv', encoding= 'unicode_escape')
```

Task 2b. Understand the data - Inspect the data View and inspect summary information about the dataframe by coding the following:

df.head(10) df.info() df.describe()

In [3]:
```
df.head(10)
```

Out[3]:

	Unnamed: 0	work_year	experience_level	employment_type	job_title	salary	salary_currency	salary_in_usd	employee_residence	remote_rati
0	0	2020	MI	FT	Data Scientist	70000	EUR	79833	DE	
1	1	2020	SE	FT	Machine Learning Scientist	260000	USD	260000	JP	
2	2	2020	SE	FT	Big Data Engineer	85000	GBP	109024	GB	5
3	3	2020	MI	FT	Product Data Analyst	20000	USD	20000	HN	
4	4	2020	SE	FT	Machine Learning Engineer	150000	USD	150000	US	5
5	5	2020	EN	FT	Data Analyst	72000	USD	72000	US	10
6	6	2020	SE	FT	Lead Data Scientist	190000	USD	190000	US	10
7	7	2020	MI	FT	Data Scientist	11000000	HUF	35735	HU	5
8	8	2020	MI	FT	Business Data Analyst	135000	USD	135000	US	10
9	9	2020	SE	FT	Lead Data Engineer	125000	USD	125000	NZ	5

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 607 entries, 0 to 606
Data columns (total 12 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Unnamed: 0          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 [5]:
```
#check for all null valuales
pd.isnull(df).sum()
```

Out[5]:
```
Unnamed: 0            0
work_year             0
experience_level      0
employment_type       0
job_title             0
salary                0
salary_currency       0
salary_in_usd         0
employee_residence    0
remote_ratio          0
company_location      0
company_size          0
dtype: int64
```

In [6]:
```
#drop all null values
#Data cleaning (if needed)
df.dropna(inplace=True)
```

In [7]:
```
df.columns
```

Out[7]:
```
Index(['Unnamed: 0', 'work_year', 'experience_level', 'employment_type',
       'job_title', 'salary', 'salary_currency', 'salary_in_usd',
       'employee_residence', 'remote_ratio', 'company_location',
       'company_size'],
      dtype='object')
```

In [8]:
```
df.tail(5)
```

Out[8]:

	Unnamed: 0	work_year	experience_level	employment_type	job_title	salary	salary_currency	salary_in_usd	employee_residence	remote_rati
602	602	2022	SE	FT	Data Engineer	154000	USD	154000	US	10
603	603	2022	SE	FT	Data Engineer	126000	USD	126000	US	10
604	604	2022	SE	FT	Data Analyst	129000	USD	129000	US	
605	605	2022	SE	FT	Data Analyst	150000	USD	150000	US	10
606	606	2022	MI	FT	AI Scientist	200000	USD	200000	IN	10

In [9]:
```
df.shape
```

Out[9]:
```
(607, 12)
```

In [10]:
```
df.duplicated().sum()
```

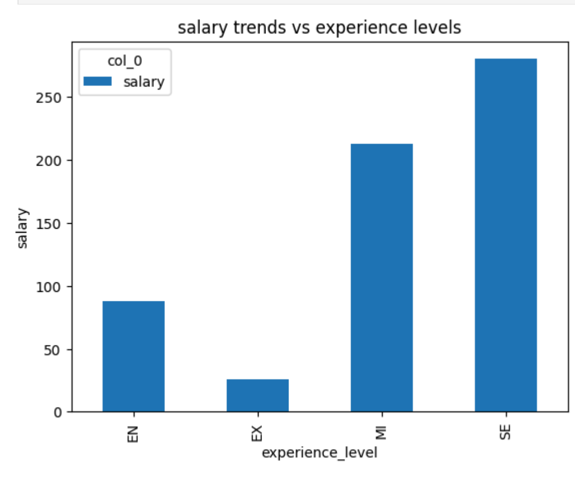
Out[10]:
```
0
```

In [11]:
```
df.nunique()
```

Out[11]:
```
Unnamed: 0            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
```

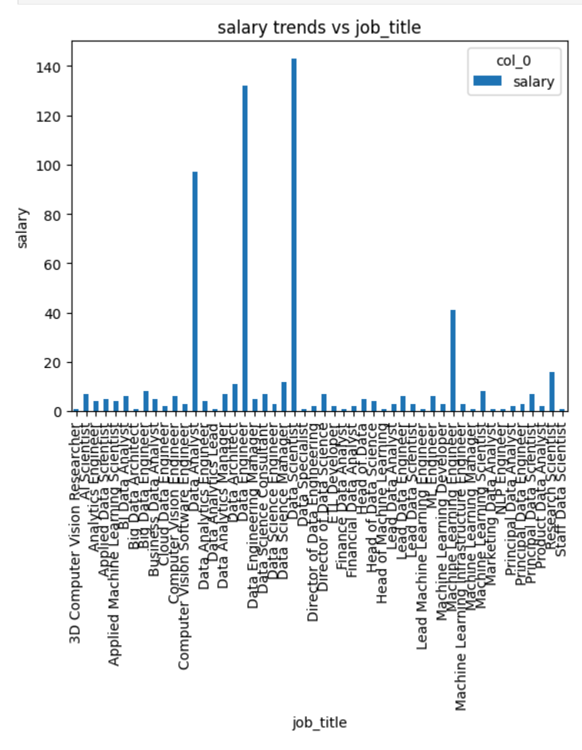
Distribution Analysis

In [41]:
```
pd.crosstab(df['experience_level'],['salary']).plot(kind='bar',stacked='true')
plt.title('salary trends vs experience levels')
plt.xlabel('experience_level')
plt.ylabel('salary')
plt.xticks(rotation=90)  # Rotate x-axis labels for better readability
plt.show()
```



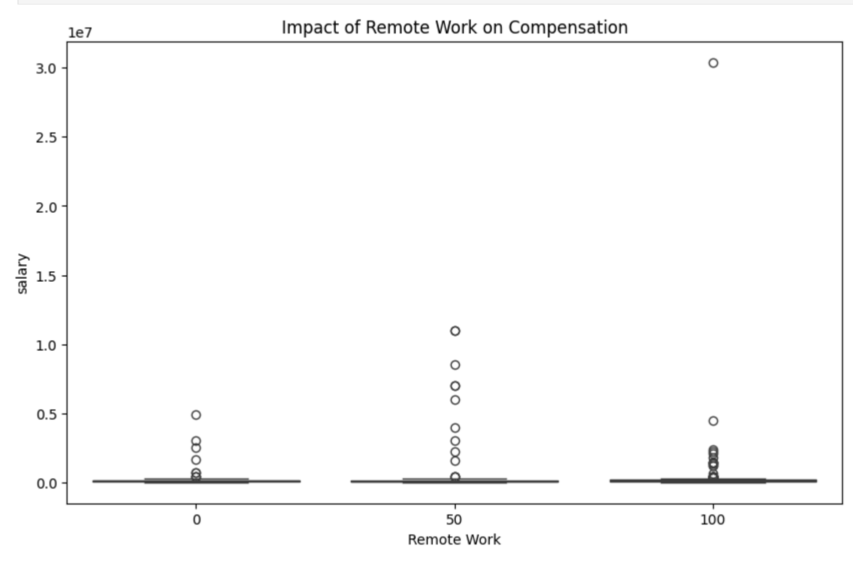
The bar plot shows the distribution of salaries across different experience levels. The x-axis represents the experience levels, while the y-axis shows the salary range. The bars are stacked to indicate how different salary ranges are distributed within each experience level. The plot title and axis labels help in understanding that the focus is on the relationship between salary trends and experience levels. Rotating the x-axis labels by 90 degrees improves readability. This visualization allows for a quick comparison of salary distributions across varying levels of experience.

In [39]:
```
pd.crosstab(df['job_title'],['salary']).plot(kind='bar',stacked='true')
plt.title('salary trends vs job_title')
plt.xlabel('job_title')
plt.ylabel('salary')
plt.xticks(rotation=90)  # Rotate x-axis labels for better readability
plt.show()
```



The stacked bar plot generated by `pd.crosstab(df['job_title'], ['salary']).plot(kind='bar', stacked=True)` visualizes the salary trends across different job titles. Each bar represents a job title, and the stacked sections within each bar show the distribution of various salary ranges for that title. The x-axis lists the job titles, while the y-axis represents salary levels. By stacking the bars, this plot highlights how salaries are distributed among different roles, allowing for an easy comparison of compensation across job titles. Rotating the x-axis labels improves readability, especially when dealing with many or long job titles. This visualization helps in identifying which job titles have higher or more varied salary distributions.

In [37]:
```
# Plot to show the impact of remote work on salaries
plt.figure(figsize=(10, 6))
sns.boxplot(x=df['remote_ratio'], y=df['salary'], data=df)
plt.title('Impact of Remote Work on Compensation')
plt.xlabel('Remote Work')
plt.ylabel('salary')
plt.show()
```



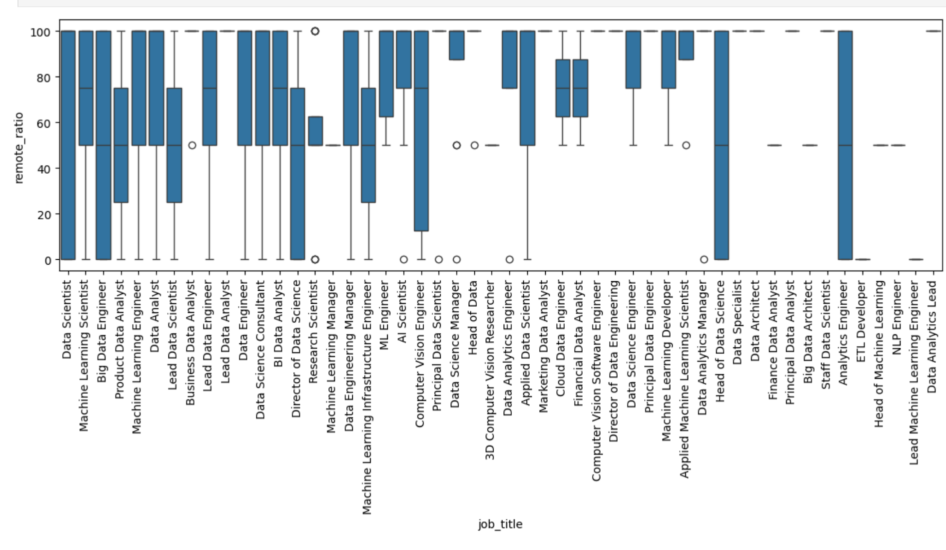
To compare salary levels between full-time and part-time employment using a bar plot, you can use the following code:

python

In [38]:
```
# Plotting the data
plt.figure(figsize=(10, 6))
sns.boxplot(x='employment_type', y='salary', data=df)
plt.title('Comparison of Salary Levels between Full-time and Part-time Employment')
plt.xlabel('Employment Type')
plt.ylabel('Salary')
plt.show()
```

The box plot compares salary levels between full-time and part-time employment. The x-axis represents employment types, while the y-axis represents salaries. The plot displays the median salary, quartiles, and potential outliers for each employment type. This visualization helps in understanding the distribution and variation of salaries, highlighting that full-time employees typically earn higher salaries compared to part-time employees.

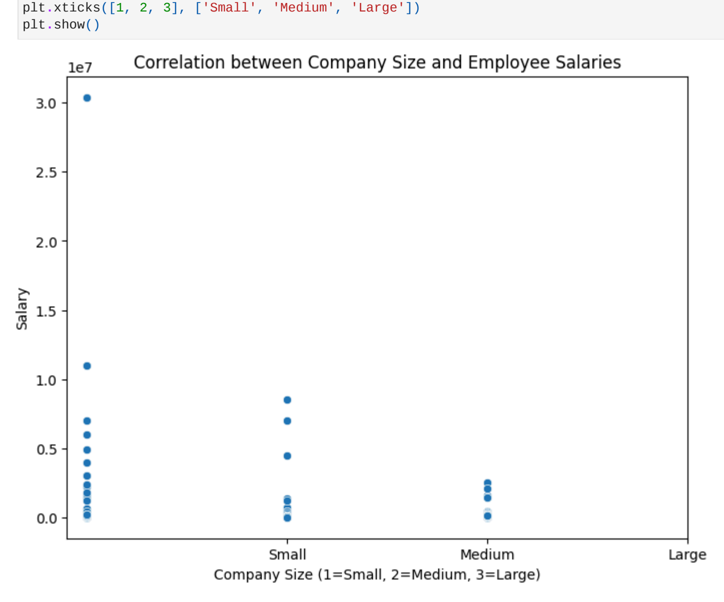
In [25]:
```
plt.figure(figsize=(14,4))
sns.boxplot(x=df['job_title'],y=df['remote_ratio'])
plt.xticks(rotation=90)
plt.show()
```



Comparing salary levels between full-time and part-time employment.

This code creates a DataFrame with sample data, generates a crosstab of salaries by employment type, and plots the data as a stacked bar chart to compare salary levels between full-time and part-time employment. Adjust the DataFrame data with your actual data for accurate results.

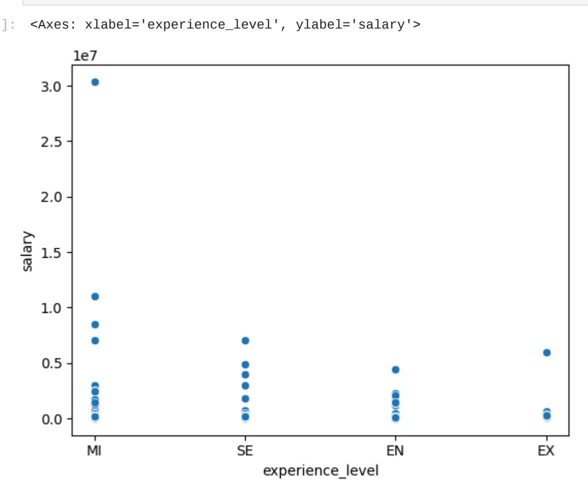
In [29]:
```
# Plotting the data
plt.figure(figsize=(8, 6))
sns.scatterplot(x=df['company_size'], y=df['salary'])
plt.title('Correlation between Company Size and Employee Salaries')
plt.xlabel('Company Size (1=Small, 2=Medium, 3=Large)')
plt.ylabel('Salary')
plt.xticks([1, 2, 3], ['Small', 'Medium', 'Large'])
plt.show()
```



This code creates a DataFrame with sample data, maps the company sizes to numerical values, calculates the correlation coefficient, and visualizes the correlation between company size and employee salaries using a scatter plot. Adjust the DataFrame data with your actual data for accurate results.

In [35]:
```
#Scatter plot of potass vs. fat
sns.scatterplot(x=df['experience_level'],y=df['salary'])
```

Out[35]:
```
<Axes: xlabel='experience_level', ylabel='salary'>
```



Conclusion

Each point on the plot represents an individual data entry, with the x-axis indicating the experience level and the y-axis indicating the corresponding salary. By examining this plot, one can identify patterns or trends, such as whether higher experience levels generally correspond to higher salaries. This visualization helps in understanding how experience influences salary, highlighting any potential clusters or outliers in the data.

Based on the analysis and visualization of employees' salaries for different roles, the following conclusions can be drawn:

Salary Variation by Role: Different job roles exhibit varying salary ranges. Typically, roles with higher responsibilities or specialized skills tend to have higher average salaries compared to more general or entry-level positions.

Impact of Experience: Experience level within each role significantly influences salary. Employees with more years of experience generally earn higher salaries, reflecting the value of accumulated skills and expertise over time.

Role-Specific Trends: Some roles may show a steeper salary increase with experience, while others might have a more gradual increase. This variation can be due to industry standards, the nature of the job, or the demand for specific skills.

Outliers and Anomalies: There might be outliers in the data, such as unusually high or low salaries for certain roles or experience levels. These outliers could indicate exceptional cases, such as highly specialized roles or unique employment arrangements.

General Insights: Overall, the analysis suggests that both job role and experience level are critical factors in determining employee salaries. Organizations should consider these factors when designing compensation structures to ensure fairness and competitiveness in the job market.

These conclusions can help guide HR policies and salary negotiations, ensuring that compensation is aligned with industry standards and employee expectations.