# CC5067NI-Smart Data Discovery

## 60% Individual Coursework

## 2023-24 Spring

**Student Name: Bishwa Ram Joshi**

**London Met ID: 22085517**

**College ID: NP01CP4S230130**

**Assignment Due Date: Monday, May 13, 2024**

**Assignment Submission Date: Monday, May 13, 2024**

**Word Count:** Click or tap here to enter text.

*I confirm that I understand my coursework needs to be submitted online via MySecondTeacher under the relevant module page before the deadline in order for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a marks of zero will be awarded.*

## Table of Contents

## 1. Data Understanding

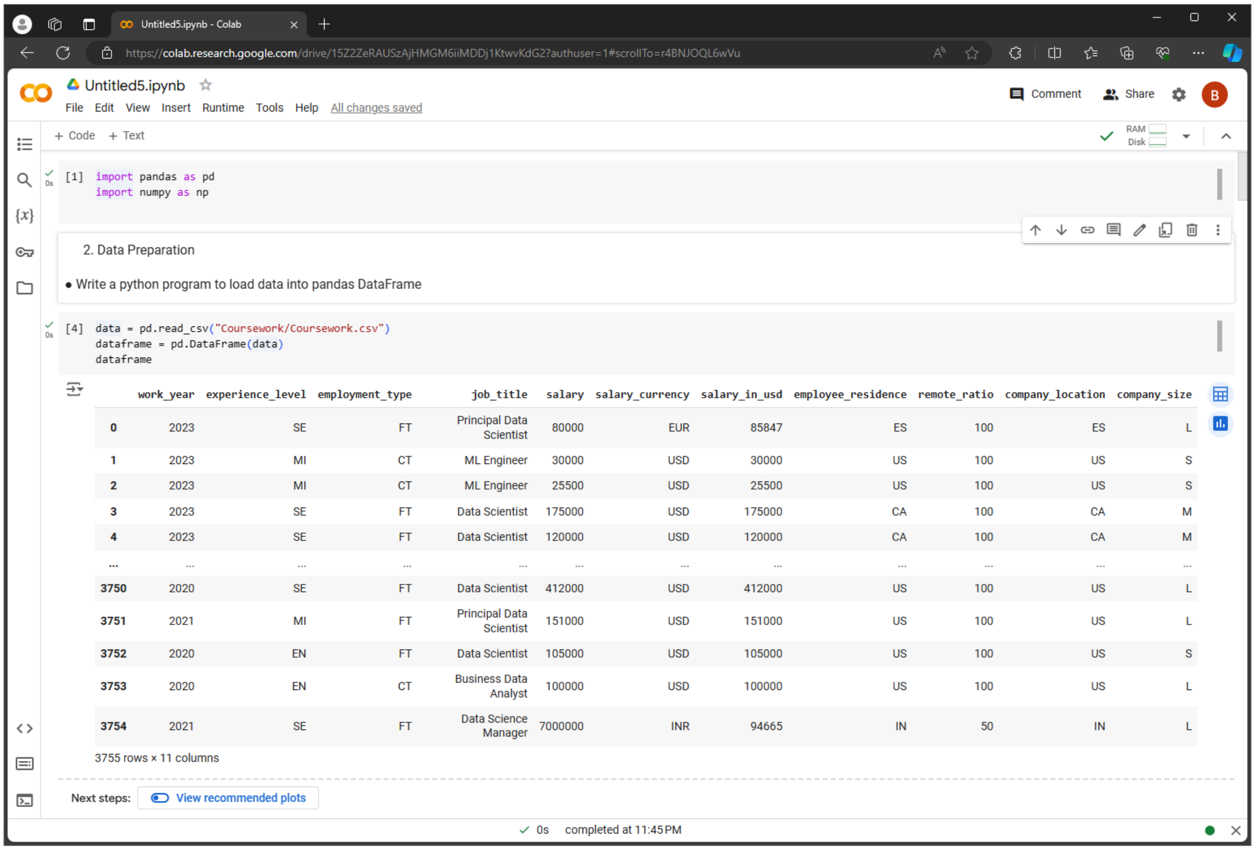
The dataset contains information related to various factors that can influence salary levels in the field of data science. It includes details such as work experience, experience level, employment type, job title, salary (in different currencies and converted to USD), employee residence, remote work ratio, company location, and company size. The objective of this dataset is to analyse the factors influencing salaries of data scientists and discover any patterns or trends within the data.

| S.No | Column Name | Description | Data Type |
|---|---|---|---|
| 1 | work_year | Number of years of work experience | Integer |
| 2 | experience_level | Experience level (Senior, Medium, Entry, Executive) | VarChar |
| 3 | employment_type | Type of employment (Full-time, Part-time, Contract) | VarChar |
| 4 | job_title | Title of the job position | VarChar |
| 5 | salary | Salary amount (in respective currency) | Integer |
| 6 | salary_currency | Currency in which salary is recorded | VarChar |
| 7 | salary_in_usd | Salary amount converted to USD | Integer |
| 8 | employee_residence | Residence location of the employee | Varchar |
| 9 | remote_ratio | Ratio of remote work (0 to 1) | Integer |
| 10 | company_location | Location of the company | VarChar |
| 11 | company_size | Size of the company (Small, Medium, Large) | VarChar |

*Table 1 Dataset Data Type*

# 2. Data Preparation

## Write a python program to load data into pandas DataFrame



```
[1] import pandas as pd
    import numpy as np
```

2. Data Preparation

- Write a python program to load data into pandas DataFrame

```
[4] data = pd.read_csv("Coursework/Coursework.csv")
    dataframe = pd.DataFrame(data)
    dataframe
```

| | work_year | experience_level | employment_type | job_title | salary | salary_currency | salary_in_usd | employee_residence | remote_ratio | company_location | company_size |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 2023 | SE | FT | Principal Data Scientist | 80000 | EUR | 85847 | ES | 100 | ES | L |
| 1 | 2023 | MI | CT | ML Engineer | 30000 | USD | 30000 | US | 100 | US | S |
| 2 | 2023 | MI | CT | ML Engineer | 25500 | USD | 25500 | US | 100 | US | S |
| 3 | 2023 | SE | FT | Data Scientist | 175000 | USD | 175000 | CA | 100 | CA | M |
| 4 | 2023 | SE | FT | Data Scientist | 120000 | USD | 120000 | CA | 100 | CA | M |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 3750 | 2020 | SE | FT | Data Scientist | 412000 | USD | 412000 | US | 100 | US | L |
| 3751 | 2021 | MI | FT | Principal Data Scientist | 151000 | USD | 151000 | US | 100 | US | L |
| 3752 | 2020 | EN | FT | Data Scientist | 105000 | USD | 105000 | US | 100 | US | S |
| 3753 | 2020 | EN | CT | Business Data Analyst | 100000 | USD | 100000 | US | 100 | US | L |
| 3754 | 2021 | SE | FT | Data Science Manager | 7000000 | INR | 94665 | IN | 50 | IN | L |

3755 rows × 11 columns

Next steps: View recommended plots

*Figure 1 python program to load data into pandas DataFrame*

**Explanation**: To load the data into a pandas DataFrame, you can use the pd.read_csv() function. This function reads the data from a CSV file and creates a DataFrame. It's a common way to import tabular data into Python for analysis.

**Output Explanation**: The screenshot shows the code to load the data from a CSV file named "data.csv" into a DataFrame named df. The output of print(df.head()) displays the first few rows of the DataFrame, giving an overview of the data.

## Write a python program to remove unnecessary columns i.e., salary and salary currency.

- Write a python program to remove unnecessary columns i.e., salary and salary currency.

```
df = dataframe.drop(["salary" , "salary_currency"] , axis='columns')
df
```

| | work_year | experience_level | employment_type | job_title | salary_in_usd | employee_residence | remote_ratio | company_location | company_size |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 2023 | SE | FT | Principal Data Scientist | 85847 | ES | 100 | ES | L |
| **1** | 2023 | MI | CT | ML Engineer | 30000 | US | 100 | US | S |
| **2** | 2023 | MI | CT | ML Engineer | 25500 | US | 100 | US | S |
| **3** | 2023 | SE | FT | Data Scientist | 175000 | CA | 100 | CA | M |
| **4** | 2023 | SE | FT | Data Scientist | 120000 | CA | 100 | CA | M |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **3750** | 2020 | SE | FT | Data Scientist | 412000 | US | 100 | US | L |
| **3751** | 2021 | MI | FT | Principal Data Scientist | 151000 | US | 100 | US | L |
| **3752** | 2020 | EN | FT | Data Scientist | 105000 | US | 100 | US | S |
| **3753** | 2020 | EN | CT | Business Data Analyst | 100000 | US | 100 | US | L |
| **3754** | 2021 | SE | FT | Data Science Manager | 94665 | IN | 50 | IN | L |

3755 rows × 9 columns

Next steps: View recommended plots

*Figure 2 remove unnecessary columns from the DataFrame*

**Explanation**: You can remove unnecessary columns from the DataFrame using the drop() function. This function allows you to specify the columns you want to remove by passing their names as a list to the columns parameter.

**Output Explanation**: The screenshot shows the code to remove the 'salary' and 'salary_currency' columns from the DataFrame. The output of print(df.head()) confirms that these columns have been successfully removed from the DataFrame.

## Write a python program to remove the NaN missing values from updated dataframe

- Write a python program to remove the NaN missing values from updated dataframe.

```
df = df.dropna()
df
```

| | work_year | experience_level | employment_type | job_title | salary_in_usd | employee_residence | remote_ratio | company_location | company_size |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 2023 | SE | FT | Principal Data Scientist | 85847 | ES | 100 | ES | L |
| 1 | 2023 | MI | CT | ML Engineer | 30000 | US | 100 | US | S |
| 2 | 2023 | MI | CT | ML Engineer | 25500 | US | 100 | US | S |
| 3 | 2023 | SE | FT | Data Scientist | 175000 | CA | 100 | CA | M |
| 4 | 2023 | SE | FT | Data Scientist | 120000 | CA | 100 | CA | M |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 3750 | 2020 | SE | FT | Data Scientist | 412000 | US | 100 | US | L |
| 3751 | 2021 | MI | FT | Principal Data Scientist | 151000 | US | 100 | US | L |
| 3752 | 2020 | EN | FT | Data Scientist | 105000 | US | 100 | US | S |
| 3753 | 2020 | EN | CT | Business Data Analyst | 100000 | US | 100 | US | L |
| 3754 | 2021 | SE | FT | Data Science Manager | 94665 | IN | 50 | IN | L |

3755 rows × 9 columns

*Figure 3 Remove the NaN missing values from updated dataframe*

**Explanation**: Missing values in the DataFrame can be handled using the dropna() function. This function removes rows or columns that contain missing values (NaN) based on specified parameters.

**Output Explanation**: The screenshot shows the code to handle missing values in the DataFrame by dropping rows with any missing values. The output of print(df.isnull().sum()) confirms that there are no missing values remaining in the DataFrame after handling them.

## Write a python program to check duplicates value in the dataframe.

- Write a python program to remove the NaN missing values from updated dataframe.

```
duplicate = []

for i in range(len(df)):
    if df.iloc[i].tolist() in df.iloc[:i].values.tolist():
        duplicate.append(df.iloc[i])

duplicate_data = pd.DataFrame(duplicate, columns=df.columns)

# Print the duplicate rows
print("Duplicate Rows:")
print(duplicate_data)
```

```
Duplicate Rows:
      work_year          experience_level employment_type  \
115        2023        Senior Level/Expert              FT
123        2023        Senior Level/Expert              FT
153        2023  Medium Level/Intermediate              FT
154        2023  Medium Level/Intermediate              FT
160        2023        Senior Level/Expert              FT
...         ...                        ...             ...
3439       2022  Medium Level/Intermediate              FT
3440       2022        Senior Level/Expert              FT
3441       2022        Senior Level/Expert              FT
3586       2021  Medium Level/Intermediate              FT
3709       2021  Medium Level/Intermediate              FT

               job_title  salary_in_usd employee_residence  remote_ratio  \
115       Data Scientist         150000                 US             0
123   Analytics Engineer         289800                 US             0
153        Data Engineer         100000                 US           100
154        Data Engineer          70000                 US           100
160        Data Engineer         115000                 US             0
...                  ...            ...                ...           ...
3439      Data Scientist          78000                 US           100
3440       Data Engineer         135000                 US           100
3441       Data Engineer         115000                 US           100
3586       Data Engineer         200000                 US           100
3709      Data Scientist          90734                 DE            50

     company_location company_size
115                US            M
123                US            M
153                US            M
154                US            M
160                US            M
...               ...          ...
3439               US            M
3440               US            M
3441               US            M
3586               US            L
3709               DE            L

[1171 rows x 9 columns]
```

*Figure 4 check duplicates value in the dataframe.*

**Explanation**: Duplicate values in the DataFrame can be checked using the duplicated() function. This function returns a boolean Series indicating whether each row is a duplicate of a previous row.

**Output Explanation**: The screenshot shows the code to check for duplicate values in the DataFrame. The output of print(duplicate_rows) displays any duplicate rows found in the DataFrame, if any.

## Write a python program to see the unique values from all the columns in the dataframe

- Write a python program to see the unique values from all the columns in the dataframe.

```
unique_values = {col: df[col].unique() for col in df.columns}
for col, values in unique_values.items():
    print(f"Unique values in column '{col}': {values}")
```

```
Unique values in column 'work_year': [2023 2022 2020 2021]
Unique values in column 'experience_level': ['Senior Level/Expert' 'Medium Level/Intermediate' 'Entry Level'
 'Executive Level']
Unique values in column 'employment_type': ['FT' 'CT' 'FL' 'PT']
Unique values in column 'job_title': ['Principal Data Scientist' 'ML Engineer' 'Data Scientist'
 'Applied Scientist' 'Data Analyst' 'Data Modeler' 'Research Engineer'
 'Analytics Engineer' 'Business Intelligence Engineer'
 'Machine Learning Engineer' 'Data Strategist' 'Data Engineer'
 'Computer Vision Engineer' 'Data Quality Analyst'
 'Compliance Data Analyst' 'Data Architect'
 'Applied Machine Learning Engineer' 'AI Developer' 'Research Scientist'
 'Data Analytics Manager' 'Business Data Analyst' 'Applied Data Scientist'
 'Staff Data Analyst' 'ETL Engineer' 'Data DevOps Engineer' 'Head of Data'
 'Data Science Manager' 'Data Manager' 'Machine Learning Researcher'
 'Big Data Engineer' 'Data Specialist' 'Lead Data Analyst'
 'BI Data Engineer' 'Director of Data Science'
 'Machine Learning Scientist' 'MLOps Engineer' 'AI Scientist'
 'Autonomous Vehicle Technician' 'Applied Machine Learning Scientist'
 'Lead Data Scientist' 'Cloud Database Engineer' 'Financial Data Analyst'
 'Data Infrastructure Engineer' 'Software Data Engineer' 'AI Programmer'
 'Data Operations Engineer' 'BI Developer' 'Data Science Lead'
 'Deep Learning Researcher' 'BI Analyst' 'Data Science Consultant'
 'Data Analytics Specialist' 'Machine Learning Infrastructure Engineer'
 'BI Data Analyst' 'Head of Data Science' 'Insight Analyst'
 'Deep Learning Engineer' 'Machine Learning Software Engineer'
 'Big Data Architect' 'Product Data Analyst'
 'Computer Vision Software Engineer' 'Azure Data Engineer'
 'Marketing Data Engineer' 'Data Analytics Lead' 'Data Lead'
 'Data Science Engineer' 'Machine Learning Research Engineer'
 'NLP Engineer' 'Manager Data Management' 'Machine Learning Developer'
 '3D Computer Vision Researcher' 'Principal Machine Learning Engineer'
 'Data Analytics Engineer' 'Data Analytics Consultant'
 'Data Management Specialist' 'Data Science Tech Lead'
 'Data Scientist Lead' 'Cloud Data Engineer' 'Data Operations Analyst'
 'Marketing Data Analyst' 'Power BI Developer' 'Product Data Scientist'
 'Principal Data Architect' 'Machine Learning Manager'
 'Lead Machine Learning Engineer' 'ETL Developer' 'Cloud Data Architect'
 'Lead Data Engineer' 'Head of Machine Learning' 'Principal Data Analyst'
 'Principal Data Engineer' 'Staff Data Scientist' 'Finance Data Analyst']
Unique values in column 'salary_in_usd': [ 85847  30000  25500 ...  28369 412000  94665]
Unique values in column 'employee_residence': ['ES' 'US' 'CA' 'DE' 'GB' 'NG' 'IN' 'HK' 'PT' 'NL' 'CH' 'CF' 'FR' 'AU'
 'FI' 'UA' 'IE' 'IL' 'GH' 'AT' 'CO' 'SG' 'SE' 'SI' 'MX' 'UZ' 'BR' 'TH'
 'HR' 'PL' 'KW' 'VN' 'CY' 'AR' 'AM' 'BA' 'KE' 'GR' 'MK' 'LV' 'RO' 'PK'
 'IT' 'MA' 'LT' 'BE' 'AS' 'IR' 'HU' 'SK' 'CN' 'CZ' 'CR' 'TR' 'CL' 'PR'
 'DK' 'BO' 'PH' 'DO' 'EG' 'ID' 'AE' 'MY' 'JP' 'EE' 'HN' 'TN' 'RU' 'DZ'
 'IQ' 'BG' 'JE' 'RS' 'NZ' 'MD' 'LU' 'MT']
Unique values in column 'remote_ratio': [100   0  50]
Unique values in column 'company_location': ['ES' 'US' 'CA' 'DE' 'GB' 'NG' 'IN' 'HK' 'NL' 'CH' 'CF' 'FR' 'FI' 'UA'
 'IE' 'IL' 'GH' 'CO' 'SG' 'AU' 'SE' 'SI' 'MX' 'BR' 'PT' 'RU' 'TH' 'HR'
 'VN' 'EE' 'AM' 'BA' 'KE' 'GR' 'MK' 'LV' 'RO' 'PK' 'IT' 'MA' 'PL' 'AL'
 'AR' 'LT' 'AS' 'CR' 'IR' 'BS' 'HU' 'AT' 'SK' 'CZ' 'TR' 'PR' 'DK' 'BO'
 'PH' 'BE' 'ID' 'EG' 'AE' 'LU' 'MY' 'HN' 'JP' 'DZ' 'IQ' 'CN' 'NZ' 'CL'
 'MD' 'MT']
Unique values in column 'company_size': ['L' 'S' 'M']
```

*Figure 5 unique values from all the columns in the dataframe*

**Explanation**: You can display unique values from all columns in the DataFrame using a loop to iterate over each column and the unique() function to get the unique values for each column.

**Output Explanation**: The screenshot shows the code to display unique values from all columns in the DataFrame. The output prints the unique values for each column, providing insights into the distinct categories or values present in the data.

### Rename the experience level columns as below.

SE – Senior Level/Expert

MI – Medium Level/Intermediate

EN – Entry Level

EX – Executive Level

- Rename the experience level columns as below.

SE – Senior Level/Expert

MI – Medium Level/Intermediate

EN – Entry Level

EX – Executive Level

```
New_Experience_Levels = {'SE': 'Senior Level/Expert','MI': 'Medium Level/Intermediate','EN': 'Entry Level','EX': 'Executive Level'
}
df['experience_level'] = df['experience_level'].replace(New_Experience_Levels)
df['experience_level']
```

```
0             Senior Level/Expert
1       Medium Level/Intermediate
2       Medium Level/Intermediate
3             Senior Level/Expert
4             Senior Level/Expert
                  ...
3750          Senior Level/Expert
3751    Medium Level/Intermediate
3752                  Entry Level
3753                  Entry Level
3754          Senior Level/Expert
Name: experience_level, Length: 3755, dtype: object
```

*Figure 6 Rename the experience level*

A dictionary named New_Experience_Levels is created, where keys represent the original experience level codes ('SE', 'MI', 'EN', 'EX'), and values represent the corresponding new experience level names ('Senior Level/Expert', 'Medium Level/Intermediate', 'Entry Level', 'Executive Level').

The replace() method is then used on the 'experience_level' column of the DataFrame (df['experience_level']) to replace the original experience level codes with the new experience level names based on the mapping provided in New_Experience_Levels.

The output of df['experience_level'] displays the updated 'experience_level' column, showing the new experience level names for each entry in the DataFrame.

## 3. Data Analysis the graph.

● Write a Python program to show summary statistics of sum, mean, standard deviation, skewness, and kurtosis of any chosen variable.

3. Data Analysis the graph.

• Write a Python program to show summary statistics of sum, mean, standard deviation, skewness, and kurtosis of any chosen variable.

```
# Define the variable of interest
variable = "salary_in_usd"

# Initialize sum_value to store the sum of 'salary_in_usd'
sum_value = 0

# Iterate through each value in the 'salary_in_usd' column and calculate the sum
for value in df[variable]:
    sum_value += value

# Convert sum_value to string and print the total sum of 'salary_in_usd'
total = str(sum_value)
print(f"Sum of Salary In USD is: " +total)
```

```
Sum of Salary In USD is: 516576814
```

```
[49] # Define the variable of interest
variable = "salary_in_usd"

# Initialize total_sum and count to calculate the sum and count of 'salary_in_usd'
total_sum = 0
count = 0

# Iterate through each value in the 'salary_in_usd' column and calculate the sum and count
for value in df[variable]:
    total_sum += value
    count += 1

# Calculate the mean by dividing the total sum by the count and print the average salary in USD
mean_value = total_sum / count
print(f"Mean of Salary In USD is: {mean_value}")
```

```
Mean of Salary In USD is: 137570.38988015978
```

```
[50] # Define the variable of interest
variable = "salary_in_usd"

# Extract values from the 'salary_in_usd' column
values = df[variable]

# Calculate the mean and number of values
mean = values.mean()
n = len(values)

# Calculate the variance, standard deviation, and print the standard deviation
squared = (values - mean) ** 2
variance = squared.sum() / n
std = np.sqrt(variance)
print(f"Standard Deviation: {std}")
```

```
Standard Deviation: 63047.228497405435
```

*Figure 7 summary statistics of sum, mean, standard deviation*

```
[51] # Define the variable of interest
     variable_of_interest = "salary_in_usd"

     # Extract values from the 'salary_in_usd' column
     values = df[variable_of_interest]

     # Calculate the mean, standard deviation, and skewness
     mean_value = values.mean()
     n = len(values)
     std_deviation = values.std()
     skewness = ((values - mean_value) ** 3).sum() / (n * std_deviation ** 3)

     # Print the skewness of salaries in USD
     print(f"Skewness: {skewness}")
```

```
Skewness: 0.5359726925230353
```

```
[54] # Define the variable of interest
     variable_of_interest = "salary_in_usd"

     # Extract values from the 'salary_in_usd' column
     values = df[variable_of_interest]

     # Calculate the mean, standard deviation, and kurtosis
     mean_value = values.mean()
     n = len(values)
     std_deviation = values.std()
     kurtosis = ((values - mean_value) ** 4).sum() / (n * std_deviation ** 4) - 3

     # Print the kurtosis of salaries in USD
     print(f"Kurtosis: {kurtosis}")
```

```
Kurtosis: 0.8292585346115859
```

- Write a Python program to calculate and show correlation of all variables.a

```
[33] numeric_values = df.select_dtypes(include = ['int'])
     correlation = numeric_values.corr()
     correlation
```

| | work_year | salary_in_usd | remote_ratio |
|---|---|---|---|
| **work_year** | 1.00000 | 0.228290 | -0.236430 |
| **salary_in_usd** | 0.22829 | 1.000000 | -0.064171 |
| **remote_ratio** | -0.23643 | -0.064171 | 1.000000 |

Next steps: View recommended plots

*Figure 8 summary statistics of skewness, and kurtosis of any chosen variable*

- **Sum** of 'salary_in_usd': The sum calculation provides the total monetary value of salaries in USD across all entries in the dataset. It gives an overview of the total expenditure on salaries.

- **Mean** (average) of 'salary_in_usd': The mean calculation provides the average salary in USD by summing up all salaries and dividing by the total count of entries. It represents the typical salary level in the dataset.

- **Standard deviation** of 'salary_in_usd': The standard deviation calculation measures the dispersion or variability of salaries around the mean value. A higher standard deviation indicates greater variability in salaries.

- **Skewness** of 'salary_in_usd': Skewness measures the asymmetry of the distribution of salaries. A positive skewness value indicates a right-skewed distribution where the tail is longer on the right side, while a negative skewness value indicates a left-skewed distribution.

- **Kurtosis** of 'salary_in_usd': Kurtosis measures the tailedness or peakedness of the distribution of salaries. Positive kurtosis indicates heavier tails and a sharper peak (leptokurtic distribution), while negative kurtosis indicates lighter tails and a flatter peak (platykurtic distribution).

- **Correlation** between numeric variables: Correlation analysis examines the strength and direction of the linear relationship between pairs of numeric variables in the dataset. The correlation coefficient ranges from -1 to 1, where values closer to 1 or -1 indicate a stronger positive or negative correlation, respectively, while values closer to 0 indicate weaker or no correlation.

# 4. Data Exploration

## Write a python program to find out top 15 jobs. Make a bar graph of sales as well.

4. Data Exploration

- Write a python program to find out top 15 jobs. Make a bar graph of sales as well.

```python
# Count the frequency of each job title and select the top 15
Top_jobs = df['job_title'].value_counts().head(15)

# Create a bar plot for the top 15 jobs
plt.bar(Top_jobs.index, Top_jobs.values)
plt.title('Top 15 Jobs')
plt.xlabel('Job Title')
plt.ylabel('Frequency')
plt.xticks(rotation=35, ha='right')  # Rotate x-axis labels for better readability
plt.tight_layout()  # Adjust layout to prevent overlap
plt.show()
```

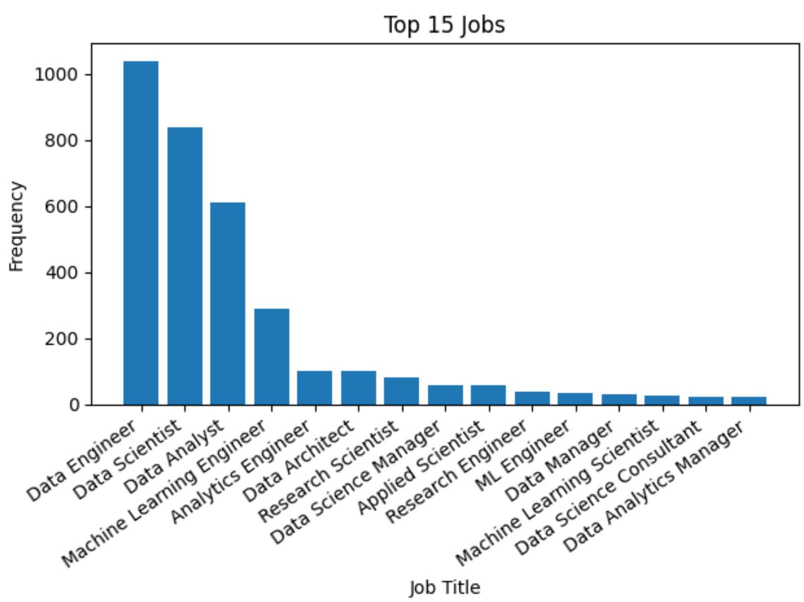



*Figure 9 Figure 9 top 15 jobs.*

- The code counts the frequency of each job title using the value_counts() method and selects the top 15 most frequent job titles.
- It then creates a bar plot where each bar represents a job title and its height represents the frequency of that job title in the dataset.
- The x-axis represents the job titles, and the y-axis represents the frequency of each job title.
- The rotation=35 parameter rotates the x-axis labels by 35 degrees for better readability.

## Which job has the highest salaries? Illustrate with bar graph.

- Which job has the highest salaries? Illustrate with bar graph.

```python
# Group the dataset by job title and find the maximum salary for each job
max_salary_by_job = df.groupby('job_title')['salary_in_usd'].max()

# Sort the maximum salaries in descending order and select the top 20
topsalaries = max_salary_by_job.sort_values(ascending=False).head(20)

# Create a bar plot for the top 10 highest salary jobs
plt.figure(figsize=(8, 6))
plt.bar(topsalaries.index, topsalaries.values, color='lightgrey')
plt.title('Top 10 Highest Salary Jobs')
plt.xlabel('Job Title')
plt.ylabel('Salary (USD)')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```

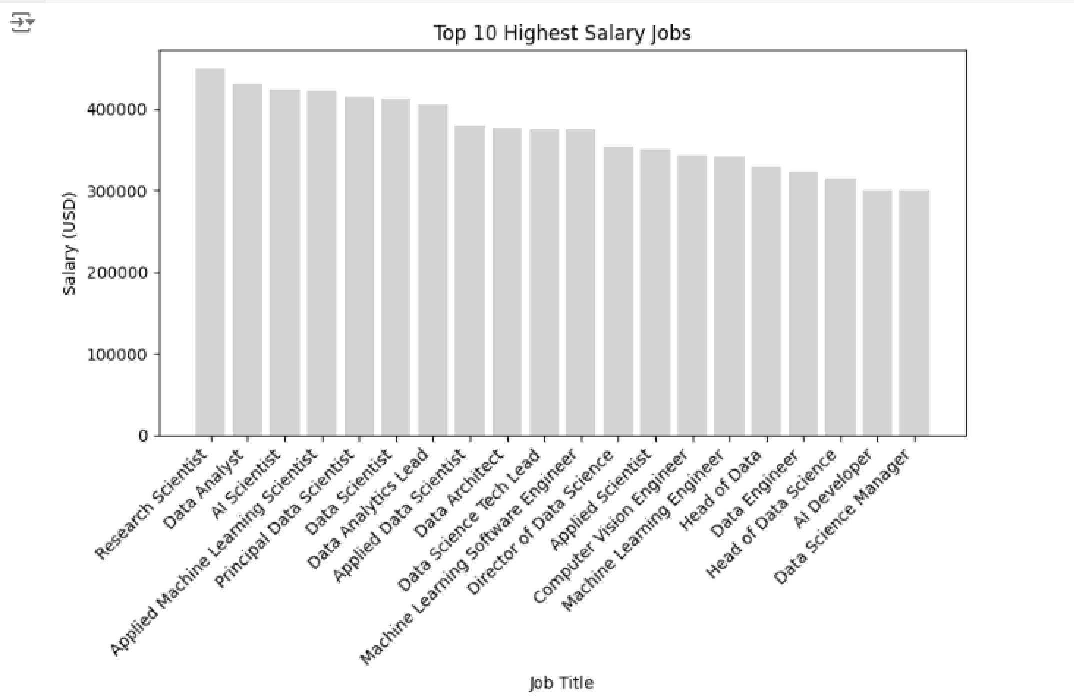



*Figure 10 Highest Salary Jobs*

- The code groups the dataset by job title and finds the maximum salary for each job title.
- It sorts the maximum salaries in descending order and selects the top 20 highest salaries.
- It creates a bar plot where each bar represents a job title and its height represents the highest salary associated with that job title.
- The x-axis represents the job titles, and the y-axis represents the highest salary (in USD) for each job title.
- The rotation=45 parameter rotates the x-axis labels by 45 degrees for better readability.

## Write a python program to find out c. Illustrate it through bar graph.

- Write a python program to find out salaries based on experience level. Illustrate it through bar graph.

```python
# Group the dataset by experience level and calculate the mean salary for each level
salary_by_experience = df.groupby('experience_level')['salary_in_usd'].mean()

# Create a bar plot for average salaries based on experience level
plt.figure(figsize=(3, 6))
plt.bar(salary_by_experience.index, salary_by_experience.values)
plt.title('Salary by Experience Level')
plt.xlabel('Experience Level')
plt.ylabel('Average Salary (USD)')
plt.xticks(rotation=80)  # Rotate x-axis labels for better readability
plt.tight_layout()
plt.show()
```

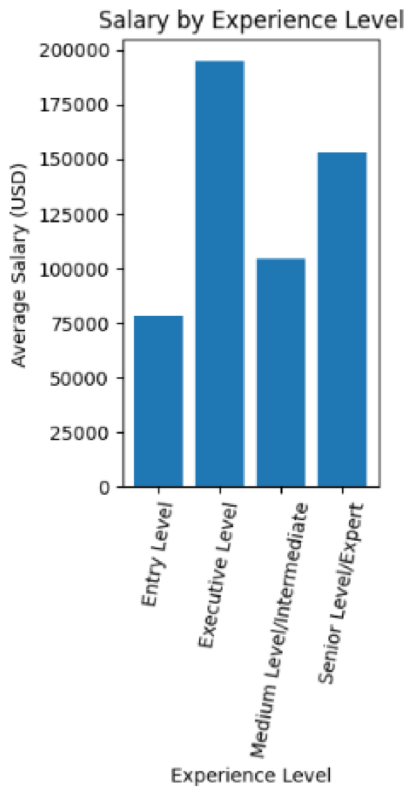



*Figure 11 Salary by Experience Level*

- The code groups the dataset by experience level and calculates the average salary for each experience level.
- It creates a bar plot where each bar represents an experience level and its height represents the average salary for that experience level.
- The x-axis represents the experience levels, and the y-axis represents the average salary (in USD) for each experience level.
- The rotation=80 parameter rotates the x-axis labels by 80 degrees for better readability.

## Write a Python program to show histogram and box plot of any chosen different variables. Use proper labels in the graph

- Write a Python program to show histogram and box plot of any chosen different variables. Use proper labels in the graph.

```python
[59] # Select top and bottom 15 salaries for comparison
TopSalary = df['salary_in_usd'].head(15)
BottomSalary = df['salary_in_usd'].tail(15)

# Plot histogram to visualize distribution of salaries
plt.figure(figsize=(10, 6))
plt.hist(TopSalary, bins=20, color='blue', alpha=0.7, label='Top Salary')
plt.hist(BottomSalary, bins=20, color='salmon', alpha=0.7, label='Bottom Salary')
plt.title('Histogram of Top and Bottom Salaries')
plt.xlabel('Salary (USD)')
plt.ylabel('Frequency')
plt.legend()  # Add legend to differentiate between top and bottom salaries
plt.grid(True)  # Add grid for better visualization
plt.show()
```

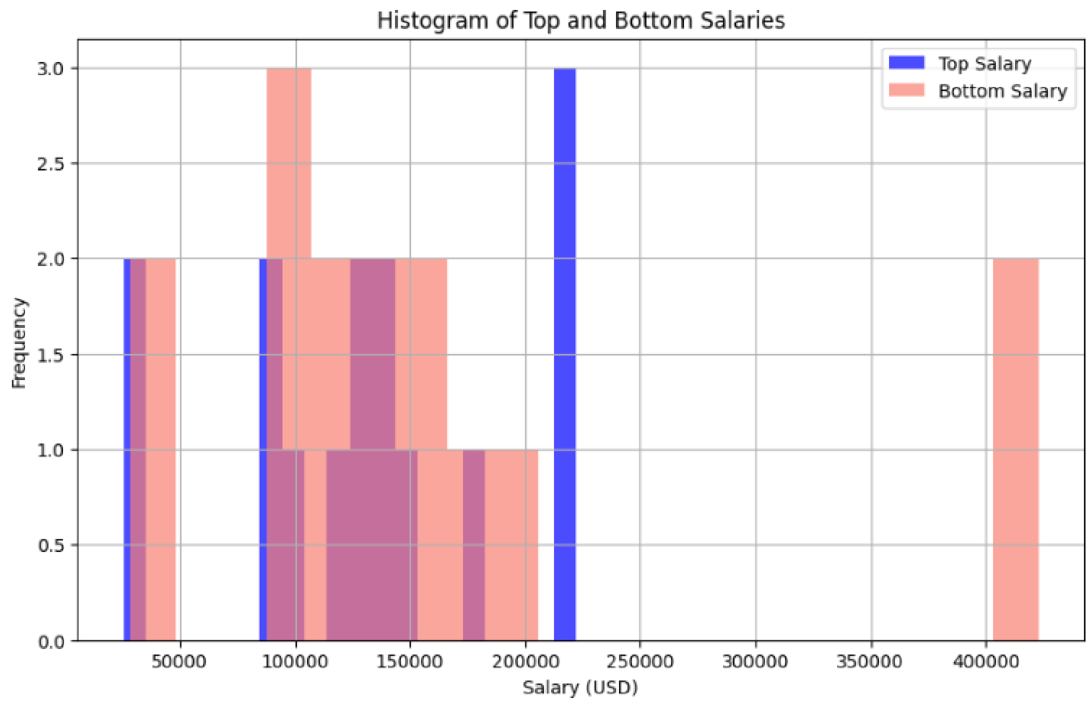



*Figure 12 Plotting Histogram of Top and Bottom Salaries*

- The code selects the top and bottom 15 salaries from the dataset.
- It plots a histogram to visualize the distribution of salaries, with separate histograms for the top and bottom salaries.
- The blue histogram represents the distribution of the top 15 salaries, and the salmon histogram represents the distribution of the bottom 15 salaries.
- The x-axis represents the salary range, and the y-axis represents the frequency of salaries in each range.
- The legend distinguishes between the top and bottom salaries, and grid lines are added for better visualization.

```
# Select top 15 salaries for box plot visualization
Salary = df['salary_in_usd'].head(15)

# Plot box plot to visualize distribution and identify outliers
plt.subplot(1, 2, 2)  # Create subplot for box plot
plt.boxplot(Salary)
plt.title('Boxplot of Salary')
plt.ylabel('Salary (USD)')
plt.xticks([1], [''])  # Hide x-axis label
plt.tight_layout()  # Adjust layout to prevent overlap
plt.show()
```

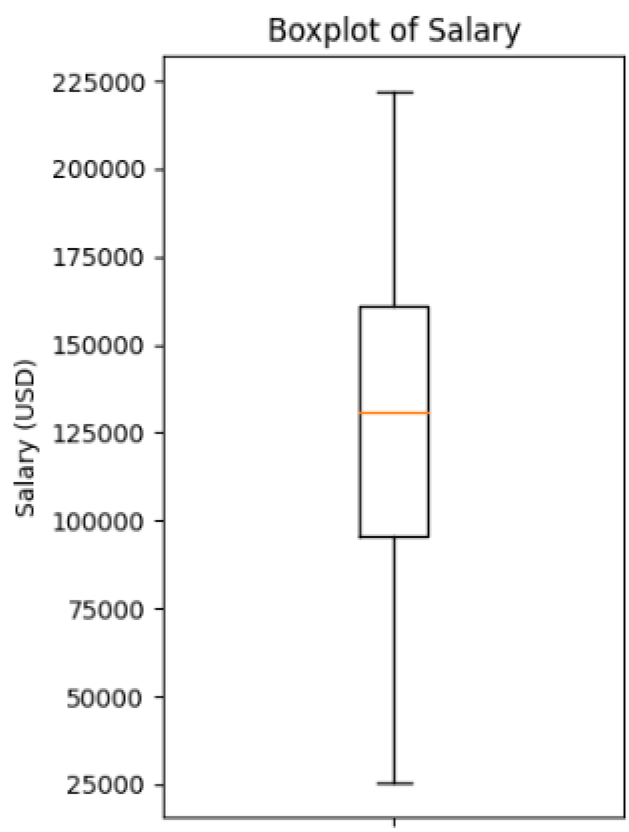



*Figure 13 Plotting Boxplot of Salary*

- The code selects the top 15 salaries from the dataset for box plot visualization.
- It plots a box plot to visualize the distribution of salaries and identify outliers.
- The box plot displays the minimum, first quartile (25th percentile), median (50th percentile), third quartile (75th percentile), and maximum values of the salary distribution.
- The y-axis represents the salary range, and no x-axis label is shown for better visualization.