LONDON METROPOLITAN UNIVERSITY

islington college
(इस्लिङ्टन कलेज)

CC5061NI Applied Data Science

60% Individual Coursework

2023-2024

Student Name: Aayesha Nakarmi

London Met ID: 22067189

College ID: NP01AI4A220011

Group: L1AI1

Assignment Due Date: Monday, May 13, 2024

Assignment Submission Date: Monday, May 13, 2024

Word Count: 3513

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 mark of zero will be awarded

Table of Contents

Table of Figures

Table of Tables

1 Introduction

The objective of this project is to analyze the salaries of individuals in the Artificial Intelligence field and identify the factors that influence their salaries. The dataset used for this analysis contains various factors such as experience, work level, job title, and many more. The primary aim of this project is to prepare the data for further data mining and analysis.

Aim

To analyse the salaries of individuals in the Artificial Intelligence field and identify the factors that influence their salaries using a dataset containing various factors such as experience, job title, employment types, different companies and locations.

Objectives

- To understand the characteristics of the dataset and identify the factors that influence the salaries of data scientists.
- To prepare the data for further analysis by removing unnecessary columns, handling missing values, and checking for duplicates.
- To perform exploratory data analysis and statistical analysis to identify patterns and correlations among different variables
- To explore the relationships between job titles, experience levels, and salaries through data visualization techniques.
- To present the findings and insights in a well-structured technical report

Tools and libraries used

- Python: The chosen programming language for data processing, analysis, and visualization.
- Pandas library: For data manipulation and analysis.
- NumPy library: For numerical computations and array operations.
- Matplotlib and seaborn library: For generating various types of visualization
- Jupyter Lab: For writing and executing Python code

2 Data Understanding

The dataset contains details about jobs, salaries, and work arrangements. It can be useful compare salaries for different job titles, experience levels, and employment types, as well as for different companies and locations.

The dataset is in a CSV(Comma Separated Values) format and has 3755 rows and 11 columns altogether. It contains the following columns: work_year, experience_level, employment_type, job_title, salary_currency, employee_residence, remote_ratio, company_location, and company_size. There are no missing values in any of the columns.

2.1 Categorical and Continuous Variables

2.1.1 Categorical Variables

- work_year: Discrete time intervals (years)
- experience_level: Senior (SE), Mid-level (MI), Entry level (EN)
- employment_type: Full-Time (FT), Part-time (PT), Contract (CT), Freelance (FL)
- job_title: 93 unique job titles
- salary_currency: 20 different currencies
- employee_residence: 78 different standardized country codes. For example, 'ES' for Spain, 'US' for the United States, 'CA' for Canada, etc.
- remote_ratio: 0, 50, 100
- company_location: 72 different standardized country codes. For example, 'ES' for Spain, 'US' for the United States, 'CA' for Canada, etc.
- company_size: Large (L), Medium (M), Small (S)

2.1.2 Continuous Variables

- salary: Ranges from 6000 to 30400000
- salary_in_usd: Ranges from 5132$ to 450000$

2.2 Column description

S.No.	Column name	Description	Data type
1	work_year	The year in which the employee's job and salaries details were recorded. This column allows for analysing trends and changes in salaries over different years. The dataset contains salary and job data of employee from 2020 to 2023.	Integer
2	experience_level	It indicates the professional experience level of the employee in their respective role. It categorizes employees into different levels of experience, It is valuable for understanding how experience impacts salary and job roles. Experience level is categorised into SE(Senior), MI(Mid-level), and EN(Entry level).	String
3	employment_type	Employment type refers to the nature of an individual's relationship with their employer or client. It describes the terms and conditions of their work, including the number of hours worked, the duration of the employment, and the benefits provided. This column provides information about potential differences in salary and job roles based on the type of employment.	String

		Employment types included in the given dataset are FT(Full-Time), PT(Part-time), CT(Contract), FL(Freelance).	
4	job_title	The specific job title held by the employee within the organization, such as Data Scientist, Machine Learning Engineer, or Data Analyst. This column facilitates analysing compensation levels and other factors across different job roles. There are 93 unique job titles within the given dataset. There is presence of some inconsistent data as	String
5	Salary	This column indicates the salary earned by each employee in their respective positions. It is redundant as there is already a column called "salary_in_usd" that provides the same information in a standardized format. Moreover, it can lead to inconsistencies due to the use of different currencies. The salary values range from 6000 to 30400000.	Integer
6	salary_currency	It indicates the currency of the employee's salary, but it is not needed for comparing salaries since "salary_in_usd" column already provides salaries in US Dollars. The dataset has already converted salaries to US Dollars for easy comparison.	String

		The dataset has 20 different currencies values such as USD (US Dollars), EUR (Euros), GBP(Great British Pound) or INR (Indian Rupees), etc.	
7	salary_in_usd	The employee's salary converted to its equivalent value in US Dollars (USD), allowing for easy comparison across job titles, experience level, and employment type, making it simple to analyze salary trends and patterns. The values range from 5132$ to 450000$.	Integer
8	employee_residence	The country where the employee resides, represented by a standardized country code (e.g., US for United States, CA for Canada, ES for Spain). It is useful for analysing salary patterns across different geographical regions. The dataset has 78 different standardized country codes for employee_residence.	String
9	remote_ratio	The ratio or percentage of the employee's work that is performed remotely, ranging from 0 (fully on-site), 50(partially remote) and 100 (fully remote). This column provides about the impact of remote work arrangements on salary and job roles within the organization.	Integer
10	company_location	The country or region where the company is headquartered, represented by a standardized country code (e.g., US for United States, DE for Germany, GB for	String

		United Kingdom). This information can be valuable for analysing salary patterns and in different geographical locations. There are 72 different standardized country codes for company_location in the dataset.	
11	company_size	The size of the company, categorized into levels such as Large (L), Medium (M), or Small (S). It can be used to explore the potential impact of company size on salary levels and job roles.	Character

Table 1 Column description

3 Data Preparation

3.1 Loading data into pandas DataFrame

Write a python program to load data into pandas DataFrame

```
# Reading the CSV file "DataScienceSalaries.csv" into a pandas DataFrame named salary_df
salary_df=pd.read_csv("DataScienceSalaries.csv")
salary_df
```

Figure 1 Loading data into pandas DataFrame

- The code reads data from a CSV file named "DataScienceSalaries.csv" into a pandas DataFrame called salary_df.
- The pd.read_csv() function is a pandas function used to read CSV files and convert them into DataFrame objects.
- A DataFrame is a two-dimensional labeled data structure with rows and columns, like a spreadsheet or SQL table.
- After executing this line of code, the salary_df DataFrame contains the data from the CSV file.
  (datacamp, 2023)

	work_year	experience_level	employment_type	job_title	salary	salary_currency	salary_in_usd	employee_residence	remote_ratio	company_location
0	2023	SE	FT	Principal Data Scientist	80000	EUR	85847	ES	100	ES
1	2023	MI	CT	ML Engineer	30000	USD	30000	US	100	US
2	2023	MI	CT	ML Engineer	25500	USD	25500	US	100	US
3	2023	SE	FT	Data Scientist	175000	USD	175000	CA	100	CA
4	2023	SE	FT	Data Scientist	120000	USD	120000	CA	100	CA
...	...	...	...	...	...	...	...	...	...	...
3750	2020	SE	FT	Data Scientist	412000	USD	412000	US	100	US
3751	2021	MI	FT	Principal Data Scientist	151000	USD	151000	US	100	US

Figure 2 Output after loading data

The output displays the first few rows of the DataFrame with the specified columns. Each row represents a record of salary data for an employee, including their work year, experience level, employment type, job title, salary information, residence, remote work ratio, company location, and company size.

3.2 Standardizing job titles to remove inconsistent data

Standardizing Data Scientist roles

```
# Getting unique job titles containing 'data scientist' (case-sensitive) in salary_df
salary_df['job_title'][salary_df['job_title'].str.contains('data scientist', case=False)].unique()
```

```
array(['Principal Data Scientist', 'Data Scientist',
       'Applied Data Scientist', 'Lead Data Scientist',
       'Data Scientist Lead', 'Product Data Scientist',
       'Staff Data Scientist'], dtype=object)
```

Figure 3 Filtering data scientist job title

This code filters the 'job_title' column of the 'salary_df' DataFrame to only include rows where the job title contains the string 'data scientist' (case-insensitive), and then returns the unique job titles from the filtered result. (SaturnCloud, 2023)

```
# Grouping specific data scientist job titles under 'Data Scientist'
replace_data_scientist_data={
    'Principal Data Scientist':'Data Scientist',
    'Applied Data Scientist':'Data Scientist',
    'Data Scientist Lead':'Data Scientist',
    'Lead Data Scientist':'Data Scientist',
    'Product Data Scientist':'Data Scientist',
    'Staff Data Scientist':'Data Scientist',
    'Lead Data Scientist':'Data Scientist'
}

# Using dictionary to replace data scientist job titles with 'Data Scientist'
salary_df['job_title']=salary_df['job_title'].replace(replace_data_scientist_data)
```

Figure 4 Standardizing data scientist job title

The code first creates a dictionary replace_data_scientist_data that maps specific data scientist job titles such as 'Principal Data Scientist', 'Applied Data Scientist', 'Data Scientist Lead', 'Lead Data Scientist', 'Product Data Scientist', and 'Staff Data Scientist' to the title 'Data Scientist'.

Then, the code replaces the job titles in the 'job_title' column of the 'salary_df' DataFrame with the corresponding values from the replace_data_scientist_data dictionary using the replace()'. (stack overflow, 2019)

All other inconsistent data has been converted to consistent data in a similar manner.

```
# Grouping specific data engineer job titles under 'Data Engineer'
replace_data_engineer_data={
    'Azure Data Engineer':'Data Engineer',
    'BI Data Engineer':'Data Engineer',
    'Big Data Engineer':'Data Engineer',
    'Cloud Data Engineer':'Data Engineer',
    'Lead Data Engineer':'Data Engineer',
    'Marketing Data Engineer':'Data Engineer',
    'Principal Data Engineer':'Data Engineer',
    'Software Data Engineer':'Data Engineer',
    'Data Operations Engineer':'Data Engineer'
}

# Using dictionary to replace data engineer job titles with 'Data Engineer'
salary_df['job_title']=salary_df['job_title'].replace(replace_data_engineer_data)
```

Figure 5 Standardizing data engineer job title

```
# Grouping specific data engineer job titles under 'Data Analyst'
replace_analyst_data={
    'Compliance Data Analyst': 'Data Analyst',
    'Business Data Analyst':'Data Analyst',
    'Staff Data Analyst' : 'Data Analyst',
    'Lead Data Analyst': 'Data Analyst',
    'Financial Data Analyst': 'Data Analyst',
    'BI Analyst': 'Data Analyst',
    'BI Data Analyst':'Data Analyst',
    'Finance Data Analyst': 'Data Analyst',
    'Marketing Data Analyst': 'Data Analyst',
    'Product Data Analyst' : 'Data Analyst',
}

# Using dictionary to replace job titles with 'Data Analyst'
salary_df['job_title']=salary_df['job_title'].replace(replace_analyst_data)
```

Figure 6 Standardizing data analyst job title

```
# Grouping specific data engineer job titles under 'Machine Learning Engineer'
replace_machine_learning_data={
    'ML Engineer':'Machine Learning Engineer',
    'Applied Machine Learning Engineer':'Machine Learning Engineer',
    'Applied Machine Learning Scientist':'Machine Learning Scientist',
    'Machine Learning Infrastructure Engineer':'Machine Learning Engineer',
    'Machine Learning Research Engineer': 'Machine Learning Engineer',
    'Machine Learning Software Engineer':'Machine Learning Engineer',
    'Principal Machine Learning Engineer':'Machine Learning Engineer',
    'Lead Machine Learning Engineer':'Machine Learning Engineer'
}

# Using dictionary to replace job titles with 'Machine Learning Engineer'
salary_df['job_title']=salary_df['job_title'].replace(replace_machine_learning_data)
```

Figure 7 Standardizing machine learning engineer job title

```
# Grouping specific data engineer job titles under 'Data Manager'
replace_manager_data = {
    'Manager Data Management': 'Data Manager',
    'Data Management Specialist': 'Data Manager'
}
# Using dictionary to replace job titles with 'Data Manager'
salary_df['job_title']=salary_df['job_title'].replace(replace_manager_data)
```

Figure 8 Standardizing data manager job title

```
# Grouping specific data research job titles under 'Researcher'
replace_research_data = {
    'Research Engineer': 'Researcher',
    'Research Scientist': 'Researcher',
    'Machine Learning Research Engineer': 'Machine Learning Researcher',
}

# Using dictionary to replace job titles with 'Researcher'
salary_df['job_title']=salary_df['job_title'].replace(replace_research_data)
```

Figure 9 Standardizing researcher job title

```
# Grouping specific data engineer job titles under 'Data Consultant'
replace_consultant_data = {
    'Data Science Consultant': 'Data Consultant',
    'Data Analytics Consultant': 'Data Consultant'
}

# Using dictionary to replace job titles with 'Data Consultant'
salary_df['job_title']=salary_df['job_title'].replace(replace_consultant_data)
```

Figure 10 Standardizing data consultant job title

```
# Standardizing short forms to full form using dictionary
replace_data = {
    'AI Programmer': 'Artificial Intelligence Programmer',
    'AI Developer':'Artificial Intelligence Developer',
    'AI Scientist':'Artificial Intelligence Scientist',
    'BI Developer': 'Business Intelligence Developer',
    'Machine Learning Research Engineer': 'Machine Learning Researcher',
}
salary_df['job_title']=salary_df['job_title'].replace(replace_data)
```

Figure 11 Standardizing full forms in job title

```
# Standardizing seniority level using dictionary
replace_lead_data ={
    'Data Science Tech Lead':'Data Science Lead',
    'Head of Data':'Data Lead',
    'Head of Data Science':'Data Science Lead',
    'Head of Machine Learning': 'Machine Learning Lead'
}
salary_df['job_title']=salary_df['job_title'].replace(replace_lead_data)
```

Figure 12 Standardizing job leads

```
salary_df['job_title'].nunique()
```

```
50
```

Figure 13 Reduced unique job title

3.3 Removing unnecessary columns

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

```
# Dropping columns 'salary' and 'salary_currency' from the DataFrame 'salary_df'
# axis=1 indicates that columns will be dropped
salary_df=salary_df.drop(['salary','salary_currency'],axis=1)
salary_df
```

Figure 14 Dropping 'salary' and 'salary_currency' column

The code drops the columns 'salary' and 'salary_currency' from the DataFrame salary_df using the drop() method along the column axis (axis=1). The resulting DataFrame contains all the other columns except for the dropped ones.

	work_year	experience_level	employment_type	job_title	salary_in_usd	employee_residence	remote_ratio	company_location	company_size
0	2023	SE	FT	Data Scientist	85847	ES	100	ES	L
1	2023	MI	CT	Machine Learning Engineer	30000	US	100	US	S
2	2023	MI	CT	Machine Learning 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	Data Scientist	151000	US	100	US	L
3752	2020	EN	FT	Data Scientist	105000	US	100	US	S
3753	2020	EN	CT	Data Analyst	100000	US	100	US	L
3754	2021	SE	FT	Data Science Manager	94665	IN	50	IN	L

3755 rows × 9 columns

Figure 15 DataFrame after dropping columns

The salary and salary_currency columns are dropped from the dataset, as the salary_in_usd column already contains the standardized salary information required for analysis. Keeping the original salary values with mixed currency units is redundant and can lead to inconsistencies.

3.4 Removing NaN missing values

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

```
# Removing rows containing any null values from the DataFrame 'salary_df'
salary_df=salary_df.dropna()
```

Figure 16 Dropping columns with NaN missing values

The code eliminates rows with any missing or null values (NaN) from the DataFrame salary_df. This ensures that the dataset used for analysis does not contain any incomplete or missing information, which could potentially affect the accuracy of the results.

```
# Calculating the number of null values in each column of the DataFrame 'salary_df'
salary_df.isnull().sum()
```

```
work_year           0
experience_level    0
employment_type     0
job_title           0
salary_in_usd       0
employee_residence  0
remote_ratio        0
company_location    0
company_size        0
dtype: int64
```

Figure 17 Number of null values in columns

After removing rows with missing values, the DataFrame salary_df contains only complete data, ensuring that all rows have values for every column.

3.5 Checking for duplicate values

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

```
# Calculating the number of duplicate rows in the DataFrame 'salary_df'
salary_df.duplicated().sum()
```

```
1192
```

Figure 18 Getting count of duplicate values

- The code salary_df.duplicated().sum() calculates the number of duplicate rows in the DataFrame salary_df using the duplicated() method from the pandas library.
- The duplicated() method returns a Series that contains True for each duplicate row and False for each unique row,
- The sum() function then adds up all the True values in the Series, which gives the total number of duplicate rows.

Therefore, the code salary_df.duplicated().sum() returns the number of duplicate rows in the DataFrame salary_df. (w3schools, 2024)

```
# Selecting duplicate rows from the salary_df dataframe
salary_df[salary_df.duplicated()]
```

	work_year	experience_level	employment_type	job_title	salary_in_usd	employee_residence	remote_ratio	company_location	company_size
115	2023	SE	FT	Data Scientist	150000	US	0	US	M
123	2023	SE	FT	Analytics Engineer	289800	US	0	US	M
153	2023	MI	FT	Data Engineer	100000	US	100	US	M
154	2023	MI	FT	Data Engineer	70000	US	100	US	M
160	2023	SE	FT	Data Engineer	115000	US	0	US	M
...	...	...	...	...	...	...	...	...	...
3439	2022	MI	FT	Data Scientist	78000	US	100	US	M
3440	2022	SE	FT	Data Engineer	135000	US	100	US	M
3441	2022	SE	FT	Data Engineer	115000	US	100	US	M
3586	2021	MI	FT	Data Engineer	200000	US	100	US	L
3709	2021	MI	FT	Data Scientist	90734	DE	50	DE	L

1192 rows × 9 columns

Figure 19 Viewing duplicate values

The code uses the boolean Series(duplicated()) as a filter to select only the rows from salary_df where the value in the duplicated() Series is True.

```
# Removing duplicate rows from the 'salary_df' DataFrame while keeping the first occurence of unique row
salary_df = salary_df.drop_duplicates(keep='first')
```

```
# Calculating the number of duplicate rows in the salary_df dataframe
salary_df.duplicated().sum()
```

```
0
```

Figure 20 Removing duplicate rows from DataFrame

- The drop_duplicates() method is a pandas function that removes duplicate rows from a DataFrame.
- The keep parameter specifies which duplicate to keep. In this case, keep='first' keeps the first occurrence of each unique row, and removes any identical rows that appear later in the DataFrame.

The resulting DataFrame salary_df contains only the unique rows from the original DataFrame, keeping only the first occurrence of each unique row. (pandas, 2024)

3.6 Viewing unique values of all columns

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

```
# Iterating over each column in the DataFrame 'salary_df'
for column in salary_df.columns:
    # Printing column name
    print(column)
    # Printing unique values of the current column
    print(salary_df[column].unique())
    print()
```

Figure 21 Getting unique values of all columns

- This code iterates over each column in the DataFrame salary_df using a for loop.
- For each column, it prints the column name using the print() function.
- It then prints the unique values in the column using the unique() function, which returns an array of the unique values in the column.

Finally, it prints an empty line using the print() function to separate the output for each column. (Stack overflow, 2015)

```
work_year
[2023 2022 2020 2021]

experience_level
['Senior Level/Expert' 'Medium Level/Intermediate' 'Entry Level'
 'Executive Level']

employment_type
['FT' 'CT' 'FL' 'PT']
```

Figure 22 Unique values of columns(1)

```
job_title
['Data Scientist' 'Machine Learning Engineer' 'Applied Scientist'
 'Data Analyst' 'Data Modeler' 'Researcher' 'Analytics Engineer'
 'Business Intelligence Engineer' 'Data Strategist' 'Data Engineer'
 'Computer Vision Engineer' 'Data Quality Analyst' 'Data Architect'
 'Artificial Intelligence Developer' 'Data Analytics Manager'
 'ETL Engineer' 'Data DevOps Engineer' 'Data Lead' 'Data Science Manager'
 'Data Manager' 'Machine Learning Researcher' 'Data Specialist'
 'Director of Data Science' 'Machine Learning Scientist' 'MLOps Engineer'
 'Artificial Intelligence Scientist' 'Autonomous Vehicle Technician'
 'Cloud Database Engineer' 'Data Infrastructure Engineer'
 'Artificial Intelligence Programmer' 'Business Intelligence Developer'
 'Data Science Lead' 'Deep Learning Researcher' 'Data Consultant'
 'Data Analytics Specialist' 'Insight Analyst' 'Deep Learning Engineer'
 'Computer Vision Software Engineer' 'Data Analytics Lead'
 'Data Science Engineer' 'NLP Engineer' 'Machine Learning Developer'
 '3D Computer Vision Researcher' 'Data Analytics Engineer'
 'Data Operations Analyst' 'Power BI Developer' 'Machine Learning Manager'
 'ETL Developer' 'Machine Learning Lead' 'Principal Data Analyst']

salary_in_usd
[ 85847  30000  25500 ...  28369 412000  94665]
```

Figure 23 Unique values of columns(2)

```
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']

remote_ratio
[100   0  50]

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']

company_size
['L' 'S' 'M']
```

Figure 24 Unique values of columns(3)

3.7 Renaming the experience level column values

Rename the experience level columns as below.

SE – Senior Level/Expert
MI – Medium Level/Intermediate
EN – Entry Level
EX – Executive Level

```
# Dictionary with experience level codes and their respective full labels
experience_level_replacement={
    "SE":"Senior Level/Expert",
    "MI":"Medium Level/Intermediate",
    "EN":"Entry Level",
    "EX":"Executive Level"
}

# Replacing experience level codes with full labels in the 'experience_level' column of the DataFrame 'salary_df'
salary_df['experience_level']=salary_df['experience_level'].replace(experience_level_replacement)
```

Figure 25 Renaming the experience level column values

- First, a dictionary experience_level_replacement is defined that maps the abbreviated experience level codes to their respective full labels.
- The replace() function takes a dictionary as a parameter, where the keys are the values to be replaced (experience level codes) and the values are the new values(full labels.).
- The code then updates the 'experience_level' column in salary_df by replacing the values using the key-value pairs in the experience_level_replacement dictionary. (stack overflow, 2019)

salary_df

	work_year	experience_level	employment_type	job_title	salary_in_usd	employee_residence	remote_ratio	company_location	company_size
0	2023	Senior Level/Expert	FT	Data Scientist	85847	ES	100	ES	L
1	2023	Medium Level/Intermediate	CT	Machine Learning Engineer	30000	US	100	US	S
2	2023	Medium Level/Intermediate	CT	Machine Learning Engineer	25500	US	100	US	S
3	2023	Senior Level/Expert	FT	Data Scientist	175000	CA	100	CA	M
4	2023	Senior Level/Expert	FT	Data Scientist	120000	CA	100	CA	M
...	...	...	...	...	...	...	...	...	...
3750	2020	Senior Level/Expert	FT	Data Scientist	412000	US	100	US	L
3751	2021	Medium Level/Intermediate	FT	Data Scientist	151000	US	100	US	L

Figure 26 Updated DataFrame after renaming the experience level column values

After execution, the 'experience_level' column in salary_df contains the full label descriptions instead of the abbreviated codes, making the data more readable and interpretable.

4 Data Analysis

4.1 Calculating summary statistics

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

```
# Dictionary to store summary statistics
summary_stats = {
    'Total': np.sum(salary_df['salary_in_usd']),  # Total sum of salaries
    'Mean': round(np.mean(salary_df['salary_in_usd']),3),  # Mean salary
    'Standard deviation': round(np.std(salary_df['salary_in_usd']),3),  # Standard deviation of salaries
    'Skewness': round((salary_df['salary_in_usd']).skew(),3),  # Skewness of salary distribution
    'Kurtosis': round((salary_df['salary_in_usd']).kurtosis(),3)  # Kurtosis of salary distribution
}

# Displaying summary statistics
for key, value in summary_stats.items():
    print(key+":", value)
```

Figure 27 Calculating summary stats

The code provided calculates and displays summary statistics for the salary_in_usd column in the salary_df DataFrame. The summary statistics are stored in a dictionary called summary_stats and include:

- Total: The total sum of all salaries in the salary_in_usd column.
- Mean: The mean salary in the salary_in_usd column.
- Standard deviation: It measures the spread of the salaries around the mean.
- Skewness: It measures the degree of asymmetry in the distribution.
- Kurtosis: It measures the degree of peakedness or flatness of the distribution. (Menon, 2023)

The summary statistics are then printed using a for loop that iterates over the summary_stats dictionary. (NumPy, 2022) (pandas, 2024) (pandas, 2024)

```
Total: 341277301
Mean: 133155.404
Standard deviation: 67185.288
Skewness: 0.626
Kurtosis: 0.836
```

Figure 28 Summary Statistics

- Total: The total is 341277301, which means that the combined salaries of all the data points in the salary_df DataFrame is 341277301 USD.
- Mean:. The average salary in the dataset is around 133,155.40 USD.
- Standard deviation: This is a measure of the spread of the salaries around the mean. In this case, the standard deviation is 67185.288, which means that the typical salary in the dataset is around 67185.288 USD away from the mean salary.
- Skewness: The skewness is 0.626, which means that the distribution of salaries is positively skewed (i.e., there are more high-salary data points than low-salary data points). (Shrishty, 2023)

If this value is between:

1. -0.5 and 0.5, the distribution of the value is almost symmetrical
2. -1 and -0.5, the data is negatively skewed, and if it is between 0.5 to 1, the data is positively skewed. The skewness is moderate.
3. If the skewness is lower than -1 (negatively skewed) or greater than 1 (positively skewed), the data is highly skewed.

Figure 29 Skewness analysis (Menon, 2023)

- Kurtosis: The kurtosis is 0.836, indicating a negative kurtosis, which means that the distribution of salaries is relatively flat (i.e., there are fewer outlier values than in a normal distribution). (Menon, 2023)

The expected value of kurtosis is 3. This is observed in a symmetric distribution. A kurtosis greater than three will indicate Positive Kurtosis. In this case, the value of kurtosis will range from 1 to infinity. Further, a kurtosis less than three will mean a negative kurtosis. The range of values for a negative kurtosis is from -2 to infinity. The greater the value of kurtosis, the higher the peak.

Figure 30 Kurtosis analysis (Menon, 2023)

4.2 Correlation of all variables

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

```
# Calculate the correlation matrix for numerical variables
correlation_matrix=salary_df.corr(numeric_only=True)
correlation_matrix
```

Figure 31 Calculating correlation matrix for numeric variables

The code provided calculates the correlation matrix for the numerical variables in the salary_df DataFrame and stores it in the correlation_matrix variable.

	work_year	salary_in_usd	remote_ratio
work_year	1.000000	0.236064	-0.215720
salary_in_usd	0.236064	1.000000	-0.082709
remote_ratio	-0.215720	-0.082709	1.000000

Figure 32 Correlation matrix

Correlation interpretation

- The diagonal elements are all 1, indicating that each variable is perfectly correlated with itself.

work_year and salary_in_usd

- work_year and salary_in_usd have a moderate positive correlation (0.236064).
- This indicates that as the year increases, the average salary in USD also tends to increase.
- This positive correlation suggests that salaries have been rising over time, possibly due to inflation, increased experience, or changes in industry standards.

work_year and remote_ratio

- work_year and remote_ratio have a moderate negative correlation (-0.215720).
- This means that as the year increases, the remote_ratio tends to decrease.

- This negative correlation implies or the percentage of employees working remotely has decreased as the year progresses.

salary_in_usd and remote_ratio

- salary_in_usd and remote_ratio have a weak negative correlation (-0.082709).
- This indicates that as the average salary in USD increases, the remote_ratio tends to decrease slightly.
- This weak negative correlation suggests that higher-paying jobs are less likely to be remote.
  (GeeksForGeeks, 2023)

```
# Setting the size of the heatmap figure
plt.figure(figsize=(5, 3))

# Setting the title of the heatmap
plt.title("Correlation heatmap")

# Ploting the heatmap
sns.heatmap(correlation_matrix)

# Displaying the heatmap
plt.show()
```

Figure 33 Creating correlation heatmap

The code sets the size of the heatmap figure, sets the title of the heatmap, plots the heatmap using seaborn's heatmap function, and displays the heatmap.

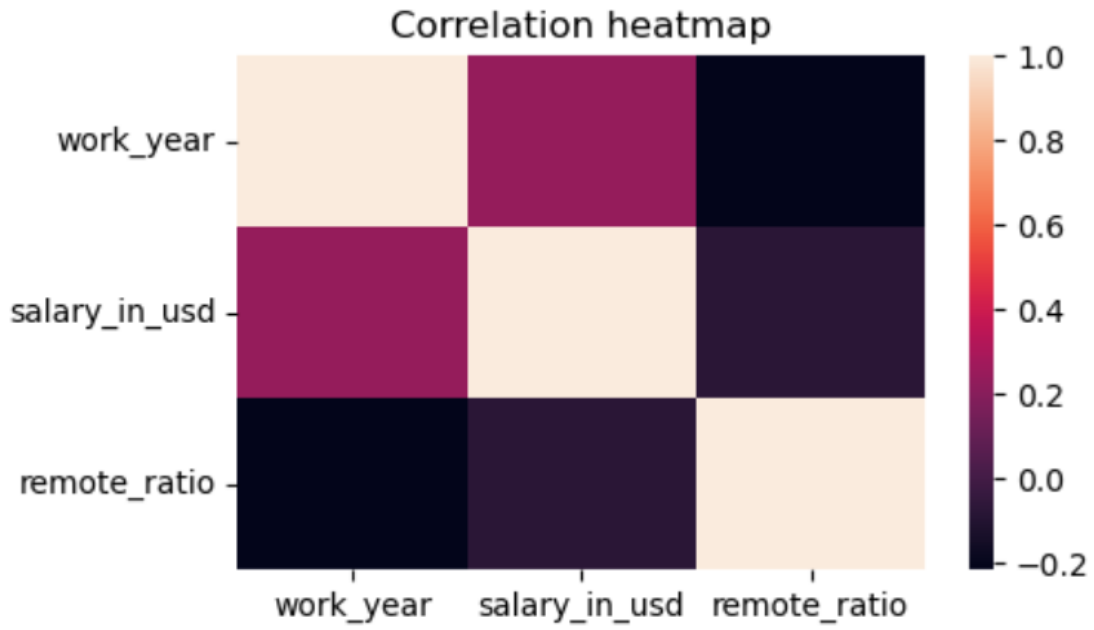


Figure 34 Correlation heatmap

The heatmap shows how strongly two variables in a dataset are correlated, with color intensity indicating the strength of the correlation. (GeeksForGeeks, 2024)

5 Data Exploration

5.1 Top 15 jobs and representation in bar graph

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

```
# Calculate the frequency of each job title and select the top 15 most frequent ones
top_15_jobs=salary_df['job_title'].value_counts().head(15)
top_15_jobs
```

Figure 35 Finding out top 15 jobs based on frequency

- This code extracts the top 15 most frequent job titles from the 'salary_df' DataFrame, where each job title's occurrence number is counted using the value_counts() function.
- The top 15 are selected using the head(15) function.

```
Data Engineer                        629
Data Scientist                       569
Data Analyst                         450
Machine Learning Engineer            259
Researcher                            94
Analytics Engineer                    91
Data Architect                        68
Data Science Manager                  52
Machine Learning Scientist            38
Applied Scientist                     31
Data Consultant                       25
Data Manager                          25
Data Analytics Manager                18
Computer Vision Engineer              18
Artificial Intelligence Scientist     16
Name: job_title, dtype: int64
```

Figure 36 Top 15 jobs based on frequency

The most frequent job title is "Data Engineer" with a frequency count of 629. The next most frequent job titles are "Data Scientist" (569), "Data Analyst" (450), "Machine Learning Engineer" (259), and "Researcher" (94).The frequency count provides information about the popularity of certain job titles in the data field.

```
# Setting the figure size
plt.figure(figsize=(8, 4))

# Plotting the horizontal bar graph
top_15_jobs.plot(kind='barh',color='lightgreen')

# Setting the title
plt.title('Top 15 Popular Jobs')

# Labeling x-axis
plt.xlabel('Frequency')

# Labeling y-axis
plt.ylabel('Job Titles')

# Setting x-axis tick marks, rotation, and alignment for better readability
plt.xticks(rotation=45, ha='right', ticks=range(0, int(top_15_jobs.max()) + 50, 50))

# Adding gridlines along the x-axis with reduced opacity
plt.grid(axis='x', alpha=0.5)

# Displaying the plot
plt.show()
```

Figure 37 Creating horizontal bar graph for Top 15 jobs

The code creates a horizontal bar plot of the top 15 popular jobs, sets the plot title, labels the x and y axes, rotates, and aligns the x-axis tick marks, adds gridlines along the x-axis, and displays the plot. The figure size is set to 8 inches width, 4 inches height, and the x-axis tick marks are spaced at intervals of 50. (pandas, n.d.)

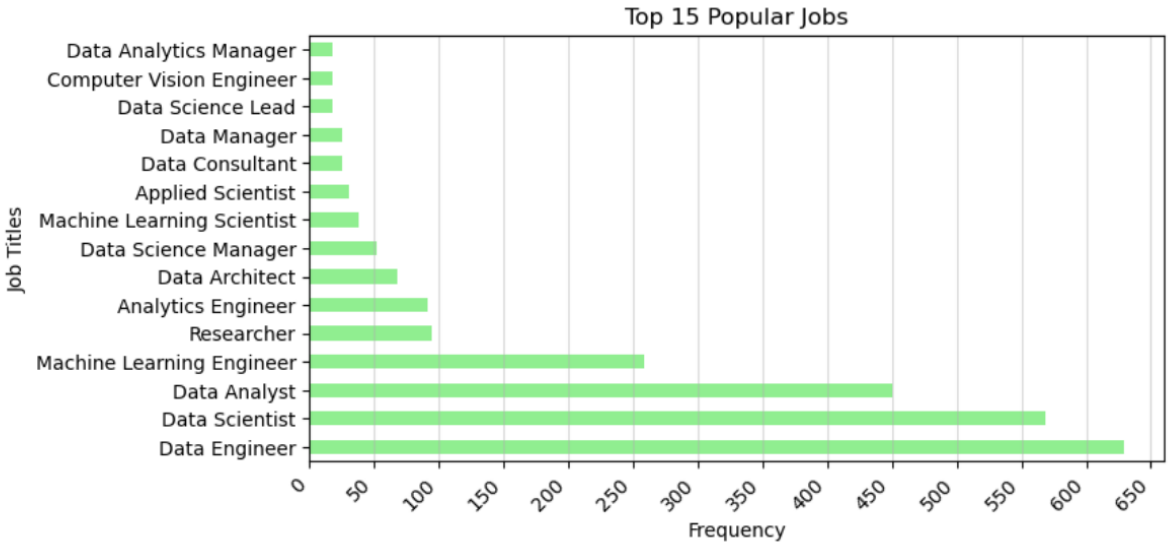


Figure 38 Horizontal Bar Graph for Top 15 jobs

The horizontal bar graph provides a clear and easy-to-read visualization of the top 15 most frequent job titles in the dataset, along with their corresponding frequency counts. The horizontal bar graph provides a clear and easy-to-read visualization of the top 15 most frequent job titles in the dataset, along with their corresponding frequency counts.

5.2 Job with highest salaries and bar graph illustration

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

```
# Grouping the DataFrame by 'job_title' and calculating the mean salary for each group,

mean_salary_df = salary_df.groupby('job_title')['salary_in_usd'].mean()

# Selecting the top job title with the highest mean salary
highest_salary_job = mean_salary_df.idxmax()

print("Job with the highest salary:", highest_salary_job)
```

```
Job with the highest salary: Data Analytics Lead
```

Figure 39 Finding out job with highest salaries

- It groups the **salary_df** DataFrame by the **job_title** column, it separates the data into different groups based on the job title.

- Then, it calculates the mean salary for each group using the **mean()** function. The result is a new DataFrame **mean_salary_df** that contains the mean salary for each job title.
- The code then selects the job title with the highest mean salary using the **idxmax()** function. This function returns the index label of the maximum value in the DataFrame.
- Finally, it prints the job title with the highest mean salary using the **print()** function.

In summary, this code calculates the mean salary for each job title and selects the job title with the highest mean salary.

```
Job with the highest salary: Data Analytics Lead
```

Figure 40 Job with highest salary

The output indicates that the job title with the highest mean salary among all job titles in the dataset is "Data Analytics Lead".

```
# Sorting the DataFrame by mean salary in descending order
# then selecting the top 15 job titles with highest mean salary
highest_salary_df = mean_salary_df.sort_values(ascending=False).head(15)
highest_salary_df
```

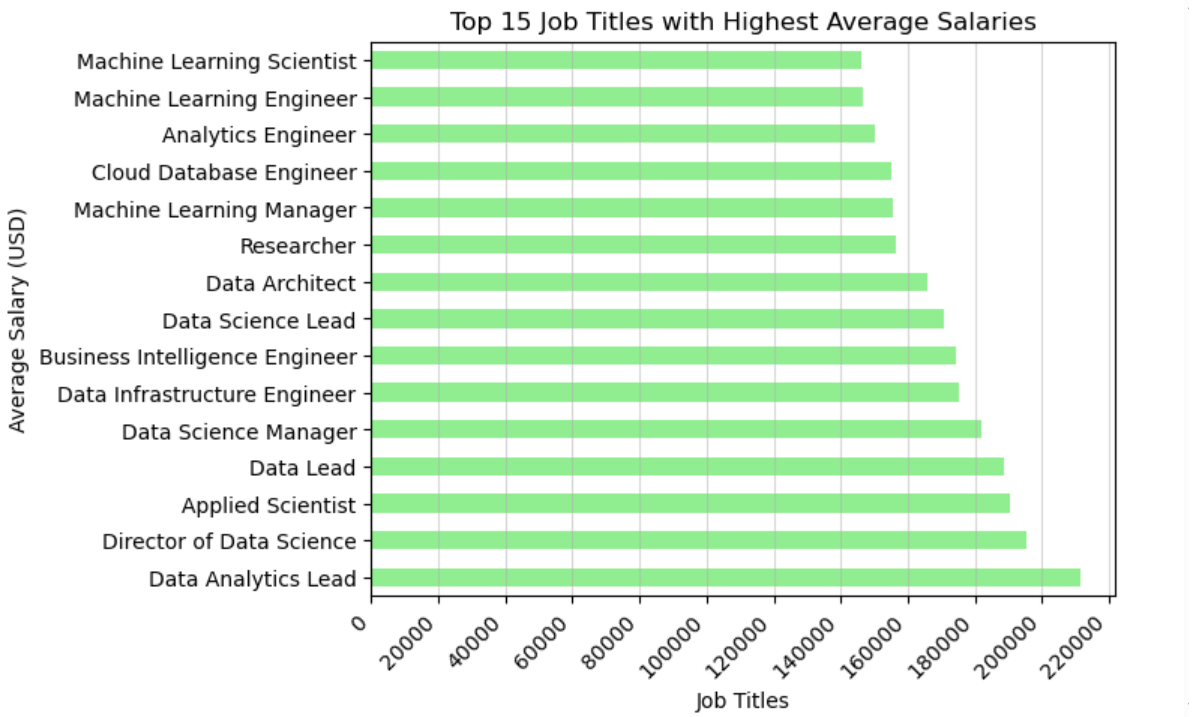
- The code first sorts the mean_salary_df DataFrame by mean salary in descending order using the sort_values() function with the ascending parameter set to False.
- The head() function is then used to select the top 15 job titles with the highest mean salary.
- The resulting DataFrame is stored in the highest_salary_df variable.

```
job_title
Data Analytics Lead               211254.500000
Director of Data Science          195140.727273
Applied Scientist                 190342.580645
Data Lead                         188631.250000
Data Science Manager              181930.173077
Data Infrastructure Engineer      175051.666667
Business Intelligence Engineer    174150.000000
Data Science Lead                 170611.111111
Data Architect                    165821.352941
Researcher                        156407.170213
Machine Learning Manager          155701.333333
Cloud Database Engineer           155000.000000
Analytics Engineer                150151.747253
Machine Learning Engineer         146375.289575
Machine Learning Scientist        146240.947368
Name: salary_in_usd, dtype: float64
```

- The top 15 job titles are displayed along with their corresponding mean salaries.
- The job titles are listed in descending order of mean salary.
- The mean salary for each job title is displayed in US dollars.

```
# Plotting a horizontal bar graph for highest salaries
highest_salary_df.plot(kind='barh', color='lightgreen')

# Setting x-axis tick marks, rotation, and alignment for better readability
plt.xticks(rotation=45, ha='right', ticks=range(0, int(highest_salary_df.max()) + 20000, 20000))

# Adding title to the bar plot
plt.title('Top 15 Job Titles with Highest Average Salaries')

# Labeling x-axis
plt.ylabel('Average Salary (USD)')

# Labeling y-axis
plt.xlabel('Job Titles')

# Adding gridlines along the x-axis with reduced opacity
plt.grid(axis='x', alpha=0.5)
```

The code creates a horizontal bar plot of the top 15 job titles with the highest average salaries, sets the x-axis tick marks, rotates, and aligns the x-axis tick marks for better readability, sets the plot title, labels the x and y axes, adds gridlines along the x-axis with reduced opacity, and displays the plot. The x-axis tick marks are spaced at intervals of 20,000 and the x-axis labels are rotated 45 degrees and horizontally aligned to the right.



Based on the bar graph made for the mean salary by job title, we can see that the job title with the highest mean salary is "Data Analytics Lead" with a mean salary of $211,254.50. The next highest mean salaries are for "Director of Data Science" ($195,140.73), "Applied Scientist" ($190,342.58), "Data Lead" ($188,631.25), and "Data Science Manager" ($181,930.17).

This suggests that certain job titles may be more in demand or require more skills, leading to higher salaries. (pandas, n.d.)

5.3 Salaries based on experience level and bar graph illustration

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

```
# Grouping the DataFrame by 'experience_level' and calculating the mean salary for each group
salary_experience_level = salary_df.groupby('experience_level')['salary_in_usd'].mean()

# Displaying the mean salary for each experience level
salary_experience_level
```

Figure 41 Calculating mean salaries bases on experience level

- The function groupby('experience_level') groups the salary_df DataFrame by the experience_level column.
- mean(): This aggregate function calculates the mean of the values in the salary_in_usd column for each group.
- The resulting DataFrame has experience_level as its index and the mean salary as its values.

```
experience_level
Entry Level                  72648.685185
Executive Level             191078.208333
Medium Level/Intermediate    101786.292609
Senior Level/Expert          153738.159061
Name: salary_in_usd, dtype: float64
```

Figure 42 Experience level with their corresponding mean salary

The experience levels are: Entry Level, Executive Level, Medium Level/Intermediate, and Senior Level/Expert.

- The Entry Level has the lowest mean salary of approximately $72,649.
- The Medium Level/Intermediate has a mean salary of around $101,786.
- The Senior Level/Expert has a mean salary of approximately $153,738.
- The Executive Level has the highest mean salary of approximately $191,078.

```
# Plotting the bar graph for salary based on experience level
salary_experience_level.plot(kind='bar', color='lightgreen')

# Rotating x-axis labels for better readability
plt.xticks(rotation=15, ha='right')

# Adding title to the bar plot
plt.title("Salary based on experience level bar graph")

# Labeling x-axis
plt.xlabel("Experience Level")

# Labeling y-axis
plt.ylabel("Mean salary")

# Adding gridlines along the y-axis for clarity
plt.grid(axis='y')

# Displaying the plot
plt.show()
```

Figure 43 Creating bar graph for salary based on experience level

The code creates a bar plot of the mean salary based on experience level, rotates the x-axis labels for better readability, sets the plot title, labels the x and y axes, adds gridlines along the y-axis, and displays the plot. The x-axis labels are rotated 15 degrees and horizontally aligned to the right. (pandas, n.d.)

Figure 44 Salary based on experience level bar graph

Overall, the graph shows that the mean salary increases with the level of experience.

5.4 Histogram and box plot of variable ‘salary_in_usd’

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

5.4.1 Box plot

```
# Create a box plot of the 'salary_in_usd' column of the 'salary_df' DataFrame
plt.boxplot(salary_df['salary_in_usd'])

# Adding labels and title
plt.xlabel('Salaries (USD)')
plt.ylabel('Frequency')
plt.title('Distribution of Salaries')

# Adding grid
plt.grid(axis='y')

# Showing the plot
plt.show()
```

Figure 45 Creating box plot

The code creates a boxplot of the column ‘salary_in_df’ and sets labels, titles, adds grid, and displays the plot.

The boxplot() method displays the distribution of data using a five number summary: minimum, Q1, median, Q3, and maximum. It also shows any outliers. The box represents the interquartile range (IQR), with the line in the middle of the box indicating the median. Outliers are displayed as individual points beyond the whiskers. (lumen, n.d.)

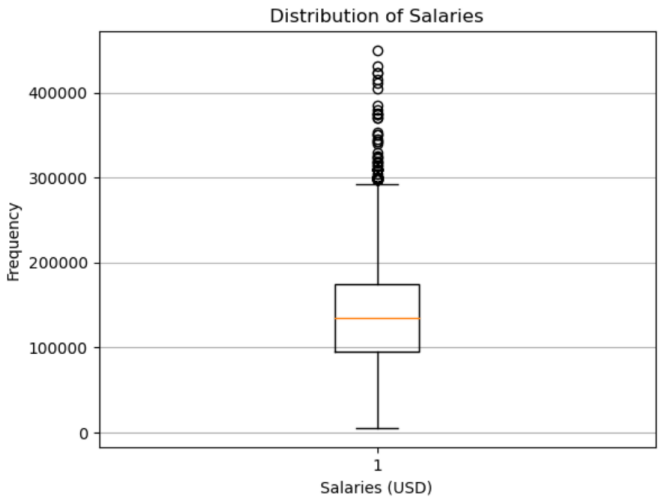


Figure 46 Box plot of salary_in_usd column

```
# Summary statistics of 'salary_in_usd' column
salary_df['salary_in_usd'].describe()
```

```
count      2563.000000
mean     133155.404214
std       67198.398697
min        5132.000000
25%       84053.000000
50%      129300.000000
75%      175000.000000
max      450000.000000
Name: salary_in_usd, dtype: float64
```

Figure 47 Summary statistics of Salary in USD

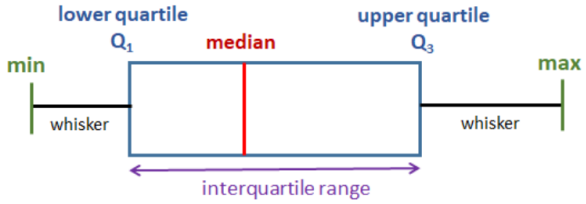


Figure 48 Box plot interpretation (online math learning, n.d.)

From the given box plot and summary statistics:

- The median salary is 129,300 USD. This is the middle value of the data, which means that half of the salaries are above this value and half are below.
- The first quartile (Q1) is 84,053 USD. This means that 25% of the salaries are below this value.
- The third quartile (Q3) is 175,000 USD. This means that 75% of the salaries are below this value.
- The minimum salary is 51,320 USD.
- The maximum salary is 450,000 USD.

5.4.2 Histogram

```
# Getting the minimum and maximum values of the 'salary_in_usd' column
min_salary = salary_df['salary_in_usd'].min()
max_salary = salary_df['salary_in_usd'].max()

# Create a figure with a specified size
plt.figure(figsize=(7, 5))

# 11 bins for histogram
bins = np.linspace(min_salary, max_salary, 12)

# Creating a histogram of the 'salary_in_usd' column of the 'salary_df' DataFrame
plt.hist(salary_df['salary_in_usd'], color='lightgreen', bins=bins, edgecolor='black')

# Adding labels and title to the plot
plt.title('Histogram of Salary in USD')
plt.xlabel('Salaries (USD)')
plt.ylabel('No. of employees')

# Setting the x-ticks to the positions of the bins
plt.xticks(bins, rotation=45, ha='right')

# Addding a grid
plt.grid(alpha=0.6)

# Displaying the plot
plt.show()
```

Figure 49 Creating histogram of salary_in_usd column

This code creates a histogram of the 'salary_in_usd' column of the 'salary_df' DataFrame. The minimum and maximum values of the column are first obtained and used to create 11 bins for the histogram. The histogram is then plotted using the plt.hist() function, with the 'salary_in_usd' column as the input data, and the number of bins, bin edges, and color specified. The x-axis label, y-axis label, and title are added to the plot, the x-ticks are set to the positions of the bins, and a grid. (stack overflow, 2024)

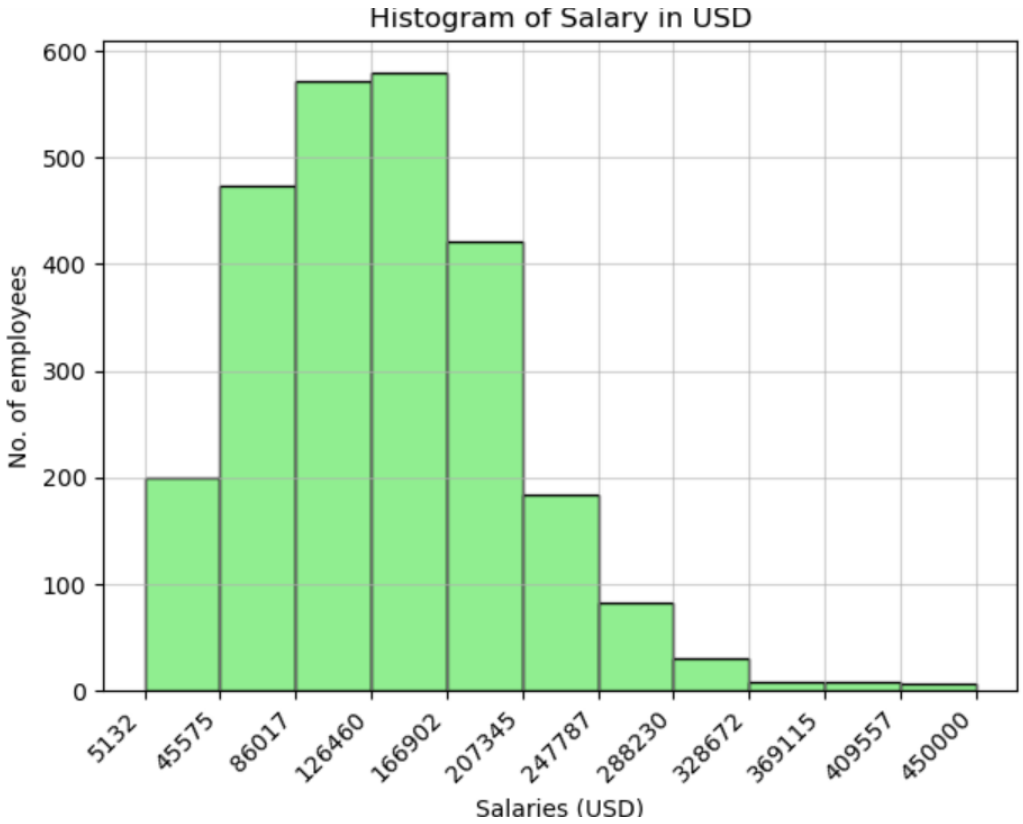


Figure 50 Histogram of salary in USD

The histogram displays a right-skewed distribution, with a noticeable decrease in the number of individuals as the salary increases.

To be more specific, there are around 200 individuals with a salary between $5132 and $45575, and approximately 480 individuals with a salary between $45575 and $86017. The number of individuals with a salary between $86017 and $126460 is also around 480, indicating a similar trend.

The number of individuals with a salary between $126460 and $166902 increases slightly to around 485, making it the bin with the highest number of individuals.

Towards the higher end of the salary range, there is a significant decrease in the number of individuals. There are only around 10 individuals with a salary between $409557 and $450000, which is the lowest among all the bins.

6 Conclusion

In conclusion, this project has been a valuable learning experience in working with data manipulation and analysis using Python and its libraries. I have gained a deeper understanding of how to clean and preprocess data, as well as how to extract meaningful insights from large datasets.

Throughout the project, I encountered several challenges that required me to seek guidance from my module instructor, Mr. Dipeshor Silwal, as well as from online resources and my peers. These challenges included handling inconsistent data formats and implementing complex data transformations.

With the help of my teacher, the internet, and my friends, I was able to overcome these challenges and successfully complete the project. Overall, this project has been a valuable learning experience that has helped me to develop my critical thinking, problem-solving, and data analysis skills. I am grateful for the guidance and support I received throughout the project, and I look forward to continuing to develop my skills in data manipulation and analysis.

7 References

3 Pillar, 2016. *Understanding the data in data science.* [Online]
Available at: https://www.3pillarglobal.com/insights/blog-posts/understanding-data-data-science/
[Accessed May 2024].

datacamp, 2023. *pandas read csv() Tutorial.* [Online]
Available at: https://www.datacamp.com/tutorial/pandas-read-csv
[Accessed May 2024].

GeeksForGeeks, 2023. *Correlation: Meaning, Significance, Types and Degree of Correlation.* [Online]
Available at: https://www.geeksforgeeks.org/correlation-meaning-significance-types-and-degree-of-correlation/
[Accessed April 2024].

GeeksForGeeks, 2024. *How to create a seaborn correlation heatmap in Python?.* [Online]
Available at: https://www.geeksforgeeks.org/how-to-create-a-seaborn-correlation-heatmap-in-python/
[Accessed April 2024].

IBM, 2021. *Writing a Data Description Report.* [Online]
Available at: https://www.ibm.com/docs/en/spss-modeler/saas?topic=data-writing-description-report#crisp_data_description_report
[Accessed May 2024].

lumen, n.d. *Box Plots.* [Online]
Available at: https://courses.lumenlearning.com/introstats1/chapter/box-plots/
[Accessed May 2024].

Menon, K., 2023. *The Complete Guide to Skewness and Kurtosis.* [Online]
Available at: https://www.simplilearn.com/tutorials/statistics-tutorial/skewness-and-kurtosis
[Accessed May 2024].

NumPy, 2022. *Statistics.* [Online]
Available at: https://numpy.org/doc/stable/reference/routines.statistics.html
[Accessed May 2024].
online math learning, n.d. *Box Plot.* [Online]
Available at: https://www.onlinemathlearning.com/box-plot.html
[Accessed 2024].
pandas, 2024. *How to calculate summary statistics.* [Online]
Available at:
https://pandas.pydata.org/docs/getting_started/intro_tutorials/06_calculate_statistics.html
[Accessed May 2024].
pandas, 2024. *pandas.DataFrame.drop_duplicates.* [Online]
Available at:
https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.drop_duplicates.html
[Accessed April 2024].
pandas, 2024. *pandas.DataFrame.kurtosis.* [Online]
Available at:
https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.kurtosis.html
[Accessed May 2024].
pandas, n.d. *pandas.DataFrame.plot.* [Online]
Available at: https://pandas.pydata.org/pandas-docs/version/0.23/generated/pandas.DataFrame.plot.html
[Accessed May 2024].
SaturnCloud, 2023. *How to Filter Pandas DataFrame by Substring Criteria.* [Online]
Available at: https://saturncloud.io/blog/how-to-filter-pandas-dataframe-by-substring-criteria/#:~:text=To%20filter%20a%20pandas%20DataFrame,column%20contains%20the%20specified%20substring.&text=This%20code%20first%20selects%20the,the%20indexing%20operator%20(%20%5B%5D
[Accessed May 2024].
Shrishty, 2023. *What Is Normal Distribution & Standard Deviation in Statistics.* [Online]
Available at: https://www.simplilearn.com/tutorials/statistics-tutorial/what-is-normal-

distribution
[Accessed May 2024].
Stack overflow, 2015. *Print unique values of every column in pandas dataframe.* [Online]
Available at: https://stackoverflow.com/questions/27241253/print-the-unique-values-in-every-column-in-a-pandas-dataframe
[Accessed May 2024].
stack overflow, 2019. *Rename column values using pandas dataframe.* [Online]
Available at: https://stackoverflow.com/questions/57165247/rename-column-values-using-pandas-dataframe
[Accessed May 2024].
stack overflow, 2024. *Matplotlib xticks not lining up with histogram.* [Online]
Available at: https://stackoverflow.com/questions/27083051/matplotlib-xticks-not-lining-up-with-histogram
[Accessed May 2024].
w3schools, 2024. *Pandas DataFrame duplicated() Method.* [Online]
Available at: https://www.w3schools.com/python/pandas/ref_df_duplicated.asp
[Accessed May 2024].