LONDON METROPOLITAN UNIVERSITY

islington college
(इस्लिङ्टन कलेज)

CC5067NI-Smart Data Discovery

60% Individual Coursework

2023-24 Spring

Student Name: Rashi Maharjan

London Met ID: 22067683

College ID: NP01CP4A220113

Assignment Due Date: Monday, May 13, 2024

Assignment Submission Date: Monday, May 13, 2024

Word Count: 3292

I confirm that I understand my coursework needs to be submitted online via MySecondTeacher under the relevant module page before the deadline 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.

Abstract

This coursework report is based on the data preparation, analysis, and exploration of a dataset of real-world data involving salaries of data science. The primary objective of this project is to prepare data for further data mining and analysis. The coursework requires reading and cleaning of data, summarizing statistics and correlation of the numeric data, and creating visualizations like bar graphs, histograms, and boxplots for better insights into data.

Acknowledgment

I would like to express our sincere gratitude to our respected teachers, Mr. Dipeshwor Silwal and Mr. Alish K.C of Smart Data Discovery Module who gave us such a wonderful opportunity to us to do such an astonishing project. Due to their active guidance, help, and encouragement, this report was able to turn out as a success.

I would also like to thank my friends who assisted me in solving some problems. This project gave us insight into the process of data preparation for further data mining and analysis of real-world data.

Table of Contents

Table of Figures

Table of tables

1. Introduction

The coursework is a demonstration of problem-solving and critical thinking skills with the exercise of applying programming knowledge and skills to data analysis tasks related to the Smart Data Discovery module. This module aims to provide comprehension of the fundamental concepts and techniques of data science and its applications in a wide range of business contexts and introduce the methods for formulating problems, preparing data, modeling data, making data-driven decisions, visualization, and forecasting.

The coursework involves the Data Science salary analysis. It requires writing a program in Python with data that contains information about various factors that can influence salary levels such as experience, work level, job title, and many more, and preparing data for further data mining and analysis. The report involves questions related to data preparation, analysis, and exploration which is essential for making business decisions in real-world scenarios.

1.1 Tools Used:

1.1.1 Anaconda Distribution

Anaconda Distribution is a free and open-source Python/R data science distribution that equips individuals with a powerful toolkit for Python and R data science and machine learning which can be conveniently accessible on a single machine. It includes components like Conda, Anaconda Navigator, and Anaconda Repository with 8000 open-source data science and machine learning packages (Anaconda , 2018). It is a robust platform for those who require a stable and versatile environment for their work and deal with complex projects that have many dependencies.

Figure 1 Anaconda logo.

1.1.2 Jupyter Notebook

Jupyter Notebook is a powerful and versatile tool for interactive computing, data analysis, and scientific exploration. It includes Jupyter Lab which is a web-based interactive development environment with modular design for notebooks, code, and data and supports over forty programming languages (Jupyter , 2024). Jupyter Notebook is used as a main IDLE as it allows you to write and run Python scripts and different cells as required.

Figure 2 Jupyter Notebook logo.

1.1.3 Python

Python is the general purpose and high-level programming language that places a strong emphasis on code readability. It supports a variety of programming paradigms, including imperative, functional, and object-oriented programming. It enables the expression of concepts more simply to produce understandable systems (Rossum, 2007). The Python programming language has been used to complete this assignment by solving the assigned questions.

Figure 3 Python logo.

1.2 Aims and Objectives

This project aims to be able to do problem-solving and critical thinking using programming knowledge for the data analysis of the Data Science salary By comprehending, preparing, exploring, and performing initial analysis.

The main objectives are listed below:

a. To prepare data for further data mining and analysis.
b. To be able to do real-world data analysis by applying programming skills.
c. To discover any regularities or tendencies within the data.
d. To obtain a better understanding of the elements that influence the salaries of data scientists.
e. To understand what your data resources are and the characteristics of those resources.

1.3 Modules and Library Used

1.3.1 Pandas

Pandas is a Python library for data analysis and manipulation that provides compatible and high-performance data structures and data analysis tools (pandas, 2024). It makes it easy to clean messy data and handle missing values, duplicate records, and anomalies seamlessly. It can read data from various sources and provides various data structures and operations for manipulating numerical data and time series.

1.3.2 Matplotlib

Matplotlib is a Python library for visualizations of graph plots. It makes static, animated, and interactive data visualization easy to analyze trends, explore results, or present results. It can customize visual styles, layouts, and labels and provides flexibility (Matplotlib, 2023). It consists of several plots like bar, histogram, etc. which is very useful in data analysis and data visualization.

2. Data Understanding

Data Understanding is the process of critical thinking that involves accessing and exploring the available raw data for mining and gaining insights. It allows us to determine and describe the quality of the data (IBM, 2021). It involves crucial activities like Identifying potential data sources, capturing aggregate data sources, reviewing the raw data, and evaluating the data structures and tools needed . This step is crucial to find issues in the data quality such as null values, duplicates, and inconsistency.

The **Data Science Salaries.csv** file consists of the data related to factors influencing the salary of Data Science. It has a detailed list of professions, and their work year, experience level, employment type job title, salary, salary currency, salary in USD, employee residence, remote ratio, company location, and company size. It has **3755 rows** along with **11 columns**.

Summary of the DataFrame:

Columns	Data Type	Description
work_year	int64	It stores the year when someone was employed or earned a salary. The common values include 2020, 2021, 2022, and 2023.
experience_level	object	It stores the level of experience in the specified job. The common values include: • SE: Senior Experience • MI: Mid-Level Experience • EN: Entry-Level Experience • EX: Executive Experience
employment_type	object	It stores the type of employment arrangement. The common values include: • FT: Full-Time

		• CT: Contract or Casual • FL: Freelance • PT: Part-Time
job_title	object	It stores data on the specific role or job title. The common values include Principal Data Scientist, ML Engineer, Data Scientist, etc. It has a total of 93 unique values.
salary	int64	It stores the total gross salary amount paid.
salary_currency	object	It stores the currency of the salary. The common values include EUR, USD, INR HKD, CHF, GBP, AUD, etc.
salary_in_usd	int64	It stores the salary converted in US dollars to compare the salaries across currencies.
employee_residence	object	It stores the country where the employee lives. The common values include ES, US, CA, DE, GB, etc.
remote_ratio	int64	It stores the overall amount of work done remotely. The common values include 0, 50, and 100.
company_location	object	It is stored in the country where the company is located. The common values include ES, US, CA, DE, GB, etc.
company_size	object	It stores the scale of the company. The common values include: • S for small • M for Medium • L for Large.

Table 1 Data Types and Description of each column.

```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3755 entries, 0 to 3754
Data columns (total 11 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   work_year           3755 non-null   int64
 1   experience_level    3755 non-null   object
 2   employment_type     3755 non-null   object
 3   job_title           3755 non-null   object
 4   salary              3755 non-null   int64
 5   salary_currency     3755 non-null   object
 6   salary_in_usd       3755 non-null   int64
 7   employee_residence  3755 non-null   object
 8   remote_ratio        3755 non-null   int64
 9   company_location    3755 non-null   object
 10  company_size        3755 non-null   object
dtypes: int64(4), object(7)
memory usage: 322.8+ KB
```

Figure 4 Summary of the DataFrame.

The summary of the data frame is obtained by using the function, **df.info()** which includes the column titles, number of non-null values, and the data type of each column. It also gives details of **memory** usage which is **322.8+ KB** here.

Summary of the Null values in the DataFrame:

```
df.isnull().sum() #count

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
```

Figure 5 Summary of null values column-wise.

```
df[df.isnull().any(axis=1)] #row
```

work_year	experience_level	employment_type	job_title	salary	salary_currency	salary_in_usd	employee_residence	remote_ratio	company_location	co

Figure 6 Summary of null values row-wise.

The summary of the null values in the row and column of the DataFrame is obtained by using the function, **df. isnull().** The **sum()** function shows the exact sum of the values and **any()** function shows the Boolean values i.e. True or False coordinated with **axis = 1** to exact the values from the row. In the DataFrame, there were not any null values present in either row or column as the output was **zero**. It is important to remove and replace null values to maintain data consistency in the DataFrame.

Summary of the Duplicate values in the DataFrame:

```
#count number of duplicate value
duplicate_count = df.duplicated().sum()
duplicate_count
```

```
1171
```

Figure 7 Number of duplicate values in the columns.

The summary of the duplicate values in the columns was obtained by using **df.duplicated().sum()** which displayed the number of existing duplicate values. Here, the output was **1171.**

```
#remove duplicate values in dataframe
df = df.drop_duplicates()
```

```
#count number of duplicate value after removing it
duplicate_count = df.duplicated().sum()
duplicate_count
```

```
0
```

Figure 8 Remove duplicate values.

The duplicate values were removed using the function **df.drop_duplicates(), which helped remove inconsistency** in the DataFrame. After removing the duplicates, the value became zero.

Statistics of the Numeric columns of the DataFrame:

```
df.describe()
```

	work_year	salary	salary_in_usd	remote_ratio
count	3755.000000	3.755000e+03	3755.000000	3755.000000
mean	2022.373635	1.906956e+05	137570.389880	46.271638
std	0.691448	6.716765e+05	63055.625278	48.589050
min	2020.000000	6.000000e+03	5132.000000	0.000000
25%	2022.000000	1.000000e+05	95000.000000	0.000000
50%	2022.000000	1.380000e+05	135000.000000	0.000000
75%	2023.000000	1.800000e+05	175000.000000	100.000000
max	2023.000000	3.040000e+07	450000.000000	100.000000

Figure 9 Statistics of the Numeric columns of the DataFrame.

The statistics of the Numeric columns of the DataFrame are obtained by using the function, **df. describe()** which describes a dataset's central tendency, dispersion, and distributional shape excluding NaN values. It includes count, mean, standard deviation, minimum, maximum, and quartile ranges i.e. 25%, 50%, and 75%.

Data Inconsistency:

The presence of data inconsistency was seen in the values of the **'job_title'** column of the DataFrame. There were job titles having the same meaning such as:
"Financial Data Analyst" and "Finance Data Analyst",
"Lead Data Scientist" and "Data Scientist Lead",
"ML Engineer" and "Machine Learning Engineer", and more.

```
#number of unique job_titles
num_unique_job_titles = df['job_title'].nunique()
num_unique_job_titles
```

```
93
```

Figure 10 Number of unique job titles before removing inconsistency.

Initially, there were 93 total unique job titles in the **'job_title'** column of the DataFrame **df.**

Thus, to remove these inconsistencies in the values, the .**replace()** function is used to replace inconsistent values and update the **'job_title'** column DataFrame **df**.

Removing Data Inconsistency

```
In [4]:
#replace to Data Analyst
df['job_title'] = df['job_title'].replace(
    ['Compliance Data Analyst', 'Business Data Analyst',
     'Staff Data Analyst', 'Lead Data Analyst',
     'Financial Data Analyst', 'Finance Data Analyst',
     'BI Data Analyst', 'Product Data Analyst',
     'Marketing Data Analyst', 'Principal Data Analyst',
     'Data Quality Analyst', 'Data Operations Analyst'], 'Data Analyst')

In [5]:
#replace to Data Scientist
df['job_title'] = df['job_title'].replace(
    ['Principal Data Scientist', 'Applied Data Scientist',
     'Lead Data Scientist', 'Data Scientist Lead',
     'Product Data Scientist', 'Staff Data Scientist'], 'Data Scientist')

In [6]:
#replace to Machine Learning Scientist
df['job_title'] = df['job_title'].replace('Applied Machine Learning Scientist', 'Machine Learning Scientist')
```

```
In [7]:
# replace to Machine Learning Engineer
df['job_title'] = df['job_title'].replace(
    ['ML Engineer', 'Applied Machine Learning Engineer',
     'Machine Learning Infrastructure Engineer', 'Machine Learning Software Engineer',
     'Machine Learning Research Engineer', 'Principal Machine Learning Engineer',
     'Lead Machine Learning Engineer'], 'Machine Learning Engineer')

In [8]:
#replace to Business Intelligence Engineer
df['job_title'] = df['job_title'].replace('BI Data Engineer', 'Business Intelligence Engineer')

In [9]:
#replace to Computer Vision Engineer
df['job_title'] = df['job_title'].replace('Computer Vision Software Engineer', 'Computer Vision Engineer')

In [10]:
#replace to Head of Data Science
df['job_title'] = df['job_title'].replace('Head of Data', 'Head of Data Science')

In [11]:
#replace to Data Engineer
df['job_title'] = df['job_title'].replace(
    ['Data Infrastructure Engineer', 'Software Data Engineer',
     'Data DevOps Engineer', 'Big Data Engineer',
     'Data Operations Engineer', 'Azure Data Engineer',
     'Marketing Data Engineer', 'Data Science Engineer',
     'Cloud Database Engineer', 'Cloud Data Engineer',
     'Data Analytics Engineer', 'Lead Data Engineer', 'Principal Data Engineer'], 'Data Engineer')
```

```
In [12]:
#replace to Data Manager
df['job_title'] = df['job_title'].replace(
    ['Manager Data Management', 'Data Analytics Manager',
     'Data Science Manager'], 'Data Manager')

In [13]:
#replace to Data Architect
df['job_title'] = df['job_title'].replace(
    ['Big Data Architect', 'Principal Data Architect',
     'Cloud Data Architect'], 'Data Architect')

In [14]:
#replace to Data Lead
df['job_title'] = df['job_title'].replace(
    ['Data Analytics Lead', 'Data Science Lead',
     'Data Science Tech Lead'], 'Data Lead')

In [15]:
#replace to Data Specialist
df['job_title'] = df['job_title'].replace(
    ['Data Analytics Specialist', 'Data Management Specialist'], 'Data Specialist')
```

Figure 11 Removing inconsistent values in the job_title column.

After the removal of inconsistent values, the total number of job titles became **40.**

```
#number of unique job_titles after removing data inconsistency
num_unique_job_titles = df['job_title'].nunique()
num_unique_job_titles
```

```
40
```

Figure 12 Number of unique job titles after removing inconsistency.

```
In [17]: #unique job titles after removing inconsistency
         unique_job_titles = df['job_title'].unique()
         unique_job_titles

Out[17]: array(['Data Scientist', 'Machine Learning Engineer', 'Applied Scientist',
                'Data Analyst', 'Data Modeler', 'Research Engineer',
                'Analytics Engineer', 'Business Intelligence Engineer',
                'Data Strategist', 'Data Engineer', 'Computer Vision Engineer',
                'Data Architect', 'AI Developer', 'Research Scientist',
                'Data Manager', 'ETL Engineer', 'Head of Data Science',
                'Machine Learning Researcher', 'Data Specialist',
                'Director of Data Science', 'Machine Learning Scientist',
                'MLOps Engineer', 'AI Scientist', 'Autonomous Vehicle Technician',
                'AI Programmer', 'BI Developer', 'Data Lead',
                'Deep Learning Researcher', 'BI Analyst',
                'Data Science Consultant', 'Insight Analyst',
                'Deep Learning Engineer', 'NLP Engineer',
                'Machine Learning Developer', '3D Computer Vision Researcher',
                'Data Analytics Consultant', 'Power BI Developer',
                'Machine Learning Manager', 'ETL Developer',
                'Head of Machine Learning'], dtype=object)
```

Figure 13 Unique job titles after removing inconsistency.

3. Data Preparation

Data Preparation is the process of sorting, cleaning, and preparing raw data for analysis and processing, it improves the quality of data which enhances the performance and saves time and resources (GeeksforGeeks, 2024).

a. Write a Python program to load data into pandas DataFrame.

```
In [1]: import pandas as pd #importing pandas library

In [4]: #loading data
        data = pd.read_csv('Data Science Salaries.csv')
        df = pd.DataFrame(data)
        df
```

Out[4]:

	work_year	experience_level	employment_type	job_title	salary	salary_cur
0	2023	SE	FT	Principal Data Scientist	80000	
1	2023	MI	CT	ML Engineer	30000	
2	2023	MI	CT	ML Engineer	25500	
3	2023	SE	FT	Data Scientist	175000	
4	2023	SE	FT	Data Scientist	120000	
...	...	...	...	...	...	
3750	2020	SE	FT	Data Scientist	412000	
3751	2021	MI	FT	Principal Data Scientist	151000	
3752	2020	EN	FT	Data Scientist	105000	
3753	2020	EN	CT	Business Data Analyst	100000	
3754	2021	SE	FT	Data Science Manager	7000000	

3755 rows × 11 columns

Figure 14 Load data into Pandas DataFrame.

A function provided by Pandas library, **read_csv()** is imported to read a CSV (Comma-Separated-Values) file named **'Data Science Salaries.csv'** and load its contents into a Pandas DataFrame called **data**. A new DataFrame called **df** is created by copying the data from the **data** DataFrame in a structured format using **pd.DataFrame**. Then the entire contents of the **df** DataFrame were displayed.

b. Write a Python program to remove unnecessary columns i.e., salary and salary currency.

```
In [3]:  #remove columns salary and salary currency
         df.drop(columns =['salary','salary_currency'], inplace= True)
         df
```

Out[3]:

	work_year	experience_level	employment_type	job_title	salary_in_usd
0	2023	SE	FT	Principal Data Scientist	85847
1	2023	MI	CT	ML Engineer	30000
2	2023	MI	CT	ML Engineer	25500
3	2023	SE	FT	Data Scientist	175000
4	2023	SE	FT	Data Scientist	120000
...	...	...	...	...	...
3750	2020	SE	FT	Data Scientist	412000
3751	2021	MI	FT	Principal Data Scientist	151000
3752	2020	EN	FT	Data Scientist	105000
3753	2020	EN	CT	Business Data Analyst	100000
3754	2021	SE	FT	Data Science Manager	94665

3755 rows × 9 columns

Figure 15 Remove unnecessary columns.

The **df.drop()** function removes the specified columns from the DataFrame. The argument **columns = ['salary', 'salary_currency']** specify the names of the columns to be removed, and **in place=True** ensures that the changes are made directly to the original DataFrame. The resulting frame will no longer contain the removed columns i.e. salary and salary_currency.

c. Write a Python program to remove the NaN missing values from the updated DataFrame.

```
In [4]: #remove the NaN missing values from updated DataFrame
        df = df.dropna()
        df
```

Out[4]:

	work_year	experience_level	employment_type	job_title	salary_in_usd
0	2023	SE	FT	Principal Data Scientist	85847
1	2023	MI	CT	ML Engineer	30000
2	2023	MI	CT	ML Engineer	25500
3	2023	SE	FT	Data Scientist	175000
4	2023	SE	FT	Data Scientist	120000
...	...	...	...	...	...
3750	2020	SE	FT	Data Scientist	412000
3751	2021	MI	FT	Principal Data Scientist	151000
3752	2020	EN	FT	Data Scientist	105000
3753	2020	EN	CT	Business Data Analyst	100000
3754	2021	SE	FT	Data Science Manager	94665

3755 rows × 9 columns

Figure 16 Remove the NaN missing values.

The **df.dropna()** function removes rows or columns containing missing values which are represented by NaN in Pandas in the DataFrame. The resulting data frame contains rows or columns without any NaN values. Here, there were not any NaN values, so no changes were made in the DataFrame.

d. Write a Python program to check duplicate values in the DataFrame.

```
In [5]:  #check duplicates value
         duplicate = df[df.duplicated()]
         duplicate
```

Out[5]:

	work_year	experience_level	employment_type	job_title	salary_in_usd
115	2023	SE	FT	Data Scientist	150000
123	2023	SE	FT	Analytics Engineer	289800
153	2023	MI	FT	Data Engineer	100000
154	2023	MI	FT	Data Engineer	70000
160	2023	SE	FT	Data Engineer	115000
...	...	...	...	...	...
3439	2022	MI	FT	Data Scientist	78000
3440	2022	SE	FT	Data Engineer	135000
3441	2022	SE	FT	Data Engineer	115000
3586	2021	MI	FT	Data Engineer	200000
3709	2021	MI	FT	Data Scientist	90734

1171 rows × 9 columns

Figure 17 Check duplicate value.

The **df.duplicated()** function identifies rows that are duplicates based on all columns in the DataFrame and stores it into a new DataFrame named **duplicate**. Then, the **duplicate** DataFrame will contain the rows that are duplicates of other rows in the original DataFrame **df**.

e. Write a Python program to see the unique values from all the columns in the DataFrame.

```
In [7]: #unique values from all the columns in the DataFrame
unique = {}
for column in df.columns:
    unique = df[column].unique()
    print(f"Unique values in column '{column}':")
    print (unique)
    print()
```

```
Unique values in column 'work_year':
[2023 2022 2020 2021]

Unique values in column 'experience_level':
['SE' 'MI' 'EN' 'EX']

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 Engin
eer'
 '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 Scie
ntist'
 'Data Analytics Manager' 'Business Data Analyst' 'Applied Data Sc
ientist'
 'Staff Data Analyst' 'ETL Engineer' 'Data DevOps Engineer' 'Head
of Data'
 'Data Science Manager' 'Data Manager' 'Machine Learning Researche
r'
 '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 Scienti
st'
 'Lead Data Scientist' 'Cloud Database Engineer' 'Financial Data A
nalyst'
 'Data Infrastructure Engineer' 'Software Data Engineer' 'AI Progr
ammer'
 'Data Operations Engineer' 'BI Developer' 'Data Science Lead'
 'Deep Learning Researcher' 'BI Analyst' 'Data Science Consultant'
 'Data Analytics Specialist' 'Machine Learning Infrastructure Engi
neer'
 '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 Develo
per'
 '3D Computer Vision Researcher' 'Principal Machine Learning Engin
eer'
```

Figure 18 See the unique values from all the columns.

The **unique = {}** initializes an empty dictionary called **unique**. Then a loop is iterated over each column in the DataFrame **df**. The **df[column].unique()** retrieves unique values present in the column and the result is stored in the **unique** dictionary. Then the **printing** is done sequentially using the print() function; firstly, the header indicates the column name, secondly the unique values from that column, and lastly an empty line for readability.

f. Rename the experience level columns as below.

SE – Senior Level/Expert

```
In [8]: #rename SE to Expert
        df['experience_level'] = df['experience_level'].replace('SE','Expert')
        df
```

Out[8]:

	work_year	experience_level	employment_type	job_title	salary_in_usd	employee_residence	remote_ratio	company_location	company_size
0	2023	Expert	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	Expert	FT	Data Scientist	175000	CA	100	CA	M
4	2023	Expert	FT	Data Scientist	120000	CA	100	CA	M
...	...	...	...	...	...	...	...	...	...
3750	2020	Expert	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	Expert	FT	Data Science Manager	94665	IN	50	IN	L

3755 rows × 9 columns

Figure 19 Rename SE to Expert

The .**replace()** function is used to replace **SE** with **Expert** in the **'experience_level'** column. Then the DataFrame **df** will update the values in the **'experience_level'** column.

MI – Medium Level/Intermediate

```
In [9]: #rename MI to Intermediate
        df['experience_level'] = df['experience_level'].replace('MI','Intermediate')
        df
```

Out[9]:

	work_year	experience_level	employment_type	job_title	salary_in_usd	employee_residence	remote_ratio	company_location	company_size
0	2023	Expert	FT	Principal Data Scientist	85847	ES	100	ES	L
1	2023	Intermediate	CT	ML Engineer	30000	US	100	US	S
2	2023	Intermediate	CT	ML Engineer	25500	US	100	US	S
3	2023	Expert	FT	Data Scientist	175000	CA	100	CA	M
4	2023	Expert	FT	Data Scientist	120000	CA	100	CA	M
...	...	...	...	...	...	...	...	...	...
3750	2020	Expert	FT	Data Scientist	412000	US	100	US	L
3751	2021	Intermediate	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	Expert	FT	Data Science Manager	94665	IN	50	IN	L

3755 rows × 9 columns

Figure 20 Rename MI to Intermediate

The **.replace()** function is used to replace **MI** with **Intermediate** in the **'experience_level'** column. Then the DataFrame **df** will update the values in the **'experience_level'** column.

EN – Entry Level

```
In [10]:
#rename EN to Entry Level
df['experience_level'] = df['experience_level'].replace('EN','Entry Level')
df
```

Out[10]:

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

3755 rows × 9 columns

Figure 21 Rename EN to Entry Level

The **.replace()** function is used to replace **EN** to **Entry Level** in the **'experience_level'** column. Then the DataFrame **df** will update the values in the **'experience_level'** column.

EX – Executive Level

```
In [11]:
#rename EX to Executive Level
df['experience_level'] = df['experience_level'].replace('EX','Executive Level')
df
```

Out[11]:

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

3755 rows × 9 columns

Figure 22 Rename EX to Executive Level

The **.replace()** function is used to replace **EX** with **Executive Level** in the **'experience_level'** column. Then the DataFrame **df** will update the values in the **'experience_level'** column.

4. Data Analysis

Data Analysis is the practice of working with data to evaluate, clean, manipulate, and model data to identify relevant data, make conclusions, and enhance decision-making. It includes a variety of methods and approaches for interpreting data from different sources, both structured and unstructured (Datacamp, 2023).

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

```
In [37]: chosen_variable = input ("Enter the desired variable: ")
summarize = df[chosen_variable].sum()
meean = df[chosen_variable].mean()
standard_deviation = df[chosen_variable].std()
skewness = df[chosen_variable].skew()
kurtosiss = df[chosen_variable].kurtosis()

# print summary statistics
print()
print(f"Summary Statistics for {chosen_variable}")
print(f"Sum: {summarize}")
print(f"Mean: {meean}")
print(f"Standard Deviation: {standard_deviation}")
print(f"Skewness: {skewness}")
print(f"Kurtosis: {kurtosiss}")

Enter the desired variable: salary_in_usd

Summary Statistics for salary_in_usd
Sum: 344729580
Mean: 133409.28018575851
Standard Deviation: 67136.83732925021
Skewness: 0.6203168790580038
Kurtosis: 0.8269400876861832
```

Figure 23 Summary statistics of sum, mean, standard deviation, skewness, and kurtosis of any chosen variable.

The user enters the desired variable in the DataFrame that they want to summarize statistics of. The sum, mean, standard deviation, skewness, and kurtosis were determined using the build-in functions i.e. like **sum(), mean(), std(), skew(), and kurtosis()**. The **print()** function is used to format strings before printing a summary of the statistics for the selected variable.

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

```
In [38]: df.corr(numeric_only = True)
```

Out[38]:

	work_year	salary_in_usd	remote_ratio
work_year	1.000000	0.236958	-0.219160
salary_in_usd	0.236958	1.000000	-0.084502
remote_ratio	-0.219160	-0.084502	1.000000

Figure 24 Display correlation.

The **corr()** function computes the correlation matrix for the numeric columns in a Pandas DataFrame. The **numeric_only = True** argument ensures that only numeric columns are considered in the correlation calculation.

The resulting matrix is symmetric with diagonal elements always equal to 1 since a variable is perfectly correlated with itself. The off-diagonal values represent the correlation between pairs of variables. The values close to 1 indicate a strong positive correlation, the values close to -1 indicate a strong negative correlation, and the values close to 0 indicate little to no linear relationship.

5. Data Exploration

Data Exploration is the process of utilizing data visualization tools and statistical techniques to analyze raw datasets to gain familiarity, insights, and context. It is an effective method for detecting enabled relationships or anomalies due to elements such as colors, lines, graphs, etc. of data visualization (Robinson, 2021).

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

```
In [22]: # Get the top 15 jobs
         top_jobs = df['job_title'].value_counts().head(15)
         top_jobs

Out[22]: job_title
         Data Engineer                653
         Data Scientist               569
         Data Analyst                 455
         Machine Learning Engineer    271
         Data Manager                  94
         Analytics Engineer            91
         Data Architect                68
         Research Scientist            65
         Machine Learning Scientist    38
         Research Engineer             33
         Applied Scientist             31
         Computer Vision Engineer      23
         Data Science Consultant       23
         Head of Data Science          19
         AI Scientist                  16
         Name: count, dtype: int64

In [20]: #create bar graph
         top_jobs.plot(kind='bar')
         plt.xlabel("Job Titles")
         plt.ylabel("Values")
         plt.title("Bar Graph of Top 15 Jobs")

Out[20]: Text(0.5, 1.0, 'Bar Graph of Top 15 Jobs')
```

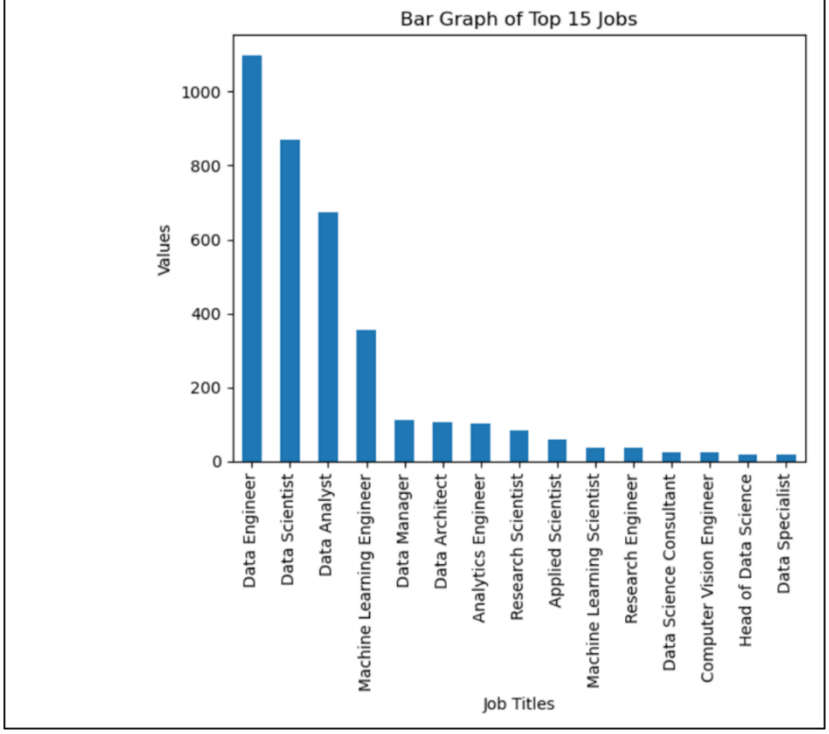

Figure 25 Find out the top 15 jobs and make a bar graph.

The **df[‘job_title’]** selects the **'job_title'** column from the DataFrame **df** and the **.value_counts()** function counts the unique values in the **'job_title'** column. The **.head(15)** function then takes the first 15 entries from the Series created by value_counts() which are usually the most frequent values. The result is stored in the new Series called **top_jobs** and displayed.

In the **top_jobs.plot(kind='bar'),** the **.plot()** function creates a visualization of the **top_jobs** Series. The **kind='bar'** parameter specifies that the plot should be a bar graph. The **plt.xlabel** sets the label for the x-axis as “**Job Titles”** and **plt.ylabel** sets the label for the y-axis as **“Values”**. The **plt.title** sets the title of the bar graph as **"Bar Graph of Top 15 Jobs".**

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

```
In [23]:
# get highest salaries
high_salary = df.sort_values(by = 'salary_in_usd', ascending = False).head(5)
high_salary [['job_title', 'salary_in_usd']]
```

Out[23]:

	job_title	salary_in_usd
3522	Research Scientist	450000
2011	Data Analyst	430967
528	AI Scientist	423834
3747	Machine Learning Scientist	423000
3675	Data Scientist	416000

```
In [24]:
#creating bar
high_salary.plot(kind='bar', x='job_title', y='salary_in_usd')
plt.xlabel("Job Titles")
plt.ylabel("Salaries")
plt.title("Bar Graph of Highest Salaries")
```

```
Out[24]: Text(0.5, 1.0, 'Bar Graph of Highest Salaries')
```

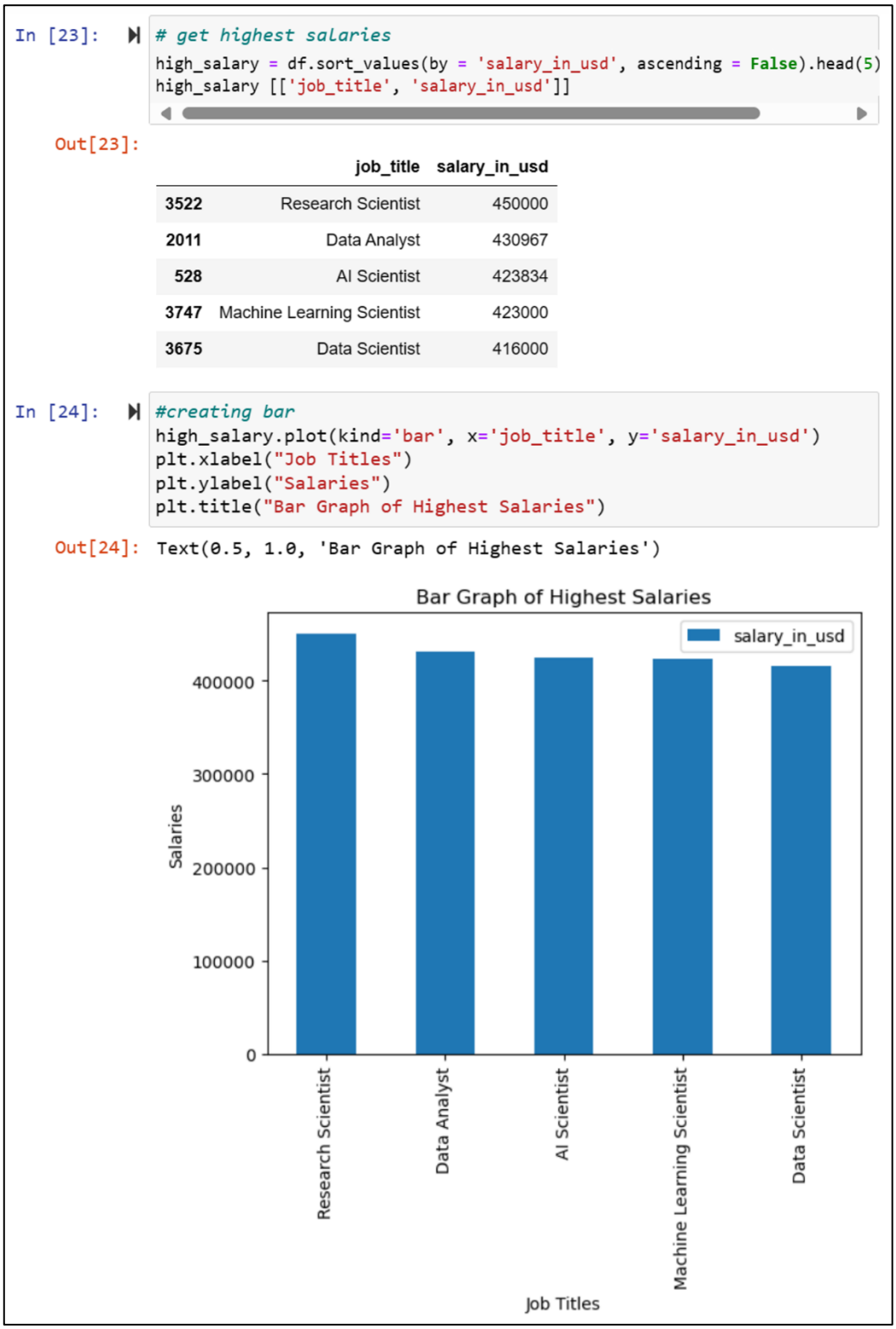

Figure 26 Find out the highest salaries and make a bar graph

The **df.sort_values(by = 'salary_in_usd', ascending = False)** function sorts the DataFrame **df** based on the column **'salary_in_usd'** in descending order i.e.

from highest to lowest. The **.head(7)** function then takes the first 7 rows from the DataFrame **df** and the result is stored in the new DataFrame called **high_salary.** The **high_salary [['job_title', 'salary_in_usd']]** selects only the **'job_title'** and **'salary_in_usd'** columns to show relevant information.

Then **.plot()** function creates a visualization of the **high_salary** DataFrame. The **kind='bar'** parameter specifies it to be a bar graph. The **x = 'job_title'** sets the column **'job_title'** from the DataFrame as the label for the x-axis and **y = 'salary_in_usd'** sets the column **'salary_in_usd'** from the DataFrame as the height of the bars representing the values on the y-axis. The **plt.xlabel** sets the label for the x-axis as **"Job Titles"** and **plt.ylabel** sets the label for the y-axis as **"Salaries".** The **plt.title** sets the title of the bar graph as **"Bar Graph of Highest Salaries".**

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

```
In [36]: # get salaries based on experience level
         experience_salary = df.groupby('experience_level')['salary_in_usd']
         experience_salary

Out[36]: experience_level
         Entry Level        300000
         Executive Level    416000
         Expert             423834
         Intermediate       450000
         Name: salary_in_usd, dtype: int64

In [37]: #create bar graph
         experience_salary.plot(kind='bar')
         plt.xlabel("Experience level")
         plt.ylabel("Salaries")
         plt.title("Bar Graph of Salaries based on Experience level")

Out[37]: Text(0.5, 1.0, 'Bar Graph of Salaries based on Experience level')
```

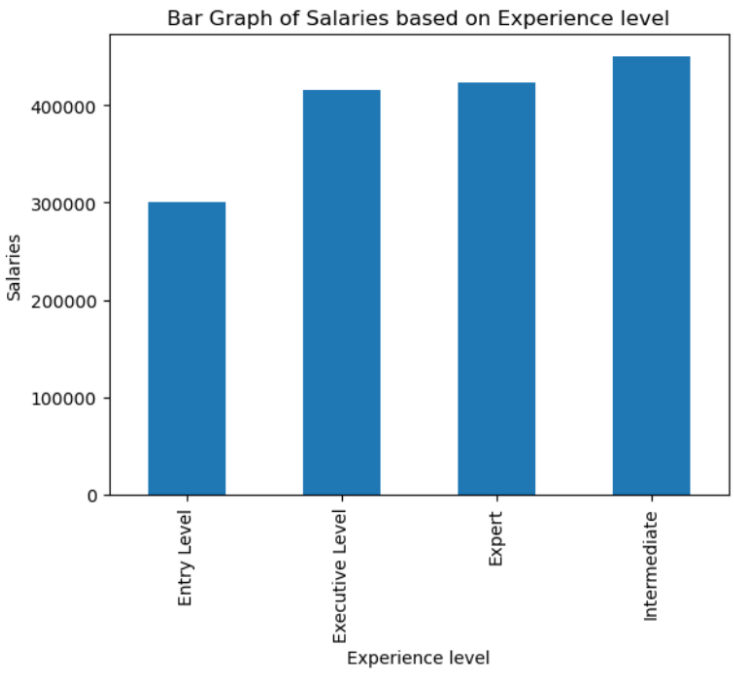

Figure 27 Salaries based on experience level.

The **df.groupby('experience_level')** groups the DataFrame **df** based on the values in the **'experience_level'** column. It produces groupings of rows consisting of values with the same experience level. Then **['salary_in_usd'].max()** selects the **'salary_in_usd'** column and applies the **max()** function to find the maximum value of 'salary_in_usd' within each group. The result is stored in the new Series called **experience_salary** and displayed.

Then **.plot()** function creates a visualization of the **experience_salary** Series. The **kind='bar'** parameter specifies it to be a bar graph. The **plt.xlabel** sets the label for the x-axis as “**Experience Level**” and **plt.ylabel** sets the label for the y-axis as **“Salaries”.** The **plt.title** sets the title of the bar graph as **"Bar Graph of Salaries based on Experience level".**

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

```
In [39]:  #create histogram
          plt.hist(df['company_size'], label="Company Size")
          plt.hist(df['employment_type'], label="Employment Type")
          plt.legend(loc="upper right")
          plt.title("Histogram of variables Company Size and Employment Type")
          plt.xlabel("Values")
          plt.ylabel("Frequency")

Out[39]: Text(0, 0.5, 'Frequency')
```

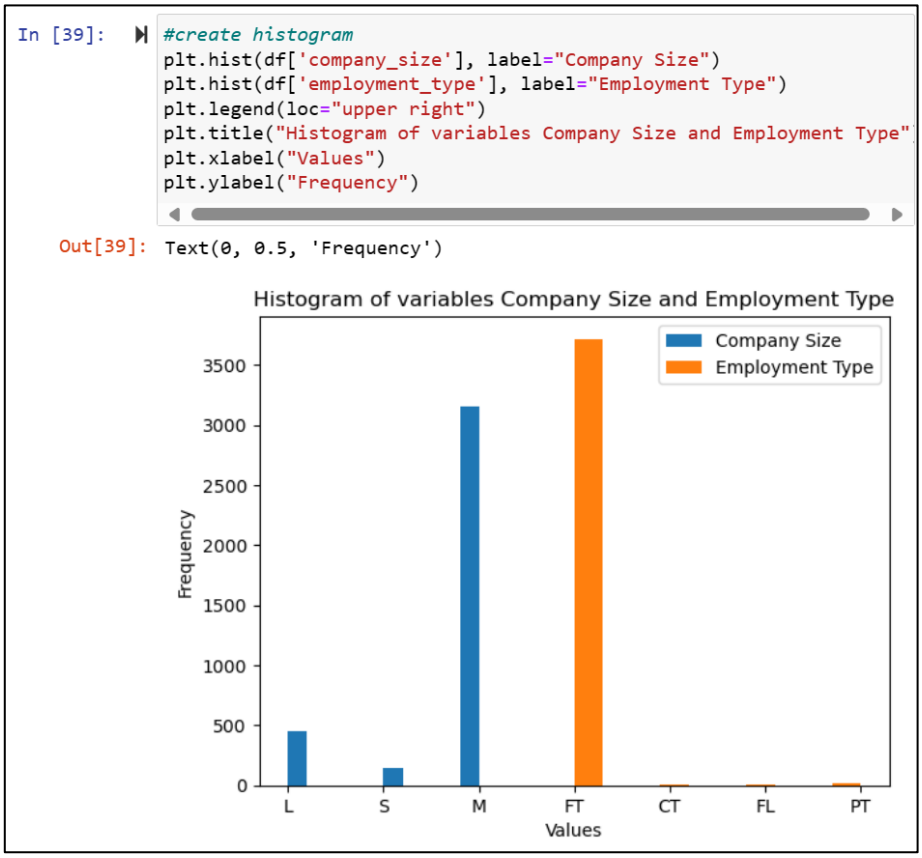

Figure 28 Histogram.

A histogram is a graphical representation of the distribution of a set of continuous data. It visualizes how frequently each data point appears in a specific range of values (JasperSoft, 2022).

The **plt.hist(df['company_size'], label="Company Size")** creates a histogram for the data in the **'company_size'** column and **plt.hist(df['employment_type'], label="Employment Type")** creates a histogram for the data in the **'employment_type'** column of the DataFrame **df.** The **label** parameter is used to identify the histogram in the legend.

The **plt.legend(loc="upper right")** adds a legend to the plot and **loc** parameter specifies the location of the legend on the plot to be upper right corner. The **plt.xlabel** sets the label for the x-axis as "**Values**" and **plt.ylabel** sets the label for the y-axis as **"Frequency".** The **plt.title** sets the title of the histogram as **"Histogram of variables Company Size and Employment Type".**

```
In [7]:  #create box plot
         df['work_year'].plot(kind='box')
         plt.ylabel("years")
         plt.title("Box Plot of Work Year")

Out[7]:  Text(0.5, 1.0, 'Box Plot of Work Year')
```

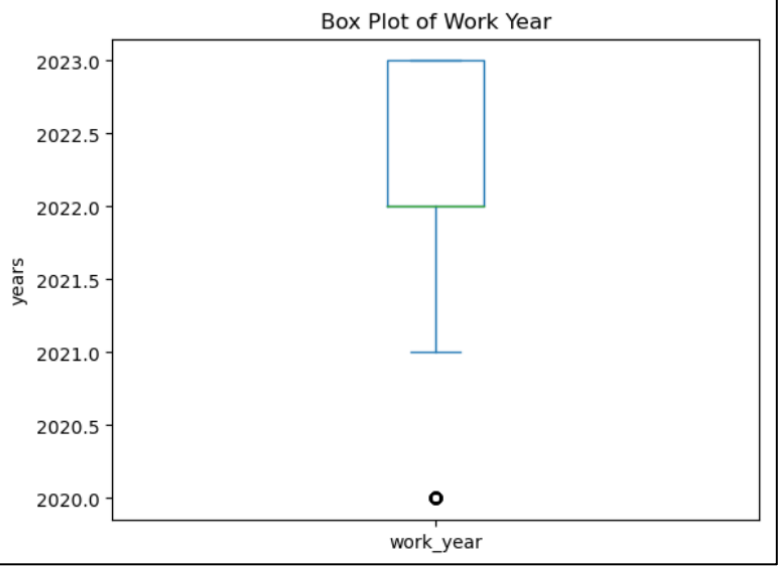

Figure 29 Box Plot.

The boxplot is a graph that displays the distribution of the dataset based on the based on five-number summary i.e. minimum, first quartile (Q1), median, third quartile (Q3), and maximum. It gives a visual indication of how a dataset's 25th percentile, 50th percentile, 75th percentile, minimum, maximum, and outlier values are spread out and compared to each other *(Galarnyk, 2023)*.

The whiskers are the lines that extend from the bottom of the box to the lowest values within a certain range. The Upper Quartile (Q3) is the top edge of the box, indicating the 75th percentile. The median is the green line inside the box, indicating the 50th percentile. The Lower Quartile (Q1) is the bottom edge of the box which is covered by the median, indicating the 25th percentile. The minimum is the end of the lower whisker, indicating the smallest value excluding any outliers. The Outlier is a data point that falls outside of the typical range.

The **df['work_year'].plot(kind='box')** creates a boxplot for the data in the **'work_year'** column of the DataFrame **df**. The **plt.ylabel** sets the label for the y-axis as "years". The **plt.title** sets the title of the histogram as **"Box Plot of Work Year"**.

6. Conclusion

This coursework involves understanding dataset Data Science salaries and writing a program in Python to read and clean data, summarize statistics and correlation, and create visualizations like histograms, bar graphs, and boxplots for data understanding, preparation, analysis, and exploration. The dataset involved data that can influence salary like experience level, employment type, work year, job title, and many more.

The Python libraries i.e. Pandas and Matplotlib were used to execute the requirements of the coursework. Pandas provided structure to the data and provided analysis tools. Matplotlib provided interactive visualization features for better data analysis.

This coursework gave us a better understanding of applying programming skills and analysis skills to real-world datasets. It gave insight into the process of data preparation for further data mining and analysis.

7. References

Anaconda , 2018. *Anaconda Distribution.* [Online]
Available at: https://docs.anaconda.com/free/anaconda/index.html
[Accessed 5 April 2024].

Datacamp, 2023. *What is Data Analysis? An Expert Guide With Examples.* [Online]
Available at: https://www.datacamp.com/blog/what-is-data-analysis-expert-guide
[Accessed 7 April 2024].

Galarnyk, M., 2023. *Understanding Boxplots.* [Online]
Available at: https://builtin.com/data-science/boxplot
[Accessed 15 April 2024].

GeeksforGeeks, 2024. *What is Data Preparation?.* [Online]
Available at: https://www.geeksforgeeks.org/what-is-data-preparation/
[Accessed 7 April 2024].

IBM, 2021. *Data Understanding Overview.* [Online]
Available at: https://www.ibm.com/docs/en/spss-modeler/saas?topic=understanding-data-overview
[Accessed 7 April 2024].

JasperSoft, 2022. *What is a Histogram Chart?.* [Online]
Available at: https://www.jaspersoft.com/articles/what-is-a-histogram-chart
[Accessed 15 April 2024].

Jupyter , 2024. *Try Jupyter.* [Online]
Available at: https://jupyter.org/try
[Accessed 5 April 2024].

Matplotlib, 2023. *Matplotlib: Visualization with Python.* [Online]
Available at: https://matplotlib.org/
[Accessed 5 April 2024].

pandas, 2024. *Pandas Documentaion.* [Online]
Available at: https://pandas.pydata.org/docs/#
[Accessed 5 April 2024].

Robinson, S., 2021. *data exploration.* [Online]
Available at: https://www.techtarget.com/searchbusinessanalytics/definition/data-exploration
[Accessed 7 April 2024].

Rossum, G. V., 2007. *Python (programming language)..* [Online]
Available at: http://kelas-karyawan-bali.kurikulum.org/IT/en/2420-2301/Python_3721_kelas-karyawan-bali-kurikulumngetesumum.html
[Accessed 6 April 2024].

we3schools, 2024. *NumPy Introduction.* [Online]
Available at: https://www.w3schools.com/python/numpy/numpy_intro.asp
[Accessed 5 April 2024].