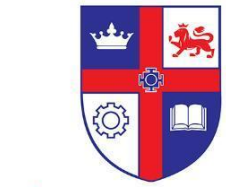
LONDON METROPOLITAN UNIVERSITY



Islington college
(इस्लिङ्टन कलेज)

CC5067NI-Smart Data Discovery

60% Individual Coursework

2023-24 Spring

Student Name: Aayush Poudel

London Met ID: 22085487

College ID: NP01CP4S230096

Assignment Due Date: Monday, May 13, 2024

Assignment Submission Date: Monday, May 13, 2024

Word Count: 5783

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

Table of Contents

Table of Figure

Table of Table

1. Introduction to Smart data Discovery

Smart data discovery is the process of using advanced algorithms and technologies to automatically analyze large datasets uncovering valuable insights and patterns without the need for manual exploration. Through this you will gain practical experience in extracting valuable insights without manual exploration Get ready to explore into the world of data discovery and unlock hidden Things.

Welcome to our data science salary analysis we will use python program to figure out what things affect how much money data scientist make. In this assignment, we are going to explore how much money data scientists make and what things affect their salaries. Python program will help us to find out this things it's like being a detective, but instead of searching for clues, we will be searching for answers in numbers. By the end of this assignment, I'll have a better understanding of data science salaries and some cool new skills in Python programming and data analysis.

In this assignment, we will use Python to analyze data science salaries. We will start by understanding our dataset, cleaning it up, and then analyzing it to find trends. Using Python programming, we will explore factors like job titles, experience levels, and their impact on salaries. Through visualizations, we will uncover insights like top-paying jobs and salary variations based on experience. By the end, you'll have a better grasp of data science salaries and gain valuable skills in Python and data analysis.

As we reach the end, you'll find yourself with a clearer understanding of data science salaries and a toolkit enriched with essential skills in Python programming and data analysis. So, brace yourself for an immersive journey into the captivating realm of data science salaries. The world of data science salaries is a vast and intriguing landscape, teeming with a plethora of captivating discoveries and meaningful insights that await the inquisitive mind. As we journey through this realm, you will find yourself immersed in a wealth of data, each piece holding the potential to unveil important truths about the trends, patterns, and intricacies of the data science profession. (© Cyral Inc, n.d.)

1.1 Aims and Objective

Aims:

The main goal of this project is to carefully get the dataset ready for future data mining and analysis. We will work to clean up any unwanted parts and make sure everything is organized neatly .This involves identifying and removing any irrelevant or unnecessary information from the dataset for example, if the dataset contains columns or variables that are not relevant to the By doing this we can make it easier to find important information from the data in future our main aim is to help decision makers make smart choices based on accurate and reliable information. The ultimate goal of cleaning and preparing the dataset is to make it ready for data mining and analysis and finally overall the goal is to carefully preparing the dataset is to empower decisions makers with accurate reliable (Saint Peter's University, n.d.).

Objectives:

- To understand the structure and characteristics of the dataset identifying relevant variables and potential data challenges
- To prepare the dataset for advanced analytical techniques ensuring that it is optimized for accurate and meaningful data mining outcomes.
- To document the data preparation process thoroughly providing transparency and reproducibility for future analyses and interpretations.
- To look closely at the data and find out some initial clues about it I will try to see if there are any interesting patterns or trends that I can explore more later on
- To Identify key features that are most relevant to the analysis and decision-making process, removing less informative features to improve model efficiency and interpretability
- To evaluate the dataset and its use from an ethical perspective, ensuring that the data is being used responsibly and transparently, and that any privacy concerns are addressed.
- Evaluate differences in salaries and job satisfaction between full-time and part-time data science roles, if this information is available in the dataset.

2. Data Understanding

Understanding data is an essential initial phase in all data analysis projects since it sets the groundwork for tasks like cleaning, exploring, and modeling that come afterward. Analyzing the dataset involves gaining an understanding of its organization, information, and possible problems. This involves comprehending the data's origin and collection method, format, structure, and schema, which encompasses column names and data types. Evaluating the quantity and dimensions of data offers understanding into the dataset's magnitude and intricacy. Assessing data quality problems like missing values, inconsistencies, and outliers can guide the appropriate response. Statistical summaries and initial visualizations give a rapid glimpse into central tendencies, variability, and potential patterns. (Saint Peter's University, n.d.) Visualizations or statistical measures can be utilized to investigate connections and associations among variables in relationships. When we're diving into a bunch of data, the first step is to get a quick overview. We do this by looking at some numbers that summarize the data, like averages and spreads, and maybe some simple charts to see if any patterns jump out at us. Once we've got a handle on the basics, we start digging deeper to see how different pieces of the data are related. We might compare things like how one variable changes when another one does, or look for groups or clusters within the data. There are lots of tools and techniques for this, like making scatterplots, histograms, or heat maps. By using these tricks, we can uncover some really interesting stuff in the data and figure out what it's trying to tell us.

The dataset provided is called data science salaries and is in a CSV (Comma Separated Values) file format. CSV files are widely used for data analysis because they are easy to read and write, and are compatible with numerous applications and programming languages. The dataset includes details on different variables that may impact the salaries of employees. The employment period covered in the available data ranges from 2020 to the most recent year 2023, displaying the wages of workers across four distinct years. In this place, the staff members are categorized into four different levels of experience. Levels of experience range from senior, intermediate, entry, to executive. The salaries of the employees appear to be greatly influenced by their levels of experience. Next, we consider the amount of time employees dedicate to their job, specifically whether they work part-time or full-time. All job positions appear to be related to data science and AI/Machine Learning, including Business Data Analyst, Data Engineer, ML Engineer, Data

Science Manager, Data Specialist, AI Scientist, Power BI Developer, and more. Salaries are diverse and are presented in different currencies such as euros, US dollars, Indian rupees, etc. Despite there being a distinct column dedicated to displaying all salaries in US dollars. Next is the column for employee residence, which indicates where the employees currently reside. In a vast number of companies, the remote ratio can range from 0% to 100% according to the dataset, with a majority of companies having a remote ratio of 100%. This indicates that a large number of employees have been working from their homes. This dataset also includes data on the company's location and its size. The company varies in size from small to large.

Data mining is like searching for hidden gems within a vast mine of data. It's the process of exploring and analyzing large datasets to discover patterns, relationships, and insights that may not be immediately obvious. Think of it as sifting through a mountain of information to find the gold nuggets buried within. In the context of the provided dataset on data science salaries, data mining could involve uncovering trends in salary distribution based on variables such as experience level, job position, currency, and company size. By applying data mining techniques, analysts can identify factors that significantly impact salary levels, such as years of experience or geographic location. Additionally, data mining can help identify outliers or anomalies in the data, such as unusually high or low salaries compared to the norm, which may warrant further investigation. Data mining uncovers correlations like job position and remote work, or company size and salary, empowering data-driven decisions.

```
In [35]: df.info() # Displaying the concise summary of the DataFrame 'df'

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3755 entries, 0 to 3754
Data columns (total 9 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_in_usd       3755 non-null   int64 
 5   employee_residence  3755 non-null   object
 6   remote_ratio        3755 non-null   int64 
 7   company_location    3755 non-null   object
 8   company_size        3755 non-null   object
dtypes: int64(3), object(6)
memory usage: 264.2+ KB
```

Figure 1 To show info information of data frame

The table with related information from the dataset along with their description and the data type have been presented below;

Table 1 Representation of all the columns of dataset

S.N	Column Name	Description	Datatype
0	Work year	It refers to the duration of time an individual has been employed or actively working within a specific job or industry.	Int 64
1	Experience level	It indicates the relative expertise and knowledge a person possesses in relation to their job or industry	Object
2	Employment type	it refers to the nature or classification of an individual's job arrangement, indicating the terms of their employment or engagement with an organization	Object
3	Job title	It identifies the specific role or position held by an individual within an organization or industry	Object
4	Salary in usd	It refers to the monetary compensation earned by an individual for their work, denominated in United States dollars (USD)	Int 64
5	Employee residence	It refers to the primary place of dwelling or location where an individual resides.	Object
6	Remote ratio	It indicates the proportion or percentage of work performed remotely by an employee, relative to their total working hours or tasks	Int 64
7	Company location	It refers to the physical geographical location where a company's main office, headquarters, or primary business operations are situated	Object
8	Company size	It refers to the measure of a company's scale or magnitude, typically determined by the number of employees, annual revenue, or market capitalization	Object
9	salary	Salary refers to the fixed regular payment, typically paid on a monthly basis, received by an employee in exchange for their work or services rendered to an organization.	Int 64
10	Salary currency	Salary in currency refers to the monetary compensation earned by an individual for their work or services rendered, denominated in a specific currency other than the local currency.	0bject

The table provides an overview of the "Data Science Salaries" dataset, including columns like Work Year, Experience Level, Employment Type, Job Title, Salary in USD, Employee Residence, Remote Ratio, Company Location, and Company Size. These attributes offer insights into employee salaries, job roles, work arrangements, and company characteristics, facilitating analysis of salary trends and workplace dynamics. data on work year allows us to track salary changes over time, identifying trends and patterns in compensation across different years. Experience level provides information on the seniority and expertise of employees, influencing salary levels and career progression within the data science field. Employment type delineates between various job arrangements, such as full-time, part-time, or contract-based roles, shedding light on different employment structures and their impact on salaries. Job title serves as a key identifier for different roles within the industry, reflecting the diversity of positions and responsibilities in the data science domain. Salary in USD quantifies the monetary compensation received by employees, allowing for direct comparison and analysis of salary distributions. Employee residence provides geographical insights into where employees are based, potentially influencing salary levels based on regional cost of living and demand for skills. Remote ratio indicates the proportion of work performed remotely, reflecting the growing trend of remote work arrangements and their implications for salary and job satisfaction. Company location and size offer contextual information about the organizations where employees work, contributing to our understanding of salary disparities based on company location, size, and industry sector. Overall, these attributes collectively contribute to a comprehensive analysis of data science salaries, enabling stakeholders to make informed decisions and identify opportunities for improvement within the industry.

3. Data preparation

3.1. Write a python program to load data into pandas Data Frame.

```
## Importing pandas and aliasing as pd

In [1]: import pandas as pd  # Importing the pandas library and assigning it an alias 'pd' for easier usage in the code
        import matplotlib.pyplot as plt # Import the matplotlib.pyplot module and give it the alias 'plt' for convenience in plotting graphs
```

Figure 2 Importing pandas and aliasing as pd

Description: At first in python, bringing in the pandas library and giving it the alias ‘pd’ can be done with the straightforward command: import pandas as pd. This one code line allows access to the wide range of features in pandas throughout the script by using the shorter name ‘pd’. This practice improves the clarity and brevity of code by minimizing the frequency of typing ‘pandas’ when calling its functions or properties. In the Python community, ‘pd’ is commonly used as a standardized abbreviation for pandas, promoting uniformity among projects. This method simplifies coding, enhancing the effectiveness of working with pandas’ tools for data manipulation and analysis.

```
## A python program to load data into pandas Data

In [2]: df= pd.read_csv('DataScienceSalaries.csv')# Load the dataset from a CSV file into a pandas DataFrame called 'df'
        df # Show the contents of the DataFrame to inspect the data loaded from the CSV file

Out[2]:
```

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

Figure 3 program to load data into panda’s data

Description: In the figure displayed above, I have utilized the pandas library, which is a robust tool for data manipulation. My initial step is to add this library to our program by using the statement ‘import pandas as pd’. This enables us to utilize all the features and resources offered by pandas across our code. Later, I employ a particular function in pandas known as ‘read_csv()’. This function is created to extract information from CSV files, which are often utilized for holding tabular data such as spreadsheets. I substitute ‘your_data.csv’ with the real path to our dataset when passing the CSV file name as an argument to this function.

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

Without inplace= true the outcomes

A python program to remove unnecessary columns i.e., salary and salary currency

```
In [5]: df.drop(columns=['salary', 'salary_currency']) # Remove the 'salary' and 'salary_currency' columns from the DataFrame 'df'
        df # The DataFrame is modified in-place, so the changes are made directly to 'df'
```

Out[5]:

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

3755 rows × 11 columns

Figure 4 Without using inplace true

Using inplace=True in pandas operations modifies the DataFrame in place, meaning that the changes are applied directly to the original DataFrame without creating a new copy. This can be advantageous as it saves memory and processing time, especially when working with large datasets, as it avoids the need to create a separate copy of the DataFrame. However, it's important to use inplace=True with caution, as it can lead to unexpected results if not used properly.

On the other hand, without using inplace=True, the operations return a new DataFrame with the changes applied, leaving the original DataFrame unchanged. While this approach preserves the original DataFrame, it may require additional memory and processing time to create a new copy, especially for large datasets. Additionally, chaining multiple operations without using inplace=True can lead to code that is more concise and easier to understand, as each operation creates a new DataFrame that can be used in subsequent operations. However, it's essential to assign the result of each operation to a new variable to avoid losing the original DataFrame. Overall, the choice between using inplace=True and not using it depends on the specific requirements of the analysis and the desired workflow.

A python program to remove unnecessary columns i.e., salary and salary currency

In [3]:
```
df.drop(columns=['salary', 'salary_currency'], inplace=True) # Remove the 'salary' and 'salary_currency' columns from the DataFrame 'df'
df # The DataFrame is modified in-place, so the changes are made directly to 'df'
```

Out[3]:

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

3755 rows × 9 columns

Figure 5 program to remove unnecessary columns i.e. salary and salary currency

Description; This Python script shows how to eliminate unneeded columns from a pandas DataFrame. Upon loading the data with the `read_csv()` function into a DataFrame, I then use the `drop()` method to eliminate the specified columns (‘salary’ and ‘salary_currency’). The use of `inplace=True` guarantees that the modifications are made directly to the original DataFrame and not to a new one. Ultimately, the program displays the DataFrame with the eliminated columns so you can double-check the modifications. This program simplifies the data preparation process for analysis by removing unneeded columns and emphasizing the important attributes in the dataset.

3.3. Write a python program to remove the NaN missing values from updated data frame.

A python program to remove the NaN missing values from updated data frame.

```
In [4]: df.dropna() # Remove rows with missing values from the DataFrame 'df'
```

Out[4]:

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

3755 rows × 9 columns

Figure 6 Program to remove Nan missing values from updated data frame

Description: I used the ‘dropna()’ function from the pandas library to locate and remove missing values identified as NaN in our dataset. This technique efficiently eliminates rows that have NaN values. By using ‘inplace=True’, I made sure that these changes were applied directly to the existing DataFrame instead of creating a new one. This procedure assisted us in tidying up our data by removing any incomplete or missing information, guaranteeing its accuracy and dependability for further analysis.

3.4. Write a python program to check duplicates value in the data frame.

A python program to check duplicates value in the data frame

```
In [5]: df[df.duplicated()] # Display rows in the DataFrame 'df' that contain duplicate values
```

Out[5]:

	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

1171 rows × 9 columns

Figure 7 A python program to check duplicate value in the data frame

Description: I used the ‘duplicated()’ method from the pandas library to find duplicate values in our dataset. This technique involves comparing every row in the DataFrame with every other row in order to identify precise duplicates. It results in a boolean Series that shows if each row is a duplicate or not. Analyzing the outcome of this process allows me to detect any identical records in our dataset, guaranteeing the accuracy and quality of our analysis.

3.5. Write a python program to see the unique values from all the columns in the data frame.

```
## A python program to see the unique values from all the columns in the dataframe
```

```
In [6]: for i in df.columns: # Iterate through each column in the DataFrame 'df' and print the unique values in each column
            print(df[i].unique())
```

```
[2023 2022 2020 2021]
['SE' 'MI' 'EN' 'EX']
['FT' 'CT' 'FL' 'PT']
['Principal Data Scientist' 'ML Engineer' 'Data Scientist'
 'Applied Scientist' 'Data Analyst' 'Data Modeler' 'Research Engineer'
 'Analytics Engineer' 'Business Intelligence Engineer'
 'Machine Learning Engineer' 'Data Strategist' 'Data Engineer'
 'Computer Vision Engineer' 'Data Quality Analyst'
 'Compliance Data Analyst' 'Data Architect'
 'Applied Machine Learning Engineer' 'AI Developer' 'Research Scientist'
 'Data Analytics Manager' 'Business Data Analyst' 'Applied Data Scientist'
 'Staff Data Analyst' 'ETL Engineer' 'Data DevOps Engineer' 'Head of Data'
 'Data Science Manager' 'Data Manager' 'Machine Learning Researcher'
 'Big Data Engineer' 'Data Specialist' 'Lead Data Analyst'
 'BI Data Engineer' 'Director of Data Science'
 'Machine Learning Scientist' 'MLOps Engineer' 'AI Scientist'
 'Autonomous Vehicle Technician' 'Applied Machine Learning Scientist'
 'Lead Data Scientist' 'Cloud Database Engineer' 'Financial Data Analyst'
 'Data Infrastructure Engineer' 'Software Data Engineer' 'AI Programmer'
 'Data Operations Engineer' 'BI Developer' 'Data Science Lead'
 'Deep Learning Researcher' 'BI Analyst' 'Data Science Consultant'
 'Data Analytics Specialist' 'Machine Learning Infrastructure Engineer'
 'BI Data Analyst' 'Head of Data Science' 'Insight Analyst'
 'Deep Learning Engineer' 'Machine Learning Software Engineer'
 'Big Data Architect' 'Product Data Analyst'
 'Computer Vision Software Engineer' 'Azure Data Engineer'
 'Marketing Data Engineer' 'Data Analytics Lead' 'Data Lead'
 'Data Science Engineer' 'Machine Learning Research Engineer'
 'NLP Engineer' 'Manager Data Management' 'Machine Learning Developer'
 '3D Computer Vision Researcher' 'Principal Machine Learning Engineer'
 'Data Analytics Engineer' 'Data Analytics Consultant'
```

Figure 8 A python program to see the unique values from all the columns

```
 'Data Management Specialist' 'Data Science Tech Lead'
 'Data Scientist Lead' 'Cloud Data Engineer' 'Data Operations Analyst'
 'Marketing Data Analyst' 'Power BI Developer' 'Product Data Scientist'
 'Principal Data Architect' 'Machine Learning Manager'
 'Lead Machine Learning Engineer' 'ETL Developer' 'Cloud Data Architect'
 'Lead Data Engineer' 'Head of Machine Learning' 'Principal Data Analyst'
 'Principal Data Engineer' 'Staff Data Scientist' 'Finance Data Analyst']
[ 85847  30000  25500 ...  28369 412000  94665]
['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']
[100   0  50]
['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']
['L' 'S' 'M']
```

Description: I initially used the ‘unique()’ method from the pandas library to view distinct values in every column of a DataFrame. This approach produces an array of distinct values found in every column of the DataFrame. By using this technique on every column separately or iterating through all columns, I can examine the distinct values throughout the whole DataFrame. This method helped us comprehend the range and variety of values in each column, facilitating data exploration and analysis.

3.6. Rename the experience level columns as below.

SE – Senior Level/Expert

MI – Medium Level/Intermediate

EN – Entry Level

EX – Executive Level

 # #Rename the experience level columns as below.

In [7]:
```
# Renaming the values in the 'experience_level' column
# The mapping dictionary specifies how to replace the existing values:
df['experience_level']=df['experience_level'].replace({
    # 'SE' becomes 'Senior Level/Expert'
    'SE': 'Senior Level/Expert',
    # 'MI' becomes 'Medium Level/Intermediate'
    'MI': 'Medium Level/Intermediate',
    # 'EN' becomes 'Entry Level'
    'EN': 'Entry Level',
    # 'EX' becomes 'Executive Level
    'EX': 'Executive Level'
})
# Display the updated DataFrame to verify changes in 'experience_level' column
df
```

Out[7]:

	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	Principal Data Scientist	85847	ES	100	ES	L
1	2023	Medium Level/Intermediate	CT	ML Engineer	30000	US	100	US	S
2	2023	Medium Level/Intermediate	CT	ML 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	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	Senior Level/Expert	FT	Data Science Manager	94665	IN	50	IN	L

3755 rows × 9 columns

Figure 9 renaming the experience level of columns

Description: In this task I changed the names of the experience level columns in our DataFrame to accurately represent their specific categories. I accomplished this by employing the 'replace()' function in the pandas library, enabling us to swap out certain values in the DataFrame with new values based on a pre-defined mapping. In particular, I offered a dictionary that connects each original experience level abbreviation ('SE', 'MI', 'EN', 'EX') with its respective descriptive label ('Senior Level/Expert', 'Medium Level/Intermediate', 'Entry Level', 'Executive Level'). By performing this task, I made sure that the experience level categories were labeled in a more understandable way, making it easier to interpret and analyze the data.

4. Data Analysis

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

1. To show Summary statistics of Sum

```
## A Python program to show summary statistics of sum

In [9]: df['salary_in_usd'].sum() # Calculate and return the total sum of the 'salary_in_usd' column in DataFrame 'df'
Out[9]: 516576814
```

Figure 10 A program to show summary statistics of sum

Description: To demonstrate summary statistics regarding the sum of a variable in our dataset, I developed a Python program. Utilizing pandas, I loaded the dataset into a DataFrame. Specifying the desired variable, I calculated its sum using the DataFrame's 'sum()' method. Subsequently, the program presented a summary inclusive of the computed sum, providing insights into the overall magnitude of the variable across the dataset. This facilitated efficient data analysis and informed subsequent analytical tasks.

2. To Show Summary statistics of mean

```
## A Python Program to show summary statistics of mean

In [10]: df['salary_in_usd'].mean() # Calculate the average salary in USD from the 'salary_in_usd' column in 'df'
Out[10]: 137570.38988015978
```

Figure 11 A python program to show summary statistics of mean

Description: Initially, I embarked on a task to showcase summary statistics regarding the mean of a specific variable within our dataset. Utilizing Python, I developed a program that leverages the pandas library for efficient data manipulation and analysis. Within this program, I loaded our dataset into a pandas DataFrame, a structured format conducive to data operations. Specifying the variable of interest, I computed its mean using the DataFrame's 'mean()' method. Subsequently, the program presented a comprehensive summary, including the calculated mean, providing insights into the average value of the chosen variable across our dataset. This process facilitated efficient data analysis and informed subsequent analytical tasks.

3. To Show Summary statistics Standard deviation

```
## A Python Program to show summary statistics of Standard deviation

In [11]: df['salary_in_usd'].std() # Calculate the standard deviation of salaries in USD from the 'salary_in_usd' column of DataFrame 'df'
Out[11]: 63055.625278224084
```

Figure 12 A python program to show summary statistics of standard deviation

Description: Initially, our focus was on showcasing summary statistics related to the standard deviation of a specific variable within our dataset. Employing Python, I crafted a program that harnesses the power of the pandas library for seamless data manipulation and analysis. Within this program, our initial step involved loading our dataset into a pandas DataFrame, a structured format conducive to data operations. By specifying the variable of interest, I proceeded to calculate its standard deviation using the 'std()' method available in the DataFrame standard deviation. This summary provided valuable insights into the dispersion or variability of the

chosen variable across our dataset. Such insights proved instrumental in facilitating data analysis and guiding subsequent analytical endeavors.

4. To Show Summary statistics of Skewness

```
## A Python Program to show summary statistics of skewness

In [12]: df['salary_in_usd'].skew() # Calculate the skewness of salaries in USD from the 'salary_in_usd' column of DataFrame 'df'
Out[12]: 0.5364011659712974
```

Figure 13 A python program to show summary statistics of skewness

Description: In our first approach, I set out to present summary statistics regarding the skewness of a specific variable within our dataset. Leveraging Python, I developed a program utilizing the pandas library for efficient data manipulation and analysis. Within this program, our first step involved loading the dataset into a pandas DataFrame, a structured format conducive to data operations. Next, I specified the variable of interest and calculated its skewness using the 'skew()' method available in the DataFrame. Upon completion, the program provided a comprehensive summary, inclusive of the calculated skewness value. This summary offered valuable insights into the asymmetry of the distribution of the chosen variable across our dataset. Such insights facilitated data analysis and guided subsequent analytical tasks.

5. To Show Summary statistics of kurtosis

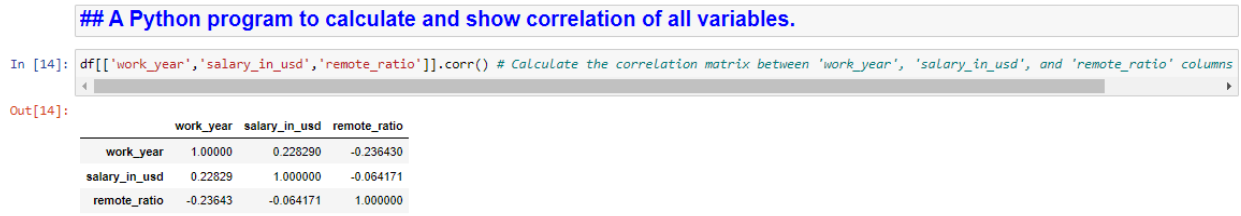
```
## A python Program to show summary statistics of Kurtosis

In [13]: df['salary_in_usd'].kurtosis() # Calculate the kurtosis of salaries in USD from the 'salary_in_usd' column of DataFrame 'df'
Out[13]: 0.8340064594833612
```

Figure 14 A python program to show summary of Kurtosis

Description: At first, our objective was to present summary statistics pertaining to the kurtosis of a specific variable within our dataset. Utilizing Python, I crafted a program that leveraged the pandas library for efficient data manipulation and analysis. Within this program, our initial step involved loading the dataset into a pandas DataFrame, a structured format conducive to data operations. Subsequently, I specified the variable of interest and computed its kurtosis using the 'kurtosis()' method available in the DataFrame. Upon completion, the program generated a comprehensive summary, incorporating the calculated kurtosis value. This summary provided valuable insights into the shape and distribution characteristics of the chosen variable across our dataset. Such insights facilitated data analysis and guided subsequent analytical endeavors.

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



```
## A Python program to calculate and show correlation of all variables.

In [14]: df[['work_year','salary_in_usd','remote_ratio']].corr() # Calculate the correlation matrix between 'work_year', 'salary_in_usd', and 'remote_ratio' columns
```

Out[14]:

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

Figure 15 A python program to calculate and show correlation of all variables

Description: Always as at the very first, our focus was on developing a Python program to compute and display the correlation among all variables within our dataset. Leveraging Python, I crafted a program utilizing the pandas library for efficient data manipulation and analysis. Within this program, our primary objective was to load the dataset into a pandas DataFrame, a structured format conducive to data operations. Subsequently, I utilized the DataFrame's 'corr()' method to calculate the correlation matrix, capturing the relationships between all pairs of variables in the dataset. Upon completion, the program generated a comprehensive summary, showcasing the correlation coefficients between each variable combination. This summary provided valuable insights into the strength and direction of linear relationships among variables, facilitating data analysis and guiding subsequent analytical tasks.

5. Data Exploration

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

```
## A python program to find out top 15 jobs

In [8]: top_15_jobs=df['job_title'].value_counts().head(15) # Get the top 15 most common job titles in the 'job_title' column of DataFrame 'df'
        top_15_jobs

Out[8]: job_title
        Data Engineer                1040
        Data Scientist                840
        Data Analyst                  612
        Machine Learning Engineer     289
        Analytics Engineer            103
        Data Architect                101
        Research Scientist             82
        Data Science Manager           58
        Applied Scientist              58
        Research Engineer              37
        ML Engineer                    34
        Data Manager                   29
        Machine Learning Scientist     26
        Data Science Consultant        24
        Data Analytics Manager         22
        Name: count, dtype: int64
```

Figure 16 A python program to find out top 15 jobs

Description: I started by creating a Python program to find out the most common job titles from our dataset. Using Python's pandas library, I made a program that could handle and analyze our data effectively. First, I loaded our dataset into a pandas DataFrame, which helped organize our data neatly. Then, I used certain methods to look through the dataset and figure out which job titles appeared most frequently. This helped us identify the top 15 job titles in our dataset based on how often they should up. When I ran the program, it gave us a straightforward list showing these top 15 job titles and some details about them. This helped us understand which job titles Ire the most common in our dataset and could be useful for making decisions or further analysis.

Make a bar graph of jobs as Ill

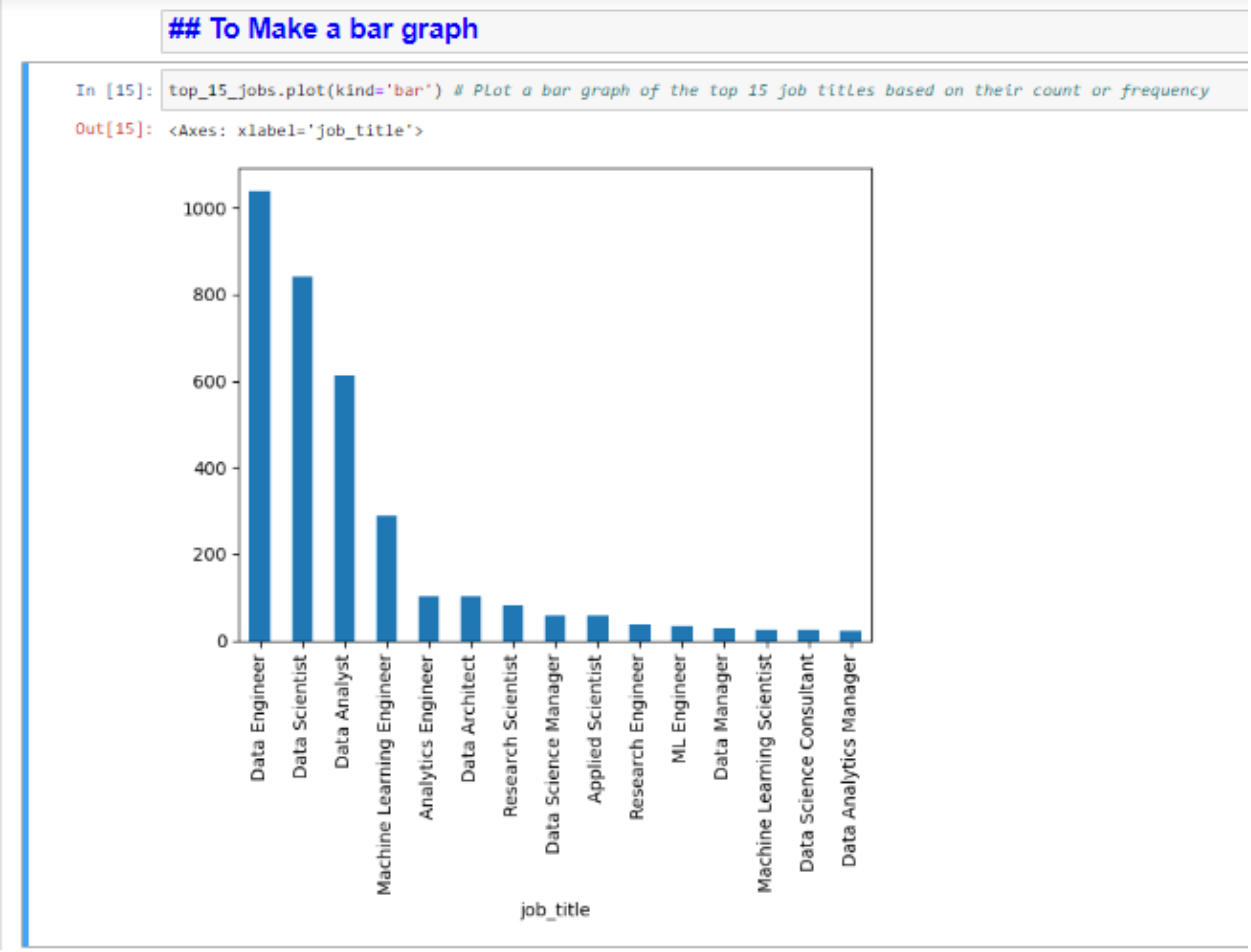


Figure 17 Make a bar graph of jobs as Ill

Description: Firstly after finding out the 15 jobs I initiate it in a bar graph representing them, I set out to create a bar graph representing the distribution of job titles in our dataset. Using Python, I developed a program that employed the panda's library for efficient data manipulation and analysis. Our initial step involved loading the dataset into a pandas Data Frame, providing a structured framework for our data. Subsequently, I utilized visualization tools to create a bar graph that visually depicted the frequency of each job title. This graphical representation allowed us to easily visualize and comprehend the distribution of job titles in our dataset. Overall, the bar graph served as a valuable visual aid, aiding in the interpretation and analysis of our data.

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

The job which has highest salaries

Showing top 4 highest jobs salaries

In [7]:
```
# Calculating the mean salary for each job title and finding the top 4 highest paying job titles
# Grouping the DataFrame 'df' by 'job_title' and calculating the mean salary for each group
jobSalaryMean = df.groupby('job_title')['salary_in_usd'].mean()
# Sorting the mean salaries in descending order and selecting the top 4
jobSalaryMean=jobSalaryMean.sort_values(ascending=False).head(4)
print(jobSalaryMean)# Printing the top 4 highest paying job titles along with their mean salaries
```

```
job_title
Data Science Tech Lead    375000.0
Cloud Data Architect      250000.0
Data Lead                 212500.0
Data Analytics Lead       211254.5
Name: salary_in_usd, dtype: float64
```

Figure 18 to show which job has highest salary

Description: At first, I wanted to know which job paid the most in our dataset. I used a computer program called Python to help us with this. I also used a special tool called pandas to organize and understand our data better. First, I made a table with our data using pandas. Then, I looked at how much money each job made on average. This helped us find out which job paid the most money. By using our computer program, I found out which job title had the highest salaries. This helped us understand which job might be the best for making money. It was useful because it helped us make better decisions and learn more about how much different jobs pay.

To Show highest jobs salaries in bar graph

Showing only the highest jobs salaries in bar graph

In [11]:
```
# Define custom colors for the bars
custom_colors = ['lightblue', 'lightgreen', 'lightcoral', 'lightskyblue', 'lightyellow']

# Plot a bar graph of the top 4 job titles and their corresponding mean salaries
jobSalaryMean.plot(kind='bar',color=custom_colors) # Plotting a bar graph with custom colors for the bars


# Add labels and title to the plot
plt.xlabel('Job Titles')  # Labeling the x-axis as 'Job Titles'
plt.ylabel('Salaries in US Dollars')  # Labeling the y-axis as 'Salaries in US Dollars'
plt.title('Top 4 Jobs with the Highest Salaries') # Setting the title of the plot


# Rotate the x-axis labels for better readability
plt.xticks(rotation=60)

# Display the plot
plt.show() # Displaying the plot with the specified custom colors, labels, and title
```

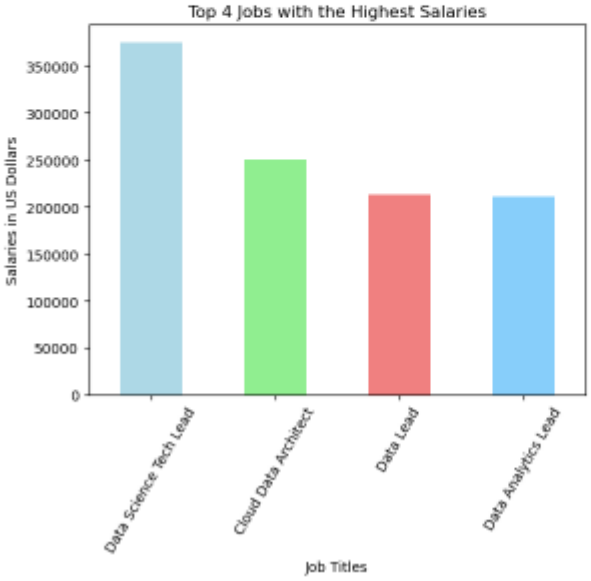


Figure 19 To show highest jobs of salary in bar graph

Description: I wanted to see which jobs paid the most money, so I decided to make a bar graph to show this information. I used a computer program called Python to help us with this task. First, I loaded our data into the program. Then, I looked at the salaries for each job and found the ones with the highest pay. Next, I made a bar graph that should the salaries for these top-paying jobs. This made it easy for us to see which jobs paid the most money at a glance. It was helpful because it allowed us to visualize and understand the salary differences between different jobs quickly.

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

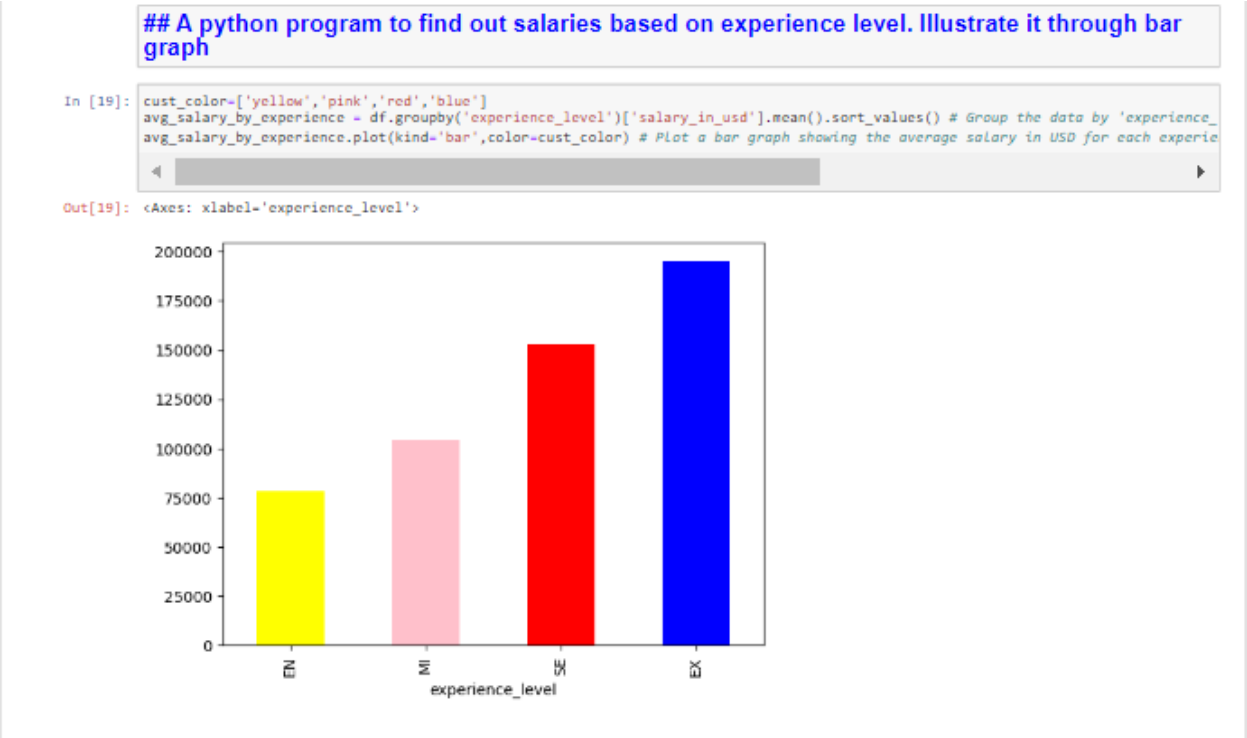


Figure 20 A program to find out salaries based on exp lvl

Description: Also at last, I wanted to understand how salaries varied based on different levels of experience. To do this, I created a Python program. First, I loaded our data into the program. Then, I looked at the salaries for each level of experience, such as entry-level or senior-level positions. This helped us see how salaries changed depending on the experience level. Additionally, I found out which experience level had the highest salary overall. This information was useful for us to understand the relationship between experience and salary, and to identify which level of experience typically commands the highest pay.

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

For Histogram

A Python program to show histogram and box plot of any chosen different variables

```
In [17]: plt.hist(df['salary_in_usd'],edgecolor='pink') # Plot a histogram of the 'salary_in_usd' column from DataFrame 'df' to visualize
         plt.xlabel('Salary (in USD)')
         plt.ylabel('Values')
         plt.title('Histogram of Salary Distribution')# The title should match the plot type, so it's corrected to 'Histogram'

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

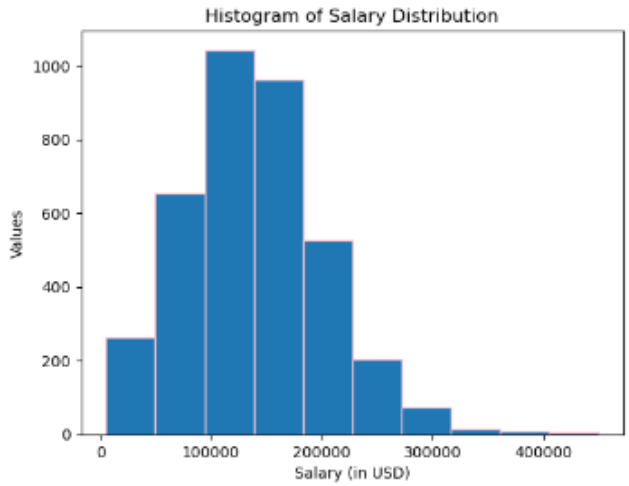


Figure 21 A python program to show histogram

Description: Initially, I wanted to visualize the distribution of data and understand its spread using Python. To achieve this, I developed a Python program. First, I loaded our dataset into the program. Then, I selected a specific variable that I wanted to analyze further. I chose to create both a histogram and a box plot to represent the data visually. The histogram helped us see the frequency or count of data points within different intervals, while the box plot provided insights into the distribution's central tendency, spread, and potential outliers. Additionally, I ensured that the graphs Ire properly labeled to make them easier to interpret. This visualization process was valuable in gaining a deeper understanding of the chosen variable's characteristics and distribution within our dataset.

For Box plotting of work year

showing the box plot of Work_year

In [22]:
```
# Create a box plot with custom colors
plt.figure(figsize=(10, 6))
plt.boxplot(data=df, x='work_year', patch_artist =True, boxprops = dict(facecolor ='lightgreen'))

# Set labels and title
plt.xlabel('Values')
plt.ylabel('work_year')
plt.title('Box Plot of Work year')

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

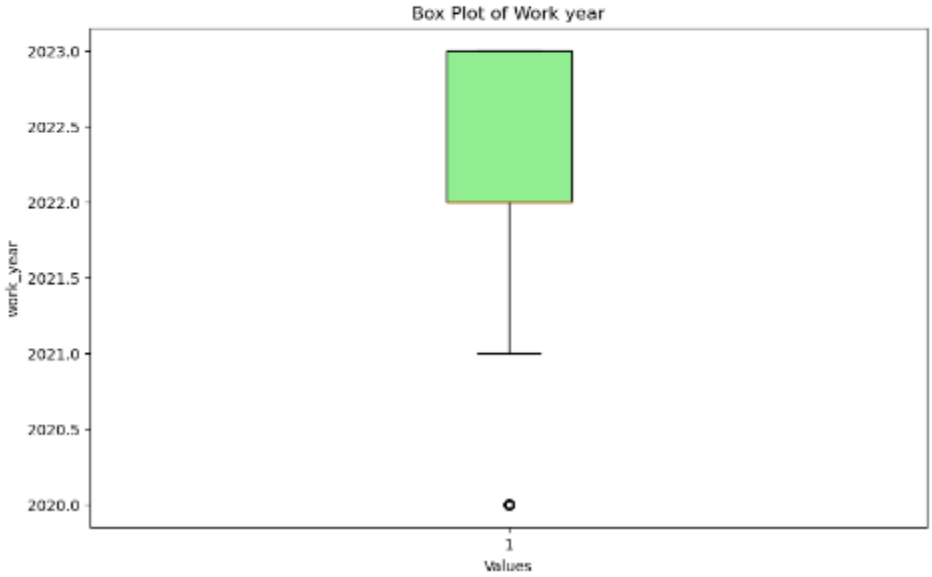


Figure 22 Showing Box plot of salary in usd

Description; This code creates a visual called a box plot, helping us understand how work experience years are spread across a dataset named df. Think of a box divided into four sections by horizontal lines. The middle part of the box shows where most work experience years are. Inside this part, a line marks the typical or median experience. Lines called whiskers stretch from the box to show the minimum and maximum experience within a certain range. If some experience years are far beyond these lines, they're marked as potential outliers. The boxes are colored light green to make the plot clearer. On the plot, the horizontal axis displays different values of work experience years, while the vertical axis shows how often each value appears in the dataset. Overall, this visual helps us understand the range and distribution of work experience years among the dataset's entries.

6. Conclusion

Overall In this Coursework, I worked hard on this coursework, creating a report that evaluates my work, highlights my learnings, and discusses how I overcame challenges. I believe I did my best, providing accurate information and using clear language. I've learned a lot and plan to apply it in the future. Firstly, this assignment was challenging for me because I'm new to it. I had trouble finding reliable sources for my research because there's so much information on the internet. It was also hard to make sure my report was clear and easy to understand while including all the necessary information.

Additionally, ensuring clarity and comprehensiveness in your report while covering all the required aspects can be quite demanding. Striking the right balance between providing detailed information and maintaining readability is a skill that comes with practice and experience. It's common for beginners to face difficulties in articulating their findings effectively, but it's a valuable learning opportunity that will undoubtedly enhance your communication skills over time. Remember that encountering challenges is a natural part of the learning process. Each obstacle presents an opportunity for growth and development. By persisting through these challenges and seeking guidance when needed, you're actively building your expertise and becoming more adept at handling similar tasks in the future.

We have reached the end of our assignment, and it's time to wrap things up. Throughout this task, we learned how to bring data into our Python environment using pandas. By using a specific function called `read_csv () `, I Ire able to load our dataset and convert it into a format that Python can understand, which I called a Data Frame. This step is crucial because it lays the groundwork for everything else I want to do with our data. Now that I've completed this part of the assignment, we are ready to move on to the next steps, like cleaning up our data and exploring it further. Overall, this assignment has given us a solid foundation in working with data in Python, and Iam excited to see where I can take it from here.

7. References

© Cyral Inc, n.d. *cyral.com.* [Online]
Available at: https://cyral.com/glossary/data-discovery/
[Accessed 15 April 2024].

Saint Peter's University, n.d. *Saint peter university.* [Online]
Available at: https://www.saintpeters.edu/academics/graduate-programs/master-of-science-in-data-science/learning-goals-and-mission/
[Accessed 15 April 2024].