



CC5067NI-Smart Data Discovery

60% Individual Coursework

2023-24 Autumn

Student Name: Divya Shrestha

London Met ID: 22085527

College ID: NP01CPS230022

Assignment Due Date: Monday, May 13, 2024

Assignment Submission Date: Sunday, May 12, 2024

Word Count: 4789

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

Table of Contents

1	Introduction	1
2	Data Understanding.....	2
3	Data Preparation.....	5
3.1	Write a python Program to load data into pandas DataFrame	6
3.2	Write a python program to remove unnecessary columns.	8
3.3	Write a python program to remove the NaN missing values from updated dataframe.	9
3.4	Write a python program to check duplicate value in the dataframe.....	11
3.5	Write a python program to see the unique values from all the columns in the dataframe.	13
3.6	Rename the experience level columns as below. SE – Senior Level/Expert MI – Medium Level/Intermediate EN – Entry Level EX – Executive Level	14
4	Data Analysis	16
4.1	Write a Python program to show summary statistics of sum, mean, standard deviation, skewness, and kurtosis of any chosen variable.....	17
4.2	Write a Python program to calculate and show correlation of all variables.....	20
5	Data Exploration	21
5.1	Write a python program to find out top 15 jobs. Make a bar graph of sales as well. 22	
5.2	Which job has the highest salaries?	24
5.3	Write a python program to find out salaries based on experience level.	26
5.4	Write a Python program to show histogram and box plot of any chosen different variables. Use proper labels in the graph.	28
6	Conclusion	31

Table of Figure

Figure 1:importing libraries.....	5
Figure 2:: Reading the csv file and converting into Data Frame.....	6
Figure 3:Removing unnecessary columns.	8
Figure 4:Dropping the rows with Nan values in it.	10
Figure 5:Removing duplicate rows from the data frame.....	11
Figure 6:Printing the unique values in a column.....	13
Figure 7:Mapping and replacing values of the column.	14
Figure 8:Total sum of a column "Salary_in_usd".	17
Figure 9:Mean of the column "Salary_in_usd"	17
Figure 10: Standard deviation of the column "salary_in_usd"	18
Figure 11:Skewness of the column "salary_in_usd"	18
Figure 12:Kurtosis of the column "salary_in_usd"	19
Figure 13:Top 15 jobs.....	22
Figure 14:Highest salaries among the top jobs.	24
Figure 15:Avarage Salary by Experience level.....	26
Figure 16:Histogram and box plot of salary_in_usd variable.....	29
Figure 17: Histogram and box plot of work_year_variable	30

1 Introduction

Currently, in the digital economy, modern companies collect large volumes of data that contain an immense sum of values, but only when processed, structured, and refined this value can be obtained. For companies trying to be successful, using data in its raw and unprocessed form or in the wrong way for making decisions is not only insufficient; it's a strategic mistake. Data acts as a very crucial service that helps in providing organizations with both the required information and the power to make the best decisions.

The main activities of data analysis include the processes of cleaning, transforming of a data set, and generating their insights. This way of analysis is a must for any decision that a person must make because of the uncertainties that are always present. Through the implementation of the complex instruments, data analysis becomes the source of sophisticated information, which is commonly reflected as graphs, charts, tables, or via visualization graphics.

The importance of data analysis exceeds just the simple decision support function; it directs companies towards stepped targeting of consumers and to a better understanding of the consumer behavior. Furthermore, it is a major stakeholder in the reduction of operational costs and the improvement of problem-solving strategies. Data analytics would make organizations see both inefficiency opportunities within the business and a possibility to finetune the marketing efforts. It would also strategically allocate the resources the business environment needs. Hence a more flexible and reactive type of environment is encouraged.

2 Data Understanding

Data understanding is when a building block of whatever data are analyzed which highlight all the features like content, shape, and attributes of the data set. This operation mostly makes use of data study which can help an expert to identify the features as well as the probable problems that may stem from the data. It is a data cleaning that is regarded as especially pivotal as it serves, first, to help avoid many problems at the stage of the data preparation. The second step is data accessing, data exploration, which is an invaluable approach and dependency in the investigating process of finding out any misfit, error, or anomaly. Understanding the set of data that addresses the reasons for salary fluctuations is about analyzing the projecting attributes solely. Such an exhaustive study headed by data mining and analysis steps are about to be executed by us. The conclusion of this stage determines the basis for further analysis through the uncovering of data patterns and the key levers we will focus on during further analysis. With integrating the obtained findings, we will, thus, aptly be in the position of intuitive and applying the accumulated data to investigate a point of inference and therefore enhance our decision-making process. Recognizing the fact that the first part is the foundation on which the efficiency and success of the research depends, we are resolute to seek for data in the next item on the agenda. As we give the pre-requisites and the second look, we identify and understand the pollution and mode of them transpiring which allows us to find answer and make progress towards development.

This dataset leans into the hidden angles of things that create the salary situation for the data science field that is a fast developing and challenging environment. A wide range of parameters has been thoughtfully collected in this series which includes parameters like seniority level job title, annual wages, number of years worked, besides many other factors. The dataset is a collection of gold mine of valuable knowledge. With its unprecedented look on this field, the dataset becomes the map to the realm of data science salaries. Through a careful examination of the interplay between these factors, it seeks to peek behind the scenes and reveal the specific patterns and trends, that form the basis of salaries in this rapidly growing sector. From

this perspective, one can embark on a journey of discovery which will unfold like unravelling the fantastic puzzle of the factors reflected in the levels of salary. There exist diverse shades of job titles as well as job experience apparently which perplex all. This is more complicated as every data point has become a lighthouse revealing the hidden path of understanding of determinants in data science. Through careful review of the vast amount of knowledge poured in the data, the researchers and other analysts can manage to discover significant fact and analysis which if implemented can affect some decisions that are critical in areas like talent recruitment and retention. For instance, these are the key showcase how data-driven approaches are gaining more importance regarding understanding and tackling the issues of modern workforce complexity. This Beyond the data scrap shot not only gives a current insight but also acts as a catalyst for investigations in future, therefore all the stakeholders must adapt and compete in the emerging data driven sphere. The following are the Dataset attributes in the csv file of our project:

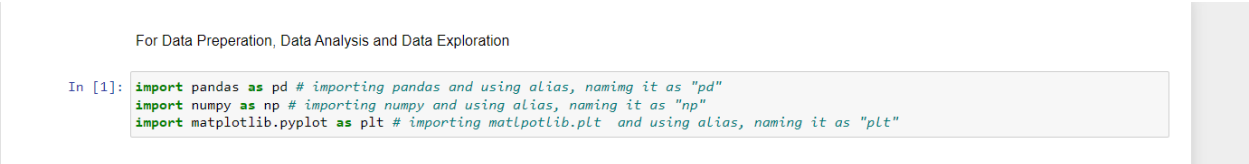
Tables of Dataset Attributes

S.No	Column Name	Description	Data Type
1	work_year	Work year of an individual	Integer
2	Experience_level	Experience level of the individual	object
3	employment_type	Employment type of an individual	object
4	job_title	Job title of an individual	object
5	salary	Salary of an individual	Integer
6	salary_currency	Currency type of an individual	object
7	salary_in_usd	Salary of an employee in USD	Integer
8	employee_residence	Residence of the employee	object
9	Remote_ratio	Remote working ration of an employee	Integer
10	company_location	Location of the company of an individual	object
11	company_size	Size of the company	object

3 Data Preparation

In the world of data science, the provision of libraries significantly affects the fast performance and ease of the data manipulation. Libraries of this type largely employ libraries to process databases coupled with the execution of manipulations on data, and formations of visuals that are efficient in transferring the information.

To empower the current assignment, we shortlist the necessary libraries i.e.: Pandas, numPy, and Matplotlib. Panda is as a data analysis cornerstone through its strong capabilities of creating and transforming the data frames. Matplotlib, on contrast, forms the backbone of creating various graphs, charts, and plots for data visualization. In addition, NumPy will be employed to efficiently handle array-based data structures, providing essential support for various mathematical operations and array manipulation tasks within the data science workflow.



```
For Data Preperation, Data Analysis and Data Exploration

In [1]: import pandas as pd # importing pandas and using alias, naming it as "pd"
import numpy as np # importing numpy and using alias, naming it as "np"
import matplotlib.pyplot as plt # importing matplotlib.pyplot and using alias, naming it as "plt"
```

Figure 1:importing libraries.

3.1 Write a python Program to load data into pandas DataFrame

When the code is executed, it reads pandas Data Frame where data will be extracted based on salaries Damascene from the CSV file named "DataScienceSalaries.csv". It further frees the files in CSV format into Data Frame which is tabular data structure such as spreadsheets for hassle free analysis. The head() function is used to depict select few rows of the Data Frame, which helps one quickly to understand the data structure and content. These methods are often called/used in the early steps of data analysis, to load an easy dataset and perform an initial inspection before other operations like data cleaning, transformation, or visualization can be executed.

```
In [2]: data = pd.read_csv("DataScienceSalaries_8b290669-a5e9-45bf-be72-d27add2eacae_93472_.csv")
# Reading the csv file from Local storage

In [3]: dataframe = pd.DataFrame(data)
# Converting the csv files into DataFrame for clear analyzing

In [4]: dataframe.head() # printing the dataframe
Out[4]:
```

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

Figure 2:: Reading the csv file and converting into Data Frame

The functions used in these questions are:

- `pd.read_csv()`: This feature is being between the pandas library (pd). It fills in data from a CSV file and returns a Data Frame. Here, an argument is passed which is an address of CSV file that is being read.
- `pd.DataFrame(data)`: This function will take as input a data Frame data object read by a `pd.read_csv()` function. It changes a data from class style to a tabular format and enables user to access and modify rows and columns even more comfortably.

- `dataFrame.head()`: The code presents a method that will be called on the `DataFrame` object that was created in the previous step (`dataFrame`). It gives the initial number of the rows of the `DataFrame` and so on. It, by default, displays only the initial 5 rows.

3.2 Write a python program to remove unnecessary columns.

Firstly, we build a list of exclusion named column to exclude, with the column names to be removed from new Data Frame being part of it. hence, a list named without column header is created to save the names of columns that are going to remain. The code loops through individual columns in original Data Frame (dataFrame) and when not in column_to_exclude list, it converts the column name to the specified units (units_conversion). In this case an 'except' clause is not used, instead the column name is added to the column_to_keep list. After running through the entire table and replicating process, code provides column list to keep. Then, a new DataFrame is formed by specifying only those columns indicated in column_to_keep, which implies the elimination of the unwanted columns. The usage of the index operation `dataFrame[column_to_keep]` will achieve it implies which selects columns by their names.

```
In [5]: # List of columns to exclude from the new DataFrame
column_to_exclude = ["salary", "salary_currency"]

# List to store columns to keep
column_to_keep = []

# Iterating through columns in the new DataFrame
for col in dataframe.columns:
    # Checking if the column is not in the exclusion list
    if col not in column_to_exclude:
        column_to_keep.append(col) # If the column is not excluded, add it to the list of columns to keep

# Printing the list of columns to keep
print(column_to_keep)

# Creating a new DataFrame by dropping the excluded columns'
dropped_columns = dataframe[column_to_keep]

['work_year', 'experience_level', 'employment_type', 'job_title', 'salary_in_usd', 'employee_residence', 'remote_ratio', 'company_location', 'company_size']
```

Figure 3: Removing unnecessary columns.

The functions used in these questions are:

- `dataFrame.columns`: This attribute returns the names of all columns in the DataFrame.
- DataFrame indexing: The indexing operation (`dataFrame[column_to_keep]`) selects columns from the DataFrame based on the column names provided in the

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

First, we check for missing values (NaN) if they are in Data Frame using `isnull()` method which gives True in case that missing value was found, and then the `any()` method which shows that at least one NaN was found. The implication is that gives a binary value called `has_null` that represents the status of missing values in terms of their existence. Secondly, the following step would have been producing a DataFrame with all the NaN values removed (`has_null` is False). The No Missing Row operation is performed through the `dropna()` function, which filters out rows with missing values in any type of integral column. The parameter (`inplace=True`) ensures that Data Frame is being edited in place and doesn't have any "side effects". In a nutshell, this code snippet is important as it can contribute to the quality of data and its integrity by filling in values of missing data and applies measures to ensure that the data is fit for analysis later.

```

In [6]: # checking if there is any nan values in the dataframe
has_null = dataframe.isnull().values.any()
print(has_null)

# If there were any NaN values in the DataFrame, 'has_null' would be True,
# indicating the presence of missing data. However, since there are no NaN values,
# 'has_null' is False, indicating that the DataFrame contains no missing data.

dataframe.dropna(inplace=True)
# To remove rows containing NaN values from the DataFrame, we can use the dropna() function.
# This function returns a new DataFrame with NaN values dropped.
dataframe

False

Out[6]:

```

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

3755 rows × 9 columns

Figure 4: Dropping the rows with Nan values in it.

The functions used in this question are:

- `isnull()`: This method completes searching for missing (NaN) values within the DataFrame and returns a [True, False] Data Frame to express the presence of these kind of values.
- `any()`: The Boolean Testing method checks if any value entries in the Boolean Data Frame created with the `isnull()` function are True, which could be an indication of missing values.
- `dropna()`: This function is there for that purpose: it removes columns in which are found missing values from the Data Frame. With "inplace=True", modifications are performed on the original Data Frame in-place instead of the returned Data Frame is a new Data Frame.

3.4 Write a python program to check duplicate value in the DataFrame

First, we continue by looking for duplicate records using the duplicate function `duplicated()`, which outputs accordingly (a Boolean specifying if there are duplicate rows). After that it sets the cover and indices list to processes the duplicate row. code is running through all rows holding in dataframe and every row is transformed into tuple for the comparison purpose. When a line It is included in the list of duplicate line indices. Following the recognition of all duplicated rows, those are deleted from the DataFrame by the `drop_duplicates()` method with `inplace=True` parameter which actually allows to change the initial Data Frame. Lastly, it rewrites the sentence again to check if there have been any duplicates. This code preserves data integrity within databases by detecting and removing misleading entries. It is significant for accurate data analysis and modeling.

```
In [7]: print("Duplicates:", dataframe.duplicated().any())

# Initializing a set to store seen rows and a list to store duplicate indices
seen = set()
duplicate_indices = []

# Iterating through rows in the DataFrame
for index, row in dataframe.iterrows():
    # Converting the row to a tuple for comparison
    row_tuple = tuple(row)
    # Checking if the row has been seen before
    if row_tuple in seen:
        # If the row is a duplicate, adding its index to the list of duplicate indices
        duplicate_indices.append(index)
    else:
        # If the row is not a duplicate, adding it to the set of seen rows
        seen.add(row_tuple)

# Printing the indices of duplicate rows
print(duplicate_indices)

# Dropping duplicate rows from the DataFrame
dataframe.drop_duplicates(inplace=True)

# Printing whether there are any duplicated rows after removing duplicates
print("Duplicates:", dataframe.duplicated().any())
```

2367, 2370, 2371, 2376, 2377, 2385, 2388, 2389, 2398, 2399, 2402, 2403, 2410, 2411, 2412, 2414, 2415, 2419, 2420, 2423, 2434,
2435, 2436, 2441, 2442, 2443, 2444, 2445, 2448, 2449, 2452, 2453, 2456, 2457, 2459, 2463, 2467, 2468, 2469, 2482, 2483, 2484,
2485, 2493, 2494, 2495, 2497, 2498, 2499, 2502, 2503, 2505, 2506, 2510, 2511, 2516, 2517, 2518, 2524, 2525, 2538, 2540, 2541,
2542, 2543, 2544, 2545, 2549, 2550, 2551, 2552, 2557, 2558, 2560, 2574, 2575, 2576, 2577, 2580, 2581, 2582, 2583, 2584, 2585,
2586, 2587, 2594, 2595, 2600, 2602, 2603, 2604, 2605, 2606, 2607, 2612, 2613, 2614, 2615, 2616, 2617, 2621, 2622, 2623, 2628,
2629, 2630, 2631, 2633, 2634, 2635, 2642, 2643, 2644, 2645, 2658, 2659, 2668, 2669, 2670, 2671, 2676, 2677, 2681, 2682, 2682,
2693, 2713, 2719, 2720, 2729, 2730, 2731, 2732, 2733, 2735, 2736, 2737, 2738, 2741, 2742, 2748, 2749, 2752, 2753, 2754, 2756,
2757, 2758, 2759, 2760, 2761, 2764, 2765, 2766, 2767, 2768, 2769, 2770, 2771, 2782, 2783, 2794, 2795, 2796, 2797, 2798, 2801,
2802, 2803, 2804, 2808, 2810, 2811, 2814, 2815, 2820, 2821, 2838, 2839, 2843, 2844, 2845, 2846, 2847, 2848, 2853, 2856, 2857,
2858, 2859, 2860, 2861, 2863, 2864, 2865, 2866, 2877, 2878, 2881, 2882, 2883, 2884, 2885, 2886, 2894, 2895, 2896, 2897, 2900,
2901, 2904, 2905, 2924, 2925, 2926, 2927, 2928, 2929, 2936, 2941, 2942, 2944, 2945, 2949, 2971, 2972, 2978, 2979, 2988, 2989,
2996, 2997, 3002, 3003, 3004, 3024, 3025, 3027, 3029, 3030, 3043, 3044, 3045, 3047, 3048, 3053, 3069, 3070, 3080, 3081, 3082,
3083, 3084, 3085, 3089, 3100, 3101, 3105, 3111, 3112, 3115, 3116, 3120, 3121, 3122, 3129, 3130, 3134, 3135, 3136, 3137, 3140,
3141, 3144, 3145, 3164, 3165, 3172, 3177, 3179, 3180, 3182, 3190, 3191, 3200, 3201, 3202, 3203, 3204, 3205, 3206, 3207, 3210,
3211, 3217, 3218, 3222, 3232, 3235, 3242, 3247, 3257, 3263, 3271, 3272, 3277, 3281, 3290, 3291, 3294, 3298, 3299, 3300, 3301,
3302, 3303, 3307, 3314, 3315, 3316, 3317, 3319, 3321, 3323, 3334, 3335, 3336, 3337, 3344, 3345, 3348, 3349, 3352, 3358, 3359,
3361, 3364, 3365, 3367, 3370, 3371, 3373, 3374, 3375, 3376, 3377, 3378, 3379, 3380, 3382, 3383, 3384, 3385, 3386, 3393, 3394,
3395, 3396, 3397, 3401, 3407, 3412, 3413, 3416, 3417, 3439, 3440, 3441, 3586, 3709]

Duplicates: False

Figure 5: Removing duplicate rows from the data frame.

The functions used in this question are:

- `duplicated()`: Checks for duplicate rows within the Data Frame and returns a boolean Series indicating which rows are duplicates.
- `any()`: Determines if any of the values in the boolean Series returned by `duplicated()` are True, indicating the presence of duplicates.
- `iterrows()`: Iterates through each row of the Data Frame, returning the index and row data as a tuple.
- `drop_duplicates()`: Removes duplicate rows from the Data Frame. When `inplace=True` is specified, it modifies the original Data Frame directly.

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

First, we use a for loop to iterate over the columns of the Data Frame using `dataFrame.columns`. Within the loop, it accesses each column individually using `dataFrame[column]` and applies the `unique()` method to retrieve an array of unique values in that column. The `unique()` function returns an array containing the unique elements of the specified column. Finally, it prints out the unique values for each column in the Data Frame. This code is helpful for quickly understanding the distinct values present in each column, aiding in data exploration and analysis.

```
In [8]: # Printing unique values for each column in the DataFrame
        for column in dataframe.columns:
            print(dataframe[column].unique())

'Data Science Manager' 'Data Manager' 'Machine Learning Researcher'
'Big Data Engineer' 'Data Specialist' 'Lead Data Analyst'
'BI Data Engineer' 'Director of Data Science'
'Machine Learning Scientist' 'MLOps Engineer' 'AI Scientist'
'Autonomous Vehicle Technician' 'Applied Machine Learning Scientist'
'Lead Data Scientist' 'Cloud Database Engineer' 'Financial Data Analyst'
'Data Infrastructure Engineer' 'Software Data Engineer' 'AI Programmer'
'Data Operations Engineer' 'BI Developer' 'Data Science Lead'
'Deep Learning Researcher' 'BI Analyst' 'Data Science Consultant'
'Data Analytics Specialist' 'Machine Learning Infrastructure Engineer'
'BI Data Analyst' 'Head of Data Science' 'Insight Analyst'
'Deep Learning Engineer' 'Machine Learning Software Engineer'
'Big Data Architect' 'Product Data Analyst'
'Computer Vision Software Engineer' 'Azure Data Engineer'
'Marketing Data Engineer' 'Data Analytics Lead' 'Data Lead'
'Data Science Engineer' 'Machine Learning Research Engineer'
'NLP Engineer' 'Manager Data Management' 'Machine Learning Developer'
'3D Computer Vision Researcher' 'Principal Machine Learning Engineer'
'Data Analytics Engineer' 'Data Analytics Consultant'
'Data Management Specialist' 'Data Science Tech Lead'
```

Figure 6:Printing the unique values in a column.

Functions used in this question are:

- `dataFrame.columns`: This attribute returns the names of all columns in the DataFrame, allowing the code to iterate over each column.
- `dataFrame[column]`: This syntax accesses the data within a specific column of the Data Frame, allowing the code to operate on each column individually during the iteration.
- `unique()`: This method, applied to a Series within the Data Frame, returns an array containing the unique elements of that Series, effectively providing the unique values present in a column.

3.6 Rename the experience level columns as below. SE – Senior Level/Expert MI – Medium Level/Intermediate EN – Entry Level EX – Executive Level

The dictionary is then initialized named `experience_level_mapping` with a key as the abbreviated label and a value as the corresponding descriptive label. Subsequently, it employs the `replace()` method in the DataFrame's "experience_level" column. The dictionary is passed in as the parameter for the `experience_level_mapping` dictionary. While this feature replaces, in the column, each abbreviated label with its more elaborate label. The `inplace=True` parameter specifies the modifications done directly to the original DataFrame. Lastly it renders the top few rows of the DataFrame having used the `head()` method to exhibit the modified "experience_level" column. This code is geared toward less complicated DataFrame linkage and makes the experience level data more convenient to understand.

```
In [9]: # Mapping experience level abbreviations to descriptive labels and replacing them in the DataFrame
experience_level_mapping = {
    "SE": "Senior Level/Expert",
    "MI": "Medium Level/Intermediate",
    "EN": "Entry Level",
    "EX": "Executive Level"
}

# Replacing the values in the "experience_level" column with their corresponding labels
# inplace=True modifies the DataFrame in place
dataFrame["experience_level"].replace(experience_level_mapping, inplace=True)

dataFrame.head()
```

Out[9]:

	work_year	experience_level	employment_type	job_title	salary	salary_currency	salary_in_usd	employee_residence	remote_work
0	2023	Senior Level/Expert	FT	Principal Data Scientist	80000	EUR	85847	ES	
1	2023	Medium Level/Intermediate	CT	ML Engineer	30000	USD	30000	US	
2	2023	Medium Level/Intermediate	CT	ML Engineer	25500	USD	25500	US	
3	2023	Senior Level/Expert	FT	Data Scientist	175000	USD	175000	CA	
4	2023	Senior Level/Expert	FT	Data Scientist	120000	USD	120000	CA	

Figure 7: Mapping and replacing values of the column.

The functions used in this question are:

- **replace()**: This method replaces specified values in a Series or Data Frame with other values. In this case, it replaces abbreviated experience level labels with their corresponding descriptive labels.
- **dataFrame.head()**: This method displays the first few rows of the Data Frame, providing a quick overview of the data after the replacement operation.

4 Data Analysis

Data analysis is the practice of working with data to glean useful information, which can then be used to make informed decisions. This procedure requires executing duties, like cleaning and processing of the data, investigating its components and connections, using statistical or machine learning methods to recognize valuable data, and finally presenting the results in a manner that is easy and unambiguous to comprehend. Data analysis provides organizations and individuals with the possibility to derive invaluable insights about their functioning, customers and markets that can then shape their decisions, help identify areas for improvement, as well as stimulate innovation and advancement.

Analysis of data (data analysis) plays a key role in different areas. To start with, the data we deal with within the broadest areas containing things like business transactions, online interactions e.g. social media, sensors, and other are so many that it could be hard to make sense of them. A particular statistical tool that enables us to investigate what is happening within the data in terms of trends, patterns, and interrelationships giving us a starting point for working out the most likely cause of this trend, pattern, or interrelationship so that we can make informed decisions. Not only that, but data analytics also helps us realize the areas for improvement, maximizes our processes, and brings the abnormalities or potential harms to our attention. Furthermore, data analysis, which is integral for business success because of the data-driven world, is currently gaining traction, while businesses may easily acquire important business insights. This focuses on the needs of consumers, identifying trends of the future and directing the strategic initiatives strongly towards success. First, the analysis of data helps individuals and organizations to significantly reduce the number of uncertainties because they largely act on evidence. Second, it supports the decision-making process because it determines more accurate strategy initial points and the direction in which to move. Moreover, it allows individuals and organizations to develop accurate control plans and to decrease amounts of waste.

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

We start with getting the "salary_in_usd" column from the Data Frame and storing it into a variable called salary_column. Firstly, it defines a variable called total_sum and assigns it to keep the sum of the salaries. The code loops through the salary value in the salary_column and adds each value to the total_sum variable. Lastly, it prints the calculated total sum of all salary amounts.

```
In [10]: salary_column = dataframe["salary_in_usd"] # Extracting the "salary_in_usd" column
        total_sum = 0

        # Iterating through each salary value in the column and adding it to the total sum
        for i in salary_column:
            total_sum = total_sum + i

        total_sum # value from the calculation

Out[10]: 344729580
```

Figure 8: Total sum of a column "Salary_in_usd".

After that, we mean is calculated by dividing the total sum of salaries and that sum is stored in the variable total_sum by the number of salary values in the column, currently counts the number using the function len(salary_column). Then, it fetches the mean (the average wage) and assigns it to a variable called salary_mean. The next line, that is the calculated average salary value is printed.

```
In [11]: # Dividing the total sum by number of rows in the salary_in_usd column to get mean value
        salary_mean = total_sum / len(salary_column)

        salary_mean # value from the calculation

Out[11]: 133409.28018575851
```

Figure 9: Mean of the column "Salary_in_usd"

After, it starts off a variable called `squared_deviations` and it will be used to calculate the sum of squared deviations. It carries out the looping through all the individual values in the `salary_column`, calculating the discrepancy with mean salary (`salary_mean`) and squaring each deviation and accumulates squared deviations in `squared_deviations`. After a for-loop has been performed for the salary values, the code then finds out the sum of squared deviations divided by the number of salary values, then standard deviation by extracting the square root of variance.

```
In [12]: squared_deviations = 0 # Initializing variable for sum of squared deviations

# Calculating the sum of squared deviations for each salary value from the mean.
for x in salary_column:
    deviation = x - salary_mean
    squared_deviation = deviation ** 2
    squared_deviations += squared_deviation

data_variance = squared_deviations / len(salary_column) # Calculating the data variance

salary_standard_deviation = data_variance ** 0.5 # Calculating the standard deviation by taking the sq
salary_standard_deviation # printing the value

Out[12]: 67123.84519805372
```

Figure 10: Standard deviation of the column "salary_in_usd"

Now, to the variable `cubed_deviations`, which in turns keeps track of the sum of the squared deviations. After that, it goes through each salary value individuality in the `salary_column`, computing the difference of each salary from the average value (`salary_mean`), squared each difference, and collects all the squared differences in `cubed_deviations`. When the code is finished iterating through all salary values, the calculation of the skewness will be done by taking the sum of the cubed deviations divided by the cube of `salary_standard_deviation` and the number of salary values.

```
In [13]: cubed_deviations = 0 #Initializing the sum of cubed deviations

# Calculating the sum of cubed deviations
for x in salary_column:
    deviation = x - salary_mean
    cube_deviation = deviation ** 3
    cubed_deviations = cube_deviation + cubed_deviations

# Calculating the skewness of salary values
salary_skew = cubed_deviations / len(salary_column) / (salary_standard_deviation ** 3)

salary_skew

Out[13]: 0.6199567299104844
```

Figure 11: Skewness of the column "salary_in_usd"

Finally, variable named `pow_deviations` is assigned for fourth-power deviation summation. Subsequently, it goes over every salary value in column of salary and multiplies each salary value by 4 the deviation (`salary_mean`) into the power of 4 and adds it into `pow_deviations`. Once all salary values are passed through the loop, the code calculates the kurtosis by dividing the sum of fourth power deviations by (power of the fourth-(`salary_standard_deviation`)). Thereafter it terms the kurtosis value, which essentially the level of the clustering of the salary values distribution against a normal distribution.

```
In [14]: pow_deviations = 0 # Initializing the sum of fourth power deviations

# Calculating the sum of fourth power deviations
for x in salary_column:
    deviation = x - salary_mean
    pow_deviation = deviation ** 4
    pow_deviations = pow_deviations + pow_deviation

# Calculating the kurtosis of salary values
salary_kurtosis = pow_deviations / len(salary_column) / (salary_standard_deviation ** 4)

salary_kurtosis # Printing the kurtosis of salary values

Out[14]: 3.823019754941828
```

Figure 12:Kutosis of the column "salary_in_usd"

Function used in this question:

- `len()` :This function is used to get the length or number of elements in an object.

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

Correlation refers to the measure of linear relationship for two variables, so here, it helps us to understand how different column of integers within this data Frame relate to each other in terms of their values.

These two codes lines in Python mainly used for calculations, and the pandas library is specifically devoted to analysis. The first line creates `int_columns` as a name of the new column. `int_columns = dataframe.select_dtypes(include=['int64'])` is the new Data Frame that is generated from the original DataFrame (`dataFrame`), while the filtering process retains only columns with data type of `int64` among them, which is a usual representation of the '64-bit' integer values. Selecting the types of records in a DataFrame session is helped by the `select_dtypes()` function. Subsequently, `int_columns.corr()` is formula that returns the correlation matrix within the `int_columns` DataFrame of the columns of `int_columns`.. Based on this investigation, patterns that lie, and interdependencies within the data form may be uncovered.

```
In [17]: int_columns = dataframe.select_dtypes(include=['int64'])
int_columns.corr()
```

```
Out[17]:
```

	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

Functions used in this method:

`corr()`: This function calculates the correlation matrix among the columns of a Data Frame.

5 Data Exploration

Data exploration being a very crucial phase of the analysis, we utilize modules of Python and its libraries like pandas and matplotlib to delve even more into salaries of data scientists. Such a stage is very important as it entails the examination of several attributes which then analyzing their underlying trends and outlying cases. The first step is determining the most presented job titles; therefore, bar graphs will be used to display the frequencies of these positions which will give an idea about the job pretense within the sector of data science. The next phase entails the study of how the experience level correlates with different median salaries, as well as constructing different bar and line charts that reflect average salary across experience groups. This allows for dialog about occupational mobility and pay disparities in the given field, not only that. Such in addition histograms and box plots are used to scrutinize salary distribution, pointing outliers and general tendencies. Such a methodical approach sets up a good background for analyzing the dataset that in the end will facilitate the taking of informed decisions and the detailed study of the current stage.

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

First, we start by code by taking the top 15 jobs in a variable, which is obtained through the “value_counts()” method of the Data Frame, and it returns a Series consists of the count of unique value. Then, it creates the bar plot by using the plot() method in which the top job titles and their frequency are plotted in bar type. Param="kind='bar'" will generate a bar chart, color="lime" will set the bars color to some shade of lime green, and edgecolor="black" will set the color of bars edges to some black shade. Further, the code provides the headline "Top 15 jobs" as the title of the plot, designates the x-axis as "Job Titles" and the y-axis as "Scale".

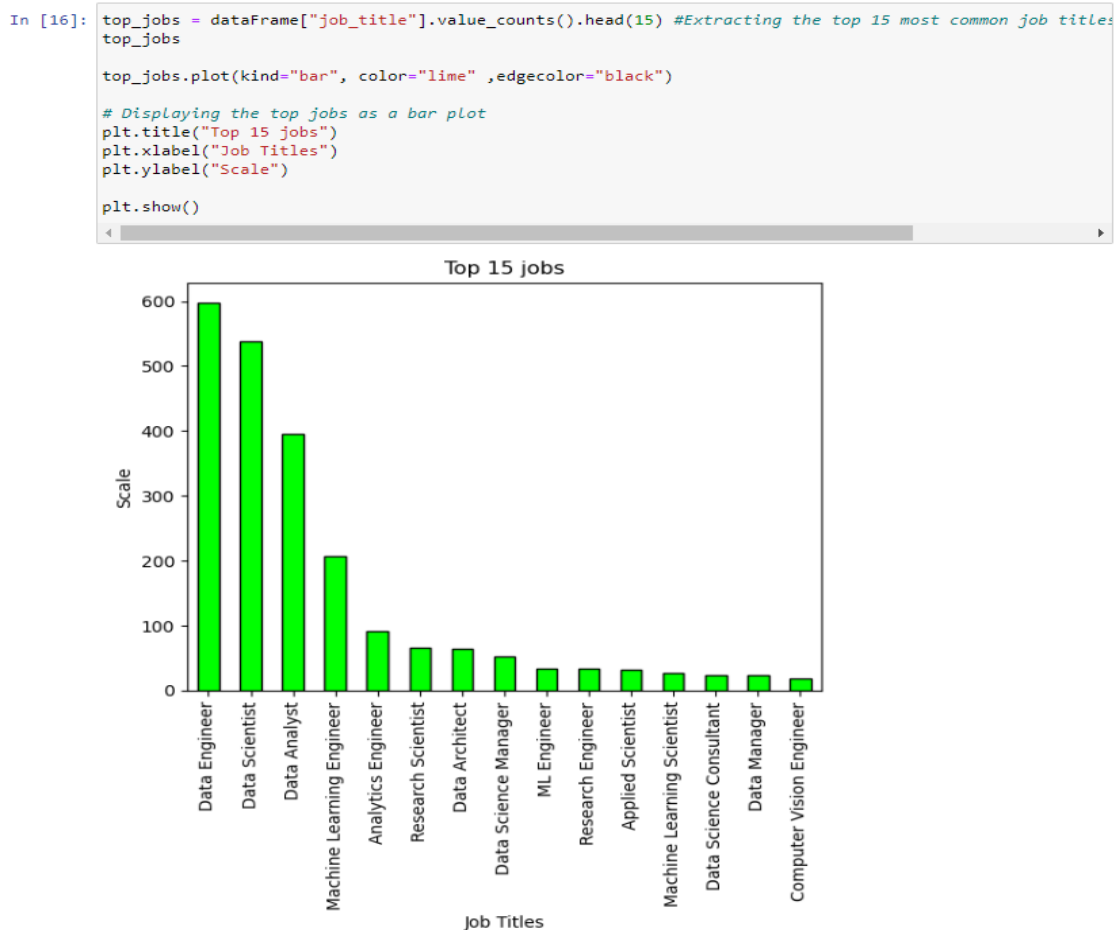


Figure 13:Top 15 jobs.

Function used in this question:

- `Value_count()` : This method counts the occurrences of unique values in a Series and returns a new Series containing counts of each unique value.

5.2 Which job has the highest salaries?

The Data Frame is grouped by using “groupby” method grouping titles of jobs and the subsequent mean is found using the function mean. This way a calculated column is added, it shows the average salary per job titles. After that, it extracts the job salaries data and then the graph of the top 15 average salaries by a job title, the values of which are selected and plotted in a bar plot in a position of the top 15 highest average salaries. The chart argument is bar, color="Lime green", with border color black. Again, it creates a title that fits this plot ("Top 15 Average Salaries by Job Title"), writes on an x-axis (which provides the data on Job Titles) and writes on y-axis (which is labeled with Average Salary) and finally depicts the plot.

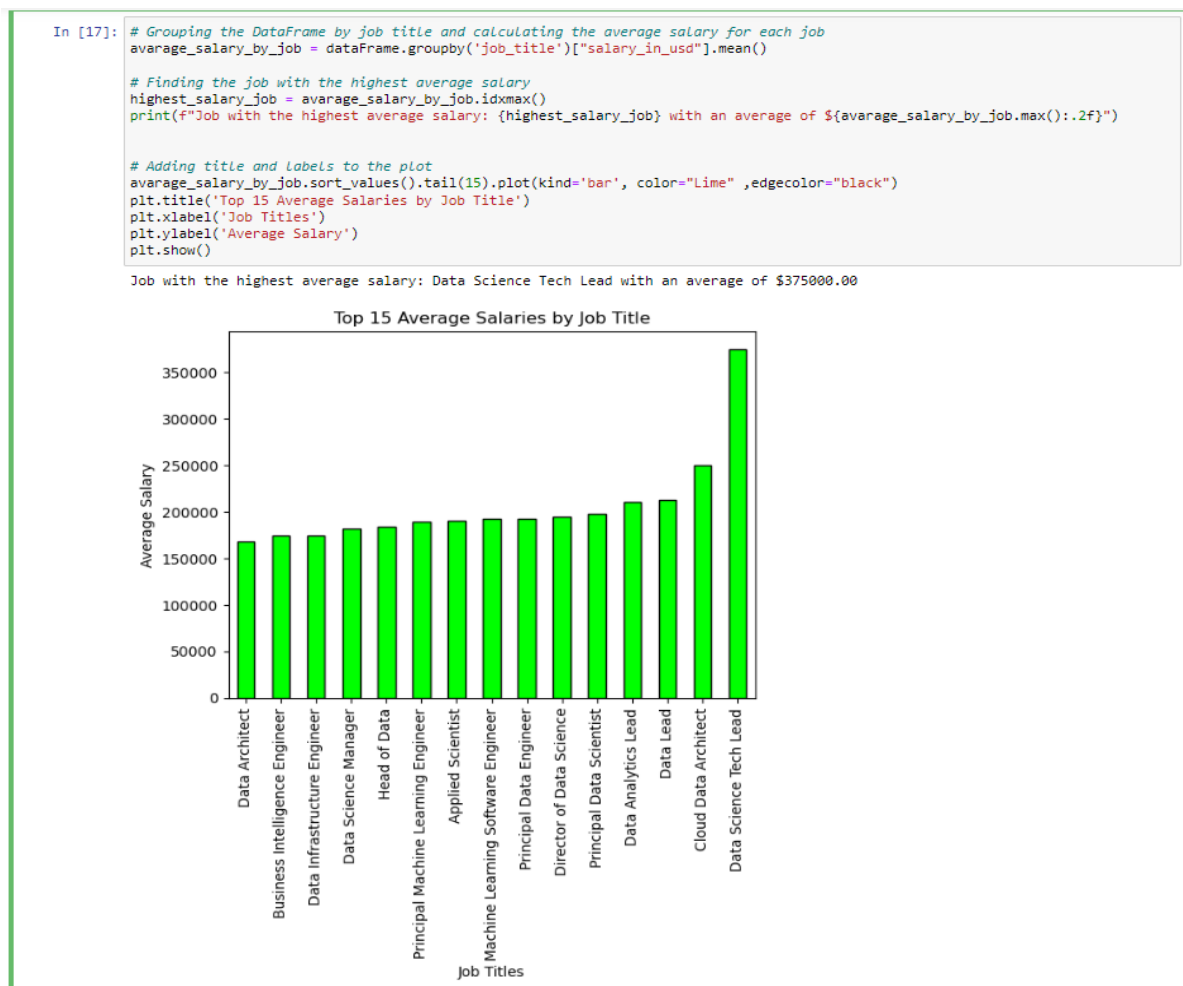


Figure 14: Highest salaries among the top jobs.

Functions used in this question are:

- `groupby()`: This method groups the DataFrame by a specified column (in this case, "job_title") and allows for operations to be applied within each group.
- `mean()`: This aggregation function calculates the mean (average) value of each group.
- `plot()`: This method generates plots based on data in a DataFrame or Series. In this case, it creates a bar plot.
- `plt.title()`, `plt.xlabel()`, `plt.ylabel()`, `plt.show()`: These functions from the matplotlib library are used to set the title, x-axis label, y-axis label, and display the plot, respectively.

5.3 Write a python program to find out salaries based on experience level.

The Data Frame is grouped by the experience level using the `groupby()` function combined with the `mean()` aggregation method. Following this, the machine will print the average salary for each level of experience. To continue, it visualizes the average wage as bar plot which displays the average salary values on the y-axis and the experience levels on the x-axis. The parameters `kind='bar'`, `color="green"`, `edgecolor="black"` define the charting type, color of the bars (lime green), edge color of the bars (black) respectively. Then, it puts a title in the plot ("Average Salary by Experience Level"), plots the x-axis as "Experience Level", and sets up the y-axis as the "Average Salary", and brings up the graph.

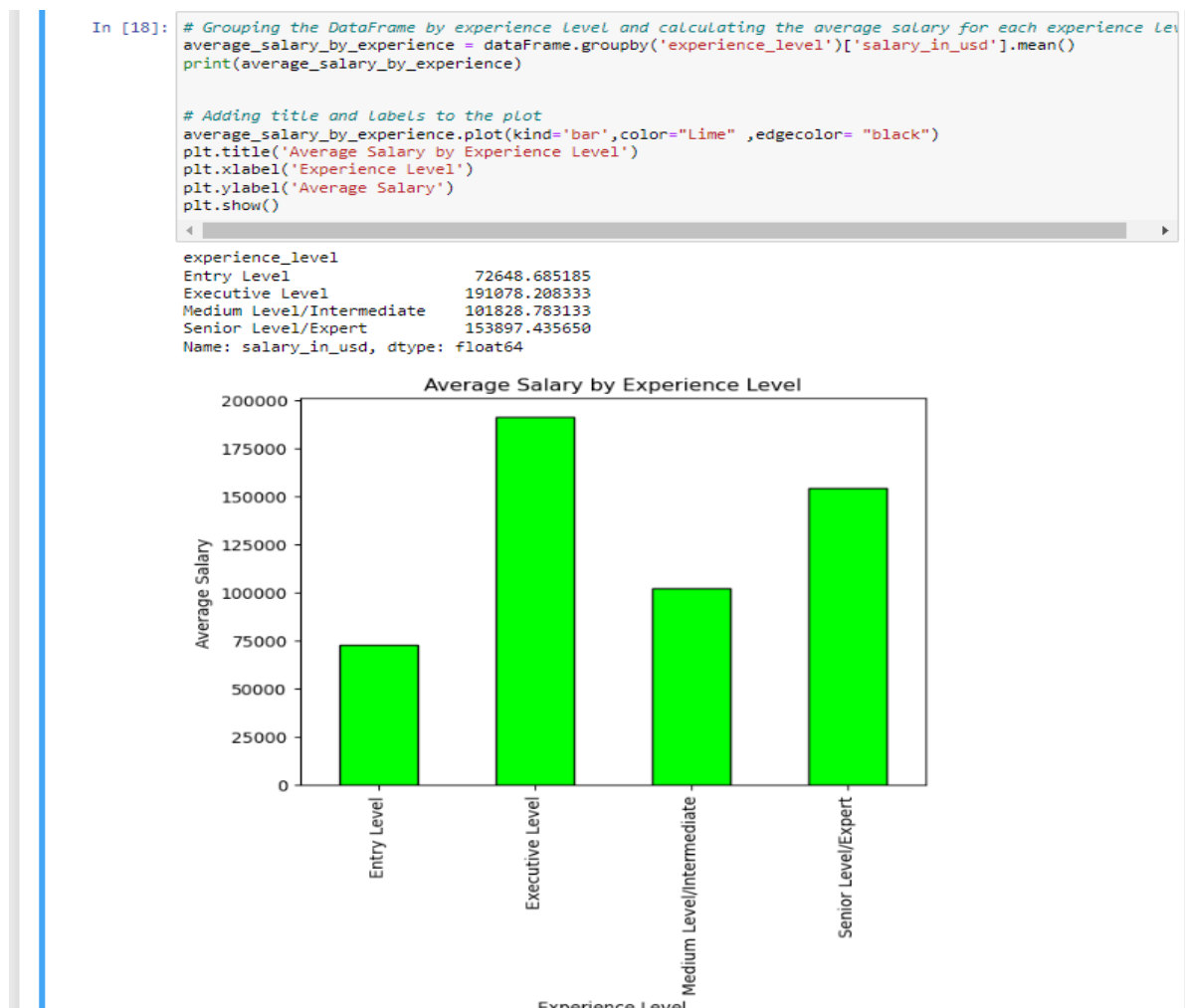


Figure 15: Avarage Salary by Experience level.

Functions used in this question are:

- `groupby()`: This method groups the DataFrame by a specified column (in this case, "experience_level") and allows for operations to be applied within each group.
- `mean()`: This aggregation function calculates the mean (average) value of each group.
- `plot()`: This method generates plots based on data in a DataFrame or Series. In this case, it creates a bar plot.
- `plt.title()`, `plt.xlabel()`, `plt.ylabel()`, `plt.show()`: These functions from the matplotlib library are used to set the title, x-axis label, y-axis label, and display the plot, respectively.

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

The first plot is a histogram, created using the `plot()` method with `kind="hist"`. It displays the distribution of salary values in the "salary_in_usd" column and work year in "work_year". The parameter `bins=20` specifies the number of bins (intervals) for the histogram, `edgecolor="black"` sets the color of the edges of the bars to black, and `color="Lime"` sets the color of the bars to lime green. Additionally, the code sets the title of the plot to "Salary Distribution" and "Work_year", labels the x-axis as "Salary" "Work_years", and labels the y-axis as "Frequency".

The second plot is a box plot, created using the `plot()` method with `kind='box'`. It provides a visual summary of the distribution of salary values, including information about the median, quartiles, and potential outliers.

```
In [19]: # Plotting a histogram to visualize the distribution of salaries
dataFrame['salary_in_usd'].plot(kind="hist", bins=20, edgecolor="black", color="lime")
plt.title('Salary Distribution')
plt.xlabel('Salary')
plt.ylabel('Frequency')
plt.show()

# Plotting a box plot to visualize the distribution of salaries
dataFrame['salary_in_usd'].plot(kind='box')
plt.title('Salary Distribution')
plt.ylabel('Salary')
plt.show()
```

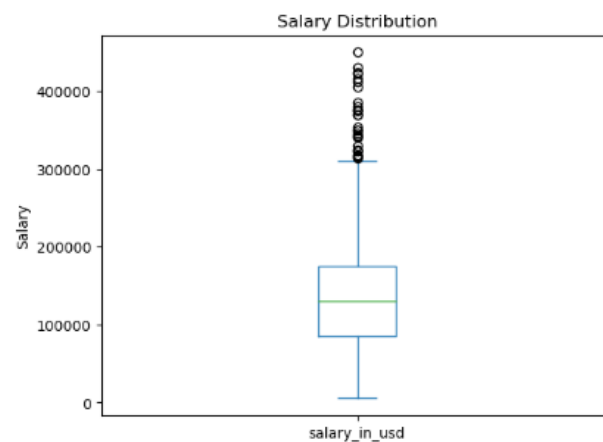
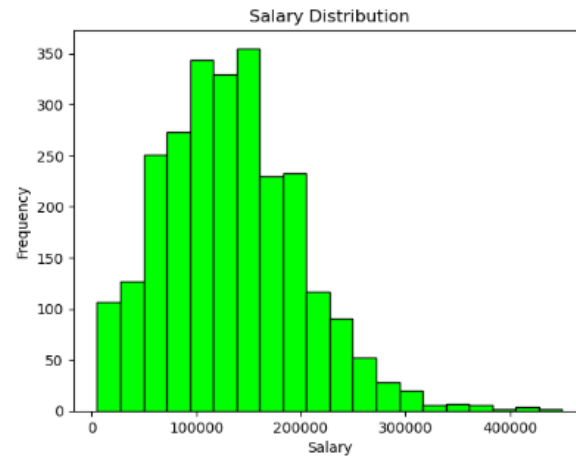


Figure 16: Histogram and box plot of salary_in_usd variable.

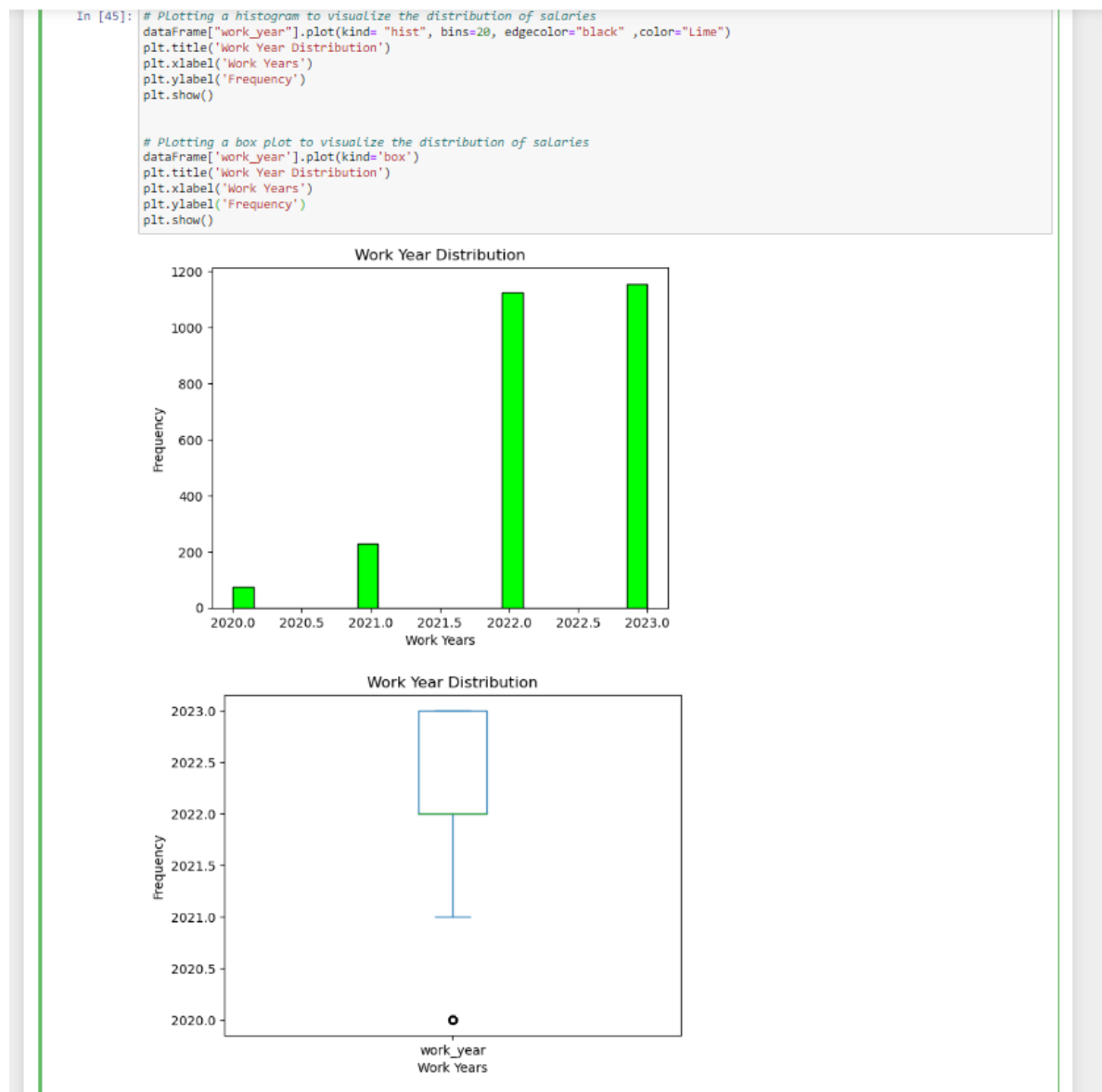


Figure 17: Histogram and box plot of work_year_variable

Functions used in this question are:

- `plot()`: This method generates plots based on data in a DataFrame or Series. In this case, it creates a bar plot.
- `plt.title()`, `plt.xlabel()`, `plt.ylabel()`, `plt.show()`: These functions from the matplotlib library are used to set the title, x-axis label, y-axis label, and display the plot, respectively.

6 Conclusion

To sum it up, the strategic deployment of data analysis is not a luxury but a must for the contemporary businesses. It is the core that transits the raw data into a valuable resource and gives the organizations the wherewithal to tackle the many complex markets that is trending today. Businesses will achieve the utmost precision in their data analysis through implementing advanced data analytics methods; while this will ensure the reliability of the strategic decisions based on the insights. This dedication to data-driven decision-making not only improves operational efficiencies and customer engagement but also puts the companies on top of the innovation and competitiveness. That is why data analysis is crucial for any organization that strives to maintain and advance on the highly competitive market as well as the data oriented modern world.