LONDON METROPOLITAN UNIVERSITY

islington college
(इस्लिङ्टन कलेज)

APPLIED DATA SCIENCE

60% Individual Coursework

2023-24 Autumn

Student Name: AASUTOSH KUMAR VERMA

London Met ID: 22085760

College ID: NP01AI4S230020

Assignment Due Date: Monday, May 13, 2024

Assignment Submission Date: Sunday, May 12, 2024

Word Count: 1855

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.

Contents

ABSTRACT

This document presents a comprehensive analysis of a given DataFrame, focusing on the extraction of meaningful insights and the application of analytical techniques. The analysis is question-driven, ensuring that the results are tailored to provide specific answers and solutions. The document covers various stages of the data analysis process, including data cleaning, exploration, visualization, and interpretation. It also discusses the application of statistical and machine learning methods to uncover patterns and trends in the data. The ultimate goal of this document is to enable readers to make data-driven decisions and strategies based on the findings

ACKNOWLEDGEMENTS

I would like to express my deepest appreciation to my mentor, Dipshor Sir, who has the attitude and the substance of a genius. He continually and convincingly conveyed a spirit of adventure in regard to data analysis and an excitement in regard to teaching. Without his guidance and persistent help, this project would not have been possible. His guidance helped me in all the time of research and writing of this document. I could not have imagined having a better advisor and mentor for my project.

INTRODUCTION

Welcome to our journey through a dataset that revolves around the fascinating world of data scientists. This dataset is like a treasure chest, filled with nuggets of information about their work years, experience levels, job titles, salaries, and more.

In the Data Understanding phase, we'll be like detectives, deciphering what each column in our dataset means. We'll also create a table that shows how complete our data is, by looking at the non-null count for each column.

Next, we'll roll up our sleeves for the Data Preparation phase. This is where we get our hands dirty with the data, loading it into a pandas DataFrame, trimming off unnecessary columns, dealing with those pesky missing values, checking for duplicate values, and giving the values in the experience level column a makeover for better understanding.

Then comes the Data Analysis phase, where we'll put on our statistician hats. We'll crunch numbers to calculate the sum, mean, standard deviation, skewness, and kurtosis of a chosen variable. We'll also look at how all the variables in our dataset interact with each other by calculating their correlations.

Finally, we'll dive deeper into the data in the Data Exploration phase. We'll find out which jobs are the top 15 in demand and illustrate this with a bar

graph. We'll also find out which job pays the highest salaries and represent this with another bar graph.

This document is like a map, guiding you through the dataset and the insights we can extract from it. It showcases the power of Python in handling and analyzing data, and the crucial role of data analysis in making informed decisions. To make the journey easier, we've included screenshots along the way.

1. Data Understanding.

- **Explanation what the dataset is about.**

work_year: This is the year the data was recorded, which is 2020 to 2023 for all entries.

experience_level: This tells us the experience level of the employee. ‘SE’ stands for Senior, ‘MI’ for Mid-level, and ‘EN’ for Entry-level.

employment_type: This indicates whether the employee is Full-Time (FT) or Contract (CT).

job_title: This is the role or position of the employee in the company, such as Data Scientist, ML Engineer, etc.

salary: This is how much the employee earns in their local currency.

salary_currency: This is the currency in which the employee is paid.

salary_in_usd: This is the employee’s salary converted into US dollars for comparison.

employee_residence: This is the country where the employee lives, represented by its two-letter country code.

remote_ratio: This tells us what percentage of the time the employee works remotely. A value of 100 means they always work remotely, and 0 means they never do.

company_location: This is the country where the company is located.

company_size: This indicates the size of the company. ‘S’ stands for Small, ‘M’ for Medium, and ‘L’ for Large.

- **Making a table mentioning below information.**

Table 1 DATA TYPE

Column No.	Column Name	Non-Null Count
0	work_year	3755 non-null
1	experience_level	3755 non-null
2	employment_type	3755 non-null
3	job_title	3755 non-null
4	salary	3755 non-null
5	salary_currency	3755 non-null
6	salary_in_usd	3755 non-null
7	employee_residence	3755 non-null
8	remote_ratio	3755 non-null
9	company_location	3755 non-null
10	company_size	3755 non-null

2. Data Preparation.

- **Question & Answer along with screenshots.**
    - Write a python program to load data into pandas DataFrame

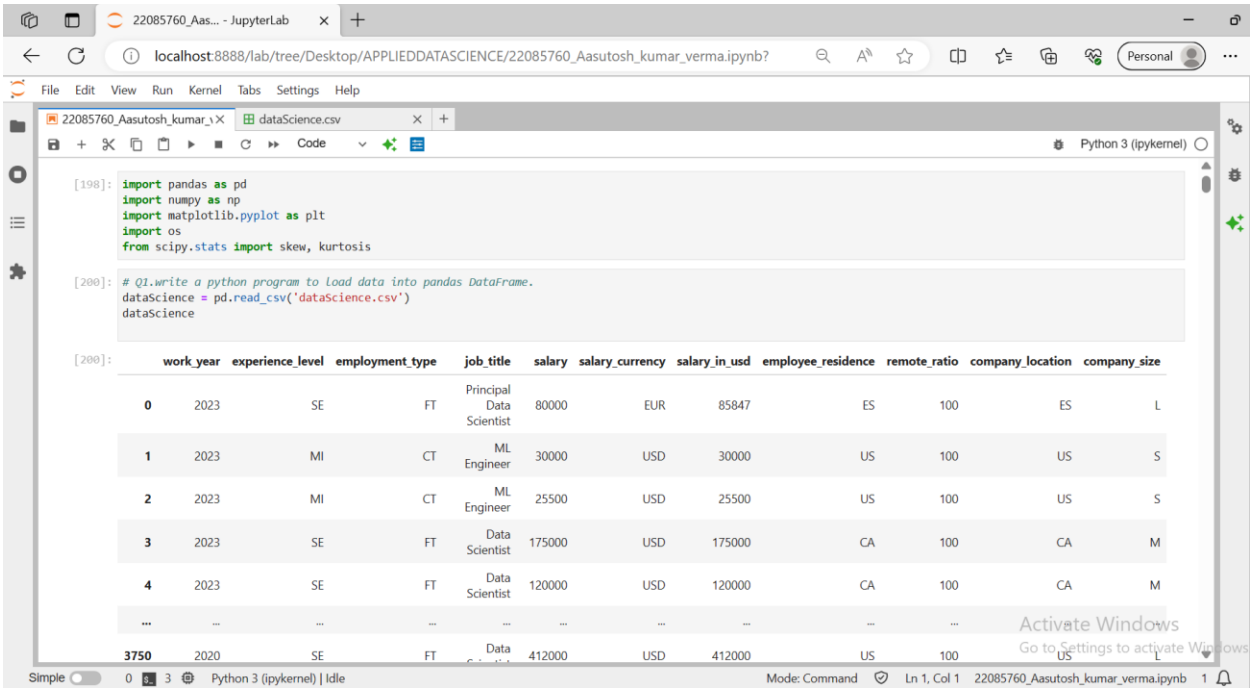

Figure 1 LOAD DATA

➔ This above screenshots shows how to load csv data file in jupyter environment.

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

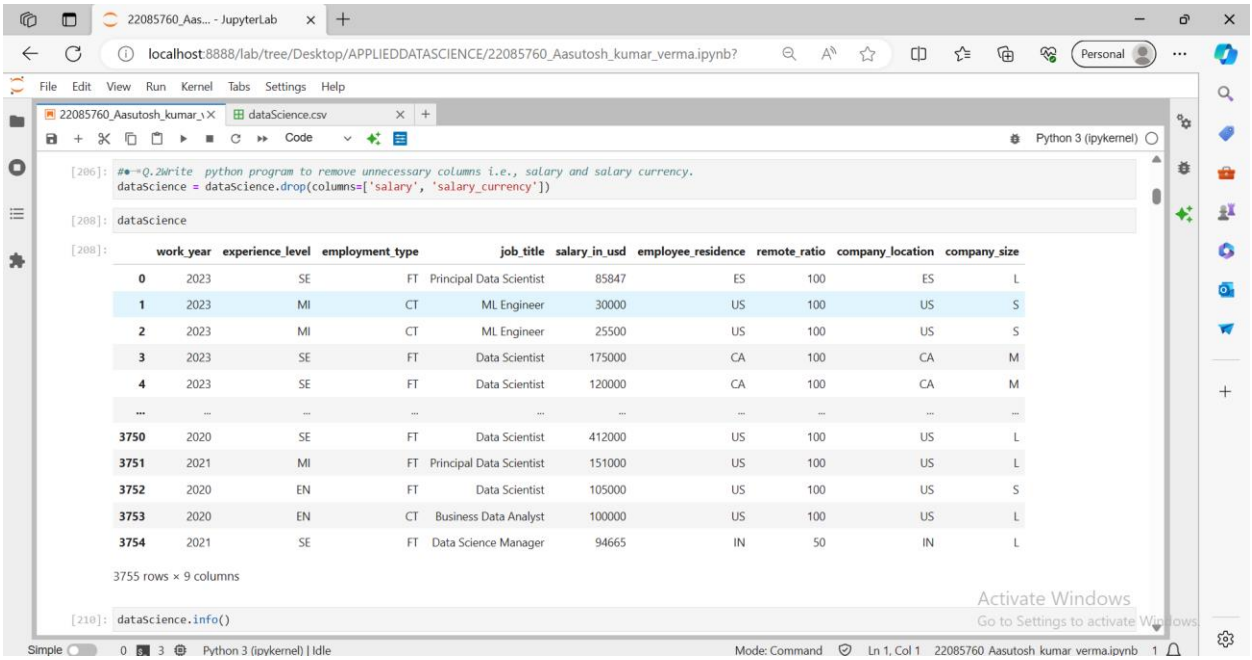

Figure 2 DROP COLUMN

➔ This above screenshots shows how to remove unwanted columns.

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

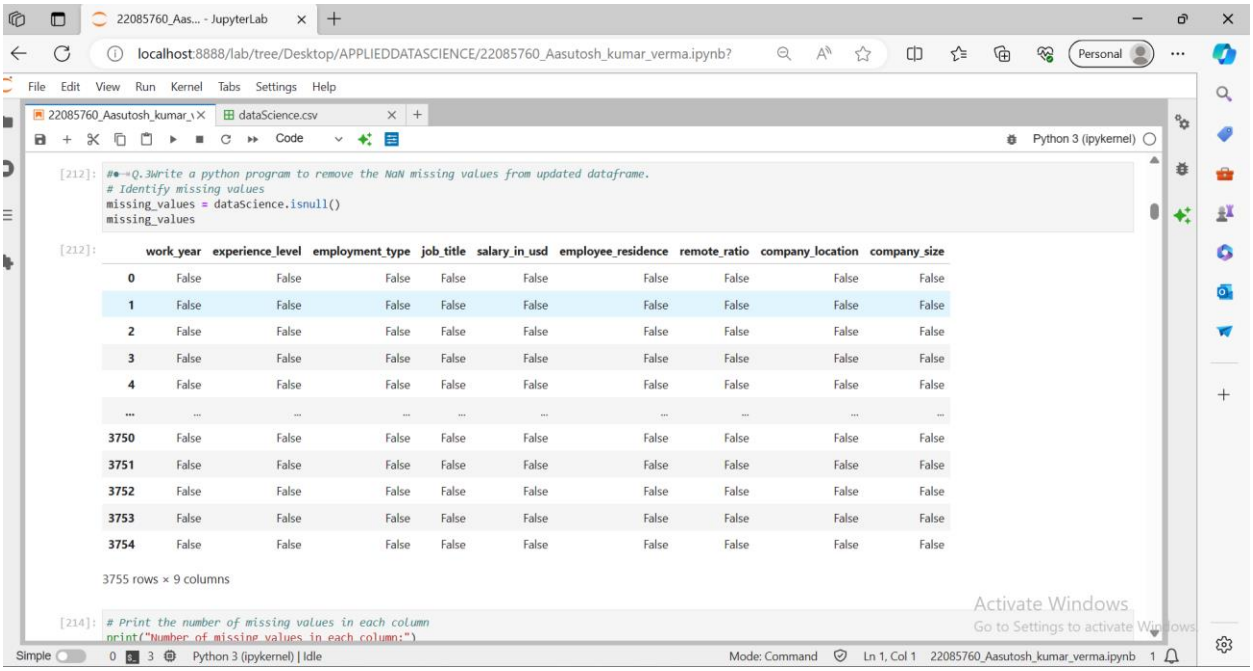

Figure 3 NAN MISSING VALUE

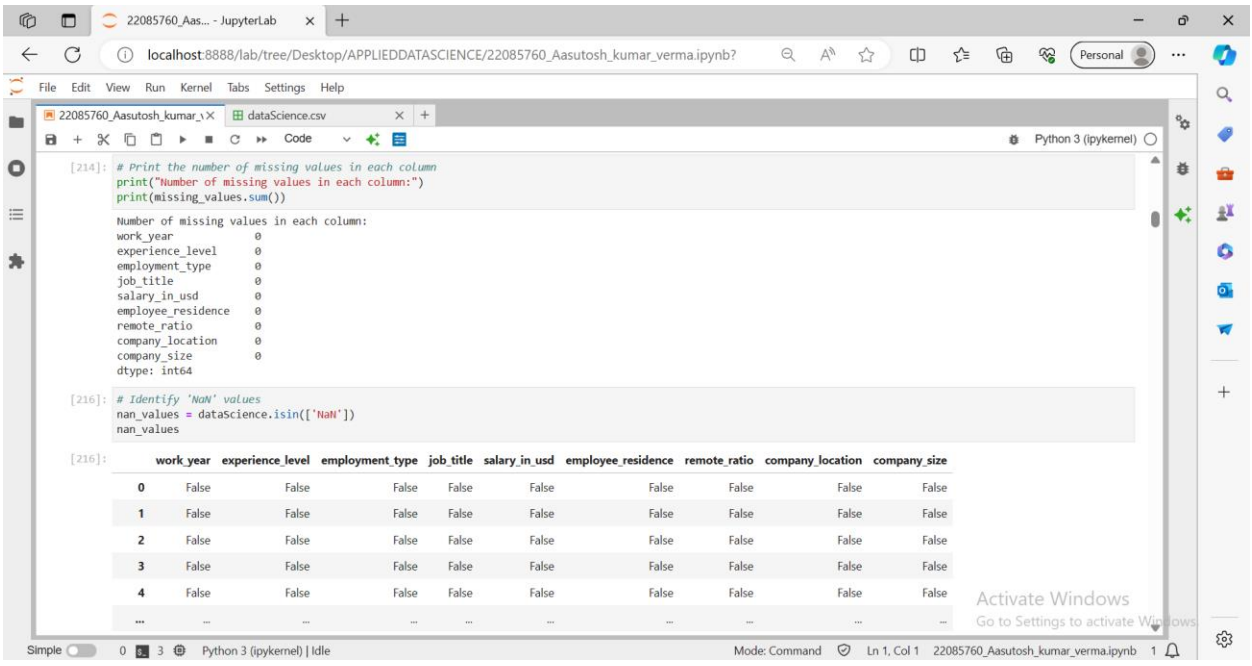

Figure 4 NAN MISSING VALUE

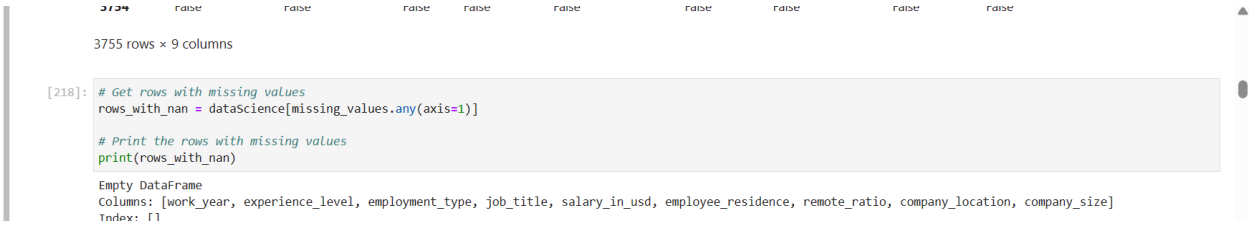

Figure 5 NAN MISSING VALUE

➔ This above screenshots shows how to check missing values with different ways as a part of data cleaning to get accurate value while analytical process.

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

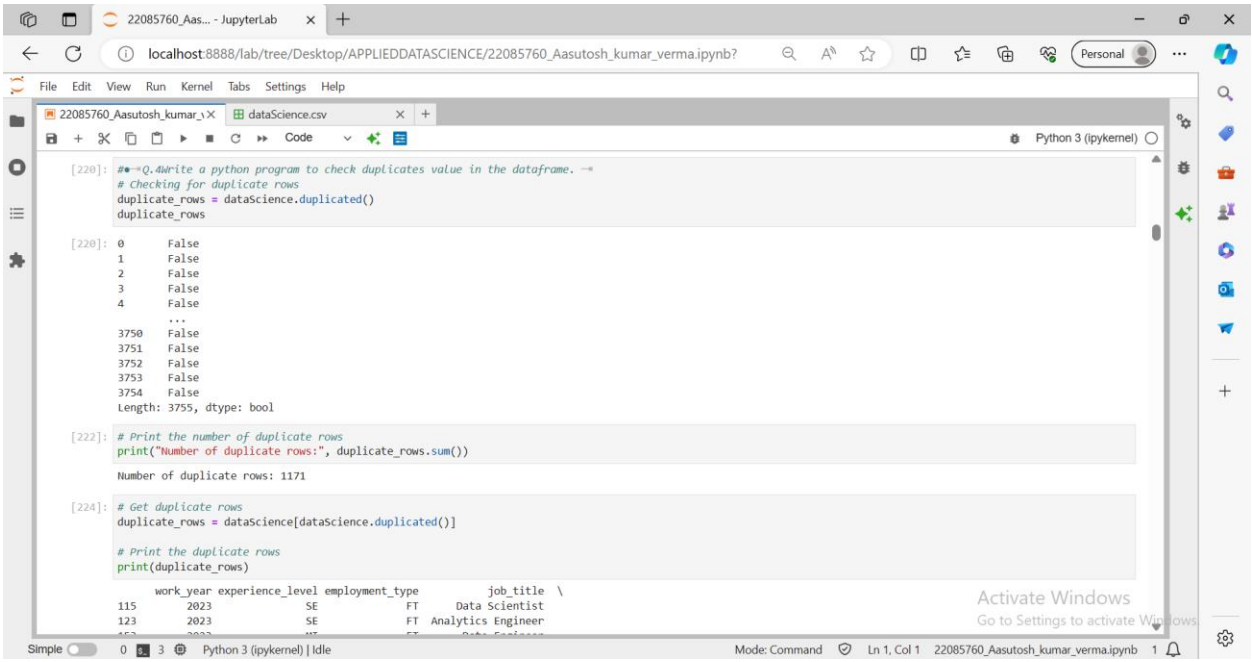

Figure 6 CHECK DUPLICATE

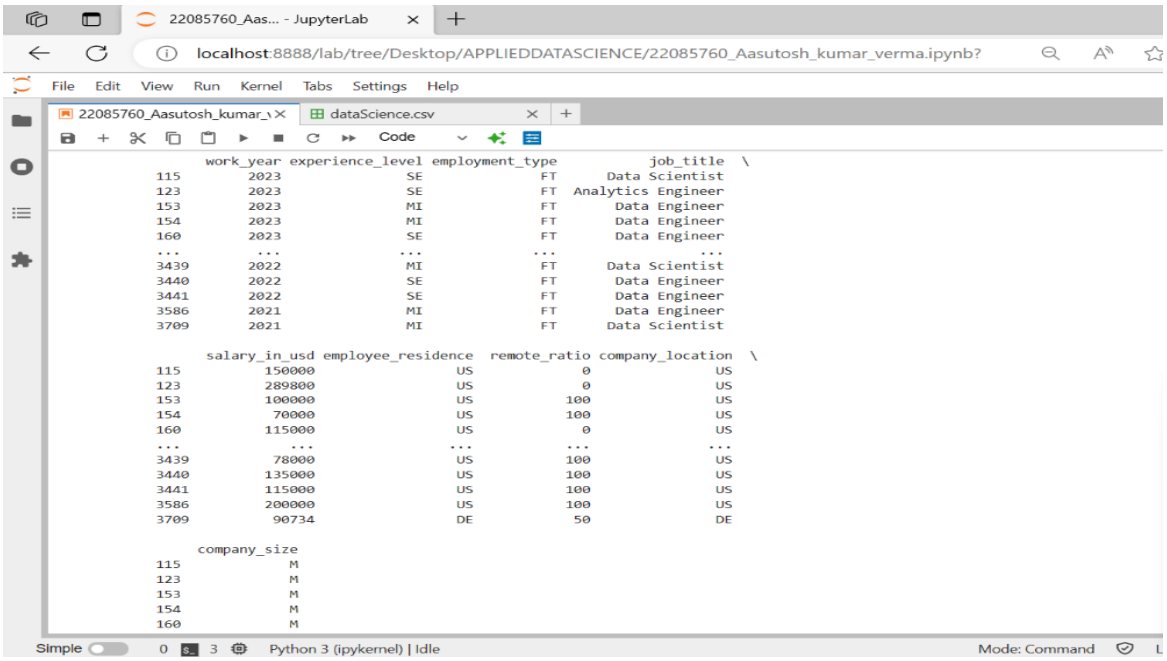

Figure 7 CHECK DUPLICATE

➔ This above screenshots shows how to check duplicates value in a dataframe.

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

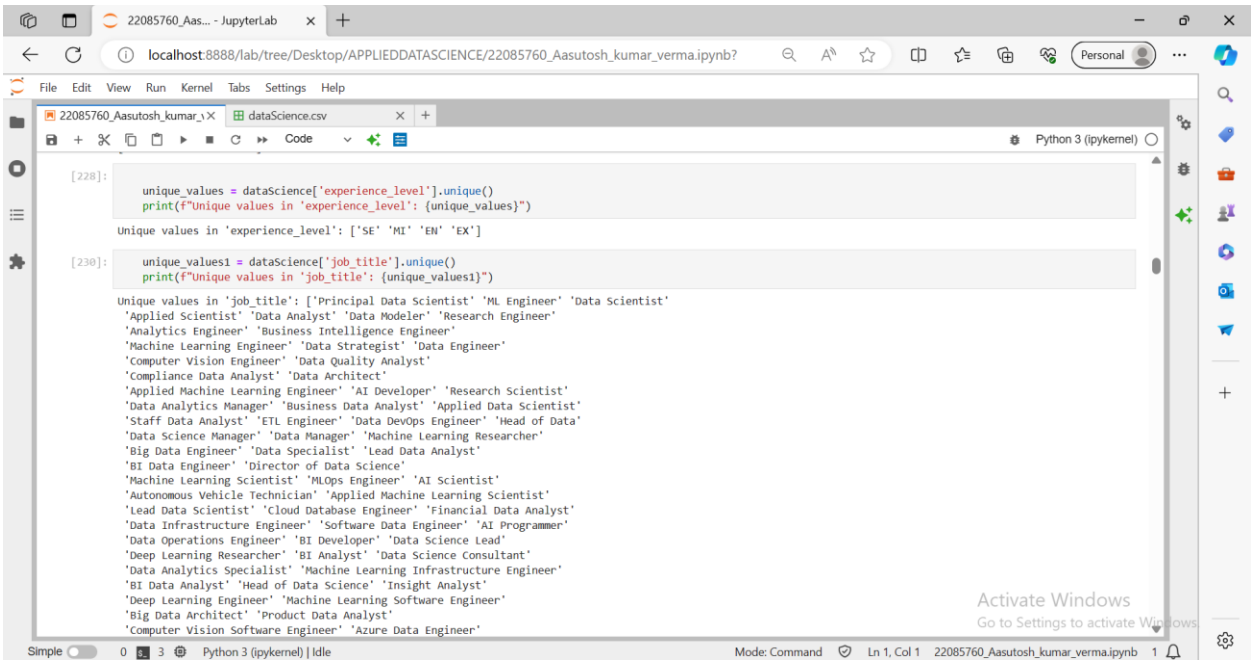

Figure 8 UNIQUE VALUES

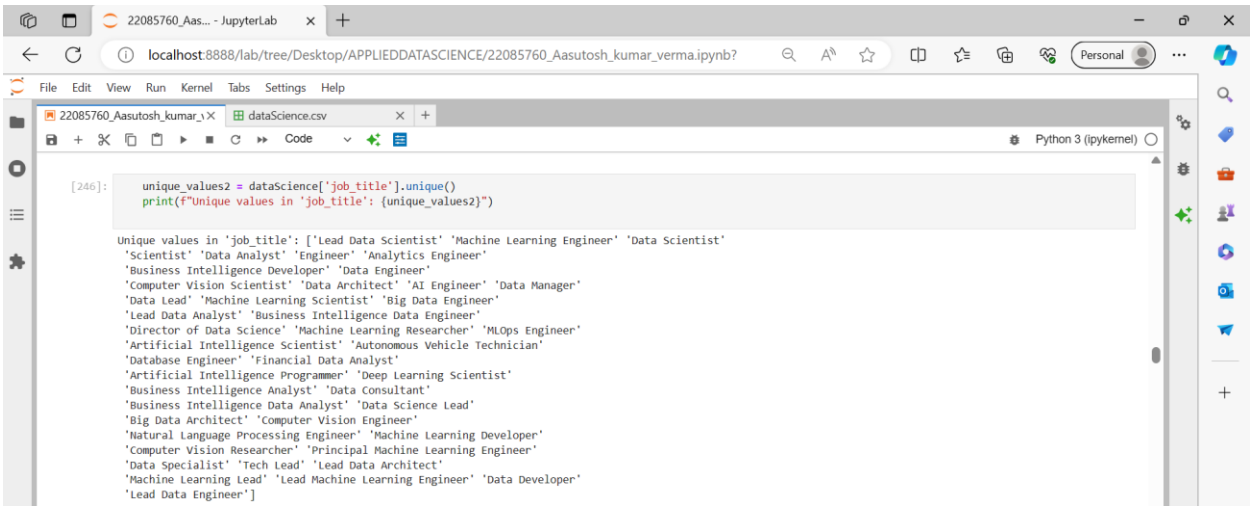

Figure 9 UNIQUE VALUES

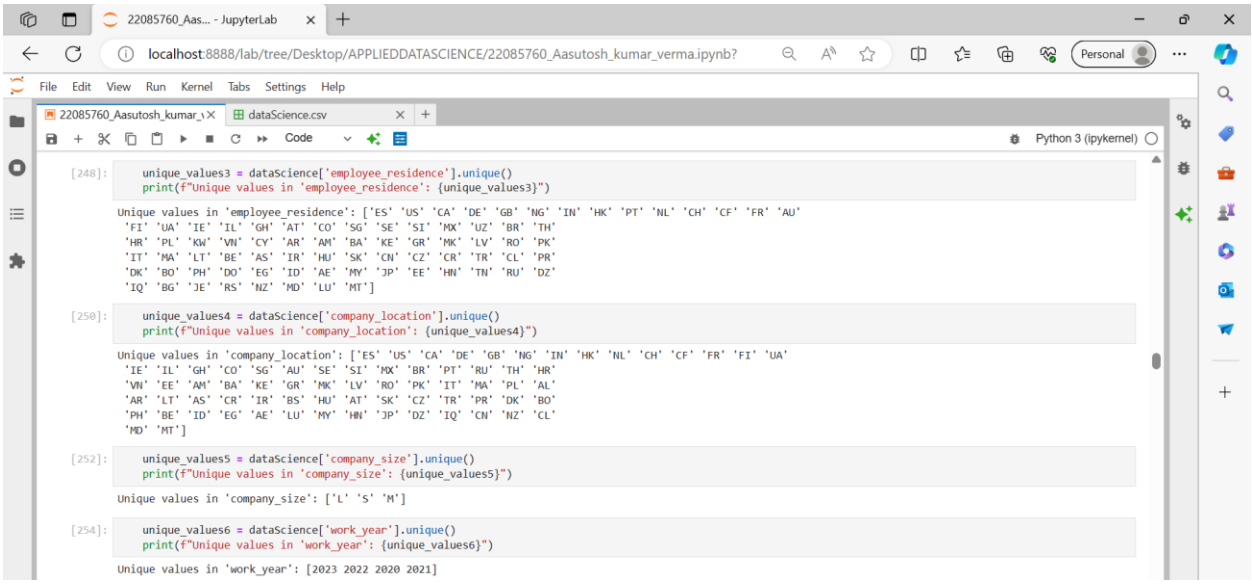

Figure 10 UNIQUE VALUES

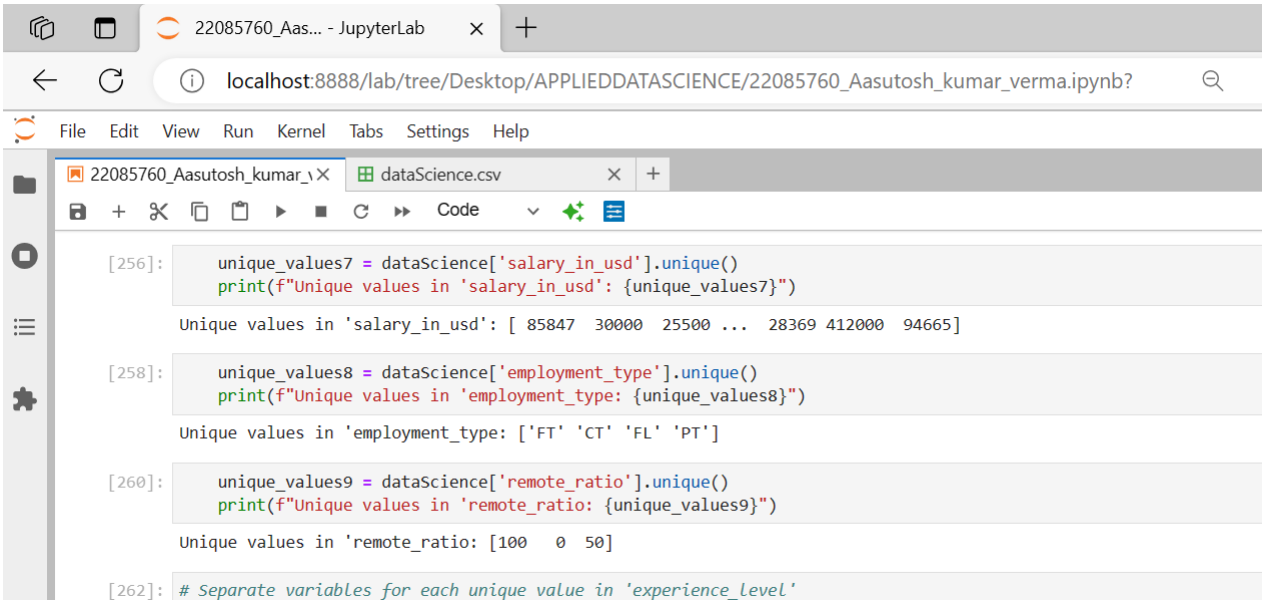

Figure 11 UNIQUE VALUES

➔ This above screenshots shows how to check unique values in a columns.

- Rename the experience level columns as below.

  SE – Senior Level/Expert

  MI – Medium Level/Intermediate

  EN – Entry Level

  EX – Executive Level

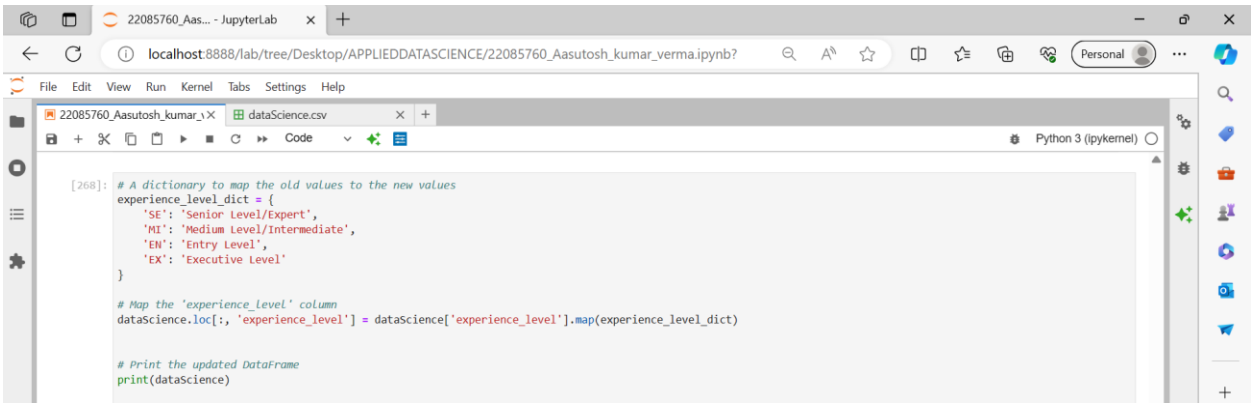

Figure 12 COLUMN VALUE RENAME

➔ This above screenshots shows how to rename column current value with new one.

- Explain of output.

3. Data Analysis.

- Question note and explain of output information.

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

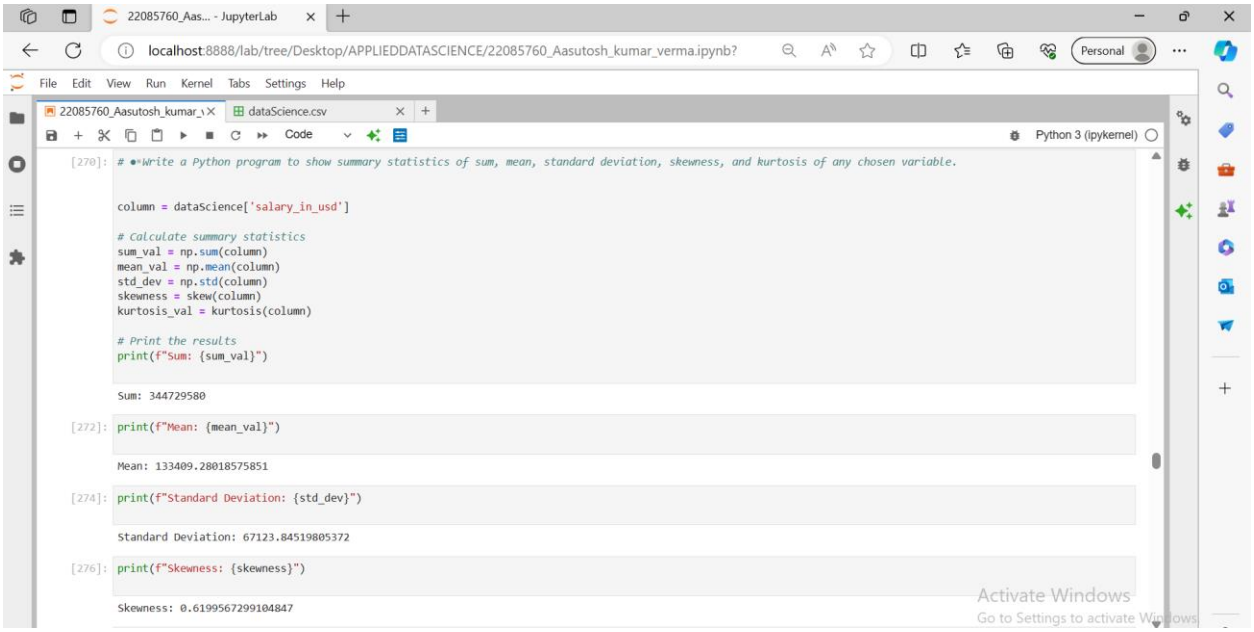

Figure 13 BUSINESS ANALYTICS

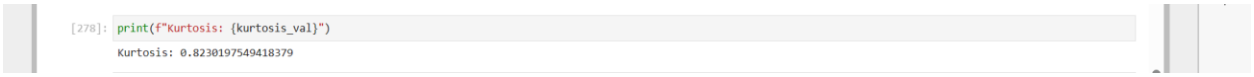

Figure 14 BUSINESS ANALYTICS

3.1. Statistics of sum.

➔ 33729580 USD.

3.2. Statistics of Mean.

➔ 133409 USD.Its shows average salary in usd a employees recived.

3.3. Statistics of standard deviation.

➔ 67123.84519805372 USD. Its shows that the company provide salaries equally or not . so if it great means company need to ensure that the salaries is equally paid.

3.4. Statistics of skewness.

➔ 0.6199567299104847. Its shows that there are number of employess that receive more salaries.

3.5. Statistics of Kurtosis.

➔ 0.8230197549418379. Its shows that the most employees in the company have salaries that are close to the average(Mean) salary.

➔ This above screenshots shows whether the analysis part or process is right or not by comparig the values received from analytical process using statistical calculations.

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

```
[284]: # only the numerical columns in the DataFrame
numerical_dataScience = dataScience[['work_year', 'salary_in_usd', 'rem

# Calculate the correlation matrix
correlation_matrix = numerical_dataScience.corr()

# Print the correlation matrix
print(correlation_matrix)
```

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

Figure 15 CORRELATION

➔ work_year and work_year (1.000000): This is the correlation of 'work_year' with itself, which is always 1. It simply means each variable perfectly correlates with itself.

➔ work_year and salary_in_usd (0.236958): This shows a positive but weak relationship between the number of years worked ('work_year') and the salary in USD. It suggests that as the number of years worked increases, the salary tends to increase slightly as well.

➔ work_year and remote_ratio (-0.219160): This shows a negative but weak relationship between the number of years worked ('work_year') and the remote work ratio. It suggests that as the number of years worked increases, the remote work ratio tends to decrease slightly.

➔ salary_in_usd and remote_ratio (-0.084502): This shows a very weak negative relationship between salary and the remote work ratio. It suggests that there's almost no relationship between the salary and the remote work ratio.

4. Data Exploration.

- **Question and explain of output information proper label of graphs.**

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

```
[94]: # Group by job_title and sum the salaries
job_salary = dataScience.groupby('job_title')['salary_in_usd'].sum()

# Sort by salary and get top 15
top_15_jobs = job_salary.sort_values(ascending=False).head(15)

# Plot
plt.figure(figsize=(10, 6))

# Generate a color palette with the same number of colors as bars
colors = plt.cm.viridis(np.linspace(0, 1, len(top_15_jobs)))

bars = plt.bar(top_15_jobs.index, top_15_jobs.values, color=colors)

# Add labels on top of bars
for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval, round(yval, 2), va='bottom', rotation=90)  # va: vertical alignment, rotation: rotate text

plt.title('Top 15 Jobs by Total Salary')
plt.xlabel('Job Title')
plt.ylabel('Total Salary in USD')

# Rotate x-axis labels
plt.xticks(rotation=90)

# Adjust y-axis limits
plt.ylim(0, max(top_15_jobs.values) * 1.2)  # Increase y-axis limit by 20% to fit text

# Add a legend
plt.legend(bars, top_15_jobs.index, loc='upper right')

plt.show()
```


Figure 16 15 JOBS

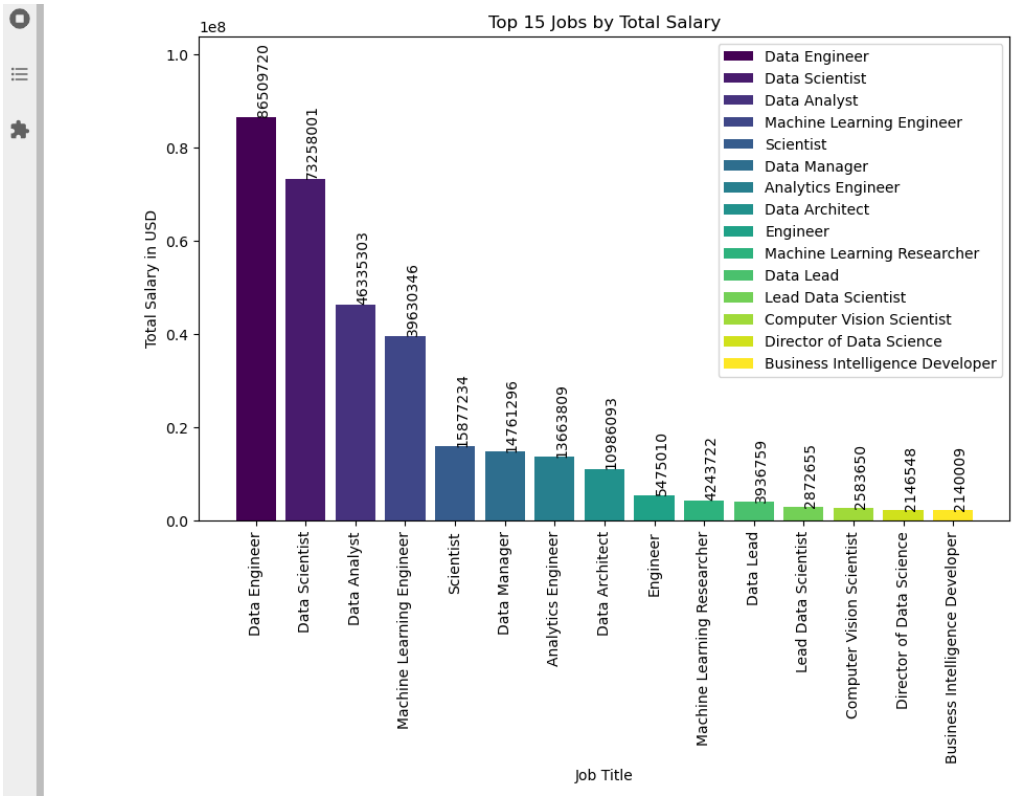

Figure 17 GRAPH OF 15 JOBS

➔ This screenshots shows that the data engineer holds higher demands out of 15 jobs.

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

```
[96]:

# Group by job_title and sum the salaries
total_salaries = dataScience.groupby('job_title')['salary_in_usd'].sum()

# Find the job with the highest total salary
highest_salary_job = total_salaries.idxmax()
highest_salary = total_salaries.max()

print(f"The job with the highest total salary is {highest_salary_job} with a total salary of ${highest_salary:,.2f}")

# Sort the total salaries
total_salaries = total_salaries.sort_values(ascending=False)

# Generate a color palette with the same number of colors as bars
colors = plt.cm.viridis(np.linspace(0, 1, len(total_salaries)))

# Create a new figure with specified figure size
fig, ax = plt.subplots(figsize=(14, 7))

# Plot the total salaries of all jobs
bars = ax.bar(total_salaries.index, total_salaries.values, color=colors)

# Add labels on top of bars
for bar in bars:
    yval = bar.get_height()
    ax.text(bar.get_x() + bar.get_width()/2, yval, round(yval, 2), va='bottom', rotation=90)  # va: vertical alignment, rotation: rotate text

ax.set_title('Total Salaries of Jobs')
ax.set_xlabel('Job Title')
ax.set_ylabel('Total Salary in USD')

# Rotate x-axis labels
plt.xticks(rotation=90)

# Adjust y-axis limits
ax.set_ylim(0, max(total_salaries.values) * 1.2)  # Increase y-axis limit by 20% to fit text

# Add a legend outside the plot area below x-axis labels
box = ax.get_position()
ax.set_position([box.x0, box.y0 + box.height * 0.1, box.width, box.height * 0.9])  # Shrink current axis's height by 10% on the bottom
ax.legend(bars, total_salaries.index, loc='upper center', bbox_to_anchor=(0.5, -0.15), fancybox=True, shadow=True, ncol=5)

plt.show()
```


Figure 18 HIGHEST SALARY

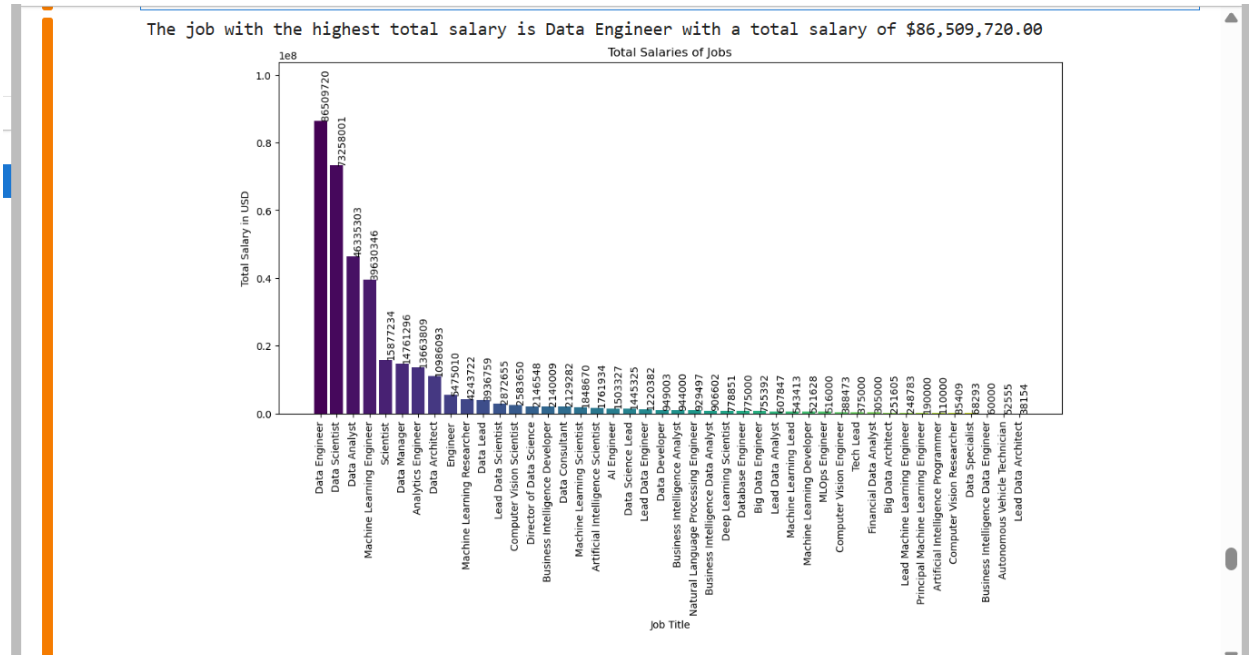

Figure 19 HIGHEST SALARY GRAPH

➔ This screenshots shows that the data engineer holds the highest salary.

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

```
[100]:

# Group the data by 'experience_level' and calculate the average salary for each group
average_salaries = dataScience.groupby('experience_level')['salary_in_usd'].mean()

# Generate a color palette with the same number of colors as bars
colors = plt.cm.viridis(np.linspace(0, 1, len(average_salaries)))

# Create a bar plot for the average salaries
fig, ax = plt.subplots(figsize=(14, 7))
bars = ax.bar(average_salaries.index, average_salaries.values, color=colors)

# Add labels on top of bars
for idx, value in enumerate(average_salaries.values):
    ax.text(idx, value, round(value, 2), ha='center', va='bottom', rotation=90)  # ha: horizontal alignment

# Add labels and title
ax.set_xlabel('Experience Level')
ax.set_ylabel('Average Salary in USD')
ax.set_title('Average Salaries by Experience Level')

# Rotate x-axis labels
plt.xticks(rotation=90)

# Adjust y-axis limits
ax.set_ylim(0, max(average_salaries.values) * 1.3)  # Increase y-axis limit by 30% to fit text

# Create a legend below the x-axis labels
box = ax.get_position()
ax.set_position([box.x0, box.y0 + box.height * 0.1, box.width, box.height * 0.9])  # Shrink current axis's height by 10% on the bottom
ax.legend(bars, average_salaries.index, loc='upper center', bbox_to_anchor=(0.5, -0.7), fancybox=True, shadow=True, ncol=5)

plt.tight_layout()
plt.show()
```

Figure 20 SALARY EXPENDITURE LEVEL

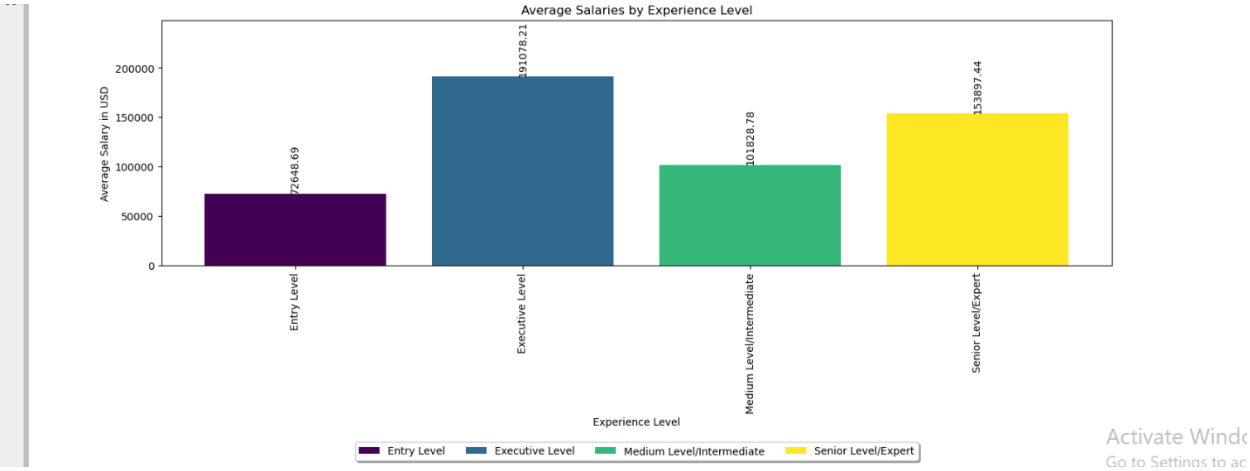

Figure 21 GRAPH OF EXPENDITURE LEVEL

➔ The screenshots shows that the employees who earns more salaries belongs to executive level.

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

```
[103]: # Create a histogram for 'work_year'
counts, bins, patches = plt.hist(dataScience['work_year'], bins = range(2020, 2024), edgecolor='black')

# Add frequency annotations on top of each bar
for count, bin, patch in zip(counts, bins, patches):
    plt.annotate(f'{int(count)}', (bin, count), xytext=(0, 5), textcoords='offset points', ha='center')

plt.title('Work Year Histogram')
plt.xlabel('Work Year')
plt.ylabel('Frequency')
plt.show()
```

Figure 22 HISTOGRAM OF WORK YEAR

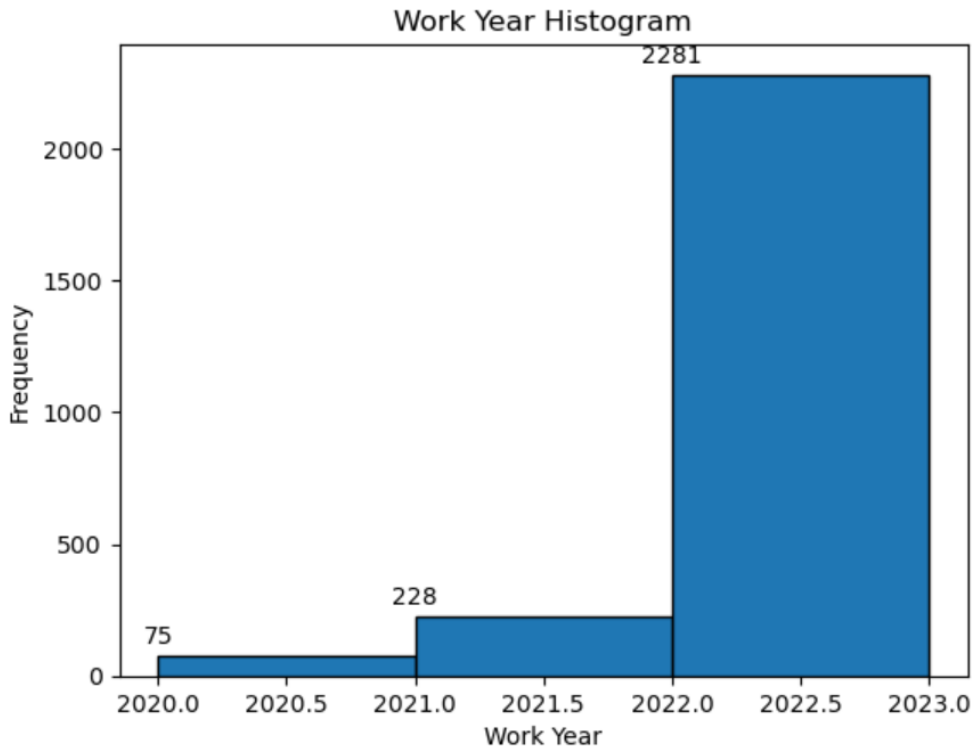

Figure 23 HISTOGRAM GRAPH OF WORK YEAR

```
[117]: # Create a box plot
plt.figure(figsize=(10, 6))
box_dict = plt.boxplot(dataScience['salary_in_usd'], vert=True, patch_artist=True, notch=True, widths=0.35)

# Change outline color, fill color and linewidth of the boxes
for patch in box_dict['boxes']:
    # Change outline color
    patch.set(color='#4286f4', linewidth=2)
    # Change fill color
    patch.set(facecolor='#e0e0e0')

# Change color and linewidth of the whiskers
for whisker in box_dict['whiskers']:
    whisker.set(color='#4286f4', linewidth=2)

# Change color and linewidth of the caps
for cap in box_dict['caps']:
    cap.set(color='#4286f4', linewidth=2)

# Change color and linewidth of the medians
for median in box_dict['medians']:
    median.set(color='#b22222', linewidth=2)

# Change the style of fliers and their fill
for flier in box_dict['fliers']:
    flier.set(marker='o', color='#e7298a', alpha=1)

# Set labels
plt.title('Box Plot of Salaries', fontsize=15)
plt.ylabel('Salary in USD', fontsize=12)
plt.xticks([1], ['Salaries'], fontsize=12)

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

Figure 24 BOX PLOT OF SALARY IN USD

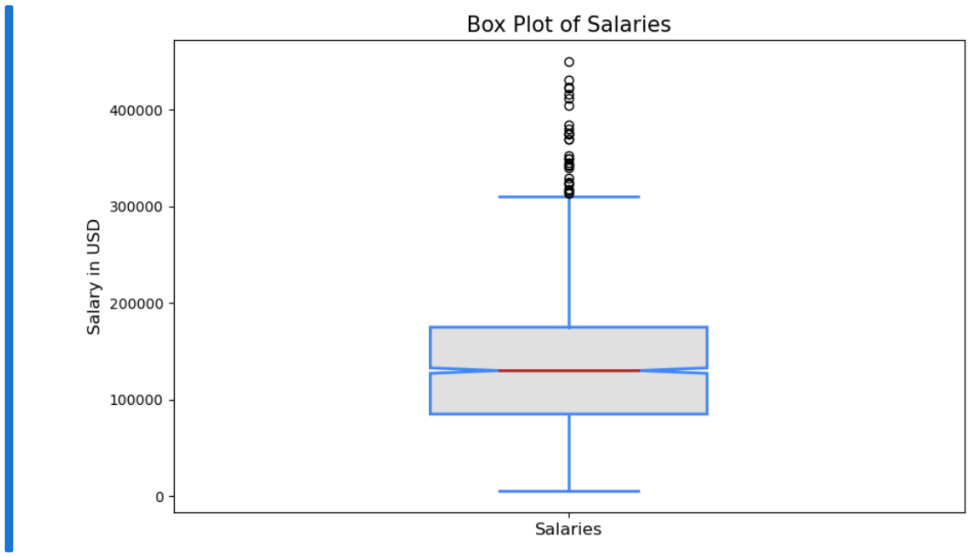

Figure 25 BOX PLOT GRAPH OF SALARY IN USD

- ➔ The screenshots of histogram shows that the increase of employess who started work between 2022 to 2023 year.
- ➔ FOR BOX PLOT:-
- ➔ Median (Middle Line): This is the middle salary. Half of the people earn less than this and half earn more.
- ➔ Box (Q1 to Q3): This is where the middle 50% of salaries fall. If the box is skewed, it means more people earn either towards the lower end (box skewed towards top) or higher end (box skewed towards bottom).
- ➔ Whiskers (Lines extending from the box): These represent the overall range of salaries, excluding extreme outliers. If the whiskers are long, it means there's a large spread in what people earn.
- ➔ Outliers (Dots): These are salaries that are unusually low or high compared to the rest. If there are many dots, it means there are many people with unusually high or low salaries.
- ➔ Notches: If the notches of two boxes do not overlap, this suggests a statistically significant difference between the medians.

5. Conclusion.

This project has highlighted the significance of the Python language in data analysis. While some believe Python may not be suitable for large-scale data analysis due to limitations in processing multiple lines of code simultaneously, this is a misconception. Python is indeed capable of handling extensive datasets with the appropriate use of libraries.

It is acknowledged that C, often referred to as the ‘mother of all languages,’ can be utilized alongside Python to optimize performance, particularly in global interpretation scenarios. Python was designed to simplify coding, and while it’s essential to critically evaluate information from scientific sources, news articles, and lectures, dismissing Python’s capabilities based on unverified claims would be misguided. Recent rumors of Google disbanding its Python team are unfounded and should not influence our perception of the language’s reliability.

The assertion that Python’s security can be compromised by C is an oversimplification. Security vulnerabilities are a concern for any language and are more dependent on the implementation rather than the language itself. It’s important to recognize that while C developers possess the knowledge to create sophisticated AI and robotics, this expertise is not exclusive to them, nor does it diminish Python’s contributions to these fields.

While Python does incorporate features from other languages, this interoperability is one of its strengths, allowing for a more comprehensive approach to problem-solving. If one’s goal is to excel in data analysis or machine learning, Python remains a strong contender due to its extensive libraries and supportive community.

In terms of data processing, this project has implemented data cleaning techniques such as removing duplicates, renaming values, and segregating unique values into separate variables for efficient access. These steps are crucial for achieving accurate results and are detailed in the code section of the report. parts as well (Mckinney, 2017).

6. References

Mckinney, W., 2017. *Python for Data Analysis.* 2nd ed. Mumbai: SHROFF.

Islington Library,Data science and analytical Books .