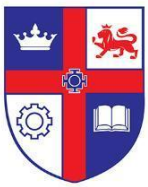
LONDON METROPOLITAN UNIVERSITY



Islington college
(इस्लिङ्टन कलेज)

CC5067NI-Smart Data Discovery

60% Individual Coursework

2023-24 Spring

Student Name: Bibek Kumar Thakur

London Met ID: 22085816

College ID: NP01CP4S230060

Assignment Due Date: Monday, May 13, 2024

Assignment Submission Date: Monday, May 13, 2024

Word Count: 1957

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

Table of Contents

Table of Figures.

#. Introduction to Coursework:

This is an individual assignment that weighted 60% of the marks of the module. From this Coursework Assignment, data analysis tasks, critical thinking/evaluation and problem-solving skills are aimed to test by applying programming knowledge and skills. This assignment consists of a data set of “Data Science salaries” in csv format, and it is needed to be prepared, explored and its initial analysis to be done. It is required to write python programs to solve problems occurring in various stages of data understanding, preparation, initial analysis, and exploration. The primary objective of this coursework is to prepare data for further data analysis and data mining.

1. Data Understanding:

Q. To understand what your data resources are and the characteristics of those resources. Write down your findings.

Ans: The objective of data understanding phase of data mining is to understand the dataset’s structure, contents, and quality of the data. The data provided shows columns, which are as follows: -

- Work Year: This column is for data spans across the year 2023.
- Experience Level: We have several experience levels represented, including Software Engineer (SE), Machine Learning Engineer (MI), and Engineer (EN).
- Employment Type: This column holds employment type. The majority are Full-Time (FT) positions.
- Job Tittle: Various job titles include Data Scientist, ML Engineer, Applied Scientist, Data Analyst, Research Engineer, Analytics Engineer, Business Intelligence Engineer, Data Strategist, Data Engineer, Computer Vision Engineer, Data Quality Analyst, and Compliance Data Analyst.
- Salary and Currency: Salaries are provided in different currencies including EUR, USD, and INR.
- Employee Residence: Employees are based in different countries including Spain (ES), the United States (US), Canada (CA), Germany (DE), Nigeria (NG), and India (IN).
- Remote Work Ratio: Some employees have 100% remote work while others do not.
- Company Location: Companies are in various countries.
- Company Size: Companies of different sizes are represented.

The dataset offers insightful information about 2023 employment trends. First of all, it draws attention to the stark differences in pay across various job types and geographical areas, with data scientists in the US earning noticeably more than their international colleagues. This highlights the significance of geographic considerations in determining salary by indicating a substantial correlation between employment location and remuneration. Furthermore, the abundance of jobs for data scientists and engineers suggests that there is a strong need for experts in these domains, demonstrating the ongoing significance of data-related occupations across a variety of businesses.

Second, with most jobs allowing remote work, the data clearly demonstrates a trend in favor of remote work choices. The global COVID-19 epidemic, which hastened the adoption of remote work arrangements, is perhaps one factor influencing this development. Even though there seems to be a relationship between experience level and pay, more research is necessary to completely comprehend this relationship. Furthermore, it's been said that larger businesses might provide more attractive benefits and pay, albeit this might differ depending on the region and industry.

These are just initial observations, and further analysis such as statistical modeling or correlation analysis could provide more insights into the factors influencing salaries and job trends in this dataset.

2. Data Preparation:

2.1. Loading data into pandas DataFrame.

Q) Write a python program to load data into pandas DataFrame.

Ans: Program Screenshot:

```
#2.Data Preparation
#2.i) Write a python program to load data into pandas DataFrame
#Importing Pandas Library.
import pandas as pd
try:
    # Specifying the file path.
    path = 'DataScienceSalaries.csv'
    #Data loading into Pandas DataFrame.
    df = pd.read_csv(path)
    print(df)
except Exception as e:
    #error message
    print("Error Occured!!")
```

Figure 1:Loading data into Pandas.

As shown in above figure, the data in CSV format is loaded into pandas DataFrame. It uses the **'read_csv'** function from the Pandas library to read the data from Pandas library and the data is stored into DataFrame **'df'**. If any error occurred, it prints an error message.

Results:

```
      work_year experience_level employment_type                 job_title  \
0          2023               SE              FT  Principal Data Scientist
1          2023               MI              CT             ML Engineer
2          2023               MI              CT             ML Engineer
3          2023               SE              FT          Data Scientist
4          2023               SE              FT          Data Scientist
...         ...              ...             ...                     ...
3750       2020               SE              FT          Data Scientist
3751       2021               MI              FT  Principal Data Scientist
3752       2020               EN              FT          Data Scientist
3753       2020               EN              CT  Business Data Analyst
3754       2021               SE              FT   Data Science Manager

       salary salary_currency  salary_in_usd employee_residence  remote_ratio  \
0       80000             EUR          85847                ES           100
1       30000             USD          30000                US           100
2       25500             USD          25500                US           100
3      175000             USD         175000                CA           100
4      120000             USD         120000                CA           100
...       ...             ...            ...               ...           ...
3750   412000             USD         412000                US           100
3751   151000             USD         151000                US           100
3752   105000             USD         105000                US           100
3753   100000             USD         100000                US           100
3754  7000000             INR          94665                IN            50

     company_location company_size
0                  ES            L
1                  US            S
2                  US            S
3                  CA            M
```

Figure 2: Loaded dataFrame.

The above image gives the output of code which displays the success of load of csv file into a dataFrame (Datacamp, 2023).

2.2. Removing columns

Q) Write a python program to remove unnecessary columns i.e. salary and salary currency.

Ans: Program coding:

```
#2 ii)  Write a python program to remove unnecessary columns i.e., salary and salary currency.

# Removing 'salary' and 'salary currency' columns using drop() function.
df= df.drop(columns= ['salary','salary_currency'])
print(df)
```

Figure 3: Removing column code.

The above python program removes the columns **'salary'** and '**salary_currency'** from the DataFrame using '**drop()**' function. After removing, it prints the updated DataFrame using '**Print()**' function.

Result:

```
      work_year experience_level employment_type                 job_title  \
0          2023               SE              FT  Principal Data Scientist
1          2023               MI              CT             ML Engineer
2          2023               MI              CT             ML Engineer
3          2023               SE              FT           Data Scientist
4          2023               SE              FT           Data Scientist
...         ...              ...             ...                      ...
3750       2020               SE              FT           Data Scientist
3751       2021               MI              FT  Principal Data Scientist
3752       2020               EN              FT           Data Scientist
3753       2020               EN              CT    Business Data Analyst
3754       2021               SE              FT     Data Science Manager

      salary_in_usd employee_residence  remote_ratio company_location  \
0             85847                ES           100               ES
1             30000                US           100               US
2             25500                US           100               US
3            175000                CA           100               CA
4            120000                CA           100               CA
...             ...               ...           ...              ...
3750         412000                US           100               US
3751         151000                US           100               US
3752         105000                US           100               US
3753         100000                US           100               US
3754          94665                IN            50               IN
```

Figure 4: Columns removed.

This is updated DataFrame after removal of those columns, there are no 'salary' and 'salary_currency' (Queirozf, 2020).

2.3. Remove NaN missing values.

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

Ans: Program Screenshot:

```
# 2. iii. Write a python program to remove the NaN missing values from updated dataframe.

# At first, Checking for NaN values in each columns
Null_values = df.isna()# NaN values column-wise.
# Printing NaN values
print("NaN values in each column:")
print(Null_values)
```

Figure 5:Checking NaN values.

The above program checks the Null (NaN) values in each column. Alternatively, some functions like **isnull(), isnan(), notnull()** , etc could also be used.

Results:

```
NaN values in each column:
      work_year  experience_level  employment_type  job_title  salary_in_usd  \
0         False             False            False      False          False
1         False             False            False      False          False
2         False             False            False      False          False
3         False             False            False      False          False
4         False             False            False      False          False
...         ...               ...              ...        ...            ...
3750      False             False            False      False          False
3751      False             False            False      False          False
3752      False             False            False      False          False
3753      False             False            False      False          False
3754      False             False            False      False          False

      employee_residence  remote_ratio  company_location  company_size
0                  False         False             False         False
1                  False         False             False         False
2                  False         False             False         False
3                  False         False             False         False
4                  False         False             False         False
...                  ...           ...               ...           ...
3750               False         False             False         False
3751               False         False             False         False
3752               False         False             False         False
3753               False         False             False         False
3754               False         False             False         False

[3755 rows x 9 columns]
```

Figure 6: No NaN values found.

The above screenshot of results shows no any column with null/NaN values as the results says '**False**' on the implication of **'isna()'** function (Queirozf, 2020).

```
#There are no NaN (Null)  values for columns so df remains same.
new_df = df.dropna()
new_df
```

	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

Figure 7: Use of dropna function.

Since there is no NaN values, the **dropna()** function applied on DataFrame, **df** gives no any update to new DataFrame, **new_df.**

2.4. Checking duplicate values.

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

Ans: Program Screenshot:

```
# 2. iv.Write a python program to check duplicates value in the dataframe.
duplicate_values = df.duplicated() #it finds duplicate rows.
print(duplicate_values)
```

Figure 8 : Checking duplicate values.

The above program checks the duplicate values in data frame using **'duplicated()'** function on df.

Results:

```
0          False
1          False
2          False
3          False
4          False
           ...
3750       False
3751       False
3752       False
3753       False
3754       False
Length: 3755, dtype: bool
```

Figure 9: No duplicate value found.

2.5. Displaying unique values from columns.

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

Ans: Program Screenshot:

```
#2. v. Write a python program to see the unique values from all the columns in the dataframe.
#Print unique values in each column
for column in df.columns:
    unique_values = df[column].unique()  # Get unique values for the current column
    print("Unique Values:")
    print("For **" + column + "** Column:")
    print()
    print(unique_values)
    print("-------------------------------------------------------------------------------")
    print()
```

Figure 10: Program to see unique values.

The above program iterates through each column in DataFrame, **'df'** and prints the unique values from each column. The 'for' loop iterates each column using **'df.columns'** and for each column, **'df[column].unique()'** function retrives the unique values. The value is printed using **'print()'** function.

Results:

The screenshots below gives the unique values presented in the each column of the DataFrame (Queirozf, 2020).

```
Unique Values:
For **work_year** Column:

[2023 2022 2020 2021]
-------------------------------------------------------------------------------

Unique Values:
For **experience_level** Column:

['SE' 'MI' 'EN' 'EX']
-------------------------------------------------------------------------------

Unique Values:
For **employment_type** Column:

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

Figure 11:Unique values output-1

```
Unique Values:
For **job_title** Column:

['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 12: Unique values output-2

```
 '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']
--------------------------------------------------------------------------------

Unique Values:
For **salary_in_usd** Column:

[ 85847  30000  25500 ...  28369 412000  94665]
--------------------------------------------------------------------------------

Unique Values:
For **employee_residence** Column:

['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']
--------------------------------------------------------------------------------
```

Figure 13: Unique values output-3

```
Unique Values:
For **remote_ratio** Column:

[100   0  50]
--------------------------------------------------------------------------------

Unique Values:
For **company_location** Column:

['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']
--------------------------------------------------------------------------------

Unique Values:
For **company_size** Column:

['L' 'S' 'M']
--------------------------------------------------------------------------------
```

Figure 14: Unique values output-4

2.6. Renaming:

Q) Rename the experience level columns as below.

SE – Senior Level/Expert

MI – Medium Level/Intermediate

EN – Entry Level

EX – Executive Level

Ans:

Program Screenshot:

```python
#2. vi. Renaming experience_level columns data:

#making a dictionary with old names (SE, MI, EN and EX) are keys and corresponding names are values
level_rename = {
    'SE': 'Senior Level/Expert',
    'MI': 'Medium Level/Intermediate',
    'EN': 'Entry Level',
    'EX': 'Executive Level'
}
#replacing the 'experience_level' columns data with corresponding values of the dictionary.
df['experience_level'] = df['experience_level'].replace(level_rename)
print(df)
```

Figure 15: Rename column data.

The above program renames the values in the ‘experience_level’ column of DataFrame, ‘df’ using the **‘level_rename’**. A dictionary is created, where the keys are the old values in the ‘experience_level’ column and the values are their corresponding new names.

The **‘replace()’** method is used on the ‘experience_level’ column. It replaces the older values with corresponding new names based on dictionary defined earlier.

Result:

```
df.columns
df # showing the changed names
```

	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

Figure 16: Name changed.

The above screenshot shows that the ‘experience_level’ column is updated. The older names are replaced with the new names that were in the dictionary (Queirozf, 2020).

3. Data Analysis:

3.1. Calculating sum, mean, S.D, skewness, and Kurtosis.

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

Ans:

Program Screenshots:

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

#choosing a variable/column, 'salary_in_usd'  for statistics and replacing with a variable.
salaries = 'salary_in_usd'

#Using Pandas' describe() method for showing the statistics of salaries.
salary_stats = df[salaries].describe()
print("The summary statistics for 'salary_in_usd' column :")
print(salary_stats)
print()

#a) for salary sum.
salary_sum = df[salaries].sum()
print("The total salary in USD is :")
print(salary_sum)
print()

#b) for mean/average of the salary.
avg_salary = df[salaries].mean()
print("The Average salary in USD is :")
print(avg_salary)
print()
```

Figure 17: Statistical Analysis program-1

```
#c) for standard deviation.
std_salary = df[salaries].std()
print("The S.D in salary is :")
print(std_salary)
print()

#d) Calculating skewness and kurtosis'
skewness = df[salaries].skew()
kurtosis = df[salaries].kurt()

#printing skewness and kurtosis
print("Skewness in salary is:")
print(skewness)
print()
print("Kurtosis in salary is:")
print(kurtosis)
```

Figure 18: Statistical Analysis program-2

The above screenshots compute the statistics (sum, mean, S.D, skewness, and kurtosis) for **'salary_in_usd'** in a DataFrame, **'df'**. The summary of Statistics is computed via 'describe()' method. Similarly, **'sum()', 'mean()', 'std()', 'skew()' and 'kurt()'** functions are used to calculate the statistical values.

Results:

```
The summary statistics for 'salary_in_usd' column :
count      3755.000000
mean     137570.389880
std       63055.625278
min        5132.000000
25%       95000.000000
50%      135000.000000
75%      175000.000000
max      450000.000000
Name: salary_in_usd, dtype: float64

The total salary in USD is :
516576814

The Average salary in USD is :
137570.38988015978

The S.D in salary is :
63055.625278224084

Skewness in salary is:
0.5364011659712974

Kurtosis in salary is:
0.8340064594833612
```

Figure 19: Result of statistical Analysis.

3.2. Correlation:

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

Program Screenshot

```
# 3. ii. Write a Python program to calculate and show correlation of all variables.
#At first, choosing only the numeric variables i.e columns in new dataframe, df2
df2 = df.select_dtypes(include=['number'])
print("Columns with Numerical Values: ")
print()
print(df2)
print()
#Now, calculating the correlation of these variables(columns)
correlation = df2.corr()
print("The correlation matrix of numeric columns: ")
print("-----------------------------------------------------")
print(correlation)
```

Figure 20: Correlation code.

The above code calculates and displays the correlation matrix of all numeric variables in the DataFrame, df.

Summary of what each part of program does:

- **df2 = df.select_dtypes(include=['number'])** :- This line of code selects the column with numeric data types and the result is assigned to a new DataFrame, **'df2'**.
- **Correlation = df2.corr(): -** This line of code calculates the matrix for all numeric variables in the DataFrame, 'df2'.
- **Print()** :- This method is used to print the results.

Result:

```
Columns with Numerical Values:

      work_year  salary_in_usd  remote_ratio
0          2023          85847           100
1          2023          30000           100
2          2023          25500           100
3          2023         175000           100
4          2023         120000           100
...         ...            ...           ...
3750       2020         412000           100
3751       2021         151000           100
3752       2020         105000           100
3753       2020         100000           100
3754       2021          94665            50

[3755 rows x 3 columns]

The correlation matrix of numeric columns:
-----------------------------------------------------
                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 21: Correlation results

The above output provides display of linear relationships between pairs of variables, with correlation coefficients ranging from -1 to 1. A closely related pair of variables with correlation coefficients close to 1 indicates strong positive relation and vice-versa. The value close to 0 indicates the no linear correlation (Geeksforgeeks, 2024).

4. Data Exploration:

4.1. Write a python program to find out top 15 jobs. Make a bar graph.

Program Screenshot:

```
#4. DATA exploration:
#4.i.Write a python program to find out top 15 jobs.Make a bar graph.

#Importing NumPy and Matplotlib libraries
import numpy as np
import matplotlib.pyplot as plt

# Group by job_title and count occurrences
job_counts = df['job_title'].value_counts().head(15)

# Sort by frequency and take the top 15
top_jobs = job_counts.nlargest(15)

#x = jobs
#y = frequency (repetition of the jobs)

# Plotting x and y
plt.figure(figsize=(12, 8))
plt.bar(top_jobs.index, top_jobs, label="Top Jobs", color="green", alpha=0.5)
plt.title('Top 15 Jobs') #Title
plt.xlabel('Job Title') #X-axis label
plt.ylabel('Frequency') #y-axis label
plt.xticks(rotation=40, ha='right')
plt.legend()
plt.tight_layout() #adjusting layout to prevent overlapping.
plt.show()
```

Figure 22: Program to find top 15 jobs

This python program uses NumPy and Matplotlib to create a bar graph displaying the top 15 most common job titles.

At first, NumPy and Matplotlib libraries are imported for data manipulation and visualization.

- **job_counts = df['job_title'].value_counts().head(15):-** This line counts the occurrences of each job title in the DataFrame and selects the top 15 using **head(15)** method.

- **top_jobs = job_counts.nlargest(15)**: - This line of code selects top 15 most frequent job titles from the calculated **'job_counts'** variables.
- The plotting part of code helps in plotting a bar graph with top 15 job titles on the x-axis and their corresponding frequency on the y-axis. X-axis is roated for better readability and prevents the overlapping (Analytics Vidhya, 2024).

Result:

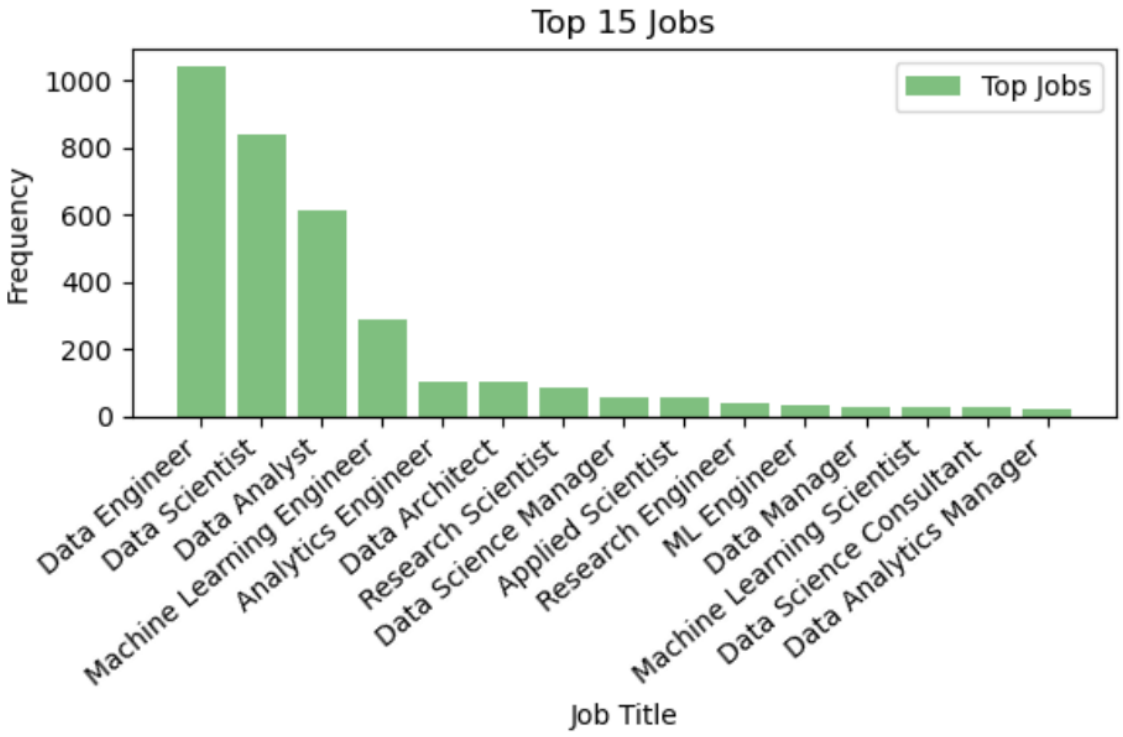


Figure 23: A bar graph showing top 15 jobs.

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

Program Screenshot:

```
#4 ii. Which job has the highest salaries? Illustrate with bar graph

#Calculating average salary by each job title
Avg_salary = df.groupby('job_title')['salary_in_usd'].mean()

#Finding top 10 jobs with highest average salary.
highest_salary = Avg_salary.nlargest(10)

# Plotting the top 10 jobs with the highest average salaries
plt.figure(figsize=(10, 6))
plt.bar(highest_salary.index, highest_salary.values, color="blue", alpha=0.3)
plt.title('Top 10 Jobs with Highest Average Salaries')
plt.xlabel('Job Title')
plt.ylabel('Highest Average Salary (USD)')
plt.xticks(rotation=40, ha='right')
plt.tight_layout()
plt.show()
```

Figure 24: Program to make bar graph top salaries jobs.

The above program shows the following flow: -

- Calculating Average Salary by Job Title:
  Avg_salary = df.groupby('job_title')['salary_in_usd'].mean() :- This line of code groups the dataframe by the **'job_title'** and calculates the mean of the **'salary_in_usd'** column for each group and results assigned to **Avg_salary** variable.

- Finding top 10 jobs with highest Average Salary:
  highest_salary = Avg_salary.nlargest(10) :- This method selects top 10 jobs with highest average salaries from **'Avg_salary'** and sorting descending order.

- Plotting Bar Graph:
  The **'plt.bar'** is used to plot the bar graph with job tiles on x-axis (**'highest_salary.index')** and their corresponding average salaries on y-axis **('highest_salary.values').**
  'Plt.xtrics()' rotates the x-axis for better readability and **'plt.tight_layout()'** adjusts the layout to prevent overlapping. Finally, **'plt.show()'** displays the plot (Analytics Vidhya, 2024).

Result:

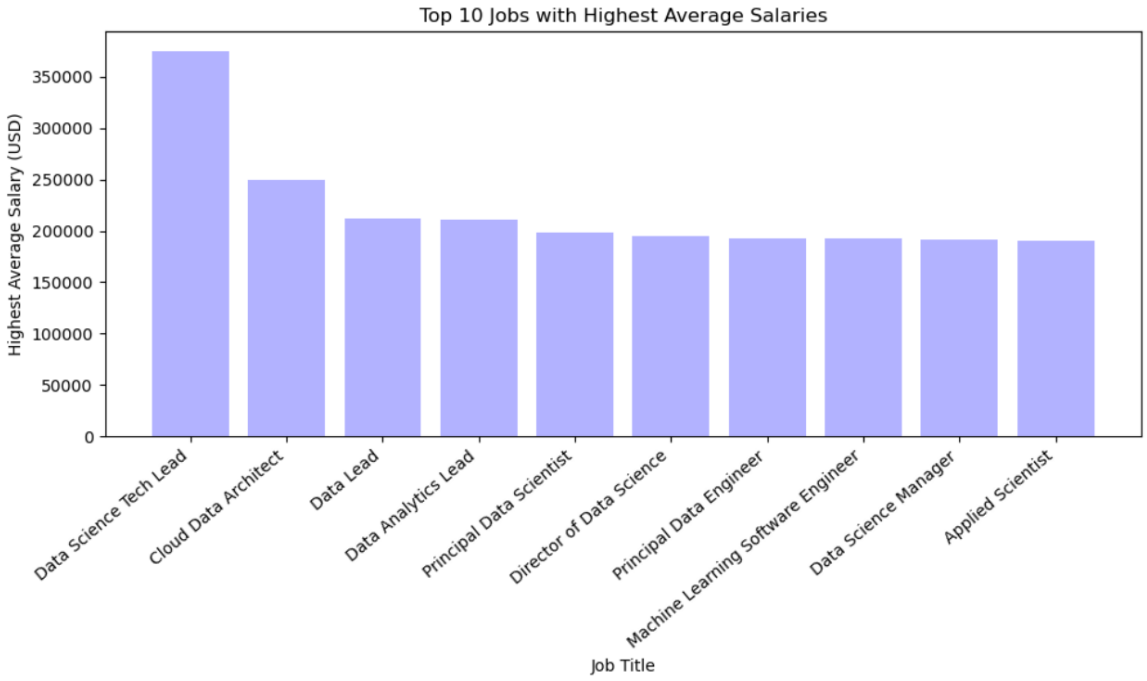


Figure 25: Bar graph showing highest salary jobs.

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

Program Screenshot:

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

# Group by experience level and calculate the average salary for each level
Avg_salary2 = df.groupby('experience_level')['salary_in_usd'].mean()

# Plotting salaries based on experience level
plt.figure(figsize=(12, 8))
plt.bar(Avg_salary2.index, Avg_salary2.values, color="skyblue", alpha=0.8)
plt.title('Salaries Based on Experience Level') # Title
plt.xlabel('Experience Level') # X-axis label
plt.ylabel('Average Salary (USD)') # Y-axis label
plt.xticks(rotation=40, ha='right')
plt.tight_layout() # Adjusting layout to prevent overlapping
plt.show()
```

Figure 26: Code to find salaries based on experience level.

The above code finds the salaries based on experience level and illustrates it through bar graph.

The code flows as following:

• Calculating Average Salary by Experience level.

. **Avg_salary2 = df.groupby('experience_level')['salary_in_usd'].mean() :-** This line of code groups the '**experience_level'** column and computes the mean of the **'salary_in_usd'** column of each group.

• Plotting Bar Graph:

'plt.bar()' is used to plot the bar graph with experience levels on the x-axis **(Avg_salary2.index)** and their corresponding average salaries on the y-axis **(Avg_salary2.values).**

'Plt.xtrics()' rotates the x-axis for better readability and **'plt.tight_layout()'** adjusts the layout to prevent overlapping. Finally**, 'plt.show()'** displays the plot (Analytics Vidhya, 2024).

Result:

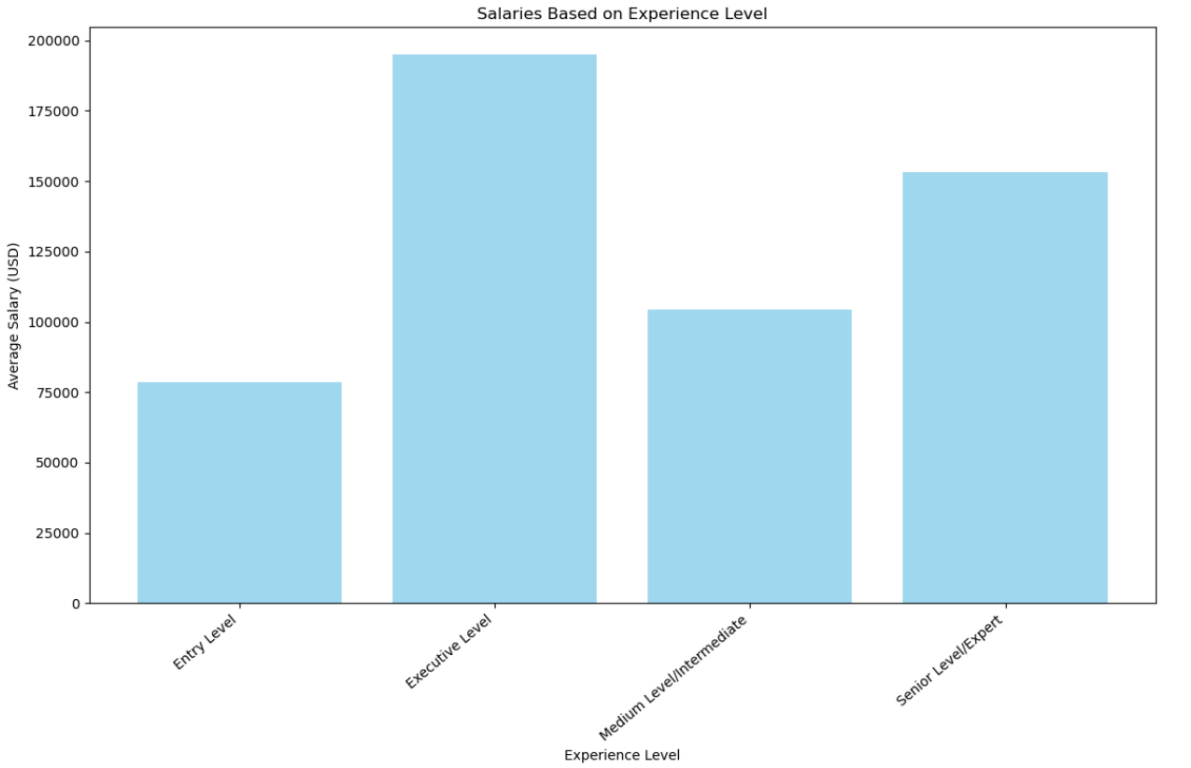


Figure 27: Bar graph showing salaries based on experience level.

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

Program Screenshot: (for salary_in_usd column)

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

# part 1: for 'salary_in_usd' column
#Extracting the chosen column for analysis and defining a variable to store 'salary_in_usd'
Salary_analysis = df['salary_in_usd']

#plotting histogram of the Salary_analysis.
plt.figure(figsize=(12, 8))
plt.hist(Salary_analysis, bins=8, color='red',alpha=0.6, edgecolor='black')
plt.title('Histogram of Salary in USD')
plt.xlabel('Salary in USD')
plt.ylabel('Frequency')

# Plot box plot of the Salary_analysis to identify outliers
plt.figure(figsize=(8, 6))
plt.boxplot(Salary_analysis)
plt.title('Box Plot of Salary in USD (Outliers)')
plt.xlabel('Salary in USD')

# Showing scatterness of the Salary_analysis
plt.figure(figsize=(12, 10))
plt.scatter(range(len(Salary_analysis)), Salary_analysis, color='green', alpha=0.5)
plt.title('Scatter Plot of Salary in USD')
plt.xlabel('Index')
plt.ylabel('Salary in USD')
# Showing the plots
plt.tight_layout()
plt.show()
```

Figure 28: part 1 code for (salary_in_usd) column.

The above program analyzes the ‘slary_in_usd’ column with histogram, box plot, and scatter plot.

Results:

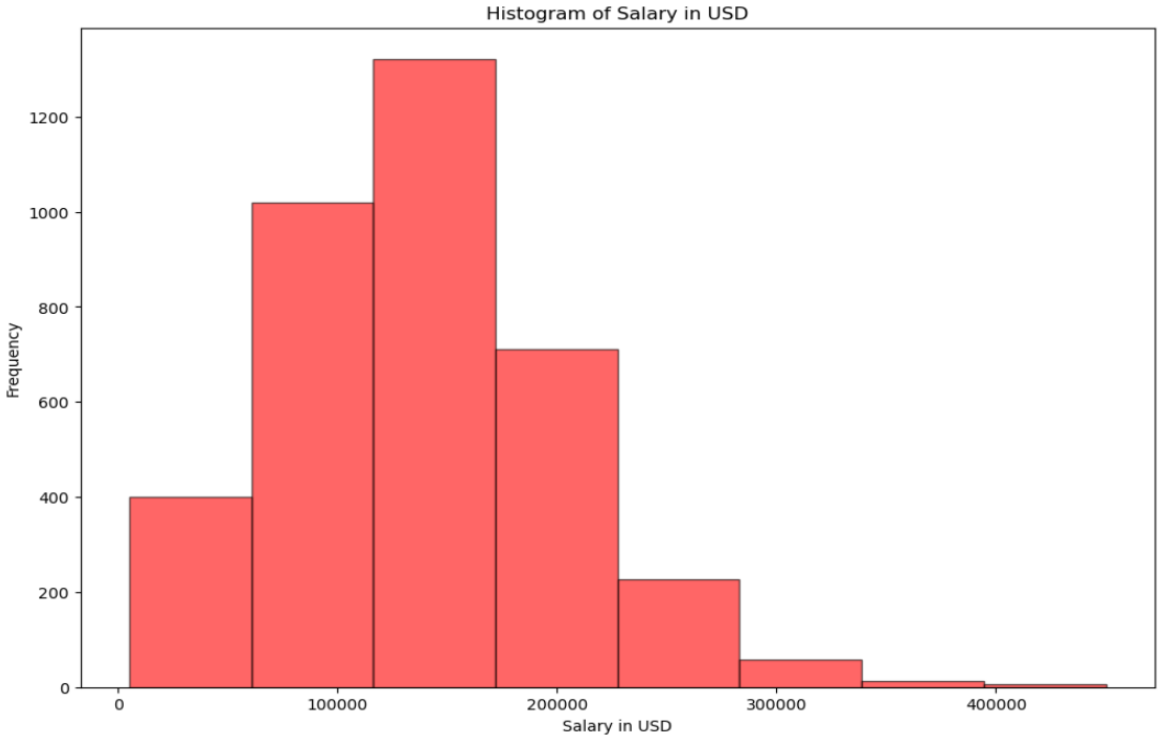


Figure 29: Histogram(1).

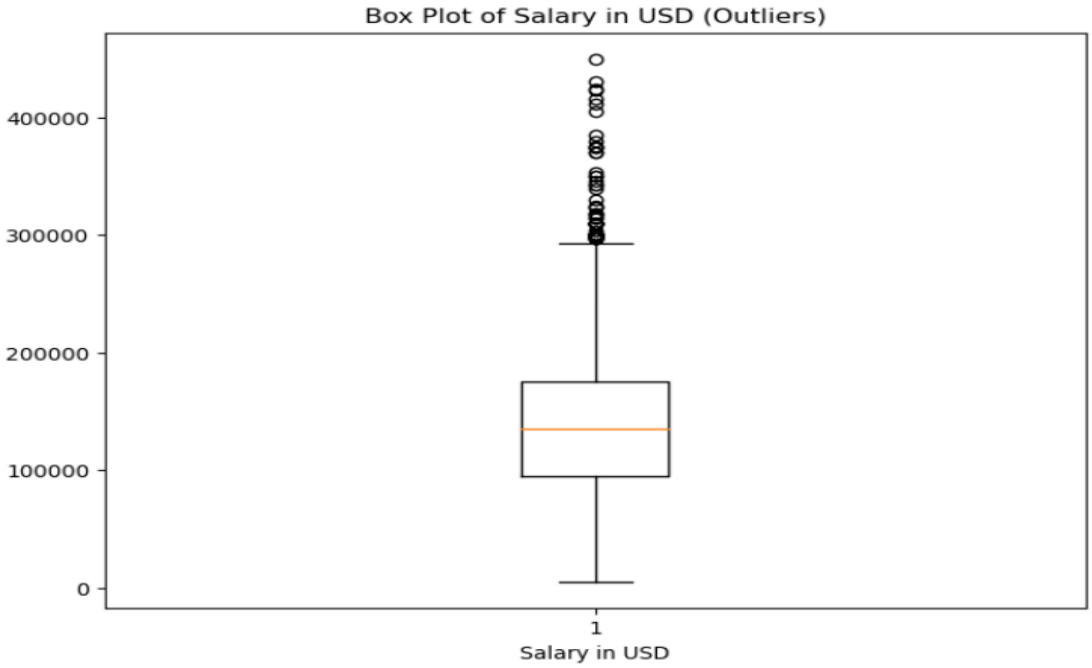


Figure 30: Box plot-1.

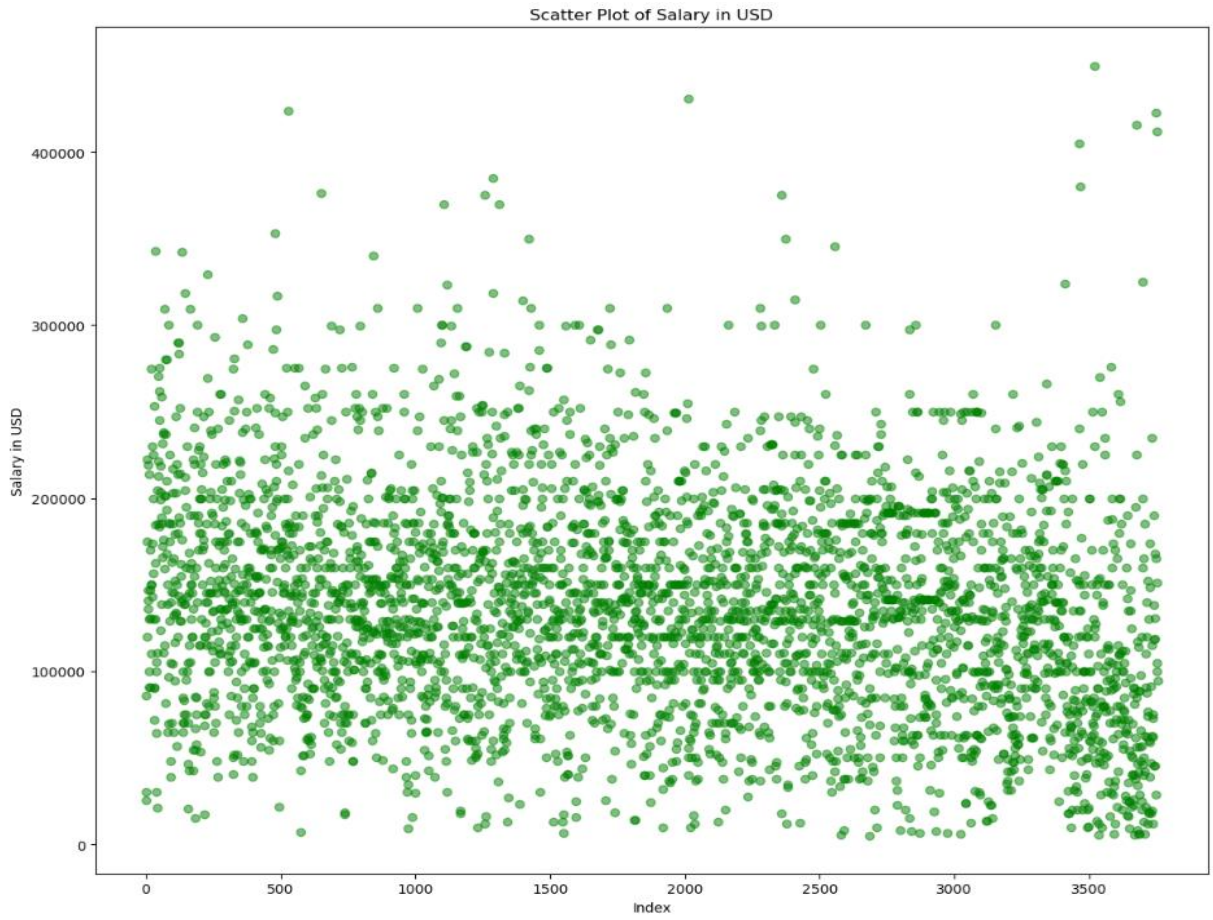


Figure 31: Scatter diagram.

Program Screenshot: (for work_year column)

```
#4.iv
# part 2: for 'work_year' column
# Extracting the chosen column for analysis and defining a variable to store 'work_year'
work_year_analysis = df['work_year']

# Plotting histogram of the work_year_analysis.
plt.figure(figsize=(8, 4))
plt.hist(work_year_analysis, bins=8, color='orange', alpha=0.6, edgecolor='blue')
plt.title('Histogram of Work Year')
plt.xlabel('Work Year')
plt.ylabel('Frequency')

# Plotting box plot of the work_year_analysis to identify outliers
plt.figure(figsize=(8, 4))
plt.boxplot(work_year_analysis)
plt.title('Box Plot of Work Year (Outliers)')
plt.xlabel('Work Year')

# Plotting scatter plot of the work_year_analysis
plt.figure(figsize=(8, 4))
plt.scatter(work_year_analysis.index, work_year_analysis, color='blue', alpha=0.5)
plt.title('Scatter Plot of Work Year')
plt.xlabel('Index')
plt.ylabel('Work Year')

# Show the plots
plt.tight_layout()
plt.show()
```

Figure 32: part 2 code for (work_year) column.

The above program analyzes the ‘work_year’ column with histogram, box plot, and scatter plot (Analytics Vidhya, 2024).

Results:

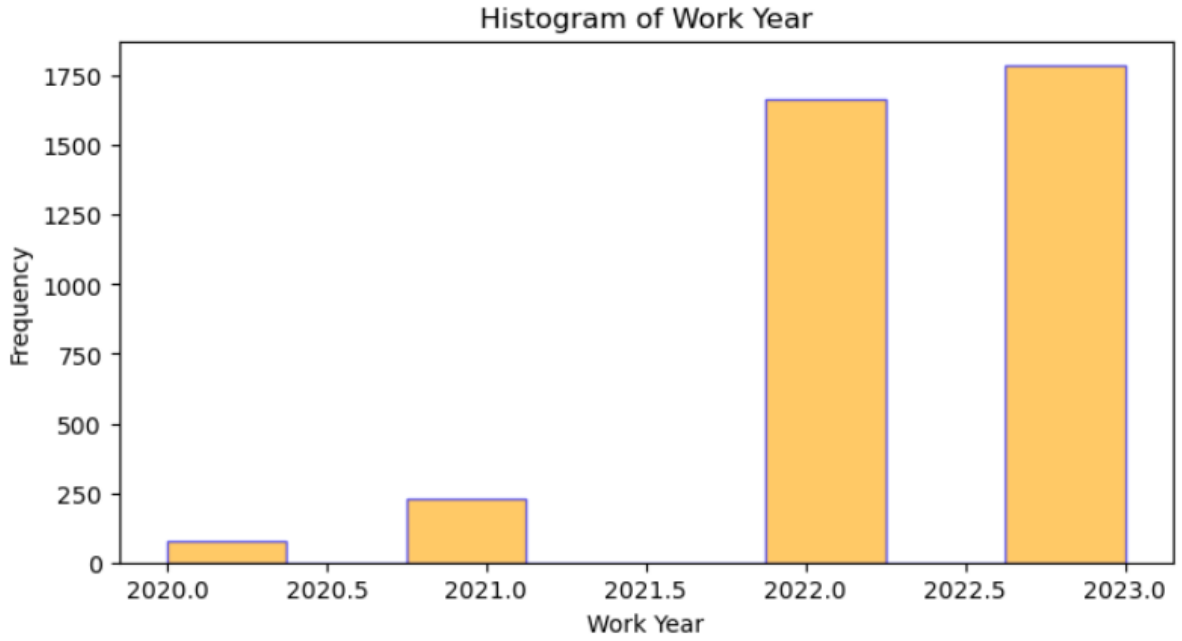


Figure 33: Histogram-2

Figure 34: Box plot-2

Figure 35: Scatter Diagram-2

5. Conclusion:

To sum up, the coursework assignment serves as a comprehensive evaluation. It emphasizes on python programming skills in data analysis, and problem-solving skills. It is focused on “data science salary” analysis. The data understanding, preparation, analysis and exploration from the CSV format of the **“Data science Salaries”.**
The primary objective of this coursework was to prepare data for further data analysis and data mining.

6. References

Analytics Vidhya, 2024. *Data exploration.* [Online]
Available at: https://www.analyticsvidhya.com/blog/2015/04/comprehensive-guide-data-exploration-sas-using-python-numpy-scipy-matplotlib-pandas/
[Accessed 05 2024].
Datacamp, 2023. *loading data into Pandas.* [Online]
Available at: https://www.datacamp.com/tutorial/pandas-read-csv
[Accessed 05 2024].
Geeksforgeeks, 2024. *Data Analysis.* [Online]
Available at: https://www.geeksforgeeks.org/data-analysis-with-python/
[Accessed 05 2024].
Queirozf, 2020. *Column Operations.* [Online]
Available at: https://queirozf.com/entries/pandas-dataframe-examples-column-operations
[Accessed 05 2024].