# CC5067NI-Smart Data Discovery

## 60% Individual Coursework

## 2023-24 Spring

**Student Name: Rohit Raut**

**London Met ID: 22068148**

**College ID:** NP01CP4A220498

**Assignment Due Date: Monday, May 13, 2024**

**Assignment Submission Date: Monday, May 13, 2024**

**Word Count: 1859**

*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

## Figure of Figures: -

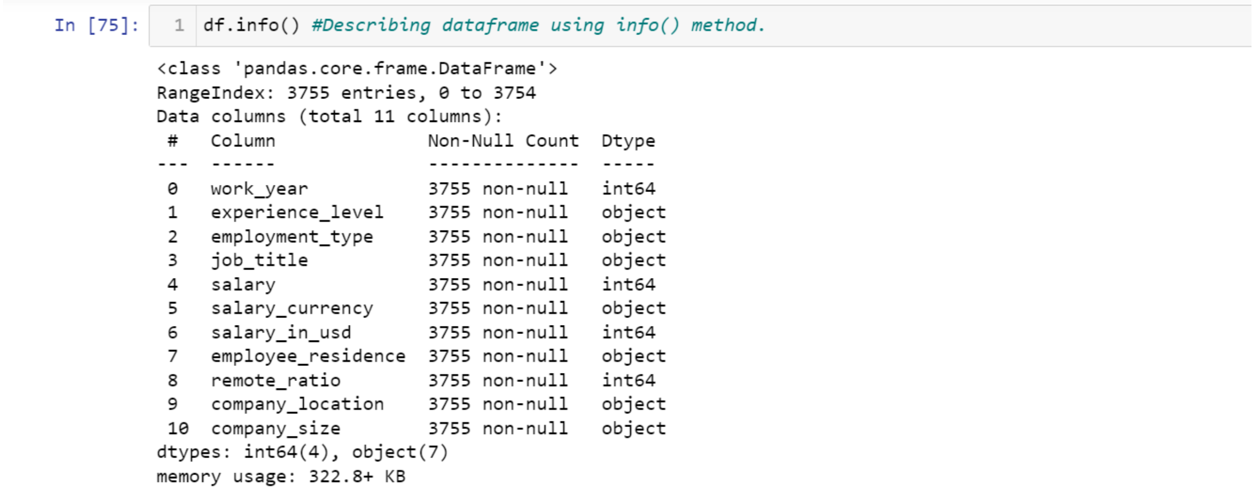
## *Table of Table*

# 1. Data Understanding

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

This coursework is about understanding the dataset which include the salary of Data Scientist in different specific field. The provided dataset is in csv file which name is DataScienceSalaries.csv. This csv extension is used for this coursework because, excel may not be available in every device for data collection or for storing data. Moreover, this extension is widely used because it can be easily opened and can be manipulated the data inside it. Despite simplicity, csv file are very effective for storing and transferring large dataset into small file size as compared to other formats.

In this csv file there are total eleven columns which include work_year, experience_level, employment_type, job_title, salary,salary_currency, salary_in_usd, employee_residence, remote_ratio, company_location, and company_size. The salaries of each employment_type is vary upon different experiences_level. This below image shows all the columns present in dataset which include columns name its datatype and counts.



```
In [75]:  df.info() #Describing dataframe using info() method.

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

Figure 1: Describing the Data Frame using info() method,

| S.no | Column Name | Description | Data Type |
|---|---|---|---|
| 1. | work_year | This column in CSV file shows the number of years of employment for every single individual, giving essential details about the duration of their stay with the company and the amount of experiences gained during that period of time. | Int64 |
| 2. | **experience_level** | The "experience_level" column in csv file classifies the employee by level of professional expertise or time served in the organization; it provides information on the distribution of skills and how they recruit. | Object |
| 3. | **employment_type** | The "experience_level" column contains information about the distribution of skills and approaches to hire, assessing individuals by their period of experience or their level of knowledge in their field. | Object |
| 4. | **job_title** | The "job_title" column of the data set tells about the particular responsibilities or jobs that the staff members have within the company, expounding the great variance of the job structures that is present in the company. | Object |
| 5. | **salary** | The currency used by this value for the salary amount is mentioned in the column "salary_currency." This column also depicts the money value in the salary information. | Int64 |
| 6. | **salary_currency** | The column "salary_currency" determines the currency that is in use for the salary values, while the display of currency values inside salary data is signified. | Object |
| 7. | **salary_in_usd** | The "salary_in_usd" column represents the values for salary converted into US dollars. This conversion enables easier comparison of salary levels on a worldwide scale. | Int64 |
| 8. | **employee_residence** | The "employee_residence" column provides employees’ place of work,<br>which include geographic separation over the world. | Object |

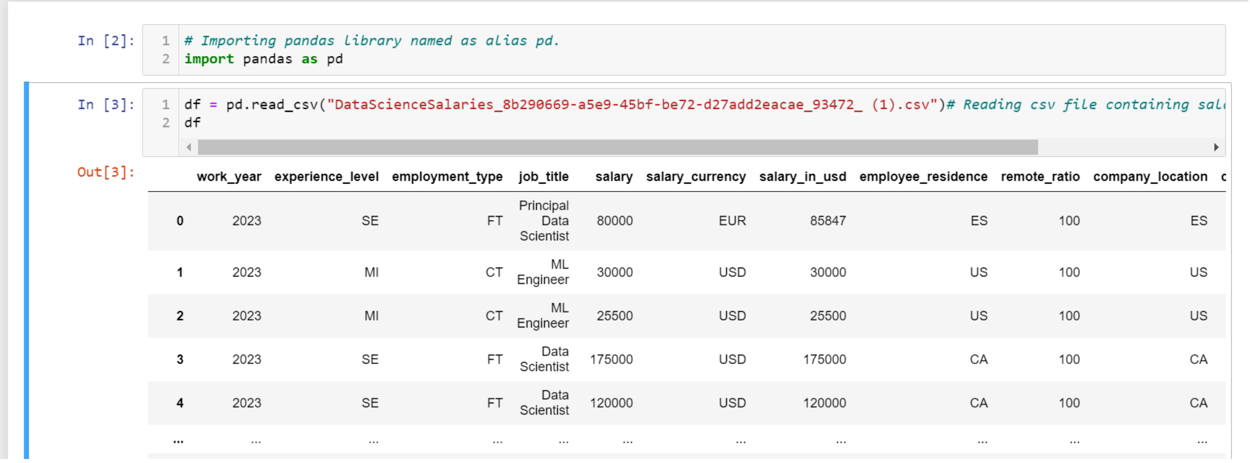
| | | | |
|---|---|---|---|
| 9. | **remote_ratio** | The "remote_ratio" column represents the ratio of remote workers within the company. | Int64 |
| 10. | **company_location** | This column describe the company location within the geographical structure. | Object |
| 11. | **company_size** | The "company_size" column in the dataset categorizes company based on their size which includes small, medium, or large. | Object |

Table 1: Description of columns available in Data Frame.

# 2. Data preparation

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

Here in this data set I first import pandas including alias named as pd. This alias is denoted for the easier referencing pandas in the code. After that, the attribute called df is created to read the csv file. The name of the csv file is "DataScienceSalaries" which contains the salaries of data science jobs position, salaries, experiences level etc and stored in it a Data frame named as df. The read_csv() is a function provided by the pandas for reading data from CSV files.



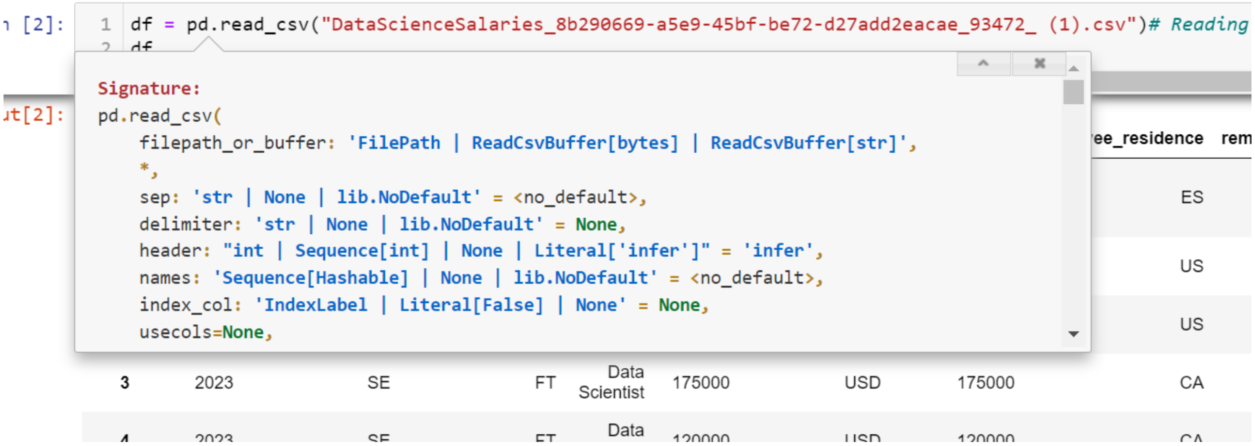
In [2]:
```
# Importing pandas library named as alias pd.
import pandas as pd
```

In [3]:
```
df = pd.read_csv("DataScienceSalaries_8b290669-a5e9-45bf-be72-d27add2eacae_93472_ (1).csv")# Reading csv file containing sal
df
```

Out[3]:

| | work_year | experience_level | employment_type | job_title | salary | salary_currency | salary_in_usd | employee_residence | remote_ratio | company_location | c |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **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 CSV file



ı [2]:
```
df = pd.read_csv("DataScienceSalaries_8b290669-a5e9-45bf-be72-d27add2eacae_93472_ (1).csv")# Reading
df
```

```
Signature:
pd.read_csv(
    filepath_or_buffer: 'FilePath | ReadCsvBuffer[bytes] | ReadCsvBuffer[str]',
    *,
    sep: 'str | None | lib.NoDefault' = <no_default>,
    delimiter: 'str | None | lib.NoDefault' = None,
    header: "int | Sequence[int] | None | Literal['infer']" = 'infer',
    names: 'Sequence[Hashable] | None | lib.NoDefault' = <no_default>,
    index_col: 'IndexLabel | Literal[False] | None' = None,
    usecols=None,
```

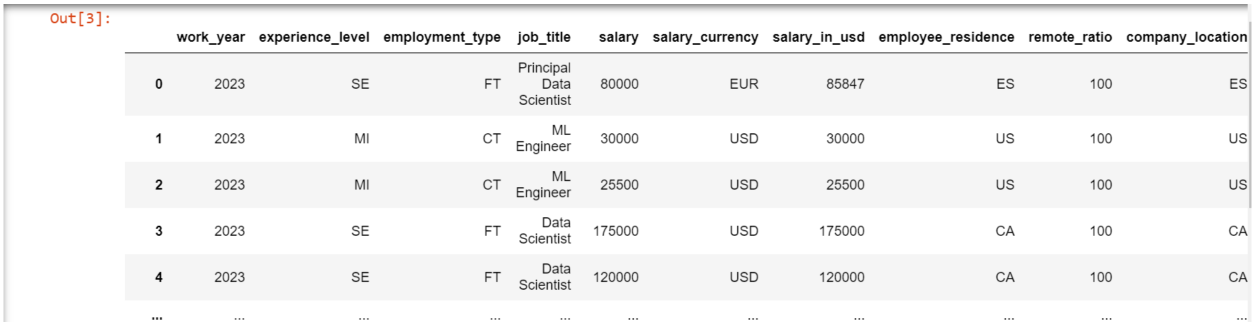
ıt[2]:

| | | | | | | | | ee_residence | rem |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | ES | |
| | | | | | | | | US | |
| | | | | | | | | US | |
| **3** | 2023 | SE | FT | Data Scientist | 175000 | USD | 175000 | CA | |
| **4** | 2023 | SE | FT | Data | 120000 | USD | 120000 | CA | |

Figure 3: Signature of read_csv function.

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

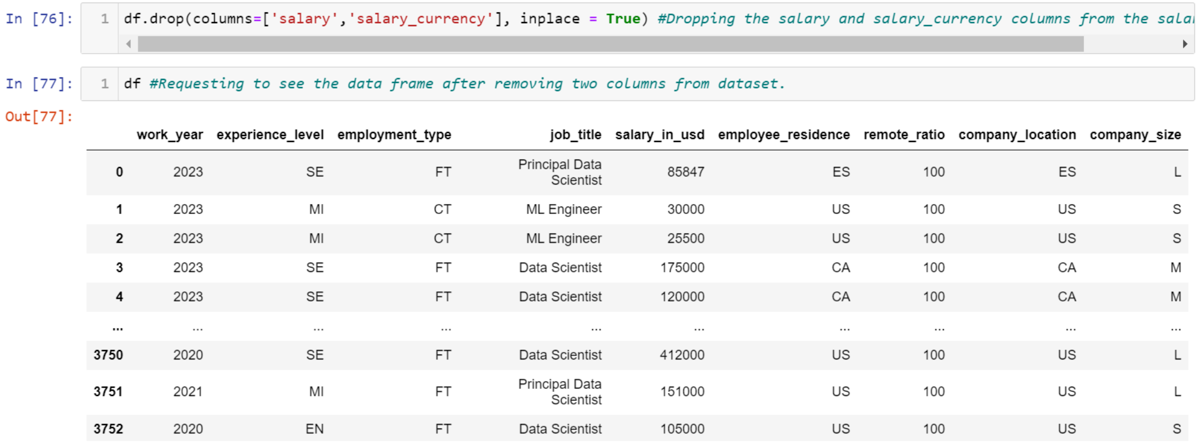
The drop () method in pandas is used to remove specific columns and rows from the data frame. Here, inside the columns the name of columns is passed to be dropped. Moreover, the inplace = ”True” parameter indicate whether to modify the provided data frame permanently or to return a data frame with the columns removed.



Out[3]:

| | work_year | experience_level | employment_type | job_title | salary | salary_currency | salary_in_usd | employee_residence | remote_ratio | company_location |
|---|---|---|---|---|---|---|---|---|---|---|
| 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 4: Data Frame before removing Columns.



```
In [76]: df.drop(columns=['salary','salary_currency'], inplace = True) #Dropping the salary and salary_currency columns from the sala

In [77]: df #Requesting to see the data frame after removing two columns from dataset.
```

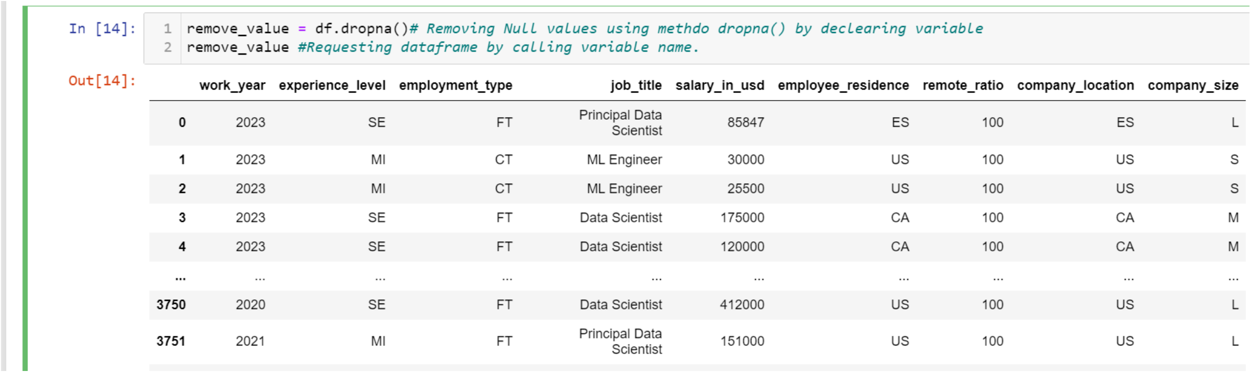
Out[77]:

| | 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 |

Figure 5: Data Frame after removing columns.

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

The dropna() method is used to remove row which containing missing values from DataFrame. Moreover, I checked the data frame using for loop and any(). The first any () method checks if there is any True values along the columns (axis= 0) and the second one checks if there are any True values within the DataFrame.

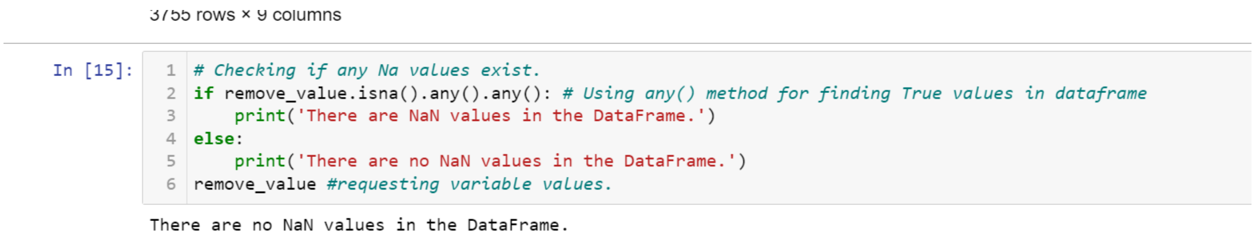


In [14]:
```
remove_value = df.dropna()# Removing Null values using methdo dropna() by declearing variable
remove_value #Requesting dataframe by calling variable name.
```

Out[14]:

| | 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 |

Figure 6: Dropping null values using drone method ().



3755 rows × 9 columns

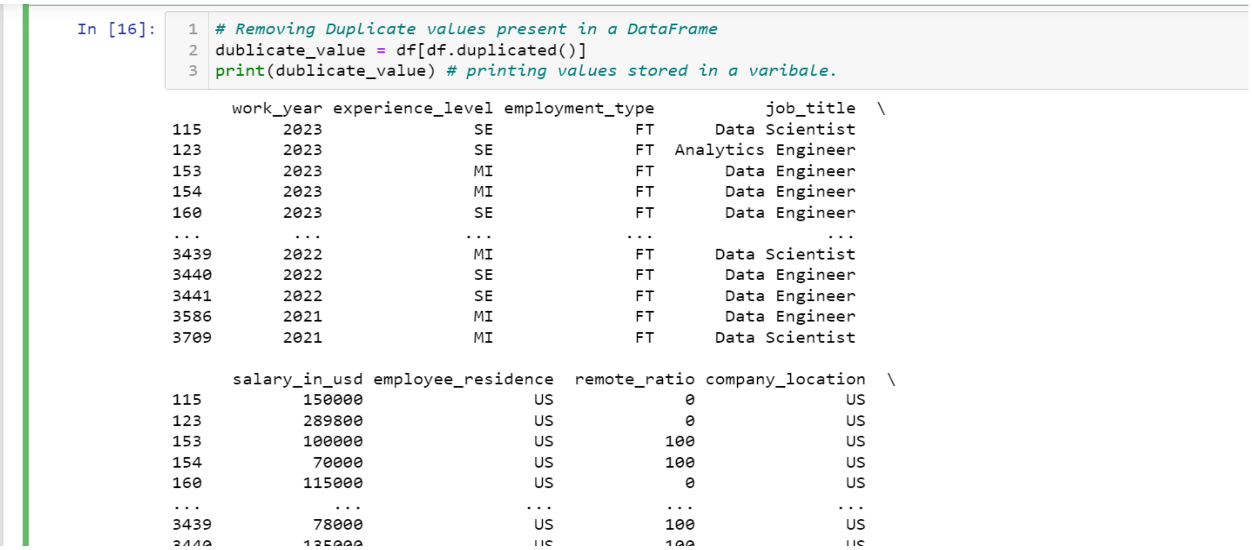
In [15]:
```
# Checking if any Na values exist.
if remove_value.isna().any().any(): # Using any() method for finding True values in dataframe
    print('There are NaN values in the DataFrame.')
else:
    print('There are no NaN values in the DataFrame.')
remove_value #requesting variable values.
```

```
There are no NaN values in the DataFrame.
```

Figure 7: Checking if Nall value exist or not.

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

This code removes duplicate rows from the data frame and store them in a variable named as dublicate_value, then it prints the values stored in that variable. The second line filters the data frame to keep only rows that are duplicate.

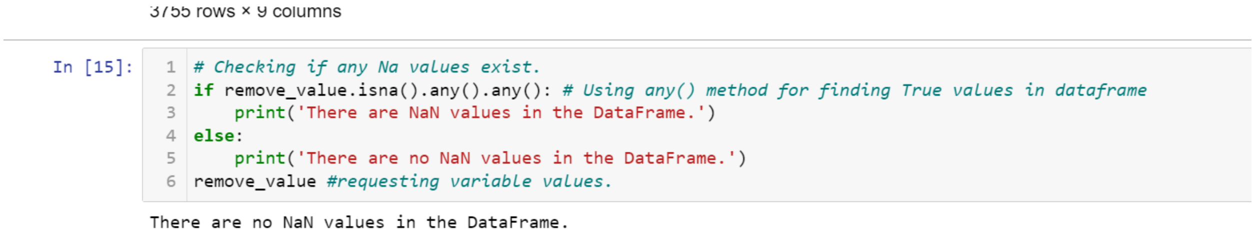


```
In [16]:
# Removing Duplicate values present in a DataFrame
dublicate_value = df[df.duplicated()]
print(dublicate_value) # printing values stored in a varibale.
```

```
      work_year experience_level employment_type          job_title  \
115        2023               SE              FT     Data Scientist
123        2023               SE              FT  Analytics Engineer
153        2023               MI              FT      Data Engineer
154        2023               MI              FT      Data Engineer
160        2023               SE              FT      Data Engineer
...         ...              ...             ...                ...
3439       2022               MI              FT     Data Scientist
3440       2022               SE              FT      Data Engineer
3441       2022               SE              FT      Data Engineer
3586       2021               MI              FT      Data Engineer
3709       2021               MI              FT     Data Scientist

      salary_in_usd employee_residence  remote_ratio company_location  \
115          150000                 US             0               US
123          289800                 US             0               US
153          100000                 US           100               US
154           70000                 US           100               US
160          115000                 US             0               US
...             ...                ...           ...              ...
3439          78000                 US           100               US
3440         135000                 US           100               US
```

Figure 8: Printing duplicate values.

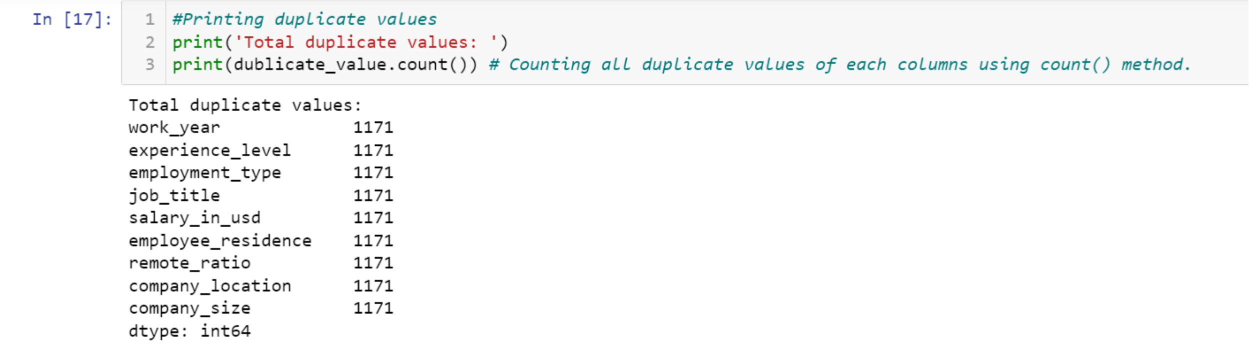


3755 rows × 9 columns

```
In [15]:
# Checking if any Na values exist.
if remove_value.isna().any().any(): # Using any() method for finding True values in dataframe
    print('There are NaN values in the DataFrame.')
else:
    print('There are no NaN values in the DataFrame.')
remove_value #requesting variable values.
```

```
There are no NaN values in the DataFrame.
```

*Figure 9 Program for checking Nan values.*



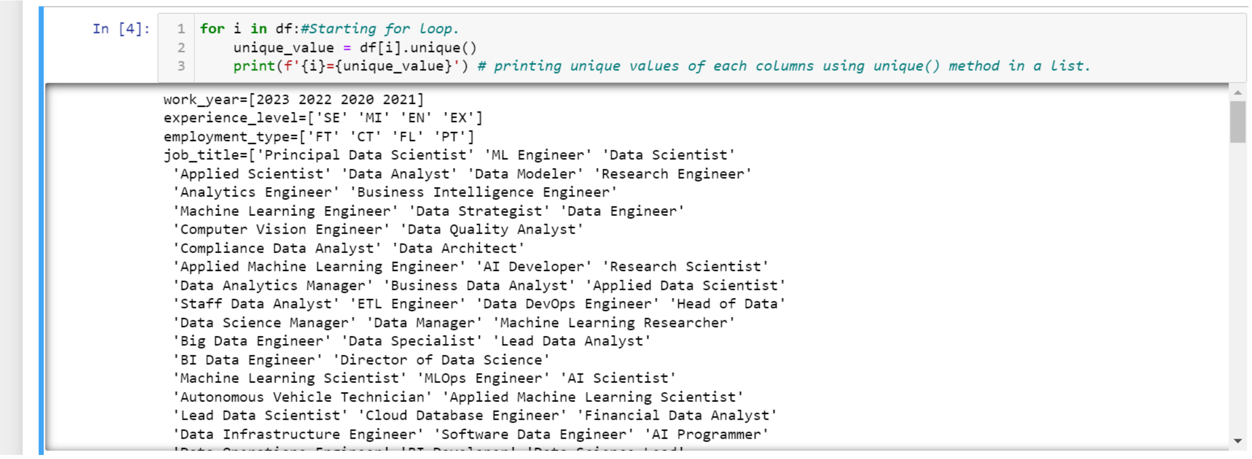
```
In [17]:
#Printing duplicate values
print('Total duplicate values: ')
print(dublicate_value.count()) # Counting all duplicate values of each columns using count() method.
```

```
Total duplicate values:
work_year             1171
experience_level      1171
employment_type       1171
job_title             1171
salary_in_usd         1171
employee_residence    1171
remote_ratio          1171
company_location      1171
company_size          1171
dtype: int64
```

*Figure 10 Counting duplicate values.*

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

This code iterates in each column in the data frame and prints the name of each column. The for loop iterates in each column and using unique () method it will find the unique value of each column. Then it prints the unique value. Here, “f” refers to string which is used to insert the column name and unique values.



```python
for i in df:#Starting for loop.
    unique_value = df[i].unique()
    print(f'{i}={unique_value}') # printing unique values of each columns using unique() method in a list.
```

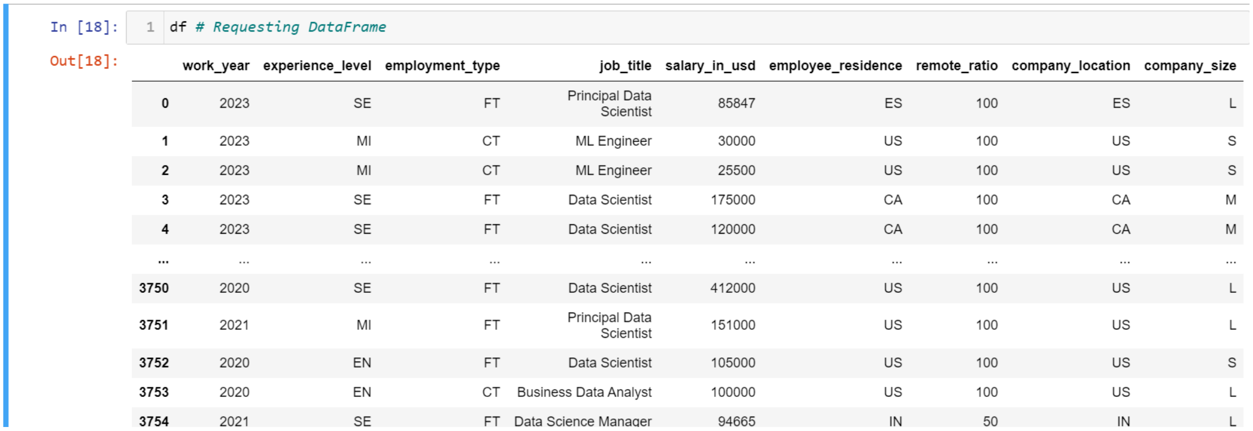
```
work_year=[2023 2022 2020 2021]
experience_level=['SE' 'MI' 'EN' 'EX']
employment_type=['FT' 'CT' 'FL' 'PT']
job_title=['Principal Data Scientist' 'ML Engineer' 'Data Scientist'
 'Applied Scientist' 'Data Analyst' 'Data Modeler' 'Research Engineer'
 'Analytics Engineer' 'Business Intelligence Engineer'
 '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'
```

*Figure 11 Printing all the unique values.*

## 2.6. Rename the experience level columns as below.

1. **SE – Senior Level/Expert**
2. **MI – Medium Level/Intermediate**
3. **EN – Entry Level**
4. **EX – Executive Level**

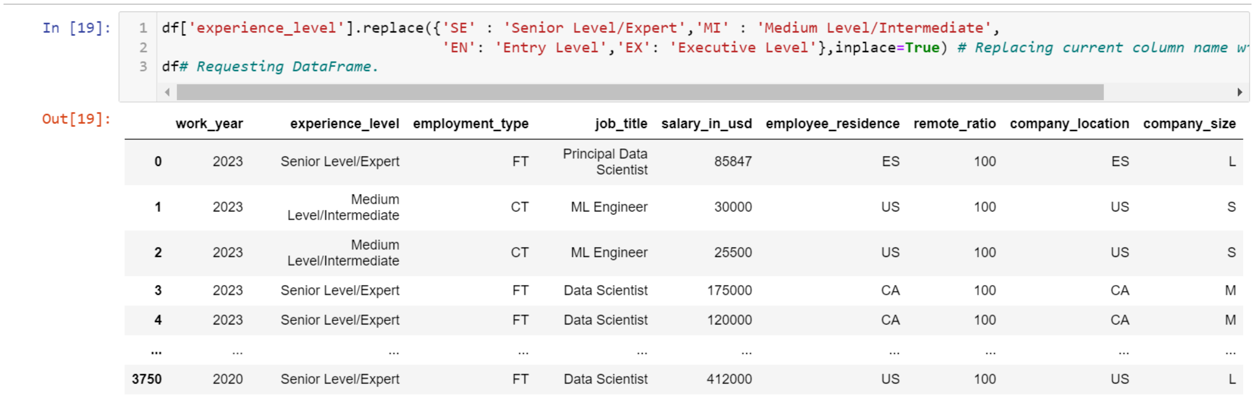
The replace () method is used to replace specific value with a new one. Here the inplace = true parameter ensure that the changes should implement directly to original Data frame.



```
In [18]: df # Requesting DataFrame
Out[18]:
```

| | 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 12 Calling Data frame before changing the experience level*



```
In [19]: df['experience_level'].replace({'SE' : 'Senior Level/Expert','MI' : 'Medium Level/Intermediate',
                                        'EN': 'Entry Level','EX': 'Executive Level'},inplace=True) # Replacing current column name w
         df# Requesting DataFrame.
Out[19]:
```

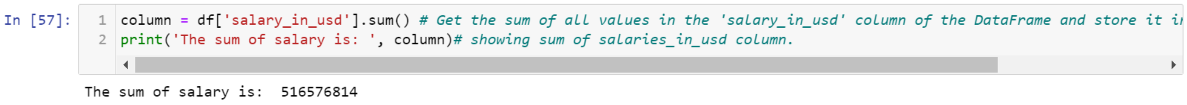
| | work_year | experience_level | employment_type | job_title | salary_in_usd | employee_residence | remote_ratio | company_location | company_size |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 2023 | Senior Level/Expert | FT | Principal Data Scientist | 85847 | ES | 100 | ES | L |
| 1 | 2023 | 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 13 Changing Experience level.*

# 3. Data Analysis

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

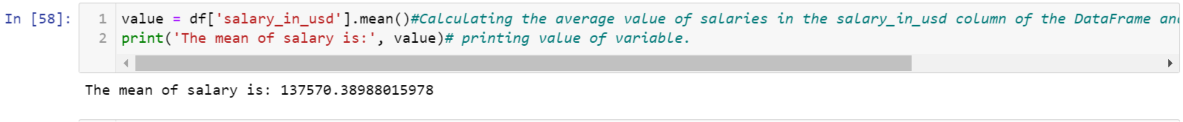
Here The sum () method is used to calculate the sum of values in a column. The mean () method is used to calculate the average of values in the salary_in_usd column. The std () method is used to calculate the standard deviation of the mention column. Next, the skew () method is used to compute the skewness of the distribution of values in a column. Then, Kurt () method is used to calculate the Kurtosis of values. Finally, describe () method provide summery of the data frame which include all method above.



```
In [57]:
column = df['salary_in_usd'].sum() # Get the sum of all values in the 'salary_in_usd' column of the DataFrame and store it i
print('The sum of salary is: ', column)# showing sum of salaries_in_usd column.
```

```
The sum of salary is:  516576814
```

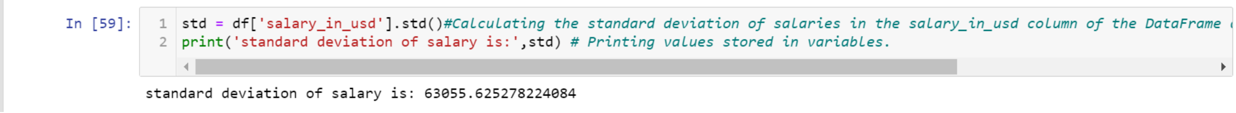
*Figure 14 Getting sum of salary in usd.*



```
In [58]:
value = df['salary_in_usd'].mean()#Calculating the average value of salaries in the salary_in_usd column of the DataFrame an
print('The mean of salary is:', value)# printing value of variable.
```

```
The mean of salary is: 137570.38988015978
```

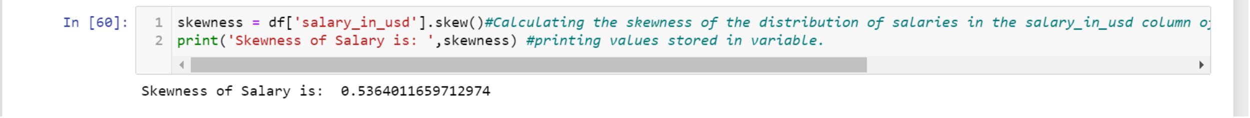
*Figure 15 getting mean value for salary in usd.*



```
In [59]:
std = df['salary_in_usd'].std()#Calculating the standard deviation of salaries in the salary_in_usd column of the DataFrame
print('standard deviation of salary is:',std) # Printing values stored in variables.
```

```
standard deviation of salary is: 63055.625278224084
```

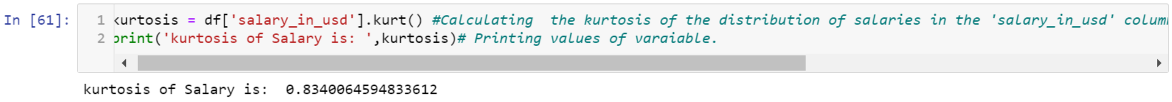
*Figure 16 Getting Standard deviation of salaries in usd.*



```
In [60]:
skewness = df['salary_in_usd'].skew()#Calculating the skewness of the distribution of salaries in the salary_in_usd column o
print('Skewness of Salary is: ',skewness) #printing values stored in variable.
```

```
Skewness of Salary is:  0.5364011659712974
```

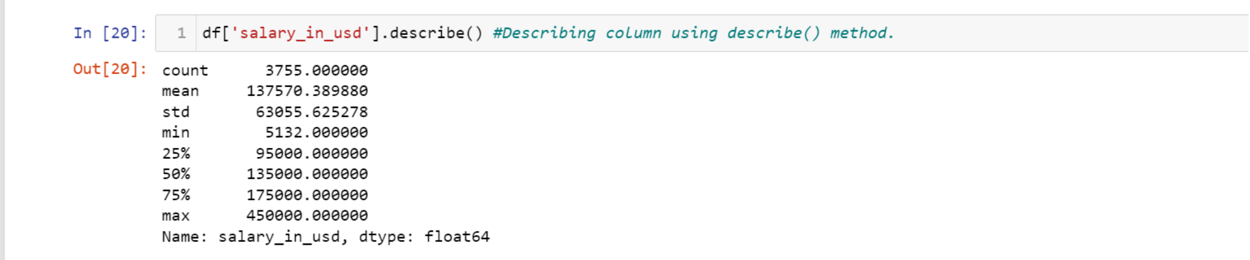
*Figure 17 Getting Skewness of salaries in usd.*



```
In [61]:
kurtosis = df['salary_in_usd'].kurt() #Calculating  the kurtosis of the distribution of salaries in the 'salary_in_usd' colum
print('kurtosis of Salary is: ',kurtosis)# Printing values of varaiable.

kurtosis of Salary is:  0.8340064594833612
```

*Figure 18 Getting krutosis in salaries in usd.*



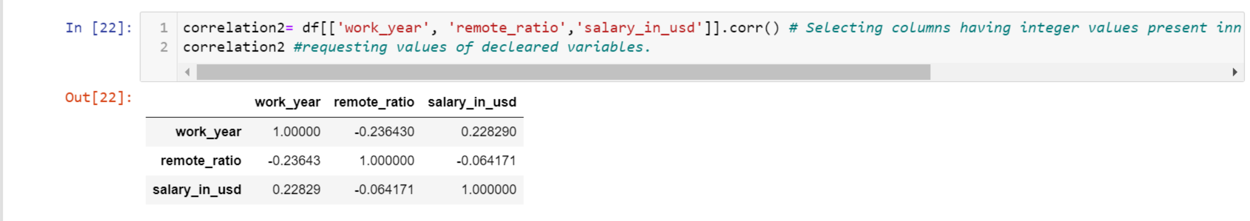
```
In [20]:
df['salary_in_usd'].describe() #Describing column using describe() method.

Out[20]:
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
```

*Figure 19 Describing columns.*

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

Here the corr() method is used to compute the correlation (linear relationship) between pair of column in a data frame. Here three column which have integer value namely work_year, remote_ratio, and salary_in_usd form the data frame.

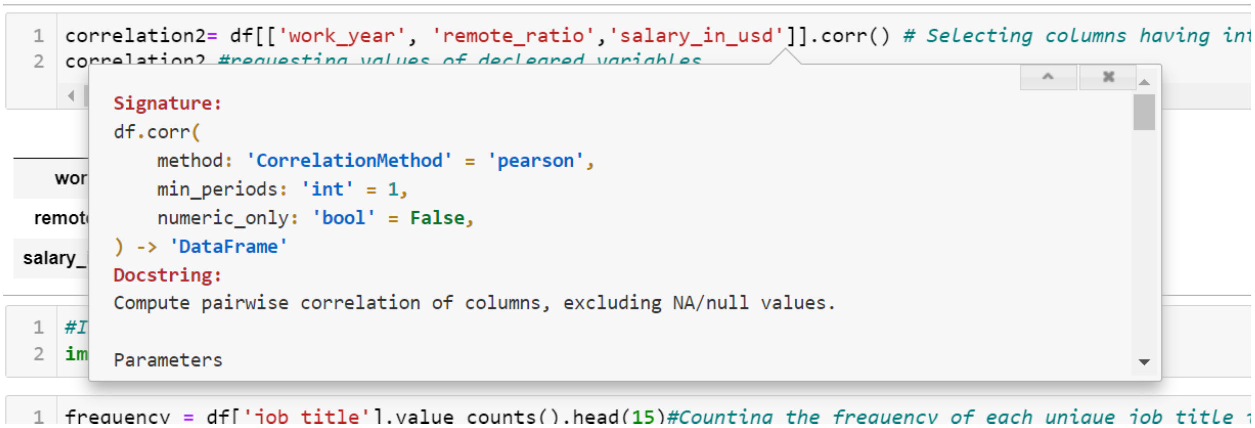


```
In [22]:
correlation2= df[['work_year', 'remote_ratio','salary_in_usd']].corr() # Selecting columns having integer values present inn
correlation2 #requesting values of decleared variables.

Out[22]:
```

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

*Figure 20 Using Code of Correlation Matrix*



```
correlation2= df[['work_year', 'remote_ratio','salary_in_usd']].corr() # Selecting columns having int
correlation2 #requesting values of decleared variables
```

wor
remot
salary_

```
Signature:
df.corr(
    method: 'CorrelationMethod' = 'pearson',
    min_periods: 'int' = 1,
    numeric_only: 'bool' = False,
) -> 'DataFrame'
Docstring:
Compute pairwise correlation of columns, excluding NA/null values.

Parameters
```

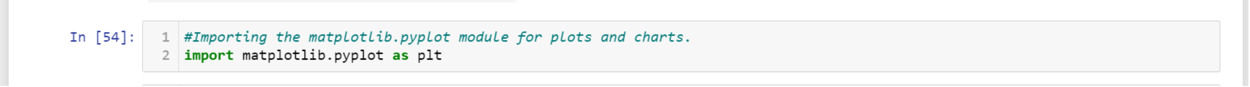
```
#I
im
```

```
frequency = df['job_title'].value_counts().head(15)#Counting the frequency of each unique job title
```

*Figure 21 Signature of Correlation matrix.*

# 4. Data Exploration

Importing matplotlib.pyplot module which is used for creating various types of plots and charts of the python. Here, the plt is an alias. This module helps in the wide range of functions and capabilities for the visualizing data which includes line plot, bar plot histograms, scatter plots etc.

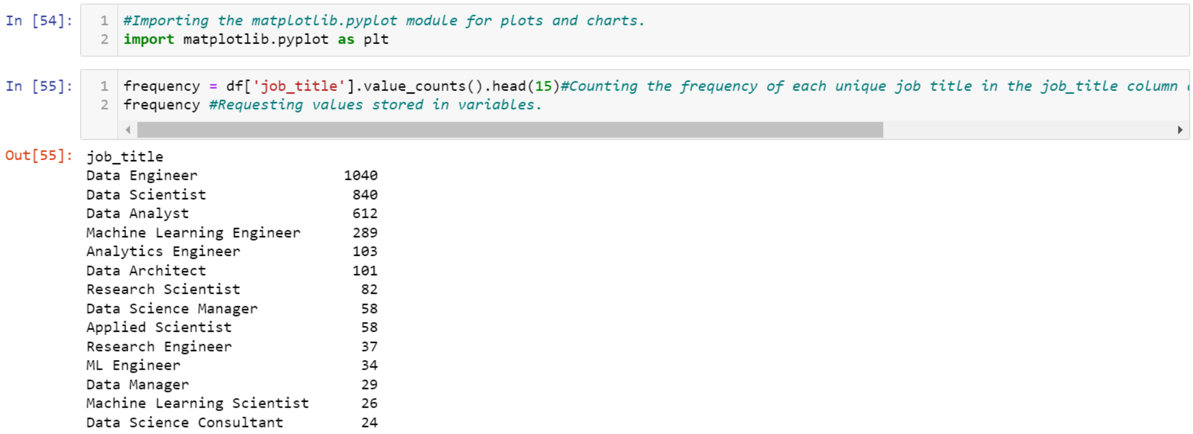


```
In [54]:
#Importing the matplotlib.pyplot module for plots and charts.
import matplotlib.pyplot as plt
```

*Figure 22 importing matplotlib.*

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

Here the value_counts() method is used to calculate the frequency of each unique value in a column and using head() method will select first 15 row serially. Plot(kind='bar) method is used to bar plot of the data where "kind = bar" parameter is used to declared type of plot to be create. Moreover, plt.figure() is used to set the size of plot here 8 is width and 5 is height to the plot. Plt.title() is used to set title of the plot. Plt.xlabel and plt.ylabel function is used to set label of x-axis and y-axis respectively. Moreover, plt.xtricks() function is used to rotate the x-axis label for better readability. Lastly, plt.show() is used to display the plot.



```
In [54]:  #Importing the matplotlib.pyplot module for plots and charts.
          import matplotlib.pyplot as plt

In [55]:  frequency = df['job_title'].value_counts().head(15)#Counting the frequency of each unique job title in the job_title column
          frequency #Requesting values stored in variables.

Out[55]:  job_title
          Data Engineer                1040
          Data Scientist                840
          Data Analyst                  612
          Machine Learning Engineer     289
          Analytics Engineer            103
          Data Architect                101
          Research Scientist             82
          Data Science Manager           58
          Applied Scientist              58
          Research Engineer              37
          ML Engineer                    34
          Data Manager                   29
          Machine Learning Scientist     26
          Data Science Consultant        24
```

*Figure 23 code to count top 15 jobs.*

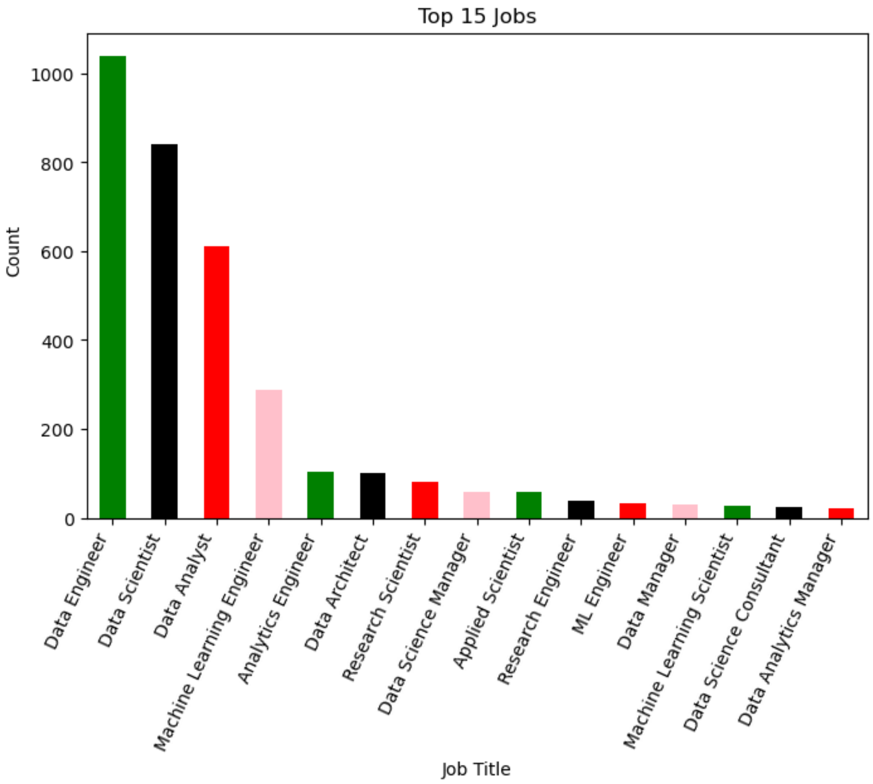
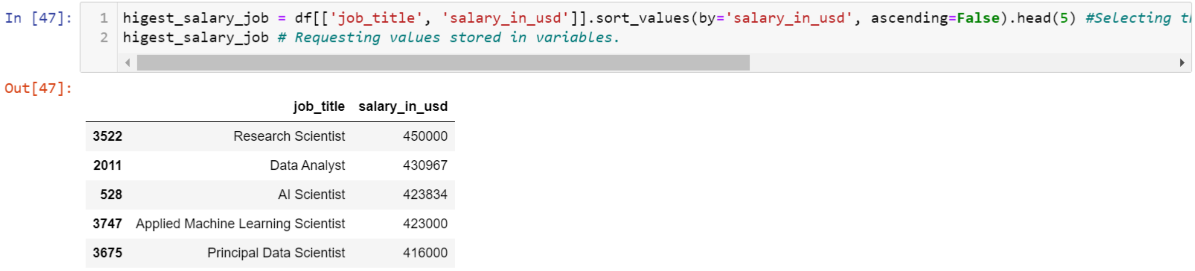


*Figure 24 Chart of all the top 15 jobs.*

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

Here the sort_values() method is used to sort the data frame by the values in the salary_in_usd column in descending order of top five with the highest salary using head() method. Plt.bar() function is used to create a bar plot showing the job title against their corresponding salaries. Moreover, plt.figure()  is used to set the size of plot here 15 is width and 9 is height to the plot. Plt.title() is used to set title of the plot. Plt.xlabel and plt.ylabel function is used to set label of x-axis and y-axis respectively. Moreover, plt.xtricks() function is used to rotate the x-axis label for better readability. Lastly, plt.show() is used to display the bar graph of the data frame.

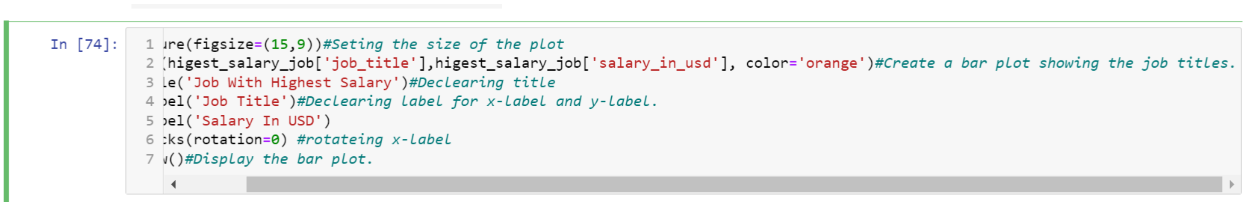


```
In [47]: higest_salary_job = df[['job_title', 'salary_in_usd']].sort_values(by='salary_in_usd', ascending=False).head(5) #Selecting ti
         higest_salary_job # Requesting values stored in variables.
```

Out[47]:

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

*Figure 25 Highest paying job based on salary.*



```
In [74]: ure(figsize=(15,9))#Seting the size of the plot
         (higest_salary_job['job_title'],higest_salary_job['salary_in_usd'], color='orange')#Create a bar plot showing the job titles.
         le('Job With Highest Salary')#Declearing title
         bel('Job Title')#Declearing label for x-label and y-label.
         bel('Salary In USD')
         cks(rotation=0) #rotateing x-label
         v()#Display the bar plot.
```

*Figure 26 Plottig bargraph for the top jobs based on salary.*

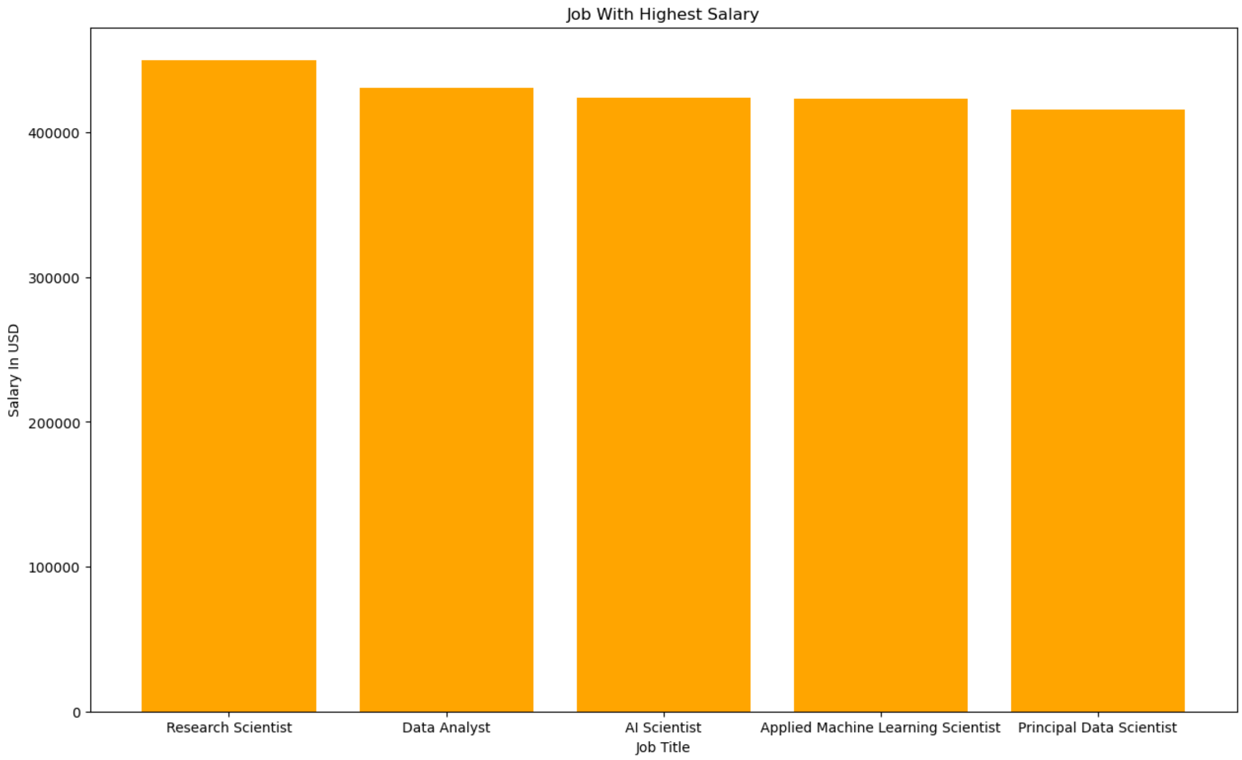
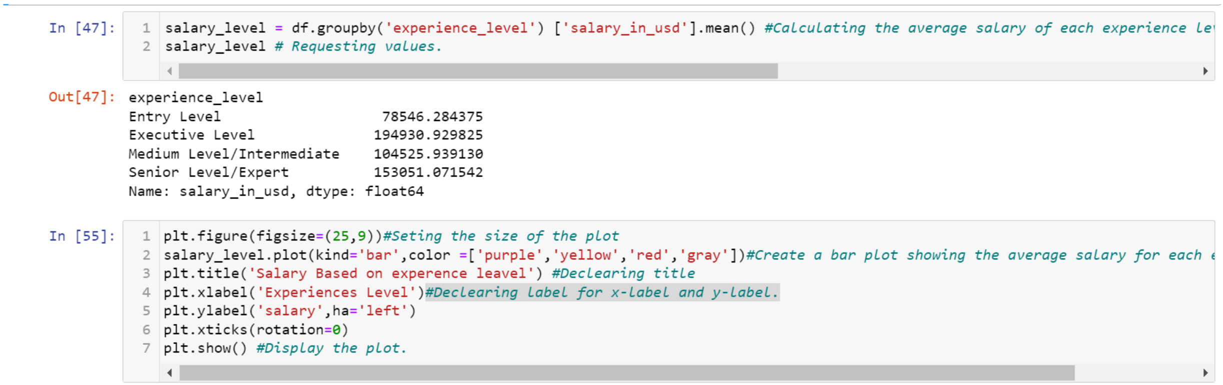


*Figure 27 bargraph of highest paying jobs.*

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

Here in this code, groupby () method is used to group the data frame in basic of experiences_level column. Mean () method is used to calculate the average salary of each experiences level. Moreover, plot () method is used to draw a plot as bar to visualized the average salary form experience level. .Plt.xlabel and plt.ylabel function is used to set label of x-axis and y-axis respectively. plt.xtricks() function is used to rotate the x-axis label for better readability. Lastly, plt.show() is used to display the bar graph.

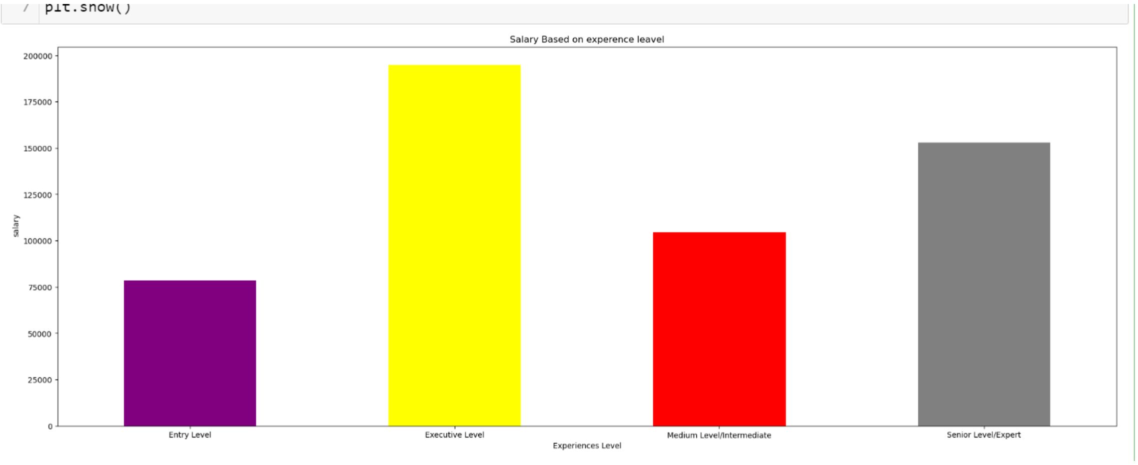


```
In [47]:  salary_level = df.groupby('experience_level') ['salary_in_usd'].mean() #Calculating the average salary of each experience Le
          salary_level # Requesting values.

Out[47]:  experience_level
          Entry Level                   78546.284375
          Executive Level              194930.929825
          Medium Level/Intermediate    104525.939130
          Senior Level/Expert          153051.071542
          Name: salary_in_usd, dtype: float64

In [55]:  plt.figure(figsize=(25,9))#Seting the size of the plot
          salary_level.plot(kind='bar',color =['purple','yellow','red','gray'])#Create a bar plot showing the average salary for each e
          plt.title('Salary Based on experence leavel') #Declearing title
          plt.xlabel('Experiences Level')#Declearing label for x-label and y-label.
          plt.ylabel('salary',ha='left')
          plt.xticks(rotation=0)
          plt.show() #Display the plot.
```

*Figure 28 code to display experience level based on total salary.*

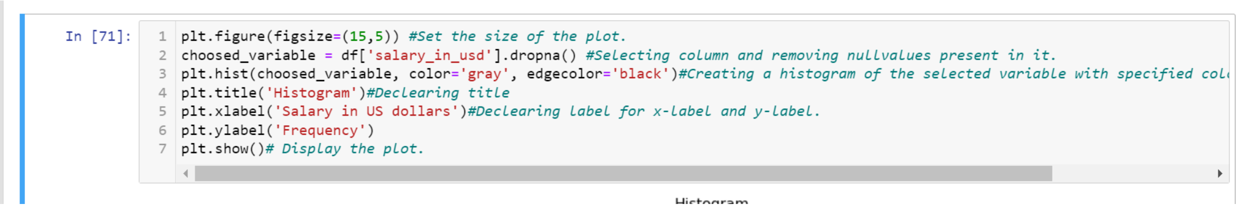


```
plt.show()
```

*Figure 29 graph of total salary 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,

For making histogram in the below code, plt.figure() is used to set size of the histogram in which 15 is width and 5 is height. Whereas dropna() method is used remove null values within the salary_in_usd column. Plt.hist() is used to create a histogram of the selected column with appropriate colour mentioned below. Lastly, plt.show() function help to display histogram.



```
In [71]:
plt.figure(figsize=(15,5)) #Set the size of the plot.
choosed_variable = df['salary_in_usd'].dropna() #Selecting column and removing nullvalues present in it.
plt.hist(choosed_variable, color='gray', edgecolor='black')#Creating a histogram of the selected variable with specified col
plt.title('Histogram')#Declearing title
plt.xlabel('Salary in US dollars')#Declearing label for x-label and y-label.
plt.ylabel('Frequency')
plt.show()# Display the plot.
```

*Figure 30 Plotting histogram of salary in usd.*

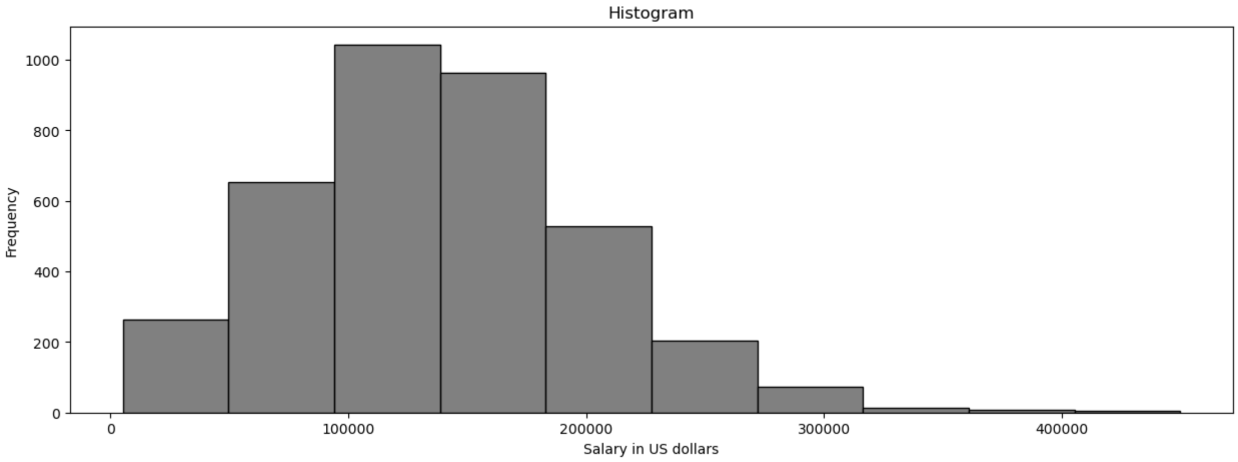
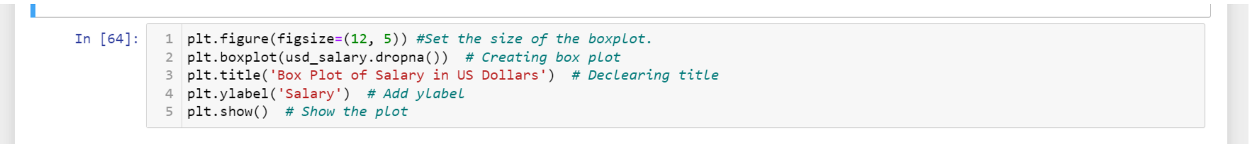


*Figure 31 graph of histogram.*

For box plot, plt.figure() is used to set the size of the box plot where 12 is with and 5 is height. Moreover, dropna() is used to remove all null values present in the selected column. Plt.boxplot() is used to create the display the box plot.



```
In [64]: plt.figure(figsize=(12, 5)) #Set the size of the boxplot.
         plt.boxplot(usd_salary.dropna())  # Creating box plot
         plt.title('Box Plot of Salary in US Dollars')  # Declearing title
         plt.ylabel('Salary')  # Add ylabel
         plt.show()  # Show the plot
```

*Figure 32 plotting boxplot for salary in usd.*

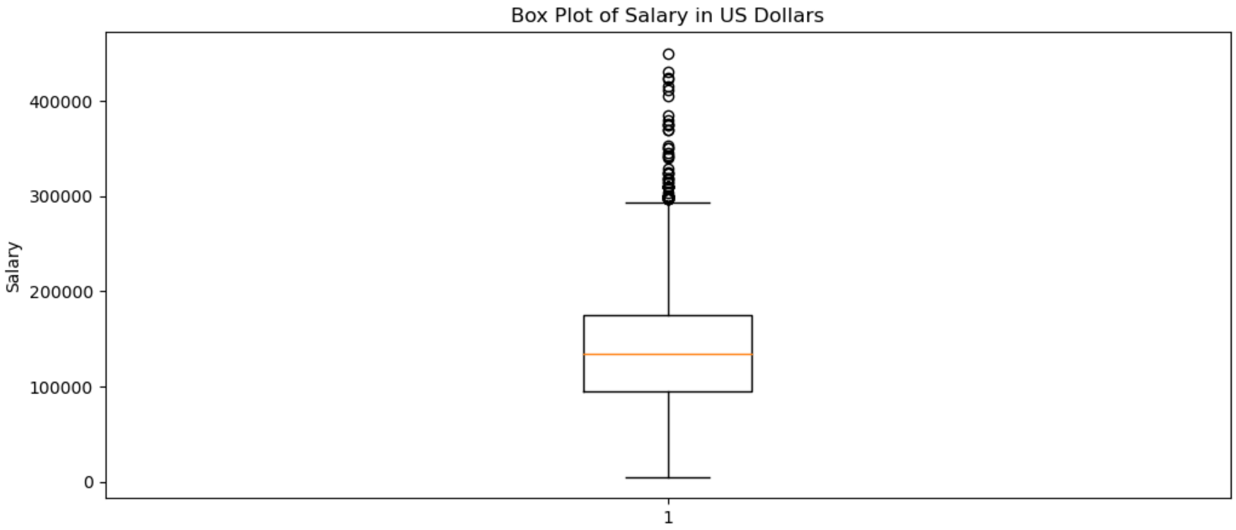


*Figure 33 boxplot of salary in us dollars.*