60% CC5067NI-Smart Data Discovery

2023-24 Autumn

Student Name: Shreyash Basnet
London Met ID: 22067847
College ID: np01cp4a220233
Assignment Due Date: Monday, May 13, 2024
Assignment Submission Date: Monday, May 13, 2024
Word Count: 5202

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

Table of Contents

Table of Figures

1. Data Understanding

The "DataScienceSalaries_8b290669-a5e9-45bf-be72-d27add2eacae_93472_.csv" file contains a lengthy list of occupations together with information on salaries, work ethics, and employee residences. There are 11 columns containing different "types" of data in the 3755 rows of data that are presented. Every one of these positions is unique, with some being remote and others requiring office hours, based on the data. Let's look at each variable and its possible significance to gain an understanding of the features of the dataset that has been provided: work_year: This variable indicates the year that the data was recorded or the year that the employee began working. When examining trends over an extended period of time, experience_level: Shows how experienced a person is in their particular role. This variable can reveal information about how employees' experience levels are distributed. employment_type: explains the kind of job (full-time, contract, part-time, etc.). It facilitates comprehension of the employment arrangements seen in the dataset. job_title: Indicates a person's position or title in the data science industry. Understanding the various roles and responsibilities within the dataset depends on this variable. pay: shows the compensation corresponding to each position or person. An important factor in examining compensation trends and discrepancies is salary. salary_currency: Indicates the currency that is used to provide salaries. It guarantees that the dataset's currency representation is consistent. salary_in_usd: This variable shows the salary in US dollars (USD). For analytical purposes, this variable can be helpful in standardising wage values. employee_residence: Identifies the place of employment for employees. Analysing geographic distributions and trends in remote work can benefit from it. remote_ratio: Indicates how much of each employee's job is done remotely. This variable aids in determining how common remote work arrangements are. company_location: Indicates the company's precise location. It is helpful for examining regional variations in pay and employment prospects. company_size: Indicates the company's size, such as small, medium, or large. This variable shed light on how the dataset's companies are distributed in terms of size. We can learn more about the features and organisation of the dataset by looking at these variables, and this knowledge can help direct future research and interpretation of the data sources.

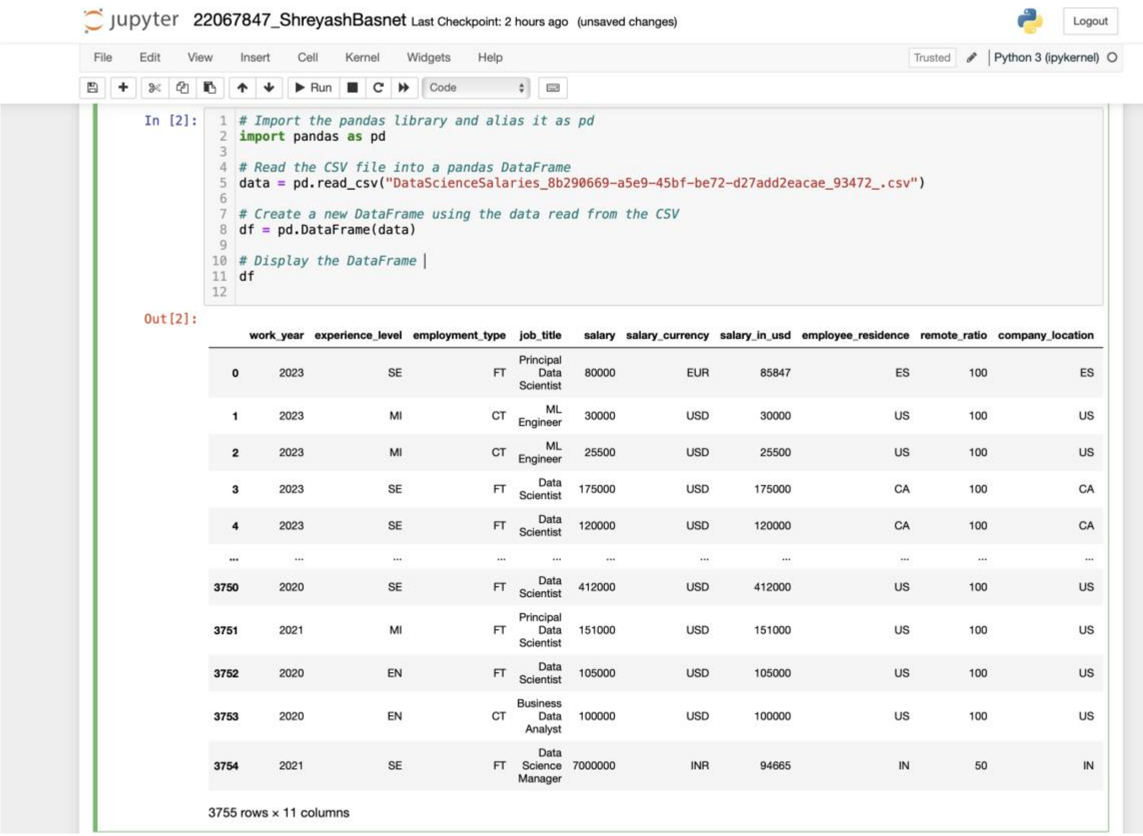
Several important insights into the dataset are revealed by the study. First of all, it lists the top 15 job titles that are most frequently associated with data science and related fields. These include occupations like Data Scientist, Data Engineer, and Machine Learning Engineer, demonstrating the importance of technical responsibilities in the sector. A look into the demand for different skill sets and areas of knowledge within the data science domain can be obtained from the frequency distribution of these job titles.

Second, it illustrates the earning potential at various stages of a person's career by showing the maximum income for each experience level. According to the data, the maximum wage increases gradually from Entry Level to Executive Level, with Senior Level positions paying a significant salary that is just somewhat less than the maximum for Medium Level positions. This discrepancy points to many variables at work, including organizational structures and the particular duties and abilities connected to each experience level.

Lastly, a thorough analysis of the pay variable, including its distributional characteristics, is provided by the summary statistics. With a mean income of almost $137,570.39, the dataset's central tendency is shown, while the pay variability is represented by a standard deviation of roughly $63,055.63. The distribution is positively skewed (skewness = 0.536), indicating that a tail extends towards higher values while the majority of wages are grouped towards the lower end. Furthermore, the salary distribution is marginally more peaked (leptokurtic) than a normal distribution, as indicated by the kurtosis value of 0.831, which may point to a concentration of incomes near the mean. All things considered; these summary figures offer insightful information about the pay structure in the data science sector.

2. Data Preparation

- Write a python program to load data into pandas DataFrame.




```python
# Import the pandas library and alias it as pd
import pandas as pd

# Read the CSV file into a pandas DataFrame
data = pd.read_csv("DataScienceSalaries_8b290669-a5e9-45bf-be72-d27add2eacae_93472_.csv")

# Create a new DataFrame using the data read from the CSV
df = pd.DataFrame(data)

# Display the DataFrame
df
```

	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
...	...	...	...	...	...	...	...	...	...	...
3750	2020	SE	FT	Data Scientist	412000	USD	412000	US	100	US
3751	2021	MI	FT	Principal Data Scientist	151000	USD	151000	US	100	US
3752	2020	EN	FT	Data Scientist	105000	USD	105000	US	100	US
3753	2020	EN	CT	Business Data Analyst	100000	USD	100000	US	100	US
3754	2021	SE	FT	Data Science Manager	7000000	INR	94665	IN	50	IN

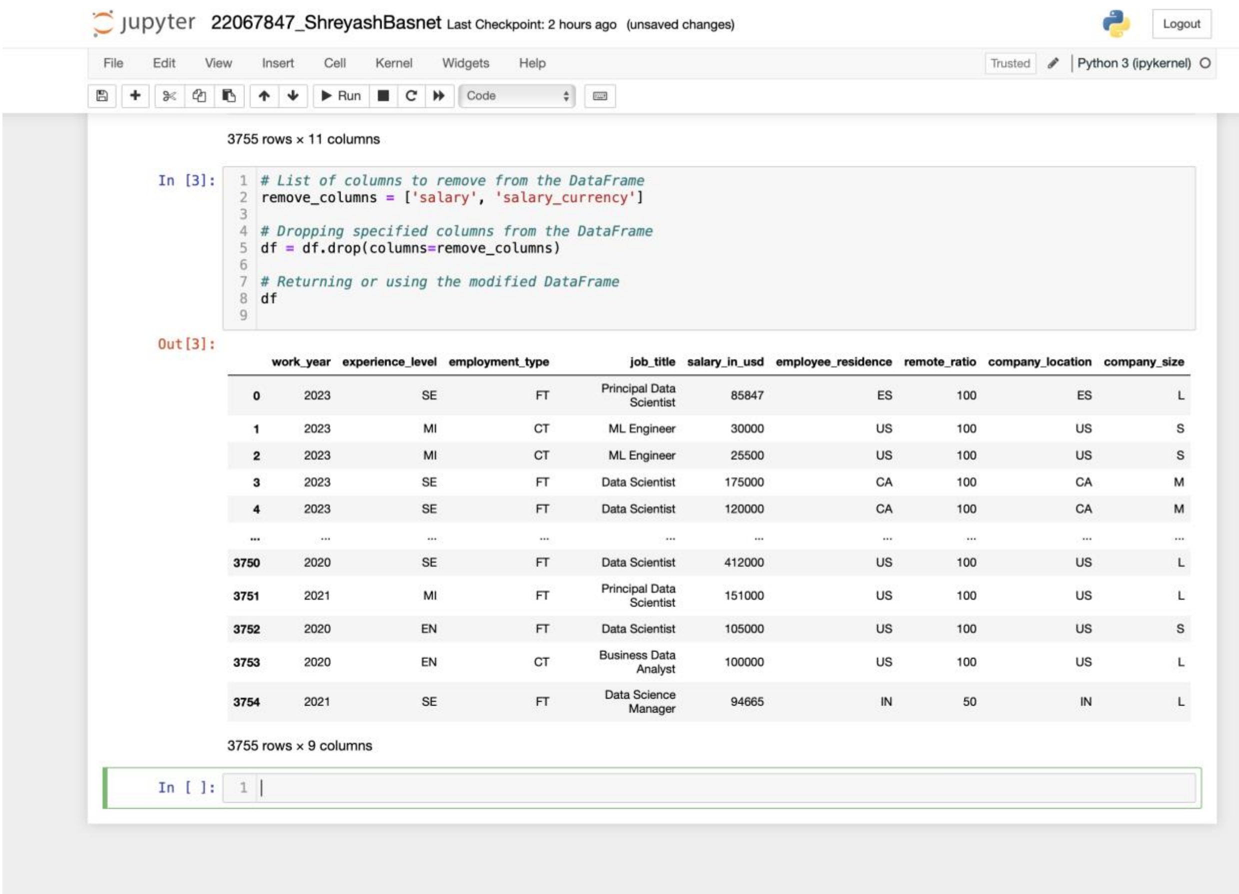
3755 rows × 11 columns

Figure 1 Program to load data into pandas DataFrame

The contents of the DataFrame "df" that was produced by reading the CSV file "DataScienceSalaries_8b290669-a5e9-45bf-be72-d27add2eacae_93472_.csv" are shown in the output.

A data entry is represented by each row of the DataFrame, and a separate attribute or feature of that entry is represented by each column. As stated in the assignment prompt, the data probably includes information concerning salaries in the data science industry. The structure of the DataFrame makes data handling and analysis simple. You can inspect the actual data items and obtain a summary of the dataset, complete with column names and associated values, by displaying the DataFrame. An overview of the dataset's contents is provided via this output, facilitating additional research and analysis.

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




```python
# List of columns to remove from the DataFrame
remove_columns = ['salary', 'salary_currency']

# Dropping specified columns from the DataFrame
df = df.drop(columns=remove_columns)

# Returning or using the modified DataFrame
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

3755 rows × 9 columns

Figure 2 Program to remove unnecessary columns i.e., salary and salary currency.

This code sample results in a modified DataFrame with the columns "salary" and "salary_currency" deleted. This is the function of every line of code:

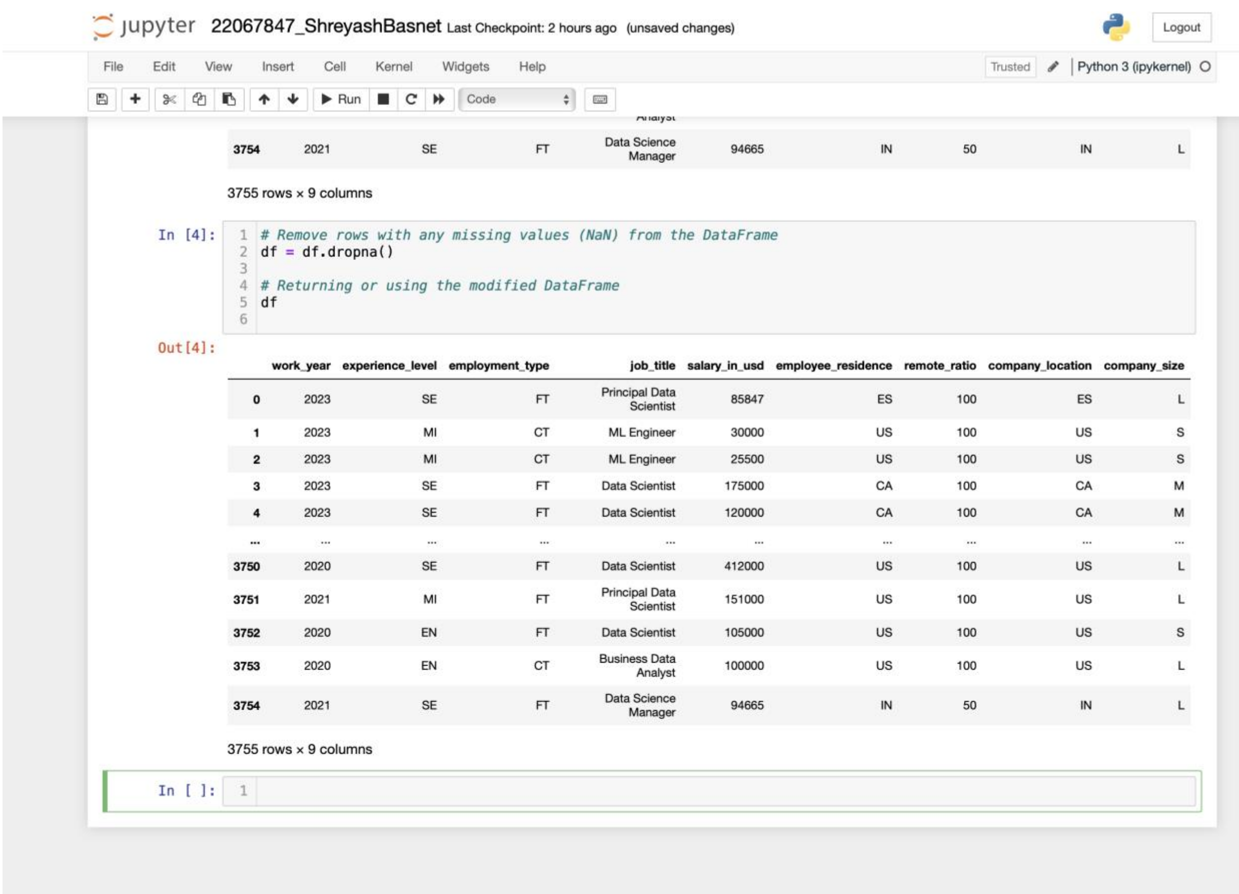
1. remove_columns = ['salary','salary_currency']: This line puts the names of the columns that need to be taken out of the DataFrame into a list.

2. df = df.drop(columns=remove_columns): This line deletes from the DataFrame df the columns listed in the remove_columns list. To eliminate rows or columns from a DataFrame, use pandas' `drop()` function. The columns to be dropped are taken out of the DataFrame by supplying the list of them as an argument to the `columns` parameter.

3. df: After removing the designated columns, this line returns or shows the updated DataFrame df.

This code line produces the updated DataFrame {df{, which only contains the remaining columns after the'salary' and'salary_currency' columns are deleted. With this change, the DataFrame is ready for additional processing or analysis without the extraneous columns pertaining to pay information.

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


jupyter 22067847_ShreyashBasnet Last Checkpoint: 2 hours ago (unsaved changes)

3754	2021	SE	FT	Data Science Manager	94665	IN	50	IN	L

3755 rows × 9 columns

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

# Returning or using the modified DataFrame
df
```

Out[4]:

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

3755 rows × 9 columns

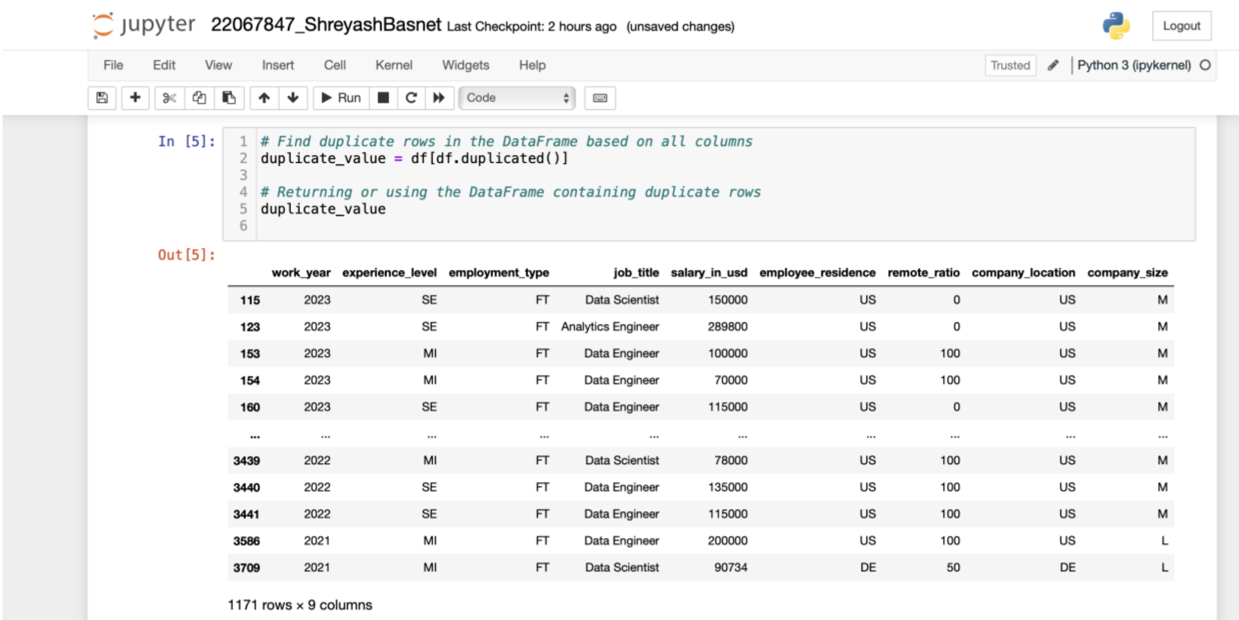
Figure 3 Program to remove the NaN missing values from updated dataframe.

The given code takes the DataFrame {df} and removes any rows that have missing values (NaN). It then either returns or uses the updated DataFrame.

Missing value rows in pandas are removed using the `dropna()` method. Any row with at least one missing element is by default removed. By removing irrelevant or incomplete data points, this process efficiently cleans the dataset and guarantees that only complete observations are kept for further analysis.

The updated DataFrame {df} with eliminated rows that contain NaN values is the result of running the function. Now that all of the rows in this DataFrame have complete information in every column, it is ready for additional processing or analysis.

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




```python
# Find duplicate rows in the DataFrame based on all columns
duplicate_value = df[df.duplicated()]

# Returning or using the DataFrame containing duplicate rows
duplicate_value

```

	work_year	experience_level	employment_type	job_title	salary_in_usd	employee_residence	remote_ratio	company_location	company_size
115	2023	SE	FT	Data Scientist	150000	US	0	US	M
123	2023	SE	FT	Analytics Engineer	289800	US	0	US	M
153	2023	MI	FT	Data Engineer	100000	US	100	US	M
154	2023	MI	FT	Data Engineer	70000	US	100	US	M
160	2023	SE	FT	Data Engineer	115000	US	0	US	M
...	...	...	...	...	...	...	...	...	...
3439	2022	MI	FT	Data Scientist	78000	US	100	US	M
3440	2022	SE	FT	Data Engineer	135000	US	100	US	M
3441	2022	SE	FT	Data Engineer	115000	US	100	US	M
3586	2021	MI	FT	Data Engineer	200000	US	100	US	L
3709	2021	MI	FT	Data Scientist	90734	DE	50	DE	L

1171 rows × 9 columns

Figure 4 Program to check duplicates value in the dataframe.

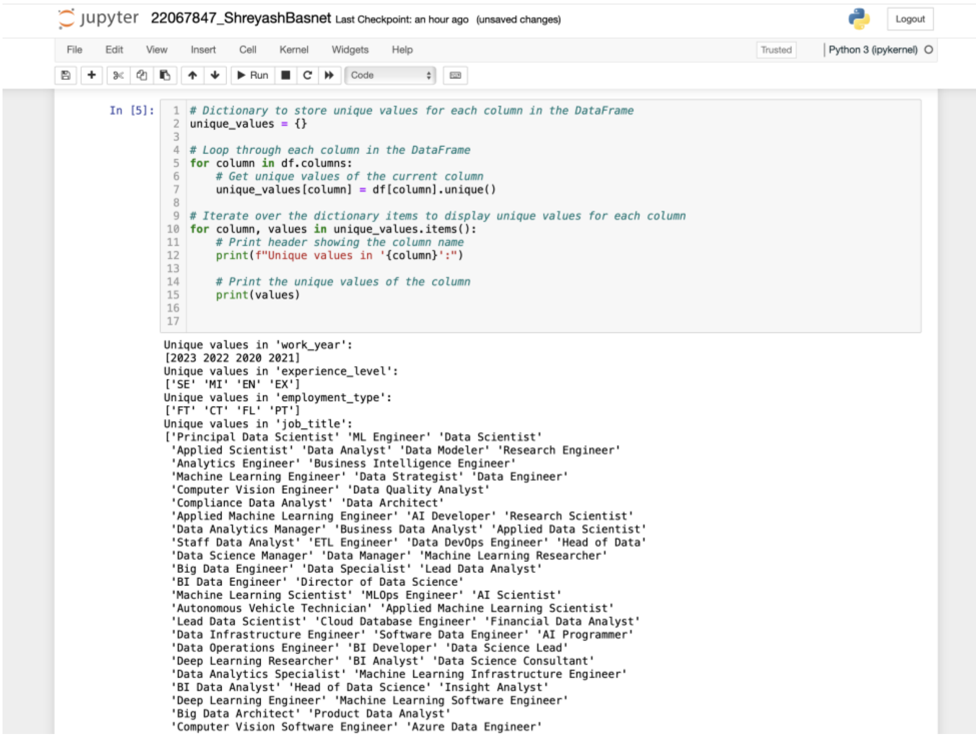
The above code searches the DataFrame for duplicate rows based on every column.

In pandas, duplicate rows in a DataFrame can be found using the `duplicated()` method. It searches for rows where all of the column values are the same as those in another row when it is used without any parameters.

Following code execution, all duplicate rows from the original DataFrame {df} are included in a DataFrame that is stored in the variable {duplicate_value}. In the original dataset, each row in this DataFrame corresponds to a duplicate observation.

The DataFrame with the duplicate rows is displayed in the `duplicate_value} output, enabling additional examination or processing as required. Data input errors or duplicate records in the dataset are examples of data quality issues that can be found and possibly resolved with the use of this information.

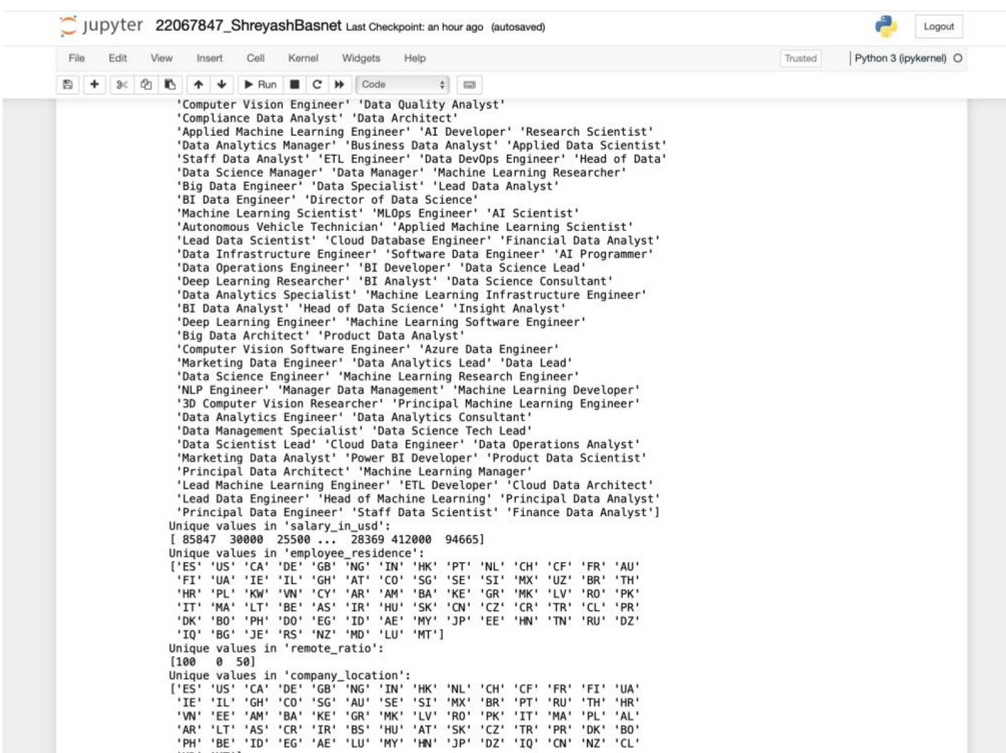
- Write a python program to see the unique values from all the columns in the dataframe.



```python
# Dictionary to store unique values for each column in the DataFrame
unique_values = {}

# Loop through each column in the DataFrame
for column in df.columns:
    # Get unique values of the current column
    unique_values[column] = df[column].unique()

# Iterate over the dictionary items to display unique values for each column
for column, values in unique_values.items():
    # Print header showing the column name
    print(f"Unique values in '{column}':")

    # Print the unique values of the column
    print(values)
```

```
Unique values in 'work_year':
[2023 2022 2020 2021]
Unique values in 'experience_level':
['SE' 'MI' 'EN' 'EX']
Unique values in 'employment_type':
['FT' 'CT' 'FL' 'PT']
Unique values in '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'
 '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'
```

Figure 5 Program to see the unique values from all the columns in the dataframe.



```
 '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'
 'Data Management Specialist' 'Data Science Tech Lead'
 'Data Scientist Lead' 'Cloud Data Engineer' 'Data Operations Analyst'
 'Marketing Data Analyst' 'Power BI Developer' 'Product Data Scientist'
 'Principal Data Architect' 'Machine Learning Manager'
 'Lead Machine Learning Engineer' 'ETL Developer' 'Cloud Data Architect'
 'Lead Data Engineer' 'Head of Machine Learning' 'Principal Data Analyst'
 'Principal Data Engineer' 'Staff Data Scientist' 'Finance Data Analyst']
Unique values in 'salary_in_usd':
[ 85847  30000  25500 ...  28369 412000  94665]
Unique values in 'employee_residence':
['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']
Unique values in 'remote_ratio':
[100   0  50]
Unique values in 'company_location':
['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']
```

Figure 6 Program to see the unique values from all the columns in the dataframe.

The purpose of this snippet of Python code is to extract and show the distinct values that are present in each column of a DataFrame. This is how it operates:

1. Initialization of the Dictionary: - {unique_values = {}}: Creates an empty dictionary named `unique_values`, in which the unique values for every column are kept.

2. Iterate through each column in the DataFrame {df} using the {for column in df.columns:} loop.

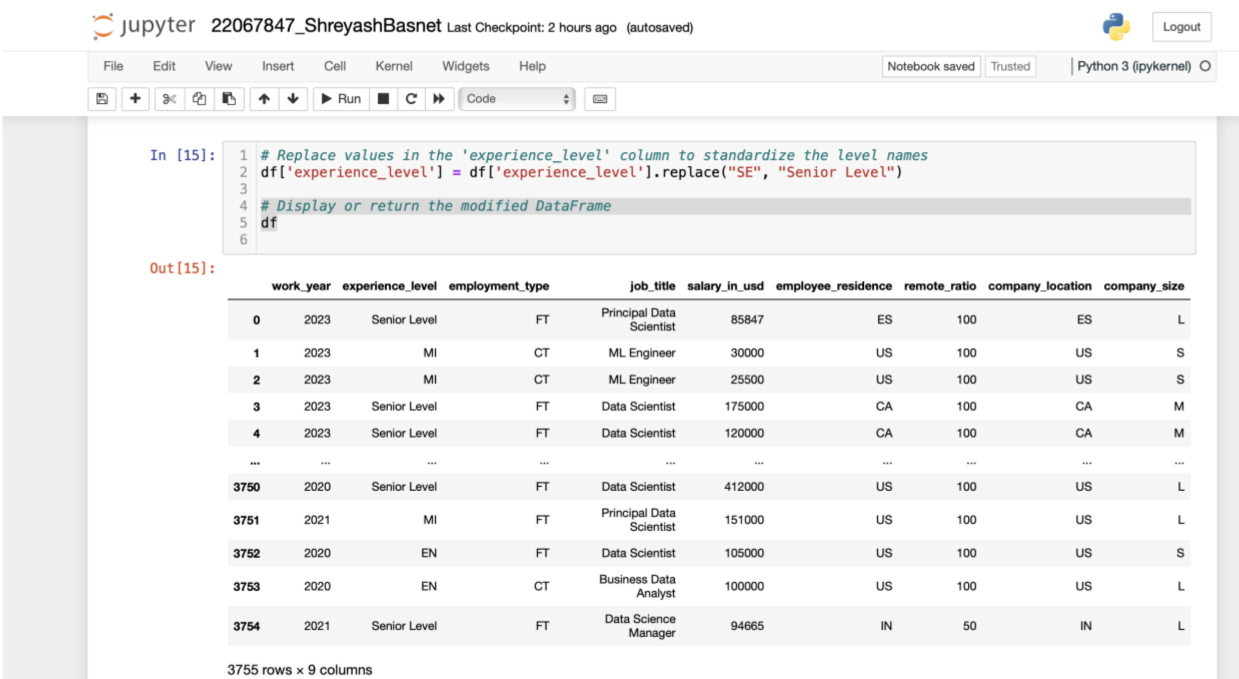
3. Determine Special Values:
   - {df[column] = unique_values[column].unique(){: It uses the pandas DataFrame function `unique()` to extract the unique values for each column, then assigns those values to the appropriate key in the `unique_values} dictionary.

4. Show Particular Values:
   - Iterates over the elements in the `unique_values` dictionary using the `for column, values in unique_values.items():` method.
     The code {print(f"Unique values in '{column}':")} generates a header with the column name printed in it.
     - {print(values)}: Outputs the column's unique values.

This code snippet will produce a structured list of unique values for each column in the DataFrame, with the column name serving as a header for each list. Giving readers a summary of the unique values found in every column makes it easier to explore and comprehend the data.

- Rename the experience level columns below.

SE – Senior Level/Expert




```python
# Replace values in the 'experience_level' column to standardize the level names
df['experience_level'] = df['experience_level'].replace("SE", "Senior Level")

# Display or return the modified DataFrame
df

```

	work_year	experience_level	employment_type	job_title	salary_in_usd	employee_residence	remote_ratio	company_location	company_size
0	2023	Senior Level	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	Senior Level	FT	Data Scientist	175000	CA	100	CA	M
4	2023	Senior Level	FT	Data Scientist	120000	CA	100	CA	M
...	...	...	...	...	...	...	...	...	...
3750	2020	Senior Level	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	Senior Level	FT	Data Science Manager	94665	IN	50	IN	L

3755 rows × 9 columns

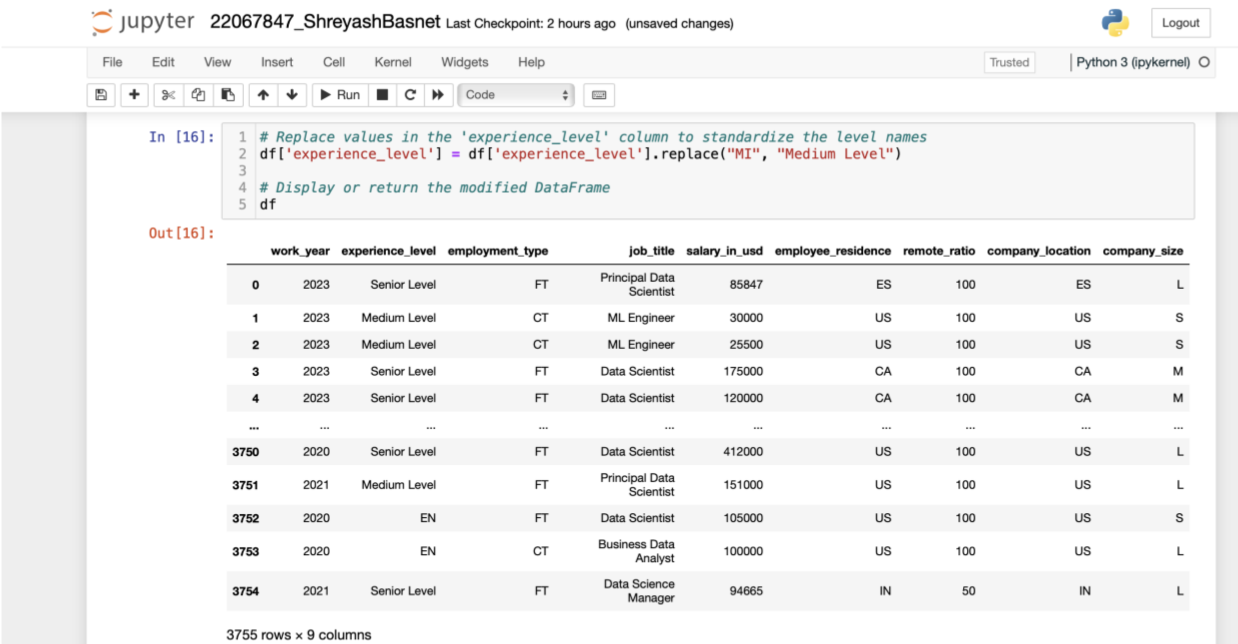
Figure 7 Rename the experience level to SE – Senior Level/Expert

This code snippet substitutes data in the DataFrame's 'experience_level' column with standardised level names. In particular, "Senior Level" is used in place of "SE". Below is a summary of the functions of the code:

1. Replacement Operation: The code replaces values in the 'experience_level' column of the DataFrame ({df{) by using the `replace()` method in pandas. {df['experience_level'].replace("SE", "Senior Level")} is the syntax that is utilised. Targeting the 'experience_level' column, this action substitutes "Senior Level" for any instance of the string "SE".
2. Standardisation: This procedure aims to harmonise how experience levels are displayed inside the DataFrame. The DataFrame's clarity and consistency are increased by using complete, descriptive names for specific codes or acronyms, such as "SE" for "Senior Level".
3. updated DataFrame: The "df" variable is used to display or return the updated DataFrame following the replacement process. This makes it possible to examine the modifications made to the 'experience_level' column.

4. Output Explanation: The DataFrame with the 'experience_level' column changed to match the standardised level names is what will be produced by the code snippet. Every instance of "SE" that existed in the "experience_level" column has been substituted with "Senior Level." The DataFrame doesn't change in any other way.

MI – Medium Level/Intermediate



```
# Replace values in the 'experience_level' column to standardize the level names
df['experience_level'] = df['experience_level'].replace("MI", "Medium Level")

# Display or return the modified DataFrame
df
```

	work_year	experience_level	employment_type	job_title	salary_in_usd	employee_residence	remote_ratio	company_location	company_size
0	2023	Senior Level	FT	Principal Data Scientist	85847	ES	100	ES	L
1	2023	Medium Level	CT	ML Engineer	30000	US	100	US	S
2	2023	Medium Level	CT	ML Engineer	25500	US	100	US	S
3	2023	Senior Level	FT	Data Scientist	175000	CA	100	CA	M
4	2023	Senior Level	FT	Data Scientist	120000	CA	100	CA	M
...	...	...	...	...	...	...	...	...	...
3750	2020	Senior Level	FT	Data Scientist	412000	US	100	US	L
3751	2021	Medium Level	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	Senior Level	FT	Data Science Manager	94665	IN	50	IN	L

3755 rows × 9 columns

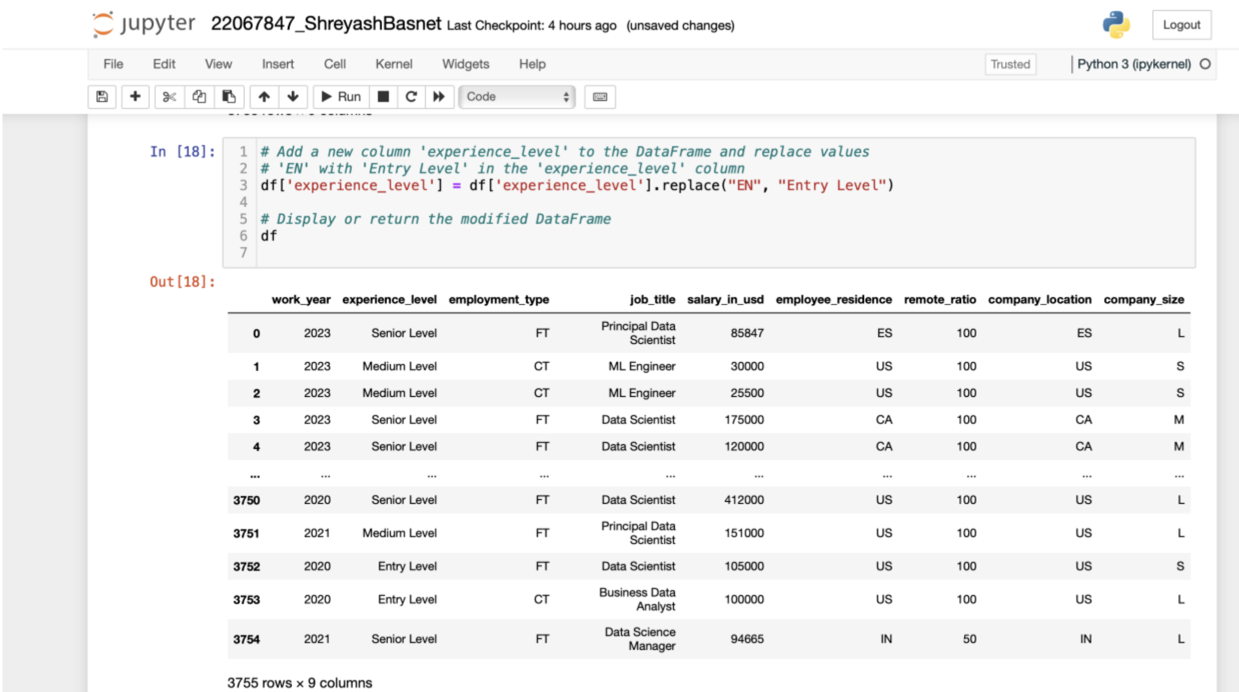
Figure 8 Rename the experience level to MI – Medium Level/Intermediate

This code snippet generates a modified DataFrame with standardised values in the 'experience_level' column. In the 'experience_level' column, "Medium Level" has been used in place of any instance of "MI".

Ensuring consistency and clarity in the representation of experience levels within the dataset is the aim of this operation. For analytical reasons, the data is made more understandable and straightforward by substituting "Medium Level" for "MI".

The DataFrame 'df' will display or return after this code has been executed, with the 'experience_level' column modified to reflect the standardised level names. Unless specifically updated by other actions, the other columns and data in the DataFrame stay the same.

EN – Entry Level



jupyter 22067847_ShreyashBasnet Last Checkpoint: 4 hours ago (unsaved changes)

In [18]:

```
# Add a new column 'experience_level' to the DataFrame and replace values
# 'EN' with 'Entry Level' in the 'experience_level' column
df['experience_level'] = df['experience_level'].replace("EN", "Entry Level")

# Display or return the modified DataFrame
df
```

Out[18]:

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

3755 rows × 9 columns

Figure 9 Rename the experience level to EN – Entry Level

The supplied code extends the DataFrame {df{ by adding a new column called 'experience_level'. 'Entry Level' is used in place of 'EN' in the new column, which is populated using the data from the current 'experience_level' column.

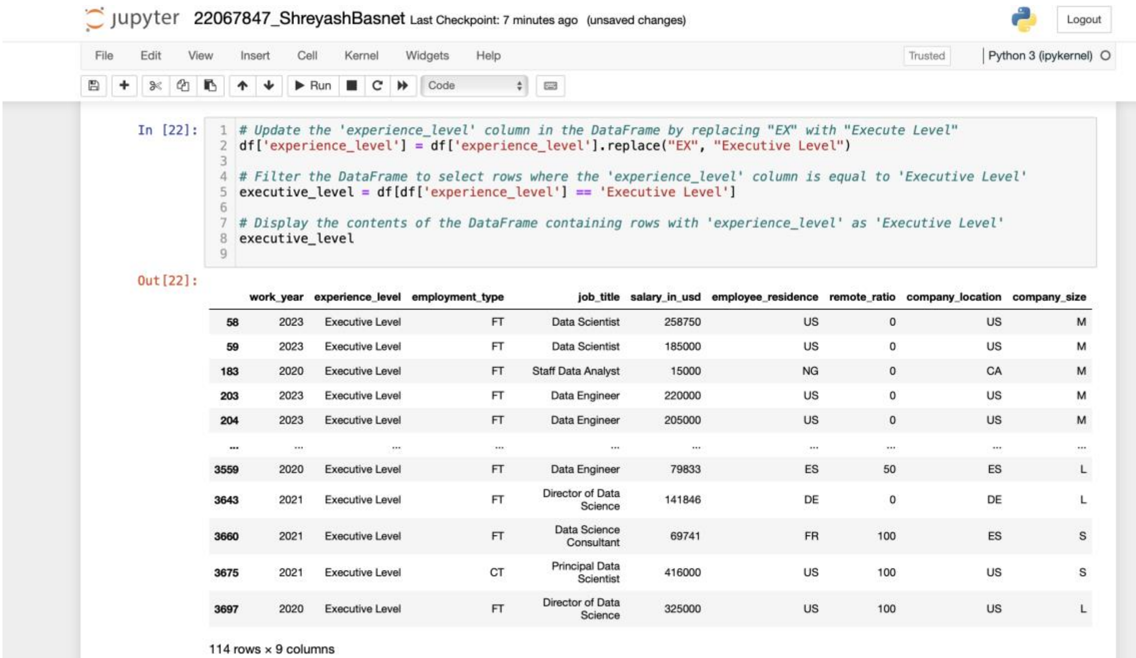
These are the functions of each code segment:

1. {df['experience_level'] = df['level of experience'].substitute("EN", "Entry Level")}: This piece of code adds a new column to the DataFrame df called 'experience_level'. 'Entry Level' is substituted for any instance of 'EN' using the values from the current 'experience_level' column. The experience level category 'EN' is essentially renamed as 'Entry Level'.

2. {df}: At last, the altered DataFrame {df} is shown or given back. When 'EN' is substituted with 'Entry Level,' the DataFrame with the newly added 'experience_level' column will be displayed.

The DataFrame {df} with the 'experience_level' column changed in accordance with the designated replacements would be the result of this code. 'Entry Level' would now take the place of every instance of 'EN' in the 'experience_level' column.

EX – Executive Level



jupyter 22067847_ShreyashBasnet Last Checkpoint: 7 minutes ago (unsaved changes)

In [22]:

```
# Update the 'experience_level' column in the DataFrame by replacing "EX" with "Execute Level"
df['experience_level'] = df['experience_level'].replace("EX", "Executive Level")

# Filter the DataFrame to select rows where the 'experience_level' column is equal to 'Executive Level'
executive_level = df[df['experience_level'] == 'Executive Level']

# Display the contents of the DataFrame containing rows with 'experience_level' as 'Executive Level'
executive_level
```

Out[22]:

	work_year	experience_level	employment_type	job_title	salary_in_usd	employee_residence	remote_ratio	company_location	company_size
58	2023	Executive Level	FT	Data Scientist	258750	US	0	US	M
59	2023	Executive Level	FT	Data Scientist	185000	US	0	US	M
183	2020	Executive Level	FT	Staff Data Analyst	15000	NG	0	CA	M
203	2023	Executive Level	FT	Data Engineer	220000	US	0	US	M
204	2023	Executive Level	FT	Data Engineer	205000	US	0	US	M
...	...	...	...	...	...	...	...	...	...
3559	2020	Executive Level	FT	Data Engineer	79833	ES	50	ES	L
3643	2021	Executive Level	FT	Director of Data Science	141846	DE	0	DE	L
3660	2021	Executive Level	FT	Data Science Consultant	69741	FR	100	ES	S
3675	2021	Executive Level	CT	Principal Data Scientist	416000	US	100	US	S
3697	2020	Executive Level	FT	Director of Data Science	325000	US	100	US	L

114 rows × 9 columns

Figure 10 Rename the experience level to EX – Executive Level

The above code manipulates the DataFrame {df} in a number of ways linked to the 'experience_level' column. The following summarises the functions of each section of the code:

1. {df['experience_level'] = df['level of experience'].replace("Executive Level", "EX")}: The DataFrame df's 'experience_level' column is updated by this line of code. Every time the word "EX" appears, "Executive Level" appears instead. In effect, this changes the category of experience level from "EX" to "Executive Level".

2. The expression {executive_level = df[df['experience_level'] == 'Executive Level']}: Using a filter, this line selects only the rows in the DataFrame df where the 'experience_level' column value is 'Executive Level'. This generates a fresh DataFrame with the name `executive_level} that exclusively includes the rows with the experience level set to 'Executive Level'.

3. {executive_level}: Lastly, the DataFrame {executive_level}'s contents are shown. When the 'experience_level' is 'Executive Level,' all the entries from the original DataFrame {df} will be displayed.

After carrying out the designated substitutions and filtering, the output of this code would be the contents of the DataFrame {executive_level}, which contains all the rows from the original DataFrame {df} where the experience level is labelled as 'Executive Level'.

3. Data Analysis

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

```
import pandas as pd
from scipy.stats import skew, kurtosis

# Read the CSV file into a pandas DataFrame
df = pd.read_csv("DataScienceSalaries_8b290669-a5e9-45bf-be72-d27add2eacae_93472_.csv")

# Function to display summary statistics of a chosen variable
def variable_summary(variable):
    # Calculate summary statistics
    sum_value = df[variable].sum()
    mean_value = df[variable].mean()
    std_dev = df[variable].std()
    skewness = skew(df[variable])
    kurt = kurtosis(df[variable])

    # Print summary statistics
    print(f"Summary statistics for variable '{variable}':")
    print(f"Sum: {sum_value}")
    print(f"Mean: {mean_value}")
    print(f"Standard Deviation: {std_dev}")
    print(f"Skewness: {skewness}")
    print(f"Kurtosis: {kurt}")

# Choose the variable you want to analyze
chosen_variable = 'salary_in_usd'

# Call the function to display summary statistics for the chosen variable
variable_summary(chosen_variable)
```

```
Summary statistics for variable 'salary_in_usd':
Sum: 516576814
Mean: 137570.38988015978
Standard Deviation: 63055.625278224084
Skewness: 0.5361868674235593
Kurtosis: 0.8312989014514311
```

Figure 11 Program to show summary statistics of sum, mean, standard deviation, skewness, and kurtosis of any chosen variable.

The code snippet that is provided gives a summary of statistics for the selected variable, in this case, "salary_in_usd." The following is what each statistic means:

1. Sum: This represents the total of all of the dataset's pay values added together. It offers an estimation of the overall profits across all the data points.

2. Mean: Also referred to as the average, it shows the wage distribution's centre tendency. It is computed by taking the total number of observations and dividing it by the sum of the salary values (BYJU's, 2024).

3. Standard Deviation: This expresses how much the compensation values deviate from the mean. Greater salary fluctuation is indicated by a higher standard deviation, whilst greater consistency is suggested by a lower value (MARSHALL HARGRAVE Full Bio Marshall

Hargrave is a stock analyst and writer with 10+ years of experience covering stocks and markets, 2023).

4. Skewness: Skewness quantifies how asymmetrically wage values are distributed. A distribution that is entirely symmetrical has a skewness value of 0. The distribution may be skewed to the right (tail towards higher values) if the skewness value is positive (>0), skewness to the left (tail towards lower values) if the skewness value is negative (<0) (Turney, 2024).

5. Kurtosis, which quantifies the distribution's "tailedness" or how sharply peaked or flat it is in relation to a normal distribution. A normal distribution is shown by a kurtosis value of 0. A peak that is flatter (platykurtic) has negative kurtosis (<0), whereas a sharper peak (leptokurtic) has positive kurtosis (>0) (KENTON, 2023).

The measures of central tendency, dispersion, skewness, and kurtosis included in these summary statistics offer a thorough overview of the income distribution and are crucial for comprehending the properties and form of the wage data.

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

jupyter 22067847_ShreyashBasnet Last Checkpoint: Last Thursday at 20:06 (unsaved changes)

In [21]:
```
import pandas as pd

# Read the CSV file into a pandas DataFrame
df = pd.read_csv("DataScienceSalaries_8b290669-a5e9-45bf-be72-d27add2eacae_93472_.csv")

# Remove non-numeric columns (assuming non-numeric columns are not relevant for correlation)
numeric_df = df.select_dtypes(include=['int64', 'float64'])

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

# Display the correlation matrix
print("Correlation Matrix:")
print(correlation_matrix)

```

```
Correlation Matrix:
               work_year    salary  salary_in_usd  remote_ratio
work_year       1.000000 -0.094724       0.228290     -0.236430
salary         -0.094724  1.000000      -0.023676      0.028731
salary_in_usd   0.228290 -0.023676       1.000000     -0.064171
remote_ratio   -0.236430  0.028731      -0.064171      1.000000
```

Figure 12 Program to calculate and show correlation of all variables.

A correlation matrix, or table displaying the correlation coefficients between pairs of variables in the dataset, is the result of the code snippet that was supplied. The linear relationship between two variables, both in intensity and direction, is quantified by correlation coefficients (policies, 2024).

In this case, the code assumes that non-numeric columns are not relevant for correlation analysis (which may not always be the case, but it's a common starting point). It reads a CSV file containing salary data for data science positions and transforms the DataFrame into a numeric-only DataFrame by selecting columns with integer and float data types.

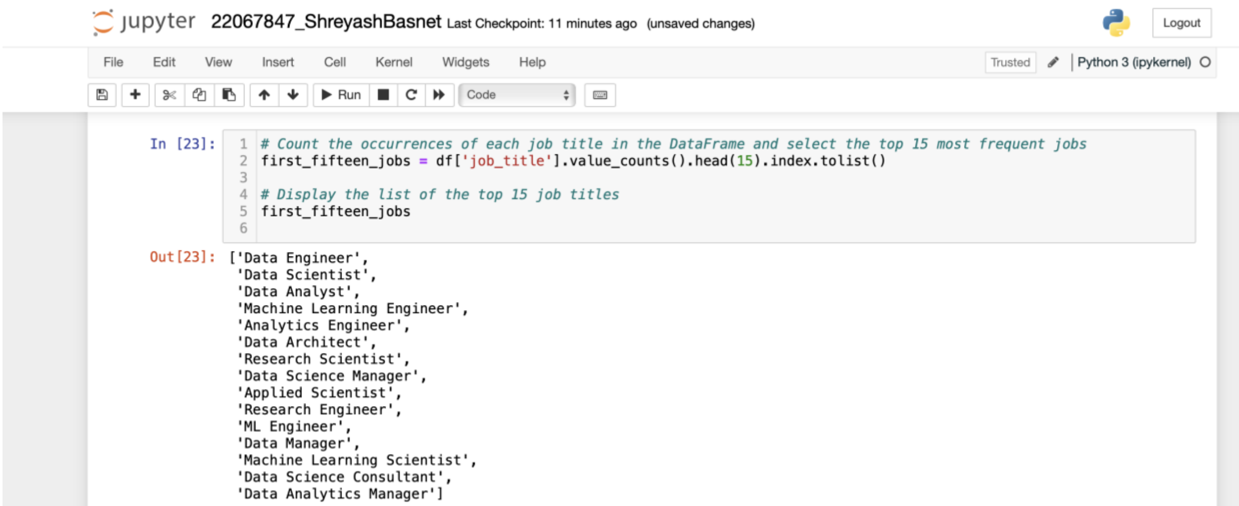
Next, the numeric DataFrame is subjected to the `corr()` method calculation to yield the correlation matrix. The correlation coefficient between two variables is represented by each cell in the matrix. The values are in the interval of -1 to 1:

- A high positive correlation is shown by a correlation coefficient that is close to 1, which means that if one variable rises, the other variable also tends to rise.
- A significant negative correlation is shown by a correlation coefficient that is near to -1, which means that as one variable rises, the other tends to fall.
- A weak or nonexistent linear correlation between the variables is indicated by a correlation coefficient that is closer to 0.

Finding relationships between the variables in the dataset may be done with the help of the correlation matrix. Negative correlations imply that the variables move against one another, whereas positive correlations indicate that they move in the same direction. Understanding the relationships between the various components in the dataset as well as using this information for future research or modelling might be beneficial.

4. Data Exploration

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



In [23]:
```python
# Count the occurrences of each job title in the DataFrame and select the top 15 most frequent jobs
first_fifteen_jobs = df['job_title'].value_counts().head(15).index.tolist()

# Display the list of the top 15 job titles
first_fifteen_jobs
```

Out[23]:
```
['Data Engineer',
 'Data Scientist',
 'Data Analyst',
 'Machine Learning Engineer',
 'Analytics Engineer',
 'Data Architect',
 'Research Scientist',
 'Data Science Manager',
 'Applied Scientist',
 'Research Engineer',
 'ML Engineer',
 'Data Manager',
 'Machine Learning Scientist',
 'Data Science Consultant',
 'Data Analytics Manager']
```

Figure 13 Program to find out the top 15 jobs

The purpose of this code snippet is to determine the top 15 most frequently occurring jobs by counting the instances of each job description in a DataFrame. Now let's dissect the code:

1. {df['job_title'].value_counts()}: The frequency of each distinct job title in the 'job_title' column of the DataFrame {df} is determined by this section of the code. In this case, the 'job_title' column is the Series, and the `value_counts()` function counts the occurrences of each unique value in the Series.
2. {.head(15)}: This function takes a sorted list of job titles from {value_counts()} and returns the top 15 most frequent job titles. The DataFrame's first fifteen rows are returned.
3. `.index.tolist()}: Lastly, the index of the resultant Series—which comprises the job titles—is transformed into a Python list using `.index.tolist()`. The top 15 most common job titles are shown here in descending order of frequency.

Thus, a list of the top 15 job titles depending on how frequently they appear in the DataFrame will be the result of this code. These are the job titles that are most commonly found in the dataset.

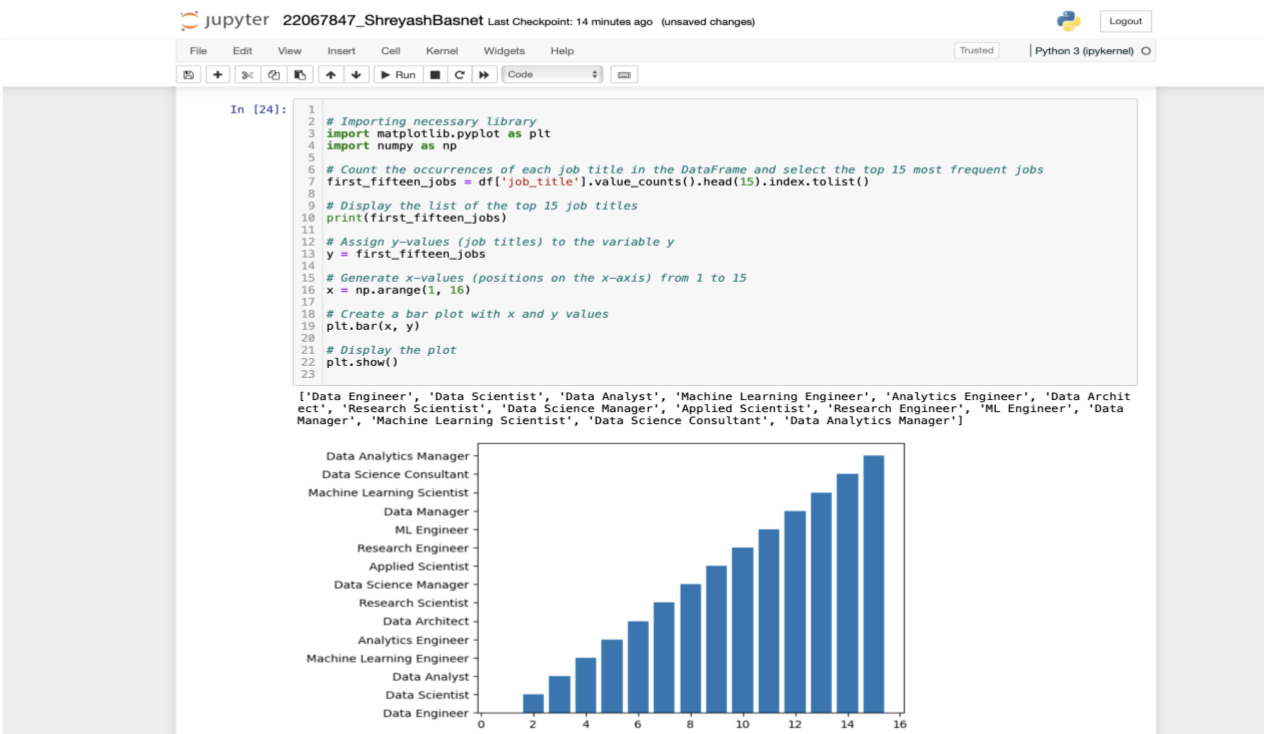


Figure 14 Program to make bar graph of sales

This code snippet uses a bar plot to display the frequency distribution of the top 15 most popular job names in a dataset. The following describes each component of the code and its result:

1. Importing Libraries: The code imports `matplotlib.pyplot` and `numpy`, which are the libraries required for plotting.

2. Counting Occurrences of Job Titles: It determines the top 15 most frequent job titles by counting the instances of each distinct job title in the DataFrame ({df}). The outcome is kept in the `first_fifteen_jobs} variable.

3. Showing the Top 15 Job Titles: `print(first_fifteen_jobs)} is used to show the list of the top 15 job titles. This gives information on the most prevalent job titles found in the sample.

4. Assigning y-values: The variable {y} is given the job titles.

5. Creating x-values: `np.arange(1, 16)} is used to create an array of x-values between 1 and 15. The locations of each job title on the x-axis are represented by these values.

6. Making the Bar Plot: To make a bar plot, use plt.bar(x, y), where x denotes the x-axis coordinates and y the corresponding job titles.

7. Showing the Plot: Lastly, the bar plot is shown using plt.show().

A bar plot displaying the distribution of the top 15 most common job titles would be the result of this code; each bar would represent a job title, and its height would indicate how frequently that title appears in the dataset. Understanding the distribution and popularity of various job titles within the data is made easier with the aid of this visualisation.

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

jupyter 22067847_ShreyashBasnet Last Checkpoint: 4 minutes ago (unsaved changes)

In [30]:

```
import matplotlib.pyplot as plt
import numpy as np

# Group the DataFrame by 'job_title' and calculate the average salary for each job
average_salary_per_job = df.groupby('job_title')['salary_in_usd'].mean()

# Sort the average salaries in descending order and select the top 15
top_jobs = average_salary_per_job.sort_values(ascending=False).head(15)

# Extract job titles and average salaries
job_titles = top_jobs.index.tolist()
average_salaries = top_jobs.values

# Create a bar plot
plt.figure(figsize=(10, 7))
plt.barh(job_titles, average_salaries, color='skyblue')
plt.xlabel('Average Salary (USD)')
plt.ylabel('Job Title')
plt.title('Average Salary by Job Title')
plt.gca().invert_yaxis()  # Invert y-axis to display the highest salary at the top
plt.show()
```

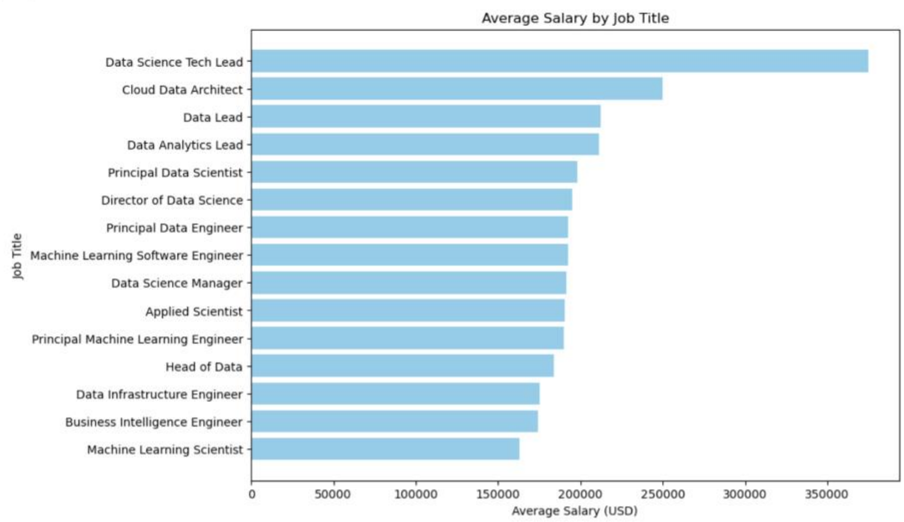


Figure 15 Job with highest salary in graph

The code provided yields a horizontal bar plot that illustrates the average income of the top 15 data science job titles. An description of each part is provided below:

1. Horizontal Bar Plot: - Matplotlib's barh() function is used to construct the plot; it creates horizontal bar plots. Every bar signifies the mean income for a certain job title.

2. Job Titles: - The job titles are plotted on the y-axis. Every bar is associated with a specific job title within the data science field. The job titles are taken out of the DataFrame and arranged in descending order of average income.

3. Mean Income (US Dollars): - The average wage in USD is displayed on the x-axis. The average pay linked to each job title is indicated by the length of each bar. The DataFrame is grouped by job title to get the mean salary for each job, which yields the average salaries.

4. Colour and Style: - The bars are sky blue in colour to improve aesthetics and visibility. - To add context and clarity, the plot includes labels for the x- and y-axes (Average Salary and Job Title), as well as a title (Average Salary by Job Title).

5. Inverted Y-Axis: - To show the job title with the greatest average income at the top of the plot, the y-axis is inverted using `plt.gca().invert_yaxis()}. This facilitates the process of visually identifying the highest-paying job titles.

Overall, this visualisation makes it easy to compare the average pay for various data science job titles quickly, which can assist stakeholders in identifying profitable career routes or areas where wage adjustments might be necessary.

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

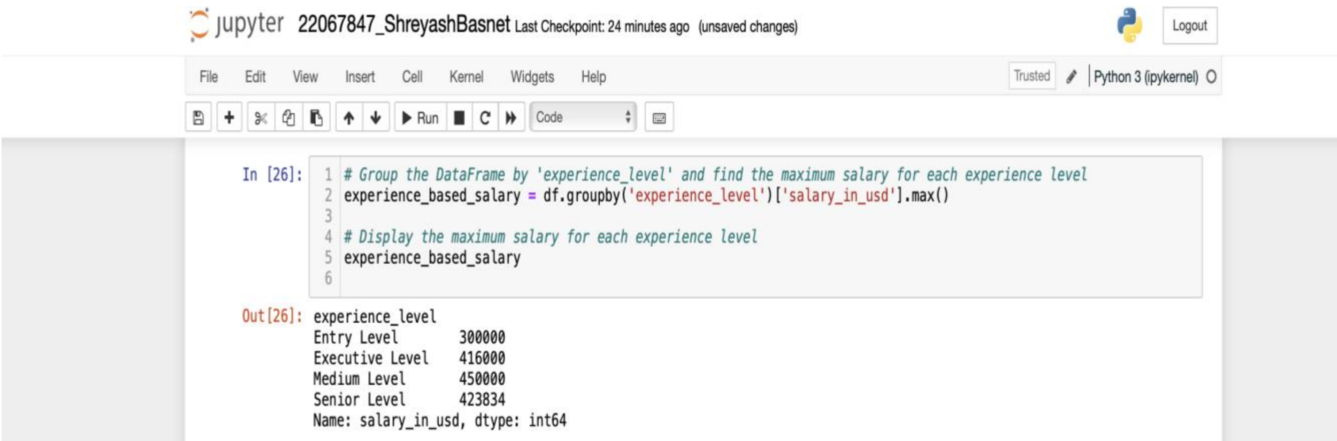


Figure 16 Program to find out salaries based on experience level

The code snippet's result computes the maximum salary for every experience level by grouping the DataFrame based on the 'experience_level' column. Here's why:

1. Grouping by 'experience_level': The DataFrame {df} is grouped by the values in the 'experience_level' column using the `groupby()} function. Through this process, the DataFrame is effectively divided into groups according to distinct values found in the 'experience_level' column.

2. Determining the Maximum Salary for Every Group: The'salary_in_usd' column in each group is subjected to an application of the `max()} function. This determines the highest salary linked to each experience level category, or the maximum salary value within each group.

3. Showing the Maximum compensation for Every Experience Level: The maximum compensation for every experience level is included in the resultant object, {experience_based_salary}. It appears as a Series or DataFrame with the maximum wages matching to the index 'experience_level'. This makes it easier to comprehend how salaries are distributed across various levels of seniority or expertise within the dataset by giving a clear picture of the greatest salary that may be earned within each experience level category.

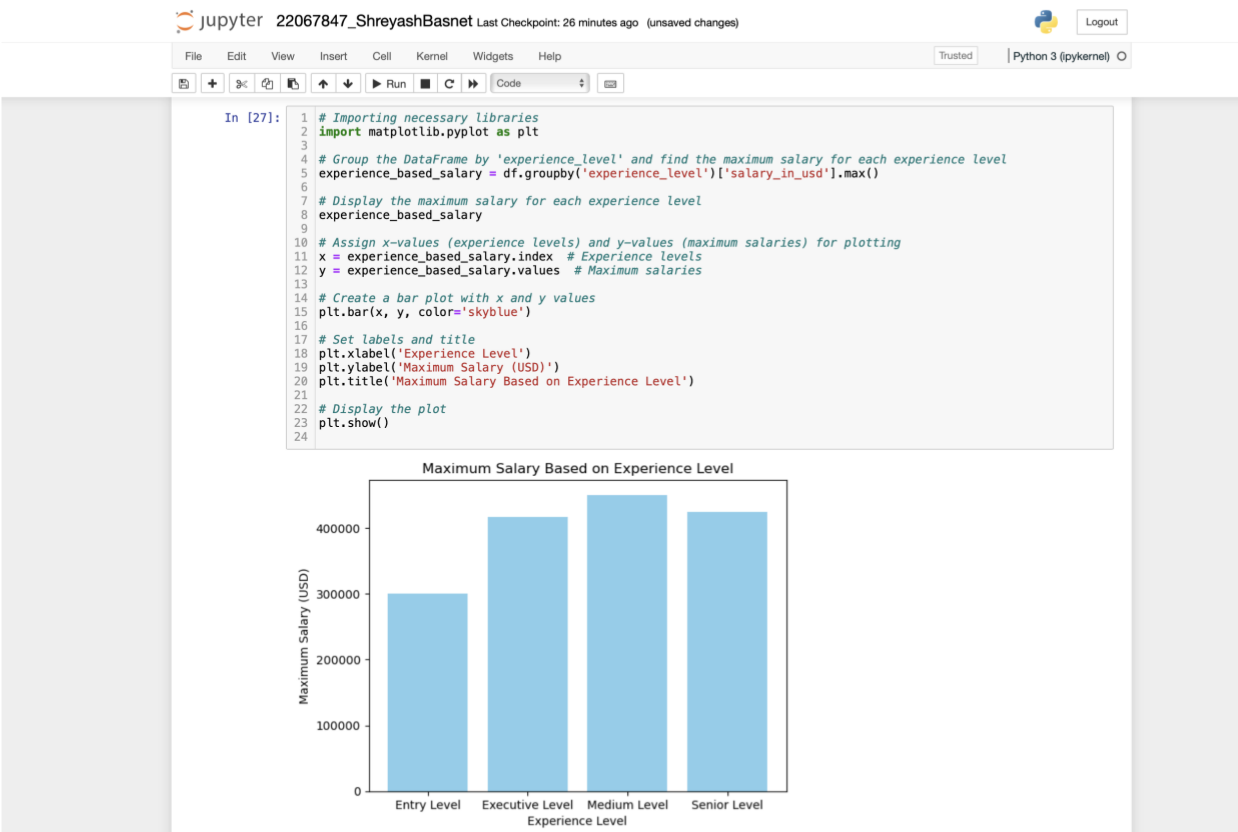


Figure 17 Program to find out salaries based on experience level. Illustrating it through a bar graph

The code that is given produces a bar plot that illustrates the highest pay for every experience level in the dataset. An description of each part is provided below:

1. Data processing: The 'experience_level' column is used to first group the DataFrame, and then the `groupby()` function and `max()` method are used to compute the maximum wage for each experience level.

2. Variables Assignment: The variable {x} is assigned the index of the `experience_based_salary` DataFrame (which corresponds to the experience levels), and the variable {y} is allocated the maximum salaries.

3. Bar Plot Creation: The plt.bar() function in Matplotlib is used to produce a bar plot. Arguments are passed in relation to the x-values (experience levels) and y-values (maximum salaries). 'skyblue' is the colour parameter set to differentiate the bars.

4. Labels and Title: The `plt.xlabel()} and `plt.ylabel()} functions are used to establish the labels for the x-axis ('Experience Level') and y-axis ('Maximum Salary (USD)'). 'Maximum Salary Based on Experience Level' is the plot title that is supplied using `plt.title()}.
5. Display: Using `plt.show()`, the plot is finally shown.

The maximum pay for each experience level is visually represented by the resulting bar plot, which makes it simple to compare and understand wage trends over a person's career in the data science industry.

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

jupyter 22067847_ShreyashBasnet Last Checkpoint: Last Thursday at 20:06 (autosaved)

In [17]:

```python
import matplotlib.pyplot as plt
import pandas as pd

# Read the CSV file into a pandas DataFrame
df = pd.read_csv("DataScienceSalaries_8b290669-a5e9-45bf-be72-d27add2eacae_93472_.csv")

# Function to display histogram and box plot for a chosen variable
def plot_variable(variable):
    plt.figure(figsize=(10, 4))

    # Histogram
    plt.subplot(1, 2, 1)
    plt.hist(df[variable], bins=10, color='skyblue', edgecolor='black')
    plt.xlabel(variable)
    plt.ylabel('Frequency')
    plt.title(f'Histogram of {variable}')

    # Box plot
    plt.subplot(1, 2, 2)
    plt.boxplot(df[variable], vert=False)
    plt.xlabel(variable)
    plt.title(f'Box plot of {variable}')

    plt.tight_layout()
    plt.show()

# Choose the variable you want to plot
chosen_variable = 'salary_in_usd'

# Call the function to plot the chosen variable
plot_variable(chosen_variable)
```

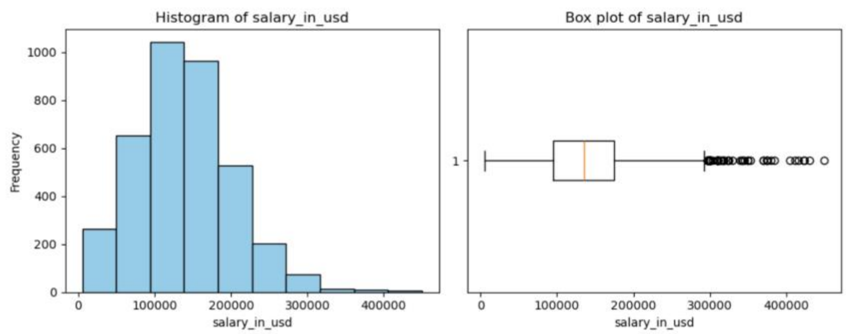


Figure 18 Program to show histogram and box plot of any chosen different variables in the graph.

Two visuals that show the distribution of salaries (in USD) in the dataset are produced by the accompanying Python code: a box plot and a histogram. What each visualisation shows is as follows:

1. Histogram: - The frequency distribution of salaries is depicted by the histogram on the left side of the figure. Salary values are plotted on the x-axis as bins (intervals), and the frequency (or count) of observations falling within each bin is plotted on the y-axis. The histogram's bars each show the quantity of data points that fall into a given pay range. In this instance, the histogram illustrates where the data tends to cluster and how it expands out, offering insights on the distribution of incomes over various ranges (Atlassian, 2024).

2. Box Plot: - Showing important statistical metrics like the median, quartiles, and possible outliers, the box plot on the right side of the figure offers a visual summary of the salary distribution. The median, also known as the second quartile, is shown by a horizontal line inside the interquartile range (IQR) box, which sits in the centre of the figure. The "whiskers" go from the box to the upper and lower bounds of a given range, usually 1.5 times the IQR. Outliers are defined as any points that are not within the whiskers. The box plot makes it possible to see the salary distribution's central tendency, variability, and possible skewness or asymmetry.
When combined, these visualisations offer contrasting viewpoints on how incomes are distributed throughout the dataset, allowing analysts to evaluate its features, spot trends, and find any oddities or inconsistencies (Khanacademy, 2024).

5. Conclusion

To sum up, the examination of the data science wage dataset has yielded significant understandings into the variables affecting pay in the industry. Numerous important conclusions have been drawn from the comprehension, preparation, analysis, and investigation of the data.

First, the dataset's features, including the several elements that can affect pay, like work level, job title, and experience level, were made clear during the data understanding step. This comprehension established the groundwork for additional examination. Python programmes were created to load the data into a pandas DataFrame, eliminate superfluous columns like salary and currency, manage missing values, look for duplicates, and investigate unique values in every field during the data preparation stage. To improve uniformity and clarity, experience level columns were also given new names that make sense. To give a quantitative picture of the dataset, summary statistics were computed for selected variables during the data analysis phase. To find possible linkages, correlations between the variables were also looked at.
The process of data exploration included ranking the top 15 jobs and using a bar graph to visualise them. It also entailed figuring out which job paid the most and using a bar graph to show it. Experience-level-based salary analysis and graphical presentation were also done. The technical report also included a brief user guide for each programme and included screenshots of testing and results along with snippets of Python code organised in an organised fashion.

Overall, this investigation has demonstrated expertise in Python programming, data processing, and visualisation techniques and has offered insightful information about the factors impacting earnings in the field of data science. The data science domain's workforce planning, talent recruiting, and wage negotiating procedures can all benefit from knowing about these findings while making decisions.

6. References:

KrebsOnSecutiry, 2015. *KrebsOnSecutiry.* [Online]
Available at: https://krebsonsecurity.com/2015/07/online-cheating-site-ashleymadison-hacked/
[Accessed 28 04 2024].
Media, A. L., 2016. *PR Newswire.* [Online]
Available at: https://www.prnewswire.com/news-releases/avid-life-media-rebrands-as-ruby---officially-drops-ashley-madison-life-is-short-have-an-affair-tagline-300297105.html
[Accessed 29 04 2024].
Anon., 2016. *Office of the Privacy Commisioner of Canada.* [Online]
Available at: https://www.priv.gc.ca/en/opc-actions-and-decisions/investigations/investigations-into-businesses/2016/pipeda-2016-005/
[Accessed 01 05 2024].
Commision, F. T., 2017. *Federal Trade Commision.* [Online]
Available at: https://www.ftc.gov/legal-library/browse/cases-proceedings/152-3284-ashley-madison
[Accessed 01 05 2024].
Lukic, D., 2020. *ID Strong.* [Online]
Available at: https://www.idstrong.com/sentinel/ashley-madison-data-breach/#:~:text=Due%20to%20the%20highly%20sensitive,hundreds%20of%20divorces%20and%20breakups.
[Accessed 01 05 2024].
Thornsen, D., 2015. *GRTherapyGroup.* [Online]
Available at: https://grandrapidstherapygroup.com/surviving-the-affair-ashley-madison/
[Accessed 01 05 2024].
ICMR, 2018. *ICMR.* [Online]
Available at:
https://www.icmrindia.org/casestudies/catalogue/Business%20Ethics/BECG161.htm#:~:text=Ashley%20Madison%20encouraged%20people%20to,meted%20out%20to%20customers%20justified%3F
[Accessed 01 05 2024].
Cultures, C. f. E. O., 2024. *Center for Ethical Organizational Cultures.* [Online]
Available at: https://harbert.auburn.edu/binaries/documents/center-for-ethical-organizational-cultures/cases/ashley-madison.pdf
[Accessed 02 05 2024].
Leader, B., 2015. *Burbank.* [Online]
Available at: https://www.latimes.com/socal/burbank-leader/opinion/tn-blr-in-theory-do-ashley-madison-hackers-have-the-moral-high-ground-20150901-story.html
[Accessed 02 05 2024].

Poe, 2024. *Poe.* [Online]
Available at: https://poe.com/p/What-are-the-major-issues-associated-with-the-Ashley-Madison-website#:~:text=Privacy%20Concerns%3A%20Ashley%20Madison's%20business,and%20search%20history%20%5B1%5D.
[Accessed 02 05 2024].
forbes, 2020. *forbes.* [Online]
Available at: https://www.forbes.com/sites/zakdoffman/2020/02/01/ashley-madison-hack-returns-to-haunt-its-victims-32-million-users-now-have-to-watch-and-wait/?sh=156b0c3b5677
[Accessed 02 05 2024].
Clapperton, 2015. *LinkedIn.* [Online]
Available at: https://www.linkedin.com/pulse/ashley-madison-how-manage-crisis-guy-clapperton-fpsa
[Accessed 02 05 2024].
Media, A. L., 2016. *PR Newswire.* [Online]
Available at: https://www.prnewswire.com/news-releases/avid-life-media-rebrands-as-ruby---officially-drops-ashley-madison-life-is-short-have-an-affair-tagline-300297105.html
[Accessed 28 04 2024].
KrebsOnSecurity, 2015. *KrebsOnSecurity.* [Online]
Available at: https://krebsonsecurity.com/2015/07/online-cheating-site-ashleymadison-hacked/
[Accessed 28 04 2024].
Atlassian, 2024. *Atlassian.* [Online]
Available at: https://www.atlassian.com/data/charts/histogram-complete-guide#:~:text=What%20is%20a%20histogram%3F,value%20within%20the%20corresponding%20bin.
[Accessed 08 05 2024].
Khanacademy, 2024. *Khanacademy.* [Online]
Available at: https://www.khanacademy.org/math/statistics-probability/summarizing-quantitative-data/box-whisker-plots/a/box-plot-review
[Accessed 08 05 2024].
policies, J. F. F. B. J. F. i. a. p. i. a. w. w. e. t. a. c. c. b. a. f. p. L. a. o. e., 2024. *Investopedia.* [Online]
Available at: https://www.investopedia.com/terms/c/correlationcoefficient.asp
[Accessed 06 05 2024].
KENTON, W., 2023. *Investopedia.* [Online]
Available at: https://www.investopedia.com/terms/k/kurtosis.asp
[Accessed 04 05 2024].
Turney, S., 2024. *Scribbr.* [Online]
Available at:
https://www.scribbr.com/statistics/skewness/#:~:text=Skewness%20is%20a%20measure%20of,negative)%2C%20or%20zero%20skewness.
[Accessed 05 05 2024].
MARSHALL HARGRAVE Full Bio Marshall Hargrave is a stock analyst and writer with 10+ years of experience covering stocks and markets, a. w. a. a. a. v. c. L. a. o. e. p., 2023. *Investopedia.*

[Online]
Available at: https://www.investopedia.com/terms/s/standarddeviation.asp
[Accessed 05 05 2024].
BYJU's, 2024. *BYJU's.* [Online]
Available at: https://byjus.com/maths/mean/
[Accessed 01 05 2024].