

Osama Marie 201911195

Mays Qasem 201910019

Salary dataset based on country and race(Salary and Demographic Information with Experience)

➤ The data mining problem and about data set:

The dataset consists of a comprehensive collection of salary and demographic information with additional details on years of experience. It offers a valuable resource for studying the relationship between income and various socio-demographic factors. The demographic attributes include age, gender, education, country, and race, providing a diverse range of variables for analysis. Researchers can explore patterns and trends in income distribution across different demographic categories, allowing for insights into potential disparities or variations in earning potential. Moreover, the dataset incorporates the crucial dimension of years of experience, enabling investigations into the impact of professional tenure on salary levels. This aspect adds a dynamic aspect to the analysis, enabling researchers to examine how income varies based on both demographic characteristics and accumulated work experience. The dataset presents a rich opportunity for conducting comprehensive studies on income diversity and understanding the multifaceted factors influencing earning potential in today's workforce.

The dataset includes various columns providing insightful information about individuals. It consists of an index column for unique identification, age for age of the individuals, gender for their gender identification, job title for their respective occupations, years of experience representing the professional tenure, salary denoting their income levels, country indicating the country of residence, and race for their racial background. This dataset offers a comprehensive view of individuals' demographic characteristics, professional attributes, and income levels, allowing researchers to explore relationships between age, gender, job title, experience, salary, country, and race in their analyses.

➤ Tool used:

Python

➤ **Column analysis:**

- **Id:** A unique identifier for each entry in the dataset Note: this not consider attribute but we mention it because it is in our data set.
- **Age:** The age of the individuals in the dataset, representing their chronological age in years.
- **Gender:** The gender identification of the individuals, indicating their gender or gender identity.
- **Education level:** The highest level of education attained by the individuals, indicating their educational qualifications or degree.
- **Job title:** The occupation or job title of the individuals, specifying their professional role or position.
- **Years of experience:** The number of years of professional experience accumulated by the individuals in their respective fields.
- **Salary:** The income level or salary earned by the individuals, denoting their monetary compensation.
- **Country:** The country of residence or origin of the individuals, providing geographical information.
- **Race:** The racial background or ethnicity of the individuals, reflecting their specific racial or ethnic group.

➤ **Number of:**

- Number of attributes :9 attributes
- Number of observations: 6706.

➤ **Types of attributes:**

- **id:** numeric(int) this not consider attribute but we mention it because it is in our data set.
- **Age:** numeric(int)
- **gender:** nominal(chr)

- Education level: nominal(chr)
- Job Title: nominal(chr)
- Years of experience: numeric(int)
- Salary: numeric(int)
- Country: nominal(chr)
- Race: nominal(chr)

➤ **link for the dataset:**

<https://www.kaggle.com/datasets/sudheerp2147234/salary-dataset-based-on-country-and-race>

➤ **problems with this dataset:**
in our data set there is many problems:

- Missing values
- Duplicated observations.
- Outliers
- Inconsistent in class attribute

➤ **Problem definition:**

- Based on the following, there are problems with our dataset, starting with the missing data, which will be treated by replacing it with the most frequent data, because it is present in fields containing text, and the number of missing data does not allow us to delete it.
- Outliers: first the percentage of outliers in age is very low so we remove the outliers, and when it is height we replace it by the mean.
- Inconsistent in class attribute: replace the wrong value with another one.
- Duplicate data we remove it.

➤ Data Description:

```
1 import pandas as pd
2 # read dataset
3 data = pd.read_csv("dataset.csv")
4 df = pd.DataFrame(data)
5 pd.set_option('display.max_columns', None)
6 data = []
7 for column in df.columns:
8     data.append([df[column].dtype])
9 dict = {'Column Name': list(df.columns),
10        'Data Type': data}
11 print(pd.DataFrame(dict))
12
```

Run: main

	Column Name	Data Type
0	Unnamed: 0	[int64]
1	Age	[float64]
2	Gender	[object]
3	Education Level	[object]
4	Job Title	[object]
5	Years of Experience	[float64]
6	Salary	[float64]
7	Country	[object]
8	Race	[object]

➤ Here number of missing value in all data set:

```
8 data = pd.read_csv("dataset.csv")
9 df = pd.DataFrame(data)
10 pd.set_option('display.max_columns', None)
11 print(f"Number of missing value in dataset is :{sum(list(df.isna().sum()))}")
```

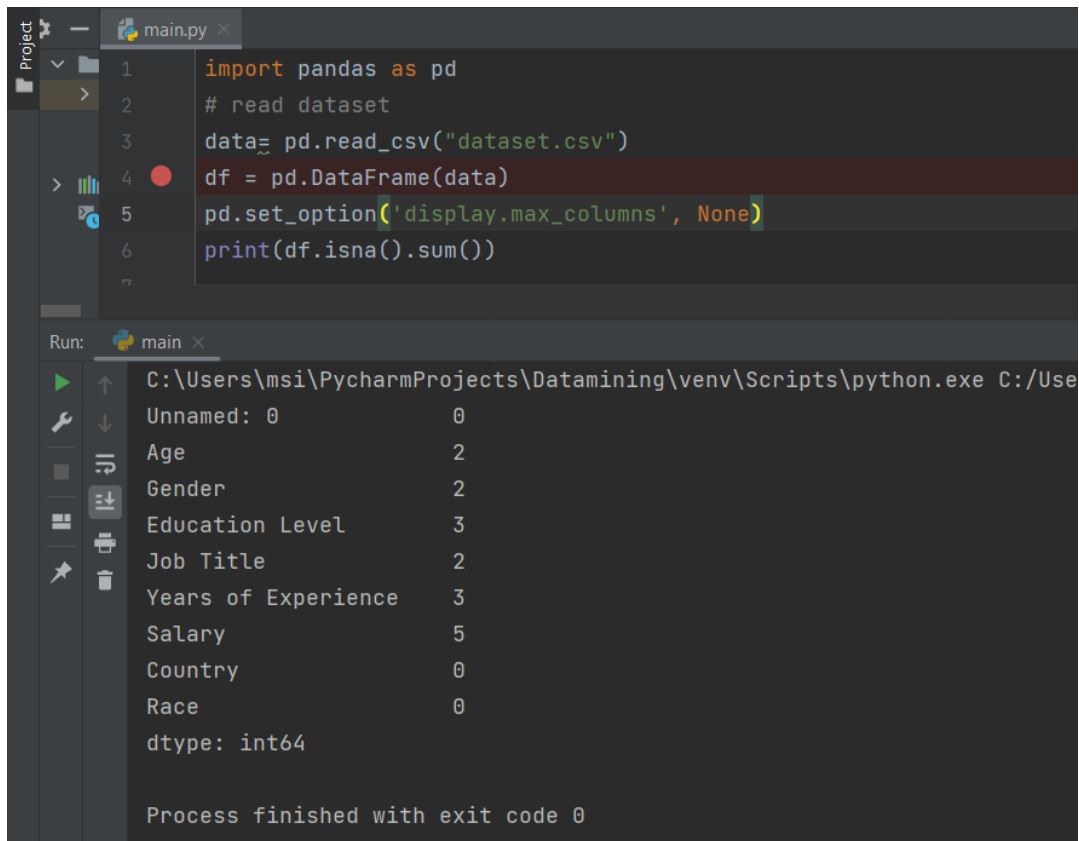
Run: main

C:\Users\msi\PycharmProjects\Datamining\venv\Scripts\python.exe C:/Users/msi/PycharmP

Number of missing value in dataset is :17

Process finished with exit code 0

➤ Here number of missing value in each column:



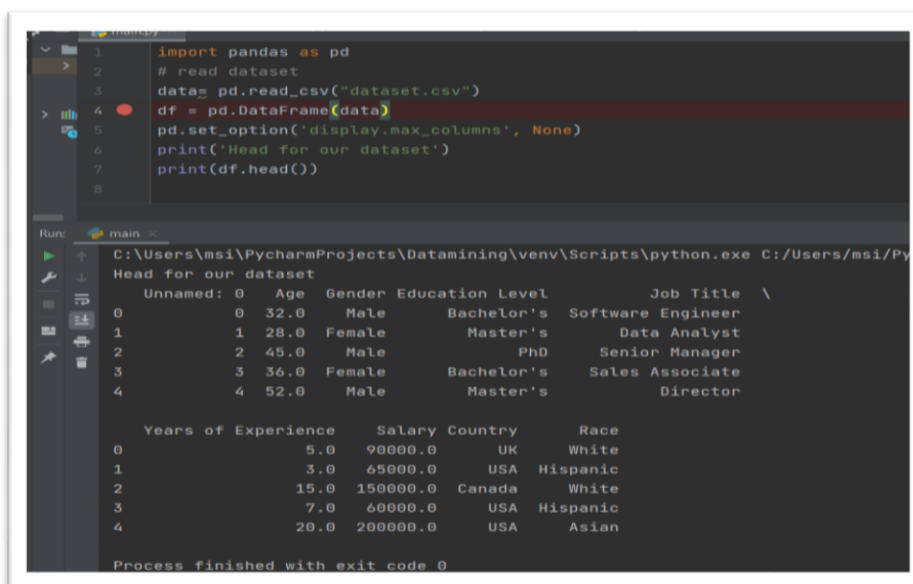
```
1 import pandas as pd
2 # read dataset
3 data = pd.read_csv("dataset.csv")
4 df = pd.DataFrame(data)
5 pd.set_option('display.max_columns', None)
6 print(df.isna().sum())
```

Run: main

```
C:\Users\msi\PycharmProjects\Datamining\venv\Scripts\python.exe C:/Use
Unnamed: 0      0
Age             2
Gender          2
Education Level  3
Job Title       2
Years of Experience  3
Salary          5
Country         0
Race            0
dtype: int64

Process finished with exit code 0
```

➤ Here is the head of data set:



```
1 import pandas as pd
2 # read dataset
3 data = pd.read_csv("dataset.csv")
4 df = pd.DataFrame(data)
5 pd.set_option('display.max_columns', None)
6 print('Head for our dataset')
7 print(df.head())
```

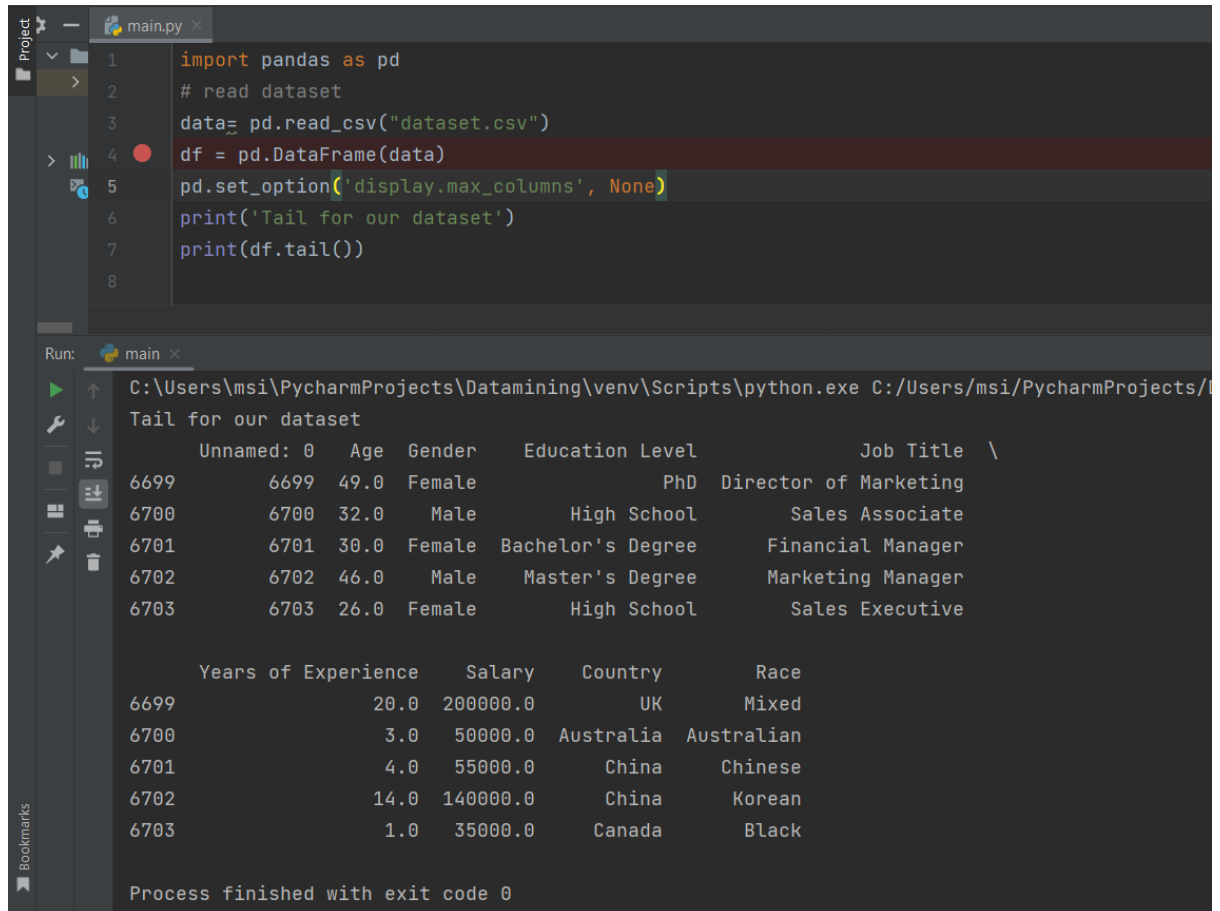
Run: main

```
C:\Users\msi\PycharmProjects\Datamining\venv\Scripts\python.exe C:/Users/msi/Py
Head for our dataset
  Unnamed: 0  Age  Gender Education Level  Job Title \
0          0  32.0   Male      Bachelor's Software Engineer
1          1  28.0  Female      Master's   Data Analyst
2          2  45.0   Male         PhD   Senior Manager
3          3  36.0  Female      Bachelor's Sales Associate
4          4  52.0   Male      Master's      Director

  Years of Experience  Salary  Country  Race
0          5.0  90000.0    UK  White
1          3.0  65000.0   USA  Hispanic
2         15.0 150000.0  Canada  White
3          7.0  60000.0   USA  Hispanic
4         20.0 200000.0   USA   Asian

Process finished with exit code 0
```

➤ Here is the tail of data set:



The screenshot shows the PyCharm IDE with a file named `main.py` open. The code in the editor is as follows:

```
1 import pandas as pd
2 # read dataset
3 data = pd.read_csv("dataset.csv")
4 df = pd.DataFrame(data)
5 pd.set_option('display.max_columns', None)
6 print('Tail for our dataset')
7 print(df.tail())
8
```

Below the editor, the Run console shows the execution of the script. The output is:

```
Run: main x
C:\Users\msi\PycharmProjects\Datamining\venv\Scripts\python.exe C:/Users/msi/PycharmProjects/
Tail for our dataset
```

	Unnamed: 0	Age	Gender	Education Level	Job Title	
6699	6699	49.0	Female	PhD	Director of Marketing	
6700	6700	32.0	Male	High School	Sales Associate	
6701	6701	30.0	Female	Bachelor's Degree	Financial Manager	
6702	6702	46.0	Male	Master's Degree	Marketing Manager	
6703	6703	26.0	Female	High School	Sales Executive	

	Years of Experience	Salary	Country	Race
6699	20.0	200000.0	UK	Mixed
6700	3.0	50000.0	Australia	Australian
6701	4.0	55000.0	China	Chinese
6702	14.0	140000.0	China	Korean
6703	1.0	35000.0	Canada	Black

Process finished with exit code 0

➤ Here is the diminution of data set:

```
1 import pandas as pd
2 # read dataset
3 data= pd.read_csv("dataset.csv")
4 df = pd.DataFrame(data)
5 pd.set_option('display.max_columns', None)
6
7 print(f"dimensions for dataset{df.shape}")
8
```

Run: main

```
C:\Users\msi\PycharmProjects\Datamining\venv\Scripts\python.exe C:/Users/msi/
dimensions for dataset(6704, 9)

Process finished with exit code 0
```

- **Description for numeric attribute (to show five-point summary):**
25% indicate to Q1, 50% indicate to Q2 and 75% indicate to Q3

```
1 import pandas as pd
2 # read dataset
3 data= pd.read_csv("dataset.csv")
4 df = pd.DataFrame(data)
5 pd.set_option('display.max_columns', None)
6 print('Print Description for dataset as mead count std min and so on-->')
7 print(df.describe())
8
```

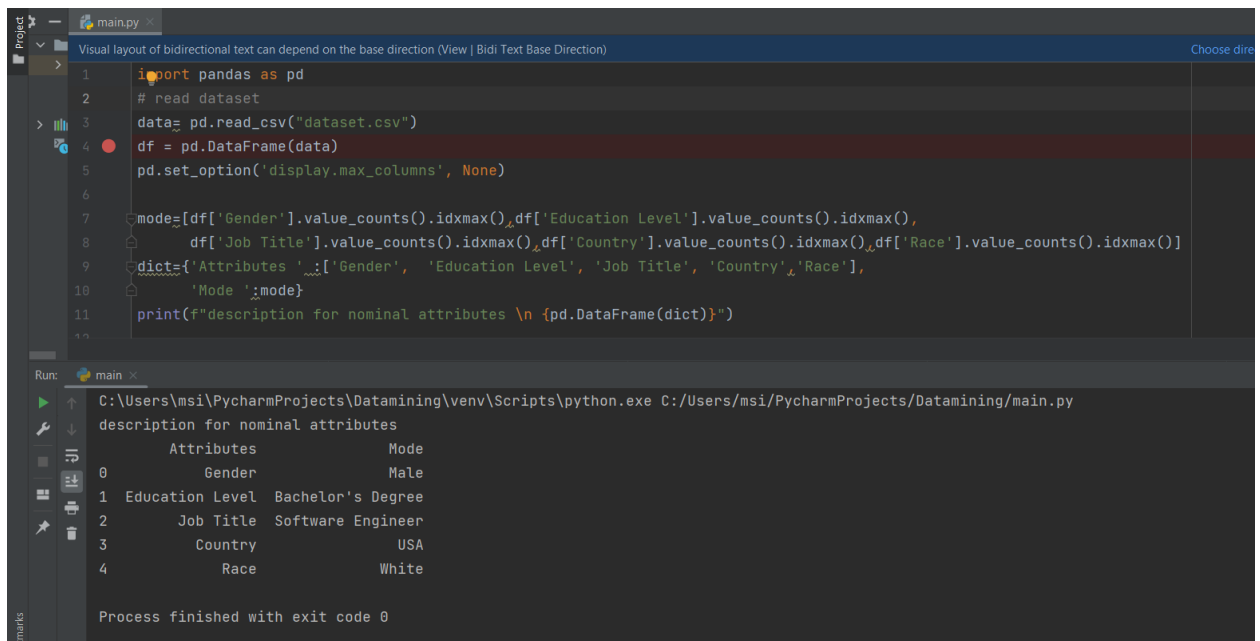
Run: main

```
C:\Users\msi\PycharmProjects\Datamining\venv\Scripts\python.exe C:/Users/msi/PycharmProjects/Datamining/main.py
Print Description for dataset as mead count std min and so on-->

   Unnamed: 0      Age  Years of Experience      Salary
count  6704.000000  6702.000000      6701.000000    6699.000000
mean   3351.500000   33.620859       8.094687   115326.964771
std    1935.422435    7.614633       6.059003    52786.183911
min      0.000000   21.000000       0.000000     350.000000
25%    1675.750000   28.000000       3.000000    70000.000000
50%    3351.500000   32.000000       7.000000   115000.000000
75%    5027.250000   38.000000      12.000000   160000.000000
max    6703.000000   62.000000      34.000000  250000.000000

Process finished with exit code 0
```

- **Description for nominal attribute:**
Because its nominal we only can compute the mod



The screenshot shows a PyCharm IDE with a Python script named `main.py` and its execution output in the Run console.

Code in `main.py`:

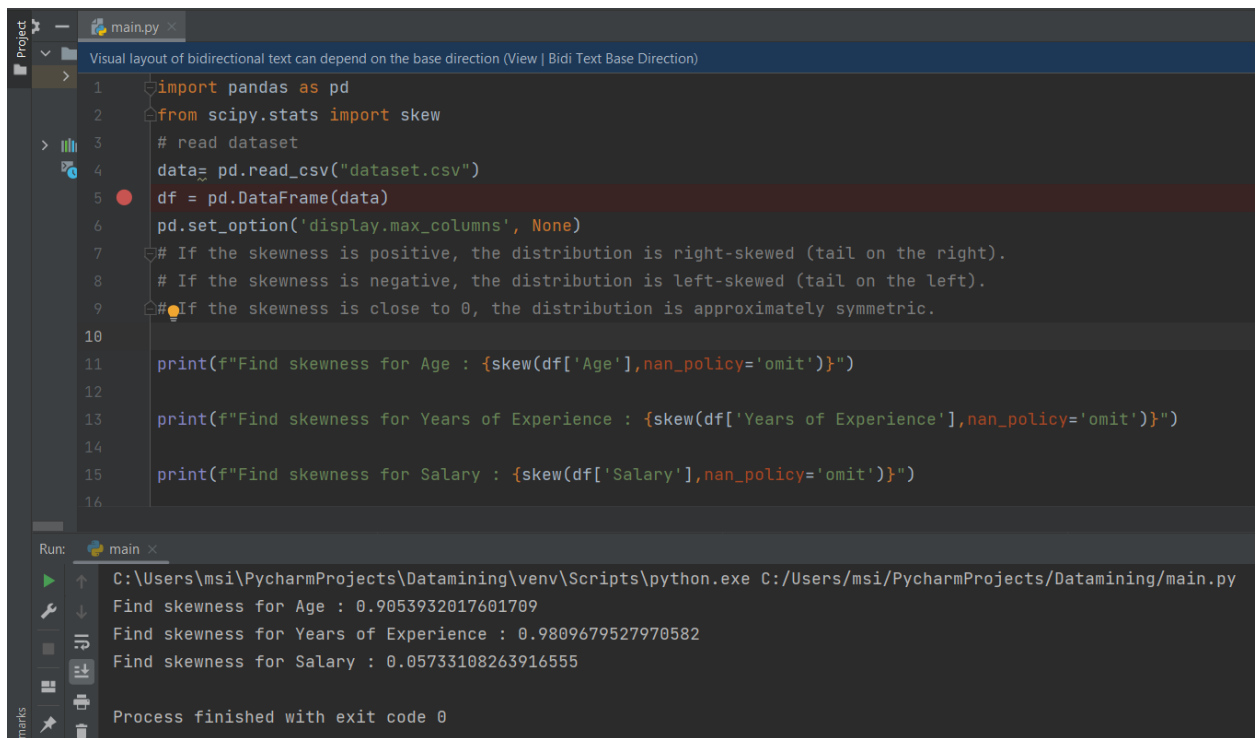
```
1 import pandas as pd
2 # read dataset
3 data = pd.read_csv("dataset.csv")
4 df = pd.DataFrame(data)
5 pd.set_option('display.max_columns', None)
6
7 mode=[df['Gender'].value_counts().idxmax(), df['Education Level'].value_counts().idxmax(),
8       df['Job Title'].value_counts().idxmax(), df['Country'].value_counts().idxmax(), df['Race'].value_counts().idxmax()]
9 dict={'Attributes ': ['Gender', 'Education Level', 'Job Title', 'Country', 'Race'],
10      'Mode ': mode}
11 print(f"description for nominal attributes \n {pd.DataFrame(dict)}")
```

Run console output:

```
C:\Users\msi\PycharmProjects\Datamining\venv\Scripts\python.exe C:/Users/msi/PycharmProjects/Datamining/main.py
description for nominal attributes
  Attributes      Mode
0      Gender      Male
1 Education Level Bachelor's Degree
2      Job Title Software Engineer
3      Country      USA
4      Race      White

Process finished with exit code 0
```

- If the skewness is positive, the distribution is right-skewed (tail on the right).
- If the skewness is negative, the distribution is left-skewed (tail on the left).
- If the skewness is close to 0, the distribution is approximately symmetric.



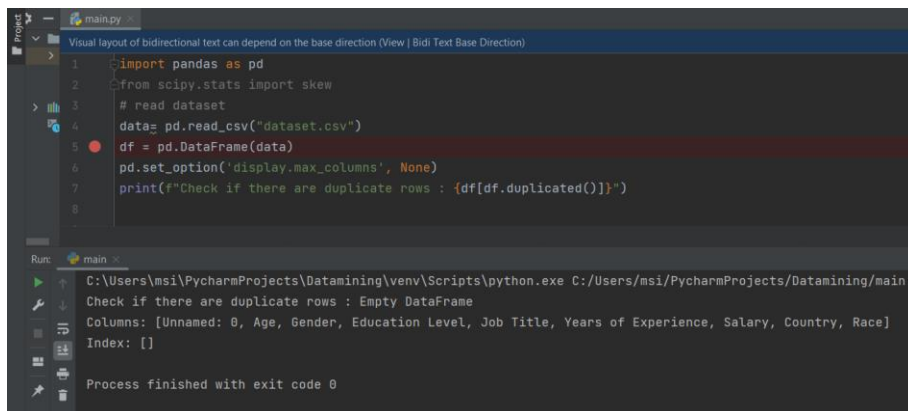
```
1 import pandas as pd
2 from scipy.stats import skew
3 # read dataset
4 data = pd.read_csv("dataset.csv")
5 df = pd.DataFrame(data)
6 pd.set_option('display.max_columns', None)
7 # If the skewness is positive, the distribution is right-skewed (tail on the right).
8 # If the skewness is negative, the distribution is left-skewed (tail on the left).
9 # If the skewness is close to 0, the distribution is approximately symmetric.
10
11 print(f"Find skewness for Age : {skew(df['Age'], nan_policy='omit')}")
12
13 print(f"Find skewness for Years of Experience : {skew(df['Years of Experience'], nan_policy='omit')}")
14
15 print(f"Find skewness for Salary : {skew(df['Salary'], nan_policy='omit')}")
16
```

Run: main

```
C:\Users\msi\PycharmProjects\Datamining\venv\Scripts\python.exe C:/Users/msi/PycharmProjects/Datamining/main.py
Find skewness for Age : 0.9053932017601709
Find skewness for Years of Experience : 0.9809679527970582
Find skewness for Salary : 0.05733108263916555
Process finished with exit code 0
```

➤ Data Cleaning:

- We Find if there is any duplicate in dataset (if there is we must remove it) As we note no duplicate in our data set:



```
1 import pandas as pd
2 from scipy.stats import skew
3 # read dataset
4 data = pd.read_csv("dataset.csv")
5 df = pd.DataFrame(data)
6 pd.set_option('display.max_columns', None)
7 print(f"Check if there are duplicate rows : {df[df.duplicated()]}")
8
```

Run: main

```
C:\Users\msi\PycharmProjects\Datamining\venv\Scripts\python.exe C:/Users/msi/PycharmProjects/Datamining/main.py
Check if there are duplicate rows : Empty DataFrame
Columns: [Unnamed: 0, Age, Gender, Education Level, Job Title, Years of Experience, Salary, Country, Race]
Index: []
Process finished with exit code 0
```

- We make cleaning for missing nominal by compute the mod and replace missing with mod:
 - For Gender: the mod equal male

```
102 # For Gender Attributes
103 print(f"Before Cleaning {df['Gender'].isna().sum()}")
104 frequent_Value=df['Gender'].value_counts().idxmax()
105 for idx in range(len(df)):
106     if df['Gender'][idx] != 'Male' and df['Gender'][idx] != 'Female':
107         df.loc[idx, 'Gender']=frequent_Value
108 print(f"After Cleaning all values is right {df['Gender'].isna().sum()}")
109
110
```

Run: main ×

C:\Users\msi\PycharmProjects\Datamining\venv\Scripts\python.exe C:/Users/msi/Pycharm
Before Cleaning 2
After Cleaning all values is right 0

➤ **For Education level: mode equal bachelor's degree**

```
109
110 # For Education Level Attributes
111
112 print(df['Education Level'].isna().sum())
113 most_frequent=df['Education Level'].value_counts().idxmax()
114 print(most_frequent)
115 df['Education Level']=df['Education Level'].fillna(most_frequent)
116 print(df['Education Level'].isna().sum())
117
```

Run: main ×

C:\Users\msi\PycharmProjects\Datamining\venv\Scripts\python.exe C:/Users
3
Bachelor's Degree
0

➤ **For Race and country as we mentioned previously there is no missing data.**

➤ **For job title:**

```
110 #For Job Title Attributes
111
112 print(df['Job Title'].isna().sum())
113 most_frequent=df['Job Title'].value_counts().idxmax()
114 print(most_frequent)
115 df['Job Title']=df['Job Title'].fillna(most_frequent)
116 print(df['Job Title'].isna().sum())
117
```

Run: main

```
C:\Users\msi\PycharmProjects\Datamining\venv\Scripts\python.exe C:/Users/msi/Py
2
Software Engineer
0
```

- We make cleaning for missing numeric by compute the mean and replace missing with mean:
 - For Age the min value is 21 and max 62 and it is acceptable values and we calculate the mean and replace missing by it:

```
31
32 # cleaning Data
33 # first Age Check if any data illogical for Age attributes
34 print(f"Max Age is :{df['Age'].max()}")
35 print(f"Min Age is :{df['Age'].min()}")
36 print(f"number of missing value :{df['Age'].isna().sum()}") # the result is 2 must fill null value with mean
37 df['Age']=df['Age'].fillna(df['Age'].mean())
38 print(df['Age'].isna().sum())
39
```

Run: main

```
C:\Users\msi\PycharmProjects\Datamining\venv\Scripts\python.exe C:/Users/msi/PycharmProjects/Datamining/main.py
Max Age is :62.0
Min Age is :21.0
number of missing value :2
0
Process finished with exit code 0
```

- For years of experience the min value is 0 and max 34 and it is acceptable values and we calculate the mean and replace missing here there is 3 values missing by it:

```
110 # For Years of Experience Attributes
111 # from description max value is 34 and min 0
112 print(df['Years of Experience'].isna().sum())
113 df['Years of Experience']=df['Years of Experience'].fillna(df['Years of Experience'].mean())
114 print(df['Country'].isna().sum())
115
116
```

Run: main x

C:\Users\msi\PycharmProjects\Datamining\venv\Scripts\python.exe C:/Users/msi/PycharmProjects/Datamining/

3

0

Process finished with exit code 0

- For salary the min is 350 and max is 250000 and it is acceptable values and we calculate the mean and replace missing here there is 5 values missing by it:

```
110 # For Salary Attributes
111 # from description max value is 250000 and min 350
112 print(df['Salary'].isna().sum())
113 df['Salary']=df['Salary'].fillna(df['Salary'].mean())
114 print(df['Salary'].isna().sum())
115
```

Run: main x

C:\Users\msi\PycharmProjects\Datamining\venv\Scripts\python.exe C:/Users/msi/PycharmProjects/Datamining/main.py

5

0

- Cleaning duplicate by remove it:

```
120
121 print("Duplicate rows:")
122 print(f"duplicated for all record: {len(df[df.duplicated()])}")
123 print(f"duplicated based on keyId: {len(df[df.duplicated(subset=['Unnamed: 0'])])}")_# based on keyId
124
```

Run: main x

C:\Users\msi\PycharmProjects\Datamining\venv\Scripts\python.exe C:/Users/msi/PycharmProjects/Datamining/main.py

Duplicate rows:

duplicated for all record: 0

duplicated based on keyId: 0

- Box blot:

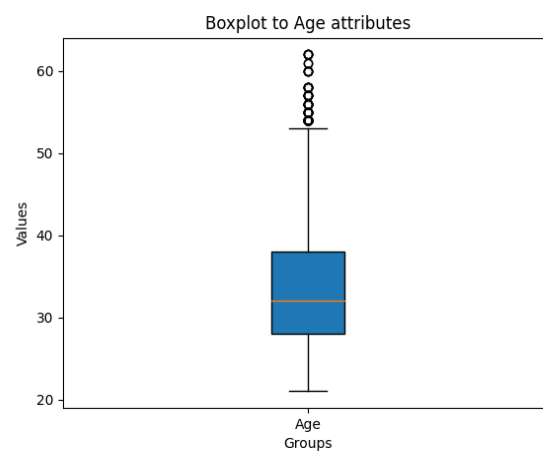
- For Age:

- We find the outlier and the first outlier is 54 as shown

- The number of outlier is very small so we remove it.

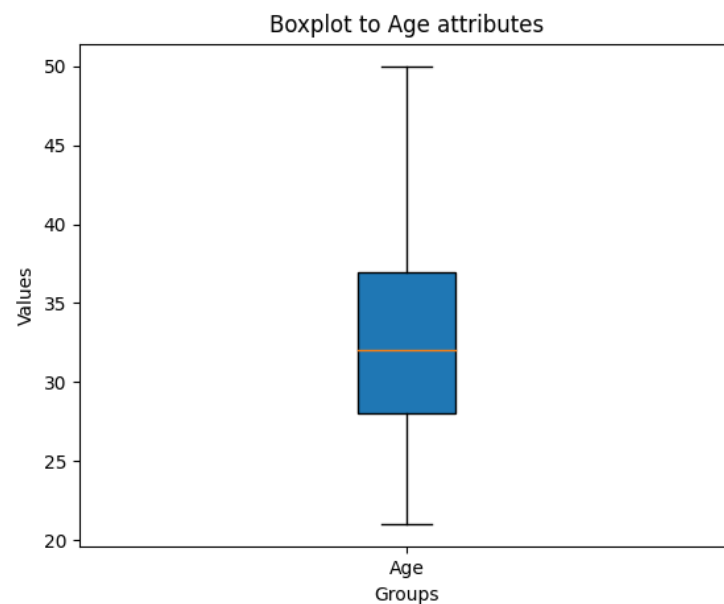
```
Datamining main.py
main.py
Visual layout of bidirectional text can depend on the base direction (View | Bidi Text Base Direction)
Q: 0 results
46
47 df_age={'Age':list(df['Age'])}
48 df_age=pd.DataFrame(df_age)
49 # Calculate the IQR
50 Q1 = df_age['Age'].quantile(0.25)
51 Q3 = df_age['Age'].quantile(0.75)
52 IQR = Q3 - Q1
53 # Define the lower and upper bounds for outliers
54 lower_bound = Q1 - 1.5 * IQR
55 upper_bound = Q3 + 1.5 * IQR
56 # Find outliers
57 outliers = df_age[(df_age['Age'] < lower_bound) | (df_age['Age'] > upper_bound)]
58 print(min(list(outliers['Age'])))
59 plt.boxplot(df_age.values, labels=df_age.columns, patch_artist=True)
60 plt.title("Boxplot to Age attributes")
61 plt.ylabel("Values")
62 plt.xlabel("Groups")
63 plt.show()

Run: main
C:\Users\msi\PycharmProjects\Datamining\venv\Scripts\python.exe C:/Users/msi/PycharmProjects/Datamining/ms
54.0
```



➤ After remove outlier for age:

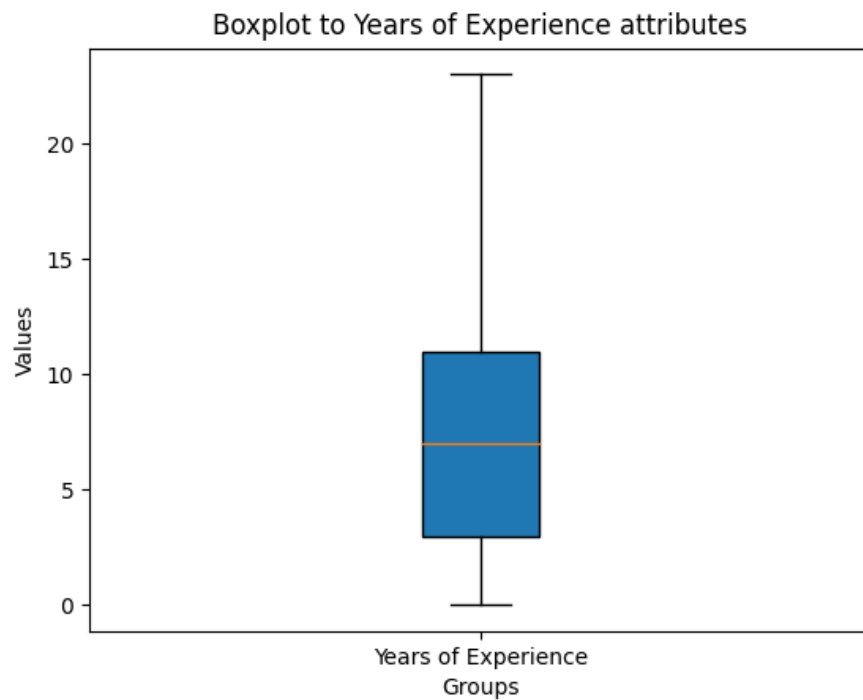
```
125 # Calculate the IQR
126 Q1 = df['Age'].quantile(0.25)
127 Q3 = df['Age'].quantile(0.75)
128 IQR = Q3 - Q1
129 # Define the lower and upper bounds for outliers
130 lower_bound = Q1 - 1.5 * IQR
131 upper_bound = Q3 + 1.5 * IQR
132 # Find outliers
133 outliers = df[(df['Age'] < lower_bound) | (df['Age'] > upper_bound)]
134 print(len(list(outliers['Age']))) # 123 outliers
135 condition = df['Age'].isin(outliers['Age'])
136 df = df[~condition] # remove outliers
137 outliers = df[(df['Age'] < lower_bound) | (df['Age'] > upper_bound)]
138 print(len(outliers)) # zero outliers
139 plt.boxplot(df['Age'].values, labels=['Age'], patch_artist=True, showfliers=False)
140 plt.title("Boxplot to Age attributes")
141 plt.ylabel("Values")
142 plt.xlabel("Groups")
143 plt.show()
```



➤ For years of experience after remove outliers:

```
141 Q1 = df['Years of Experience'].quantile(0.25)
142 Q3 = df['Years of Experience'].quantile(0.75)
143 IQR = Q3 - Q1
144 # Define the lower and upper bounds for outliers
145 lower_bound = Q1 - 1.5 * IQR
146 upper_bound = Q3 + 1.5 * IQR
147 # Find outliers
148 outliers = df[(df['Years of Experience'] < lower_bound) | (df['Years of Experience'] > upper_bound)]
149 print(f"number of outliers before cleaning: {len(list(outliers['Age']))}") # 123 outliers
150 condition = df['Years of Experience'].isin(outliers['Years of Experience'])
151 df = df[~condition] # remove outliers
152 outliers = df[(df['Years of Experience'] < lower_bound) | (df['Years of Experience'] > upper_bound)]
153 print(f"number of outliers after cleaning: {len(outliers)}") # zero outliers
154 plt.boxplot(df['Years of Experience'].values, labels=['Years of Experience'], patch_artist=True, showfliers=False)
155 plt.title("Boxplot to Years of Experience attributes")
156 plt.ylabel("Values")
157 plt.xlabel("Groups")
158 plt.show()
```

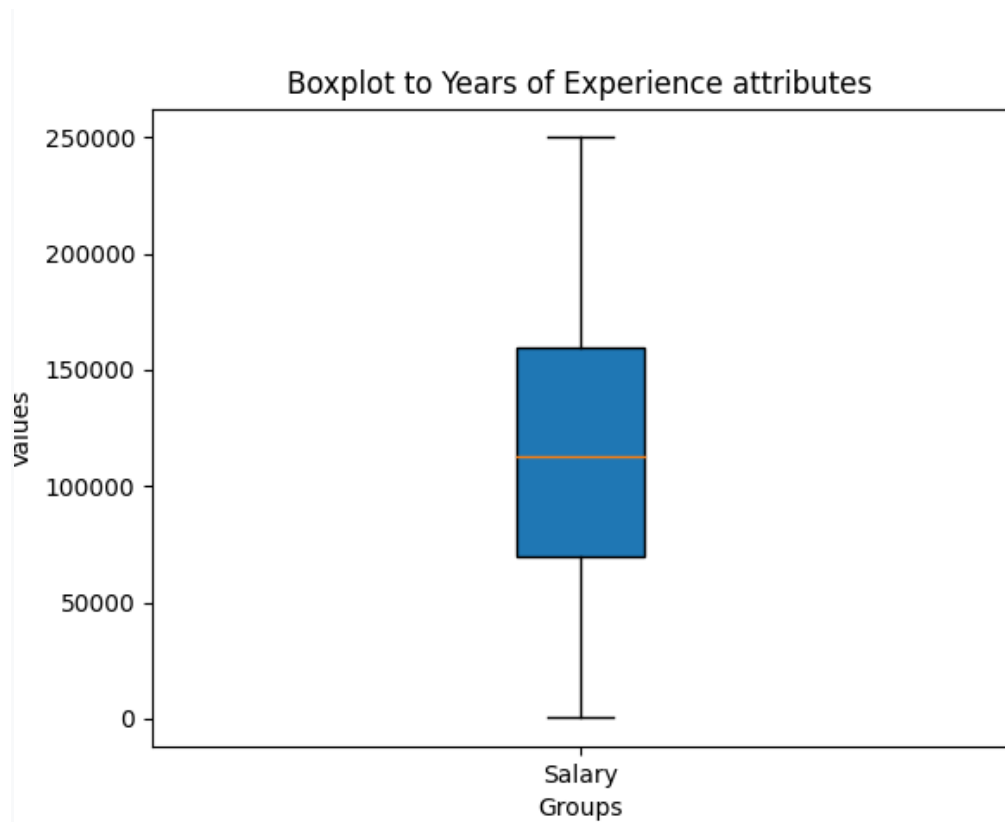
Run: C:\Users\msi\PycharmProjects\Datamining\venv\Scripts\python.exe C:\Users\msi\PycharmProjects\Datamining\main.py
number of outliers before cleaning: 25
number of outliers after cleaning : 0



➤ For salary after remove outliers:

```
main.py
Visual layout of bidirectional text can depend on the base direction (View | Bidi Text Base Direction)
Q- 0 results
156 Q1 = df['Salary'].quantile(0.25)
157 Q3 = df['Salary'].quantile(0.75)
158 IQR = Q3 - Q1
159 # Define the lower and upper bounds for outliers
160 lower_bound = Q1 - 1.5 * IQR
161 upper_bound = Q3 + 1.5 * IQR
162 # Find outliers
163 outliers = df[(df['Salary'] < lower_bound) | (df['Salary'] > upper_bound)]
164 print(f"number of outliers before cleaning: {len(list(outliers['Salary']))}") # 0 outliers
165 plt.boxplot(df['Salary'].values, labels=['Salary'], patch_artist=True, showfliers=False)
166 plt.title("Boxplot to Years of Experience attributes")
167 plt.ylabel("Values")
168 plt.xlabel("Groups")
169 plt.show()
170

Run: main
C:\Users\msi\PycharmProjects\Datamining\venv\Scripts\python.exe C:/Users/msi/PycharmProjects/Datamini
number of outliers before cleaning: 0
```



- Dimension after cleaning:
Note in the page 6 we put the dimension before smoothing.

```
173
174     print(f"Dimention for data set after cleaning {df.shape}")
175
```

Run: main x

C:\Users\msi\PycharmProjects\Datamining\venv\Scripts\python.exe C:/Users/msi/PycharmProjects/Datamining/main.py

Dimention for data set after cleaning (6556, 9)

Process finished with exit code 0

- To determine the biasing, we compute the mod for every attribute
And the percentage and the highest percentage the data biasing it:
For gender:

```
185     mode_value = df['Gender'].mode().iloc[0]
186     count = (df['Gender'] == mode_value).sum()
187     print(f" The mode is :{mode_value} and the Freq is: {count}")
188     print(f"the Perc is:{count/df['Gender'].value_counts().sum()}")
189
190
```

Run: main x

C:\Users\msi\PycharmProjects\Datamining\venv\Scripts\python.exe C:/Users/msi/PycharmProjects/Datamining/main.py

The mode is :Male and the Freq is: 3576

the Perc is:0.5454545454545454

For job title:

```
185     mode_value = df['Job Title'].mode().iloc[0]
186     count = (df['Job Title'] == mode_value).sum()
187     print(f" The mode is :{mode_value} and the Freq is: {count}")
188     print(f"the Perc is:{count/df['Job Title'].value_counts().sum()}")
189
190
```

Run: main x

C:\Users\msi\PycharmProjects\Datamining\venv\Scripts\python.exe C:/Users/msi/PycharmProjects/Datamining/main.py

The mode is :Software Engineer and the Freq is: 520

the Perc is:0.07931665649786455

Process finished with exit code 0

For education level:

```
185 mode_value = df['Education Level'].mode().iloc[0]
186 count = (df['Education Level'] == mode_value).sum()
187 print(f" The mode is :{mode_value} and the Freq is: {count}")
188 print(f"the Perc is:{count/df['Education Level'].value_counts().sum()}")
189
190
```

Run: main ×

C:\Users\msi\PycharmProjects\Datamining\venv\Scripts\python.exe C:/Users/msi/PycharmProjects/Datamining/main.py

The mode is :Bachelor's Degree and the Freq is: 2257

the Perc is:0.3442647956070775

For country:

```
185 mode_value = df['Country'].mode().iloc[0]
186 count = (df['Country'] == mode_value).sum()
187 print(f" The mode is :{mode_value} and the Freq is: {count}")
188 print(f"the Perc is:{count/df['Country'].value_counts().sum()}")
189
190
```

Run: main ×

C:\Users\msi\PycharmProjects\Datamining\venv\Scripts\python.exe C:/Users/msi/PycharmProjects/Datamining/main.py

The mode is :USA and the Freq is: 1335

the Perc is:0.20363026235509457

For Race:

```
185 mode_value = df['Race'].mode().iloc[0]
186 count = (df['Race'] == mode_value).sum()
187 print(f" The mode is :{mode_value} and the Freq is: {count}")
188 print(f"the Perc is:{count/df['Race'].value_counts().sum()}")
189
190
```

Run: main ×

C:\Users\msi\PycharmProjects\Datamining\venv\Scripts\python.exe C:/Users/msi/PycharmProjects/Datamining/main.py

The mode is :White and the Freq is: 1923

the Perc is:0.2933190970103722

For Salary:

```
185 mode_value = df['Salary'].mode().iloc[0]
186 count = (df['Salary'] == mode_value).sum()
187 print(f" The mode is :{mode_value} and the Freq is: {count}")
188 print(f"the Perc is:{count/df['Salary'].value_counts().sum()}")
189
190
```

Run: main ×

C:\Users\msi\PycharmProjects\Datamining\venv\Scripts\python.exe C:/Users/msi/PycharmProjects/Datamining/main.py

The mode is :140000.0 and the Freq is: 287

the Perc is:0.04377669310555216

For Years of experience:

```
185 mode_value = df['Years of Experience'].mode().iloc[0]
186 count = (df['Years of Experience'] == mode_value).sum()
187 print(f" The mode is :{mode_value} and the Freq is: {count}")
188 print(f"the Perc is:{count/df['Years of Experience'].value_counts().sum()}")
189
190
```

Run: main ×

C:\Users\msi\PycharmProjects\Datamining\venv\Scripts\python.exe C:/Users/msi/PycharmProjects/Datamining/main.py

The mode is :2.0 and the Freq is: 613

the Perc is:0.09350213544844417

For Age:

```
185 mode_value = df['Age'].mode().iloc[0]
186 count = (df['Age'] == mode_value).sum()
187 print(f" The mode is :{mode_value} and the Freq is: {count}")
188 print(f"the Perc is:{count/df['Age'].value_counts().sum()}")
189
190
```

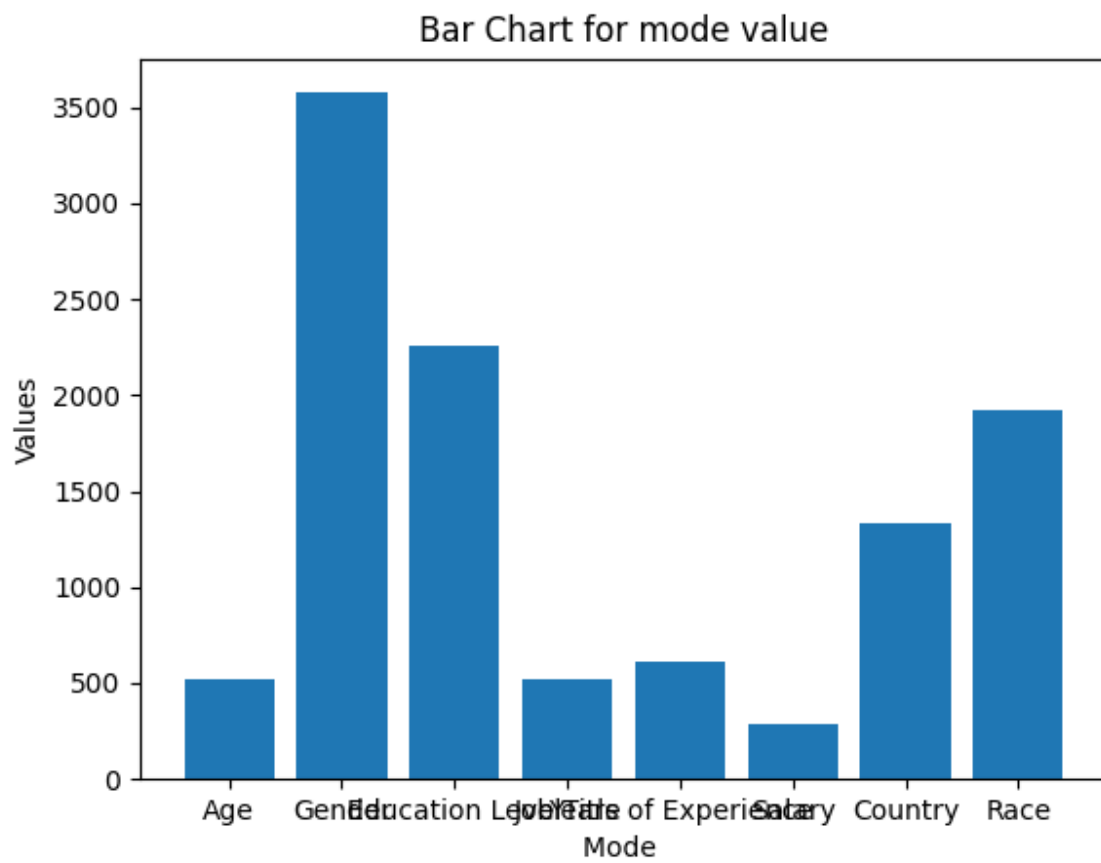
Run: main ×

C:\Users\msi\PycharmProjects\Datamining\venv\Scripts\python.exe C:/Users/msi/PycharmProjects/Datamining/main.py

The mode is :27.0 and the Freq is: 517

the Perc is:0.07885906040268456

As we notice the gender is most percentage so there is biasing to it (the data biasing to gender male). And the diagram represent that the mod of the gender is highest:



Part 2: - Select a sub-dataset of 100 objects described by two nominal attributes, to apply FP-growth and find the frequent patterns, and strong associations. (Select appropriate thresholds according to your data)

```
162
163 sub_df=[list(df['Country'])[300:400]],
164         list(df['Education Level'])[300:400]])
165 te = TransactionEncoder()
166 te_ary = te.fit(sub_df).transform(sub_df)
167 df_res=pd.DataFrame(te_ary,columns=te.columns_)
168 frequent_item=fpgrowth(df_res, min_support=0.4, use_colnames=True)
169 print(frequent_item)
```

Run: main

	support	itemsets
0	0.5	(USA)
1	0.5	(UK)
2	0.5	(China)
3	0.5	(Canada)
4	0.5	(Australia)
5	0.5	(PhD)
6	0.5	(Master's)
7	0.5	(Bachelor's)
8	0.5	(USA, UK)
9	0.5	(China, UK)
10	0.5	(USA, China)
11	0.5	(USA, China, UK)

15	0.5	(China, Canada, UK)
16	0.5	(USA, China, Canada)
17	0.5	(USA, Canada, UK)
18	0.5	(USA, China, Canada, UK)
19	0.5	(Australia, Canada)
20	0.5	(China, Australia)
21	0.5	(Australia, UK)
22	0.5	(USA, Australia)
23	0.5	(China, Australia, Canada)
24	0.5	(UK, Australia, Canada)
25	0.5	(USA, Australia, Canada)
26	0.5	(China, Australia, UK)
27	0.5	(USA, China, Australia)
28	0.5	(USA, Australia, UK)
29	0.5	(UK, China, Australia, Canada)
30	0.5	(USA, China, Australia, Canada)
31	0.5	(UK, USA, Australia, Canada)
32	0.5	(USA, China, Australia, UK)
33	0.5	(USA, China, Australia, Canada, UK)
34	0.5	(PhD, Master's)
35	0.5	(Bachelor's, Master's)
36	0.5	(Bachelor's, PhD)
37	0.5	(Bachelor's, PhD, Master's)

To find strong association rule:

The threshold for confidence =0.8

The threshold for support =0.4

```

170 rules = association_rules(frequent_item, metric="confidence", min_threshold=1) # all Association
171 print("\nAssociation Rules:")
172 # as in strong role is confidence > .8 and support > .4 as you want
173 strong_rules = rules[
174     (rules['support'] > .4) & (rules['confidence'] > 0.8)
175 ]
176 print(strong_rules)

```

Run: main ×

C:\Users\msi\PycharmProjects\Datamining\venv\Scripts\python.exe C:/Users/msi/PycharmProjects/Datamining/main.py

Association Rules:

	antecedents	consequents	antecedent support	\
0	(UK)	(USA)	0.5	
1	(USA)	(UK)	0.5	
2	(UK)	(China)	0.5	
3	(China)	(UK)	0.5	
4	(China)	(USA)	0.5	
..	
187	(Bachelor's, Master's)	(PhD)	0.5	
188	(PhD, Master's)	(Bachelor's)	0.5	
189	(Bachelor's)	(PhD, Master's)	0.5	
190	(PhD)	(Bachelor's, Master's)	0.5	
191	(Master's)	(Bachelor's, PhD)	0.5	

Run: main ×

	consequent support	support	confidence	lift	leverage	conviction	\
0	0.5	0.5	1.0	2.0	0.25	inf	
1	0.5	0.5	1.0	2.0	0.25	inf	
2	0.5	0.5	1.0	2.0	0.25	inf	
3	0.5	0.5	1.0	2.0	0.25	inf	
4	0.5	0.5	1.0	2.0	0.25	inf	
..	
187	0.5	0.5	1.0	2.0	0.25	inf	
188	0.5	0.5	1.0	2.0	0.25	inf	
189	0.5	0.5	1.0	2.0	0.25	inf	
190	0.5	0.5	1.0	2.0	0.25	inf	
191	0.5	0.5	1.0	2.0	0.25	inf	

zhangs_metric

0	1.0
1	1.0
2	1.0
3	1.0
4	1.0
..	...
187	1.0
188	1.0
189	1.0
190	1.0
191	1.0

