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

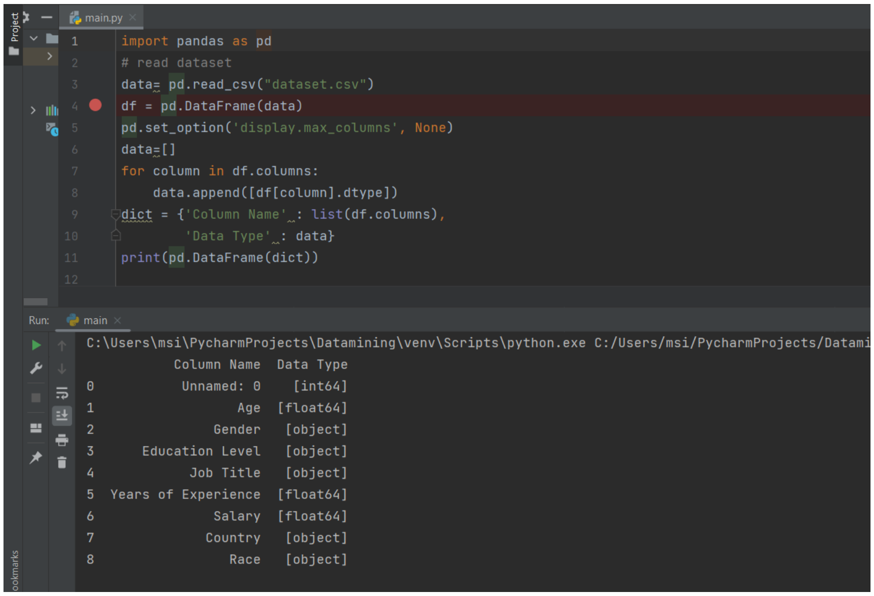
➢ **problems with this dataset:**
in our data set there is many problems:

- 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:**



```
import pandas as pd
# read dataset
data= pd.read_csv("dataset.csv")
df = pd.DataFrame(data)
pd.set_option('display.max_columns', None)
data=[]
for column in df.columns:
    data.append([df[column].dtype])
dict = {'Column Name' : list(df.columns),
        'Data Type' : data}
print(pd.DataFrame(dict))
```

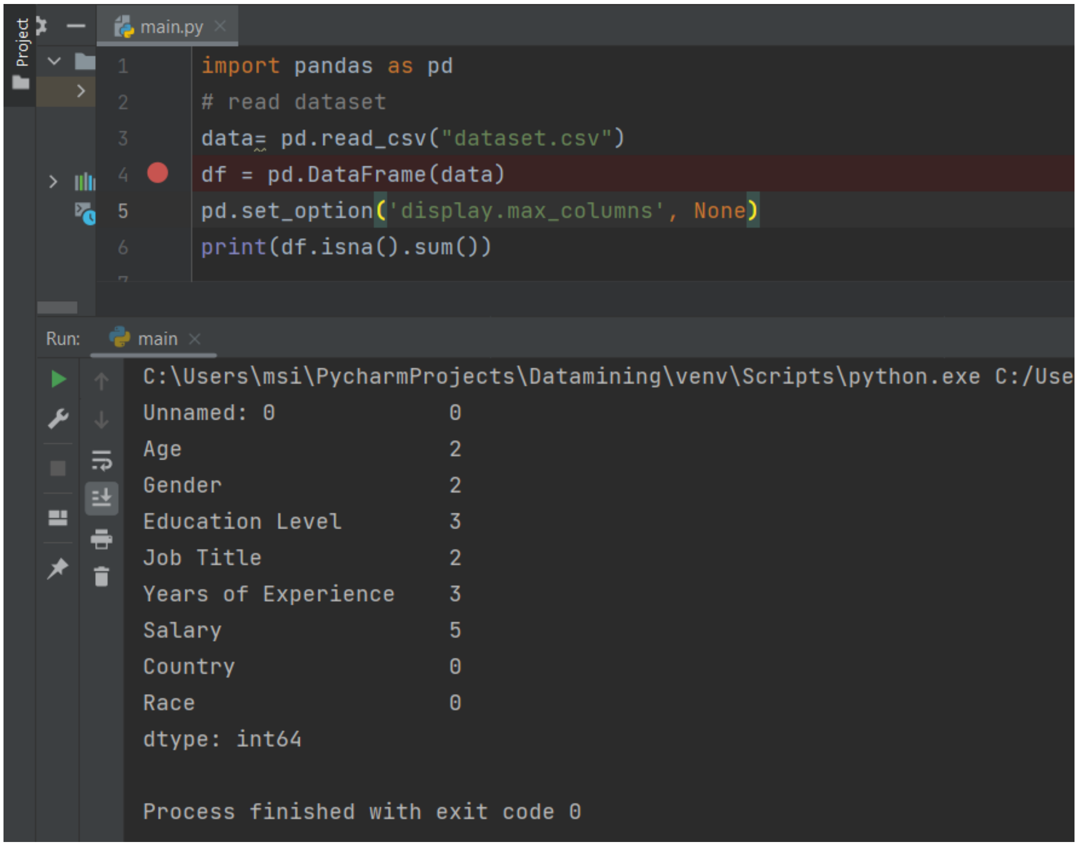
```
C:\Users\msi\PycharmProjects\Datamining\venv\Scripts\python.exe C:/Users/msi/PycharmProjects/Datami
          Column Name  Data Type
0          Unnamed: 0    [int64]
1                 Age  [float64]
2              Gender   [object]
3     Education Level   [object]
4           Job Title   [object]
5 Years of Experience  [float64]
6              Salary  [float64]
7             Country   [object]
8                Race   [object]
```

- **Here number of missing value in all data set:**

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

```
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
```

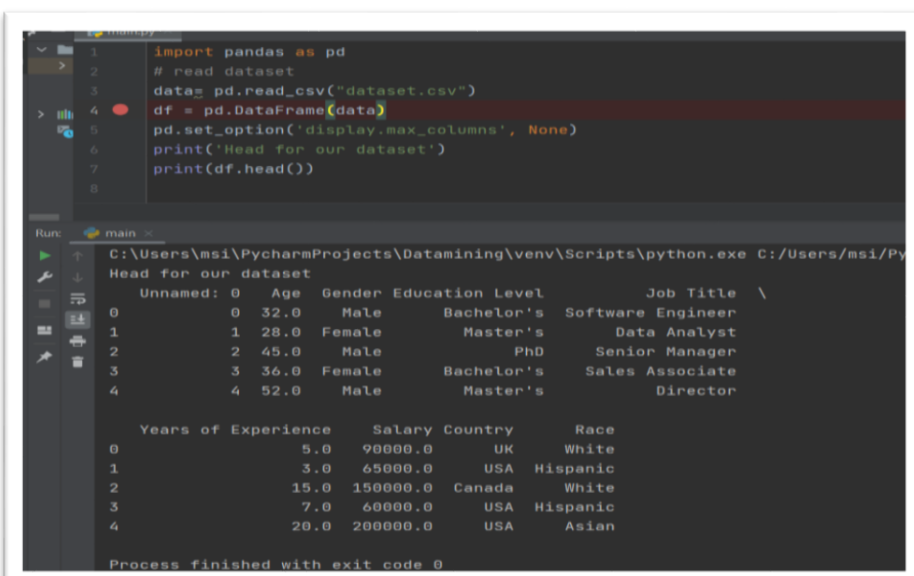
- **Here number of missing value in each column:**



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

```
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
```

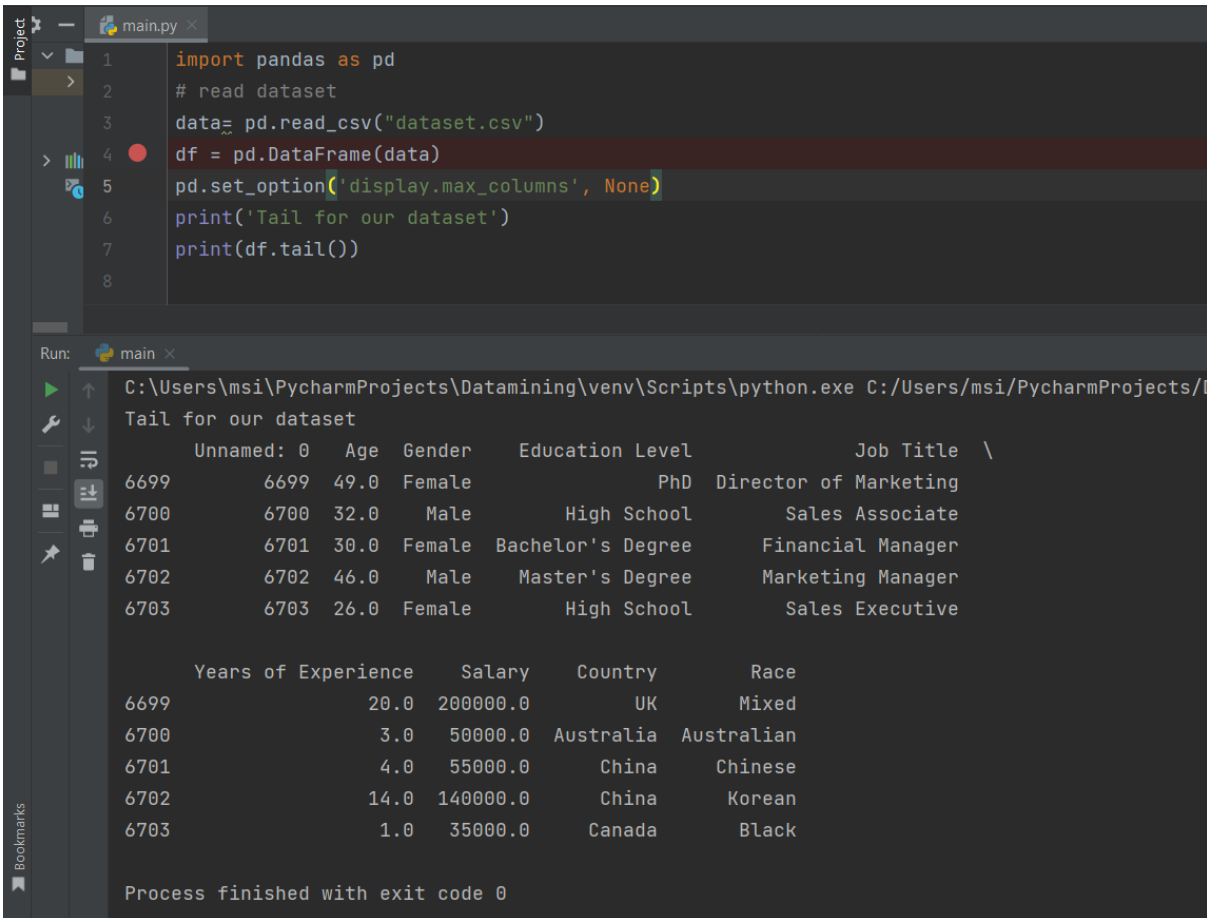
- **Here is the head of data set:**



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

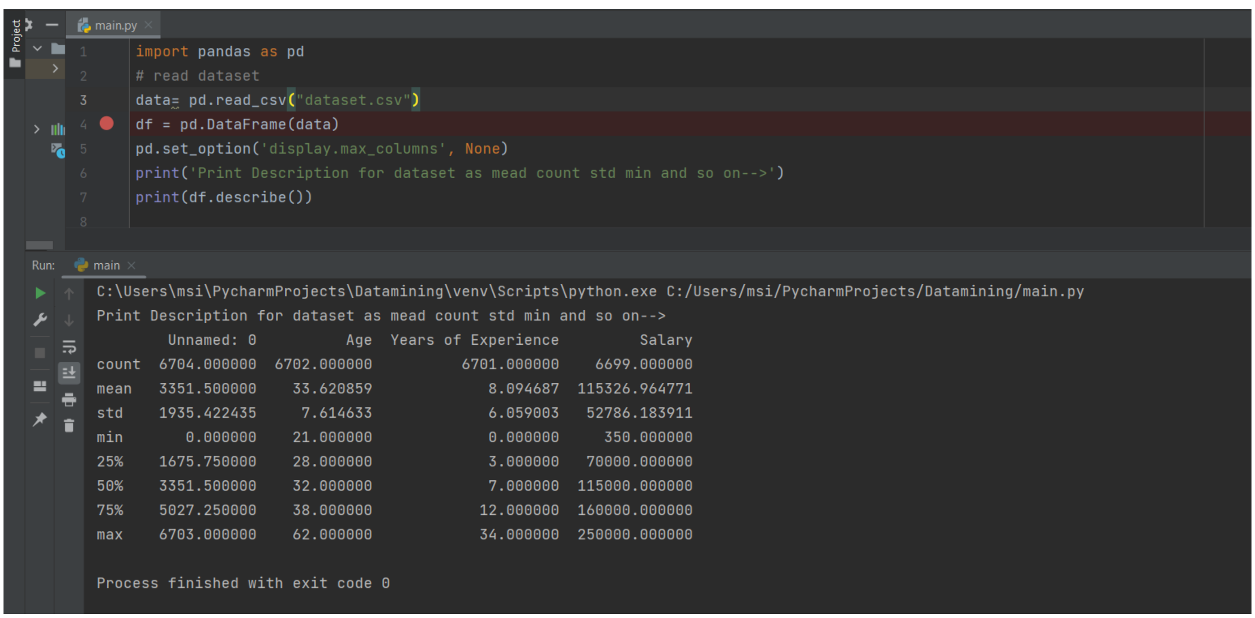
```
C:\Users\msi\PycharmProjects\Datamining\venv\Scripts\python.exe C:/Users/msi/Py
Head for our dataset
   Unnamed: 0   Age  Gender Education Level          Job Title  \
0           0  32.0    Male      Bachelor's  Software Engineer
1           1  28.0  Female        Master's       Data Analyst
2           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:**

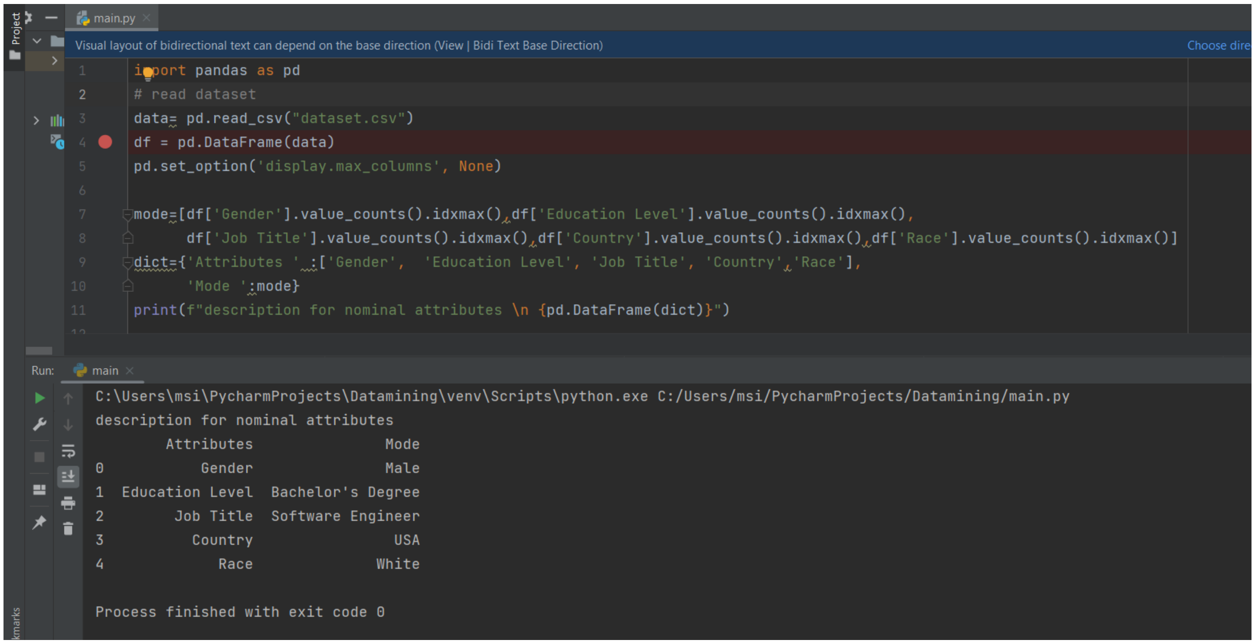


- **Here is the diminution of data set:**

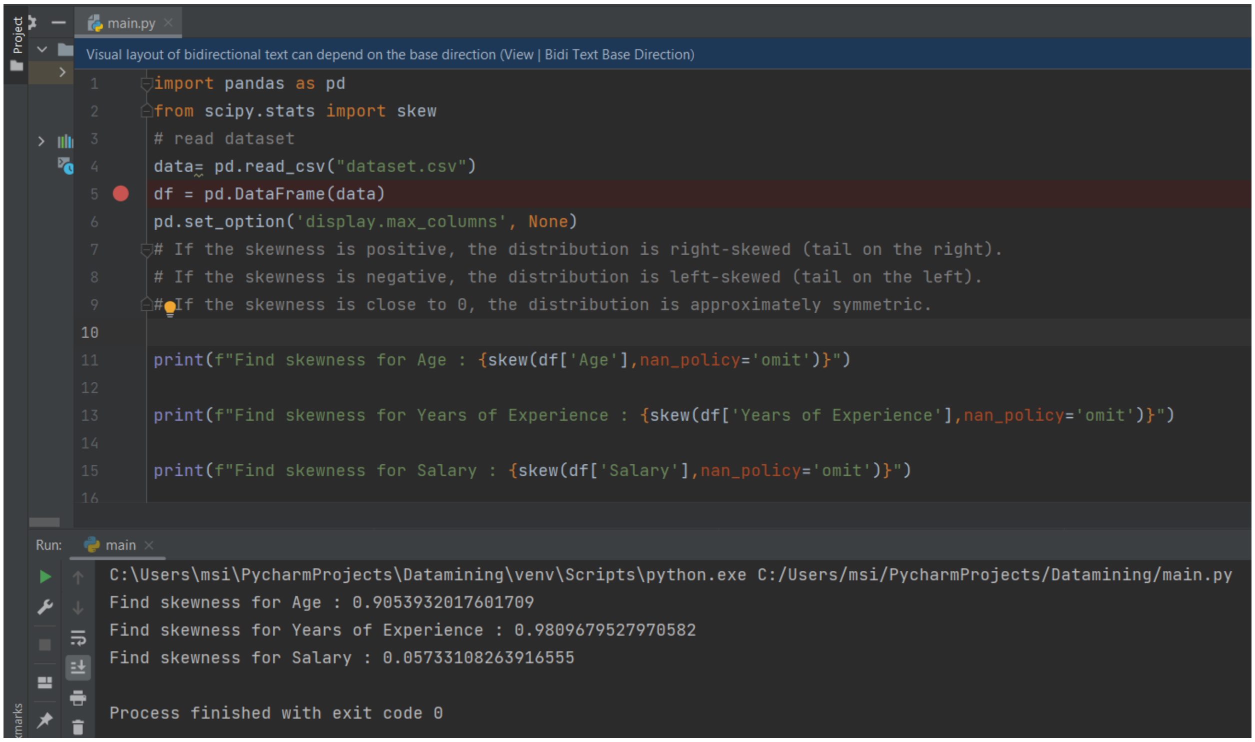
- **Description for numeric attribute (to show five-point summary): 25% indicate to Q1,50% indicate to Q2 and 75% indicate to Q3**



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

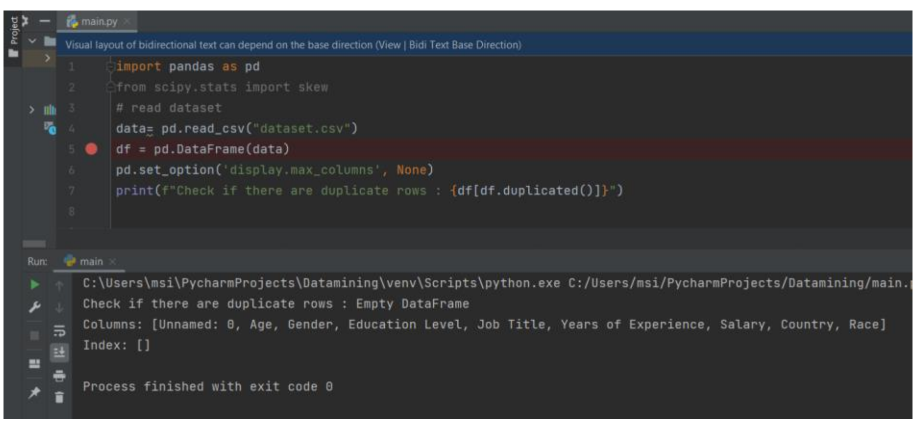


- If the skewness is positive, the distribution is right-skewed (tail on the right).
- If the skewness is negative, the distribution is left-skewed (tail on the left).
- If the skewness is close to 0, the distribution is approximately symmetric.

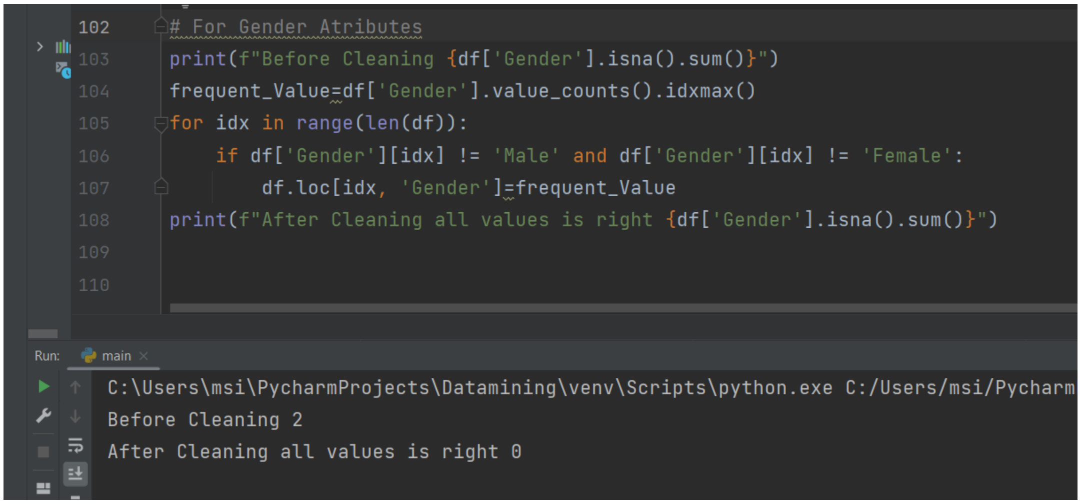


➢ Data Cleaning:

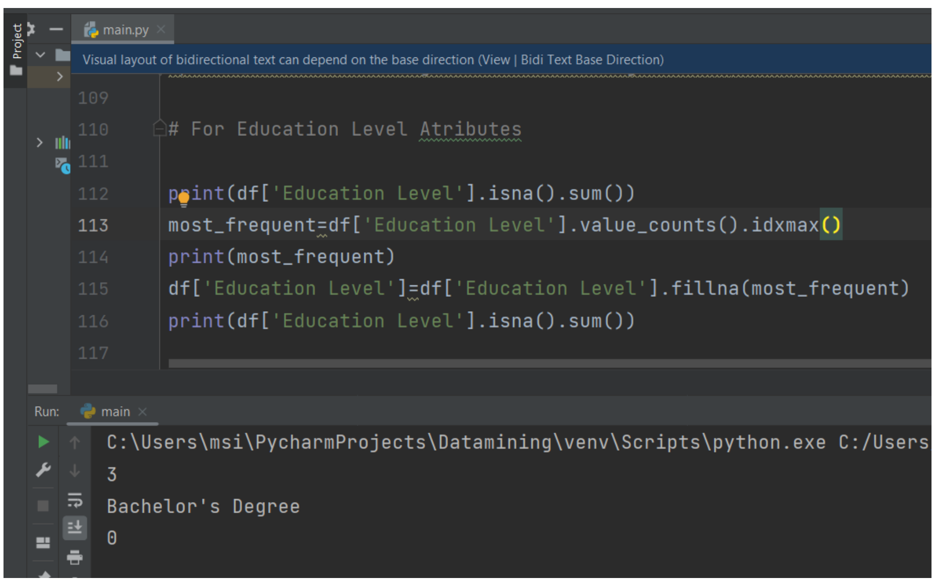
- **We Find if there is any duplicate in dataset (if there is we must remove it) As we note no duplicate in our data set:**



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

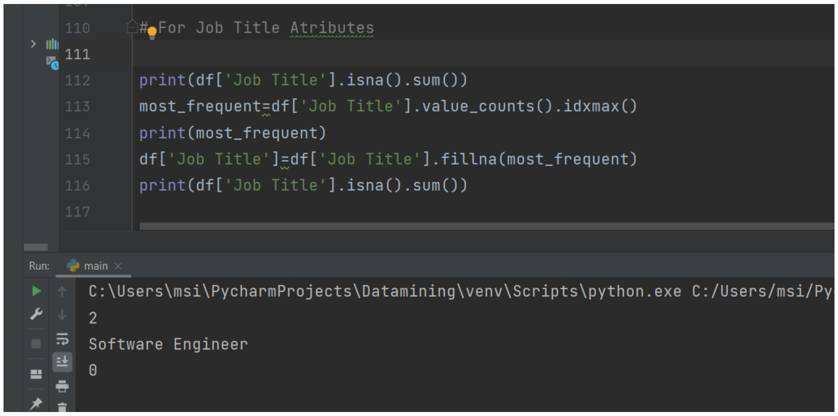


- **For Education level: mode equal bachelor's degree**

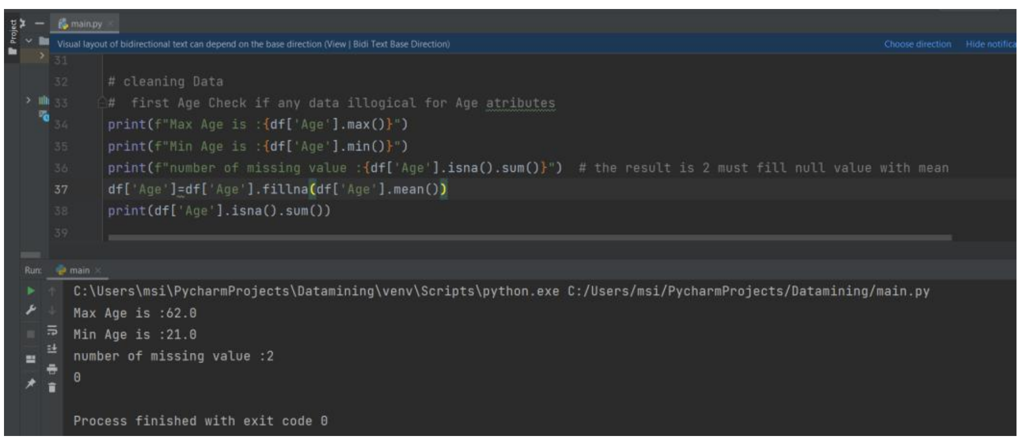


- **For Race and country as we mentioned previously there is no missing data.**

- **For job title:**

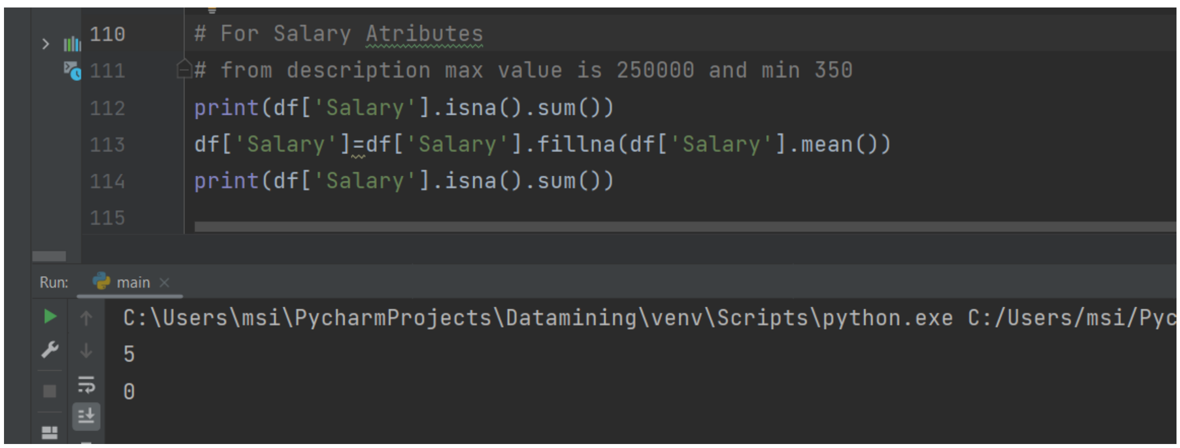


- **We make cleaning for missing numeric by compute the mean and replace missing with mean:**
  - **For Age the min value is 21 and max 62 and it is acceptable values and we calculate the mean and replace missing by it:**

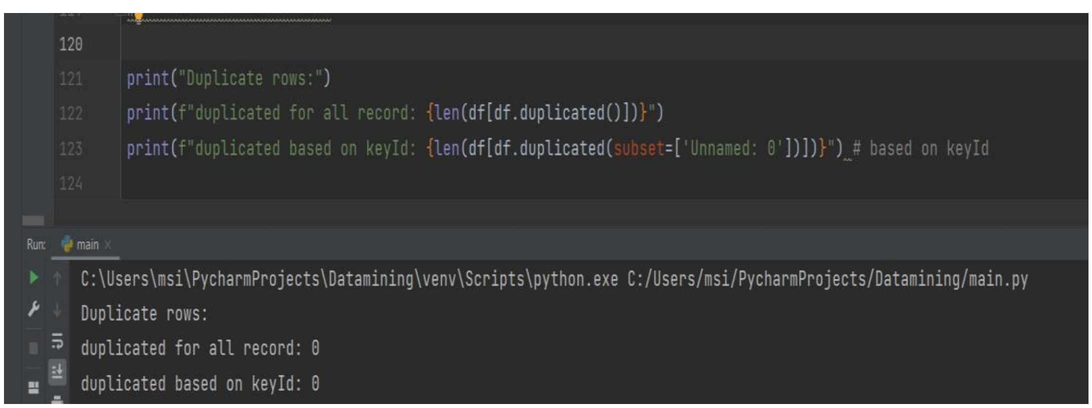


  - **For years of experience the min value is 0 and max 34 and it is acceptable values and we calculate the mean and replace
    missing here there is 3 values missing by it:**

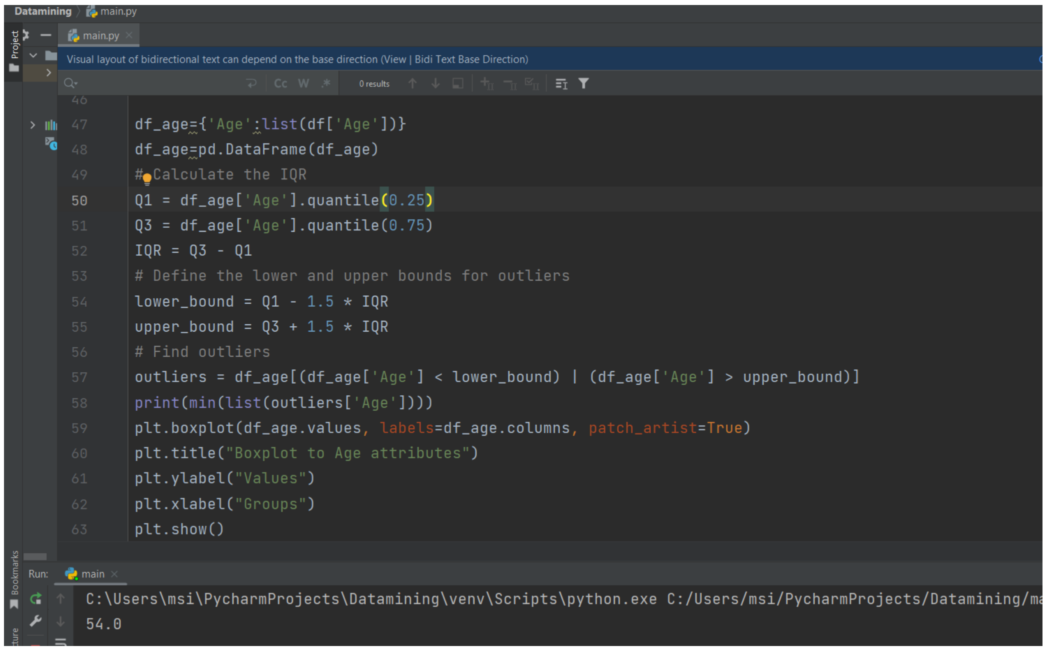
➢ **For salary the min is 350 and max is 250000 and it is acceptable values and we calculate the mean and replace missing here there is 5 values missing by it:**



➢ Cleaning duplicate by remove it:

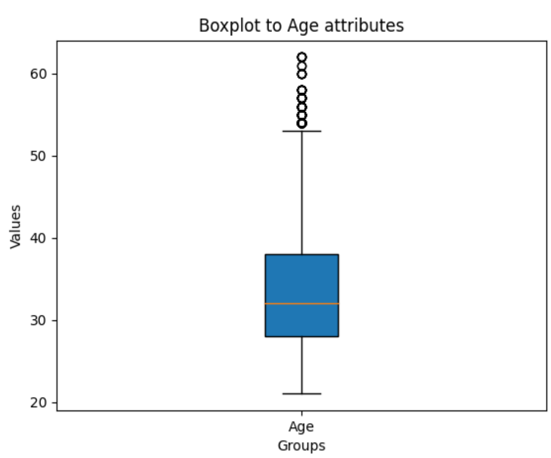


- Box blot:
    - For Age:
      We find the outlier and the first outlier is 54 as shown
      The number of outlier is very small so we remove it.

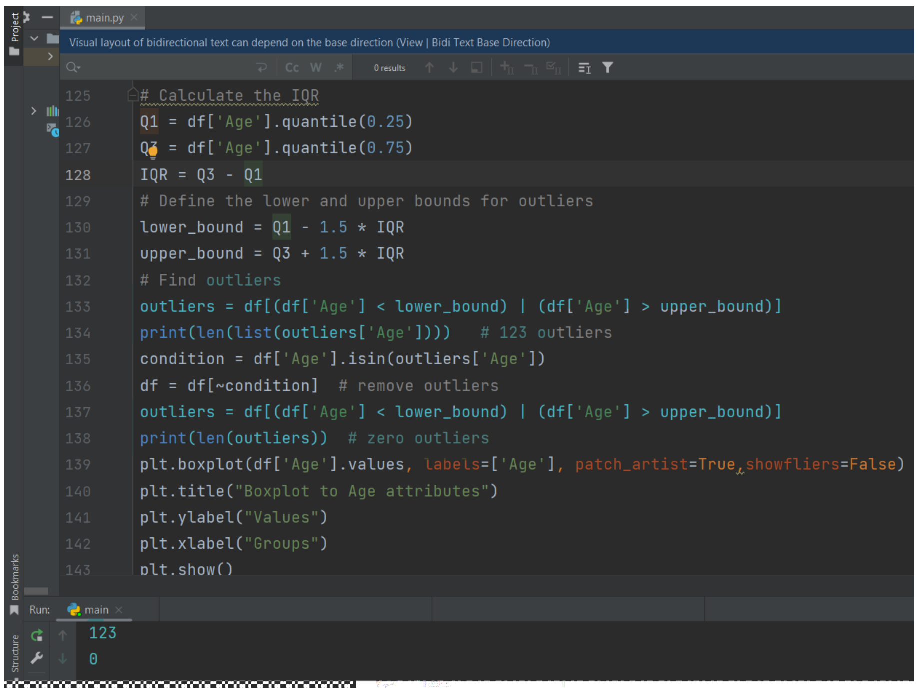


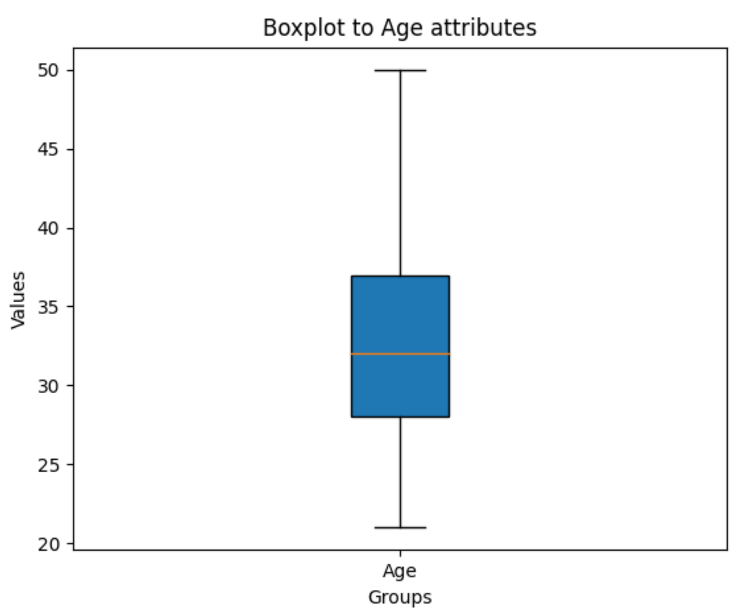

```
df_age={'Age':list(df['Age'])}
df_age=pd.DataFrame(df_age)
#Calculate the IQR
Q1 = df_age['Age'].quantile(0.25)
Q3 = df_age['Age'].quantile(0.75)
IQR = Q3 - Q1
# Define the lower and upper bounds for outliers
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
# Find outliers
outliers = df_age[(df_age['Age'] < lower_bound) | (df_age['Age'] > upper_bound)]
print(min(list(outliers['Age'])))
plt.boxplot(df_age.values, labels=df_age.columns, patch_artist=True)
plt.title("Boxplot to Age attributes")
plt.ylabel("Values")
plt.xlabel("Groups")
plt.show()
```

```
C:\Users\msi\PycharmProjects\Datamining\venv\Scripts\python.exe C:/Users/msi/PycharmProjects/Datamining/ma
54.0
```

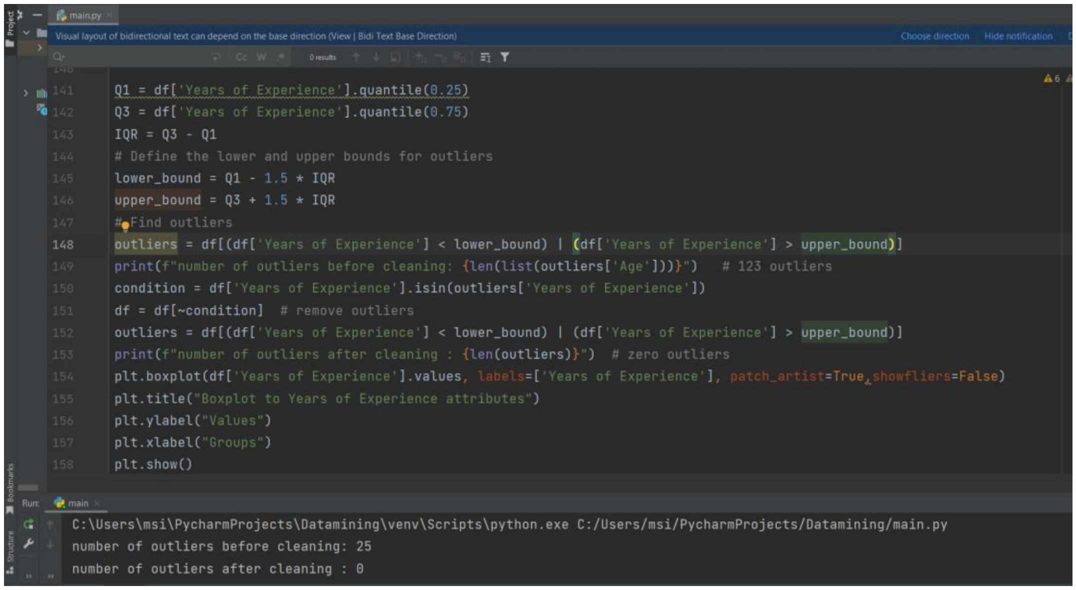


➢ After remove outlier for age:



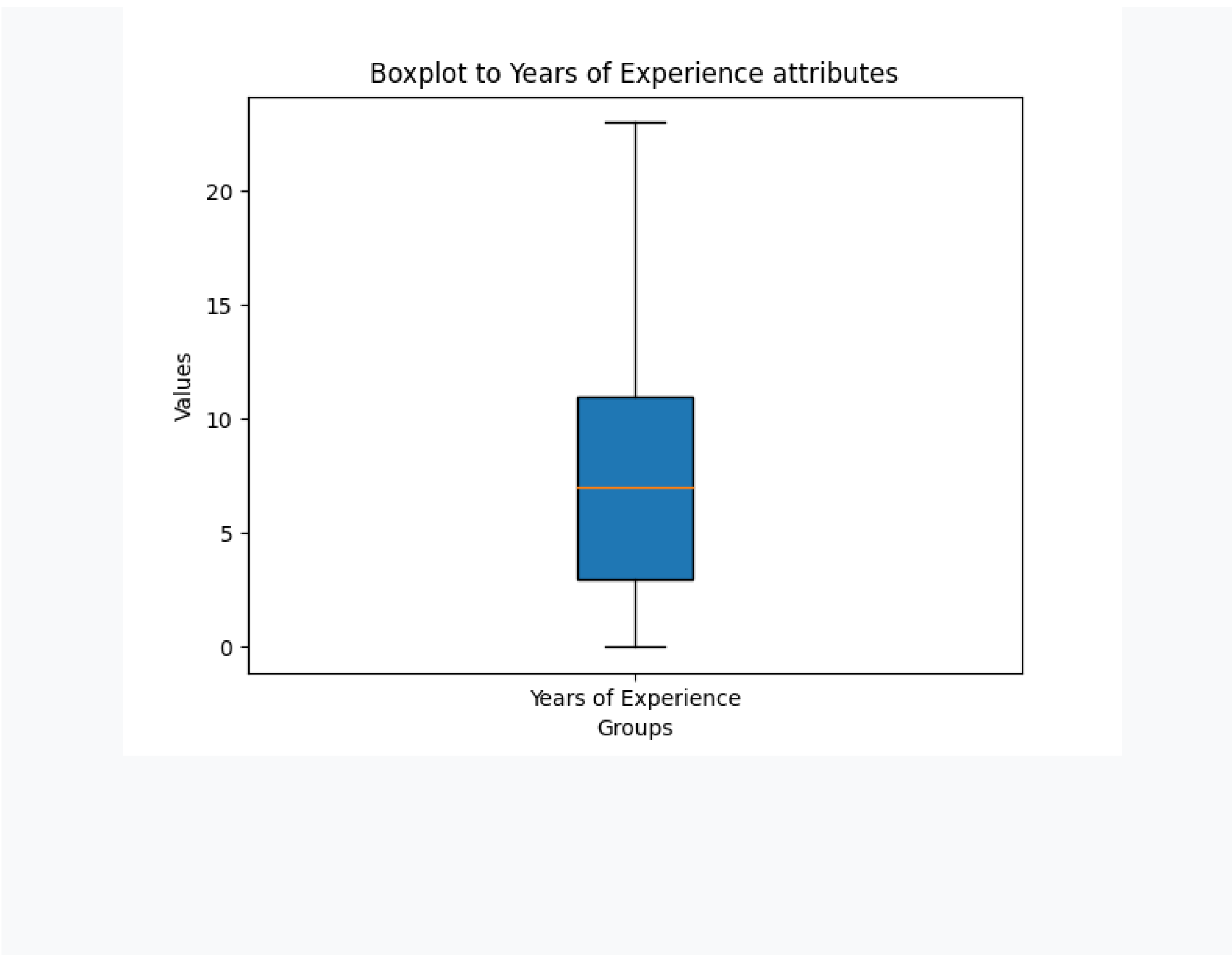


➢ **For years of experience after remove outliers:**

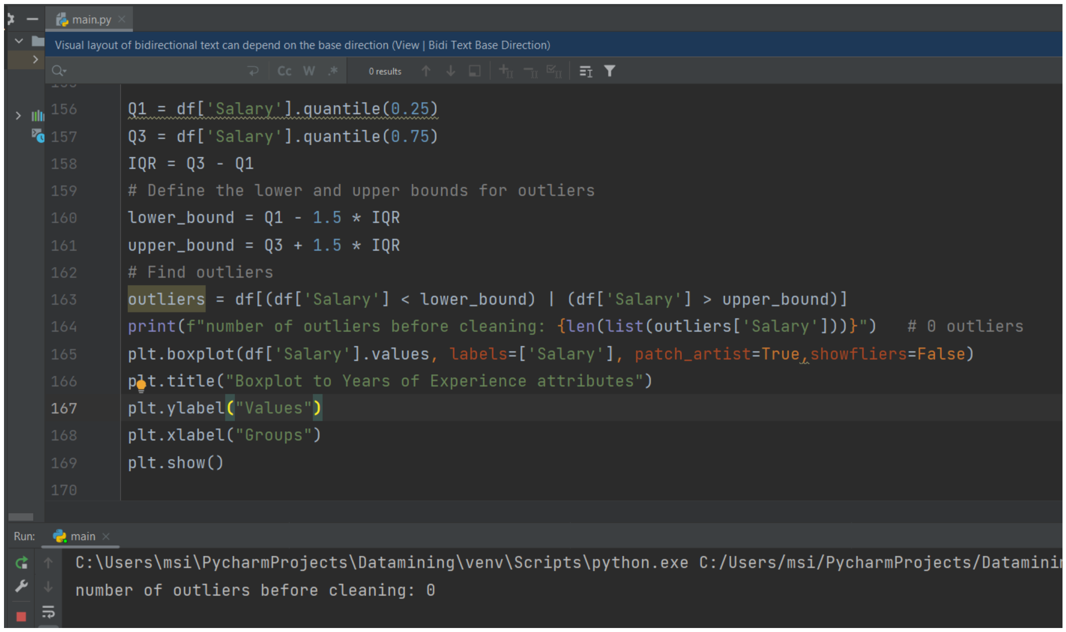


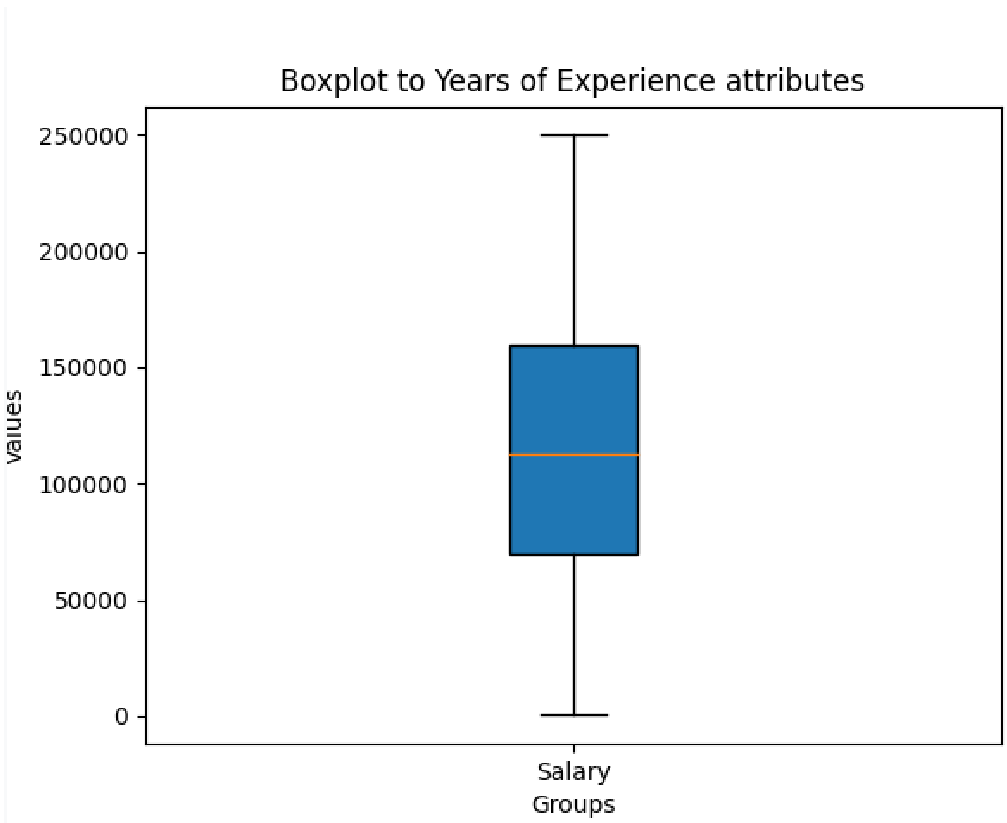
```
Q1 = df['Years of Experience'].quantile(0.25)
Q3 = df['Years of Experience'].quantile(0.75)
IQR = Q3 - Q1
# Define the lower and upper bounds for outliers
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
#Find outliers
outliers = df[(df['Years of Experience'] < lower_bound) | (df['Years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

```
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
```

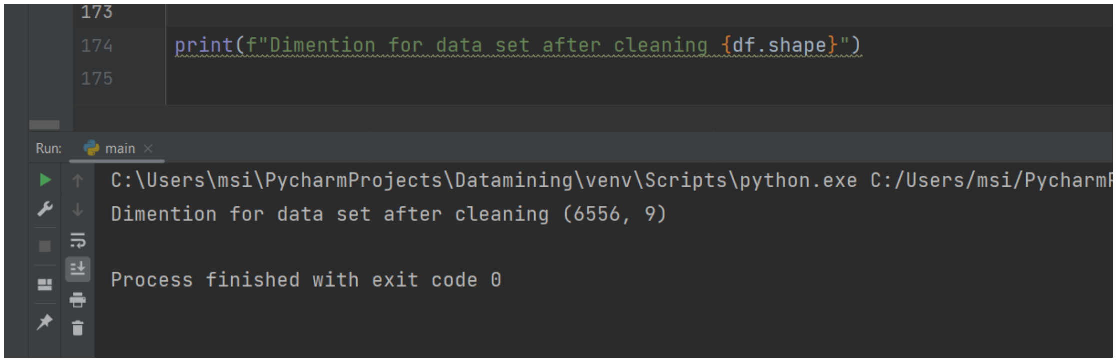


➢ **For salary after remove outliers:**





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



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

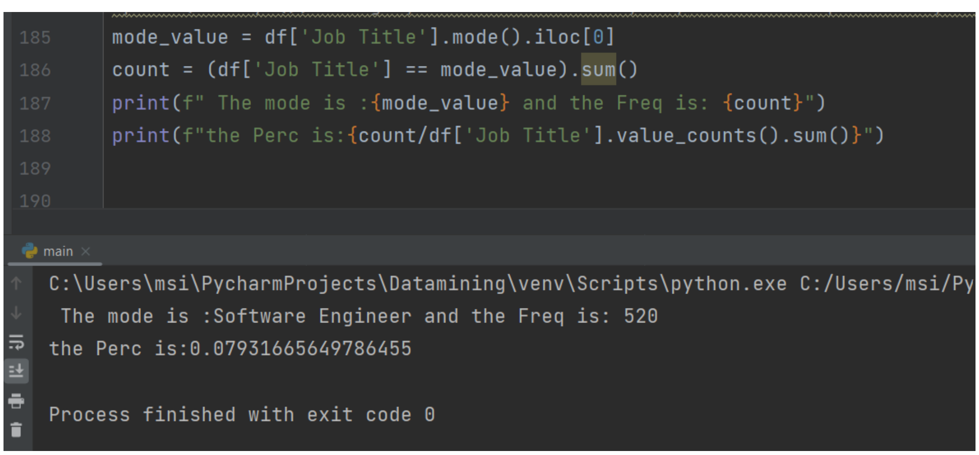
```
C:\Users\msi\PycharmProjects\Datamining\venv\Scripts\python.exe C:/Users/msi/Pycharm
Dimention for data set after cleaning (6556, 9)

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:

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

```
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:



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

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

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

```
C:\Users\msi\PycharmProjects\Datamining\venv\Scripts\python.exe C:/Users/msi/PycharmProje
 The mode is :Bachelor's Degree and the Freq is: 2257
the Perc is:0.3442647956070775
```

For country:

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

```
C:\Users\msi\PycharmProjects\Datamining\venv\Scripts\python.exe C:/Users/msi/Pycha
 The mode is :USA and the Freq is: 1335
the Perc is:0.20363026235509457
```

For Race:

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

```
C:\Users\msi\PycharmProjects\Datamining\venv\Scripts\python.exe C:/Users/msi
 The mode is :White and the Freq is: 1923
the Perc is:0.2933190970103722
```

For Salary:

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

```
Run:  main
C:\Users\msi\PycharmProjects\Datamining\venv\Scripts\python.exe C:/Users/msi/Pyc
 The mode is :140000.0 and the Freq is: 287
the Perc is:0.04377669310555216
```

For Years of experience:

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

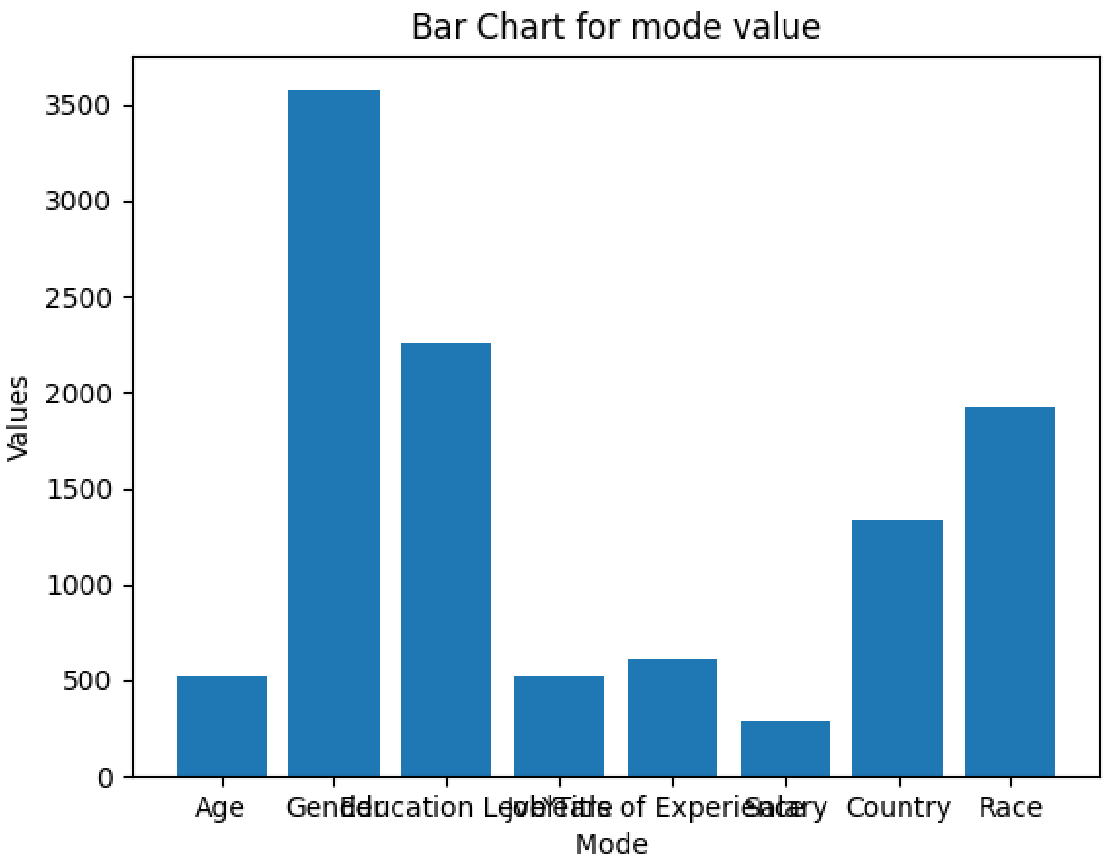
```
un:  main
C:\Users\msi\PycharmProjects\Datamining\venv\Scripts\python.exe C:/Users/msi/PycharmProjects/Dat
 The mode is :2.0 and the Freq is: 613
the Perc is:0.09350213544844417
```

For Age:

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

```
Run:  main
C:\Users\msi\PycharmProjects\Datamining\venv\Scripts\python.exe C:/Users/msi/PycharmProjects/Datamini
 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 basing to it (the data biasing to gender male). And the diagram represent that the mod of the gender is highest:



Bar Chart for mode value
Values
3500
3000
2500
2000
1500
1000
500
0
Age
Country
Race
Mode

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)

```python
sub_df=[list(df['Country'][300:400]),
        list(df['Education Level'][300:400])]
te = TransactionEncoder()
te_ary = te.fit(sub_df).transform(sub_df)
df_res=pd.DataFrame(te_ary,columns=te.columns_)
frequent_item=fpgrowth(df_res, min_support=0.4, use_colnames=True)
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

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