# Exploratory Data Analysis Report

Shweta Ajay Shinde
Masters in Data Analytics, San Jose State University
Data 230: Data Visualization
Instructor: Venkata Duvvuri
8th Sept 2024

# Exploratory Data Analysis Report

## 1.Introduction

This report presents an exploratory analysis of the data set contained in **Salaries.csv**. This dataset includes details about various salaries, and the primary objective is to explore and understand the data's structure and key attributes using google colab.

## 2. Data Loading and Inspection

### 2.1 Loading the Dataset

First, we'll need to load the dataset. We will first read the CSV file into a Pandas Data Frame for which we will use **pd.read_csv** command.

```
import pandas as pd
from google.colab import drive

# Reading the CSV file into a DataFrame
salaries = pd.read_csv('/content/drive/MyDrive/Salaries.csv')
```

### 2.2 Viewing the Data

To get an initial understanding of the dataset, we can view the first few records. We will use **head ()** command to view data. '*salaries*' is the name given to the data frame.

- First 10 records:-**salaries.head(10)**

```
# Viewing the first 10 records
print("First 10 records:")
print(salaries.head(10))
```

```
First 10 records:
        rank discipline  phd  service   sex  salary
0       Prof          B   56       49  Male  186960
1       Prof          A   12        6  Male   93000
2       Prof          A   23       20  Male  110515
3       Prof          A   40       31  Male  131205
4       Prof          B   20       18  Male  104800
5       Prof          A   20       20  Male  122400
6  AssocProf          A   20       17  Male   81285
7       Prof          A   18       18  Male  126300
8       Prof          A   29       19  Male   94350
9       Prof          A   51       51  Male   57800
```

- First 20 records: **salaries.head(20)**

```
[4] # Viewing the first 20 records
    print("First 20 records:")
    print(salaries.head(20))
```

```
First 20 records:
       rank discipline  phd  service   sex  salary
0      Prof          B   56       49  Male  186960
1      Prof          A   12        6  Male   93000
2      Prof          A   23       20  Male  110515
3      Prof          A   40       31  Male  131205
4      Prof          B   20       18  Male  104800
5      Prof          A   20       20  Male  122400
6  AssocProf         A   20       17  Male   81285
7      Prof          A   18       18  Male  126300
8      Prof          A   29       19  Male   94350
9      Prof          A   51       51  Male   57800
10     Prof          B   39       33  Male  128250
11     Prof          B   23       23  Male  134778
12 AsstProf          B    1        0  Male   88000
13     Prof          B   35       33  Male  162200
14     Prof          B   25       19  Male  153750
15     Prof          B   17        3  Male  150480
16 AsstProf          B    8        3  Male   75044
17 AsstProf          B    4        0  Male   92000
18     Prof          A   19        7  Male  107300
19     Prof          A   29       27  Male  150500
```

- First 50 records: **salaries.head(50)**

```
# Viewing the first 50 records
print("First 50 records:")
print(salaries.head(50))
```

```
First 50 records:
       rank discipline  phd  service     sex  salary
0      Prof          B   56       49    Male  186960
1      Prof          A   12        6    Male   93000
2      Prof          A   23       20    Male  110515
3      Prof          A   40       31    Male  131205
4      Prof          B   20       18    Male  104800
5      Prof          A   20       20    Male  122400
6  AssocProf         A   20       17    Male   81285
7      Prof          A   18       18    Male  126300
8      Prof          A   29       19    Male   94350
9      Prof          A   51       51    Male   57800
10     Prof          B   39       33    Male  128250
11     Prof          B   23       23    Male  134778
12 AsstProf          B    1        0    Male   88000
13     Prof          B   35       33    Male  162200
14     Prof          B   25       19    Male  153750
15     Prof          B   17        3    Male  150480
16 AsstProf          B    8        3    Male   75044
17 AsstProf          B    4        0    Male   92000
18     Prof          A   19        7    Male  107300
19     Prof          A   29       27    Male  150500
20 AsstProf          B    4        4    Male   92000
21     Prof          A   33       30    Male  103106
22 AsstProf          A    4        2    Male   73000
23 AsstProf          A    2        0    Male   85000
24     Prof          A   30       23    Male   91100
25     Prof          B   35       31    Male   99418
26     Prof          A   38       19    Male  148750
27     Prof          A   45       43    Male  155865
28 AsstProf          B    7        2    Male   91300
29     Prof          B   21       20    Male  123683
30 AssocProf         B    9        7    Male  107008
31     Prof          B   22       21    Male  155750
32     Prof          A   27       19    Male  103275
33     Prof          B   18       18    Male  120000
34 AssocProf         B   12        8    Male  119800
35     Prof          B   28       23    Male  126933
36     Prof          B   45       45    Male  146856
37     Prof          A   20        8    Male  102000
38 AsstProf          B    4        3    Male   91000
39     Prof          B   18       18  Female  129000
40     Prof          A   39       36  Female  137000
41 AssocProf         A   13        8  Female   74830
42 AsstProf          B    4        2  Female   80225
43 AsstProf          B    5        0  Female   77000
44     Prof          B   23       19  Female  151768
45     Prof          B   25       25  Female  140096
46 AsstProf          B   11        3  Female   74692
47 AssocProf         B   11       11  Female  103613
48     Prof          B   17       17  Female  111512
49     Prof          B   17       18  Female  122960
```

### 2.3 Viewing the Records in given range

To view the records in given range, use the **iloc()** method.
**salaries.iloc(start_index:end_index)**

**salaries.iloc(20:61):** This will slice the row from salary to include rows from index 20 up to 60, but not including index 61.

```
#view data for a given range[start_intex:end_index]
middle_slice = salaries.iloc[20:61]
print("Records from 20 to 60")
print(middle_slice)
```

```
Records from 20 to 60
           rank discipline  phd  service     sex  salary
20     AsstProf          B    4        4    Male   92000
21         Prof          A   33       30    Male  103106
22     AsstProf          A    4        2    Male   73000
23     AsstProf          A    2        0    Male   85000
24         Prof          A   30       23    Male   91100
25         Prof          B   35       31    Male   99418
26         Prof          A   38       19    Male  148750
27         Prof          A   45       43    Male  155865
28     AsstProf          B    7        2    Male   91300
29         Prof          B   21       20    Male  123683
30    AssocProf          B    9        7    Male  107008
31         Prof          B   22       21    Male  155750
32         Prof          A   27       19    Male  103275
33         Prof          B   18       18    Male  120000
34    AssocProf          B   12        8    Male  119800
35         Prof          B   28       23    Male  126933
36         Prof          B   45       45    Male  146856
37         Prof          A   20        8    Male  102000
38     AsstProf          B    4        3    Male   91000
39         Prof          B   18       18  Female  129000
40         Prof          A   39       36  Female  137000
41    AssocProf          A   13        8  Female   74830
42     AsstProf          B    4        2  Female   80225
43     AsstProf          B    5        0  Female   77000
44         Prof          B   23       19  Female  151768
45         Prof          B   25       25  Female  140096
46     AsstProf          B   11        3  Female   74692
47    AssocProf          B   11       11  Female  103613
48         Prof          B   17       17  Female  111512
49         Prof          B   17       18  Female  122960
50     AsstProf          B   10        5  Female   97032
51         Prof          B   20       14  Female  127512
52         Prof          A   12        0  Female  105000
53     AsstProf          A    5        3  Female   73500
54    AssocProf          A   25       22  Female   62884
55     AsstProf          A    2        0  Female   72500
56    AssocProf          A   10        8  Female   77500
57     AsstProf          A    3        1  Female   72500
58         Prof          B   36       26  Female  144651
59    AssocProf          B   12       10  Female  103994
60     AsstProf          B    3        3  Female   92000
```

### 2.4 Viewing the Last Few Records

To view the last few records, use the **tail()** method.
Last 10 records: **salaries.tail(10)**

```
# Viewing the last 10 records
print("Last 10 records:")
print(salaries.tail(10))
```

```
Last 10 records:
           rank discipline  phd  service     sex  salary
68     AsstProf          A    4        2  Female   77500
69         Prof          A   28        7  Female  116450
70     AsstProf          A    8        3  Female   78500
71    AssocProf          B   12        9  Female   71065
72         Prof          B   24       15  Female  161101
73         Prof          B   18       10  Female  105450
74    AssocProf          B   19        6  Female  104542
75         Prof          B   17       17  Female  124312
76         Prof          A   28       14  Female  109954
77         Prof          A   23       15  Female  109646
```

## 3. Data Summary

### 3.1 Number of Records

To find out how many records (rows) are in the Data Frame we use **shape** command
Number of records**:-salaries.shape[0]**

```
# Number of records
num_records = salaries.shape[0]
print(f"Number of records: {num_records}")
```

```
Number of records: 78
```

Here shape returns tuple of array dimensions. (rows , columns), we just want rows, so we get **shape[0]**

```
print(salaries.shape)
```

```
(78, 6)
```

**f** in print(f"Number …...:{num_records}") indicates an f-string. F-strings let you insert variables directly into strings using curly braces {}, making it easy to include values in your text.

### 3.2 Number of Elements

The total number of elements can be calculated as the product of the number of rows and columns. For this we use the **size** command.
Number of elements: **salaries.size**

```
# Number of elements
num_elements = salaries.size
print(f"Number of elements: {num_elements}")
```

```
Number of elements: 468
```

### 3.3 Column Names

To get the column names we use **columns.tolist()**
**tolist()** will convert the column names of the Data Frame to a list.

```
# Column names
column_names = salaries.columns.tolist()
print(f"Column names: {column_names}")
```

```
Column names: ['rank', 'discipline', 'phd', 'service', 'sex', 'salary']
```

Or we can use **list(salaries.columns)**

```
# Column names
column_list = list(salaries.columns)
print(f"Column names: {column_list}")
```

```
Column names: ['rank', 'discipline', 'phd', 'service', 'sex', 'salary']
```

### 3.4 Data Types of Columns

To view the types of each column we use **dtype** command.
Column data types: **salaries.dtype**

```
# Column data types
column_types = salaries.dtypes
print("Column data types:")
print(column_types)
```

```
Column data types:
rank          object
discipline    object
phd            int64
service        int64
sex           object
salary         int64
dtype: object
```

Here's a brief description of data type used:

- **object**: Represents categorical data or text. Used for strings and mixed types.
- **int64**: Represents integer numbers, which can be large, without decimal points**.**

## 4. Calculation on Data

### 4.1 Basic Statistics for the Salary Column

To calculate basic statistics (mean, median, standard deviation, etc.) for the salary column we use **salaries.['salary'].describe()** command which provides summary statistics for the 'salary' column. The **round()** function rounds numbers to the specified number of decimal places given as a parameter.

```
# Basic statistics for the 'salary' column
salary_stats = salaries['salary'].describe()
#Round upto 4 decimal places
salary_stats_rounded = salary_stats.round(4)
print("Basic statistics for the 'salary' column:")
print(salary_stats_rounded)
```

```
Basic statistics for the 'salary' column:
count        78.0000
mean     108023.7821
std       28293.6610
min       57800.0000
25%       88612.5000
50%      104671.0000
75%      126774.7500
max      186960.0000
Name: salary, dtype: float64
```

- **float64**: Represents numbers with decimals, allowing for precise measurements and fractional values.

### 4.2 Count of Values in the Salary Column

To find out how many values are present in the salary column we use **salaries.['salary'].count()** command which returns the number of non-null values in the 'salary' column.

```
# Count of values in the 'salary' column
salary_count = salaries['salary'].count()
print(f"Number of values in the 'salary' column: {salary_count}")
```

```
Number of values in the 'salary' column: 78
```

### 4.3 Average Salary

To calculate the average salary, we use **salaries.['salary'].mean()** command. It will return the average salary.

```
# Average salary
average_salary = salaries['salary'].mean()
print(f"Average salary: {average_salary:.2f}")
```

```
Average salary: 108023.78
```

Here **{average_salary:.2f} : .2f** displays value up to 2 decimal place

## 5. Conclusion

Using Pandas, we analyzed a dataset with 78 records and different data types like float, int, object. We calculated important statistics, such as averages and ranges. Pandas made it easy to handle and understand the data.

## References

Pandas Development Team. (n.d.). *Getting started with pandas*. Retrieved, from **https://pandas.pydata.org/pandas-docs/stable/getting_started/index.html**

Google Colab. command line code. *Assignment_3.ipynb*, **https://colab.research.google.com/drive/1OX39kH4Vwvr_CPYJG2os94gTYWymZB1M#scrollTo=et99_DyZEUR9**