Data Visualization Techniques

Introduction

Data visualization is a powerful tool that allows us to represent data in a graphical format, making it easier to understand and derive meaningful insights. By visually representing data, we can identify patterns, trends, and relationships that might not be apparent in raw data. In this notebook, we will explore various data visualization techniques using Python and popular libraries such as Matplotlib, Seaborn, and Plotly.

Description

This notebook aims to provide a comprehensive overview of different data visualization techniques and how they can be implemented using Python. The notebook includes code examples and explanations for each visualization technique. The following types of Data Visualization Techniques are addressed in this notebook:

- Bar Chart
- Comparative Bar Chart
- Horizontal Bar Chart
- Stacked Bar Chart
- Pie Chart
- Dot Plot
- Line Plot
- Scatter Plot
- Box plot
- Histogram Plot

Explaination with code

- Importing the Dataset

In [2]:

```
import pandas as pd
import matplotlib.pyplot as plt
data = pd.read_csv('emp_visu.csv')
data
```

Out[2]:

	First Name	Gender	Salary	Bonus %	Senior Management	Team	Age	Experi
0	Maria	Female	130590	11.858	0	Finance	26	
1	Angela	Female	54568	18.523	1	Business Development	27	
2	Allan	Male	125792	5.042	0	Client Services	28	
3	Rohan	Female	45906	11.598	1	Finance	28	
4	Douglas	Male	97308	6.945	1	Marketing	28	
5	Brandon	Male	112807	17.492	1	Human Resources	30	
6	Diana	Female	132940	19.082	0	Client Services	31	
7	Frances	Female	139852	7.524	1	Business Development	34	
8	Matthew	Male	100612	13.645	0	Marketing	34	
9	Larry	Male	101004	1.389	1	Client Services	35	
10	Joshua	Male	90816	18.816	1	Client Services	35	
11	Jerry	Male	72000	9.340	1	Finance	35	
12	Lois	Female	64714	4.934	1	Legal	35	
13	Dennis	Male	115163	10.125	0	Legal	36	
14	John	Male	97950	13.873	0	Client Services	37	
15	Thomas	Male	61933	10.945	1	Marketing	38	
16	Shawn	Male	111737	6.414	0	Human Resources	39	
17	Gary	Male	109831	5.831	0	Product	39	
18	Jeremy	Male	90370	7.369	0	Human Resources	42	
19	Kimberly	Female	41426	7.450	1	Finance	44	
20	Louise	Female	63241	15.132	1	Business Development	45	
21	Donna	Female	81014	1.894	0	Product	49	
22	Ruby	Female	65476	10.012	1	Product	54	
23	Lillian	Female	59414	1.256	0	Product	55	

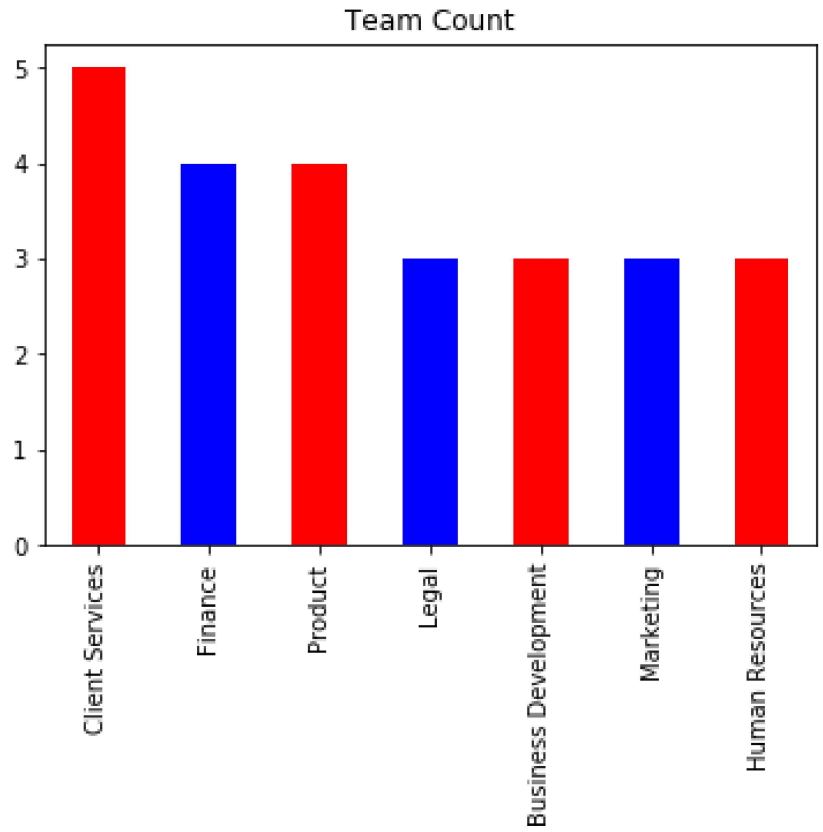
	First Name	Gender	Salary	Bonus %	Senior Management	Team	Age	Experi
24	Julia	Female	102508	12.637	1	Legal	58	

1) Bar Chart: A bar chart is a graphical representation of categorical data using rectangular bars. Each bar represents a category, and the length of the bar corresponds to the value or count of that category. Bar charts are commonly used to compare and display discrete data sets, making it easy to visualize and compare values across different categories.

Bar Chart with Team and its Count: This question involves creating a bar chart to visualize the count of teams.

In [3]:

```
# Draw a bar chart with Team and its count (use different colors for each te
chart = data['Team'].value_counts().plot(kind='bar', title="Team Count", col
plt.show()
```



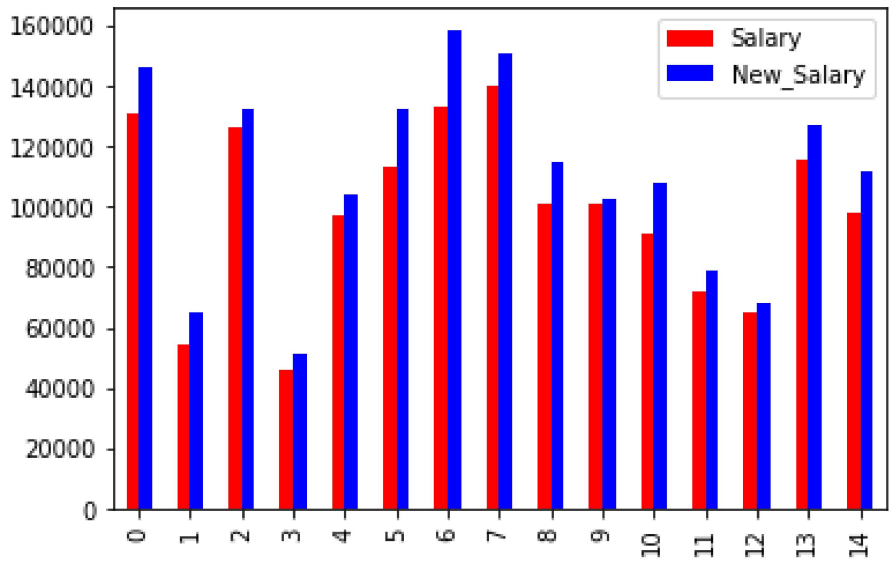
2) Comparative Bar Chart: A comparative bar chart is used to compare two or more variables or data sets side by side. Each variable or data set is represented by a set of bars, and the height or length of each bar corresponds to the value of that variable. Comparative bar charts are useful for visualizing and comparing data across different categories and identifying patterns or differences between variables.

Comparative Bar Chart for Salary and New_Salary: This question focuses on creating a comparative bar chart to compare the salary and new salary values for each person.

In [4]:

```
# Draw a comparative bar chart for Salary and New_Salary against each person
chart = data[['Salary', 'New_Salary']].head(15).plot(kind='bar',color=['red'
print(chart)
```

```
AxesSubplot(0.125,0.125;0.775x0.755)
```



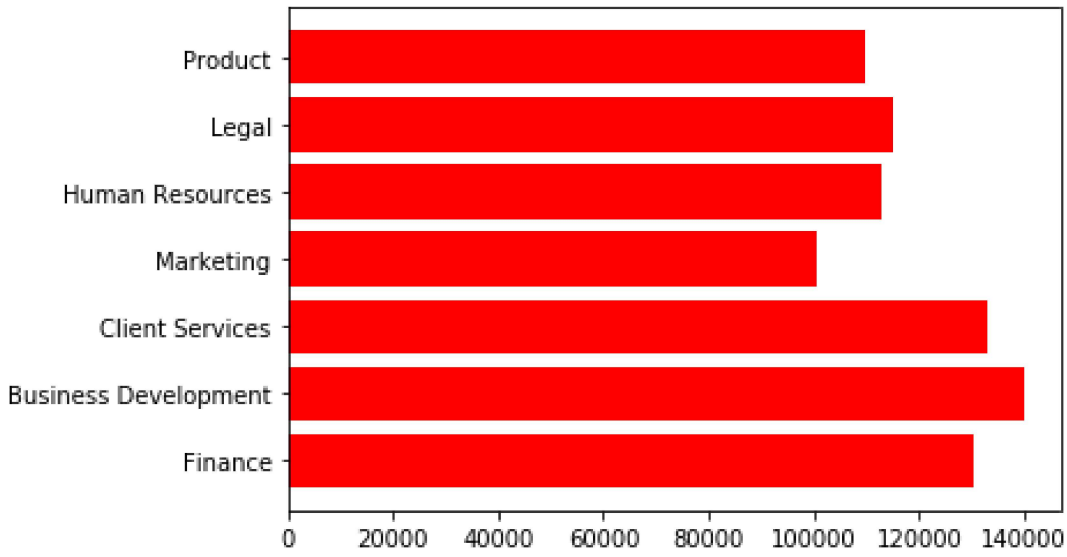
3) Horizontal Bar Chart: A horizontal bar chart is similar to a regular bar chart, but the bars are oriented horizontally instead of vertically. The length of each bar represents the value or count of a category, and the bars are arranged horizontally from longest to shortest. Horizontal bar charts are particularly useful when there are long category labels or when comparing a large number of categories.

Horizontal Bar Chart for Team and Salary: Here, we create a horizontal bar chart to display the relationship between teams and salaries.

In [5]:

```
# Draw a horizontal bar chart for Team and Salary
chart = plt.barh(data['Team'], data['Salary'], color=['red'])
print(chart)
```

```
<BarContainer object of 25 artists>
```



4) Stacked Bar Chart: A stacked bar chart is used to visualize the composition of a whole based on subcategories. Each bar represents the total value of a category, and it is divided into segments representing the subcategories. The length of each segment corresponds to the proportion or value of the subcategory within the total category. Stacked bar charts are effective for displaying the relative contributions of different components to the whole.

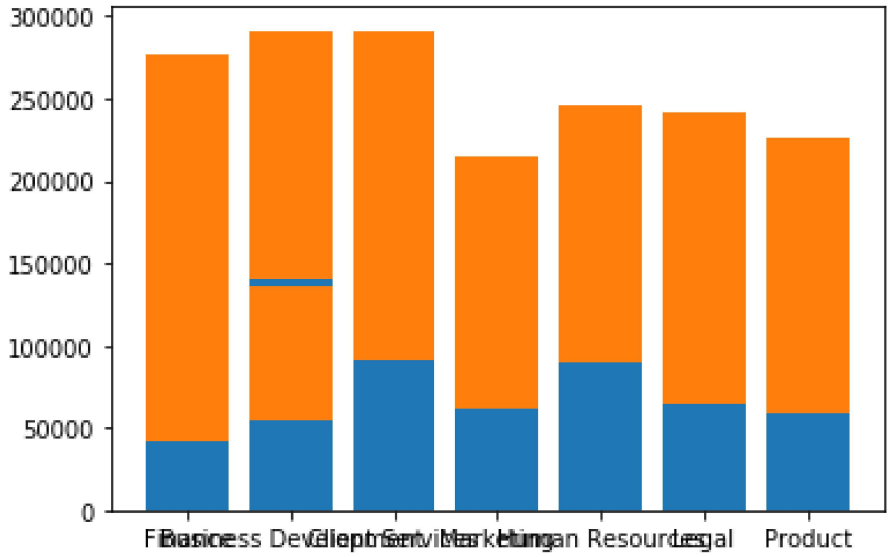
Stacked Bar Chart for Salary and New_Price: This question deals with creating a stacked bar chart to show the relationship between salary, new price, and each person.

In [6]:

```
# Draw a stacked barchart for Salary and New_Price against the person (first
plt.bar(data["Team"], data["Salary"])
plt.bar(data["Team"], data["New_Salary"], bottom=data["Salary"])
```

Out[6]:

```
<BarContainer object of 25 artists>
```



5) Pie Chart: A pie chart is a circular chart that represents data as slices of a pie. Each slice corresponds to a category, and the size of the slice represents the proportion or percentage of that category relative to the whole. Pie charts are useful for displaying the distribution or composition of data and comparing the relative sizes of different categories.

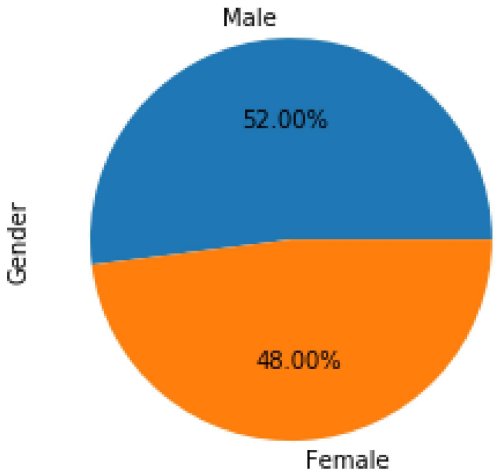
Pie Chart with Gender and its Count: This question involves creating a pie chart to visualize the distribution of genders.

In [7]:

```
# Draw a pie chart with Gender and its count
data['Gender'].value_counts().plot(kind='pie', autopct='%.2f%%')
```

Out[7]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2fac8e6198>
```



6) Dot Plot: A dot plot is a simple chart that represents individual data points as dots along a horizontal or vertical axis. Each dot corresponds to a data point, and its position on the axis represents its value. Dot plots are useful for visualizing the distribution of data, identifying outliers, and comparing individual values.

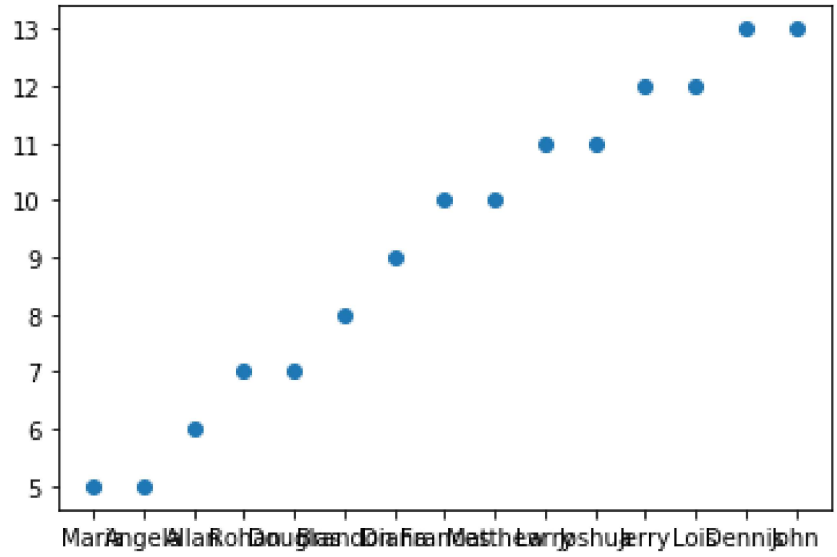
Dot Plot between Person and Experience: Here, we create a dot plot to depict the relationship between persons and their experience.

In [8]:

```
# Draw the dot plot between person and experience (first 15 persons)
plt.plot(data['First Name'].head(15),data['Experience'].head(15),linewidth=0
```

Out[8]:

```
[<matplotlib.lines.Line2D at 0x7f2fac8654a8>]
```



7) Line Plot: A line plot, also known as a line graph, is used to display data as a series of data points connected by straight lines. Line plots are particularly useful for showing trends or changes in data over time or across continuous variables. They are commonly used to represent time series data or continuous data sets.

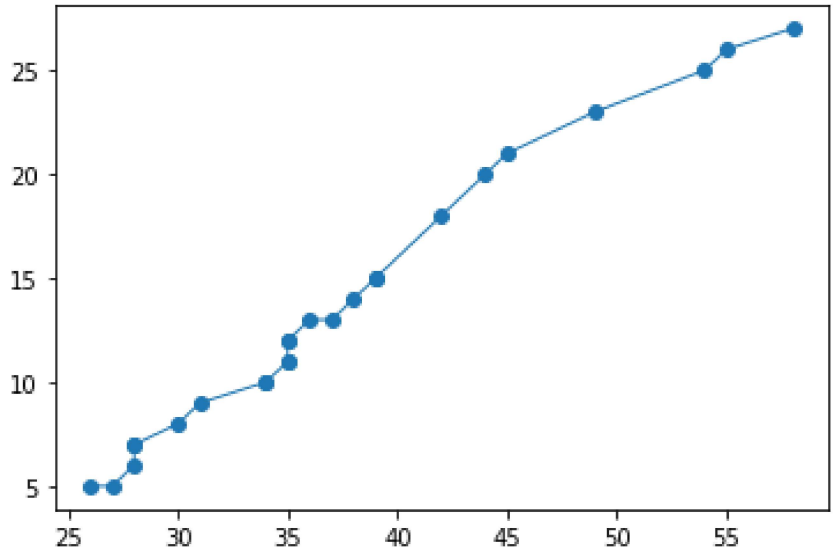
Line Plot between Age and Experience: This question focuses on creating a line plot to illustrate the relationship between age and experience.

In [9]:

```
# Draw the line plot between age and experience. Observe the trend line.
plt.plot(data['Age'],data['Experience'],linewidth=1,marker='o')
```

Out[9]:

```
[<matplotlib.lines.Line2D at 0x7f2fac7e0550>]
```



8) Scatter Plot: A scatter plot is used to visualize the relationship between two continuous variables. Each data point is represented by a marker on a Cartesian coordinate system, with one variable plotted on the x-axis and the other variable plotted on the y-axis. Scatter plots are effective for identifying patterns, correlations, clusters, or outliers in the data.

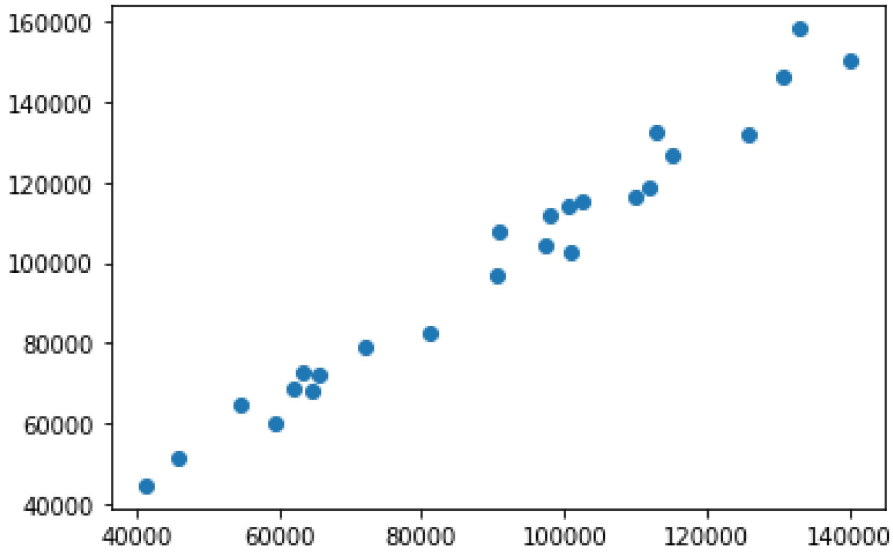
Scatter Plot between Salary and New_Salary: This question involves creating a scatter plot to observe the correlation between salary and new salary values.

In [10]:

```
# Draw the scatter plot between Salary and New_Salary. Observe the correlati
plt.scatter(data["Salary"], data["New_Salary"])
```

Out[10]:

```
<matplotlib.collections.PathCollection at 0x7f2fac7955f8>
```

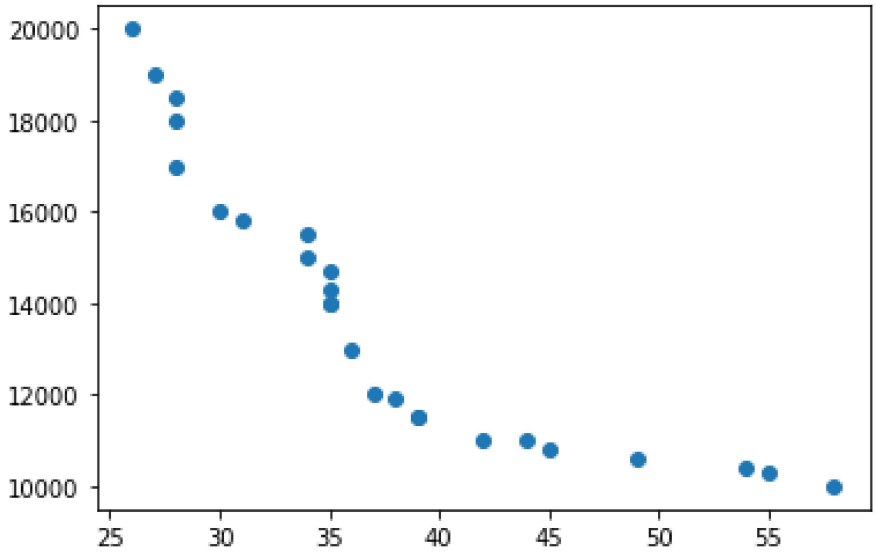


9) Box Plot: A box plot, also known as a box-and-whisker plot, is a visual representation of the statistical summary of a dataset. It displays the distribution of data by dividing it into quartiles. The plot consists of a rectangular box (the interquartile range) with a line inside (the median), and two lines (whiskers) that extend to the minimum and maximum values. Box plots provide insights into the central tendency, spread, and skewness of the data, as well as the presence of outliers.

Box Plot for Statistical Summary of Age Column: Here, we create a box plot to showcase the statistical summary of the age column.

In [11]:

```
# Draw the scater plot between Age and Incentive. Observe the correlation
chart = plt.scatter(data["Age"], data["Incentive"])
```

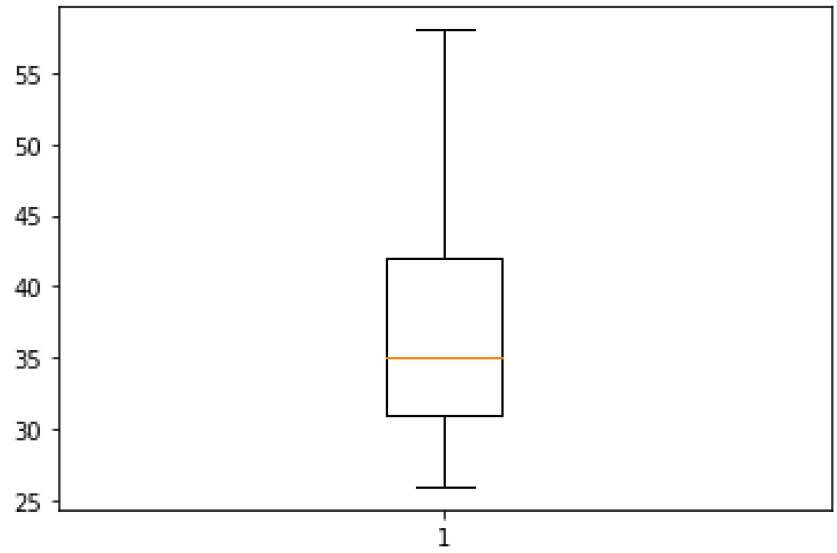


In [12]:

```
# Draw the box plot to show the statistical summary of Age column and verify
plt.boxplot(data["Age"])
data["Age"].describe()
```

Out[12]:

```
count    25.000000
mean     37.680000
std       8.938307
min      26.000000
25%      31.000000
50%      35.000000
75%      42.000000
max      58.000000
Name: Age, dtype: float64
```



10) Histogram Plot: A histogram plot is used to visualize the distribution of continuous or discrete data. It represents data as bars, where each bar corresponds to a range or bin of values, and the height of the bar represents the frequency or count of data points falling within that range. Histograms are useful for understanding the shape, central tendency, and variability of the data distribution. They can help identify patterns such as normal distribution, skewness, or multimodality.

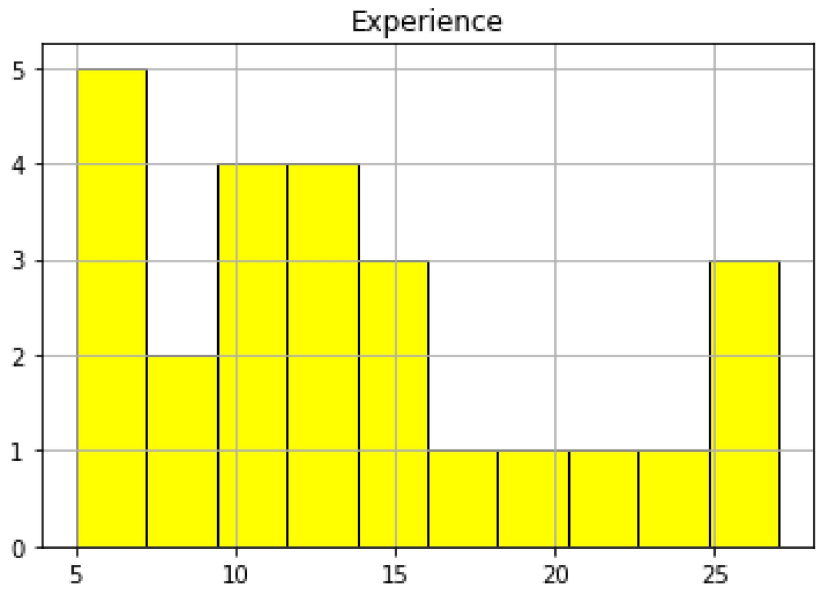
Histogram Plot for Experience Column: This question deals with creating a histogram plot to visualize the distribution of values in the experience column.

In [13]:

```
# Draw the histogram plot for Experience column.
data.hist("Experience", color="Yellow", edgecolor="black")
```

Out[13]:

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7f2
fac643518>]],
      dtype=object)
```

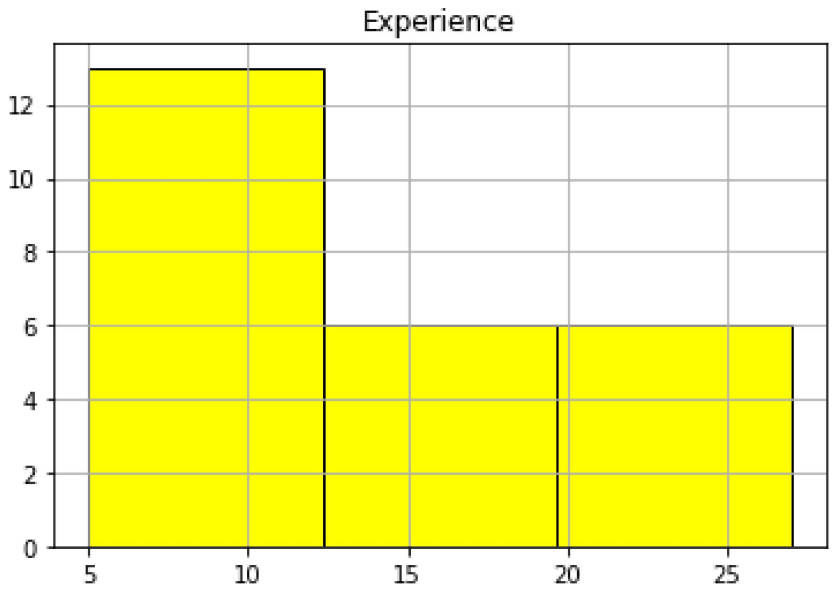


In [14]:

```
# Draw the histogram plot for Experience column with bin value and PDF
data.hist("Experience", color="Yellow", edgecolor="black", bins=3)
```

Out[14]:

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7f2
fac5cda90>]],
      dtype=object)
```

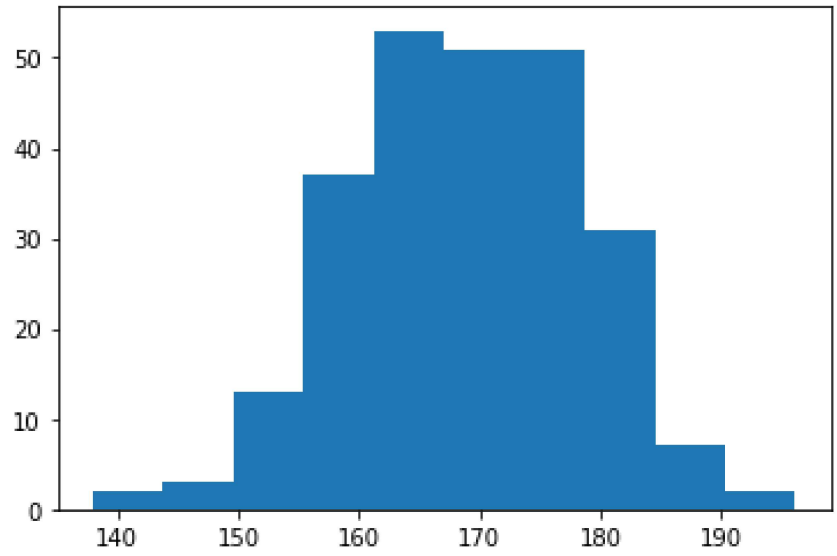


In [4]:

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

data1 = pd.read_csv('emp_visu.csv')
data1

x = np.random.normal(170, 10, 250)
plt.hist(x)
plt.show()
```



Each of these visualization techniques offers unique ways to represent and analyze data, allowing us to gain insights and communicate findings effectively.

Conclusion

Data visualization is a crucial aspect of data analysis and communication. It helps us understand complex data sets, identify patterns, trends, and outliers, and communicate our findings effectively. By utilizing various visualization techniques, we can unlock valuable insights and make data-driven decisions. In this notebook, we explored a range of visualization techniques, including bar charts, comparative bar charts, horizontal bar charts, stacked bar charts, pie charts, dot plots, line plots, scatter plots, box plots, and histograms. Each technique serves its unique purpose and provides a visual representation of different aspects of the data. With these tools at our disposal, we can effectively explore, analyze, and present data in a meaningful and visually appealing manner.

📧 **Contact Information:**

For any inquiries, feedback, or collaboration opportunities regarding the Data Visualization Techniques IPYNB file, please feel free to reach out to me through the following channels:

📩 Email: info@rubangino.in (mailto:info@rubangino.in)

🌐 LinkedIn: ruban-gino-singh (https://www.linkedin.com/in/ruban-gino-singh/)

🐦 Twitter: Rubangino (https://twitter.com/Rubangino)

🌐 GitHub: Ruban2205 (https://github.com/Ruban2205)

I'm open to discussions, questions, and suggestions related to data visualization, Python programming, and data analysis. Don't hesitate to connect with me and start a conversation. Let's explore the fascinating world of data visualization together!

Looking forward to connecting with you and sharing insights on data visualization techniques.