project 5,6,7

In [3]:
```
import pandas as pd
```

In [5]:
```
df=pd.read_csv(r"c:\Users\yadit\Downloads\adarsh data sheet .csv")
print(df.head())
print(df.info())
```

```
   Employee_ID    Employee_Name Department  Experience_Years  Monthly_Sales  \
0         1001           adarsh         HR                15          41834
1         1002  pratyush singh   Marketing                13          38047
2         1003             Amit         IT                 1          46105
3         1004            Priya  Marketing                 9          95766
4         1005            Karan  Marketing                 7          35707

   Customer_Satisfaction
0                     20
1                     15
2                      7
3                     10
4                      9
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 6 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Employee_ID            100 non-null    int64
 1   Employee_Name          100 non-null    object
 2   Department             100 non-null    object
 3   Experience_Years       100 non-null    int64
 4   Monthly_Sales          100 non-null    int64
 5   Customer_Satisfaction  100 non-null    int64
dtypes: int64(4), object(2)
memory usage: 4.8+ KB
None
```

Question 1.Load the dataset and display the first 10 rows

In [6]:
```
df.head(5)
```

Out[6]:

	Employee_ID	Employee_Name	Department	Experience_Years	Monthly_Sales
0	1001	adarsh	HR	15	41834
1	1002	pratyush singh	Marketing	13	38047
2	1003	Amit	IT	1	46105
3	1004	Priya	Marketing	9	95766
4	1005	Karan	Marketing	7	35707

Question 2.Display summary statistics of the numerical column

In [7]:
```
print(df.describe())
```

```
       Employee_ID  Experience_Years  Monthly_Sales  Customer_Satisfaction
count   100.000000        100.000000     100.000000             100.000000
mean   1050.500000          8.450000   72030.750000               5.060000
std      29.011492          4.349329   30111.719996               3.299281
min    1001.000000          1.000000   20854.000000               1.000000
25%    1025.750000          5.000000   44832.000000               3.000000
50%    1050.500000          8.500000   72697.500000               5.000000
75%    1075.250000         12.000000  101270.000000               7.250000
max    1100.000000         15.000000  118506.000000              20.000000
```

In [ ]:
```
Question 3.Check how many rows and columns the dataset has
```

In [8]:
```
print(df.shape)
```

```
(100, 6)
```

Question 4.Select only the Employee_ID , Experience_Years , Monthly_Sales , Customer_Satisfaction and balance columns .

In [9]:
```
result = [['Employee_ID','Experience_Years','Monthly_Sales','Customer_Satisfac
print(result)
```

```
[['Employee_ID', 'Experience_Years', 'Monthly_Sales', 'Customer_Satisfaction']]
```

Question 5.Filter rows where monthly_sales greater than 20.

In [ ]:
```
result = df[df['Monthly_Sales'] > 20]
print(result)
```

```
    Employee_ID    Employee_Name Department  Experience_Years  Monthly_Sales  \
0          1001           adarsh         HR                15          41834
1          1002  pratyush singh   Marketing                13          38047
2          1003             Amit         IT                 1          46105
3          1004            Priya  Marketing                 9          95766
4          1005            Karan  Marketing                 7          35707
..          ...              ...        ...               ...            ...
95         1096            Megha    Finance                12          93656
96         1097           Pritam    Finance                12          59384
97         1098           Ramesh  Marketing                 4          67254
98         1099          Shivani      Sales                14          41918
99         1100            Niraj  Marketing                14         105981

    Customer_Satisfaction
0                      20
1                      15
2                       7
3                      10
4                       9
..                    ...
95                      6
96                      7
97                      2
98                     10
99                      2

[100 rows x 6 columns]
```

Question 6. Find employees whose department is HR .

In [19]:
```
result = df[df['Department'] == 'HR']
print(result)
```

	Employee_ID	Employee_Name	Department	Experience_Years	Monthly_Sales	Customer_Satisfaction
5	1006	Divya	HR	9	41976	4
14	1015	Anjali	HR	1	29474	3
18	1019	Suman	HR	1	114856	8
23	1024	Manoj	HR	10	86199	4
31	1032	Monika	HR	1	71005	9
46	1047	Shweta	HR	15	110084	8
51	1052	Rohan	HR	10	107092	1
52	1053	Ira	HR	7	70859	3
54	1055	Krishna	HR	14	107455	7
56	1057	Sunny	HR	11	90467	6
66	1067	Niharika	HR	7	54698	3
68	1069	Aarav	HR	4	42671	3
72	1073	Sandeep	HR	5	106202	6
76	1077	Prakash	HR	15	69811	8
77	1078	Mitali	HR	11	22811	1
79	1080	Jay	HR	4	92082	3
80	1081	Vandana	HR	13	54754	1
90	1091	Sujit	HR	9	108614	10
92	1093	Shalini	HR	6	57504	8

Question 7.Count the frequency of each employee_name column value.

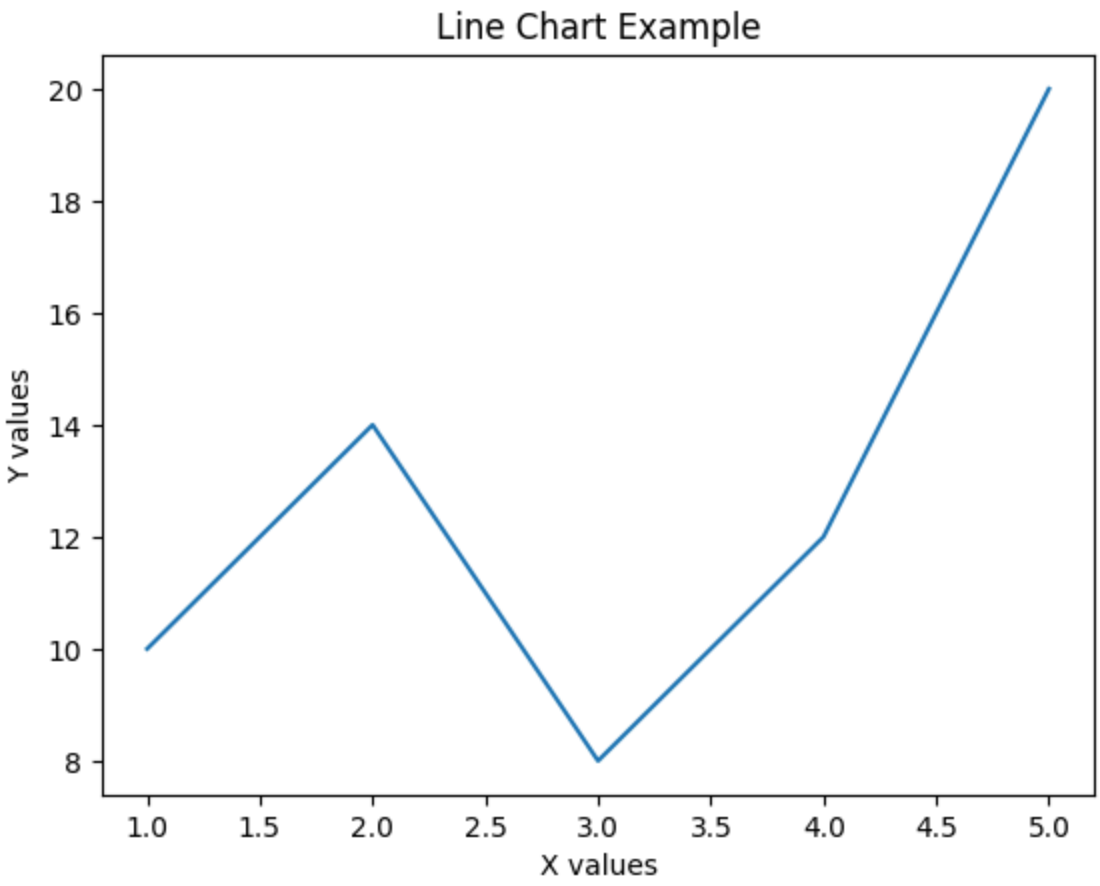
In [25]:
```
counts = df['Employee_Name'].value_counts()
print(counts)
```

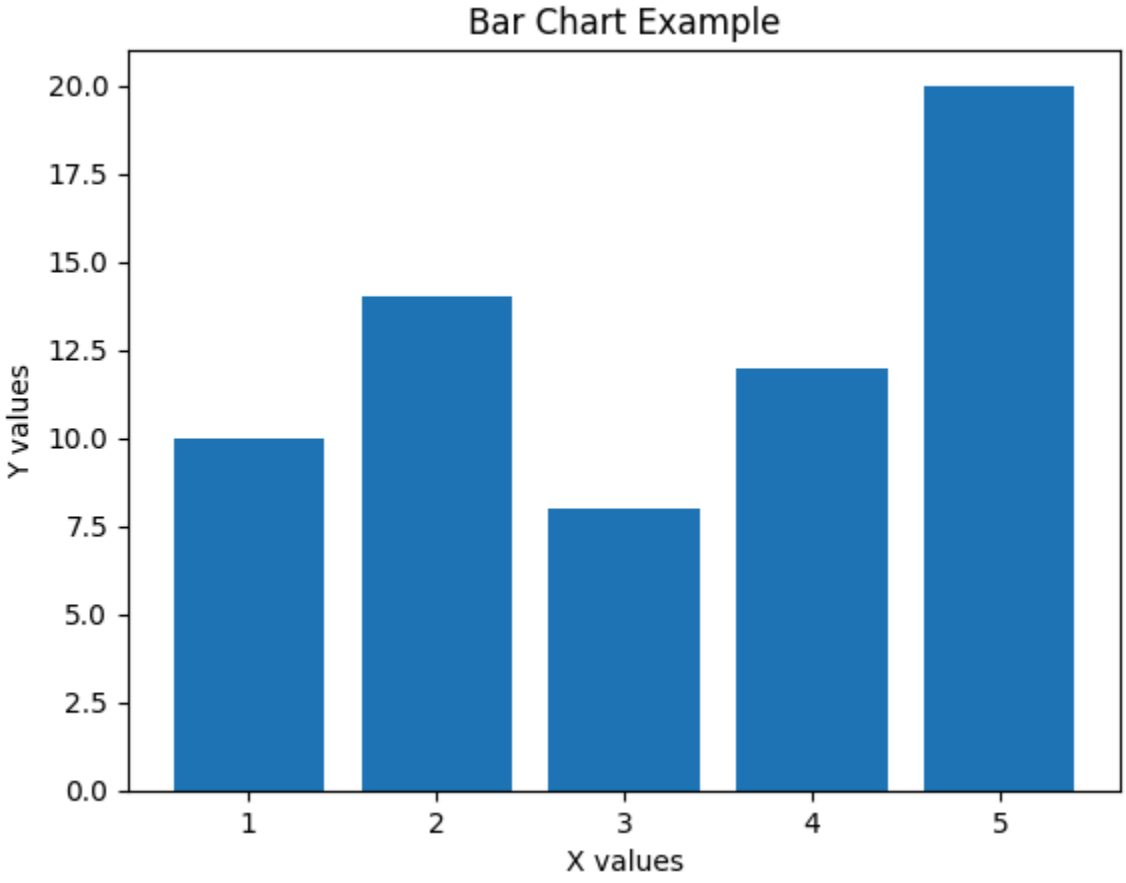
```
Employee_Name
adarsh             1
pratyush singh     1
Amit               1
Priya              1
Karan              1
                  ..
Megha              1
Pritam             1
Ramesh             1
Shivani            1
Niraj              1
Name: count, Length: 100, dtype: int64
```

project 6,7 fills

6. Create a line chart and a bar chart using Matplotlib library

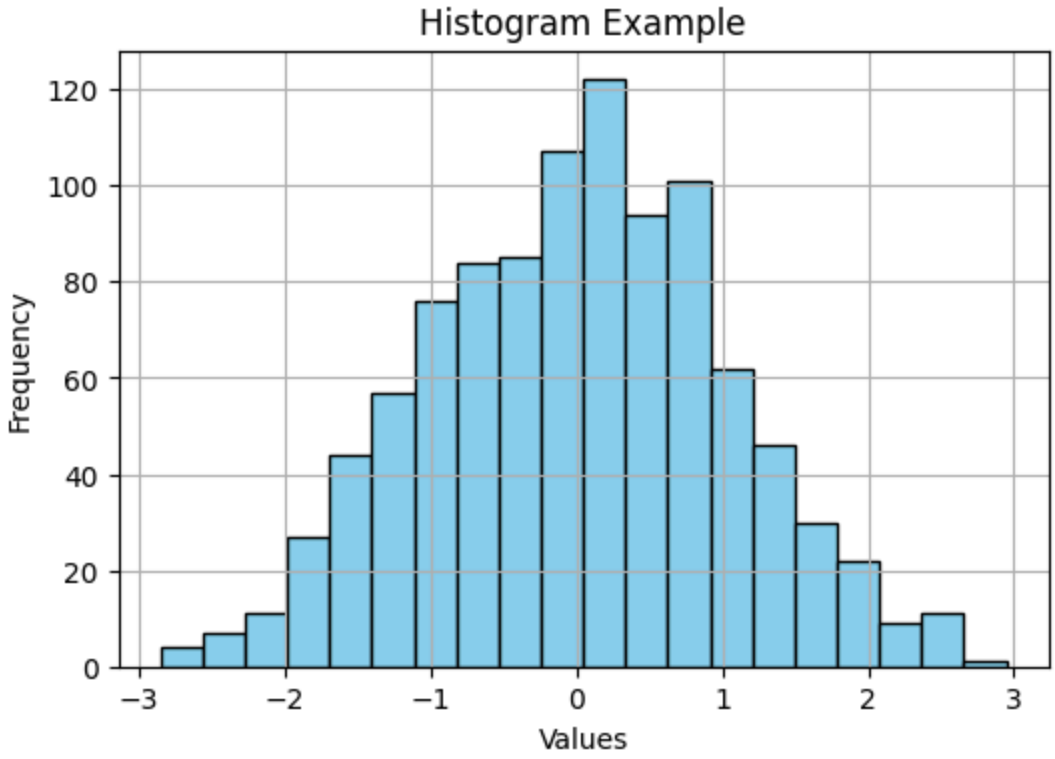
In [1]:
```python
import matplotlib.pyplot as plt

# Common data
x = [1, 2, 3, 4, 5]
y = [10, 14, 8, 12, 20]

# ---- Line Chart ----
plt.figure()
plt.plot(x, y)
plt.title("Line Chart Example")
plt.xlabel("X values")
plt.ylabel("Y values")
plt.show()

# ---- Bar Chart ----
plt.figure()
plt.bar(x, y)
plt.title("Bar Chart Example")
plt.xlabel("X values")
plt.ylabel("Y values")
plt.show()
```

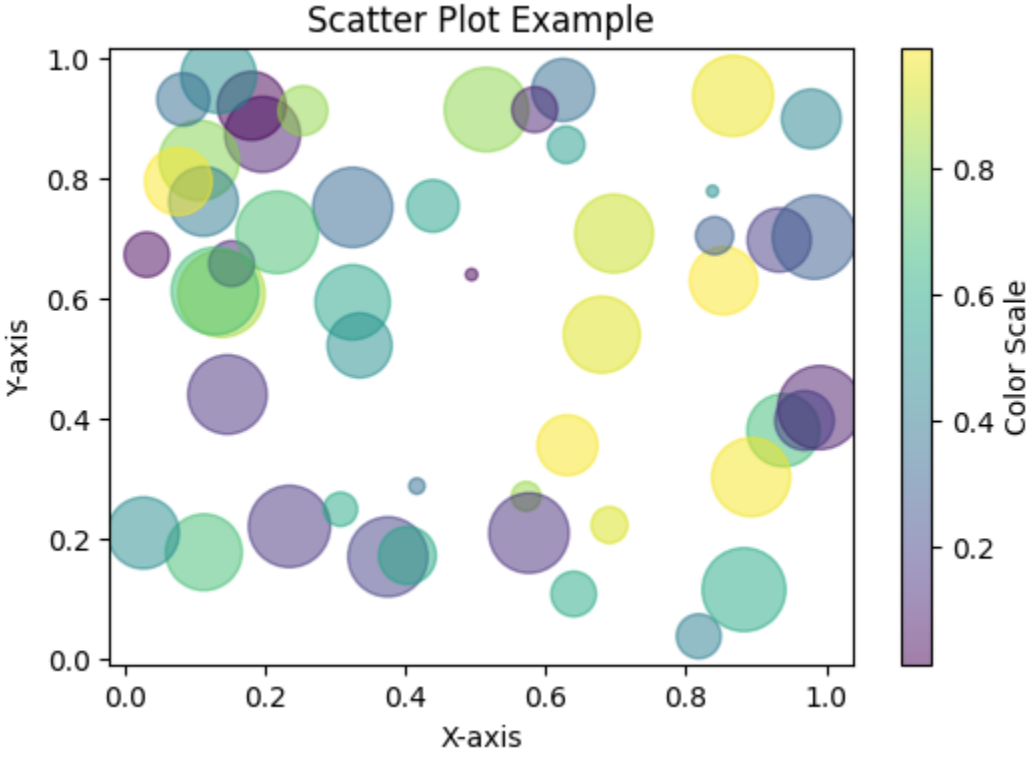

Line Chart Example
Y values
X values
20
18
16
14
12
10
8
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0

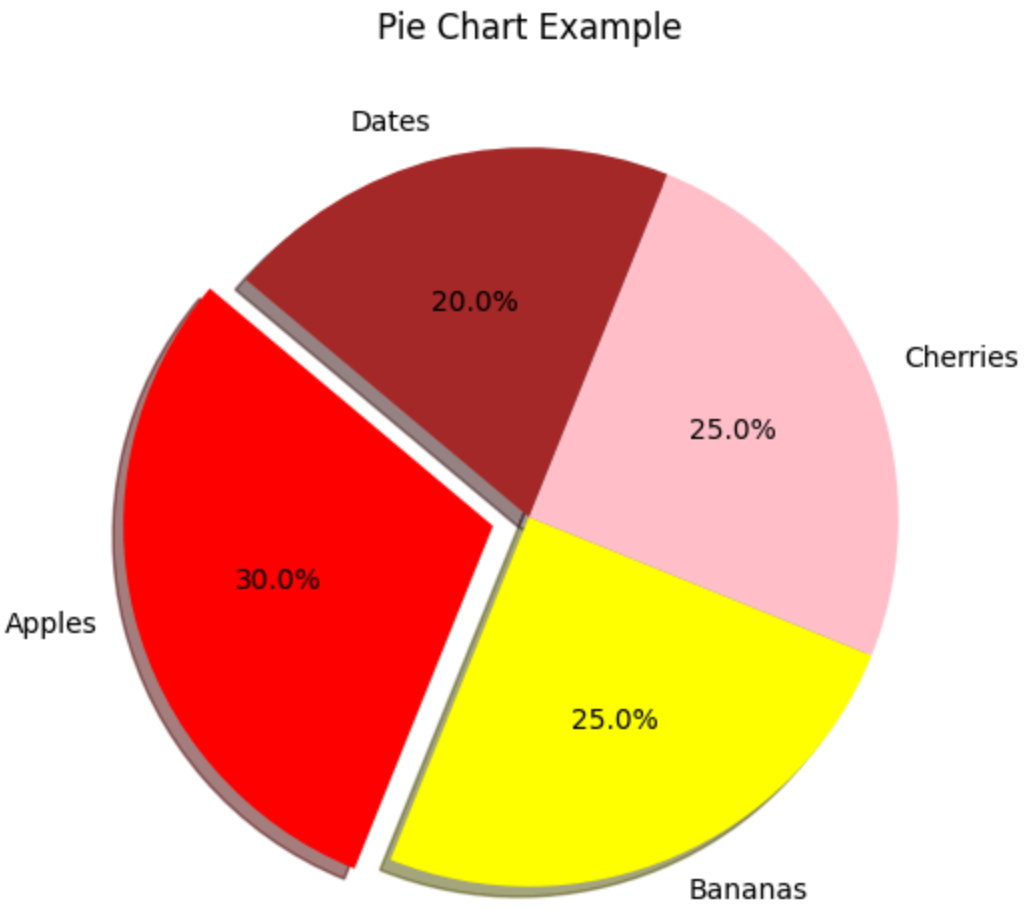

Bar Chart Example
Y values
X values
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
1
2
3
4
5

7. Create a histogram , scatter plot,pie chart using Matplotlib

In [2]:
```
# Import necessary libraries
import matplotlib.pyplot as plt
import numpy as np

# ------------------------------
# 1️⃣ Histogram
# ------------------------------
# Generate some random data
data = np.random.randn(1000)  # 1000 random numbers (Normal Distribution)

plt.figure(figsize=(6,4))
plt.hist(data, bins=20, color='skyblue', edgecolor='black')
plt.title('Histogram Example')
plt.xlabel('Values')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()

# ------------------------------
# 2️⃣ Scatter Plot
# ------------------------------
# Generate random data for scatter plot
x = np.random.rand(50)
y = np.random.rand(50)
colors = np.random.rand(50)
sizes = 1000 * np.random.rand(50)

plt.figure(figsize=(6,4))
plt.scatter(x, y, c=colors, s=sizes, alpha=0.5, cmap='viridis')
plt.title('Scatter Plot Example')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.colorbar(label='Color Scale')
plt.show()

# ------------------------------
# 3️⃣ Pie Chart
# ------------------------------
# Data for pie chart
labels = ['Apples', 'Bananas', 'Cherries', 'Dates']
sizes = [30, 25, 25, 20]
colors = ['red', 'yellow', 'pink', 'brown']
explode = (0.1, 0, 0, 0)  # explode first slice

plt.figure(figsize=(6,6))
plt.pie(sizes, labels=labels, colors=colors, explode=explode,
        autopct='%1.1f%%', shadow=True, startangle=140)
plt.title('Pie Chart Example')
plt.show()
```


Histogram Example
Frequency
Values
0
20
40
60
80
100
120
−3
−2
−1
0
1
2
3


Scatter Plot Example
Y-axis
X-axis
Color Scale
0.0
0.2
0.4
0.6
0.8
1.0


Pie Chart Example
Dates
20.0%
Cherries
25.0%
Apples
30.0%
25.0%
Bananas