1.To test whether the average weight of a species of birds differs from 150 grams. Procedure:

1. Null Hypothesis The average weight of the birds is 150 grams.
2. Alternative Hypothesis The average weight of the birds is not 150 grams.
3. Sample: Measure the weights of 30 birds randomly selected from the population.
4. Z-Test: Conduct a Z-test to compare the sample mean to 150 grams.
5. Decision Rule: Use a significance level of = 0.05.

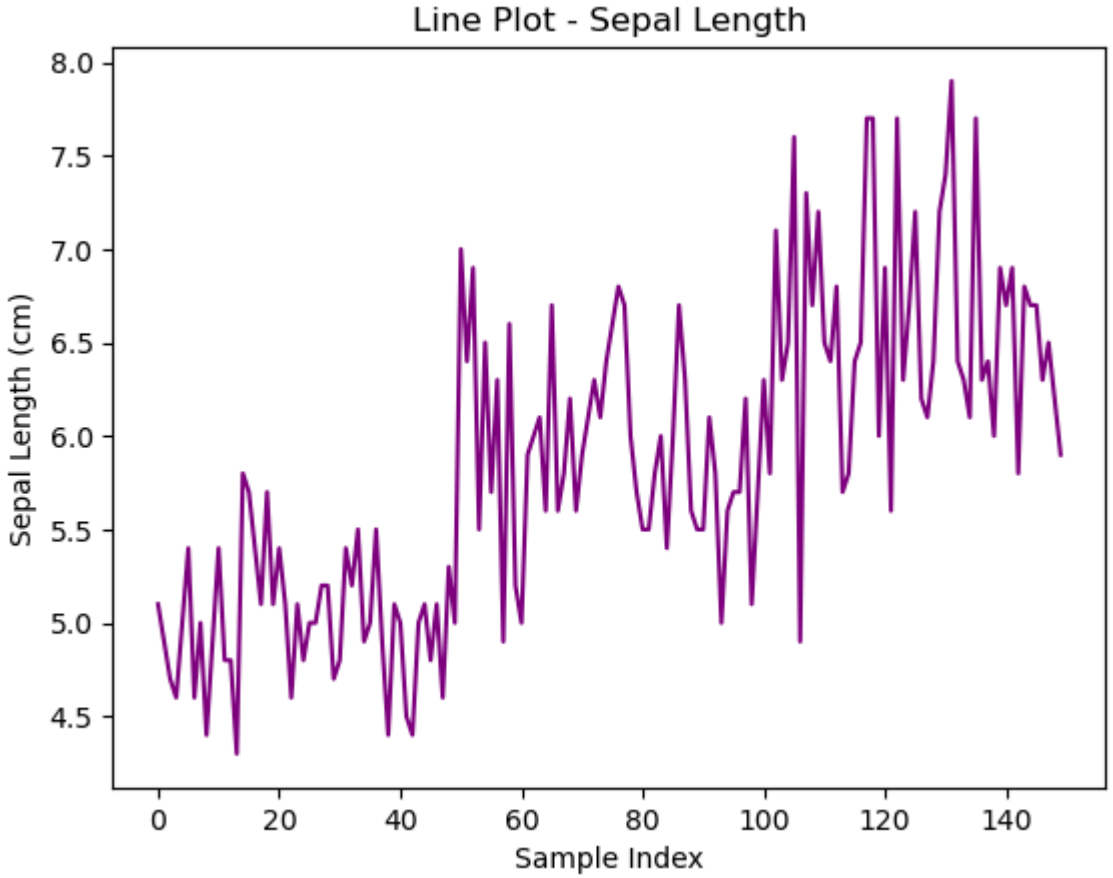
In [3]:
```
import numpy as np
from scipy import stats
sample_data = np.array([153, 151, 152, 154, 150, 153, 152, 155, 151,
152,
154, 153, 151, 150, 152, 153, 154, 152, 153,
151,
155, 152, 153, 151, 154, 152, 153, 154, 151,
152])
hypothesized_mean = 150
t_statistic, p_value = stats.ttest_1samp(a=sample_data,
popmean=hypothesized_mean)
print(f"Sample Mean: {np.mean(sample_data):.2f}")
print(f"T-Statistic: {t_statistic:.4f}")
print(f"P-Value: {p_value:.4f}")
alpha = 0.05
if p_value < alpha:
 print("Reject the null hypothesis: The average weight is significantly different from 150 grams.")
else:
 print("Fail to reject the null hypothesis: There is no significant difference in average weight from 150 grams.")
```

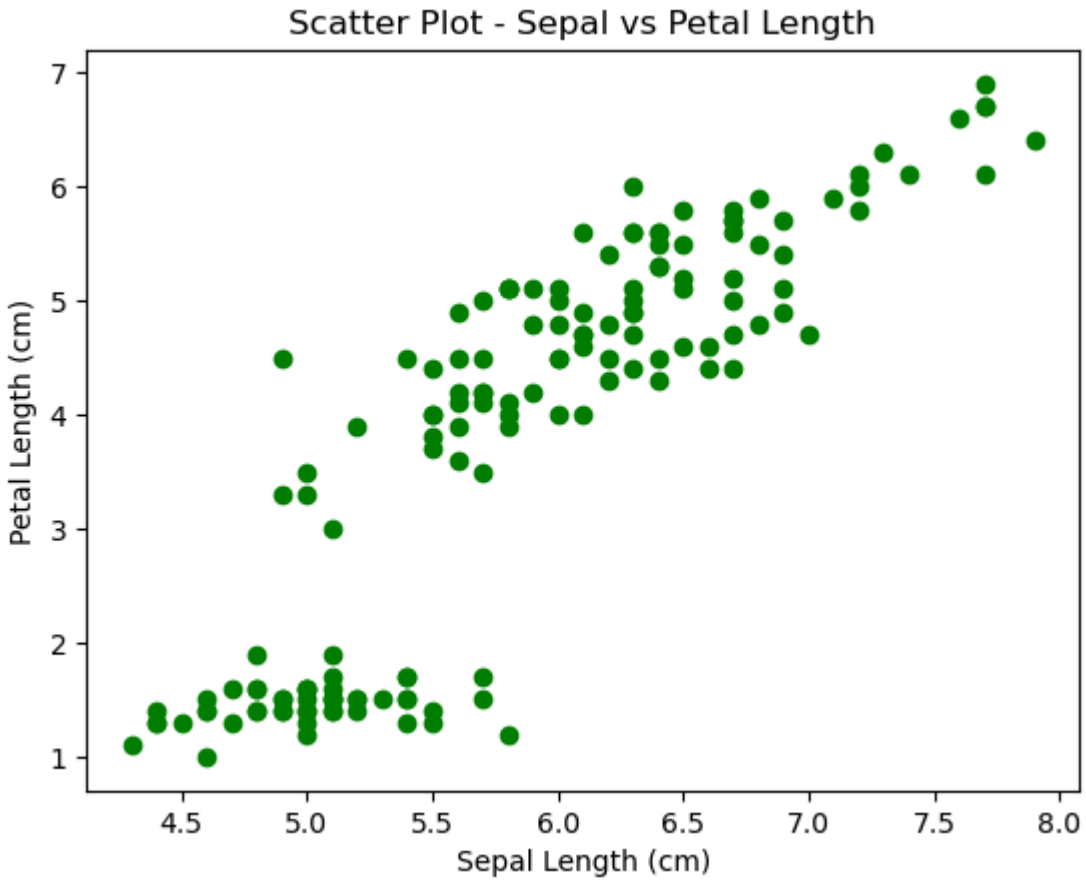
```
Sample Mean: 152.43
T-Statistic: 9.8249
P-Value: 0.0000
Reject the null hypothesis: The average weight is significantly different from 150 grams.
```

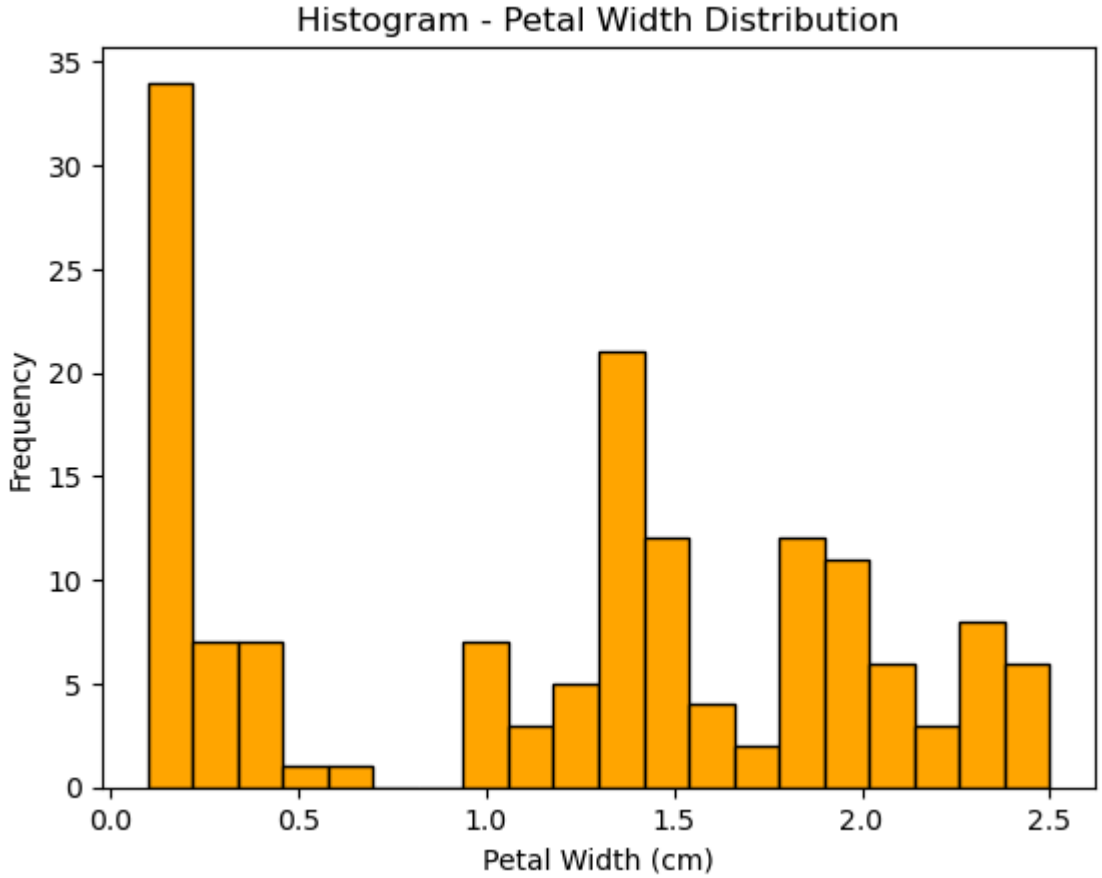
2.Experiment to understand Matplotlib library use cases in Data Science through visualization Description: Use Iris data set to understand the Matplot lib library

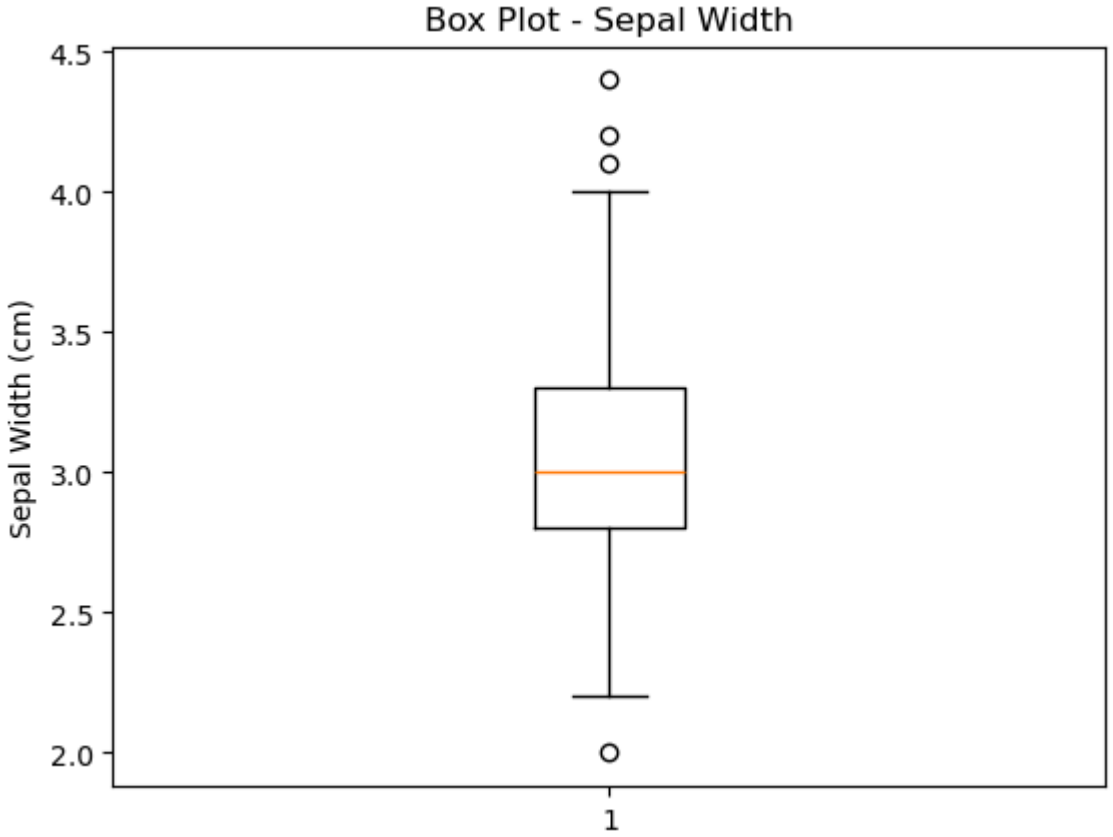
In [4]:
```
import seaborn as sns
import matplotlib.pyplot as plt
iris = sns.load_dataset('iris')

print(iris.head())
plt.plot(iris['sepal_length'], color='purple')
plt.title('Line Plot - Sepal Length')
plt.xlabel('Sample Index')
plt.ylabel('Sepal Length (cm)')
plt.show()
plt.scatter(iris['sepal_length'], iris['petal_length'], color='green')
plt.title('Scatter Plot - Sepal vs Petal Length')
plt.xlabel('Sepal Length (cm)')
plt.ylabel('Petal Length (cm)')
plt.show()
plt.hist(iris['petal_width'], bins=20, color='orange',
edgecolor='black')
plt.title('Histogram - Petal Width Distribution')
plt.xlabel('Petal Width (cm)')
plt.ylabel('Frequency')
plt.show()
plt.boxplot(iris['sepal_width'])
plt.title('Box Plot - Sepal Width')
plt.ylabel('Sepal Width (cm)')
plt.show()
species_count = iris['species'].value_counts()
plt.bar(species_count.index, species_count.values, color=['red',
'green', 'blue'])
plt.title('Bar Chart - Count of Each Iris Species')
plt.xlabel('Species')
plt.ylabel('Count')
plt.show()
```

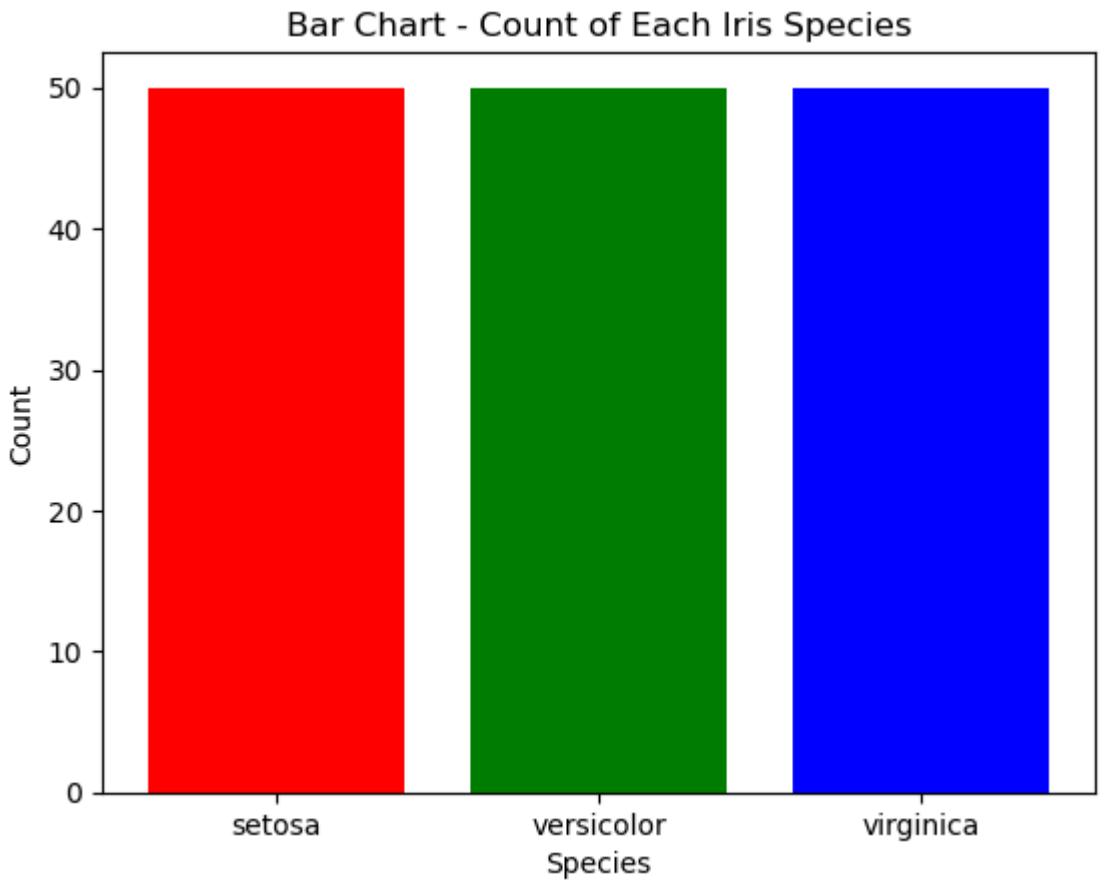
```
   sepal_length  sepal_width  petal_length  petal_width species
0           5.1          3.5           1.4          0.2  setosa
1           4.9          3.0           1.4          0.2  setosa
2           4.7          3.2           1.3          0.2  setosa
3           4.6          3.1           1.5          0.2  setosa
4           5.0          3.6           1.4          0.2  setosa
```


Line Plot - Sepal Length
Sepal Length (cm)
8.0
7.5
7.0
6.5
6.0
5.5
5.0
4.5
0
20
40
60
80
100
120
140
Sample Index


Scatter Plot - Sepal vs Petal Length
Petal Length (cm)
Sepal Length (cm)
1
2
3
4
5
6
7
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0


Histogram - Petal Width Distribution
Frequency
Petal Width (cm)
0
5
10
15
20
25
30
35
0.0
0.5
1.0
1.5
2.0
2.5


Box Plot - Sepal Width
Sepal Width (cm)
4.5
4.0
3.5
3.0
2.5
2.0
1



3.Experiment to understand array function in Data science. Description: Understand array function using Numpy library

In [5]:
```
import numpy as np
array = np.random.randint(1, 100, 9)
print(array)
print(np.sqrt(array))
print(array.ndim)
new_array = array.reshape(3, 3)
print(new_array)
print(new_array.ndim)
print(new_array.ravel())
newm = new_array.reshape(3, 3)
```

```
print(newm[2, 1:3])
print(newm[1:2, 1:3])

print(new_array[0:3, 0:0])
print(new_array[0:2, 0:1])
print(new_array[0:3, 0:1])
print(new_array[1:3])
```

```
[33 13 64 56 46 74 20 10 96]
[5.74456265 3.60555128 8.         7.48331477 6.78232998 8.60232527
 4.47213595 3.16227766 9.79795897]
1
[[33 13 64]
 [56 46 74]
 [20 10 96]]
2
[33 13 64 56 46 74 20 10 96]
[10 96]
[[46 74]]
[]
[[33]
 [56]]
[[33]
 [56]
 [20]]
[[56 46 74]
 [20 10 96]]
```

In [ ]:
```
4.Experiment to understand pandas library use cases in Data science. Description: Understand
data frame use cases using pandas library
```

In [7]:
```
import numpy as np
import pandas as pd

# First DataFrame creation and modification
lst = [[1, 'Smith', 50000], [2, 'Jones', 60000]]
df = pd.DataFrame(lst)
print(df)

df.columns = ['EmpId', 'Name', 'Salary']
print(df)
```

```
print(df.info())

# Second DataFrame with employee data
employee_data = {
    'emp id': [1, 2, 3, 4, 5, 6, 7],
    'name': ['SREE VARSSINI K S', 'SREEMATHI B', 'SREYA G',
             'SREYASKARI MULLAPUDI', 'SRI AKASH U G',
             'SRI HARSHAVARDHANAN R', 'SRI HARSHAVARDHANAN R'],
    'salary': [5000, 6000, 7000, 5000, 8000, 3000, 6000]
}

df = pd.DataFrame(employee_data)
print(df.head())
print(df.tail())
print(df.info())
print(df.salary)
print(type(df.salary))
print(df.salary.mean())
print(df.salary.median())
print(df.salary.mode())
print(df.salary.var())
print(df.salary.std())
print(df.describe())
print(df.describe(include='all'))

empCol = df.columns
print(empCol)

emparray = df.values
print(emparray)

employee_DF = pd.DataFrame(emparray, columns=empCol)
print(employee_DF)
```

```
   0      1      2
0  1  Smith  50000
1  2  Jones  60000
   EmpId   Name  Salary
0      1  Smith   50000
1      2  Jones   60000
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2 entries, 0 to 1
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   EmpId   2 non-null      int64
 1   Name    2 non-null      object
 2   Salary  2 non-null      int64
dtypes: int64(2), object(1)
memory usage: 180.0+ bytes
None
   emp id                  name  salary
0       1     SREE VARSSINI K S    5000
1       2           SREEMATHI B    6000
2       3               SREYA G    7000
3       4  SREYASKARI MULLAPUDI    5000
4       5         SRI AKASH U G    8000
   emp id                   name  salary
2       3                SREYA G    7000
3       4   SREYASKARI MULLAPUDI    5000
4       5          SRI AKASH U G    8000
5       6  SRI HARSHAVARDHANAN R    3000
6       7  SRI HARSHAVARDHANAN R    6000
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7 entries, 0 to 6
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   emp id  7 non-null      int64
 1   name    7 non-null      object
 2   salary  7 non-null      int64
dtypes: int64(2), object(1)
memory usage: 300.0+ bytes
None
0    5000
```

```
1    6000
2    7000
3    5000
4    8000
5    3000
6    6000
Name: salary, dtype: int64
<class 'pandas.core.series.Series'>
5714.285714285715
6000.0
0    5000
1    6000
Name: salary, dtype: int64
2571428.5714285714
1603.5674514745463
         emp id       salary
count  7.000000     7.000000
mean   4.000000  5714.285714
std    2.160247  1603.567451
min    1.000000  3000.000000
25%    2.500000  5000.000000
50%    4.000000  6000.000000
75%    5.500000  6500.000000
max    7.000000  8000.000000
          emp id                    name       salary
count   7.000000                       7     7.000000
unique       NaN                       6          NaN
top          NaN  SRI HARSHAVARDHANAN R          NaN
freq         NaN                       2          NaN
mean    4.000000                     NaN  5714.285714
std     2.160247                     NaN  1603.567451
min     1.000000                     NaN  3000.000000
25%     2.500000                     NaN  5000.000000
50%     4.000000                     NaN  6000.000000
75%     5.500000                     NaN  6500.000000
max     7.000000                     NaN  8000.000000
Index(['emp id', 'name', 'salary'], dtype='object')
[[1 'SREE VARSSINI K S' 5000]
 [2 'SREEMATHI B' 6000]
 [3 'SREYA G' 7000]
 [4 'SREYASKARI MULLAPUDI' 5000]
```

```
[5 'SRI AKASH U G' 8000]
[6 'SRI HARSHAVARDHANAN R' 3000]
[7 'SRI HARSHAVARDHANAN R' 6000]]
```

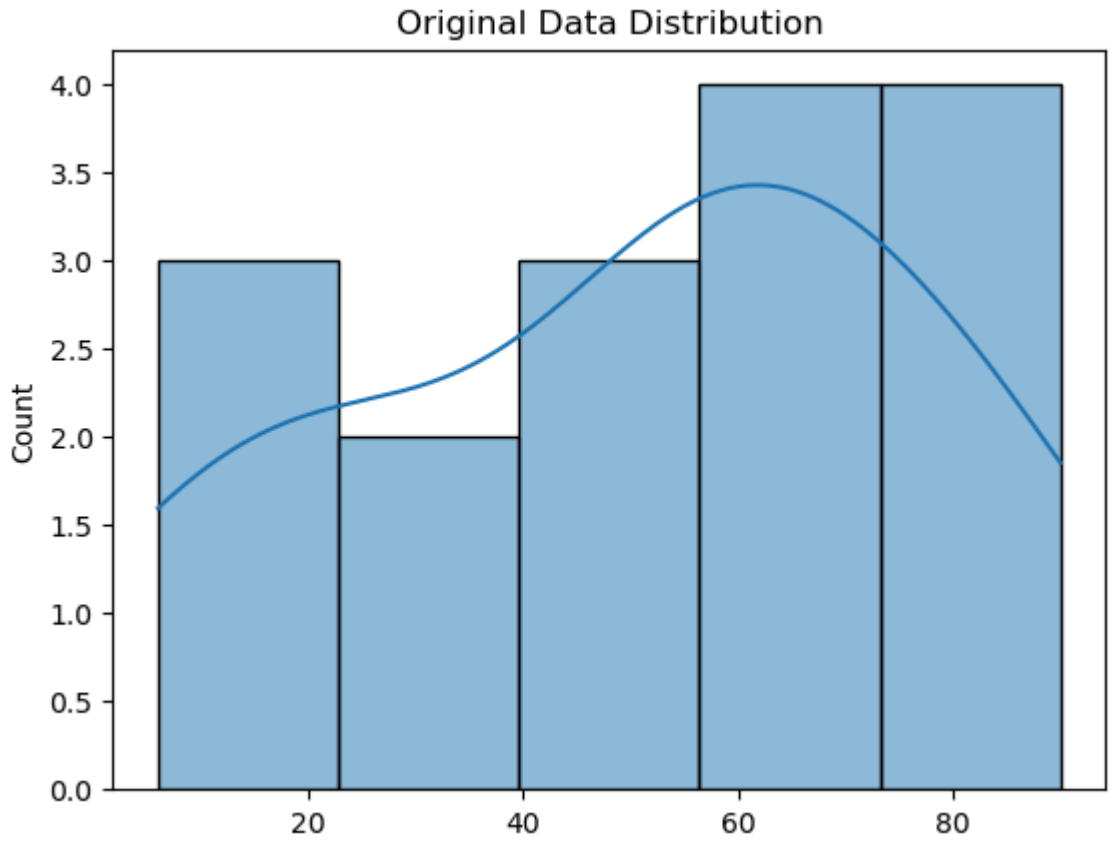
	emp id	name	salary
0	1	SREE VARSSINI K S	5000
1	2	SREEMATHI B	6000
2	3	SREYA G	7000
3	4	SREYASKARI MULLAPUDI	5000
4	5	SRI AKASH U G	8000
5	6	SRI HARSHAVARDHANAN R	3000
6	7	SRI HARSHAVARDHANAN R	6000

In [ ]:
```
5.Experiment to detect outliers in a given data set. Description: Understand the procedure to
identify the outliers in a given dataset
```

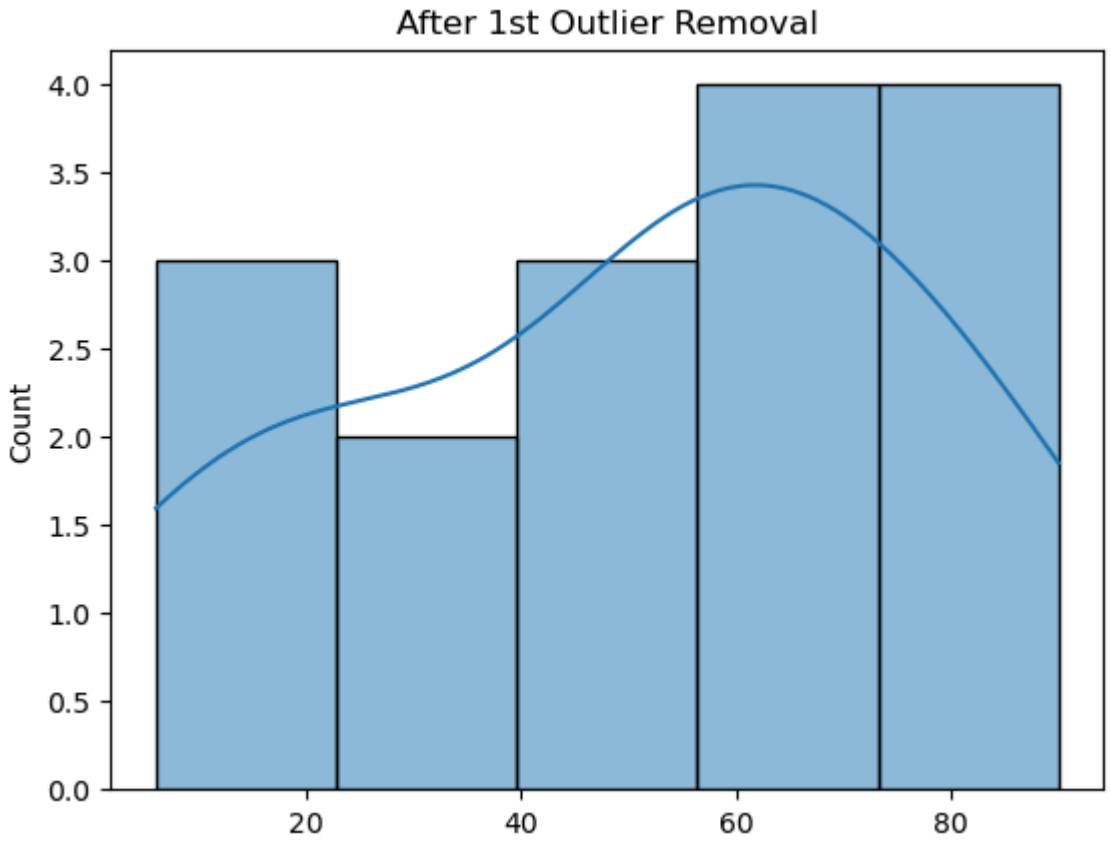
In [9]:
```
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

array = np.array([27, 50, 44, 6, 58, 61, 23, 86, 67, 20, 75, 7, 79, 61, 90, 54])
print(array)
print("Mean:", array.mean())
print("25th percentile:", np.percentile(array, 25))
print("50th percentile (median):", np.percentile(array, 50))
print("75th percentile:", np.percentile(array, 75))
print("100th percentile (max):", np.percentile(array, 100))

def outDetection(arr):
    Q1, Q3 = np.percentile(arr, [25, 75])
    IQR = Q3 - Q1
    lr = Q1 - (1.5 * IQR)
    ur = Q3 + (1.5 * IQR)
    return lr, ur

lr, ur = outDetection(array)
print(f"Lower range: {lr}, Upper range: {ur}")

# Plot original data distribution
sns.histplot(array, kde=True)
plt.title("Original Data Distribution")
```

```
plt.show()

# Filter outliers
new_array = array[(array > lr) & (array < ur)]
print("After 1st outlier removal:", new_array)

sns.histplot(new_array, kde=True)
plt.title("After 1st Outlier Removal")
plt.show()

lr1, ur1 = outDetection(new_array)
print(f"New lower range: {lr1}, New upper range: {ur1}")

# Further filter the array if necessary
final_array = new_array[(new_array > lr1) & (new_array < ur1)]
print("After 2nd outlier removal:", final_array)

sns.histplot(final_array, kde=True)
plt.title("After 2nd Outlier Removal")
plt.show()
```

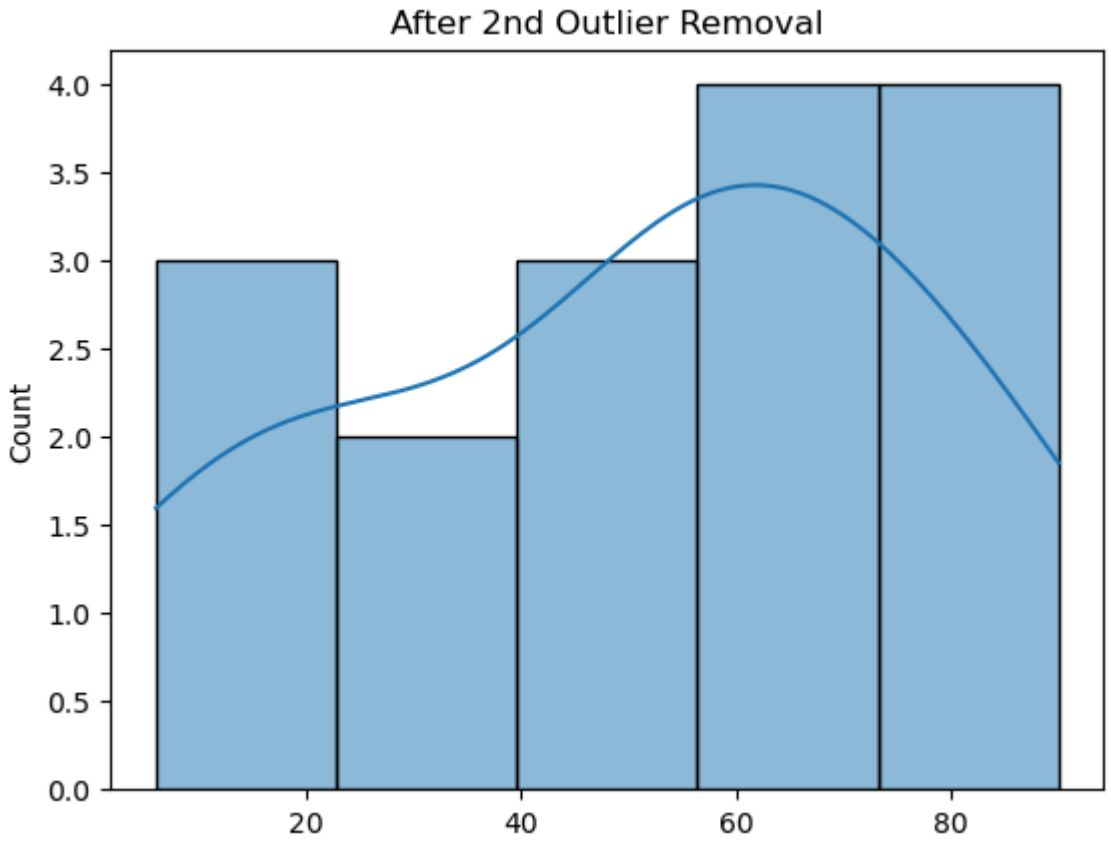
```
[27 50 44  6 58 61 23 86 67 20 75  7 79 61 90 54]
Mean: 50.5
25th percentile: 26.0
50th percentile (median): 56.0
75th percentile: 69.0
100th percentile (max): 90.0
Lower range: -38.5, Upper range: 133.5
```




```
After 1st outlier removal: [27 50 44  6 58 61 23 86 67 20 75  7 79 61 90 54]
```




```
New lower range: -38.5, New upper range: 133.5
After 2nd outlier removal: [27 50 44  6 58 61 23 86 67 20 75  7 79 61 90 54]
```


After 2nd Outlier Removal
Count
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
20
40
60
80

In [ ]: