1: Conduct Z test for the Data Given using Python Objective: To test whether the average weight of a species of birds differs from 150 grams.

In [3]:

```
import numpy as np
import scipy.stats as stats
sample_data = np.array([152, 148, 151, 149, 147, 153, 150, 148, 152, 149,
151, 150, 149, 152, 151, 148, 150, 152, 149, 150,
148, 153, 151, 150, 149, 152, 148, 151, 150, 153])
       population_mean = 150 sample_mean = np.mean(sample_data) sample_std =
np.std(sample_data,   ddof=1)   n   =   len(sample_data)   z_statistic    =
(sample_mean - population_mean) / (sample_std / np.sqrt(n)) p_value = 2 *
(1    -    stats.norm.cdf(np.abs(z_statistic)))    print(f"Sample    Mean:
{sample_mean:.2f}")  print(f"Z-Statistic:  {z_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 diffe
else:
    print("Fail to reject the null hypothesis: There is no significant differenc
```

```
Sample Mean: 150.20
Z-Statistic: 0.6406
P-Value: 0.5218
Fail to reject the null hypothesis: There is no significant difference in average
weight from 150 grams.
```

jupyter EXP-10 Last Checkpoint: 5 minutes ago

File Edit View Run Kernel Settings Help

Code JupyterLab Python 3 (ipykernel) ○

Exp No:10 Experiment to understand array function in Data science. Description: Understan
array function using Numpy library

[5]:
```
import numpy as np
array=np.random.randint(1,100,9)
array
```

[5]:
```
array([63, 44, 70, 75, 89, 59, 75, 19, 41], dtype=int32)
```

[6]:
```
np.sqrt(array)
```

[6]:
```
array([7.93725393, 6.63324958, 8.36660027, 8.66025404, 9.43398113,
       7.68114575, 8.66025404, 4.35889894, 6.40312424])
```

[3]:
```
array.ndim
```

[3]:
```
1
```

[4]:
```
import numpy as np
array=np.random.randint(1,100,9)
new_array=array.reshape(3,3)
new_array
```

[4]:
```
array([[20, 18, 49],
[15, 65, 62],
[46, 69, 59]], dtype=int32)
```

[7]:
```
new_array.ravel()
```

[7]:
```
array([20, 18, 49, 15, 65, 62, 46, 69, 59], dtype=int32)
```

[8]:
```
newm=new_array.reshape(3,3)
newm
```

Experiment to understand pandas library use cases in Data science. Description: Understand data frame use cases using pandas library

In [2]:

```
import pandas as pd
import numpy as np
data = {
    'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Eva'],
    'Age': [25, 30, 35, np.nan, 40],
    'City': ['New York', 'Paris', 'London', 'Berlin', 'Tokyo'],
    'Salary': [50000, 54000, 58000, 62000, np.nan]
}

df = pd.DataFrame(data)
print("Original DataFrame:\n", df)
print("\nBasic Info:")
print(df.info())
print("\nSummary Statistics:")
print(df.describe())
print("\nFirst 3 Rows:")
print(df.head(3))
print("\nLast 3 rows:")
print(df.tail(3))
df['Age'].fillna(df['Age'].mean())
df['Salary'].fillna(df['Salary'].mean())
sorted_df = df.sort_values(by='Salary', ascending=False)
print("\nCleaned and Sorted DataFrame:\n", sorted_df)
print("\nAges:\n", df['Age'])
high_salary = df[df['Salary'] > 55000]
print("\nEmployees with Salary > 55000:\n", high_salary)
df['Experience'] = [2, 4, 6, 8, 10]
print("\nAverage Salary by City:\n", df.groupby('City')['Salary'].mean())
```

```
Original DataFrame:
       Name   Age      City   Salary
0   1   2 AliCharlie5.0 New York 50000.0
35.0 NaNBEvba 340.00      Paris 54000.0
                      London 58000.0
3    David            Berlin 62000.0
4                    Tokyo       NaN

Basic Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5 entries, 0 to 4
Data columns (total 4 columns):
 #       Column Non-Null Count Dtype
    --- ------ ------------- -----
Na0me          5 non-null     object
Ag1e           4 non-null     float64
Ci2ty         5 non-null     object
S3alary 4 non-null           float64
dtypes: float64(2), object(2)
memory usage: 292.0+ bytes
None

Summary Statistics:
              Age       Salary
count   4.000000      4.000000
mean     32.500000 56000.000000
std     6.454972    5163.977795
min      25.000000 50000.000000
25%      28.750000 53000.000000
50%      32.500000 56000.000000
75%      36.250000 59000.000000
max      40.000000 62000.000000

First 3 Rows:
      Name   Age      City    Salary
0   1   2 AliCharlie5.0 New York 50000.0
35.0    Bob 30.0      Paris 54000.0
                      London 58000.0

Last 3 rows:
      Name   Age    City    Salary
     2 Charlie 35.0 London 58000.0
3    David      NaN Berlin 62000.0
4      Eva 40.0    Tokyo       NaN

Cleaned and Sorted DataFrame:
       Name   Age      City   Salary
3    David   NaN      Berlin 62000.0
2 Charlie 35.0        London 58000.0
1Bob 30.0              Paris 54000.0
0       Alice 25.0 New York 50000.0
4      Eva 40.0      Tokyo       NaN

Ages:
0 1 2 25.0
3 4  30.0
     35.0
      NaN
     40.0
Name: Age, dtype: float64
```

```
Employees with Salary > 55000:
       Name   Age    City   Salary
     2 Charlie 35.0 London 58000.0
3    David      NaN Berlin 62000.0

Average Salary by City:
 City
Berlin      62000.0
London      58000.0
New York    50000.0
Paris       54000.0
Tokyo           NaN
Name: Salary, dtype: float64
```

In [ ]:

Experiment to detect outliers in a given data set.

In [39]:
```
import numpy as np
array=np.random.randint(1,100,16)
array
```

Out[39]:
```
array([ 2, 79, 19, 17, 1, 79, 97, 11, 46, 80, 45, 46, 27, 28,   7, 65],
      dtype=int32)
```

In [40]:
```
array.mean()
```

Out[40]:
```
np.float64(40.5625)
```

In [41]:
```
np.percentile(array,25)
```

Out[41]:
```
np.float64(15.5)
```

In [42]:
```
np.percentile(array,50)
```

Out[42]:
```
np.float64(36.5)
```

In [43]:
```
np.percentile(array,75)
```

Out[43]:
```
np.float64(68.5)
```

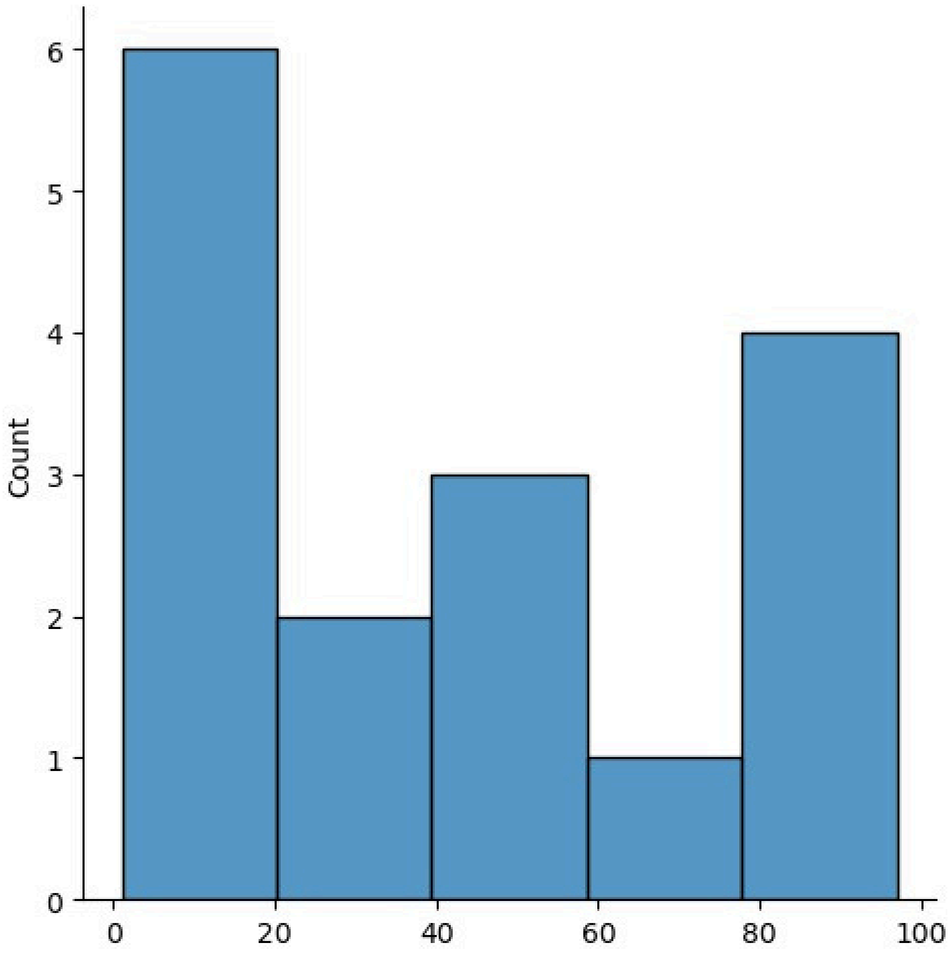
In [44]:
```
np.percentile(array,100)
```

Out[44]:
```
np.float64(97.0)
```

In [45]:
```
def outDetection(array):
    sorted(array)
    Q1,Q3=np.percentile(array,[25,75])
    IQR=Q3-Q1   lr=Q1-(1.5*IQR)   ur=Q3+
    (1.5*IQR) return lr,ur


lr,ur=outDetection(array)
lr,ur
```

Out[45]:
```
(np.float64(-64.0), np.float64(148.0))
```

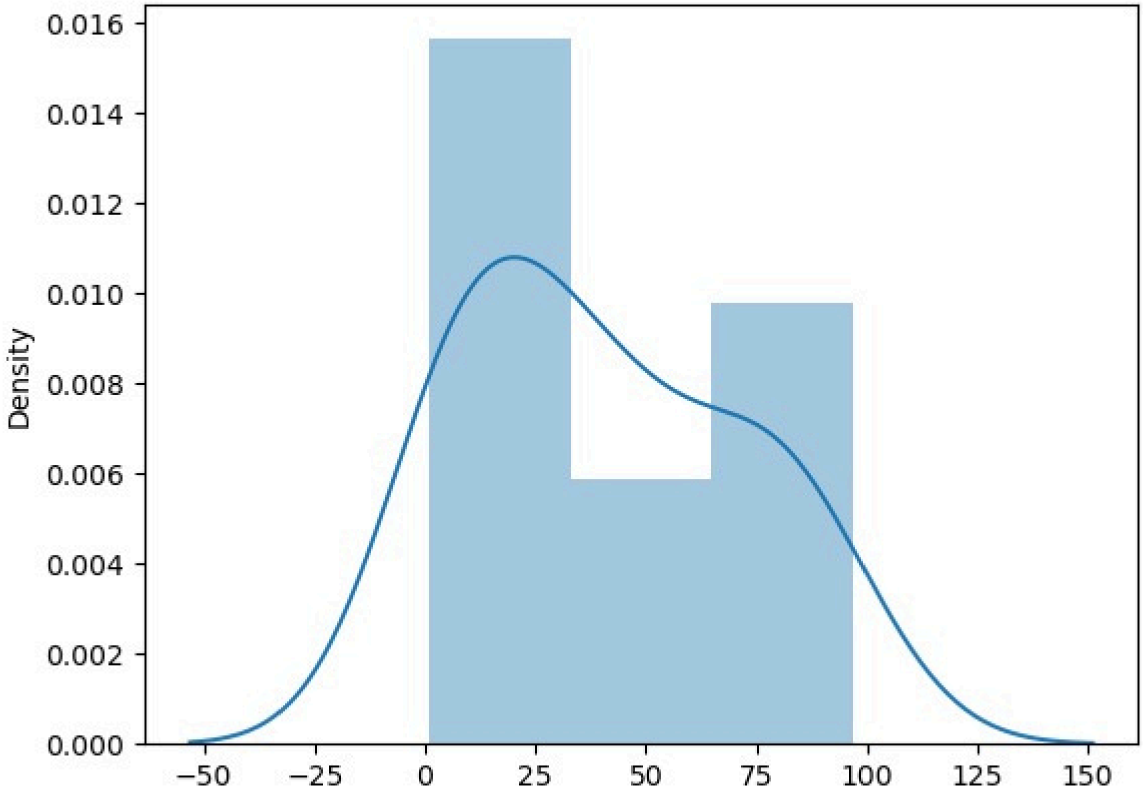
In [46]:
```
import seaborn as sns
import matplotlib.pyplot as plt
sns.displot(array)
plt.show()
```



In [47]:

```
sns.distplot(array)
plt.show()
```

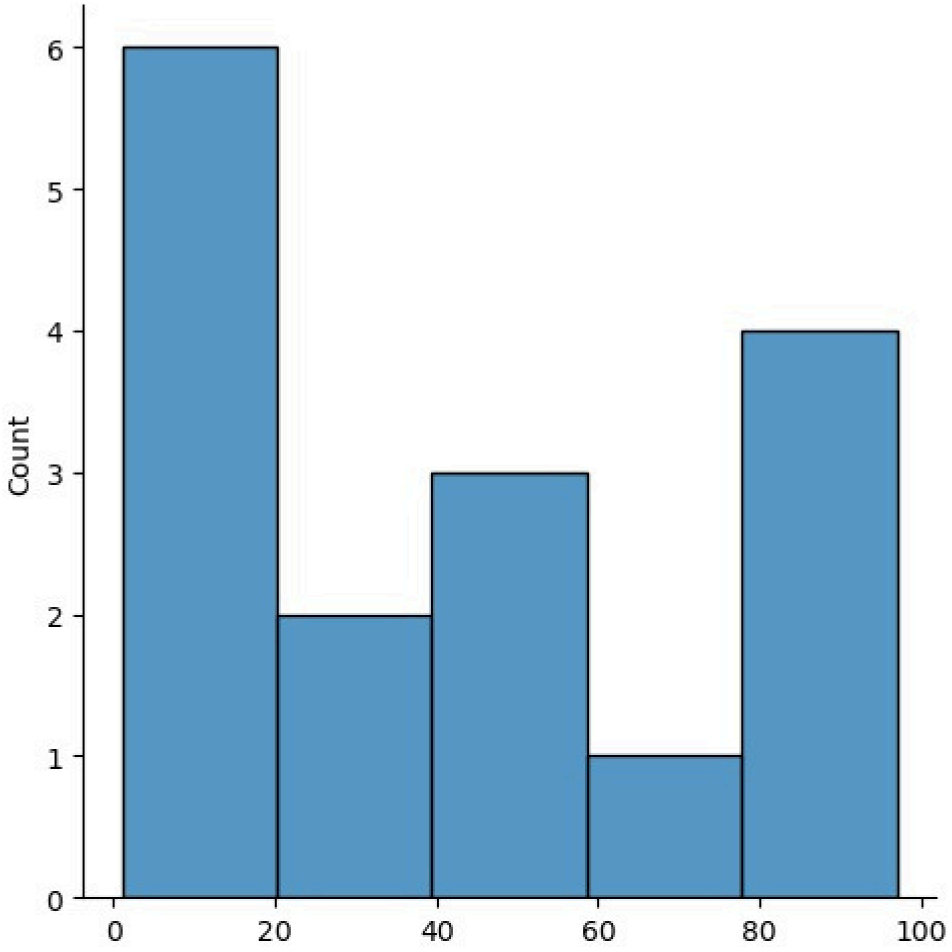
```
C:\Users\Kaviya\AppData\Local\Temp\ipykernel_35088\2106234926.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(array)
```



```
In [51]: new_array=array[(array>lr)&(array<ur)]
         new_array
```

```
Out[51]: array([ 2, 79, 19, 17, 1, 79, 97, 11, 46, 80, 45, 46, 27, 28,   7, 65],
               dtype=int32)
```

```
In [52]: sns.displot(new_array)
         plt.show()
```



```
In [53]: lr1,ur1=outDetection(new_array)
         lr1,ur1
```

```
Out[53]: (np.float64(-64.0), np.float64(148.0))
```