NAME : DARSHAN SHASHIKANT

PRN: 123M1H041

ADVANCE DATA SCIENCE ASSIGNMENT-1

In [ ]:

```
import pandas as pd

# Load the dataset
file_path = '/content/top_intelligent_people_in_the_world_5000.csv'
df = pd.read_csv(file_path)

# Display column names and first few rows of the dataset
print("Columns in the dataset:")
print(df.columns)

print("\nFirst few rows of the dataset:")
print(df.head())
```

```
Columns in the dataset:
Index(['Name', 'Country', 'Field of Expertise', 'IQ', 'Achievements',
       'Birth Year', 'Gender', 'Notable Works', 'Awards', 'Education',
       'Influence'],
      dtype='object')

First few rows of the dataset:
                Name  Country Field of Expertise   IQ  \
0       Enrico Fermi  Austria           Polymath  199
1         Max Planck    Italy          Chemistry  159
2         Paul Dirac       UK            Physics  177
3  Erwin Schrödinger    Italy            Physics  130
4         Paul Dirac       UK            Physics  163

                Achievements  Birth Year  Gender  \
0  Father of Computer Science        1968  Female
1         Theory of Evolution        1986  Female
2           Quantum Mechanics        1927  Female
3   Electromagnetic Induction        1921  Female
4               Wave Equation        1964  Female

                             Notable Works               Awards  \
0                                     E=mc²  Numerous Posthumous
1                               Bohr Model          Nobel Prize
2                                   Cosmos          Nobel Prize
3  Discovery of Electromagnetic Induction          Nobel Prize
4                    On Computable Numbers          Nobel Prize

                 Education                           Influence
0              Self-taught      Popularizing science and cosmology
1       Ph.D. in Astronomy  Foundational work in quantum mechanics
2      Ph.D. in Mathematics      Foundation of classical mechanics
3  University of Cambridge  Iconic Renaissance artist and inventor
4         Ph.D. (honorary)  Foundational work in quantum mechanics
```

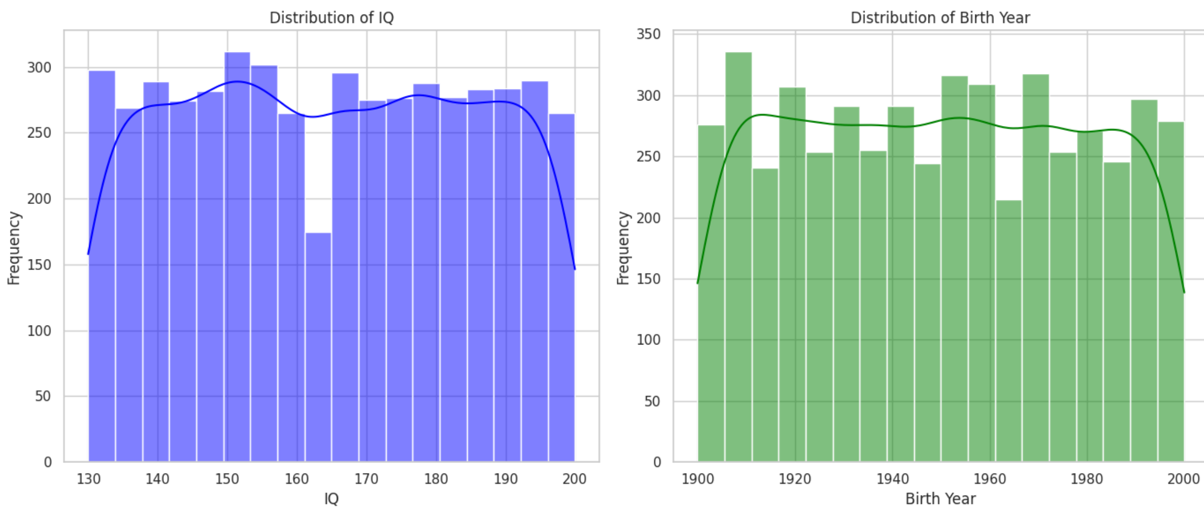
In [ ]:

```
df['IQ'] = pd.to_numeric(df['IQ'], errors='coerce')
df['Birth Year'] = pd.to_numeric(df['Birth Year'], errors='coerce')

# Check for missing values
missing_values = df.isnull().sum()
print("Missing values in each column:")
print(missing_values)

# Drop rows with missing 'IQ' or 'Birth Year'
df_cleaned = df.dropna(subset=['IQ', 'Birth Year'])

# Check data types and cleaned dataset
print("\nData types of each column:")
print(df_cleaned.dtypes)

print("\nFirst few rows of the cleaned dataset:")
print(df_cleaned.head())
```

```
Missing values in each column:
Name                     0
Country                  0
Field of Expertise       0
IQ                       0
Achievements             0
Birth Year               0
Gender                   0
Notable Works            0
Awards                1249
Education                0
Influence                0
dtype: int64

Data types of each column:
Name                  object
Country               object
Field of Expertise    object
IQ                     int64
Achievements          object
Birth Year             int64
Gender                object
Notable Works         object
Awards                object
Education             object
Influence             object
dtype: object

First few rows of the cleaned dataset:
                Name  Country Field of Expertise   IQ  \
0       Enrico Fermi  Austria           Polymath  199
1         Max Planck    Italy          Chemistry  159
2         Paul Dirac       UK            Physics  177
3  Erwin Schrödinger    Italy            Physics  130
4         Paul Dirac       UK            Physics  163

                 Achievements  Birth Year  Gender  \
0  Father of Computer Science        1968  Female
1         Theory of Evolution        1986  Female
2           Quantum Mechanics        1927  Female
3   Electromagnetic Induction        1921  Female
4               Wave Equation        1964  Female

                            Notable Works               Awards  \
0                                    E=mc²  Numerous Posthumous
1                               Bohr Model          Nobel Prize
2                                   Cosmos          Nobel Prize
3  Discovery of Electromagnetic Induction          Nobel Prize
4                     On Computable Numbers          Nobel Prize

                 Education                               Influence
0              Self-taught      Popularizing science and cosmology
1       Ph.D. in Astronomy  Foundational work in quantum mechanics
2     Ph.D. in Mathematics       Foundation of classical mechanics
3  University of Cambridge  Iconic Renaissance artist and inventor
4         Ph.D. (honorary)  Foundational work in quantum mechanics
```

In [ ]:

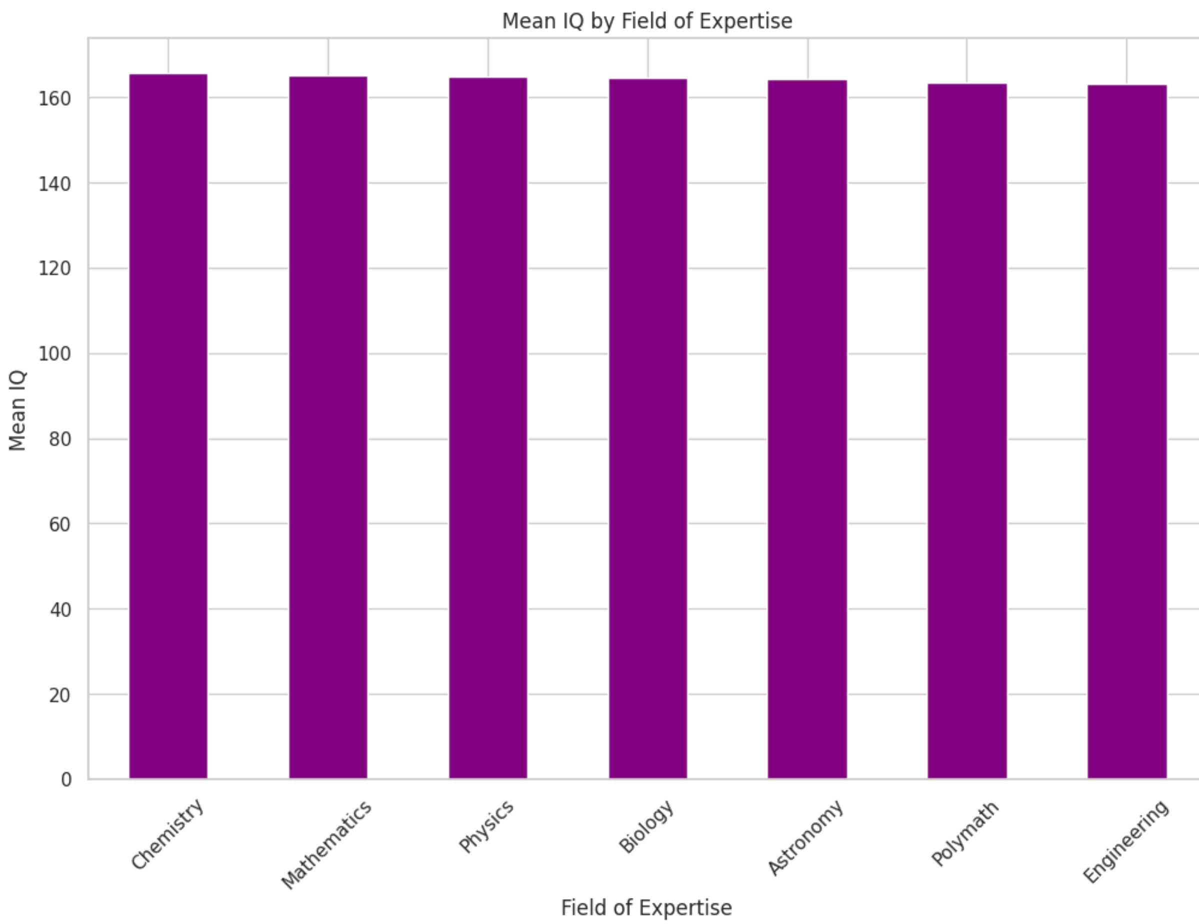
```
import matplotlib.pyplot as plt
import seaborn as sns

# Set the style of the visualization
sns.set(style="whitegrid")

# Create histograms for 'IQ' and 'Birth Year'
plt.figure(figsize=(14, 6))

# Plot histogram for 'IQ'
plt.subplot(1, 2, 1)
sns.histplot(df_cleaned['IQ'], kde=True, color='blue')
plt.title('Distribution of IQ')
plt.xlabel('IQ')
plt.ylabel('Frequency')
# Plot histogram for 'Birth Year'
plt.subplot(1, 2, 2)
sns.histplot(df_cleaned['Birth Year'], kde=True, color='green')
plt.title('Distribution of Birth Year')
plt.xlabel('Birth Year')
plt.ylabel('Frequency')

plt.tight_layout()
plt.show()
```



In [ ]:

```
field_of_expertise_mean_iq = df_cleaned.groupby('Field of Expertise')['IQ'].mean().

# Display mean IQ by field of expertise
print("Mean IQ by Field of Expertise:")
print(field_of_expertise_mean_iq)

# Plot mean IQ by field of expertise
plt.figure(figsize=(12, 8))
field_of_expertise_mean_iq.plot(kind='bar', color='purple')
plt.title('Mean IQ by Field of Expertise')
plt.xlabel('Field of Expertise')
plt.ylabel('Mean IQ')
plt.xticks(rotation=45)
plt.show()
```

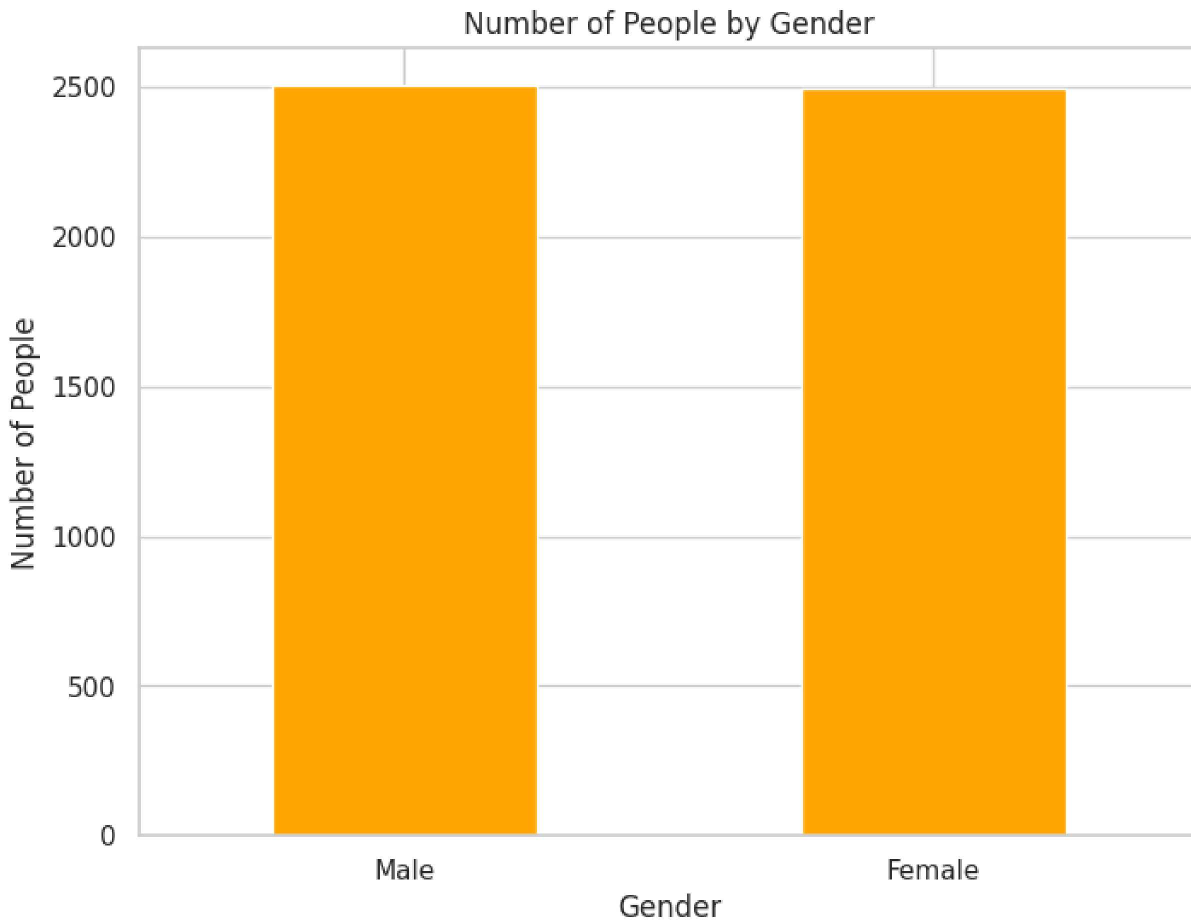
```
Mean IQ by Field of Expertise:
Field of Expertise
Chemistry      165.648330
Mathematics    165.058586
Physics        164.898328
Biology        164.629344
Astronomy      164.297552
Polymath       163.362869
Engineering    163.160920
Name: IQ, dtype: float64
```



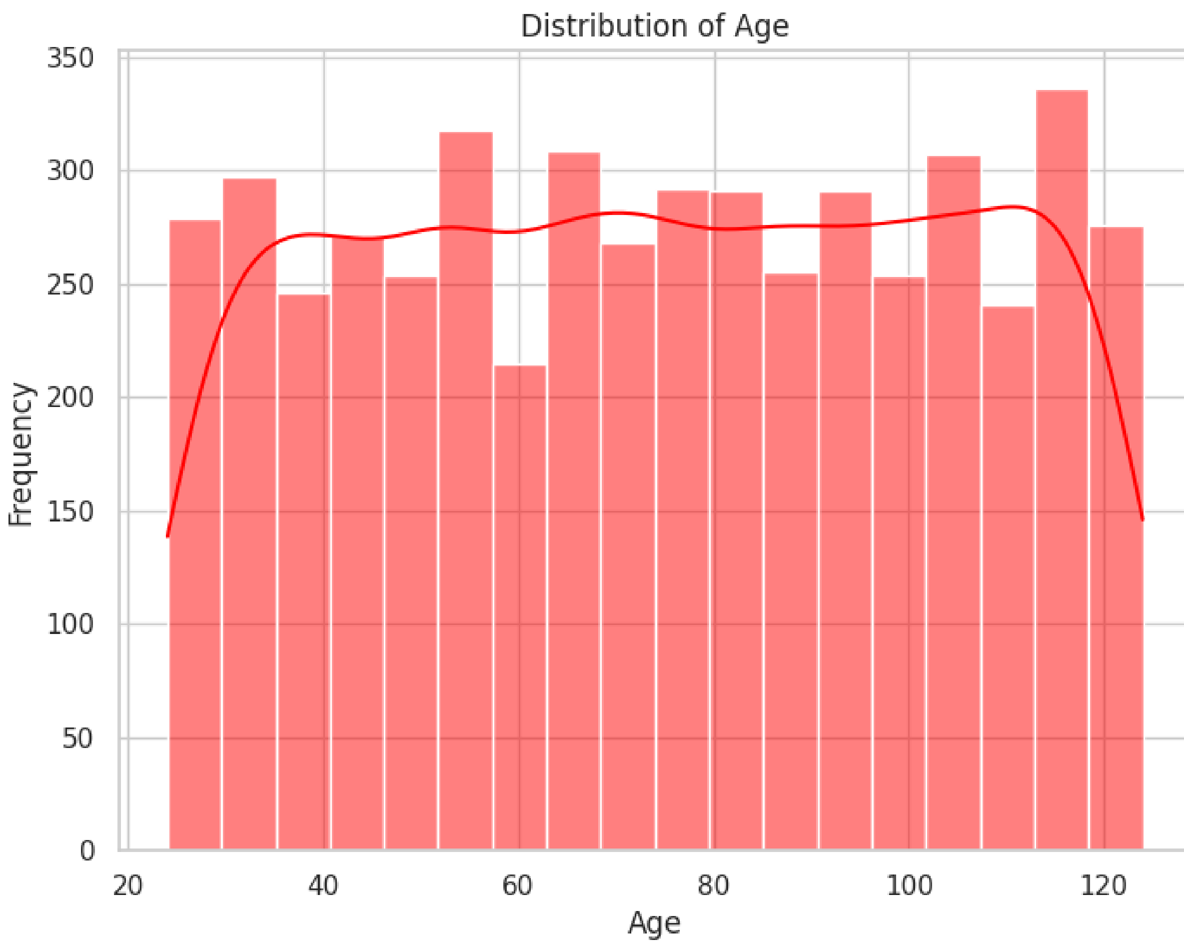
In [ ]:

```
gender_counts = df_cleaned['Gender'].value_counts()

# Display number of people by gender
print("\nNumber of people by Gender:")
print(gender_counts)

# Plot number of people by gender
plt.figure(figsize=(8, 6))
gender_counts.plot(kind='bar', color='orange')
plt.title('Number of People by Gender')
plt.xlabel('Gender')
plt.ylabel('Number of People')
plt.xticks(rotation=0)
plt.show()
```

```
Number of people by Gender:
Gender
Male      2506
Female    2494
Name: count, dtype: int64
```



In [ ]:
```python
current_year = pd.Timestamp.now().year
df_cleaned['Age'] = current_year - df_cleaned['Birth Year']

# Plot histogram for 'Age'
plt.figure(figsize=(8, 6))
sns.histplot(df_cleaned['Age'], kde=True, color='red')
plt.title('Distribution of Age')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.show()
```



2) Assignments based on Python Programming and Statistical Concepts: [COI]

In [ ]:
```
#a) Write a Python program to create/access/update the lists, tuples, sets, and dic

# Lists
my_list = [1, 2, 3, 'apple', 'banana']

# Accessing elements
print("First element:", my_list[0])
print("Last element:", my_list[-1])

# Updating elements
my_list[2] = 'orange'
print("Updated list:", my_list)

# Adding elements
my_list.append('grape')
print("List after adding element:", my_list)

# Removing elements
my_list.remove('apple')
```

```
print("List after removing element:", my_list)


# Tuples
my_tuple = (1, 2, 3, 'apple', 'banana')

# Accessing elements
print("First element:", my_tuple[0])
print("Last element:", my_tuple[-1])

# Tuples are immutable, so elements cannot be updated


# Sets
my_set = {1, 2, 3, 'apple', 'banana'}

# Adding elements
my_set.add('orange')
print("Set after adding element:", my_set)

# Removing elements
my_set.remove('apple')
print("Set after removing element:", my_set)


# Dictionaries
my_dict = {'name': 'John', 'age': 30, 'city': 'New York'}

# Accessing values
print("Name:", my_dict['name'])

# Updating values
my_dict['age'] = 35
print("Updated dictionary:", my_dict)

# Adding new key-value pairs
my_dict['country'] = 'USA'
print("Dictionary after adding key-value pair:", my_dict)

# Removing key-value pairs
del my_dict['city']
print("Dictionary after removing key-value pair:", my_dict)
```

```
First element: 1
Last element: banana
Updated list: [1, 2, 'orange', 'apple', 'banana']
List after adding element: [1, 2, 'orange', 'apple', 'banana', 'grape']
List after removing element: [1, 2, 'orange', 'banana', 'grape']
First element: 1
Last element: banana
Set after adding element: {1, 2, 3, 'banana', 'apple', 'orange'}
Set after removing element: {1, 2, 3, 'banana', 'orange'}
Name: John
Updated dictionary: {'name': 'John', 'age': 35, 'city': 'New York'}
Dictionary after adding key-value pair: {'name': 'John', 'age': 35, 'city': 'New Yor
k', 'country': 'USA'}
Dictionary after removing key-value pair: {'name': 'John', 'age': 35, 'country': 'US
A'}
```

In [ ]:

```
# 2b) Write a Python program to apply various functions on the basic data structure

# Lists
my_list = [1, 2, 3, 'apple', 'banana']

# Accessing elements
print("First element:", my_list[0])
print("Last element:", my_list[-1])

# Updating elements
my_list[2] = 'orange'
print("Updated list:", my_list)

# Adding elements
my_list.append('grape')
print("List after adding element:", my_list)

# Removing elements
my_list.remove('apple')
print("List after removing element:", my_list)


# Tuples
my_tuple = (1, 2, 3, 'apple', 'banana')

# Accessing elements
print("First element:", my_tuple[0])
print("Last element:", my_tuple[-1])

# Tuples are immutable, so elements cannot be updated


# Sets
my_set = {1, 2, 3, 'apple', 'banana'}

# Adding elements
my_set.add('orange')
print("Set after adding element:", my_set)

# Removing elements
```

```python
my_set.remove('apple')
print("Set after removing element:", my_set)


# Dictionaries
my_dict = {'name': 'John', 'age': 30, 'city': 'New York'}

# Accessing values
print("Name:", my_dict['name'])

# Updating values
my_dict['age'] = 35
print("Updated dictionary:", my_dict)

# Adding new key-value pairs
my_dict['country'] = 'USA'
print("Dictionary after adding key-value pair:", my_dict)

# Removing key-value pairs
del my_dict['city']
print("Dictionary after removing key-value pair:", my_dict)
```

```
First element: 1
Last element: banana
Updated list: [1, 2, 'orange', 'apple', 'banana']
List after adding element: [1, 2, 'orange', 'apple', 'banana', 'grape']
List after removing element: [1, 2, 'orange', 'banana', 'grape']
First element: 1
Last element: banana
Set after adding element: {1, 2, 3, 'banana', 'apple', 'orange'}
Set after removing element: {1, 2, 3, 'banana', 'orange'}
Name: John
Updated dictionary: {'name': 'John', 'age': 35, 'city': 'New York'}
Dictionary after adding key-value pair: {'name': 'John', 'age': 35, 'city': 'New Yor
k', 'country': 'USA'}
Dictionary after removing key-value pair: {'name': 'John', 'age': 35, 'country': 'US
A'}
```

In [ ]:

```python
# 2.c) Write a Python function to compute the mean, median, and mode of a list of n

import statistics

def compute_stats(data):
  """Computes the mean, median, and mode of a list of numbers.

  Args:
    data: A list of numbers.

  Returns:
    A tuple containing the mean, median, and mode.
  """
  mean = statistics.mean(data)
  median = statistics.median(data)
  try:
    mode = statistics.mode(data)
  except statistics.StatisticsError:
```

```
    mode = "No unique mode found"
  return mean, median, mode

# Example usage
numbers = [1, 2, 2, 3, 4, 4, 4, 5, 5]
mean, median, mode = compute_stats(numbers)

print("Mean:", mean)
print("Median:", median)
print("Mode:", mode)
```

```
Mean: 3.3333333333333335
Median: 4
Mode: 4
```

In [ ]:

```
#2.D Write a Python function to compute variance and standard deviation of a list o

import statistics

def compute_variance_std(data):
  """Computes the variance and standard deviation of a list of numbers.

  Args:
    data: A list of numbers.

  Returns:
    A tuple containing the variance and standard deviation.
  """
  variance = statistics.variance(data)
  std_dev = statistics.stdev(data)
  return variance, std_dev

# Example usage
numbers = [1, 2, 2, 3, 4, 4, 4, 5, 5]
variance, std_dev = compute_variance_std(numbers)

print("Variance:", variance)
print("Standard Deviation:", std_dev)
```

```
Variance: 2
Standard Deviation: 1.4142135623730951
```

In [ ]:

```
# 2.e) Consider the ungrouped dataset of your choice and use the statistical concep
# i) Determine the range of the raw data.
# ii) Determine the number of classes for frequency distribution table.
# m) Determine the width of each class intervals.
# iv) Determine the midpoint of each class intervals.

import pandas as pd

# Ungrouped dataset
data = [75, 82, 68, 90, 72, 85, 78, 92, 88, 70, 76, 81, 83, 79, 86, 95, 73, 80, 89,

# i) Determine the range of the raw data
data_range = max(data) - min(data)
print("Range:", data_range)
```

```python
# ii) Determine the number of classes (Sturges' rule)
num_classes = int(1 + 3.322 * len(data))
print("Number of classes:", num_classes)

# iii) Determine the width of each class interval
class_width = int(data_range / num_classes) + 1
print("Class width:", class_width)

# Create class intervals
lower_bounds = range(min(data), max(data) + class_width, class_width)
class_intervals = [(lower, lower + class_width - 1) for lower in lower_bounds]

# iv) Determine the midpoint of each class interval
midpoints = [(lower + upper) / 2 for lower, upper in class_intervals]

# Generate frequency distribution table
frequency_dist = {}
for lower, upper in class_intervals:
  frequency_dist[(lower, upper)] = 0
  for value in data:
    if lower <= value <= upper:
      frequency_dist[(lower, upper)] += 1

# Create a Pandas DataFrame for better visualization
frequency_table = pd.DataFrame({
    'Class Interval': class_intervals,
    'Midpoint': midpoints,
    'Frequency': frequency_dist.values()
})

print("\nFrequency Distribution Table:")
print(frequency_table)
```

```
Range: 27
Number of classes: 67
Class width: 1

Frequency Distribution Table:
   Class Interval  Midpoint  Frequency
0        (68, 68)      68.0          1
1        (69, 69)      69.0          0
2        (70, 70)      70.0          1
3        (71, 71)      71.0          0
4        (72, 72)      72.0          1
5        (73, 73)      73.0          1
6        (74, 74)      74.0          0
7        (75, 75)      75.0          1
8        (76, 76)      76.0          1
9        (77, 77)      77.0          1
10       (78, 78)      78.0          1
11       (79, 79)      79.0          1
12       (80, 80)      80.0          1
13       (81, 81)      81.0          1
14       (82, 82)      82.0          1
15       (83, 83)      83.0          1
16       (84, 84)      84.0          0
17       (85, 85)      85.0          1
18       (86, 86)      86.0          1
19       (87, 87)      87.0          0
20       (88, 88)      88.0          1
21       (89, 89)      89.0          1
22       (90, 90)      90.0          1
23       (91, 91)      91.0          0
24       (92, 92)      92.0          1
25       (93, 93)      93.0          0
26       (94, 94)      94.0          0
27       (95, 95)      95.0          1
```

In [ ]:

```python
# 2.e) Consider the ungrouped dataset of your choice and use the statistical concep
# v) Determine the relative frequency of the classes of a frequency distribution.
# vi)Determine the cumulative frequency of the classes of a frequency distribution.
# vii) Populate the complete frequency distribution table for the given

import pandas as pd

# Ungrouped dataset (using the same data from the previous task)
data = [75, 82, 68, 90, 72, 85, 78, 92, 88, 70, 76, 81, 83, 79, 86, 95, 73, 80, 89,

# ... (Previous calculations for range, number of classes, class width, class inter

# Generate frequency distribution table (including relative and cumulative frequenc
frequency_dist = {}
for lower, upper in class_intervals:
  frequency_dist[(lower, upper)] = {'frequency': 0, 'relative_frequency': 0, 'cumul
  for value in data:
    if lower <= value <= upper:
      frequency_dist[(lower, upper)]['frequency'] += 1

total_frequency = len(data)
```

```python
cumulative_frequency = 0

for interval, values in frequency_dist.items():
  values['relative_frequency'] = values['frequency'] / total_frequency
  cumulative_frequency += values['frequency']
  values['cumulative_frequency'] = cumulative_frequency

# Create a Pandas DataFrame for better visualization
frequency_table = pd.DataFrame({
    'Class Interval': class_intervals,
    'Midpoint': midpoints,
    'Frequency': [values['frequency'] for values in frequency_dist.values()],
    'Relative Frequency': [values['relative_frequency'] for values in frequency_dis
    'Cumulative Frequency': [values['cumulative_frequency'] for values in frequency
})

print("\nComplete Frequency Distribution Table:")
print(frequency_table)
```

Complete Frequency Distribution Table:

	Class Interval	Midpoint	Frequency	Relative Frequency \
0	(68, 68)	68.0	1	0.05
1	(69, 69)	69.0	0	0.00
2	(70, 70)	70.0	1	0.05
3	(71, 71)	71.0	0	0.00
4	(72, 72)	72.0	1	0.05
5	(73, 73)	73.0	1	0.05
6	(74, 74)	74.0	0	0.00
7	(75, 75)	75.0	1	0.05
8	(76, 76)	76.0	1	0.05
9	(77, 77)	77.0	1	0.05
10	(78, 78)	78.0	1	0.05
11	(79, 79)	79.0	1	0.05
12	(80, 80)	80.0	1	0.05
13	(81, 81)	81.0	1	0.05
14	(82, 82)	82.0	1	0.05
15	(83, 83)	83.0	1	0.05
16	(84, 84)	84.0	0	0.00
17	(85, 85)	85.0	1	0.05
18	(86, 86)	86.0	1	0.05
19	(87, 87)	87.0	0	0.00
20	(88, 88)	88.0	1	0.05
21	(89, 89)	89.0	1	0.05
22	(90, 90)	90.0	1	0.05
23	(91, 91)	91.0	0	0.00
24	(92, 92)	92.0	1	0.05
25	(93, 93)	93.0	0	0.00
26	(94, 94)	94.0	0	0.00
27	(95, 95)	95.0	1	0.05

	Cumulative Frequency
0	1
1	1
2	2
3	2
4	3
5	4
6	4
7	5
8	6
9	7
10	8
11	9
12	10
13	11
14	12
15	13
16	13
17	14
18	15
19	15
20	16
21	17
22	18
23	18

24	19
25	19
26	19
27	20