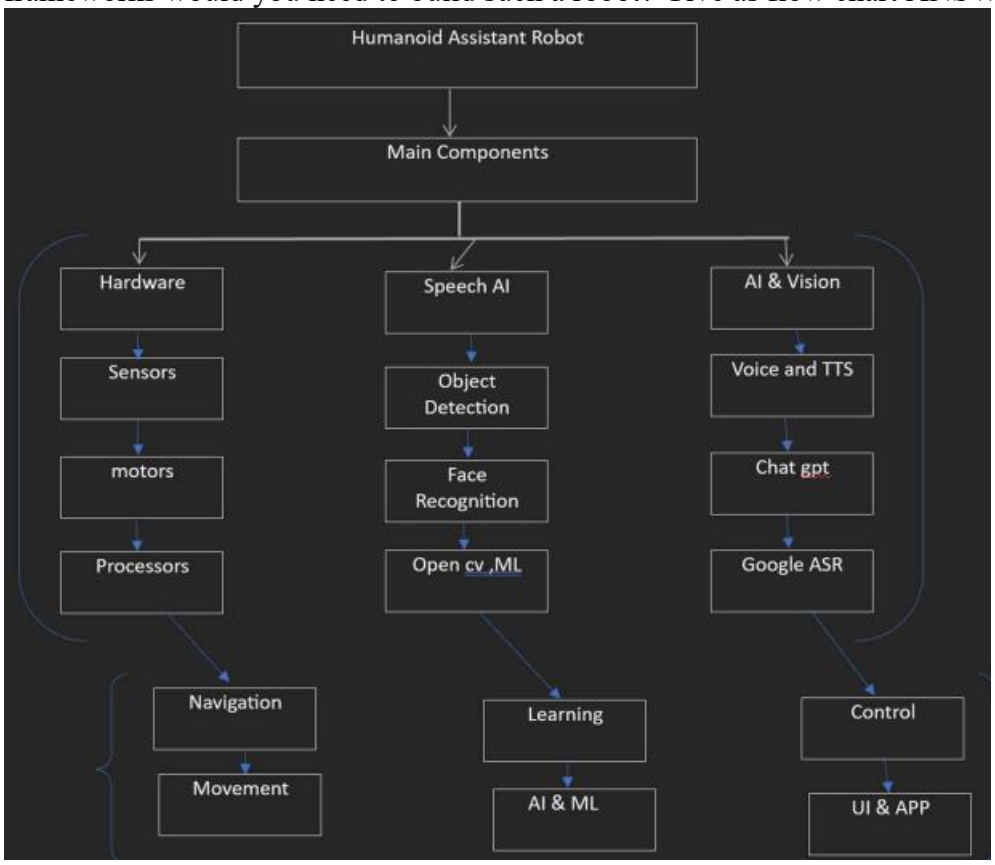
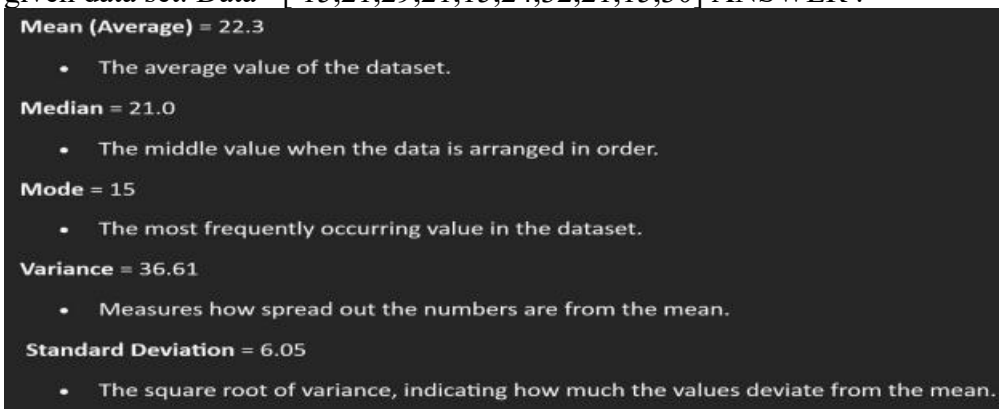
1. Imagine you are tasked with designing a humanoid robot to assist in a home or office environment. The robot must be capable of interacting with people by talking and listening, walking to different locations, seeing and recognizing objects, and learning from its surroundings to adapt its behavior. What technologies, tools, and frameworks would you need to build such a robot? Give as flow chart ANSWER :



2. Calculate and interpret mean, median, mode, variance and standard deviation for a given data set. Data =[15,21,29,21,15,24,32,21,15,30] ANSWER :



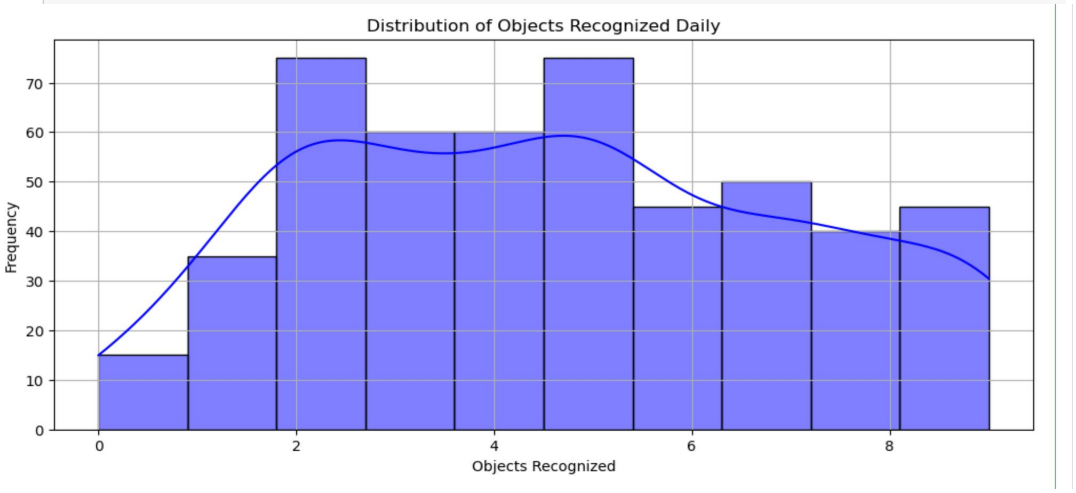
3. You are analyzing a data set that captures the daily performance and activity of a humanoid robot in a simulated environment. The data set link robot_data set(robot_data set)_1.csv includes the following attributes Interaction_Count: Number of conversations the robot had daily. Steps_Walked: Total steps taken each day. Objects_Recognized: Number of objects successfully identified by the robot. Learning_Sessions: Number of learning tasks completed. Energy_Consumption (kWh): Daily energy usage of robots. Perform Basic Statistical Operations:

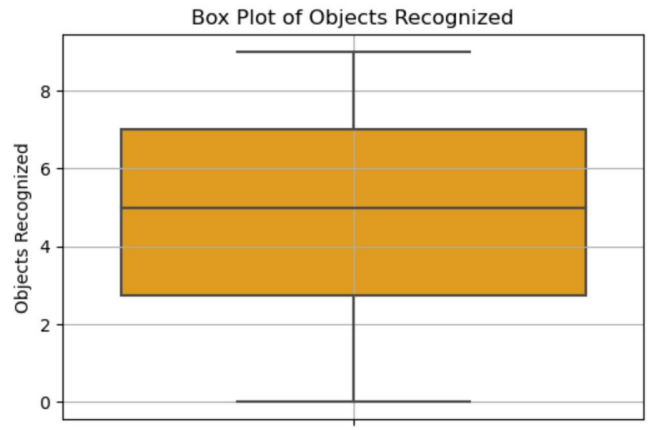
1) What is the average (mean) number of conversations the robot has daily?
2) Find the total steps walked by the robot over a given period.
3) Determine the maximum and minimum energy consumption in the data set.
4) Calculate the correlation between the number of steps walked and energy consumption.
5) Analyze the distribution of objects recognized daily (e.g., histogram or box plot).
6) What is the variance in the number of learning sessions completed?
ANSWER:

```
   Robot_ID    Task_Type Component_ID Sensor_Type             Sensor_Data  \
0  RBT_001   Inspection       CMP_460        LIDAR  1 (obstacle detected)
1  RBT_002     Assembly       CMP_252     Thermal             85.3 (Â°C)
2  RBT_003   Inspection       CMP_248     Thermal          92% (visual fit)
3  RBT_004      Welding       CMP_433      Camera       98% (defect-free)
4  RBT_005     Assembly       CMP_992      Camera          92% (visual fit)

   Processing_Time (s)  Accuracy (%) Environmental_Status  \
0                 67.0          90.4                Stable
1                 71.2          98.1                Stable
2                 49.2          95.3              Unstable
3                 74.5          90.2                Stable
4                 64.5          97.2              Unstable

   Energy_Consumption (kWh) Human_Intervention_Needed Obstacle_Detected  \
0                        2.2                       No                Yes
1                        2.7                      Yes                 No
2                        2.4                       No                 No
3                        2.4                      Yes                 No
4                        1.8                       No                 No

  Defect_Detected  Interaction_Count  Steps_Walked  Objects_Recognized  \
0             Yes                  2            10                   2
1              No                  6            23                   4
2              No                  2            25                   3
3             Yes                  7            34                   5
4              No                  8            35                   3

   Learning_Sessions  Energy_Consumption (kWh).1
0                 10                         100
1                 12                         123
2                 45                         321
3                 12                         456
4                 32                         654
Average Daily Conversations: 5.51
Total Steps Walked: 14379
Maximum Energy Consumption: 3.0 kWh
Minimum Energy Consumption: 1.0 kWh
Correlation between Steps Walked and Energy Consumption: 0.0015478137393313933
```

In [1]:

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
df = pd.read_csv("RB.csv")
print(df.head())
average_conversations = df["Interaction_Count"].mean()
print(f"Average Daily Conversations: {average_conversations}")
total_steps = df["Steps_Walked"].sum()
print(f"Total Steps Walked: {total_steps}")
max_energy = df["Energy_Consumption (kWh)"].max()
min_energy = df["Energy_Consumption (kWh)"].min()
print(f"Maximum Energy Consumption: {max_energy} kWh")
print(f"Minimum Energy Consumption: {min_energy} kWh")
correlation = df["Steps_Walked"].corr(df["Energy_Consumption (kWh)"])
print(f"Correlation between Steps Walked and Energy Consumption: {correlation}")
plt.figure(figsize=(12, 5))
sns.histplot(df["Objects_Recognized"], bins=10, kde=True, color='blue')
plt.title("Distribution of Objects Recognized Daily")
plt.xlabel("Objects Recognized")
plt.ylabel("Frequency")
plt.grid(True)
plt.show()
plt.figure(figsize=(6, 4))
sns.boxplot(y=df["Objects_Recognized"], color="orange")
plt.title("Box Plot of Objects Recognized")
plt.ylabel("Objects Recognized")
plt.grid(True)
plt.show()
learning_sessions_variance = df["Learning_Sessions"].var()
print(f"Variance in Learning Sessions: {learning_sessions_variance}")
```






```
Variance in Learning Sessions: 391.9422845691385
```

4 Write a Python program that declares variables of different data types (e.g., string, integer, float, and boolean). Output the variables in a sentence format using print() and f-strings.
ANSWER:

```
name = "SURYA"
age = 19
height = 6.0
is_student = True

print(f"My name is {name}, I am {age} years old, my height is {height} feet, and it is {is_student} that I am a student.")
```

```
My name is SURYA, I am 19 years old, my height is 6.0 feet, and it is True that I am a student.
```

5.Write a Python program that takes an integer input and checks whether the number is positive, negative, or zero using conditional statements (if-else).
ANSWER:

```
num = int(input("Enter an integer: "))

if num > 0:
    print("The number is positive.")
elif num < 0:
    print("The number is negative.")
else:
    print("The number is zero.")
```

```
Enter an integer: 7
The number is positive.
```

6. Write a Python program that takes a number as input and prints the multiplication table for that number (from 1 to 10).
ANSWER:

```
In [7]: num = int(input("Enter a number: "))
        for i in range(1, 11):
            print(f"{num} x {i} = {num * i}")

        Enter a number: 7
        7 x 1 = 7
        7 x 2 = 14
        7 x 3 = 21
        7 x 4 = 28
        7 x 5 = 35
        7 x 6 = 42
        7 x 7 = 49
        7 x 8 = 56
        7 x 9 = 63
        7 x 10 = 70
```

7. Create a Python list that contains the names of 5 different fruits. Perform the given operations on the list.
ANSWER:

```
fruits = ["Apple", "Banana", "Cherry", "Mango", "Orange"]

fruits.append("Grapes")
fruits.remove("Banana")
fruits[2] = "Pineapple"

print("Updated fruit list:", fruits)
print("First fruit:", fruits[0])
print("Last fruit:", fruits[-1])
```

```
Updated fruit list: ['Apple', 'Cherry', 'Pineapple', 'Orange', 'Grapes']
First fruit: Apple
Last fruit: Grapes
```

8. Write a Python program that creates a tuple containing 5 numbers. Perform the given operations on the tuple.
ANSWER:

```
numbers = (10, 20, 30, 40, 50)

print("Tuple:", numbers)
print("First element:", numbers[0])
print("Last element:", numbers[-1])
print("Sum of all elements:", sum(numbers))
print("Index of 30:", numbers.index(30))
```

```
Tuple: (10, 20, 30, 40, 50)
First element: 10
Last element: 50
Sum of all elements: 150
Index of 30: 2
```

9. Create a dictionary that stores the names of 3 students as keys and their marks in mathematics as values. Perform the given operations.
ANSWER:

```
students = {"SURYA": 85, "SHYAM": 92, "ALFY": 78}

students["SURYA"] = 88
students["SHYAM"] = 90
del students["ALFY"]

print("Updated Dictionary:", students)
print("Marks of SHYAM:", students["SHYAM"])
print("All student names:", list(students.keys()))
print("All marks:", list(students.values()))
```

```
Updated Dictionary: {'SURYA': 88, 'SHYAM': 90}
Marks of SHYAM: 90
All student names: ['SURYA', 'SHYAM']
All marks: [88, 90]
```

10. Create two sets of integers. Perform the given set operations. ANSWER:

```
set1 = {1, 2, 3, 4, 5}
set2 = {4, 5, 6, 7, 8}

union_set = set1 | set2
intersection_set = set1 & set2
difference_set = set1 - set2
symmetric_difference_set = set1 ^ set2

print("Union:", union_set)
print("Intersection:", intersection_set)
print("Difference (set1 - set2):", difference_set)
print("Symmetric Difference:", symmetric_difference_set)
```

```
Union: {1, 2, 3, 4, 5, 6, 7, 8}
Intersection: {4, 5}
Difference (set1 - set2): {1, 2, 3}
Symmetric Difference: {1, 2, 3, 6, 7, 8}
```

11. Write a Python function called find_largest() that takes a list of numbers as input and returns the largest number from the list. Test the function with a sample list. ANSWER:

```
def find_largest(numbers):
    return max(numbers)

sample_list = [10, 25, 67, 89, 45]
largest_number = find_largest(sample_list)

print("Largest number:", largest_number)
```

```
Largest number: 89
```

12. Use list comprehension to create a list of squares of all even numbers between 1 and 20.

ANSWER:

```
squares = [x**2 for x in range(2, 21, 2)]
print("Squares of even numbers:", squares)
```

```
Squares of even numbers: [4, 16, 36, 64, 100, 144, 196, 256, 324, 400]
```

13. Write a Python script that uses a lambda function to calculate the product of two numbers provided by the user.

ANSWER:

```
product = lambda a, b: a * b
num1 = int(input("Enter first number: "))
num2 = int(input("Enter second number: "))
print("Product:", product(num1, num2))
```

```
Enter first number: 7
Enter second number: 2
Product: 14
```

14. Write a Python program to create a one-dimensional, two-dimensional, and three-dimensional NumPy array. Print the shape and dimensions of each array.

ANSWER:

```
import numpy as np

one_d = np.array([1, 2, 3, 4, 5])
two_d = np.array([[1, 2, 3], [4, 5, 6]])
three_d = np.array([[[1, 2], [3, 4]], [[5, 6], [7, 8]]])

print("1D Array:\n", one_d)
print("Shape:", one_d.shape, "Dimensions:", one_d.ndim)

print("\n2D Array:\n", two_d)
print("Shape:", two_d.shape, "Dimensions:", two_d.ndim)

print("\n3D Array:\n", three_d)
print("Shape:", three_d.shape, "Dimensions:", three_d.ndim)
```

```
1D Array:
 [1 2 3 4 5]
Shape: (5,) Dimensions: 1

2D Array:
 [[1 2 3]
 [4 5 6]]
Shape: (2, 3) Dimensions: 2

3D Array:
 [[[1 2]
  [3 4]]

 [[5 6]
  [7 8]]]
Shape: (2, 2, 2) Dimensions: 3
```

15. Write a Python program to create a 5x5 NumPy array of random integers and Perform array indexing as given.
ANSWER:

```
import numpy as np

array = np.random.randint(1, 100, (5, 5))

print("5x5 Random Integer Array:\n", array)

print("\nElement at row 2, column 3:", array[1, 2])
print("First row:", array[0])
print("Last column:", array[:, -1])
print("Subarray (rows 2-4, columns 1-3):\n", array[1:4, 0:3])
```

```
5x5 Random Integer Array:
 [[19 86 21 20  2]
 [40 86 52 54 69]
 [59 92 13 80 34]
 [43 42 75  5  4]
 [40 62 12  6 66]]

 52ment at row 2, column 3:
First row: [19 86 21 20  2]
Last column: [ 2 69 34  4 66]
Subarray (rows 2-4, columns 1-3):
 [[40 86 52]
 [59 92 13]
 [43 42 75]]
```

16. create a NumPy array of shape (4, 4) containing numbers from 1 to 16. Use slicing to extract for the given conditions
ANSWER:

```python
import numpy as np

array = np.arange(1, 17).reshape(4, 4)

print("Original 4x4 Array:\n", array)

print("\nFirst two rows:\n", array[:2])
print("\nLast two columns:\n", array[:, -2:])
print("\nSubarray (middle 2x2 block):\n", array[1:3, 1:3])
print("\nElements from second and fourth row, first and third column:\n", array[[1, 3], [0, 2]])
```

```
Original 4x4 Array:
 [[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]
 [13 14 15 16]]

First two rows:
 [[1 2 3 4]
 [5 6 7 8]]

Last two columns:
 [[ 3  4]
 [ 7  8]
 [11 12]
 [15 16]]

Subarray (middle 2x2 block):
 [[ 6  7]
 [10 11]]

Elements from second and fourth row, first and third column:
 [ 5 15]
```

17. Write a Python program that creates a 2D array of shape (6, 2) using np.arange() and then reshapes it into a 3D array of shape (2, 3, 2). Flatten the reshaped array and print the result.
ANSWER:

```
import numpy as np

array_2d = np.arange(1, 13).reshape(6, 2)
array_3d = array_2d.reshape(2, 3, 2)
flattened_array = array_3d.flatten()

print("Original 2D Array (6x2):\n", array_2d)
print("\nReshaped 3D Array (2x3x2):\n", array_3d)
print("\nFlattened Array:\n", flattened_array)
```

```
Original 2D Array (6x2):
 [[ 1  2]
 [ 3  4]
 [ 5  6]
 [ 7  8]
 [ 9 10]
 [11 12]]

Reshaped 3D Array (2x3x2):
 [[[ 1  2]
  [ 3  4]
  [ 5  6]]

 [[ 7  8]
  [ 9 10]
  [11 12]]]

Flattened Array:
 [ 1  2  3  4  5  6  7  8  9 10 11 12]
```

18. Write a Python program to demonstrate broadcasting. Create an array of shape (3, 3) and add a one dimensional array of shape (1, 3) to it using broadcasting.
ANSWER:

```python
import numpy as np

array_2d = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
array_1d = np.array([10, 20, 30])

result = array_2d + array_1d

print("Original 2D Array:\n", array_2d)
print("\n1D Array:\n", array_1d)
print("\nResult After Broadcasting:\n", result)
```

```
Original 2D Array:
 [[1 2 3]
 [4 5 6]
 [7 8 9]]

1D Array:
 [10 20 30]

Result After Broadcasting:
 [[11 22 33]
 [14 25 36]
 [17 28 39]]
```

19. Create two NumPy arrays of the same shape, A and B. Perform the following arithmetic operations: Element-wise addition. Element-wise subtraction. Element-wise multiplication. Element-wise division. ANSWER:

```python
import numpy as np

A = np.array([[2, 4, 6], [8, 10, 12], [14, 16, 18]])
B = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])

addition = A + B
subtraction = A - B
multiplication = A * B
division = A / B

print("Element-wise Addition:\n", addition)
print("\nElement-wise Subtraction:\n", subtraction)
print("\nElement-wise Multiplication:\n", multiplication)
print("\nElement-wise Division:\n", division)
```

```
Element-wise Addition:
 [[ 3  6  9]
 [12 15 18]
 [21 24 27]]

Element-wise Subtraction:
 [[1 2 3]
 [4 5 6]
 [7 8 9]]

Element-wise Multiplication:
 [[  2   8  18]
 [ 32  50  72]
 [ 98 128 162]]

Element-wise Division:
 [[2. 2. 2.]
 [2. 2. 2.]
```

20. Create a Pandas DataFrame with the given Name and marks of 3 courses: Add a new column named 'Total' that represents the sum of all the courses. Add 'Grade' based on the values of the 'Total'. Print the updated DataFrame with the new 'Total' and 'Grade' column. ANSWER:

```python
import pandas as pd

data = {
    "Name": ["Alice", "Bob", "Charlie"],
    "Math": [85, 78, 92],
    "Science": [88, 74, 95],
    "English": [90, 80, 85]
}

df = pd.DataFrame(data)

df["Total"] = df["Math"] + df["Science"] + df["English"]

def assign_grade(total):
    if total >= 270:
        return "A"
    elif total >= 240:
        return "B"
    elif total >= 210:
        return "C"
    else:
        return "D"

df["Grade"] = df["Total"].apply(assign_grade)

print(df)
```

```
      Name  Math  Science  English  Total Grade
0    Alice    85       88       90    263     B
1      Bob    78       74       80    232     C
2  Charlie    92       95       85    272     A
```