In [2]:
```
#Pandas and Numpy functions
```

In [1]:
```
import pandas as pd
heart=pd.read_csv("report.csv")
```

In [13]:
```
df=pd.DataFrame(heart)
df.describe()
```

Out[13]:

	Age	Cholesterol	Diabities
count	400.000000	400.00000	400.000000
mean	41.045000	217.40900	256.640000
std	23.061604	17.47693	86.922374
min	1.000000	180.40000	124.000000
25%	21.000000	203.92500	175.750000
50%	41.500000	221.70000	261.000000
75%	61.000000	232.10000	341.250000
max	80.000000	250.00000	401.000000

In [16]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 6 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   Name         400 non-null    object 
 1   Age          400 non-null    int64  
 2   BP           400 non-null    object 
 3   Cholesterol  400 non-null    float64
 4   Diabities    400 non-null    int64  
 5   Result       400 non-null    object 
dtypes: float64(1), int64(2), object(3)
memory usage: 18.9+ KB
```

In [17]:
```
1 df.head(10)
```

Out[17]:

	Name	Age	BP	Cholesterol	Diabities	Result
0	John	67	120/80	228.8	143	Yes
1	Sarah	12	130/85	237.6	234	No
2	Rajesh	31	115/70	219.3	176	Yes
3	Sarah	58	140/90	248.5	321	No
4	Rajesh	42	110/75	230.2	267	Yes
5	Maria	19	125/82	200.7	189	No
6	Amit	75	135/88	205.9	355	Yes
7	Elena	5	118/76	189.4	289	No
8	Suresh	64	128/84	231.5	156	Yes
9	Sophia	27	122/78	194.6	398	No

In [18]:
```
1 df.tail(10)
```

Out[18]:

	Name	Age	BP	Cholesterol	Diabities	Result
390	Daniel	26	130/85	214.5	352	Yes
391	Kavita	74	125/81	225.8	232	No
392	Ella	1	145/94	241.3	169	No
393	Naveen	56	121/80	205.0	149	Yes
394	Ava	32	140/92	223.7	363	Yes
395	Ritu	51	112/72	196.2	271	Yes
396	James	43	132/86	232.1	199	No
397	Lila	20	116/75	190.6	352	No
398	Amitabh	61	138/88	221.7	232	Yes
399	Lucas	39	113/71	238.0	169	Yes

In [20]:
```
1 print(df.index)
```

```
RangeIndex(start=0, stop=400, step=1)
```

In [25]:
```
1 df.columns
```

Out[25]:
```
Index(['Name', 'Age', 'BP', 'Cholesterol', 'Diabities', 'Result'], dtype='obj
ect')
```

In [26]:
```
1 df.shape
```

Out[26]:
```
(400, 6)
```

In [28]:
```
1 df.dtypes
```

Out[28]:
```
Name            object
Age              int64
BP              object
Cholesterol    float64
Diabities        int64
Result          object
dtype: object
```

In [30]:
```
1 df.head(3)
```

Out[30]:

	Name	Age	BP	Cholesterol	Diabities	Result
0	John	67	120/80	228.8	143	Yes
1	Sarah	12	130/85	237.6	234	No
2	Rajesh	31	115/70	219.3	176	Yes

In [33]:
```
1 df['Age'].mean
```

Out[33]:
```
<bound method Series.mean of 0      67
1      12
2      31
3      58
4      42
       ..
395    51
396    43
397    20
398    61
399    39
Name: Age, Length: 400, dtype: int64>
```

In [38]:
```
1 df['Cholesterol'].median
```

Out[38]:
```
<bound method Series.median of 0      228.8
1      237.6
2      219.3
3      248.5
4      230.2
       ...
395    196.2
396    232.1
397    190.6
398    221.7
399    238.0
Name: Cholesterol, Length: 400, dtype: float64>
```

In [39]:
```
1 df['Diabities'].mode
```

Out[39]:
```
<bound method Series.mode of 0      143
1      234
2      176
3      321
4      267
      ...
395    271
396    199
397    352
398    232
399    169
Name: Diabities, Length: 400, dtype: int64>
```

In [36]:
```
1 df.max()
```

Out[36]:
```
Name              Zoe
Age                80
BP             147/96
Cholesterol     250.0
Diabities         401
Result            Yes
dtype: object
```

In [37]:
```
1 df.min()
```

Out[37]:
```
Name           Abigail
Age                  1
BP               105/65
Cholesterol       180.4
Diabities           124
Result               No
dtype: object
```

In [40]:
```
1 #Numpy functions
```

In [41]:
```
1 import numpy as np
2 empty_array = np.empty((3, 3))
3 filled_array = np.ones((2, 2))
4 print("Empty Array:")
5 print(empty_array)
6 print("\nFilled Array:")
7 print(filled_array)
```

```
Empty Array:
[[0.0e+000 4.9e-324 9.9e-324]
 [1.5e-323 2.0e-323 2.5e-323]
 [3.0e-323 3.5e-323 4.0e-323]]

Filled Array:
[[1. 1.]
 [1. 1.]]
```

In [42]:
```
def remove_non_numeric_rows(arr):
    mask = np.all(np.isreal(arr), axis=1)
    return arr[mask]
original_array = np.array([[1, 2, 3], [4, 5, 6], ['a', 'b', 'c'], [7, 8, 9

cleaned_array = remove_non_numeric_rows(original_array)

print("Original Array")
print(original_array)

print("\nCleaned Array")
print(cleaned_array)
```

```
Original Array
[['1' '2' '3']
 ['4' '5' '6']
 ['a' 'b' 'c']
 ['7' '8' '9']]

Cleaned Array
[]
```

In [43]:
```
def count_occurrences(sequence, array):
    occurrences = np.where(np.correlate(array, sequence, 'valid') == np.su
    return len(occurrences[0])
sequence = np.array([1, 2, 3])
array = np.array([1, 2, 3, 1, 2, 3, 1, 2, 3])

result = count_occurrences(sequence, array)
print(f"The sequence {sequence} occurs {result} times in the array")
```

```
The sequence [1 2 3] occurs 0 times in the array
```

In [44]:
```
array1 = np.array([1, 2, 3])
array2 = np.array([4, 5, 6])
sum_array = array1 + array2
print("Resultant Array:", sum_array)
```

```
Resultant Array: [5 7 9]
```

In [45]:
```
sample_array = np.array([1, 2, 3, 4, 5, 6])
condition = sample_array > 3
indices = np.where(condition)
print("Indices where condition is satisfied:", indices)
```

```
Indices where condition is satisfied: (array([3, 4, 5], dtype=int64),)
```

In [46]:
```
def multiply_matrix_by_scalar(matrix, scalar):
    result = matrix * scalar
    return result
matrix = np.array([[1, 2], [3, 4]])
scalar = 2
result_matrix = multiply_matrix_by_scalar(matrix, scalar)
print(result_matrix)
```

```
[[2 4]
 [6 8]]
```

In [47]:
```
def redimension_array(data, new_shape):
    result = data.reshape(new_shape)
    return result
a = np.array([[2.5, 3.8, 1.5], [4.7, 2.9, 1.56]])
new_shape = (3, 2)
result_array = redimension_array(a, new_shape)
print(result_array)
```

```
[[2.5  3.8 ]
 [1.5  4.7 ]
 [2.9  1.56]]
```

In [48]:
```
def get_boolean_array(binary_array):
    boolean_array = binary_array.astype(bool)
    return boolean_array
a = np.array([[1, 0, 0],
              [1, 1, 1],
              [0, 0, 0]])
result_boolean_array = get_boolean_array(a)
print(result_boolean_array)
```

```
[[ True False False]
 [ True  True  True]
 [False False False]]
```

In [49]:
```
def horizontal_stack_arrays(arrays):
    result = np.hstack(arrays)
    return result
array1 = np.array([1, 2, 3])
array2 = np.array([4, 5, 6])
array3 = np.array([7, 8, 9])
result_array = horizontal_stack_arrays([array1, array2, array3])
print(result_array)
```

```
[1 2 3 4 5 6 7 8 9]
```

In [ ]:
```

```