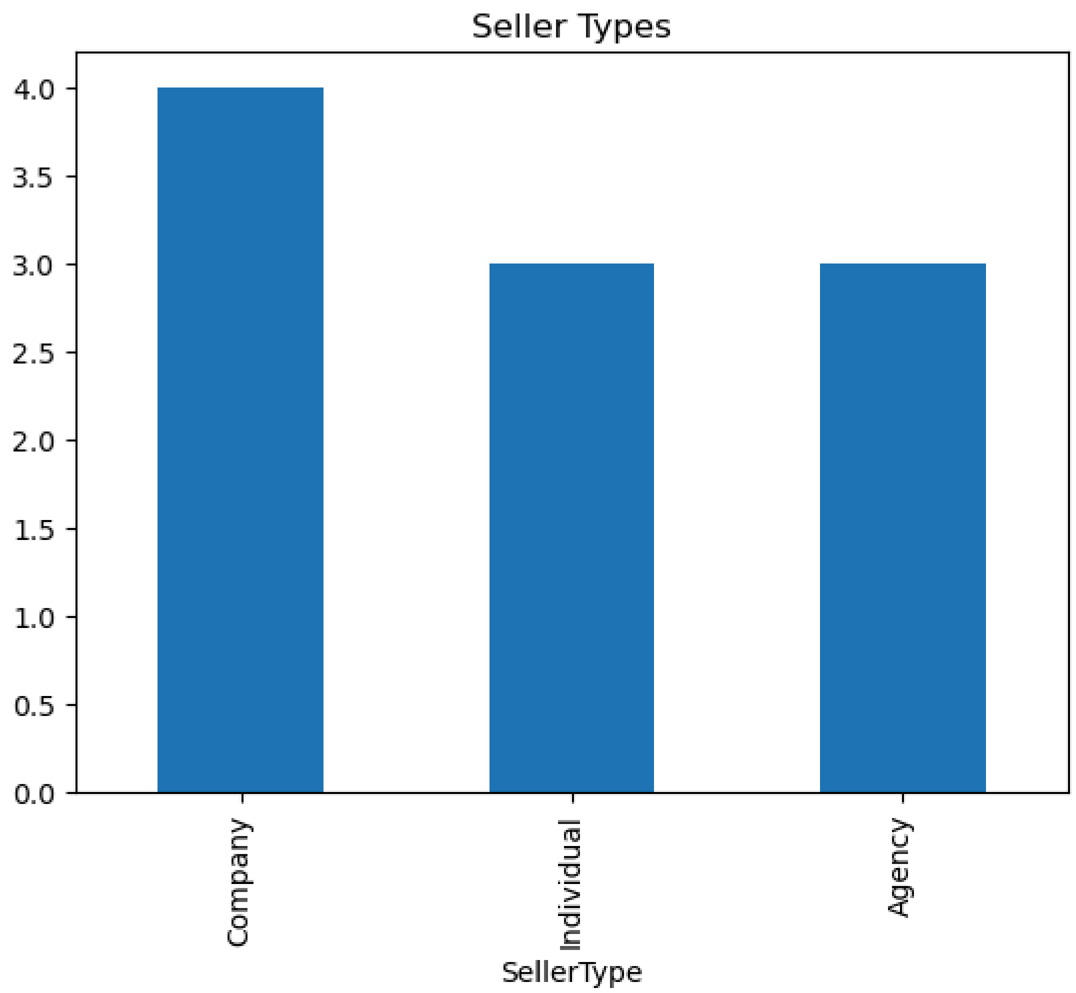
In [ ]:

```
#slip20
# Q1) Read "StudentsPerformance.csv" and display dataset information

import pandas as pd

# Read the CSV file
data = pd.read_csv("StudentsPerformance.csv")

# Display the shape of the dataset (rows, columns)
print("Shape of dataset (rows, columns):", data.shape)

# Display the first 5 rows of the dataset
print("\nTop rows of the dataset:")
print(data.head())

# Display column names
print("\nColumns in dataset:")
print(data.columns.tolist())
```

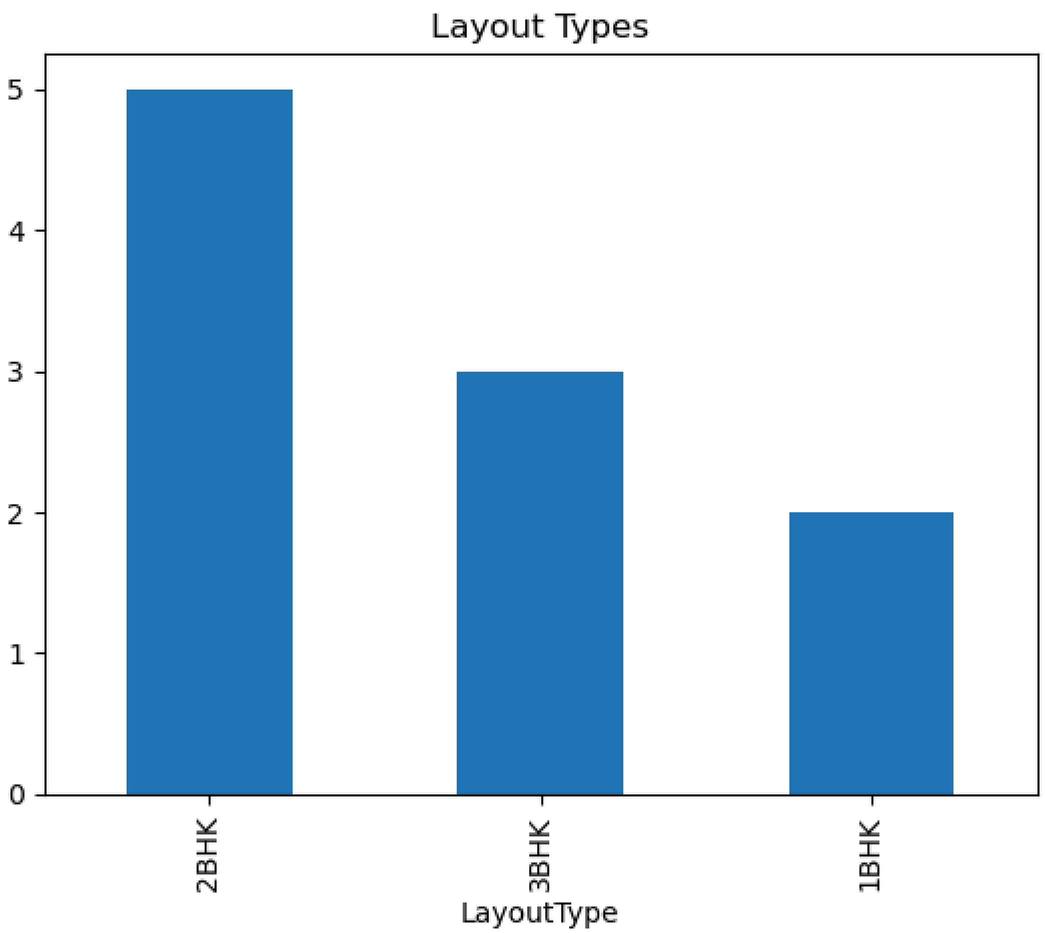
In [3]:

```
#slip19
import pandas as pd
import matplotlib.pyplot as plt

# create sample dataset (if CSV given, use pd.read_csv("Seller.csv"))
data = pd.DataFrame({
    "SellerType": ["Individual","Company","Individual","Company","Agency",
                   "Agency","Individual","Company","Agency","Company"],
    "LayoutType": ["2BHK","3BHK","1BHK","2BHK","3BHK",
                   "2BHK","1BHK","2BHK","3BHK","2BHK"]
})

# a) Count different seller types
print("Different seller types:\n", data["SellerType"].value_counts())
data["SellerType"].value_counts().plot(kind="bar", title="Seller Types")
plt.show()

# b) Seller with minimum records
print("\nSeller with minimum records:\n", data["SellerType"].value_counts().idxm

# c) Count of layout types
print("\nDifferent layout types:\n", data["LayoutType"].value_counts())
data["LayoutType"].value_counts().plot(kind="bar", title="Layout Types")
plt.show()
```

```
Different seller types:
 SellerType
Company       4
Individual    3
Agency        3
Name: count, dtype: int64
```

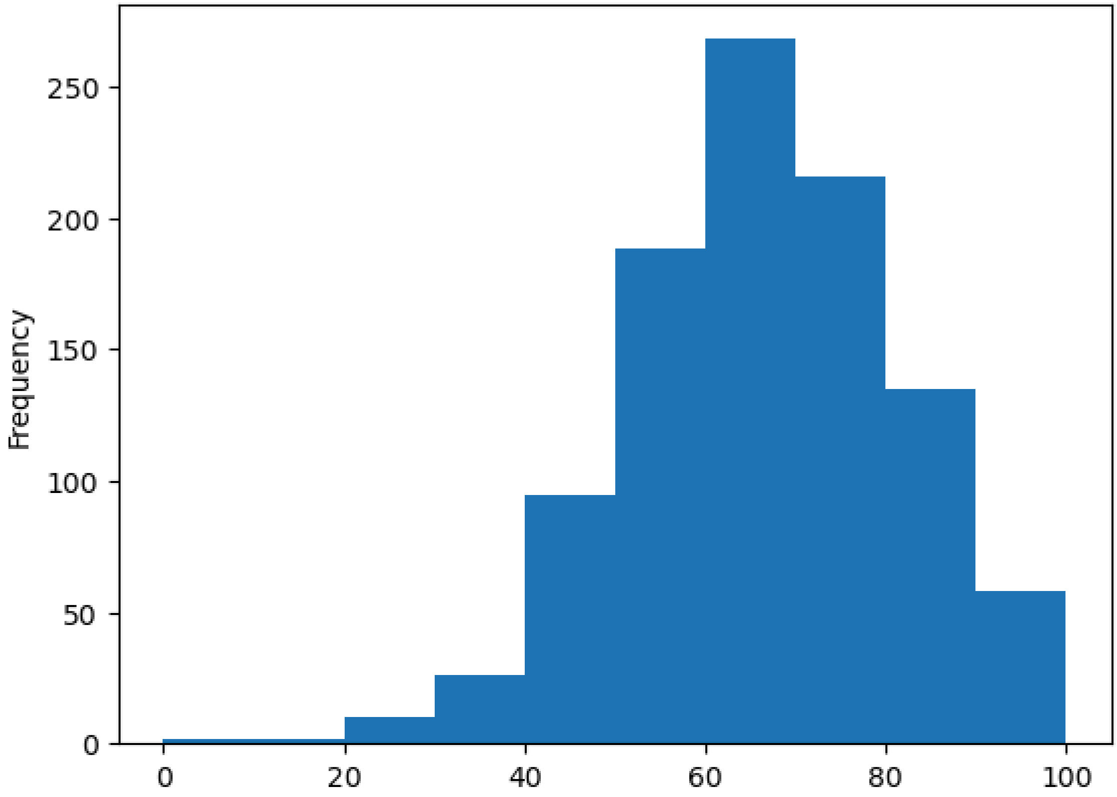


```
Seller with minimum records:
 Individual

Different layout types:
 LayoutType
2BHK    5
3BHK    3
1BHK    2
Name: count, dtype: int64
```

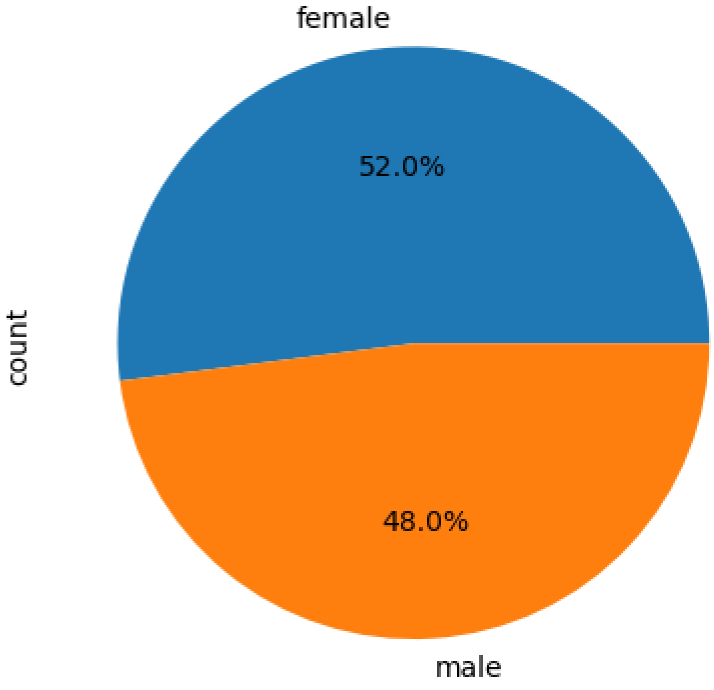


In [4]:

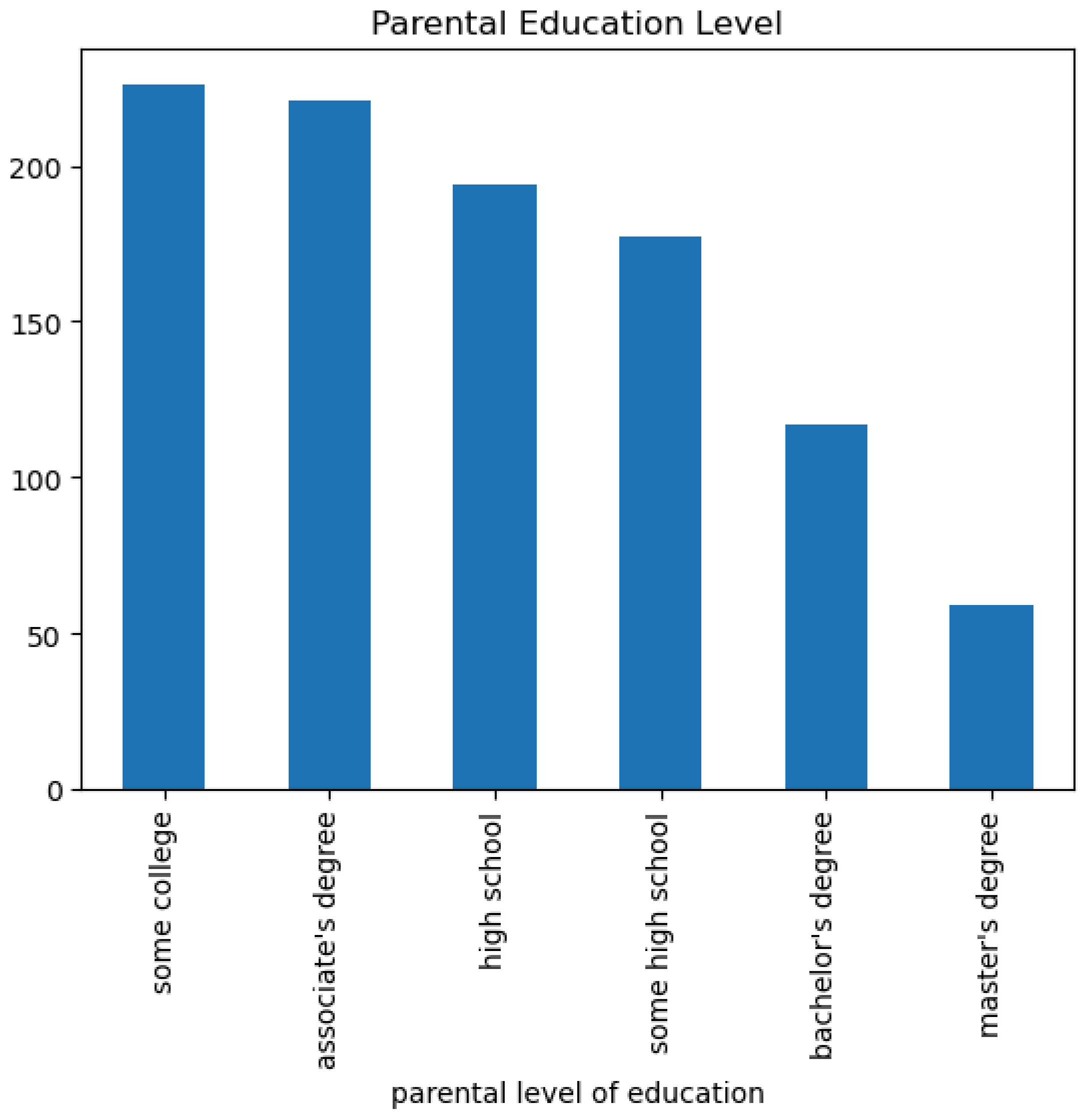
```
#slip 16
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Read CSV (or create dummy data)
data = pd.read_csv("Student.csv")

# 1) Histogram of Math Score
data['math score'].plot(kind='hist', title='Math Score Histogram', bins=10)
plt.show()

# 2) Pie Chart of Gender
data['gender'].value_counts().plot(kind='pie', autopct='%1.1f%%', title='Gender
plt.show()

# 3) Bar Graph of Parental Level of Education
data['parental level of education'].value_counts().plot(kind='bar', title='Paren
plt.show()

# 4) Heatmap of correlation between scores
sns.heatmap(data[['math score','reading score','writing score']].corr(), annot=T
plt.title("Heatmap of Scores")
plt.show()
```

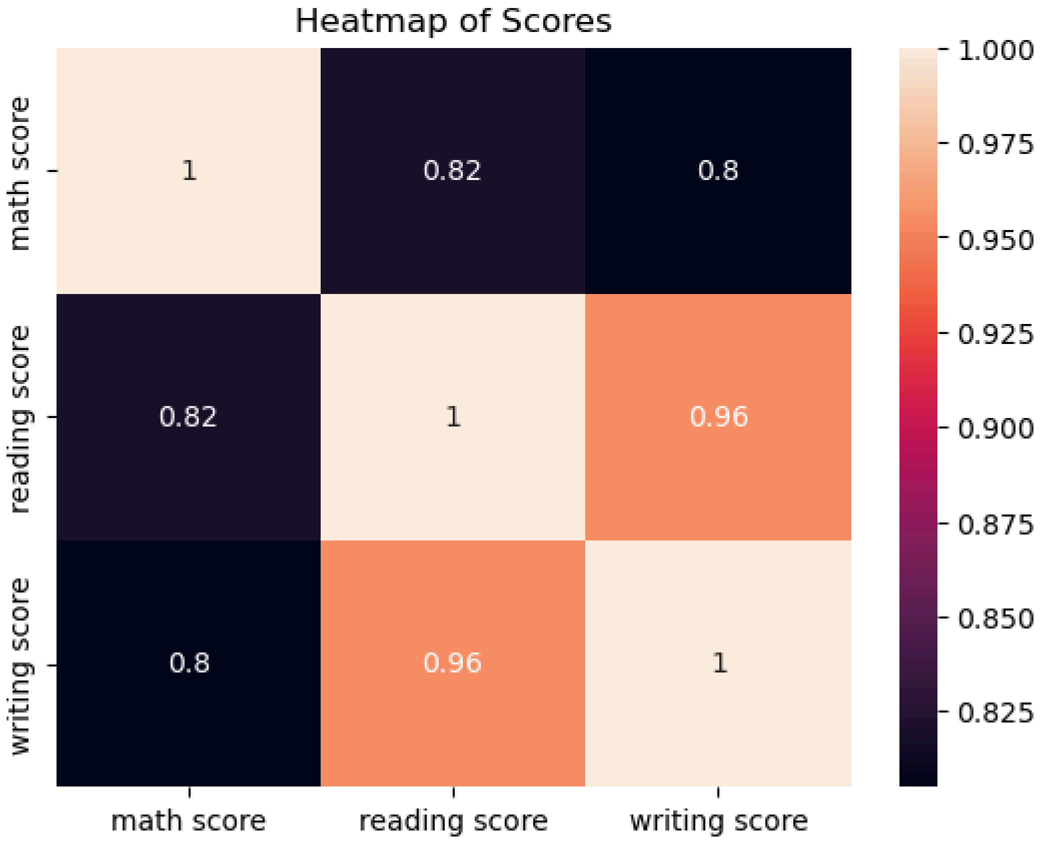


Math Score Histogram



Gender Distribution


Parental Education Level


Heatmap of Scores

In [5]:

```python
#slip17
import numpy as np

# 1) np.array()
arr1D = np.array([1, 2, 3, 4, 5])       # 1D array
arr2D = np.array([[1, 2], [3, 4]])      # 2D array
print("1D array:\n", arr1D)
print("2D array:\n", arr2D)

# 2) np.arange()
arr_range = np.arange(0, 10, 2)         # 0 to 10 step 2
print("\nArange array:\n", arr_range)

# 3) np.zeros()
zeros1D = np.zeros(5)                   # 1D zeros
zeros2D = np.zeros((2,3))               # 2D zeros
print("\n1D zeros:\n", zeros1D)
print("2D zeros:\n", zeros2D)

# 4) np.ones()
ones1D = np.ones(5)                     # 1D ones
ones2D = np.ones((2,3))                 # 2D ones
print("\n1D ones:\n", ones1D)
print("2D ones:\n", ones2D)
```

```
1D array:
 [1 2 3 4 5]
2D array:
 [[1 2]
 [3 4]]

Arange array:
 [0 2 4 6 8]

1D zeros:
 [0. 0. 0. 0. 0.]
2D zeros:
 [[0. 0. 0.]
 [0. 0. 0.]]

1D ones:
 [1. 1. 1. 1. 1.]
2D ones:
 [[1. 1. 1.]
 [1. 1. 1.]]
```

In [6]:

```python
#slip14
import numpy as np

# Create a sample NumPy array
arr = np.array([10, 20, 30, 40, 50])

# 1) Sum
print("Sum:", np.sum(arr))

# 2) Mean
print("Mean:", np.mean(arr))

# 3) Median
print("Median:", np.median(arr))
```

```
# 4) Variance
print("Variance:", np.var(arr))

# 5) Standard Deviation
print("Std Dev:", np.std(arr))

# 6) Minimum
print("Min:", np.min(arr))

# 7) Maximum
print("Max:", np.max(arr))
```

```
Sum: 150
Mean: 30.0
Median: 30.0
Variance: 200.0
Std Dev: 14.142135623730951
Min: 10
Max: 50
```

In [7]:

```
#slip13
import pandas as pd
import numpy as np

# 1D Array → DataFrame
arr1D = np.array([10, 20, 30, 40, 50])
df1D = pd.DataFrame(arr1D, columns=["Numbers"])
print("1D DataFrame:\n", df1D)

# Indexing 1D
print("\n1D Indexing (3rd element):", df1D.iloc[2,0])

# Slicing 1D
print("1D Slicing (2nd to 4th element):\n", df1D.iloc[1:4, 0])

# 2D Array → DataFrame
arr2D = np.array([[1,2,3],[4,5,6],[7,8,9]])
df2D = pd.DataFrame(arr2D, columns=["A","B","C"])
print("\n2D DataFrame:\n", df2D)

# Indexing 2D
print("\n2D Indexing (row 2, column B):", df2D.iloc[1,1])

# Slicing 2D
print("2D Slicing (first 2 rows, first 2 columns):\n", df2D.iloc[0:2, 0:2])
```

```
1D DataFrame:
    Numbers
0       10
1       20
2       30
3       40
4       50

1D Indexing (3rd element): 30
1D Slicing (2nd to 4th element):
 1    20
2    30
3    40
Name: Numbers, dtype: int64

2D DataFrame:
    A  B  C
0  1  2  3
1  4  5  6
2  7  8  9

2D Indexing (row 2, column B): 5
2D Slicing (first 2 rows, first 2 columns):
    A  B
0  1  2
1  4  5
```

In [8]:

```python
#slip11
import pandas as pd

# Create Employee DataFrame with 7 columns
data = {
    "EmpID": [101, 102, 103, 104, 105],
    "Name": ["Amit", "Sneha", "Ravi", "Priya", "Karan"],
    "Age": [25, 28, 24, 27, 30],
    "Department": ["HR", "IT", "Finance", "IT", "Marketing"],
    "Salary": [50000, 60000, 55000, 58000, 62000],
    "Experience": [2, 5, 1, 4, 6],
    "City": ["Mumbai", "Delhi", "Pune", "Bangalore", "Hyderabad"]
}

df = pd.DataFrame(data)

# Display the DataFrame
print(df)
```

```
   EmpID   Name  Age Department  Salary  Experience       City
0    101   Amit   25         HR   50000           2     Mumbai
1    102  Sneha   28         IT   60000           5      Delhi
2    103   Ravi   24    Finance   55000           1       Pune
3    104  Priya   27         IT   58000           4  Bangalore
4    105  Karan   30  Marketing   62000           6  Hyderabad
```

In [9]:

```python
#slip10
import numpy as np

# Create a 2D array
arr = np.array([[1, 2, 3, 4], [5, 6, 7, 8]])
print("Original Array:\n", arr)
```

```
# 1) Reshape (change shape without changing data)
reshaped = arr.reshape(4, 2)    # 4 rows, 2 columns
print("\nReshaped Array (4x2):\n", reshaped)

# 2) Resize (change shape and can change total size)
resized = np.resize(arr, (3, 3))  # 3x3 array
print("\nResized Array (3x3):\n", resized)

# 3) Transpose (swap rows and columns)
transposed = arr.T
print("\nTransposed Array:\n", transposed)
```

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

Reshaped Array (4x2):
 [[1 2]
 [3 4]
 [5 6]
 [7 8]]

Resized Array (3x3):
 [[1 2 3]
 [4 5 6]
 [7 8 1]]

Transposed Array:
 [[1 5]
 [2 6]
 [3 7]
 [4 8]]
```

In [10]:

```
#slip9
import numpy as np

# Create a 2D array
arr = np.array([[1, 2, 3], [4, 5, 6]])
print("2D Array:\n", arr)

# Shape of the array (rows, columns)
print("\nShape:", arr.shape)

# Total number of elements
print("Size:", arr.size)

# Number of dimensions
print("Dimensions:", arr.ndim)
```

```
2D Array:
 [[1 2 3]
 [4 5 6]]

Shape: (2, 3)
Size: 6
Dimensions: 2
```

In [11]:

```
#slip8
import pandas as pd

# Create sample dataset
```

```
data = pd.DataFrame({
    "Student": ["Amit", "Sneha", "Ravi", "Priya", "Karan"],
    "Marks": [32, 28, 35, 40, 30],
    "Attempts": [1, 2, 1, 3, 1]
})

print("Dataset:\n", data)

# a) Rows where marks are between 30 and 35
marks_30_35 = data[(data["Marks"] >= 30) & (data["Marks"] <= 35)]
print("\nMarks between 30 and 35:\n", marks_30_35)

# b) Rows where attempts < 2 and marks > 30
attempts_marks = data[(data["Attempts"] < 2) & (data["Marks"] > 30)]
print("\nAttempts < 2 and Marks > 30:\n", attempts_marks)

# c) Sum of examination attempts
total_attempts = data["Attempts"].sum()
print("\nTotal examination attempts:", total_attempts)
```

```
Dataset:
   Student  Marks  Attempts
0    Amit     32         1
1   Sneha     28         2
2    Ravi     35         1
3   Priya     40         3
4   Karan     30         1

Marks between 30 and 35:
   Student  Marks  Attempts
0    Amit     32         1
2    Ravi     35         1
4   Karan     30         1

Attempts < 2 and Marks > 30:
   Student  Marks  Attempts
0    Amit     32         1
2    Ravi     35         1

Total examination attempts: 8
```

In [12]:

```
#slip7
import pandas as pd

# Create sample DataFrame
df = pd.DataFrame({
    "Math": [50, 60, 70],
    "Science": [55, 65, 75]
})

print("Original DataFrame:\n", df)

# a) apply() → apply function to each column
print("\nUsing apply() to add 5 to each column:")
print(df.apply(lambda x: x + 5))

# b) applymap() → apply function to each element
print("\nUsing applymap() to add 10 to each element:")
print(df.applymap(lambda x: x + 10))
```

```
# c) map() → apply function to a single column
print("\nUsing map() on Math column to double the values:")
print(df["Math"].map(lambda x: x * 2))
```

```
Original DataFrame:
    Math  Science
0    50       55
1    60       65
2    70       75

Using apply() to add 5 to each column:
   Math  Science
0    55       60
1    65       70
2    75       80

Using applymap() to add 10 to each element:
   Math  Science
0    60       65
1    70       75
2    80       85

Using map() on Math column to double the values:
0    100
1    120
2    140
Name: Math, dtype: int64
```

```
C:\Users\User\AppData\Local\Temp\ipykernel_29848\2794539646.py:18: FutureWarning:
DataFrame.applymap has been deprecated. Use DataFrame.map instead.
  print(df.applymap(lambda x: x + 10))
```

In [13]:

```
#slip4
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

# Data
x = np.array([1,2,3,4,5,6,7,8]).reshape(-1,1)  # x must be 2D
y = np.array([7,14,15,18,19,21,26,23])

# Create Linear Regression model
model = LinearRegression()
model.fit(x, y)

# Estimated coefficients
m = model.coef_[0]     # slope
c = model.intercept_  # intercept
print("Estimated slope (m):", m)
print("Estimated intercept (c):", c)

# Predictions
y_pred = model.predict(x)

# Performance metrics
mse = mean_squared_error(y, y_pred)
r2 = r2_score(y, y_pred)
print("\nMean Squared Error (MSE):", mse)
print("R-squared (R2 score):", r2)
```

```
Estimated slope (m): 2.2738095238095237
Estimated intercept (c): 7.642857142857142

Mean Squared Error (MSE): 3.4657738095238084
R-squared (R2 score): 0.8867741072947811
```

In [14]:

```python
#slip3
import pandas as pd

# Create a dictionary
data = {
    "EmpID": [101, 102, 103],
    "Name": ["Amit", "Sneha", "Ravi"],
    "Department": ["HR", "IT", "Finance"],
    "Salary": [50000, 60000, 55000]
}

# Convert dictionary to DataFrame
df = pd.DataFrame(data)

# Display the DataFrame
print("Employee Data:\n", df)
```

```
Employee Data:
    EmpID   Name Department  Salary
0    101   Amit         HR   50000
1    102  Sneha         IT   60000
2    103   Ravi    Finance   55000
```

In [15]:

```python
#slip1
import pandas as pd

# Create sample dictionary
data = {
    "Employee": ["Amit", "Sneha", "Ravi", "Amit", "Ravi"],
    "Department": ["HR", "IT", "Finance", "HR", "Finance"],
    "Salary": [50000, 60000, 55000, 52000, 58000]
}

# Convert dictionary to DataFrame
df = pd.DataFrame(data)

# Create Pivot Table: sum of Salary by Employee and Department
pivot = pd.pivot_table(df, index="Employee", columns="Department", values="Salar

print("Pivot Table:\n", pivot)
```

```
Pivot Table:
 Department  Finance      HR     IT
Employee
Amit              0  102000      0
Ravi         113000       0      0
Sneha             0       0  60000
```

In [ ]: