1. Create a DataFrame using the following data data = { 'Name': ['Alice', 'Bob', 'Charlie', 'David'], 'Age': [24, 27, 22, 32], 'Salary': [50000, 60000, 45000, 80000] } Write a program to: a) Create a new column, Tax, which is 10% of the Salary. b) Create another column, Net Salary, as Salary – Tax.

```
import pandas as pd

# Creating a DataFrame
data = {
    'Name': ['Alice', 'Bob', 'Charlie'],
    'Age': [25, 30, 35],
    'City': ['New York', 'Los Angeles', 'Chicago']
}

df = pd.DataFrame(data)
print(df)
```

```
      Name  Age         City
0    Alice   25     New York
1      Bob   30  Los Angeles
2  Charlie   35      Chicago
```

```
print(df['Name'])
```

```
0      Alice
1        Bob
2    Charlie
Name: Name, dtype: object
```

```
print(df.iloc[0])  # Access by position
print(df.loc[0])   # Access by label (index)
```

```
Name       Alice
Age           25
City    New York
Name: 0, dtype: object
Name       Alice
Age           25
City    New York
Name: 0, dtype: object
```

2. Given the following DataFrame data = { 'Sex': ['male', 'female', 'female', 'male', 'female'], 'Class': ['First', 'Second', 'Third', 'First', 'Second'], 'Fare': [100, 50, 20, 120, 60] } df = pd.DataFrame(data) Write a program to calculate the mean ticket fare price for each combination of Sex and Class.

```
import pandas as pd

# Sample DataFrame
data = {'Name': ['Alice', 'Bob', 'Charlie'],
        'Age': [24, 27, 22],
        'City': ['New York', 'Los Angeles', 'Chicago']}
df = pd.DataFrame(data, index=['A', 'B', 'C'])
print(df,"\n")
# Using loc (label-based)
print(df.loc['A'])  # Access row with index label 'A'
print("\n")
print(df.loc['A', 'Age'])  # Access 'Age' column for row 'A'
print("\n")
print(df.loc['A':'B', 'Name'])  # Slice rows 'A' to 'B' and 'Name' column
```

```
      Name  Age         City
A    Alice   24     New York
B      Bob   27  Los Angeles
C  Charlie   22      Chicago

Name       Alice
Age           24
City    New York
Name: A, dtype: object
```

```
24


A    Alice
B      Bob
Name: Name, dtype: object
```

```
data = {
'Name': ['Alice', 'Bob', 'Charlie', 'David'],
'Age': [24, 27, 22, 32],
'Salary': [50000, 60000, 45000, 80000]
}
df = pd.DataFrame(data);
print(df)
```

```
      Name  Age  Salary
0    Alice   24   50000
1      Bob   27   60000
2  Charlie   22   45000
3    David   32   80000
```

```
df['Tax'] = df['Salary']*0.1
print(df)
```

```
      Name  Age  Salary     Tax
0    Alice   24   50000  5000.0
1      Bob   27   60000  6000.0
2  Charlie   22   45000  4500.0
3    David   32   80000  8000.0
```

```
df['net_salary'] = df['Salary']-df['Tax']
print(df)
```

```
      Name  Age  Salary     Tax  net_salary
0    Alice   24   50000  5000.0     45000.0
1      Bob   27   60000  6000.0     54000.0
2  Charlie   22   45000  4500.0     40500.0
3    David   32   80000  8000.0     72000.0
```

Double-click (or enter) to edit

```
data = {
'Sex': ['male', 'female', 'female', 'male', 'female'],
'Class': ['First', 'Second', 'Third', 'First', 'Second'],
'Fare': [100, 50, 20, 120, 60]
}
df = pd.DataFrame(data)
print(df)
```

```
      Sex   Class  Fare
0    male   First   100
1  female  Second    50
2  female   Third    20
3    male   First   120
4  female  Second    60
```

```
male_fare = df.loc[df['Sex']=='male','Fare']
female_fare = df.loc[df['Sex']=='female','Fare']
male_fare_mean = np.mean(male_fare);
female_fare_mean = np.mean(female_fare)
print(male_fare)
print(female_fare)
print(male_fare_mean)
print(round(female_fare_mean,2))
```

```
0    100
3    120
Name: Fare, dtype: int64
1    50
2    20
4    60
Name: Fare, dtype: int64
110.0
43.33
```

3.Using the same DataFrame as Question 2, write a program to count the number of passengers in each Class.

```
class_count = df['Class'].value_counts()
print(class_count)
```

```
Class
First     2
Second    2
Third     1
Name: count, dtype: int64
```

4. Given the following DataFrame data = { 'Student': ['Alice', 'Bob', 'Charlie'], 'Math': [85, 90, 95], 'Science': [88, 92, 96] } df = pd.DataFrame(data) Write a program to reshape the DataFrame from wide format to long format such that each row corresponds to a Student, a Subject, and their respective Score.

```
import pandas as pd

# Original DataFrame
data = {
    'Student': ['Alice', 'Bob', 'Charlie'],
    'Math': [85, 90, 95],
    'Science': [88, 92, 96]
}
df = pd.DataFrame(data)

# Reshape the DataFrame to long format
df_long = pd.melt(df, id_vars=['Student'], var_name='Subject', value_name='Score')

# Display the reshaped DataFrame
print(df_long)
```

```
   Student  Subject  Score
0    Alice     Math     85
1      Bob     Math     90
2  Charlie     Math     95
3    Alice  Science     88
4      Bob  Science     92
5  Charlie  Science     96
```

5. Using the reshaped DataFrame from Question 4, write a program to convert it back to wide format with Math and Science as separate columns.

```
import pandas as pd

# Reshaped (long format) DataFrame
data_long = {
    'Student': ['Alice', 'Bob', 'Charlie', 'Alice', 'Bob', 'Charlie'],
    'Subject': ['Math', 'Math', 'Math', 'Science', 'Science', 'Science'],
    'Score': [85, 90, 95, 88, 92, 96]
}
df_long = pd.DataFrame(data_long)

# Convert back to wide format
df_wide = df_long.pivot(index='Student', columns='Subject', values='Score').reset_index()

# Display the wide format DataFrame
print(df_wide)
```

```
Subject  Student  Math  Science
0          Alice    85       88
1            Bob    90       92
2        Charlie    95       96
```

6. Given the following two DataFrames data1 = {'ID': [1, 2, 3], 'Name': ['Alice', 'Bob', 'Charlie']} data2 = {'ID': [2, 3, 4], 'Score': [85, 90, 95]} df1 = pd.DataFrame(data1) df2 = pd.DataFrame(data2) Write a program to: a) Perform an inner join on ID. b) Perform a left join on ID.

```
# DataFrames
data1 = {'ID': [1, 2, 3], 'Name': ['Alice', 'Bob', 'Charlie']}
data2 = {'ID': [2, 3, 4], 'Score': [85, 90, 95]}
df1 = pd.DataFrame(data1)
df2 = pd.DataFrame(data2)

# a) Perform an inner join on ID
inner_join_result = pd.merge(df1, df2, on='ID', how='inner')

# b) Perform a left join on ID
left_join_result = pd.merge(df1, df2, on='ID', how='left')

# Display the results
print("Inner Join Result:")
print(inner_join_result)

print("\nLeft Join Result:")
print(left_join_result)
```

```
Inner Join Result:
   ID     Name  Score
0   2      Bob     85
1   3  Charlie     90

Left Join Result:
   ID     Name  Score
0   1    Alice    NaN
1   2      Bob   85.0
2   3  Charlie   90.0
```

7. Given the following two DataFrames data1 = {'ID': [1, 2, 3], 'Name': ['Alice', 'Bob', 'Charlie']} data2 = {'ID': [2, 3, 4], 'Score': [85, 90, 95]} df1 = pd.DataFrame(data1) df2 = pd.DataFrame(data2) Write a program to: a) Perform an inner join on ID. b) Perform a left join on ID.

```
# Define DataFrames
data1 = {'ID': [1, 2, 3], 'Name': ['Alice', 'Bob', 'Charlie']}
data2 = {'ID': [2, 3, 4], 'Score': [85, 90, 95]}
df1 = pd.DataFrame(data1)
df2 = pd.DataFrame(data2)

# a) Perform an inner join on ID
inner_join_result = pd.merge(df1, df2, on='ID', how='inner')

# b) Perform a left join on ID
left_join_result = pd.merge(df1, df2, on='ID', how='left')

# Display the results
print("Inner Join Result:")
print(inner_join_result)

print("\nLeft Join Result:")
print(left_join_result)
```

```
Inner Join Result:
   ID     Name  Score
0   2      Bob     85
1   3  Charlie     90

Left Join Result:
   ID     Name  Score
0   1    Alice    NaN
1   2      Bob   85.0
2   3  Charlie   90.0
```

8. Using the following DataFrame: data = { 'Department': ['HR', 'IT', 'Finance', 'HR', 'Finance', 'IT'], 'Employee': ['Alice', 'Bob', 'Charlie', 'David', 'Eve', 'Frank'], 'Salary': [50000, 60000, 55000, 52000, 59000, 61000] } df = pd.DataFrame(data) Write a program to: a) Calculate the total salary for each department. b) Calculate the average salary for each department

```
# Define the DataFrame
data = {
    'Department': ['HR', 'IT', 'Finance', 'HR', 'Finance', 'IT'],
    'Employee': ['Alice', 'Bob', 'Charlie', 'David', 'Eve', 'Frank'],
    'Salary': [50000, 60000, 55000, 52000, 59000, 61000]
}
df = pd.DataFrame(data)

# a) Calculate the total salary for each department
total_salary = df.groupby('Department')['Salary'].sum()

# b) Calculate the average salary for each department
average_salary = df.groupby('Department')['Salary'].mean()

# Display the results
print("Total Salary for Each Department:")
print(total_salary)

print("\nAverage Salary for Each Department:")
print(average_salary)
```

```
Total Salary for Each Department:
Department
Finance    114000
HR         102000
IT         121000
Name: Salary, dtype: int64

Average Salary for Each Department:
Department
Finance    57000.0
HR         51000.0
IT         60500.0
Name: Salary, dtype: float64
```

9. Given the following DataFrame: data = { 'Name': ['Alice', 'Bob', 'Charlie', 'David'], 'Age': [24, np.nan, 22, 32], 'Salary': [50000, 60000, np.nan, 80000] } df = pd.DataFrame(data) Write a program to: a) Replace the missing values in the Age column with the mean age. b) Drop rows where the Salary column has missing value

```
# Define the DataFrame
data = {
    'Name': ['Alice', 'Bob', 'Charlie', 'David'],
    'Age': [24, np.nan, 22, 32],
    'Salary': [50000, 60000, np.nan, 80000]
}
df = pd.DataFrame(data)

# a) Replace missing values in the Age column with the mean age
mean_age = df['Age'].mean()
df['Age'].fillna(mean_age, inplace=True)

# b) Drop rows where the Salary column has missing values
df = df.dropna(subset=['Salary'])

# Display the results
print("DataFrame after replacing missing Age values and dropping Salary rows:")
print(df)
```

10. Using the following DataFrames data = {'Month': ['Jan', 'Feb', 'Mar', 'Apr', 'May'], 'Sales': [300, 400, 500, 600, 700]}

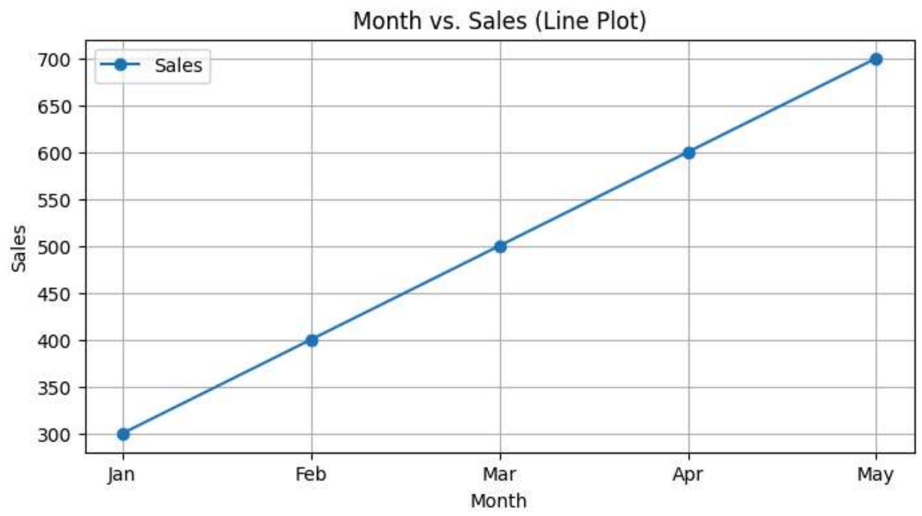
df = pd.DataFrame(data) Write a program to create: a) A line plot for Month vs. Sales. b) A bar plot for the same data.

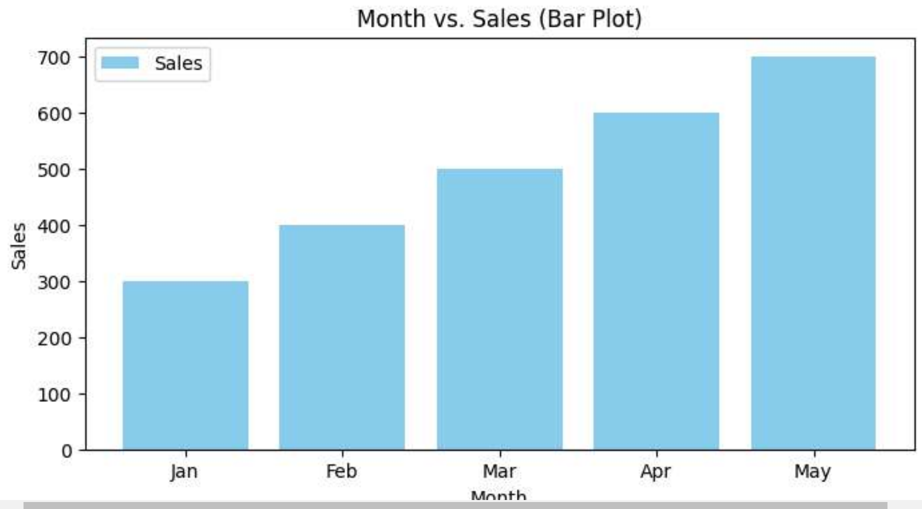
```
import matplotlib.pyplot as plt

# Define the DataFrame
data = {'Month': ['Jan', 'Feb', 'Mar', 'Apr', 'May'], 'Sales': [300, 400, 500, 600, 700]}
df = pd.DataFrame(data)

# a) Line plot for Month vs. Sales
plt.figure(figsize=(8, 4))
plt.plot(df['Month'], df['Sales'], marker='o', label='Sales')
plt.title('Month vs. Sales (Line Plot)')
plt.xlabel('Month')
plt.ylabel('Sales')
plt.grid(True)
plt.legend()
plt.show()

# b) Bar plot for Month vs. Sales
plt.figure(figsize=(8, 4))
plt.bar(df['Month'], df['Sales'], color='skyblue', label='Sales')
plt.title('Month vs. Sales (Bar Plot)')
plt.xlabel('Month')
plt.ylabel('Sales')
plt.legend()
plt.show()
```





11. Create a DataFrame with a Date column containing the following dates ['2023-01-01', '2023-01-02', '2023-01-03', '2023-01-04', '2023-01-05'] Add a column Sales with random integers between 100 and 500. Write a program to: a) Convert the Date column to a datetime object. b) Filter the rows where Sales are greater than 300.

```
import numpy as np
# Create the DataFrame
data = {
    'Date': ['2023-01-01', '2023-01-02', '2023-01-03', '2023-01-04', '2023-01-05'],
    'Sales': np.random.randint(100, 500, size=5)
}
df = pd.DataFrame(data)

# a) Convert the Date column to a datetime object
df['Date'] = pd.to_datetime(df['Date'])

# b) Filter rows where Sales are greater than 300
filtered_df = df[df['Sales'] > 300]

# Display the results
```

```
print("Original DataFrame:")
print(df)

print("\nFiltered DataFrame (Sales > 300):")
print(filtered_df)
```

```
Original DataFrame:
        Date  Sales
0 2023-01-01    280
1 2023-01-02    456
2 2023-01-03    157
3 2023-01-04    215
4 2023-01-05    118

Filtered DataFrame (Sales > 300):
        Date  Sales
1 2023-01-02    456
```

12. Create a series of 4 capital cities where the index is the name of corresponding country.

```
import pandas as pd

# Create the Series
capitals = pd.Series(
    data=['Washington, D.C.', 'London', 'Paris', 'Tokyo'],
    index=['USA', 'UK', 'France', 'Japan']
)

# Display the Series
print(capitals)
```

```
USA       Washington, D.C.
UK                  London
France               Paris
Japan                Tokyo
dtype: object
```

13. Create a dataframe of (at least 4) countries, with 2 variables: population and capital. Country name should be the index.

```
# Create the DataFrame
data = {
    'Population': [331002651, 67886011, 65273511, 126476461],  # Population values
    'Capital': ['Washington, D.C.', 'London', 'Paris', 'Tokyo']  # Capitals
}
countries = pd.DataFrame(data, index=['USA', 'UK', 'France', 'Japan'])

# Display the DataFrame
print(countries)
```

```
        Population           Capital
USA      331002651  Washington, D.C.
UK        67886011            London
France    65273511             Paris
Japan    126476461             Tokyo
```

14. How many columns are printed? How many variables does the dataframe contain? Dataset: titanic.csv

```
# Load the Titanic dataset
titanic_df = pd.read_csv('path_to_titanic.csv')  # Provide the correct file path

# Get the number of columns and variables
num_columns = len(titanic_df.columns)
num_variables = titanic_df.shape[1]

print(f"Number of columns: {num_columns}")
print(f"Number of variables: {num_variables}")
```

15. Create a matrix of data, and create a data frame from it using pd.DataFrame. Specify index (row names) and columns (variable names). Include at least 3 cities and 3 variables (e.g. population in millions, size in km2, and population density people per km2).

```
# Create a matrix of data
data = [
    [8.4, 789, 10600],  # City 1 (New York): population in millions, size in km^2, population density
    [3.9, 607, 6400],   # City 2 (London)
    [14.5, 174, 83000]   # City 3 (Tokyo)
]

# Create the DataFrame
cities = ['New York', 'London', 'Tokyo']
columns = ['Population (millions)', 'Size (km²)', 'Population Density (people per km²)']
df = pd.DataFrame(data, index=cities, columns=columns)

# Display the DataFrame
print(df)
```

```
          Population (millions)  Size (km²)  \
New York                    8.4         789
London                      3.9         607
Tokyo                      14.5         174

          Population Density (people per km²)
New York                                10600
London                                   6400
Tokyo                                   83000
```

16. Take your own city matrix and city data frame. From both of these extracts: a) population density (for all cities) b) data for the third city. For the data frame do it in two ways: using index, and using row number!

```
# Create a matrix of data for 3 cities
data = [
    [5.2, 400, 13000],  # City A: population in millions, size in km^2, population density
    [3.1, 500, 6200],   # City B
    [9.8, 600, 16333]   # City C
]

# Create the DataFrame
cities = ['City A', 'City B', 'City C']
columns = ['Population (millions)', 'Size (km²)', 'Population Density (people per km²)']
df = pd.DataFrame(data, index=cities, columns=columns)

# a) Population density for all cities
population_density = df['Population Density (people per km²)']

# b) Data for the third city (City C)
# 1) Using index (by city name)
city_c_data_by_index = df.loc['City C']

# 2) Using row number (3rd row in the DataFrame)
city_c_data_by_row_number = df.iloc[2]

# Display the results
print("Population Density for all cities:")
print(population_density)

print("\nData for the third city (City C) using index:")
print(city_c_data_by_index)

print("\nData for the third city (City C) using row number:")
print(city_c_data_by_row_number)
```

```
Population Density for all cities:
City A    13000
City B     6200
City C    16333
Name: Population Density (people per km²), dtype: int64

Data for the third city (City C) using index:
Population (millions)                      9.8
Size (km²)                               600.0
Population Density (people per km²)    16333.0
Name: City C, dtype: float64

Data for the third city (City C) using row number:
Population (millions)                      9.8
```

```
Size (km²)                              600.0
Population Density (people per km²)    16333.0
Name: City C, dtype: float64
```