**Pandas**

Pandas is a powerful and open-source Python library. The Pandas library is used for data manipulation and analysis. Pandas consist of data structures and functions to perform efficient operations on data.

Pandas is well-suited for working with tabular data, such as spreadsheets or SQL tables.

The name "Pandas" has a reference to both "Panel Data", and "Python Data Analysis" and was created by Wes McKinney in 2008.

Pandas allows us to analyze big data and make conclusions based on statistical theories.

Pandas can clean messy data sets, and make them readable and relevant.

Relevant data is very important in data science.

**Steps**

1. Open Anaconda Prompt
2. Type ‘jupyter notebook’
3. It will redirect to Home page of Jupyter notebook on browser
4. In Files tab open Desktop folder
5. Create new folder using New button on top right and name it
6. Open the folder and click on Upload button
7. Upload the dataset file (.csv , .xlsv) on which you have to perform tasks
8. Create a Python file by clicking on new button and choosing ‘Python 3’ file
9. On creating, file will open in new tab of the browser
10. Import pandas
11. Store dataset file in a variable

**Perform the following operations on dataset**

Before performing tasks on dataset we need to

1. Import pandas

import pandas as pd

1. Store dataset file in a variable

ds = pd.read\_csv("Student Depression Dataset.csv")

1. **What are the first five rows of the dataset?**

df.head()

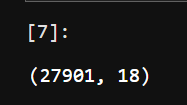
**OUTPUT:**

****

1. **What is the total number of rows and columns in the dataset?**

df.shape # Returns (number of rows, number of columns)

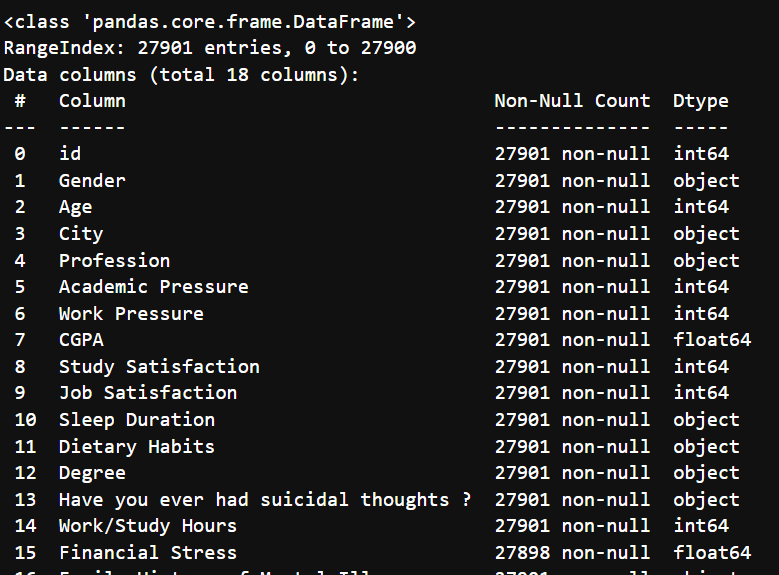
**OUTPUT:**



**3.How to Get Detailed Information About the Dataset?**

df.info() # Displays column names, data types, and non-null values

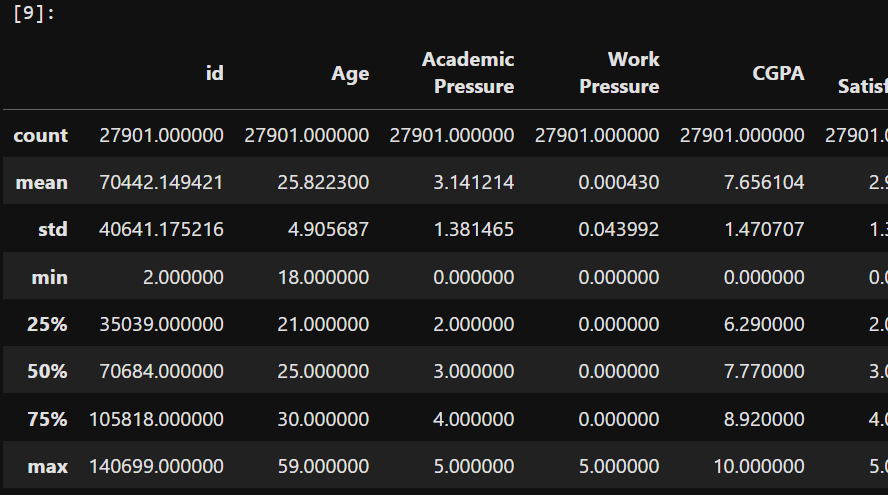
**OUTPUT:**

****

**5. How to Get Summary Statistics of the Dataset?**

df.describe() # Provides mean, median, std, min, max, etc.

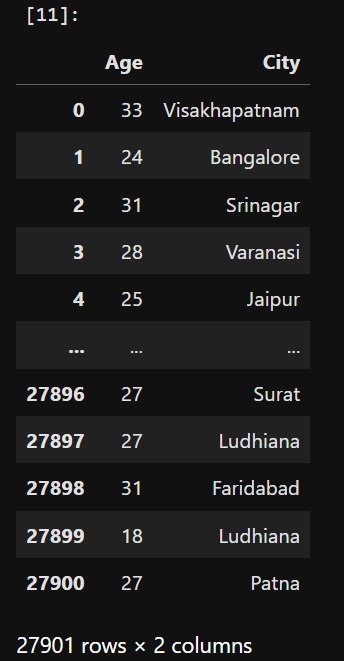
**OUTPUT:**

****

**6. How to Select Specific Columns (e.g., Age and City)?**

df[['Age', 'City']]

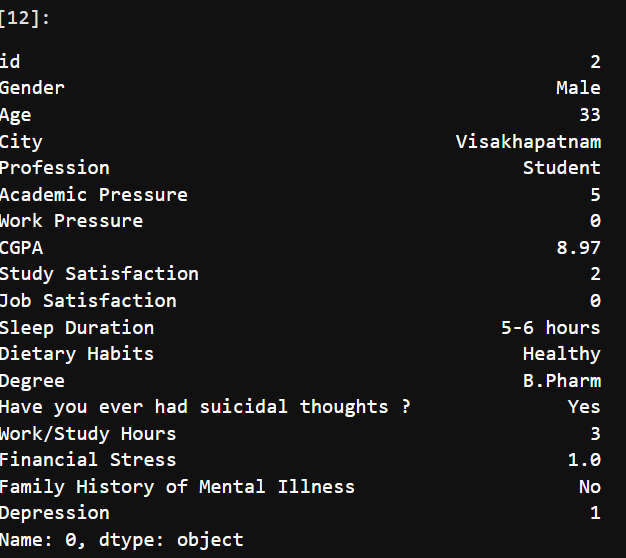
**OUTPUT:**

****

**7. How to Select a Specific Row by Index?**

df.iloc[0] # First row

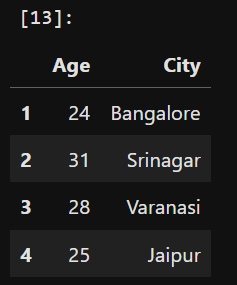
**OUTPUT :**

****

**8. How to Select Specific Rows and Columns?**

df.iloc[1:5, 2:4] # Select rows 1-4 and columns 2-3

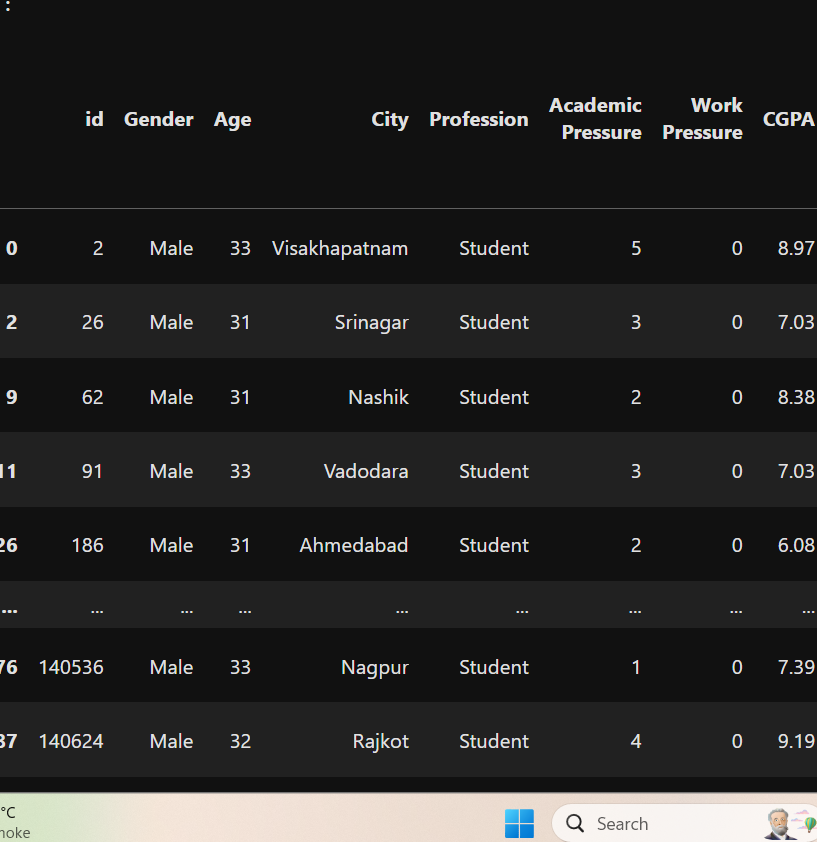
**OUTPUT:**

****

**9. How to Filter Students Older Than 30?**

df[df['Age'] > 30]

**OUTPUT:**

****

**10. How to Check for Missing Values?**

df.isnull().sum() # Counts NaN values per column

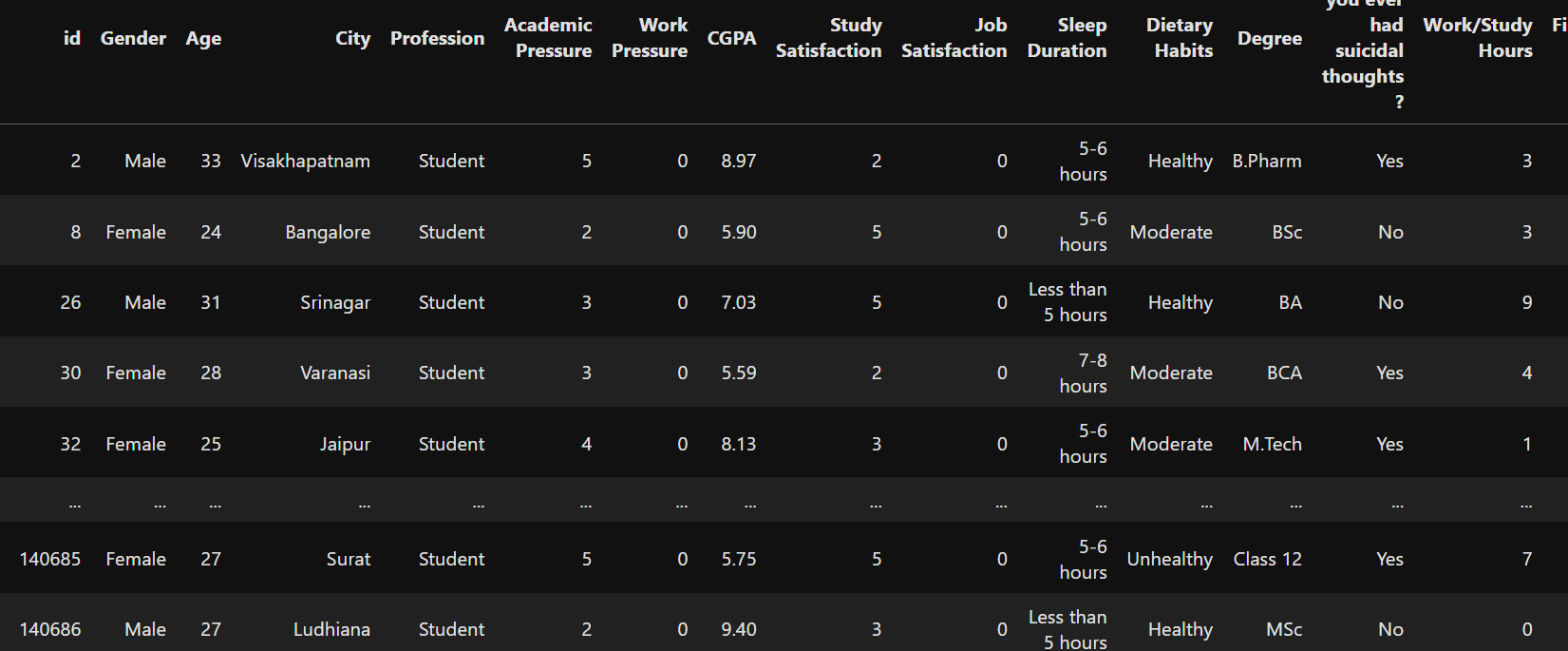
**OUTPUT:**

****

**11.How to Drop Rows with Missing Values?**

df.dropna(inplace=True)

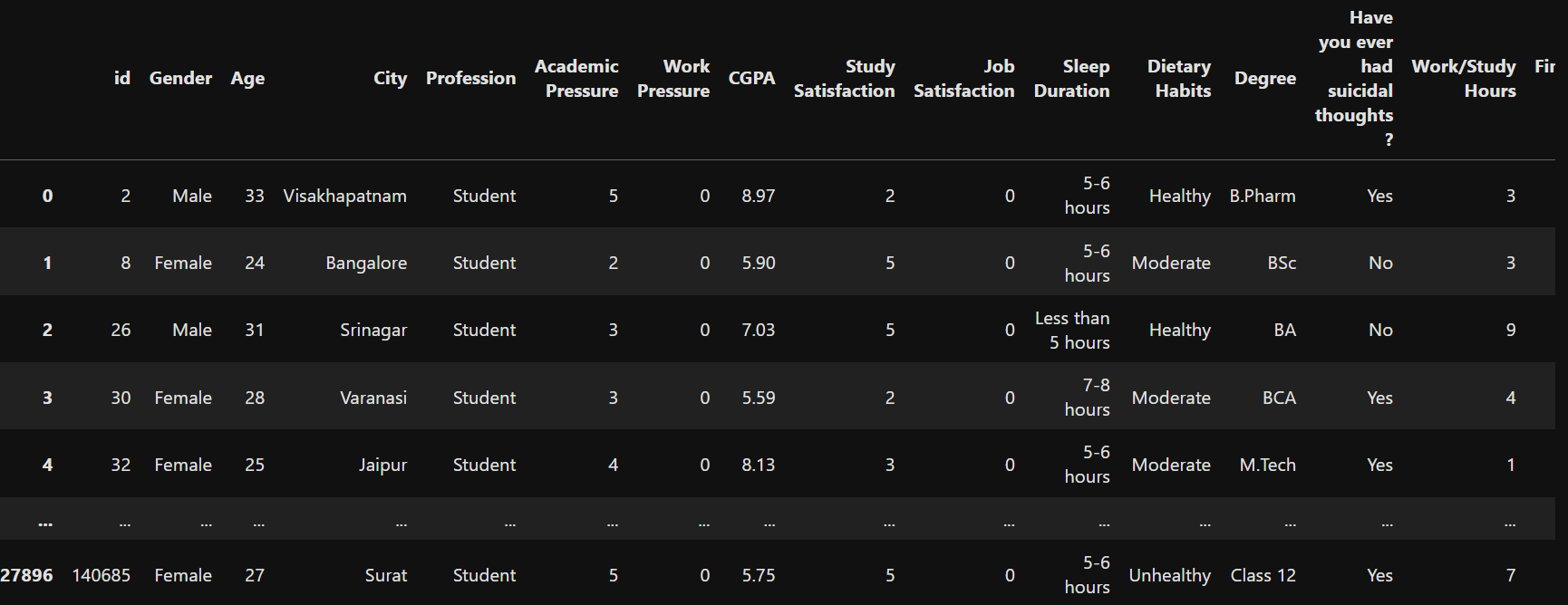
**OUTPUT:**

****

1. **How to Create a New Column Using Existing Columns?**

df['mine'] = df['id'] + df['Age']

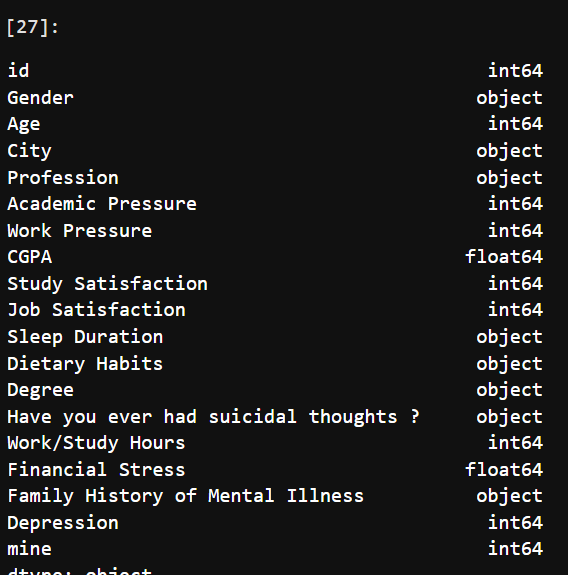
**OUTPUT:**

****

1. **How to Get Data Types of Each Column?**

df.dtypes

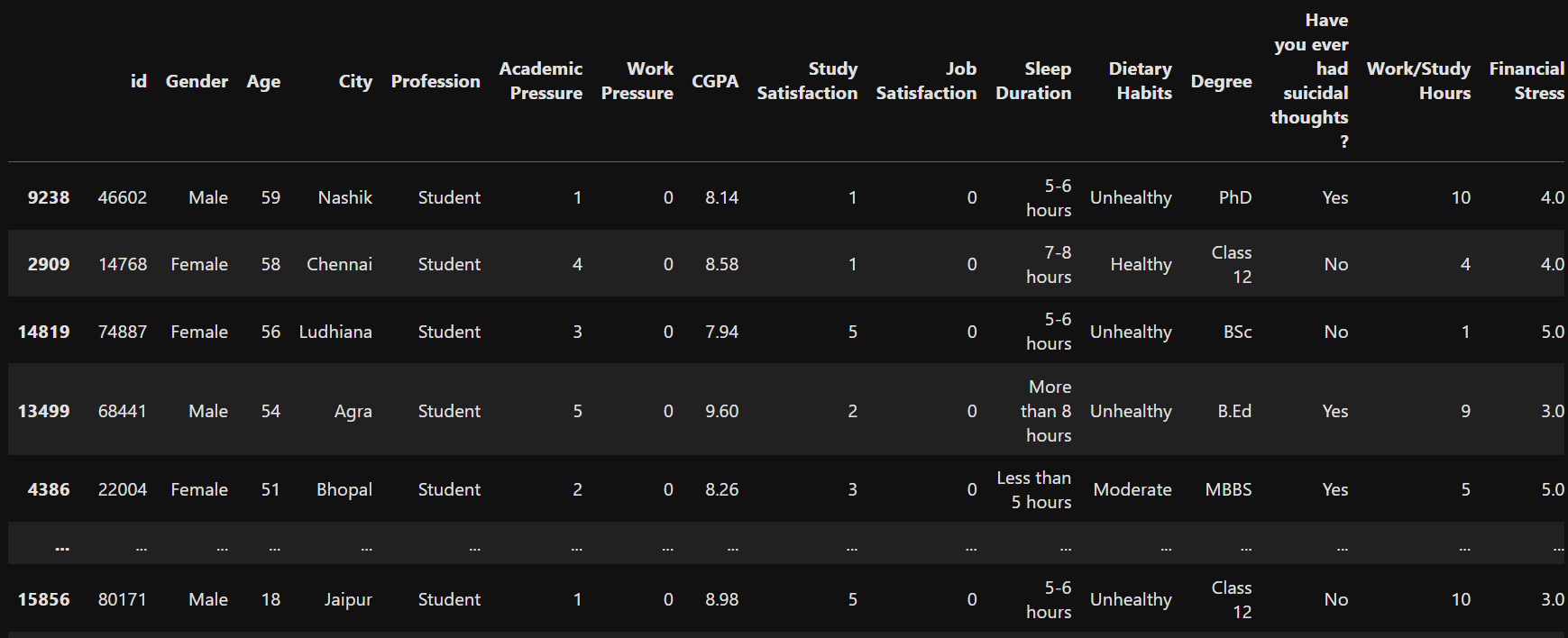
**OUTPUT:**

****

1. **How to Sort the Data by Age in Descending Order?**

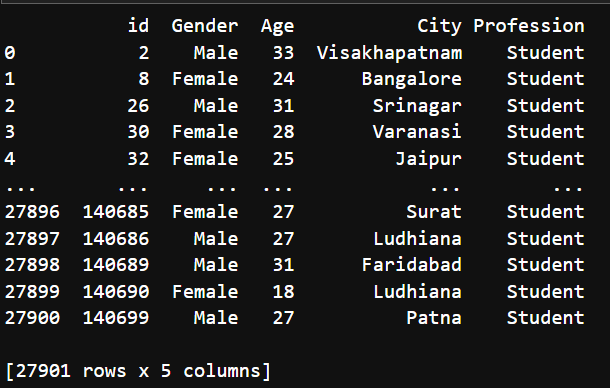
df.sort\_values(by='Age', ascending=False)

**OUTPUT:**

****

1. **How to Display Only the First 5 Columns?**

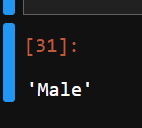
print(df.iloc[:, :5])

**OUTPUT:** ****

1. **How to Select a Specific Value Using Row and Column Index?**

df.iloc[0, 1] # Selects the element in first row, second column

**OUTPUT:**

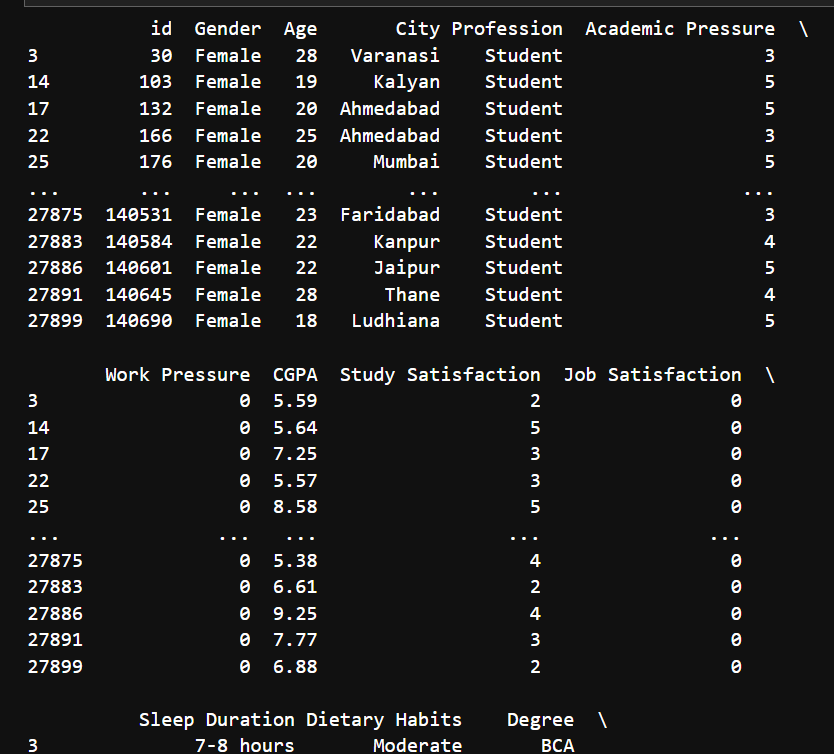
****

1. **How to Find All Rows Where Gender is Female and Depression is 1?**

female\_depressed = df[(df['Gender'] == 'Female') & (df['Depression'] == 1)]

print(female\_depressed)

**OUTPUT:**

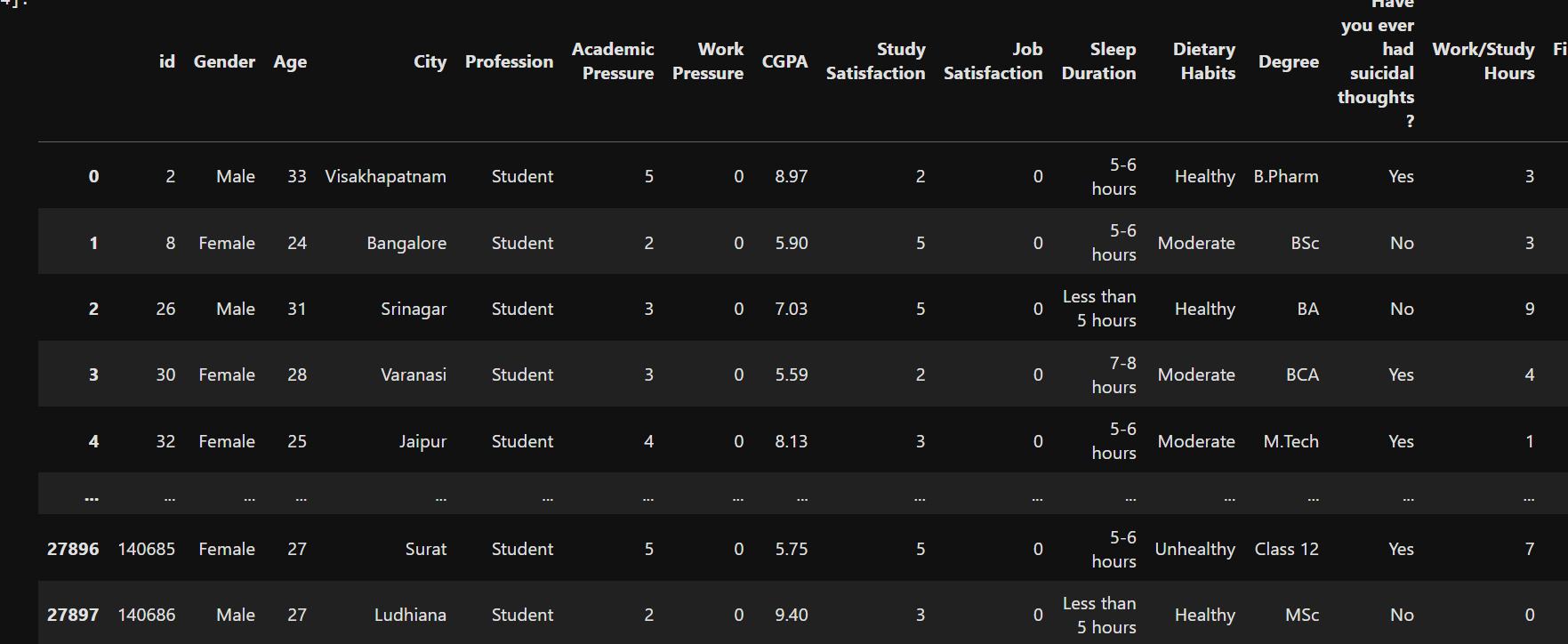
****

1. **How to Count Unique Values in the "City" Column?**

unique\_cities = df['City'].nunique()

print(f"Unique cities: {unique\_cities}")

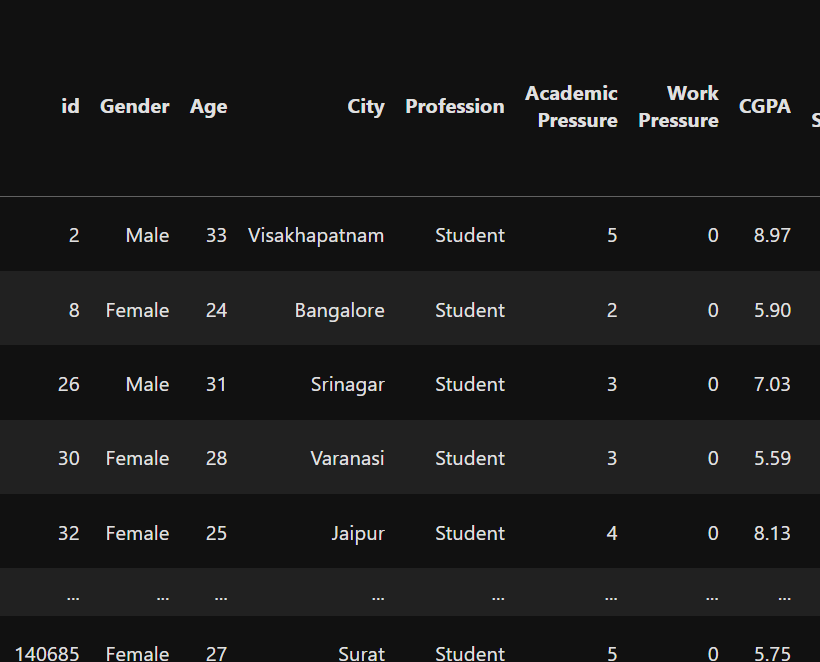
**OUTPUT:**

****

**19.How to Fill Missing Values with 0?**

df.fillna(0, inplace=True)

**OUTPUT**:



**20.How to Calculate the Average CGPA?**

average\_cgpa = df['CGPA'].mean()

print(f"Average CGPA: {average\_cgpa}")

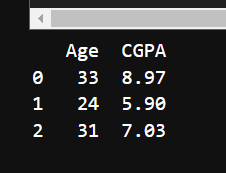
**OUTPUT:**



**21.How to Select the First 3 Rows of Age and CGPA?**

df.loc[:2, ['Age', 'CGPA']]

**OUTPUT:**

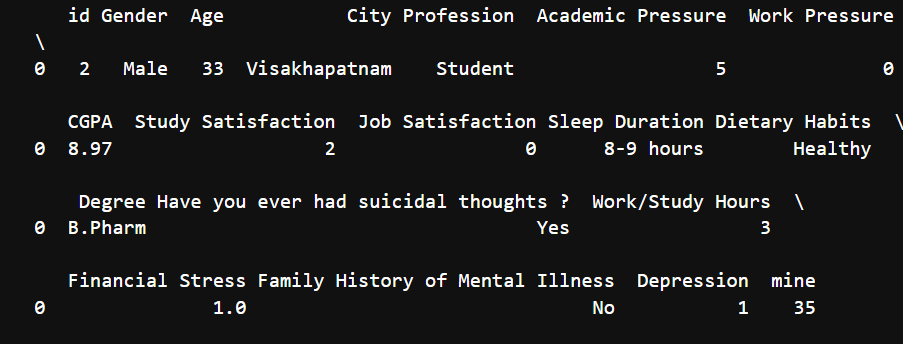
****

**22. How to Update a Student's Sleep Duration (id=2 to "8-9 hours")?**

df.loc[df['id'] == 2, 'Sleep Duration'] = "8-9 hours"

print(df.loc[df['id'] == 2])

**OUTPUT:**

****

**23. How to Check If a Row with id=0 Exists and Modify Its Sleep Duration?**

if (df['id'] == 0).any():

df.loc[df['id'] == 0, 'Sleep Duration'] = "8-9 hours"

print(df.loc[df['id'] == 0])

else:

print("No row with id == 0 found.")

**OUTPUT:**

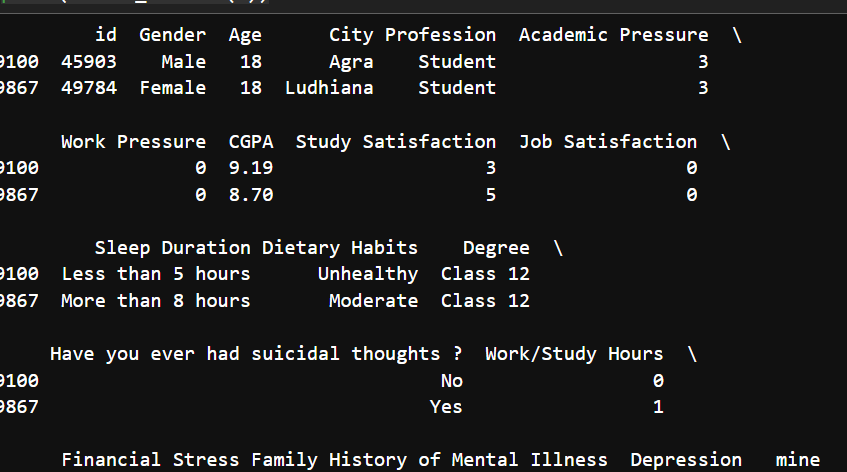
****

**24. How to Sort the Data by Age in Ascending Order?**

sorted\_df = df.sort\_values(by='Age')

print(sorted\_df.head(2))

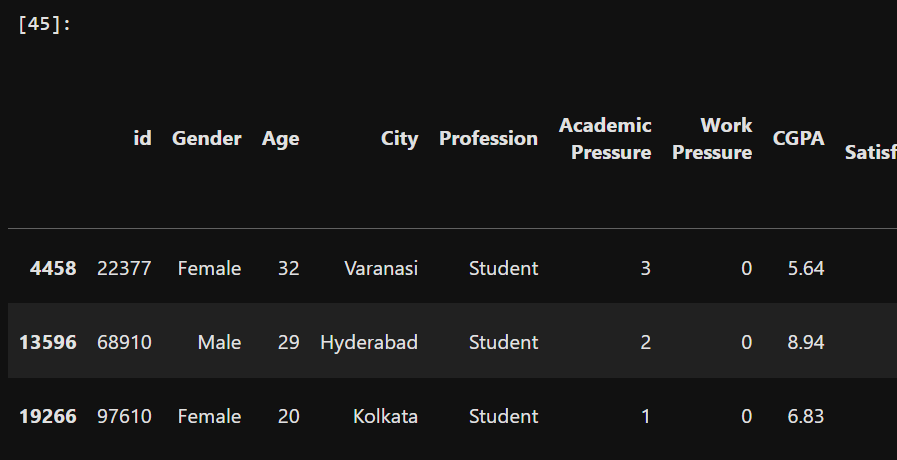
**OUTPUT:**

****

**25.How to Find Rows with Missing or Empty Values?**

df[df.isnull().any(axis=1) | (df == '').any(axis=1)]

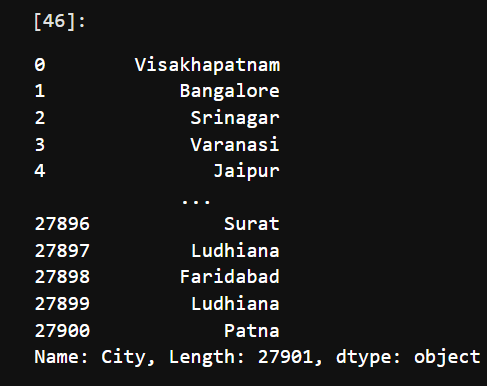
**OUTPUT:**

****

**26.How to Fill Missing Values in the "City" Column with the Mode of "Financial Stress"?**

df['City'].fillna(df['Financial Stress'].mode()[0]) # Filling NaN with mode

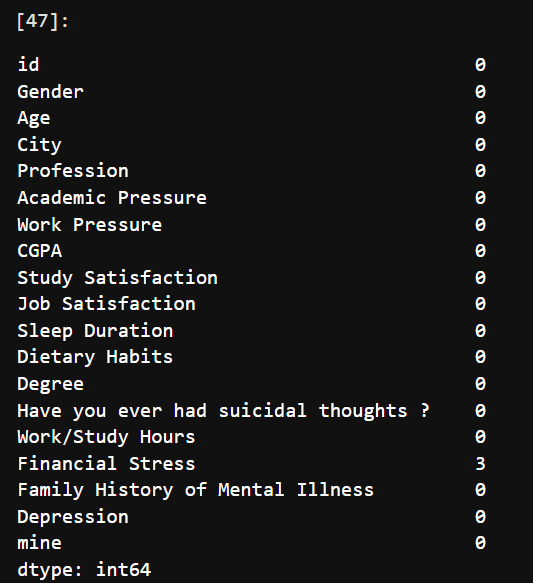
**OUTPUT:**

****

**27. How to Check the Count of Missing Values in Each Column?**

df.isnull().sum()

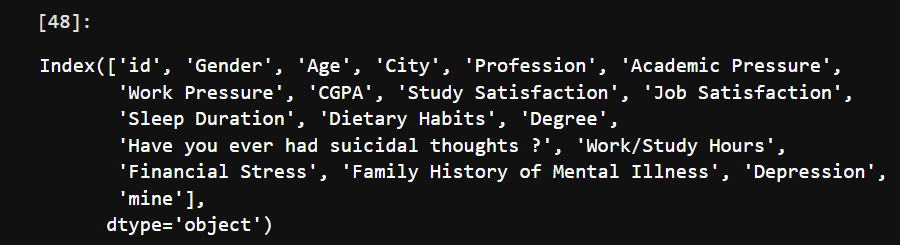
**OUTPUT:**

****

**28.How to Display All Column Names in the Dataset?**

df.columns

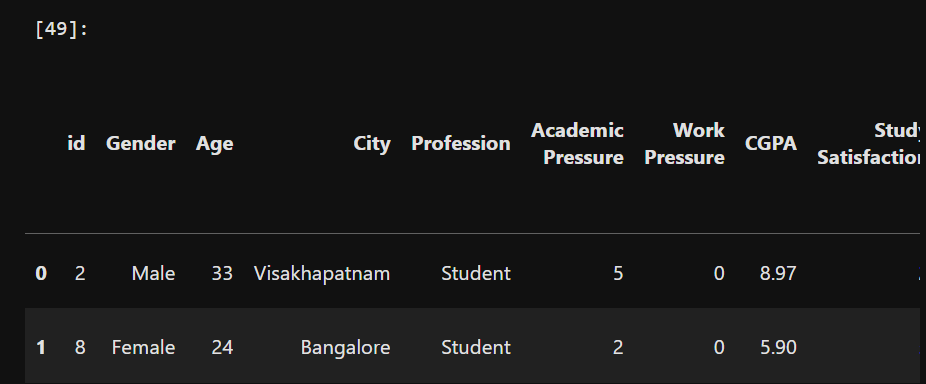
**OUTPUT:**

****

**29.How to Select Multiple Rows by Index?**

df.iloc[[0,1]] # Selects the first and second row

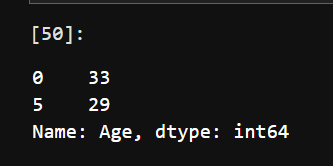
**OUTPUT:**

****

**30.How to Select a Specific Value Using Row and Column Index?**

df.iloc[[0,5],2] # Selects the value at row indices 0 and 5, column index 2

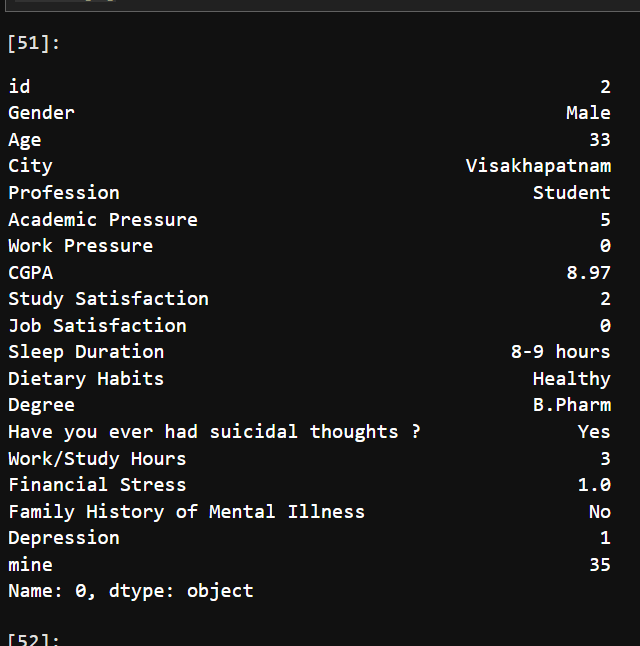
**OUTPUT:**

****

**31.How to Select a Row by Index Using .loc[]?**

df.loc[0] # Selects the row where index = 0

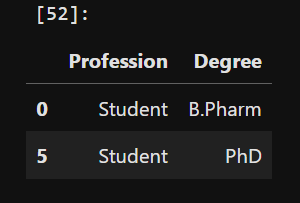
**OUTPUT:**

****

**32. How to Select Specific Rows and Columns Using .loc[]?**

df.loc[[0,5],['Profession','Degree']]

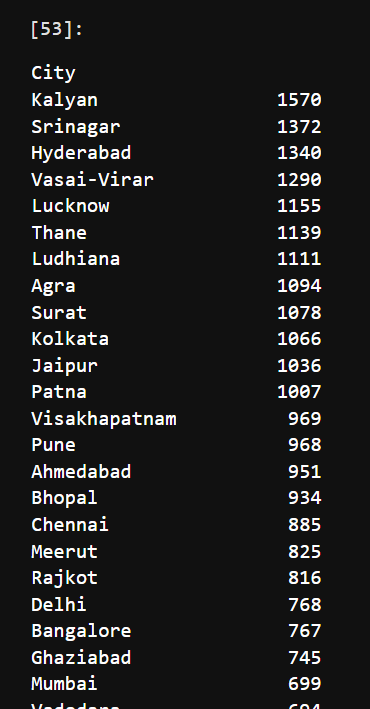
**OUTPUT:**

****

**33.How to Get the Count of Unique Values in the "City" Column?**

df['City'].value\_counts()

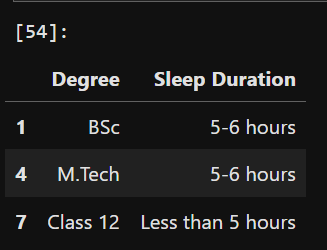
**OUTPUT:**

****

**34. How to Select Specific Rows and Columns?**

df.loc[[1,4,7],[‘Degree’,’Sleep Duration’]]

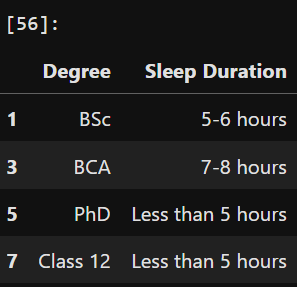
**OUTPUT:**

****

**35. How to Select a Range of Rows with Specific Columns Using .loc[]?**

df.loc[1:8:2, ['Degree','Sleep Duration']]

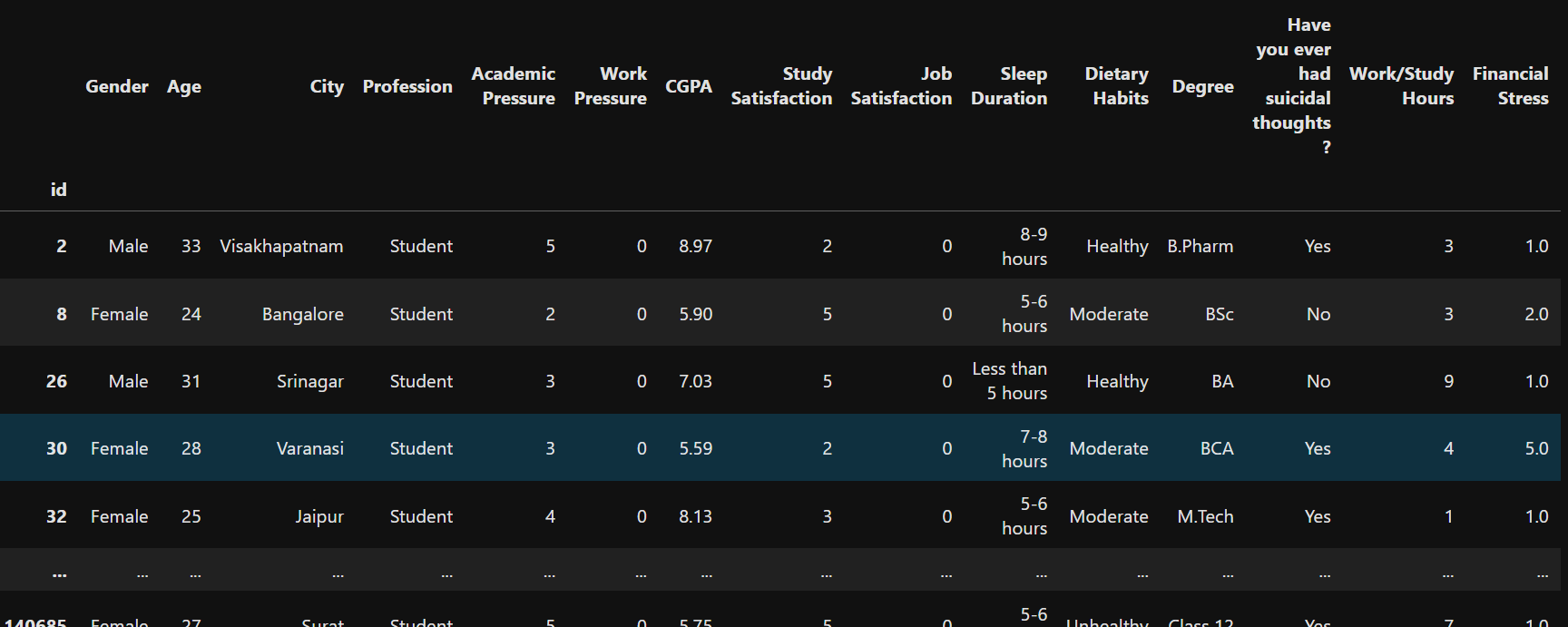
**OUTPUT:**

****

**36. How to Set the "id" Column as the Index?**

df.set\_index('id', inplace=True)

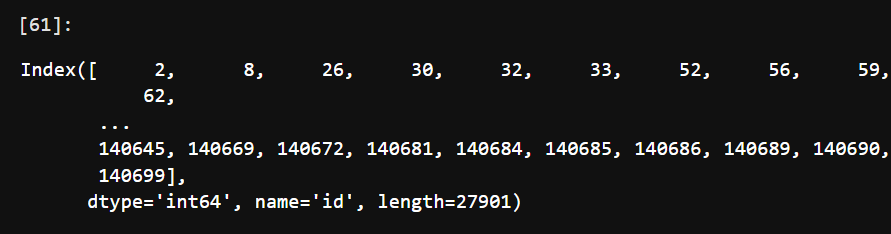
**OUTPUT:**

****

**37. How to Display the Current Index of the DataFrame?**

df.index

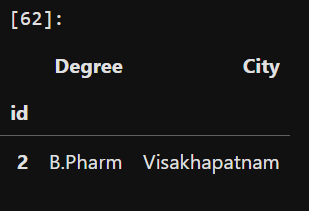
**OUTPUT:**

****

**38. How to Select a Specific Row and Columns Using .loc[]?**

df.loc[[2],['Degree','City']]

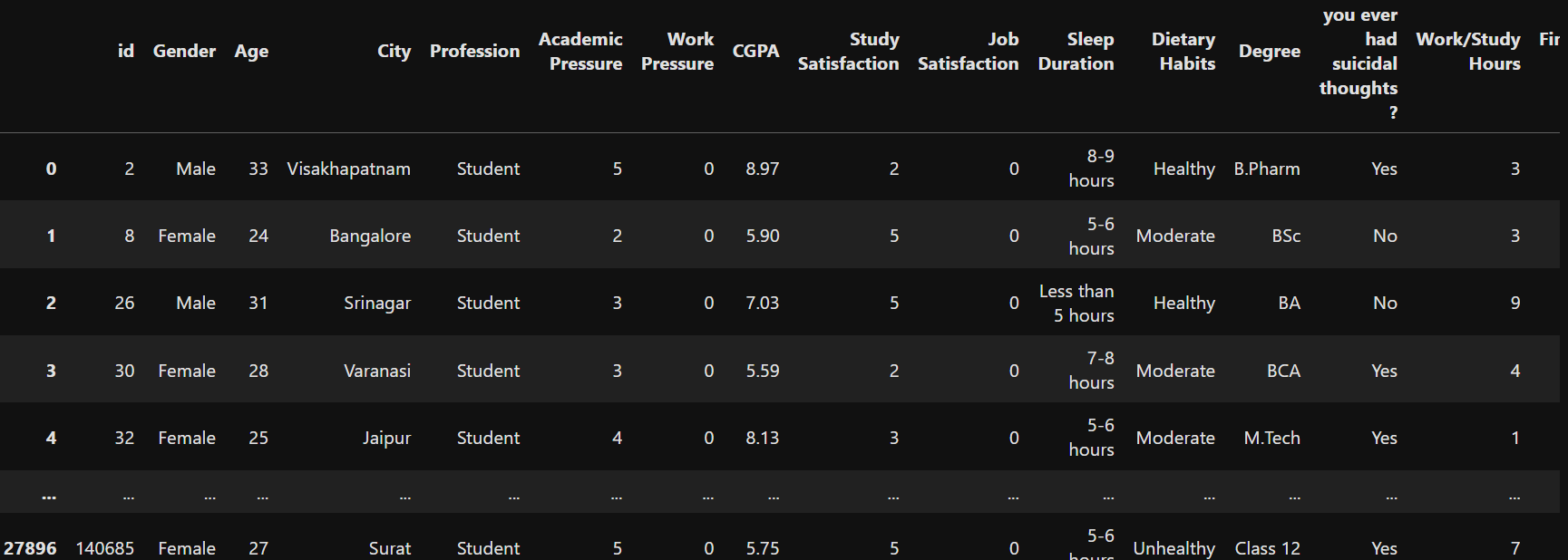
**OUTPUT:**

****

**39. How to Reset the Index of the DataFrame?**

df.reset\_index(inplace=True)

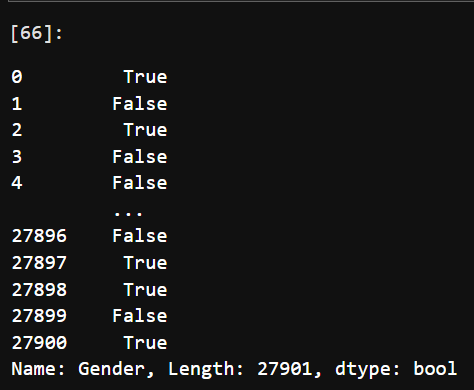
**OUTPUT:**



**40. How to Filter Male Students Using a Boolean Condition?**

df[df['Gender'] == 'Male']

**OUTPUT:**

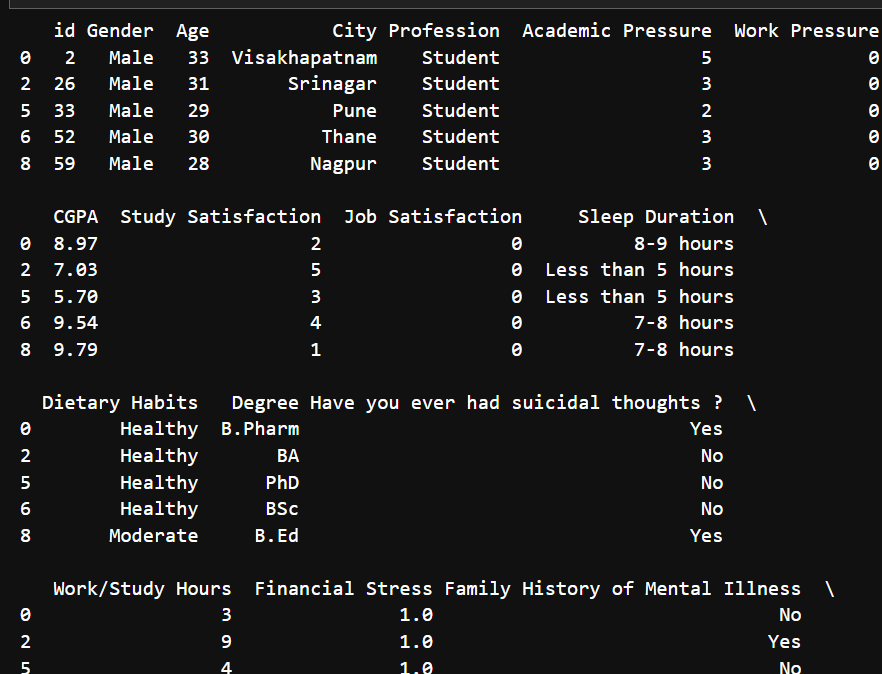
****

**41.How to Select Only Male Students Using .query()?**

p = df.query('Gender == "Male"')

print(p.head())

**OUTPUT:**

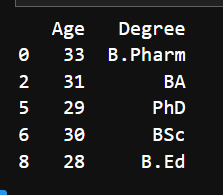
****

**42.How to Select Only Male Students and Display Their "Age" and "Degree"?**

p = df.query('Gender == "Male"')[['Age','Degree']]

print(p.head())

**OUTPUT:**

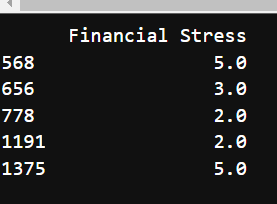
****

**43.How to Find Students from "Kalyan" With a "BCA" Degree and Their Financial Stress?**

a = df[(df['City'] == "Kalyan") & (df['Degree'] == "BCA")][['Financial Stress']]

print(a.head())

**OUTPUT:**

****

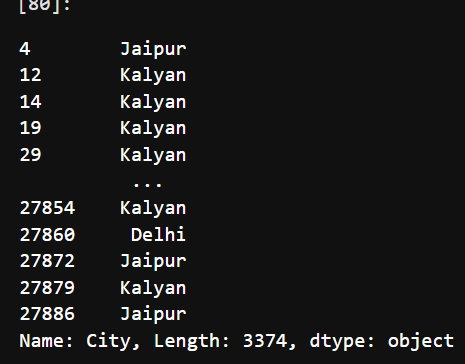
**44.How to Filter Students From a List of Cities (Kalyan, Jaipur, Delhi)?**

ci\_ty = ['Kalyan','Jaipur','Delhi']

filter = df['City'].isin(ci\_ty)

df.loc[filter, 'City']

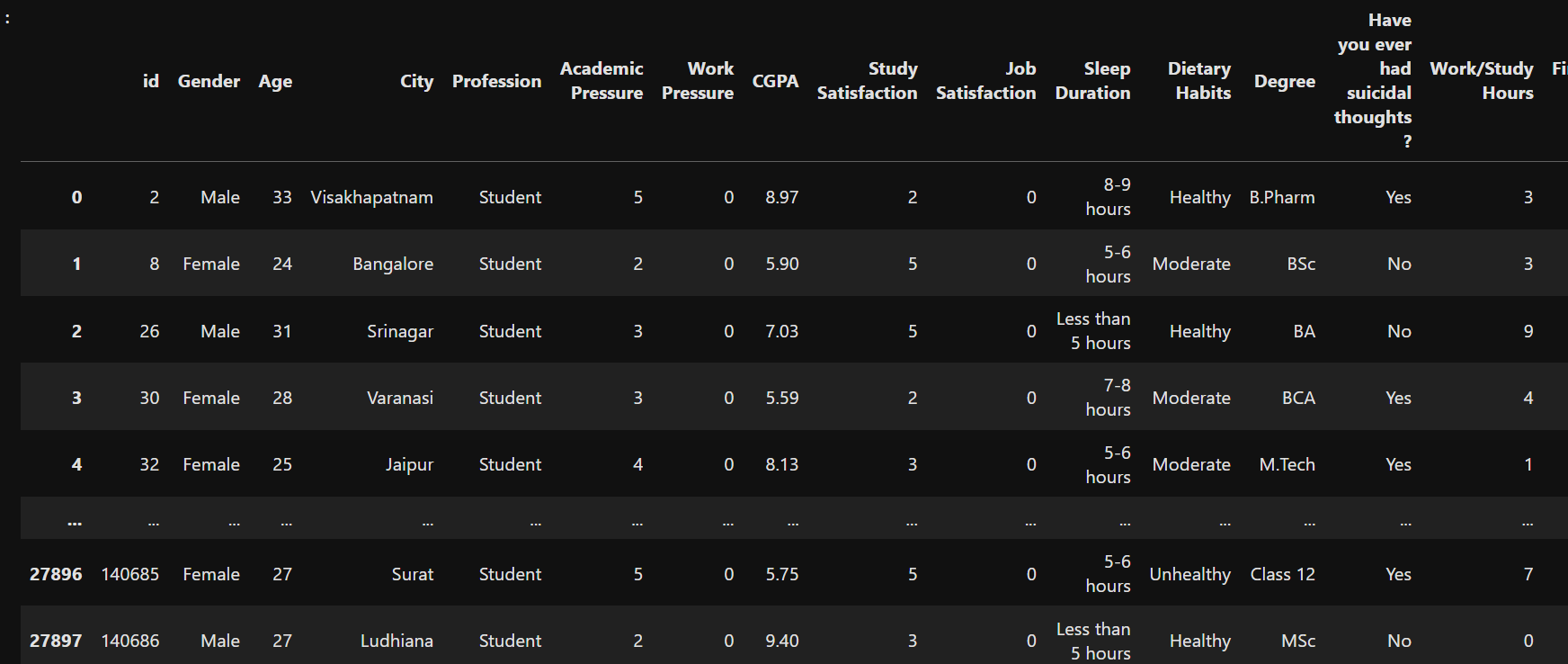
**OUTPUT:**

****

**45.How to Check If Any Student Has an MSc Degree?**

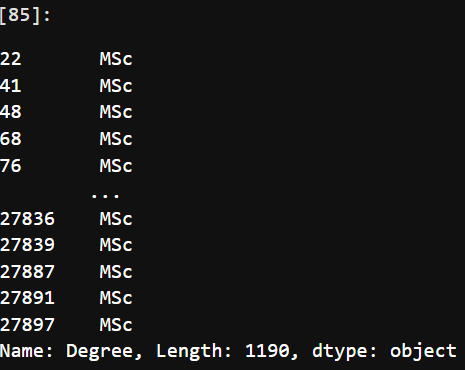
m = df['Degree'].str.contains('MSc', na=False)

**OUTPUT**:



df.loc[m, 'Degree']

**OUTPUT**:



**Questions on DataFrame Creation:**

**1. What are the different methods used in the code to create a Pandas DataFrame?**

There are four methods used to create a Pandas DataFrame:

* Using a dictionary
* Using the zip() function with lists
* Using NumPy arrays
* Using a list of dictionaries
* **Code for creating a Pandas DataFrame using a dictionary:**

import pandas as pd

# Creating a dictionary with student data

data = {

"Name": ["Sana", "Sejal", "Riya", "Sneha"],

"Age": [20, 30, 50, 80],

"Grade": ["A", "B", "C", "D"],

"City": ["Manikpur", "Kalamboli", "Nerul", "Pune"]

}

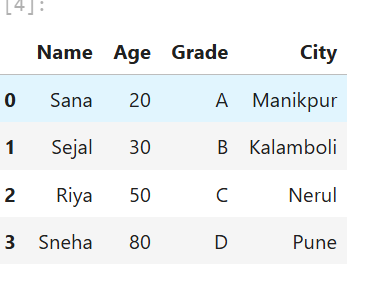
# Creating DataFrame from the dictionary

df = pd.DataFrame(data)

# Display the DataFrame

print(df)

**OUTPUT:**

****

* Using the zip() function with lists

The zip() function pairs corresponding elements from multiple lists and allows us to create a DataFrame by converting them into tuples.

**MANUAL APPROACH**

# Data for each column

names=["Raaj","Harshali","Sunita"]

ages=[20,50,60]

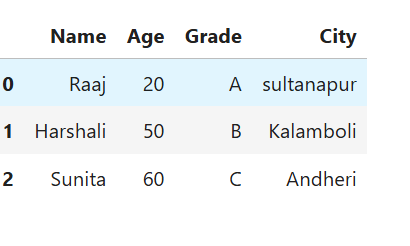
grades=["A","B","C"]

cities=["sultanapur","Kalamboli","Andheri"]

df=pd.DataFrame(list(zip(names,ages,grades,cities)),columns=["Name","Age","Grade","City"])

df

**OUTPUT:**



* Using NumPy arrays

numpy.array() is used to store column data efficiently and perform mathematical operations more effectively.

**# USING NUMPY ARRAY METHOD**

import pandas as pd

import numpy as np

# Using numpy arrays

names=np.array(["Atharv","Akash","Amit"])

ages=np.array([20,50,60])

grades=np.array(["A","D","G"])

cities=np.array(["Panvel","Sultana","Mumbai"])

df=pd.DataFrame({

"Names":names,

"Age":ages,

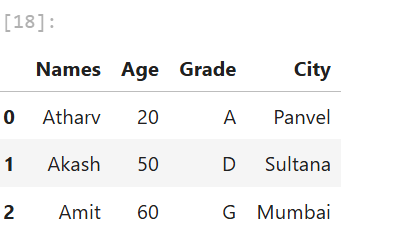
"Grade":grades,

"City":cities

})

df

**OUTPUT:**



* Using a list of dictionaries

A list of dictionaries represents each row as a dictionary, making it easy to create records dynamically. Unlike the dictionary method, where keys are column names, each dictionary in the list represents a row.

**# USING LIST OF DICTIONARIES**

import pandas as pd

# List of dictionaries representing each student record

data=[

{"Name":"Arav","Age":20,"Grade":"A","City":"Manikpur"},

{"Name":"Arnav","Age":30,"Grade":"B","City":"Snikpur"}

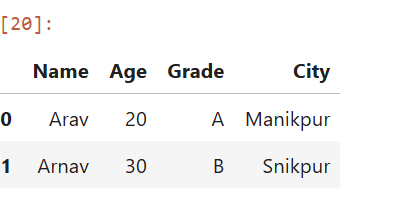
]

# Creating DataFrame

df = pd.DataFrame(data)

df

**OUTPUT:**

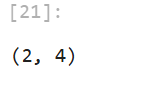


**2. What does df.shape return, and how can we use it to find the number of rows and columns?**

df.shape returns a tuple (rows, columns), where the first value is the number of rows and the second is the number of columns.

df.shape

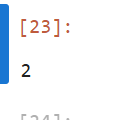
**OUTPUT:**



rows, columns = df.shape

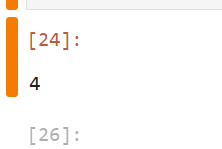
rows

**OUTPUT:**

****

Columns

**OUTPUT:**

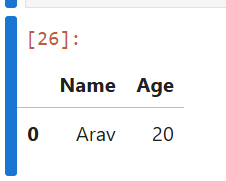


**3. How can we filter out only those students who have received an "A" grade?**

We can filter the DataFrame using a condition on the "Grade" column

df[['Name', 'Age']][df['Grade'] == 'A']

**OUTPUT:**



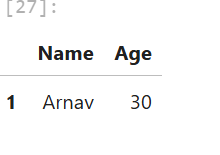
**4. How can we find students whose age is the maximum in the DataFrame?**

We can use the max() function to get the maximum age and filter rows accordingly.

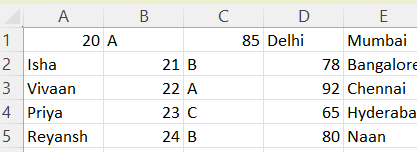
#SET INDEX

df[['Name' ,'Age']] [df['Age']==df['Age'].max()]

**OUTPUT:**



**Importing Data from Excel-Sheet:**



**Q: What does the skiprows=1 argument do in pd.read\_csv()?**

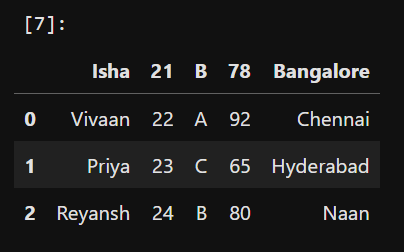
**A:** It skips the first row of the CSV file while reading the data.

import pandas as pd

df=pd.read\_csv("C:\\Users\\Sana\\OneDrive\\Desktop\\Book1.csv",skiprows=1)

df

**OUTPUT:**

****

**Q: What is the effect of using header=None when reading a CSV file?**

**A:** It treats the first row as data instead of column headers.

df=pd.read\_csv("C:\\Users\\Sana\\OneDrive\\Desktop\\Book1.csv",header=None)

df

**OUTPUT:**

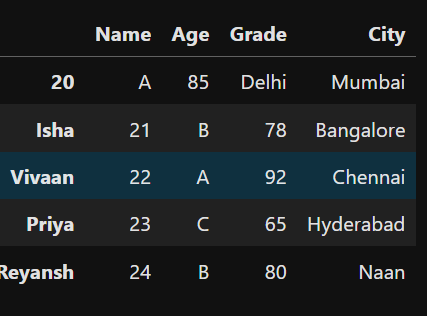
****

**Q: How does specifying names=["Name","Age","Grade","City"] affect the DataFrame?**

**A:** It assigns custom column names instead of using the default ones from the file.

df=pd.read\_csv("C:\\Users\\Sana\\OneDrive\\Desktop\\Book1.csv",names=["Name","Age","Grade","City"])

**OUTPUT**:

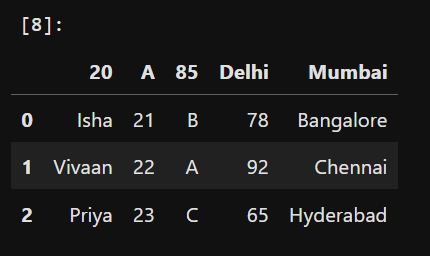


**Q: What happens when nrows=3 is used while reading the CSV file?**

**A:** It only reads the first three rows of the dataset.

df=pd.read\_csv("C:\\Users\\Sana\\OneDrive\\Desktop\\Book1.csv",nrows=3)

**OUTPUT:**

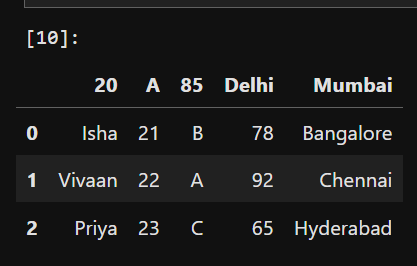


**Q: What does df.to\_csv("Book2.csv", index=False) do?**

**A:** It saves the DataFrame to a CSV file named "Book2.csv", excluding the index column.

df.to\_csv("Book2.csv",index=False)

**OUTPUT:**



**Data Conversion**

**Q: What is the purpose of the convert\_City\_cell function?**

**A:** It replaces any cell with the string 'NaN' in the "City" column with 'Hello'.

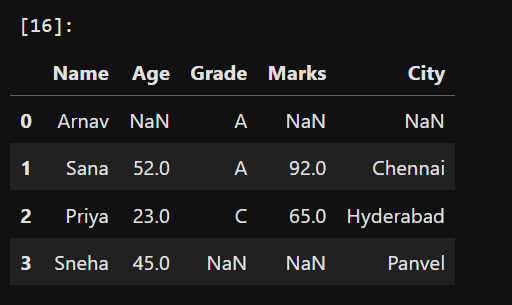
import pandas as pd

import numpy as np

df=pd.read\_csv("C:\\Users\\Sana\\pandas\\Book2.csv",skiprows=0)

df

**OUTPUT:**



**Q: What is the purpose of the convert\_City\_cell function?**

**A:** It replaces any cell with the string 'NaN' in the "City" column with 'Hello'.

# Define the converter function

def convert\_City\_cell(cell):

if cell=='NaN': # Check if the cell is NaN

return 'Hello'

return cell

**Q: How does the convert\_Grade\_cell function modify the data?**

**A:** It replaces any cell with the string 'NaN' in the "Grade" column with 'zero'.

def convert\_Grade\_cell(cell):

if cell=='NaN': # Check if the cell is NaN

return 'zero'

return cell

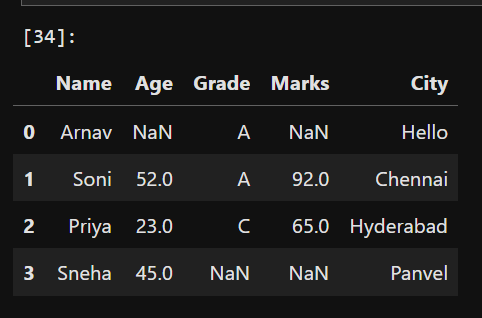
**Q: What will happen if a cell contains 'NaN' in the 'City' or 'Grade' column?**

**A:** It will be replaced with 'Hello' in the "City" column and 'zero' in the "Grade" column.

# Read the CSV file using the converter for the 'City' column

df = pd.read\_csv('C:\\Users\\Sana\\pandas\\Book2.csv', converters={'City': convert\_City\_cell,'Grade': convert\_Grade\_cell})

**OUTPUT:**

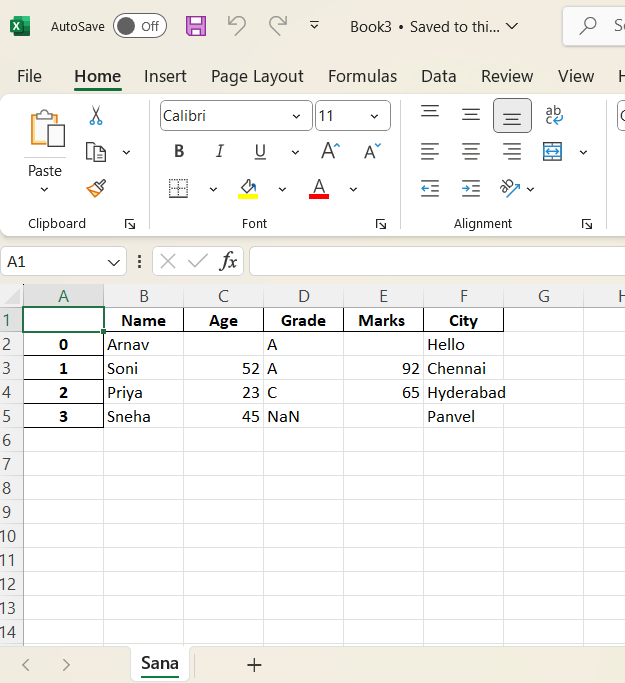


**What does df.to\_excel("Book3.xlsx", sheet\_name='Sana') do?**

**A:** It saves the DataFrame to an Excel file named "Book3.xlsx" with the sheet name "Sana".

df.to\_excel("Book3.xlsx",sheet\_name='Sana')

**OUTPUT**:

Faker:

**Faker** is a library used to generate realistic fake data for testing, development, and data science applications. It can create names, addresses, emails, phone numbers, and even structured datasets, helping developers and data scientists work without using real or sensitive data. Available in multiple programming languages like Python, JavaScript, and R, Faker is widely used for mock data generation, machine learning model training, and privacy protection.

Faker allows customization by defining specific data formats and locales, making it adaptable for different regions and use cases. It supports generating structured datasets, such as user profiles, credit card details, and product descriptions, which can be useful for simulating real-world scenarios. Additionally, Faker can create large volumes of synthetic data quickly, making it a valuable tool for stress testing applications and populating databases with meaningful sample data.

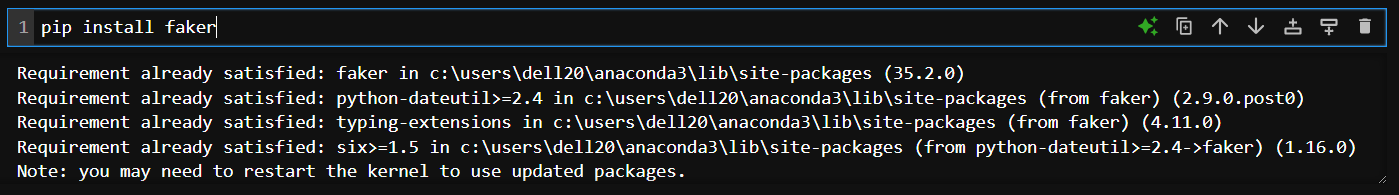
Steps:

1. Open Anaconda Prompt
2. Type ‘jupyter notebook’
3. It will redirect to Home page of Jupyter notebook on browser
4. In Files tab open Desktop folder
5. Create new folder using New button on top right and name it
6. Open the folder and click on Upload button
7. Upload the dataset file (.csv , .xlsv) on which you have to perform tasks
8. Create a Python file by clicking on new button and choosing ‘Python 3’ file
9. On creating, file will open in new tab of the browser
10. Import faker
11. Store dataset file in a variable

Performing some operation on dataset:

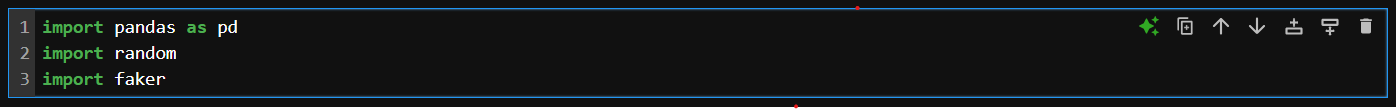
To install faker

* pip install faker



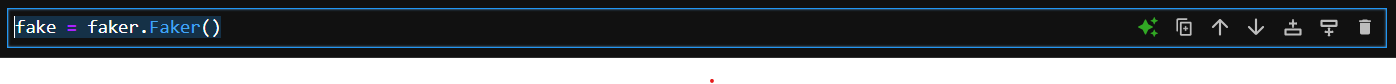
Import libraries

* import pandas as pd
* import random
* import faker



Instance of the faker class to generate fake data

* fake = faker.Faker()



Q.1.

* data ={

"first Name":["Arav","Isha","Mohammad","Zoya","Ayann"],

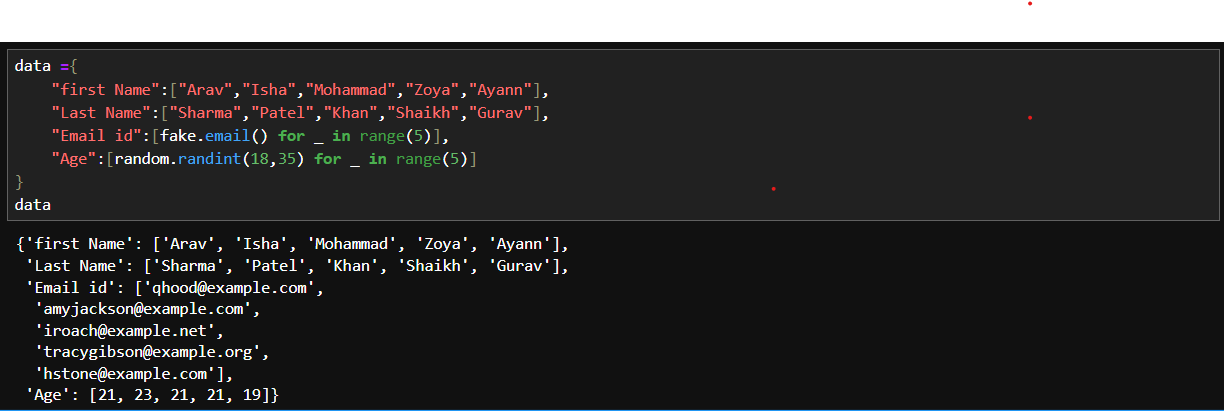
"Last Name":["Sharma","Patel","Khan","Shaikh","Gurav"],

"Email id":[fake.email() for \_ in range(5)],

"Age":[random.randint(18,35) for \_ in range(5)]

}

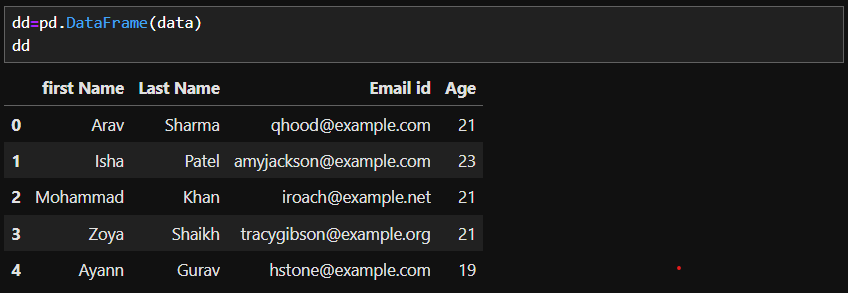
* data



Q.2.

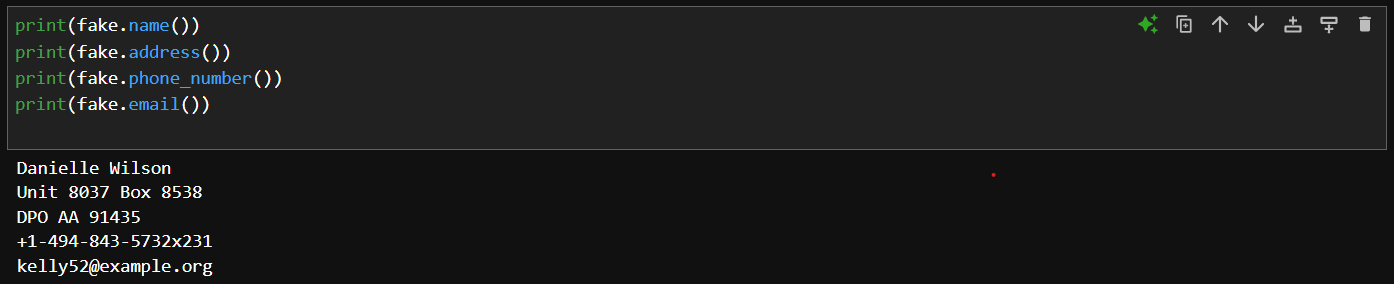
* dd=pd.DataFrame(data)

dd



Q.3. Display fake name, address, phone number, email

* print(fake.name())
* print(fake.address())
* print(fake.phone\_number())
* print(fake.email())



Q.4. display 6 fake data of name, address, email, date-of-birth, phone number, company name.

* info = []

for \_ in range(5):

record = {

"Name": fake.name(),

"Address": fake.address(),

"Date of Birth": fake.date\_of\_birth(),

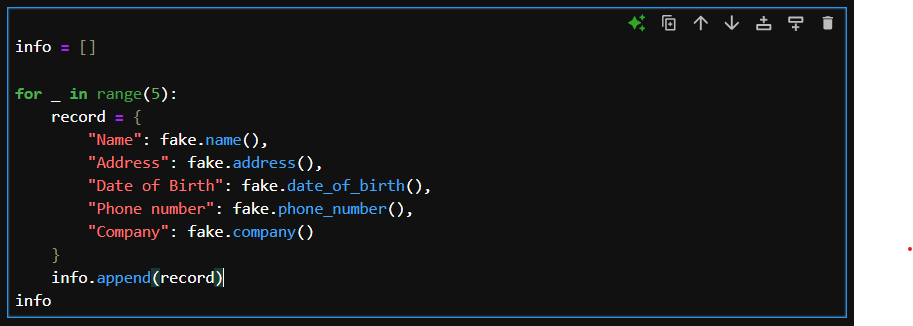
"Phone number": fake.phone\_number(),

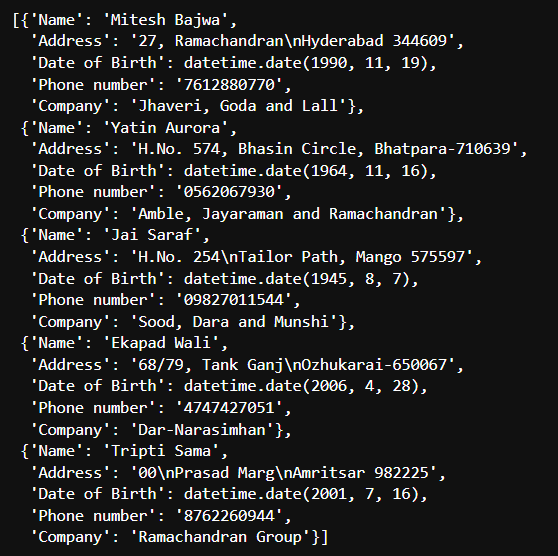
"Company": fake.company()

}

info.append(record)

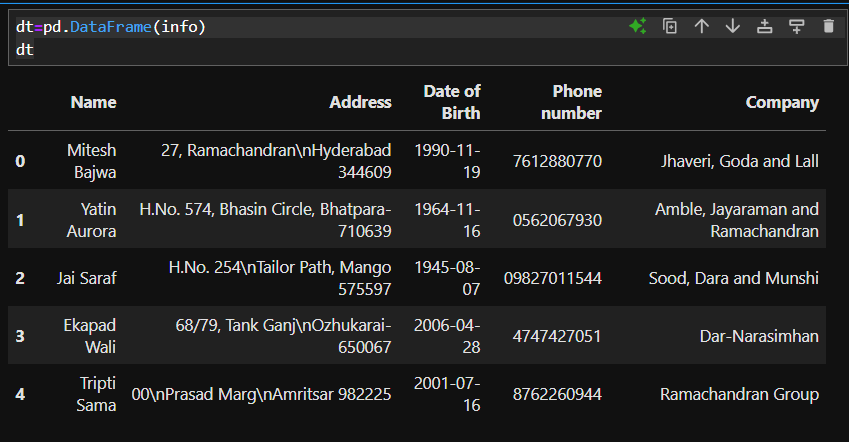
* info





Q.5. Converts the fake profile data into a structured Pandas DataFrame.

* dt=pd.DataFrame(info)
* dt



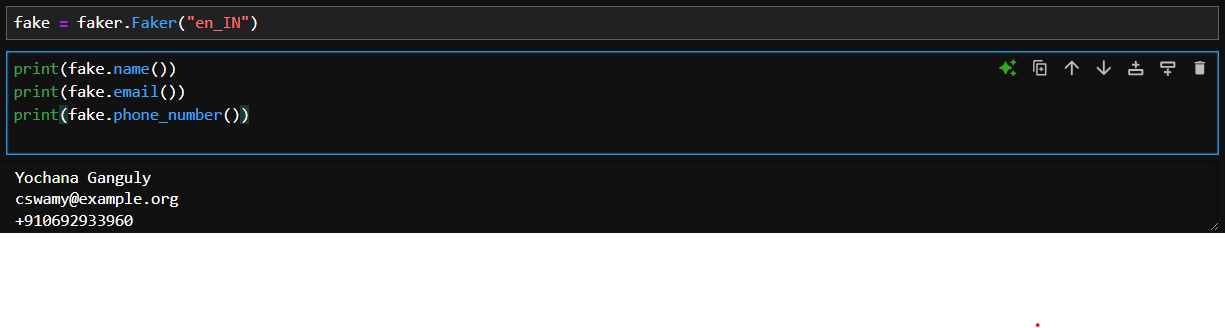
Q.6. To create fake data that contain indian name

* fake = faker.Faker("en\_IN")

print(fake.name())

print(fake.email())

print(fake.phone\_number())



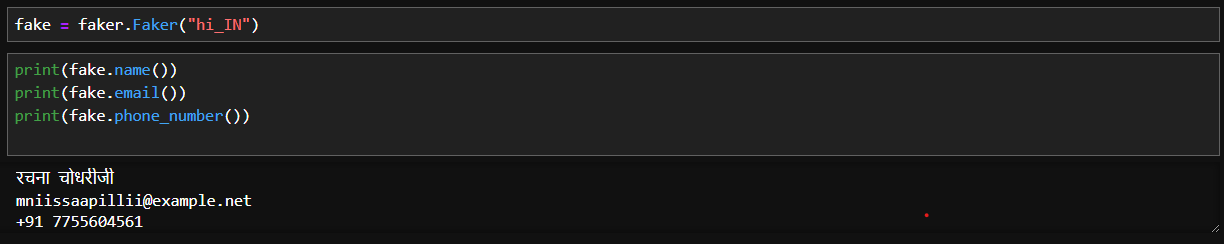
Q.7. To create fake data that contain indian (Hindi) name

* fake = faker.Faker("hi\_IN")

print(fake.name())

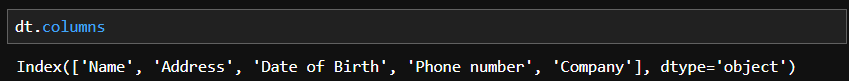
print(fake.email())

print(fake.phone\_number())



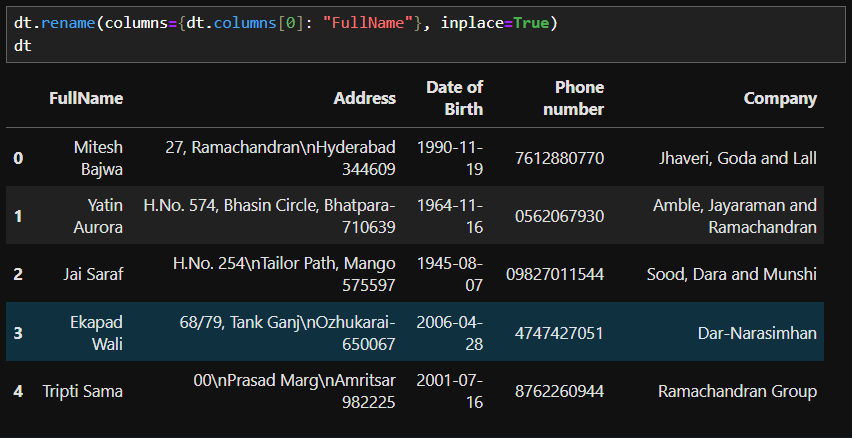
Q.8. Displaying column name

* dt.columns



Q.9. Command for replacing column name.

* dt.rename(columns={dt.columns[0]: "First Name"}, inplace=True)
* dt



Q.10.

* dt.rename(columns={

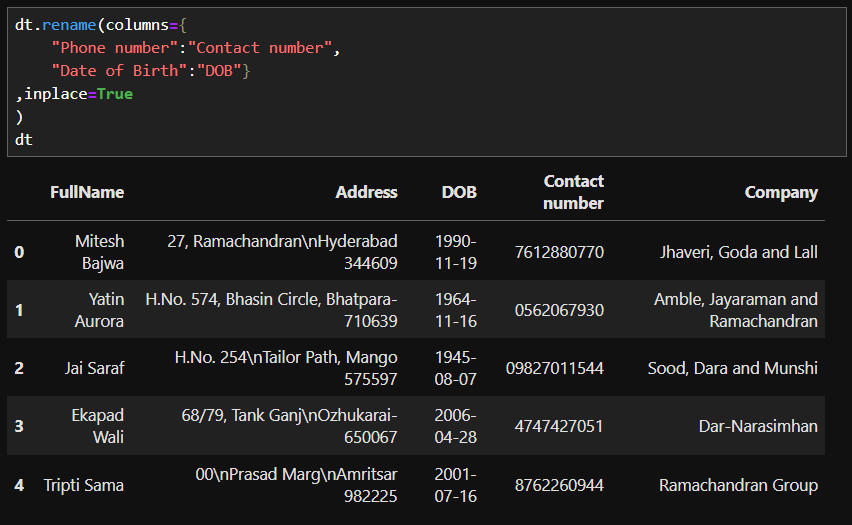
"Phone number":"Contact number",

"Date of Birth":"DOB"}

,inplace=True

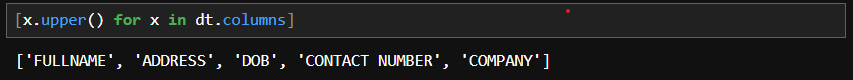
)

* dt



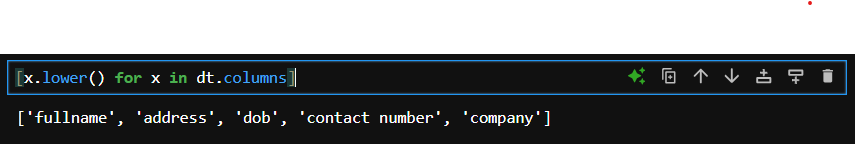
Q.11. Display all columns name in UPPER case

* [x.upper() for x in dt.columns]



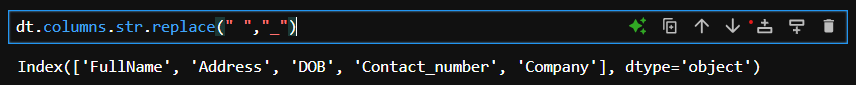
Q.12. Display all columns name in lower case

* [x.lower() for x in dt.columns]



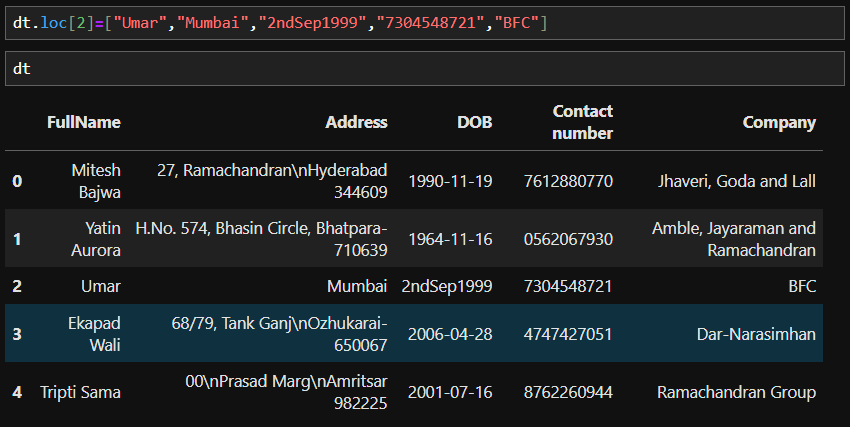
Q.13. Replace all with space with underscore “\_” and display the column

* dt.columns.str.replace(" ","\_")



Q.14. Replace the second column for the table with ne data

* dt.loc[2]=["Umar","Mumbai","2ndSep1999","7304548721","BFC"]
* dt



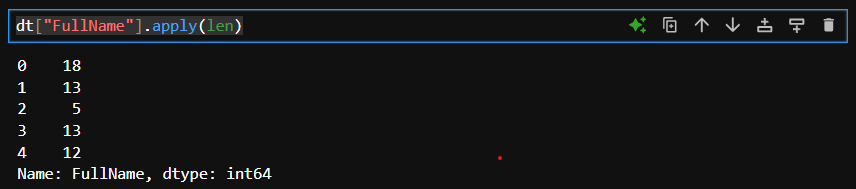
Q.15. Replace the name form column of fullname

* dt.loc[2,["FullName"]]=["Onkar"]
* dt



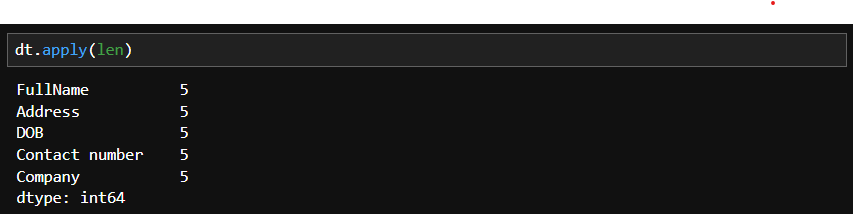
Q.16. Display the length of the column that in given table

* dt["FullName"].apply(len)



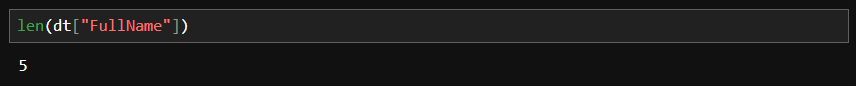
Q.17. Displaying the length of each column have

* dt.apply(len)



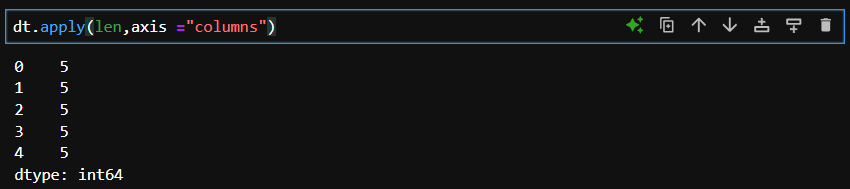
Q.18. Displaying the each column contain five value

* len(dt["FullName"])



Q.19.

* dt.apply(len,axis ="columns")

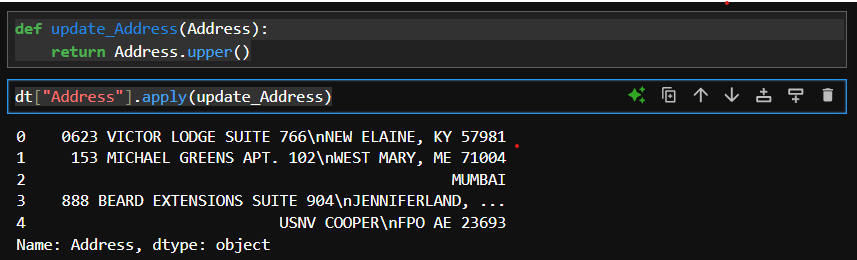


Q.20.

* def update\_Address(Address):

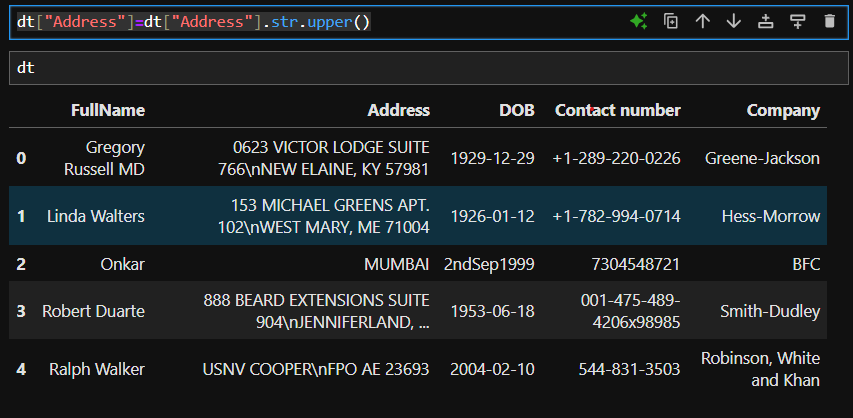
return Address.upper()

* dt["Address"].apply(update\_Address)



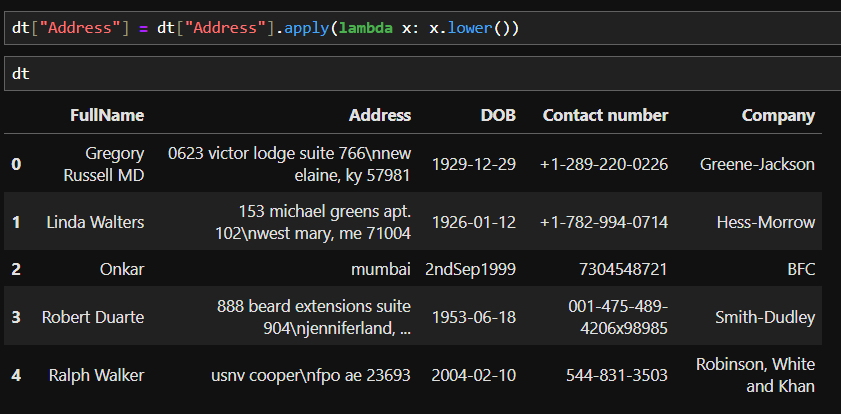
Q.21. Display the address column in upper case

* dt["Address"]=dt["Address"].str.upper()
* dt



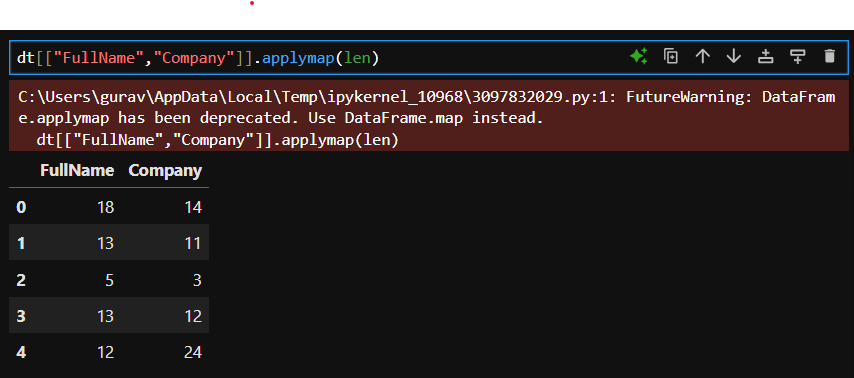
Q.22.

* dt["Address"] = dt["Address"].apply(lambda x: x.lower())
* dt



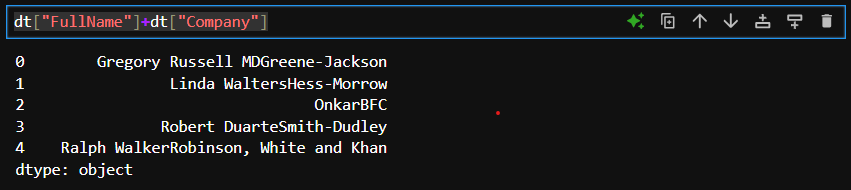
Q.23.

* dt[["FullName","Company"]].applymap(len)



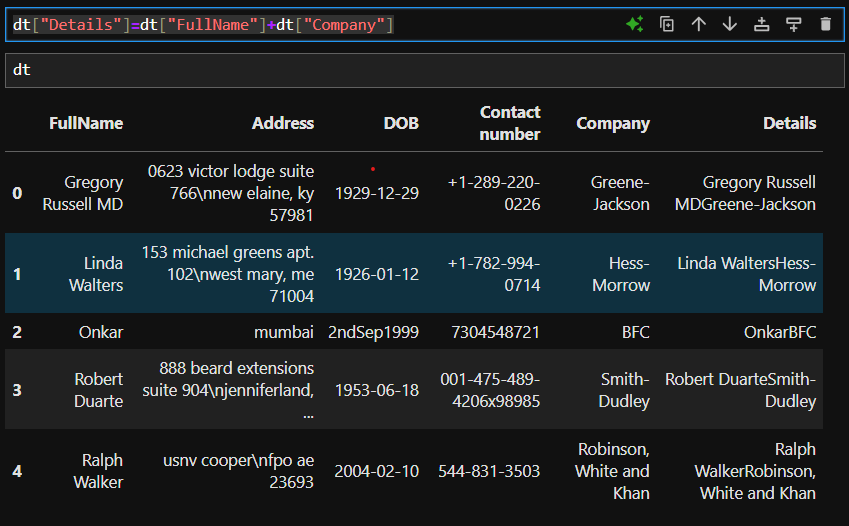
Q.24.

* dt["FullName"]+dt["Company"]



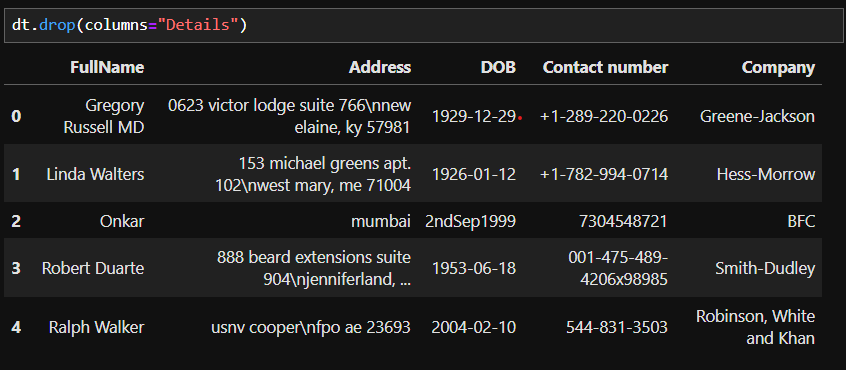
Q.25.

* dt["Details"]=dt["FullName"]+dt["Company"]
* dt



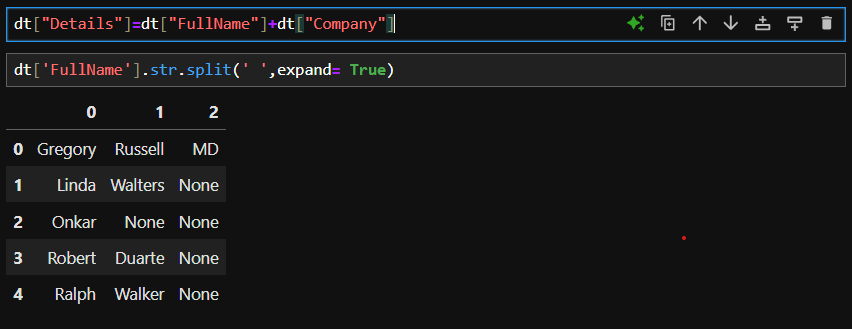
Q.26. Droping the Detail column form the given table

* dt.drop(columns="Details")



Q.27.

* dt["Details"]=dt["FullName"]+dt["Company"]
* dt['FullName'].str.split(' ',expand= True)



Q.28.

* import pandas as pd

import faker

fake = Faker('en\_IN')

information = []

for \_ in range(10):

record = {

"First Name": fake.first\_name(),

"Surname": fake.last\_name(),

"Age": fake.age(),

"City": fake.city(),

"Hobby": fake.word(), # Using random word as a placeholder for hobby

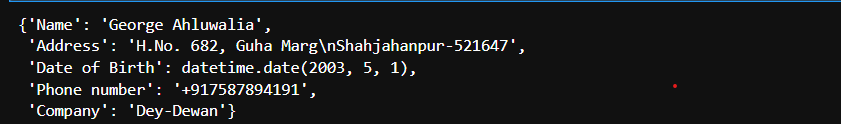
"Bank Name": fake.bank\_name(),

"Gender": fake.random\_element(elements=("Male", "Female"))

}

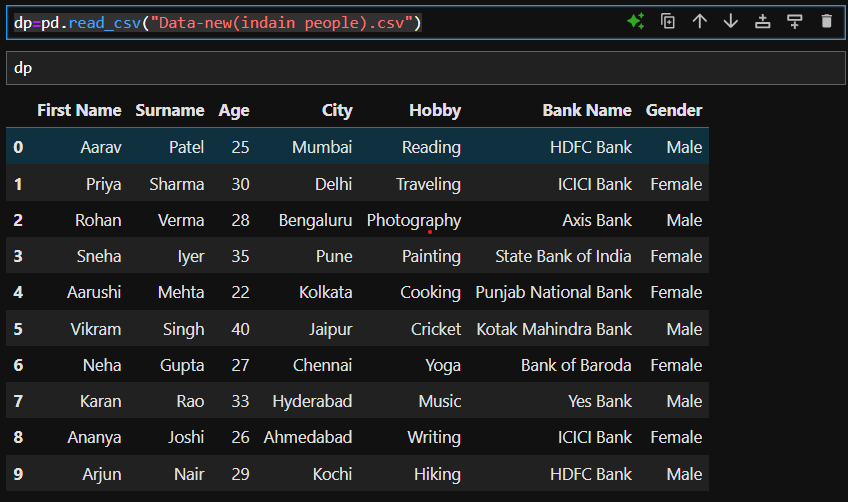
information.append(record)

* record



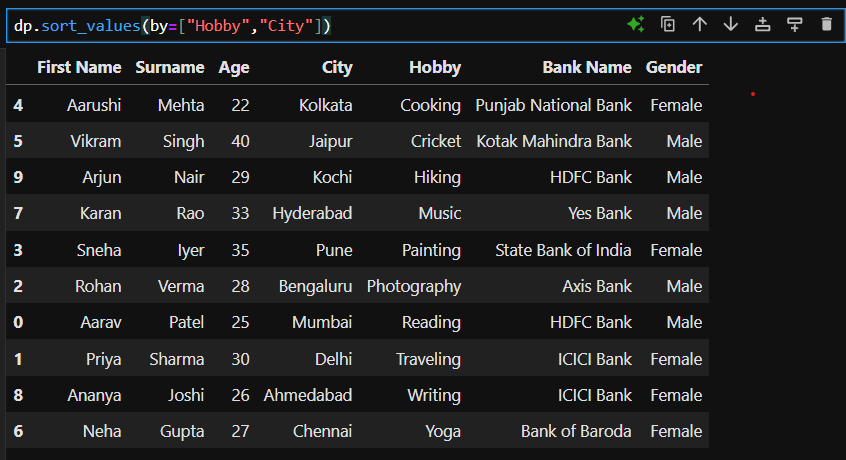
Q.29.

* dp=pd.read\_csv("Data-new(indain people).csv")
* dp



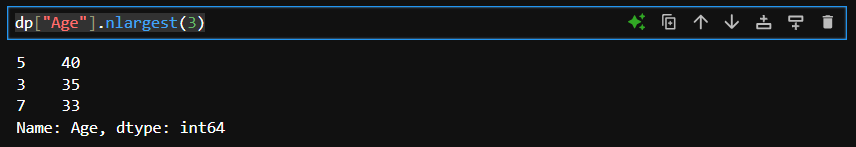
Q.30.

* dp.sort\_values(by=["Hobby","City"])



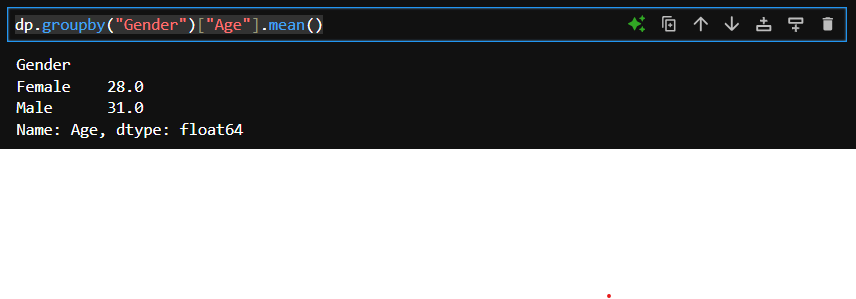
Q.31.

* dp["Age"].nlargest(3)



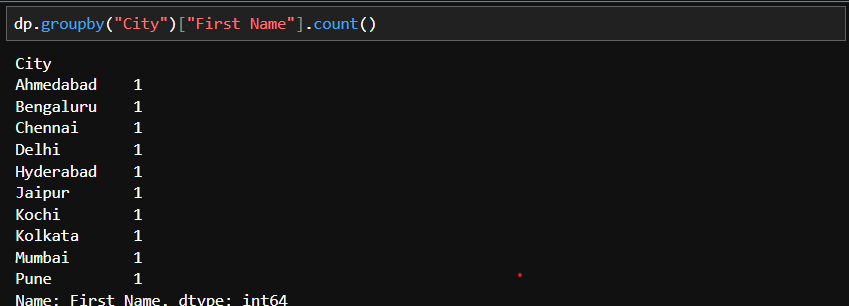
Q.32.

* dp.groupby("Gender")["Age"].mean()



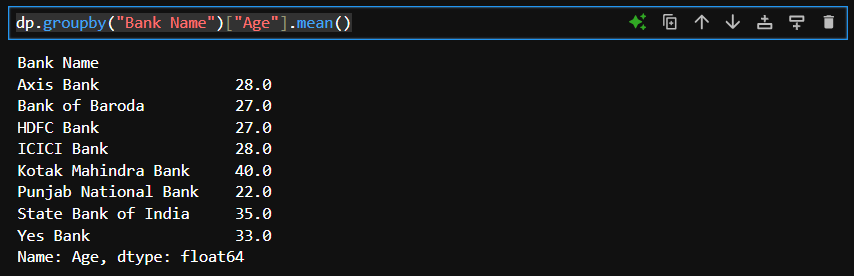
Q.33.

* dp.groupby("City")["First Name"].count()



Q.34.

* dp.groupby("Bank Name")["Age"].mean()



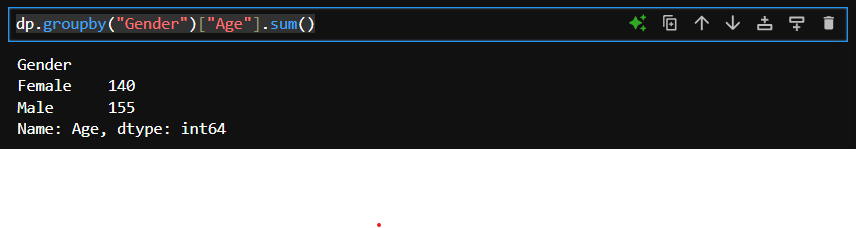
Q.35.

* dp.groupby(["Gender","Hobby"])["Age"].mean



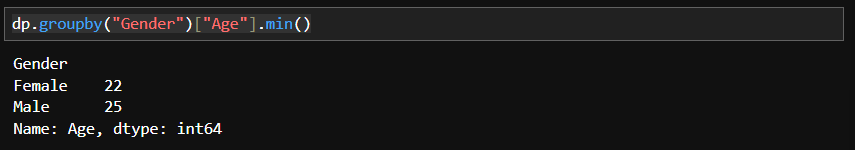
Q.36.

* dp.groupby("Gender")["Age"].sum()



Q.37.

* dp.groupby("Gender")["Age"].min()



Q.38.

* dp.groupby("Gender").agg({

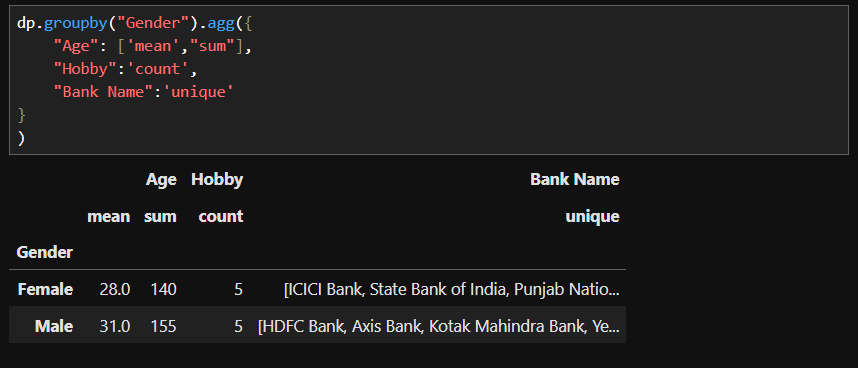
"Age": ['mean',"sum"],

"Hobby":'count',

"Bank Name":'unique'

}

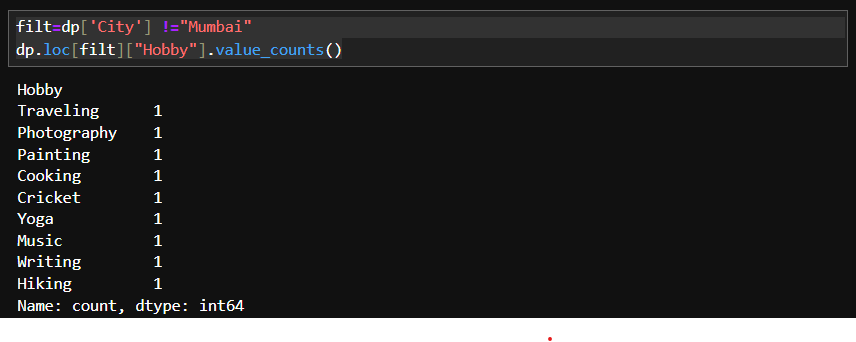
)



Q.39.

* filt=dp['City'] !="Mumbai"

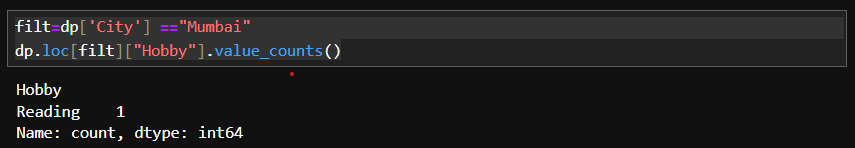
dp.loc[filt]["Hobby"].value\_counts()



Q.40.

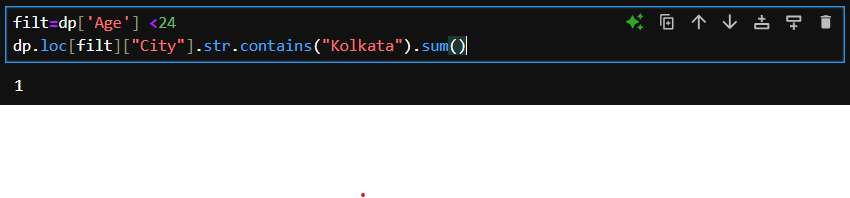
* filt=dp['City'] =="Mumbai"

dp.loc[filt]["Hobby"].value\_counts()



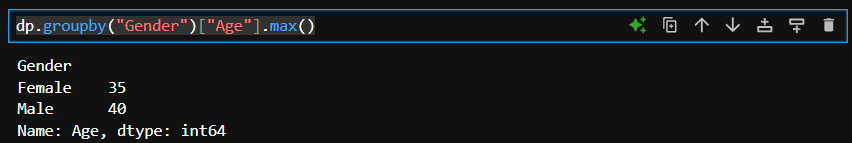
Q.41.

* filt=dp['Age'] <24
* dp.loc[filt]["City"].str.contains("Kolkata").sum



Q.42.

* dp.groupby("Gender")["Age"].max()



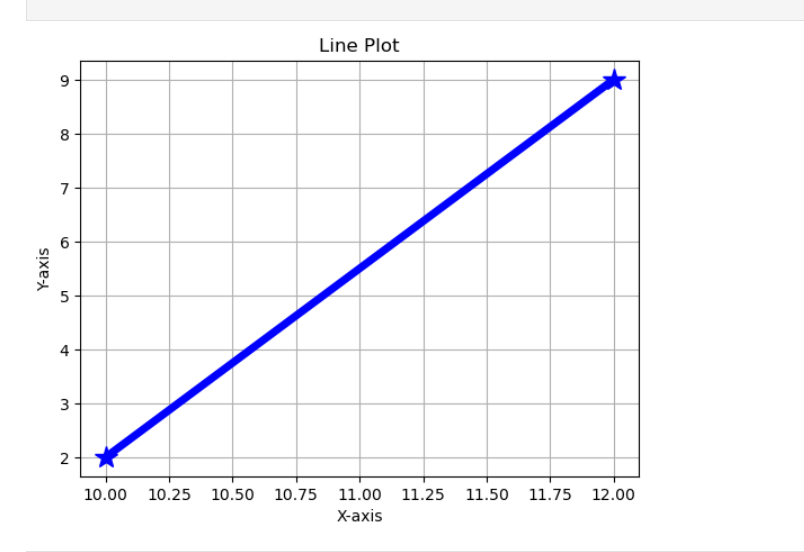
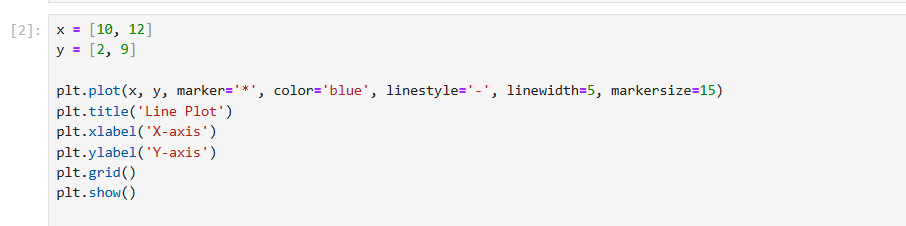
**DATA ANALYSIS**

Graph

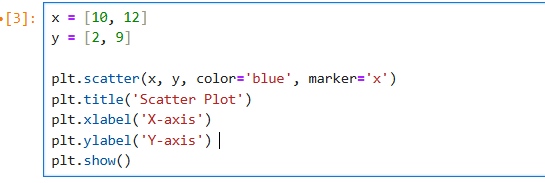
Q1. Import pandas for data manipulation and matplotlib.pyplot for plotting.

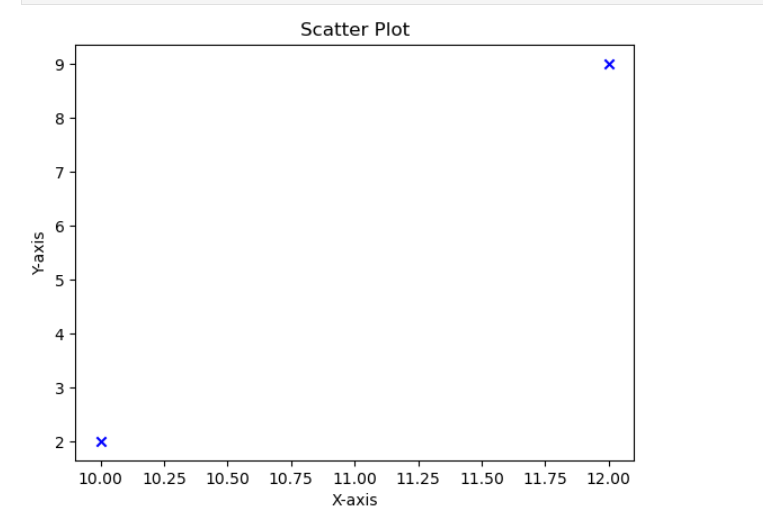


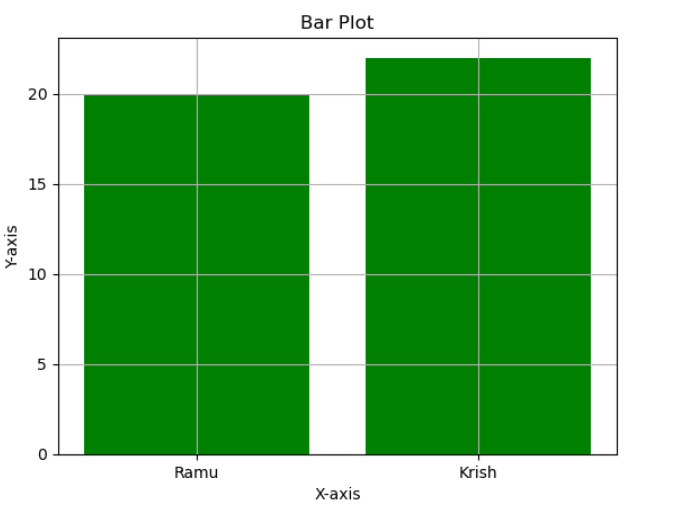
Q2. Plot a line graph and display it.



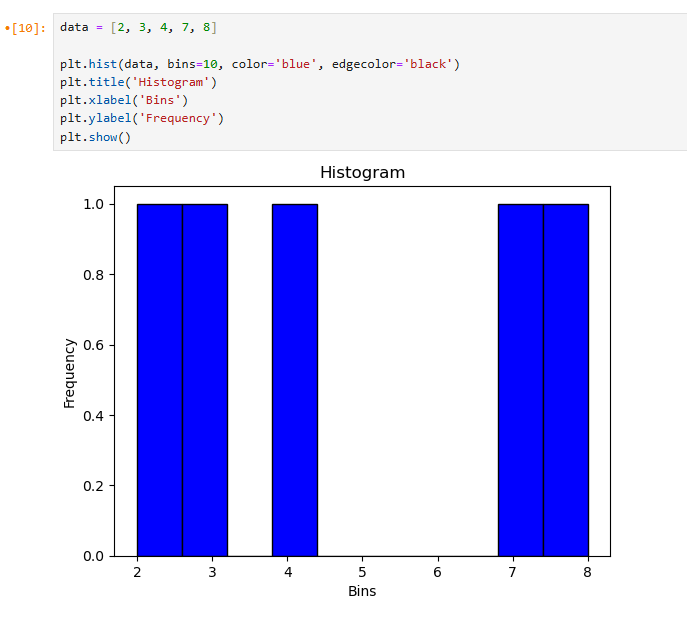
Q3. Create a scatter plot and display it.



Q4. Create a bar plot for Name and age then display it.



Q5. Create a red histogram with black edges and display it.



Q6. Create a pie chart for sizes with labels showing percentages and a title.

