Data Loading and Overview

1. Read a CSV file into a Pandas DataFrame
➢ Load data.csv and display the first 5 rows.

```
import pandas as pd

df = pd.read_csv('data 3.csv')
print(df)
print("First 5 rows are:")
print(df.head(5))
```

o/p:

```
      Name  Age  Salary       City Technology
0     Abhi   21    2000  Bangalore        c++
1   Ananya   22    3000    Chennai          C
2  Bhumika   25    4000        Goa     Python
3     Esha   26    5000  Hyderabad       Java
4  Bhargav   25    4000     Mumbai       .net
5  Prateek   28    3000     Mysore  Fullstack
6    Abhay   19    6000       Pune   Frontend
7  Karthik   23    8000      Delhi      React
8    Vibha   27    3500  Mangalore       Html
```

```
First 5 rows are:
      Name  Age  Salary       City Technology
0     Abhi   21    2000  Bangalore        c++
1   Ananya   22    3000    Chennai          C
2  Bhumika   25    4000        Goa     Python
3     Esha   26    5000  Hyderabad       Java
4  Bhargav   25    4000     Mumbai       .net
```

2. Check the shape of a DataFrame

➢ How many rows and columns are present?

```
import pandas as pd
df = pd.read_csv("data 3.csv")
print("Shape of the DataFrame is:")
print(df.shape)
```

o/p:Shape of the DataFrame is:
(9, 5)

3. Get summary statistics
➢ Use a method to get min, max, mean, etc., for numeric columns.

```
import pandas as pd
df = pd.read_csv("data 3.csv")
print("Summary statistics is:")
print(df.describe())
```

Summary statistics is:

	Age	Salary
count	9.00000	9.000000
mean	24.00000	4277.777778
std	2.95804	1821.934259
min	19.00000	2000.000000
25%	22.00000	3000.000000
50%	25.00000	4000.000000
75%	26.00000	5000.000000
max	28.00000	8000.000000

Data Selection and Filtering

4. Select a single column
➢ Extract the "Age" column as a Series.

```
import pandas as pd
df = pd.read_excel('data a1.xlsx')
print(df)
```

```
# Extract Age column
age = df["Age"]
print("Age column:")
print(age)
```

o/p:

	Name	Age	Salary	Department
0	Abhi	30	2000	Developer
1	Ananya	31	3000	HR
2	Bhumika	34	4000	Trainee
3	Esha	32	5000	Trainee
4	Bhargav	25	4000	HR
5	Prateek	39	3000	Trainee
6	Abhay	23	6000	Developer
7	Karthik	35	8000	Trainee
8	Vibha	27	3500	HR

Age column:

0 30
1 31
2 34
3 32
4 25
5 39
6 23
7 35
8 27

Name: Age, dtype: int64

5. Filter rows based on condition

➢ Show rows where "Salary" > 50000.

```
import pandas as pd
df = pd.read_excel('data a1.xlsx')
salary = df[df["Salary"] > 5000]
print("\nEmployees with Salary > 5000:")
print(salary)
```

o/p:Employees with Salary > 5000:

```
      Name  Age  Salary Department
6    Abhay   23    6000  Developer
7  Karthik   35    8000    Trainee
```

6. Filter multiple conditions

➢ Display rows where "Department" == 'HR' and "Age" > 30.

```
import pandas as pd
df = pd.read_excel('data a1.xlsx')
hr = df[(df["Department"] == "HR") & (df["Age"] > 30)]
print("\nHR with Age > 30 are:")
print(hr)
```

o/p:

```
HR with Age > 30 are:
    Name  Age  Salary Department
1  Ananya   31    3000         HR
```

Data Cleaning

7. Check for missing values

➢ Find which columns have NaN values and how many.

```
import pandas as pd

df = pd.read_excel('dataa.xlsx')
print(df)
missing_values = df.isnull().sum()
print("Missing values :")
print(missing_values)
```

o/p:

```
     Name   Age  Salary       City
```

```
0     Abhi  21.0  2000.0  Bangalore
1   Ananya  22.0  3000.0        NaN
2  Bhumika  25.0  4000.0        Goa
3     Esha   NaN     NaN        NaN
4  Bhargav  25.0  4000.0     Mumbai
5  Karthik  23.0  8000.0      Delhi
6    Vibha  27.0  3500.0  Mangalore
Missing values :
Name      0
Age       1
Salary    1
City      2
dtype: int64
```

8. Replace missing values

➤ Fill NaN values in "Salary" with 0.

```python
import pandas as pd

df = pd.read_excel('dataa.xlsx')
df['Salary'] = df['Salary'].fillna(0)

print("Updated Salary column with 0:")
print(df[['Name', 'Salary']])
```

o/p:

```
Updated Salary column with 0:
      Name  Salary
0     Abhi  2000.0
1   Ananya  3000.0
2  Bhumika  4000.0
3     Esha     0.0
4  Bhargav  4000.0
5  Karthik  8000.0
6    Vibha  3500.0
```

9. Remove duplicate rows

➢ Drop duplicates and reset the index.

```
import pandas as pd

df = pd.read_excel('dataa.xlsx')
print(df)
df_clean = df.drop_duplicates().reset_index(drop=True)
print("updated data")
print(df_clean)
```

o/p:

```
     Name   Age  Salary       City
0     Abhi  21.0  2000.0  Bangalore
1   Ananya  22.0  3000.0        NaN
2  Bhumika  25.0  4000.0        Goa
3     Esha   NaN     NaN        NaN
4  Bhargav  25.0  4000.0     Mumbai
5  Karthik  23.0  8000.0      Delhi
6     Abhi  21.0  2000.0  Bangalore

updated data
     Name   Age  Salary       City
0     Abhi  21.0  2000.0  Bangalore
1   Ananya  22.0  3000.0        NaN
2  Bhumika  25.0  4000.0        Goa
3     Esha   NaN     NaN        NaN
4  Bhargav  25.0  4000.0     Mumbai
5  Karthik  23.0  8000.0      Delhi
```

Data Aggregation and Sorting

10. Sort the DataFrame by a column

➢ Sort rows by "Age" in descending order.

```
import pandas as pd
df = pd.read_excel('data a1.xlsx')
sorted_df = df.sort_values(by="Age", ascending=False)
print("sorted data of Age in descending order:")
print(sorted_df)
```

o/p:sorted data of Age in descending order:

	Name	Age	Salary	Department
5	Prateek	39	3000	Trainee
7	Karthik	35	8000	Trainee
2	Bhumika	34	4000	Trainee
3	Esha	32	5000	Trainee
1	Ananya	31	3000	HR
0	Abhi	30	2000	Developer
8	Vibha	27	3500	HR
4	Bhargav	25	4000	HR
6	Abhay	23	6000	Developer

11. Group by and aggregate

➢ Group by "Department" and find the average "Salary".

```
import pandas as pd
df = pd.read_excel('data a1.xlsx')
avg_salary = df.groupby("Department")["Salary"].mean()
print("Average Salary by Department is:")
print(avg_salary)
```

o/p:

Average Salary by Department is:

Department

Developer 4000.0

HR 3500.0

Trainee 5000.0

12. Count unique values

➢ How many unique departments are there in the "Department" column?

```
import pandas as pd
df = pd.read_excel('data a1.xlsx')
unique = df["Department"].nunique()
print("Number of unique departments:",unique)
```

o/p:

Number of unique departments: 3

Q1.create a table with columns and rows

column name : students,technology,marks

add the column teacher

and delete the technology later

and try to to practice on all the example methods

```
var={'students':("abhi","bhumi","esha","bhargs","anu","vibha
","harini","joshi","prateek","abhay","hemanth","rashmi","shr
eya","ananya","aditya"),
'technology':("java","python","c++","maths","english","kanna
da","sql","react","aiml",".net","c","automation","testing","
fullstack","frontend"),
'marks':(15,20,55,90,80,70,60,45,50,40,20,95,65,77,55)}
df=pd.DataFrame(var)
print(df)

df['teacher']=['vidhya','eshalaksmi','anvitha','abhijith','d
ivya','lisha','roopa','nithya','manisha','paul','john','amit
h','rohan','rahul','dev']
print(df['teacher'])
df.drop("technology",axis=1,inplace=True)
print(df)
print(df.describe())
print(df.head(5))
print(df.tail(5))
print(df.shape)
print(df.dtypes)
print(df.columns)
```

O/P:Name: teacher, dtype: object

	students	marks	teacher
0	abhi	15	vidhya
1	bhumi	20	eshalaksmi
2	esha	55	anvitha
3	bhargs	90	abhijith

4 anu 80 divya
5 vibha 70 lisha
6 harini 60 roopa
7 joshi 45 nithya
8 prateek 50 manisha
9 abhay 40 paul
10 hemanth 20 john
11 rashmi 95 amith
12 shreya 65 rohan
13 ananya 77 rahul
14 aditya 55 dev
marks
count 15.000000
mean 55.800000
std 24.891766
min 15.000000
25% 42.500000
50% 55.000000
75% 73.500000
max 95.000000
students marks teacher
0 abhi 15 vidhya
1 bhumi 20 eshalaksmi
2 esha 55 anvitha
3 bhargs 90 abhijith
4 anu 80 divya
students marks teacher
10 hemanth 20 john
11 rashmi 95 amith
12 shreya 65 rohan
13 ananya 77 rahul
14 aditya 55 dev
(15, 3)
students object
marks int64
teacher object

```
dtype: object
Index(['students', 'marks', 'teacher'], dtype='object')
```