Program	B.Pharmacy
Semester	First
CourseTitle /Subject Name	Basics of Python Programming for Pharmaceutical Sciences (Theory)
Course Code	BP101T
Module	Data Handling with Pandas
Course Coordinator	Ms. Ramanjit Kaur
Mobile no	8968729307

Course Objectives:

The objectives of this course are to:

1. Introduce the fundamentals of python programming for pharmaceutical sciences.
2. Develop basic programming skills using control structures, functions ,and data structures.
3. Provide knowledge of data handling techniques for structured dataset management.
4. Familiarize students with data analysis tools such as NumPy and Pandas for healthcare datasets.
5. Enable students to visualize and interpret pharmaceutical data.

Course Outcomes (CO):

CO No.	Upon successful completion of this course, the students will be able to:
BP101.1	Explain the fundamentals of Python programming, including variables, data types, operators, and libraries.
BP101.2	Analyze program logic using control structures and functions.
BP101.3	Organize, manipulate, and retrieve data using data structures and file handling techniques.
BP101.4	Analyze pharmaceutical datasets using Python libraries.
BP101.5	Visualize and interpret pharmaceutical data using graphical tools.

Learning Outcomes:

After completing this unit, students will be able to:

LO	Learning Outcomes	CO
1	Understand the Pandas library	BP101.4
2	Create and use Pandas Series and Data Frame	BP101.4
3	Read and import CSV and Excel files	BP101.4
4	Inspect and summarize datasets	BP101.4
5	Perform data cleaning operations	BP101.4
6	Filter, select, and extract relevant data	BP101.4
7	Group and aggregate data	BP101.4
8	Apply Pandas techniques to pharmaceutical datasets	BP101.4
9	Interpret processed data	BP101.4
10	Develop practical skills	BP101.4

Data Handling with Pandas

What is Pandas?

Pandas is a powerful open-source Python library used for **data analysis and data manipulation**. It provides easy-to-use data structures and functions for working with structured data such as tables, spreadsheets, and CSV files.

Pandas is widely used in:

- Data analysis
- Data cleaning
- Data visualization
- Machine learning
- Healthcare and pharmaceutical data analysis

Why Use Pandas?

- Handles large datasets efficiently
- Reads and writes data from various file formats (CSV, Excel, JSON, etc.)
- Performs data cleaning and transformation
- Supports statistical analysis
- Integrates well with NumPy, Matplotlib, and Scikit-learn

Example: Creating a Series

- A **Series** is a one-dimensional array-like structure.

```
</>python
import pandas as pd
numbers = pd.Series([10, 20, 30, 40, 50])
print(numbers)
```

Output

```
0   10
1   20
2   30
3   40
4   50
dtype: int64
```

Pandas Series and DataFrame Structures

1. Pandas Series

Definition

A Series is a one-dimensional labeled array capable of holding data of any type such as integers, floats, strings, or booleans. Each element in a Series has an associated index.

Syntax

```
</>pyhton
import pandas as pd
series_name = pd.Series(data)
```

Example

```
</>pyhton
import pandas as pd
marks = pd.Series([85, 90, 78, 92])
print(marks)
```

Output

```
0    85
1    90
2    78
3    92
dtype: int64
```

Series with Custom Index

```
</>python
import pandas as pd
marks = pd.Series([85, 90, 78], index=["A", "B", "C"])
print(marks)
```

Output

```
A    85
B    90
C    78
dtype: int64
```

Characteristics of Series

- One-dimensional data structure.
- Stores a single column of data.
- Has labels called indexes.
- Can store different data types.

---

2. Pandas DataFrame

Definition

A DataFrame is a two-dimensional labeled data structure consisting of rows and columns. It is similar to a spreadsheet or database table.

Syntax

```
</>python
import pandas as pd
df = pd.DataFrame(data)
```

Example

```
</>python
import pandas as pd
student_data = {
   "Name": ["Aman", "Riya", "John"],
   "Age": [20, 21, 22],
   "Marks": [85, 90, 88]
}
df = pd.DataFrame(student_data)
print(df)
```

Output

```
   Name  Age  Marks
0  Aman   20     85
1  Riya   21     90
2  John   22     88
```

Characteristics of DataFrame

- Two-dimensional structure.
- Contains rows and columns.
- Different columns can have different data types.
- Easy to filter, sort, and analyze data.
- Most commonly used Pandas object.

Difference between Series and DataFrame

Feature	Series	DataFrame
Dimension	One-dimensional	Two-dimensional
Structure	Single column	Multiple columns
Index	Yes	Rows and columns both have indexes
Data Type	One data column	Multiple data columns
Example	Marks of students	Complete student record

Reading CSV and Excel Files – PK Study Datasets and ADR Reports

Introduction

In pharmaceutical and healthcare data analysis, Pharmacokinetic (PK) study datasets and Adverse Drug Reaction (ADR) reports are often stored in CSV or Excel files. The Pandas library provides simple functions to read, analyze, and manipulate these datasets.

1. Reading a CSV File

A CSV (Comma-Separated Values) file stores data in tabular form.

Syntax

```
</>python
import pandas as pd
df = pd.read_csv("filename.csv")
```

Example: Reading a PK Study Dataset

Suppose a file named **pk_study.csv** contains:

Patient_ID	Drug	Concentration
P001	Drug A	12.5
P002	Drug A	15.2
P003	Drug B	10.8

Python Code

```
</python>
import pandas as pd
pk_data = pd.read_csv("pk_study.csv")
print(pk_data)
```

Output

```
Patient_ID      Drug      Concentration
0          P001  Drug A          12.5
1          P002  Drug A          15.2
2          P003  Drug B          10.8
```

2. Reading an Excel File

Excel files usually have the extension .xlsx.

Syntax

```
</python>
import pandas as pd
df = pd.read_excel("filename.xlsx")
```

Example: Reading an ADR Report

Suppose **adr_report.xlsx** contains:

Patient_ID	Drug	ADR
P101	Aspirin	Nausea
P102	Penicillin	Rash
P103	Ibuprofen	Headache

Python Code

```
</>python
import pandas as pd
adr_data = pd.read_excel("adr_report.xlsx")
print(adr_data)
```

Output

	Patient_ID	Drug	ADR
0	P101	Aspirin	Nausea
1	P102	Penicillin	Rash
2	P103	Ibuprofen	Headache

1. **Viewing Dataset Information**

```
</>python
 print(pk_data.info())
```

Output

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3 entries, 0 to 2
Data columns (total 3 columns):
```

This provides information about:

- Number of rows and columns
- Data types
- Missing values

4. Displaying First Five Records

```
</>python
print(pk_data.head())
```

Output

Patient_ID	Drug	Concentration
0	P001 Drug A	12.5
1	P002 Drug A	15.2
2	P003 Drug B	10.8

5. Basic Analysis of PK Dataset

Calculate average drug concentration:

```
</>python
average = pk_data["Concentration"].mean()
print("Average Concentration:", average)
```

Output

```
Average Concentration: 12.83
```

6. Basic Analysis of ADR Reports

Count the number of ADR cases:

```
</>python
print(adr_data["ADR"].value_counts())
```

Output

```
Nausea      1
Rash        1
Headache    1
```

Applications in Pharmaceutical Studies

PK Study Datasets

- Drug concentration analysis
- Bioavailability studies
- Pharmacokinetic parameter calculations
- Clinical trial data management

ADR Reports

- Monitoring drug safety
- Identifying adverse reactions
- Pharmacovigilance studies
- Regulatory reporting

Inspecting datasets using functions such as head(), tail(), info(), and describe().

Introduction

After loading a dataset into a Pandas DataFrame, it is important to inspect and understand the data before performing analysis. Pandas provides several useful functions for exploring datasets quickly:

- head() – View the first few rows
- tail() – View the last few rows
- info() – Get information about the dataset
- describe() – Generate statistical summaries

Sample Dataset

```
</>python
import pandas as pd
data = {
   "Patient_ID": ["P001", "P002", "P003", "P004", "P005"],
   "Age": [25, 30, 28, 35, 40],
   "Weight": [60, 72, 68, 75, 80],
   "Drug_Concentration": [12.5, 15.2, 10.8, 18.0, 16.4]
}
df = pd.DataFrame(data)
print(df)
```

Output

```
  Patient_ID  Age  Weight  Drug_Concentration
0       P001   25      60                12.5
1       P002   30      72                15.2
2       P003   28      68                10.8
3       P004   35      75                18.0
4       P005   40      80                16.4
```

1. head() Function

Definition

The head() function displays the first five rows of a dataset by default.

Syntax

```
</>python
df.head()
```

Example

```
</>python
print(df.head())
```

Output

	Patient_ID	Age	Weight	Drug_Concentration
0	P001	25	60	12.5
1	P002	30	72	15.2
2	P003	28	68	10.8
3	P004	35	75	18.0
4	P005	40	80	16.4

Display Specific Number of Rows

```
</>python
print(df.head(3))
```

Output

	Patient_ID	Age	Weight	Drug_Concentration
0	P001	25	60	12.5
1	P002	30	72	15.2
2	P003	28	68	10.8

2. tail() Function

Definition

The tail() function displays the last five rows of a dataset by default.

Syntax

```
</>python
   df.tail()
```

Example

```
</>python
print(df.tail())
```

Output

	Patient_ID	Age	Weight	Drug_Concentration
0	P001	25	60	12.5
1	P002	30	72	15.2
2	P003	28	68	10.8
3	P004	35	75	18.0
4	P005	40	80	16.4

Display Last Three Rows

```
</>python
print(df.tail(3))
```

Output

	Patient_ID	Age	Weight	Drug_Concentration
2	P003	28	68	10.8
3	P004	35	75	18.0
4	P005	40	80	16.4

3. info() Function

Definition

The info() function provides a concise summary of the DataFrame.

Syntax

```
</>python
df.info()
```

Example

```
</>python
print(df.info())
```

Output

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5 entries, 0 to 4
Data columns (total 4 columns):
   Column            Non-Null Count  Dtype

 0         Patient_ID          5 non-null      object
 1         Age                 5 non-null      int64
 2         Weight               5 non-null     int64
 3   Drug_Concentration  5 non-null      float64
```

Information Provided by info()

- Number of rows and columns
- Column names
- Data types
- Non-null values
- Memory usage

4. describe() Function

Definition

The describe() function generates statistical summaries for numerical columns.

Syntax

```
</>python
df.describe()
```

Example

```
</>python
print(df.describe())
```

Output

	Age	Weight	Drug_Concentration
count	5.000000	5.000000	5.000000
mean	31.600000	71.000000	14.580000
std	5.941381	7.382412	2.857971
min	25.000000	60.000000	10.800000
25%	28.000000	68.000000	12.500000
50%	30.000000	72.000000	15.200000
75%	35.000000	75.000000	16.400000
max	40.000000	80.000000	18.000000

Statistics Provided by describe()

- **count** – Number of values
- **mean** – Average value
- **std** – Standard deviation
- **min** – Minimum value
- **25%** – First quartile
- **50%** – Median
- **75%** – Third quartile
- **max** – Maximum value

Data Cleaning Techniques and Handling Missing Values in Pandas

Introduction

Data cleaning is the process of identifying and correcting errors, inconsistencies, and missing values in a dataset. Clean data improves the accuracy and reliability of data analysis and machine learning models.

In pharmaceutical and healthcare datasets, missing or incorrect data can occur due to:

- Incomplete patient records
- Data entry errors
- Equipment malfunction
- Lost clinical trial information

Common Data Cleaning Techniques

1. Identifying Missing Values

Pandas provides the isnull() function to detect missing values.

Example

```
</>python
import pandas as pd
data = {
   "Patient_ID": ["P001", "P002", "P003"],
   "Age": [25, None, 30],
   "Weight": [60, 70, None]
}
df = pd.DataFrame(data)
print(df.isnull())
```

Output

```
   Patient_ID     Age    Weight
0      False     False   False
1      False      True   False
2      False     False    True
```

2. Counting Missing Values

Use isnull().sum() to count missing values in each column.

```
</>python
print(df.isnull().sum())
```

Output

```
Patient_ID      0
Age             1
Weight          1
dtype: int64
```

3. Removing Missing Values

Using dropna()

Rows containing missing values can be removed.

```
</>python
clean_df = df.dropna()
print(clean_df)
```

Output

```
  Patient_ID        Age       Weight
0       P001        25.0       60.0
```

4. Replacing Missing Values

Fill with a Specific Value

```
</>python
df.fillna(0, inplace=True)
print(df)
```

Output

```
  Patient_ID        Age       Weight
0       P001        25.0       60.0
1       P002        0.0        70.0
2       P003        30.0        0.0
```

5. Replacing Missing Values with Mean

This method is commonly used for numerical data.

```
</>python
df["Age"].fillna(df["Age"].mean(), inplace=True)
```

Example

```
import pandas as pd
data = {
   "Age": [25, None, 30, 35]
}
df = pd.DataFrame(data)
df["Age"].fillna(df["Age"].mean(), inplace=True)
print(df)
```

Output

```
    Age
0  25.0
1  30.0
2  30.0
3  35.0
```

6. Removing Duplicate Records

Example

```
</>python
import pandas as pd

data = {
   "Drug": ["Paracetamol", "Aspirin", "Aspirin"],
   "Price": [25, 30, 30]
}
df = pd.DataFrame(data)
df = df.drop_duplicates()
print(df)
```

Output

```
        Drug          Price
0  Paracetamol        25
1      Aspirin        30
```

7. Renaming Columns

```
</>python
df.rename(columns={"Price": "Drug_Price"}, inplace=True)
```

Output

```
        Drug          Drug_Price
0   Paracetamol         25
1      Aspirin         30
```

Pharmaceutical Dataset Example

Before Cleaning

```
</>python
import pandas as pd
data = {
   "Patient_ID": ["P001", "P002", "P002"],
   "Drug_Concentration": [12.5, None, None],
   "Age": [25, 30, 30]
}
df = pd.DataFrame(data)
print(df)
```

Output

	Patient_ID	Drug_Concentration	Age
0	P001	12.5	25
1	P002	NaN	30
2	P002	NaN	30

Cleaning Steps

```
</>python
 Fill missing values with mean
df["Drug_Concentration"].fillna(
    df["Drug_Concentration"].mean(),
    inplace=True
)
 Remove duplicates
df = df.drop_duplicates()
print(df)
```

Output

	Patient_ID	Drug_Concentration	Age
0	P001	12.5	25
1	P002	12.5	30

Filtering and Selecting Data Based on Conditions in Pandas

Introduction

Filtering and selecting data are important operations in Pandas that allow users to extract specific rows and columns from a dataset based on certain conditions. This helps in analyzing only the relevant data.

Sample Dataset

```
</>python
import pandas as pd
data = {
    "Patient_ID": ["P001", "P002", "P003", "P004"],
    "Age": [25, 35, 28, 40],
    "Drug": ["Aspirin", "Paracetamol", "Ibuprofen", "Aspirin"],
    "Concentration": [12.5, 18.0, 10.8, 20.5]
}
df = pd.DataFrame(data)
print(df)
```

Output

	Patient_ID	Age	Drug	Concentration
0	P001	25	Aspirin	12.5
1	P002	35	Paracetamol	18.0
2	P003	28	Ibuprofen	10.8
3	P004	40	Aspirin	20.5

1. Selecting a Single Column

Example

```
</>python
print(df["Drug"])
```

Output

```
0       Aspirin
1    Paracetamol
2     Ibuprofen
3       Aspirin
```

2. Selecting Multiple Columns

Example

```
</>python
print(df[["Patient_ID", "Drug"]])
```

Output

	Patient_ID	Drug
0	P001	Aspirin
1	P002	Paracetamol
2	P003	Ibuprofen
3	P004	Aspirin

3. Filtering Rows Based on One Condition

Example: Age Greater Than 30

```
</>pytrhon
filtered_data = df[df["Age"] > 30]
print(filtered_data)
```

Output

	Patient_ID	Age	Drug	Concentration
1	P002	35	Paracetamol	18.0
3	P004	40	Aspirin	20.5

4. Filtering Rows Based on Drug Name

Example

```
</>python
aspirin_patients = df[df["Drug"] == "Aspirin"]
print(aspirin_patients)
```

Output

	Patient_ID	Age	Drug	Concentration
0	P001	25	Aspirin	12.5
3	P004	40	Aspirin	20.5

5. Filtering with Multiple Conditions (AND)

Use & for AND conditions.

Example

```
</>python
result = df[(df["Age"] > 30) & (df["Drug"] == "Aspirin")]
print(result)
```

Output

	Patient_ID	Age	Drug	Concentration
3	P004	40	Aspirin	20.5

6. Filtering with Multiple Conditions (OR)

Use | for OR conditions.

Example

```
</>python
result = df[(df["Drug"] == "Aspirin") | (df["Drug"] == "Ibuprofen")]
print(result)
```

Output

```
  Patient_ID   Age      Drug   Concentration
0       P001    25   Aspirin           12.5
2       P003    28 Ibuprofen           10.8
3       P004    40   Aspirin           20.5
```

7. Selecting Rows Using loc[]

loc[] selects rows and columns using labels.

Example

```
</>python
print(df.loc[0])
```

Output

```
Patient_ID          P001
Age                   25
Drug             Aspirin
Concentration       12.5
```

Selecting Specific Rows and Columns

```
</>python
print(df.loc[0:2, ["Patient_ID", "Drug"]])
```

Output

```
  Patient_ID       Drug
0       P001    Aspirin
1       P002 Paracetamol
2       P003  Ibuprofen
```

8. Selecting Rows Using iloc[]

iloc[] selects data based on index positions.

Example

```
</>python
print(df.iloc[1])
```

Output

```
Patient_ID      P002
Age              35
Drug          Paracetamol
Concentration    18.0
```

Selecting Multiple Rows and Columns

```
</>python
print(df.iloc[0:3, 0:2])
```

Output

```
   Patient_ID  Age
0        P001   25
1        P002   35
2        P003   28
```

Pharmaceutical Example

Filter Patients with Drug Concentration Greater Than 15

high_concentration = df[df["Concentration"] > 15]

```
</>python
print(high_concentration)
```

Output

```
  Patient_ID  Age  Drug        Concentration
1       P002   35  Paracetamol  18.0
3       P004   40  Aspirin       20.5
```

Grouping Data and Performing Aggregation Functions in Pandas

Introduction

Grouping is the process of dividing data into groups based on one or more columns. After grouping, aggregation functions are used to calculate summary statistics such as sum, mean, count, minimum, and maximum values for each group.

When working with large datasets it's used to group and summarize the data to make analysis easier. Pandas a popular Python library provides powerful tools for this. groupby() and aggregation functions step by step with clear explanations and practical examples

In Pandas, grouping is performed using the groupby() function.

This technique is widely used in pharmaceutical and healthcare data analysis to summarize patient records, drug usage, clinical trial results, and ADR reports.

Aggregation in Pandas

Aggregation means applying a mathematical function to summarize data. It can be used to get a summary of columns in our dataset like getting sum, minimum, maximum etc. from a particular column of our dataset. The function used for aggregation is **agg()** the parameter is the function we want to perform. Some functions used in the aggregation are:

Function	Description
sum()	Compute sum of column values
min()	Compute min of column values
max()	Compute max of column values
mean()	Compute mean of column
size()	Compute column sizes
describe()	Generates descriptive statistics
first()	Compute first of group values
last()	Compute last of group values
count()	Compute count of column values
std()	Standard deviation of column
var()	Compute variance of column
sem()	Standard error of the mean of column

What is groupby()?

The groupby() function groups rows that have the same value in a specified column.

Syntax

```
</>python
df.groupby("Column_Name")
```

Sample Dataset

```
</>python
import pandas as pd
data = {
   "Drug": ["Aspirin", "Paracetamol", "Aspirin", "Ibuprofen", "Paracetamol"],
   "Patient_ID": ["P001", "P002", "P003", "P004", "P005"],
   "Concentration": [12.5, 18.0, 15.5, 10.8, 20.0]
}
df = pd.DataFrame(data)
print(df)
```

Output

	Drug	Patient_ID	Concentration
0	Aspirin	P001	12.5
1	Paracetamol	P002	18.0
2	Aspirin	P003	15.5
3	Ibuprofen	P004	10.8
4	Paracetamol	P005	20.0

1. Calculating Mean Using groupby()

Example

```
</>python
mean_concentration = df.groupby("Drug")["Concentration"].mean()
print(mean_concentration)
```

Output

```
Drug
Aspirin       14.0
Ibuprofen     10.8
Paracetamol   19.0
Name: Concentration, dtype: float64
```

This calculates the average drug concentration for each drug.

2. Calculating Sum

Example

```
</>python
total_concentration = df.groupby("Drug")["Concentration"].sum()
print(total_concentration)
```

Output

```
Drug
Aspirin       28.0
Ibuprofen     10.8
Paracetamol   38.0
```

3. Counting Records

Example

```
</>python
patient_count = df.groupby("Drug")["Patient_ID"].count()
print(patient_count)
```

Output

```
Drug
Aspirin       2
Ibuprofen     1
Paracetamol   2
```

This shows the number of patients receiving each drug.

4. Finding Maximum Value

Example

```
</>python
max_value = df.groupby("Drug")["Concentration"].max()
print(max_value)
```

Output

```
Drug
Aspirin       15.5
Ibuprofen     10.8
Paracetamol   20.0
```

5. Finding Minimum Value

Example

```
</>python
min_value = df.groupby("Drug")["Concentration"].min()
print(min_value)
```

Output

```
Drug
Aspirin         12.5
Ibuprofen       10.8
Paracetamol     18.0
```

6. Multiple Aggregation Functions

Use agg() to apply several functions at once.

Example

```
</>python
summary = df.groupby("Drug")["Concentration"].agg(
    ["count", "mean", "min", "max", "sum"]
)
print(summary)
```

Output

Drug	count	mean	min	max	sum
Aspirin	2	14.0	12.5	15.5	28.0
Ibuprofen	1	10.8	10.8	10.8	10.8
Paracetamol	2	19.0	18.0	20.0	38.0

7. Grouping by Multiple Columns

Example

```
</>python
data = {
   "Drug": ["Aspirin", "Aspirin", "Paracetamol", "Paracetamol"],
   "Gender": ["Male", "Female", "Male", "Female"],
   "Concentration": [12, 15, 18, 20]
}
df = pd.DataFrame(data)
result = df.groupby(["Drug", "Gender"])["Concentration"].mean()
print(result)
```

Output

Drug	Gender	
Aspirin	Female	15.0
	Male	12.0
Paracetamol	Female	20.0
	Male	18.0

Pharmaceutical Example

PK Study Dataset

```
</>python
import pandas as pd

pk_data = {
   "Drug": ["Drug A", "Drug A", "Drug B", "Drug B"],
   "Concentration": [12.5, 15.0, 10.2, 11.8]
}
df = pd.DataFrame(pk_data)
result = df.groupby("Drug")["Concentration"].mean()
print(result)
```

Output

```
Drug
Drug A    13.75
Drug B    11.00
```

Multiple Choice Questions

1. What is Pandas?

A) Database
B) Python library for data analysis
C) Operating System
D) Compiler
Answer: B

2. How is Pandas commonly imported?

A) import numpy as np
B) import pandas as pd
C) import pandas
D) import pd
Answer: B

3. Which data structure is one-dimensional in Pandas?

A) DataFrame
B) Series
C) Array
D) Table
Answer: B

4. Which data structure is two-dimensional in Pandas?

A) Series
B) List
C) DataFrame
D) Tuple
Answer: C

5. A DataFrame consists of:

A) Only rows
B) Only columns
C) Rows and columns
D) None
Answer: C

6. Which function reads a CSV file?

A) pd.read_excel()
B) pd.read_csv()
C) pd.read_table()
D) pd.open()
Answer: B

7. Which function reads an Excel file?

A) pd.read_excel()
B) pd.read_csv()
C) pd.read_data()
D) pd.excel()
Answer: A

8. Which function displays the first five rows?

A) tail()
B) info()
C) head()
D) describe()
Answer: C

9. Which function displays the last five rows?

A) head()
B) tail()
C) info()
D) shape()
Answer: B

10. Which function provides dataset information?

A) info()
B) describe()
C) head()
D) tail()
Answer: A

11. Which function gives statistical summary?

A) info()
B) mean()
C) describe()
D) shape()
Answer: C

12. Which function checks missing values?

A) isnull()
B) fillna()
C) dropna()
D) count()
Answer: A

13. Which function removes missing values?

A) isnull()
B) fillna()
C) dropna()
D) replace()
Answer: C

14. Which function fills missing values?

A) fillna()
B) dropna()
C) null()
D) remove()
Answer: A

15. Missing values in Pandas are represented by:

A) 0
B) NULL
C) NaN
D) Empty
Answer: C

16. Which operator is used to filter records greater than 50?

A) <
B) >
C) =
D) !=
Answer: B

17. What does df["Age"] represent?

A) Row
B) Column Age
C) File
D) Index
Answer: B

18. Which function is used to group data?

A) aggregate()
B) groupby()
C) summarize()
D) classify()
Answer: B

19. Which function calculates average?

A) mean()
B) sum()
C) count()
D) max()
Answer: A

20. Which function calculates total?

A) count()
B) max()
C) sum()
D) mean()
Answer: C

21. Which function counts records?

A) count()
B) mean()
C) sum()
D) min()
Answer: A

22. Which function finds maximum value?

A) min()
B) max()
C) mean()
D) count()
Answer: B

23. Which function finds minimum value?

A) max()
B) mean()
C) min()
D) sum()
Answer: C

24. Which function shows the dimensions of a DataFrame?

A) shape
B) info()
C) size()
D) len()
Answer: A

25. Which attribute returns column names?

A) rows
B) columns
C) names
D) labels
Answer: B

26. Which method displays data types?

A) dtype
B) info()
C) types()
D) datatypes()
Answer: B

27. A Series can contain:

A) One column of data
B) Multiple tables
C) Multiple sheets
D) Databases
Answer: A

28. DataFrame is similar to:

A) Dictionary
B) Excel Spreadsheet
C) Tuple
D) String
Answer: B

29. Which function returns the number of rows and columns?

A) shape
B) size
C) len
D) count
Answer: A

30. Which command creates a Series?

A) pd.DataFrame()
B) pd.Series()
C) pd.Array()
D) pd.List()
Answer: B

31. Which command creates a DataFrame?

A) pd.Series()
B) pd.DataFrame()
C) pd.Table()
D) pd.Frame()
Answer: B

32. Which file format is commonly used for PK study datasets?

A) CSV
B) MP3
C) JPG
D) PNG
Answer: A

33. ADR stands for:

A) Adverse Drug Reaction
B) Advanced Drug Research
C) Analytical Drug Report
D) Active Drug Response
Answer: A

34. Which function is useful before data analysis?

A) head()
B) tail()
C) info()
D) All of these
Answer: D

35. Data cleaning improves:

A) Data quality
B) Data errors
C) Data duplication
D) None
Answer: A

36. Which method replaces missing values with mean?

A) fillna(mean)
B) meanfill()
C) replace()
D) average()
Answer: A

37. Which function can remove duplicate rows?

A) remove()
B) drop_duplicates()
C) unique()
D) delete()
Answer: B

38. Which function returns unique values?

A) unique()
B) distinct()
C) values()
D) count()
Answer: A

39. Which function sorts data?

A) arrange()
B) sort_values()
C) sort()
D) organize()
Answer: B

40. Filtering data means:

A) Removing files
B) Selecting records based on conditions
C) Creating DataFrames
D) Saving data
Answer: B

41. Which symbol represents “equal to” in filtering?

A) =
B) ==
C) ===
D) :=
Answer: B

42. Which symbol represents “not equal to”?

A) <>
B) !=
C) ~=
D) =!
Answer: B

43. Which function can display random rows?

A) sample()
B) random()
C) select()
D) choose()
Answer: A

44. Which aggregation function gives median?

A) mean()
B) median()
C) average()
D) middle()
Answer: B

45. Which aggregation function gives standard deviation?

A) std()
B) var()
C) mean()
D) count()
Answer: A

46. Which function combines grouping and aggregation?

A) groupby()
B) aggregate()
C) Both A and B
D) None
Answer: C

47. Pandas is mainly used for:

A) Web browsing
B) Data analysis
C) Game development
D) Networking
Answer: B

48. Which function returns column-wise statistics?

A) describe()
B) head()
C) tail()
D) shape()
Answer: A

49. Which type of data is best handled by Pandas?

A) Structured tabular data
B) Audio files
C) Video files
D) Images
Answer: A

50. Why is Pandas important in pharmaceutical data analysis?

A) It helps analyze PK and ADR datasets efficiently
B) It creates operating systems
C) It designs hardware
D) It develops games
Answer: A

Short Questions and Answers

1. What is Pandas?
Answer: Pandas is a Python library used for data manipulation, analysis, and handling structured datasets.
2. Why is Pandas important in data analysis?
Answer: It provides efficient tools for cleaning, organizing, and analyzing data.
3. What is a Pandas Series?
Answer: A Series is a one-dimensional labeled array that can store different types of data.
4. What is a DataFrame?
Answer: A DataFrame is a two-dimensional tabular data structure with rows and columns.
5. Differentiate between Series and DataFrame.
Answer: A Series is one-dimensional, whereas a DataFrame is two-dimensional.
6. How do you import the Pandas library?
Answer: `import pandas as pd`
7. Which function is used to read a CSV file?
Answer: `pd.read_csv()`
8. Which function is used to read an Excel file?
Answer: `pd.read_excel()`
9. What is the purpose of the head() function?
Answer: It displays the first five rows of a dataset.
10. What is the purpose of the tail() function?
Answer: It displays the last five rows of a dataset.
11. What information does the info() function provide?
Answer: It provides details about columns, data types, and missing values.
12. What is the use of the describe() function?
Answer: It generates statistical summaries of numerical data.
13. What are missing values in a dataset?
Answer: Missing values are data entries that are not available or are left blank.
14. How can missing values be identified in Pandas?
Answer: By using the `isnull()` function.
15. How can missing values be removed?
Answer: By using the `dropna()` function.
16. How can missing values be replaced?
Answer: By using the `fillna()` function.
17. What is data filtering?
Answer: Data filtering is the process of selecting records that satisfy specific conditions.
18. How can you select patients older than 50 years from a dataset?
Answer:
df[df["Age"] > 50]
19. What is the purpose of the groupby() function?
Answer: It groups data based on one or more columns for analysis.
20. Name any three aggregation functions used in Pandas.
Answer: `mean()`, `sum()`, and `count()`.

Long Questions

1. Explain the Pandas library and discuss its importance in pharmaceutical data analysis.
2. Describe the structure, features, and applications of Pandas Series and DataFrame with suitable examples.
3. Differentiate between Series and DataFrame in Pandas. Explain their creation and practical applications.
4. Explain how CSV and Excel files can be imported into Pandas. Discuss their importance in PK study datasets and ADR reports.
5. Describe the various functions used for dataset inspection in Pandas. Explain the significance of head(), tail(), info(), and describe().
6. Explain data cleaning in Pandas. Discuss various techniques used for handling missing values with examples.
7. Discuss the methods used for identifying, removing, and replacing missing values in healthcare datasets using Pandas.
8. Explain filtering and selecting data in Pandas using conditional statements. Illustrate with suitable examples.
9. What is data grouping in Pandas? Explain the groupby() function and its applications in pharmaceutical research.
10. Explain the complete process of reading, inspecting, cleaning, and analyzing a pharmaceutical dataset using Pandas.

Case-Based Learning (CBL)

Case Study: Analysis of PK Study Dataset

Case:
A pharmaceutical researcher has a PK (Pharmacokinetic) study dataset stored in a CSV file. The researcher wants to view the data, check its structure, and generate a summary report.

Python Program

```
</>python

import pandas as pd

df = pd.read_csv("pk_study.csv")

print(df.head())
print(df.tail())
print(df.info())
print(df.describe())
```

Learning Outcomes

- Understand the Pandas library.
- Read CSV files.
- Inspect datasets using head(), tail(), info(), and describe().
- Analyze pharmaceutical datasets.

Q1. What is the purpose of this program?
Answer: To view, inspect, and summarize a PK study dataset.
Q2. Which library is used in the program?
Answer: Pandas.
Q3. How is Pandas imported?
Answer: `import pandas as pd`
Q4. Which function is used to read the CSV file?
Answer: `pd.read_csv()`
Q5. What is the name of the DataFrame variable?
Answer: `df`
Q6. What does `head()` display?
Answer: The first five rows of the dataset.
Q7. What does `tail()` display?
Answer: The last five rows of the dataset.
Q8. What does `info()` show?
Answer: Information about columns, data types, and non-null values.

Q9. What does `describe()` show?
Answer: Statistical summary of numerical data.
Q10. What type of file is `pk_study.csv`**?**
Answer: A CSV (Comma-Separated Values) file.

Problem-Based Learning (PBL)

Problem 1: Introduction to Pandas Library

Scenario

A pharmaceutical company has collected data from 500 patients in a clinical trial. The researcher needs a tool to organize and analyze the data efficiently.

Problem

Which Python library can help manage and analyze this data?

Solution

The **Pandas** library is used for data manipulation and analysis. It helps organize data into tables and perform various operations easily.

```
</>python
import pandas as pd
```

Learning Outcome

Understand the importance of Pandas in data analysis.

Problem 2: Pandas Series and DataFrame Structures

Scenario

A pharmacist wants to store the daily sales of a medicine and also maintain a table containing medicine names and prices.

Problem

Which Pandas structures should be used?

Solution

- **Series**: For a single column of data.

- **DataFrame**: For multiple columns of related data.

```
</>python
import pandas as pd
sales = pd.Series([100, 120, 130])
data = {
   "Medicine": ["Paracetamol", "Aspirin"],
   "Price": [20, 25]
}
df = pd.DataFrame(data)
```

Learning Outcome

Differentiate between Series and DataFrame.

Problem 3: Reading PK Study Datasets and ADR Reports

Scenario

A researcher receives a PK study dataset in CSV format and an ADR report in Excel format.

Problem

How can these files be imported into Python?

Solution

```
</>python
import pandas as pd
pk_data = pd.read_csv("pk_study.csv")
adr_data = pd.read_excel("adr_report.xlsx")
```

Learning Outcome

Learn to import CSV and Excel files.

Short PBL Questions

Q1. Which library is used for data analysis in Python?
Answer: Pandas.
Q2. What is a Series?
Answer: A one-dimensional labeled array.
Q3. What is a DataFrame?
Answer: A two-dimensional table with rows and columns.
Q4. Which function reads a CSV file?
Answer: `read_csv()`.
Q5. Which function reads an Excel file?
Answer: `read_excel()`.

Q6. What does head() display?
Answer: The first five rows of a dataset.
Q7. What does tail() display?
Answer: The last five rows of a dataset.
Q8. What does info() provide?
Answer: Dataset structure, column names, and data types.
Q9. What does describe() provide?
Answer: Statistical summary of numerical data.
Q10. Which function checks missing values?
Answer: `isnull()`.
Q11. Which function fills missing values?
Answer: `fillna()`.
Q12. Which function removes missing values?
Answer: `dropna()`.
Q13. How can records be filtered?
Answer: Using conditional statements such as `df[df["Age"] > 50]`.
Q14. Which function is used to group data?
Answer: `groupby()`.
Q15. Name three aggregation functions.
Answer: `mean()`, `sum()`, and `count()`.