

**Pandas Exam Paper 1 (Total 30 Questions - 2 Marks Each)**



**Section A: Data Creation and Importing (7 Questions)**

1. **Creating a DataFrame**   
Create a DataFrame using pd.DataFrame() with columns: 'Name', 'Age', and 'City' and values for 3 individuals.

**Answer: import pandas as pd**

**# Creating the DataFrame**

**data = {**

**'Name': ['Ravi', 'Sneha', 'Arjun'],**

**'Age': [28, 24, 30],**

**'City': ['Hyderabad', 'Mumbai', 'Bangalore']**

**}**

**df = pd.DataFrame(data)**

**# Displaying the DataFrame**

**print(df)**

2. **Reading CSV File**   
 Write the command to read a CSV file named data.csv into a DataFrame.

**Answer:** **import pandas as pd**

**# Reading the CSV file into a DataFrame**

**df = pd.read\_csv('data.csv')**

**# Displaying the DataFrame**

**print(df)**

3. **Reading Excel File**   
 How would you load data from an Excel file called data.xlsx into a DataFrame?

**Answer:** **import pandas as pd**

**# Reading the Excel file into a DataFrame**

**df = pd.read\_excel('data.xlsx')**

**# Displaying the DataFrame**

**print(df)**

4. **Reading JSON File**   
 Load a JSON file named data.json into a DataFrame.

**Answer: import pandas as pd**

**# Reading the JSON file into a DataFrame**

**df = pd.read\_json('data.json')**

**# Displaying the DataFrame**

**print(df)**

5. **Reading HTML Table**   
 Parse an HTML file containing a table and return it as a DataFrame.

**Answer:** **import pandas as pd**

**# Reading the HTML table from a file**

**dfs = pd.read\_html('data.html') # Returns a list of DataFrames**

**# Selecting the first table (if multiple tables exist in the file)**

**df = dfs[0]**

**# Displaying the DataFrame**

**print(df)**

6. **Creating DataFrame from a Dictionary**   
Create a DataFrame using a dictionary with two columns: 'Product' and 'Price', containing 3 items.

**Answer: import pandas as pd**

**# Creating a dictionary with product names and prices**

**data = {**

**'Product': ['Laptop', 'Smartphone', 'Headphones'],**

**'Price': [75000, 30000, 2000]**

**}**

**# Converting the dictionary into a DataFrame**

**df = pd.DataFrame(data)**

**# Displaying the DataFrame**

**print(df)**

7. **Exploring DataFrame from CSV**   
After loading a CSV into a DataFrame, what command would you use to see the first 5 rows?

**Answer:** **import pandas as pd**

**# Reading the CSV file**

**df = pd.read\_csv('data.csv')**

**# Displaying the first 5 rows**

**print(df.head()**



**Section B: Data Inspection (7 Questions)**

8. **Viewing First Few Rows**   
 Use the appropriate command to display the first 10 rows of a DataFrame df .

**Answer: import pandas as pd**

**# Creating a sample DataFrame**

**data = {**

**'Name': ['Ravi', 'Sneha', 'Arjun', 'Meera', 'Vikram', 'Aditi', 'Kiran', 'Rahul', 'Neha', 'Aman'],**

**'Age': [28, 24, 30, 27, 26, 25, 29, 23, 22, 31]**

**}**

**df = pd.DataFrame(data)**

**# Displaying the first 10 rows**

**print(df.head(10))**

9. **Viewing Last Few Rows**   
 Show the last 3 rows of the DataFrame df .

**Answer: import pandas as pd**

**# Creating a sample DataFrame**

**data = {**

**'Name': ['Ravi', 'Sneha', 'Arjun', 'Meera', 'Vikram', 'Aditi', 'Kiran', 'Rahul', 'Neha', 'Aman'],**

**'Age': [28, 24, 30, 27, 26, 25, 29, 23, 22, 31]**

**}**

**df = pd.DataFrame(data)**

**# Displaying the last 3 rows**

**print(df.tail(3))**

10. **Checking DataFrame Information**   
Which command provides concise information about the DataFrame, such as data types and memory usage?

**Answer: import pandas as pd**

**# Creating a sample DataFrame**

**data = {**

**'Name': ['Ravi', 'Sneha', 'Arjun', 'Meera'],**

**'Age': [28, 24, 30, 27],**

**'City': ['Hyderabad', 'Mumbai', 'Bangalore', 'Chennai']**

**}**

**df = pd.DataFrame(data)**

**# Displaying concise information about the DataFrame**

**df.info()**

11. **Descriptive Statistics**   
 How do you generate descriptive statistics like mean, median, and standard deviation for

numeric columns in a DataFrame?   
**Answer: import pandas as pd**

**# Creating a sample DataFrame**

**data = {**

**'Age': [25, 30, 35, 40, 45],**

**'Salary': [50000, 60000, 70000, 80000, 90000]**

**}**

**df = pd.DataFrame(data)**

**# Displaying descriptive statistics**

**print(df.describe())**

**# Getting specific statistics**

**print("Mean:\n", df.mean()) # Mean of each numeric column**

**print("Median:\n", df.median()) # Median of each numeric column**

**print("Standard Deviation:\n", df.std()) # Standard deviation of each numeric column**

12. **Checking Data Types**   
 What command returns the data types of each column in the DataFrame?

**Answer: import pandas as pd**

**# Creating a sample DataFrame**

**data = {**

**'Name': ['Ravi', 'Sneha', 'Arjun', 'Meera'],**

**'Age': [28, 24, 30, 27],**

**'Salary': [50000.5, 60000.75, 70000.0, 80000.25],**

**'Joining Date': ['2022-05-10', '2021-06-15', '2020-08-20', '2019-12-05']**

**}**

**df = pd.DataFrame(data)**

**# Checking data types of each column**

**print(df.dtypes)**

13. **Checking DataFrame Shape**   
 How do you find the number of rows and columns in the DataFrame?

**Answer: import pandas as pd**

**# Creating a sample DataFrame**

**data = {**

**'Name': ['Ravi', 'Sneha', 'Arjun', 'Meera'],**

**'Age': [28, 24, 30, 27],**

**'City': ['Hyderabad', 'Mumbai', 'Bangalore', 'Chennai']**

**}**

**df = pd.DataFrame(data)**

**# Checking the shape of the DataFrame**

**print(df.shape)**

14. **DataFrame Summary**   
 Explain what df.info() does and what kind of information it provides.

**Answer:** The df.info() method provides a **concise summary** of a **Pandas DataFrame**, including details about **column names, data types, non-null values, and memory usage**.

1. **DataFrame Structure**
   * Shows <class 'pandas.core.frame.DataFrame'>, confirming it's a DataFrame.
2. **Index Range**
   * RangeIndex: 4 entries, 0 to 3 → Indicates **total rows** (here, 4).
3. **Column Details**
   * Lists **column names**, **number of non-null values**, and **data types**.
4. **Non-Null Count**
   * Helps identify **missing values** (Salary has **3 non-null values**, meaning **1 missing**).
5. **Data Types**
   * Shows **int64**, **float64**, and **object** (string/text).
6. **Memory Usage**
   * Indicates **memory consumed** by the DataFrame (e.g., 260.0 bytes).



**Section C: Indexing and Selecting Data (8 Questions)**

15. **Setting an Index**   
 Set the 'ID' column as the index for the DataFrame df .

**Answer:** **import pandas as pd**

**# Creating a sample DataFrame**

**data = {**

**'ID': [101, 102, 103, 104],**

**'Name': ['Ravi', 'Sneha', 'Arjun', 'Meera'],**

**'Age': [28, 24, 30, 27],**

**'City': ['Hyderabad', 'Mumbai', 'Bangalore', 'Chennai']**

**}**

**df = pd.DataFrame(data)**

**# Setting 'ID' as the index**

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

**# Displaying the updated DataFrame**

**print(df)**

16. **Resetting an Index**   
 How do you reset the index of the DataFrame and return it to the default integer index?

**Answer:**

import pandas as pd

# Creating a sample DataFrame with 'ID' as index

data = {

'ID': [101, 102, 103, 104],

'Name': ['Ravi', 'Sneha', 'Arjun', 'Meera'],

'Age': [28, 24, 30, 27],

'City': ['Hyderabad', 'Mumbai', 'Bangalore', 'Chennai']

}

df = pd.DataFrame(data).set\_index('ID') # Setting 'ID' as index

print("Before Resetting Index:")

print(df)

# Resetting index

df.reset\_index(inplace=True)

print("\nAfter Resetting Index:")

print(df)

17. **Selecting Data by Position**   
 Retrieve the third row of the DataFrame using iloc[] .

**Answer: import pandas as pd**

**# Creating a sample DataFrame**

**data = {**

**'Name': ['Ravi', 'Sneha', 'Arjun', 'Meera'],**

**'Age': [28, 24, 30, 27],**

**'City': ['Hyderabad', 'Mumbai', 'Bangalore', 'Chennai']**

**}**

**df = pd.DataFrame(data)**

**# Selecting the third row (index 2)**

**third\_row = df.iloc[2]**

**print(third\_row)**

18. **Selecting Data by Label**   
 Use loc[] to access all rows where the 'Age' column is greater than 30.

**Answer: import pandas as pd**

**# Creating a sample DataFrame**

**data = {**

**'Name': ['Ravi', 'Sneha', 'Arjun', 'Meera', 'Rajesh'],**

**'Age': [28, 24, 30, 27, 35], # Rajesh is the only one older than 30**

**'City': ['Hyderabad', 'Mumbai', 'Bangalore', 'Chennai', 'Pune']**

**}**

**df = pd.DataFrame(data)**

**# Selecting rows where 'Age' > 30**

**filtered\_df = df.loc[df['Age'] > 30]**

**print(filtered\_df)**

19. **Querying the DataFrame**   
 Use query() to select rows where the 'Salary' is greater than 50000.

**Answer: import pandas as pd**

**# Creating a sample DataFrame**

**data = {**

**'Name': ['Ravi', 'Sneha', 'Arjun', 'Meera', 'Rajesh'],**

**'Age': [28, 24, 30, 27, 35],**

**'Salary': [45000, 52000, 48000, 60000, 75000]**

**}**

**df = pd.DataFrame(data)**

**# Selecting rows where 'Salary' > 50000 using query()**

**high\_salary\_df = df.query("Salary > 50000")**

**print(high\_salary\_df)**

20. **Sorting Values**   
 Sort the DataFrame df by the 'Price' column in ascending order.

**Answer: import pandas as pd**

**# Creating a sample DataFrame**

**data = {**

**'Product': ['Laptop', 'Phone', 'Tablet', 'Smartwatch'],**

**'Price': [80000, 30000, 40000, 20000]**

**}**

**df = pd.DataFrame(data)**

**# Sorting by 'Price' in ascending order**

**sorted\_df = df.sort\_values(by='Price', ascending=True)**

**print(sorted\_df)**

21. **Selecting Top N Rows by Value**   
 Select the top 3 rows with the highest values in the 'Marks' column using nlargest() .

**Answer:** **import pandas as pd**

**# Creating a sample DataFrame**

**data = {**

**'Student': ['Ravi', 'Sneha', 'Arjun', 'Meera', 'Rajesh'],**

**'Marks': [85, 92, 78, 95, 88]**

**}**

**df = pd.DataFrame(data)**

**# Selecting the top 3 students with the highest marks**

**top\_students = df.nlargest(3, 'Marks')**

**print(top\_students)**

22. **Selecting Smallest N Rows by Value**   
 Use nsmallest() to return the bottom 2 rows based on the 'Age' column.

**Answer:** **import pandas as pd**

**# Creating a sample DataFrame**

**data = {**

**'Name': ['Ravi', 'Sneha', 'Arjun', 'Meera', 'Rajesh'],**

**'Age': [28, 24, 30, 27, 35]**

**}**

**df = pd.DataFrame(data)**

**# Selecting the bottom 2 rows based on 'Age'**

**youngest\_people = df.nsmallest(2, 'Age')**

**print(youngest\_people)**



**Section D: Data Cleaning (8 Questions)**

23. **Detecting Missing Values**   
 Write the command to detect missing values in the DataFrame df .

**Answer:** **import pandas as pd**

**# Creating a sample DataFrame with missing values**

**data = {**

**'Name': ['Ravi', 'Sneha', 'Arjun', 'Meera', None],**

**'Age': [28, None, 30, 27, 35],**

**'City': ['Hyderabad', 'Mumbai', None, 'Chennai', 'Pune']**

**}**

**df = pd.DataFrame(data)**

**# Detecting missing values**

**missing\_values = df.isnull()**

**print(missing\_values)**

24. **Removing Missing Values**   
 Remove rows with missing values in the DataFrame df .

**Answer:** **import pandas as pd**

**# Creating a sample DataFrame with missing values**

**data = {**

**'Name': ['Ravi', 'Sneha', 'Arjun', 'Meera', None],**

**'Age': [28, None, 30, 27, 35],**

**'City': ['Hyderabad', 'Mumbai', None, 'Chennai', 'Pune']**

**}**

**df = pd.DataFrame(data)**

**print("Original DataFrame:")**

**print(df)**

**# Removing rows with missing values**

**df\_cleaned = df.dropna()**

**print("\nDataFrame after removing missing values:")**

**print(df\_cleaned)**

25. **Filling Missing Values**   
 Fill missing values in the 'Salary' column with the mean salary value.

**Answer: import pandas as pd**

**# Creating a sample DataFrame with missing values in 'Salary' column**

**data = {**

**'Name': ['Ravi', 'Sneha', 'Arjun', 'Meera', 'Rajesh'],**

**'Salary': [50000, None, 60000, 55000, None]**

**}**

**df = pd.DataFrame(data)**

**print("Original DataFrame:")**

**print(df)**

**# Filling missing 'Salary' values with the mean salary**

**df['Salary'].fillna(df['Salary'].mean(), inplace=True)**

**print("\nDataFrame after filling missing Salary values:")**

**print(df)**

26. **Dropping Duplicate Rows**   
 How do you remove duplicate rows from the DataFrame?

**Answer: import pandas as pd**

**# Creating a sample DataFrame with duplicate rows**

**data = {**

**'Name': ['Ravi', 'Sneha', 'Arjun', 'Meera', 'Ravi'],**

**'Age': [28, 25, 30, 27, 28],**

**'City': ['Hyderabad', 'Mumbai', 'Delhi', 'Chennai', 'Hyderabad']**

**}**

**df = pd.DataFrame(data)**

**print("Original DataFrame:")**

**print(df)**

**# Removing duplicate rows**

**df.drop\_duplicates(inplace=True)**

**print("\nDataFrame after removing duplicate rows:")**

**print(df)**

27. **Replacing Values**   
 Replace all occurrences of the value 'M' in the 'Gender' column with 'Male'.

**Answer: import pandas as pd**

**# Creating a sample DataFrame**

**data = {**

**'Name': ['Ravi', 'Sneha', 'Arjun', 'Meera'],**

**'Gender': ['M', 'F', 'M', 'F'],**

**'Age': [28, 25, 30, 27]**

**}**

**df = pd.DataFrame(data)**

**print("Original DataFrame:")**

**print(df)**

**# Replacing 'M' with 'Male' in the 'Gender' column**

**df['Gender'].replace('M', 'Male', inplace=True)**

**print("\nDataFrame after replacing values:")**

**print(df)**

28. **Converting Data Types**   
 Convert the 'Age' column to integers using astype() .

**Answer:**

import pandas as pd

# Creating a sample DataFrame with 'Age' as string

data = {

'Name': ['Ravi', 'Sneha', 'Arjun', 'Meera'],

'Age': ['28', '25', '30', '27'] # Age as string

}

df = pd.DataFrame(data)

print("Original DataFrame:")

print(df)

print("\nData Types Before Conversion:")

print(df.dtypes)

# Converting 'Age' column to integer

df['Age'] = df['Age'].astype(int)

print("\nDataFrame after conversion:")

print(df)

print("\nData Types After Conversion:")

print(df.dtypes)

29. **Handling Missing Values in Specific Column**   
 Remove rows where the 'Age' column contains missing values.

**Answer: import pandas as pd**

**# Creating a DataFrame with missing values in the 'Age' column**

**data = {**

**'Name': ['Ravi', 'Sneha', 'Arjun', 'Meera', 'Kiran'],**

**'Age': [28, None, 30, 27, None] # Missing values (None)**

**}**

**df = pd.DataFrame(data)**

**print("Original DataFrame:")**

**print(df)**

**# Removing rows where 'Age' column has missing values**

**df.dropna(subset=['Age'], inplace=True)**

**print("\nDataFrame after removing rows with missing 'Age':")**

**print(df)**

30. **Filling Missing Values Using Forward Fill**   
 Use the forward fill method to fill missing values in the DataFrame df .

**Answer: import pandas as pd**

**# Creating a DataFrame with missing values**

**data = {**

**'Name': ['Ravi', 'Sneha', 'Arjun', 'Meera', 'Kiran'],**

**'Age': [28, None, 30, None, 26], # Missing values (None/NaN)**

**'Salary': [50000, None, 60000, None, 45000] # Missing values**

**}**

**df = pd.DataFrame(data)**

**print("Original DataFrame:")**

**print(df)**

**# Filling missing values using forward fill**

**df.fillna(method='ffill', inplace=True)**

**print("\nDataFrame after forward fill:")**

**print(df)**

