

**Pandas Exam Paper 2 - (Total Marks 30 Questions - 2**

**Marks Each)**



**Section A: Data Manipulation (7 Questions)**

1. **Applying Functions to Columns**   
 Apply a function to double the values of the 'Price' column using apply() .

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

**# Sample DataFrame**

**data = {'Product': ['A', 'B', 'C'], 'Price': [10, 20, 30]}**

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

**# Applying a function to double the 'Price' column**

**df['Price'] = df['Price'].apply(lambda x: x \* 2)**

**print(df)**

2. **Mapping Values in Series**   
Use map() to replace all occurrences of 'Yes' in the 'Passed' column with True and 'No' with False .

**Answer: import pandas as pd**

**# Sample DataFrame**

**data = {'Name': ['Alice', 'Bob', 'Charlie', 'David'],**

**'Passed': ['Yes', 'No', 'Yes', 'No']}**

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

**# Mapping 'Yes' to True and 'No' to False**

**df['Passed'] = df['Passed'].map({'Yes': True, 'No': False})**

**print(df)**

3. **Lowercase Strings**   
 Convert all strings in the 'Names' column to lowercase.

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

**# Sample DataFrame**

**data = {'Names': ['Alice', 'Bob', 'Charlie', 'David']}**

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

**# Converting all strings in 'Names' column to lowercase**

**df['Names'] = df['Names'].str.lower()**

**print(df)**

4. **Uppercase Strings**   
 Convert the 'City' column to uppercase.

**Answer: import pandas as pd**

**# Sample DataFrame**

**data = {'City': ['New York', 'London', 'Tokyo', 'Paris']}**

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

**# Converting the 'City' column to uppercase**

**df['City'] = df['City'].str.upper()**

**print(df)**

5. **Splitting Strings**   
 Split the 'FullName' column into 'FirstName' and 'LastName' using a space as the delimiter.

**Answer:**

import pandas as pd

# Sample DataFrame

data = {'FullName': ['Alice Johnson', 'Bob Smith', 'Charlie Brown', 'David Lee']}

df = pd.DataFrame(data)

# Splitting 'FullName' into 'FirstName' and 'LastName'

df[['FirstName', 'LastName']] = df['FullName'].str.split(' ', expand=True)

print(df)

6. **String Contains**   
 Filter rows where the 'Email' column contains '@gmail.com'.

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

**# Sample DataFrame**

**data = {'Name': ['Alice', 'Bob', 'Charlie', 'David'],**

**'Email': ['alice@gmail.com', 'bob@yahoo.com', 'charlie@gmail.com', 'david@outlook.com']}**

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

**# Filtering rows where 'Email' contains '@gmail.com'**

**gmail\_users = df[df['Email'].str.contains('@gmail.com', na=False)]**

**print(gmail\_users)**

7. **Replacing String Patterns**   
Use str.replace() to replace the domain in all emails from '@example.com' to '@newdomain.com'.

**Answer: import pandas as pd**

**# Sample DataFrame**

**data = {'Name': ['Alice', 'Bob', 'Charlie', 'David'],**

**'Email': ['alice@example.com', 'bob@example.com', 'charlie@example.com', 'david@example.com']}**

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

**# Replacing '@example.com' with '@newdomain.com'**

**df['Email'] = df['Email'].str.replace('@example.com', '@newdomain.com', regex=False)**

**print(df)**



**Section B: Grouping and Aggregation (8 Questions)**

8. **Grouping Data**   
Group the DataFrame by the 'Department' column and calculate the mean salary for each department.

**Answer: import pandas as pd**

**# Sample DataFrame**

**data = {'Employee': ['Alice', 'Bob', 'Charlie', 'David', 'Eve'],**

**'Department': ['HR', 'IT', 'IT', 'HR', 'Finance'],**

**'Salary': [50000, 70000, 80000, 55000, 90000]}**

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

**# Grouping by 'Department' and calculating mean salary**

**mean\_salary = df.groupby('Department')['Salary'].mean().reset\_index()**

**print(mean\_salary)**

9. **Aggregating Data**   
 Apply multiple aggregate functions (mean, max) to the 'Sales' column using agg() .

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

**# Sample DataFrame**

**data = {'Store': ['A', 'B', 'C', 'A', 'B', 'C'],**

**'Sales': [1000, 1500, 2000, 1200, 1800, 2500]}**

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

**# Applying multiple aggregate functions (mean, max) to the 'Sales' column**

**sales\_summary = df['Sales'].agg(['mean', 'max'])**

**print(sales\_summary)**

10. **Aggregate Multiple Functions**   
 Use aggregate() to calculate both the sum and count of the 'Marks' column.

**Answer:**

import pandas as pd

# Sample DataFrame

data = {'Student': ['Alice', 'Bob', 'Charlie', 'David', 'Eve'],

'Marks': [85, 90, 78, 92, 88]}

df = pd.DataFrame(data)

# Applying multiple aggregate functions (sum, count) to the 'Marks' column

marks\_summary = df['Marks'].agg(['sum', 'count'])

print(marks\_summary)

11. **Filtering with isin()**   
 Filter rows where the 'City' column is either 'New York' or 'Los Angeles' using isin() .

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

**# Sample DataFrame**

**data = {'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Eve'],**

**'City': ['New York', 'Los Angeles', 'Chicago', 'Houston', 'New York']}**

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

**# Filtering rows where 'City' is either 'New York' or 'Los Angeles'**

**filtered\_df = df[df['City'].isin(['New York', 'Los Angeles'])]**

**print(filtered\_df)**

12. **Grouping and Aggregating**   
Group the DataFrame by 'Gender' and calculate the sum of the 'Marks' column for each group.

**Answer: import pandas as pd**

**# Sample DataFrame**

**data = {'Student': ['Alice', 'Bob', 'Charlie', 'David', 'Eve'],**

**'Gender': ['Female', 'Male', 'Male', 'Male', 'Female'],**

**'Marks': [85, 90, 78, 92, 88]}**

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

**# Grouping by 'Gender' and calculating the sum of 'Marks'**

**marks\_sum = df.groupby('Gender')['Marks'].sum().reset\_index()**

**print(marks\_sum)**

13. **Multiple Aggregations on Multiple Columns**   
 Perform multiple aggregations (min, max, mean) on the 'Age' and 'Salary' columns.

**Answer: import pandas as pd**

**# Sample DataFrame**

**data = {'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Eve'],**

**'Age': [25, 30, 35, 40, 28],**

**'Salary': [50000, 60000, 75000, 80000, 55000]}**

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

**# Applying multiple aggregations (min, max, mean) on 'Age' and 'Salary'**

**summary = df[['Age', 'Salary']].agg(['min', 'max', 'mean'])**

**print(summary)**

14. **Grouping and Counting**   
 Group by 'City' and count the number of entries in each city.

**Answer: import pandas as pd**

**# Sample DataFrame**

**data = {'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Eve', 'Frank'],**

**'City': ['New York', 'Los Angeles', 'New York', 'Chicago', 'Los Angeles', 'Chicago']}**

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

**# Grouping by 'City' and counting entries**

**city\_counts = df.groupby('City').size().reset\_index(name='Count')**

**print(city\_counts)**

15. **Using apply() with Groupby**   
Apply a custom function to find the range (max-min) of the 'Salary' column for each department.

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

**# Sample DataFrame**

**data = {'Employee': ['Alice', 'Bob', 'Charlie', 'David', 'Eve', 'Frank'],**

**'Department': ['HR', 'IT', 'IT', 'HR', 'Finance', 'Finance'],**

**'Salary': [50000, 70000, 80000, 55000, 90000, 95000]}**

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

**# Custom function to calculate range (max - min)**

**def salary\_range(salaries):**

**return salaries.max() - salaries.min()**

**# Grouping by 'Department' and applying the custom function**

**salary\_ranges = df.groupby('Department')['Salary'].apply(salary\_range).reset\_index(name='Salary Range')**

**print(salary\_ranges)**



**Section C: Merging, Joining, and Concatenating (5 Questions)**

16. **Concatenating DataFrames**   
 Concatenate two DataFrames df1 and df2 along rows.

import pandas as pd

# Creating first DataFrame

df1 = pd.DataFrame({'ID': [1, 2], 'Name': ['Alice', 'Bob'], 'Age': [25, 30]})

# Creating second DataFrame

df2 = pd.DataFrame({'ID': [3, 4], 'Name': ['Charlie', 'David'], 'Age': [35, 40]})

# Concatenating along rows

df\_combined = pd.concat([df1, df2], ignore\_index=True)

print(df\_combined)

**Answer:**

17. **Merging DataFrames**   
 Merge two DataFrames df1 and df2 on the 'ID' column.

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

**# Creating first DataFrame**

**df1 = pd.DataFrame({'ID': [1, 2, 3], 'Name': ['Alice', 'Bob', 'Charlie'], 'Age': [25, 30, 35]})**

**# Creating second DataFrame**

**df2 = pd.DataFrame({'ID': [1, 2, 4], 'Salary': [50000, 60000, 70000]})**

**# Merging DataFrames on 'ID'**

**df\_merged = pd.merge(df1, df2, on='ID', how='inner')**

**print(df\_merged\_**

18. **Merging with Different Keys**   
 Merge DataFrames on different column names: 'df1' has 'EmployeeID' and 'df2' has 'ID'.

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

**# Creating first DataFrame**

**df1 = pd.DataFrame({'EmployeeID': [1, 2, 3], 'Name': ['Alice', 'Bob', 'Charlie'], 'Age': [25, 30, 35]})**

**# Creating second DataFrame**

**df2 = pd.DataFrame({'ID': [1, 2, 4], 'Salary': [50000, 60000, 70000]})**

**# Merging DataFrames with different column names**

**df\_merged = pd.merge(df1, df2, left\_on='EmployeeID', right\_on='ID', how='inner')**

**print(df\_merged)**

19. **Concatenating Along Columns**   
 Concatenate two DataFrames df1 and df2 along columns.

**Answer: import pandas as pd**

**# Creating first DataFrame**

**df1 = pd.DataFrame({'ID': [1, 2, 3], 'Name': ['Alice', 'Bob', 'Charlie']})**

**# Creating second DataFrame**

**df2 = pd.DataFrame({'Age': [25, 30, 35], 'Salary': [50000, 60000, 70000]})**

**# Concatenating along columns**

**df\_combined = pd.concat([df1, df2], axis=1)**

**print(df\_combined)**

20. **Joining DataFrames**   
 Join df1 and df2 on the 'ID' column with an outer join.

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

**# Creating first DataFrame**

**df1 = pd.DataFrame({'ID': [1, 2, 3], 'Name': ['Alice', 'Bob', 'Charlie']})**

**# Creating second DataFrame**

**df2 = pd.DataFrame({'ID': [2, 3, 4], 'Salary': [60000, 70000, 80000]})**

**# Performing an outer join on 'ID'**

**df\_joined = pd.merge(df1, df2, on='ID', how='outer')**

**print(df\_joined)**



**Section D: Reshaping and Input/Output (10 Questions)**

21. **Transposing DataFrames**   
 Transpose the rows and columns of the DataFrame df .

**Answer:**

import pandas as pd

# Sample DataFrame

df = pd.DataFrame({

'Name': ['Alice', 'Bob', 'Charlie'],

'Age': [25, 30, 35],

'Salary': [50000, 60000, 70000]

})

# Using .transpose() method to transpose

df\_transposed = df.transpose()

print(df\_transposed)

22. **Using T Attribute**   
 Use the T attribute to transpose the DataFrame df .

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

**# Sample DataFrame**

**df = pd.DataFrame({**

**'Name': ['Alice', 'Bob', 'Charlie'],**

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

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

**})**

**# Using the T attribute to transpose**

**df\_transposed = df.T**

**print(df\_transposed)**

23. **Writing to CSV**   
 Save the DataFrame df to a file called output.csv .

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

**# Sample DataFrame**

**df = pd.DataFrame({**

**'Name': ['Alice', 'Bob', 'Charlie'],**

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

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

**})**

**# Saving DataFrame to a CSV file**

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

**print("DataFrame saved to output.csv successfully!")**

24. **Writing to Excel**   
 Export the DataFrame df to an Excel file named output.xlsx .

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

**# Sample DataFrame**

**df = pd.DataFrame({**

**'Name': ['Alice', 'Bob', 'Charlie'],**

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

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

**})**

**# Saving DataFrame to an Excel file**

**df.to\_excel("output.xlsx", index=False)**

**print("DataFrame saved to output.xlsx successfully!")**

25. **Writing to JSON**   
 Convert the DataFrame df to a JSON file named output.json .

**Answer: import pandas as pd**

**# Sample DataFrame**

**df = pd.DataFrame({**

**'Name': ['Alice', 'Bob', 'Charlie'],**

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

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

**})**

**# Saving DataFrame to a JSON file**

**df.to\_json("output.json", orient="records", indent=4)**

**print("DataFrame saved to output.json successfully!")**

26. **Rendering DataFrame as HTML**   
 Convert the DataFrame df to an HTML table and save it as output.html .

**Answer: import pandas as pd**

**# Sample DataFrame**

**df = pd.DataFrame({**

**'Name': ['Alice', 'Bob', 'Charlie'],**

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

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

**})**

**# Saving DataFrame to an HTML file**

**df.to\_html("output.html", index=False)**

**print("DataFrame saved to output.html successfully!")**

27. **Loading CSV File**   
 Load a CSV file named student\_data.csv into a DataFrame.

**Answer:**

import pandas as pd

# Load the CSV file into a DataFrame

df = pd.read\_csv("student\_data.csv")

# Display the first few rows

print(df.head())

28. **Loading Excel File**   
 Load an Excel file named sales\_data.xlsx into a DataFrame.

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

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

**df = pd.read\_excel("sales\_data.xlsx")**

**# Display the first few rows**

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

29. **Saving a DataFrame as CSV**   
Save the DataFrame df to a CSV file called employees.csv , including only the 'Name' and 'Salary' columns.

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

**# Sample DataFrame**

**df = pd.DataFrame({**

**'Name': ['Alice', 'Bob', 'Charlie'],**

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

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

**})**

**# Saving only 'Name' and 'Salary' columns to a CSV file**

**df[['Name', 'Salary']].to\_csv("employees.csv", index=False)**

**print("DataFrame saved to employees.csv successfully!")**

30. **Saving a DataFrame as JSON with Specific Columns**   
Save the DataFrame df as a JSON file, but only include the 'Name' and 'Department' columns.

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

**# Sample DataFrame**

**df = pd.DataFrame({**

**'Name': ['Alice', 'Bob', 'Charlie'],**

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

**'Department': ['HR', 'IT', 'Finance'],**

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

**})**

**# Saving only 'Name' and 'Department' columns to a JSON file**

**df[['Name', 'Department']].to\_json("employees.json", orient="records", indent=4)**

**print("DataFrame saved to employees.json successfully!")**

