# 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

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

2. Mapping Values in Series

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

with False .

Answer:

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

3. Lowercase Strings

Convert all strings in the 'Names' column to lowercase.

Answer:

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

4. Uppercase Strings

Convert the 'City' column to uppercase.

Answer:

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

5. Splitting Strings

Split the 'FullName' column into 'FirstName' and 'LastName' using a space as the delimiter.

Answer:

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

6. String Contains

Filter rows where the 'Email' column contains '@gmail.com'.

Answer:

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

7. Replacing String Patterns

Use str.replace()  to replace the domain in all emails from '@example.com' to '@newdomain.com'.

Answer:

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

## 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:

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

9. Aggregating Data

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

Answer:

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

10. Aggregate Multiple Functions

Use aggregate()  to calculate both the sum and count of the 'Marks' column.

Answer:

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

11. Filtering with isin()

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

Answer:

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

12. Grouping and Aggregating

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

group.

Answer:

marks\_sum\_by\_gender = df.groupby('Gender')['Marks'].sum()

13. Multiple Aggregations on Multiple Columns

Perform multiple aggregations (min, max, mean) on the 'Age' and 'Salary' columns.

Answer:

multiple\_agg = df.agg({'Age': ['min', 'max', 'mean'], 'Salary': ['min', 'max', 'mean']})

14. Grouping and Counting

Group by 'City' and count the number of entries in each city.

Answer:

count\_by\_city = df.groupby('City').size()

15. Using apply() with Groupby

Apply a custom function to find the range (max-min) of the 'Salary' column for each department.

Answer:

salary\_range\_by\_department = df.groupby('Department')['Salary'].apply(lambda x: x.max() - x.min())

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

16. Concatenating DataFrames

Concatenate two DataFrames df1 and df2 along rows.

Answer:

df\_concat = pd.concat([df1, df2], axis=0)

17. Merging DataFrames

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

Answer:

df\_merge = df1.merge(df2, on='ID')

18. Merging with Different Keys

Merge DataFrames on different column names: 'df1' has 'EmployeeID' and 'df2' has 'ID'

Answer:

df\_merge\_diff\_keys = df1.merge(df2, left\_on='EmployeeID', right\_on='ID')

19. Concatenating Along Columns

Concatenate two DataFrames df1 and df2 along columns.

Answer:

df\_concat\_cols = pd.concat([df1, df2], axis=1)

20. Joining DataFrames

Join df1  and df2  on the 'ID' column with an outer join.

Answer:

df\_join = df1.join(df2.set\_index('ID'), on='ID', how='outer')

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

21. Transposing DataFrames

Transpose the rows and columns of the DataFrame df .

Answer:

df\_transposed = df.T

22. Using T Attribute

Use the T attribute to transpose the DataFrame df .

Answer:

df\_transposed\_T = df.T

23. Writing to CSV

Save the DataFrame df to a file called output.csv .

Answer:

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

24. Writing to Excel

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

Answer:

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

25. Writing to JSON

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

Answer:

df.to\_json('output.json', orient='records')

26. Rendering DataFrame as HTML

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

Answer:

df.to\_html('output.html')

27. Loading CSV File

Load a CSV file named student\_data.csv into a DataFrame.

Answer:

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

28. Loading Excel File

Load an Excel file named sales\_data.xlsx into a DataFrame.

Answer:

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

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:

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

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:

df[['Name', 'Department']].to\_json('output.json', orient='records')