# 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 &#39;Price&#39; 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 &#39;Yes&#39; in the &#39;Passed&#39; column with True and &#39;No&#39;

with False .

Answer:

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

3. Lowercase Strings

Convert all strings in the &#39;Names&#39; column to lowercase.

Answer:

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

4. Uppercase Strings

Convert the &#39;City&#39; column to uppercase.

Answer:

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

5. Splitting Strings

Split the &#39;FullName&#39; column into &#39;FirstName&#39; and &#39;LastName&#39; using a space as the delimiter.

Answer:

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

6. String Contains

Filter rows where the &#39;Email&#39; column contains

&#39;@gmail.com&#39;.

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 &#39;@example.com&#39; to

&#39;@newdomain.com&#39;.

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 &#39;Department&#39; 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 &#39;Sales&#39; 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 &#39;Marks&#39; column.

Answer:

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

11. Filtering with isin()

Filter rows where the &#39;City&#39; column is either &#39;New York&#39; or &#39;Los Angeles&#39; using isin() .

Answer:

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

12. Grouping and Aggregating

Group the DataFrame by &#39;Gender&#39; and calculate the sum of the &#39;Marks&#39; column for each

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 &#39;Age&#39; and &#39;Salary&#39; columns.

Answer:

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

14. Grouping and Counting

Group by &#39;City&#39; 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 &#39;Salary&#39; 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 &#39;ID&#39; column.

Answer:

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

18. Merging with Different Keys

Merge DataFrames on different column names: &#39;df1&#39; has &#39;EmployeeID&#39; and &#39;df2&#39; has

&#39;ID&#39;.

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 &#39;ID&#39; 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 &#39;Name&#39; and

&#39;Salary&#39; 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 &#39;Name&#39; and &#39;Department&#39;

columns.

Answer:

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