
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
```

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

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

```
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
```

```
data = {'Student': ['Alice', 'Bob', 'Charlie'], 'Passed': ['Yes', 'No', 'Yes']}
```

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

```
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
```

```
data = {'Names': ['Alice', 'BOB', 'Charlie'], 'Age': [25, 30, 22]}  
df = pd.DataFrame(data)  
df['Names'] = df['Names'].str.lower()  
print(df)
```

4. Uppercase Strings

Convert the 'City' column to uppercase.

Answer:

```
import pandas as pd  
data = {'City': ['Agra', 'Banglore', 'Chicago']}  
df = pd.DataFrame(data)  
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

data = {'Full Name': ['Alice Johnson', 'Bob Smith', 'Charlie Brown']}

df = pd.DataFrame(data)

df[['First Name', 'Last Name']] = df['Full Name'].str.split(' ', expand=True)

print(df)
```

6. String Contains

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

Answer:

```
import pandas as pd

data = {'Name': ['Alice', 'Bob', 'Charlie'],
        'Email': ['alice@gmail.com', 'bob@yahoo.com', 'charlie@gmail.com']}

df = pd.DataFrame

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

data = {'Name': ['Alice', 'Bob', 'Charlie'],
        'Email': ['alice@example.com', 'bob@example.com', 'charlie@example.com']}

df = pd.DataFrame(data)

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

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

data = {'Employee': ['Alice', 'Bob', 'Charlie', 'David', 'Eve'],
        'Department': ['HR', 'IT', 'IT', 'HR', 'Finance'],
        'Salary': [50000, 70000, 80000, 55000, 60000]}

df = pd.DataFrame(data)

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

print(mean_salaries)
```

9. Aggregating Data

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

Answer:

```
import pandas as pd

data = {'Employee': ['Alice', 'Bob', 'Charlie', 'David', 'Eve'],
        'Sales': [5000, 7000, 8000, 5500, 6000]}

df = pd.DataFrame(data)

sales_agg = df['Sales'].agg(['mean', 'max'])

print(sales_agg)
```

10. Aggregate Multiple Functions

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

Answer:

```
import pandas as pd
```

```
data = {'Student': ['Alice', 'Bob', 'Charlie', 'David', 'Eve'],  
        'Marks': [85, 90, 78, 92, 88]}  
  
df = pd.DataFrame(data)  
  
marks_agg = df['Marks'].aggregate(['sum', 'count'])  
  
print(marks_agg)
```

11. Filtering with `isin()`

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

Answer:

```
import pandas as pd

data = {'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Eve'],
        'City': ['New York', 'Los Angeles', 'Chicago', 'New York', 'Houston']}

df = pd.DataFrame(data)

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

data = {'Student': ['Alice', 'Bob', 'Charlie', 'David', 'Eve'],
        'Gender': ['Female', 'Male', 'Male', 'Male', 'Female'],
        'Marks': [85, 90, 78, 92, 88]}

df = pd.DataFrame(data)

marks_sum = df.groupby('Gender')['Marks'].sum()

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

data = {'Employee': ['Alice', 'Bob', 'Charlie', 'David', 'Eve'],
        'Age': [25, 30, 35, 40, 28],
        'Salary': [50000, 60000, 70000, 80000, 55000]}

df = pd.DataFrame(data)

agg_result = df[['Age', 'Salary']].agg(['min', 'max', 'mean'])
```

```
print(agg_result)
```

14. Grouping and Counting

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

Answer:

```
import pandas as pd

data = {'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Eve', 'Frank'],
        'City': ['New York', 'Los Angeles', 'New York', 'Chicago', 'Los Angeles', 'Chicago']}

df = pd.DataFrame(data)

city_counts = df.groupby('City').size()

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

data = {'Employee': ['Alice', 'Bob', 'Charlie', 'David', 'Eve'],
        'Department': ['HR', 'IT', 'IT', 'HR', 'Finance'],
        'Salary': [50000, 70000, 80000, 55000, 60000]}

df = pd.DataFrame(data)

salary_range = df.groupby('Department')['Salary'].agg(lambda x: x.max() - x.min())

print(salary_range)
```

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

16. Concatenating DataFrames

Concatenate two DataFrames df1 and df2 along rows.

Answer:

```
import pandas as pd
df1 = pd.DataFrame({'ID': [1, 2], 'Name': ['Alice', 'Bob']})
df2 = pd.DataFrame({'ID': [3, 4], 'Name': ['Charlie', 'David']})
df_combined = pd.concat([df1, df2], axis=0)
print(df_combined)
```

17. Merging DataFrames

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

Answer:

```
import pandas as pd

df1 = pd.DataFrame({'ID': [1, 2], 'Name': ['Alice', 'Bob']})
df2 = pd.DataFrame({'ID': [3, 4], 'Name': ['Charlie', 'David']})

df_combined = pd.merge(df1, df2, axis=1)

print(df_combined)
```

18. Merging with Different Keys

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

Answer:

```
import pandas as pd

df1 = pd.DataFrame({'EmployeeID': [1, 2, 3], 'Name': ['Alice', 'Bob', 'Charlie']})
df2 = pd.DataFrame({'ID': [1, 2, 3], 'Department': ['HR', 'IT', 'Finance']})

df_merged = pd.merge(df1, df2, left_on='EmployeeID', right_on='ID')

print(df_merged)
```

19. Concatenating Along Columns

Concatenate two DataFrames df1 and df2 along columns.

Answer:

```
import pandas as pd
```



```
df1 = pd.DataFrame({'ID': [1, 2], 'Name': ['Alice', 'Bob']})
df2 = pd.DataFrame({'ID': [3, 4], 'Name': ['Charlie', 'David']})
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
```

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

```
df2 = pd.DataFrame({'ID': [2, 3, 4], 'Department': ['IT', 'Finance', 'HR']})
```

```
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
```

```
df = pd.DataFrame({
    'Name': ['Alice', 'Bob', 'Charlie'],
    'Age': [25, 30, 35],
    'Salary': [50000, 60000, 70000]
})
```

```
df_transposed = df.T
```

```
print(df_transposed)
```

22. Using T Attribute

Use the T attribute to transpose the DataFrame df .

Answer:

```
import pandas as pd

df = pd.DataFrame({
    'Name': ['Alice', 'Bob', 'Charlie'],
    'Age': [25, 30, 35],
    'Salary': [50000, 60000, 70000]
})

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

df = pd.DataFrame({
    'Name': ['Alice', 'Bob', 'Charlie'],
    'Age': [25, 30, 35],
    'Salary': [50000, 60000, 70000]
})

df.to_csv('output.csv', index=False)

print("DataFrame saved as output.csv")
```

24. Writing to Excel

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

Answer:

```
import pandas as pd

df = pd.DataFrame({
    'Name': ['Alice', 'Bob', 'Charlie'],
    'Age': [25, 30, 35],
    'Salary': [50000, 60000, 70000]
})

df.to_excel('output.xlsx', index=False)

print("DataFrame saved as output.xlsx")
```

25. Writing to JSON

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

Answer:

```
import pandas as pd

df = pd.DataFrame({
    'Name': ['Alice', 'Bob', 'Charlie'],
    'Age': [25, 30, 35],
    'Salary': [50000, 60000, 70000]
})

df.to_csv('output.csv', index=False)

print("DataFrame saved as output.json")
```

26. Rendering DataFrame as HTML

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

Answer:

```
import pandas as pd

df = pd.DataFrame({
    'Name': ['Alice', 'Bob', 'Charlie'],
    'Age': [25, 30, 35],
    'Salary': [50000, 60000, 70000]
})

df.to_csv('output.csv', index=False)

print("DataFrame saved as output.html")
```

27. Loading CSV File

Load a CSV file named `student_data.csv` into a DataFrame.

Answer:

```
import pandas as pd

df = pd.read_csv('student_data.csv')

print(df.head())
```

28. Loading Excel File

Load an Excel file named `sales_data.xlsx` into a `DataFrame`.

Answer:

```
import pandas as pd

df = pd.read_excel("sales_data.xlsx")

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

df = pd.DataFrame({
    'Name': ['Alice', 'Bob', 'Charlie'],
    'Age': [25, 30, 35],
    'Salary': [50000, 60000, 70000]
})

df[['Name', 'Salary']].to_csv('employees.csv', index=False)

print("DataFrame saved as employees.csv")
```

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

df = pd.DataFrame({
    'Name': ['Alice', 'Bob', 'Charlie'],
```

```
'Department': ['HR', 'IT', 'Finance'],  
'Salary': [50000, 60000, 70000]  
)  
df[['Name', 'Department']].to_json('employees.json', orient='records', indent=4)  
print("DataFrame saved as employees.json")
```
