**CODING ASSESSMENT PYTHON(PANDAS) :**

**29-07-2025**

1.CREATED A ‘your\_data.csv’ FILE AND USED IT.

id,name,age,city

1,Lokanya,28,New York

2,Abdul Basith,28,Los Angeles

3,Gayathri ,45,Chicago

4,Ajay,29,Houston

5,Sai Vignesh,32,Phoenix

**2.Handling missing values**

import pandas as pd

# Load your data

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

# Check for missing values

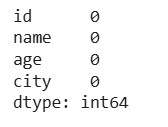
print(df.isnull().sum())

# Drop rows with missing values

df\_cleaned = df.dropna()

# Alternatively, fill missing values

df\_filled = df.fillna(value={'column\_name': 'default\_value'})



**3.Removing duplicates**

# Remove duplicate rows

df\_no\_duplicates = df\_cleaned.drop\_duplicates()

**4.Data type conversion**

import pandas as pd

# Load your data

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

# Check the DataFrame

print("Original DataFrame:")

print(df)

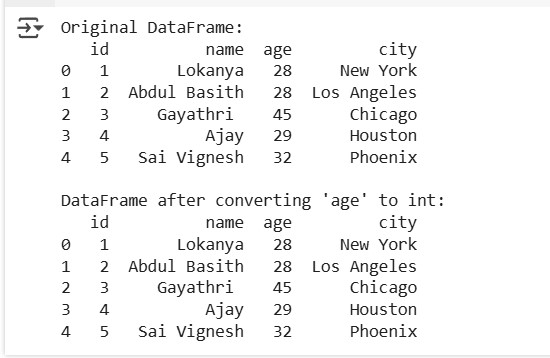
# Convert the 'age' column to integer type

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

# Check the DataFrame after conversion

print("\nDataFrame after converting 'age' to int:")

print(df)



5.**Inner Join**

An inner join returns only the rows with matching values in both DataFrames.

import pandas as pd

# Sample DataFrames

data1 = {'key': ['K0', 'K1', 'K2'], 'value1': ['A', 'B', 'C']}

data2 = {'key': ['K1', 'K2', 'K3'], 'value2': ['D', 'E', 'F']}

df1 = pd.DataFrame(data1)

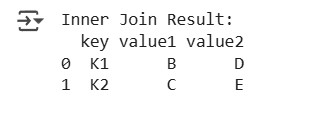
df2 = pd.DataFrame(data2)

# Inner Join

result\_inner = pd.merge(df1, df2, on='key', how='inner')

print("Inner Join Result:")

print(result\_inner)



6. **Left Join**

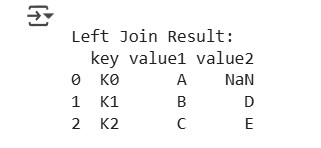
A left join returns all rows from the left DataFrame and the matched rows from the right DataFrame.

# Left Join

result\_left = pd.merge(df1, df2, on='key', how='left')

print("\nLeft Join Result:")

print(result\_left)



7. **Right Join**

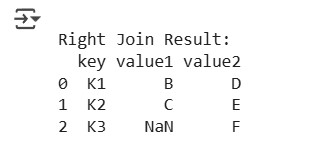
A right join returns all rows from the right DataFrame and the matched rows from the left DataFrame.

# Right Join

result\_right = pd.merge(df1, df2, on='key', how='right')

print("\nRight Join Result:")

print(result\_right)



8. **Outer Join**

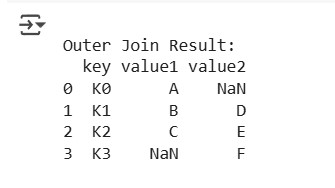
An outer join returns all rows from both DataFrames, filling in NaNs for missing matches.

# Outer Join

result\_outer = pd.merge(df1, df2, on='key', how='outer')

print("\nOuter Join Result:")

print(result\_outer)



9. **Using .join() Method**

The **.join()** method is useful for joining DataFrames based on their indices.

# Sample DataFrames with indices

df1.set\_index('key', inplace=True)

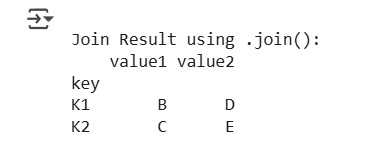
df2.set\_index('key', inplace=True)

# Join using .join()

result\_join = df1.join(df2, how='inner', lsuffix='\_left', rsuffix='\_right')

print("\nJoin Result using .join():")

print(result\_join)



10. **Concatenating DataFrames**

You can also concatenate DataFrames vertically or horizontally.

# Concatenating DataFrames

result\_concat = pd.concat([df1, df2], axis=0, ignore\_index=True)

print("\nConcatenated DataFrame:")

print(result\_concat)

