**LAB - 4**

**Name:Rhythm Kachhwaha**

**SAP:500107218**

**Batch - 4 AIML**

**Q1. Loading and Inspecting Data:**

**· Load various data formats (CSV, Excel, JSON) into Pandas DataFrames.**

**· Explore DataFrame attributes like shape, columns, dtypes, head, tail, info, describe.**

**· Practice selecting columns and rows using different methods (indexing, slicing, loc, iloc).**

**1.**

import pandas as pd

# Create sample data

data = {

    'Name': ['Rhythm', 'Anmol', 'Sid', 'Maith', 'Charvi','Mayank'],

    'Age': [19, 20, 21, 18, 12, 20],

    'City': ['Jodhpur', 'Jaipur', 'Meerut', 'Mumbai', 'Jodhpur','Roorkee']

}

# Create a DataFrame

df = pd.DataFrame(data)

# Save as CSV

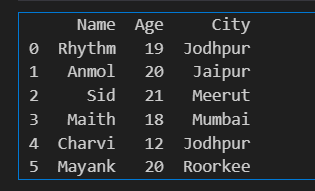
df.to\_csv('sample\_data.csv', index=False)

# Load the CSV file back

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

print(df\_csv)

Output:



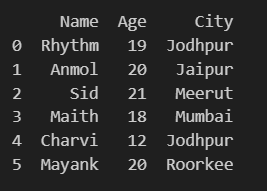
# Save as Excel

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

# Load the Excel file back

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

print(df\_excel)



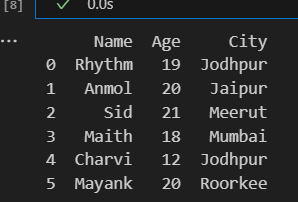
# Save as JSON

df.to\_json('sample\_data.json', orient='records', lines=True)

# Load the JSON file back

df\_json = pd.read\_json('sample\_data.json', lines=True)

print(df\_json)



2.

# Shape of the DataFrame

print("Shape:", df.shape)

# Columns of the DataFrame

print("Columns:", df.columns)

# Data types of each column

print("Data types:\n", df.dtypes)

# First few rows of the DataFrame

print("First 5 rows:\n", df.head())

# Last few rows of the DataFrame

print("Last 5 rows:\n", df.tail())

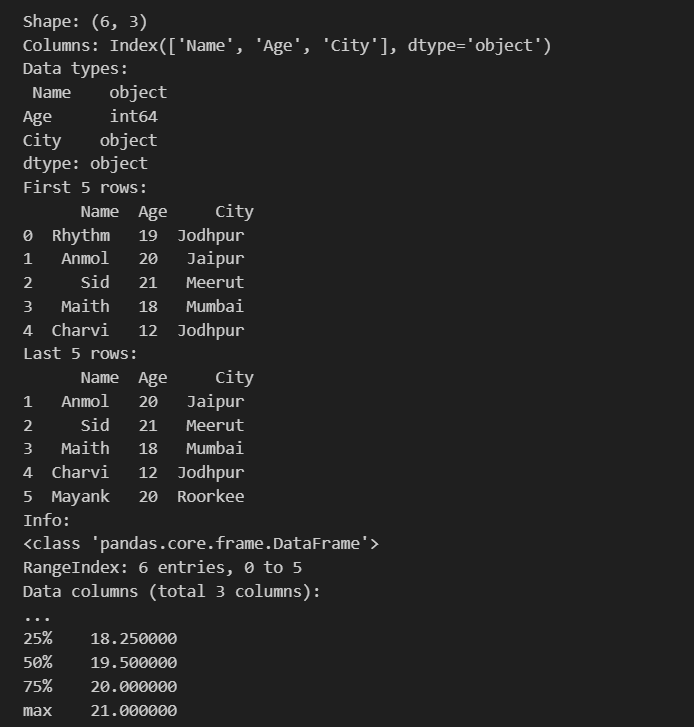
# Information about the DataFrame

print("Info:")

df.info()

# Summary statistics of the DataFrame

print("Describe:\n", df.describe())



3.

# Select a single column

print("Column 'Name':\n", df['Name'])

# Select specific rows using iloc (indexing)

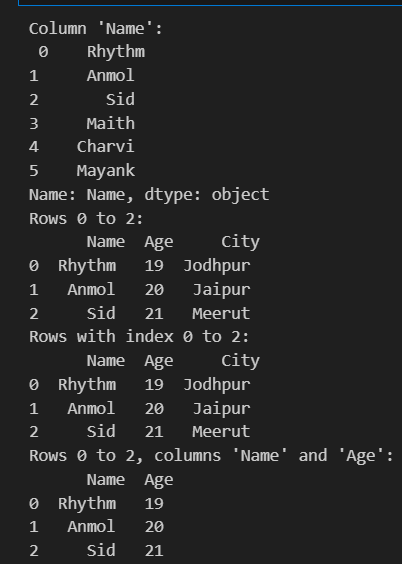
print("Rows 0 to 2:\n", df.iloc[0:3])

# Select specific rows using loc (label-based)

print("Rows with index 0 to 2:\n", df.loc[0:2])

# Select specific rows and columns

print("Rows 0 to 2, columns 'Name' and 'Age':\n", df.loc[0:2, ['Name', 'Age']])



**Q2. Data Cleaning and Preparation**

**Identify missing values using isnull and isna.**

**Handle missing values using fillna, dropna, interpolation.**

**Apply scaling techniques (min-max, z-score) to numerical columns.**

**Create dummy variables for categorical columns.**

import pandas as pd

import numpy as np

# Create sample data with missing values

data = {

    'Name': ['Rhythm', 'Anmol', 'Sid', 'Maith', np.nan],

    'Age': [19, 20, np.nan, 18, 12],

    'City': ['Jodhpur', 'Jaipur', 'Meerut', np.nan, 'Jodhpur'],

    'Salary': [50000, 60000, np.nan, 80000, 70000]

}

df = pd.DataFrame(data)

# Identify missing values

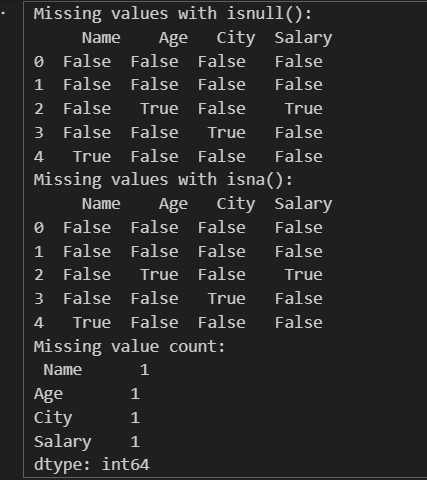
print("Missing values with isnull():\n", df.isnull())

# Alternatively, use isna()

print("Missing values with isna():\n", df.isna())

# Count the number of missing values in each column

print("Missing value count:\n", df.isnull().sum())



# Fill missing values with a constant (e.g., 0)

df\_filled = df.fillna(0)

print("Filled with 0:\n", df\_filled)

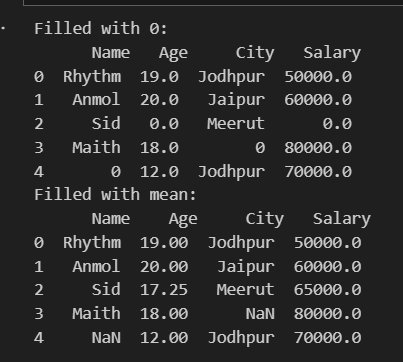
# Fill missing values with the mean of the column

df\_filled\_mean = df.copy()

df\_filled\_mean['Age'] = df['Age'].fillna(df['Age'].mean())

df\_filled\_mean['Salary'] = df['Salary'].fillna(df['Salary'].mean())

print("Filled with mean:\n", df\_filled\_mean)



# Drop rows with any missing values

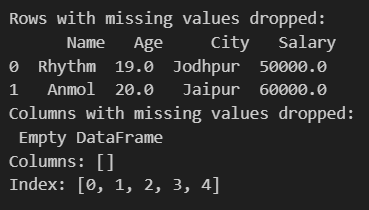
df\_dropped = df.dropna()

print("Rows with missing values dropped:\n", df\_dropped)

# Drop columns with missing values

df\_dropped\_cols = df.dropna(axis=1)

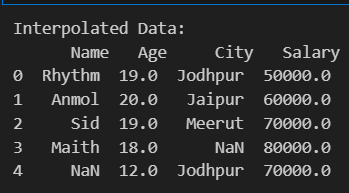
print("Columns with missing values dropped:\n", df\_dropped\_cols)



# Interpolate missing values

df\_interpolated = df.interpolate()

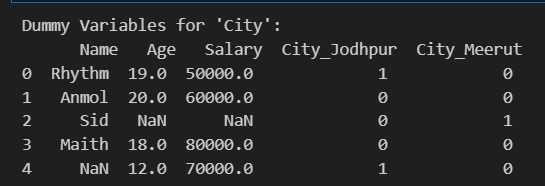
print("Interpolated Data:\n", df\_interpolated)



# Convert categorical 'City' column to dummy variables

df\_with\_dummies = pd.get\_dummies(df, columns=['City'], drop\_first=True)

print("Dummy Variables for 'City':\n", df\_with\_dummies)



**Q3. Aggregation and Grouping:**

**Calculate summary statistics (mean, median, count, etc.) using groupby.**

**Create pivot tables for data summarization.**

**Combine DataFrames using concat, merge, and join.**

**Practice different join types (inner, outer, left, right).**

import pandas as pd

data = {

    'Name': ['Rhythm', 'Anmol', 'Sid', 'Maith', 'Charvi','Mayank','Harsh'],

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

    'Age': [19, 20, 21, 18, 12, 20, 23],

    'Salary': [50000, 60000, 55000, 80000, 70000, 65000, 90000]

}

df = pd.DataFrame(data)

# Group by 'Department' and calculate the mean salary

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

print("Mean salary by department:\n", mean\_salary\_by\_dept)

# Group by 'Department' and calculate multiple summary statistics

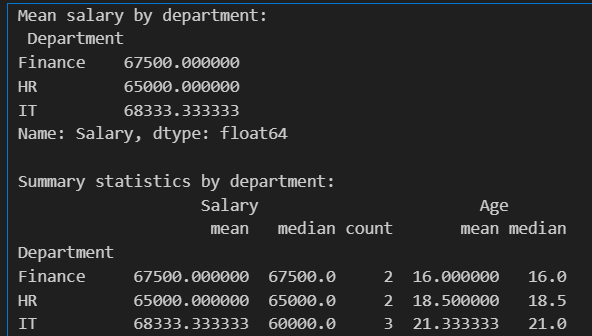
summary\_stats\_by\_dept = df.groupby('Department').agg({

    'Salary': ['mean', 'median', 'count'],

    'Age': ['mean', 'median']

})

print("\nSummary statistics by department:\n", summary\_stats\_by\_dept)



# Pivot table to summarize mean salary by 'Department'

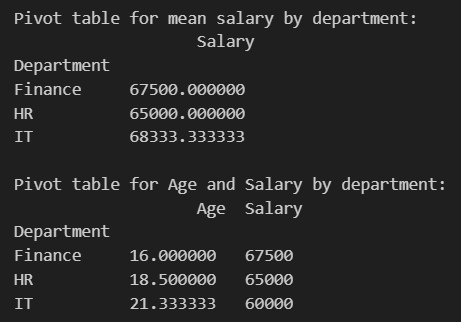
pivot\_salary = pd.pivot\_table(df, values='Salary', index='Department', aggfunc='mean')

print("\nPivot table for mean salary by department:\n", pivot\_salary)

# Pivot table to summarize both 'Age' and 'Salary' by 'Department'

pivot\_summary = pd.pivot\_table(df, values=['Age', 'Salary'], index='Department', aggfunc={'Age': 'mean', 'Salary': 'median'})

print("\nPivot table for Age and Salary by department:\n", pivot\_summary)



df1 = pd.DataFrame({'A': ['A0', 'A1', 'A2'], 'B': ['B0', 'B1', 'B2']})

df2 = pd.DataFrame({'A': ['A3', 'A4', 'A5'], 'B': ['B3', 'B4', 'B5']})

# Concatenate along rows (axis=0)

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

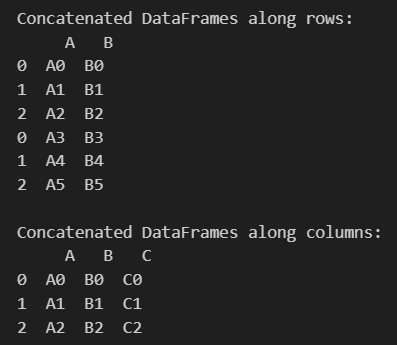
print("\nConcatenated DataFrames along rows:\n", df\_concat\_rows)

# Concatenate along columns (axis=1)

df3 = pd.DataFrame({'C': ['C0', 'C1', 'C2']})

df\_concat\_cols = pd.concat([df1, df3], axis=1)

print("\nConcatenated DataFrames along columns:\n", df\_concat\_cols)



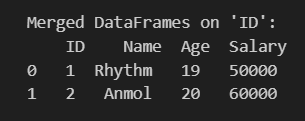
df\_left = pd.DataFrame({'ID': [1, 2, 3], 'Name': ['Rhythm', 'Anmol', 'Sid'], 'Age': [19,20,21]})

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

# Merge DataFrames on 'ID'

df\_merged = pd.merge(df\_left, df\_right, on='ID')

print("\nMerged DataFrames on 'ID':\n", df\_merged)



# Set 'ID' as index in both DataFrames

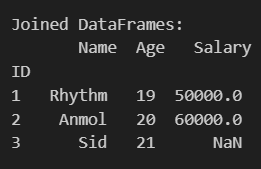
df\_left.set\_index('ID', inplace=True)

df\_right.set\_index('ID', inplace=True)

# Perform a join

df\_joined = df\_left.join(df\_right)

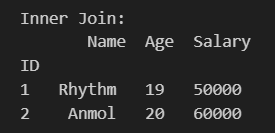
print("\nJoined DataFrames:\n", df\_joined)



#Inner Join

df\_inner = pd.merge(df\_left, df\_right, on='ID', how='inner')

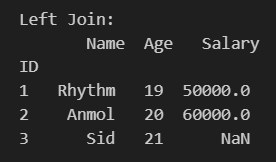
print("\nInner Join:\n", df\_inner)



#Left join

df\_left\_join = pd.merge(df\_left, df\_right, on='ID', how='left')

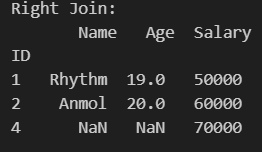
print("\nLeft Join:\n", df\_left\_join)



#Right join

df\_right\_join = pd.merge(df\_left, df\_right, on='ID', how='right')

print("\nRight Join:\n", df\_right\_join)



#Outer Join

df\_outer = pd.merge(df\_left, df\_right, on='ID', how='outer')

print("\nOuter Join:\n", df\_outer)

