**Name:** Deepanshu Miglani **SAP ID:**500106264  **BTech CSE AIML B4(NH)**

**Lab 4**

**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).**

**CODE:**

import pandas as pd

# Load the CSV file

df\_csv = pd.read\_csv(r'C:/Users/mehak/Downloads/data.csv', delimiter=',', skiprows=1)

# Explore DataFrame attributes

print("Shape:", df.shape)

print("Columns:", df.columns)

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

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

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

print("Info:\n", df.info())

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

# Selecting columns

name\_column = df['Name']

occupation\_salary\_columns = df[['Occupation', 'Salary']]

# Selecting rows

first\_row = df.iloc[0]

first\_five\_rows = df[0:5]

rows\_label\_0\_to\_4 = df.loc[0:4]

rows\_iloc\_0\_to\_4 = df.iloc[0:5]

print("\nSelected Columns:")

print(name\_column)

print(occupation\_salary\_columns)

print("\nSelected Rows:")

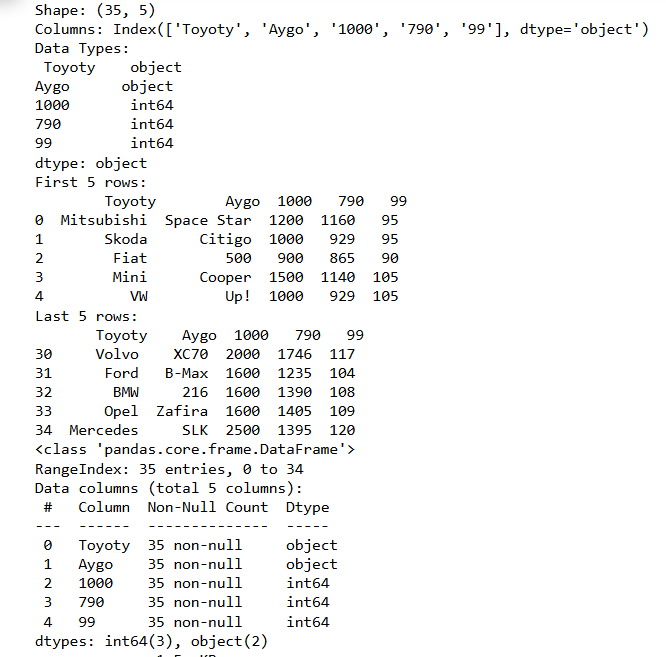
print("First row:\n", first\_row)

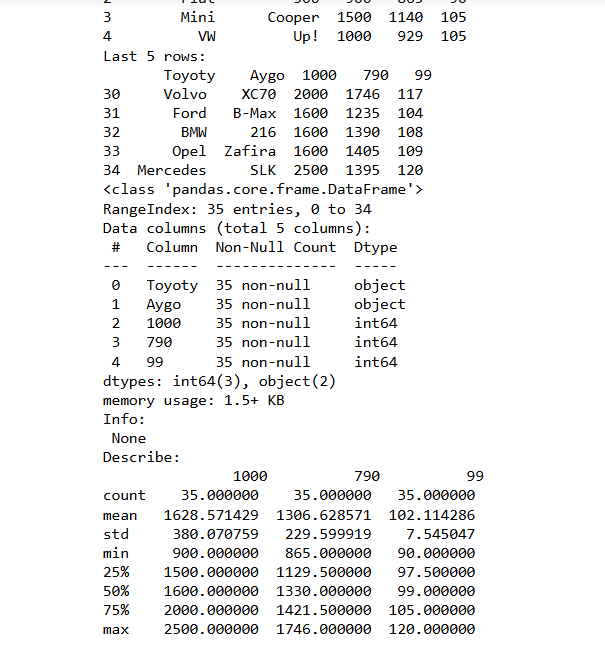
print("First five rows:\n", first\_five\_rows)

print("Rows 0 to 4 (loc):\n", rows\_label\_0\_to\_4)

print("Rows 0 to 4 (iloc):\n", rows\_iloc\_0\_to\_4)

**OUTPUT:**





**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.**

**CODE:**

import pandas as pd

from sklearn.preprocessing import MinMaxScaler, StandardScaler

# Load a sample CSV file (fixing skiprows and file path format)

df = pd.read\_csv(r'C:/Users/mehak/Downloads/data.csv', delimiter=',', skiprows=1)

# Data Cleaning and Preparation

# 1. Identify missing values using isnull() and isna()

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

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

# 2. Handle missing values

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

df\_filled\_constant = df.fillna(0)

# Fill missing values with the mean of the column (replace 'column\_name' with actual column)

# Ensure the column you want to fill with mean is numerical

if 'column\_name' in df.columns: # Replace 'column\_name' with the actual column

df['column\_name'] = df['column\_name'].fillna(df['column\_name'].mean())

# Drop rows with any missing values

df\_dropped = df.dropna()

# Interpolate missing values (linear interpolation for numerical data)

df\_interpolated = df.interpolate()

# 3. Scaling numerical columns (Min-Max and Z-score scaling)

# Selecting numerical columns

numerical\_cols = df.select\_dtypes(include=['float64', 'int64']).columns

# Apply Min-Max scaling (scales data to range [0, 1])

scaler\_min\_max = MinMaxScaler()

df\_min\_max\_scaled = df.copy()

df\_min\_max\_scaled[numerical\_cols] = scaler\_min\_max.fit\_transform(df[numerical\_cols])

print("Data after Min-Max Scaling:\n", df\_min\_max\_scaled.head())

# Apply Z-score scaling (Standardization: mean=0, std=1)

scaler\_zscore = StandardScaler()

df\_zscore\_scaled = df.copy()

df\_zscore\_scaled[numerical\_cols] = scaler\_zscore.fit\_transform(df[numerical\_cols])

print("Data after Z-score Scaling:\n", df\_zscore\_scaled.head())

# 4. Create dummy variables for categorical columns

# Identifying categorical columns

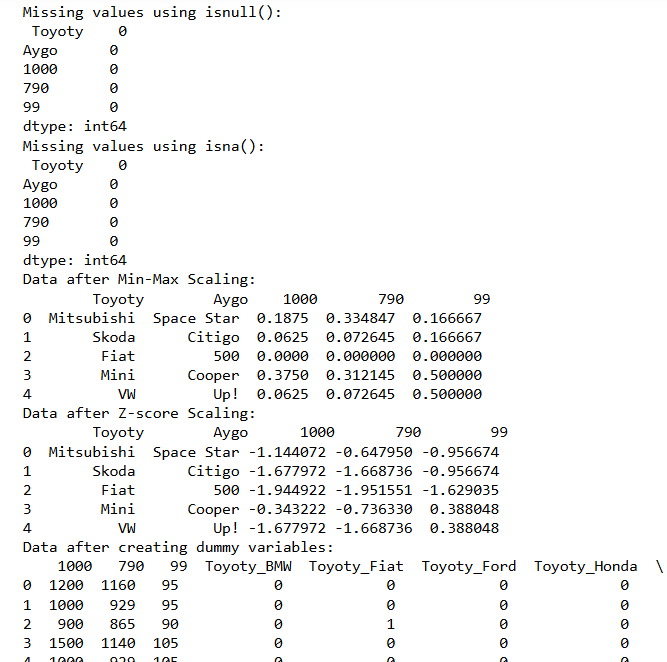
categorical\_cols = df.select\_dtypes(include=['object']).columns

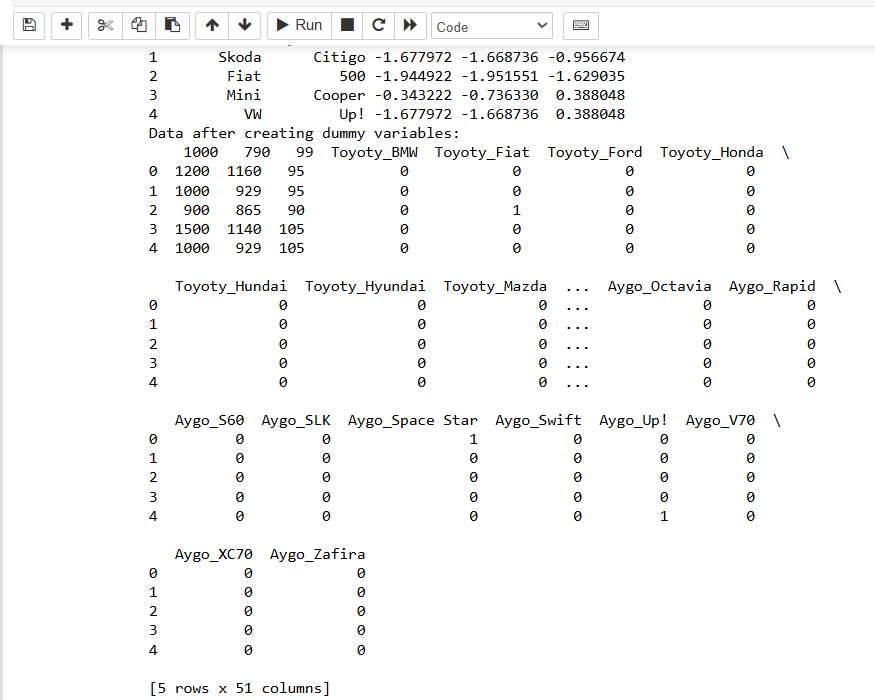
# Creating dummy variables (one-hot encoding)

df\_with\_dummies = pd.get\_dummies(df, columns=categorical\_cols, drop\_first=True)

print("Data after creating dummy variables:\n", df\_with\_dummies.head())

**OUTPUT:**





**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).**

**CODE:**

import pandas as pd

# Sample data to work with

data = {

'Category': ['A', 'B', 'A', 'C', 'B', 'C', 'A', 'C'],

'SubCategory': ['X', 'Y', 'X', 'Z', 'Y', 'Z', 'X', 'Z'],

'Sales': [200, 150, 300, 400, 500, 250, 100, 300],

'Profit': [20, 30, 50, 60, 70, 10, 20, 15],

'Quantity': [1, 2, 3, 4, 5, 2, 1, 2]

}

df = pd.DataFrame(data)

# Aggregation and Grouping

# 1. Calculate summary statistics using groupby

# Group by 'Category' and calculate mean, median, count

grouped = df.groupby('Category').agg({

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

'Profit': ['sum', 'mean'],

'Quantity': 'sum'

})

print("Grouped Summary Statistics:\n", grouped)

# 2. Create pivot tables for data summarization

# Create a pivot table summarizing 'Sales' and 'Profit' by 'Category' and 'SubCategory'

pivot\_table = pd.pivot\_table(df, values=['Sales', 'Profit'],

index=['Category'],

columns=['SubCategory'],

aggfunc='sum',

fill\_value=0)

print("\nPivot Table:\n", pivot\_table)

# 3. Combine DataFrames using concat, merge, and join

# Create a second DataFrame to demonstrate merging

data2 = {

'Category': ['A', 'B', 'C', 'D'],

'Region': ['North', 'South', 'East', 'West']

}

df2 = pd.DataFrame(data2)

# Concatenation (vertically stacking two DataFrames)

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

print("\nConcatenated DataFrame:\n", df\_concat)

# Merge DataFrames using different join types

# Inner join

df\_inner = pd.merge(df, df2, on='Category', how='inner')

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

# Outer join

df\_outer = pd.merge(df, df2, on='Category', how='outer')

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

# Left join

df\_left = pd.merge(df, df2, on='Category', how='left')

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

# Right join

df\_right = pd.merge(df, df2, on='Category', how='right')

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

# Joining using 'join' function (joining on indices)

df\_joined = df.set\_index('Category').join(df2.set\_index('Category'), how='left')

print("\nJoin on Indices:\n", df\_joined)

**OUTPUT:**

