	Marwadi University **Faculty of Engineering & Technology** **Department of Information and Communication Technology**	
Subject: Programming With Python (01CT1309)	**Aim:** Practical based on Data Loading, Storage and File Formats	
Experiment No: 22	**Date:**	**Enrollment No:92400133131**

GITHUB LINK:- https://github.com/farhan-web404/farhankaladiya.git

Aim: Practical based on Data Loading, Storage and File Formats

IDE: load, manipulate, and store data using Python (over reading and writing CSV, JSON, and Excel files)

Library Installation pip

install pandas openpyxl

Sample Data:

Create a folder for this experiment and add the following sample data files:

sample_data.csv (Name,Age,City

Alice,30,New York

Bob,25,Los Angeles

Charlie,35,Chicago)

sample_data.json ([

{"Name": "David", "Age": 28, "City": "San Francisco"},

{"Name": "Eve", "Age": 22, "City": "Seattle"}

])

sample_data.xlsx (you can create this using Excel with similar data**)**

Loading Data from CSV

Read the CSV file and perform basic data manipulation.

```
import pandas as pd

# Load data from CSV csv_file_path

= 'sample_data.csv' df_csv =

pd.read_csv(csv_file_path)


# Display the DataFrame

print("CSV Data:") print(df_csv)


# Basic data manipulation: Filter by age filtered_data

= df_csv[df_csv['Age'] > 30] print("\nFiltered Data

(Age > 30):") print(filtered_data)
```

Loading Data from JSON

Read the JSON file and manipulate the data.

```
# Load data from JSON json_file_path =

'sample_data.json' df_json =

pd.read_json(json_file_path)


# Display the DataFrame

print("\nJSON Data:") print(df_json)
```

```
# Basic data manipulation: Find the average age

average_age = df_json['Age'].mean() print("\nAverage

Age:", average_age)
```

Loading Data from Excel

Read the Excel file and display its contents.

```
# Load data from Excel excel_file_path =

'sample_data.xlsx' df_excel =

pd.read_excel(excel_file_path)


# Display the DataFrame

print("\nExcel Data:") print(df_excel)


# Basic data manipulation: Count the number of entries

entry_count = df_excel.shape[0] print("\nNumber of

entries in Excel file:", entry_count)
```

Writing Data to Different Formats

Save manipulated DataFrames to new files in different formats.

```
# Save filtered CSV data to a new file

filtered_data.to_csv('filtered_data.csv', index=False) print("\nFiltered

data saved to 'filtered_data.csv'.") # Save DataFrame to a new JSON
```

file df_json.to_json('new_data.json', orient='records', lines=True)

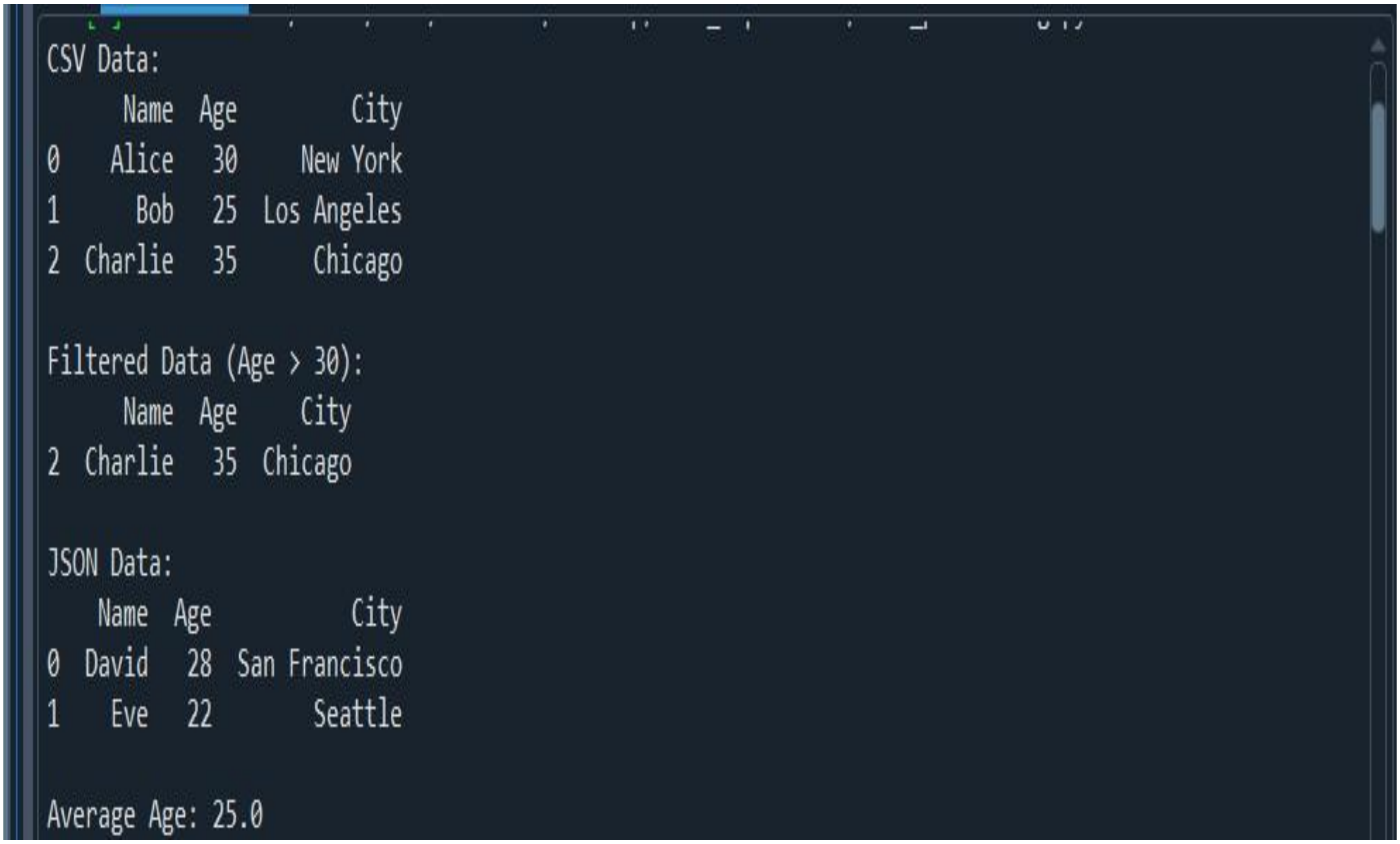
print("JSON data saved to 'new_data.json'.")

# Save DataFrame to a new Excel file

df_excel.to_excel('new_data.xlsx', index=False) print("Excel

data saved to 'new_data.xlsx'.")

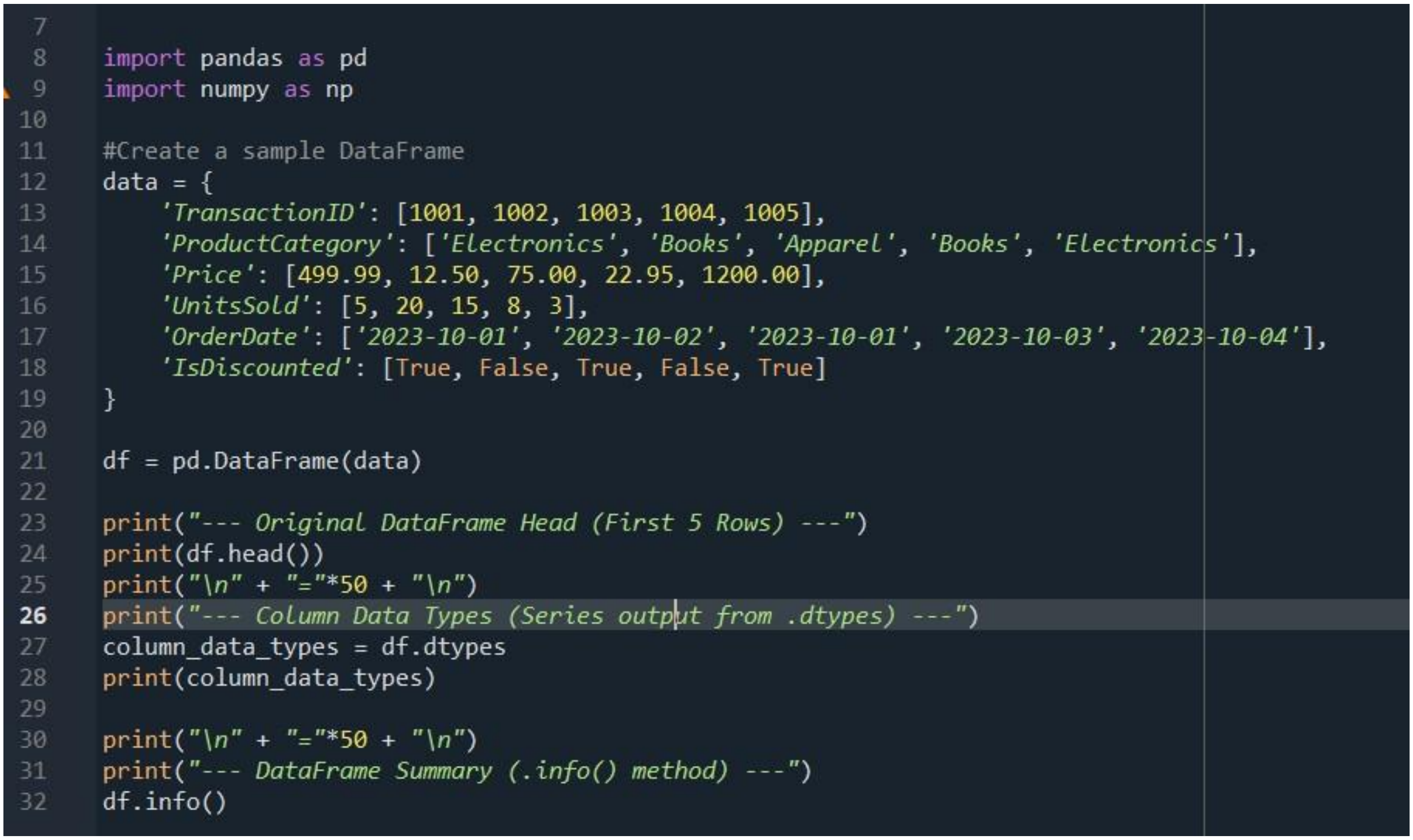
output:-



```
CSV Data:
      Name  Age         City
0    Alice   30     New York
1      Bob   25  Los Angeles
2  Charlie   35      Chicago

Filtered Data (Age > 30):
      Name  Age     City
2  Charlie   35  Chicago

JSON Data:
    Name  Age           City
0  David   28  San Francisco
1    Eve   22        Seattle

Average Age: 25.0
```

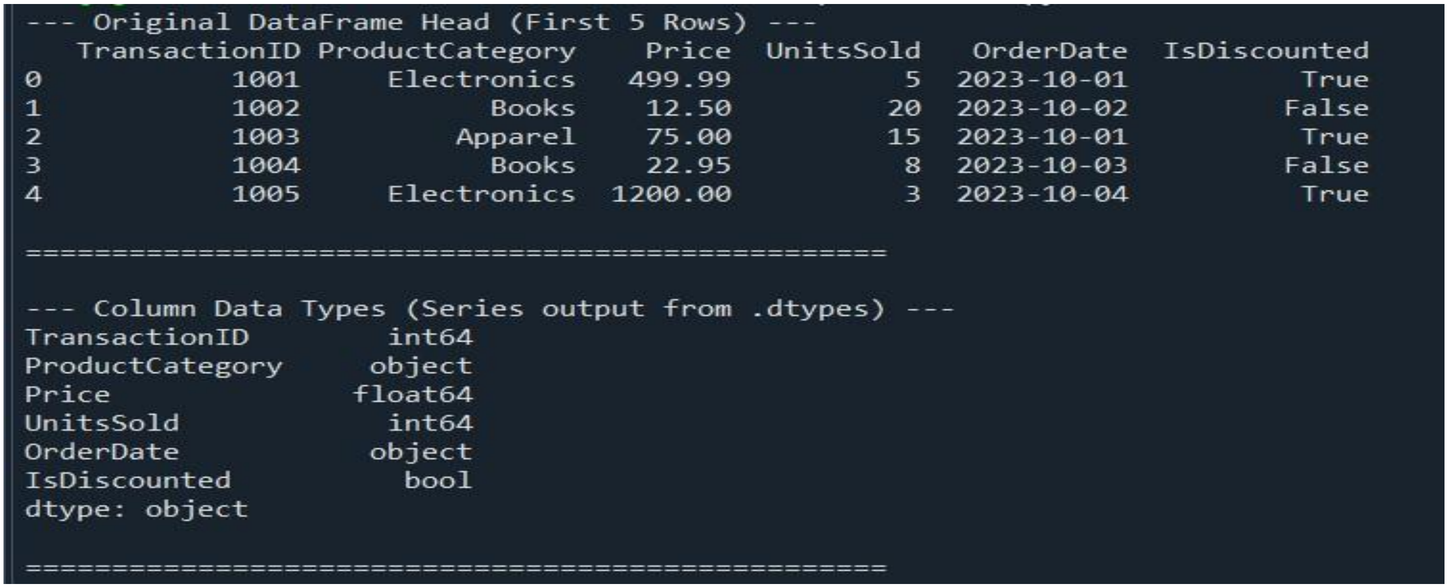
Post Lab:

Write a code snippet to check the data types of each column in a DataFrame. Code:-

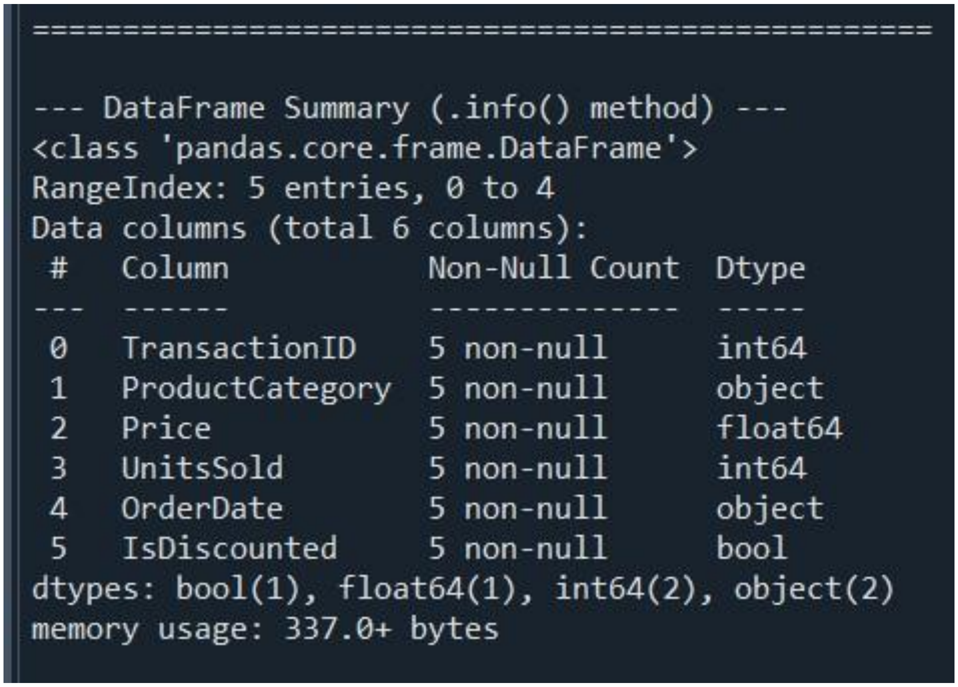


```
import pandas as pd
import numpy as np

#Create a sample DataFrame
data = {
    'TransactionID': [1001, 1002, 1003, 1004, 1005],
    'ProductCategory': ['Electronics', 'Books', 'Apparel', 'Books', 'Electronics'],
    'Price': [499.99, 12.50, 75.00, 22.95, 1200.00],
    'UnitsSold': [5, 20, 15, 8, 3],
    'OrderDate': ['2023-10-01', '2023-10-02', '2023-10-01', '2023-10-03', '2023-10-04'],
    'IsDiscounted': [True, False, True, False, True]
}

df = pd.DataFrame(data)

print("--- Original DataFrame Head (First 5 Rows) ---")
print(df.head())
print("\n" + "="*50 + "\n")
print("--- Column Data Types (Series output from .dtypes) ---")
column_data_types = df.dtypes
print(column_data_types)

print("\n" + "="*50 + "\n")
print("--- DataFrame Summary (.info() method) ---")
df.info()
```

Output:-



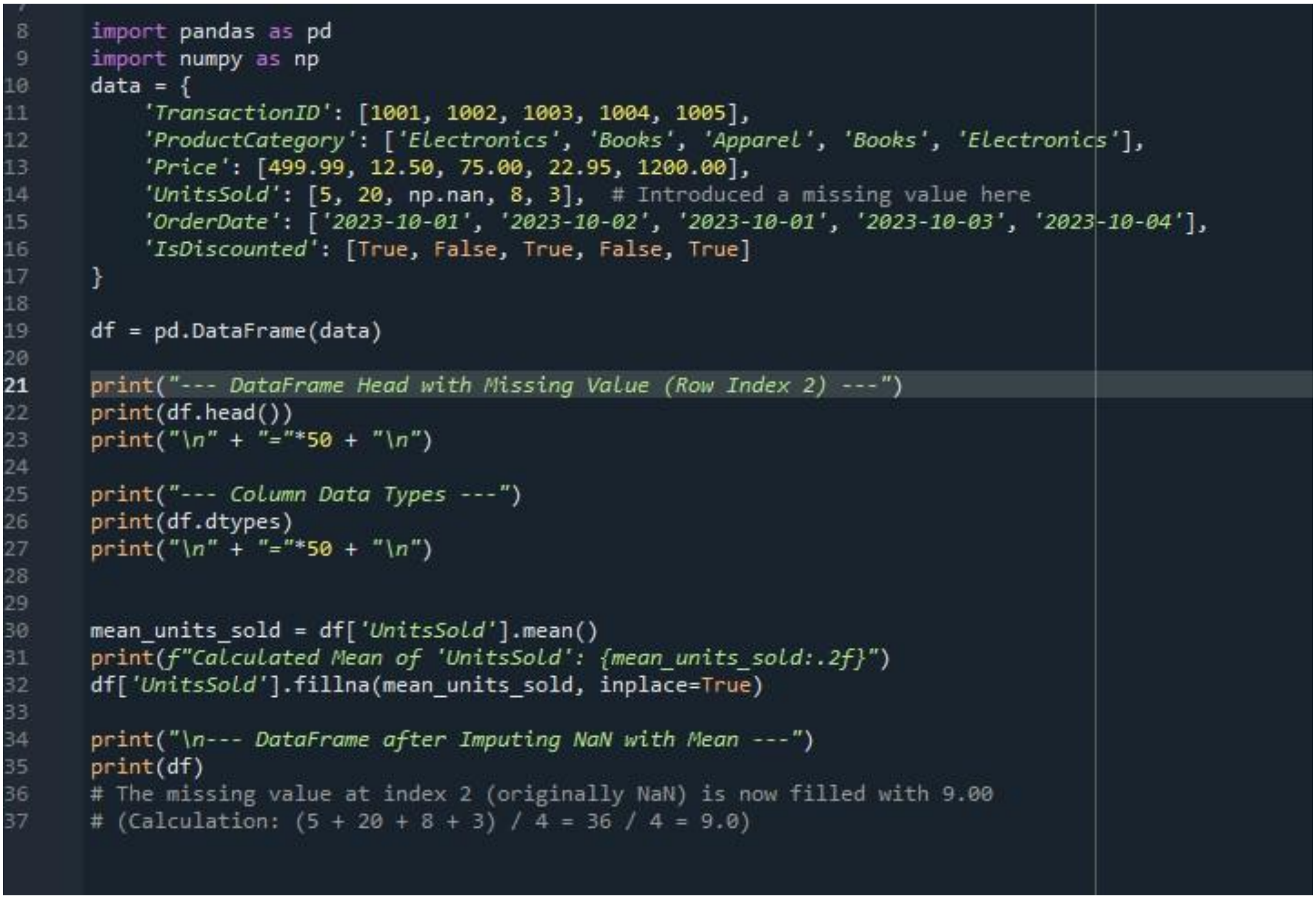
```
--- Original DataFrame Head (First 5 Rows) ---
   TransactionID ProductCategory    Price  UnitsSold   OrderDate  IsDiscounted
0           1001     Electronics   499.99          5  2023-10-01          True
1           1002           Books    12.50         20  2023-10-02         False
2           1003         Apparel    75.00         15  2023-10-01          True
3           1004           Books    22.95          8  2023-10-03         False
4           1005     Electronics  1200.00          3  2023-10-04          True

==================================================

--- Column Data Types (Series output from .dtypes) ---
TransactionID        int64
ProductCategory     object
Price              float64
UnitsSold            int64
OrderDate           object
IsDiscounted          bool
dtype: object

==================================================
```



```
==================================================

--- DataFrame Summary (.info() method) ---
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5 entries, 0 to 4
Data columns (total 6 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   TransactionID    5 non-null      int64  
 1   ProductCategory  5 non-null      object 
 2   Price            5 non-null      float64
 3   UnitsSold        5 non-null      int64  
 4   OrderDate        5 non-null      object 
 5   IsDiscounted     5 non-null      bool   
dtypes: bool(1), float64(1), int64(2), object(2)
memory usage: 337.0+ bytes
```

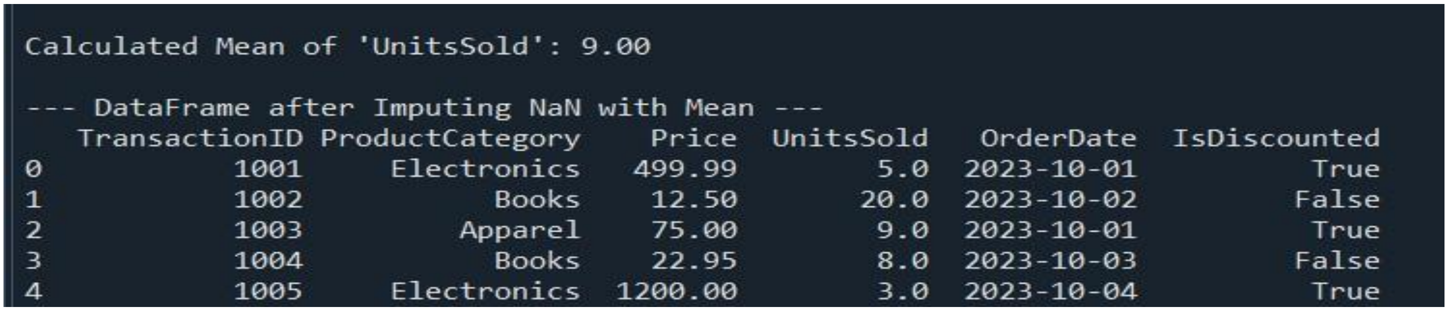
Write a code snippet that demonstrates how to fill missing values with the mean of a column.

Code:-



```
import pandas as pd
import numpy as np
data = {
    'TransactionID': [1001, 1002, 1003, 1004, 1005],
    'ProductCategory': ['Electronics', 'Books', 'Apparel', 'Books', 'Electronics'],
    'Price': [499.99, 12.50, 75.00, 22.95, 1200.00],
    'UnitsSold': [5, 20, np.nan, 8, 3],  # Introduced a missing value here
    'OrderDate': ['2023-10-01', '2023-10-02', '2023-10-01', '2023-10-03', '2023-10-04'],
    'IsDiscounted': [True, False, True, False, True]
}

df = pd.DataFrame(data)

print("--- DataFrame Head with Missing Value (Row Index 2) ---")
print(df.head())
print("\n" + "="*50 + "\n")

print("--- Column Data Types ---")
print(df.dtypes)
print("\n" + "="*50 + "\n")


mean_units_sold = df['UnitsSold'].mean()
print(f"Calculated Mean of 'UnitsSold': {mean_units_sold:.2f}")
df['UnitsSold'].fillna(mean_units_sold, inplace=True)

print("\n--- DataFrame after Imputing NaN with Mean ---")
print(df)
# The missing value at index 2 (originally NaN) is now filled with 9.00
# (Calculation: (5 + 20 + 8 + 3) / 4 = 36 / 4 = 9.0)
```

Output:-



```
Calculated Mean of 'UnitsSold': 9.00

--- DataFrame after Imputing NaN with Mean ---
   TransactionID ProductCategory    Price  UnitsSold   OrderDate  IsDiscounted
0           1001     Electronics   499.99        5.0  2023-10-01          True
1           1002           Books    12.50       20.0  2023-10-02         False
2           1003         Apparel    75.00        9.0  2023-10-01          True
3           1004           Books    22.95        8.0  2023-10-03         False
4           1005     Electronics  1200.00        3.0  2023-10-04          True
```