DATA SCIENCE

Name: K Vinoth Kumar

Roll No: CH.EN.U4ARE22011

AIM :

The aim of the code is to perform various data manipulation and preprocessing tasks using pandas and scikit-learn libraries.

The Code used in this assignment can be found here

CODE :

Import necessary libraries for data manipulation, numerical operations, and preprocessing.

Load the dataset from a CSV file into a pandas DataFrame and print its contents.

```
#Import Necessary Libraries
import pandas as pd
import numpy as np
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import MinMaxScaler,StandardScaler

df  = pd.read_csv('datasets/dataset1.csv')     # Read the file
print(df)
```

✓ 0.0s

	Person	Age	City	Bool	Marks
0	Person2	24.0	Bangalore	No	47
1	Person3	19.0	Delhi	No	89
2	Person4	NaN	Mumbai	Yes	93
3	Person5	24.0	Hyderabad	No	85
4	Person1	17.0	Chennai	Yes	98

df.describe - Generate descriptive statistics

df.shape - Representing the dimensionality of the DataFrame

df.isnull - Return a boolean same-sized object indicating if the values are NA

```
print(df['Age'])    # Access Coloumn
print('\n')

print(df.describe())  # Generate descriptive statistics4
print('\n')

print(df.shape)                    #representing the dimensionality of the DataFrame
print('\n')

print(df.isnull())                 #Return a boolean same-sized object indicating if the values are NA
```

[18] ✓ 0.0s

```
0    24.0
1    19.0
2     NaN
3    24.0
4    17.0
Name: Age, dtype: float64


             Age      Marks
count   4.000000   5.000000
mean   21.000000  82.400000
std     3.559026  20.366639
min    17.000000  47.000000
25%    18.500000  85.000000
50%    21.500000  89.000000
75%    24.000000  93.000000
max    24.000000  98.000000


(5, 5)


   Person    Age   City   Bool  Marks
0   False  False  False  False  False
1   False  False  False  False  False
2   False   True  False  False  False
3   False  False  False  False  False
4   False  False  False  False  False
```

df.head - Returns first n rows

df.loc - Access a group of rows and columns by label(s)

df.iloc - Purely integer-location based indexing for selection by position.

```
print(df); print('\n')

print(df.head(3))                      # Returns first n rows
print('\n')

print(df.loc[df['City']=='Bangalore'])  # Access a group of rows and columns by label(s)
print('\n')

print(df.iloc[1,1])                    # Purely integer-location based indexing for selection by position.
```

[13] ✓ 0.0s

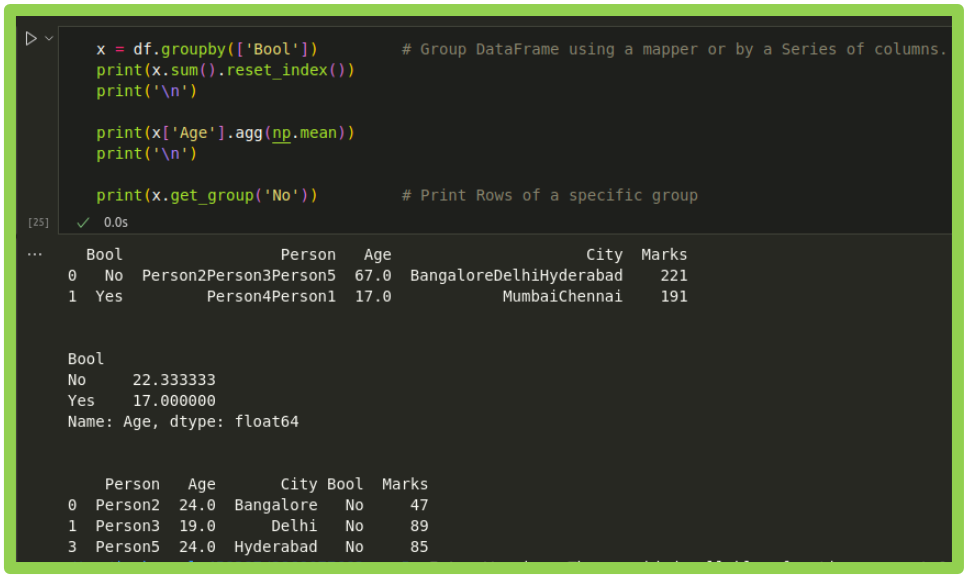
```
    Person   Age       City Bool  Marks
0  Person2  24.0  Bangalore   No     47
1  Person3  19.0      Delhi   No     89
2  Person4   NaN     Mumbai  Yes     93
3  Person5  24.0  Hyderabad   No     85
4  Person1  17.0    Chennai  Yes     98


    Person   Age       City Bool  Marks
0  Person2  24.0  Bangalore   No     47
1  Person3  19.0      Delhi   No     89
2  Person4   NaN     Mumbai  Yes     93


    Person   Age       City Bool  Marks
0  Person2  24.0  Bangalore   No     47


19.0
```

df.groupby - Group DataFrame using a mapper or by a Series of columns.



```
x = df.groupby(['Bool'])        # Group DataFrame using a mapper or by a Series of columns.
print(x.sum().reset_index())
print('\n')

print(x['Age'].agg(np.mean))
print('\n')

print(x.get_group('No'))        # Print Rows of a specific group
```

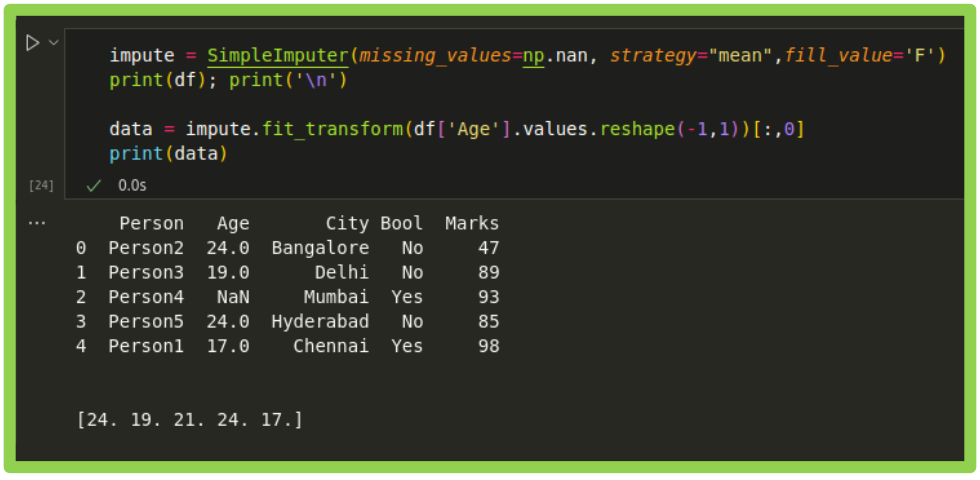
[25] ✓ 0.0s

```
   Bool                Person   Age                    City  Marks
0    No  Person2Person3Person5  67.0  BangaloreDelhiHyderabad    221
1   Yes          Person4Person1  17.0            MumbaiChennai    191


Bool
No     22.333333
Yes    17.000000
Name: Age, dtype: float64


    Person   Age       City Bool  Marks
0  Person2  24.0  Bangalore   No     47
1  Person3  19.0      Delhi   No     89
3  Person5  24.0  Hyderabad   No     85
```

Simple Imputer : Replace missing values using a descriptive statistic (e.g. mean, median, or most frequent) along each column or using a constant value.



```
impute = SimpleImputer(missing_values=np.nan, strategy="mean",fill_value='F')
print(df); print('\n')

data = impute.fit_transform(df['Age'].values.reshape(-1,1))[:,0]
print(data)
```

[24] ✓ 0.0s

```
    Person   Age       City Bool  Marks
0  Person2  24.0  Bangalore   No     47
1  Person3  19.0      Delhi   No     89
2  Person4   NaN     Mumbai  Yes     93
3  Person5  24.0  Hyderabad   No     85
4  Person1  17.0    Chennai  Yes     98


[24. 19. 21. 24. 17.]
```

get_dummies() - Each variable is converted in as many 0/1 variables as there are different values. Columns in the output are each named after a value; if the input is a DataFrame, the name of the original variable is prepended to the value.

```
df1 = pd.read_csv('datasets/cars.csv')              #Reads cars.csv file

new_cars = pd.get_dummies(df1[['Model']])
print(new_cars.to_string())
#print(new_cars)
```

0.0s Python

	Model_1	Model_2	Model_3	Model_5	Model_500	Model_A-Class	Model_A1	Model_A4	Model_A6	Model_Astra	Model_Aygo	Model_B-Max	Model_C-Class	Model_CLA	Model_Citigo	Model_Civic	Model_Cooper	Model_E-Cl
0	False	False	False	False	False	False	False	False	False	False	True	False	False	False	False	False	False	Fa
1	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	Fa
2	False	False	False	False	False	False	False	False	False	False	False	False	False	False	True	False	False	Fa
3	False	False	False	False	True	False	False	False	False	False	False	False	False	False	False	False	False	Fa
4	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	True	Fa
5	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	Fa
6	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	Fa
7	False	False	False	False	False	True	False	False	False	False	False	False	False	False	False	False	False	Fa
8	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	Fa
9	False	False	False	False	False	False	True	False	False	False	False	False	False	False	False	False	False	Fa
10	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	Fa
11	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	Fa
12	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	Fa
13	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	True	False	Fa
14	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	Fa
15	False	False	False	False	False	False	False	False	False	True	False	False	False	False	False	False	False	Fa
16	True	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	Fa
17	False	False	True	False	False	False	False	False	False	False	False	False	False	False	False	False	False	Fa
18	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	Fa
19	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	Fa
20	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	Fa
21	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	Fa
22	False	False	False	False	False	False	False	False	False	False	False	False	True	False	False	False	False	Fa
23	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	Fa
...																		
32	False	False	False	False	False	False	False	False	False	False	False	True	False	False	False	False	False	Fa
33	False	True	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	Fa
34	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	Fa
35	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	Fa

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

MinMaxScaler :

Transform features by scaling each feature to a given range.

This estimator scales and translates each feature individually such that it is in the given range on the training set, e.g. between zero and one.

The transformation is given by::

X_std = (X - X.min(axis=0)) / (X.max(axis=0) - X.min(axis=0))

X_scaled = X_std * (max - min) + min

where min, max = feature_range.

This transformation is often used as an alternative to zero mean, unit variance scaling.

StandardScaler :

Standardize features by removing the mean and scaling to unit variance.

The standard score of a sample `x` is calculated as:

z = (x - u) / s

where `u` is the mean of the training samples or zero if `with_mean=False`, and `s` is the standard deviation of the training samples or one if `with_std=False`.

```
cars = pd.read_csv('datasets/cars.csv')
Vol_data = cars[['Volume']]
Weight_data = cars[['Weight']]

scaler = MinMaxScaler()
stdscaler = StandardScaler()
```

[31] ✓ 0.0s

MinMaxScaler.fit_transform - Fit to data, then transform it.

StandardScaler.fit_transform - Fit to data, then transform it.

```
scaler.fit(Vol_data)
mm_data_vol = scaler.transform(Vol_data)
print(mm_data_vol)
print('\n')

scaler.fit(Weight_data)
mm_data_weight = scaler.transform(Weight_data)
print(mm_data_weight)
print('\n')

stdscaler.fit(Vol_data)
stddata_vol = stdscaler.transform(Vol_data)
print(stddata_vol)
print('\n')

stdscaler.fit(Weight_data)
stddata_weight = stdscaler.transform(Weight_data)
print(stddata_weight)
```

[39] ✓ 0.0s

```
[[0.0625]
 [0.1875]
 [0.0625]
 [0.    ]
 [0.375 ]
 [0.0625]
 [0.3125]
 [0.375 ]
 [0.375 ]
 [0.4375]
 [0.125 ]
 [0.25  ]
 [0.0625]
 [0.4375]
 [0.4375]
 [0.4375]
 [0.4375]
 [0.8125]
 [0.4375]
 [0.6875]
 [0.4375]
 [0.6875]
 [0.75  ]
 [0.4375]
 [0.6875]
```

RESULT :

The results show successful execution of data preprocessing steps including loading, inspecting, exploring, and transforming data.

Key operations such as handling missing values and scaling numerical features were performed, preparing the data for subsequent analysis or machine learning tasks.

These steps are crucial for ensuring the quality and consistency of the dataset, ultimately leading to more reliable and accurate modeling outcomes.

LEARNING OUTCOMES :

1. Gained proficiency in loading and manipulating datasets using pandas DataFrame operations.

2. Acquired the ability to inspect datasets for missing values and understand the importance of handling incomplete data.

3. Gained knowledge of different data scaling techniques, including Min-Max scaling and Standard scaling.

4. Learned to fit and transform data using these scalers to prepare features for machine learning algorithms.

5. Enhanced ability to document code and explain its functionality, purpose, and output