Name: Viven Hotwani Class:D15C/17

Experiment 1

Aim: Introduction to Data science and Data preparation using Pandas steps.

Theory:

Data science is the study of data that helps us derive useful insight for business decision making. Data Science is all about using tools, techniques, and creativity to uncover insights hidden within data. It combines math, computer science, and domain expertise to tackle real-world challenges in a variety of fields.

<u>Data science involves these key steps:</u>

- **Data Collection:** Gathering raw data from various sources, such as databases, sensors, or user interactions.
- **Data Cleaning:** Ensuring the data is accurate, complete, and ready for analysis.
- **Data Analysis:** Applying statistical and computational methods to identify patterns, trends, or relationships.
- **Data Visualization:** Creating charts, graphs, and dashboards to present findings clearly.
- **Decision-Making:** Using insights to inform strategies, create solutions, or predict outcomes.

Dataset Overview:

The dataset consists of 21 columns, each providing valuable insights into vehicle details, registration data, and vehicle types across different locations. Below is a breakdown of the dataset's columns and their significance:

- **ID**: A unique identifier assigned to each vehicle, distinguishing them in the inventory.
- **Plate Type**: Represents the type of plate assigned to the vehicle, reflecting its registration or classification.
- **Primary Customer City**: The city where the primary customer resides, giving insights into regional distribution.
- **Primary Customer State**: The state where the primary customer resides, which helps in analyzing geographic patterns.
- **Registration Start Date**: The start date of the vehicle's registration, indicating the time frame for its use.
- **Registration Expiration Date**: The expiration date of the vehicle's registration, helping track the validity of the registration.

- **Registration Usage**: Represents the usage category of the vehicle's registration, such as regular or commercial use.
- **Vehicle Type**: The classification of the vehicle, such as Motorcycle or Passenger, giving insights into vehicle segmentation.
- **Vehicle Weight**: The weight of the vehicle, which could be relevant for logistics and fuel efficiency considerations.
- **Vehicle Year**: The year the vehicle was manufactured, which is crucial for understanding its age and market trends.
- **Vehicle Make**: The manufacturer or brand of the vehicle, which is vital for brand-specific market analysis.
- **Vehicle Model**: The specific model of the vehicle, helping identify product features and target consumer segments.
- **Vehicle Body**: Describes the type of body the vehicle has, such as Sedan (SD) or SUV, aiding in vehicle categorization.
- **Primary Color**: The primary color of the vehicle, which can influence customer preference and aesthetics.
- **Vehicle Declared Gross Weight**: The gross weight of the vehicle as declared by the manufacturer, important for regulatory and logistical purposes.
- **Fuel Code**: Represents the fuel type used by the vehicle, such as Electric (E00) or Hybrid (H04), helping to analyze energy consumption patterns.
- **Vehicle Recorded GVWR**: The Gross Vehicle Weight Rating (GVWR) recorded for the vehicle, a key measure for vehicle classification.
- **Vehicle Name**: The official name or model name of the vehicle, assisting in product-specific analysis.
- **Type**: The vehicle's classification type, such as BEV (Battery Electric Vehicle) or PHEV (Plug-in Hybrid Electric Vehicle), influencing environmental analysis.
- **Vehicle Category**: The classification of the vehicle based on its size and usage, such as Light-Duty (Class 1-2).

Problem Statement:

The dataset provides detailed information about various vehicles, including their registration, attributes, and classifications. The primary objectives of analyzing this dataset are:

- **Vehicle Performance**: Identifying which vehicle types, makes, and models perform better in terms of registration and customer preferences.
- **Customer Insights**: Understanding the impact of customer location (city and state) on vehicle preferences and registrations.

- **Fuel Type Analysis**: Investigating how different fuel types (BEV, PHEV, etc.) influence vehicle demand and registration trends.
- **Pricing and Brand Impact**: Analyzing how vehicles make, model, and type correlate with pricing strategies and customer behavior.
- **Sales and Registration Trends**: Exploring how vehicle year and type affect registration volume over time, helping in forecasting and inventory management.

By processing and analyzing this dataset, the goal is to uncover trends that can assist in strategic decisions related to vehicle marketing, inventory, and customer targeting.

Code:

1] This returns a tuple indicating the number of rows and columns in the DataFrame. This code prints "dataset info" and displays the DataFrame's structure including data types and non-null counts using `df.info()`.
2] It then prints "dataset description" and shows summary statistics like mean, standard deviation, and percentiles for numerical columns with `df.describe()`.

Analyzing its dimensions

```
[3] df.shape
```

```
(52691, 20)
```

```
print('datadet info')
df.info()
print('\ndataset description')
df.describe()
```

```
datadet info
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 52691 entries, 0 to 52690
Data columns (total 20 columns):
 #   Column                         Non-Null Count  Dtype
---  ------                         --------------  -----
 0   ID                             52691 non-null  int64
 1   Plate Type                     52691 non-null  object
 2   Primary Customer City          52657 non-null  object
 3   Primary Customer State         52657 non-null  object
 4   Registration Start Date        52691 non-null  object
 5   Registration Expiration Date   52691 non-null  object
 6   Registration Usage             52691 non-null  object
 7   Vehicle Type                   52691 non-null  object
 8   Vehicle Weight                 52691 non-null  int64
 9   Vehicle Year                   52691 non-null  int64
 10  Vehicle Make                   52691 non-null  object
 11  Vehicle Model                  52691 non-null  object
 12  Vehicle Body                   52691 non-null  object
 13  Primary Color                  52691 non-null  object
 14  Vehicle Declared Gross Weight  52691 non-null  int64
 15  Fuel Code                      52691 non-null  object
 16  Vehicle Recorded GVWR          52691 non-null  int64
 17  Vehicle Name                   52691 non-null  object
 18  Type                           52691 non-null  object
 19  Vehicle Category               52691 non-null  object
dtypes: int64(5), object(15)
```

Name: Viven Hotwani Class:D15C/17

dataset description

	ID	Vehicle Weight	Vehicle Year	Vehicle Declared Gross Weight	Vehicle Recorded GVWR
count	5.269100e+04	52691.000000	52691.000000	5.269100e+04	52691.000000
mean	1.541024e+06	662.548633	2021.632119	6.362825e+05	806.937466
std	1.039725e+06	2127.894845	2.471569	1.643281e+06	2300.669623
min	9.300000e+01	0.000000	1998.000000	0.000000e+00	0.000000
25%	5.487550e+05	0.000000	2021.000000	0.000000e+00	0.000000
50%	1.673766e+06	0.000000	2022.000000	0.000000e+00	0.000000
75%	2.613891e+06	0.000000	2023.000000	0.000000e+00	0.000000
max	2.934513e+06	80000.000000	2025.000000	7.214477e+06	80000.000000

3] This code checks each column in the DataFrame for missing values and sums them up.

To check null values and remove maximum missing values

```
df.isnull().sum()
```

	0
ID	0
Plate Type	0
Primary Customer City	34
Primary Customer State	34
Registration Start Date	0
Registration Expiration Date	0
Registration Usage	0
Vehicle Type	0
Vehicle Weight	0
Vehicle Year	0
Vehicle Make	0
Vehicle Model	0
Vehicle Body	0
Primary Color	0
Vehicle Declared Gross Weight	0
Fuel Code	0
Vehicle Recorded GVWR	0
Vehicle Name	0
Type	0

Name: Viven Hotwani Class:D15C/17

4] This code replaces zeros in the DataFrame with NA values, then removes all rows containing any missing data to create a cleaned DataFrame.
It prints the cleaned DataFrame and confirms no missing values remain by summing NA entries for each column.

```
df.replace(0, pd.NA, inplace=True)
df_cleaned = df.dropna()
print(df_cleaned.isna().sum())
```

```
Plate Type                   0
Primary Customer City        0
Registration Usage           0
Vehicle Type                 0
Vehicle Weight               0
Primary Color                0
Vehicle Category             0
Z_Score                      0
Vehicle Weight Normalized    0
dtype: int64
```

Dropping unnecessary features

```
df.drop(['ID', 'Primary Customer State', 'Registration Start Date', 'Registration Expiration Date',
         'Vehicle Year', 'Vehicle Make', 'Vehicle Model', 'Vehicle Body',
         'Vehicle Declared Gross Weight', 'Fuel Code', 'Vehicle Recorded GVWR',
         'Vehicle Name', 'Type'], axis=1, inplace=True)
```

This code removes duplicate rows from the cleaned DataFrame.

```
[35] df = df_cleaned.drop_duplicates()
     df.head()
```

	Plate Type	Primary Customer City	Registration Usage	Vehicle Type	Vehicle Weight	Primary Color	Vehicle Category	Z_Score	Vehicle Weight Normalized
71	Passenger	SOUTHINGTON	Regular	SUV	5530	Gray	Light-Duty (Class 1-2)	-0.047744	0.376398
233	Passenger	GLASTONBURY	Regular	SUV	4774	Gray	Light-Duty (Class 1-2)	-0.533065	0.282484
276	Passenger	WALLINGFORD	Regular	Passenger	5200	Black	Light-Duty (Class 1-2)	-0.25959	0.335404
279	Passenger	WILLIMANTIC	Regular	Passenger	4200	Black	Light-Duty (Class 1-2)	-0.901549	0.21118
337	Passenger	WINDSOR	Regular	SUV	4800	Gray	Light-Duty (Class 1-2)	-0.516374	0.285714

```
df.shape
```

```
(678, 20)
```

Name: Viven Hotwani Class:D15C/17

5] This code creates dummy data out of plate type as commercial and passenger. This helps to convert categorical data to numerical data and helps in analysis in the algorithm

Creating a dummy variable for primary color

```
df_dummies = pd.get_dummies(df, columns=['Plate Type'], drop_first=True)
print(df_dummies)
```

```
      Primary Customer City Registration Usage Vehicle Type Vehicle Weight  \
71              SOUTHINGTON            Regular          SUV           5530
233             GLASTONBURY            Regular          SUV           4774
276             WALLINGFORD            Regular    Passenger           5200
279             WILLIMANTIC            Regular    Passenger           4200
337                 WINDSOR            Regular          SUV           4800
...                     ...                ...          ...            ...
52470                WESTON            Regular        Truck           8450
52609               PRESTON            Regular        Truck           8000
52648                GOSHEN            Regular          SUV           6450
52670             STRATFORD            Regular        Truck           8250
52684             WOODSTOCK            Regular    Passenger           5300

      Primary Color        Vehicle Category  Plate Type_Commercial  \
71             Gray  Light-Duty (Class 1-2)                  False
233            Gray  Light-Duty (Class 1-2)                  False
276           Black  Light-Duty (Class 1-2)                  False
279           Black  Light-Duty (Class 1-2)                  False
337            Gray  Light-Duty (Class 1-2)                  False
...             ...                     ...                    ...
52470         Black  Light-Duty (Class 1-2)                  False
52609          Gray  Light-Duty (Class 1-2)                  False
52648          Blue  Light-Duty (Class 1-2)                  False
52670          Gray  Light-Duty (Class 1-2)                  False
52684          Blue  Light-Duty (Class 1-2)                  False

       Plate Type_Passenger
71                     True
233                    True
276                    True
279                    True
337                    True
...                     ...
52470                  True
52609                  True
52648                  True
52670                  True
52684                  True

[678 rows x 8 columns]
```

Name: Viven Hotwani Class:D15C/17

6] The `head()` method to display the first ten rows of the DataFrame, to identify outliers manually we use the standardization approach (z score method). We find mean and standard deviation of the vehicle weight and calculate its z score; if its less than -3 or greater than 3 means its an outlier.

Detect outliers manually

```
df.head(10)
```

	Plate Type	Primary Customer City	Registration Usage	Vehicle Type	Vehicle Weight	Primary Color	Vehicle Category
71	Passenger	SOUTHINGTON	Regular	SUV	5530	Gray	Light-Duty (Class 1-2)
233	Passenger	GLASTONBURY	Regular	SUV	4774	Gray	Light-Duty (Class 1-2)
276	Passenger	WALLINGFORD	Regular	Passenger	5200	Black	Light-Duty (Class 1-2)
279	Passenger	WILLIMANTIC	Regular	Passenger	4200	Black	Light-Duty (Class 1-2)
337	Passenger	WINDSOR	Regular	SUV	4800	Gray	Light-Duty (Class 1-2)
416	Passenger	BRISTOL	Regular	SUV	5530	Silver	Light-Duty (Class 1-2)
420	Passenger	WEST SIMSBURY	Regular	Passenger	5600	Blue	Light-Duty (Class 1-2)
616	Passenger	NEW HAVEN	Regular	Passenger	3840	Black	Light-Duty (Class 1-2)
782	Passenger	GLASTONBURY	Regular	SUV	4800	Silver	Light-Duty (Class 1-2)
857	Passenger	GUILFORD	Regular	SUV	4709	Red	Light-Duty (Class 1-2)

Name: Viven Hotwani Class:D15C/17

```
#By Z-score method
mean_vehicle_weight = df['Vehicle Weight'].mean()
std_vehicle_weight = df['Vehicle Weight'].std()

print(f"Mean of Vehicle Weight: {mean_vehicle_weight}")
print(f"Standard Deviation of Vehicle Weight: {std_vehicle_weight}")

# Calculate the Z-score for each vehicle weight
df['Z_Score'] = (df['Vehicle Weight'] - mean_vehicle_weight) / std_vehicle_weight

print(df[['Vehicle Weight', 'Z_Score']])

# Identify outliers based on the Z-score
outliers = df[df['Z_Score'].abs() > 3]
print(outliers)
```

```
Mean of Vehicle Weight: 5604.371681415929
Standard Deviation of Vehicle Weight: 1557.7315042063165
      Vehicle Weight   Z_Score
71              5530 -0.047744
233             4774 -0.533065
276             5200  -0.25959
279             4200 -0.901549
337             4800 -0.516374
...              ...       ...
52470           8450  1.826777
52609           8000  1.537896
52648           6450  0.542859
52670           8250  1.698385
52684           5300 -0.195394

[678 rows x 2 columns]
        Plate Type Primary Customer City Registration Usage Vehicle Type  \
15639  Combination             THOMASTON        Combination          SUV
23230  Combination             FAIRFIELD        Combination        Truck
32996  Combination           WILLIMANTIC        Combination          Van
50047  Combination               WINDSOR        Combination        Truck

      Vehicle Weight Primary Color         Vehicle Category   Z_Score
15639          10550         White   Light-Duty (Class 1-2)  3.174891
23230          10550         White   Light-Duty (Class 1-2)  3.174891
32996          10360        Orange  Medium-Duty (Class 3-6)  3.052919
50047          10500         White  Medium-Duty (Class 3-6)  3.142793
```

Name: Viven Hotwani Class:D15C/17

7] We normalize the data across the vehicle weights on a scale of 0 to 1.

Normalization

```
# Min-Max Normalization
min_vehicle_weight = df['Vehicle Weight'].min()
max_vehicle_weight = df['Vehicle Weight'].max()

# Apply normalization
df['Vehicle Weight Normalized'] = (df['Vehicle Weight'] - min_vehicle_weight) / (max_vehicle_weight - min_vehicle_weight)

print(df[['Vehicle Weight', 'Vehicle Weight Normalized']])
```

```
       Vehicle Weight  Vehicle Weight Normalized
71               5530                   0.376398
233              4774                   0.282484
276              5200                   0.335404
279              4200                    0.21118
337              4800                   0.285714
...               ...                        ...
52470            8450                    0.73913
52609            8000                    0.68323
52648            6450                   0.490683
52670            8250                   0.714286
52684            5300                   0.347826

[678 rows x 2 columns]
```

8] This is our normalized data with respect to vehicle weight and can be used to analyse the vehicle weight distribution across its type and color.

```
[38] print("The first 10 rows of normalized data are :")
     df.head(10)
```

The first 10 rows of normalized data are :

	Plate Type	Primary Customer City	Registration Usage	Vehicle Type	Vehicle Weight	Primary Color	Vehicle Category	Z_Score	Vehicle Weight Normalized
71	Passenger	SOUTHINGTON	Regular	SUV	5530	Gray	Light-Duty (Class 1-2)	-0.047744	0.376398
233	Passenger	GLASTONBURY	Regular	SUV	4774	Gray	Light-Duty (Class 1-2)	-0.533065	0.282484
276	Passenger	WALLINGFORD	Regular	Passenger	5200	Black	Light-Duty (Class 1-2)	-0.25959	0.335404
279	Passenger	WILLIMANTIC	Regular	Passenger	4200	Black	Light-Duty (Class 1-2)	-0.901549	0.21118
337	Passenger	WINDSOR	Regular	SUV	4800	Gray	Light-Duty (Class 1-2)	-0.516374	0.285714
416	Passenger	BRISTOL	Regular	SUV	5530	Silver	Light-Duty (Class 1-2)	-0.047744	0.376398
420	Passenger	WEST SIMSBURY	Regular	Passenger	5600	Blue	Light-Duty (Class 1-2)	-0.002806	0.385093
616	Passenger	NEW HAVEN	Regular	Passenger	3840	Black	Light-Duty (Class 1-2)	-1.132655	0.16646
782	Passenger	GLASTONBURY	Regular	SUV	4800	Silver	Light-Duty (Class 1-2)	-0.516374	0.285714
857	Passenger	GUILFORD	Regular	SUV	4709	Red	Light-Duty (Class 1-2)	-0.574792	0.27441

Conclusion:
Thus we have pre-processed our dataset by various techniques mentioned and can be used for analysis and trained under algorithms for predictions.