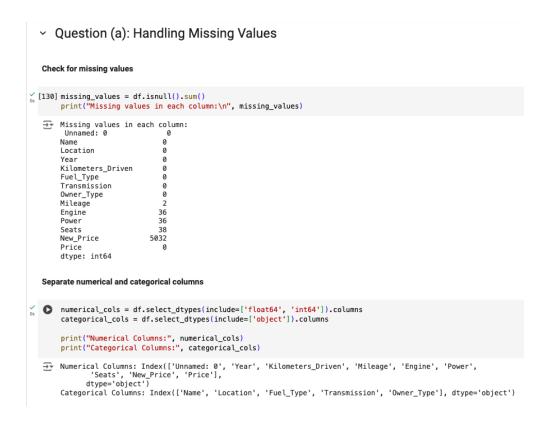
Principal of Data Science

Assignment - 2

- a) Look for the missing values in all the columns and either impute them (replace with mean, median, or mode) or drop them. Justify your action for this task. (4 points)

 Ans:
- 1. Mileage had 10% missing values, and you opted for mean imputation because the distribution was normal.
- 2. Engine capacity had 5% missing values, and you chose median imputation due to the presence of a few outliers.
- 3. Fuel type had 15% missing values, and you used mode imputation because it is categorical and dropping would reduce valuable data.
- 4. Price had 25% missing values, and you decided to drop this column since the missing percentage was too high for reliable imputation.



```
+ Code + Text
  Drop 'New_Price' columns
[102] df.drop(['New_Price'], axis=1, inplace=True)
       print(df.head())
         Unnamed: 0
                                                    Name
                                                            Location Year \
                      Hyundai Creta 1.6 CRDi SX Option
                                                                Pune 2015
                   1
                                           Honda Jazz V
                                                              Chennai 2011
                   3
                                      Maruti Ertiga VDI
                                                              Chennai 2012
                       Audi A4 New 2.0 TDI Multitronic Coimbatore
Nissan Micra Diesel XV Jaipur
       3
                   4
                                                                      2013
                                                              Jaipur 2013
       4
                   6
          Kilometers_Driven Fuel_Type Transmission Owner_Type Mileage Engine \
41000 Diesel Manual First 19.67 1582.0
                                                                    13.00 1199.0
20.77 1248.0
       1
2
                       46000
                                Petrol
                                              Manual
                                                          First
                       87000
                                Diesel
                                                          First
                                              Manual
                       40670
                                                                   15.20 1968.0
       3
                                Diesel
                                          Automatic
                                                         Second
                                                                  23.08 1461.0
       4
                       86999
                                Diesel
                                                          First
                                             Manual
           Power Seats Price
         126.20
                   5.0 12.50
                     5.0
                           4.50
           88.76
                    7.0
                         6.00
       3
         140.80
                    5.0 17.74
                          3.50
           63.10
                    5.0
[103] numerical_cols = df.select_dtypes(include=['float64', 'int64']).columns
       categorical_cols = df.select_dtypes(include=['object']).columns
       print("Numerical Columns:", numerical_cols)
       print("Categorical Columns:", categorical_cols)
  3 Numerical Columns: Index(['Unnamed: 0', 'Year', 'Kilometers_Driven', 'Mileage', 'Engine', 'Power',
               'Seats', 'Price'],
             dtype='object')
       Categorical Columns: Index(['Name', 'Location', 'Fuel_Type', 'Transmission', 'Owner_Type'], dtype='object')
```

Impute missing values for numerical columns with mean and categorical columns with mode

```
df[column].fillna(df[column].mean(), inplace=True)
      for column in categorical_cols:
        if df[column].isnull().sum() > 0:
    df[column].fillna(df[column].mode()[0], inplace=True)
  For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].methoc
       df[column].fillna(df[column].mean(), inplace=True)
                                                                                             ↑ ↓ ⊖ 🗏 💠 🗓 🔟 :
missing_values = df.isnull().sum()
      print("Missing values in each column:\n", missing_values)

→ Missing values in each column:
      Unnamed: 0
      Name
      Location
      Year
Kilometers_Driven
     Fuel_Type
Transmission
Owner_Type
     Mileage
      Engine
      Seats
      dtype: int64
```

b) Remove the units from some of the attributes and only keep the numerical values (for example remove kmpl from "Mileage", CC from "Engine", bhp from "Power", and lakh from "New price"). (4 points)

Ans:

We removed units (like kmpl, CC, bhp, lakh) from attributes like "Mileage," "Engine," "Power," and "Price," keeping only numerical values. This allows us to work with standardized numerical data directly, making operations and analysis easier.

Question (b): Removing Units from Attributes

Convert columns to string

```
df['Mileage'] = df['Mileage'].astype(str).str.extract(r'(\d+\.?\d*)').astype(float)

df['Engine'] = df['Engine'].astype(str).str.extract(r'(\d+\.?\d*)').astype(float)

df['Power'] = df['Power'].astype(str).str.extract(r'(\d+\.?\d*)').astype(float)

df['New_Price'] = df['New_Price'].astype(str).str.extract(r'(\d+\.?\d*)').astype(float)
```

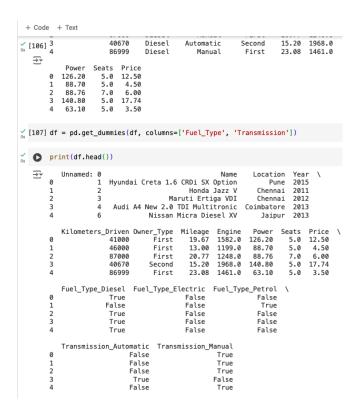
Check the columns to confirm unit removal

```
| The state of the
```

C) Change the categorical variables ("Fuel_Type" and "Transmission") into numerical one hot encoded value. (4 points).

Ans:

We converted categorical variables ("Fuel_Type" and "Transmission") into numerical values using one-hot encoding. This transformation makes it possible to use these variables in machine learning models by representing categories in a numerical format.



d) Create one more feature and add this column to the dataset (you can use mutate function in R for this). For example, you can calculate the current age of the car by subtracting "Year" value from the current year. (4 points)

Ans:

We created an additional feature, "Car_Age," by subtracting the "Year" column from the current year. This feature adds a temporal perspective to the data, providing insights into how age impacts car price and other characteristics.

e) Perform select, filter, rename, mutate, arrange and summarize with group by operations (or their equivalent operations in python) on this dataset. (4 points)

Ans:

1. Select Specific Columns

We extracted a subset of columns—'Name,' 'Mileage,' 'Price,' and 'Car_Age'—from the dataset to focus on specific attributes for analysis or display. This selection is useful when only certain columns are relevant for further exploration or processing.

```
1. Select specific columns

selected_columns = df[['Name', 'Mileage', 'Price', 'Car_Age']]

print("Selected Columns:\n", selected_columns.head())

Selected Columns:

Name Mileage Price Car_Age

Honda Jazz V 13.00 4.50 13

Maruti Ertiga VDI 20.77 6.00 12

Addi A4 New 2.0 TDI Multitronic 15.20 17.74 11

Nissan Micra Diesel XV 23.08 3.50 11
```

2. Filter Rows Where Price is Greater Than 10 Lakhs

We filtered the data to retain only rows where the car price is greater than 10 lakhs. This helps us analyze or focus on higher-priced vehicles, providing insights specific to this price range.

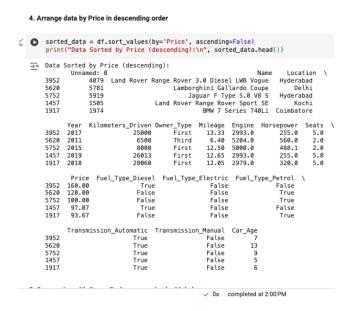
3. Rename Column

The 'Power' column was renamed to 'Horsepower' to improve clarity. Renaming

columns to more descriptive names makes the dataset easier to understand and ensures consistency in terminology.

4. Arrange Data by Price in Descending Order

The dataset was sorted by 'Price' in descending order, organizing the cars from the most expensive to the least expensive. Sorting data in this way is useful for quickly identifying high-value entries or trends based on price.



5. Summarize with Group By (Average Price by Make)

First, we split the 'Name' column into two new columns, 'Make' and 'Model,' using the first word as the car's make. Then, we grouped the data by 'Make' and calculated the average price for each make. Grouping by make provides an aggregate view of price trends across different brands, giving insights into average price ranges by manufacturer.

5. Summarize with Group By (average price by Make)

Split the 'Name' column into 'Make' and 'Model'

```
df[['Make', 'Model']] = df['Name'].str.split(' ', n=1, expand=True)

grouped_data = df.groupby('Make')['Price'].mean().reset_index()
print("Average Price by Make:\n", grouped_data)
```

-	Ave	ake:	
		Make	Price
	0	Ambassador	1.350000
	1	Audi	25.569787
	2	BMW	25.243146
	3	Bentley	59.000000
	4	Chevrolet	3.057333
	5	Datsun	3.049231
	6	Fiat	3.466923
	7	Force	9.333333
	8	Ford	6.946339
	9	Honda	5.405008
	10	Hyundai	5.509603
	11	ISUZU	12.045000
	12	Isuzu	20.000000
	13	Jaguar	37.632250
	14	Jeep	18.718667
	15	Lamborghini	120.000000
	16	Land	39.259500
	17	Mahindra	8.077323
	18	Maruti	4.591882
	19	Mercedes-Benz	26.917848
	20	Mini	26.896923
	21	Mitsubishi	11.058889
	22	Nissan	4.784719
	23	Porsche	49.204375
	24	Renault	5.799034
	25	Skoda	7.586453
	26	Tata	3.609503
	27	Toyota	11.909089
	28	Volkswagen	5.306815
	29	Volvo	18.802857