load the dataset

```
import pandas as pd

# Load the dataset
df = pd.read_csv("train.csv")

# Show the shape and first few rows
print("Shape of the dataset:", df.shape)
df.head()
```

```
Shape of the dataset: (5847, 14)
```

	Unnamed: 0	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type	Mileage	Engine	Power	Seats	New_P
0	1	Hyundai Creta 1.6 CRDi SX Option	Pune	2015	41000	Diesel	Manual	First	19.67 kmpl	1582 CC	126.2 bhp	5.0	
1	2	Honda Jazz V	Chennai	2011	46000	Petrol	Manual	First	13 km/kg	1199 CC	88.7 bhp	5.0	8.61
2	3	Maruti Ertiga VDI	Chennai	2012	87000	Diesel	Manual	First	20.77 kmpl	1248 CC	88.76 bhp	7.0	
3	4	Audi A4 New 2.0 TDI	Coimbatore	2013	40670	Diesel	Automatic	Second	15.2 kmpl	1968 CC	140.8 bhp	5.0	

Next steps: Generate code with df | View recommended plots | New interactive sheet

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)

```
# Check for missing values
missing = df.isnull().sum()
print("Missing values:\n", missing)
```

```
# Example imputation logic
df["Mileage"].fillna(df["Mileage"].mode()[0], inplace=True)
df["Engine"].fillna(df["Engine"].mode()[0], inplace=True)
df["Power"].fillna(df["Power"].mode()[0], inplace=True)
df["Seats"].fillna(df["Seats"].mode()[0], inplace=True)
df["New_Price"].fillna(df["New_Price"].mode()[0], inplace=True)

# Drop rows with too many missing values (optional)
df.dropna(inplace=True)
print("Remaining missing values:\n", df.isnull().sum())
```

Show hidden output

```
# Define a function to extract numeric part only
def extract_numeric(val):
    if pd.isnull(val):
        return np.nan
    # Keep only digits and dot
    return float(''.join(ch for ch in str(val) if (ch.isdigit() or ch == '.')))

# Apply the function to each column
df["Mileage"] = df["Mileage"].apply(extract_numeric)
df["Engine"] = df["Engine"].apply(extract_numeric)
df["Power"] = df["Power"].apply(extract_numeric)

# For New_Price (handle both Lakh and Cr properly)
def convert_price(price):
    try:
        if "Lakh" in price:
            return float(price.replace(" Lakh", "")) * 100000
        elif "Cr" in price:
            return float(price.replace(" Cr", "")) * 10000000
    except:
        return np.nan

df["New_Price"] = df["New_Price"].apply(convert_price)
```

Remove Units from Attributes (Task b)

```
df["Mileage"].fillna(df["Mileage"].median(), inplace=True)
df["Engine"].fillna(df["Engine"].median(), inplace=True)
df["Power"].fillna(df["Power"].median(), inplace=True)
df["New_Price"].fillna(df["New_Price"].median(), inplace=True)
```

Show hidden output

One-Hot Encode Categorical Variables (Task c)

```
# One-hot encode Fuel_Type and Transmission
df = pd.get_dummies(df, columns=["Fuel_Type", "Transmission"], drop_first=True)

df.head()
```

	Unnamed: 0	Name	Location	Year	Kilometers_Driven	Owner_Type	Mileage	Engine	Power	Seats	New_Price	Price	Fuel_Type_E
0	1	Hyundai Creta 1.6 CRDi SX Option	Pune	2015	41000	First	19.67	1582.0	126.20	5.0	478000.0	12.50	
1	2	Honda Jazz V	Chennai	2011	46000	First	13.00	1199.0	88.70	5.0	861000.0	4.50	
2	3	Maruti Ertiga VDI	Chennai	2012	87000	First	20.77	1248.0	88.76	7.0	478000.0	6.00	
3	4	Audi A4 New 2.0 TDI Multitronic	Coimbatore	2013	40670	Second	15.20	1968.0	140.80	5.0	478000.0	17.74	
4	6	Nissan Micra Diesel XV	Jaipur	2013	86999	First	23.08	1461.0	63.10	5.0	478000.0	3.50	

Next steps: Generate code with df | View recommended plots | New interactive sheet

Add New Feature - Car Age (Task d)

```
# Add a new column 'Car_Age'
current_year = 2025
df["Car_Age"] = current_year - df["Year"]

df[["Year", "Car_Age"]].head()
```

	Year	Car_Age
0	2015	10
1	2011	14
2	2012	13
3	2013	12
4	2013	12

Perform Data Operations (Task e)

```
# SELECT: Pick a few columns
selected_df = df[["Name", "Location", "Year", "Car_Age", "Mileage", "Price"]]

# FILTER: Cars older than 10 years
filtered_df = df[df["Car_Age"] > 10]

# RENAME: Rename Price to Selling_Price
renamed_df = df.rename(columns={"Price": "Selling_Price"})

# MUTATE: Add new column price per kilometer
renamed_df["Price_per_km"] = renamed_df["Selling_Price"] / renamed_df["Kilometers_Driven"]

# ARRANGE: Sort by Price descending
sorted_df = renamed_df.sort_values(by="Selling_Price", ascending=False)

# SUMMARIZE with GROUP BY: Average price by location
grouped_df = df.groupby("Location")["Price"].agg(["mean", "median", "count"]).reset_index()


df.head(10)
```

	Unnamed: 0	Name	Location	Year	Kilometers_Driven	Owner_Type	Mileage	Engine	Power	Seats	New_Price	Price	Fuel_Type_
0	1	Hyundai Creta 1.6 CRDi SX Option	Pune	2015	41000	First	19.67	1582.0	126.20	5.0	478000.0	12.50	
1	2	Honda Jazz V	Chennai	2011	46000	First	13.00	1199.0	88.70	5.0	861000.0	4.50	
2	3	Maruti Ertiga VDI	Chennai	2012	87000	First	20.77	1248.0	88.76	7.0	478000.0	6.00	
3	4	Audi A4 New 2.0 TDI Multitronic	Coimbatore	2013	40670	Second	15.20	1968.0	140.80	5.0	478000.0	17.74	
4	6	Nissan Micra Diesel XV	Jaipur	2013	86999	First	23.08	1461.0	63.10	5.0	478000.0	3.50	
5	7	Toyota Innova Crysta 2.8 GX AT 8S	Mumbai	2016	36000	First	11.36	2755.0	171.50	8.0	2100000.0	17.50	
6	8	Volkswagen Vento Diesel Comfortline	Pune	2013	64430	First	20.54	1598.0	103.60	5.0	478000.0	5.20	
7	9	Tata Indica Vista Quadrajet LS	Chennai	2012	65932	Second	22.30	1248.0	74.00	5.0	478000.0	1.95	
8	10	Maruti Ciaz Zeta	Kochi	2018	25692	First	21.56	1462.0	103.25	5.0	1065000.0	9.95	
9	11	Honda City 1.5 V AT Sunroof	Kolkata	2012	60000	First	16.80	1497.0	116.30	5.0	478000.0	4.49	