Import the required Libraries

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder
from google.colab import files
```

```
uploaded = files.upload()
```

```
Choose Files train.csv
• train.csv(text/csv) - 644523 bytes, last modified: 04/11/2023 - 100% done
Saving train.csv to train.csv
```

Load the data

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

Task a) Handle missing values

```
# Check for missing values
print(df.isnull().sum())

# Impute or drop missing values
for column in df.columns:
    if pd.api.types.is_numeric_dtype(df[column]):
        df[column].fillna(df[column].median(), inplace=True)
    else:
        df[column].fillna(df[column].mode()[0], inplace=True)
```

```
Unnamed: 0             0
Name                   0
Location               0
Year                   0
Kilometers_Driven      0
Fuel_Type              0
Transmission           0
Owner_Type             0
Mileage                2
Engine                36
Power                 36
Seats                 38
New_Price           5032
Price                  0
dtype: int64
```

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

```
Unnamed: 0          0
Name                0
Location            0
Year                0
Kilometers_Driven   0
Fuel_Type           0
Transmission        0
Owner_Type          0
Mileage             0
Engine              0
Power               0
Seats               0
New_Price           0
Price               0
dtype: int64
```

Task b) Remove units from attributes

```
# Remove units and convert columns to appropriate data types
df['Mileage'] = df['Mileage'].replace(r'\s+kmpl|\s+km/kg', '', regex=True).astype(float)
df['Engine'] = df['Engine'].replace('CC', '', regex=True).astype(int)
```

```
df['Power'] = df['Power'].replace('bhp', '', regex=True).astype(float)
```

```
def convert_price_to_float(price):
    if pd.isnull(price):
        return np.nan
    if isinstance(price, float):  # If it's already a float, return as is.
        return price
    try:
        price = price.strip()  # Remove any leading/trailing whitespace
        # Remove ' Lakh' and convert to float
        if 'Lakh' in price:
            return float(price.replace('Lakh', ''))
        # Convert 'Cr' to float, assuming 1 Cr = 100 Lakh
        elif 'Cr' in price:
            return float(price.replace('Cr', '')) * 100
    except ValueError as e:
        # Log or print any values that could not be converted
        print(f"Cannot convert {price}: {e}")
        return np.nan

# Apply this conversion to the New_Price column
df['New_Price'] = df['New_Price'].apply(lambda x: convert_price_to_float(x))
```

Task c) Encode categorical variables

```
# Perform one-hot encoding on categorical variables
df = pd.get_dummies(df, columns=['Fuel_Type', 'Transmission'], drop_first=True)
```

Task d) Create an additional feature

```
# Creating a new feature - 'Car_Age'
current_year = pd.to_datetime('today').year
df['Car_Age'] = current_year - df['Year']
```

```
# Check the resulting DataFrame
print(df.head())
```

```
   Unnamed: 0                              Name    Location  Year  \
0           1  Hyundai Creta 1.6 CRDi SX Option        Pune  2015
1           2                      Honda Jazz V     Chennai  2011
2           3                 Maruti Ertiga VDI     Chennai  2012
3           4   Audi A4 New 2.0 TDI Multitronic  Coimbatore  2013
4           6            Nissan Micra Diesel XV      Jaipur  2013

   Kilometers_Driven Owner_Type  Mileage  Engine   Power  Seats  New_Price  \
0              41000      First    19.67    1582  126.20    5.0       4.78
1              46000      First    13.00    1199   88.70    5.0       8.61
2              87000      First    20.77    1248   88.76    7.0       4.78
3              40670     Second    15.20    1968  140.80    5.0       4.78
4              86999      First    23.08    1461   63.10    5.0       4.78

   Price  Fuel_Type_Electric  Fuel_Type_Petrol  Transmission_Manual  Car_Age
0  12.50                   0                 0                    1        8
1   4.50                   0                 1                    1       12
2   6.00                   0                 0                    1       11
3  17.74                   0                 0                    0       10
4   3.50                   0                 0                    1       10
```