DSBDL Assignment 01 - Data Wrangling 1

Perform the following operations using Python on any open-source dataset (e.g., data.csv)

1. Import all the required Python Libraries.
2. Locate an open-source data from the web (e.g. https://www.kaggle.com). Provide a clear description of the data and its source (i.e., URL of the web site).
3. Load the Dataset into pandas' data frame.
4. Data Preprocessing: check for missing values in the data using pandas isnull (), describe() function to get some initial statistics. Provide variable descriptions. Types of variables etc. Check the dimensions of the data frame.
5. Data Formatting and Data Normalization: Summarize the types of variables by checking the data types (i.e., character, numeric, integer, factor, and logical) of the variables in the data set. If variables are not in the correct data type, apply proper type conversions.
6. Turn categorical variables into quantitative variables in Python.

In addition to the codes and outputs, explain every operation that you do in the above steps and explain everything that you do to import/read/scrape the data set.

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Dataset

Source: https://www.kaggle.com/datasets/pouyamofidi/carsalesextendedmissingdata

Features: 5

Instances: 1000

```
ds = pd.read_csv('/content/drive/My Drive/DSBDL/Assignment1/car-sales.csv')
ds
```

	Make	Colour	Odometer (KM)	Doors	Price
0	Honda	White	35431.0	4.0	15323.0
1	BMW	Blue	192714.0	5.0	19943.0
2	Honda	White	84714.0	4.0	28343.0
3	Toyota	White	154365.0	4.0	13434.0
4	Nissan	Blue	181577.0	3.0	14043.0
...	...	...	...	...	...
995	Toyota	Black	35820.0	4.0	32042.0
996	NaN	White	155144.0	3.0	5716.0
997	Nissan	Blue	66604.0	4.0	31570.0
998	Honda	White	215883.0	4.0	4001.0
999	Toyota	Blue	248360.0	4.0	12732.0

1000 rows × 5 columns

Next steps: Generate code with ds | View recommended plots

```
ds['Odometer (KM)'] = ds['Odometer (KM)'].astype('Int64')
ds['Doors'] = ds['Doors'].astype('Int64')
ds['Price'] = ds['Price'].astype('Int64')
ds
```

	Make	Colour	Odometer (KM)	Doors	Price
0	Honda	White	35431	4	15323
1	BMW	Blue	192714	5	19943
2	Honda	White	84714	4	28343
3	Toyota	White	154365	4	13434
4	Nissan	Blue	181577	3	14043
...	...	...	...	...	...
995	Toyota	Black	35820	4	32042
996	NaN	White	155144	3	5716
997	Nissan	Blue	66604	4	31570
998	Honda	White	215883	4	4001
999	Toyota	Blue	248360	4	12732

1000 rows × 5 columns

Next steps: Generate code with ds | View recommended plots

```
ds.shape
```

```
(1000, 5)
```

```
ds.dtypes
```

```
Make             object
Colour           object
Odometer (KM)     Int64
Doors             Int64
Price             Int64
dtype: object
```

```
ds.describe()
```

	Odometer (KM)	Doors	Price
count	950.0	950.0	950.0
mean	131253.237895	4.011579	16042.814737
std	69094.857187	0.382539	8581.695036
min	10148.0	3.0	2796.0
25%	70391.25	4.0	9529.25
50%	131821.0	4.0	14297.0
75%	192668.5	4.0	20806.25
max	249860.0	5.0	52458.0

```
ds.isna().sum()
```

```
Make             49
Colour           50
Odometer (KM)    50
Doors            50
Price            50
dtype: int64
```

```
ds['Make'].fillna(value = ds['Make'].mode()[0], inplace = True)
ds['Colour'].fillna(value = ds['Colour'].mode()[0], inplace = True)
ds['Doors'].fillna(value = ds['Doors'].mode()[0], inplace = True)
ds['Odometer (KM)'].fillna(value = ds['Odometer (KM)'].mode()[0], inplace = True)
ds['Price'].fillna(value = ds['Price'].mode()[0], inplace = True)
ds
```

	Make	Colour	Odometer (KM)	Doors	Price
0	Honda	White	35431	4	15323
1	BMW	Blue	192714	5	19943
2	Honda	White	84714	4	28343
3	Toyota	White	154365	4	13434
4	Nissan	Blue	181577	3	14043
...	...	...	...	...	...
995	Toyota	Black	35820	4	32042
996	Toyota	White	155144	3	5716
997	Nissan	Blue	66604	4	31570
998	Honda	White	215883	4	4001
999	Toyota	Blue	248360	4	12732

1000 rows × 5 columns

Next steps: Generate code with ds | View recommended plots

```
ds.isna().sum()
```

```
Make             0
Colour           0
Odometer (KM)    0
Doors            0
Price            0
dtype: int64
```

```
def normalize(col_name):
  ds[col_name] = (ds[col_name] - ds[col_name].min()) / (ds[col_name].max() - ds[col_name].min())

normalize("Odometer (KM)")
normalize("Doors")
normalize("Price")
```

```
pip install category_encoders
```

```
Collecting category_encoders
  Downloading category_encoders-2.6.3-py2.py3-none-any.whl (81 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 81.9/81.9 kB 3.1 MB/s eta 0:00:00
Requirement already satisfied: numpy>=1.14.0 in /usr/local/lib/python3.10/dist-packages (from category_encoders) (1.25.2)
Requirement already satisfied: scikit-learn>=0.20.0 in /usr/local/lib/python3.10/dist-packages (from category_encoders) (1.2.2)
Requirement already satisfied: scipy>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from category_encoders) (1.11.4)
Requirement already satisfied: statsmodels>=0.9.0 in /usr/local/lib/python3.10/dist-packages (from category_encoders) (0.14.1)
Requirement already satisfied: pandas>=1.0.5 in /usr/local/lib/python3.10/dist-packages (from category_encoders) (1.5.3)
Requirement already satisfied: patsy>=0.5.1 in /usr/local/lib/python3.10/dist-packages (from category_encoders) (0.5.6)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.0.5->category_encoc
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.0.5->category_encoders) (202:
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from patsy>=0.5.1->category_encoders) (1.16.0)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.20.0->category_encoder
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.20.0->category_
Requirement already satisfied: packaging>=21.3 in /usr/local/lib/python3.10/dist-packages (from statsmodels>=0.9.0->category_encoder
Installing collected packages: category_encoders
Successfully installed category_encoders-2.6.3
```

```
from category_encoders import OrdinalEncoder
```

```
ds['Make'].value_counts()
```

```
Toyota    428
Honda     292
Nissan    183
BMW        97
Name: Make, dtype: int64
```

```
mp1 = [{'col': 'Make', 'mapping': {"Toyota": 1,  "Honda": 2, "Nissan": 3, "BMW": 4}}]

ds = OrdinalEncoder(cols=['Make'], mapping=mp1).fit(ds).transform(ds)
ds.head()
```

	Make	Colour	Odometer (KM)	Doors	Price
0	2	White	0.105472	0.5	0.252245
1	4	Blue	0.761606	1.0	0.345274
2	2	White	0.311065	0.5	0.514417
3	1	White	0.601626	0.5	0.214208
4	3	Blue	0.715146	0.0	0.226471

Next steps: Generate code with ds View recommended plots

```
ds['Colour'].value_counts()
```

```
White    440
Blue     302
Black     95
Red       88
Green     75
Name: Colour, dtype: int64
```

```
mp2 =  [{'col': 'Colour', 'mapping': {"White": 1,  "Blue": 2, "Black": 3, "Red": 4, "Green": 5}}]

ds = OrdinalEncoder(cols=['Colour'], mapping=mp2).fit(ds).transform(ds)
ds.head()
```

	Make	Colour	Odometer (KM)	Doors	Price
0	2	1	0.105472	0.5	0.252245
1	4	2	0.761606	1.0	0.345274
2	2	1	0.311065	0.5	0.514417
3	1	1	0.601626	0.5	0.214208
4	3	2	0.715146	0.0	0.226471

Next steps: Generate code with ds View recommended plots

Done!!!