CAR SALES EDA

In [1]:
```
#pip install missingno
```

In [2]:
```
# Import Libraries

import numpy as np                      # implement multi-dimensional arrays and matrices
import pandas as pd                     # for data manupulation and analysis
import matplotlib.pyplot as plt         # for data visualization
import seaborn as sns                   # provide high_level of interface for drawing attractive and informative statistics
%matplotlib inline
import missingno as msno
sns.set()
from subprocess import check_output    # for viewing profile report
#import warnings   to remove all warnings
#warnings.filterwarnings("ignore")
```

In [3]:
```
# Load data and view top 5

carsales = pd.read_excel(r"C:\Users\EZ FARMING\Desktop\Seun Personal Docs\DATA SCIENCE\PYTHON\EXPLORATIVE DATA ANALYSIS\SORAN AUT

carsales.head()
```

Out[3]:

	car	price	body	mileage	engV	engType	registration	year	model	drive
0	Ford	15500.0	crossover	68	2.5	Gas	yes	2010	Kuga	full
1	Mercedes-Benz	20500.0	sedan	173	1.8	Gas	yes	2011	E-Class	rear
2	Mercedes-Benz	35000.0	other	135	5.5	Petrol	yes	2008	CL 550	rear
3	Mercedes-Benz	17800.0	van	162	1.8	Diesel	yes	2012	B 180	front
4	Mercedes-Benz	33000.0	vagon	91	NaN	Other	yes	2013	E-Class	NaN

Overview of dataset

In [4]:
```
carsales.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9576 entries, 0 to 9575
Data columns (total 10 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   car           9576 non-null   object 
 1   price         9576 non-null   float64
 2   body          9576 non-null   object 
 3   mileage       9576 non-null   int64  
 4   engV          9142 non-null   float64
 5   engType       9576 non-null   object 
 6   registration  9576 non-null   object 
 7   year          9576 non-null   int64  
 8   model         9576 non-null   object 
 9   drive         9065 non-null   object 
dtypes: float64(2), int64(2), object(6)
memory usage: 748.2+ KB
```

In [5]:
```
# Dimensionality
carsales.shape
```

Out[5]: (9576, 10)

In [6]:
```
# statistical summary of the dataset
carsales.describe().astype(int)
```

Out[6]:

	price	mileage	engV	year
count	9576	9576	9142	9576
mean	15633	138	2	2006
std	24106	98	5	7
min	0	0	0	1953
25%	4999	70	1	2004
50%	9200	128	2	2008
75%	16700	194	2	2012
max	547800	999	99	2016

In [7]:
```
#missing value count
carsales.isna().sum()
```

Out[7]:
```
car               0
price             0
body              0
mileage           0
engV            434
engType           0
registration      0
year              0
model             0
drive           511
dtype: int64
```

Preprofiling

In [8]:
```
pip install -U pandas-profiling
```

```
Requirement already satisfied: pandas-profiling in c:\users\ez farming\anaconda\lib\site-packages (3.6.6)
Requirement already satisfied: ydata-profiling in c:\users\ez farming\anaconda\lib\site-packages (from pandas-profiling) (4.1.
2)
Requirement already satisfied: scipy<1.10,>=1.4.1 in c:\users\ez farming\anaconda\lib\site-packages (from ydata-profiling->pand
as-profiling) (1.9.1)
Requirement already satisfied: matplotlib<3.7,>=3.2 in c:\users\ez farming\anaconda\lib\site-packages (from ydata-profiling->pa
ndas-profiling) (3.5.2)
Requirement already satisfied: statsmodels<0.14,>=0.13.2 in c:\users\ez farming\anaconda\lib\site-packages (from ydata-profilin
g->pandas-profiling) (0.13.2)
Requirement already satisfied: imagehash==4.3.1 in c:\users\ez farming\anaconda\lib\site-packages (from ydata-profiling->pandas
-profiling) (4.3.1)
Requirement already satisfied: visions[type_image_path]==0.7.5 in c:\users\ez farming\anaconda\lib\site-packages (from ydata-pr
ofiling->pandas-profiling) (0.7.5)
Requirement already satisfied: pydantic<1.11,>=1.8.1 in c:\users\ez farming\anaconda\lib\site-packages (from ydata-profiling->p
andas-profiling) (1.10.7)
Requirement already satisfied: jinja2<3.2,>=2.11.1 in c:\users\ez farming\anaconda\lib\site-packages (from ydata-profiling->pan
das-profiling) (2.11.3)
Requirement already satisfied: numpy<1.24,>=1.16.0 in c:\users\ez farming\anaconda\lib\site-packages (from ydata-profiling->pan
das-profiling) (1.21.5)
Requirement already satisfied: PyYAML<6.1,>=5.0.0 in c:\users\ez farming\anaconda\lib\site-packages (from ydata-profiling->pand
as-profiling) (6.0)
Requirement already satisfied: seaborn<0.13,>=0.10.1 in c:\users\ez farming\anaconda\lib\site-packages (from ydata-profiling->p
andas-profiling) (0.11.2)
Requirement already satisfied: requests<2.29,>=2.24.0 in c:\users\ez farming\anaconda\lib\site-packages (from ydata-profiling->
pandas-profiling) (2.28.1)
Requirement already satisfied: htmlmin==0.1.12 in c:\users\ez farming\anaconda\lib\site-packages (from ydata-profiling->pandas-
profiling) (0.1.12)
Requirement already satisfied: phik<0.13,>=0.11.1 in c:\users\ez farming\anaconda\lib\site-packages (from ydata-profiling->pand
as-profiling) (0.12.3)
Requirement already satisfied: tqdm<4.65,>=4.48.2 in c:\users\ez farming\anaconda\lib\site-packages (from ydata-profiling->pand
as-profiling) (4.64.1)
Requirement already satisfied: typeguard<2.14,>=2.13.2 in c:\users\ez farming\anaconda\lib\site-packages (from ydata-profiling-
>pandas-profiling) (2.13.3)
Requirement already satisfied: pandas!=1.4.0,<1.6,>1.1 in c:\users\ez farming\anaconda\lib\site-packages (from ydata-profiling-
>pandas-profiling) (1.4.4)
Requirement already satisfied: multimethod<1.10,>=1.4 in c:\users\ez farming\anaconda\lib\site-packages (from ydata-profiling->
pandas-profiling) (1.9.1)
Requirement already satisfied: pillow in c:\users\ez farming\anaconda\lib\site-packages (from imagehash==4.3.1->ydata-profiling
->pandas-profiling) (9.2.0)
Requirement already satisfied: PyWavelets in c:\users\ez farming\anaconda\lib\site-packages (from imagehash==4.3.1->ydata-profi
ling->pandas-profiling) (1.3.0)
Requirement already satisfied: networkx>=2.4 in c:\users\ez farming\anaconda\lib\site-packages (from visions[type_image_path]==
0.7.5->ydata-profiling->pandas-profiling) (2.8.4)
Requirement already satisfied: attrs>=19.3.0 in c:\users\ez farming\anaconda\lib\site-packages (from visions[type_image_path]==
0.7.5->ydata-profiling->pandas-profiling) (21.4.0)
Requirement already satisfied: tangled-up-in-unicode>=0.0.4 in c:\users\ez farming\anaconda\lib\site-packages (from visions[typ
e_image_path]==0.7.5->ydata-profiling->pandas-profiling) (0.2.0)
Requirement already satisfied: MarkupSafe>=0.23 in c:\users\ez farming\anaconda\lib\site-packages (from jinja2<3.2,>=2.11.1->yd
ata-profiling->pandas-profiling) (2.0.1)
Requirement already satisfied: packaging>=20.0 in c:\users\ez farming\anaconda\lib\site-packages (from matplotlib<3.7,>=3.2->yd
ata-profiling->pandas-profiling) (21.3)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\ez farming\anaconda\lib\site-packages (from matplotlib<3.7,>=3.
2->ydata-profiling->pandas-profiling) (2.8.2)
Requirement already satisfied: cycler>=0.10 in c:\users\ez farming\anaconda\lib\site-packages (from matplotlib<3.7,>=3.2->ydata
-profiling->pandas-profiling) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\ez farming\anaconda\lib\site-packages (from matplotlib<3.7,>=3.2->
ydata-profiling->pandas-profiling) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\ez farming\anaconda\lib\site-packages (from matplotlib<3.7,>=3.2->
ydata-profiling->pandas-profiling) (1.4.2)
Requirement already satisfied: pyparsing>=2.2.1 in c:\users\ez farming\anaconda\lib\site-packages (from matplotlib<3.7,>=3.2->y
data-profiling->pandas-profiling) (3.0.9)
Requirement already satisfied: pytz>=2020.1 in c:\users\ez farming\anaconda\lib\site-packages (from pandas!=1.4.0,<1.6,>1.1->yd
ata-profiling->pandas-profiling) (2022.1)
Requirement already satisfied: joblib>=0.14.1 in c:\users\ez farming\anaconda\lib\site-packages (from phik<0.13,>=0.11.1->ydata
-profiling->pandas-profiling) (1.1.0)
Requirement already satisfied: typing-extensions>=4.2.0 in c:\users\ez farming\anaconda\lib\site-packages (from pydantic<1.11,>
=1.8.1->ydata-profiling->pandas-profiling) (4.3.0)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\ez farming\anaconda\lib\site-packages (from requests<2.29,>=2.
24.0->ydata-profiling->pandas-profiling) (1.26.11)
Requirement already satisfied: idna<4,>=2.5 in c:\users\ez farming\anaconda\lib\site-packages (from requests<2.29,>=2.24.0->yda
ta-profiling->pandas-profiling) (3.3)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\ez farming\anaconda\lib\site-packages (from requests<2.29,>=2.24.
0->ydata-profiling->pandas-profiling) (2022.9.14)
Requirement already satisfied: charset-normalizer<3,>=2 in c:\users\ez farming\anaconda\lib\site-packages (from requests<2.29,>
=2.24.0->ydata-profiling->pandas-profiling) (2.0.4)
Requirement already satisfied: patsy>=0.5.2 in c:\users\ez farming\anaconda\lib\site-packages (from statsmodels<0.14,>=0.13.2->
ydata-profiling->pandas-profiling) (0.5.2)
Requirement already satisfied: colorama in c:\users\ez farming\anaconda\lib\site-packages (from tqdm<4.65,>=4.48.2->ydata-profi
ling->pandas-profiling) (0.4.5)
Requirement already satisfied: six in c:\users\ez farming\anaconda\lib\site-packages (from patsy>=0.5.2->statsmodels<0.14,>=0.1
3.2->ydata-profiling->pandas-profiling) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

In [9]:
```
from pandas_profiling import ProfileReport
```

```
C:\Users\EZ FARMING\AppData\Local\Temp\ipykernel_15868\2274191625.py:1: DeprecationWarning: `import pandas_profiling` is going
to be deprecated by April 1st. Please use `import ydata_profiling` instead.
  from pandas_profiling import ProfileReport
```

In [10]:
```
carsales_profile = ProfileReport(carsales, title='carsales_before_data_cleaning')
carsales_profile
```

Summarize dataset: 100% 36/36 [00:14<00:00, 3.40it/s, Completed]

Generate report structure: 100% 1/1 [00:04<00:00, 4.83s/it]

Render HTML: 100% 1/1 [00:07<00:00, 7.20s/it]

Overview

Dataset statistics

Number of variables	10
Number of observations	9576
Missing cells	945
Missing cells (%)	1.0%
Duplicate rows	110
Duplicate rows (%)	1.1%
Total size in memory	748.2 KiB
Average record size in memory	80.0 B

Variable types

Categorical	4
Numeric	4
Boolean	1
Unsupported	1

Alerts

Dataset has 110 (1.1%) duplicate rows	Duplicates
`car` has a high cardinality: 87 distinct values	High cardinality
`price` is highly overall correlated with `year`	High correlation
`mileage` is highly overall correlated with `year`	High correlation
`year` is highly overall correlated with `price` and 1 other fields (price, mileage)	High correlation
`car` is highly overall correlated with `drive`	High correlation
`body` is highly overall correlated with `drive`	High correlation

Out[10]:

Data Cleaning

In [11]:
```
# make a copy of df - .copy()
carsales1 = carsales.copy()
carsales1.head(2)
```

Out[11]:

	car	price	body	mileage	engV	engType	registration	year	model	drive
0	Ford	15500.0	crossover	68	2.5	Gas	yes	2010	Kuga	full
1	Mercedes-Benz	20500.0	sedan	173	1.8	Gas	yes	2011	E-Class	rear

Rename column names

In [12]:
```
carsales.columns
```
Out[12]:
```
Index(['car', 'price', 'body', 'mileage', 'engV', 'engType', 'registration',
       'year', 'model', 'drive'],
      dtype='object')
```

In [13]:
```
carsales.columns = ['car_band', 'price', 'body', 'mileage', 'engineValues', 'engineType', 'registration',
       'year', 'model', 'drive']
carsales.head(2)
```
Out[13]:

	car_band	price	body	mileage	engineValues	engineType	registration	year	model	drive
0	Ford	15500.0	crossover	68	2.5	Gas	yes	2010	Kuga	full
1	Mercedes-Benz	20500.0	sedan	173	1.8	Gas	yes	2011	E-Class	rear

Handling Duplicates

In [14]:
```
carsales.duplicated().sum()
```
Out[14]:
```
113
```

In [15]:
```
# drop duplicate rows
carsales.drop_duplicates(inplace=True)
```

In [16]:
```
carsales.shape
```
Out[16]:
```
(9463, 10)
```

In [ ]:

Handling missing values

Type *Markdown* and LaTeX: α^2

In [17]:
```
# filling missing values for drive
# find the mode
carsales["drive"].mode()
```
Out[17]:
```
0    front
Name: drive, dtype: object
```

In [18]:
```
#fill missing values with the mode "front"
carsales["drive"] =  carsales["drive"].fillna('front')
```

In [19]:
```
#fill missing values for engineValue with the median
carsales["engineValues"] = carsales.groupby(['car_band', 'body'])['engineValues'].transform(lambda x: x.fillna(x.median()))
```

In [20]:
```
#Let's see what we did
carsales.isna().sum()
```
Out[20]:
```
car_band         0
price            0
body             0
mileage          0
engineValues    10
engineType       0
registration     0
year             0
model            0
drive            0
dtype: int64
```

In [21]:
```
#drop the remaining NaN values
carsales.dropna(subset=['engineValues'],inplace=True)
carsales.isnull().sum()
```
Out[21]:
```
car_band        0
price           0
body            0
mileage         0
engineValues    0
engineType      0
registration    0
year            0
model           0
drive           0
dtype: int64
```

In [22]:
```
carsales.price[carsales.price==0].count()
```
Out[22]: 238

In [23]:
```
# Dropping entries with price <= 0

carsales = carsales.drop(carsales[carsales.price <=0].index)
```

In [24]:
```
carsales.price[carsales.price==0].count()
```
Out[24]: 0

In [ ]:

In [ ]:

Post profiling

In [25]:
```
carsales_profile2 = ProfileReport(carsales, title='carsales_after_data_cleaning')
carsales_profile2
```

Summarize dataset: 100% 35/35 [00:06<00:00, 3.85it/s, Completed]

Generate report structure: 100% 1/1 [00:04<00:00, 4.29s/it]

Render HTML: 100% 1/1 [00:01<00:00, 1.59s/it]

Overview

Dataset statistics

Number of variables	10
Number of observations	9215
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	24
Duplicate rows (%)	0.3%
Total size in memory	1.0 MiB
Average record size in memory	116.7 B

Variable types

Categorical	4
Numeric	4
Boolean	1
Unsupported	1

Alerts

Dataset has 24 (0.3%) duplicate rows	Duplicates
`car_band` has a high cardinality: 84 distinct values	High cardinality
`price` is highly overall correlated with `year`	High correlation
`mileage` is highly overall correlated with `year`	High correlation
`year` is highly overall correlated with `price` and 1 other fields (price, mileage)	High correlation
`car_band` is highly overall correlated with `drive`	High correlation
`body` is highly overall correlated with `drive`	High correlation

Out[25]:

In [ ]:

Questions:

- Which type of cars are sold maximum?
- What is the correlation between price and mileage?
- How many cars are registered?
- Does the registeration status influence car price?
- What is the car price distribution based on Engine Value?
- Which car type has the highest pricing?

In [ ]:

```
```

EDA

In [26]:

```
# 1. Which type of cars are sold most?
```

In [27]:

```
carsales['car_band'].value_counts().head(1)
```

Out[27]:

```
Volkswagen    899
Name: car_band, dtype: int64
```

In [28]:

```
# 2. What is the correlation between price and mileage?
```

In [29]:

```
carsales['price'].corr(carsales['mileage'])
```

Out[29]:

```
-0.3267339848197728
```

In [30]:

```
# 3. How many cars are registered?
```

In [31]:

```
len(carsales[carsales['registration'] == 'yes'])
```

Out[31]:

```
8661
```

In [32]:

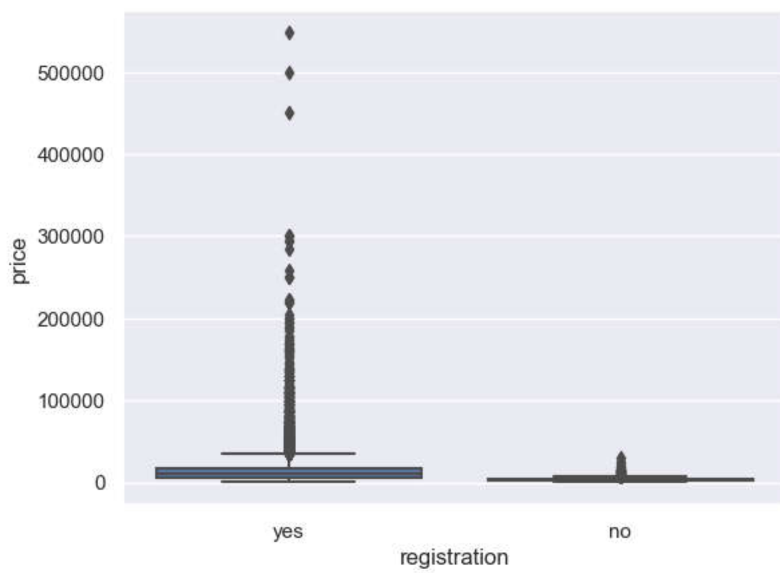
```
# 4. Does the registration status influence car price?.
```

In [33]:

```
sns.boxplot(x="registration", y="price", data=carsales)
```

Out[33]:

```
<AxesSubplot:xlabel='registration', ylabel='price'>
```



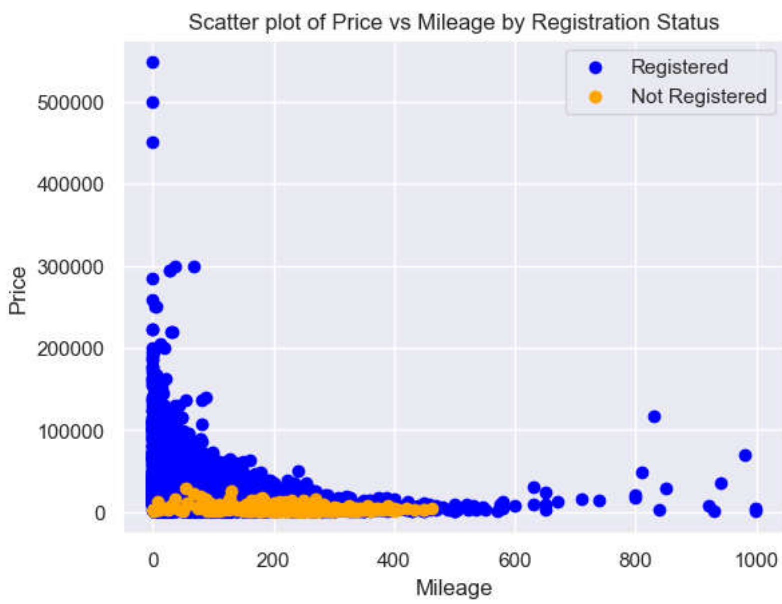
In [34]:
```
# separate data by registration status
registered = carsales[carsales['registration'] == 'yes']
not_registered = carsales[carsales['registration'] == 'no']

# plot scatter plot with different colors for each registration status
plt.scatter(registered['mileage'], registered['price'], color='blue', label='Registered')
plt.scatter(not_registered['mileage'], not_registered['price'], color='orange', label='Not Registered')

# set x and y labels
plt.xlabel('Mileage')
plt.ylabel('Price')

# set title and legend
plt.title('Scatter plot of Price vs Mileage by Registration Status')
plt.legend()

# display plot
plt.show()
```



In [35]:
```
# 5. What is the car price distribution based on Engine Value?
```

In [36]:
```
# Group the data by Engine Value and calculate the mean price
engine_groups = carsales.groupby('engineValues')['price'].mean()

# Create a scatter plot with Engine Value on x-axis and Price on y-axis
plt.scatter(x=carsales['engineValues'], y=carsales['price'], c=carsales['engineValues'], cmap='viridis', alpha=0.5)

# Add a horizontal line for the mean price of each Engine Value group
for engine_value, mean_price in engine_groups.iteritems():
    plt.axhline(mean_price, color='gray', linestyle='dashed', linewidth=1)

# Set the axis labels and title
plt.xlabel('Engine Value')
plt.ylabel('Price')
plt.title('Car Price Distribution by Engine Value')

plt.show()
```

In [37]:
```
engine_price_dist = carsales.groupby('engineValues')['price'].describe()
engine_price_dist
```

Out[37]:

engineValues	count	mean	std	min	25%	50%	75%	max
0.10	1.0	54000.000000	NaN	54000.0	54000.0	54000.0	54000.0	54000.0000
0.11	2.0	16700.000000	1414.213562	15700.0	16200.0	16700.0	17200.0	17700.0000
0.14	1.0	24300.000000	NaN	24300.0	24300.0	24300.0	24300.0	24300.0000
0.60	17.0	3870.529412	692.821957	2300.0	3450.0	3999.0	4200.0	5500.0000
0.65	1.0	38888.000000	NaN	38888.0	38888.0	38888.0	38888.0	38888.0000
...	...	...	...	...	...	...	...	...
74.00	1.0	7500.000000	NaN	7500.0	7500.0	7500.0	7500.0	7500.0000
75.00	1.0	11350.000000	NaN	11350.0	11350.0	11350.0	11350.0	11350.0000
85.00	1.0	2800.000000	NaN	2800.0	2800.0	2800.0	2800.0	2800.0000
90.00	1.0	6100.000000	NaN	6100.0	6100.0	6100.0	6100.0	6100.0000
99.99	27.0	5546.866319	3987.433118	550.0	2700.0	4600.0	7850.0	15195.3906

119 rows × 8 columns

In [38]:
```
carsales.columns
```

Out[38]:
```
Index(['car_band', 'price', 'body', 'mileage', 'engineValues', 'engineType',
       'registration', 'year', 'model', 'drive'],
      dtype='object')
```

In [39]:
```
# 6. Which car type has the highest pricing?
```

In [40]:
```
# Assuming the car sales data is stored in a DataFrame called 'carsales'
car_type_prices = carsales.groupby('body')['price'].mean().sort_values(ascending=False)
highest_priced_car_type = car_type_prices.index[0]

print(f"The car type with the highest pricing is {highest_priced_car_type}")
```

```
The car type with the highest pricing is crossover
```

In [ ]: