Coursework Specification (1CWK100)

6G7V0026 Principles of Data Science

Data Understanding, Exploration, and Preparation of Car Sale Adverts Dataset

The aim of our project is to discover associations and group differences that significantly affect the valuation of cars by using the Car Sale Adverts information given by AutoTrader. It is a .csv file having around 400K rows. The dataset includes a collection of anonymous advertisements for vehicles that include brand, type, colour, mileage, and the selling price.

Importing and Configuring Essential Packages

- The essential modules
- for data visualisation: `matplotlib` , `seaborn`
- for data manipulation: `pandas` , `numpy`
- for machine learning: `scikit-learn`
- `sns.set` sets up the visualisation engine (e.g., default size of figures)

In [108]:
```
%config InlineBackend.figure_format = 'retina'
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
sns.set(rc={'figure.figsize': (6,4)},
        style='ticks', context='talk', font_scale=0.8)
```

In [178]:
```
car_adv = pd.read_csv(
    'C:\\Users\\user\\OneDrive - MMU\\Jesty Lab\Assignment\\adverts.csv'
)
```

The data file is read by `pandas` , which creates a DataFrame object that we assign to the variable `car_adv` for future use.

1.0 Data Understanding and Exploration

When dealing with a new dataset, data exploration and understanding is an important phase in the data science process. A dataset can be understood and explored using a variety of methods as follows

- Viewing few rows
- Checking the data types
- Summary statistics
- Checking for missing values

- Visualization
- Data Cleaning
- Data Transformation
- Data Reduction

1.1. Meaning and Type of Features; Analysis of Distributions

1.1.1 Meaning of features by viewing rows

Viewing the rows of dataset gives an overall structure and content of the dataset. From the structure of data, the meaning of features can be analysed. Some common techniques to view data includes:

- `car_adv.head()` : View the initial few rows of DataFrame
- `car_adv.sample()` : View the random sample rows of DataFrame
- `car_adv.tail()` : View the last few rows of DataFrame

In [110]:
```
car_adv.head()
```

Out[110]:

	public_reference	mileage	reg_code	standard_colour	standard_make	standard_model	v
0	202006039777689	0.0	NaN	Grey	Volvo	XC90	
1	202007020778260	108230.0	61	Blue	Jaguar	XF	
2	202007020778474	7800.0	17	Grey	SKODA	Yeti	
3	202007080986776	45000.0	16	Brown	Vauxhall	Mokka	
4	202007161321269	64000.0	64	Grey	Land Rover	Range Rover Sport	

1.1.2 Type of features

Quantitative and Qualitative features of a dataset can be identified by checking the data types of each column. This also aids in locating any columns that might need to be cleaned up or converted before further analysis. dtypes attribute can be used to find the types of features.

In [111]:
```
car_adv.dtypes
```

Out[111]:
```
public_reference           int64
mileage                  float64
reg_code                  object
standard_colour           object
standard_make             object
standard_model            object
vehicle_condition         object
year_of_registration     float64
price                      int64
body_type                 object
crossover_car_and_van       bool
fuel_type                 object
dtype: object
```

1.1.3 Analysis of distribution

The describe() function can be used to view the target column's summary statistics. In this dataset the target column is 'price'. The value counts() method can be used to view the unique values and their counts of categorical column. To understand and explore the data, data visualisation can be a powerful resource.Box plot, histograms, scatter plots, and bar plots can all be made using the matplotlib or seaborn libraries.

In [112]:
```
car_adv[['price','mileage']].describe()
```

Out[112]:

	price	mileage
count	4.020050e+05	401878.000000
mean	1.734197e+04	37743.595656
std	4.643746e+04	34831.724018
min	1.200000e+02	0.000000
25%	7.495000e+03	10481.000000
50%	1.260000e+04	28629.500000
75%	2.000000e+04	56875.750000
max	9.999999e+06	999999.000000

In [113]:
```
car_adv['fuel_type'].value_counts()
```

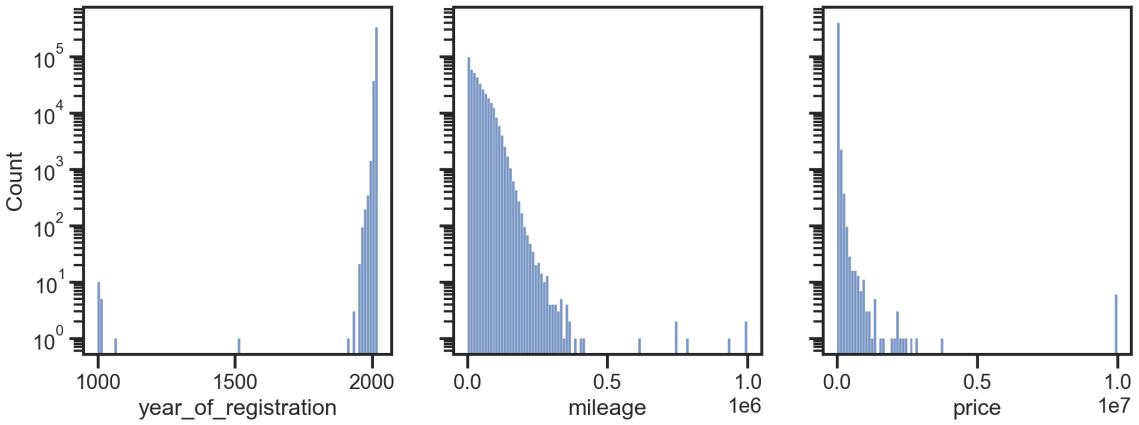
Out[113]:
```
Petrol                  216929
Diesel                  158120
Petrol Hybrid            13602
Petrol Plug-in Hybrid     6160
Electric                  4783
Diesel Hybrid             1403
Bi Fuel                    221
Diesel Plug-in Hybrid      185
Natural Gas                  1
Name: fuel_type, dtype: int64
```

1.1.3.1 Analysis of distribution of numerical features

Analyzing the distribution of numerical features in a dataset can be done using a variety of statistical approaches, such as histograms and box plots. The distribution of each numerical feature, including the form of the distribution, the existence of outliers, and the range of values, will be represented visually by these plots. Since the given dataaset is very large, histograms are used to analyse the distribution of numerical features

- The given dataset has the following numerical features:
  - public_reference
  - mileage
  - year_of_registration
  - price

In [114]:
```
fig, axs = plt.subplots(1, 3, figsize=(12,4), sharey=True)
sns.histplot(data=car_adv, x='price', bins=100)
sns.histplot(data=car_adv, x='year_of_registration',ax=axs[0], bins=100)
sns.histplot(data=car_adv, x='mileage',ax=axs[1], bins=100)
plt.yscale('log')
```



Above figure shows the distribution of numerical data in the dataset.

The majority of the data is within the span of 1900 to 2020. It is possible to classify the years displayed below 1600 as outliers. The inaccuracies in this column can be examined and corrected using the provided registration code.

The majority of vehicles have a mileage less 400000.

The price in the dataset is positively skewed. Skewness is a measure of the asymmetry of a real-valued random variable's probability distribution relative to its mean. Skewness can be used to evaluate the asymmetry of the price distribution in this dataset. Positive skewness means that the probability density function's right side's tail is longer or fatter than its left side.

1.1.3.2 Analysis of distribution of categorical features

Analyzing the distribution of numerical features in a dataset can be done using a variety of statistical approaches, such as Frequency tables and Bar charts

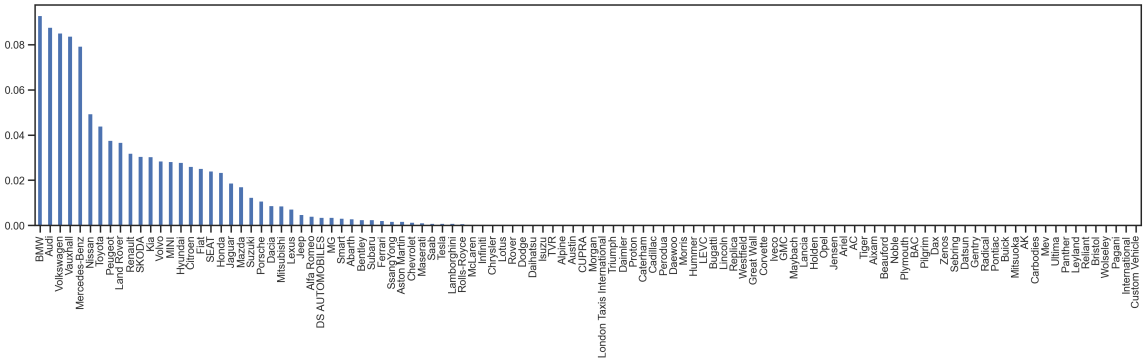
- Frequency tables display how frequently each category in a feature occurs
- A frequency table is graphically represented by a bar chart. It can be used to compare the distribution of a single feature across various datasets or the distribution of various

categorical features.

- The given dataset has the following categorical features:
  - reg_code
  - standard_colour
  - standard_make
  - standard_model
  - vehicle_condition
  - body_type
  - fuel_type

In [115]:
```
plt.figure(figsize=(25, 5))
car_adv['standard_make'].value_counts(normalize=True).plot.bar()
```

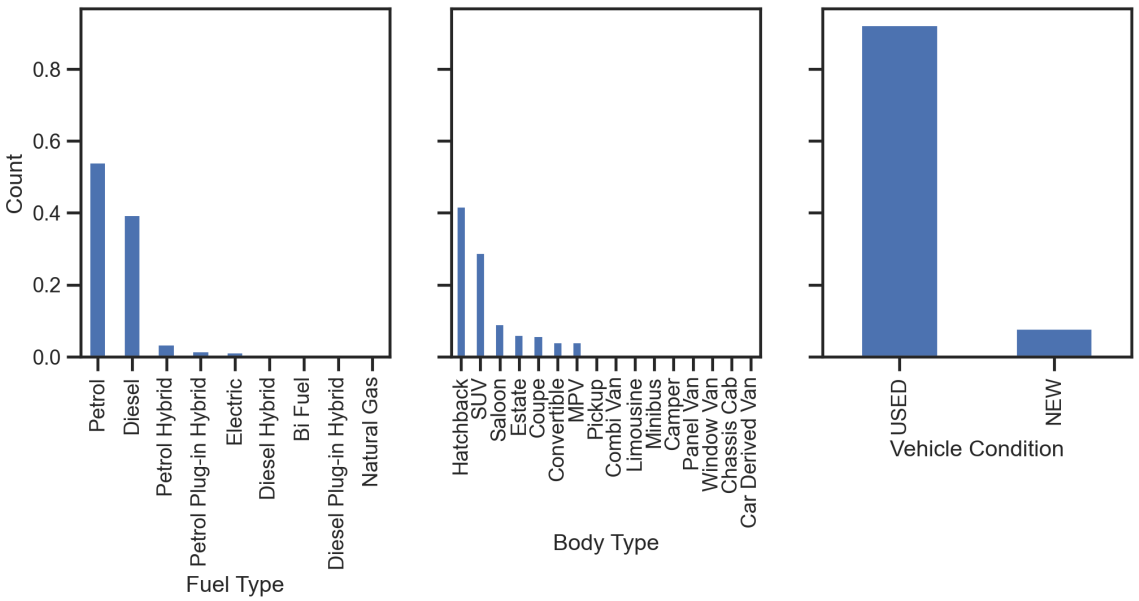
Out[115]: <AxesSubplot:>



From the above figure, it is visualized that most occurred Brand of dataset are premium group of vehicles such as BMW, Audi, Volkswagen, Vauxhall, Mercedese Benz etc.

In [180]:
```
fig, axs = plt.subplots(1, 3, figsize=(12,4), sharey=True)
car_adv['fuel_type'].value_counts(normalize=True).plot.bar(
    ax=axs[0], xlabel='Fuel Type', ylabel='Count')
car_adv['body_type'].value_counts(normalize=True).plot.bar(
    ax=axs[1], xlabel='Body Type')
car_adv['vehicle_condition'].value_counts(normalize=True).plot.bar(
    ax=axs[2], xlabel='Vehicle Condition')
```

Out[180]:
```
<AxesSubplot:xlabel='Vehicle Condition'>
```

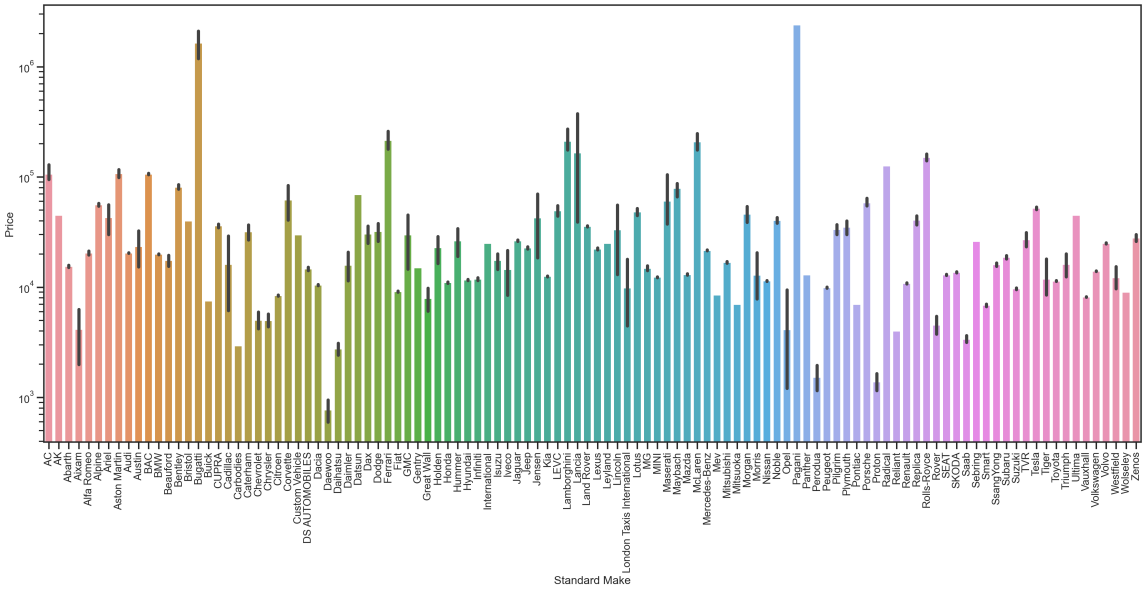


From the above figure, it is clear that most of the sample are used Hatchback or SUV type vehicle with petrol or diesel fuel.

1.1.4 Data visualization

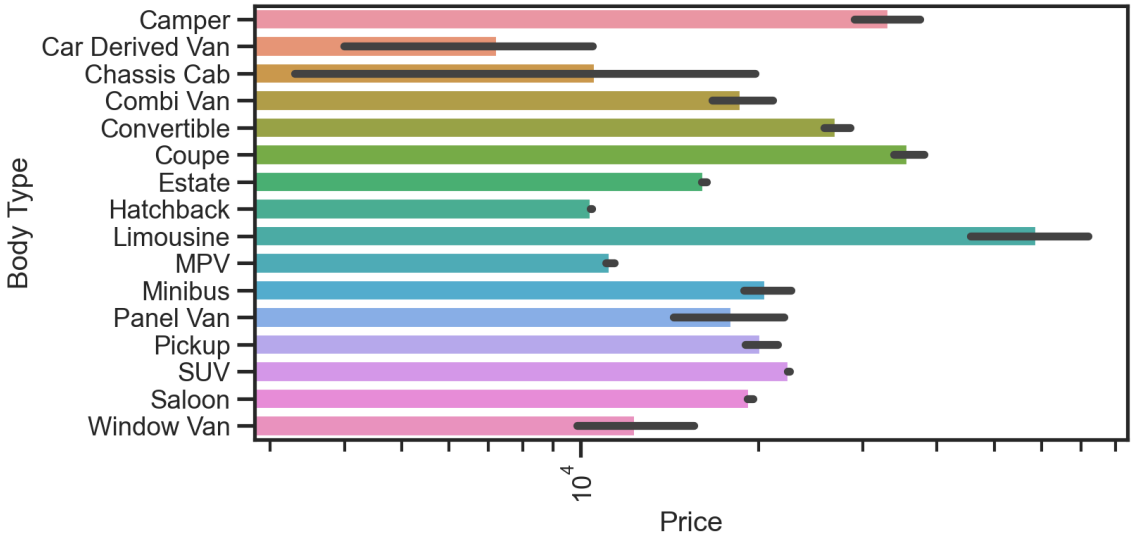
In [117]:
```
plt.figure(figsize=(25, 10))
sns.barplot(x='standard_make', y='price',
            data=car_adv.sort_values(by='standard_make'))
plt.xticks(rotation=90)
plt.yscale('log')
plt.xlabel('Standard Make')
plt.ylabel('Price')
```

Out[117]:
```
Text(0, 0.5, 'Price')
```



According to the above figure, the most costly car brands in the provided dataset are Pagani and Bugatti. Following closely behind are Ferrari, Lamborghini, McLaren, and Rolls-Royce.

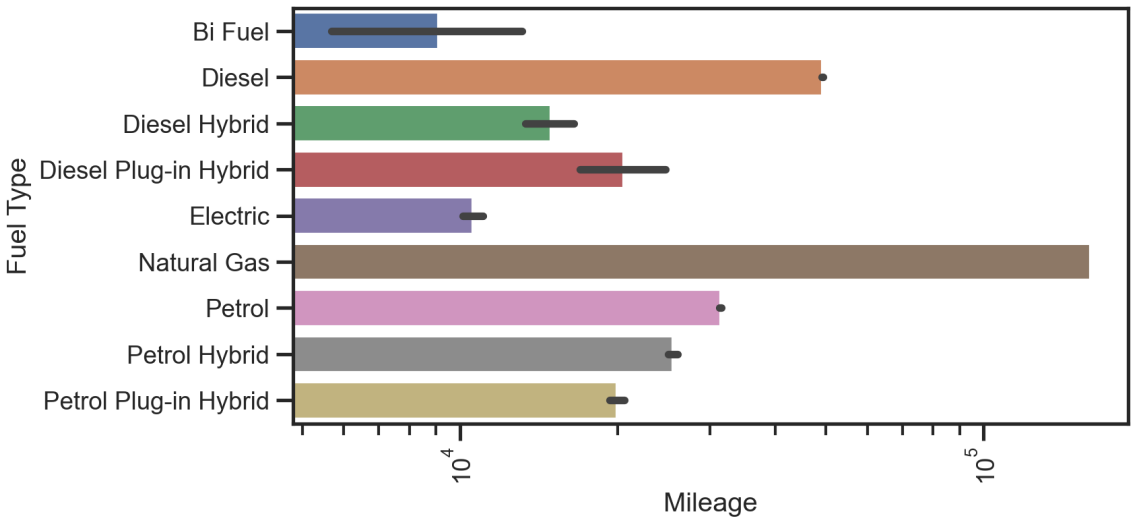
In [118]:
```
plt.figure(figsize=(8, 4))
sns.barplot(y='body_type', x='price',
            data=car_adv.sort_values(by='body_type'))
plt.xticks(rotation=90)
plt.xscale('log')
plt.ylabel('Body Type')
plt.xlabel('Price');
```



On the comparison of Body type Vs Price in the above figure, it can be concluded that

Limousine vehicle are the most expensive category it the given dataset and is followed by Coupe, Camper, Convertible and SUV respectively.

In [119]:
```
plt.figure(figsize=(8, 4))
sns.barplot(y='fuel_type', x='mileage',
            data=car_adv.sort_values(by='fuel_type'))
plt.xticks(rotation=90)
plt.xscale('log')
plt.ylabel('Fuel Type')
plt.xlabel('Mileage');
```



From the comparison of Fuel type Vs Mileage, it is evident that Natural gas cars are the most fuel-efficient vehicle in the dataset. It is followed by Diesel and Petrol vehicles respectively.

1.2. Identification/Commenting on Missing Values

It's crucial to recognise and deal with missing values in a dataset since they might significantly affect the analysis's precision and interpretability. Identifying missing values in a dataset can be done in a number of ways as follows.

- .isnull()
- Heat map

In [120]:
```
car_adv.isnull().sum()
```

Out[120]:
```
public_reference            0
mileage                   127
reg_code                31857
standard_colour          5378
standard_make               0
standard_model              0
vehicle_condition           0
year_of_registration    33311
price                       0
body_type                 837
crossover_car_and_van       0
fuel_type                 601
dtype: int64
```

The .isnull() method is used to determine whether the elements of a DataFrame are null or NaN (Not a Number). The.sum() function is used to find the total count.

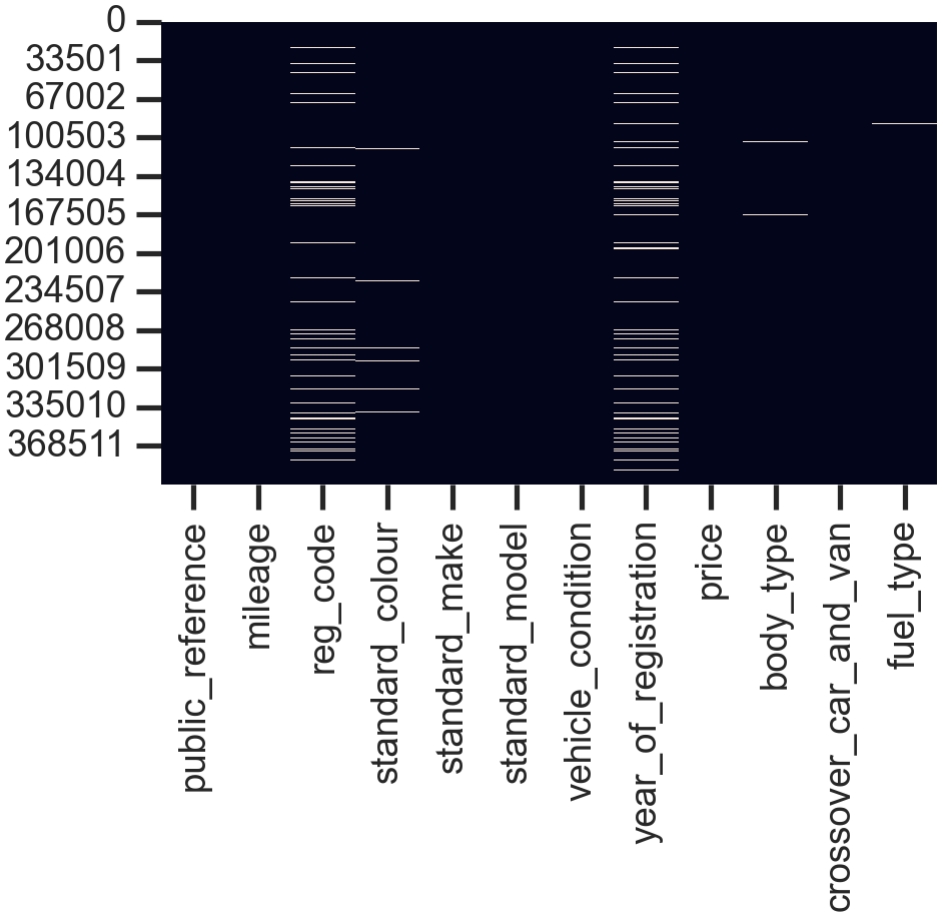
In the given dataset, the maximum number of missing values occured in the `year_of_registration` which can be fixed by the given `reg_code` .

The missing values in the `mileage` can be replaced with the mean value of that column since it is a numerical value.

Similarly, the missing values in the `standard_colour` , `body_type` , `fuel_type` can be replaced by the mod values of repective columns since they are categorical values.

In [121]:
```
plt.figure(figsize=(5, 3))
sns.heatmap(car_adv.isna(), cbar=False)
```

Out[121]: <AxesSubplot:>



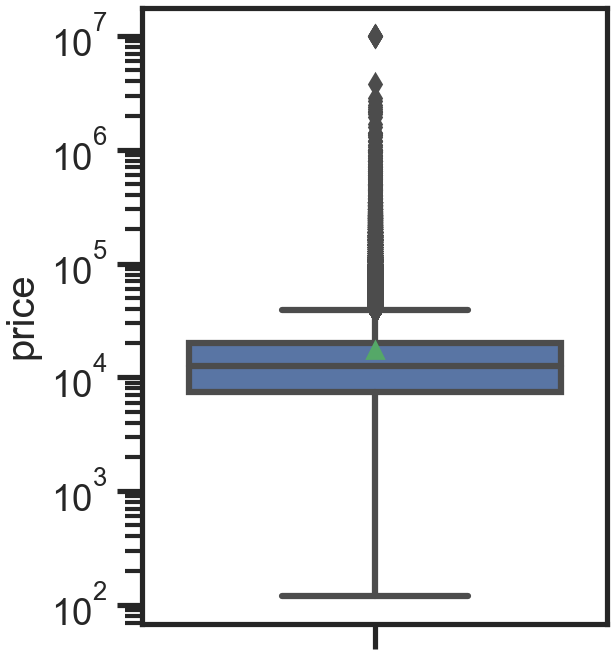
The above heat map gives the visualization of null values in the given dataset.

1.3. Identification/Commenting on Outliers and Noise

It's critical to recognise and deal with outliers and noise in a way that preserves the analysis' accuracy and interpretability. There are a number of approaches to spot outliers and noise in a dataset, including the following.

1.3.1 Identification/Commenting on Outliers

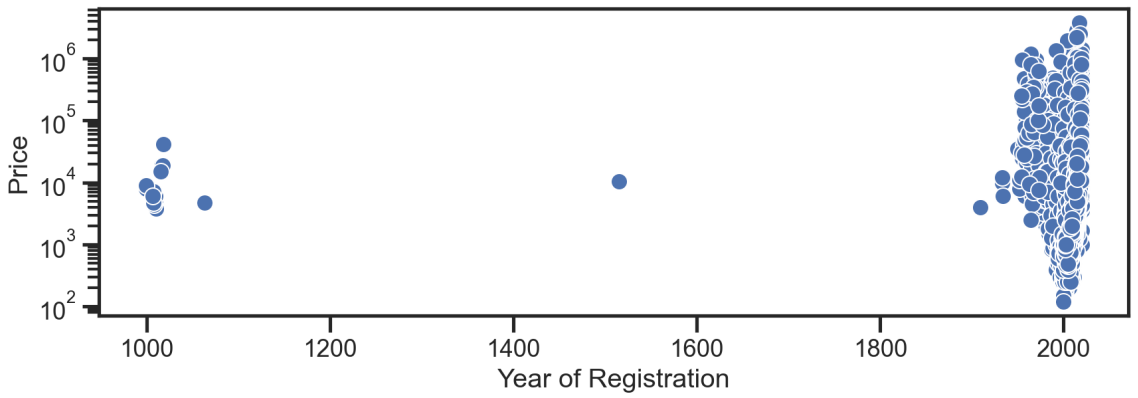
In [122]:
```
plt.figure(figsize=(3, 4))
sns.boxplot(y='price', showmeans=True, data=car_adv).set_yscale('log')
```



The boxplot in the image above displays the `price` outliers. These outliers are thought to be the result of a few expensive cars' prices in the dataset.

In [123]:
```
plt.figure(figsize=(10, 3))
sns.scatterplot(data=car_adv, y = "price", x = "year_of_registration")
plt.yscale('log')
plt.xlabel('Year of Registration')
plt.ylabel('Price')
```

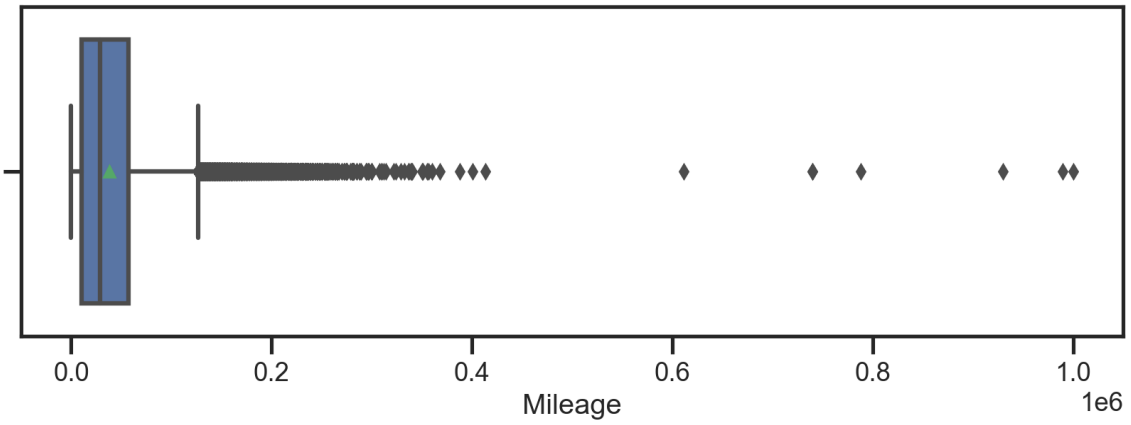
Out[123]:
```
Text(0, 0.5, 'Price')
```



The outliers in `year of registration` are displayed in the scatter plot above. There must be a data entry error for the years displayed below 1600. The reg_code provided in the dataset can be used to correct these issues.

In [124]:
```
plt.figure(figsize=(10, 3))
sns.boxplot(x='mileage', showmeans=True, data=car_adv)
plt.xlabel('Mileage')
```

Out[124]:
```
Text(0.5, 0, 'Mileage')
```



Data points outside the lower and upper quartile ranges, indicated by the lower and upper hinges of the box, are known as outliers in a boxplot. They could be the result of a number of things, including measurement error, data entry error, or just being a valid extreme value in the dataset. The figure shows the outliers in `mileage`. I made the choice to exclude the rows with mileage of more than 400000.

1.3.2 Identification/Commenting on Noice

In [125]:
```
car_adv.duplicated().sum()
```

Out[125]:
```
0
```

The `.duplicated()` function is used to locate the noises in the provided dataset. The outcome demonstrates that the dataset has no duplicate values.

2.0 Data Processing

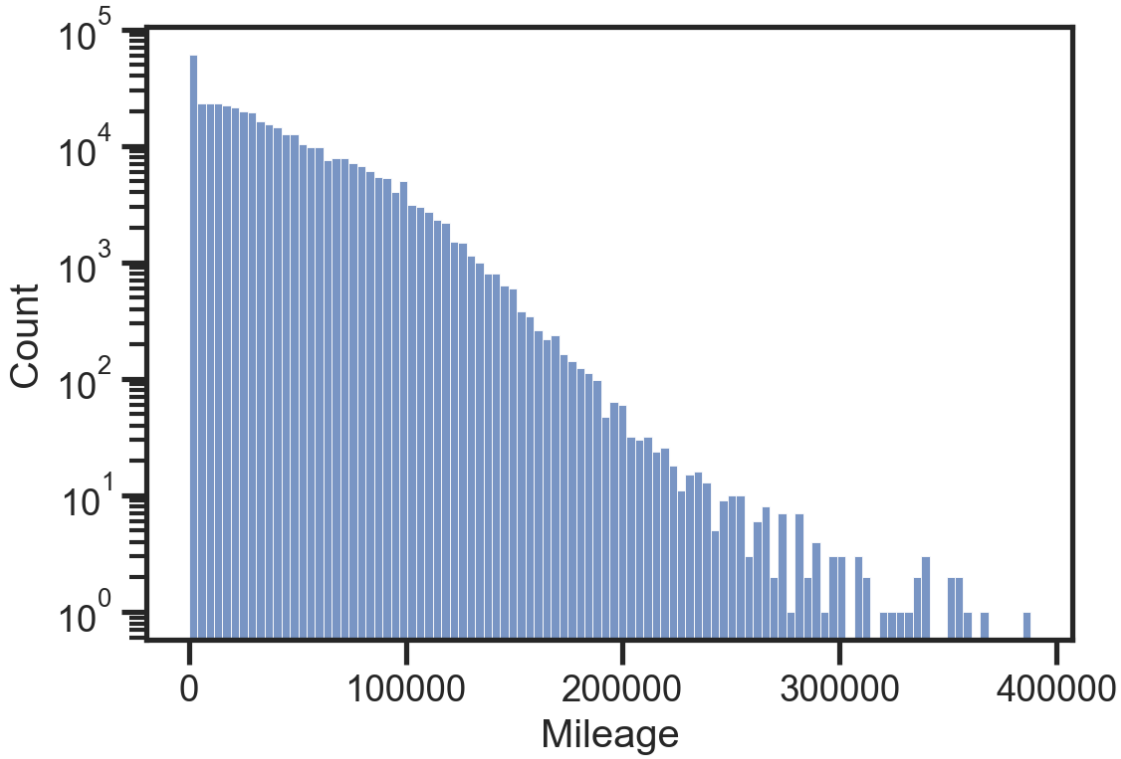
2.1. Dealing with Missing Values, Outliers, and Noise

- Replacing missing values and removing the outliers in mileage

In [126]:
```
car_adv.loc[:, 'mileage'] = (
    car_adv
    .loc[:, 'mileage']
    .fillna(car_adv.loc[:, 'mileage'].mean())
)

car_adv = car_adv.drop(car_adv[car_adv['mileage'] > 400000].index, axis=0)
sns.histplot(data=car_adv, x='mileage', bins=100).set_yscale('log')
plt.xlabel('Mileage')
```

Out[126]:
```
Text(0.5, 0, 'Mileage')
```



By using the above code, the `mileage` column's missing values are replaced with the column's mean value, and the rows that include mileage outliers are eliminated. The current distribution of the mileage column is displayed in the histogram.

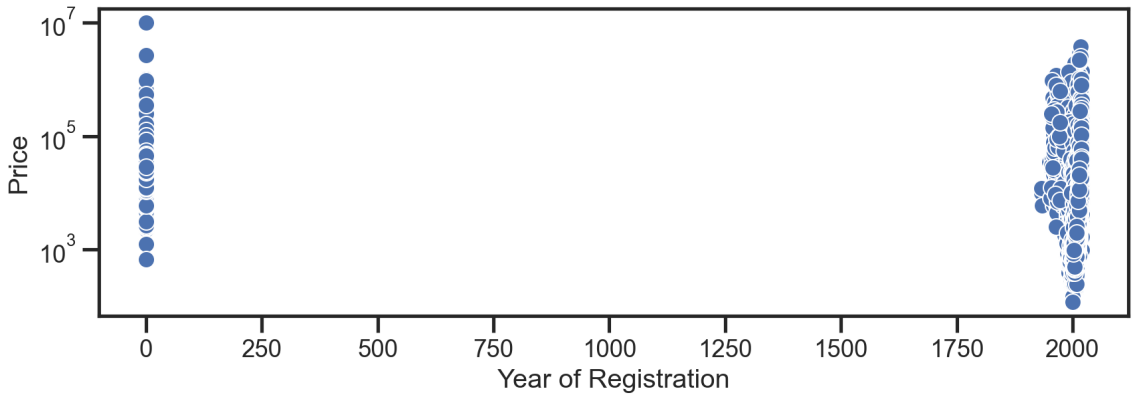
- Replacing missing values and outliers in year of registration

In [127]:
```
# Defining a function to find year_of_registration
def find_year(year_of_registration,reg_code,vehicle_condition):
    if(vehicle_condition!="NEW"):
        if reg_code.isdigit():
            if 0< int(reg_code) < 23 :
                out = int(reg_code)+2000
            elif 23 <= int(reg_code) < 73:
                out = int(reg_code)+1950
            else:
                  out=year_of_registration
        else:
            out=year_of_registration
    else :
            out=year_of_registration
    return out

# Calling the function
car_adv['year_of_registration'] =car_adv.apply(
    lambda x: find_year((x['year_of_registration']),
                        str(x['reg_code']),(x['vehicle_condition'])),
    axis=1)
```

In [128]:
```
# Replacing the missing values with zero
car_adv['year_of_registration'] = car_adv['year_of_registration'].fillna(0)
plt.figure(figsize=(10, 3))
sns.scatterplot(data=car_adv, y = "price",
                x = "year_of_registration").set_yscale('log')
plt.xlabel('Year of Registration')
plt.ylabel('Price')
```

Out[128]:
```
Text(0, 0.5, 'Price')
```



The seven-character registration index used in the current automobile registration system in Great Britain has a specific format. Initially a local memory tag, or area code, composed of up to two letters that collectively identify the local registration office. Then a two-digit age identifier that shifts twice a year, on the first of March and September, respectively. If the code is issued between March and August, it is either the final two digits of the current year or, if it is issued between September and February of the next year, it is the current year plus 50. Finally a three-letter sequence that specifically identifies each of the cars displaying the same starting four-character area and age sequence.

In the given dataset, the two-digit age identifiers of every vehicle are given as `reg_code` .

The function to determine the year of registration for all rows other than those with a vehicle

condition of "NEW" is first defined in the code above. The 'NEW' vehicle condition designates a brand-new vehicle, and the NEW car's `year_of_registration` column is empty (null). Then the function executed for each row. Running the code allows the `year_of_registration` column's outliers to be updated with the appropriate year of registration.

In the next part of code, the null values in the `year_of_registration` column are replaced by zero. The outliers shown in above figure are caused by the replace 0 for new cars.

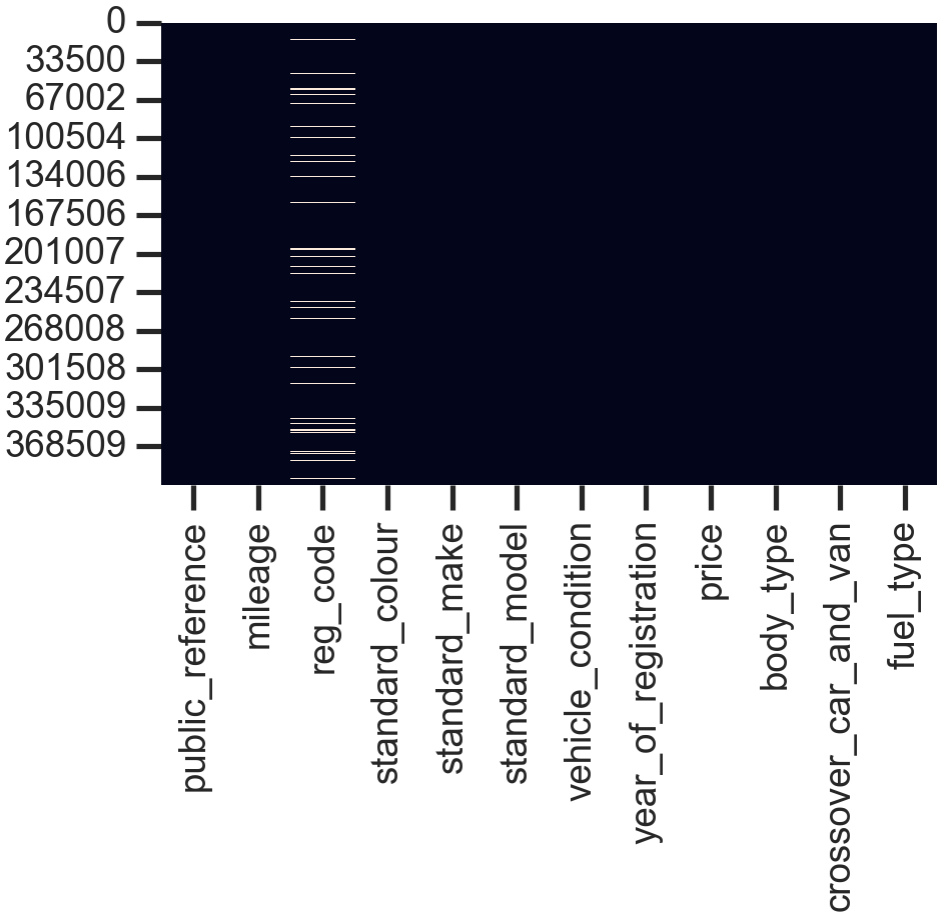
- Replacing missing values in categorical features with mode value

In [129]:
```
car_adv['fuel_type'] = car_adv['fuel_type'].fillna(
    car_adv['fuel_type'].mode().iloc[0])
car_adv['standard_colour'] = car_adv['standard_colour'].fillna(
    car_adv['standard_colour'].mode().iloc[0])
car_adv['body_type'] = car_adv['body_type'].fillna(
    car_adv['body_type'].mode().iloc[0])
```

The category columns with empty values are `fuel_type` , `standard_colour` , and `body_type` . All of these missing values are substituted with the appropriate column's mode value.

In [130]:
```
plt.figure(figsize=(5, 3))
sns.heatmap(car_adv.isna(), cbar=False)
```

Out[130]: <AxesSubplot:>



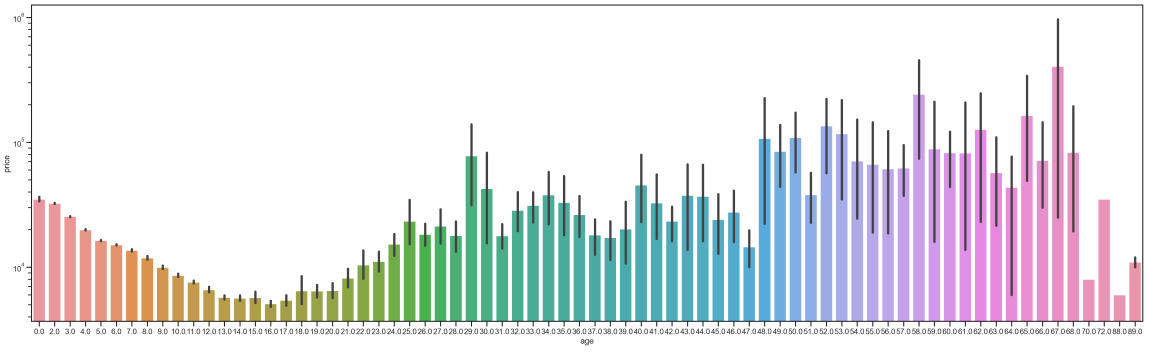
The dataset is depicted in the heatmap above after missing values, outliers, and noise have been addressed. Since the `reg_code` column is solely needed to determine the year of registration, missing entries there can be ignored.

2.2 Feature Engineering, Data Transformations

In machine learning, feature engineering is a crucial stage. The effectiveness of a model can be significantly impacted by feature engineering. In order to enhance the performance of the model, features are created, tested, and modified during this phase.

- Creating new features

In [131]:
```
def age(year_of_registration):
    if (year_of_registration == 0):
        out = 0
    else:
        out = 2022-year_of_registration
    return out
car_adv['age'] = car_adv.apply(lambda x: age((x['year_of_registration'])),
                               axis=1)
plt.figure(figsize=(35, 10))
sns.barplot(x='age', y='price', data=car_adv)
plt.yscale('log')
```



The year of the car is determined for each row using the code above, and a new column named `age` is generated.

In [132]:
```
def vehicle_condition(vehicle_condition, age, mileage):
    if (vehicle_condition == "USED"):
        if age <= 15 and mileage <=150000:
            return 1
        elif 15 <= age <= 25 and (0 < mileage < 500000):
            return 2
        else:
            return 3
    else:
        return 0
car_adv["condition"] = car_adv.apply(lambda x: vehicle_condition(
    (x['vehicle_condition']),(x['age']),(x['mileage'])), axis=1)
car_adv.sample(1)
```

Out[132]:

	public_reference	mileage	reg_code	standard_colour	standard_make	standard_mod
312071	202010074732342	6107.0	67	Grey	Renault	Kadj

To further categorise the dataset, a new column named `condition` is constructed based on the `vehicle_condition` column. The above code divides the dataset into four categories based on age and mileage.

"0" for brand-new vehicles.

"1" for fair vehicles under 15 years old with fewer than 150000 miles.

"2" for typical vehicles between the ages of 15 and 25 and with mileage between 0 and 500,000.

"3" for classic cars is where all other vehicles fall.

- Dropping unnecessory columns

In [133]:
```
car_adv = car_adv.drop(labels=['public_reference','reg_code',
                               'vehicle_condition','standard_model'],
                       axis=1)
car_adv.sample(1)
```

Out[133]:

	mileage	standard_colour	standard_make	year_of_registration	price	body_type	cr
252194	20343.0	Black	Hyundai	2017.0	13295	SUV	

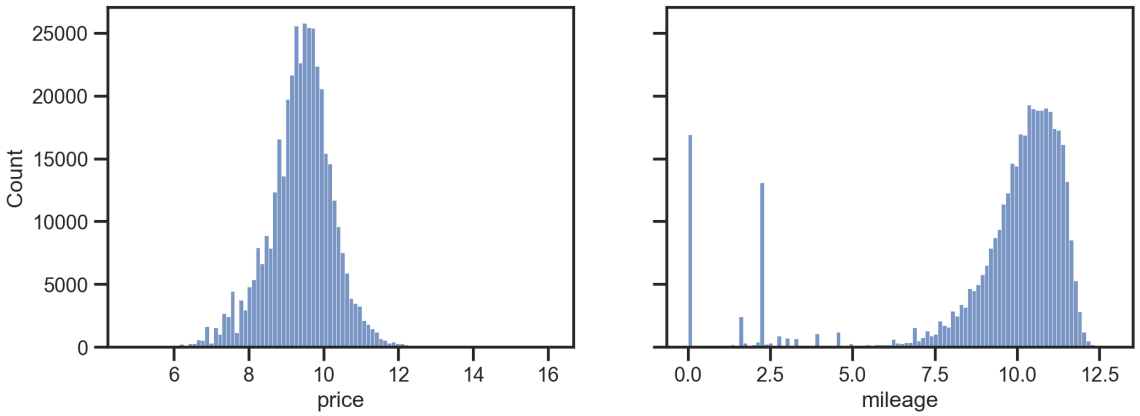
The columns `public reference`, `reg code`, `vehicle_condition`, and `standard model` are no longer relevant to our model. So they're taken out

- Data Transformation using natural logarithm

A typical transformation to alter a variable's distribution is the natural logarithm, especially when the distribution of the value is skewed. In our dataset both price and mileage are skewed towards the right. For better performance of our model both the features are transformed to natural logarithm.

In [134]:
```
def mileage(mileage):
    if (mileage == 0):
        out = mileage
    else:
        out = np.log(mileage)
    return out
car_adv['mileage'] =car_adv.apply(lambda x: mileage((x['mileage'])),
                                  axis=1)
car_adv['price'] = np.log(car_adv['price'])
fig, axs = plt.subplots(1, 2, figsize=(12,4), sharey=True)
sns.histplot(data=car_adv, x='price', bins=100, ax=axs[0])
sns.histplot(data=car_adv, x='mileage', bins=100, ax=axs[1])
```

Out[134]:
```
<AxesSubplot:xlabel='mileage', ylabel='Count'>
```



From the above histograms it is evident thath the skewness of both price and mileage is decreased. Price is normally distributed now. Mileage is having sight skewness towards left which may be caused by the new cars in the dataset.

- Data Transformation using label encoding

Majority of features in the given dataset are categorical. Since machine learning algorithms can only work with numerical values, these features must be transformed into numerical forms in order to be used in these methods. After transformation using label encoding, a unique number will be assigned to each distinct category.

In [135]:
```
car_adv_copy = car_adv.copy()
car_adv.select_dtypes(include=['object']).columns
```

Out[135]:
```
Index(['standard_colour', 'standard_make', 'body_type', 'fuel_type'], dtyp
e='object')
```

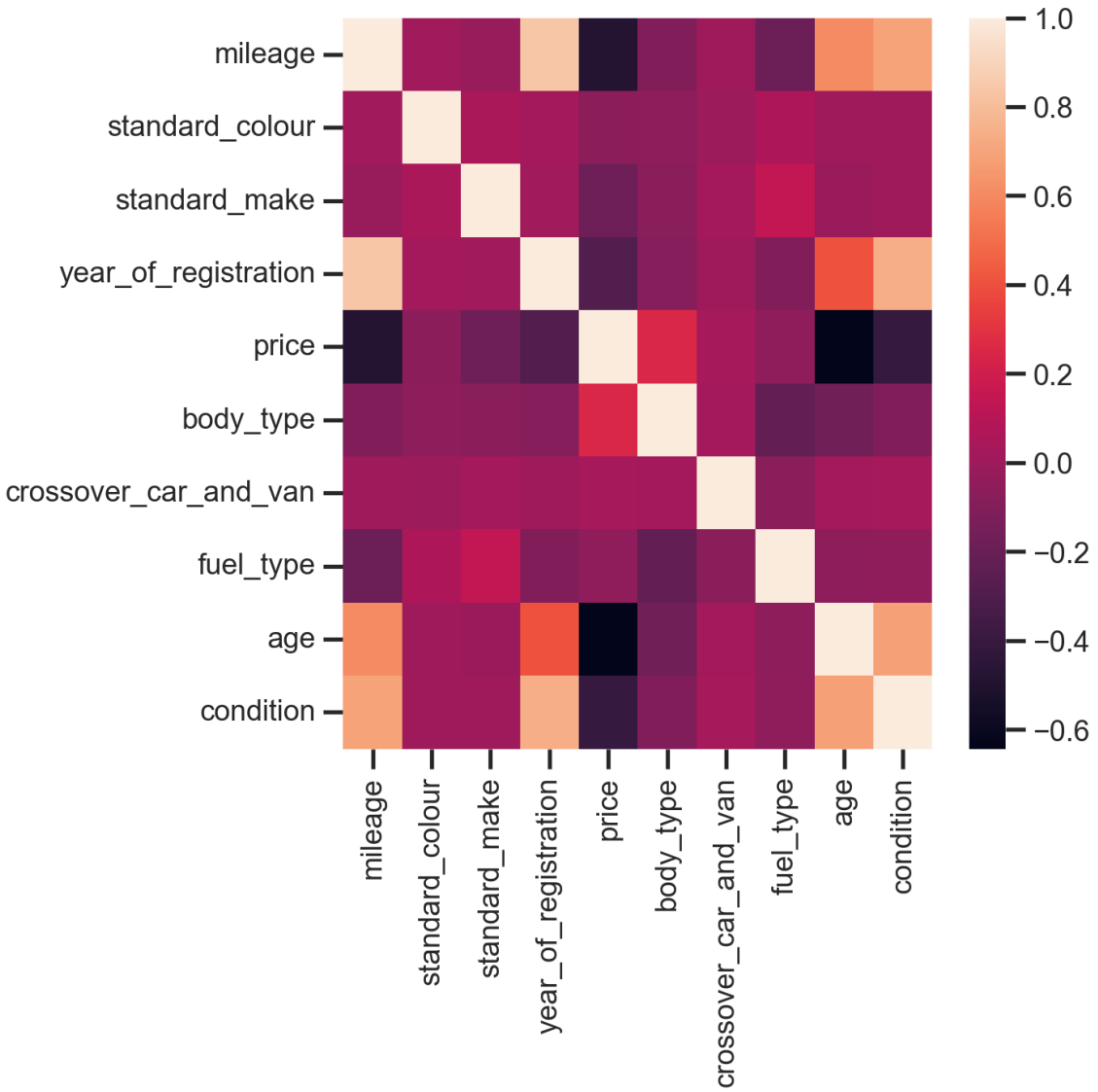
In [136]:
```
from sklearn.preprocessing import LabelEncoder
# Initialize the encoder
encoder = LabelEncoder()

# Select the categorical columns
categorical_cols = ["standard_make", 'standard_colour', 'body_type',
                    'fuel_type']

# Apply the encoder to each column
for col in categorical_cols:
    car_adv[col] = encoder.fit_transform(car_adv[col])
```

In [137]:
```
plt.figure(figsize=(6, 6))
sns.heatmap(car_adv.corr(), annot=False)
```

Out[137]: <AxesSubplot:>



The correlation between the features of the dataset is displayed in the heatmap created after data transformation. This map clearly demonstrates the negative correlation between the target class and the attributes `age`, `mileage`, `condition`, `year_of_registration`, `standard_make`, `standard_colur`, and `fuel_type`. Hence, negatively affecting the `price`.

2.3 Subsetting (e.g., Feature Selection, Data Sampling)

A subset of data is chosen from a larger dataset through the process of subsetting. This may be carried out for a number of purposes, such as feature selection (selecting a subset of features to be used in a model) or data sampling (randomly selecting a subset of data to use for training or testing a model). A model's performance, computational efficiency, and complexity can all be improved using subsetting.

- Selecting Features

In [147]:
```
Brand_new_car = car_adv.loc[car_adv['condition'] == 0]
Brand_new_car.shape
```

Out[147]: (31249, 10)

In [148]:
```
Fair_car = (car_adv.loc[car_adv['condition'] == 1 ])
Fair_car.shape
```

Out[148]: (351936, 10)

In [149]:
```
Average_car = car_adv.loc[car_adv['condition'] == 2]
Average_car.shape
```

Out[149]: (16176, 10)

In [150]:
```
Other_car = car_adv.loc[car_adv['condition'] == 3]
Other_car.shape
```

Out[150]: (2635, 10)

The dataset is split into four subsets in accordance with the condition generated from the age and mileage of the cars, as these two factors have the greatest impact on price prediction. Each category may employ a different set of machine learning techniques for price prediction. For instance, the vehicles included in the Vintage automobile subset can have fancy price. So they may be more expensive than those in the other subsets. According to the characteristics of the subset, we must choose the techniques.

- Removing Features

In [151]:
```
Brand_new_car = Brand_new_car.drop(labels=['year_of_registration', 'age',
                                           'condition'], axis=1)
Fair_car = Fair_car.drop(labels=['condition'], axis=1)
Average_car = Average_car.drop(labels=['condition'], axis=1)
Other_car = Other_car.drop(labels=['condition'], axis=1)
```

The unnecessary features are eliminated after subsetting the dataset.

- Smapling

The act of sampling is the random selection of a subset of data from a larger dataset.

Making a representative sample of the data for training and testing models is a standard practise in machine learning

In [152]:
```
sample_data = car_adv.sample(frac=0.1, random_state=50)
sample_data.shape
```

Out[152]:
```
(40200, 10)
```

10% of the original dataset with a random state of 50 is sampled from "car adv".

3.0 Association and Group Differences Analysis

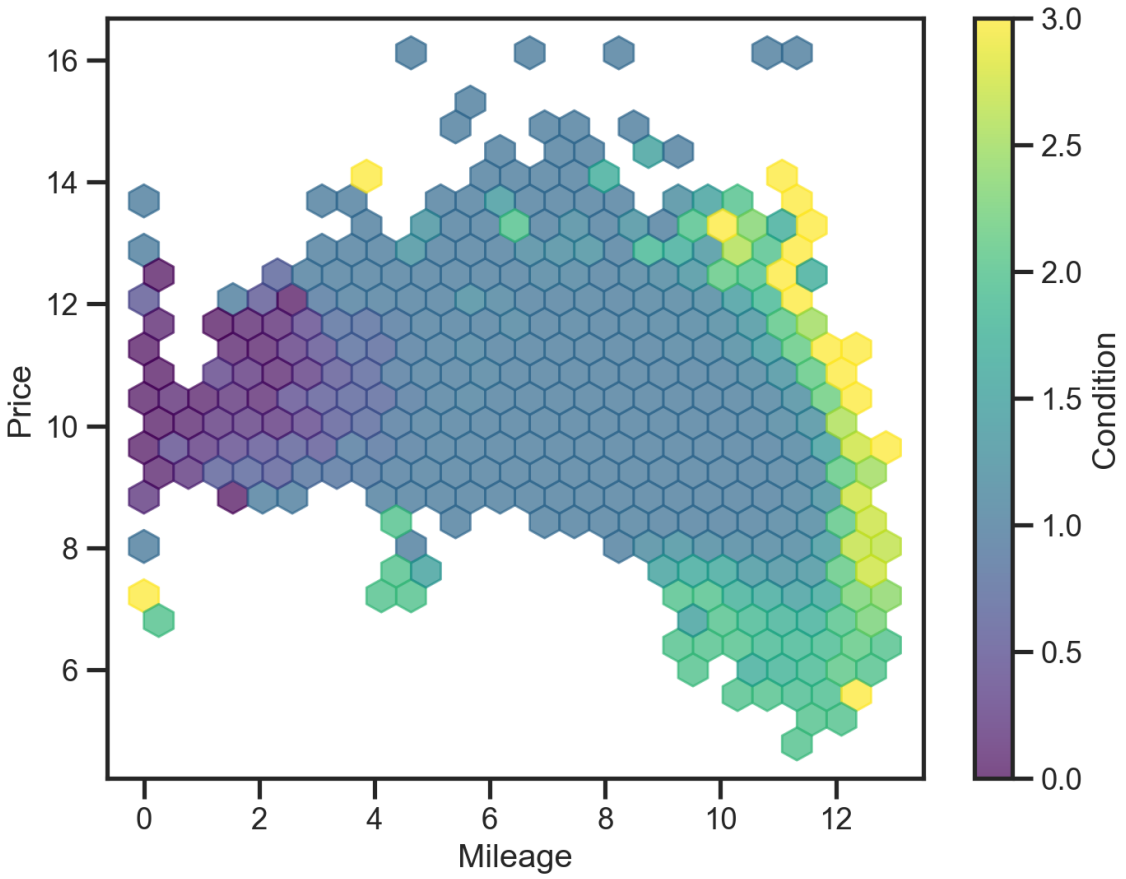
To ascertain the relationship between two or more variables, association analysis is utilised. On the other hand, group differences analysis is performed to identify whether there are significant differences between two or more groups of data.

3.1. Quantitative-Quantitative

The relationship between two numerical variables is referred to as quantitative-quantitative association. It is used to establish whether the two variables are correlated or associated.

In [164]:
```
plt.figure(figsize=(8,6))
plt.hexbin(car_adv['mileage'], car_adv['price'], C=car_adv['condition'],
           gridsize=25, cmap='viridis', alpha=0.7)
plt.ylabel('Price')
plt.xlabel('Mileage')
plt.colorbar(label='Condition')
```

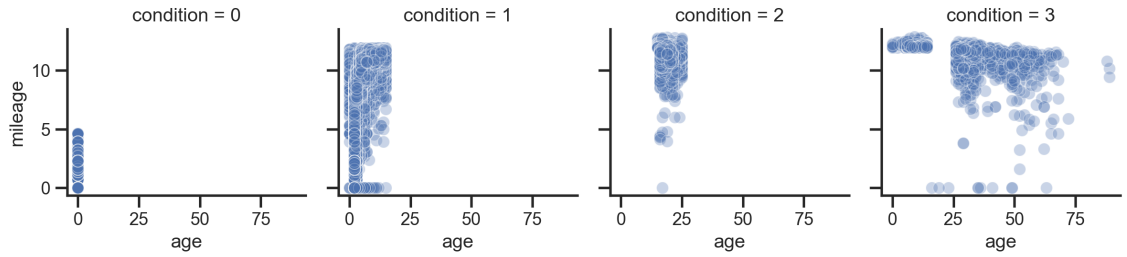
Out[164]:
```
<matplotlib.colorbar.Colorbar at 0x46dd76a940>
```



Above figure shows the relationship between mileage, price and condition of the dataset. It is clear that as the mileage increases, the price get decreases. The brand new cars (condition = 0) are costly compare to the other groups whereas the price of fair cars (condition = 1) varies from medium to high.The average cars (condition = 2) are cheaper compare to other groups since thair mileage is very high. The cars which are not fall in these three groups (condition = 4) not showing a specific patern. Most of them are having high mileage, but their price is varies from low to high. The high cost of this group may be caused by the classic cars in that group.

In [165]:
```
g = sns.FacetGrid(car_adv, col='condition')
g.map(sns.scatterplot, 'age', 'mileage', alpha=.3)
g.add_legend()
```

Out[165]:
```
<seaborn.axisgrid.FacetGrid at 0x46d2ecec40>
```



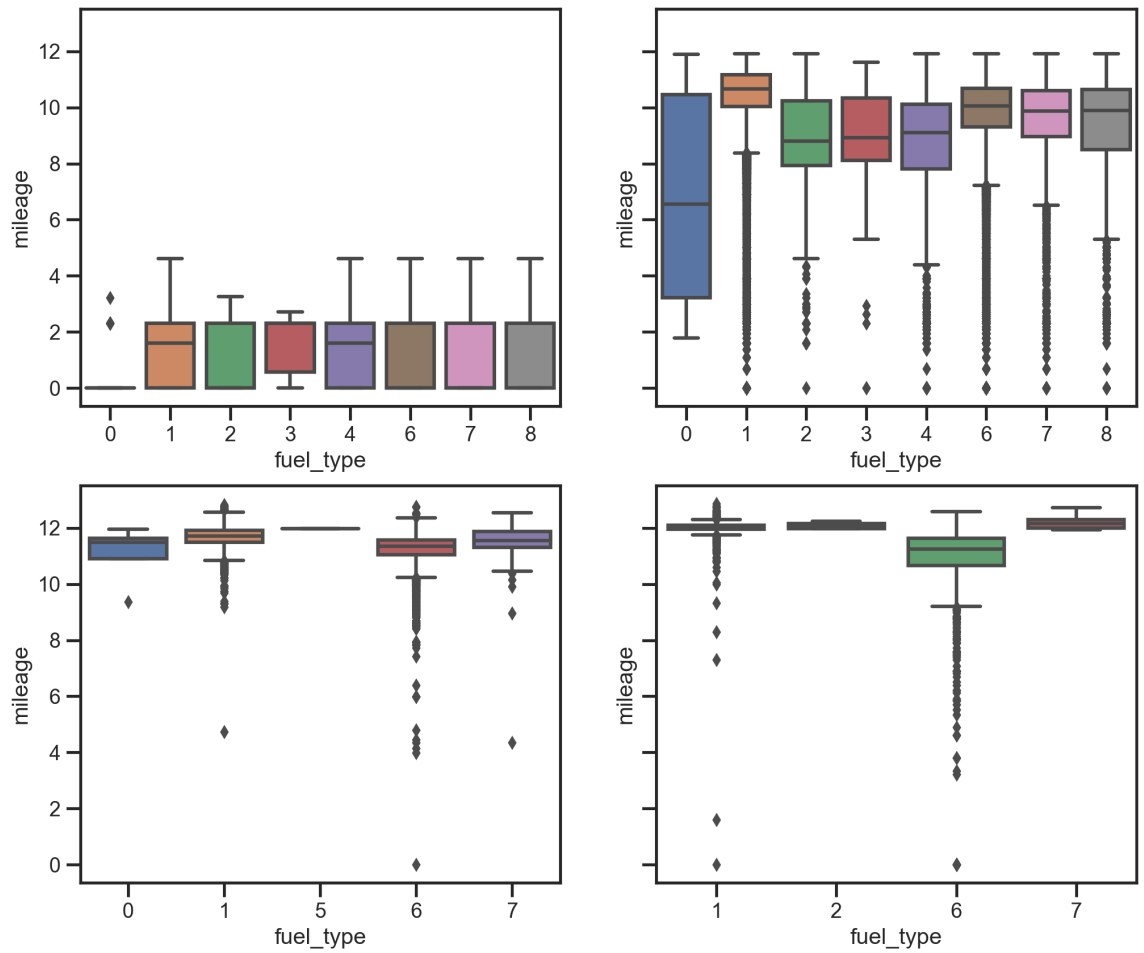
Above figures shows the relationship between age and price of the dataset. As the age increases mileage is also increases. This will affect the price of a car.

3.2. Quantitative-Categorical

In [168]:
```
fig, axs = plt.subplots(2, 2, figsize=(12,10), sharey=True)

sns.boxplot(data=Brand_new_car, x='fuel_type', y='mileage', ax=axs[0,0])
sns.boxplot(data=Fair_car, x='fuel_type', y='mileage', ax=axs[0,1])
sns.boxplot(data=Average_car, x='fuel_type', y='mileage', ax=axs[1,0])
sns.boxplot(data=Other_car, x='fuel_type', y='mileage', ax=axs[1,1])
```

Out[168]:
```
<AxesSubplot:xlabel='fuel_type', ylabel='mileage'>
```

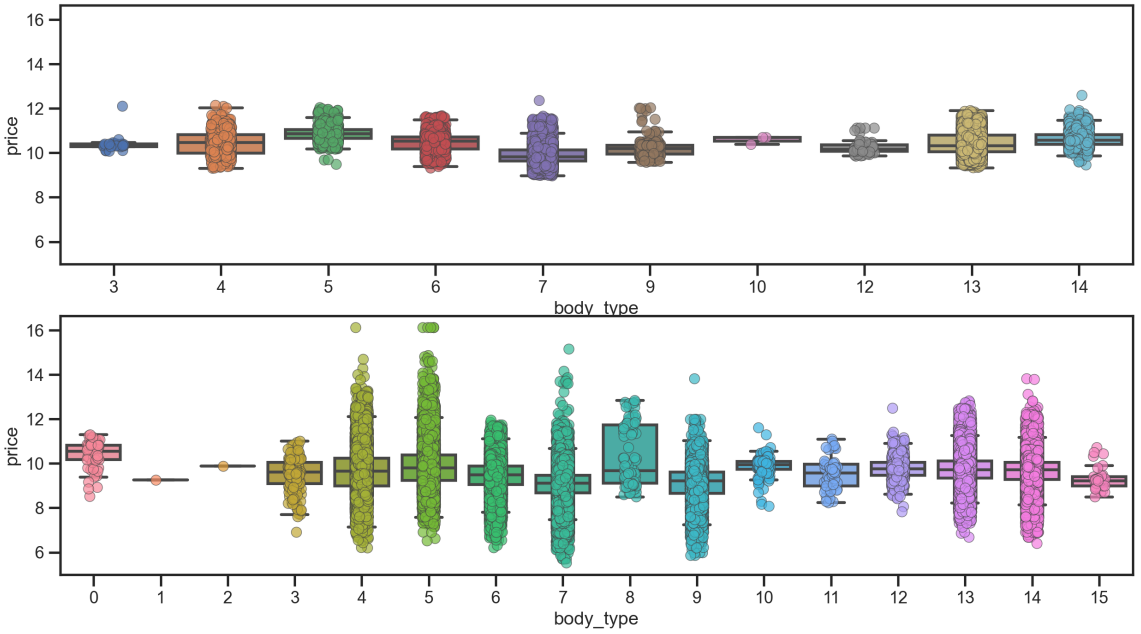


Above figures shows the relationship between fuel_type and mileage in four subsets of the dataset.

In [169]:
```
fig, axs = plt.subplots(2, 1, figsize=(15,8), sharey=True)
sns.boxplot(data=Brand_new_car, x='body_type', y='price', ax=axs[0],
            showfliers=False)
sns.stripplot(data=Brand_new_car, x='body_type', y='price', ax=axs[0],
              linewidth=0.5, jitter=0.1, size=8, alpha=0.7)

sns.boxplot(data=Fair_car, x='body_type', y='price', ax=axs[1],
            showfliers=False)
sns.stripplot(data=Fair_car, x='body_type', y='price', ax=axs[1],
              linewidth=0.5, jitter=0.1, size=8, alpha=0.7)
```

Out[169]:
```
<AxesSubplot:xlabel='body_type', ylabel='price'>
```

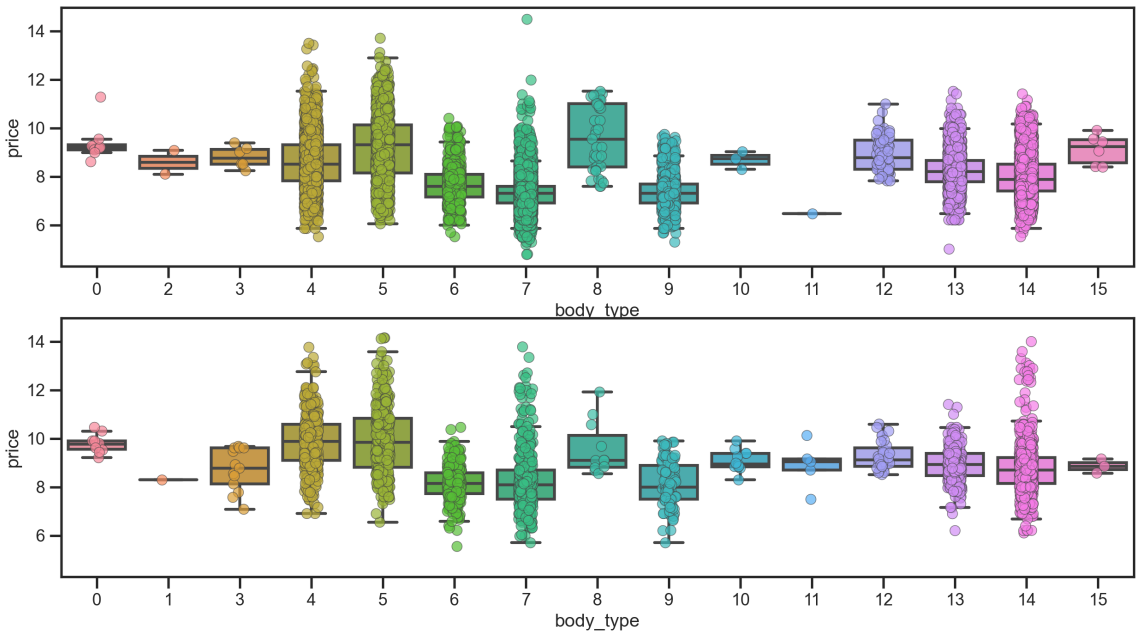


In [171]:
```
fig, axs = plt.subplots(2, 1, figsize=(15,8), sharey=True)

sns.boxplot(data=Average_car, x='body_type', y='price', ax=axs[0],
            showfliers=False)
sns.stripplot(data=Average_car, x='body_type', y='price', ax=axs[0],
              linewidth=0.5, jitter=0.1, size=8, alpha=0.7)

sns.boxplot(data=Other_car, x='body_type', y='price', ax=axs[1],
            showfliers=False)
sns.stripplot(data=Other_car, x='body_type', y='price', ax=axs[1],
              linewidth=0.5, jitter=0.1, size=8, alpha=0.7)
```

Out[171]:
```
<AxesSubplot:xlabel='body_type', ylabel='price'>
```

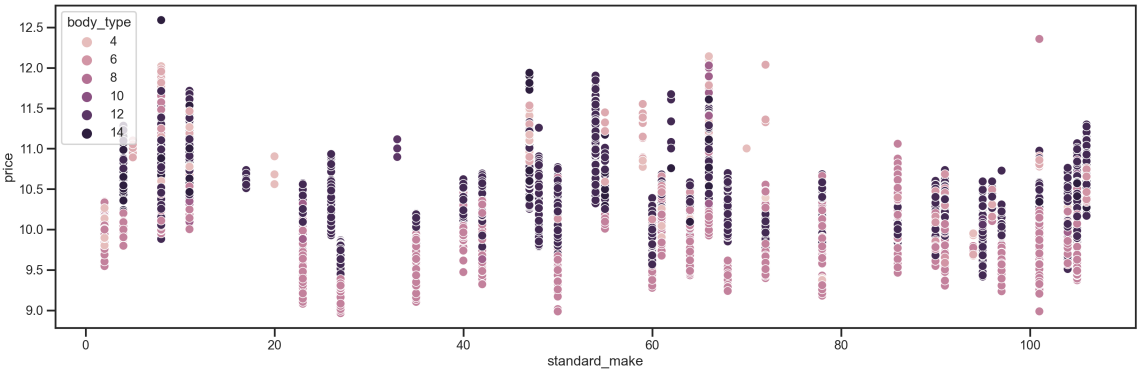


Above figures shows the relationship between body_type and price in four subsets of the dataset.

3.3. Categorical-Categorical

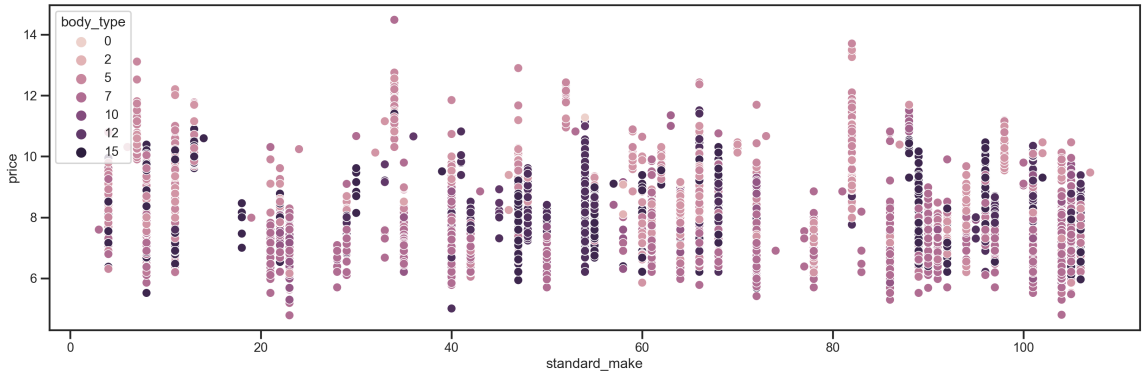
In [182]:
```
plt.figure(figsize=(20, 6))
sns.scatterplot(x='standard_make', y='price', hue='body_type',
                data=Brand_new_car)
```

Out[182]:
```
<AxesSubplot:xlabel='standard_make', ylabel='price'>
```



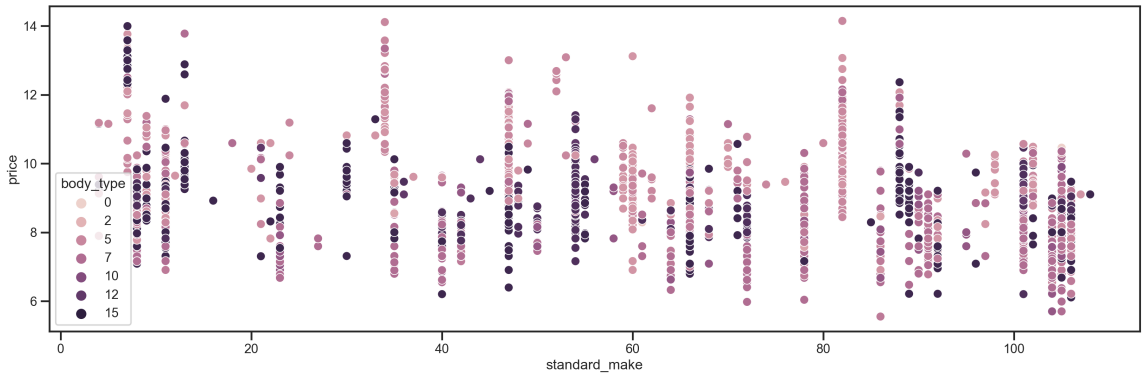
In [184]:
```
plt.figure(figsize=(20, 6))
sns.scatterplot(x='standard_make', y='price', hue='body_type',
```

Out[184]:
```
<AxesSubplot:xlabel='standard_make', ylabel='price'>
```



In [185]:
```
plt.figure(figsize=(20, 6))
sns.scatterplot(x='standard_make', y='price', hue='body_type',
```

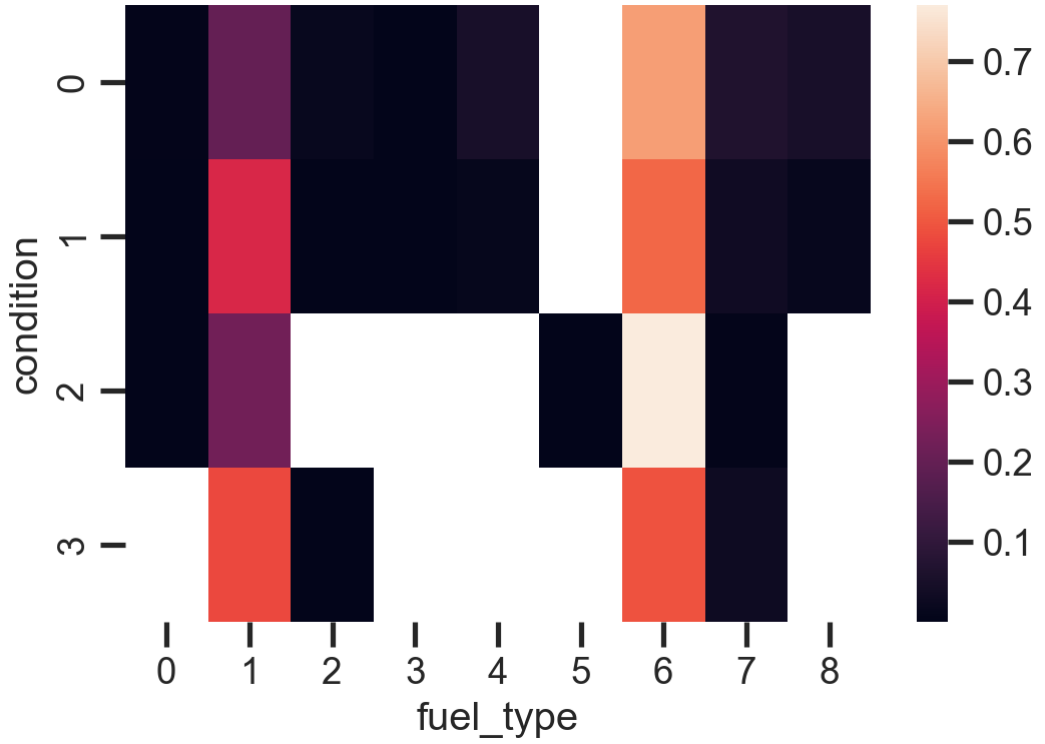
Out[185]:
```
<AxesSubplot:xlabel='standard_make', ylabel='price'>
```



Above figures shows the relationship between standard_make, body_type and price in four subsets of the dataset. It is clear that the price of same standard_make can vary according to their body_type.

In [175]:
```
sns.heatmap((car_adv.value_counts(['fuel_type', 'condition']) /
             car_adv.value_counts(['condition'])).unstack())
```

Out[175]:
```
<AxesSubplot:xlabel='fuel_type', ylabel='condition'>
```



Above heat map compairs the fuel_type in four subsets of the dataset.

Data Science Question 1

What are the best predictors of the price of a vehicle? In other words, of the individual and interactions of pairs of features in the dataset, what are those that seem to have to strongest association with the feature price, and what are the explanations and insights behind those findings?

A vehicle's pricing might vary depending on a number of features. I believe that the age, mileage, standard make, body type, and fuel type are the most important predictors.

Age:

A car's age may be a reliable indicator of how much it will cost because newer cars are usually more expensive than older ones. Because newer cars are frequently believed to be in better condition and to have fewer problems than older ones. Additionally, newer vehicles may have superior safety features and cutting-edge technology than older ones. However, certain classic and antique vehicles may also be highly valuable.

Mileage:

A vehicle's mileage can be a good indicator of its pricing. Prices for vehicles with lesser mileage are often greater than those with higher mileage. This is so because vehicles with lower mileage are typically thought to be in better condition and to have a longer lifespan

than those with higher mileage.

Standard make:

The brand of a car can also affect how much it costs. The cost of luxury automobiles, like those produced by Pagani,, Bugatti, Ferrari, Lamborghini, McLaren, Rolls-Royce etc, is typically higher than that of automobiles from other manufacturers. This is due to the perception that luxury cars are higher quality and have more cutting-edge safety features than cars from other manufacturers.

Body type:

A car's body style, such as a sedan, hatchback, or SUV, can have an impact on how much it costs. In general, sedans and hatchbacks are less expensive than SUVs, which are viewed as more practical and adaptable automobiles.

Fuel type:

A vehicle's pricing may also be influenced by the type of fuel it uses. The cost of gasoline-powered cars is typically less than the cost of electric or hybrid vehicles that use alternative fuels. This may be a result of the vehicle's higher price as a result of the technology and battery costs

Data Science Question 2

What are interesting groupings of the data, involving one or two features, that show significant differences (e.g., trends, averages) in price? What can we learn from them and how useful could these findings be for the business?

Age & mileage, fuel type & mileage, and standard make & body type are all interesting groupings of data that can show significant differences in the price of a car.

Age, Mileage and Price:

By dividing the data into groups according to age and mileage, we can see how the cost of an automobile changes as it ages and as its mileage rises. The price of the car will probably be lower the older it is and the higher its mileage. For a company looking to buy or sell used cars, knowing the going rate for vehicles with various age and mileage ranges could be helpful.

Fuel type, Mileage and Price:

By dividing the data into groups based on fuel type and mileage, we can observe how the cost of a car varies depending on the kind of fuel it runs on and how far it travels. Automobiles with lower mileage and more cost-effective fuel options could be more expensive than those with higher mileage and inefficient fuel options. An organisation wanting to acquire or sell cars could benefit from knowing the average pricing for vehicles with various fuel types and mileages.

Standard Make, Body Type and Price:

Standard make and body type: By dividing the data into these two categories, we can

examine how the cost of an automobile varies depending on its make and body style. Certain manufacturers and body styles of vehicles may cost more than others. Having knowledge of the typical prices for automobiles of various kinds and body styles could be helpful for a company wanting to acquire or sell cars.