Pandas Lab Exercise (Kaggle Automobile Dataset)

We shall now test your skills in using Pandas package. We will be using the automobiles Dataset from Kaggle.

Answer each question asked below wrt the automobiles dataset. Load pandas as pd and upload the Automobile.csv file as auto

In [12]:
```
import pandas as pd
```

Load the Automobile dataset into variable "auto"

In [13]:
```
auto=pd.read_csv("Automobile.csv")
type(auto)
```

Out[13]:
```
pandas.core.frame.DataFrame
```

Check the head of the DataFrame.

In [14]:
```
auto.head()
```

Out[14]:

	symboling	normalized_losses	make	fuel_type	aspiration	number_of_doors	body_s
0	3	168	alfa-romero	gas	std	two	conver
1	3	168	alfa-romero	gas	std	two	conver
2	1	168	alfa-romero	gas	std	two	hatchl
3	2	164	audi	gas	std	four	se
4	2	164	audi	gas	std	four	se

5 rows × 26 columns

How many rows and columns are there?

In [15]:
```
auto.shape
```

Out[15]:
```
(201, 26)
```

In [ ]:
```
```

What is the average Price of all cars in the dataset?

In [16]:
```
auto['price'].mean()
```

```
Out[16]: np.float64(13207.129353233831)
```

Which is the cheapest make and costliest make of car in the lot?

```
In [20]: auto['price'].max() ,auto['price'].idxmax()
```

```
Out[20]: (np.int64(45400), 71)
```

```
In [21]: auto['price'].min(),auto['price'].idxmin()
```

```
Out[21]: (np.int64(5118), 134)
```

How many cars have horsepower greater than 100?

```
In [22]: horse_power=auto[auto['horsepower']>100]
         horse_power
```

Out[22]:

	symboling	normalized_losses	make	fuel_type	aspiration	number_of_doors	bod
0	3	168	alfa-romero	gas	std	two	con
1	3	168	alfa-romero	gas	std	two	con
2	1	168	alfa-romero	gas	std	two	hat
3	2	164	audi	gas	std	four	
4	2	164	audi	gas	std	four	
...	...	...	...	...	...	...	
196	-1	95	volvo	gas	std	four	
197	-1	95	volvo	gas	turbo	four	
198	-1	95	volvo	gas	std	four	
199	-1	95	volvo	diesel	turbo	four	
200	-1	95	volvo	gas	turbo	four	

90 rows × 26 columns

How many hatchback cars are in the dataset ?

```
In [23]: hatchback_count=auto[auto['body_style']=='hatchback'].shape[0]
```

```
In [24]: hatchback_count
```

```
Out[24]: 68
```

What are the 3 most commonly found cars in the dataset?

In [25]:
```
most_common_cars=auto['make'].value_counts().head(3)
```

In [26]:
```
most_common_cars
```

Out[26]:
```
make
toyota    32
nissan    18
mazda     17
Name: count, dtype: int64
```

Someone purchased a car for 7099, what is the make of the car?

In [27]:
```
make_Car=auto[auto['price']==7099]['make']
make_Car
```

Out[27]:
```
87    nissan
Name: make, dtype: object
```

Which cars are priced greater than 40000?

In [28]:
```
greater_than_40k=auto[auto['price']>40000][['make','price']]
greater_than_40k
```

Out[28]:

	make	price
15	bmw	41315
70	mercedes-benz	40960
71	mercedes-benz	45400

Which are the cars that are both a sedan and priced less than 7000?

In [30]:
```
cond = auto[(auto['body_style'] == 'sedan') & (auto['price'] < 7000)][['body_styl
cond
```

Out[30]:

	body_style
19	sedan
24	sedan
42	sedan
50	sedan
82	sedan
86	sedan
88	sedan
89	sedan
118	sedan
152	sedan

Count the number of unique values in the `fuel_type` column.

In [31]:

```
unique_vals=auto['fuel_type'].unique()
unique_vals
```

Out[31]:

```
array(['gas', 'diesel'], dtype=object)
```

List all the cars that have a horsepower between 100 and 200, and display their `make` , `horsepower` , and `price` .

In [33]:

```
cond_2= auto[(auto['horsepower']>100) & (auto['horsepower']<=200) ]
cond_2[['make','horsepower','price']]
```

Out[33]:

	make	horsepower	price
0	alfa-romero	111	13495
1	alfa-romero	111	16500
2	alfa-romero	154	16500
3	audi	102	13950
4	audi	115	17450
...	...	...	...
196	volvo	114	16845
197	volvo	160	19045
198	volvo	134	21485
199	volvo	106	22470
200	volvo	114	22625

86 rows × 3 columns

Find the average `city_mpg` and `highway_mpg` for each `body_style` .

In [34]:
```
avg_mpg = auto.groupby('body_style')[['city_mpg','highway_mpg']].mean()
avg_mpg
```

Out[34]:

	city_mpg	highway_mpg
body_style		
convertible	20.500000	26.000000
hardtop	21.625000	27.250000
hatchback	26.602941	32.382353
sedan	25.053191	30.574468
wagon	24.040000	28.720000

What is the median `price` for each `make` ?

In [35]:
```
median_price = auto.groupby('make')['price'].mean()
median_price
```

Out[35]:

```
make
alfa-romero      15498.333333
audi             17859.166667
bmw              26118.750000
chevrolet         6007.000000
dodge             7875.444444
honda             8184.692308
isuzu             8916.500000
jaguar           34600.000000
mazda            10652.882353
mercedes-benz    33647.000000
mercury          16503.000000
mitsubishi        9239.769231
nissan           10415.666667
peugot           15489.090909
plymouth          7963.428571
porsche          31400.500000
renault           9595.000000
saab             15223.333333
subaru            8541.250000
toyota            9885.812500
volkswagen       10077.500000
volvo            18063.181818
Name: price, dtype: float64
```

List all cars that have a `wheel_base` greater than 100 and a `curb_weight` less than 2500.

In [36]:

```
cond_3 = auto[(auto['wheel_base'] >100 ) & (auto['curb_weight']<2500)]
cond_3
```

Out[36]:

	symboling	normalized_losses	make	fuel_type	aspiration	number_of_doors	body
9	2	192	bmw	gas	std	two	
10	0	192	bmw	gas	std	four	
169	-1	65	toyota	gas	std	four	
170	-1	65	toyota	diesel	turbo	four	
171	-1	65	toyota	gas	std	four	hatc
172	-1	65	toyota	gas	std	four	
173	-1	65	toyota	gas	std	four	hatc

7 rows × 26 columns

Create a new column `price_per_hp` that calculates the price of the car per horsepower.

In [37]:

```
auto['price_per_hp']= auto['price']/auto['horsepower']
auto.head()
```

Out[37]:

	symboling	normalized_losses	make	fuel_type	aspiration	number_of_doors	body_s
0	3	168	alfa-romero	gas	std	two	conver
1	3	168	alfa-romero	gas	std	two	conver
2	1	168	alfa-romero	gas	std	two	hatchb
3	2	164	audi	gas	std	four	se
4	2	164	audi	gas	std	four	se

5 rows × 27 columns

Count how many cars have a `number_of_doors` as `four`.

In [41]:

```
four_nofdoors = auto[auto['number_of_doors'] == 'four'].shape[0]
four_nofdoors
```

Out[41]:

```
114
```

Find the top 5 cars based on their `highway_mpg` and `price`.

In [42]:

```
top_5_cars=auto.sort_values(['highway_mpg','price'],ascending=[False,False]).head
top_5_cars
```

Out[42]:

	symboling	normalized_losses	make	fuel_type	aspiration	number_of_doors	bo
29	2	137	honda	gas	std	two	h
17	2	121	chevrolet	gas	std	two	h
87	1	128	nissan	diesel	std	two	
155	0	91	toyota	diesel	std	four	h
156	0	91	toyota	gas	std	four	

5 rows × 27 columns

How many cars have missing values in the `normalized_losses` column?

In [43]:

```
missing_vals = auto['normalized_losses'].isnull().sum()
missing_vals
```

Out[43]:

```
np.int64(0)
```

Create a new column `car_age` that calculates the age of the car based on the `year_of_manufacture` (assume the current year is 2025).

In [46]:

```
current_year = 2025
auto['car_age'] = current_year - auto['year_of_manufacture']
auto.head()
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
File C:\Python311\Lib\site-packages\pandas\core\indexes\base.py:3805, in Index.get
_loc(self, key)
   3804 try:
-> 3805     return self._engine.get_loc(casted_key)
   3806 except KeyError as err:

File index.pyx:167, in pandas._libs.index.IndexEngine.get_loc()

File index.pyx:196, in pandas._libs.index.IndexEngine.get_loc()

File pandas\\_libs\\hashtable_class_helper.pxi:7081, in pandas._libs.hashtable.PyO
bjectHashTable.get_item()

File pandas\\_libs\\hashtable_class_helper.pxi:7089, in pandas._libs.hashtable.PyO
bjectHashTable.get_item()

KeyError: 'year_of_manufacture'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
Cell In[46], line 2
      1 current_year = 2025
----> 2 auto['car_age'] = current_year - auto['year_of_manufacture']
      3 auto.head()

File C:\Python311\Lib\site-packages\pandas\core\frame.py:4102, in DataFrame.__geti
tem__(self, key)
   4100 if self.columns.nlevels > 1:
   4101     return self._getitem_multilevel(key)
-> 4102 indexer = self.columns.get_loc(key)
   4103 if is_integer(indexer):
   4104     indexer = [indexer]

File C:\Python311\Lib\site-packages\pandas\core\indexes\base.py:3812, in Index.get
_loc(self, key)
   3807     if isinstance(casted_key, slice) or (
   3808         isinstance(casted_key, abc.Iterable)
   3809         and any(isinstance(x, slice) for x in casted_key)
   3810     ):
   3811         raise InvalidIndexError(key)
-> 3812     raise KeyError(key) from err
   3813 except TypeError:
   3814     # If we have a listlike key, _check_indexing_error will raise
   3815     #  InvalidIndexError. Otherwise we fall through and re-raise
   3816     #  the TypeError.
   3817     self._check_indexing_error(key)

KeyError: 'year_of_manufacture'
```

The END