Vehicle Dataset

KATE expects your code to define variables with specific names that correspond to certain things we are interested in.

KATE will run your notebook from top to bottom and check the latest value of those variables, so make sure you don't overwrite them.

- Remember to uncomment the line assigning the variable to your answer and don't change the variable or function names.
- Use copies of the original or previous DataFrames to make sure you do not overwrite them by mistake.

You will find instructions below about how to define each variable.

Once you're happy with your code, upload your notebook to KATE to check your feedback.

In [1]:
```
import pandas as pd
```

First, we will load the dataset from `data/cars.csv` into a DataFrame.

In [2]:
```
df = pd.read_csv('data/cars.csv')
df.head()
```

Out[2]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	origin	na
0	18.0	8	307.0	130.0	3504	12.0	70	usa	chevro cheve mal
1	15.0	8	350.0	165.0	3693	11.5	70	usa	bu skyl 3
2	18.0	8	318.0	150.0	3436	11.0	70	usa	plymo satel
3	16.0	8	304.0	150.0	3433	12.0	70	usa	a rebel
4	17.0	8	302.0	140.0	3449	10.5	70	usa	f tor

Dataset stats

1. What's the mean of the values in the `weight` column?

Store the answer in a variable called `mean_weight`

In [3]:
```
# Add your code below
mean_weight = df['weight'].mean()
mean_weight
```

Out[3]:
```
2973.946835443038
```

2. What's the maximum value in the `horsepower` column?

Store the answer in a variable called `max_horsepower`

In [4]:
```
# Add your code below
max_horsepower = df['horsepower'].max()
max_horsepower
```

Out[4]:
```
230.0
```

3. How many cars have a `weight` of equal to or greater than 3500 ?

Store the answer in a variable called `heavy_cars`

In [5]:
```
# Add your code below
df
weight_mask=df['weight']>=3500
heavy = df[weight_mask]
heavy_cars=len(heavy)
heavy_cars
```

Out[5]:
```
109
```

4. Create a new DataFrame with an additional column called `ratio` , which equals `horsepower` divided by `weight`

Call the new DataFrame `df_ratio`

In [6]:
```
# We made a copy of df to start with, so you don't risk modifying the origin
df_ratio = df.copy()
# Add your code below
df_ratio['ratio']=df['horsepower']/df['weight']
df_ratio
```

Out[6]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	origin
0	18.0	8	307.0	130.0	3504	12.0	70	usa
1	15.0	8	350.0	165.0	3693	11.5	70	usa
2	18.0	8	318.0	150.0	3436	11.0	70	usa
3	16.0	8	304.0	150.0	3433	12.0	70	usa
4	17.0	8	302.0	140.0	3449	10.5	70	usa
...	...	...	...	...	...	...	...	...

Dataset sorting and filtering

5. Create a new DataFrame containing only cars with an `origin` of 'usa'

We'll start with a copy of the original DataFrame to avoid modifying the original. Call the new DataFrame `df_usa`

In [7]:
```
df_usa = df.copy()
# Add your code below
mask_usa=df_usa['origin']=='usa'
df_usa = df_usa[mask_usa]
df_usa
```

Out[7]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	origin	
0	18.0	8	307.0	130.0	3504	12.0	70	usa	che ch n
1	15.0	8	350.0	165.0	3693	11.5	70	usa	sl
2	18.0	8	318.0	150.0	3436	11.0	70	usa	plyr sa
3	16.0	8	304.0	150.0	3433	12.0	70	usa	reb
4	17.0	8	302.0	140.0	3449	10.5	70	usa	
...	...	...	...	...	...	...	...	...	
389	27.0	4	151.0	90.0	2950	17.3	82	usa	che ca
390	27.0	4	140.0	86.0	2790	15.6	82	usa	mu
392	32.0	4	135.0	84.0	2295	11.6	82	usa	c ram
393	28.0	4	120.0	79.0	2625	18.6	82	usa	r
394	31.0	4	119.0	82.0	2720	19.4	82	usa	che

246 rows × 9 columns

6. What's the mean `mpg` of cars of origin `usa` ?

Remember that we can use the `df_usa` DataFrame just created, which only contains these cars.

Store your answer in a variable called `mean_mpg_usa`

In [8]:
```
# Add your code below
mean_mpg_usa=df_usa['mpg'].mean()
mean_mpg_usa
```

Out[8]:
```
20.04308943089431
```

7. How many cars of origin `usa` have 8 `cylinders` ?

Store your answer in a variable called `eight_cyl_usa`

In [9]:
```
# Add your code below
mask_cylinder=df_usa['cylinders']==8
eight_cyl = df_usa[mask_cylinder]
eight_cyl_usa=len(eight_cyl)
eight_cyl_usa
```

Out[9]:
```
103
```

In [10]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 395 entries, 0 to 394
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   mpg           395 non-null    float64
 1   cylinders     395 non-null    int64  
 2   displacement  395 non-null    float64
 3   horsepower    390 non-null    float64
 4   weight        395 non-null    int64  
 5   acceleration  395 non-null    float64
 6   model_year    395 non-null    int64  
 7   origin        395 non-null    object 
 8   name          395 non-null    object 
dtypes: float64(4), int64(3), object(2)
memory usage: 27.9+ KB
```

We can see from `df.info()` that we have some missing values in the `horsepower` column.

8. create a new DataFrame (from the original `df`) which does not contain the rows with a missing value

Call the new DataFrame `df_horsepower`

In [11]:
```
df_horsepower = df.copy()
df_horsepower = df_horsepower.dropna(axis=0)
df_horsepower
```

Out[11]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	origin
0	18.0	8	307.0	130.0	3504	12.0	70	usa
1	15.0	8	350.0	165.0	3693	11.5	70	usa
2	18.0	8	318.0	150.0	3436	11.0	70	usa
3	16.0	8	304.0	150.0	3433	12.0	70	usa
4	17.0	8	302.0	140.0	3449	10.5	70	usa
...	...	...	...	...	...	...	...	...

9. What's the first (or only) mode value for `horsepower` in `df_horsepower` ?

Store your answer in a variable called `mode_hp`

Hint: i.e. the value found using the `.mode()` method on the given column; note that because there may be more than one mode, the method returns an array. We can access the first value using `[0]` , like we would with a list.

In [12]:
```
# Add your code below
mode_hp = df_horsepower['horsepower'].mode()[0]
mode_hp
```

Out[12]:
```
150.0
```

10. Create a DataFrame containing only cars with a horsepower greater than or equal to `mode_hp` in `df_horsepower`

Call the new DataFrame `df_high_hp`

In [13]:
```
df_high_hp = df_horsepower.copy()
# Add your code below
mask_high = df_high_hp['horsepower']>=mode_hp
df_high_hp = df_high_hp[mask_high]
df_high_hp
```

Out[13]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	origin	
1	15.0	8	350.0	165.0	3693	11.5	70	usa	sl
2	18.0	8	318.0	150.0	3436	11.0	70	usa	plyr sa
3	16.0	8	304.0	150.0	3433	12.0	70	usa	reb
5	15.0	8	429.0	198.0	4341	10.0	70	usa	ga
6	14.0	8	454.0	220.0	4354	9.0	70	usa	che ir
...	...	...	...	...	...	...	...	...	
228	15.5	8	350.0	170.0	4165	11.4	77	usa	che r la
229	15.5	8	400.0	190.0	4325	12.2	77	usa	ch co
261	17.7	6	231.0	165.0	3445	13.4	78	usa	((f
287	16.9	8	350.0	155.0	4360	14.9	79	usa	e w
290	18.5	8	360.0	150.0	3940	13.0	79	usa	ch le to cc

67 rows × 9 columns

11. What percentage of the cars in `df_high_hp` have 8 `cylinders` ?

Store your answer in a variable called `percentage_eight_cyl`

Your answer should be a float, and should be for example 56.0 rather than 0.56 for 56%.

In [14]:
```
# Add your code below
#df_high_hp
mask_eight=df_high_hp['cylinders']==8
eight_cylinder=df_high_hp[mask_eight]
#eight_cylinder
percentage_eight_cyl = (len(eight_cylinder)/len(df_high_hp))*100
percentage_eight_cyl
```

Out[14]:
```
98.50746268656717
```

Dataset manipulation

We can see from the output below that some car names have more than one entry in the DataFrame:

In [15]:
```
df['name'].value_counts()
```

Out[15]:
```
toyota corolla         5
amc matador            5
ford maverick          5
toyota corona          4
chevrolet chevette     4
                      ..
chevrolet monza 2+2    1
ford mustang ii        1
pontiac astro          1
amc pacer              1
chevy s-10             1
Name: name, Length: 306, dtype: int64
```

12. Add a column called `name_year` to a copy of `df`, with each entry containing a string in the following format:

```
name + ' - 19' + model_year
```

So for example, `'chevrolet chevelle malibu - 1970'`

Call the new DataFrame `df_name`

Hint: you may find the .astype() method useful

In [16]:
```
df_name = df.copy()
df_name
df_name['name_year']=df_name['name']+ ' - 19'+df_name['model_year'].astype(s
df_name
```

Out[16]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	origin	
0	18.0	8	307.0	130.0	3504	12.0	70	usa	che ch
1	15.0	8	350.0	165.0	3693	11.5	70	usa	s
2	18.0	8	318.0	150.0	3436	11.0	70	usa	ply s
3	16.0	8	304.0	150.0	3433	12.0	70	usa	re
4	17.0	8	302.0	140.0	3449	10.5	70	usa	
...	...	...	...	...	...	...	...	...	
390	27.0	4	140.0	86.0	2790	15.6	82	usa	mu
391	44.0	4	97.0	52.0	2130	24.6	82	europe	
392	32.0	4	135.0	84.0	2295	11.6	82	usa	rar
393	28.0	4	120.0	79.0	2625	18.6	82	usa	
394	31.0	4	119.0	82.0	2720	19.4	82	usa	ch

395 rows × 10 columns

Looking at value_counts() on the `name_year` column, we should now see that there are no duplicated entries:

In [17]:
```
df_name['name_year'].value_counts()
```

Out[17]:
```
chevrolet chevelle malibu - 1970    1
datsun 200-sx - 1978                1
plymouth sapporo - 1978             1
toyota celica gt liftback - 1978    1
dodge omni - 1978                   1
                                   ..
ford pinto - 1974                   1
datsun b210 - 1974                  1
chevrolet nova - 1974               1
amc hornet - 1974                   1
chevy s-10 - 1982                   1
Name: name_year, Length: 395, dtype: int64
```

13. On a copy of the `df_name` DataFrame, set the index of the DataFrame as the `name_year` column

Call you new DataFrame `df_car_index`

Hint: if using the set_index method, either use `inplace=True` or assign the result to a variable, otherwise the new index won't be stored.

In [18]:
```python
# Add your code below
df_car = df_name.copy()
df_car_index = df_car.set_index('name_year')
df_car_index
```

Out[18]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	ori
name_year								
chevrolet chevelle malibu - 1970	18.0	8	307.0	130.0	3504	12.0	70	u
buick skylark 320 - 1970	15.0	8	350.0	165.0	3693	11.5	70	u
plymouth satellite - 1970	18.0	8	318.0	150.0	3436	11.0	70	u
amc rebel sst - 1970	16.0	8	304.0	150.0	3433	12.0	70	u
ford torino - 1970	17.0	8	302.0	140.0	3449	10.5	70	u
...	...	...	...	...	...	...	...	
ford mustang gl - 1982	27.0	4	140.0	86.0	2790	15.6	82	u
vw pickup - 1982	44.0	4	97.0	52.0	2130	24.6	82	eurc
dodge rampage - 1982	32.0	4	135.0	84.0	2295	11.6	82	u
ford ranger - 1982	28.0	4	120.0	79.0	2625	18.6	82	u
chevy s-10 - 1982	31.0	4	119.0	82.0	2720	19.4	82	u

395 rows × 9 columns

14. Create a function which takes `name_year` as the only parameter, and returns the `acceleration` for any car in `df_car_index`

```
In [19]: # Add your code below
         def acceleration(name_year):
         #      pass


             acceleration = df_car_index.loc[name_year, 'acceleration']

             return acceleration
```

You can test your function using the following cell:

```
In [20]: acceleration('ford torino - 1970')
```

```
Out[20]: 10.5
```

```
In [ ]:
```