Vehicle Dataset

- 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.

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

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

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

Dataset stats

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

Store the answer in a variable called `mean_weight`

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

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

Store the answer in a variable called `max_horsepower`

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

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

Store the answer in a variable called `heavy_cars`

In [ ]:
```
df.shape
```

In [ ]:
```
# Add your code below
heavy_cars = df[df['weight'] >= 3500]['weight'].count()
heavy_cars
```

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

Call the new DataFrame `df_ratio`

In [ ]:
```
# We made a copy of df to start with, so you don't risk modifying the original df
df_ratio = df.copy()

# Add your code below
df_ratio['ratio'] = df_ratio['horsepower'] / df_ratio['weight']
df_ratio
```

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 [ ]:
```
df_usa = df.copy()
# Add your code below

df_usa = df_usa[df_usa['origin'] == 'usa']
df_usa
```

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

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

Store your answer in a variable called `eight_cyl_usa`

In [ ]:
```
df_usa[df_usa['cylinders'] == 8].shape[0]
```

In [ ]:
```
# Add your code below
eight_cyl_usa = df_usa[df_usa['cylinders'] == 8].shape[0]
eight_cyl_usa
```

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

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 [ ]:
```
df_horsepower = df.copy()

# Add your code below
df_horsepower = df_horsepower[~df_horsepower['horsepower'].isna()]
df_horsepower
```

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

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 [ ]:
```
df_high_hp = df_horsepower.copy()

# Add your code below
df_high_hp = df_horsepower[df_horsepower['horsepower'] >= mode_hp]
df_high_hp
```

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 [ ]:
```
# Add your code below
total_count = df_high_hp['cylinders'].count()
eight_count = df_high_hp['cylinders'].value_counts()[8]

percentage_eight_cyl = eight_count / total_count * 100
percentage_eight_cyl
```

Dataset manipulation

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

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

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 [ ]:
```
df_name = df.copy()

# Add your code below
df_name = df_name['name_year'] = df_name['name'] + ' - 19' + df_name['model_year'].ast
df_name
```

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

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

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 [ ]:
```
df_car_index = df_name.copy()

# Add your code below
df_car_index.set_index('name_year', inplace=True)
df_car_index
```

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

In [ ]:
```
# Add your code below
def acceleration(name_year):
    return df_car_index.loc[name_year, 'acceleration']
```

You can test your function using the following cell:

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