Fixing `air_pollution_score` Data Type

- 2008: convert string to float
- 2018: convert int to float

Load datasets `data_08_v3.csv` and `data_18_v3.csv`. You should've created these data files in the previous section: *Fixing Data Types Pt 1*.

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

# load datasets

df_08 = pd.read_csv('data_08_v3.csv')
df_18 = pd.read_csv('data_18_v3.csv')
df_08.head()
```

Out[1]:

	model	displ	cyl	trans	drive	fuel	veh_class	air_pollution_score	city_mpg	hwy_mpg
0	ACURA MDX	3.7	6	Auto-S5	4WD	Gasoline	SUV	7	15	20
1	ACURA RDX	2.3	4	Auto-S5	4WD	Gasoline	SUV	7	17	22
2	ACURA RL	3.5	6	Auto-S5	4WD	Gasoline	midsize car	7	16	24
3	ACURA TL	3.2	6	Auto-S5	2WD	Gasoline	midsize car	7	18	26
4	ACURA TL	3.5	6	Auto-S5	2WD	Gasoline	midsize car	7	17	26

In [2]:
```
df_18.head()
```

Out[2]:

	model	displ	cyl	trans	drive	fuel	veh_class	air_pollution_score	city_mpg	hwy_m
0	ACURA RDX	3.5	6	SemiAuto-6	2WD	Gasoline	small SUV	3	20	
1	ACURA RDX	3.5	6	SemiAuto-6	4WD	Gasoline	small SUV	3	19	
2	ACURA TLX	2.4	4	AMS-8	2WD	Gasoline	small car	3	23	
3	ACURA TLX	3.5	6	SemiAuto-9	2WD	Gasoline	small car	3	20	
4	ACURA TLX	3.5	6	SemiAuto-9	4WD	Gasoline	small car	3	21	

In [3]:
```
# try using pandas' to_numeric or astype function to convert the
# 2008 air_pollution_score column to float -- this won't work
df_08.air_pollution_score = df_08.air_pollution_score.astype(float)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-3-39951888cec4> in <module>()
      1 # try using pandas' to_numeric or astype function to convert the
      2 # 2008 air_pollution_score column to float -- this won't work
----> 3 df_08.air_pollution_score = df_08.air_pollution_score.astype(float)

/opt/conda/lib/python3.6/site-packages/pandas/util/_decorators.py in wrapper(*args, **kwargs)
    176                 else:
    177                     kwargs[new_arg_name] = new_arg_value
--> 178             return func(*args, **kwargs)
    179         return wrapper
    180     return _deprecate_kwarg

/opt/conda/lib/python3.6/site-packages/pandas/core/generic.py in astype(self, dtype, copy, errors, **kwargs)
   4999             # else, only a single dtype is given
   5000             new_data = self._data.astype(dtype=dtype, copy=copy, errors=errors,
-> 5001                                          **kwargs)
   5002             return self._constructor(new_data).__finalize__(self)
   5003

/opt/conda/lib/python3.6/site-packages/pandas/core/internals.py in astype(self, dtype, **kwargs)
   3712
   3713     def astype(self, dtype, **kwargs):
-> 3714         return self.apply('astype', dtype=dtype, **kwargs)
   3715
   3716     def convert(self, **kwargs):

/opt/conda/lib/python3.6/site-packages/pandas/core/internals.py in apply(self, f, axes, filter, do_integrity_check, consolidate, **kwargs)
   3579
   3580             kwargs['mgr'] = self
-> 3581             applied = getattr(b, f)(**kwargs)
   3582             result_blocks = _extend_blocks(applied, result_blocks)
   3583

/opt/conda/lib/python3.6/site-packages/pandas/core/internals.py in astype(self, dtype, copy, errors, values, **kwargs)
    573     def astype(self, dtype, copy=False, errors='raise', values=None, **kwargs):
    574         return self._astype(dtype, copy=copy, errors=errors, values=values,
--> 575                             **kwargs)
    576
    577     def _astype(self, dtype, copy=False, errors='raise', values=None,
```

```
/opt/conda/lib/python3.6/site-packages/pandas/core/internals.py in _ast
ype(self, dtype, copy, errors, values, klass, mgr, **kwargs)
    662
    663                 # _astype_nansafe works fine with 1-d only
--> 664                 values = astype_nansafe(values.ravel(), dtype,
 copy=True)
    665                 values = values.reshape(self.shape)
    666

/opt/conda/lib/python3.6/site-packages/pandas/core/dtypes/cast.py in as
type_nansafe(arr, dtype, copy)
    728
    729     if copy:
--> 730         return arr.astype(dtype, copy=True)
    731     return arr.view(dtype)
    732

ValueError: could not convert string to float: '6/4'
```

Figuring out the issue

Looks like this isn't going to be as simple as converting the datatype. According to the error above, the air pollution score value in one of the rows is "6/4" - let's check it out.

In [4]:
```
df_08[df_08.air_pollution_score == '6/4']
```

Out[4]:

	model	displ	cyl	trans	drive	fuel	veh_class	air_pollution_score	city_mpg	h
582	MERCEDES-BENZ C300	3.0	6	Auto-L7	2WD	ethanol/gas	small car	6/4	13/18	

It's not just the air pollution score!

The mpg columns and greenhouse gas scores also seem to have the same problem - maybe that's why these were all saved as strings! According to this link (http://www.fueleconomy.gov/feg/findacarhelp.shtml#airPollutionScore), which I found from the PDF documentation:

> "If a vehicle can operate on more than one type of fuel, an estimate is provided for each fuel type."

Ohh... so all vehicles with more than one fuel type, or hybrids, like the one above (it uses ethanol AND gas) will have a string that holds two values - one for each. This is a little tricky, so I'm going to show you how to do it with the 2008 dataset, and then you'll try it with the 2018 dataset.

In [5]:
```
# First, let's get all the hybrids in 2008
hb_08 = df_08[df_08['fuel'].str.contains('/')]
hb_08
```

Out[5]:

	model	displ	cyl	trans	drive	fuel	veh_class	air_pollution_score	city_mpg	h
582	MERCEDES-BENZ C300	3.0	6	Auto-L7	2WD	ethanol/gas	small car	6/4	13/18	

Looks like this dataset only has one! The 2018 has MANY more - but don't worry - the steps I'm taking here will work for that as well!

In [6]:
```
# hybrids in 2018
hb_18 = df_18[df_18['fuel'].str.contains('/')]
hb_18
```

Out[6]:

	model	displ	cyl	trans	drive	fuel	veh_class	air_pollution_score
52	BMW 330e	2.0	4	SemiAuto-8	2WD	Gasoline/Electricity	small car	3
78	BMW 530e	2.0	4	SemiAuto-8	2WD	Gasoline/Electricity	small car	7
79	BMW 530e	2.0	4	SemiAuto-8	4WD	Gasoline/Electricity	small car	7
92	BMW 740e	2.0	4	SemiAuto-8	4WD	Gasoline/Electricity	large car	3
189	CHEVROLET Impala	3.6	6	SemiAuto-6	2WD	Ethanol/Gas	large car	5
195	CHEVROLET Silverado 15	4.3	6	Auto-6	2WD	Ethanol/Gas	pickup	5
196	CHEVROLET Silverado 15	4.3	6	Auto-6	4WD	Ethanol/Gas	pickup	5
197	CHEVROLET Silverado 15	5.3	8	Auto-6	2WD	Ethanol/Gas	pickup	3
212	CHEVROLET Suburban 1500	5.3	8	Auto-6	2WD	Ethanol/Gas	standard SUV	3
214	CHEVROLET Suburban 1500	5.3	8	Auto-6	4WD	Ethanol/Gas	standard SUV	3
216	CHEVROLET Tahoe 1500	5.3	8	Auto-6	2WD	Ethanol/Gas	standard SUV	3
218	CHEVROLET Tahoe 1500	5.3	8	Auto-6	4WD	Ethanol/Gas	standard SUV	3
225	CHEVROLET Volt	1.5	4	CVT	2WD	Gasoline/Electricity	small car	3
226	CHEVROLET Volt	1.5	4	CVT	2WD	Gasoline/Electricity	small car	7
227	CHRYSLER 300	3.6	6	Auto-8	2WD	Ethanol/Gas	large car	3
229	CHRYSLER 300	3.6	6	Auto-8	4WD	Ethanol/Gas	large car	3
244	DODGE Charger	3.6	6	Auto-8	2WD	Ethanol/Gas	large car	3
246	DODGE Charger	3.6	6	Auto-8	4WD	Ethanol/Gas	large car	3
300	FORD Fusion Energi Plug-in Hybrid	2.0	4	CVT	2WD	Gasoline/Electricity	midsize car	7
326	GMC Sierra 15	4.3	6	Auto-6	2WD	Ethanol/Gas	pickup	5
327	GMC Sierra 15	4.3	6	Auto-6	4WD	Ethanol/Gas	pickup	5

	model	displ	cyl	trans	drive	fuel	veh_class	air_pollution_score
328	GMC Sierra 15	5.3	8	Auto-6	2WD	Ethanol/Gas	pickup	3
345	GMC Yukon 1500	5.3	8	Auto-6	2WD	Ethanol/Gas	standard SUV	3
347	GMC Yukon 1500	5.3	8	Auto-6	4WD	Ethanol/Gas	standard SUV	3
351	GMC Yukon 1500 XL	5.3	8	Auto-6	2WD	Ethanol/Gas	standard SUV	3
354	GMC Yukon XL 1500	5.3	8	Auto-6	4WD	Ethanol/Gas	standard SUV	3
442	JEEP Cherokee	2.4	4	Auto-9	2WD	Ethanol/Gas	small SUV	3
444	JEEP Cherokee	2.4	4	Auto-9	4WD	Ethanol/Gas	small SUV	3
462	KARMA Revero	2.0	4	Auto-1	2WD	Gasoline/Electricity	small car	1
571	MERCEDES-BENZ CLA250 4Matic	2.0	4	AutoMan-7	4WD	Ethanol/Gas	small car	5
578	MERCEDES-BENZ GLA250 4Matic	2.0	4	AutoMan-7	4WD	Ethanol/Gas	small SUV	5
584	MERCEDES-BENZ GLE350 4Matic	3.5	6	Auto-7	4WD	Ethanol/Gas	standard SUV	3
616	MINI Cooper SE Countryman All4	1.5	3	SemiAuto-6	4WD	Gasoline/Electricity	midsize car	3
742	TOYOTA Sequoia FFV	5.7	8	SemiAuto-6	4WD	Ethanol/Gas	standard SUV	5
747	TOYOTA Tundra FFV	5.7	8	SemiAuto-6	4WD	Ethanol/Gas	pickup	5
777	VOLVO S90	2.0	4	SemiAuto-8	4WD	Gasoline/Electricity	midsize car	7
789	VOLVO XC 60	2.0	4	SemiAuto-8	4WD	Gasoline/Electricity	small SUV	7
793	VOLVO XC 90	2.0	4	SemiAuto-8	4WD	Gasoline/Electricity	standard SUV	7

We're going to take each hybrid row and split them into two new rows - one with values for the first fuel type (values before the "/"), and the other with values for the second fuel type (values after the "/"). Let's separate them with two dataframes!

In [7]:
```
# create two copies of the 2008 hybrids dataframe
df1 = hb_08.copy()  # data on first fuel type of each hybrid vehicle
df2 = hb_08.copy()  # data on second fuel type of each hybrid vehicle

# Each one should look like this
df1
```

Out[7]:

	model	displ	cyl	trans	drive	fuel	veh_class	air_pollution_score	city_mpg	h
582	MERCEDES-BENZ C300	3.0	6	Auto-L7	2WD	ethanol/gas	small car	6/4	13/18	

For this next part, we're going use pandas' apply function. See the docs here (https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.apply.html).

In [8]:
```
# columns to split by "/"
split_columns = ['fuel', 'air_pollution_score', 'city_mpg', 'hwy_mpg',
'cmb_mpg', 'greenhouse_gas_score']

# apply split function to each column of each dataframe copy
for c in split_columns:
    df1[c] = df1[c].apply(lambda x: x.split("/")[0])
    df2[c] = df2[c].apply(lambda x: x.split("/")[1])
```

In [9]:
```
# this dataframe holds info for the FIRST fuel type of the hybrid
# aka the values before the "/"s
df1
```

Out[9]:

	model	displ	cyl	trans	drive	fuel	veh_class	air_pollution_score	city_mpg	hwy_
582	MERCEDES-BENZ C300	3.0	6	Auto-L7	2WD	ethanol	small car	6	13	

In [10]:
```
# this dataframe holds info for the SECOND fuel type of the hybrid
# aka the values after the "/"s
df2
```

Out[10]:

	model	displ	cyl	trans	drive	fuel	veh_class	air_pollution_score	city_mpg	hwy_mp
582	MERCEDES-BENZ C300	3.0	6	Auto-L7	2WD	gas	small car	4	18	2

In [11]:
```
# combine dataframes to add to the original dataframe
new_rows = df1.append(df2)

# now we have separate rows for each fuel type of each vehicle!
new_rows
```

Out[11]:

	model	displ	cyl	trans	drive	fuel	veh_class	air_pollution_score	city_mpg	hwy_
582	MERCEDES-BENZ C300	3.0	6	Auto-L7	2WD	ethanol	small car	6	13	
582	MERCEDES-BENZ C300	3.0	6	Auto-L7	2WD	gas	small car	4	18	

In [12]:
```
# drop the original hybrid rows
df_08.drop(hb_08.index, inplace=True)

# add in our newly separated rows
df_08 = df_08.append(new_rows, ignore_index=True)
```

In [13]:
```
# check that all the original hybrid rows with "/"s are gone
df_08[df_08['fuel'].str.contains('/')]
```

Out[13]:

	model	displ	cyl	trans	drive	fuel	veh_class	air_pollution_score	city_mpg	hwy_mpg	cmb_

In [14]:
```
df_08.shape
```

Out[14]:
```
(987, 13)
```

Repeat this process for the 2018 dataset

In [15]:
```
# create two copies of the 2018 hybrids dataframe, hb_18
df1 = hb_18.copy()
df2 = hb_18.copy()
```

Split values for `fuel`, `city_mpg`, `hwy_mpg`, `cmb_mpg`

You don't need to split for `air_pollution_score` or `greenhouse_gas_score` here because these columns are already ints in the 2018 dataset.

In [16]:
```
# list of columns to split
split_columns = ['fuel','city_mpg','hwy_mpg','cmb_mpg']

# apply split function to each column of each dataframe copy
for c in split_columns:
    df1[c] = df1[c].apply(lambda x: x.split("/")[0])
    df2[c] = df2[c].apply(lambda x: x.split("/")[1])
```

In [17]:
```
# append the two dataframes
new_rows = df1.append(df2)

# drop each hybrid row from the original 2018 dataframe
# do this by using pandas' drop function with hb_18's index
df_18.drop(hb_18.index, inplace=True)

# append new_rows to df_18
df_18 = df_18.append(new_rows, ignore_index=True)
```

In [18]:
```
# check that they're gone
df_18[df_18['fuel'].str.contains('/')]
```

Out[18]:

	model	displ	cyl	trans	drive	fuel	veh_class	air_pollution_score	city_mpg	hwy_mpg	cmb_

In [19]:
```
df_18.shape
```

Out[19]: (832, 13)

Now we can comfortably continue the changes needed for `air_pollution_score`! Here they are again:

- 2008: convert string to float
- 2018: convert int to float

In [20]:
```
# convert string to float for 2008 air pollution column
df_08.air_pollution_score = df_08.air_pollution_score.astype(float)
```

In [21]:
```
# convert int to float for 2018 air pollution column
df_18.air_pollution_score = df_18.air_pollution_score.astype(float)
```

In [22]:
```
df_08.to_csv('data_08_v4.csv', index=False)
df_18.to_csv('data_18_v4.csv', index=False)
```

In [ ]:
```

```