Pivot Tables:

A pivot table is a way of summarizing data in a dataframe for a particular purpose. It makes heavy use of the aggregation function. A pivot table is itself a dataframe, where the rows represent one variable that you're interested in, the columns another, and the cell's some aggregate value.

A pivot table also tends to includes marginal values as well, which are the sums for each column and row.

This allows you to be able to see the relationship between two variables at just a glance.

Let's take a look at an example.

Here we'll load a new data set, cars.csv.

This data set comes from the Open Data Initiative of the Canadian government. And has information about the efficiency of different electric cars which are available for purchase.

When we look at the head of the dataframe, we'll see that there are model years, vendors, sizes of cars, and statistics, like how big the battery is in kilowatt hours.

```
#http://open.canada.ca/data/en/dataset/98f1a129-f628-4ce4-b24d-6f16bf24dd64
import numpy as np
df = pd.read_csv('cars.csv')
```

```
df.head()
```

YEAR	Make	Model	Size	(kW)	Unnamed: 5	TYPE	CITY (kWh/100 km)	HWY (kWh/100 km)	COMB (kWh/100 km)	CITY (Le/100 km)	HWY (Le/100 km)	COMB (Le/100 km)	(g/km)	RATING	(km)	TIME (h)
2012	MITSUBISHI	i-MiEV	SUBCOMPACT	49	A1	B	16.9	21.4	18.7	1.9	2.4	2.1	0	NaN	100	7
2012	NISSAN	LEAF	MID-SIZE	80	A1	B	19.3	23.0	21.1	2.2	2.6	2.4	0	NaN	117	7
2013	FORD	FOCUS ELECTRIC	COMPACT	107	A1	B	19.0	21.1	20.0	2.1	2.4	2.2	0	NaN	122	4
2013	MITSUBISHI	i-MiEV	SUBCOMPACT	49	A1	B	16.9	21.4	18.7	1.9	2.4	2.1	0	NaN	100	7
2013	NISSAN	LEAF	MID-SIZE	80	A1	B	19.3	23.0	21.1	2.2	2.6	2.4	0	NaN	117	7

A pivot table allows us to pivot out one of these columns into a new column headers and compare it against another column as row indices.

For instance, let's say we wanted to compare the makes of electric vehicles versus the years and that we wanted to do this comparison in terms of battery capacity.

To do this, we tell pandas we want the values to be kilowatts, the index to be the year and the columns to be the make. Then we specify that the aggregation function, and here we'll use the NumPy mean.

Here's the results.

```
df.pivot_table(values='(kW)', index='YEAR', columns='Make', aggfunc=np.mean)
```

Make	BMW	CHEVROLET	FORD	KIA	MITSUBISHI	NISSAN	SMART	TESLA
YEAR								
2012	NaN	NaN	NaN	NaN	49.0	80.0	NaN	NaN
2013	NaN	NaN	107.0	NaN	49.0	80.0	35.0	280.000000
2014	NaN	104.0	107.0	NaN	49.0	80.0	35.0	268.333333
2015	125.0	104.0	107.0	81.0	49.0	80.0	35.0	320.666667
2016	125.0	104.0	107.0	81.0	49.0	80.0	35.0	409.700000

We see there are NaN values for vendors who didn't have an entry in a given year like Ford in 2012.

And we see that most vendors don't have a change in battery capacity over the years, except for Tesla, as they've introduced several new models.

Now, pivot tables aren't limited to one function that you might want to apply.

You can pass *aggfunc*, a list of the different functions to apply, and pandas will provide you with the result using hierarchical column names.

Here, I'll also pass margins equals true.

And that you can see for each of the functions there's now an all category, which shows the overall mean and the minimum values for a given year and a given vendor.

```
df.pivot_table(values='(kW)', index='YEAR', columns='Make', aggfunc=[np.mean,np.min], margins=True)
```

	mean									amin					
Make	BMW	CHEVROLET	FORD	KIA	MITSUBISHI	NISSAN	SMART	TESLA	All	BMW	CHEVROLET	FORD	KIA	MITSUBISHI	NISSAN
YEAR															
2012	NaN	NaN	NaN	NaN	49.0	80.0	NaN	NaN	64.500000	NaN	NaN	NaN	NaN	49.0	80.0
2013	NaN	NaN	107.0	NaN	49.0	80.0	35.0	280.000000	158.444444	NaN	NaN	107.0	NaN	49.0	80.0
2014	NaN	104.0	107.0	NaN	49.0	80.0	35.0	268.333333	135.000000	NaN	104.0	107.0	NaN	49.0	80.0
2015	125.0	104.0	107.0	81.0	49.0	80.0	35.0	320.666667	181.428571	125.0	104.0	107.0	81.0	49.0	80.0
2016	125.0	104.0	107.0	81.0	49.0	80.0	35.0	409.700000	252.263158	125.0	104.0	107.0	81.0	49.0	80.0
All	125.0	104.0	107.0	81.0	49.0	80.0	35.0	345.478261	190.622642	125.0	104.0	107.0	81.0	49.0	80.0

SMART	TESLA	All
NaN	NaN	49
35.0	270.0	35
35.0	225.0	35
35.0	280.0	35
35.0	283.0	35
35.0	225.0	35

So that's pivot tables.

This has been a pretty short description, but they're incredibly useful when dealing with numeric data. And of course, you can pass any function you want to the aggregate function, including those that you define yourself.