Group by walkthrough

By now, you'd better know that you need to import Agate.

```
import agate
```

And you need data.

```
mountainlions = agate.Table.from_csv('../../Data/mountainlions
.csv')
print(mountainlions)
```

```
|---------------+---------------|
|  column_names | column_types  |
|---------------+---------------|
|  ID           | Number        |
|  Cofirm Type  | Text          |
|  COUNTY       | Text          |
|  Date         | Date          |
|---------------+---------------|
```

First, let's group them together using `group_by`. We'll create a new table, called `by_county` and populate it with the results of our `group_by` statement. The new table is a subset of our first table we created on import, so we'll have to be sure to use that.

```
by_county = mountainlions.group_by('COUNTY')
```

```
print(len(by_county))
```

```
42
```

So that means 42 counties had mountain lion sightings. Out of 93 Nebraska counties. That's a chunk. However, it helps to know what by using group_by, we haven't actually created a new table. We've created a TableSet. Which is why you can't print it and see it.

```
by_county.print_table()
```

```
---------------------------------------------------------------
-------------

AttributeError                            Traceback (most rece
nt call last)

<i-input-9-27a489864901> in <module>()
----> 1 by_county.print_table()


/Users/mattwaite/anaconda/envs/homework/lib/3.5/site-packages/
agate/tableset.py in __getattr__(self, name)
    115         # Proxy table methods
    116         if name in Table.__dict__:
--> 117             if Table.__dict__[name].allow_tableset_pro
xy:
    118                 return TableMethodProxy(self, name)
    119


AttributeError: 'function' object has no attribute 'allow_tabl
eset_proxy'
```

To make something out of a `group_by` TableSet, we have to run some aggregates on it. That looks like this:

```
county_totals = by_county.aggregate([
    ('count', agate.Length())
])
```

```
county_totals.print_table()
```

```
|---------------+--------|
|  COUNTY       | count  |
|---------------+--------|
|  Dawes        |   111  |
|  Sioux        |    52  |
|  Scotts Bluff |    26  |
|  Box Butte    |     4  |
|  Howard       |     3  |
|  Brown        |    15  |
|  Douglas      |     2  |
|  Cherry       |    30  |
|  Thomas       |     5  |
|  Keya Paha    |    20  |
|  Dakota       |     3  |
|  Sarpy        |     1  |
|  Custer       |     8  |
|  Sheridan     |    35  |
|  Banner       |     6  |
|  Knox         |     8  |
|  Nance        |     1  |
|  Platte       |     1  |
|  Dawson       |     5  |
|  Rock         |    11  |
|  Hooker       |     1  |
|  Lincoln      |    10  |
|  Polk         |     1  |
|  Valley       |     1  |
|  Sherman      |     1  |
|  Blaine       |     3  |
|  Saunders     |     2  |
|  Buffalo      |     3  |
|  sheridan     |     2  |
|  Thurston     |     1  |
|  Dixon        |     3  |
|  Holt         |     2  |
|  Kimball      |     1  |
|  Morrill      |     2  |
```

```
|  Cedar         |      1  |
|  Keith         |      1  |
|  Merrick       |      1  |
|  Hall          |      1  |
|  Wheeler       |      1  |
|  Richardson    |      2  |
|  Nemaha        |      5  |
|  Frontier      |      1  |
|----------------+---------|
```

That's more like it. But it's not in order, so it is sort of bothersome. We can fix that just like we did in the first walkthrough -- with ordering.

```
sorted_counties = county_totals.order_by('count', reverse=True
)
sorted_counties.print_table()
```

```
|----------------+---------|
|  COUNTY        | count   |
|----------------+---------|
|  Dawes         |   111   |
|  Sioux         |    52   |
|  Sheridan      |    35   |
|  Cherry        |    30   |
|  Scotts Bluff  |    26   |
|  Keya Paha     |    20   |
|  Brown         |    15   |
|  Rock          |    11   |
|  Lincoln       |    10   |
|  Custer        |     8   |
|  Knox          |     8   |
|  Banner        |     6   |
|  Thomas        |     5   |
|  Dawson        |     5   |
|  Nemaha        |     5   |
|  Box Butte     |     4   |
|  Howard        |     3   |
|  Dakota        |     3   |
|  Blaine        |     3   |
```

```
|  Buffalo      |     3  |
|  Dixon        |     3  |
|  Douglas      |     2  |
|  Saunders     |     2  |
|  sheridan     |     2  |
|  Holt         |     2  |
|  Morrill      |     2  |
|  Richardson   |     2  |
|  Sarpy        |     1  |
|  Nance        |     1  |
|  Platte       |     1  |
|  Hooker       |     1  |
|  Polk         |     1  |
|  Valley       |     1  |
|  Sherman      |     1  |
|  Thurston     |     1  |
|  Kimball      |     1  |
|  Cedar        |     1  |
|  Keith        |     1  |
|  Merrick      |     1  |
|  Hall         |     1  |
|  Wheeler      |     1  |
|  Frontier     |     1  |
|---------------+--------|
```

A note on `print_table` : You can limit the number of rows you print by addng max_rows=X to the print_table in the parenthesis, like this:

```
sorted_counties.print_table(max_rows=25)
```

```
|---------------+--------|
|  COUNTY       | count  |
|---------------+--------|
|  Dawes        |   111  |
|  Sioux        |    52  |
|  Sheridan     |    35  |
|  Cherry       |    30  |
|  Scotts Bluff |    26  |
|  Keya Paha    |    20  |
|  Brown        |    15  |
|  Rock         |    11  |
|  Lincoln      |    10  |
|  Custer       |     8  |
|  Knox         |     8  |
|  Banner       |     6  |
|  Thomas       |     5  |
|  Dawson       |     5  |
|  Nemaha       |     5  |
|  Box Butte    |     4  |
|  Howard       |     3  |
|  Dakota       |     3  |
|  Blaine       |     3  |
|  Buffalo      |     3  |
|  Dixon        |     3  |
|  Douglas      |     2  |
|  Saunders     |     2  |
|  sheridan     |     2  |
|  Holt         |     2  |
|  ...          |   ...  |
|---------------+--------|
```

Assignment

1. Copy your homework notebook from last time, the one with UNL salaries. Rename it to something else. I called my SecondAgateHomework.ipynb
2. Using what you've done already, let's extend it. We've calculated the median and mean salary for all UNL employees, but that doesn't tell the whole story. The mean includes football and basketball coaches. The medians don't show

the differences between jobs at the university. So using what you've learned in this walkthrough, group the salaries by job title.

3. Aggregate a count, a median and an average for each job title. Hint: You can do this all in one aggregate table. See here. One gotcha on that multiple aggregates in a single table thing: Watch out for commas. You need one at the end of every line EXCEPT the last aggregate.
4. Sort the table by the count, putting the most common job title at the top.
5. Print the table out. Limit it to the 50 most common jobs.
6. Make sure you describe each step taken in Markdown between the commands.