Lecture 12

Table Examples and Advanced Where
Announcements

- Project 1 out, part 1 due 3/19, part 2 due 3/26.
  - Get started early! Don’t put it off!
  - You can work in pairs.
  - Vocareum is not Google Docs. You cannot both work on the assignment at the same time and have both people’s work be saved.

- This week:
  - No class Wednesday (Wellness day)
  - No labs Wednesday or Thursday
  - No office hours Tuesday or Wednesday
Prelim 1

Prelim 1 on Tuesday, March 16, 8:30PM-10PM
● Here in this room (Call Auditorium) for all Ithaca resident students (whether in-person or online); assigned seating
● Same time online for all Ithaca non-resident students
● Coverage: From Lecture 1 to Lecture 11 (last Friday)
● Format:
  ○ Short answer (e.g. write a line of Python that does this…), multiple choice
  ○ Closed ‘book’, but you may bring one page (8.5” x 11”)
    double-sided set of notes that you write yourself
  ○ You will be provided with a sheet of standard Python function definitions
Prelim 1 resources

- Study guide (with the list of table functions to be given to you in the exam) will be posted today on Canvas
- Practice exam to be posted today
- Review session/`ask me anything (about 1380)' session late this week TBA (Saturday?)
Combining Table Methods
Important Table Methods

```
t.select(column, ...) or t.drop(column, ...)
t.take([row, ...]) or t.exclude([row, ...])
t.sort(column, descending=False, distinct=False)
t.where(column, are.condition(...))
t.apply(function, column, ...)
t.group(column) or t.group(column, function)
t.groups([column, ...]) or t.groups([column, ...], function)
t.pivot(cols, rows) or t.pivot(cols, rows, vals, function)
t.join(column, other_table, other_table_column)
```
The **apply** method creates an array by calling a function on every element in input column(s)

- First argument: Function to apply
- Other arguments: The input column(s)

```
table_name.apply(function_name, 'column_label')
```

(Demo)
The **group** method aggregates all rows with the same value for a column into a single row in the result.

- **First argument:** Which column to group by
- **Second argument:** (Optional) How to combine values
  - `len` — number of grouped values (default)
  - `sum` — total of all grouped values
  - `list` — list of all grouped values

(Demo)
Grouping By Two Columns

The `groups` method can also aggregate all rows that share the combination of values in multiple columns

- First argument: A list of which columns to group by
- Second argument: (Optional) How to combine values

(Demo)
Pivot

- Cross-classifies according to two categorical variables
- Produces a grid of counts or aggregated values
- Two required arguments:
  - First: variable that forms column labels of grid
  - Second: variable that forms row labels of grid
- Two optional arguments (include both or neither)
  - `values` = 'column_label_to_aggregate'
  - `collect` = function_with_which_to_aggregate

(Demo)
## Joining Two Tables

Keep all rows in the table that have a match ...

```python
drinks.join('Cafe', discounts, 'Location')
```

... for the value in this column ...
... somewhere in this other table's ...
... column that contains matching values.

<table>
<thead>
<tr>
<th>drinks</th>
<th></th>
<th>discounts</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Drink</td>
<td>Cafe</td>
<td>Price</td>
<td>Coupon</td>
</tr>
<tr>
<td>Milk tea</td>
<td>Panda Tea</td>
<td>4</td>
<td>25%</td>
</tr>
<tr>
<td>Espresso</td>
<td>Gimme</td>
<td>2</td>
<td>50%</td>
</tr>
<tr>
<td>Latte</td>
<td>Gimme</td>
<td>3</td>
<td>5%</td>
</tr>
<tr>
<td>Espresso</td>
<td>Cafe Gola</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

The joined column is sorted automatically

(Demo)
### Discussion Question

Generate a table with one row per cafe that has the name and discounted price of its cheapest discounted drink.

<table>
<thead>
<tr>
<th>Drinks</th>
<th>Cafe</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Milk tea</td>
<td>Panda Tea</td>
<td>4</td>
</tr>
<tr>
<td>Espresso</td>
<td>Gimme</td>
<td>2</td>
</tr>
<tr>
<td>Coffee</td>
<td>Gimme</td>
<td>3</td>
</tr>
<tr>
<td>Espresso</td>
<td>Cafe Gola</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Discounts</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coupon</td>
<td>Location</td>
</tr>
<tr>
<td>25%</td>
<td>Panda Tea</td>
</tr>
<tr>
<td>50%</td>
<td>Gimme</td>
</tr>
<tr>
<td>5%</td>
<td>Gimme</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cheapest</th>
<th>Cafe</th>
<th>Drink</th>
<th>Discounted Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cafe</td>
<td>Drink</td>
<td>Discounted Price</td>
<td></td>
</tr>
<tr>
<td>Panda Tea</td>
<td>Milk Tea</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Gimme</td>
<td>Espresso</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
Booleans and Advanced Where
Comparison Operators

The result of a comparison expression is a `bool` value

```
x = 2           y = 3
x > 1           x > y          y >= 3
x == y          x != 2         2 < x < 5
```

Assignment statements

Comparison expressions

`t.where(array_of_bool_values)` returns a table with only the rows of `t` for which the corresponding `bool` is `True`.

(Demo)
'Dog' > 'Cat' > 'Catastrophe'

True

False
Some Table Questions
Table Questions

<table>
<thead>
<tr>
<th>start</th>
<th>end</th>
<th>duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Metropolitan Ave &amp; Bedford Ave</td>
<td>Bedford Ave &amp; Nassau Ave</td>
<td>6.06667</td>
</tr>
<tr>
<td>Lafayette St &amp; E 8 St</td>
<td>2 Ave &amp; E 104 St</td>
<td>35.7</td>
</tr>
<tr>
<td>Schermerhorn St &amp; Court St</td>
<td>Court St &amp; Nelson St</td>
<td>5.46667</td>
</tr>
</tbody>
</table>

- What is the average duration of all trips?
- What is the average duration of all trips that started and ended at the same station?
- What is the average duration of all trips that started and ended at different stations?
- What is the name of the station where the most rentals ended? (Assume no ties.)
- For how many stations was the average duration of a trip ending at that station at least 10 minutes?
<table>
<thead>
<tr>
<th>TEAM</th>
<th>C</th>
<th>PF</th>
<th>PG</th>
<th>SF</th>
<th>SG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Atlanta Hawks</td>
<td>Al Horford</td>
<td>Paul Millsap</td>
<td>Jeff Teague</td>
<td>Thabo Sefolosha</td>
<td>Kyle Korver</td>
</tr>
<tr>
<td>Boston Celtics</td>
<td>Tyler Zeller</td>
<td>Jonas Jerebko</td>
<td>Avery Bradley</td>
<td>Jae Crowder</td>
<td>Evan Turner</td>
</tr>
<tr>
<td>Brooklyn Nets</td>
<td>Andrea Bargnani</td>
<td>Thaddeus Young</td>
<td>Jarrett Jack</td>
<td>Joe Johnson</td>
<td>Bojan Bogdanovic</td>
</tr>
<tr>
<td>Charlotte Hornets</td>
<td>Al Jefferson</td>
<td>Marvin Williams</td>
<td>Kemba Walker</td>
<td>Michael Kidd-Gilchrist</td>
<td>Nicolas Batum</td>
</tr>
<tr>
<td>Chicago Bulls</td>
<td>Joakim Noah</td>
<td>Nikola Mirotic</td>
<td>Derrick Rose</td>
<td>Doug McDermott</td>
<td>Jimmy Butler</td>
</tr>
<tr>
<td>Cleveland Cavaliers</td>
<td>Tristan Thompson</td>
<td>Kevin Love</td>
<td>Kyrie Irving</td>
<td>LeBron James</td>
<td>Iman Shumpert</td>
</tr>
<tr>
<td>Dallas Mavericks</td>
<td>Zaza Pachulia</td>
<td>David Lee</td>
<td>Deron Williams</td>
<td>Chandler Parsons</td>
<td>Justin Anderson</td>
</tr>
<tr>
<td>Denver Nuggets</td>
<td>JJ Hickson</td>
<td>Kenneth Faried</td>
<td>Jameer Nelson</td>
<td>Danilo Gallinari</td>
<td>Gary Harris</td>
</tr>
<tr>
<td>Detroit Pistons</td>
<td>Aron Baynes</td>
<td></td>
<td>Reggie Jackson</td>
<td>Stanley Johnson</td>
<td>Jodie Meeks</td>
</tr>
<tr>
<td>Golden State Warriors</td>
<td>Andrew Bogut</td>
<td>Draymond Green</td>
<td>Stephen Curry</td>
<td>Andre Iguodala</td>
<td>Klay Thompson</td>
</tr>
</tbody>
</table>