Homework 4: Functions, Histograms, and Groups

Recommended Reading from our book, *Computational and Inferential Thinking*:

- Visualizing Numerical Distributions
- Functions and Tables

Please complete this notebook by filling in the cells provided. Before you begin, execute the following cell to load the provided tests and import necessary modules. Each time you start your kernel, you will need to execute this cell again to load the tests.

Throughout this homework and all future ones, please be sure to not re-assign variables throughout the notebook! For example, if you use `max_temperature` in your answer to one question, do not reassign it later on. Moreover, please be sure to only put your written answers in the provided cells.

```
# Don't change this cell; just run it.

import numpy as np
from datascience import *

# These lines do some fancy plotting magic.\n",
import matplotlib
%matplotlib inline
import matplotlib.pyplot as plt
plt.style.use('fivethirtyeight')
```

Burrito-ful San Diego

Tam, Margaret and Winifred are trying to use Data Science to find the best burritos in San Diego! Their friends Irene and Maya provided them with two comprehensive datasets on many burrito establishments in the San Diego area taken from (and cleaned from): https://www.kaggle.com/srcole/burritos-in-san-diego/data

The following cell reads in a table called `ratings` which contains names of burrito restaurants, their Yelp rating, Google rating, as well as their Overall rating. It also reads in a table called `burritos_types` which contains names of burrito restaurants, their menu items, and the cost of the respective menu item at the restaurant.

```
from google.colab import drive
drive.mount('/content/drive')

     Mounted at /content/drive
```

```
#Just run this cell
ratings = Table.read_table("/content/drive/MyDrive/ratings.csv")
ratings.show(5)
burritos_types = Table.read_table("/content/drive/MyDrive/burritos_types.csv")
burritos_types.show(5)
```

Name	Yelp	Google	Overall
Albertacos	3.5	3.9	3.45
Burrito Factory	4.5	4.8	3.5
Burros and Fries	3.5	4.1	3.575
Caliente Mexican Food	3.5	4.4	3.25
California Burrito Company	3.5	4.4	3.2

... (77 rows omitted)

Name	Menu_Item	Cost
Albertacos	California	5.7
Albertacos	Carne asada	5.25
Alberto's 623 N Escondido Blvd, Escondido, CA 92025	Carne Asada	4.59
Burrito Box	Steak with guacamole	11.5
Burrito Factory	Steak everything	7.35

... (244 rows omitted)

Question 1. It would be easier if we could combine the information in both tables. Assign `burritos` to the result of joining the two tables together.

Hint: If you need refreshers on table methods, look at the python reference.

```
burritos = ratings.join("Name", burritos_types, "Name")
burritos.show(5)
```

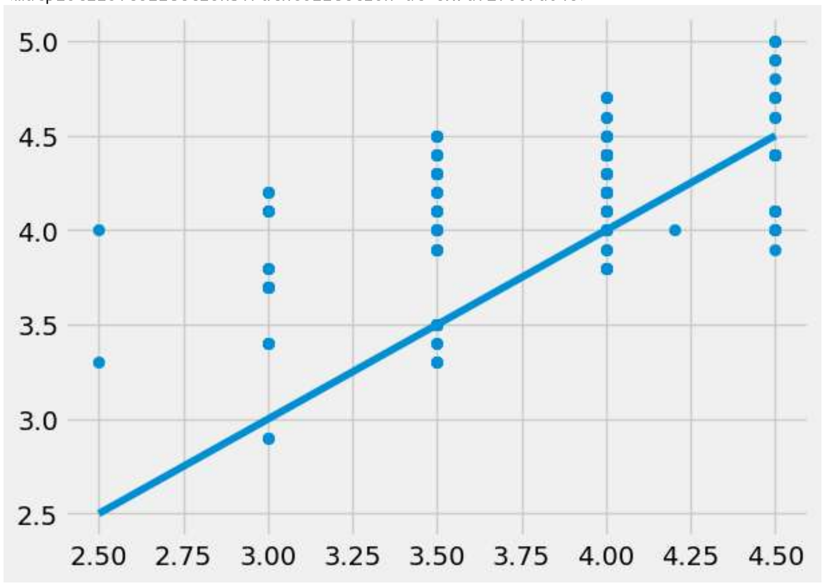
Name	Yelp	Google	Overall	Menu_Item	Cost
Albertacos	3.5	3.9	3.45	California	5.7
Albertacos	3.5	3.9	3.45	Carne asada	5.25
Burrito Factory	4.5	4.8	3.5	Steak everything	7.35
Burros and Fries	3.5	4.1	3.575	California	7.19
Burros and Fries	3.5	4.1	3.575	Carne asada	6.89

... (207 rows omitted)

Question 2. Let's look at how the Yelp scores compare to the Google scores in the `burritos` table. First, assign `yelp_and_google` to a table only containing the columns `Yelp` and `Google`. Then, make a scatter plot with Yelp scores on the x-axis and the Google scores on the y-axis.

```
yelp_and_google = burritos.select("Yelp", "Google")
yelp_and_google
# Don't change/edit/remove the following line.
# To help you make conclusions, we have plotted a straight line on the graph (y=x)
plt.plot(np.arange(2.5,5,.5), np.arange(2.5,5,.5));
plt.scatter(yelp_and_google.column(0), yelp_and_google.column(1))
```

```
<matplotlib.collections.PathCollection at 0x7df2f667d840>
```

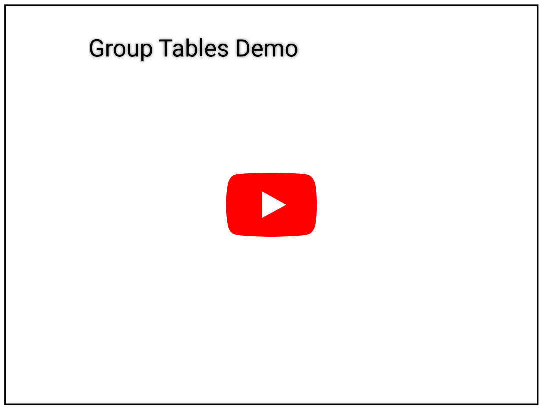


Question 3. Looking at the scatter plot you just made in Question 1.2, do you notice any pattern(s) (i.e. is one of the two types of scores consistently higher than the other one)? If so, describe them **briefly** in the cell below.

Yes, The google reviews are consistently higher- there is no label but google is my Y axis and its range is moved up by .5- not to mention most of its values sit on the higher end of that range.

Here's a refresher on how `.group` works! You can read how `.group` works in the textbook, or you can view the video below. The video resource was made by Divyesh Chotai, a member of the team who authored our textbook.

```
from IPython.display import YouTubeVideo
YouTubeVideo("HLoYTCUP0fc")
```



Question 4. From the `burritos` table, some of the restaurant locations have multiple reviews. Winifred thinks California burritos are the best type of burritos, and wants to see the average overall rating for California burritos at each location. Create a table that has two columns: the name of the restaurant and the average overall rating of California burritos at each location.

Tip: Revisit the burritos table to see how California burritos are represented.

Note: you can break up the solution into multiple lines, as long as you assign the final output table to `california_burritos`! For reference however, the staff solution only used one line.

```
california_burritos = burritos.where("Menu_Item", are.equal_to("California")).select("Name", "Overall").group("Name", np.mean)
california_burritos
```

Name	Overall mean
Burros and Fries	3.575
Caliente Mexican Food	3.25
California Burrito Company	3.2
Cancun Mexican & Seafood	4.1
Cotixan	3.6
Don Carlos Taco Shop	3.3
El Dorado Mexican Food	4.025
El Indio	4
El Pueblo Mexican Food	4.3
El Zarape	3.54815

... (36 rows omitted)

Question 5. Given this new table `california_burritos`, Winifred can figure out the name of the restaurant with the highest overall average rating! Assign `best_restaurant` to a line of code that evaluates to a string that corresponds to the name of the restaurant with the highest overall average rating.

```
best_restaurant = california_burritos.where("Overall mean", are.equal_to(max(california_burritos.column(1)))).column(0)[0]
best_restaurant
```

```
'Mikes Taco Club'
```

Question 6. Using the `burritos` table, assign `menu_average` to a table that has three columns that uniquely pairs the name of the restaurant, the menu item featured in the review, and the average Overall score for that menu item at that restaurant.

Hint: Use .group, and remember that you can group by multiple columns. Here's an example from the textbook.

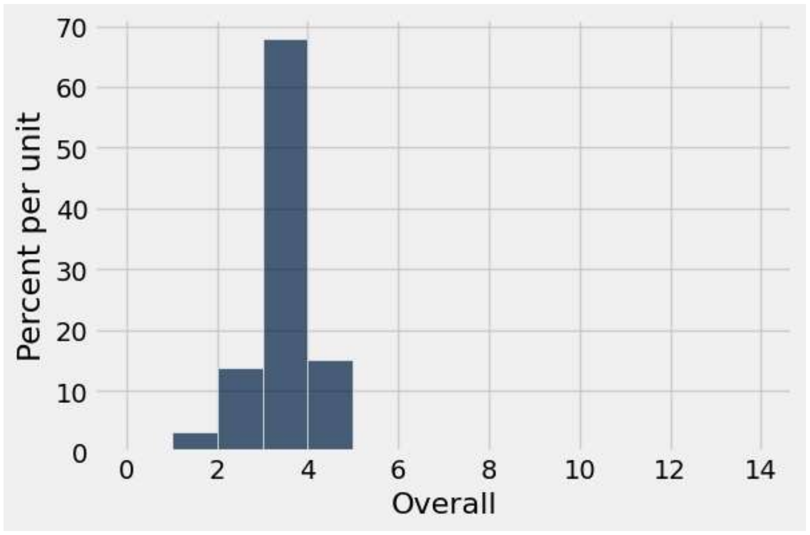
```
menu_average = burritos.select("Name", "Menu_Item", "Overall").group(["Name","Menu_Item"], np.mean)
menu_average
```

Name	Menu_Item	Overall mean
Albertacos	California	3.45
Albertacos	Carne asada	3.45
Burrito Factory	Steak everything	3.5
Burros and Fries	California	3.575
Burros and Fries	Carne asada	3.575
Burros and Fries	Shrimp california	3.575
Caliente Mexican Food	California	3.25
Caliente Mexican Food	carne asada	3.25
Caliente Mexican Food	fried fish	3.25
California Burrito Company	California	3.2

... (196 rows omitted)

Question 7. Tam thinks that burritos in San Diego are cheaper (and taste better) than the burritos in Berkeley. Plot a histogram that visualizes that distribution of the costs of the burritos from San Diego in the `burritos` table. Also use the provided `bins` variable when making your histogram, so that visually the histogram is more informative.

```
bins = np.arange(0, 15, 1)
# Please also use the provided bins
burritos.hist("Overall", overlay=False, bins=bins)
```

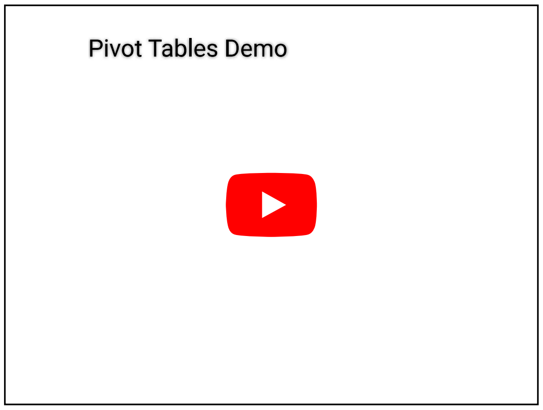


2. Faculty salaries

This exercise is designed to give you practice using the Table methods `pivot` and `group`. Here is a link to the Python reference page in case you need a quick refresher.

Run the cell below to view a demo on how you can use pivot on a table. (Thank you to textbook author team member Divyesh Chotai)

```
from IPython.display import YouTubeVideo
YouTubeVideo("4WzXo8eKLAg")
```



In the next cell, we load a dataset created by the Daily Cal which contains UC Berkeley faculty, their departments, their positions, and their gross salaries in 2015.

```
raw_profs = Table.read_table("/content/drive/MyDrive/faculty.csv").where("year", are.equal_to(2015)).drop("year", "title")
profs = raw_profs.relabeled("title_category", "position")
profs
```

name	department	position	gross_salary
CYNTHIA ABAN	South & Southeast Asian Studies	lecturer	64450
PIETER ABBEEL	Computer Science	associate professor	184998
SALLY ABEL	Law	lecturer	3466
ELIZABETH ABEL	English	professor	138775
DOR ABRAHAMSON	Education	associate professor	100300
KATHRYN ABRAMS	Law	professor	319693
BARBARA ABRAMS	Public Health	professor	191162
SARAH ACCOMAZZO	Social Welfare	lecturer	14779
CHARISMA ACEY	City and Regional Planning	assistant professor	101567
DAVID ACKERLY	Biology	professor	182288

... (2049 rows omitted)

We want to use this table to generate arrays with the names of each professor in each department.

Question 1. Set `prof_names` to a table with two columns. The first column should be called `department` and have the name of every department once, and the second column should be called `faculty` with each row in that second column containing an *array* of the names of all faculty members in that department.

Hint: Think about how `group` works: it collects values into an array and then applies a function to that array. We have defined two functions below for you, and you will need to use one of them in your call to `group`.

```
# Pick one of the two functions defined below in your call to group.
def identity(array):
    '''Returns the array that is passed through'''
    return array

def first(array):
    '''Returns the first item'''
    return array.item(0)

# Make a call to group using one of the functions above when you define prof_names
prof_names = raw_profs.select("department", "name").group("department", identity)
prof_names
```

department	name identity
African American Studies	['AYA DE LEON' 'CHIYUMA ELLIOTT' 'NIKKI JONES' 'DAVID KY ...
Agricultural and Resource Economics and Policy	['MAXIMILIAN AUFFHAMMER' 'CHARLES GIBBONS' 'JEFFREY PERL ...
Anthroplogy	['SABRINA AGARWAL' 'STANLEY BRANDES' 'CHARLES BRIGGS' ' ...
Architecture	['MARK ANDERSON' 'JACOB ATHERTON' 'WILLIAM ATWOOD' 'R.GA ...
Art History	['DILIANA ANGELOVA' 'PATRICIA BERGER' 'JULIA BRYAN-WILSO ...
Art Practice	['ALLAN DESOUZA' 'AIDA GAMEZ' 'RANDY HUSSONG' 'JENNIFER ...
Astronomy	['GIBOR BASRI' 'STEVEN BECKWITH' 'LEO BLITZ' 'EUGENE CHI ...

Understanding the code you just wrote in 2.1 is important for moving forward with the class! If you made a lucky guess, take some time to look at the code, step by step.

Question 2. At the moment, the `name` column of the `profs` table is sorted by last name. Would the arrays you generated in the `faculty` column of the previous part be the same if we had sorted by first name instead before generating them? Two arrays are the **same** if they contain the same number of elements and the elements located at corresponding indexes in the two arrays are identical. An example of arrays that are

They would be different arrays because elements would have different indices- despite being the same size.

Question 3. Set `department_ranges` to a **pivot table** containing departments as the rows, and the position as the columns. The values in the rows should correspond to a salary range, where range is defined as the **difference between the highest salary and the lowest salary in the department for that position**.

Hint: First you'll need to define a new function `salary_range` which takes in an array of salaries and returns the range of salaries in that array.

```
# Define salary_range first
def salary_range(array):
    return max(array)-min(array)
department_ranges = raw_profs.pivot("title_category", "department", "gross_salary", salary_range)
department_ranges
```

```
department  assistant professor  associate professor  lecturer  professor
```