

HW 04 Autograder

● Graded

Student

Sangwon Ji

Total Points

60 / 60 pts

Autograder Score

60.0 / 60.0

Passed Tests

Public Tests

Question 2

Early Submission Bonus

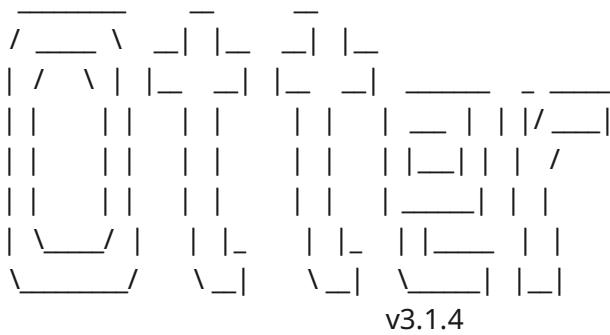
0 / 0 pts

+ 5 pts Early Submission Bonus

✓ + 0 pts No bonus

Autograder Results

Autograder Output



<IPython.core.display.HTML object>
<IPython.core.display.HTML object>
<IPython.core.display.HTML object>
<IPython.core.display.HTML object>
<IPython.core.display.HTML object>

----- GRADING SUMMARY -----

Error encountered while trying to verify scores with log:
'TestCaseResult' object has no attribute 'hidden'

Successfully uploaded submissions for: sangwon@berkeley.edu

Total Score: 60.000 / 60.000 (100.000%)

	name	score	max_score
0	Public Tests	NaN	NaN
1	q1_1	8.0	8.0
2	q1_2	1.0	1.0
3	q1_4	9.0	9.0
4	q1_5	8.0	8.0
5	q1_7	8.0	8.0
6	q2_1	9.0	9.0
7	q2_5	9.0	9.0
8	q2_3	9.0	9.0

Public Tests

q1_1 results: All test cases passed!

q1_2 results: All test cases passed!

q1_4 results: All test cases passed!

q1_5 results: All test cases passed!

q1_7 results: All test cases passed!

q2_1 results: All test cases passed!

q2_5 results: All test cases passed!

q2_3 results: All test cases passed!

Submitted Files

In [35]:

```
# Initialize Otter
import otter
grader = otter.Notebook("hw04.ipynb")
```

Homework 4: Functions, Tables, and Groups

Please complete this notebook by filling in the cells provided. Before you begin, execute the previous cell to load the provided tests.

Helpful Resource:

- [Python Reference](#): Cheat sheet of helpful array & table methods used in Data 8!

Recommended Readings:

- [Visualizing Numerical Distributions](#)
- [Functions and Tables](#)

Please complete this notebook by filling in the cells provided. Before you begin, execute the following cell to setup the notebook by importing some helpful libraries. Each time you start your server, you will need to execute this cell again.

For all problems that you must write explanations and sentences for, you **must** provide your answer in the designated space. **Moreover, 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. Otherwise, you will fail tests that you thought you were passing previously!

Deadline:

This assignment is due **Tuesday, 7/5 at 11:59pm PT**. Turn it in by Monday, 7/4 at 11:59pm PT for 5 extra credit points. Late work will not be accepted as per the [policies](#) page.

Note: This homework has hidden tests on it. That means even though the tests may say 100% passed, it doesn't mean your final grade will be 100%. We will be running more tests for correctness once everyone turns in the homework.

Directly sharing answers is not okay, but discussing problems with the course staff or with other students is encouraged. Refer to the policies page to learn more about how to learn cooperatively.

You should start early so that you have time to get help in OH if you're stuck. The schedule appears on <http://data8.org/su22/office-hours.html>.

1. Burrito-ful San Diego

In [36]:

```
# Run this cell to set up the notebook, but please don't change it.

# These lines import the Numpy and Datascience modules.
import numpy as np
from datascience import *
import d8error

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

import warnings
warnings.simplefilter('ignore', FutureWarning)
warnings.filterwarnings("ignore")
```

Mira, Sofia, and Sara are trying to use Data Science to find the best burritos in San Diego! Their friends Jessica and Sonya 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. The `Overall` rating is not an average of the `Yelp` and `Google` ratings, but rather it is the overall rating of the customers that were surveyed in the study above.

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.

In [37]:

```
# Just run this cell
```

```
ratings = Table.read_table("ratings.csv")
ratings.show(5)
burritos_types = Table.read_table("burritos_types.csv").drop(0)
burritos_types.show(5)
```

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

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, so that we have a table with the ratings for every corresponding menu item from every restaurant. Each menu item has the same rating as the restaurant from which it is from. **(8 Points)**

Note: It doesn't matter which table you put in as the argument to the table method, either order will work for the autograder tests.

Hint: If you need help on using the `join` method, refer to the [Python Reference Sheet](#) or [Section 8.4](#) in the textbook.

In [38]:

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

<IPython.core.display.HTML object>

In [39]:

```
grader.check("q1_1")
```

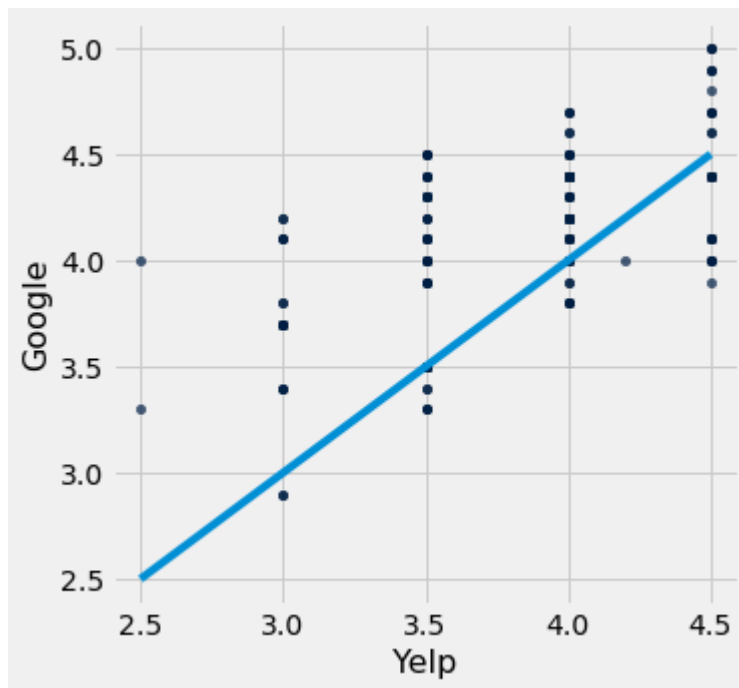
Out [39]: q1_1 results: All test cases passed!

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. **(8 Points)**

In [40]:

```
yelp_and_google = burritos.select('Yelp', 'Google')
yelp_and_google.scatter('Yelp', '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));
```



In [41]: `grader.check("q1_2")`

Out [41]: q1_2 results: All test cases passed!

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. (8 Points)

Not all of them, but mostly, the scores seems to come out higher in the Google's score more. With the scatter plot above, it is easy to find out that looking at the x-axis, Yelp scores and comparing it with the scores that come out on Google, Google's ones are higher. Also, with the blue line that shows the equal number for Yelp and Google, majority of the scores seems to be above. This seems to be a consisten pattern as numbers doesn't go that below to the ones of Yelp ones.

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 a past staff member, Divyesh Chotai!

You can also use the [Table Functions Visualizer](#) to get some more hands-on experience with the `.group` function.

In [42]: `from IPython.display import YouTubeVideo`

Out [42]:

Flavor	
chocolate	[light brown, dark brown]
strawberry	[pink, pink]
vanilla	

cones.group("Flavor")

Flavor	Color
strawberry	pink
chocolate	light brown
chocolate	dark brown
strawberry	pink
chocolate	dark brown
vanilla	white

cones

Question 4. There are so many types of California burritos in the `burritos` table! Sara wants to consider her options for burritos based on rankings. For the sake of these questions, we are treating each menu item's rating the same as its respective restaurant's, as we do not have the rating of every single item at these restaurants. You do not need to worry about this fact, but we thought to mention it!

Create a table with two columns: the first column include the names of the burritos and the second column should contain the average overall rating of that burrito across restaurants. **In your calculations, you should only compare burritos that contain the word "California"**. For example, there are "California" burritos, "California Breakfast" burritos, "California Surf And Turf" burritos, etc. **(9 Points)**

Hint 1: If multiple restaurants serve the "California - Chicken" burrito, what table method can we use to aggregate those together and find the average overall rating?

Hint 2: "California" is case sensitive!

Note: For reference, the staff solution only used one line. However, feel free to break up the solution into multiple lines and steps; just make sure you assign the final output table to `california_burritos`!


```
In [43]: california_burritos = burritos.where("Menu_Item",
are.containing("California")).select("Menu_Item", "Overall").group("Menu_Item",
np.average)
california_burritos
```

```
Out [43]: Menu_Item          | Overall average
California          | 3.5242
California (Only Cheese) | 4.1
California + Guac + Sour Cream | 3.4
California - Chicken    | 3.45839
California - Pork Adobada | 3.26429
California - Steak      | 3.26429
California Breakfast    | 2.75833
California Chicken      | 3.54815
California Chipotle     | 4.36667
California Everything    | 4.1
... (9 rows omitted)
```

```
In [44]: grader.check("q1_4")
```

```
Out [44]: q1_4 results: All test cases passed!
```

Question 5. Given this new table `california_burritos`, Sara can figure out the name of the California burrito with the highest overall average rating! Assign `best_california_burrito` to a line of code that outputs the string that represents the name of the California burrito with the highest overall average rating. If multiple burritos satisfy this criteria, you can output any one of them. **(8 Points)**

```
In [45]: best_california_burrito = california_burritos.sort("Overall average",
descending=True).column("Menu_Item").item(0)
best_california_burrito
```

```
Out [45]: 'California Chipotle'
```

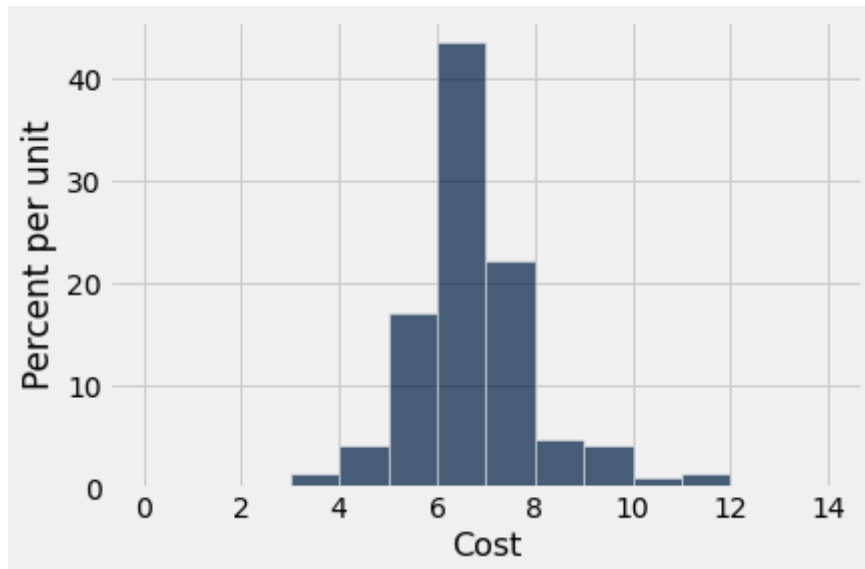
```
In [46]: grader.check("q1_5")
```

```
Out [46]: q1_5 results: All test cases passed!
```

Question 6. Mira 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 the histogram is more visually informative. **(8 Points)**

```
In [47]: bins = np.arange(0, 15, 1)
# Please also use the provided bins
burritos.hist("Cost", bins = bins)
```



Question 7. What percentage of burritos in San Diego are less than \$6? Assign `burritos_less_than_6` to your answer, which should be between 0 and 100. You should only use the histogram above to answer the question. Do not use code on the table to find the answer, just eyeball the heights and use Python to evaluate your arithmetic! **(8 Points)**

Note: Your answer does not have to be exact, but it should be within a couple percentages of the staff answer.

```
In [48]: burritos_less_than_6 = 22.2
burritos_less_than_6
```

Out [48]: 22.2

```
In [49]: grader.check("q1_7")
```

Out [49]: q1_7 results: All test cases passed!

2. San Francisco City Employee Salaries

This exercise is designed to give you practice with using the Table methods `.pivot` and `.group`. Here is a link to the [Python Reference Sheet](#) in case you need a quick refresher. The [Table Function Visualizer](#) may also be a helpful tool.

Run the cell below to view a demo on how you can use pivot on a table.
(Thank you to past staff Divyesh Chotai!)

```
In [50]: from IPython.display import YouTubeVideo
YouTubeVideo("4WzXo8eKLA")
```

Out [50]:



Color	chocolate	strawberry	vanilla
dark brown			
light brown	[3]		
pink		[2.5]	
white			

Flavor	Color	Price
strawberry	pink	2.5
chocolate	light brown	3
chocolate	dark brown	3.5
strawberry	pink	2.5
chocolate	dark brown	3.5
vanilla	white	3

cones.pivot("Flavor", "Color", "Price", np.sum) cones

The data source we will use within this portion of the homework is [publicly provided](#) by the City of San Francisco. We have filtered it to retain just the relevant columns and restricted the data to the calendar year 2019. Run the following cell to load our data into a table called `full_sf`.

```
In [51]: full_sf = Table.read_table("sf2019.csv")
full_sf.show(5)
```

<IPython.core.display.HTML object>

The table has one row for each of the 44,525 San Francisco government employees in 2019.

The first four columns describe the employee's job. For example, the employee in the third row of the table had a job called "IS Business Analyst-

Senior". We will call this the employee's *position* or *job title*. The job was in a Job Family called Information Systems (hence the IS in the job title), and was in the Adult Probation Department that is part of the Public Protection Organization Group of the government. You will mostly be working with the `Job` column.

The next three columns contain the dollar amounts paid to the employee in the calendar year 2019 for salary, overtime, and benefits. Note that an employee's salary does not include their overtime earnings.

The last column contains the total compensation paid to the employee. It is the sum of the previous three columns:

$$\text{Total Compensation} = \text{Salary} + \text{Overtime} + \text{Benefits}$$

For this homework, we will be using the following columns: 1.

`Organization Group`: A group of departments. For example, the Public Protection Org. Group includes departments such as the Police, Fire, Adult Protection, District Attorney, etc. 2. `Department`: The primary organizational unit used by the City and County of San Francisco. 3. `Job`: The specific position that a given worker fills. 4. `Total Compensation`: The sum of a worker's salary, overtime, and benefits in 2019.

Run the following cell to select the relevant columns and create a new table named `sf`.

In [52]:

```
sf = full_sf.select("Job", "Department", "Organization Group", "Total  
Compensation")  
sf.show(5)
```

<IPython.core.display.HTML object>

We want to use this table to generate arrays with the job titles of the members of each **Organization Group**.

Question 1. Set `job_titles` to a table with two columns. The first column should be called `Organization Group` and have the name of every "Organization Group" once, and the second column should be called `Jobs` with each row in that second column containing an *array* of the names of all the job titles within that "Organization Group". Don't worry if there are multiple of the same job titles. **(9 Points)**

Hint 1: 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`.

Hint 2: It might be helpful to create intermediary tables and experiment with the given functions.

In [53]:

```
# Pick one of the two functions defined below in your call to group.
def first_item(array):
    """Returns the first item"""
    return array.item(0)

def full_array(array):
    """Returns the array that is passed through"""
    return array

# Make a call to group using one of the functions above when you define
job_titles
job_titles = sf.group("Organization Group", full_array).drop("Department
full_array", "Total Compensation full_array").reabeled("Job full_array", "Jobs")
job_titles
```

Out [53]:

Organization Group	Jobs
Community Health	['Painter Supervisor 1' 'Painter' 'Painter' ... 'Nursing
Culture & Recreation	['Electrician' 'Executive Secretary 2' 'Bldgs & Groun
General Administration & Finance	['Painter' 'Painter' 'Electrician' ... 'Investigator
Human Welfare & Neighborhood Development	['Dept Head I' 'Administrative Analys
Public Protection	['IS Trainer-Journey' 'IS Engineer-Assistant' 'IS Busine
Public Works, Transportation & Commerce	['Heavy Equip Ops Asst Sprv' 'Heavy Equi
...	

In [54]:

```
grader.check("q2_1")
```

Out [54]:

q2_1 results: All test cases passed!

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. Office hours is always a great resource!

Question 2. At the moment, the `Job` column of the `sf` table is not sorted (no particular order). Would the arrays you generated in the `Jobs` column of the previous question be the same if we had sorted alphabetically instead before generating them? Explain your answer. To receive full credit, your answer should reference *how* the `.group` method works, and how sorting the `Jobs` column would affect this. **(8 Points)**

Note: 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 NOT the same:

```
array([1,2]) != array([2,1]).
```

The results won't come out the same as it will display different result since the whole array will also come out differently. .group works by collecting and rowing the value in a column by using values and combinations. Sorting the Jobs column will affect this as it is located differently and would change the rows, eventually showing the different results.

Question 3. Set `department_ranges` to a table containing departments as the rows, and the organization groups as the columns. The values in the rows should correspond to a total compensation range, where range is defined as the **difference between the highest total compensation and the lowest total compensation in the department for that organization group.** (9 Points)

Hint 1: First you'll need to define a new function `compensation_range` which takes in an array of compensations and returns the range of compensations in that array.

Hint 2: What table function allows you to specify the rows and columns of a new table? You probably watched a video on it earlier in the homework!

In [55]:

```
# Define compensation_range first
def compensation_range(array):
    return max(array)- min(array)

department_ranges = sf.pivot("Organization Group","Department",
values="Total Compensation", collect=compensation_range)
department_ranges
```

Out [55]:

Department	Community Health	Culture & Recreation	General Administration
Academy Of Sciences	0	199121	0
Administrative Services	0	0	478784
Adult Probation	0	0	0
Airport Commission	0	0	0
Art Commission	0	251823	0
Asian Art Museum	0	298230	0
Assessor	0	0	277385
Board Of Appeals	0	0	0
Board Of Supervisors	0	0	293773

Question 4. Give an explanation as to why some of the row values are 0 in the `department_ranges` table from the previous question. **(8 Points)**

It is because there isn't a number that's existing for that department in the organization group. There is no data for it as one department belongs to one organization group. There is why the pivot is outputting the data 0.

Question 5. Find the number of departments appearing in the `sf` table that have an average total compensation of greater than 125,000 dollars; assign this value to the variable `num_over_125k`. **(9 Points)**

Hint: The variable names provided are meant to help guide the intermediate steps and general thought process. Feel free to delete them if you'd prefer to start from scratch, but make sure your final answer is assigned to `num_over_125k`!

```
In [56]: num_over_125k = sf.drop("Organization Group", "Job").group("Department",  
np.mean).where("Total Compensation mean", are.above(125000)).num_rows  
num_over_125k
```

Out [56]: 23

```
In [57]: grader.check("q2_5")
```

Out [57]: q2_5 results: All test cases passed!

You're done with Homework 4!

Important submission steps: 1. Run the tests and verify that they all pass. 2. Choose **Save Notebook** from the **File** menu, then **run the final cell**. 3. Click the link to download the zip file. 4. Go to [Gradescope](#) and submit the zip file to the corresponding assignment. The name of this assignment is "Homework 4 Autograder".

It is your responsibility to make sure your work is saved before running the last cell.

To double-check your work, the cell below will rerun all of the autograder tests.

In [58]: `grader.check_all()`

Out [58]: q1_1 results: All test cases passed!
q1_2 results: All test cases passed!
q1_4 results: All test cases passed!
q1_5 results: All test cases passed!
q1_7 results: All test cases passed!
q2_1 results: All test cases passed!
q2_3 results: All test cases passed!
q2_5 results: All test cases passed!

Submission

Make sure you have run all cells in your notebook in order before running the cell below, so that all images/graphs appear in the output. The cell below will generate a zip file for you to submit. **Please save before exporting!**


In []: `# Save your notebook first, then run this cell to export your submission.
grader.export(pdf=False)`

▼ .OTTER_LOG

 Download

1 Binary file hidden. You can download it using the button above.

▼ __zip_filename__

 Download

1 `hw04_2022_07_05T14_39_01_504467.zip`