Data 8
Summer 2018
Worksheet 1
6/30/2018
Conceptual Office Hours

Name: ______________________

This worksheet can serve as a general overview of materials you have learned this week and give you an opportunity to practice solving them in an exam-like format. NOTE: This worksheet is a collection of problems that come from resources all students have access to, resources only staff members of previous iterations of the course have access to, and some that I have made up myself. This is not necessarily representative of what will be tested on the actual exams. For any issues or questions, contact Robert Sweeney Blanco at robertsweeneyblanco@berkeley.edu.

For all problems, you may assume you have imported datascience and numpy as np.

1. Which lines of code evaluates to an array of the first five even numbers starting at 2; 2,...,10 (Check all that apply).

   ○ `np.arange(10)`

   ○ `np.arange(2, 10, 2)`

   ○ np.arange(2, 11, 2)

   ○ np.arange(2, 12, 2)

   ○ np.arange(1, 6) * 2

   ○ `np.arange(1, 5) * 2`

Suppose the table below is set to the variable *Fall2018*. This table shows the name of the class, its size in students, and the professor's name.

Class	Size	Professor
CS61A	1625	John Denero
CS61B	987	Josh Hug
Data 8	1256	Ani Adhikari

2. Suppose you are not interested in who is teaching the class (fatal mistake), which of the following does NOT evaluate to a new table without the 'professor' column?
   - ○ classes.drop("Professor")
   - [✓ classes.columns("Class", "Size")]
   - ○ classes.select("Class", "Size")
   - ○ classes.select(0,1)

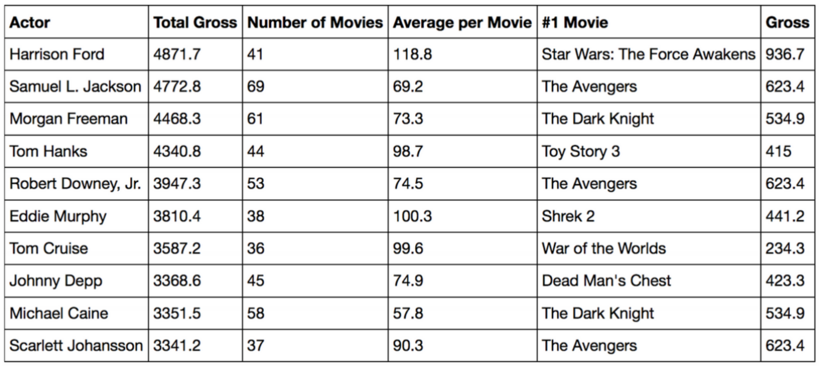
3. Suppose it is projected that all class sizes at UC Berkeley will increase 7% in the Spring of 2019. Write code that returns a new table with *Fall2018* info with a new column "Projected" with the projected sizes (don't worry about decimals).
   Fall2018.with_column("Projected", Fall2018.column("Size") * 1.07)

   Suppose course staffs from different classes form basketball teams and play in a tournament. The number of games each team wins is recorded, as well as the total number of points scored. The results are stored in the table *results*.

Class	Wins	Points
CS10	5	225
CS61A	9	270
CS61B	9	265
CS61C	5	202
Data8	12	285
CS70	5	190

4. Write a line of code that calculates the average number of points scored by both teams in a game.
   sum(results.column("Points")) / sum(results.column("Wins"))

Suppose we have the following table set to the variable *actors*.



Actor	Total Gross	Number of Movies	Average per Movie	#1 Movie	Gross
Harrison Ford	4871.7	41	118.8	Star Wars: The Force Awakens	936.7
Samuel L. Jackson	4772.8	69	69.2	The Avengers	623.4
Morgan Freeman	4468.3	61	73.3	The Dark Knight	534.9
Tom Hanks	4340.8	44	98.7	Toy Story 3	415
Robert Downey, Jr.	3947.3	53	74.5	The Avengers	623.4
Eddie Murphy	3810.4	38	100.3	Shrek 2	441.2
Tom Cruise	3587.2	36	99.6	War of the Worlds	234.3
Johnny Depp	3368.6	45	74.9	Dead Man's Chest	423.3
Michael Caine	3351.5	58	57.8	The Dark Knight	534.9
Scarlett Johansson	3341.2	37	90.3	The Avengers	623.4

5. Write a line of code to find the actor who has made the most movies. Do not return a table with the actors name; just return the actors name.

```
actors.sort(”Number of Movies”, descending=True).column(”Actor”).item(0)
OR
actors.where(”Number of Movies”, max(actors.column(”Number of movies”))).column(”Actor”).item(0)
```

6. What is Tom Hanks’ #1 movie? Write a line of code to find out.

```
actors.where(”Actor”, are.equal to(”Tom Hanks”)).column(”#1 Movie”).item(0)
```

7. We’ll finish this table off with a pretty involved query; Write a line of code which returns a table that contains only the actors column. The elements in the actors column are the names of actors who have made more than 40 movies and have a total gross below 3000.

```
over40 = actors.where(”Number of Movies”, are.above(40))
over40.where(”Total Gross”, are.below(3000)).select(”Actor”)
```

Suppose the following table of NBA data is set to the variable *players*.

Rk	Player	Pos	Age	Team
1	Stephen Curry	PG	27	GSW
2	James Harden	SG	26	HOU
3	Kevin Durant	SF	27	OKC
4	DeMarcus Cousins	C	25	SAC
5	LeBron James	SF	31	CLE
6	Damian Lillard	PG	25	POR
7	Anthony Davis	PF	22	NOP
8	DeMar DeRozan	SG	26	TOR
9	Russell Westbrook	PG	27	OKC
10	Paul George	SF	25	IND

... (466 rows omitted)

8. Define a function which takes in three NBA players found in the nba table and returns the oldest player. Don't be afraid of using multiple lines!

```
def oldest(p1,p2,p3):
    player_table = nba.where(‘Player’,are.contained_in(make_array(p1,p2,p3))
    return player_table.sort(‘Age’,descending=True).column(‘Player’).item(0)
```

9. Write a line of code that evaluates to the NBA team with the youngest average age.

```
players.group(”Team”, np.mean).sort(”Age mean”).column(”Team”).item(0)
```

The *cafe* table (left) describes the Yelp reviews for three cafes on Euclid. Every cafe has a count for the number of 3-star, 4-star, and 5-star reviews, in that order. The *price* table (right) describes coffee prices.

name	stars	count
Nefeli	3	37
Nefeli	4	75
Nefeli	5	50
Brewed	3	56
Brewed	4	71
Brewed	5	37
Abe	3	1
Abe	4	2
Abe	5	17

name	$
Nefeli	3
Brewed	3
Abe	2

Complete the Python expressions below to compute each result. For example, if the result prompt said, "The total number of reviews of all cafes," then you would write: sum (cafe.column(2)) *** **You must fit your solution into the lines and spaces provided to receive full credit.** *** The last line of each answer should evaluate to the result requested; you never need to call print.

10. The total number of reviews of the cafe named Nefeli.

    sum(cafe.where('name', 'Nefeli').column('count'))

11. The total number of reviews of the cafe with the fewest reviews

    min(cafe.group('name', sum).column(2))

12. An array containing the names of all cafes that have above-average coffee prices.

    price.where('$', are.above(np.average(price.column('$')))).column('name')

13. Among all reviews of cafes with $3 coffee, the proportion that are 3-star reviews.

    j = cafe.join('name', price).where('$', 3)

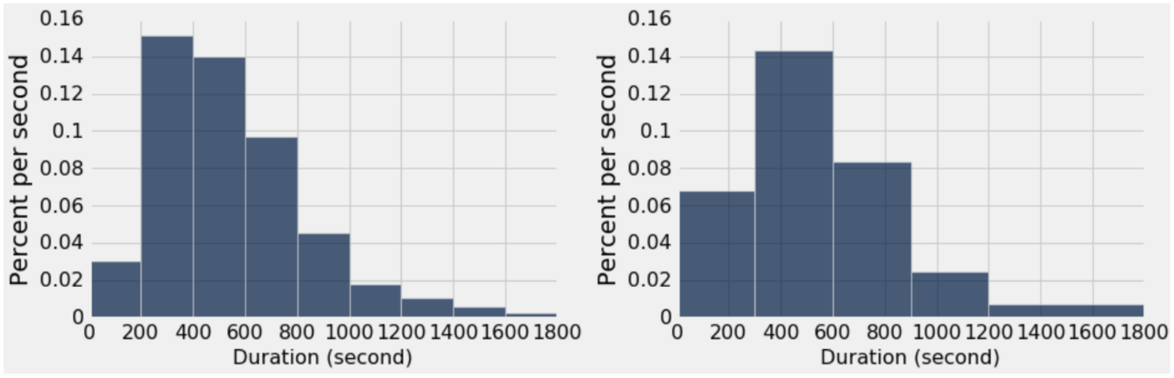
    sum(j.where('stars', 3).column('count')) / sum(j.column('count'))

14. The table below, in which each row describes the number of reviews with a particular star rating for every cafe.

stars	Abe	Brewed	Nefeli
3	1	56	37
4	2	71	75
5	17	37	50

cafe.pivot('name', 'stars', 'count', sum))

The two histograms of bike trip durations below were both generated by trip.hist(...) using different bins.



Write the proportion of trips that fall into each range of durations below. Show your work. If it is not possible to tell from the histograms, instead write Not enough information.

15. Between 200 (inclusive) and 400 (exclusive) seconds

    0.0015 * 200 == 0.30 or 30%

16. Between 300 (inclusive) and 900 (exclusive) seconds

    0.0014 * 300 + 0.0008 * 300 == 0.66 or 66%

17. Between 400 (inclusive) and 900 (exclusive) seconds
    0.0014 * 200 + 0.0008 * 300 == 0.52 or 52%

18. Between 200 (inclusive) and 300 (exclusive) seconds
    0.0015 * 200 + 0.0014 * 200 - 0.0014 * 300 == 0.16 or 16%
    Also accepted,
    0.0007 * 300 - 0.0003 * 200 == 0.15 or 15%