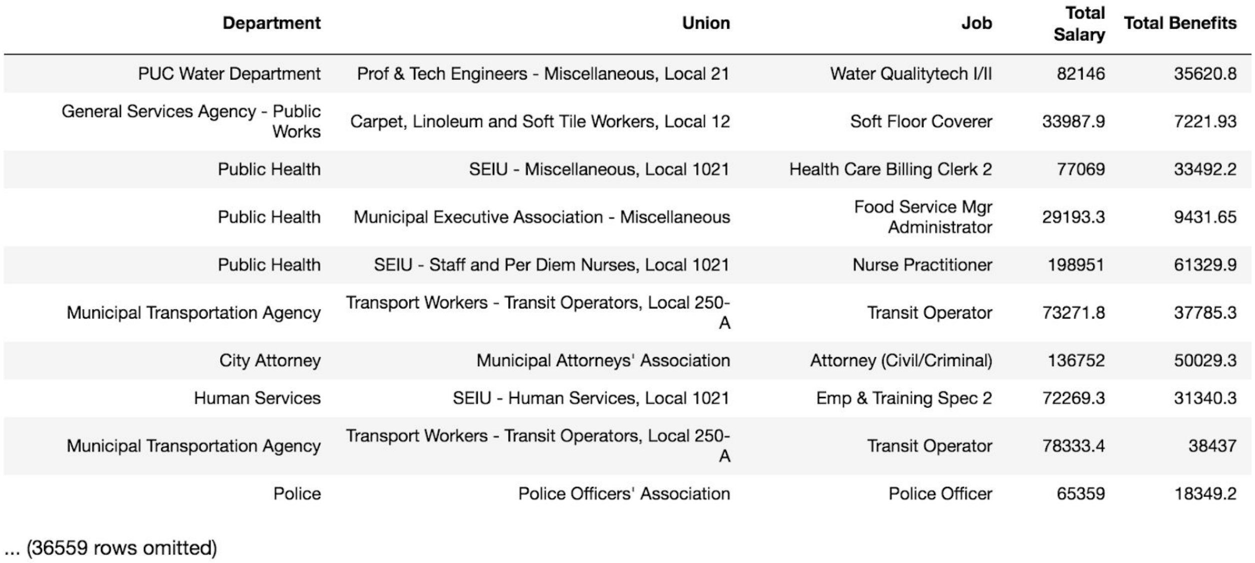
Data 8 Summer 2018 Coding Worksheet

For all the questions below, assume we have imported numpy as `np` and `datascience`.

1.

   The table `pay` (shown below), contains information on a large random sample employees of the City of San Francisco. These include medical professionals, firefighters, elected officials, and so forth.



Department	Union	Job	Total Salary	Total Benefits
PUC Water Department	Prof & Tech Engineers - Miscellaneous, Local 21	Water Qualitytech I/II	82146	35620.8
General Services Agency - Public Works	Carpet, Linoleum and Soft Tile Workers, Local 12	Soft Floor Coverer	33987.9	7221.93
Public Health	SEIU - Miscellaneous, Local 1021	Health Care Billing Clerk 2	77069	33492.2
Public Health	Municipal Executive Association - Miscellaneous	Food Service Mgr Administrator	29193.3	9431.65
Public Health	SEIU - Staff and Per Diem Nurses, Local 1021	Nurse Practitioner	198951	61329.9
Municipal Transportation Agency	Transport Workers - Transit Operators, Local 250-A	Transit Operator	73271.8	37785.3
City Attorney	Municipal Attorneys' Association	Attorney (Civil/Criminal)	136752	50029.3
Human Services	SEIU - Human Services, Local 1021	Emp & Training Spec 2	72269.3	31340.3
Municipal Transportation Agency	Transport Workers - Transit Operators, Local 250-A	Transit Operator	78333.4	38437
Police	Police Officers' Association	Police Officer	65359	18349.2

... (36559 rows omitted)

Based on this table alone, we want to estimate the average compensation (the sum of an employee's salary and benefits) of *everyone* employed by the City of San Francisco using the bootstrap.

a. First, let's write a function to compute our test statistic. It should take in `pay_sample,` a table containing the columns "`Total Salary`" and "`Total Benefits,`" and output the average of an array that contains the sum of these two columns

```
def compensation(pay_sample):

    salary = pay_sample._______(_________)

    Benefits = pay_sample._______(_________)

    return __________(_____________________________)
```

b. Next, let's write a function to bootstrap the means. It should take in a table that contains the original sample (a table like `pay`), and a number of repetitions, then return an array that contains all the bootstrapped means.

```
def bootstrapped_means(original_tbl, reps):
     necessary_columns = original_tbl.select(___,  ____)

     means = __________

     for i in ___________:

          resampled = ___________.________

          resampled_mean = compensation(_________)

          _________(__________, ____________)

     ____________
```

c. Construct a 90% confidence interval for the means generated by 5,000 bootstrap resamples of `pay`. Store it in an array called `ninety_ci`

```
pay_means = _____________(_______, _______)

upper_bound = ____________(_______, _______)

lower_bound = ____________(_______, _______)

ninety_ci = ___________(__________,__________)
```

2.

Using `pay`, answer the following

a. Set `best_ten_benefits` equal to a table with 3 columns: Department, Union, and Greatest Benefits. Each row should contain a unique combination of department and union, as well as the greatest benefits package offered to an employee in both. Select only the rows for the 10 greatest benefits packages.

```
___________ =
pay._______________(_________________________)

___________ =
_________._____________(_______________________)

___________ =
_________._____________(_______________________)
```

```
___________ =
_________.____________(______________________)

best_ten_benefits =
_________.__________(_________________)
```

b. Define a function `pay_attributes` that takes 2 arguments: a string that’s either ‘Total Salary’ or ‘Total Benefits’ and another function, and returns a table with 3 Columns. There should be one row for every combination of department and union. The values in the 3rd column should be equal to the value returned by the aggregation function (i.e. if max is passed in, the 3rd column should contain the greatest benefits package offered to members of a specific combination of department and union)

```
def pay_attributes(column_name, f):
     result =
pay.___________(____________________________)

     .___________(________________________________________
)

     return result
```

c. Using the function you defined in part b, create a histogram for the distribution of the average salaries for each combination of department and union. If you aren’t sure about your answer to b, assume you have a working version of `pay_attributes.`

```
pay_attributes(___________________________________________
_)


     .____________(_________________________________)
```

3. You’ve been buying jelly beans from a local candy store that only sells watermelon jelly beans. However, you notice that sometimes there are licorice jelly beans in your order. You suspect that the jelly bean machine is broken, and there’s a 1% chance that a licorice jelly bean is produced instead of a watermelon jelly bean.

a. Suppose there is indeed a 1% chance of getting a licorice jelly bean instead of watermelon. If you pick out 50 jelly beans chosen at random from among all the jelly beans, what is the chance that you find at least one licorice jelly bean? (You may assume that jelly beans are chosen with replacement from a population in which 1% of jelly beans are licorice flavored). Use simulation to compute the probability.

Let beans be an array containing 99 copies of the number 0 (to represent watermelon) and 1 copy of the number 1 (to represent liquorish).

```
beans = np.append(0*np.arange(99), 1)

trials = 5,000
licorice = ________________

for ______ in ___________________:

     chosen_beans = __________________

     licorice = __________________________________

chance_of_at_least_one = ________________________________
```

b. Define the eat function, which should simulate taking n number of jelly beans with replacement from a population in which 1% of jelly beans are licorice flavored. The eat function returns the probability of getting exactly k licorice jelly beans.

```
beans = np.append(0*np.arange(99), 1)

def eat(k, n, trials):
     """ Repeatedly pick n jelly beans and find the chance of
     getting exactly k licorice."""

     k_licorice = _______________

     for ______ in __________________:

          chosen_beans = ____________________

          k_licorice = np.append(____________,
                                np.count_nonzero(____________________
                           )

     chance_of_exactly_k_licorice =
__________________________________
```

c. You find a new store that sells a variety of jelly bean flavors. You buy a bag of 50 jelly beans, containing 10 watermelon, 20 cotton candy, and 20 blueberry. If you pick 2 jelly beans from the bag uniformly at random with replacement, what is the chance that you draw at least one that is either watermelon or cotton candy? Write your answer as a Python expression that computes the result exactly (no simulation).

4. The jelly_bean table contains the color and count of the jelly beans you want to buy.

Flavor	Count
watermelon	10
cotton candy	20
blueberry	20

The store table contains the jelly beans that each store sells and their price.

Store	Flavor	Price
A	watermelon	0.1
A	cotton candy	0.5
A	blueberry	0.3
B	watermelon	0.2
B	cotton candy	0.3
B	blueberry	0.1
C	watermelon	0.5
C	cotton candy	0.6
C	blueberry	0.8

a. Find the total cost of buying all the jelly beans you want from store A.

```
join  = store . _____________________________________
```

```
cost = sum(__________________________ *_____________________________)
```

b. Define the function cost so that the provided line of code returns a table that contains a row for each store and the cost of buying all the jelly beans you want from the store.

```
def cost(arr):

     return  ___________________________________

store_cost = store.drop(‘Flavor’).group(‘Store’, cost)
```

You want to be frugal and make sure you get the best deal on the jelly beans.

c. The function frugal takes in a flavor and returns the store that sells it at the lowest price. Fill in the blanks. Do not worry about duplicate prices.

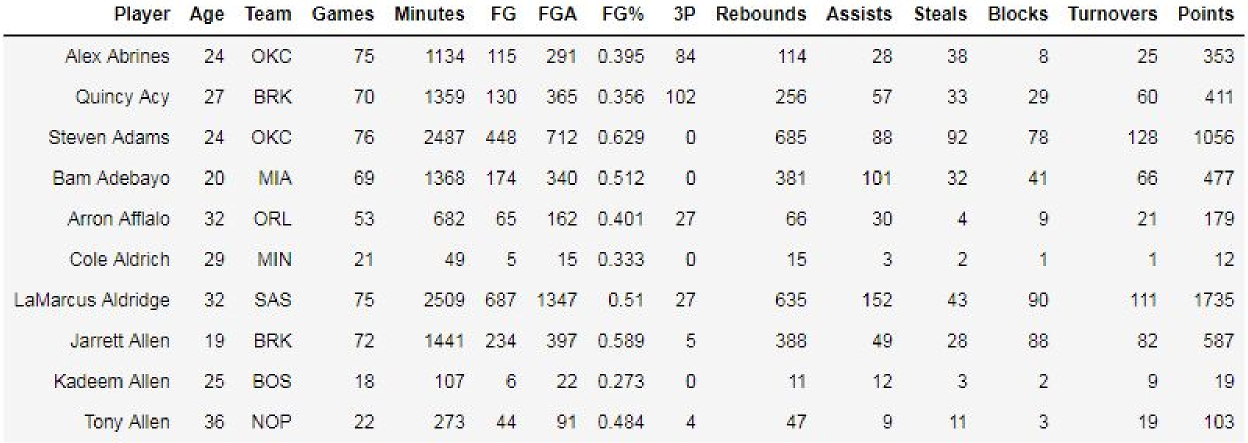
```
def frugal (flavor):

     return ___________________________________________
```

d. Use the frugal function to update the jelly_bean table so that it has a third column called ‘Cheapest Store’ which contains that name of the store that sells the flavor for the lowest price.

```
jelly_bean =

jelly_bean.with_column(_____________,________________________)
```

3.

The NBA table below contains data from the 2017-2018 season for every active player. Each row represents the totals of each players’ statistics over the whole season.



Player	Age	Team	Games	Minutes	FG	FGA	FG%	3P	Rebounds	Assists	Steals	Blocks	Turnovers	Points
Alex Abrines	24	OKC	75	1134	115	291	0.395	84	114	28	38	8	25	353
Quincy Acy	27	BRK	70	1359	130	365	0.356	102	256	57	33	29	60	411
Steven Adams	24	OKC	76	2487	448	712	0.629	0	685	88	92	78	128	1056
Bam Adebayo	20	MIA	69	1368	174	340	0.512	0	381	101	32	41	66	477
Arron Afflalo	32	ORL	53	682	65	162	0.401	27	66	30	4	9	21	179
Cole Aldrich	29	MIN	21	49	5	15	0.333	0	15	3	2	1	1	12
LaMarcus Aldridge	32	SAS	75	2509	687	1347	0.51	27	635	152	43	90	111	1735
Jarrett Allen	19	BRK	72	1441	234	397	0.589	5	388	49	28	88	82	587
Kadeem Allen	25	BOS	18	107	6	22	0.273	0	11	12	3	2	9	19
Tony Allen	36	NOP	22	273	44	91	0.484	4	47	9	11	3	19	103

a. eFG% is a an advanced analytic commonly used over FG% since it weighs three-point shots more than two-pointers due to their extra value. Calculate eFG% using this formula (FG+0.5*3P)/FGA and append the values as the column `eFG%` to this table.

```
efg =
(nba.____________+(_____________________))/________________

nba_efg = nba.__________(_____________)
```

b. Find the team with the highest average eFG% (return the name only)

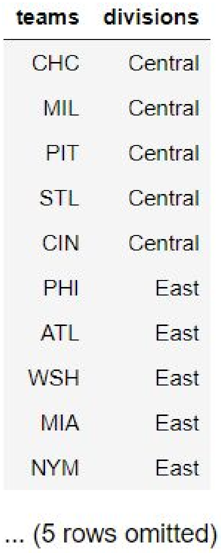
```
nba_efg._________(_______, ________).sort(_________, descending
=

                      _______).__________(__________)._____(__)
```

c. What proportion of points scored were by players who had an eFG% of at least 60%? Set `answer` to your final proportion.

```
At_least_sixty = _____(nba_efg._______(___________,

     _____._______________(_____)).__________(___________))

total = _____(nba_efg._________(___________))

answer = _______________________________
```

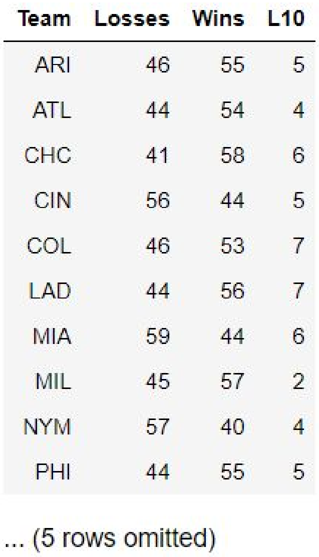
We have two tables, both shown below. The first table is called `mlb_teams` which contains 15 rows, one for each team in the National League. The first column contains the teams' abbreviated names and the second column indicates which division the team is in.

The second table is called `stats` and has four columns. The first column has the team's name, the second column has the number of wins the team has as of July 24th, 2018, the third column has the number of losses at the same time, and the fourth column contains the number of wins the team has in its ten most recent games.



teams	divisions
CHC	Central
MIL	Central
PIT	Central
STL	Central
CIN	Central
PHI	East
ATL	East
WSH	East
MIA	East
NYM	East

... (5 rows omitted)



Team	Losses	Wins	L10
ARI	46	55	5
ATL	44	54	4
CHC	41	58	6
CIN	56	44	5
COL	46	53	7
LAD	44	56	7
MIA	59	44	6
MIL	45	57	2
NYM	57	40	4
PHI	44	55	5

... (5 rows omitted)

Question 1. First combine the two tables into one table without losing any of the data. Name this table `combined`. Then reorder and rename the columns into a new table called `mlb_stats` that has the same data as `combined` but has column labels of `['Team', 'Division', 'Wins', 'Losses', 'L10']` in that order.

```
combined = _________________._______(_________________________________)

mlb_stats = combined.____________________________________________________

      ___________________________________________________________________
```

Question 2. Set `percentages` equal to an array that contains the win percentage for each team. The win percentage is the number of wins by a team divided by the number of games they have played. Then create a new table called `mlb` that has all of the same columns as `mlb_stats` plus a new column called ‘`Percentage`’ with the win percentages.

```
percentages = _________________________________/(_______________________

      __________________________________________________________________)

mlb = _________________.____________________(‘Percentage’, percentages)
```

Question 3. We're interested in knowing which division has the highest win percentage. First make a two column table with three rows. The first column has the division name and the second column has the average win percentage for all the teams in that division. Each row should represent one of the three divisions. Set `division` equal to this table.

```
division = mlb.____________________________________________________________
```

Question 4. Which division has the highest win percentage? Set `best` equal to the division's name.

```
best = division.______________(___________________________________________)

       .__________________(____________________________)._____________
```

Question 5. Next we want to know the number of teams that have a win percentage above the 75th percentile of all win percentages in the National League. Set `percentile_75` to the 75th percentile of all win percentages in our data. Set `above_75` to the number of teams with win percentages above the 75th percentile.

```
percentile_75 = ___________________(_____________,_______________________)

above_75 = ____________.______________(_____________________,_____________

      _________________________________________.___________________________
```

Question 6. We want to be able to look up whether or not a team has a winning record or a losing record. A team has a winning record when they've won 50% or more of their games. Assume that we've defined a function called `winning_and_losing` which takes in a single team name as an argument and returns "`Winning`" if the team has a winning record and "`Losing`" otherwise. Use `winning_and_losing` along with some other functions to add another column to `mlb` called '`W/L`' that says whether a team has a '`Winning`' or '`Losing`' record.

```
mlb.___________________(‘W/L’,___________________________________________)
```

Question 7. We believe that how well a team has played in their 10 most recent games is an indicator of how well they will do in upcoming games. When looking at two random teams, we want to know which team will have the advantage based on this belief.

Write a function called `advantage` that takes in two randomly selected rows of `mlb` table and outputs which team has the perceived advantage based on the number of wins in their respective last 10 games. If the home team has the advantage, have the function output ‘`Home Team`’. If the away team has the advantage, have the function output ‘`Away Team`’. If both teams have the same number of wins in the last 10 games, have the function output ‘`Neither`’. For convenience, assume the first row selected is the home team and the second row selected is the away team.

```
def advantage(two_rows):
      home_team = _____________________________

      away_team = _____________________________

      ___________________________________________________________________:

            ________________________________

      ___________________________________________________________________:

            ________________________________

      ______

            ________________________________

advantage(mlb.sample(2))
```