DATA 6
Summer 2025

Jedi Tsang
Final

Solutions last updated: Thursday, August 14, 2025

Your name: ______________________________

Your student ID: ______________________________

Your Berkeley email: ______________________________

Your room location: ______________________________

Student ID of the person to your left: ______________________________

Student ID of the person to your right: ______________________________

You have 110 minutes. There are 6 questions of varying credit. (62 points total)

Question:	HC	1	2	3	4	5	6	Total
Points:	1	9	15	16	11	10	0	62

For questions with **circular bubbles**, you may select only one choice.

○ Unselected option (Completely unfilled)

☑ Don't do this (it will be graded as incorrect)

● Only one selected option (completely filled)

For questions with **square checkboxes**, you may select one or more choices.

■ You can select

■ multiple squares

☑ (Don't do this)

Anything you write outside the answer boxes or you ~~cross out~~ will not be graded. If you write multiple answers, your answer is ambiguous, or the bubble/checkbox is not entirely filled in, we will grade the worst interpretation. For coding questions with blanks, you may write at most one statement per blank and you may not use more blanks than provided.

As a member of the UC Berkeley community, I act with honesty, integrity, and respect for others. I will follow the rules of this exam.

I have read and agree to the honor code above.

(1 point) Sign your name: ______________________________

Q1 *Potpourri (True/False)* (9 points)

Q1.1 (1 point) TF-IDF can be used to help create embeddings (numerical representations) of words.

● True ○ False

Q1.2 (1 point) TF-IDF can be used as Exploratory Data Analysis to better understand what a dataset is about.

● True ○ False

Q1.3 (1 point) TF-IDF can be used for interrater reliability.

○ True ● False

Q1.4 (1 point) We can call the `join` method for tables with only one argument being passed in.

○ True ● False

Q1.5 (1 point) **Internal validity** is primarily concerned with how applicable the results of a study are to a broader population.

○ True ● False

Q1.6 (1 point) Aggregating data is more similar to grouping than disaggregating data.

● True ○ False

Q1.7 (1 point) It is easier to go from aggregated data to disaggregated data than the other way around.

○ True ● False

Q1.8 (1 point) Concepts are measurable, whereas variables are not.

○ True ● False

Q1.9 (1 point) The presence of confounding variables allows us to have confidence in the relationship between two variables.

○ True ● False

Q2 *Sliver or Cheeseboard (Sampling, Variables, and Table Methods)* **(15 points)**

Q2.1 (1 point) Maryam decides to team up with John to investigate the sentiment of current undergraduate Berkeley students around an age-old debate: Sliver or Cheeseboard?

To do so, they ask the registrar for a full roster of all enrolled UC Berkeley students, and randomly pick 100 students with equal probability. However, they make sure that they get 50 underclassmen and 50 upperclassmen as a part of their study.

What type of sampling method is this?

○ Simple random sampling

● Stratified random sampling

○ Convenience sampling

○ Quota sampling

○ Snowball sampling

> **Solution:** They sample randomly **within strata** (under/upperclassmen) with fixed counts per stratum → stratified sampling.

Here are the results of the study and a description of the columns:

Year	Preference	Confidence Level	Amount Spent
Underclassman	Cheeseboard	32	57.46
Upperclassman	Sliver	39	280.44
Underclassman	Sliver	49	151.92
Underclassman	Sliver	32	340.33
Underclassman	Sliver	4	128.46

Table 1: `data`

... (95 rows omitted)

- `Year`: A string representing whether the respondent is an underclassman or an upperclassman.
- `Preference`: That student's preference between Cheeseboard or Sliver.
- `Confidence Level`: How confident that student is about that preference (1 indicates only a slight preference, 50 indicates strong preference).
- `Amount Spent`: The amount (to the nearest cent) that student has spent on both Cheeseboard and Sliver to date.

Q2.2 (1 point) What is the variable type of `Year`?

● Categorical Ordinal ○ Quantitative Discrete

○ Categorical Nominal ○ Quantitative Continuous

Q2.3 (1 point) What is the variable type of `Confidence Level`?

● Categorical Ordinal ○ Categorical Nominal

○ Quantitative Discrete

○ Quantitative Continuous

Q2.4 (1 point) What is the variable type of `Preference`?

○ Categorical Ordinal ○ Quantitative Discrete

● Categorical Nominal ○ Quantitative Continuous

Q2.5 (1 point) What is the variable type of `Amount Spent`?

○ Categorical Ordinal ○ Quantitative Discrete

○ Categorical Nominal ● Quantitative Continuous

Q2.6 - Q2.8 (3 points) We want to figure out if underclassmen generally spent more money than upperclassmen. Write a line of code that can help us easily visualize the average amount of money that underclassmen spent on pizza compared to the amount that upperclassmen spent.

Year	Amount Spent mean
Underclassman	274.204
Upperclassman	299.463

```
data.group("Year", np.mean).select("Year", "Amount Spent mean")
       Q2.6        Q2.7            Q2.8
```

Q2.9 - Q2.13 (7 points) Let's add a **new column** to `data` called `Confidence Ordinal` that converts the confidence level into an ordinal variable, "High Confidence" or "Low Confidence". If a given respondent's confidence level is `>= 25`, the value should be **"High Confidence"**; otherwise, **"Low Confidence"**.

First, implement `to_ordinal(num)`, then add the column to `data` using the function.

```
def to_ordinal(num):

    if num >= 25:
         Q2.9

        return "High Confidence"
                  Q2.10

    return "Low Confidence"
               Q2.11


data = data.with_column("Confidence Ordinal",
               Q2.12
data.apply(to_ordinal, "Confidence Level"))
                  Q2.13
```

Q3 *Sliver and Cheeseboard (Visualizations)* **(16 points)**

Q3.1 (6 points) Let's visualize the distribution of how much students spent via a histogram. You should set the density parameter to `False` and set your own `bins` depending on the **minimum** and **maximum** value in that column. For example, if the `Amount Spent` column has a minimum value of 0 and a maximum value of 100, your bins should start at 0 and stop at 100 (inclusive on both ends), with a bin size of 20.

```
data.hist("Amount Spent",
                Q3.2
          bins =
np.arange(min(data.column("Amount Spent")), max(data.column("Amount Spent") + 20), 20),
                                                            Q3.3
          density=False)
               Q3.4
```

Q3.5 (1 point) `True` or `False`: Increasing the bin sizes will smoothen out the distribution, lessening the effect of outliers.

● True ○ False

Solution: Larger bins aggregate more values per bin, smoothing noise and visually dampening isolated outliers (at the cost of detail).

Q3.6 (2 points) In 20 words or less, briefly explain whether it would be more appropriate to use the "Confidence Level" or "Confidence Ordinal" column to create a horizontal bar chart.

Use Confidence Ordinal because bar charts summarize categorical counts where Confidence Level is numeric.

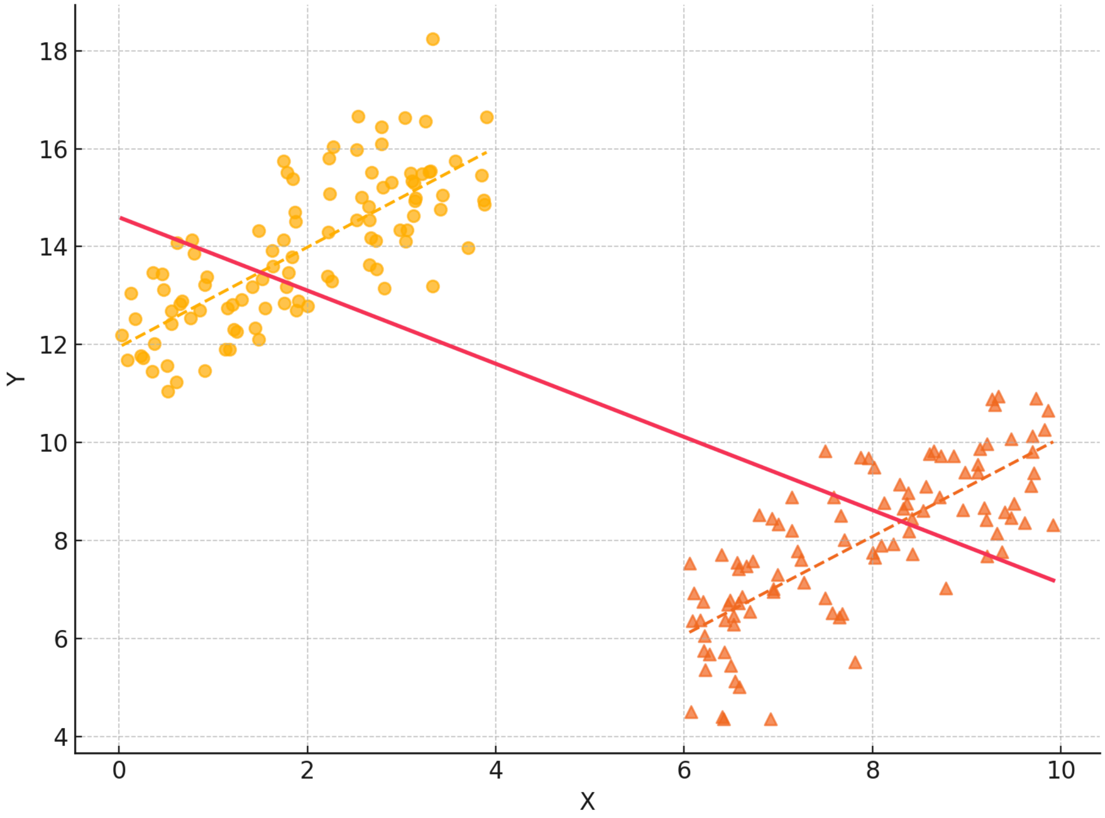
Q3.7 - Q3.8 (2 points) Fill in the following line of code to just get the number of rows where the Amount Spent are greater than or equal to 100 and less than 200.

```
data.where("Amount Spent", are.between(100, 200)).num_rows
                                  Q3.7               Q3.8
```

Q3.9 - Q3.11 (3 points) Finally, create a scatter plot visualizing the relationship between `Confidence Level` (on the x-axis) and `Amount  Spent` (on the y-axis), where each point gets a color depending on whether they are an **Underclassman** or **Upperclassman**.

```
data.scatter("Confidence Level", "Amount Spent", group = "Year")
                    Q3.9               Q3.10           Q3.11
```

Q3.12 (2 points) Suppose below is the image showing the relationship between the confidence level and amount spent. After overlaying the fit line, we conclude that there is a negative correlation between the Confidence Level and the Amount Spent. What phenomenon is this describing?



● Simpson's paradox

○ Overfitting

○ Ecological fallacy

○ Jittering

Solution: The aggregated trend reverses relative to subgroup trends → Simpson's paradox.

Q4 *Sliver and Cheeseboard (LLMs)* **(11 points)**

Q4.1 - Q4.4 (5 points) Fill in the implementation of `cumulative_sum`, which takes in a **numpy array** and returns the cumulative sum of that numpy array.

For example, `cumulative_sum(make_array(1, 3, 4))` should return a **numpy array** with the numbers `[1, 1+3, 1+3+4] = [1, 4, 8]`.

```
def cumulative_sum(arr):

    result = make_array()
                 Q4.1
    total = 0

    for elem in arr :
               Q4.2

        total += elem
                 Q4.3

        result = np.append(result, total)
                           Q4.4
    return result
```

Q4.5 (2 points) Suppose we decide to use a Large Language Model (our favorite!) to predict whether a given student will prefer Cheeseboard or Sliver. Liberty decides to first measure their level of agreement with an interrater reliability score.

	(LLM) Cheeseboard	(LLM) Sliver
(Liberty) Cheeseboard	10	60
(Liberty) Sliver	30	40

Suppose the Cohen's Kappa coefficient is −0.2857. This means that the level of agreement was...

○ Better than expected by chance

○ No better than randomly assigning

● Worse than expected by chance

○ Perfect agreement

Q4.6 - Q4.7 (4 points) Turns out, Liberty's predictions were exactly correct! Calculate the precision and recall for the LLM, treating Cheeseboard as the positive samples. You may leave your answer as an unsimplified fraction.

```
precision = 10 / 40
              Q4.6
recall = 10 / 70
           Q4.7
f1_score = (2 * precision * recall) / (precision + recall)
```

Q5 *Unique Usernames* (10 points)

Q5.1 - Q5.10 (10 points) Implement `to_username`, which takes in a **numpy array** of `names` (formatted as "FirstName LastName"), and returns a **dictionary** mapping each name to its corresponding username.

We will define a username as the combination of a person's first initial and last name, followed by the length of their last name.

For example,

Andrew Chen becomes AChen4

Earn Maneenop becomes EManeenop8

Jedi Tsang becomes JTsang5

Suppose that a given username is already taken. We will then increment the number in the username until we find a unique username.

For example, if we had the user Amy Chen (which would become AChen4), the username would then be AChen5.

In other words, `to_username(make_array("Andrew Chen", "Earn Maneenop", "Jedi Tsang", "Amy Chen"))` should return a **dictionary** with the following key-value pairings:

`{"Andrew Chen": "AChen4", "Earn Maneenop": "EManeenop8", "Jedi Tsang": "JTsang5", "Amy Chen": "AChen5"}`

```
def to_username(names):

    result =  {}
              Q5.1
    for name in names:
                 Q5.2
        first_name = name.split()[0][0]
                             Q5.3
        last_name = name.split()[1]
                          Q5.4
        last_number = len(last_name)
                           Q5.5
        proposed_username = first_name + last_name + str(last_number)
                                                           Q5.6
        while proposed_username in result.values():
         Q5.7
            last_number += 1

            proposed_username = first_name + last_name + str(last_number)
                                                   Q5.8
        result[name] = proposed_username
                      Q5.9
    return result
        Q5.10
```

Q6 *Just for fun!* **(0 points)**

Q6.1 Draw something fun, or write a message for the staff!

Everyone had cool drawings/messages! :D