Midterm Exam - DSC 80, Fall 2023

Instructions:

- This exam consists of 5 questions. A total of 100 points are available.
- Write name in the top right of each page in the space provided.
- Please write neatly in the provided answer boxes. We will not grade work that appears elsewhere.
- Completely fill in bubbles and square boxes.
  ◯ A bubble means that you should only **select one choice**.
  ☐ A square box means you should **select all that apply**.
- You may refer to one 8.5" × 11" sheet of notes of your own creation. No other resources or technology (including calculators) are permitted.
- Do not turn the page until instructed to do so.

Last name	
First name	
Student ID number	
UCSD email	
Name of the person to your left	
Name of the person to your right	
All the work on this exam is my own. **(please sign)**	

Name: _____

This page is intentionally left blank, but feel free to use it as scratch paper.

Name: _____

Question 1 ..... *24 points*

Fill in Python code below so that the last line of each part evaluates to each desired result, assuming that the following tables are both stored as Pandas DataFrames. **You may not use `for` or `while` loops in any answer for this question. Only the first few rows are shown for each table.**

The `df` table (left) records what people ate in kilograms (kg) on each date in 2023. For example, the first row records that Sam ate 0.2 kg of Ribeye on Jan 1, 2023. The `foods` table (right) records the carbon dioxide (CO_2) emissions it takes to produce each kind of food. For example, the first row in the `foods` table shows that growing 1 kg of mung beans produces 0.1 kg of CO_2.

	date	name	food	weight
0	2023-01-01	Sam	Ribeye	0.20
1	2023-01-01	Sam	Pinto beans	0.10
2	2023-01-01	Lauren	Mung beans	0.25
3	2023-01-02	Lauren	Lima beans	0.30
4	2023-01-02	Sam	Sirloin	0.30

name	co2/kg
Mung beans	0.1
Fava beans	2.5
NY strip	100.0
Sirloin	80.0

(a) (3 points) Find the total kg of food eaten for each day and each person in `df` as a Series.

```
df.groupby(['date', 'name'])['weight'].sum()
```

(b) (3 points) Find all the rows in `df` where Tina was the person eating.

```
df.loc[df['name'] == 'Tina']
```

(c) (5 points) Find all the unique people who **did not** eat any food containing the word "beans".

```
def foo(x):
    return not x['food'].str.contains('beans').any()

df.groupby('name').filter(foo)['name'].unique()
```

(d) (5 points) Create a copy of `df` that has one extra column called `words` that contains the number of words for each value in the `food` column. Assume that words are separated by one space character. For example, "Pinto beans" has two words.

```
def f(x):
    return len(x.split())

df.assign(words=df['food'].apply(f))
```

(e) (8 points) Find the total kg of CO_2 produced by each person in `df`. If a food in `df` doesn't have a matching value in `foods`, assume that the food generates 100 kg of CO_2 per kg of food.

```
df2 = df.merge(foods, left_on='food', right_index=True, how='left')

(df2.assign(c=df2['weight'] * df2['co2/kg'].fillna(100))

.groupby('name')['c'].sum())
```

Name: _____

Question 2 ..... ***16 points***

For this question, we'll continue using the `df` and `foods` tables from Question 1. Dylan and Giorgia want to compare their CO_2 emissions. They added a new column called `bean` to `df` that containing `True` if the food was a bean (e.g. "Pinto beans") and `False` otherwise. Then, they compute the following pivot table:

	Dylan's co2/kg	Giorgia's co2/kg
bean=True	5	10
bean=False	50	80

Each entry in the pivot table is the average CO_2 emissions for Dylan and Giorgia per kg of food they ate (CO_2/kg) for both bean and non-bean foods.

(a) (8 points) Suppose that overall, Dylan produced an average of 41 CO_2/kg of food he ate, while Giorgia produced an average of 38 CO_2/kg. Determine whether each statement is definitely true (T), definitely false (F), or whether more information is needed (M) beyond this information and the pivot table above.

◯ **T** ◯ F ◯ M This is an example of Simpson's Paradox.

◯ T ◯ F ◯ **M** Dylan ate at least as many kg of bean foods compared to Giorgia.

◯ **T** ◯ F ◯ M Giorgia ate a higher proportion of bean foods than Dylan.

◯ T ◯ F ◯ **M** Dylan emitted more kg of CO_2 than Giorgia overall.

(b) (8 points) Dylan and Giorgia want to figure out exactly when Simpson's paradox occurs for their data. Suppose that 0.2 proportion of Dylan's food was bean foods. What range of proportions for Giorgia's bean food would cause Simpson's paradox to occur?

Show your work in the space below, **then write your final answer in the blanks at the bottom of the page**. Your final answers should be between 0 and 1. Leave your answers as simplified fractions.

Solution: Dylan's overall CO_2 usage is:

$$0.2 \times 5 + (1 - 0.2) \times 50 = 41$$

For Simpson's paradox to occur, Giorgia's overall CO_2 usage must be less than Dylan's. If Giorgia's proportion of bean foods is p, we have:

$$\begin{aligned} p \times 10 + (1 - p) \times 80 &< 41 \\ 80 - 70p &< 41 \\ p &> \frac{39}{70} \end{aligned}$$

Since p cannot be greater than 1, our final answer is:

Between $\frac{39}{70}$ and 1.

Name: _____

Question 3 ..... ***20 points***

The `donkeys` table contains data from a research study about donkey health. The researchers measured the attributes of 544 donkeys. The next day, they selected 30 donkeys to reweigh. The first few rows of `donkeys` table are shown below (left), and the table contains the following columns (right):

	id	BCS	Age	Weight	WeightAlt
0	d01	3.0	<2	77	NaN
1	d02	2.5	<2	100	NaN
2	d03	1.5	<2	74	NaN

`id`	A unique identifier for each donkey (`d01`, `d02`, etc.).
`BCS`	Body condition score: from 1 (emaciated) to 3 (healthy) to 5 (obese) in increments of 0.5.
`Age`	Age in years: <2, 2–5, 5–10, 10–15, 15–20, and over 20 years.
`Weight`	Weight in kilograms.
`WeightAlt`	Second weight measurement taken for 30 donkeys. NaN if the donkey was not reweighed.

(a) (10 points) What is the feature type of each column in `donkeys`?

`id`: ◯ Discrete continuous ◯ Continuous ◯ Ordinal ◯ **Nominal**

`BCS`: ◯ Discrete continuous ◯ Continuous ◯ **Ordinal** ◯ Nominal

`Age`: ◯ Discrete continuous ◯ Continuous ◯ **Ordinal** ◯ Nominal

`Weight`: ◯ Discrete continuous ◯ **Continuous** ◯ Ordinal ◯ Nominal

`WeightAlt`: ◯ Discrete continuous ◯ **Continuous** ◯ Ordinal ◯ Nominal

(b) (10 points) Consider the following scenarios for how the researchers chose the 30 donkeys to reweigh. Select the correct missingness mechanism for the `WeightAlt` column in each scenario[1].

The researchers chose the 30 donkeys with the largest `Weight` values to reweigh. ◯ NMAR ◯ **MAR** ◯ MCAR

The researchers drew 30 donkeys uniformly at random without replacement from the donkeys with `BCS` scores of 4 or greater. ◯ NMAR ◯ **MAR** ◯ MCAR

The researchers set `i` as a number drawn uniformly at random between 0 and 514, then reweighed the donkeys in `donkeys.iloc[i:i+30]`. ◯ NMAR ◯ MAR ◯ **MCAR**

The researchers reweighed all the donkeys, but deleted all the values in `WeightAlt` except for the 30 lowest values. ◯ **NMAR** ◯ MAR ◯ MCAR

The researchers split up the donkeys into the 6 different age groups, then sampled 5 donkeys uniformly at random without replacement within each age group. ◯ NMAR ◯ **MAR** ◯ **MCAR** **(Both MAR and MCAR were marked correct for this question.)**

[1] Although the missing data are missing by design from the perspective of the original researchers, since we can't directly recover the missing values from our other data, we can treat the missing data as NMAR, MAR, or MCAR.

Name: _____

Question 4 ..... ***40 points***

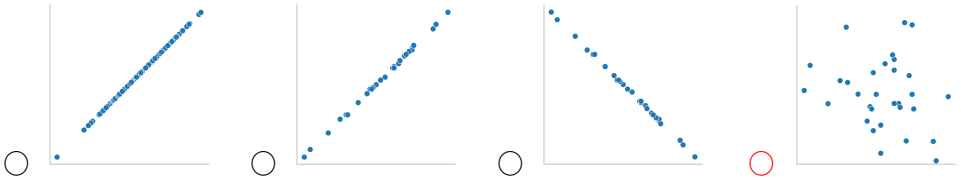
In this question, we will continue to work with the `donkeys` dataset from Question 3. The first few rows of the table and column descriptions are shown below for convenience.

	id	BCS	Age	Weight	WeightAlt
0	d01	3.0	<2	77	NaN
1	d02	2.5	<2	100	NaN
2	d03	1.5	<2	74	NaN

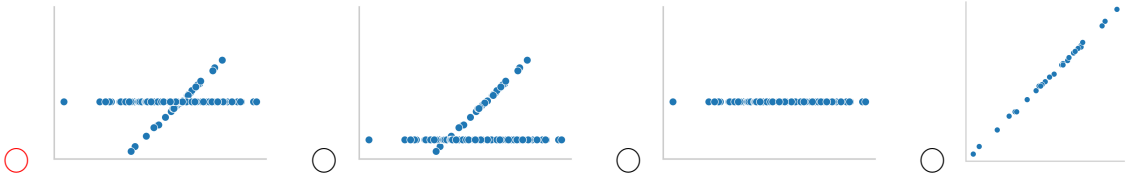
`id`	A unique identifier for each donkey (`d01`, `d02`, etc.).
`BCS`	Body condition score: from 1 (emaciated) to 3 (healthy) to 5 (obese) in increments of 0.5.
`Age`	Age in years: <2, 2–5, 5–10, 10–15, 15–20, and over 20 years.
`Weight`	Weight in kilograms.
`WeightAlt`	Second weight measurement taken for 30 donkeys. NaN if the donkey was not reweighed.

For this question, assume that the researchers chose the 30 donkeys to reweigh by drawing **a simple random sample of 30 underweight donkeys: donkeys with `BCS` values of 1, 1.5, or 2.** The researchers weighed these 30 donkeys one day later and stored the results in the `WeightAlt` column.

(a) (3 points) Which of the following shows the scatter plot of `WeightAlt - Weight` on the y-axis and `Weight` on the x-axis? Assume that missing values are not plotted.



(b) (4 points) Suppose we use mean imputation to fill in the missing values in `WeightAlt`. Select the scatter plot of `WeightAlt` on `Weight` after imputation.



(c) (12 points) Alan wants to see whether donkeys with `BCS >= 3` have larger `Weight` values on average compared to donkeys that with `BCS < 3`. Select **all the possible test statistics** that Alan could use to conduct this hypothesis test. Let μ_1 be the mean weight of donkeys with `BCS >= 3` and μ_2 be the mean weight of donkeys with `BCS < 3`.

- ☐ μ_1
- ■ $\mu_1 - \mu_2$
- ■ $2\mu_2 - \mu_1$
- ☐ $|\mu_1 - \mu_2|$
- ☐ Total variation distance
- ☐ Kolmogorov–Smirnov test statistic

(d) (4 points) To generate a single sample under his null hypothesis, Alan should:

- ◯ Resample 744 donkeys with replacement from `donkeys`.
- ◯ Resample 372 donkeys with replacement from `donkeys` with `BCS < 3`, and another 372 donkeys with `BCS >= 3`.
- ◯ **Randomly permute the `Weight` column.**

Name: _____

(e) (17 points) Doris wants to use multiple imputation to fill in the missing values in `WeightAlt`. She knows that `WeightAlt` is MAR conditional on `BCS` and `Age`, so she will perform multiple imputation conditional on `BCS` and `Age` – each missing value will be filled in with values from a random `WeightAlt` value **from a donkey with the same `BCS` and `Age`**. Assume that all `BCS` and `Age` combinations have observed `WeightAlt` values.

Fill in the blanks in the code below to estimate the median of `WeightAlt` using multiple imputation conditional on `BCS` and `Age` with 100 repetitions. A function `impute` is also partially filled in for you, and you should use it in your answer.

```
def impute(col):
    col = col.copy()

    n = _____col.isna().sum()_____

    fill = np.random.choice(_____col.dropna(), n_____)

    col[_____col.isna()_____] = fill
    return col

results = []

for i in range(_____100_____):
    imputed = (donkeys

    ._____groupby_____(_____['BCS', 'Age']_____)

    ['WeightAlt']

    ._____transform_____(_____impute_____)
    )
    results.append(imputed.median())
```

Name: _____

Question 5 ..... ***0 points***

Optional: Draw a Picture About UCSD Data Science (or use this page for scratch work)