Midterm Exam - DSC 10, Fall 2024

Full Name:

PID:

Exam Time: ◯ A (9AM) ◯ B (10AM) ◯ C (1PM)

Instructions:

- This exam consists of 12 questions, worth a total of 83 points, plus one extra credit question worth 3 points.
- Write your PID in the top right corner of each page in the space provided.
- Please write **clearly** in the provided answer boxes; we will not grade work that appears elsewhere. Completely fill in bubbles and square boxes; if we cannot tell which option(s) you selected, you may lose points.

  ◯ A bubble means that you should only **select one choice**.

  ☐ A square box means you should **select all that apply**.
- For full credit, your solutions must use methods of the course.
- You may use one page of double-sided handwritten notes. Aside from this, you may not refer to any other resources or technology during the exam. No calculators!

By signing below, you are agreeing that you will behave honestly and fairly during and after this exam.

Signature:

Version A

Please do not open your exam until instructed to do so.

Important: Before proceeding, make sure to rip off the last page of this exam packet and read the data description.

Question 1 (3 pts)

Which of the following columns would be an appropriate index for the `treat` DataFrame?

◯ `"address"` ◯ `"candy"` ◯ `"neighborhood"` ◯ None of these.

Question 2 (4 pts)

Which of the following expressions evaluate to `"M&M"`? Select all that apply.

☐ `treat.get("candy").iloc[1]`

☐ `treat.sort_values(by="candy", ascending = False).get("candy").iloc[1]`

☐ `treat.sort_values(by="candy", ascending = False).get("candy").loc[1]`

☐ `treat.set_index("candy").index[-1]`

☐ None of these.

Question 3 (6 pts)

Consider the code below.

```
street = treats.get("address").str.contains("Street")
sour = treats.get("candy").str.contains("Sour")
```

a) (3 pts) What is the data type of `street`?

◯ `int` ◯ `bool` ◯ `str` ◯ Series ◯ DataFrame

b) (3 pts) What does the following expression evaluate to? Write your answer **exactly** how the output would appear in Python.

```
np.count_nonzero(street & sour) > sour.sum()
```

PID: _____

Question 4 (9 pts)

The `"address"` column contains quite a bit of information. All houses are in `"San Diego, CA"`, but the street address and the zip code vary. Note that the "street address" includes both the house number and street name, such as `"820 Opal Street"`. All addresses are formatted in the same way, for example, `"820 Opal Street, San Diego, CA, 92109"`.

a) (6 pts) Fill in the blanks in the function `address_part` below. The function has two inputs: a value in the index of `treat` and a string `part`, which is either `"street"` or `"zip"`. The function should return the appropriate part of the address at the given index value, as a string. Example behavior is given below.

```
>>> address_part(4, "street")
"8575 Jade Coast Drive"

>>> address_part(1, "zip")
"92109"
```

The function already has a return statement included. You should not add the word `return` anywhere else!

```
def address_part(index_value, part):
    if part == "street":
        var = 0
    else:
        ___(a)___
    return treat.get("address").loc[___(b)___].___(c)___
```

(a): _____

(b): _____

(c): _____

b) (3 pts) Suppose we had a different function called `zip_as_int` that took as input a single address, formatted exactly as the addresses in `treat`, and returned the zip code as an `int`. Write a Python expression using the `zip_as_int` function that evaluates to a Series with the zip codes of all the addresses in `treat`.

_____

Question 5 (5 pts)

Write a Python expression that evaluates to the address of the house with the most pieces of candy available (the most **pieces**, not the most varieties).

It's okay if you need to write on multiple lines, but your code should represent a single expression in Python.

Question 6 (9 pts)

Suppose you visit a house that has 40 Twix, 50 M&Ms, and 10 KitKats in a bowl. You take three pieces of candy from this bowl.

a) (3 pts) What is the probability you get all Twix?

- ◯ $\frac{40}{100} \cdot \frac{39}{100} \cdot \frac{38}{100}$
- ◯ $\frac{40}{100} \cdot \frac{40}{99} \cdot \frac{40}{98}$
- ◯ $\frac{40}{100} \cdot \frac{40}{100} \cdot \frac{40}{100}$
- ◯ $\frac{40}{100} \cdot \frac{39}{99} \cdot \frac{38}{98}$

b) (3 pts) What is the probability you get no Twix? Leave your answer **completely unsimplified**, similar to the answer choices for part (a).

c) (3 pts) Let a be your answer to part (a) and let b be your answer to part (b). Write a mathematical expression in terms of a and/or b that evaluates to the probability of getting some Twix and some non-Twix candy from this house.

PID: _____

Question 7 (9 pts)

Suppose you visit another house and their candy bowl is composed of 2 Twix, 3 Rolos, 1 Snickers, 3 M&Ms, and 1 KitKat. You do the same as before and take 3 candies from the bowl at random.

Fill in the blanks in the code below so that `prob_all_same` evaluates to an estimate of the probability that you get three of the same type of candy.

```
candy_bowl = np.array(["Twix", "Twix", "Rolo", "Rolo", "Rolo",
"Snickers", "M&M", "M&M", "M&M", "KitKat"]

repetitions = 10000
prob_all_same = 0
for i in np.arange(repetitions):
    grab = np.random.choice(___(a)___)
    if ___(b)___:
        prob_all_same = prob_all_same + 1
prob_all_same = ___(c)___
```

a) (3 pts) What goes in blank (a)?

- ◯ `candy_bowl, len(candy_bowl), replace=False`
- ◯ `candy_bowl, 3, replace=False`
- ◯ `candy_bowl, 3, replace=True`
- ◯ `candy_bowl, repetitions, replace=True`

b) (3 pts) What goes in blank (b)?

- ◯ `grab[0] == "Rolo" and grab[1] == "Rolo" and grab[2] == "Rolo"`
- ◯ `grab[0] == grab[1] and grab[0] == grab[2]`
- ◯ `grab[0] == grab[1] or grab[0] == grab[2]`
- ◯ `grab == "Rolo" | grab == "M&M"`

c) (3 pts) What goes in blank (c)?

- ◯ `prob_all_same.mean()`
- ◯ `prob_all_same/len(candy_bowl)`
- ◯ `prob_all_same/repetitions`
- ◯ `prob_all_same/3`

Question 8 (4 pts)

Select the correct way to fill in the blank such that the code below evaluates to `True`.

```
treat.groupby(______).mean().shape[0] == treat.shape[0]
```

◯ `"address"` ◯ `["address", "candy"]`

◯ `"candy"` ◯ `["candy", "neighborhood"]`

◯ `"neighborhood"` ◯ `["address", "neighborhood"]`

Question 9 (4 pts)

Assume that all houses in `treat` give out the same size candy, say fun-sized. Suppose we have an additional DataFrame, `trick`, which is indexed by `"candy"` and has one column, `"price"`, containing the cost in dollars of a **single piece** of fun-sized candy, as a `float`.

Suppose that:

- `treat` has 200 rows total, and includes 15 distinct types of candies.
- `trick` has 25 rows total: 15 for the candies that appear in `treat`, plus 10 additional rows that correspond to candies not represented in `treat`.

Consider the following line of code:

```
trick_or_treat = trick.merge(treat, left_index = True, right_on = "candy")
```

How many rows does `trick_or_treat` have?

◯ 15 ◯ 25 ◯ 200 ◯ 215 ◯ 225 ◯ 3000 ◯ 5000

PID: _____

Question 10 (14 pts)

Recall from the last problem that the DataFrame `trick_or_treat` includes a column called `"price"` with the cost in dollars of a **single piece** of fun-sized candy, as a `float`.

Assume we have run the line of code `tot = trick_or_treat` to reassign `trick_or_treat` to the shorter variable name `tot`.

In this problem, we'll use `tot` to calculate the total amount of money that each house spent on Halloween candy. This number is always less than \$80 for the houses in our data set.

a) (5 pts) Fill in the blanks below so that the following block of code plots a histogram that displays the distribution of the total amount of money that houses spent on Halloween candy, in dollars.

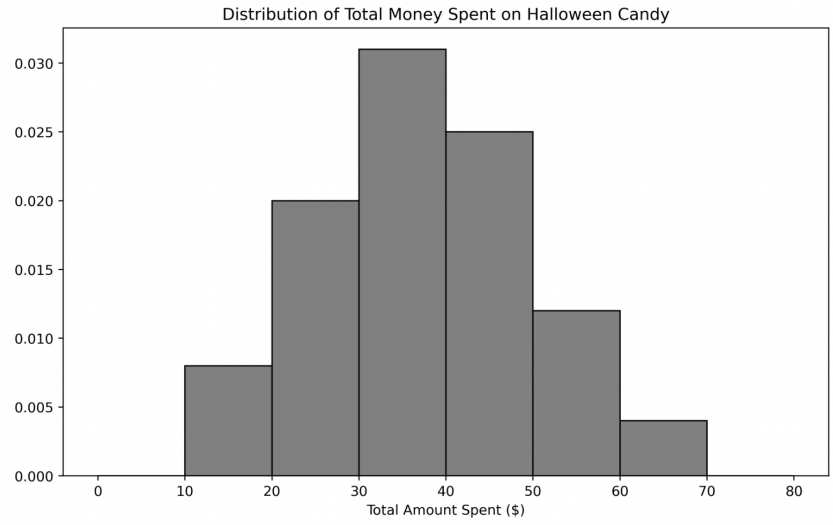
```
total = (tot.assign(total_spent = ___(a)___)
            .groupby(___(b)___).___(c)___)
total.plot(kind = "hist",  y = "total_spent", density = True,
           bins = np.arange(0, 90, 10))
```

(a):

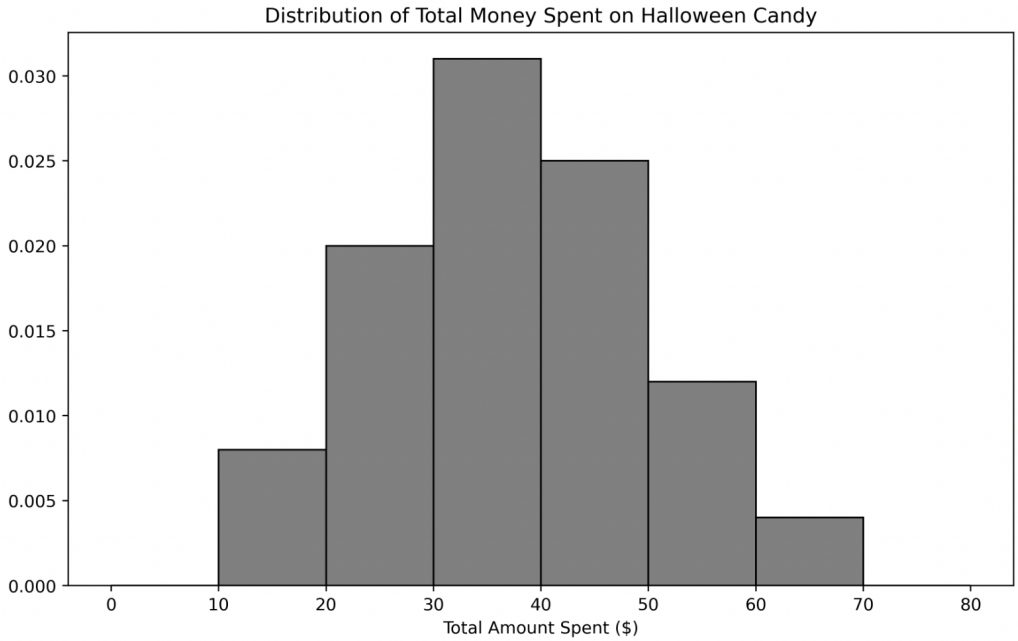
(b):

(c):

The histogram below displays the distribution of the total amount of money that houses spent on Halloween candy; it is the histogram that would be generated from the code snippet above, assuming the blanks were filled in correctly.



The histogram from the previous page is included again below for your reference.



b) (3 pts) Which two adjacent bins in the histogram represent about 50% of the houses?

◯ `[10, 20)` and `[20, 30)`

◯ `[20, 30)` and `[30, 40)`

◯ `[30, 40)` and `[40, 50)`

◯ `[40, 50)` and `[50, 60)`

◯ `[50, 60)` and `[60, 70)`

◯ Not possible to determine.

c) (3 pts) Suppose we create a new histogram, using the same code as above but with `bins = np.arange(0, 90, 20)` instead of `bins = np.arange(0, 90, 10)`. In the box below, approximate the height of the tallest bar in this new histogram. If this is impossible, leave the box blank and instead fill in the bubble for "Not possible to determine."

height = ☐ ◯ Not possible to determine.

d) (3 pts) Suppose we create a new histogram, using the same code as above but substituting `bins = np.arange(0, 90, 5)` for `bins = np.arange(0, 90, 10)`. In the box below, approximate the height of the tallest bar in this new histogram. If this is impossible, leave the box blank and instead fill in the bubble for "Not possible to determine."

height = ☐ ◯ Not possible to determine.

PID: _____

Question 11 (10 pts)

As in the last problem, we'll continue working with the `tot` DataFrame that came from merging `trick` with `treat`. The `"price"` column contains the cost in dollars of a **single piece** of fun-sized candy, as a `float`.

In this problem, we want to use `tot` to calculate the **average cost per piece** of Halloween candy at each house. For example, suppose one house has 30 Twix, which cost \$0.20 each, and 20 Laffy Taffy, which cost \$0.10 each. Then this house spent \$8.00 on 50 pieces of candy, for an average cost of \$0.16 per piece.

Which of the following correctly sets `ac` to a DataFrame indexed by `"address"` with a column called `"avg_cost"` that contains the average cost per piece of Halloween candy at each address? Select all that apply.

Way 1:

```
ac = tot.groupby("address").sum()
ac = ac.assign(avg_cost = ac.get("price") /
                          ac.get("how_many")).get(["avg_cost"])
```

Way 2:

```
ac = tot.assign(x = tot.get("price") / tot.get("how_many"))
ac = ac.groupby("address").sum()
ac = ac.assign(avg_cost = ac.get("x").mean()).get(["avg_cost"])
```

Way 3:

```
ac = tot.assign(x = tot.get("price") / tot.get("how_many"))
ac = ac.groupby("address").sum()
ac = ac.assign(avg_cost = ac.get("x") /
                          ac.get("how_many")).get(["avg_cost"])
```

Way 4:

```
ac = tot.assign(x = tot.get("how_many") * tot.get("price"))
ac = ac.groupby("address").sum()
ac = ac.assign(avg_cost = ac.get("x").mean()).get(["avg_cost"])
```

Way 5:

```
ac = tot.assign(x = tot.get("how_many") * tot.get("price"))
ac = ac.groupby("address").sum()
ac = ac.assign(avg_cost = ac.get("x") /
                          ac.get("how_many")).get(["avg_cost"])
```

☐ Way 1 ☐ Way 2 ☐ Way 3 ☐ Way 4 ☐ Way 5

Question 12 (6 pts)

a) (3 pts) What would be the best type of plot to visualize the distribution of `"neighborhood"` among the houses represented in `treat`?

◯ scatter plot ◯ line plot ◯ bar chart ◯ histogram

b) (3 pts) Suppose we had access to historical data about the price of fun-sized candies over time. If we wanted to compare the prices of Milky Way and Skittles over time, which would be the best type of visualization to plot?

◯ overlaid scatter plot ◯ overlaid line plot

◯ overlaid bar chart ◯ overlaid histogram

Question 13 (Extra Credit - 3 points)

Define the variable `double` as follows.

```
double = treat.groupby("candy").count().groupby("address").count()
```

Now, suppose you know that

`double.loc[1].get("how_many")` evaluates to `5`.

Which of the following is a valid interpretation of this information? Select all that apply.

☐ There are five houses that are each giving out only one type of candy.

☐ There are five types of candy that are each being given out by only one house.

☐ There is only one house that is giving out five types of candy.

☐ There is only one type of candy that is being given out by five houses.

☐ None of these.

Grading Note: Extra credit will be graded as all-or-nothing. You need the exact set of correct responses to earn the 3 points.

Trick or Treat

Trick-or-treating is a Halloween tradition, where children wear costumes and walk around their neighborhood from house to house to collect candy. In this exam, you'll work with a data set representing the candy given out on Halloween. Each row represents one type of candy given out by one house in San Diego.

The columns of `treat` are as follows:

- `"address"` (`str`): The address of the house giving out candy.
- `"candy"` (`str`): The type of candy that is being given out.
- `"how_many"` (`int`): How many pieces of candy are being given out.
- `"neighborhood"` (`str`): The neighborhood that the house is in.

The first few rows of `treat` are shown below, though `treat` has many more rows than pictured.

	address	candy	how_many	neighborhood
0	820 Opal Street, San Diego, CA, 92109	Twix	40	Pacific Beach
1	820 Opal Street, San Diego, CA, 92109	M&M	50	Pacific Beach
2	820 Opal Street, San Diego, CA, 92109	KitKat	10	Pacific Beach
3	4011 Arizona Street, San Diego, CA, 92104	M&M	30	North Park
4	8575 Jade Coast Drive, San Diego, CA, 92126	Skittles	25	Mira Mesa

Throughout this exam, we will refer to `treat` repeatedly.

Assume that we have already run `import babypandas as bpd` and `import numpy as np`.