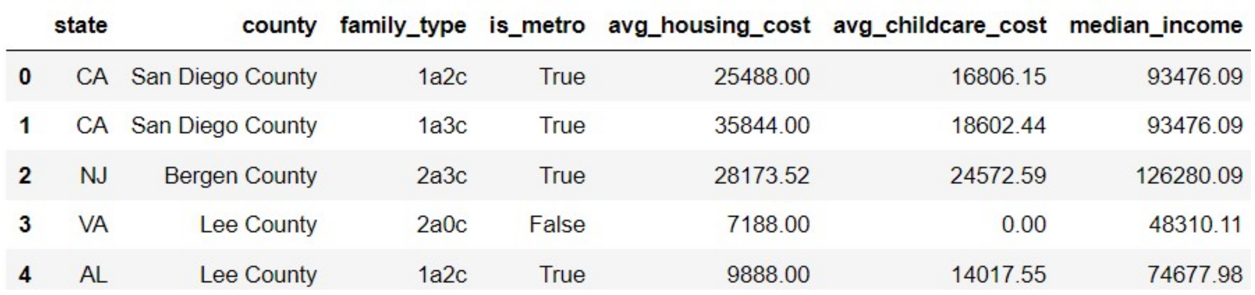
Data Overview: Cost of Living

An area's cost of living describes how expensive it is to live in that area. In this exam, we'll work with the DataFrame `living_cost`, which describes the typical cost of living for different types of families across all 3143 counties in the 50 United States. The first few rows of `living_cost` are shown below, but `living_cost` has **many more rows** than shown.



	state	county	family_type	is_metro	avg_housing_cost	avg_childcare_cost	median_income
0	CA	San Diego County	1a2c	True	25488.00	16806.15	93476.09
1	CA	San Diego County	1a3c	True	35844.00	18602.44	93476.09
2	NJ	Bergen County	2a3c	True	28173.52	24572.59	126280.09
3	VA	Lee County	2a0c	False	7188.00	0.00	48310.11
4	AL	Lee County	1a2c	True	9888.00	14017.55	74677.98

Two counties in the same state will never have the same name, but as rows 4 and 5 above illustrate, there are some counties in different states with the same name, like Lee County.

The `"family_type"` column uses a code to describe the number of adults and children in a family. For example, a value of `"2a1c"` represents families with two adults and one child. There are ten unique values, as follows: `"1a0c"`, `"1a1c"`, `"1a2c"`, `"1a3c"`, `"1a4c"`, `"2a0c"`, `"2a1c"`, `"2a2c"`, `"2a3c"`, `"2a4c"`. We will assume that **all families fall into one of these ten categories**, and **all ten family structures are present in each US county**.

Each of the 31430 rows of the DataFrame represents a unique combination of `"state"`, `"county"`, and `"family_type"`. As a result, there will be more than one row with a `"state"` of `"CA"` and a `"county"` of `"San Diego"`, corresponding to different values of `"family_type"`. Similarly, there will be many rows such that `"family_type"` is `"2a1c"`, all corresponding to different counties. There is **only one row**, however, where `"state"` is `"CA"`, `"county"` is `"San Diego"`, and `"family_type"` is `"1a2c"`

In addition to the `"state"`, `"county"`, and `"family_type"` columns, `living_cost` includes the following columns.

- `"is_metro" (bool)`: `True` if the county is part of a metropolitan (urban) area, `False` otherwise. This value is the same for all rows of the DataFrame corresponding to the same county and state.
- `"avg_housing_cost" (int)`: The average yearly cost of housing, in dollars, for families of the given size in the given county and state.
- `"avg_childcare_cost" (int)`: The average yearly cost of childcare, in dollars, for families of the given size in the given county and state.
- `"median_income" (int)`: The median annual income, in dollars, for families of the given size in the given county and state.

Throughout the exam, assume we have already run `import babypandas as bpd` and `import numpy as np`.

Midterm Exam - DSC 10, Fall 2023

Full Name:	Solutions
PID:	A12345678

Lecture: ◯ A (10AM) ◯ B (9AM) ◯ C (1PM) ◯ D (8AM)

Instructions:

- This exam consists of 8 questions, worth a total of 80 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**.
- You may refer to the Reference Sheet that was provided to you. Other than that, you may not refer to any other resources or technology during the exam (no notes, 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.

Question 1 (14 pts)

a) (2 pts) You're interested in comparing the "`avg_housing_cost`" across different "`family_type`" groups for San Diego County, CA specifically. Which type of visualization would be most appropriate?

- ○ Scatter plot
- ○ Line plot
- [✓ Bar chart]
- ○ Histogram

b) (3 pts) Suppose we run the three lines of code below.

```
families = living_cost.groupby("family_type").median()
sorted_families = families.sort_values(by="avg_housing_cost")
result = sorted_families.get("avg_childcare_cost").iloc[0]
```

Which of the following does `result` evaluate to?

- [✓ The median "`avg_childcare_cost`" of the "`family_type`" with the lowest median "`avg_housing_cost`".]
- ○ The median "`avg_childcare_cost`" of the "`family_type`" with the highest median "`avg_housing_cost`".
- ○ The median "`avg_housing_cost`" of the "`family_type`" with the lowest median "`avg_childcare_cost`".
- ○ The median "`avg_housing_cost`" of the "`family_type`" with the highest median "`avg_childcare_cost`".

c) (3 pts) Suppose we define `another_result` as follows.

```
another_result = (living_cost.groupby("state").count()
                  .sort_values(by="median_income", ascending=False)
                  .get("median_income").index[0])
```

What does `another_result` represent?

- ○ The state with the highest median income.
- ○ The median income in the state with the highest median income.
- [✓ The state with the most counties.]
- ○ The median income in the state with the most counties.

d) (3 pts) Which of the following DataFrames has exactly four columns?

- ○ `living_cost.groupby("family_type").min()`
- [✓ ● `living_cost.groupby("family_type").sum()`]
- ○ `living_cost.groupby("family_type").count()`
- ○ None of the above.

e) (3 pts) Suppose we define the Series `three_columns` to be the concatenation of three columns of the `living_cost` DataFrame as follows.

```
three_columns = (living_cost.get("state") + " " +
                 living_cost.get("county") + " " +
                 living_cost.get("family_type"))
```

For example, the first element of `three_columns` is the string `"CA San Diego County 1a2c"` (refer back to the first row of `living_cost` provided in the data overview). What does the following expression evaluate to?

```
(living_cost.assign(geo_family=three_columns)
            .groupby("geo_family").count()
            .shape[0])
```

- ○ 10, the number of distinct `"family_type"` values.
- ○ 50, the number of states in the US.
- ○ 500, the number of combinations of states in the US and `"family_type"` values.
- ○ 3143, the number of counties in the US.
- [✓ ● 31430, the number of rows in the `living_cost` DataFrame.]

Question 2 (8 pts)

Suppose we define the three variables below.

```
J = living_cost.get("county") == "Benton County"
K = living_cost.get("state") == "IN"
L = living_cost.get("family_type") == "1a2c"
```

Feel free to use these variables in your solutions to the following questions.

a) (4 pts) Fill in the blanks so that the expression below evaluates to the average yearly childcare cost for families with one adult and two children in Benton County, IN.

```
__(a)__.__(b)__.iloc[0]
```

(i) What goes in blank (a)?

Solution: `living_cost[J & K & L]`

(ii) What goes in blank (b)?

Solution: `get("avg_childcare_cost")`

b) (4 pts) Fill in the blanks so that the expression below evaluates to the number of states with a county named Benton County.

```
__(c)__.__(d)__ / 10
```

(i) What goes in blank (c)?

Solution: `living_cost[J]`

(ii) What goes in blank (d)?

Solution: `shape[0]`

PID: ____________________

Question 3 (12 pts)

Suppose we want to assign a new column named `"family_size"` to `living_cost` that contains the total number of people in each family, stored as an int. We do so as follows.

```
living_cost = living_cost.assign(
              family_size=living_cost.get("family_type").apply(num_people))
```

Which of the following options correctly define the function `num_people` such that the line above adds the `"family_size"` column as desired? **Select all that apply.**

Hint: You can access an individual character in a string using the position number in square brackets. For example, `"midterm"[0]` evaluates to `"m"` and `"midterm"[1]` evaluates to `"i"`.

```
# Option 1
def num_people(fam):
    return int(fam[0]) + int(fam[2])
------------------------------------
# Option 2
def num_people(fam):
    return int(fam[0] + fam[2])
------------------------------------
# Option 3
def num_people(fam):
    x = int(fam[0] + fam[2])
    return int(x / 10) + x % 10
------------------------------------
# Option 4
def num_people(fam):
    x = fam.strip("c").split("a")
    return int(x[0]) + int(x[1])
------------------------------------
# Option 5
def num_people(fam):
    x = 0
    for i in fam:
        if i % 2 == 0:
            x = x + 1
    return x
------------------------------------
# Option 6
def num_people(fam):
    x = 0
    for i in np.arange(len(fam)):
        if i % 2 == 0:
            x = x + int(fam[i])
    return x
```

- [✓ Option 1]
- ☐ Option 2
- [✓ Option 3]
- [✓ Option 4]
- ☐ Option 5
- [✓ Option 6]
- ☐ None of the above.

Question 4 (6 pts)

For those who plan on having children, an important consideration when deciding whether to live in an area is the cost of raising children in that area. The DataFrame `expensive`, defined below, contains all of the rows in `living_cost` where the `"avg_childcare_cost"` is at least $20,000.

```
expensive = living_cost[living_cost.get("avg_childcare_cost")
                        >= 20000]
```

We'll call a county an "expensive county" if there is **at least one** `"family_type"` in that county with an `"avg_childcare_cost"` of at least $20,000. Note that all expensive counties appear in the `expensive` DataFrame, but some may appear multiple times (if they have multiple `"family_type"`s with an `"avg_childcare_cost"` of at least $20,000).

Recall that the `"is_metro"` column contains Boolean values indicating whether or not each county is part of a metropolitan (urban) area. For all rows of `living_cost` (and, hence, `expensive`) corresponding to the same geographic location, the value of `"is_metro"` is the same. For instance, every row corresponding to San Diego County has an `"is_metro"` value of `True`.

Fill in the blanks below so that the result is a DataFrame indexed by `"state"` where the `"is_metro"` column gives the **proportion of expensive counties in each state that are part of a metropolitan area**. For example, if New Jersey has five expensive counties and four of them are metropolitan, the row corresponding to New Jersey should have a value of 0.8 in the `"is_metro"` column.

```
(expensive.groupby(____(a)____).max()
          .reset_index()
          .groupby(____(b)____).____(c)____)
```

a) (2 pts) What goes in blank (a)?

Solution: `["state", "county"]` or `["county", "state"]`

b) (2 pts) What goes in blank (b)?

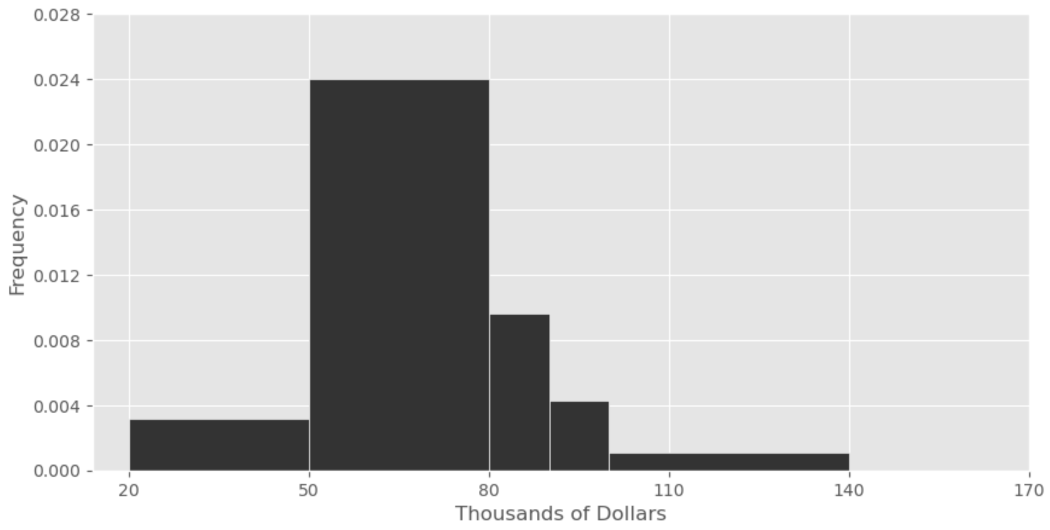
Solution: `"state"`

c) (2 pts) What goes in blank (c)?

Solution: `mean()`

Question 5 (7 pts)

The rows in `living_cost` with a `"family_type"` value of `"1a0c"` correspond to families that consist of individuals living on their own. We'll call such families "solo families." Below, we've visualized the distribution of the `"median_income"` column, but only for rows corresponding to solo families. Instead of visualizing median incomes in dollars, we've visualized them in thousands of dollars.



Suppose we're interested in splitting the $[50, 80)$ bin into two separate bins — a $[50, 70)$ bin and a $[70, 80)$ bin.

Let h_1 be the height of the new bar corresponding to the $[50, 70)$ bin and let h_2 be the height of the new bar corresponding to the $[70, 80)$ bin.

a) (4 pts) What are the minimum and maximum possible values of h_2? Give your answers as **decimals rounded to three decimal places**.

minimum = $\boxed{0}$ maximum = $\boxed{0.072}$

b) (3 pts) Suppose that the number of counties in which the median income of solo families is in the interval $[50, 70)$ is r times the number of counties in which the median income of solo families is in the interval $[70, 80)$. Given this fact, what is the value of

$$\frac{h_1}{h_2},$$

the ratio of the heights of the two new bars?

○ $\frac{1}{r}$ ○ $\frac{2}{r}$ ○ $\frac{3}{r}$ [✓ $\frac{r}{2}$] ○ $\frac{r}{3}$ ○ $2r$ ○ $3r$

Question 6 (4 pts)

Recall that `living_cost` has 31430 rows, one for each of the ten possible `"family_type"` values in each of the 3143 US counties.

Consider the function `state_merge`, defined below.

```
def state_merge(A, B):
    state_A = living_cost[living_cost.get("state") == A]
    state_B = living_cost[living_cost.get("state") == B]
    return state_A.merge(state_B, on="family_type").shape[0]
```

Suppose Montana (`"MT"`) has 5 counties, and suppose `state_merge("MT", "NV")` evaluates to `1050`. How many counties does Nevada (`"NV"`) have? Give your answer as an integer.

$\boxed{21}$

Question 7 (15 pts)

King Triton had four children, and each of his four children started their own families. These four families organize a Triton family reunion each year. The compositions of the four families are as follows:

- Family W: `"1a4c"`
- Family X: `"2a1c"`
- Family Y: `"2a3c"`
- Family Z: `"1a1c"`

Suppose we choose one of the fifteen people at the Triton family reunion at random.

a) (3 pts) Given that the chosen individual is from a family with one child, what is the probability that they are from Family X? Give your answer as a simplified fraction.

$\boxed{\frac{3}{5}}$

b) (3 pts) Consider the events A and B, defined below.

- A: The chosen individual is an adult.
- B: The chosen individual is a child.

True or False: Events A and B are independent.

○ True [✓ False]

c) (3 pts) Consider the events C and D, defined below.

- C: The chosen individual is a child.
- D: The chosen individual is from family Y.

True or False: Events C and D are independent.

[✓ True] ○ False

d) (6 pts) At the reunion, the Tritons play a game that involves placing the four letters into a hat (W, X, Y, and Z, corresponding to the four families). Then, **five times**, they draw a letter from the hat, write it down on a piece of paper, and place it back into the hat.

Let $p = \frac{1}{4}$ in the questions that follow.

(i) What is the probability that Family W is selected all 5 times?

[✓ p^5] ○ $1 - p^5$ ○ $1 - (1-p)^5$ ○ None of these.
○ $(1-p)^5$ ○ $p \cdot (1-p)^4$ ○ $p^4(1-p)$

(ii) What is the probability that Family W is selected at least once?

○ p^5 ○ $1 - p^5$ [✓ $1 - (1-p)^5$] ○ None of these.
○ $(1-p)^5$ ○ $p \cdot (1-p)^4$ ○ $p^4(1-p)$

(iii) What is the probability that Family W is selected exactly once, as the last family that is selected?

○ p^5 ○ $1 - p^5$ ○ $1 - (1-p)^5$ ○ None of these.
○ $(1-p)^5$ [✓ $p \cdot (1-p)^4$] ○ $p^4(1-p)$

Question 8 (14 pts)

After the family reunion, Family Y gets together with nine other families to play a game. All ten families (which we'll number 1 through 10) have a composition of `"2a3c"`. Within each family, the three children are labeled `"oldest"`, `"middle"`, or `"youngest"`.

In this game, the numbers 1 through 10, representing the ten families, are placed into a hat. Then, **five times**, they draw a number from the hat, write it down on a piece of paper, and place it back into the hat. If a family's number is written down on the paper at least twice, then two of the three children in that family are randomly selected to win a prize. The same child cannot be selected to win a prize twice.

Chiachan is the middle child in Family 4. He writes a simulation, which is partially provided on the next page. Fill in the blanks so that after running the simulation,

- `np.count_nonzero(outcomes == "Outcome Q") / repetitions` gives an estimate of the probability that Chiachan wins a prize.
- `np.count_nonzero(outcomes == "Outcome R") / repetitions` gives an estimate of the probability that both of Chiachan's siblings win a prize, but Chiachan does not.
- `np.count_nonzero(outcomes == "Outcome S") / repetitions` gives an estimate of the probability that nobody from Chiachan's family wins a prize.

PID: ____________________

```
ages = np.array(["oldest", "middle", "youngest"])
outcomes = np.array([])
repetitions = 10000
for i in np.arange(repetitions):
    fams = np.random.choice(np.arange(1, 11), 5, ____(a)____)
    if ____(b)____:
        children = np.random.choice(ages, 2, ____(c)____)
        if not "middle" in children:
            outcomes = np.append(outcomes, ____(d)____)
        else:
            outcomes = np.append(outcomes, ____(e)____)
    else:
        outcomes = np.append(outcomes, ____(f)____)
```

a) (2 pts) What goes in blank (a)?

[✓ `replace=True`] `replace=False`

b) (4 pts) What goes in blank (b)?

Solution: `np.count_nonzero(fams == 4) >= 2` or equivalent

c) (2 pts) What goes in blank (c)?

`replace=True` [✓ `replace=False`]

d) (2 pts) What goes in blank (d)?

`"Outcome Q"` [✓ `"Outcome R"`] `"Outcome S"`

e) (2 pts) What goes in blank (e)?

[✓ `"Outcome Q"`] `"Outcome R"` `"Outcome S"`

f) (2 pts) What goes in blank (f)?

`"Outcome Q"` `"Outcome R"` [✓ `"Outcome S"`]