Econ 148, Midterm

Spring 2023

Name: _____

Email: _____@berkeley.edu

Student ID: _____

Name and SID of left neighbor: _____

Name and SID of right neighbor: _____

Instructions:

This midterm exam consists of **40 points** spread out over **3 sections** and the Honor Code and must be completed in the **50 minute** time period ending at **2:00 PM**, unless you have accommodations supported by a DSP letter.

Note that some questions have circular bubbles to select a choice. This means that you should only **select one choice**. Other questions have boxes. This means you should **select all that apply**. Please shade in the box/circle to mark your answer.

Honor Code [1 Pt]:

As a member of the UC Berkeley community, I act with honesty, integrity, and respect for others. I am the person whose name is on the exam and I completed this exam in accordance with the Honor Code.

Signature: _____

1 Hawthorne Effects [7 Pts]

Does completing a household survey change the later behavior of those surveyed? In the Waterguard study, researchers provide evidence from a variety of settings that the act of being surveyed can affect behavior and confound estimates of parameters that initially motivated the data collection.

(a) [2 Pts] In 3 sentences or less, describe a randomized controlled trial research experiment to quantify the Hawthorne measurement effect.
Hint: Do you want to split people into different groups?

(b) [3 Pts] In the project in Kenya in Lab 3, the Hawthorne effect was observed to affect two different outcome variables, and in different directions. What were the outcomes and what was the direction of the Hawthorne effect?

(c) [2 Pts] In Lab 3 there were two related datasets with different dimensions, can you describe how the dimensions (or granularity) of the datasets relates to the sample design of the study?

2 An Intergalactic Analysis [16 Pts]

A new season of "The Mandalorian" has aired recently (no spoiler in this question) and as an ultimate Star Wars fan, Leon has embarked on a journey to dig deeper into the backstory of the show. He found several secretly hidden datasets online that contain information about the planets, ships, and other intriguing local records.
Throughout this question, we are dealing with pandas DataFrame and Series objects. All code for this question, where applicable, must be written in Python, unless explicitly stated otherwise. You may assume that pandas has been imported as `pd`.

(a) [1 Pt] Leon wants to do his analysis in the Python Jupyter Notebook. But when he tries to import the dataset he found online, it returns some weird error saying that "utf-8 cannot decode ...". He realizes that the datasets are actually coded in Galactic Basic (with a codex called `sw-gbc`). Help Leon to import one of the dataset `farm.csv` into the notebook.

Fill in the code below.

```
# load in `farm.csv` that is encoded in `sw-gbc`
nevarro_farm = _____
```

(b) [1 Pt] Now we got the data of fruits and vegetables on planet Nevarro. Leon is only interested in the fruits of Nevarro and the fruits must be red. Filter the dataframe so that it contains only data on fruits that are red.

	Crop	Type	Price	Is red?
0	Starfruit	Fruit	750	False
1	Parsnip	Vegetable	35	True
2	Sweet Gem Berry	Fruit	3000	True
3	Red Cabbage	Vegetable	260	True

Fill in the code below.

```
nevarro_farm_red_fruits_only = \
nevarro_farm[_____]
```

(c) [2 Pts] Leon did something else with this dataframe, but unfortunately his kernel was dead on datahub and lost all his code. But he remembered that the output was "True False True False". Which of the following code can produce this output? Select all that apply.

	Crop	Type	Price	Is red?
0	Starfruit	Fruit	750	False
1	Parsnip	Vegetable	35	True
2	Sweet Gem Berry	Fruit	3000	True
3	Red Cabbage	Vegetable	260	True

→

```
0     True
1    False
2     True
3    False
```

- ☐ `nevarro_farm["Crop"].str.startswith("S")`
- ☐ `nevarro_farm["Is red?"]`
- ☐ `nevarro_farm[nevarro_farm["Type"] == "Fruit"]`
- ☐ `nevarro_farm["Type"] == "Fruit"`
- ☐ None of the above

(d) [4 Pts] Leon is interested in smuggling some of those Red Star Fruits through the blockade. Leon found another dataset called `ships`, that describes all the types of ships in the universe.

	name	model	manufacturer	cost_in_credits	length	max_atmosphering_speed	cre
0	CR90 corvette	CR90 corvette	Corellian Engineering Corporation	3500000	150	950	30-16
1	Star Destroyer	Imperial I-class Star Destroyer	Kuat Drive Yards	150000000	1,600	975	47,06
2	Sentinel-class landing craft	Sentinel-class landing craft	Sienar Fleet Systems, Cyngus Spaceworks	240000	38	1000	
3	Death Star	DS-1 Orbital Battle Station	Imperial Department of Military Research, Sien...	1000000000000	120000	n/a	342,95

```
print (ships.info())
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 36 entries, 0 to 35
Data columns (total 18 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   name                  36 non-null     object
 1   model                 36 non-null     object
 2   manufacturer          36 non-null     object
 3   cost_in_credits       36 non-null     object
 4   length                36 non-null     object
 5   max_atmosphering_speed 36 non-null    object
 6   crew                  36 non-null     object
 7   passengers            36 non-null     object
 8   cargo_capacity        36 non-null     object
 9   consumables           36 non-null     object
 10  hyperdrive_rating     36 non-null     object
 11  MGLT                  36 non-null     object
 12  starship_class        36 non-null     object
 13  pilots                36 non-null     object
 14  films                 36 non-null     object
 15  created               36 non-null     object
 16  edited                36 non-null     object
 17  url                   36 non-null     object
```

Leon wants to do some analysis of the ships. Write some Pandas commands to do the following:

A Order the ships from biggest to smallest based on cargo capacity. Return all columns.

B Find the average cost by manufacturer. Return the manufacturer names and the average costs.

C Find the lowest-cost ship with hyperdrive rating of 4. Return all columns for that ship (if there are multiple ships with the lowest cost, you can return whichever one)

D Find any ship that can carry a cargo of at least 1 million kilos but with a length less than 200. Return all columns for those ships.

(e) [4 Pts] There's another database called `ships` in SQL with the same data structure (or schema) as the one above! Now do some additional analyses with SQL.

	name	model	manufacturer	cost_in_credits	length	max_atmosphering_speed	cre
0	CR90 corvette	CR90 corvette	Corellian Engineering Corporation	3500000	150	950	30-16
1	Star Destroyer	Imperial I-class Star Destroyer	Kuat Drive Yards	150000000	1,600	975	47,06
2	Sentinel-class landing craft	Sentinel-class landing craft	Sienar Fleet Systems, Cyngus Spaceworks	240000	38	1000	
3	Death Star	DS-1 Orbital Battle Station	Imperial Department of Military Research, Sien...	1000000000000	120000	n/a	342,95

```
print (ships.info())
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 36 entries, 0 to 35
Data columns (total 18 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   name                    36 non-null     object
 1   model                   36 non-null     object
 2   manufacturer            36 non-null     object
 3   cost_in_credits         36 non-null     object
 4   length                  36 non-null     object
 5   max_atmosphering_speed  36 non-null     object
 6   crew                    36 non-null     object
 7   passengers              36 non-null     object
 8   cargo_capacity          36 non-null     object
 9   consumables             36 non-null     object
 10  hyperdrive_rating       36 non-null     object
 11  MGLT                    36 non-null     object
 12  starship_class          36 non-null     object
 13  pilots                  36 non-null     object
 14  films                   36 non-null     object
 15  created                 36 non-null     object
 16  edited                  36 non-null     object
 17  url                     36 non-null     object
```

Write some SQL queries to do the following:

A Order the ships from biggest to smallest based on cargo capacity. Return all columns.

B Find the average cost by manufacturer. Return the manufacturer names and the average costs.

C Find the lowest-cost ship with hyperdrive rating of 4. Return all columns for that ship (if there are multiple ships with the lowest cost, you can return whichever one)

D Find any ship that can carry a cargo of at least 1 million kilos but with a length less than 200. Return all columns for those ships.

Leon came across a dataset called `planets` and started to look at the description of the planets. Leon is interested in finding a suitable planet for a young padawan to train. He has a theory that planets with a temperate climate have a relatively shorter orbital period.

```
planets
```

	name	rotation_period	orbital_period	diameter	climate	gravity	terrain	s
0	Tatooine	23	304	10465	arid	1 standard	desert	
1	Alderaan	24	364	12500	temperate	1 standard	grasslands, mountains	
2	Yavin IV	24	4818	10200	temperate, tropical	1 standard	jungle, rainforests	
3	Hoth	23	549	7200	frozen	1.1 standard	tundra, ice caves, mountain ranges	

(f) [2 Pts] How can Leon test whether planets with a temperate climate have a different orbital period than planets with other types of climate? Answer in both words and code.

(g) [2 Pts] How can Leon use a violin graph to test this hypothesis? Answer in both words and code.

3 Phillips Curve [16 Pts]

Two officials at the New York Fed – J Bow and G Pow – are having a heated debate over what policy they should set to combat the soaring inflation. First they want to look at several macro indicators that can summarize the current state of the economy.

All code for this question, where applicable, must be written in Python, unless explicitly stated otherwise. You may assume that pandas has been imported as `pd`.

(a) [1 Pt] Name one macro indicator that can represent the inflation rate?

(b) [2 Pts] Which two economic variables does the Phillips Curve capture? Draw a classical Phillips curve (label your plot properly).

(c) [2 Pts] In 3 sentences or less, describe the underlying mechanisms of the hypothesized Phillips curve.

J Bow and G Pow are suspicious of the theory presented in the textbook. They want to see if the Phillips Curve is actually true using empirical evidence – with some datasets.

(d) [2 Pts] In 2 sentences or less, describe your optimal dataset to test the hypothesized Phillips curve (also note the time frame and frequency). Name one data source that you can think of that contains the dataset of your interest.

(e) [2 Pts] You discovered that there is an API from Econ148.org that contains the relevant data series for the Phillips curve. In this API, the relevant data series id are `x-series` and `y-series` respectively; and there's a valid API key called `DEMO_KEY`.

Look at the API documentation provided below, and fetch relevant data for `x-series` from Jan. 1st, 1960 to Mar. 10th, 2023.

The url is `https://econ148.org`.
The endpoint is `series/observations`.

Parameters

1. api_key: The API Keys for this data source.
2. series_id: The id for a series.
3. observation_start: The start of the observation period as YYYY-MM-DD formatted string, optional, default: 1776-07-04 (earliest available)
4. observation_end: The end of the observation period as YYYY-MM-DD formatted string, optional, default: 9999-12-31 (latest available)

Fill in the code below.

```
import requests
from urllib.parse import urlencode

base_url = _____________________
endpoint = _____________________
params = {
_________________ : _________________,
_________________ : _________________,
_________________ : _________________,
_________________ : _________________
}

url_params = urlencode(params)
url = _____________________

# fires off the request
res = requests.get(url)

# return the content of the response
return res.json()

(additional data processing omitted)
```

(f) [2 Pts] You have collected data for J Bow and G Pow, and now you want to produce a visualization of the hypothesized Phillips Curve. Which type of visualization would you choose to use (line plot, histogram, scatterplot, barchart, etc.) and why?

(g) [3 Pts] Given two datasets (`x.csv` and `y.csv`) that you have collected and stored on your laptop, fill out the following code in Python to generate a plot you describe above. In the final plot, only include points where there is x and y data for the corresponding date in both of the original dataframes. You may use any methods in Pandas and plotting libraries in Python.

The dataframes look like the following:

dataframe x

	DATE	x
0	1948-01-01	3.4
1	1948-02-01	3.8
2	1948-03-01	4.0
3	1948-04-01	3.9
4	1948-05-01	3.5
...	...	...
895	2022-08-01	3.7
896	2022-09-01	3.5
897	2022-10-01	3.7
898	2022-11-01	3.6
899	2022-12-01	3.5

900 rows × 2 columns

dataframe y

	DATE	y
0	1958-01-01	2.79720
1	1958-04-01	2.42775
2	1958-07-01	2.06659
3	1958-10-01	1.82232
4	1959-01-01	1.81406
...	...	...
255	2021-10-01	5.00808
256	2022-01-01	6.29779
257	2022-04-01	6.01857
258	2022-07-01	6.29602
259	2022-10-01	5.98561

260 rows × 2 columns

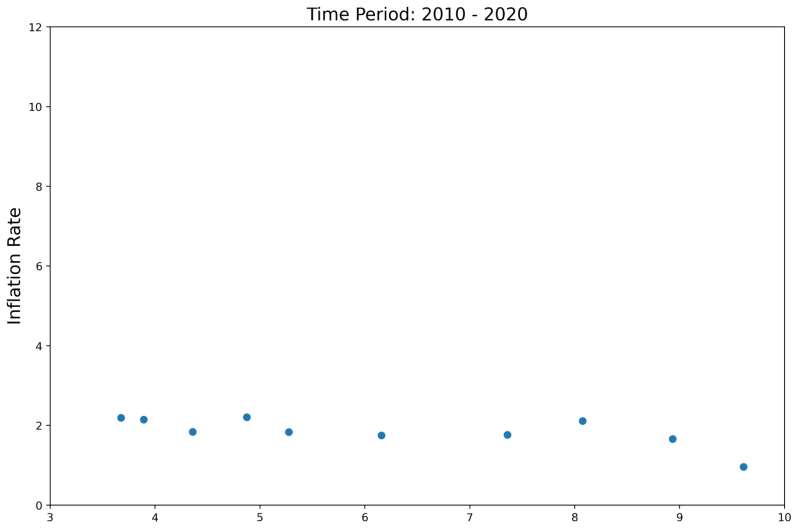
Assume dataframe y is the inflation rate, and dataframe x is for the other variable in the Phillips Curve.

```
# load in the datasets
x = _____________________
y = _____________________

# merge the datasets
pc_df = _____________________

# make a visualization of the Phillips curve
(label both axes and title properly)
_____________________________________________
_____________________________________________
_____________________________________________
_____________________________________________
_____________________________________________
_____________________________________________
_____________________________________________
(You may not need all the lines)
```

(h) [2 Pts] Below is a graph that G Pow found online for data in 2010-2020. Does this piece of empirical data support the hypothesized Phillips curve? If not, explain one potential way to reconcile this difference.



Spring 2023 Econ 148 Midterm Reference Sheet

Pandas

DataFrames & Series

In Pandas, tables are called `DataFrames`. We can think of them as a sequence of columns called `Series`.

This is a DataFrame:

```
farm = pd.DataFrame(
{"Crop":["Starfruit", …],
"Price":[750,…]})
```

This is a series:

```
farm["Price"]
```

	Crop	Price
0	Starfruit	750
1	Sweet Gem Berry	3000
2	Red Cabbage	260

.loc and .iloc accessors

We have two main ways of accessing rows and columns.

`.loc[ ]` lets us grab entries by their label:

```
df.loc[row_names, col_names]
>> farm.loc[1:2, :]
```

`.iloc[ ]` lets us grab entries by their index:

```
df.iloc[row_indices, col_indices]
>> farm.iloc[1:3, :]
```

	Crop	Price
1	Sweet Gem Berry	3000
2	Red Cabbage	260

Note that `iloc` is right-end exclusive!

Boolean filtering

We can filter out rows of our DataFrame using a Boolean array of True and False values.

First, apply a Boolean operator to the Series we want to use for filtering:

```
df["column_name"] (<, >, ==, etc.) value
>> farm_bool = farm["Price"] <= 1000
```

Then, use square brackets to filter out all False values from the DataFrame:

```
df[boolean_array]
>> farm[farm_bool]
or farm[farm["Price"] <= 1000]
```

	Crop	Price
0	Starfruit	750
2	Red Cabbage	260

Joining DataFrames using .merge

We can join two DataFrames using the .merge method. The DataFrames will pair up rows that share a common column.

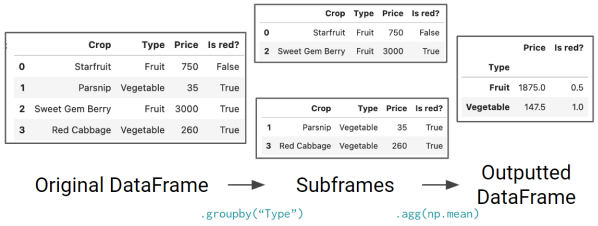
```
pd.merge(df1, df2, left_on="column_name", \
right_on="column_name", how=join_type)
```

You'll learn more about join types and primary/foreign key relationships when we study SQL later in the course.

Grouping with .groupby

If we want to group all entries by their type in a certain column, we can call df.groupby()

```
df.groupby("column_name").aggregator_func(func)
>> produce.groupby("Type").agg(np.mean)
```



We can use many aggregator functions on a GroupBy object:

```
gb.agg(func)
gb.mean()
gb.max()/gb.min()
gb.sum()
gb.first()/gb.last()
gb.filter(func)
```

Filtering groups using .filter

Sometimes we only want to keep rows that belong to a group satisfying some condition.

```
produce.groupby("Type").filter(lambda df: df["Price"].mean() < 200)
```

Here, our filter function takes in a DataFrame (a GroupBy subframe). It outputs one Boolean value. If `True`, all rows belonging to this group are kept in the final DataFrame. If `False`, all rows in this group are omitted.

	Crop	Type	Price
1	Parsnip	Vegetable	35
3	Red Cabbage	Vegetable	260

Mean price of vegetables: 147.5
Mean price of fruit: 1875
So, only the vegetables are kept!

Creating pivot tables with .pivot_table

Sometimes we want to group our data by two columns:

```
pd.pivot_table(data=produce, index="Type", columns="Is
red?", values="Price", aggfunc=sum)
```

Is red?	False	True
Type		
Fruit	750.0	3000.0
Vegetable	NaN	295.0

`index` gives the rows of the table
`columns` gives the columns

To fill out the cells, we apply `aggfunc` to `values`

Importing and Exporting Dataframes

CSV: pd.read_csv reads a comma-separated values (csv) file into DataFrame.

```
pandas.read_csv(filepath_or_buffer, sep=...,
delimiter=..., encoding=..., low_memory=True, ...)
```

Write object to a comma-separated values (csv) file.

```
DataFrame.to_csv(path_or_buf, sep=',', na_rep='',
float_format=None, columns=None, header=True,
index=True, index_label=None, mode='w', encoding=None)
```

Manipulating strings with .str

The `.str` accessory tells Pandas to perform operators on a Series of string data. This lets us manipulate every single string element in the Series, all at once. The process returns a new Series containing the manipulated strings.

```
df["column_name"].str.str_func()
>> produce["Crop"].str.startswith("S")
```

	Crop	Type	Price	Is red?
0	Starfruit	Fruit	750	False
1	Parsnip	Vegetable	35	True
2	Sweet Gem Berry	Fruit	3000	True
3	Red Cabbage	Vegetable	260	True

→

```
0    True
1   False
2    True
3   False
```

We can use many functions with `.str`:

`.split("delim")`

`.contains("val")`

`.startswith("val")`

`.slice(start, end)`

`[start:end]`

A short(ish) list of important Pandas methods:

`df.head()` - gives the first n rows of the DataFrame

`df.tail()` - gives the last n rows of the DataFrame

`df.shape` - gives the dimensions of the DataFrame

`df.rename()` - renames the rows/columns of the DataFrame

`df.set_index()` - sets the index to the specified column

`df.reset_index()` - resets the index to the default 0, 1, 2...

`df.relabel()` - relabels specific entries in the DataFrame

`df.drop()` - removes the specified rows/cols from the DataFrame

`df.sort_values()` - sorts rows by the specified column

`df.isna()` - checks if values in the DataFrame are NaN

`df.to_datetime()` - converts times to Datetime objects

`df.index` - returns the index of the DataFrame

`df.columns` - returns an array of the column labels

`df.copy()` - creates a copy of the DataFrame

`df.value_counts()` - summarizes the count of each column combo

SQL

1. `SELECT <column list>` - **select** columns in `<column list>` to keep
   a. `[DISTINCT]` - keep only **distinct** rows (filter out duplicates)
2. `FROM <table1>` - which table are we drawing data **from**
3. `[WHERE <predicate>]` - only keep rows **where** `<predicate>` is satisfied
4. `[GROUP BY <column list>]` - **group** together rows **by** value of columns in `<column list>`
5. `[HAVING <predicate>]` - only keep groups **having** `<predicate>` satisfied
6. `[ORDER BY <column list> [DESC/ASC]]` - **order** the output **by** value of the columns in `<column list>`, ASCending by default
7. `[LIMIT <amount>]` - **limit** the output to just the first `<amount>` rows

Visualization

Function	Description
`plt.plot(x, y)`	Creates a line plot of `x` against `y`
`plt.scatter(x, y)`	Creates a scatter plot of `x` against `y`
`plt.hist(x, bins=None)`	Creates a histogram of `x`
`plt.bar(x, height)`	Creates a bar plot

Regular Expressions

Operator	Description
`.`	Matches any character except **\n**
`\\`	Escapes metacharacters
`|`	Matches expression on either side of expression; has lowest priority of any operator
`\d`, `\w`, `\s`	Predefined character group of digits (0-9), alphanumerics (a-z, A-Z, 0-9, and underscore), or whitespace, respectively
`*`	Matches preceding character/group zero or more times
`?`	Matches preceding character/group zero or one times
`+`	Matches preceding character/group one or more times
`^`, `$`	Matches the beginning and end of the line, respectively
`( )`	Capturing group used to create a sub-expression
`[ ]`	Character class used to match any of the specified characters or range (e.g. `[abcde]` is equivalent to `[a-e]`)
`[^ ]`	Invert character class; e.g. `[^a-c]` matches all characters except `a`, `b`, `c`

Function	Description
`sns.countplot(data, x)`	Create a barplot of value counts of variable `x` from `data`
`sns.histplot(data, x, kde=False)` `sns.displot(x, data, rug = True, kde = True)`	Creates a histogram of `x` from `data`; optionally overlay a kernel density estimator. `displot` is similar but can optionally overlay a rug plot.
`sns.boxplot(data, x=None, y)` `sns.violinplot(data, x=None, y)`	Create a boxplot of y, optionally factoring by categorical `x`, from `data`. `violinplot` is similar but also draws a kernel density estimator of `y`.
`sns.scatterplot(data, x, y)`	Create a scatterplot of `x` versus `y` from `data`
`sns.lmplot(x, y, data, fit_reg=True)`	Create a scatterplot of `x` versus `y` from `data`, and by default overlay a least-squares regression line
`sns.jointplot(x, y, data, kind)`	Combine a bivariate scatterplot of `x` versus `y` from `data`, with univariate density plots of each variable overlaid on the axes; `kind` determines the visualization type for the distribution plot, can be `scatter`, `kde` or `hist`