Midterm Exam - EECS 398-003, Fall 2024

Full Name:	
Uniqname:	
UMID:	

Room: ○ CHRYS 133 ○ IOE 1610 ○ IOE 1680 ○ COOL 1940

Instructions:

- This exam consists of 14 questions, worth a total of 100 points. You have 120 minutes to complete the exam.
- Write your uniqname 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 a single two-sided handwritten notes sheet. Other than that, you may not refer to any other resources or technology during the exam (no phones, watches, or calculators).

You are to abide by the University of Michigan/Engineering Honor Code. To receive a grade, please sign below to signify that you have kept the honor code pledge.

I have neither given nor received aid on this exam, nor have I concealed any violations of the Honor Code.

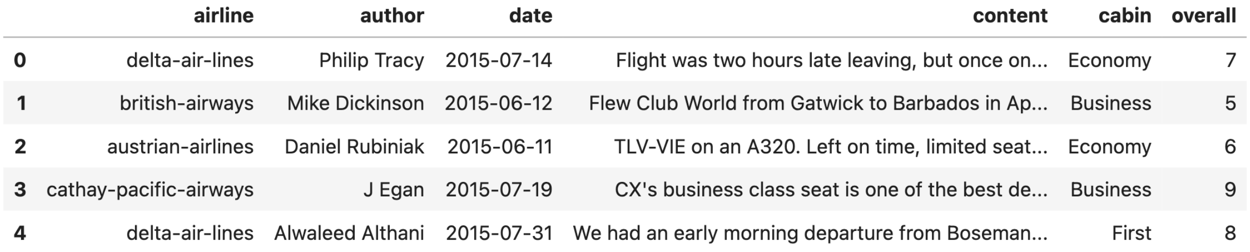
Signature:

Version A

Data Overview: Flight Reviews

Skytrax is a website that allows anyone to submit a review for a flight they took. In this exam, we'll work with the DataFrame `reviews`, which contains flight reviews taken from Skytrax.

The first few rows of `reviews` are shown below, but `reviews` has many more rows than are shown.



	airline	author	date	content	cabin	overall
0	delta-air-lines	Philip Tracy	2015-07-14	Flight was two hours late leaving, but once on...	Economy	7
1	british-airways	Mike Dickinson	2015-06-12	Flew Club World from Gatwick to Barbados in Ap...	Business	5
2	austrian-airlines	Daniel Rubiniak	2015-06-11	TLV-VIE on an A320. Left on time, limited seat...	Economy	6
3	cathay-pacific-airways	J Egan	2015-07-19	CX's business class seat is one of the best de...	Business	9
4	delta-air-lines	Alwaleed Althani	2015-07-31	We had an early morning departure from Boseman...	First	8

The columns in `reviews` are as follows:

- `"airline" (str)`: The airline flown. Note that an airline can be reviewed multiple times.
- `"author" (str)`: The author of the review. Note that an author can write multiple reviews (and can even write multiple reviews of the same airline!).
- `"date" (str)`: The date on which the review was written. Most of the reviews were written in 2015.
- `"content" (str)`: The content of the review.
- `"cabin" (str)`: The cabin the author flew; either `"Economy"`, `"Premium Economy"`, `"Business"`, or `"First"`.
- `"overall" (int)`: The overall rating the author gave their flight experience, between 0 and 10 (inclusive).

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

Make sure you have read the Data Overview before beginning!

Question 1 (6 pts)

Fill in the blanks so that `santas_airlines` evaluates to an array containing the **unique** names of the airlines that Santa Ono has reviewed.

```
santas_airlines = reviews.loc[__(i)__, __(ii)__].__(iii)__
```

(i):

(ii):　　　　(iii):

Question 2 (6 pts)

Consider the function `operate`, defined below.

```
def operate(df):
    df["content"] = df["content"].str.split().str.len()
```

Consider the following six answer choices:

A. Nothing, because we didn't reassign `reviews` after calling `operate(reviews)`.

B. `reviews["content"]` now contains the number of characters per review.

C. `reviews["content"]` now contains the number of words per review.

D. We see an error because we're trying to use `.str` methods on an invalid column.

E. We see an error because a column with the name `"content"` already exists.

F. We see an error because we're assuming `df` is a DataFrame but it is `None` (null).

All three parts of this question are **independent** from one another; the code in earlier parts should not influence your answers in later parts.

a) (2 pts) What happens after running the following block of code?

```
operate(reviews)
```

○ A　○ B　○ C　○ D　○ E　○ F

b) (2 pts) What happens after running the following block of code?

```
operate(reviews)
operate(reviews)
```

○ A　○ B　○ C　○ D　○ E　○ F

c) (2 pts) What happens after running the following block of code?

```
reviews = operate(reviews)
reviews = operate(reviews)
```

○ A　○ B　○ C　○ D　○ E　○ F

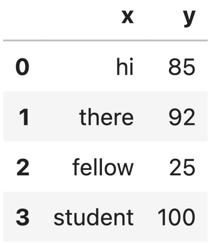
Question 3 (8 pts)

Suppose we define `n = reviews.shape[0]`.

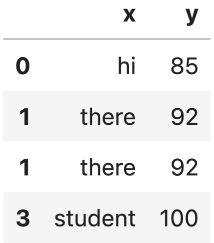
In each of the four parts of this question, we provide an expression that evaluates to a new DataFrame. Depending on the expression, the new DataFrame:

- May or may not have the same number of rows as `reviews`.
- May have some rows from `reviews` appear multiple times (i.e. may have duplicated rows). Note that `reviews` itself **does not** have any duplicated rows.
- May have rows that appear in a different order than they appeared in `reviews`.

For example, if `df` is on the left, the DataFrame on the right has the same number of rows as `df`, has some duplicated rows (row 1 appears twice), and has the same row order as `df` (all row 0s before row 1s, all row 1s before row 2s, etc.).



	x	y
0	hi	85
1	there	92
2	fellow	25
3	student	100



	x	y
0	hi	85
1	there	92
1	there	92
3	student	100

Select the options that correctly describe the DataFrame that results from each of the following expressions.

a) (2 pts) `reviews.loc[np.random.choice([True, False], size=n, replace=True)]`

(i) Same number of rows as `reviews`? ○ Yes ○ No
(ii) Possibility of duplicated rows? ○ Yes ○ No
(iii) Row order guaranteed to be the same as `reviews`? ○ Yes ○ No

b) (2 pts) `reviews.loc[np.random.permutation(np.arange(n))]`

(i) Same number of rows as `reviews`? ○ Yes ○ No
(ii) Possibility of duplicated rows? ○ Yes ○ No
(iii) Row order guaranteed to be the same as `reviews`? ○ Yes ○ No

c) (2 pts) `reviews.loc[np.random.choice(np.arange(n), size=n, replace=True)]`

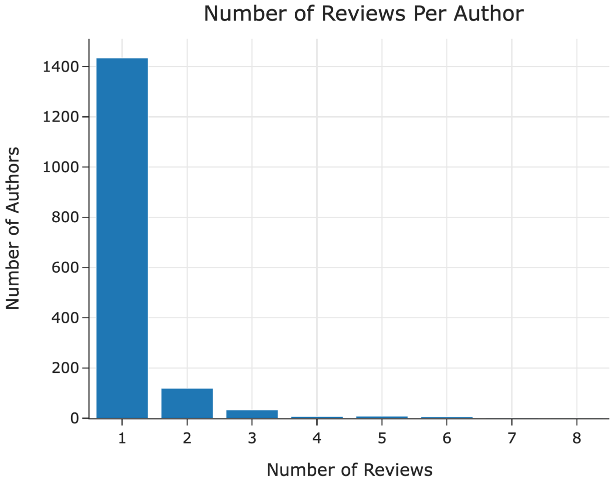
(i) Same number of rows as `reviews`? ○ Yes ○ No
(ii) Possibility of duplicated rows? ○ Yes ○ No
(iii) Row order guaranteed to be the same as `reviews`? ○ Yes ○ No

d) (2 pts) `reviews.loc[np.random.choice(np.arange(n), size=n, replace=False)]`

(i) Same number of rows as `reviews`? ○ Yes ○ No
(ii) Possibility of duplicated rows? ○ Yes ○ No
(iii) Row order guaranteed to be the same as `reviews`? ○ Yes ○ No

Question 4 (8 pts)

The bar chart below shows the distribution of the number of reviews per `"author"` in `reviews`. For instance, it's telling us that approximately 120 `"author"`s wrote 2 reviews. **Assume that the height of the bar at 8 on the x-axis is exactly 1, and that this is the shortest bar (with no ties).**



Hint: The **values** in the Series that results from calling `value_counts()` are sorted in descending order.

a) (2 pts) Fill in the blank below so that `W` evaluates to the value `7`.

```
W = reviews["author"].value_counts().__(i)__
```

`(i)`: □

Now, consider the values defined below.

```
W = reviews["author"].value_counts().value_counts().loc[1]
X = reviews["author"].value_counts().value_counts().iloc[1]
Y = reviews["author"].value_counts().value_counts().index[1]
Z = reviews["author"].value_counts().value_counts().index[-1]
```

b) (2 pts) Which value is equal to **8**?

○ `W` ○ `X` ○ `Y` ○ `Z` ○ None of these.

c) (2 pts) Which value is equal to **the height of the tallest bar**?

○ `W` ○ `X` ○ `Y` ○ `Z` ○ None of these.

d) (2 pts) Which value is equal to **the height of the second-tallest bar**?

○ `W` ○ `X` ○ `Y` ○ `Z` ○ None of these.

Question 5 (4 pts)

Select all visualization techniques we *could* use to visualize the distribution of `"overall"` ratings, separately for Delta Air Lines and United Airlines. At least one answer is correct.

□ side-by-side box plots

□ side-by-side bar charts

□ a scatter plot

□ a pie chart

□ overlaid histograms

Question 6 (8 pts)

We define the **robust mean** of a collection of values to be the mean of all values, once the largest and smallest values are removed. For example, the robust mean of $2, 2, 3, 7, 12$ is $\frac{2+3+7}{3} = 4$.

In part (a), fill in the blanks so that `worst_robustly` evaluates to the `"airline"` with the lowest robust mean of `"overall"` ratings. Assume there are no ties in robust ratings, and that each `"airline"` has at least 3 ratings.

```
def funky(x):
    return __(iii)__

worst_robustly = reviews.__(i)__["overall"].__(ii)__(funky).idxmin()
```

a) (6 pts)

(i): ○ `sort_values("overall")` ○ `sort_values("airline")`
○ `groupby("overall")` ○ `groupby("airline")`
○ `value_counts("overall")` ○ `value_counts("airline")`

(ii): ○ `agg` ○ `filter` ○ `transform`

(iii):

b) (2 pts) What are the input and output types of the function `funky`?

○ Input: Series, Output: Number

○ Input: Series, Output: Series

○ Input: Series, Output: DataFrame

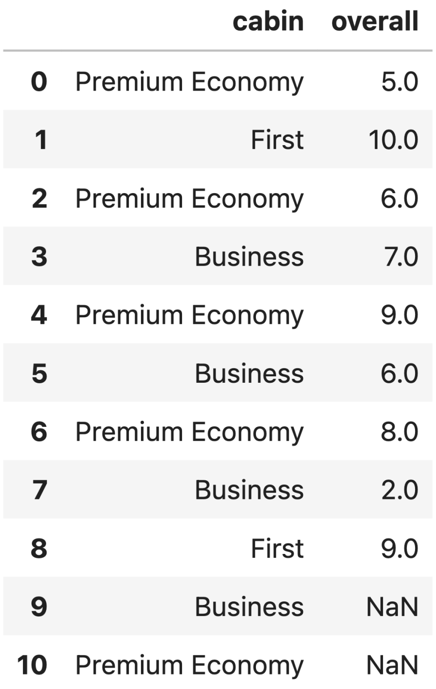
○ Input: DataFrame, Output: Number

○ Input: DataFrame, Output: Series

○ Input: DataFrame, Output: DataFrame

Question 7 (9 pts)

Consider the DataFrame `imp`, shown in its entirety below.



	cabin	overall
0	Premium Economy	5.0
1	First	10.0
2	Premium Economy	6.0
3	Business	7.0
4	Premium Economy	9.0
5	Business	6.0
6	Premium Economy	8.0
7	Business	2.0
8	First	9.0
9	Business	NaN
10	Premium Economy	NaN

As in the previous question, we define the **robust mean** of a collection of values to be the mean of all values, once the largest and smallest values are removed. For example, the robust mean of $2, 2, 3, 7, 12$ is $\frac{2+3+7}{3} = 4$.

a) (6 pts) Suppose we use **robust mean imputation, conditional on `"cabin"`** to fill in the missing values in the `"overall"` column. In doing so, what are the missing values in the `"overall"` column filled in with? Give your answers as **numbers**.

Missing value in row `9`: □ Missing value in row `10`: □

b) (3 pts) Suppose `imp` represents the DataFrame above, with 2 missing values, and `filled` represents the DataFrame that results from imputing the missing values in `imp` in the way described in the previous part. In the following code block, what is the value of `comp`?

```
old_means = imp.groupby("cabin")["overall"].mean()
comp = old_means == filled.groupby("cabin")["overall"].mean()
```

○ `True`
○ The Series `[True, True, True]`
○ The Series `[True, True, False]`
○ The Series `[False, True, True]`
○ The Series `[False, True, False]`
○ `False`
○ The Series `[False, False, False]`
○ The Series `[True, False, True]`
○ The Series `[False, False, True]`
○ The Series `[True, False, False]`

Hint: When grouping, the index is automatically sorted in ascending order.

Question 8 (8 pts)

Consider the DataFrame `framer`, defined below.

```
framer = reviews.pivot_table(index="airline",
                             columns="cabin",
                             values="author",
                             aggfunc="max")
```

a) (3 pts) In one English sentence, describe the meaning of the **value** that the following expression evaluates to.

```
framer.loc["emirates", "Economy"]
```

b) (2 pts) What is the value of the following expression?

```
framer.isna().sum(axis=0).shape[0]
```

○ The number of unique values in the `"airline"` column of `reviews`.

○ The number of unique values in the `"cabin"` column of `reviews`.

○ The number of unique values in the `"author"` column of `reviews`.

○ The number of unique values in the `"overall"` column of `reviews`.

○ None of the above.

c) (3 pts) Consider the following information.

```
>>> framer.isna().sum(axis=0).sum()
320
>>> reviews["airline"].nunique()
125
>>> reviews["cabin"].nunique()
4
```

What is the value of the following expression? Show your work, and box your final answer, which should be a **positive integer**.

```
reviews.groupby(["airline", "cabin"])["overall"].mean().shape[0]
```

Question 9 (10 pts)

Consider the DataFrame `lines`, created by keeping the rows in `reviews` corresponding to reviews written between 2015-11-24 and 2015-11-28. Remember, each row in `lines` corresponds to a review of an **airline**.

Also consider the DataFrame `ports`, in which each row corresponds to a review of a particular **airport**. `ports` has a `"date"` column, too, and also only has reviews written between 2015-11-24 and 2015-11-28.

Consider the following information.

```
>>> lines["date"].value_counts().sort_index()
2015-11-24      2
2015-11-25      7
2015-11-26      1
2015-11-27      4
>>> ports.shape[0]
17
>>> ports["date"].unique()
array(["2015-11-26", "2015-11-27", "2015-11-28"])
>>> lines.merge(ports, on="date", how="inner").shape[0]
29
>>> lines.merge(ports, on="date", how="outer").shape[0]
41
```

Below, give your answers as **integers**.

a) (2 pts) How many values in `ports["date"]` are equal to `"2015-11-24"`?

b) (2 pts) How many values in `ports["date"]` are equal to `"2015-11-25"`?

c) (2 pts) How many values in `ports["date"]` are equal to `"2015-11-26"`?

d) (2 pts) How many values in `ports["date"]` are equal to `"2015-11-27"`?

e) (2 pts) How many values in `ports["date"]` are equal to `"2015-11-28"`?

Question 10 (4 pts)

Consider the following SQL query.

```
SELECT author, AVG(overall) AS average_overall FROM reviews
GROUP BY author
HAVING AVG(overall) >= 8 AND COUNT(*) >= 5
ORDER BY average_overall;
```

Now, consider the following four code blocks.

- Block 1:

```
f = lambda x: x["overall"].mean() >= 8 and x.shape[0] >= 5
pop_auth = (reviews.groupby("author").filter(f)
                   .groupby("author")[["overall"]].mean()
                   .sort_values("overall").reset_index())
```

- Block 2:

```
g = lambda x: x.shape[0] >= 5
pop_auth = (reviews[reviews["overall"] >= 8]
            .groupby("author").filter(g)
            .groupby("author")[["overall"]].mean()
            .sort_values("overall").reset_index())
```

- Block 3:

```
pop_auth = (reviews["author"].value_counts().reset_index()
            .merge(
                reviews.groupby("author")[["overall"]].mean().reset_index(),
                on="author")
            .pipe(lambda x: x[(x["overall"] >= 8)])
            .sort_values("overall")
            .pipe(lambda r: r[r["count"] >= 5])
            [["author", "overall"]])
```

- Block 4:

```
temp = reviews.groupby("author")["overall"].agg(["count", "mean"])
pop_auth = (temp[(temp["count"] >= 5) & (temp["mean"] >= 8)]
            .sort_values("mean")
            .reset_index()[["author", "mean"]])
```

Three of the blocks above correctly assign **pop_auth** to a DataFrame that is equivalent to the table outputted by the SQL query at the top of the page (ignoring the names of the resulting columns and the values in the index). One of the options **is incorrect**.

Which block is **incorrect** — that is, which block **does not** assign **pop_auth** so that it is equivalent to the result of the SQL query at the top of the page?

○ Block 1 ○ Block 2 ○ Block 3 ○ Block 4

Question 11 (8 pts)

Suppose we define `soup` to be a BeautifulSoup object that is instantiated using the HTML document below. (To save space, we've omitted the tags `<html>` and `</html>`.)

```
<title>Aalborg Airport Customer Reviews - SKYTRAX</title>
<a href="https://www.airlinequality.com" rel="nofollow">
    <img src="https://www.airlinequality.com/images/skytrax.svg" alt="SKYTRAX">
</a>
<div class="skytrax-ratings">
    <table class="review-ratings">
        <tr>
            <td class="terminal-cleanliness">Terminal Cleanliness</td>
            <td>4</td>
        </tr>
        <tr>
            <td class="food-beverages">Food Beverages</td>
            <td>9</td>
        </tr>
        <tr>
            <td class="airport-staff">SKYTRAX Staff</td>
            <td>3</td>
        </tr>
    </table>
</div>
```

In parts (a) and (b), fill in the blanks so that each expression evaluates to `"SKYTRAX"`.

a) (2 pts) `soup.__(i)__("alt")`

(i):

b) (3 pts) `soup.find("td", __(i)__).text.__(ii)__`

(i):

(ii):

c) (3 pts) Fill in the blanks so that `avg_rating` evaluates to the mean of the three ratings in the document above. (In the document, the three ratings are 4, 9, and 3, but you **shouldn't** hard-code these values.)

```
texts = [tag.text for tag in soup.find_all("td")]
avg_rating = np.mean([__(i)__ for j in range(__(ii)__)])
```

(i):

(ii):

Question 12 (4 pts)

All airports have IATA codes, which are strings of three uppercase letters that uniquely identify the airport. For example, Detroit's IATA code is `"DTW"`. Many of the entries in the `"content"` column of `reviews` use IATA codes to describe the route flown.

We define a **route string** as being a pair of IATA codes separated by `"-"` or `" to "` (pay close attention to the spacing). For example, in `test_review` there are two route strings, `"YWG-LHR"` and `"EDI to YWG"`:

```
test_review = """I recently traveled YWG-LHR and returned home from
                 EDI to YWG with AC Rouge. I must say I was
                 pleasantly surprised with how well the trip went
                 in both directions, but I'm glad I didn't have to
                 transit in ORD."""
```

Fill in the blank below with a regular expression that matches valid route strings.

"   "

Question 13 (8 pts)

Consider the string `thoughts`, defined below:

```
thoughts = """Brought my Sony A5000 for my trip
              the Airbus A220v300's bathroom has a window
              My favorite jet is the B737-900ER flew it like x 239
              all stan the queen B747, 120% the B333st
              I like the chonky A380
              but I love the A350"""
```

There are exactly 5 valid **plane codes** in `thoughts`: `"A220"`, `"B737"`, `"B747"`, `"A380"`, and `"A350"`. For each regular expression `exp` below,

- specify the number of valid plane codes that the expression `re.findall(exp, thoughts)` extracts in their entirety, and
- specify the number of **other, invalid** strings that `re.findall(exp, thoughts)` also extracts.

The first example has been done for you.

a) `exp = r"B\d{3}"`

Valid plane codes extracted: 2 Invalid strings extracted: 1

b) (2 pts) `exp = r"[AB]\d{3}"`

Valid plane codes extracted: ______ Invalid strings extracted: ______

c) (2 pts) `exp = r"A\d{2}0|B7\d7"`

Valid plane codes extracted: ______ Invalid strings extracted: ______

d) (2 pts) `exp = r"A[23]\d0|B7\d7"`

Valid plane codes extracted: ______ Invalid strings extracted: ______

e) (2 pts) `exp = r"(A[23]\d0|B7\d7)$"`

Valid plane codes extracted: ______ Invalid strings extracted: ______

Question 14 (9 pts)

The following bag of words matrix represents the frequencies of various words across four reviews. Assume, just for this question, that the only five words that appear in any review are the columns of this matrix, and **all calculations use a base-2 logarithm**.

	"landing"	"snacks"	"turbulence"	"attendant"	"movies"
Review 0	2	1	3	0	A
Review 1	0	6	B	2	0
Review 2	4	8	0	8	0
Review 3	C	1	3	1	0

The matrix has three unknown **integer** values, A, B, and C. However, we're given that:

- The TF-IDF of `"movies"` in Review 0 is $\frac{5}{4}$.
- The TF-IDF of `"landing"` in Review 2 is $\frac{1}{5}$.
- The cosine similarity between the bag of words vectors for Reviews 1 and 2 is $\frac{16}{21}$.

a) (3 pts) What is the value of A? Show your work, and $\boxed{\text{box}}$ your final answer.

b) (3 pts) What is the value of B? Show your work, and $\boxed{\text{box}}$ your final answer.

c) (3 pts) What is the value of C? Show your work, and $\boxed{\text{box}}$ your final answer.

Make sure you've written your uniqname in the space provided in the top right corner of every page of this exam.

Congrats on finishing the exam! Feel free to draw us a picture about EECS 398 below :)