# DS-100 Midterm Exam

Spring 2018

Name: ________________________________________

Email: __________________________________`@berkeley.edu`

Student ID: ______________________________________

## Instructions:

- This midterm exam must be completed in the **80 minute time** period ending at **12:30PM**, unless you have accommodations supported by a DSP letter.
- Note that some questions have 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**.
- When selecting your choices, you must **fully shade** in the box/circle. Check marks will likely be mis-graded.
- You may use a one-sheet (two-sided) study guide.
- Work quickly through each question. There are a total of 189 points on this exam.

## Honor Code:

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: ______________________________________

# Syntax Reference

## Regular Expressions

**"^"** matches the position at the beginning of string (unless used for negation "[^]")

**"$"** matches the position at the end of string character.

**"?"** match preceding literal or sub-expression 0 or 1 times. When following "+" or "*" results in non-greedy matching.

**"+"** match preceding literal or sub-expression *one* or more times.

**"*"** match preceding literal or sub-expression *zero* or more times

**"."** match any character except new line.

**"[ ]"** match any one of the characters inside, accepts a range, e.g., "[a-c]".

**"( )"** used to create a sub-expression

**"\d"** match any *digit* character. "\D" is the complement.

**"\w"** match any *word* character (letters, digits, underscore). "\W" is the complement.

**"\s"** match any *whitespace* character including tabs and newlines. \S is the complement.

**"\b"** match boundary between words

Some useful `re` and `requests` package functions.

**`re.findall(pattern, st)`** return the list of all sub-strings in `st` that match `pattern`.

**`requests.get(url, auth, params, data)`** makes a *GET* requests with `params` in the header and `data` in the body.

**`requests.post(url, auth, params, data)`** makes a *POST* requests with `params` in the header and `data` in the body.

## Useful Pandas Syntax

```
df.loc[row_selection, col_list]  # row selection can be boolean
df.iloc[row_selection, col_list] # row selection can be boolean
df.groupby(group_columns)[['colA', 'colB']].sum()
pd.merge(df1, df2, on='hi') # Merge df1 and df2 on the 'hi' column


pd.pivot_table(df,                    # The input dataframe
               index=out_rows,        # values to use as rows
               columns=out_cols,      # values to use as cols
               values=out_values,     # values to use in table
               aggfunc="mean",        # aggregation function
               fill_value=0.0)        # value used for missing comb.
```

# Data Design and Bias

1. [1 Pt] Your letter grade (e.g., A+, A, ...) in a class that grades on a curve is most accurately described as what kind of data?

   ○ Nominal ○ Ordinal ○ Quantitative ○ Numerical

2. [1 Pt] The **number** of gold medals won by each country in the 2018 Olympics is an example of what kind of data:

   ○ Nominal ○ Ordinal ○ Qualitative ○ Quantitative

3. A discussion leader with **32 students** in her section would like to sample a single student that is representative of the total **population of students in her section**. She enumerates her students 0 to 31 and follows one of the following procedures:

   (a) [4 Pts] She flips a fair coin 31 times and records the number of heads. She then selects the student with the number that matches the number of heads. What type of sample has the discussion leader taken? **Select all that apply.**

   □ Simple random sample □ Probability sample □ Convenience sample
   □ None of the above

   (b) [4 Pts] She flips a fair coin 5 times and records the sequence of heads and tails as 1's and 0's, respectively. She then selects the student whose number corresponds to the binary sequence. For example, if she flipped $[1, 1, 0, 0, 1]$ then she would select:

   $$1 * 2^0 \;+\; 1 \times 2^1 \;+\; 0 \times 2^2 \;+\; 0 \times 2^3 \;+\; 1 \times 2^4 = \text{student } 21$$

   What type of sample has the discussion leader taken? **Select all that apply.**

   □ Simple random sample □ Probability sample □ Convenience sample
   □ None of the above

4. **Sampling True/False** For each of the following select true or false:

   (a) [1 Pt] If each element/member of the population has an equal chance of being chosen, then we have a simple random sample.

   ○ True ○ False

(b) [1 Pt] In cluster sampling, each cluster has an equal chance of being chosen.

○ True ○ False

(c) [1 Pt] In stratified sampling, each element of the population is a assigned to exactly one stratum.

○ True ○ False

(d) [1 Pt] A small simple random sample can often be more representative of the population than a very large convenience sample.

○ True ○ False

5. We would like to understand the sleeping habits on university students living in campus dorms across the United States.

(a) [2 Pts] To keep costs down we randomly sample a subset of dorms across the United States and then construct a simple random sample of students within each of the selected dorms. This is an example of which sampling procedure:

○ Simple random sample ○ Stratified sample ○ Cluster sample

(b) [2 Pts] Which of the following sampling procedures would ensure that we have good coverage of both male and female students within each dorm.

○ Simple random sample ○ Stratified sample ○ Cluster sample

# Pandas

6. **Pandas True/False**

(a) [1 Pt] If the pandas DataFrame `df` has 10 columns, then `df.iloc[:, 0:5]` will return a DataFrame with 5 columns.

○ True ○ False

(b) [1 Pt] Assuming that `len(df1) == 100` and `len(df2) == 100` are both true, then `df1.merge(df2, how='outer')` produces at most 200 rows.

○ True ○ False

(c) [1 Pt] The return type of the `pandas.DataFrame.groupby` function can either be a DataFrame or a Series object.

○ True ○ False

7. The tables **food** and **store** contain information regarding different ingredients and where to buy them. You may assume all strings are strings and numbers are floats.

   *This is preview of the first 5 rows of the DataFrames. You may assume it has many more rows than what is shown, with the same structure and no missing data.*

**food**

| index | name | color | calories | food_group |
|---|---|---|---|---|
| 0 | broccoli | green | 25 | vegetable |
| 1 | chicken | pink | 200 | meat |
| 2 | cheddar | yellow | 350 | dairy |
| 3 | mango | yellow | 40 | fruit |
| 4 | carrot | orange | 50 | vegetable |

**store**

| index | food_name | store_name | distance | price |
|---|---|---|---|---|
| 0 | broccoli | yasai | 1 | 1.5 |
| 1 | broccoli | safeway | 2 | 2 |
| 2 | cheddar | trader_joes | 1 | 4 |
| 3 | mango | berkeley_bowl | 3 | 1 |
| 4 | carrot | costco | 6 | 5 |

(a) [5 Pts] Which of the following expressions returns a **Series** containing only the **names** of all the **red vegetables** in the `food` DataFrame? **Select all that apply.**

```
□ food[(food["color"] == "red") |
       (food["food_group"] == "vegetable")]["name"]
□ food[(food["color"] == "red") &
       (food["food_group"] == "vegetable")]["name"]
□ food[(food["color"] == "red") &
       (food["food_group"] == "vegetable")]
□ food[(food["name"].isin(store["food_name"])) &
       (food["food_group"] == "vegetable")]
```

□ None of the above.

(b) [5 Pts] **Select all** of the following expressions that generate a DataFrame containing only rows of fruit.

```
□ food.set_index("food_group").loc["fruit", :]
□ food.where(food["food_group"] == "fruit")
□ food[food["food_group"] == "fruit"]
□ food["food_group"] == "fruit"
```

□ None of the above.

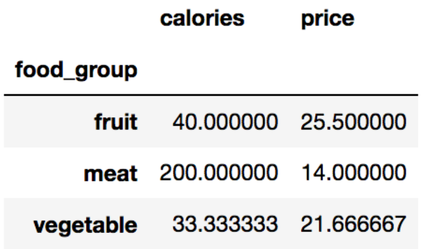
(c) [5 Pts] **Select all** true statements about the following expression.

```
cal100_foods = food[food["calories"] <= 100]
nearby_stores = store[store["distance"] <= 2]
output_df = cal100_foods.merge(nearby_stores,
                               how = "left",
                               left_on="name",
                               right_on="food_name")
```

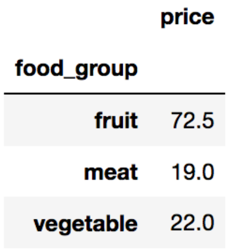
- □ `output_df['name']` and `output_df['food_name']` are always the same.
- □ `output_df` could contain NaN values.
- □ `nearby_stores` always contains the same number of rows as the `output_df`.
- □ `output_df` could contain more rows than the original food DataFrame.
- □ None of the above.

(d) [4 Pts] Which of the following tables is represented by `agg_df`?

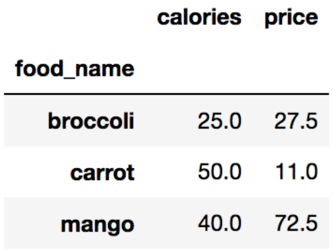
```
safeway_food = store[store["store_name"] == "safeway"]
merged_df = pd.merge(food, safeway_food, left_on="name",
                    right_on="food_name")
agg_df = (merged_df.groupby("food_group")
                   .mean()
                   .drop(columns="distance")
         )
```

○ 

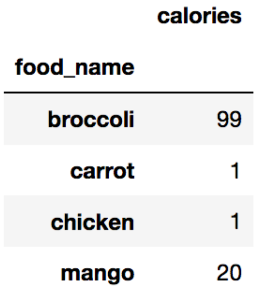
| | calories | price |
|---|---|---|
| food_group | | |
| fruit | 40.000000 | 25.500000 |
| meat | 200.000000 | 14.000000 |
| vegetable | 33.333333 | 21.666667 |

○ 

| | price |
|---|---|
| food_group | |
| fruit | 72.5 |
| meat | 19.0 |
| vegetable | 22.0 |

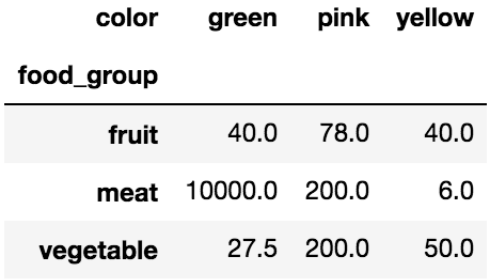
○ 

| | calories | price |
|---|---|---|
| food_name | | |
| broccoli | 25.0 | 27.5 |
| carrot | 50.0 | 11.0 |
| mango | 40.0 | 72.5 |

○ 

| | calories |
|---|---|
| food_name | |
| broccoli | 99 |
| carrot | 1 |
| chicken | 1 |
| mango | 20 |

(e) [4 Pts] Which of the following expressions would generate the following table?



| color | green | pink | yellow |
|---|---|---|---|
| **food_group** | | | |
| **fruit** | 40.0 | 78.0 | 40.0 |
| **meat** | 10000.0 | 200.0 | 6.0 |
| **vegetable** | 27.5 | 200.0 | 50.0 |

○
```
(food.groupby(["food_group", "color"])[["calories"]]
      .median())
```

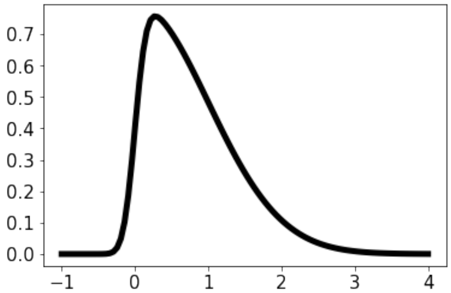
○
```
pd.pivot_table(food, values="calories",
     index="food_group", columns="color",
     aggfunc=np.median)
```

○
```
(food.set_index("food_group")
      .groupby("color")[["calories"]]
      .mean())
```

○
```
pd.pivot_table(food, values="calories",
     index="color", columns="food_group",
     aggfunc=np.median)
```

# EDA and Visualization

8. [5 Pts] Which of the following claims are true for the distribution shown below? **Select all that apply.**



□ It is left skewed □ It is unimodal □ The right tail is longer than the left tail □ It is symmetric □ None of the above

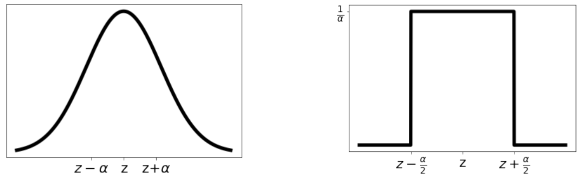
9. [5 Pts] We wish to compare the results of kernel density estimation using a gaussian kernel and a boxcar kernel. For $\alpha > 0$, which of the following statements are true? Choose all that apply.

Gaussian Kernel:

$$K_\alpha(x, z) = \frac{1}{\sqrt{2\pi\alpha^2}} \exp\left(-\frac{(x-z)^2}{2\alpha^2}\right)$$

Box Car Kernel:

$$B_\alpha(x, z) = \begin{cases} \frac{1}{\alpha} & \text{if } -\frac{\alpha}{2} \leq x - z \leq \frac{\alpha}{2} \\ 0 & \text{else} \end{cases}$$



(a) Gaussian (b) Box Car

- □ Decreasing $\alpha$ for a gaussian kernel decreases the smoothness of the KDE.
- □ The gaussian kernel is always better than the boxcar kernel for KDEs.
- □ Because the gaussian kernel is smooth, we can safely use large $\alpha$ values for kernel density estimation without worrying about the actual distribution of data
- □ The area under the box car kernel is 1, regardless of the value of $\alpha$

□ None of the above

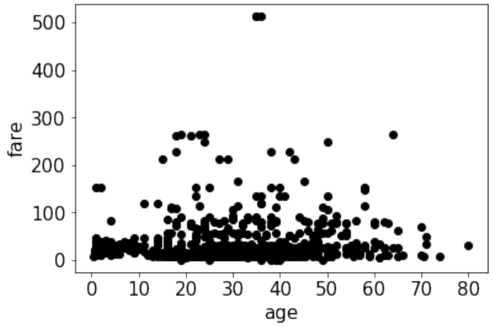
10. [5 Pts] Which of the following styles of plots are good for visualizing the distribution of a continuous variable? Choose all that apply.

□ Pie Charts □ Box Plots □ Bar Plots □ Histogram □ None of the above

11. [2 Pts] Suppose you wish to compare the number of homes homeowners in the US own and their respective salaries. Which style of plot would be the best?

○ Scatter Plot ○ Overlaid Line Plots ○ Side by Side Box Plots ○ Stacked Bar Plot

12. [5 Pts] Consider the plot below. What are some ways to improve the plot? Choose all that apply. Assume each is done individually.



□ Remove outliers and then plot on a different scale

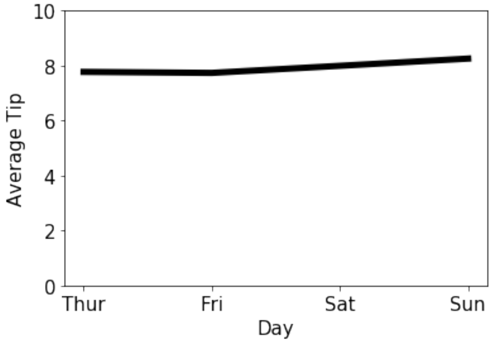
□ Plot as a line plot instead of a scatterplot.

□ Jitter the data with noise sampled from a uniform distribution of (-1, 1)

□ Utilize transparency

□ None of the above

13. [5 Pts] Consider the plot below which visualizes day of the week versus the average tip given in dollars. What are serious visualization errors made with this plot? Choose all that apply.



□ Area perception □ Jittering □ Overplotting □ Stacking □ None of the above

14. **True/False**

(a) [1 Pt] A data scientist must always consider potential sources of bias in a given dataset.

○ True ○ False

(b) [1 Pt] It is always reasonable to drop missing values.

○ True ○ False

15. Use the following dataset to answer the following questions:

```
id,diet,pulse,time,kind
1,low fat,85,1 min,rest
1,low fat,85,15 min,rest
1,low fat,88,30 min,rest
2,low fat,90,1 min,rest
2,low fat,92,15 min,rest
2,low fat,93,30 min,rest
3,low fat,97,1 min,rest
3,low fat,97,15 min,rest
```

(a) [1 Pt] Which of the following **best** describes the format of this file?

○ Raw text
○ Tab Separated Values (TSV)
○ Comma Separated Values (CSV)
○ JSON

(b) [4 Pts] Select **all** the true statements.

□ From the data available, the `id` seems to be a primary key.
□ There appear to be no missing values.
□ There are nested records.
□ None of the above.

16. [5 Pts] Select **all** the true statements about the following **XML** file:

```
< email >
        <to>Mr. Garcia
                <body>Hello there! How are we today?</to>
        </body>
< /email >
< email >
        <to>Mr. Garcia
                <body>Hello there! How are we today?</to>
        </body>
< /email >
```

□ This XML file is correctly formatted.

□ Tags are not properly nested.

□ This XML file is missing one root node that contains all the other nodes

□ The email tag on lines 1, 5, 6 and 10 should not have spaces between {<, >} and tag name.

□ None of the above are true.

17. Use the following **JSON** file `classes.json` printed below:

```
 1  [{
 2      "Prof": "Gonzalez",
 3      "Classes": [ "CS186",
 4          {
 5              "Name": "Data100",
 6              "Year": [2017, 2018]
 7          }],
 8      "Tenured": false
 9  },
10  {
11      "Prof": "Nolan",
12      "Classes": ["Stat133", "Stat153", "Stat198", "Data100"],
13      "Tenured": true
14  }]
```

(a) [5 Pts] Select **all** the true statements.

□ This JSON file is correctly formatted.

□ The `Classes` list defined on line 3 contains strings and dictionaries which is not permitted.

□ The dates 2017 and 2018 on lines 6 should be quoted.

□ the dictionary keys (e.g., `"Prof"`, `"Classes"`) should not be quoted.

□ None of the above statements are true.

(b) [3 Pts] What would be the output of the following block of code:

```
import json
with open("classes.json", "r") as f:
    x = json.load(f)
len(x[0]["Classes"][0])
```

○ 1 ○ 2 ○ 4 ○ 5 ○ None of the above.

18. [6 Pts] Which data formats would be well suited for nested data? **Select all that apply.**
□ `*.csv` □ `*.xml` □ `*.py` □ `*.json` □ `*.tsv` □ None of the above.

19. [6 Pts] Which of the following are reasonable motivations for applying a **log** transformation? **Select all that apply**:

    □ Perform dimensionality reduction on the data.

    □ To help straighten relationships between pairs of variables.

    □ Remove missing values.

    □ Bring data distribution closer to random sampling.

    □ To help visualize highly skewed distributions.

    □ None of the above.

20. [4 Pts] Which of the of the following record is the most **coarse** grained?

    ○ `{"Location": "Downtown Berkeley", "avg_income": 83000}`

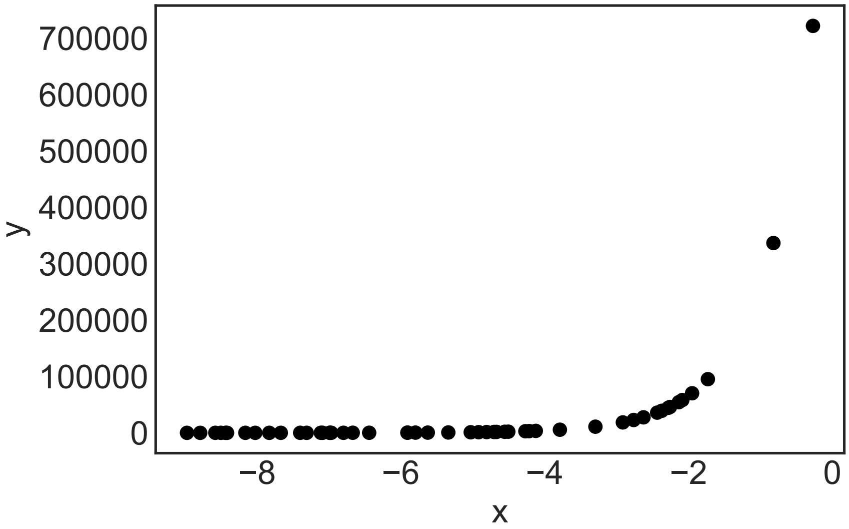
    ○ `{"Location": "Los Angeles, CA", "avg_income": 75042}`

    ○ `{"Location": "Bay Area, CA", "avg_income": 73042}`

    ○ `{"Location": "California", "avg_income": 50001}`

21. [4 Pts] Which of the following transformations would be best suited to linearize the relationship shown in the plot below? Note that all $y > 0$.:

    ○ Plotting $\log(y)$ vs $\log(x)$. ○ Plotting $\log(y)$ vs $x$. ○ Plotting $\exp(y)$ vs $\exp(x)$.
    ○ Plotting $\exp(y)$ vs $\log(x)$. ○ Plotting $y$ vs $\log(x)$. ○ Plotting $\log(y)$ vs $\log(\log(x))$



# Regular Expressions and String Manipulation

22. What would the following lines of code return? There are no spaces in any of the strings.

(a) [3 Pts] `re.findall(r"\..*", "VIXX-Error.mp3.bak")`

○ `[]` ○ `['bak']` ○ `['.bak']` ○ `['.mp3', '.bak']`
○ `['.mp3.bak']` ○ `['VIXX-Error.mp3.bak']`

(b) [3 Pts] `re.findall(r"[cat|dog]", "bobcat")`

○ `[]` ○ `['cat']` ○ `['c', 'a', 't']` ○ `['o', 'cat']`
○ `['o', 'c', 'a', 't']` ○ None of the above

(c) [3 Pts] `re.findall(r"a?p*[le]$", "apple")`

○ `[]` ○ `['e']` ○ `['appl']` ○ `['appe']`
○ `['a', 'pp', 'l', 'e']` ○ None of the above

(d) [3 Pts] `re.findall(r"</[^>]*>|<[^/]*/>",`
`"<body><h1>text</h1><img/></body>")`

○ `[]` ○ `['<body>', '<h1>']` ○ `['body', 'h1']`
○ `['</h1>', '<img/>', '</body>']` ○ `['</h1>', '</body>']`
○ `['<body>', '<h1>', '</h1>', '<img/>', '</body>']`
○ `['body', 'h1', '/h1', 'img/', '/body']` ○ None of the above

23. [9 Pts] On which of the following words would the regular expression `r"^\w[^p].*r"` return a match (on part or all of the word) instead of `None`? **Choose all that apply.**

□ sporous □ sooloos □ murdrum □ repaper □ hydroaviation
□ defendress □ gourmet □ level □ redder

24. [5 Pts] Which regular expression would match part or all of the words on the left but NONE the ones on the right? Choose all that apply

| | |
|---|---|
| flossy | baronet |
| beefin | oriole |
| ghost | scupper |

□ ^.{5}[^e]?$ □ ^.+[^e]?$ □ [a-z]5[^e]?$ □ [fh] □ None of the Above

# Relational Databases and SQL

We have decided to launch *join.we* a new *relational* dating service for x-people. To keep track of users we created the following schema:

```
CREATE TABLE users (
    uid INT PRIMARY KEY,
    name TEXT,
    age INTEGER,
    state CHAR(2));
```

```
CREATE TABLE dates (
    did INT,
    uid INT REFERENCES users(uid),
    score REAL,
    PRIMARY KEY (did, uid));
```

The users table consists of the user id (uid), their name, age, and the state in which they reside. The dates table consists of the date id (did), the user id (uid) of one of the users that went on that date, and the score that user gave for the date. We will use the following data:

| uid | name | age | state |
|---|---|---|---|
| 1 | Storm | 31 | CA |
| 2 | Iceman | 22 | VT |
| 3 | Frenzy | 24 | CA |
| 4 | Kitty | 28 | CA |
| 6 | Surge | 25 | VA |

(a) users

| did | uid | score |
|---|---|---|
| 1 | 1 | 5.0 |
| 1 | 2 | 9.0 |
| 2 | 1 | 2.0 |
| 2 | 3 | 8.0 |
| 3 | 4 | 7.0 |
| 3 | 6 | 1.0 |

(b) dates

25. [5 Pts] **Select all** of the true statements about the above schema:

□ More than one user can have the same name.

□ It would **not** be possible to have more than two people on the same date (did).

□ A user can only provide a single score for a given date (did).

□ Every uid in the dates table has a matching uid in the users table.

□ None of the above statements are true.

26. [4 Pts] What does the following query compute?

```
SELECT did, AVG(score) AS avg_score
FROM dates
GROUP BY did
```

○ The date id and total score for each date.

○ The date id and average score for each user.

○ The date id and average score for each date.

○ This query is invalid because the score variable occurs in the SELECT clause.

Repeating the schema and data from the previous page for easy reference:

```
users (uid, name, age, state          dates (did, uid, score,
    PRIMARY KEY (uid));                   PRIMARY KEY (did, uid));
```

| uid | name | age | state |
|---|---|---|---|
| 1 | Storm | 31 | CA |
| 2 | Iceman | 22 | VT |
| 3 | Frenzy | 24 | CA |
| 4 | Kitty | 28 | CA |
| 6 | Surge | 25 | VA |

(a) users

| did | uid | score |
|---|---|---|
| 1 | 1 | 5.0 |
| 1 | 2 | 9.0 |
| 2 | 1 | 2.0 |
| 2 | 3 | 8.0 |
| 3 | 4 | 7.0 |
| 3 | 6 | 1.0 |

(b) dates

27. [4 Pts] What does the following query compute?

```
SELECT users.name, AVG(score) AS avg_score
FROM users, dates
WHERE dates.uid = users.uid AND users.state = 'CA'
GROUP BY users.uid
```

○

| name | avg_score |
|---|---|
| Storm | 5.0 |
| Frenzy | 8.0 |
| Kitty | 7.0 |

○

| name | avg_score |
|---|---|
| Storm | 3.5 |
| Frenzy | 8.0 |
| Kitty | 7.0 |

○

| name | avg_score |
|---|---|
| Iceman | 9.0 |
| Frenzy | 8.0 |
| Kitty | 7.0 |
| Surge | 1.0 |
| Storm | 3.5 |

○ None of the above are correct.

28. [4 Pts] What does the following query compute?

```
WITH couples AS (
    SELECT d1.did AS did, d1.uid AS uid1, d2.uid AS uid2
    FROM dates AS d1, dates AS d2
    WHERE d1.did = d2.did AND d1.uid > d2.uid
)
SELECT c.did, u1.name AS name1, u2.name AS name2
FROM couples AS c, users AS u1, users AS u2
WHERE (c.uid1 = u1.uid AND c.uid2 = u2.uid)
```

○ The date id and names of all possible pairs of users.

○ The date id and names of pairs of users that went on dates.

○ The date id and user ids of users that went on dates.

○ This is an invalid query because you cannot join a table with itself.

○ None of the above.

# PUT, GET, and the REST of the Web

29. [4 Pts] **Querying the GitHub REST API**

GitHub's REST API describes that, starting with the endpoint `https://api.github.com`, all the notifications for a user can be listed with the query:

```
GET /notifications
```

This API endpoint also supports the following parameters:

- `before`: a string in `YYYY-MM-DD` format, to only show notifications before the given date.
- `since`: a string in `YYYY-MM-DD` format, to only show notifications after the given date.

If I want to query for all notifications for the month of August 2017, which query is the best option, from the choices below?

Assume that the variables `user` and `pwd` contain my correct GitHub username and password, and that `import requests` has already been run.

○
```
requests.put('https://api.github.com/GET/notifications',
              auth=(user, pwd),
              params={'since': '2017-08-01',
                      'before': '2017-09-01'})
```

○
```
requests.get('https://api.github.com/notifications',
             auth=(user, pwd),
             params={'since': '2017-08-01',
                     'before': '2017-09-01'})
```

○
```
requests.put('https://api.github.com/notifications',
             auth=(user, pwd),
             get=True,
             since='2017-08-01',
             before='2017-09-01')
```

○
```
requests.get('https://api.github.com/notifications',
             auth=(user, pwd),
             since='2017-08-01',
             before='2017-09-01')
```

# Modeling and Estimation

30. Let $x_1, \ldots, x_n$ denote any collection of numbers with average $\overline{x} = \frac{1}{n}\sum_{i=1}^n x_i$.

(a) [3 Pts] $\sum_{i=1}^n (x_i - \overline{x})^2 \le \sum_{i=1}^n (x_i - c)^2$ for all $c$.

○ True ○ False

(b) [3 Pts] $\sum_{i=1}^n |x_i - \overline{x}| \le \sum_{i=1}^n |x_i - c|$ for all $c$.

○ True ○ False

31. Consider the following loss function based on data $x_1, \ldots, x_n$:

$$\ell(\mu, \sigma) = \log(\sigma^2) + \frac{1}{n\sigma^2}\sum_{i=1}^n (x_i - \mu)^2.$$

(a) [5 Pts] Which estimator $\widehat{\mu}$ is a minimizer for $\mu$, i.e. satisfies $\ell(\widehat{\mu}, \sigma^2) \le \ell(\mu, \sigma^2)$ for any $\mu, \sigma$?

○ $\widehat{\mu} = 0$

○ $\widehat{\mu} = \frac{1}{n}\sum_{i=1}^n x_i$

○ $\widehat{\mu} = \frac{1}{n}\sum_{i=1}^n x_i + \log\left(\frac{1}{n}\sum_{i=1}^n x_i\right)^2$

○ $\widehat{\mu} = \frac{1}{n\sigma^2}\sum_{i=1}^n x_i + \log\left(\sigma^2\right)$

○ $\widehat{\mu} = \texttt{median}(x_1, \ldots, x_n)$.

(b) [10 Pts] Which of the following is the result of solving $\frac{\partial \ell}{\partial \sigma} = 0$ for $\sigma$ (for fixed $\mu$)? Show your work in the box below.

○ $\sigma = \frac{1}{n}\sum_{i=1}^n (x_i - \mu)^2$.

○ $\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^n (x_i - \mu)^2}$.

○ $\sigma = \frac{2}{n}\sum_{i=1}^n (\mu - x_i)$.

○ $\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^n \sum_{j=1}^n (x_i - x_j)^2}$.

32. [10 Pts] Consider the following loss function based on data $x_1, \ldots, x_n$ with mean $\overline{x}$:

$$\ell(\beta) = \log \beta + \frac{\overline{x}}{\beta} + \frac{1}{n} \sum_{i=1}^{n} e^{-x_i/\beta}$$

Given an estimate $\beta^{(t)}$, write out the update $\beta^{(t+1)}$ after one iteration of gradient descent with step size $\alpha$. Show your work in the box below.