DS-100 Final Exam, Version A

Fall 2018

Name: ____

Email: ___________________________________`@berkeley.edu`

Student ID: ______________________________________

Instructions:

- This final exam must be completed in the **170 minute** time period ending at **2:30 PM**, 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. We reserve the right to deny regrade requests if an answer choice is not completely filled in.
- Write **clearly and legibly** when filling in free response questions.
- You may use a two-sheet (each two-sided) study guide.

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 expression on either side of symbol. Has lowest priority.

"\\" match the following character literally.

"?" match preceding literal or sub-expression 0 or 1 times.

"+" 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]"`. All characters inside treated literally.

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

"{n}" preceding expression repeated n times.

Some useful Python functions and syntax

`re.findall(pattern, st)` returns the list of all non-overlapping sub-strings in `st` that match `pattern`.

`np.random.choice(a, replace, size)` Generates a random sample from `a` consisting of `size` values (with replacement if `replace=True`). `a` can be 1-D array-like or int.

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
pd.get_dummies(data) # Convert categorical variable into indicator values
df.groupby(group_columns)[['colA', 'colB']].agg(agg_func)
df.groupby(group_columns)[['colA', 'colB']].filter(filter_func)
```

Variance and Expected Value

The expected value of X is $\mathbb{E}[X] = \sum_{j=1}^{m} x_j p_j$. The variance of X is $Var[X] = \mathbb{E}[(X - \mathbb{E}[X])^2] = \mathbb{E}[X^2] - \mathbb{E}[X]^2$. The standard deviation of X is $SD[X] = \sqrt{Var[X]}$.

Misc

For calculations involving percentiles of collections of numbers, we will use the following convention from Data 8: Let p be a number between 0 and 100. The p^{th} percentile is the smallest number in the collection that is at least as large as $p\%$ of all the values.

The logistic equation is $\sigma(x) = \frac{1}{1+\exp(-x)}$ and the KL divergence for two distributions is $D(P||Q) = \sum_{k=1}^{K} P(k)\log(P(k)/Q(k))$

Score Breakdown

Page	Points
4	8
5	3
6	6
8	7
9	9
Total:	33

Tabular Data

1. For this section, we will be working with the UC Berkeley Undergraduate Career Survey dataset. Each year, the UC Berkeley career center surveys graduating seniors for their plans after graduating. Below is a sample of the full dataset. The full dataset contains many thousands of rows.

j_name	c_name	c_location	m_name
Llama Technician	Google	MOUNTAIN VIEW	EECS
Software Engineer	Salesforce	SF	EECS
Open Source Maintainer	Github	SF	Computer Science
Big Data Engineer	Microsoft	REDMOND	Data Science
Data Analyst	Startup	BERKELEY	Data Science
Analyst Intern	Google	SF	Philosophy

Table 1: `survey` Table

Each record of the `survey` table is an entry corresponding to a student. We have the student's major information (`m_name`), company information (`c_name, c_location`), and the job title (`j_name`).

(a) [3 Pts] Write a SQL query that selects all data science major graduates that got jobs in Berkeley. The result generated by your query should include all 4 columns.

```
__________________ FROM survey
__________________________________________________________
__________________________________________________________
```

Solution:
```
SELECT * FROM survey
WHERE m_name = 'Data Science'
AND c_location = 'Berkeley'
```

(b) [5 Pts] Write a SQL query to find the top 5 popular companies that data science graduates will work at, from most popular to 5th most popular.

```
SELECT c_name, ________________________ as count
FROM survey
WHERE ____________________ = 'Data␣Science'
GROUP BY ____________________
ORDER BY ____________________
LIMIT 5
```

Solution:
```
SELECT c_name, COUNT(*) AS count
FROM survey
WHERE m_name = 'Data Science'
```

```
GROUP BY c_name
ORDER BY count DESC
LIMIT 5;
```

(c) [3 Pts] Suppose our table has 9,000 rows, with 3,000 unique job names, 1,700 unique company names, 817 unique locations, and 105 unique major names. The table above has many redundancies. Suppose we wanted to instead use the star schema idea from lecture, where we have one fact table and many dimension tables. How many dimension tables would we end up with? How many rows would there be in our fact table? How many columns would there be in our fact table? There may be more than one correct answer.

i. Number of dimension tables: **4**

ii. Number of rows in fact table: **9000**

iii. Number of columns in fact table: **4**

(d) [3 Pts] Consider the pandas expression below, where `nunique` returns the number of unique elements in a group.

`survey.groupby('c_name')['m_name'].nunique().max().`

What does it return?

◯ **A. One value: The number of unique majors for the company with the most unique majors.**

◯ B. One value: The number of unique companies for the major with the most hires.

◯ C. Many values: For each company, the count of the number of hires for the most popular major.

◯ D. Many values: For each major, the count of the number of hires by the most popular company.

(e) [3 Pts] Which of the SQL expressions below is equivalent to the pandas code from above?

◯ **A.**
```
SELECT MAX(count)
FROM (
  SELECT c_name, COUNT(DISTINCT m_name) AS count
  FROM survey
  GROUP BY c_name
  );
```

◯ B.
```
SELECT c_name, MAX(COUNT(DISTINCT m_name)) AS count
FROM survey
GROUP BY c_name;
```

◯ C.
```
SELECT c_name, COUNT(DISTINCT m_name) AS count
FROM survey
GROUP BY c_name
HAVING MAX(count);
```

◯ D.
```
SELECT MAX(count)
FROM (
  SELECT c_name, COUNT(DISTINCT m_name) AS count
  FROM survey
  GROUP BY c_name
  )
WHERE count >= MAX(count);
```

Cleaning, EDA, Visualization

Let's take a look at the California Air Quality Index (AQI) for 2017. The following cells and outputs are for your reference.

```
aq = pd.read_csv("./air_quality_final.csv", index_col=0)
aq.head()
```

	Date	AQI	COUNTY_CODE	COUNTY	LAT	LON
0	01/01/2017	24.0	1	Alameda	37.687526	-121.784217
1	01/02/2017	19.0	1	Alameda	37.687526	-121.784217
2	01/03/2017	NaN	1	Alameda	37.687526	-121.784217
3	01/04/2017	15.0	1	Alameda	0.000000	0.000000
4	01/05/2017	20.0	1	NaN	37.687526	-121.784217

```
aq.iloc[49437:49442]
```

	Date	AQI	COUNTY_CODE	COUNTY	LAT	LON
49437	01/01/2017	NaN	113	Yolo	38.534450	-121.773400
49438	01/02/2017	15.0	113	Yolo	38.534450	-121.773400
49439	01/03/2017	36.0	113	Yolo	38.534450	-121.773400
49440	01/04/2017	18.0	113	Yolo	37.995239	-121.756812
49441	01/05/2017	16.0	113	NaN	38.534450	-121.773400

```
aq.describe()
```

	AQI	COUNTY_CODE	LAT	LON
count	49810.000000	49812.000000	49812.000000	49812.000000
mean	38.270167	56.169678	36.294056	-119.859815
std	24.750558	30.486150	2.235560	2.099002
min	0.000000	1.000000	0.000000	-124.203470
25%	21.000000	29.000000	34.144350	-121.618549
50%	35.000000	65.000000	36.487823	-119.828400
75%	52.000000	77.000000	37.960400	-118.147294
max	537.000000	113.000000	41.756130	0.000000

```
print(aq['COUNTY'].nunique())
Output: 51
```

2. [3 Pts] Select all that apply.

   ☐ **A. Supposing that there is a one to one mapping from COUNTY_CODE to COUNTY, we can extrapolate the value of COUNTY for index 4.**

   ☐ B. Grouping by COUNTY is equivalent to grouping by LAT, LON.

   ☐ C. The primary key in this dataset is the DATE.

   ☐ D. None of the above

Solution:
A: True
B: No, there are different latitude and longitudes for a county
C: Dates are not unique.

For all following questions, assume we have finished cleaning the dataset (filled in or removed missing, NaN, etc.).

3. [2 Pts] Which of the following correctly verifies that the mapping from COUNTY_CODE to COUNTY is 1 to 1? Select only one.

   ◯ A. `len(aq['COUNTY'].value_counts()) == len(aq['COUNTY_CODE'].value_counts())`

   ◯ B. `len(set(aq['COUNTY'])) == len(set(aq['COUNTY_CODE']))`

   ◯ C. `len(aq['COUNTY'].unique()) == len(aq['COUNTY_CODE'].unique())`

   ◯ **D.** `len(aq.groupby(['COUNTY', 'COUNTY_CODE'])) == len(set(aq['COUNTY'])) and len(set(aq['COUNTY'])) == len(set(aq['COUNTY_CODE']))`

   ◯ E. None of the above

Solution:
A-C: Having 51 unique COUNTY values and 51 COUNTY_CODE values does not imply a 1 to 1 mapping
D: Correct

4. [2 Pts] In the questions below, select the best plot to visualize a certain aspect of our data.

   (a) visualize the AQI for Los Angeles, San Diego, San Francisco, Humboldt, and Inyo counties over the first 7 days of January 2017.

   ◯ A. Stacked bar plot

   ◯ **B. Side by side line plot**

   ◯ C. KDE plot

- ◯ D. Side by side violin plot

(b) visualize the distribution of site locations by latitude and longitude.

- ◯ A. Histogram
- [✓ **B. Scatter plot**]
- ◯ C. Bar plot
- ◯ D. 1D KDE plot

(c) visualize the average AQI over all counties for each day of January.

- ◯ A. Overlaid line plot
- [✓ **B. Line plot**]
- ◯ C. Side by side histogram
- ◯ D. Side by side box plot

5. [9 Pts] We wish to visualize the mean AQI measurements taken for Alameda, San Francisco and Yolo county over the entire period. Fill in the code below to accomplish this. Use choices from the following table.

aq	'Date'	:	'AQI'	'COUNTY_CODE'
'COUNTY'	'LAT'	'LON'	Alameda	San Francisco
Yolo	str	apply	match	groupby
agg	findall	count	sum	mean
|	==	or	and	filter

```
reg = r'__________ __________ __________ __________ __________'
temp = (__________
        .loc[
                __________[__________].str.__________(__________),
                __________
        ]
        .__________(__________)
        .__________()
        .reset_index())
sns.barplot(x = __________, y=__________, data=data);
```

Solution:

```
reg = r'Alameda|San Francisco|Yolo'
data = aq.loc[aq['COUNTY'].str.match(reg), :].groupby('COUNTY').mean().res
sns.barplot(x ='COUNTY', y='AQI', data=temp);
```