Data 100, Spring 2024

Discussion #11

Note: Your TA will probably not cover all the problems. This is totally fine, the discussion worksheets are not designed to be finished within an hour. They are deliberately made slightly longer so they can serve as resources you can use to practice, reinforce, and build upon concepts discussed in lectures, labs, and homework.

SQL Syntax

All SQL queries should follow this basic framework. Note that the order of the clauses matter.

```
SELECT [DISTINCT] ___<columns>___
FROM ___<tables>___
[WHERE ___<predicate>___]
[GROUP BY ___<columns>___]
[HAVING ___<predicate>___]
[ORDER BY ___<columns>___]
[LIMIT ___<number of rows>___]
```

1. For this question, we will be working with the UC Berkeley Undergraduate Career Survey dataset, named `survey`. Each year, the UC Berkeley Career Center surveys graduating seniors for their plans after graduating. Below is a sample of the full dataset that contains many thousands of rows.

	j_name	c_name	c_location	m_name
0	Llama Technician	Google	Mountain View	Applied Math
1	Software Engineer	Salesforce	SF	ORMS
2	Open Source Maintainer	Github	SF	Computer Science
3	Big Data Engineer	Microsoft	Redmond	Data Science
4	Data Analyst	Startup	Berkeley	Data Science
5	Analyst Intern	Google	SF	Philosophy

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

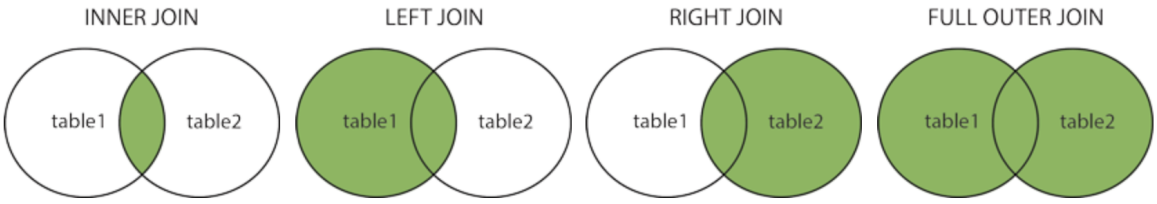
   (a) Write an SQL query that contains all data science major graduates who got jobs in Berkeley. The result generated by your query should include all 4 columns.

   ```
   _____ FROM survey
   _____
   _____
   ```

(b) Write an SQL query to find the top 2 most popular companies that data science graduates will work at, from most popular to 2nd most popular.

```
SELECT c_name, _____________ AS count
FROM survey
WHERE _____________ = 'Data Science'
GROUP BY _____________
ORDER BY _____________
LIMIT 2;
```

Joins



Note: You do not need the JOIN keyword to join SQL tables. The following are equivalent:

```
SELECT column1, column2
FROM table1, table2
WHERE table1.id = table2.id;
```

```
SELECT column1, column2
FROM table1 JOIN table2
ON table1.id = table2.id;
```

2. In the figure above, assume 'table1' has m records, while 'table2' has n records. Describe which records are returned from each type of join. What is the **maximum** possible number of records returned in each join? Consider the cases where on the joined field, (1) both tables have unique values, and (2) both tables have duplicated values. Finally, what is the **minimum** possible number of records returned in each join?

3. Consider the following real estate schema (underlined column names have unique values and no duplicates):

```
Homes(home_id int, city text, bedrooms int, bathrooms int,
area text)
Transactions(home_id int, buyer_id int, seller_id int,
transaction_date date, sale_price int)
Buyers(buyer_id int, name text)
Sellers(seller_id int, name text)
```

Fill in the blanks in the SQL query to find the home_id, selling price, and area for each home in Berkeley with an area greater than 600. If the home has not been sold yet and has an area greater than 600, it should still be included in the table with **the price as None**.

```
SELECT _____________________________________
FROM _____________________
_________ JOIN  _____________________
ON _____________________________
WHERE _____________________________
AND _____________________________________;
```

More SQL Queries

Note: The database for this question in the Discussion 11 Jupyter notebook is set up incorrectly - in particular, owner_id is specified as an integer above but is of text type in the notebook. The solutions presented below do not necessarily work in the notebook for that reason, please disregard the Question 4 section of the notebook.

4. Examine this schema for these two tables:

```
CREATE TABLE cat_owners (
    id integer,
    name text,
    age integer,
    PRIMARY KEY (id)
);
```

```
CREATE TABLE cats (
id integer,
owner_id integer,
name text,
breed text,
age integer,
PRIMARY KEY (id),
FOREIGN KEY (owner_id)
        REFERENCES cat_owners
);
```

   (a) Write an SQL query to create an almost identical table as cats, except with an additional column 'Nickname' that has the value 'Kitten' for cats less than or equal to the age of 1, 'Catto' for cats between 1 and 15, and 'Wise One' for cats older than or equal to 15.

   (b) Considering only cats with ages strictly greater than 1, write an SQL query that returns the owner_ids of owners that own more than one cat.

(c) Write an SQL query to figure out the owner_id of the one cat owner who owns the most cats.

(d) Write an SQL query to figure out the names of all of the cat owners who have a cat named Pishi.

(e) Is it possible to have a cat with an owner_id not in the cat_owners table? Explain your answer.

◯ A. True ◯ B. False

(f) Write an SQL query to select all rows from the `cats` table that have cats of the top 2 most popular cat breeds.