CS 88 Fall 2018 John Yang
Prof. David Culler Friday 2-4 PM

SQL

Link to Discussion Slides: https://goo.gl/kpa8ua

This is one of the last topics on SQL, so congratulations on making it so far! This last section of CS 88 introduces a style of programming that is a bit of a deviation from what we're traditionally used to with regards to what we've been writing in Python.

1 Imperative vs. Declarative

Up until this point in the class, we've been writing in a style known as "imperative programming". Python itself is an imperative programming language. What this means is that, to the computer, the code that we're writing is understood as a set of commands and instructions that the computer is expected to perform.

For instance, a real life example of imperative programming would be a Lego instruction manual! When you tear open a new box of the latest Lego Star Wars set, the instruction manual is a direct set of step by step instructions that tell you what pieces to put together.

On the other hand is programming style known as **declarative** programming. In this style of programming, the focus is on *what* the program should accomplish and what result it should generate, as opposed to the *how* to achieve the result within imperative programming.

Let's go back to the analogy. In this case, declarative programming would be akin to someone given you a pile of Legos and a picture, and someone tells you to recreate the picture using Legos. In this case, there's no formal set of commands or directions that tell you *how* to recreate the picture. The only thing you know is *what* to recreate.

2 SQL (Structured Query Language)

Introducing, SQL! SQL is a language that is most commonly used these days to communicate with a database in the form of storing, manipulating, and retrieving data. The reason we're introducing SQL in CS88 is for a couple of reasons.

First and foremost, we're a data science class, and in the context of this field, SQL is a fantastic language to learn that will familiarize yourself with how many modern-day company databases work. Secondly, the declarative nature of SQL provides a nice contrast to the imperative nature of Python, and this disparity encourages you to think about how the purpose of different languages affects their respective syntax and styles.

3 SQL Commands

While there's a whole host of SQL commands out there, for this class, you will only really need to understand six different kinds of SQL commands. The following table tells you the name of the command, the kinds of arguments it takes, and its purpose when it comes to defining the expected results of a search query.

Command	**Arguments**	**Description**
SELECT	<column name>	Select desired column(s) of a table by column name.
FROM	<table name>	Select desired table to retrieve columns and data from.
WHERE	<conditions>	Select row values using conditions on columns' values.
ORDER BY	<column>[DESC]	Specify order of results by column(s).
LIMIT	<number>	Specify number of results to display from search query.
DISTINCT	<column>	Choose unique values of the column, removes duplicates.
Aliases	<table name>AS <alias>	Specify an alias that can reference a table's data.
AND, OR	WHERE	Write 2+ conditions, relate them using Boolean operators.

4 SQL Practice Problems

The best way to learn SQL is with some practice! Below is a table that we can title as "cities" for now. Using this table and your knowledge of SQL commands, answer each of the queries below!

latitude	**longitude**	**name**	**country**	**population**
30	40	Berkeley	USA	300000
-170	60	New York	USA	1000000
-20	190	Shanghai	China	1500000
15	-10	London	UK	700000

Write a query that lists the location (latitude, longitude) of each city in the table.

SELECT latitude, longitude FROM cities;

Write a query that lists the name of all cities located in the USA.

SELECT name FROM cities WHERE country = 'USA';

Write a query that lists the name and population of each city listed from greatest to least population. Only include cities that are located in the Northern Hemisphere (latitude greater than 0).

SELECT name, population FROM cities WHERE latitude > 0 ORDER BY population DESC;

For the previous query, what kind of output do you expect? Write the output row by row on the lines below. If there is no output generated, write "no output".

London, 700000

Berkeley, 300000

Write a query that lists the name and country of a city, only including cities where either the population is greater than or equal to a million, or the city is located in the Southern Hemisphere and has a longitude greater than 0. Only include countries located in the U.S., and list the results in order of latitude from greatest to least.

SELECT city, country FROM cities WHERE population >= 1000000 OR (latitude < 0 AND longitude > 0) AND country = 'USA' ORDER BY latitude DESC;

For the previous query, what kind of output do you expect? Write the output row by row on the lines below. If there is no output generated, write "no output".

No output generated.

5 Joins

Hopefully in the previous set of problems, you gained a bit of insight into how SQL could be extremely useful in the future when working with real databases and more extensive values. Imagine how many *for* loops you'd need to write for every *where* condition in a query! Declarative languages because it masks the functionality and provides users with the luxury just saying exactly what they want to have happen.

Similarly, in the real world, you may encounter situations where you want to combine data between two different tables or on the same table itself by using aliases. At its core, a join allows you to create an amalgamate of the data from multiple tables.

A join is implicitly execute when there is more than one table listed in the FROM clause. Under the hood, two tables are combined through a cross product. The cross product matches every row from the first table with every row in the second table. If Table A has 3 rows and Table B has 4 rows, a cross product will yield a table with 12 rows, since each row in Table A is matched with 4 rows from Table B, so 3 x 4 = 12. Click here for a straightforward diagram of what joins look like. To learn more about the inner workings of SQL and joins, take CS 186!

6 Join Practice Problems

To practice joins, let's introduce a new table that we'll call "facts". Use this table and the "cities" table from before to answer the questions below.

mayor	**name**	**geography**
Paul	Berkeley	Rolling Hills
Fred	New York	Urban
Michelle	Shanghai	Coastal Climate
Jasmine	London	Inland Grassland

Write a query that lists the name, population, and geography of each city.

```
SELECT a.name, a.population, b.geography FROM cities as a, facts as b WHERE a.name = b.name;
```

Write a query that lists the mayor, country, and population of each city for cities not located in the USA.

```
SELECT b.mayor, a.country, a.population FROM cities as a, facts as b
WHERE a.name = b.name AND a.country != 'USA';
```