SQL Solutions

CS 61A

August 6, 2021

1 SQL

SQL Overview

SQL (Standardized Query Language) is a declarative programming language that allows us to store, access, and manipulate data stored in databases. Each database contains tables, which can store many rows of data that all share the same properties (columns).

To create a table, we can use the `CREATE TABLE` operation. For example, if we want to make a table with 2 columns 'name' and 'number' and fill it with 3 rows of data, we could do the following:

```
CREATE TABLE numbers AS
    SELECT "Papa John's Pizza" AS name, 5108457272 AS number
       UNION
    SELECT "UCPD", 5106426760 UNION
    SELECT "Foothill Mailroom", 5106429703;
```

We can then filter and aggregate data using queries which have the following general structure:

```
SELECT col1, col2, ... FROM table WHERE conditions GROUP BY
   column HAVING conditions ORDER BY column [DESC] LIMIT num;
```

1. SELECT chooses specific columns to include in the output. Column names can be changed using the AS operation (for example, `SELECT number as phone` would rename the number column to 'phone'.)
2. FROM chooses which table(s) to select data from. If multiple tables are included, then they are joined together such that every possible combination of rows are outputted. The same table can also be joined to itself if aliasing is used (e.g. `SELECT * FROM numbers as a, numbers as b`).
3. WHERE restricts which rows appear in the output. Valid conditions include less than/greater than/equal to (<, >,), AND/OR, and not equal (<>). All comparisons involving aggregations (for example, `COUNT(*) > 1`) must go in the HAVING clause instead of WHERE.
4. GROUP BY aggregates the table by combining all rows with the same value into one group. Properties of this group can then be accessed using COUNT, MIN, MAX, etc.
5. ORDER BY sorts rows using the values of the specified column. If the DESC keyword is included, then rows will be sorted from largest to smallest.
6. LIMIT restricts the maxiumum number of rows in the output table. This is most often used with ORDER BY to get the top 10 entries, for example.

CS 61A wants to start a fish hatchery, and we need your help to analyze the data we've collected for the fish populations! Running a hatchery is expensive – we'd like to make some money on the side by selling some seafood (only older fish of course) to make delicious sushi.

The table `fish` contains a subset of the data that has been collected. The SQL column names are listed in brackets.

Table name: `fish`*

Species [species]	Population [pop]	Breeding Rate [rate]	$/piece [price]	# of pieces per fish [pieces]
Salmon	500	3.3	4	30
Eel	100	1.3	4	15
Yellowtail	700	2.0	3	30
Tuna	600	1.1	3	20

*(This was made with fake data, do not actually sell fish at these rates)

1. Write a query to find the three most populated fish species.

```
SELECT species FROM fish ORDER BY -pop LIMIT 3;
```

2. Write a query to find the total number of fish in the ocean. Additionally, include the number of species we summed. Your output should have the number of species and the total population.

```
SELECT COUNT(species), SUM(pop) FROM fish;
```

3. Profit is good, but more profit is better. Write a query to select the species that yields the most number of pieces for each price. Your output should include the species, price, and pieces.

```
SELECT species, price, MAX(pieces) FROM fish GROUP BY price;
```

Business is good, but a bunch of competition has sprung up! Through some cunning corporate espionage, we have determined one such competitor's selling prices. The table `competitor` contains the competitor's `price` for each `species`.

Species [species]	**$/piece [price]**
Salmon	2
Eel	3.4
Yellowtail	3.2
Tuna	2.6

1. Write a query that returns, for each species, the difference between our hatchery's revenue versus the competitor's revenue for one whole fish.

```
SELECT fish.species, (fish.price - competitor.price) * pieces
    FROM fish, competitor
    WHERE fish.species = competitor.species;
```

For the following two questions, you have access to two tables.

Grades, which contains three columns: `day`, **`class`**, and `score`. Each row represents the `score` you got on a midterm for some **`class`** that you took on some `day`.

Outfits, which contains two columns: `day` and `color`. Each row represents the `color` of the shirt you wore on some `day`. Assume you have a row for each possible day.

Table name: `grades`

Day	Class	Score
10/31	Music 70	88
9/20	Math 1A	72

Table name: `outfits`

Day	Color
11/5	Blue
9/13	Red
10/31	Orange

1. You want to find out which classes you need to prepare for the most by determining how many points you have so far. However, you only want to do so for classes where you did relatively poorly

   Write a query that will output the sum of your midterm scores for each class along with the corresponding class, but only for classes in which you scored less than 80 points on at least one midterm. List the output from highest to lowest total score.

   ```
   SELECT SUM(score), class
       FROM grades GROUP BY class
       HAVING MIN(score) < 80 ORDER BY -SUM(score);
   ```

2. Instead of actually studying for your finals, you decide it would be the best use of your time to determine what your "lucky shirt" is. Suppose you're pretty happy with your exam scores this semester, so you define your lucky shirt as the shirt you wore to the most exams.

   Write a query that will output the color of your lucky shirt and how many times you wore it.

   ```
   SELECT color, count(g.day) AS cnt
       FROM outfits AS o, grades AS g
       WHERE o.day = g.day
       GROUP BY color
       ORDER BY cnt desc
       LIMIT 1;
   ```

After more than 100 years of operation, the Ringling Bros. circus is closing. A victory for animal rights advocates, the circus' closure poses a challenge for the zoologists tasked with moving the circus' animals to more suitable habitats.

The zoologists must first take the animals in a freight elevator with a weight limit of 2000. In order to speed up the process, the zoologists prefer to take groups of animals of the same species in the elevator, rather than one animal at a time.

Assume the zoologists will only put all of the animals of a particular species in the elevator, or take animals of that particular species one at a time.

You have access to the table `animals`, with columns containing the animals' names, heights, weights, and species.

Name	Height	Weight	Species
Wilbur	4.1	150	pig
Tigress	4.4	700	tiger
Phil	3.3	79	pig
Dug	3.5	40	dog
Buddy	4	51	dog
Marty	4.9	300	zebra
Richard Parker	5.2	918	tiger

1. Write a query that returns the collective weight and species of animals in a group where there is more than one animal of a particular species in a group, and the collective weight of the animals in the group is less than 2000.

   Your query should yield the following result.

```
91 dog
229 pig
1618 tiger
```

```
SELECT SUM(weight), species FROM animals
    GROUP BY species HAVING COUNT(*) > 1 and SUM(weight) <
       2000;
```

2. To take the animals to their new habitats, the zoologists load the animals into trucks. The zoologists would like to reduce the number of trips taken by pairing up the animals for the trip (species does not matter). However, the trucks have two restrictions:

   1. The maximum height of any animal is 5.0.
   2. The total weight of both animals cannot exceed 300.

   Your query should yield the following result. To match the table below, make sure that the left column animals' names come earlier in the alphabet than their partners.

```
Buddy     Dug
Buddy     Phil
Buddy     Wilbur
Dug       Phil
Dug       Wilbur
Phil      Wilbur
```

```
SELECT a.name, b.name
    FROM animals AS a, animals AS b
    WHERE a.name < b.name AND a.weight + b.weight <= 300 AND a
       .height <= 5 AND b.height <= 5;
```