Nicole Lawrence
Database Design & Development
CS225 Week2 Ponder Assignment

A Restaurant Analogy for SQL Commands

When visiting a restaurant, customers enter, request a table, and order their favorite dishes. The customer, while being able to complete each of these steps to ensure a delicious meal, is ignorant of the actions taken behind the scenes in response to these steps. Interacting with a database is very similar to this restaurant scenario. Any user is abstracted away from the internal workings of the server and has to give very specific SQL directions (queries) to retrieve data results. Queries are begun and ended with the SELECT clause in that it signals to the database the beginning of a query continuing to the semicolon; after evaluating all the conditions requested, the SELECT statement is evaluated, signaling the end of the query. At this point, the server returns the requested data to the user.

To continue with the restaurant analogy, a customer desiring wine would request a bottle. If there's a preference for a certain region of origin, the customer mentions it in the request. This is equivalent, roughly, to the FROM clause in SQL which instructs the server which data table to access. If a certain year of wine is preferred, the restauranteur will use that information in his search for a suitable range of choices. This parallels the WHERE clause which details what data should or should not be included in the returned results. Here, the natural language operators of AND, OR, and NOT constrain and filter the data. If AND, all conditions are required to be true. OR indicates that only one of the conditions must be true and NOT is a non-negotiable filter. For example, if 'the cat is hungry AND the dog is hungry' limits the query to data where the cat and the dog are both hungry. If 'the cat is hungry OR the dog is hungry', data where a cat, a dog, or both a cat and dog are hungry is acceptable. An example of NOT: if 'the cat is hungry NOT the dog is hungry', it doesn't matter how hungry the dog gets, the only data returned relates to a hungry cat. While a dining analogy would be more appropriate here, these examples are very clear and easy to understand with the AND, OR, and NOT operators.

The final conditions that bear discussion control the ordering, filtering, and display of the results gathered. GROUP BY groups rows together by common values. Returning to our restaurant, this is equivalent to seating diners by certain characteristics, the most common being smoking and non-smoking. Although, this is the *only* legal way to separate customers, thankfully. HAVING is a final filter for the gathered data, a last chance to refine the results. This might correspond to throwing out an obnoxious diner who is disrupting the dining experience for everyone else. Or upgrading a local celebrity to a private dining room. Finally, if the customer requests that all the staff bringing their meal wear lederhosen, this would be the same as using the ORDER BY clause. As absurd as this example is, using ORDER BY dictates how the data retrieved, after conditions and filters, is ultimately presented to the user. In the case of a database, you would usually sort by last and/or first name, dates, etc. Although lederhosen could conceivably be an ORDER BY parameter in a very odd database.

A quick note: for readability purposes, column titles can be renamed with aliases. This creates results that are clear and unambiguous. These can be added without any notation but using AS makes the aliases clear to future readers of the code. To complete the analogy, aliases could be compared to the practice of calling meat from a pig, pork or raw meat, tartare.