Advanced SQL

Sean Sall

May 11th, 2016

Today's Objectives:

- Be **aware** of a little bit of the advanced functionality available in SQL

Agenda

- Morning questions/review
- **UNION**, **EXCEPT**, and **INTERSECTION**
- **Window** Functions

Why does this matter?

- If you're interviewing for or working at a SQL-heavy job, this advanced SQL functionality can give you a leg up and/or make your life a little easier.

  - Technically, you can get all of the functionality of this "advanced" stuff out of what you already know.

SQL Tables

Recall the tables we're working with:

```
CREATE TABLE disney_chars (
    id INTEGER PRIMARY KEY,
    name VARCHAR(255),
    princess BOOLEAN,
    prince BOOLEAN,
    mulan BOOLEAN
);

CREATE TABLE users (
    id INTEGER PRIMARY KEY,
    name VARCHAR(255),
    nickname VARCHAR(255),
    fav_disney_char INTEGER references disney_chars(id),
    tswift_rating INTEGER
);
```

UNION, EXCEPT, INTERSECT

UNION, **EXCEPT**, and **INTERSECT** allow us to combine multiple queries with a `set-like` syntax.

- Let's grab the names of those Disney characters that **don't** have an equivalent in our `users` table:

```
SELECT name FROM disney_chars
EXCEPT SELECT name FROM users;
```

- Let's grab the names of those Disney characters that **do** have an equivalent in our `users` table:

```
SELECT name FROM disney_chars
INTERSECT SELECT name FROM users;
```

Window Functions

A **window** function performs a calculation across a set of table rows that are somehow related to the current row.

- Compute each person's `tswift_rating` with the average rating for users with the same `fav_disney_char`:

```
SELECT name, fav_disney_char, tswift_rating,
    AVG(tswift_rating) OVER (PARTITION BY fav_disney_char)
FROM users;
```

- Rank each of the users by their `tswift_rating`, but **WITHIN** their `fav_disney_char` group:

```
SELECT name, fav_disney_char, tswift_rating,
    rank() OVER (PARTITION BY fav_disney_char
                 ORDER BY tswift_rating)
FROM users;
```