SQL Tale Transformation Set Operations

Unions allow us to utilize information from multiple tables in our queries.

Union combines the result of two or more SELECT statements, using the following syntax:

```
SELECT column_name(s) FROM table1
UNION
SELECT column_name(s) FROM table2;
```

Each SELECT statement within the UNION must have the same number of columns with similar data types. **The columns in each SELECT statement must be in the same order.** By default, **the UNION operator selects only distinct values.**

Suppose we are a growing ecommerce store and recently acquired another store to diversify our offering. The product data still exists in two separate tables: a legacy_products table and a new_products table. To get the complete list of product names from both tables, we can perform the following union.

```
SELECT item_name FROM legacy_products
UNION
SELECT item_name FROM new_products;
```

What if we wanted to **allow duplicate values**? We can do this by using the ALL keyword with UNION, with the following syntax:

```
SELECT column_name(s) FROM table1
UNION ALL
SELECT column_name(s) FROM table2;
```

We can perform an analysis on top of the combined result set, like finding the total count of order items.

```
SELECT count(*) FROM (
  SELECT id, sale_price FROM order_items
  UNION ALL
  SELECT id, sale_price FROM order_items_historic) as a;
```

or **find the average sale price** over both order_items and order_items_historic tables.

```
SELECT id, avg(a.sale_price) FROM (
SELECT id, sale_price FROM order_items
UNION ALL
SELECT id, sale_price FROM order_items_historic) AS a
```

GROUP BY 1;

INTERSECT is used to combine **two SELECT statements,** but **returns rows only from the first SELECT** statement that are identical to **a row in the second SELECT** statement. This means that it **returns only common rows** returned by the two SELECT statements.

SELECT column_name(s) FROM table1
INTERSECT
SELECT column_name(s) FROM table2;

For instance, we might want to know what brands in our newly acquired store are also in our legacy store. We can do so using the following query:

SELECT brand FROM new_products
INTERSECT
SELECT brand FROM legacy_products;

EXCEPT is constructed in the same way, but **returns distinct rows from the first SELECT** statement **that aren't output by the second SELECT** statement.

SELECT column_name(s) FROM table1
EXCEPT
SELECT column_name(s) FROM table2;
//select category for items that are in legacy_products, and are not in new_products
SELECT category FROM legacy_products
EXCEPT
SELECT category FROM new_products;

- **The UNION clause allows us to utilize information from multiple tables in our queries.**
- **The UNION ALL clause allows us to utilize information from multiple tables in our queries, including duplicate values.**
- **INTERSECT is used to combine two SELECT statements, but returns rows only from the first SELECT statement that are identical to a row in the second SELECT statement.**
- **EXCEPT returns distinct rows from the first SELECT statement that aren't output by the second SELECT statement**