EXISTS, NOT EXISTS

- Used in a subquery to check if rows exists (or not) matching the subquery's SELECT statement.

EXISTS: Get all recipes that use onions

```
select * from Recipes
where exists (
      select * from Recipe_Ingredients
      inner join Ingredients
            on Recipe_Ingredients.IngredientID =
                  Ingredients.IngredientID
      where Ingredients.IngredientName = 'Onion'
      and Recipe_Ingredients.RecipeID = Recipes.RecipeID
);
```

NOT EXISTS: Get all recipes that do not use onions

```
select * from Recipes
where not exists (
      select * from Recipe_Ingredients
      inner join Ingredients
            on Recipe_Ingredients.IngredientID =
                  Ingredients.IngredientID
      where Ingredients.IngredientName = 'Onion'
      and Recipe_Ingredients.RecipeID = Recipes.RecipeID
);
```

Subqueries with ANY, SOME, ALL

- https://dev.mysql.com/doc/refman/8.0/en/with.html

Aggregate functions

- COUNT
- MAX
- MIN
- AVG
- SUM
- STDDEV
- …

Aggregate functions

```
use sakila;

select max(amount),
    min(amount),
    avg(amount),
    sum(amount)
from payment;
```

Aggregate functions

```
use sakila;

-- get the most recent payment date
select max(payment_date) from payment;

-- get the first payment date
select min(payment_date) from payment;
```

Aggregate functions

```
use sakila;

-- get all payments on the most recent payment
-- date
select * from payment
where payment_date = (
    select max(payment_date) from payment
);
```

Aggregate functions

```
use sakila;

-- get all payments that were above average
select * from payment
where amount > (
    select avg(amount) from payment
);
```

The GROUP BY clause

```
-- get all aggregates GROUPED BY customer

select customer_id,
     max(amount) as maximum_payed,
     min(amount) as minimum_payed,
     avg(amount) as average_payed,
     sum(amount) as total_payed,
     max(payment_date) as most_recent_payment_date,
     min(payment_date) as first_payment_date
from payment
group by customer_id;
```

The GROUP BY clause

```
-- get the count of films grouped by rating
-- order by is optional

select rating, count(film_id) from film
group by rating
order by rating;
```

The GROUP BY clause

```
-- get the count of films grouped by rating AND
-- special_features

select rating, special_features, count(film_id)
from film
group by rating, special_features
order by rating;
```

Exercises

- Using **sakila**
  - Use COUNT() and GROUP BY to get the number of customers in each store
  - Get the number of cities (using the **city** table) grouped by country
- Using **RecipesExample**
  - Get the number of recipes in each recipe class