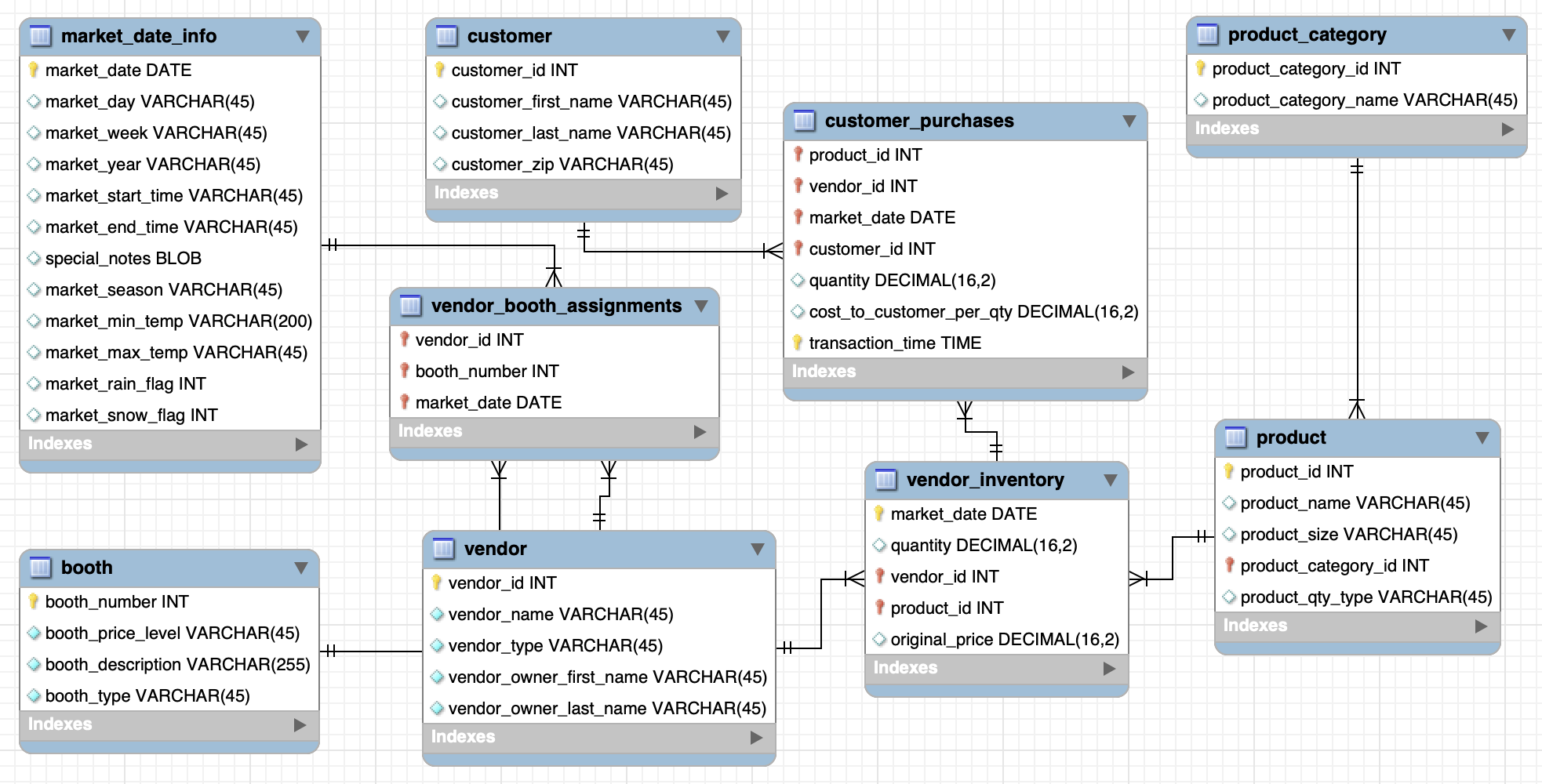
Filtering contd. and Subqueries

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Problem Statement:

You are a Data Analyst at Amazon Fresh. You have been tasked to study the Farmer’s Market.

Dataset: Farmer’s Market database



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

There can be situations when you have multiple duplicate records in a table. While fetching such records, it makes more sense to fetch only the unique ones.

In that case, you can use the **DISTINCT** keyword in conjunction with the SELECT statement to **eliminate all the duplicate rows**.

**Syntax:**

SELECT DISTINCT col1, col2, …

FROM DB.table

Even if you want to fetch the **unique** values from a single column, you can do so using the DISTINCT keyword.

### Questions: Your manager wants to see all the **unique** customer IDs present in the `customer\_purchases` table. How would you get this data?

SELECT

DISTINCT customer\_id

FROM farmers\_market.customer\_purchases

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Order of Execution** of a SQL query :

* **FROM** - The database gets the data from tables in FROM clause.
* **WHERE** - The data is filtered based on the conditions specified in the WHERE clause. Rows that do not meet the criteria are excluded.
* **SELECT** - After filtering is done, the SELECT statement determines which columns to include in the final result set.
* **DISTINCT** - The DISTINCT keyword is applied within the SELECT clause to ensure that only unique values are returned for the specified columns.
* **ORDER BY** - It allows you to sort the result set based on one or more columns, either in ascending or descending order.
* **OFFSET** - The specified number of rows are skipped from the beginning of the result set.
* **LIMIT** - After skipping the rows, the LIMIT clause is applied to restrict the number of rows returned.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### Question: Find all of the products from the `product` table which don’t have their sizes mentioned.

The absence of any value in a cell is represented by the **NULL** keyword.

SELECT \*

FROM farmers\_market.product

WHERE product\_size IS NULL

NOTE: Keep in mind that **“blank”** and **NULL** are not the same thing in database terms.

* If someone asked you to find all products that didn’t have product sizes, you might also want to check for blank strings, which would equal “”, or rows where someone entered a space or any number of spaces into that field.
* The **TRIM()** function **removes excess spaces from the beginning or end of a string value**, so if you use a combination of the TRIM() function and blank string comparison, you can find any row that is blank or contains only spaces.

SELECT \*

FROM farmers\_market.product

WHERE

product\_size IS NULL

OR TRIM(product\_size) = “”

* You might wonder why the comparison operator IS NULL is used instead of equals NULL just like numbers.
* **NULL is not actually a value, it’s the absence of a value, so it can’t be compared to any existing value.**
* If you wanted to return all records that don’t have NULL values in a field, you could use the condition “[field name] **IS NOT NULL**” in the WHERE clause.

SELECT \*

FROM farmers\_market.product

WHERE

product\_size IS NOT NULL

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Question: What if you’re asked to fetch all the product IDs from the `products` table and set a default value “medium” in rows where the product size is NULL?

We can use the **IFNULL()** function to check for null values in a column and if found, replace them with some other value.

SELECT

product\_id,

IFNULL(product\_size, “medium”)

FROM farmers\_market.product

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Subqueries**

Question: Analyze purchases made at the market on days when it rained.

There is a value in the *market\_date\_info* **table** called *market\_rain\_flag* that has a value of 0 if it didn’t rain while the market was open and a value of 1 if it did.

* 0 - it didn’t rain
* 1 - it did

First, let’s write a query that gets a list of market dates when it rained, using this query:

SELECT market\_date, market\_rain\_flag

FROM farmers\_market.market\_date\_info

WHERE market\_rain\_flag = 1

Now let’s use the list of dates generated by that query to return purchases made on those dates.

SELECT \*

FROM `farmers\_market.customer\_purchases`

WHERE market\_date IN (SELECT market\_date

FROM `farmers\_market.market\_date\_info`

WHERE market\_rain\_flag = 1 )

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Question: List down all the product details where product\_category\_name contains “Fresh” in it.

* Capture the list category\_ids where the category\_name contains “Fresh”
* Get the products which belong to the shortlisted category\_ids

SELECT \*

FROM `farmers\_market.product`

WHERE product\_category\_id IN ( SELECT product\_category\_id

FROM `farmers\_market.product\_category`

WHERE LOWER(product\_category\_name) LIKE "%fresh%")

Similarly, you could use the **NOT IN** operator if you want to get the product details where the product\_category\_name does not contain “Fresh”.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**CASE statement**

What if you want to take some action based on a certain condition.

For example -

“If [one condition] is true, then [take this action].

Otherwise, [take this other action].”

That’s where CASE statements come into play.

**Syntax:**

SELECT *cols*,

CASE

WHEN [first conditional statement]

THEN [value or calculation]

WHEN [second conditional statement]

THEN [value or calculation]

ELSE [value or calculation]

END AS *alias*

FROM *table*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**IF() function**

**Syntax:**

IF(condition, true\_value, false\_value)

**Parameters used:**

* **condition** – It is used to specify the condition to be evaluated.
* **true\_value** – It is an optional parameter that is used to specify the value to be returned if the condition evaluates to be true.
* **false\_value** – It is an optional parameter that is used to specify the value to be returned if the condition evaluates to be false.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Question: Find out which vendors primarily sell fresh products and which don’t.

You have to add **“Fresh Produce”** for vendors who sell fresh products and **“Other”** for those who don’t.

SELECT

vendor\_id,

vendor\_name,

vendor\_type,

CASE

WHEN LOWER(vendor\_type) LIKE '%fresh%'

THEN 'Fresh Produce'

ELSE 'Other'

END AS category

FROM farmers\_market.vendor

We can do the same using the **IF() function** as well -

SELECT \*,

IF(LOWER(vendor\_type) LIKE “%fresh%”, “Fresh”, “Not Fresh”) AS type FROM `farmers\_market.vendor`

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

What if we want to add **1** for vendors who sell fresh products and **0** for those who don’t?

SELECT vendor\_id, vendor\_name, vendor\_type,

CASE

WHEN LOWER(vendor\_type) LIKE "%fresh%" THEN 1

ELSE 0

END AS category

FROM `farmers\_market.vendor`

We can do the same using the **IF() function** as well -

SELECT \*,

IF(LOWER(vendor\_type) LIKE “%fresh%”, 1, 0) AS type FROM `farmers\_market.vendor`

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Question: Put the total cost to customer purchases into bins of -

● under $5.00,

● $5.00–$9.99,

● $10.00–$19.99, or

● $20.00 and over.

SELECT

market\_date, customer\_id,

quantity, cost\_to\_customer\_per\_qty,

CASE

WHEN quantity \* cost\_to\_customer\_per\_qty < 5

THEN "Under $5"

WHEN quantity \* cost\_to\_customer\_per\_qty BETWEEN 5 AND 9.99  
 THEN "$5 - $9.99"

WHEN quantity \* cost\_to\_customer\_per\_qty BETWEEN 10 AND 19.99  
 THEN "$10 - $19.99"

ELSE "Above $20"

END AS price\_bin

FROM `farmers\_market.customer\_purchases`

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_