The WHERE Clause: Filtering Results

- The WHERE clause follows the FROM statement.
- It precedes any GROUP BY, ORDER BY, or LIMIT statements in a SELECT query.

```
SELECT
    market_date, customer_id,
    vendor_id, product_id,
    quantity,
    quantity * cost_to_customer_per_qty AS price
FROM farmers_market.customer_purchases
WHERE customer_id = 4
ORDER BY market_date, vendor_id, product_id
LIMIT 5
```

Table 3.1

market_date	customer_id	vendor_id	product_id	quantity	price
2019-04-03	4	7	4	1.00	4.0000
2019-04-06	4	8	5	1.00	6.5000
2019-04-10	4	7	4	3.00	12.0000
2019-04-10	4	7	4	5.00	20.0000
2019-04-10	4	8	7	2.00	36.0000

- The customer_id values in the table are integers, not strings.
- If customer_id values were strings, the comparison value need to be a string, meaning '4'.

Filtering on Multiple Conditions

- Combine multiple conditions with "AND," "OR," or "AND NOT".
- The query for filtering customer id 4 or 3 would be:

```
SELECT
    market_date, customer_id,
    vendor_id, product_id,
    quantity,
    quantity * cost_to_customer_per_qty AS price
FROM farmers_market.customer_purchases
WHERE customer_id = 3 OR customer_id = 4
ORDER BY market_date, customer_id, vendor_id, product_id
LIMIT 5
```

Table 3.2

market_date	customer_id	vendor_id	product_id	quantity	price
2019-04-03	3	7	4	1.00	4.0000
2019-04-03	4	7	4	1.00	4.0000
2019-04-06	4	8	5	1.00	6.5000
2019-04-10	4	7	4	5.00	20.0000
2019-04-10	4	7	4	3.00	12.0000

- What happens if the WHERE clause condition is "customer_id = 3 AND customer_id = 4"?
  - This means "Return each row where the customer ID is 3 and 4."
  - Since a single customer_id cannot be both 3 and 4, no rows are returned.
- When using the AND operator:
  - All conditions with AND must be TRUE for a row to be returned.
  - Example: "WHERE customer_id > 3 AND customer_id <= 5."

```
SELECT
    market_date, customer_id,
    vendor_id, product_id,
    quantity,
    quantity * cost_to_customer_per_qty AS price
FROM farmers_market.customer_purchases
WHERE customer_id > 3 AND customer_id <= 5
ORDER BY market_date, customer_id, vendor_id, product_id
LIMIT 5
```

Table 3.3

market_date	customer_id	vendor_id	product_id	quantity	price
2019-04-03	4	7	4	1.00	4.0000
2019-04-03	5	7	4	3.00	12.0000
2019-04-03	5	8	8	1.00	18.0000
2019-04-06	4	8	5	1.00	6.5000
2019-04-06	5	7	4	1.00	4.0000

- You can combine multiple AND, OR, and NOT conditions.
- Use parentheses to control their evaluation order.
- Conditions inside parentheses are evaluated first.

```
SELECT product_id, product_name
FROM farmers_market.product
WHERE
    product_id = 10
    OR (product_id > 3
    AND product_id < 8)
```

Table 3.4

product_id	product_name
4	Banana Peppers - Jar
5	Whole Wheat Bread
6	Cut Zinnias Bouquet
7	Apple Pie
10	Eggs

```
SELECT product_id, product_name
FROM farmers_market.product
WHERE
    (product_id = 10
    OR product_id > 3)
    AND product_id < 8
```

Table 3.5

product_id	product_name
4	Banana Peppers - Jar
5	Whole Wheat Bread
6	Cut Zinnias Bouquet
7	Apple Pie

- The row with product_id 10 is only returned by the first query.

Multi-Column Conditional Filtering

- So far, only one field is refered for conditions.
- WHERE clauses can use values from multiple columns.

```
SELECT
    market_date,
    customer_id, vendor_id,
    quantity * cost_to_customer_per_qty AS price
FROM farmers_market.customer_purchases
WHERE customer_id = 4 AND vendor_id = 7
LIMIT 5
```

Table 3.6

market_date	customer_id	vendor_id	price
2019-07-06	4	7	1.8873
2019-07-10	4	7	14.8887
2019-07-17	4	7	21.1797
2019-08-03	4	7	0.5592
2019-09-04	4	7	15.9372

```
SELECT *
FROM farmers_market.vendor_booth_assignments
WHERE
    vendor_id = 9
    AND market_date <= '2019-04-20'
ORDER BY market_date
```

Table 3.7

vendor_id	booth_number	market_date
9	8	2019-04-03
9	8	2019-04-06
9	8	2019-04-10
9	8	2019-04-13
9	8	2019-04-17
9	8	2019-04-20

More Ways to Filter

BETWEEN

```
SELECT *
FROM farmers_market.vendor_booth_assignments
WHERE
    vendor_id = 9
    AND market_date BETWEEN '2020-09-09' and '2020-09-23'
ORDER BY market_date
```

Table 3.8

vendor_id	booth_number	market_date
9	8	2020-09-09
9	8	2020-09-12
9	8	2020-09-16
9	8	2020-09-19
9	8	2020-09-23

IN

- The first query:

```
SELECT
    customer_id,
    customer_first_name,
    customer_last_name
FROM farmers_market.customer
WHERE
    customer_last_name = 'Diaz'
    OR customer_last_name = 'Edwards'
    OR customer_last_name = 'Wilson'
ORDER BY customer_last_name, customer_first_name
```

- The second query:

```sql
SELECT
    customer_id,
    customer_first_name,
    customer_last_name
FROM farmers_market.customer
WHERE
    customer_last_name IN ('Diaz' , 'Edwards', 'Wilson')
ORDER BY customer_last_name, customer_first_name
```

Table 3.9

customer_id	customer_first_name	customer_last_name
17	Carlos	Diaz
2	Manuel	Diaz
10	Russell	Edwards
3	Bob	Wilson

- Use the IN list comparison to search for a person when unsure of the spelling.

```sql
SELECT
    customer_id,
    customer_first_name,
    customer_last_name
FROM farmers_market.customer
WHERE
    customer_first_name IN ('Renee', 'Rene', 'Renée', 'René', 'Renne')
```

LIKE

- If you know a customer's name starts with "Jer" but aren't sure if it's "Jerry," "Jeremy," or "Jeremiah":
    - the % wildcard represents any number of characters (including none).
    - LIKE 'Jer%' will search for strings that start with "Jer" and have any (or no) characters after "r".

```sql
SELECT
    customer_id,
    customer_first_name,
    customer_last_name
FROM farmers_market.customer
WHERE
    customer_first_name LIKE 'Jer%'
```

Table 3.10

customer_id	customer_first_name	customer_last_name
13	Jeremy	Gruber
18	Jeri	Mitchell

IS NULL

- It's useful to find rows where a field is blank or NULL.

```
SELECT *
FROM farmers_market.product
WHERE product_size IS NULL
```

Table 3.11

product_id	product_name	product_size	product_category_id	product_qty_type
14	Red Potatoes	NULL	1	NULL

- The TRIM() function removes spaces from the beginning or end of a string.
- Use TRIM() and a blank string comparison to find rows that are blank or contain only spaces.

```
SELECT *
FROM farmers_market.product
WHERE
    product_size IS NULL
    OR TRIM(product_size) = ''
```

Table 3.12

product_id	product_name	product_size	product_category_id	product_qty_type
14	Red Potatoes	NULL	1	NULL
15	Red Potatoes - Small	NULL	1	NULL

A Warning About Null Comparisons

- Nothing "equals" NULL, not even NULL.
- This is important for other types of comparisons as well.
- Ideally, the database should prevent the quantity value from being NULL.
- To return all records that don't have NULL values in a field, use the condition "[field name] IS NOT NULL" in the WHERE clause.

Filtering Using Subqueries

```
SELECT
    market_date,
    customer_id, vendor_id,
    quantity * cost_to_customer_per_qty price
FROM farmers_market.customer_purchases
WHERE
    market_date IN
        (
        SELECT market_date
        FROM farmers_market.market_date_info
        WHERE market_rain_flag = 1
        )
ORDER BY market_date
LIMIT 5
```

Table 3.13

market_date	customer_id	vendor_id	price
2019-07-31	3	7	18.4536
2019-07-31	8	7	26.7717
2019-07-31	19	7	25.7931
2019-07-31	22	7	7.4793
2019-07-31	3	7	8.1666

Table 3.13

market_date	customer_id	vendor_id	price
2019-07-31	3	7	18.4536
2019-07-31	8	7	26.7717
2019-07-31	19	7	25.7931
2019-07-31	22	7	7.4793
2019-07-31	3	7	8.1666

Exercises

1. Use the data in Table 3.1. Write a query to return all customer purchases of product IDs 4 and 9.
2. Use the data in Table 3.1. Write two queries:
   - One using two conditions with an AND operator.
   - One using the BETWEEN operator.
   - Return all customer purchases from vendors with vendor IDs between 8 and 10 (inclusive).
3. Think of two ways to change the final query in the chapter to return purchases from days when it wasn't raining.