CASE Statement Syntax

- You use conditional reasoning daily such as "If [condition] is true, then [action]. Otherwise, [other action]."
- The syntax in SQL is as follows:

```
CASE
    WHEN [first conditional statement]
      THEN [value or calculation]
    WHEN [second conditional statement]
      THEN [value or calculation]
    ELSE [value or calculation]
END
```

- The `WHEN` conditions are evaluated from top to bottom.
  - The `ELSE` part is optional.
  - If not included, the result will be `NULL` if no conditions evaluate to `TRUE`.
- You should always alias columns with `CASE` statements.
- To illustrate, consider this query:

```
SELECT
    CASE
        WHEN 1=1 THEN 'Yes'
        WHEN 2=2 THEN 'No'
    END
```

- This query will always evaluate to "Yes" because `1=1` is always `TRUE`.
- The `2=2` conditional statement is never evaluated, even though it is also true.
- Let's label vendors:
  - Vendors with "Fresh" are labeled as "Fresh Produce."
  - Others are labeled as "Other."

```
SELECT
    vendor_id, vendor_name, vendor_type,
    CASE
        WHEN LOWER(vendor_type) LIKE '%fresh%'
          THEN 'Fresh Produce'
        ELSE 'Other'
    END AS vendor_type_condensed -- Alias the column
FROM farmers_market.vendor
```

Table 4.1

vendor_id	vendor_name	vendor_type	vendor_type_condensed
1	Chris's Sustainable Eggs & Meats	Eggs & Meats	Other
2	Hernández Salsa & Veggies	Fresh Variety: Veggies & More	Fresh Produce
3	Mountain View Vegetables	Fresh Variety: Veggies & More	Fresh Produce
4	Fields of Corn	Fresh Focused	Fresh Produce
5	Seashell Clay Shop	Arts & Jewelry	Other
6	Mother's Garlic & Greens	Fresh Variety: Veggies & More	Fresh Produce
7	Marco's Peppers	Fresh Focused	Fresh Produce
8	Annie's Pies	Prepared Foods	Other
9	Mediterranean Bakery	Prepared Foods	Other

- You can use `UPPER()` with '%FRESH%'`.
- If a new vendor type containing `fresh` is added to the database:
  - The query using the `LIKE` comparison will categorize it as `Fresh Produce` in the `vendor_type_condensed` column.
- To restrict the labeling to existing vendor types:
  - Use the `IN` keyword.
  - Explicitly list the vendor types to be labeled as `Fresh Produce`.

Creating Binary Flags Using CASE

- The Farmer's Markets occur on Wednesday evenings or Saturday mornings.

```
SELECT
    market_date,
    CASE
        WHEN market_day = 'Saturday' OR market_day = 'Sunday'
          THEN 1
        ELSE 0
    END AS weekend_flag
FROM farmers_market.market_date_info
LIMIT 5
```

Table 4.2

market_date	weekend_flag
2019-03-02	1
2019-03-09	1
2019-03-13	0
2019-03-16	1
2019-03-20	0

Grouping or Binning Continuous Values

- Indicate whether the cost was over $50:

```sql
SELECT
    market_date,
    customer_id, vendor_id,
    quantity * cost_to_customer_per_qty AS price,
    CASE
        WHEN quantity * cost_to_customer_per_qty > 50
          THEN 1
        ELSE 0
    END AS price_over_50
FROM farmers_market.customer_purchases
LIMIT 5
```

Table 4.3

market_date	customer_id	vendor_id	price	price_over_50
2019-07-03	14	7	6.9201	0
2019-07-03	14	7	15.2382	0
2019-07-03	15	7	10.6947	0
2019-07-03	16	7	14.1198	0
2019-07-03	22	7	4.6134	0

- To group line-item customer purchases into price bins:

```sql
SELECT
    market_date,
    customer_id, vendor_id,
    quantity * cost_to_customer_per_qty AS price,
    CASE
        WHEN quantity * cost_to_customer_per_qty < 5.00
          THEN 'Under $5'
        WHEN quantity * cost_to_customer_per_qty < 10.00
          THEN '$5-$9.99'
        WHEN quantity * cost_to_customer_per_qty < 20.00
          THEN '$10-$19.99'
        WHEN quantity * cost_to_customer_per_qty >= 20.00
          THEN '$20 and Up'
    END AS price_bin
FROM farmers_market.customer_purchases
LIMIT 5
```

Table 4.4

market_date	customer_id	vendor_id	price	price_bin
2019-07-03	14	7	6.9201	$5-$9.99
2019-07-03	14	7	15.2382	$10-$19.99
2019-07-03	15	7	10.6947	$10-$19.99
2019-07-03	16	7	14.1198	$10-$19.99
2019-07-03	22	7	4.6134	Under $5

- To output the bottom end of the numeric range for the bins:

```sql
SELECT
    market_date,
    customer_id, vendor_id,
    quantity * cost_to_customer_per_qty AS price,
    CASE
        WHEN quantity * cost_to_customer_per_qty < 5.00
          THEN 0
        WHEN quantity * cost_to_customer_per_qty < 10.00
          THEN 5
        WHEN quantity * cost_to_customer_per_qty < 20.00
          THEN 10
        WHEN quantity * cost_to_customer_per_qty >= 20.00
          THEN 20
    END AS price_bin_lower_end
FROM farmers_market.customer_purchases
LIMIT 5
```

Table 4.5

market_date	customer_id	vendor_id	price	price_bin_lower_end
2019-07-03	14	7	6.9201	5
2019-07-03	14	7	15.2382	10
2019-07-03	15	7	10.6947	10
2019-07-03	16	7	14.1198	10
2019-07-03	22	7	4.6134	0

- One query generates a new column of strings.
- The other generates a new column of numbers.
- Including both columns in your query can be useful for reports:
    - The `price_bin` column provides explanatory labels but sorts alphabetically.
    - The numeric column sorts bins correctly.
- If a price is mis-entered or a refund is recorded:
    - Negative values will fall into the "Under $5" or 0 bin.
    - This makes `price_bin_lower_end` a misnomer.
    - Ensure your `CASE` statements handle unexpected values appropriately.

Categorical Encoding Using CASE

- Convert the string variables into numeric values representing that order.
- For example:
    - The vendor booth price levels labeled "A," "B," and "C" can be converted into numeric values 1, 2, 3.

```sql
SELECT
    booth_number, booth_price_level,
    CASE
        WHEN booth_price_level = 'A' THEN 1
        WHEN booth_price_level = 'B' THEN 2
        WHEN booth_price_level = 'C' THEN 3
    END AS booth_price_level_numeric
FROM farmers_market.booth
LIMIT 5
```

Table 4.6

booth_number	booth_price_level	booth_price_level_numeric
1	A	1
2	A	1
3	B	2
4	C	3
5	C	3

- If categories have no rank order, like vendor types:
  - Use "one-hot encoding."
    - This creates a new column for each category.
      - Assign a binary value of 1 if a row falls into that category.
      - Assign a binary value of 0 otherwise.
    - These columns are called "dummy variables."

```
SELECT
    vendor_id, vendor_name, vendor_type,
    CASE WHEN vendor_type =  'Arts & Jewelry'
        THEN 1
        ELSE 0
    END AS arts_jewelry,
    CASE WHEN vendor_type =  'Eggs & Meats'
        THEN 1
        ELSE 0
    END AS eggs_meats,
    CASE WHEN vendor_type =  'Fresh Focused'
        THEN 1
        ELSE 0
    END AS fresh_focused,
    CASE WHEN vendor_type =  'Fresh Variety: Veggies & More'
        THEN 1
        ELSE 0
    END AS fresh_variety,
    CASE WHEN vendor_type = 'Prepared Foods'
        THEN 1
        ELSE 0
    END AS prepared
FROM farmers_market.vendor
```

Table 4.7

vendor_id	vendor_name	vendor_type	arts_jewelry	eggs_meats	fresh_focused	fresh_variety
1	Chris's Sustainable Eggs & Meats	Eggs & Meats	0	1	0	0
2	Hernández Salsa & Veggies	Fresh Variety: Veggies & More	0	0	0	1
3	Mountain View Vegetables	Fresh Variety: Veggies & More	0	0	0	1
4	Fields of Corn	Fresh Focused	0	0	1	0
5	Seashell Clay Shop	Arts & Jewelry	1	0	0	0
6	Mother's Garlic & Greens	Fresh Variety: Veggies & More	0	0	0	1
7	Marco's Peppers	Fresh Focused	0	0	1	0
8	Annie's Pies	Prepared Foods	0	0	0	0
9	Mediterranean Bakery	Prepared Foods	0	0	0	0

CASE Statement Summary

- Query 1:

```
SELECT
  customer_id,
  CASE
      WHEN customer_zip = '22801' THEN 'Local'
      ELSE 'Not Local'
  END customer_location_type
FROM farmers_market.customer
LIMIT 5
```

Table 4.8

customer_id	customer_location_type
1	Local
2	Not Local
3	Not Local
4	Local
5	Local

- Query 2:

```
SELECT
  booth_number,
  CASE WHEN booth_price_level = 'A'
      THEN 1
      ELSE 0
  END booth_price_level_A,
  CASE WHEN booth_price_level = 'B'
      THEN 1
      ELSE 0
  END booth_price_level_B,
  CASE WHEN booth_price_level = 'C'
      THEN 1
      ELSE 0
  END booth_price_level_C
FROM farmers_market.booth
LIMIT 5
```

Table 4.9

booth_number	booth_price_level_A	booth_price_level_B	booth_price_level_C
1	1	0	0
2	1	0	0
3	0	1	0
4	0	0	1
5	0	0	1

Excercises

- Look back at Figure 2.1 in Chapter 2 for sample data and column names for the `product` table referenced in these exercises.

1. Products can be sold individually or in bulk (e.g., lbs or oz).
    - Write a query that outputs the `product_id` and `product_name` columns from the `product` table.
    - Add a column called `prod_qty_type_condensed` that displays "unit" if `product_qty_type` is "unit," and "bulk" otherwise.
2. Flag all types of pepper products sold at the market.
    - Add a column to the previous query called `pepper_flag` that outputs 1 if `product_name` contains the word "pepper" (case-insensitive), and 0 otherwise.
3. Can you think of a situation where a pepper product might not be flagged as a pepper product using the code from the previous exercise?