🍕 Case Study #2C: Pizza Ingredients

C. Ingredient Optimisation

1. What are the standard ingredients for each pizza?

Steps:

- Use the `pizza_recipes` table to list all standard ingredients for each pizza type.
- Split the `toppings` column into individual `topping_id` values using the `REGEXP_SPLIT_TO_TABLE` function.
- Join the resulting data with the `pizza_toppings` table to retrieve ingredient names.

```
SELECT
  pr.pizza_id,
  pt.topping_name
FROM pizza_runner.pizza_recipes pr
JOIN LATERAL REGEXP_SPLIT_TO_TABLE(pr.toppings, '[,\s]+') AS
t(topping_id)
  ON TRUE
JOIN pizza_runner.pizza_toppings pt
  ON t.topping_id::INTEGER = pt.topping_id
ORDER BY pr.pizza_id, pt.topping_name;
```

Answer:

pizza_id	topping_name
1	BBQ Sauce
1	Bacon
1	Beef
1	Cheese
1	Chicken
1	Mushrooms
1	Pepperoni
1	Salami
2	Cheese
2	Mushrooms
2	Onions
2	Peppers
2	Tomato Sauce
2	Tomatoes

2. What was the most commonly added extra?

Steps:

- Extract the `extras` column from the `customer_orders` table.
- Split the values in `extras` into individual `topping_id` values using the `REGEXP_SPLIT_TO_TABLE` function.
- Join the resulting data with the `pizza_toppings` table to get topping names.
- Count the occurrences of each topping and order the results in descending order.

```
WITH extras_cte AS (
  SELECT
    REGEXP_SPLIT_TO_TABLE(c.extras, '[,\s]+')::INTEGER AS top
```

```
ping_id
  FROM customer_orders AS c
  WHERE c.extras <> ' '
)
SELECT
  pt.topping_name,
  COUNT(ec.topping_id) AS extra_count
FROM extras_cte AS ec
JOIN pizza_runner.pizza_toppings pt
  ON ec.topping_id = pt.topping_id
GROUP BY pt.topping_name
ORDER BY extra_count DESC
LIMIT 1;
```

Answer: The most commonly added extra was **Bacon**, with 4 occurrences.

topping_name	extra_count
Bacon	4

3. What was the most common exclusion?

Steps:

- Extract the `exclusions` column from the `customer_orders` table.
- Split the values in `exclusions` into individual `topping_id` values using the `REGEXP_SPLIT_TO_TABLE` function.
- Join the resulting data with the `pizza_toppings` table to get topping names.
- Count the occurrences of each topping and order the results in descending order.

```
WITH exclusions_cte AS (
  SELECT
    REGEXP_SPLIT_TO_TABLE(c.exclusions, '[,\s]+')::INTEGER AS
topping_id
```

```
  FROM customer_orders_temp AS c
  WHERE TRIM(c.exclusions) <> ''
)
SELECT
  pt.topping_name,
  COUNT(ec.topping_id) AS exclusion_count
FROM exclusions_cte AS ec
JOIN pizza_runner.pizza_toppings pt
  ON ec.topping_id = pt.topping_id
GROUP BY pt.topping_name
ORDER BY exclusion_count DESC
LIMIT 1;
```

Answer: The most commonly excluded ingredient was **Cheese**, with 4 occurrences.

topping_name	exclusion_count
Cheese	4

---

4. Generate an order item for each record in the `customer_orders` table

Steps:

- Join the `customer_orders` table with `pizza_names` to get the pizza name for each order.
- Use string concatenation to combine pizza names with exclusions and extras, if applicable.

```
SELECT
  c.order_id,
  p.pizza_name ||
  CASE
    WHEN c.exclusions <> ' ' THEN ' - Exclude ' || c.exclusio
ns
```

```
    ELSE ''
  END ||
  CASE
    WHEN c.extras <> ' ' THEN ' - Extra ' || c.extras
    ELSE ''
  END AS order_item
FROM customer_orders AS c
JOIN pizza_runner.pizza_names p
  ON c.pizza_id = p.pizza_id;
```

Answer:

order_id	order_item
10	Meatlovers - Exclude 2, 6 - Extra 1, 4
10	Meatlovers
9	Meatlovers - Exclude 4 - Extra 1, 5
8	Meatlovers
5	Meatlovers - Extra 1
4	Meatlovers - Exclude 4
4	Meatlovers - Exclude 4

5. Generate an alphabetically ordered ingredient list for each pizza order

Steps:

- Split the `toppings` column from `pizza_recipes` into individual `topping_id` values using `REGEXP_SPLIT_TO_TABLE`.
- Join the split data with `pizza_toppings` to get topping names.

- Alphabetically order and format the ingredient list for each pizza, adding a `2x` prefix for duplicate toppings.

```
WITH toppings_cte AS (
  SELECT
    c.order_id,
    pt.topping_name,
    COUNT(*) AS topping_count
  FROM customer_orders AS c
  JOIN pizza_runner.pizza_recipes pr
    ON c.pizza_id = pr.pizza_id
  JOIN pizza_runner.pizza_toppings pt
    ON REGEXP_SPLIT_TO_TABLE(pr.toppings, '[,\s]+')::INTEGER
= pt.topping_id
  GROUP BY c.order_id, pt.topping_name
)
SELECT
  order_id,
  STRING_AGG(
    CASE
      WHEN topping_count > 1 THEN '2x' || topping_name
      ELSE topping_name
    END, ', ' ORDER BY topping_name
  ) AS ingredient_list
FROM toppings_cte
GROUP BY order_id;
```

Answer:

order_id	ingredient_list
1	BBQ Sauce, Bacon, Beef, Cheese, Chicken, Mushrooms, Pepperoni, Salami
2	BBQ Sauce, Bacon, Beef, Cheese, Chicken, Mushrooms, Pepperoni, Salami
3	BBQ Sauce, Bacon, Beef, 2xCheese, Chicken, 2xMushrooms, Onions, Pepperoni, Peppers, Salami, Tomato Sauce, Tomatoes
4	2xBBQ Sauce, 2xBacon, 2xBeef, 2xCheese, 2xChicken, 2xMushrooms, Onions, 2xPepperoni, Peppers, 2xSalami, Tomato Sauce, Tomatoes
5	BBQ Sauce, Bacon, Beef, Cheese, Chicken, Mushrooms, Pepperoni, Salami
6	Cheese, Mushrooms, Onions, Peppers, Tomato Sauce, Tomatoes
7	Cheese, Mushrooms, Onions, Peppers, Tomato Sauce, Tomatoes

6. What is the total quantity of each ingredient used in all delivered pizzas?

Steps:

- Split the `toppings` column from `pizza_recipes` into individual `topping_id` values using `REGEXP_SPLIT_TO_TABLE`.
- Join the split data with `pizza_toppings` to get topping names.
- Join the `runner_orders` table to filter for delivered pizzas (`distance > 0`).
- Sum the quantities of each topping.

```
WITH ingredient_totals_cte AS (
  SELECT
    pt.topping_name,
    COUNT(*) AS total_quantity
  FROM customer_orders AS c
  JOIN runner_orders AS r
    ON c.order_id = r.order_id
  JOIN pizza_runner.pizza_recipes pr
    ON c.pizza_id = pr.pizza_id
  JOIN pizza_runner.pizza_toppings pt
    ON REGEXP_SPLIT_TO_TABLE(pr.toppings, '[,\s]+')::INTEGER
= pt.topping_id
  WHERE r.distance > 0
```

```
  GROUP BY pt.topping_name
)
SELECT
  topping_name,
  total_quantity
FROM ingredient_totals_cte
ORDER BY total_quantity DESC;
```

Answer:

topping_name	total_quantity
Cheese	12
Mushrooms	12
Salami	9
Bacon	9
BBQ Sauce	9
Beef	9
Pepperoni	9