```sql
-- Quering runners table
SELECT *
FROM pizza_runner.dbo.runners;
```

```
runner_id   registration_date
----------- -----------------
1           2021-01-01
2           2021-01-03
3           2021-01-08
4           2021-01-15
```

```sql
-- Quering customer_orders table
SELECT *
FROM pizza_runner.dbo.customer_orders;
```

```
order_id    customer_id pizza_id    exclusions extras order_time
----------- ----------- ----------- ---------- ------ -----------------------
1           101         1           NULL       NULL   2020-01-01 18:05:02.000
2           101         1           NULL       NULL   2020-01-01 19:00:52.000
3           102         1           NULL       NULL   2020-01-02 23:51:23.000
3           102         2           NULL       NULL   2020-01-02 23:51:23.000
4           103         1           4          NULL   2020-01-04 13:23:46.000
4           103         1           4          NULL   2020-01-04 13:23:46.000
4           103         2           4          NULL   2020-01-04 13:23:46.000
5           104         1           NULL       1      2020-01-08 21:00:29.000
6           101         2           NULL       NULL   2020-01-08 21:03:13.000
7           105         2           NULL       1      2020-01-08 21:20:29.000
8           102         1           NULL       NULL   2020-01-09 23:54:33.000
9           103         1           4          1, 5   2020-01-10 11:22:59.000
10          104         1           NULL       NULL   2020-01-11 18:34:49.000
10          104         1           2, 6       1, 4   2020-01-11 18:34:49.000
```

```sql
-- Quering runner_orders table
SELECT *
FROM pizza_runner.dbo.runner_orders;
```

```
order_id    runner_id   pickup_time             distance_km           duration_min cancellation
----------- ----------- ----------------------- --------------------- ------------ -----------------------
1           1           2020-01-01 18:15:34.000 20                    32           NULL
2           1           2020-01-01 19:10:54.000 20                    27           NULL
3           1           2020-01-03 00:12:37.000 13.4                  20           NULL
4           2           2020-01-04 13:53:03.000 23.4                  40           NULL
5           3           2020-01-08 21:10:57.000 10                    15           NULL
6           3           NULL                    NULL                  NULL         Restaurant Cancellation
7           2           2020-01-08 21:30:45.000 25                    25           NULL
8           2           2020-01-10 00:15:02.000 23.4                  15           NULL
9           2           NULL                    NULL                  NULL         Customer Cancellation
10          1           2020-01-11 18:50:20.000 10                    10           NULL
```

```sql
-- Quering pizza_names table
SELECT *
FROM pizza_runner.dbo.pizza_names;
```

```
pizza_id    pizza_name
----------- ------------
1           Meatlovers
2           Vegetarian
```

```
-- Quering pizza_recipes table
SELECT *
FROM pizza_runner.dbo.pizza_recipes;

pizza_id    toppings
----------- ----------
1           1, 2, 3, 4, 5, 6, 8, 10
2           4, 6, 7, 9, 11, 12

-- Quering pizza_toppings table
SELECT *
FROM pizza_runner.dbo.pizza_toppings;


topping_id  topping_name
----------- --------------
1           Bacon
2           BBQ Sauce
3           Beef
4           Cheese
5           Chicken
6           Mushrooms
7           Onions
8           Pepperoni
9           Peppers
10          Salami
11          Tomatoes
12          Tomato Sauce


-- PIZZA METRICS

-- 1. How many pizzas were ordered?
SELECT COUNT(pizza_id) AS num_of_pizza_ordered
FROM pizza_runner.dbo.customer_orders;

num_of_pizza_ordered
--------------------
14


-- 2. How many unique customer orders were made?
SELECT COUNT(DISTINCT(order_id)) AS unique_customer_orders
FROM pizza_runner.dbo.customer_orders;

unique_customer_orders
----------------------
10


-- 3. How many successful orders were delivered by each runner?
SELECT runner_id,
              COUNT(DISTINCT(order_id)) AS num_of_orders_delivered
FROM pizza_runner.dbo.runner_orders
WHERE distance_km IS NOT NULL
GROUP BY runner_id
ORDER BY num_of_orders_delivered DESC;
```

```
runner_id   num_of_orders_delivered
----------- -----------------------
1           4
2           3
3           1


-- 4. How many of each type of pizza was delivered?
SELECT pizza_id,
              COUNT(CO.order_id) AS num_of_pizzas
FROM pizza_runner.dbo.customer_orders AS CO
INNER JOIN pizza_runner.dbo.runner_orders AS RO
ON CO.order_id = RO.order_id
WHERE RO.distance_km IS NOT NULL
GROUP BY pizza_id
ORDER BY num_of_pizzas DESC;

pizza_id    num_of_pizzas
----------- -------------
1           9
2           3


-- 5. How many Vegetarian and Meatlovers were ordered by each customer?
SELECT customer_id,
              pizza_name,
              num_of_pizzas
FROM (
       SELECT customer_id, pizza_id,
                     COUNT(*) AS num_of_pizzas
       FROM pizza_runner.dbo.customer_orders
       GROUP BY customer_id, pizza_id
       ) AS vm_pizza
INNER JOIN pizza_runner.dbo.pizza_names AS pn
ON vm_pizza.pizza_id = pn.pizza_id
ORDER BY customer_id ASC;

customer_id pizza_name     num_of_pizzas
----------- ------------- ---------------
101         Meatlovers    2
101         Vegetarian    1
102         Meatlovers    2
102         Vegetarian    1
103         Meatlovers    3
103         Vegetarian    1
104         Meatlovers    3
105         Vegetarian    1


-- 6. What was the maximum number of pizzas delivered in a single order?
SELECT TOP 1 order_id,
              COUNT(pizza_id) AS maximum_num_of_pizzas
FROM pizza_runner.dbo.customer_orders
GROUP BY order_id
ORDER BY maximum_num_of_pizzas DESC;

order_id    maximum_num_of_pizzas
----------- ---------------------
4           3
```

```
-- 7. For each customer, how many delivered pizzas had at least 1 change and how many
had no changes?
WITH changes_in_pizza AS (
              SELECT customer_id,
                     CASE
                            WHEN exclusions IS NULL AND extras IS NULL THEN
'no_change'
                            ELSE 'change'
                     END AS pizzas_with
              FROM pizza_runner.dbo.customer_orders AS CO
              INNER JOIN pizza_runner.dbo.runner_orders AS RO
              ON CO.order_id = RO.order_id
              WHERE RO.distance_km IS NOT NULL
              )

SELECT customer_id,
              pizzas_with,
              COUNT(*) AS num_of_pizzas
FROM changes_in_pizza
GROUP BY customer_id, pizzas_with
ORDER BY customer_id ASC;

customer_id pizzas_with num_of_pizzas
----------- ----------- -------------
101         no_change   2
102         no_change   3
103         change      3
104         change      2
104         no_change   1
105         change      1

-- 8. How many pizzas were delivered that had both exclusions and extras?
SELECT COUNT(pizza_id) AS pizzas_delivered_with_exclusions_and_extras
FROM pizza_runner.dbo.customer_orders AS CO
INNER JOIN pizza_runner.dbo.runner_orders AS RO
ON CO.order_id = RO.order_id
WHERE RO.distance_km IS NOT NULL
       AND exclusions IS NOT NULL AND extras IS NOT NULL;

pizzas_delivered_with_exclusions_and_extras
-------------------------------------------
1


-- 9. What was the total volume of pizzas ordered for each hour of the day?
SELECT DATEPART(HOUR, order_time) AS hour_of_day,
              COUNT(pizza_id) AS num_of_pizzas
FROM pizza_runner.dbo.customer_orders
GROUP BY DATEPART(HOUR, order_time)
ORDER BY hour_of_day ASC;


hour_of_day num_of_pizzas
----------- -------------
11          1
13          3
18          3
19          1
21          3
23          3
```

```sql
-- 10. What was the volume of orders for each day of the week?
SELECT DATENAME(WEEKDAY, order_time) AS day_of_week,
              COUNT(DISTINCT(order_id)) AS num_of_orders
FROM pizza_runner.dbo.customer_orders
GROUP BY DATENAME(WEEKDAY, order_time);
```

```
day_of_week                    num_of_orders
------------------------------ -------------
Friday                         1
Saturday                       2
Thursday                       2
Wednesday                      5
```

```sql
-- RUNNER AND CUSTOMER EXPERIENCE

-- 1. How many runners signed up for each 1 week period? (i.e. week starts 2021-01-01)
SELECT DATEPART(WEEK, DATEPART(day, registration_date)) AS week_number,
              COUNT(runner_id) AS runners_registered
FROM pizza_runner.dbo.runners
GROUP BY DATEPART(WEEK, DATEPART(day, registration_date));
```

```
week_number runners_registered
----------- ------------------
1           2
2           1
3           1
```

```sql
-- 2. What was the average time in minutes it took for each runner to arrive at the
Pizza Runner HQ to pickup the order?
WITH order_pickup_time AS (
              SELECT DISTINCT CO.order_id,
                     RO.runner_id,
                     order_time,
                     pickup_time,
                     DATEPART(MINUTE, pickup_time - order_time) as time_difference
              FROM pizza_runner.dbo.customer_orders AS CO
              INNER JOIN pizza_runner.dbo.runner_orders AS RO
              ON CO.order_id = RO.order_id
              WHERE pickup_time IS NOT NULL
              )

SELECT runner_id,
              AVG(time_difference) AS avg_pickup_time
FROM order_pickup_time
GROUP BY runner_id;
```

```
runner_id   avg_pickup_time
----------- ---------------
1           14
2           19
3           10
```

```
-- 3. Is there any relationship between the number of pizzas and how long the order
takes to prepare?
WITH pizza_preparation AS (
              SELECT CO.order_id, COUNT(pizza_id) AS pizzas_ordered,
                            AVG(DATEPART(MINUTE, pickup_time - order_time)) AS
avg_time_difference
              FROM pizza_runner.dbo.customer_orders AS CO
              INNER JOIN pizza_runner.dbo.runner_orders AS RO
              ON CO.order_id = RO.order_id
              GROUP BY CO.order_id
              )

SELECT pizzas_ordered,
              AVG(avg_time_difference) AS avg_time_to_prepare_pizza
FROM pizza_preparation
GROUP BY pizzas_ordered;


pizzas_ordered avg_time_to_prepare_pizza
-------------- -------------------------
1              12
2              18
3              29
```

**There is a positive correlation between them. As number of pizzas ordered increases, the average time to
prepare the order also increases.**

```
-- 4. What was the average distance travelled for each customer?
SELECT CO.customer_id,
              ROUND(AVG(RO.distance_km), 2) AS avg_distance_travelled
FROM pizza_runner.dbo.customer_orders AS CO
INNER JOIN pizza_runner.dbo.runner_orders AS RO
ON CO.order_id = RO.order_id
GROUP BY CO.customer_id;

customer_id avg_distance_travelled
----------- ----------------------
101         20
102         16.73
103         23.4
104         10
105         25


-- 5. What was the difference between the longest and shortest delivery times for all
orders?
WITH Total_delivery_time AS (
              SELECT DISTINCT RO.order_id,
                            DATEPART(MINUTE, pickup_time - order_time) AS
pickup_time_min,
                            RO.duration_min AS delivery_time_min
              FROM pizza_runner.dbo.customer_orders AS CO
              INNER JOIN pizza_runner.dbo.runner_orders AS RO
              ON CO.order_id = RO.order_id
              )

SELECT MAX(pickup_time_min + delivery_time_min) - MIN(pickup_time_min +
delivery_time_min) AS delivery_time_range
FROM Total_delivery_time;
```

```
delivery_time_range
-------------------
44


-- 6. What was the average speed for each runner for each delivery and do you notice
any trend for these values?
SELECT runner_id,
              COUNT(order_id) AS num_of_orders,
              ROUND(AVG((distance_km / (CAST(duration_min AS FLOAT) / 60))), 2) AS
runner_speed_km_h
FROM pizza_runner.dbo.runner_orders
WHERE distance_km IS NOT NULL
GROUP BY runner_id;

runner_id   num_of_orders runner_speed_km_h
----------- ------------- ----------------------
1           4             45.54
2           3             62.9
3           1             40
```

**Trends I notice here is runner with id 2 rides very fast to deliver the order as compared to other runners.**

```
-- 7. What is the successful delivery percentage for each runner?
SELECT runner_id,
              CAST(COUNT(distance_km) AS FLOAT) / COUNT(*) * 100 AS
percentage_of_successful_delivery
FROM pizza_runner.dbo.runner_orders
GROUP BY runner_id;

runner_id   percentage_of_successful_delivery
----------- ---------------------------------
1           100
2           75
3           50



-- INGREDIENT OPTIMISATION

-- 1. What are the standard ingredients for each pizza?
SELECT pizza_name,
              PT.topping_name AS standard_ingredients
FROM pizza_runner.dbo.pizza_recipes AS PR
INNER JOIN pizza_runner.dbo.pizza_names AS PN
ON PR.pizza_id = PN.pizza_id
CROSS APPLY STRING_SPLIT(CAST(PR.toppings AS VARCHAR), ',') AS SS
INNER JOIN pizza_runner.dbo.pizza_toppings AS PT
ON SS.VALUE = PT.topping_id;
```

```
pizza_name   standard_ingredients
------------ ----------------------
Meatlovers   Bacon
Meatlovers   BBQ Sauce
Meatlovers   Beef
Meatlovers   Cheese
Meatlovers   Chicken
Meatlovers   Mushrooms
Meatlovers   Pepperoni
Meatlovers   Salami
Vegetarian   Cheese
Vegetarian   Mushrooms
Vegetarian   Onions
Vegetarian   Peppers
Vegetarian   Tomatoes
Vegetarian   Tomato Sauce


-- 2. What was the most commonly added extra?
WITH common_add AS (
              SELECT CAST(PT.topping_name AS VARCHAR) AS topping,
                            CAST(VALUE AS INT) AS id
              FROM pizza_runner.dbo.customer_orders
              CROSS APPLY STRING_SPLIT(extras, ',') AS SS
              INNER JOIN pizza_runner.dbo.pizza_toppings AS PT
              ON SS.VALUE = PT.topping_id
              )

SELECT TOP 1 topping AS most_common_extra,
              COUNT(id) AS added_times
FROM common_add
GROUP BY topping
ORDER BY added_times DESC;


most_common_extra              added_times
------------------------------ -----------
Bacon                          4


-- 3. What was the most common exclusion?
WITH common_remove AS (
              SELECT CAST(PT.topping_name AS VARCHAR) AS topping,
                            CAST(VALUE AS INT) AS id
              FROM pizza_runner.dbo.customer_orders
              CROSS APPLY STRING_SPLIT(exclusions, ',') AS SS
              INNER JOIN pizza_runner.dbo.pizza_toppings AS PT
              ON SS.VALUE = PT.topping_id
              )

SELECT TOP 1 topping AS most_common_exclusion,
              COUNT(id) AS exclusion_times
FROM common_remove
GROUP BY topping
ORDER BY exclusion_times DESC;

most_common_exclusion          exclusion_times
------------------------------ ---------------
Cheese                         4
```

```sql
-- 4. Generate an order item for each record in the customers_orders table in the
format of one of the following:
	-- Meat Lovers
	-- Meat Lovers - Exclude Beef
	-- Meat Lovers - Extra Bacon
	-- Meat Lovers - Exclude Cheese, Bacon - Extra Mushroom, Peppers

WITH orders AS (
		SELECT order_id,
				pizza_id,
				LEFT(exclusions, 1) AS excl1,
				CASE
					WHEN RIGHT(exclusions, 1) = LEFT(exclusions, 1)
THEN NULL
					ELSE RIGHT(exclusions, 1)
				END AS excl2,
				LEFT(extras, 1) AS ext1,
				CASE
					WHEN RIGHT(extras, 1) = LEFT(extras, 1) THEN NULL
					ELSE RIGHT(extras, 1)
				END AS ext2
				FROM pizza_runner.dbo.customer_orders),

	order_details AS (
		SELECT order_id,
				CASE
					WHEN CAST(PN.pizza_name AS NVARCHAR) = 'Meatlovers'
THEN 'Meat Lovers'
					ELSE 'Vegetarian'
				END AS pizza_name,
				CONCAT(' - Exclude ', PT1.topping_name) AS exclude1,
				PT2.topping_name AS exclude2,
				CONCAT(' - Extra ', PT3.topping_name) AS extra1,
				PT4.topping_name AS extra2
		FROM orders AS CO
		LEFT JOIN pizza_runner.dbo.pizza_toppings AS PT1
		ON CO.excl1 = PT1.topping_id
		LEFT JOIN pizza_runner.dbo.pizza_toppings AS PT2
		ON CO.excl2 = PT2.topping_id
		LEFT JOIN pizza_runner.dbo.pizza_toppings AS PT3
		ON CO.ext1 = PT3.topping_id
		LEFT JOIN pizza_runner.dbo.pizza_toppings AS PT4
		ON CO.ext2 = PT4.topping_id
		LEFT JOIN pizza_runner.dbo.pizza_names AS PN
		ON CO.pizza_id = PN.pizza_id
		),

		order_details_2 AS (
			SELECT *,
				CASE
					WHEN exclude2 IS NULL THEN exclude1
					ELSE CONCAT(exclude1, ', ', exclude2)
				END AS excluding,
				CASE
					WHEN extra2 IS NULL THEN extra1
					ELSE CONCAT(extra1, ', ', extra2)
				END AS extra
		FROM order_details
		),

		order_details_3 AS (
		SELECT *,
                            CASE
                                   WHEN excluding = ' - Exclude' THEN NULL
                                   ELSE excluding
                            END AS exclude_toppings,
                            CASE
                                   WHEN extra = ' - Extra' THEN NULL
                                   ELSE extra
                            END AS extra_toppings
              FROM order_details_2
              )

SELECT order_id,
              CONCAT(pizza_name, exclude_toppings, extra_toppings) AS detailed_order
FROM order_details_3;


order_id    detailed_order
----------- -------------------------------------------------------------------
1           Meat Lovers
2           Meat Lovers
3           Meat Lovers
3           Vegetarian
4           Meat Lovers - Exclude Cheese
4           Meat Lovers - Exclude Cheese
4           Vegetarian - Exclude Cheese
5           Meat Lovers - Extra Bacon
6           Vegetarian
7           Vegetarian - Extra Bacon
8           Meat Lovers
9           Meat Lovers - Exclude Cheese - Extra Bacon, Chicken
10          Meat Lovers
10          Meat Lovers - Exclude BBQ Sauce, Mushrooms - Extra Bacon, Cheese


-- 5. Generate an alphabetically ordered comma separated ingredient list for each
pizza order
       -- from the customer_orders table and add a 2x in front of any relevant
ingredients
       -- For example: "Meat Lovers: 2xBacon, Beef, ... , Salami"

WITH excl_toppings AS (
              SELECT order_id,
                            CO.pizza_id,
                            toppings,
                            LEFT(extras, 1) as e1,
                            CASE
                            WHEN LEFT(extras, 1) = RIGHT(extras, 1) THEN NULL
                            ELSE RIGHT(extras, 1)
                            END AS e2,
                            COALESCE(REPLACE(REPLACE(cast(toppings as varchar),
LEFT(exclusions, 1), ''), RIGHT(exclusions, 1), ''), toppings) AS excluded_toppings,
                            ROW_NUMBER() OVER(ORDER BY order_id) AS rn
              FROM pizza_runner.dbo.customer_orders AS CO
              LEFT JOIN pizza_runner.dbo.pizza_recipes AS PR
              ON CO.pizza_id = PR.pizza_id
              ),

       all_toppings AS (
              SELECT order_id,
                            pizza_id,
                            rn,
                            CASE
```

```
                                    WHEN e1 IS NOT NULL AND e2 IS NULL THEN
CONCAT(excluded_toppings, ', ', e1)
                                    WHEN e1 IS NULL AND e2 IS NOT NULL THEN
CONCAT(excluded_toppings, ', ', e2)
                                    WHEN e1 IS NOT NULL AND e2 IS NOT NULL THEN
CONCAT(excluded_toppings, ', ', e1, ', ', e2)
                                    ELSE toppings
                              END AS total_toppings
              FROM excl_toppings
              ),

       toppings_count AS (
              SELECT order_id,
                              pizza_id,
                              rn,
                              COUNT(value) AS num_of_toppings,
                              CAST(PT.topping_name AS VARCHAR) AS topping_name
              FROM all_toppings
              CROSS APPLY STRING_SPLIT(CAST(total_toppings AS VARCHAR), ',') AS SS1
              INNER JOIN pizza_runner.dbo.pizza_toppings AS PT
              ON SS1.value = PT.topping_id
              GROUP BY order_id, rn, CAST(PT.topping_name AS VARCHAR), pizza_id
              ),

       all_toppings_name AS (
              SELECT order_id,
                              rn,
                              pizza_id,
                              CASE
                              WHEN num_of_toppings = 1 THEN topping_name
                              ELSE CONCAT(num_of_toppings, '*', topping_name)
                              END AS total_toppings
              FROM toppings_count
              ),

       all_pizza_ingredients AS (
              SELECT order_id,
                              rn,
                              STRING_AGG(CAST(total_toppings AS VARCHAR), ', ') AS
ingredients,
                              CASE
                              WHEN CAST(PN.pizza_name AS VARCHAR) = 'Meatlovers' THEN 'Meat Lovers:
'
                              ELSE 'Vegetarian: '
                              END AS pizza_name_new
              FROM all_toppings_name AS AN
              INNER JOIN pizza_runner.dbo.pizza_names AS PN
              ON AN.pizza_id = PN.pizza_id
              GROUP BY order_id, CAST(PN.pizza_name AS VARCHAR), rn
                     )

SELECT order_id,
              CONCAT(pizza_name_new, ingredients) AS full_ingredients_list
FROM all_pizza_ingredients;
```

```
order_id    full_ingredients_list
----------- --------------------------------------------------------------------------------
1           Meat Lovers: Bacon, BBQ Sauce, Beef, Cheese, Chicken, Mushrooms, Pepperoni, Salami
2           Meat Lovers: Bacon, BBQ Sauce, Beef, Cheese, Chicken, Mushrooms, Pepperoni, Salami
3           Meat Lovers: Bacon, BBQ Sauce, Beef, Cheese, Chicken, Mushrooms, Pepperoni, Salami
3           Vegetarian: Cheese, Mushrooms, Onions, Peppers, Tomato Sauce, Tomatoes
4           Meat Lovers: Bacon, BBQ Sauce, Beef, Cheese, Chicken, Mushrooms, Pepperoni, Salami
4           Meat Lovers: Bacon, BBQ Sauce, Beef, Cheese, Chicken, Mushrooms, Pepperoni, Salami
4           Vegetarian: Cheese, Mushrooms, Onions, Peppers, Tomato Sauce, Tomatoes
5           Meat Lovers: 2*Bacon, BBQ Sauce, Beef, Cheese, Chicken, Mushrooms, Pepperoni, Salami
6           Vegetarian: Cheese, Mushrooms, Onions, Peppers, Tomato Sauce, Tomatoes
7           Vegetarian: Bacon, Cheese, Mushrooms, Onions, Peppers, Tomato Sauce, Tomatoes
8           Meat Lovers: Bacon, BBQ Sauce, Beef, Cheese, Chicken, Mushrooms, Pepperoni, Salami
9           Meat Lovers: 2*Bacon, BBQ Sauce, Beef, 2*Chicken, Mushrooms, Pepperoni, Salami
10          Meat Lovers: Bacon, BBQ Sauce, Beef, Cheese, Chicken, Mushrooms, Pepperoni, Salami
10          Meat Lovers: 2*Bacon, Beef, 2*Cheese, Chicken, Pepperoni, Salami
```

```
-- 6. What is the total quantity of each ingredient used in all delivered pizzas
sorted by most frequent first?

WITH excl_toppings AS (
                SELECT order_id,
                                toppings,
                                LEFT(extras, 1) as e1,
                                CASE
                                WHEN LEFT(extras, 1) = RIGHT(extras, 1) THEN NULL
                                ELSE RIGHT(extras, 1)
                                END AS e2,
                                COALESCE(REPLACE(REPLACE(cast(toppings as varchar),
                LEFT(exclusions, 1), ''), RIGHT(exclusions, 1), ''), toppings) AS
                excluded_toppings,

                                ROW_NUMBER() OVER(ORDER BY order_id) AS rn
                FROM pizza_runner.dbo.customer_orders AS CO
                LEFT JOIN pizza_runner.dbo.pizza_recipes AS PR
                ON CO.pizza_id = PR.pizza_id
                WHERE order_id IN (SELECT order_id
                                        FROM pizza_runner.dbo.runner_orders
                                        WHERE distance_km IS NOT NULL)
                ),

        all_toppings AS (
                SELECT order_id,
                                rn,
                                CASE
                                WHEN e1 IS NOT NULL AND e2 IS NULL THEN
CONCAT(excluded_toppings, ', ', e1)
                                WHEN e1 IS NULL AND e2 IS NOT NULL THEN
CONCAT(excluded_toppings, ', ', e2)
                                WHEN e1 IS NOT NULL AND e2 IS NOT NULL THEN
CONCAT(excluded_toppings, ', ', e1, ', ', e2)
                                        ELSE toppings
                                END AS total_toppings
                FROM excl_toppings
                )

SELECT CAST(topping_name AS VARCHAR) AS topping_name,
                                COUNT(value) AS num_of_toppings
FROM all_toppings
```

```
CROSS APPLY STRING_SPLIT(CAST(total_toppings AS VARCHAR), ',') AS SS1
INNER JOIN pizza_runner.dbo.pizza_toppings AS PT
ON SS1.value = PT.topping_id
GROUP BY CAST(topping_name AS VARCHAR)
ORDER BY num_of_toppings DESC;

topping_name                   num_of_toppings
------------------------------ ---------------
Cheese                         13
Bacon                          12
Mushrooms                      11
Pepperoni                      9
Chicken                        9
Salami                         9
Beef                           9
BBQ Sauce                      8
Peppers                        3
Onions                         3
Tomato Sauce                   3
Tomatoes                       3



-- PRICING AND RATINGS

-- 1. If a Meat Lovers pizza costs $12 and Vegetarian costs $10 and there were no
charges for changes - how much money has Pizza Runner made so far if there are no
delivery fees?

WITH Restaurant_earnings AS (
              SELECT CAST(PN.pizza_name AS NVARCHAR) AS type_of_pizza,
                            CASE
                                   WHEN CAST(PN.pizza_name AS NVARCHAR)= 'Meatlovers'
THEN 12
                                   ELSE 10
                            END AS Earnings
              FROM pizza_runner.dbo.customer_orders AS CO
              INNER JOIN pizza_runner.dbo.pizza_names AS PN
              ON CO.pizza_id = PN.pizza_id
              WHERE order_id IN (SELECT order_id
                                          FROM pizza_runner.dbo.runner_orders
                                          WHERE distance_km IS NOT NULL)
                            )

SELECT CONCAT('$ ', SUM(Earnings)) AS Total_Earnings
FROM Restaurant_earnings;

Total_Earnings
--------------
$ 138


-- 2. What if there was an additional $1 charge for any pizza extras?

WITH delivered_pizzas AS (
              SELECT CO.pizza_id,
                            pizza_name,
                            LEFT(extras, 1) as e1,
                            CASE
                                   WHEN LEFT(extras, 1) = RIGHT(extras, 1) THEN NULL
                                   ELSE RIGHT(extras, 1)
                            END AS e2
              FROM pizza_runner.dbo.customer_orders AS CO
              INNER JOIN pizza_runner.dbo.pizza_names AS PN
              ON CO.pizza_id = PN.pizza_id
              WHERE order_id IN (SELECT order_id
                                                              FROM
pizza_runner.dbo.runner_orders
                                                              WHERE distance_km IS NOT
NULL)
              ),

       total_pizza_costs AS (
              SELECT pizza_name,
                            CASE
                                   WHEN CAST(pizza_name AS VARCHAR) = 'Meatlovers'
THEN 12
                                   ELSE 10
                            END AS pizza_cost,
                            CASE
                                   WHEN CAST(PT1.topping_name AS VARCHAR) IS NOT NULL
THEN 1
                                   ELSE 0
                            END AS topping_1_cost,
                            CASE
                                   WHEN CAST(PT2.topping_name AS VARCHAR) IS NOT NULL
THEN 1
                                   ELSE 0
                            END AS topping_2_cost
              FROM delivered_pizzas AS P
              LEFT JOIN pizza_runner.dbo.pizza_toppings AS PT1
              ON P.e1 = PT1.topping_id
              LEFT JOIN pizza_runner.dbo.pizza_toppings AS PT2
              ON P.e2 = PT2.topping_id
              )

SELECT CONCAT('$ ', SUM(pizza_cost + topping_1_cost + topping_2_cost)) AS
restaurant_earnings
FROM total_pizza_costs;

restaurant_earnings
-------------------
$ 142


-- 2.1 Add cheese is $1 extra

WITH delivered_pizzas AS (
              SELECT CO.pizza_id,
                            pizza_name,
                            LEFT(extras, 1) as e1,
                            CASE
                                   WHEN LEFT(extras, 1) = RIGHT(extras, 1) THEN NULL
                                   ELSE RIGHT(extras, 1)
                            END AS e2
              FROM pizza_runner.dbo.customer_orders AS CO
              INNER JOIN pizza_runner.dbo.pizza_names AS PN
              ON CO.pizza_id = PN.pizza_id
              WHERE order_id IN (SELECT order_id
                                                              FROM
pizza_runner.dbo.runner_orders
                                                              WHERE distance_km IS NOT
NULL)
              ),
       total_pizza_costs AS (
              SELECT pizza_name,
                            CASE
                                   WHEN CAST(pizza_name AS VARCHAR) = 'Meatlovers'
THEN 12
                                   ELSE 10
                            END AS pizza_cost,
                            CASE
                                   WHEN CAST(PT1.topping_name AS VARCHAR) IS NOT NULL
AND CAST(PT1.topping_name AS VARCHAR) = 'Cheese' THEN 2
                                   WHEN CAST(PT1.topping_name AS VARCHAR) IS NOT NULL
THEN 1
                                   ELSE 0
                            END AS topping_1_cost,
                            CASE
                                   WHEN CAST(PT2.topping_name AS VARCHAR) IS NOT NULL
AND CAST(PT2.topping_name AS VARCHAR) = 'Cheese' THEN 2
                                   WHEN CAST(PT2.topping_name AS VARCHAR) IS NOT NULL
THEN 1
                                   ELSE 0
                            END AS topping_2_cost
              FROM delivered_pizzas AS P
              LEFT JOIN pizza_runner.dbo.pizza_toppings AS PT1
              ON P.e1 = PT1.topping_id
              LEFT JOIN pizza_runner.dbo.pizza_toppings AS PT2
              ON P.e2 = PT2.topping_id
              )

SELECT CONCAT('$ ', SUM(pizza_cost + topping_1_cost + topping_2_cost)) AS
restaurant_earnings
FROM total_pizza_costs;

restaurant_earnings
-------------------
$ 143


-- 3. The Pizza Runner team now wants to add an additional ratings system that allows
customers to rate their runner, how would you design an additional table for this new
dataset - generate a schema for this new table and insert your own data for ratings
for each successful customer order between 1 to 5.

DROP TABLE IF EXISTS pizza_runner.dbo.runner_ratings;
CREATE TABLE pizza_runner.dbo.runner_ratings (
  "order_id" INTEGER,
  "customer_id" INTEGER,
  "runner_id" INTEGER,
  "rating" INTEGER
);

INSERT INTO pizza_runner.dbo.runner_ratings
  ("order_id", "customer_id", "runner_id", "rating")
VALUES
  (1, 101, 1, 4),
  (2, 101, 1, 5),
  (3, 102, 1, 3),
  (4, 103, 2, 3),
  (5, 104, 3, 4),
  (7, 105, 2, 5),
  (8, 102, 2, 5),
  (10, 104, 1, 5);
```

```
order_id    customer_id runner_id   rating
----------- ----------- ----------- -----------
1           101         1           4
2           101         1           5
3           102         1           3
4           103         2           3
5           104         3           4
7           105         2           5
8           102         2           5
10          104         1           5
```

```sql
-- 4. Using your newly generated table - can you join all of the information together
to form a table which has the following information for successful deliveries?
       -- customer_id
       -- order_id
       -- runner_id
       -- rating
       -- order_time
       -- pickup_time
       -- Time between order and pickup
       -- Delivery duration
       -- Average speed
       -- Total number of pizzas

SELECT CO.customer_id,
              CO.order_id,
              RR.runner_id,
              RR.rating,
              CO.order_time,
              RO.pickup_time,
              DATEPART(minute, pickup_time - order_time) AS difference,
              duration_min AS delivery_time_mins,
              ROUND(AVG(distance_km / (CAST(duration_min AS FLOAT) / 60)), 2) AS
avg_speed_kmph,
              COUNT(CO.pizza_id) AS total_num_of_pizzas
FROM pizza_runner.dbo.runner_ratings AS RR
INNER JOIN pizza_runner.dbo.customer_orders AS CO
ON RR.order_id = CO.order_id
INNER JOIN pizza_runner.dbo.runner_orders AS RO
ON RR.order_id = RO.order_id
GROUP BY CO.customer_id, CO.order_id, RR.runner_id, rating, order_time, pickup_time,
DATEPART(minute, pickup_time - order_time), duration_min;
```

| customer_id | order_id | runner_id | rating | order_time | pickup_time | difference | delivery_time_mins | avg_speed_kmph | total_num_of_pizzas |
|---|---|---|---|---|---|---|---|---|---|
| ------------- | ----------- | ------------ | --------- | ----------------------- | ------------------------- | ------------ | --------------------- | --------------------- | ----------------------- |
| 101 | 1 | 1 | 4 | 2020-01-01 18:05:02.0 | 2020-01-01 18:15:34.000 | 10 | 32 | 37.5 | 1 |
| 101 | 2 | 1 | 5 | 2020-01-01 19:00:52.0 | 2020-01-01 19:10:54.000 | 10 | 27 | 44.44 | 1 |
| 102 | 3 | 1 | 3 | 2020-01-02 23:51:23.0 | 2020-01-03 00:12:37.000 | 21 | 20 | 40.2 | 2 |
| 102 | 8 | 2 | 5 | 2020-01-09 23:54:33.0 | 2020-01-10 00:15:02.000 | 20 | 15 | 93.6 | 1 |
| 103 | 4 | 2 | 3 | 2020-01-04 13:23:46.0 | 2020-01-04 13:53:03.00 | 29 | 40 | 35.1 | 3 |
| 104 | 5 | 3 | 4 | 2020-01-08 21:00:29.0 | 2020-01-08 21:10:57.000 | 10 | 15 | 40 | 1 |
| 104 | 10 | 1 | 5 | 2020-01-11 18:34:49.0 | 2020-01-11 18:50:20.000 | 15 | 10 | 60 | 2 |
| 105 | 7 | 2 | 5 | 2020-01-08 21:20:29.0 | 2020-01-08 21:30:45.000 | 10 | 25 | 60 | 1 |

```sql
-- 5. If a Meat Lovers pizza was $12 and Vegetarian $10 fixed prices with no cost for
extras and each runner is paid $0.30 per kilometre traveled
       -- how much money does Pizza Runner have left over after these deliveries?
```

```
WITH costs AS (
              SELECT CAST(PN.pizza_name AS NVARCHAR) AS type_of_pizza,
                            CASE
                                   WHEN CAST(PN.pizza_name AS NVARCHAR)= 'Meatlovers'
THEN 12
                                   ELSE 10
                            END AS Earnings,
                            RO.distance_km
              FROM pizza_runner.dbo.customer_orders AS CO
              INNER JOIN pizza_runner.dbo.pizza_names AS PN
              ON CO.pizza_id = PN.pizza_id
              INNER JOIN pizza_runner.dbo.runner_orders AS RO
              ON CO.order_id = RO.order_id
              WHERE RO.distance_km IS NOT NULL
              )

SELECT CONCAT('$ ', SUM(Earnings - (distance_km * 0.3))) AS restaurant_earnings
FROM costs;
```

```
restaurant_earnings
-------------------------
$ 73.38
```