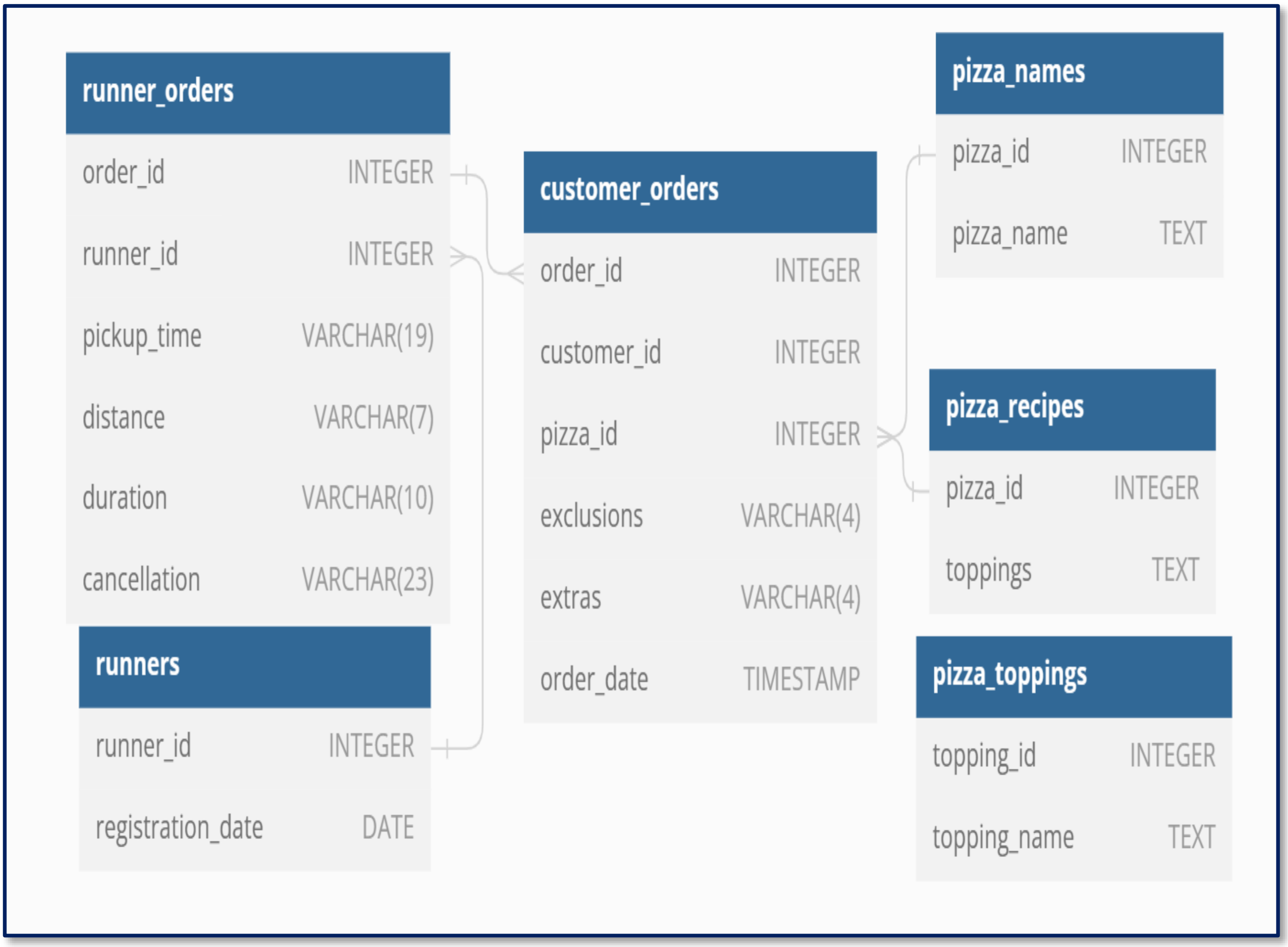
Case Study #2 - Pizza Runner

SCHEMA



runner_orders	
order_id	INTEGER
runner_id	INTEGER
pickup_time	VARCHAR(19)
distance	VARCHAR(7)
duration	VARCHAR(10)
cancellation	VARCHAR(23)

runners	
runner_id	INTEGER
registration_date	DATE

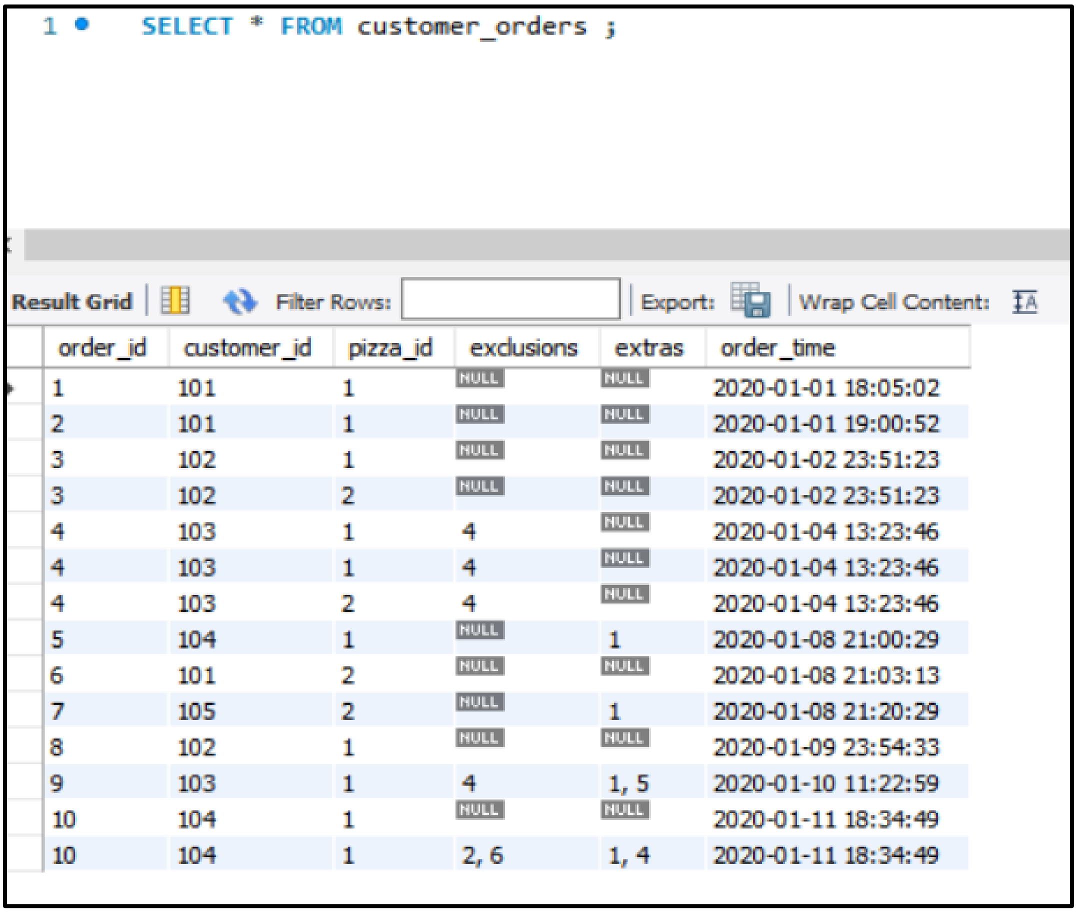
customer_orders	
order_id	INTEGER
customer_id	INTEGER
pizza_id	INTEGER
exclusions	VARCHAR(4)
extras	VARCHAR(4)
order_date	TIMESTAMP

pizza_names	
pizza_id	INTEGER
pizza_name	TEXT

pizza_recipes	
pizza_id	INTEGER
toppings	TEXT

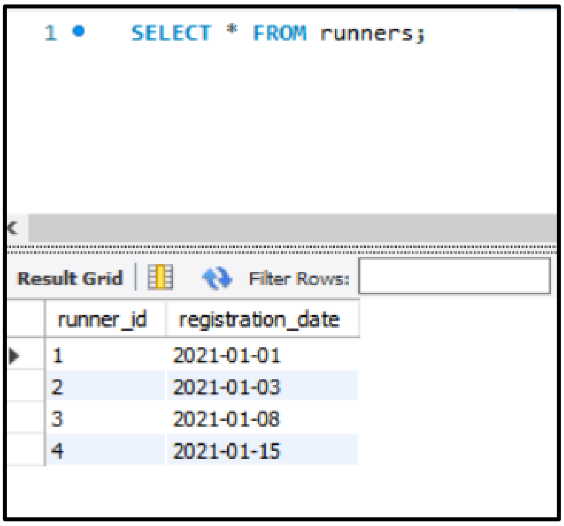
pizza_toppings	
topping_id	INTEGER
topping_name	TEXT

TABLES



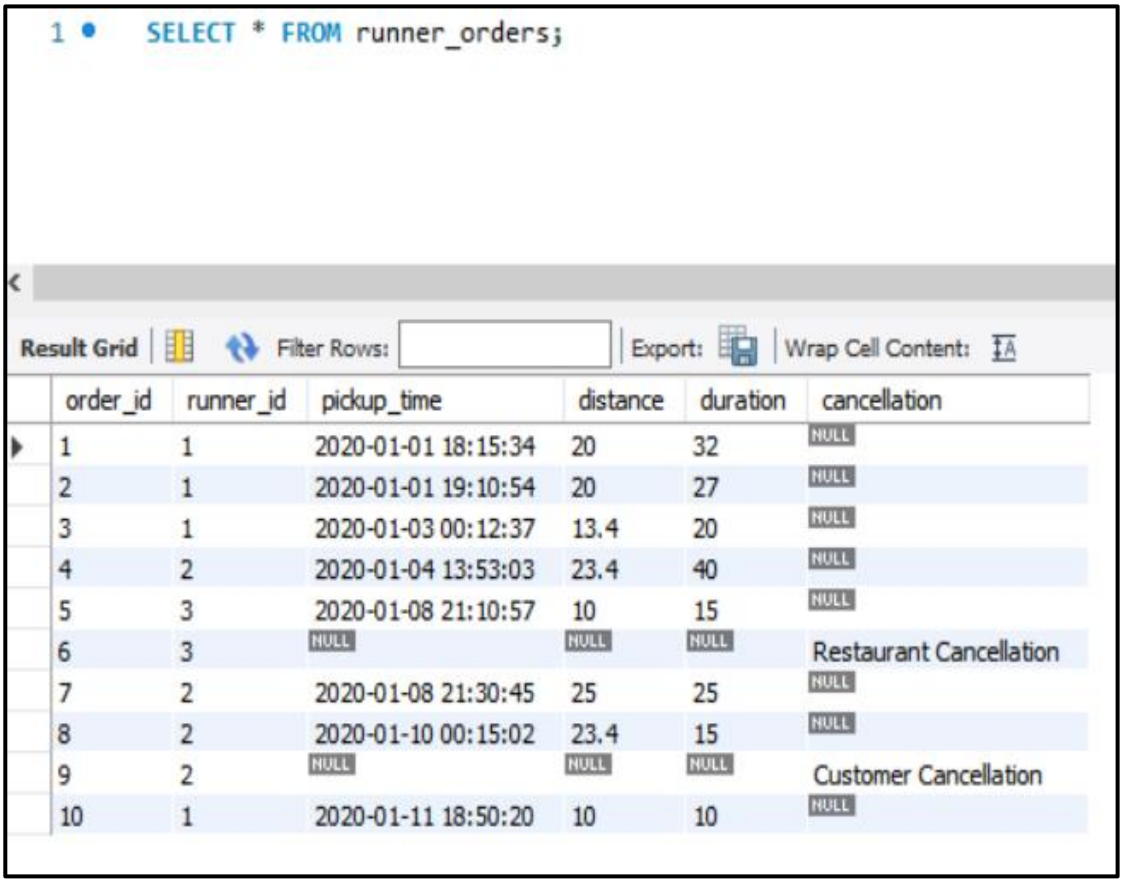
```sql
SELECT * FROM customer_orders ;
```

order_id	customer_id	pizza_id	exclusions	extras	order_time
1	101	1	NULL	NULL	2020-01-01 18:05:02
2	101	1	NULL	NULL	2020-01-01 19:00:52
3	102	1	NULL	NULL	2020-01-02 23:51:23
3	102	2	NULL	NULL	2020-01-02 23:51:23
4	103	1	4	NULL	2020-01-04 13:23:46
4	103	1	4	NULL	2020-01-04 13:23:46
4	103	2	4	NULL	2020-01-04 13:23:46
5	104	1	NULL	1	2020-01-08 21:00:29
6	101	2	NULL	NULL	2020-01-08 21:03:13
7	105	2	NULL	1	2020-01-08 21:20:29
8	102	1	NULL	NULL	2020-01-09 23:54:33
9	103	1	4	1, 5	2020-01-10 11:22:59
10	104	1	NULL	NULL	2020-01-11 18:34:49
10	104	1	2, 6	1, 4	2020-01-11 18:34:49



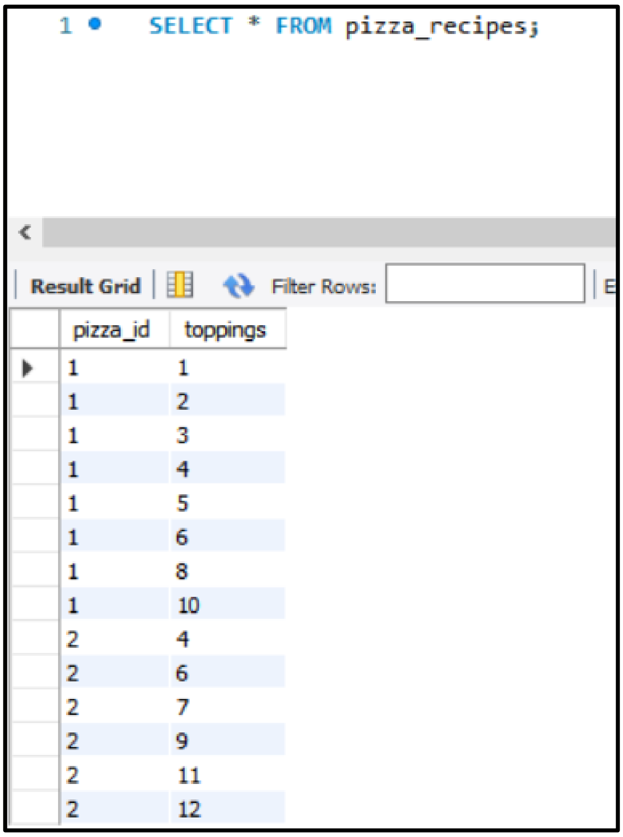
```sql
SELECT * FROM runners;
```

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



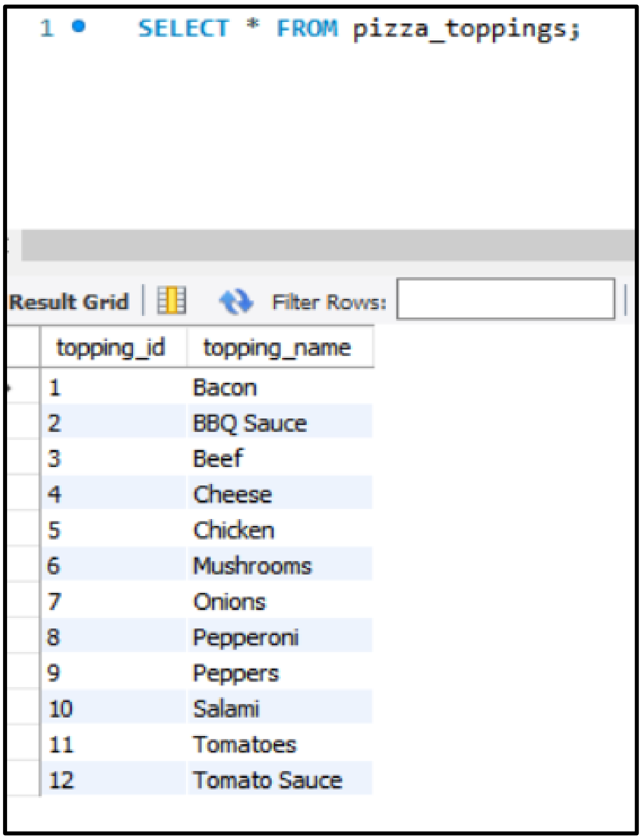
```sql
SELECT * FROM runner_orders;
```

order_id	runner_id	pickup_time	distance	duration	cancellation
1	1	2020-01-01 18:15:34	20	32	NULL
2	1	2020-01-01 19:10:54	20	27	NULL
3	1	2020-01-03 00:12:37	13.4	20	NULL
4	2	2020-01-04 13:53:03	23.4	40	NULL
5	3	2020-01-08 21:10:57	10	15	NULL
6	3	NULL	NULL	NULL	Restaurant Cancellation
7	2	2020-01-08 21:30:45	25	25	NULL
8	2	2020-01-10 00:15:02	23.4	15	NULL
9	2	NULL	NULL	NULL	Customer Cancellation
10	1	2020-01-11 18:50:20	10	10	NULL



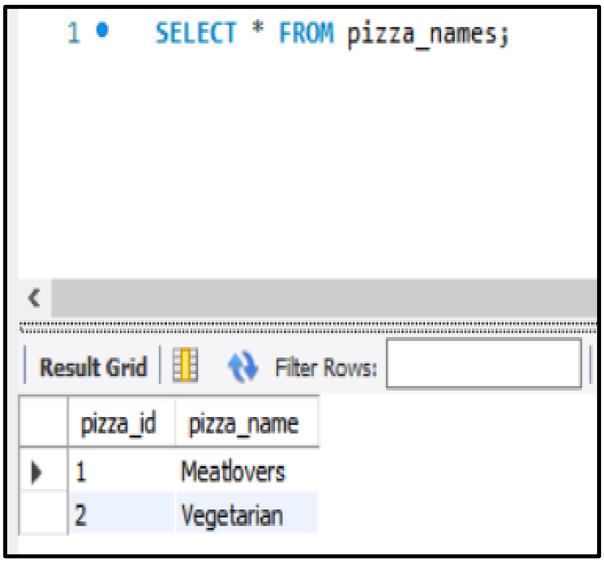
```sql
SELECT * FROM pizza_recipes;
```

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



```sql
SELECT * FROM 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



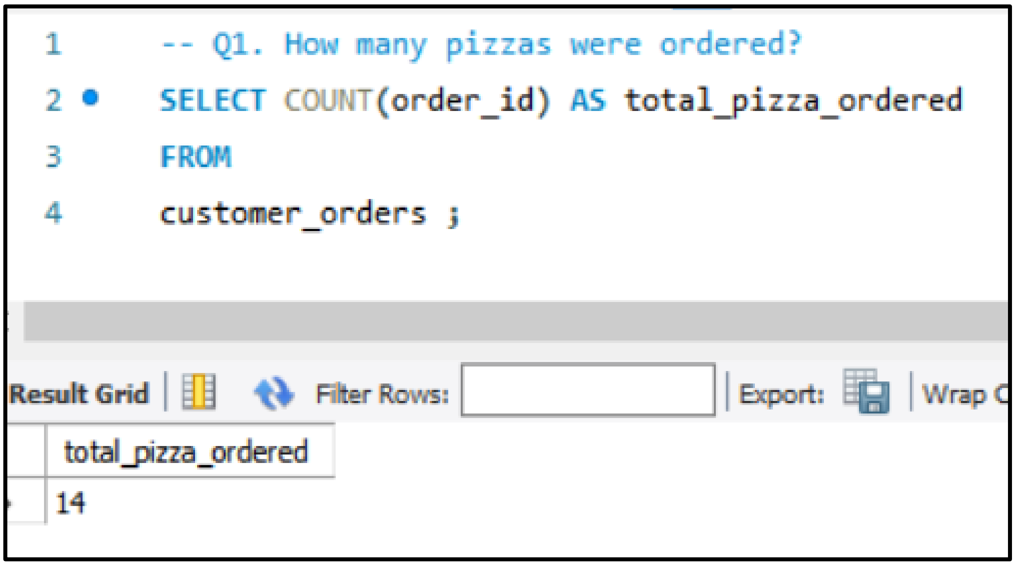
```sql
SELECT * FROM pizza_names;
```

pizza_id	pizza_name
1	Meatlovers
2	Vegetarian

Answer to Questions:

Pizza Metrics Analysis:

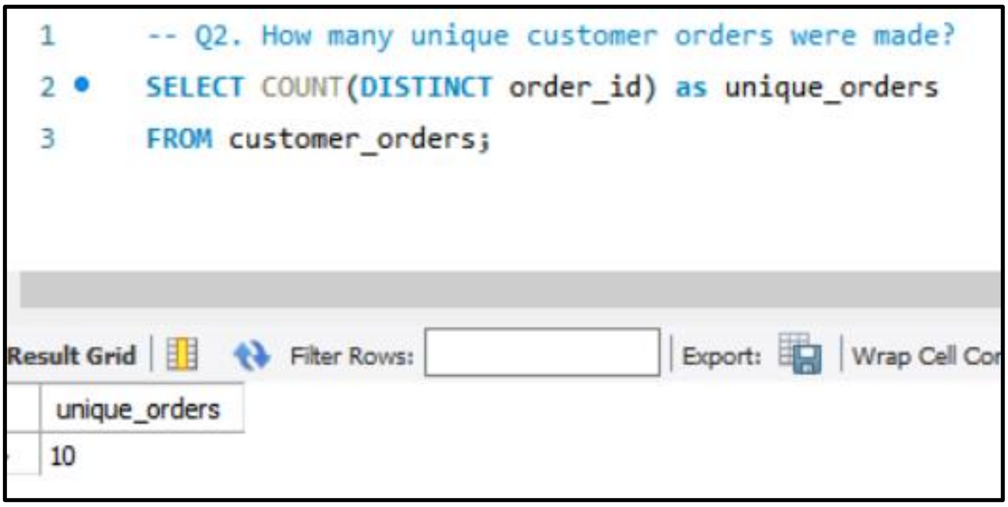
Q1. How many pizzas were ordered?



```sql
-- Q1. How many pizzas were ordered?
SELECT COUNT(order_id) AS total_pizza_ordered
FROM
customer_orders ;
```

total_pizza_ordered
14

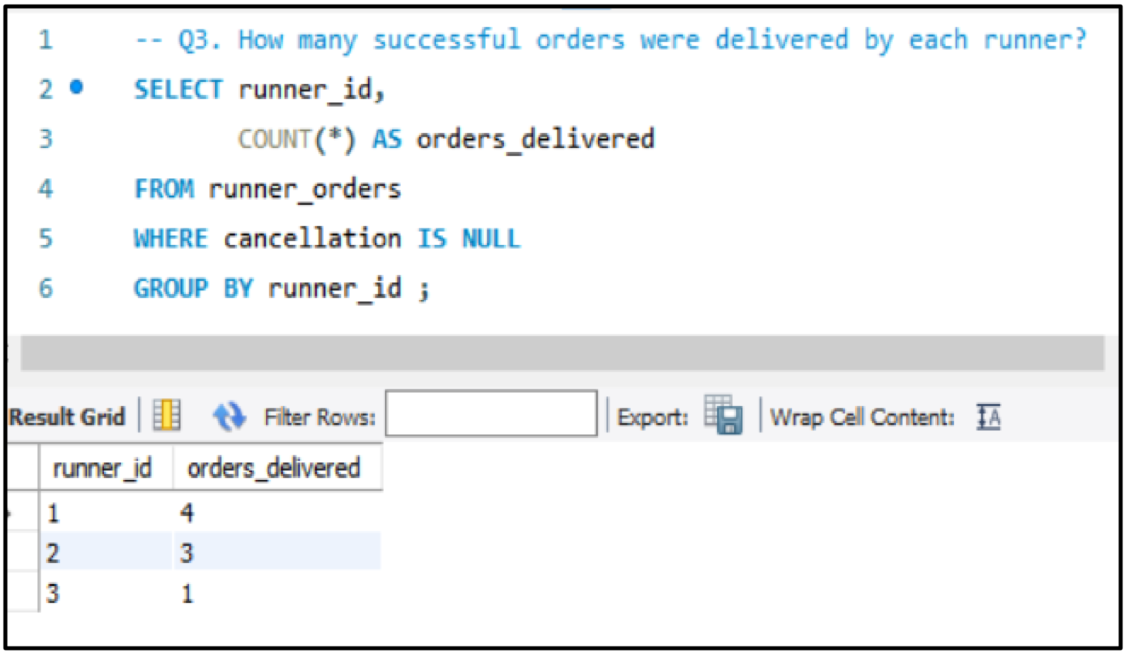
Q2. How many unique customer orders were made?



```sql
-- Q2. How many unique customer orders were made?
SELECT COUNT(DISTINCT order_id) as unique_orders
FROM customer_orders;
```

unique_orders
10

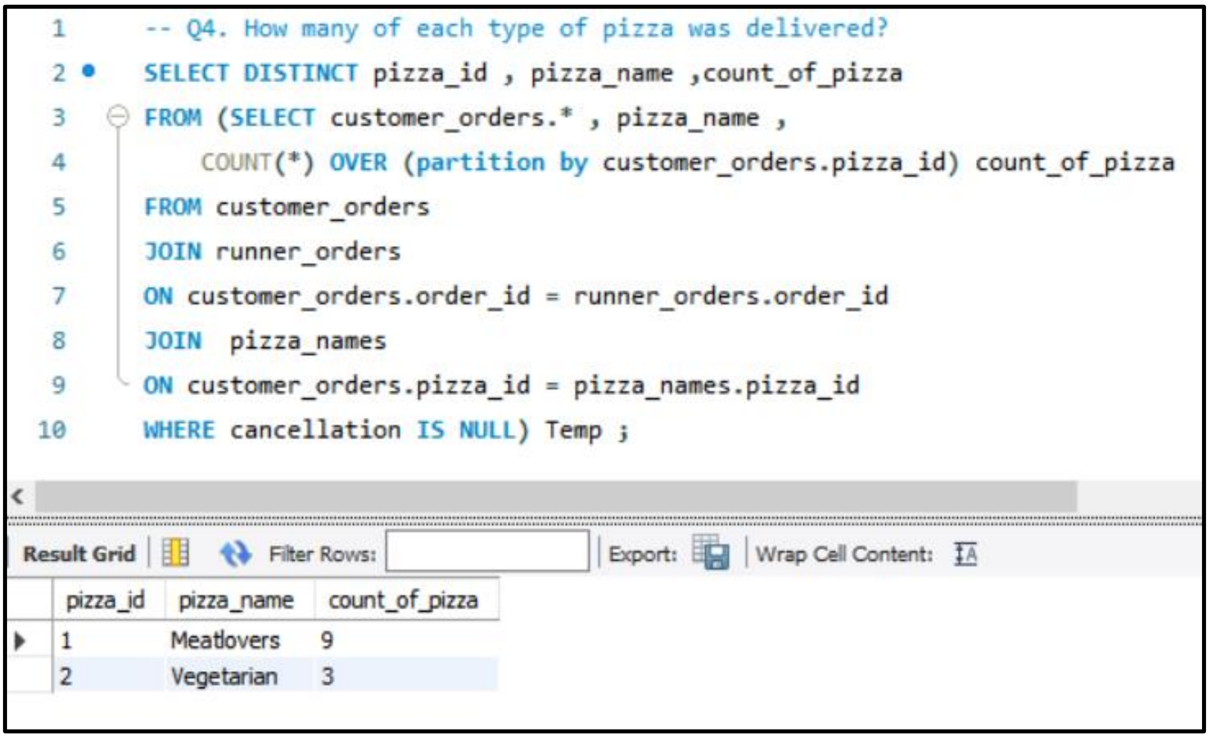
Q3. How many successful orders were delivered by each runner?



```sql
-- Q3. How many successful orders were delivered by each runner?
SELECT runner_id,
       COUNT(*) AS orders_delivered
FROM runner_orders
WHERE cancellation IS NULL
GROUP BY runner_id ;
```

runner_id	orders_delivered
1	4
2	3
3	1

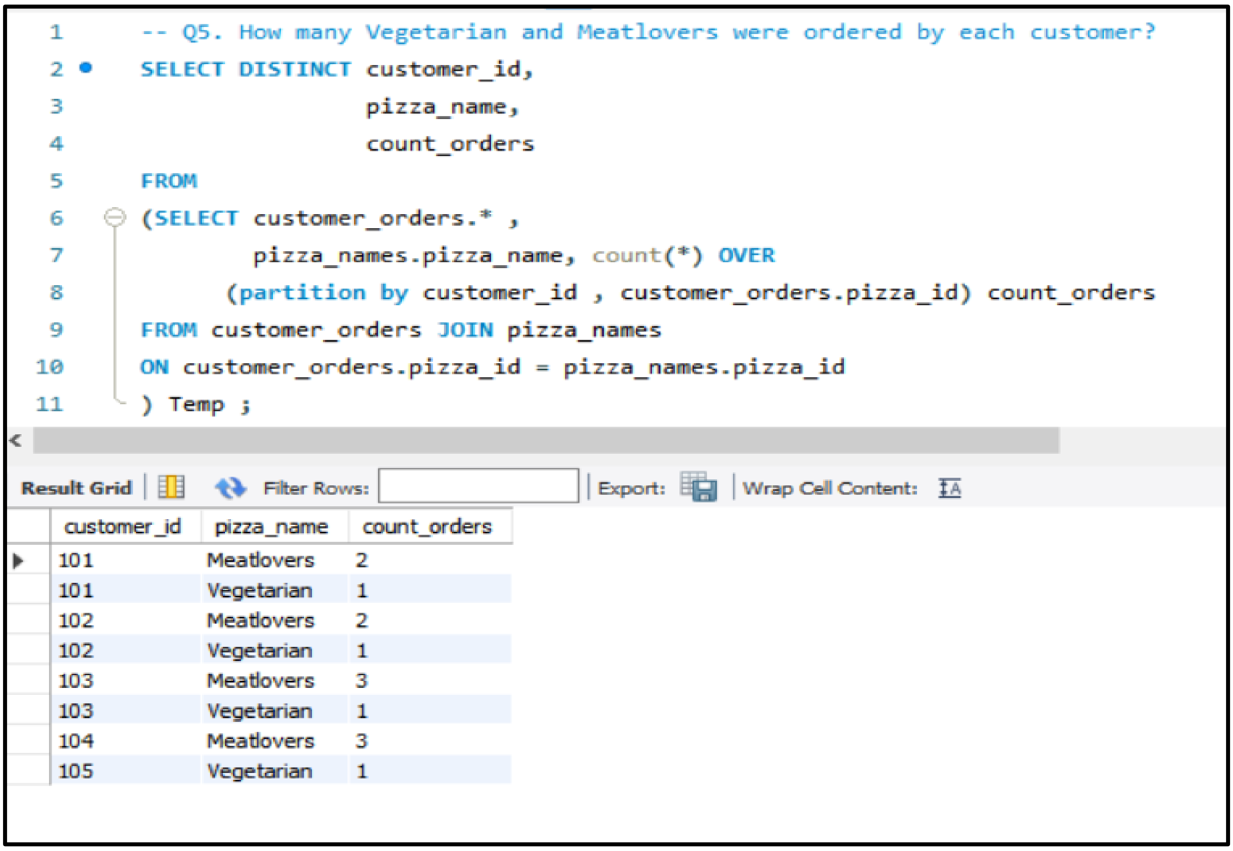
Q4. How many of each type of pizza was delivered?



```sql
-- Q4. How many of each type of pizza was delivered?
SELECT DISTINCT pizza_id , pizza_name ,count_of_pizza
FROM (SELECT customer_orders.* , pizza_name ,
    COUNT(*) OVER (partition by customer_orders.pizza_id) count_of_pizza
FROM customer_orders
JOIN runner_orders
ON customer_orders.order_id = runner_orders.order_id
JOIN  pizza_names
ON customer_orders.pizza_id = pizza_names.pizza_id
WHERE cancellation IS NULL) Temp ;
```

pizza_id	pizza_name	count_of_pizza
1	Meatlovers	9
2	Vegetarian	3

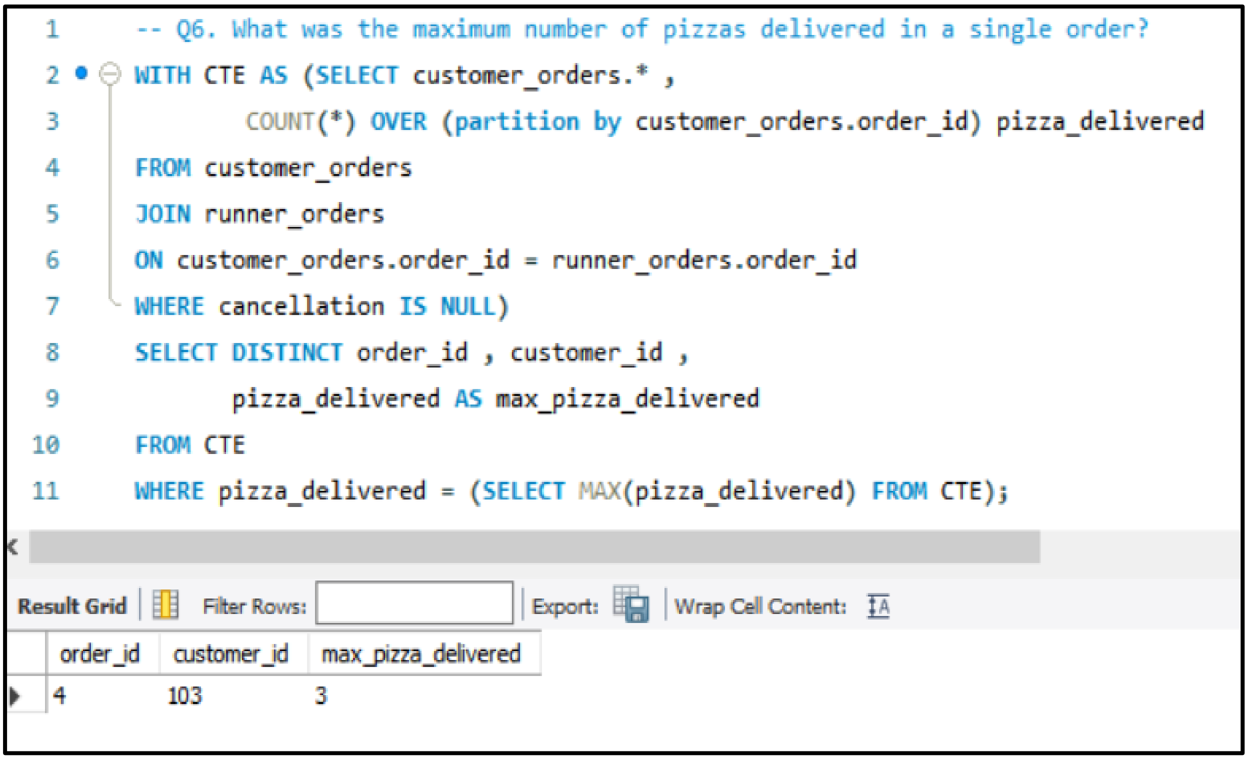
Q5. How many Vegetarian and Meat lovers were ordered by each customer?



```sql
-- Q5. How many Vegetarian and Meatlovers were ordered by each customer?
SELECT DISTINCT customer_id,
                pizza_name,
                count_orders
FROM
(SELECT customer_orders.* ,
        pizza_names.pizza_name, count(*) OVER
      (partition by customer_id , customer_orders.pizza_id) count_orders
FROM customer_orders JOIN pizza_names
ON customer_orders.pizza_id = pizza_names.pizza_id
) Temp ;
```

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

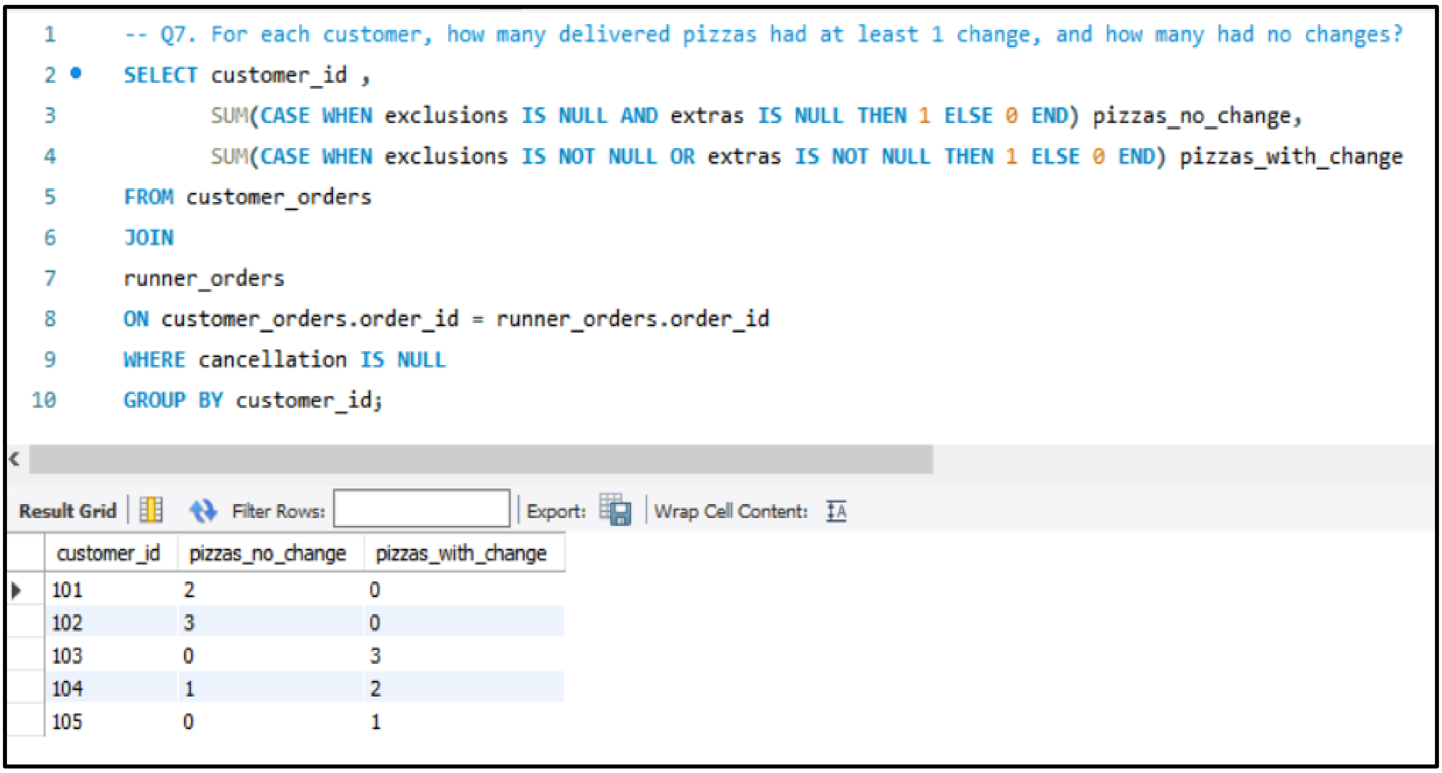
Q6 What was the maximum number of pizzas delivered in a single order?



```sql
-- Q6. What was the maximum number of pizzas delivered in a single order?
WITH CTE AS (SELECT customer_orders.* ,
        COUNT(*) OVER (partition by customer_orders.order_id) pizza_delivered
FROM customer_orders
JOIN runner_orders
ON customer_orders.order_id = runner_orders.order_id
WHERE cancellation IS NULL)
SELECT DISTINCT order_id , customer_id ,
       pizza_delivered AS max_pizza_delivered
FROM CTE
WHERE pizza_delivered = (SELECT MAX(pizza_delivered) FROM CTE);
```

order_id	customer_id	max_pizza_delivered
4	103	3

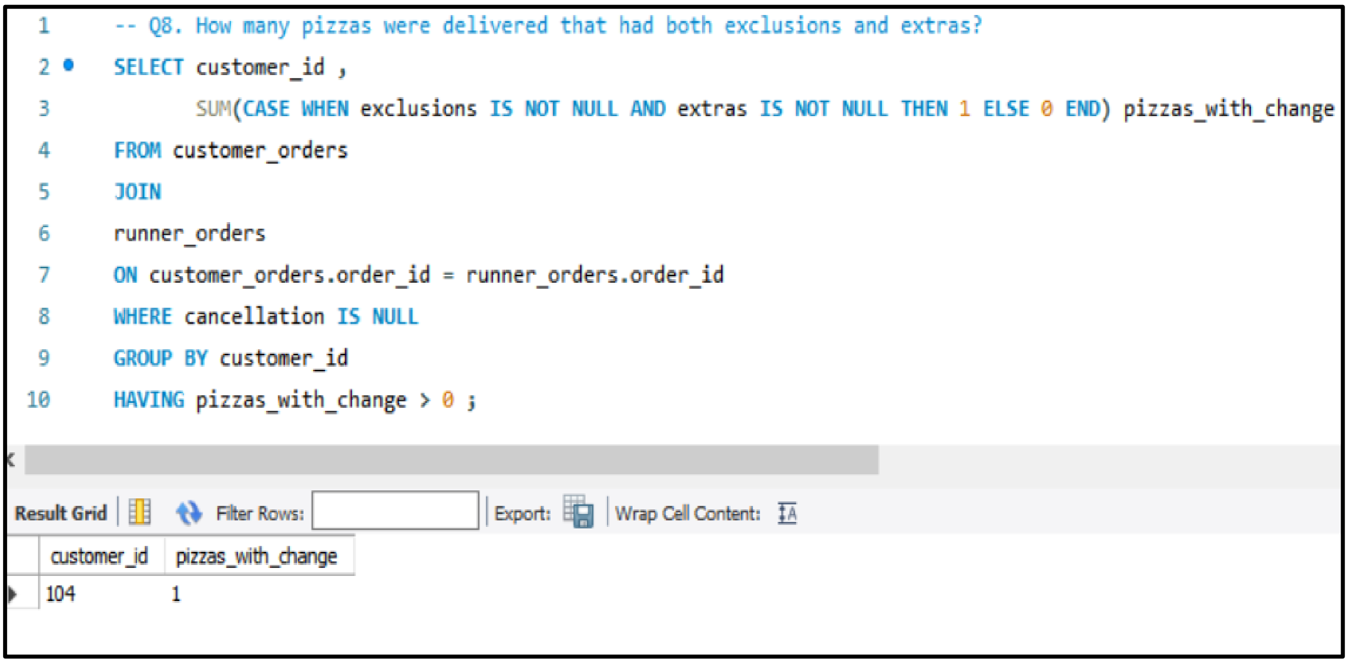
Q7. For each customer, how many delivered pizzas had at least 1 change, and how many had no changes?



```sql
-- Q7. For each customer, how many delivered pizzas had at least 1 change, and how many had no changes?
SELECT customer_id ,
       SUM(CASE WHEN exclusions IS NULL AND extras IS NULL THEN 1 ELSE 0 END) pizzas_no_change,
       SUM(CASE WHEN exclusions IS NOT NULL OR extras IS NOT NULL THEN 1 ELSE 0 END) pizzas_with_change
FROM customer_orders
JOIN
runner_orders
ON customer_orders.order_id = runner_orders.order_id
WHERE cancellation IS NULL
GROUP BY customer_id;
```

customer_id	pizzas_no_change	pizzas_with_change
101	2	0
102	3	0
103	0	3
104	1	2
105	0	1

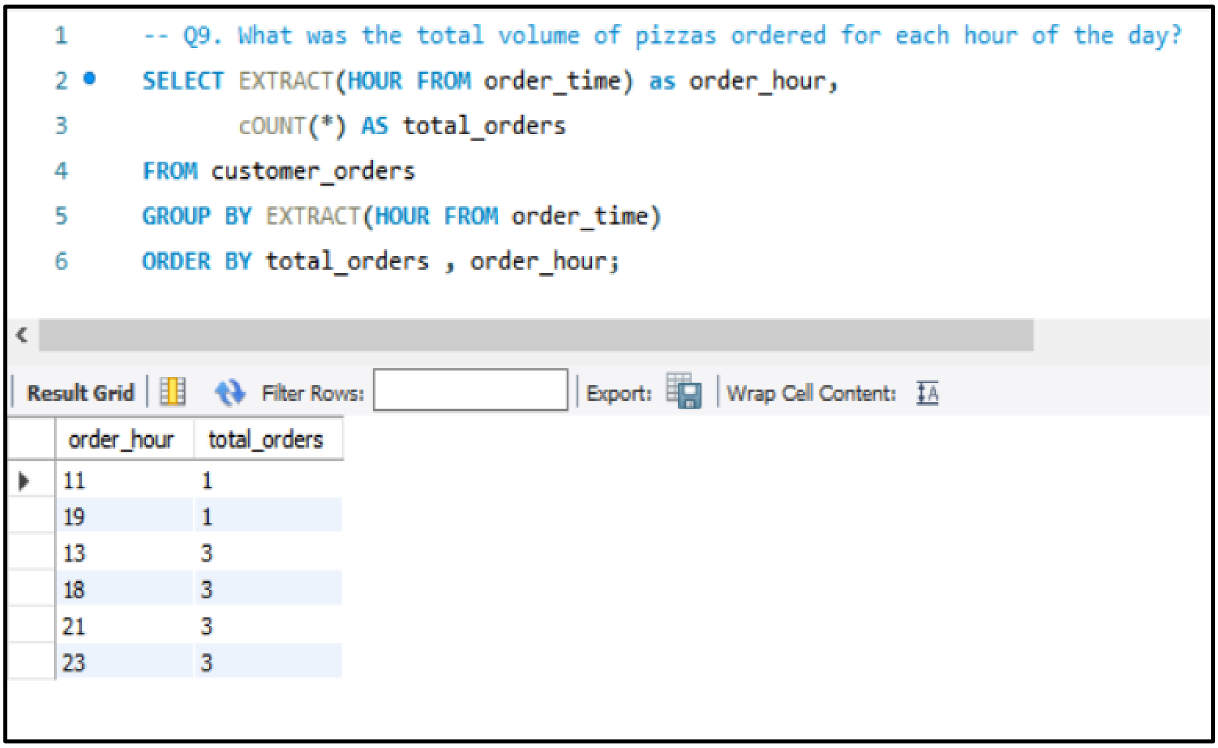
Q8 How many pizzas were delivered that had both exclusions and extras?



```
-- Q8. How many pizzas were delivered that had both exclusions and extras?
SELECT customer_id ,
       SUM(CASE WHEN exclusions IS NOT NULL AND extras IS NOT NULL THEN 1 ELSE 0 END) pizzas_with_change
FROM customer_orders
JOIN
runner_orders
ON customer_orders.order_id = runner_orders.order_id
WHERE cancellation IS NULL
GROUP BY customer_id
HAVING pizzas_with_change > 0 ;
```

customer_id	pizzas_with_change
104	1

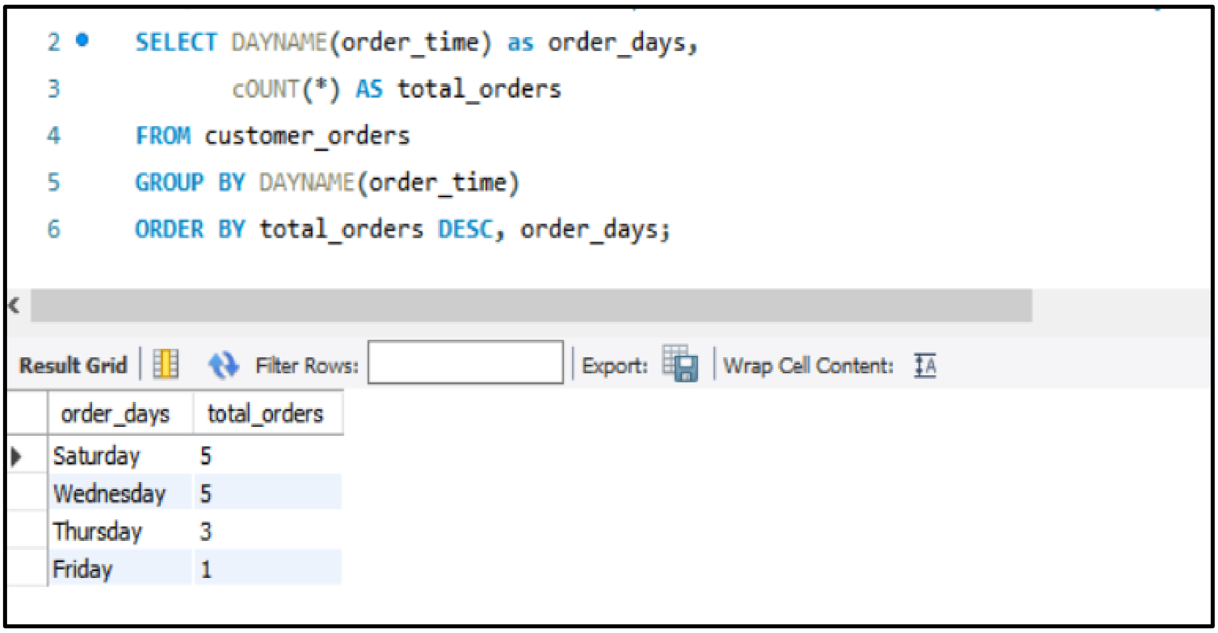
Q9. What was the total volume of pizzas ordered for each hour of the day?



```
-- Q9. What was the total volume of pizzas ordered for each hour of the day?
SELECT EXTRACT(HOUR FROM order_time) as order_hour,
       cOUNT(*) AS total_orders
FROM customer_orders
GROUP BY EXTRACT(HOUR FROM order_time)
ORDER BY total_orders , order_hour;
```

order_hour	total_orders
11	1
19	1
13	3
18	3
21	3
23	3

Q10. What was the volume of orders for each day of the week?

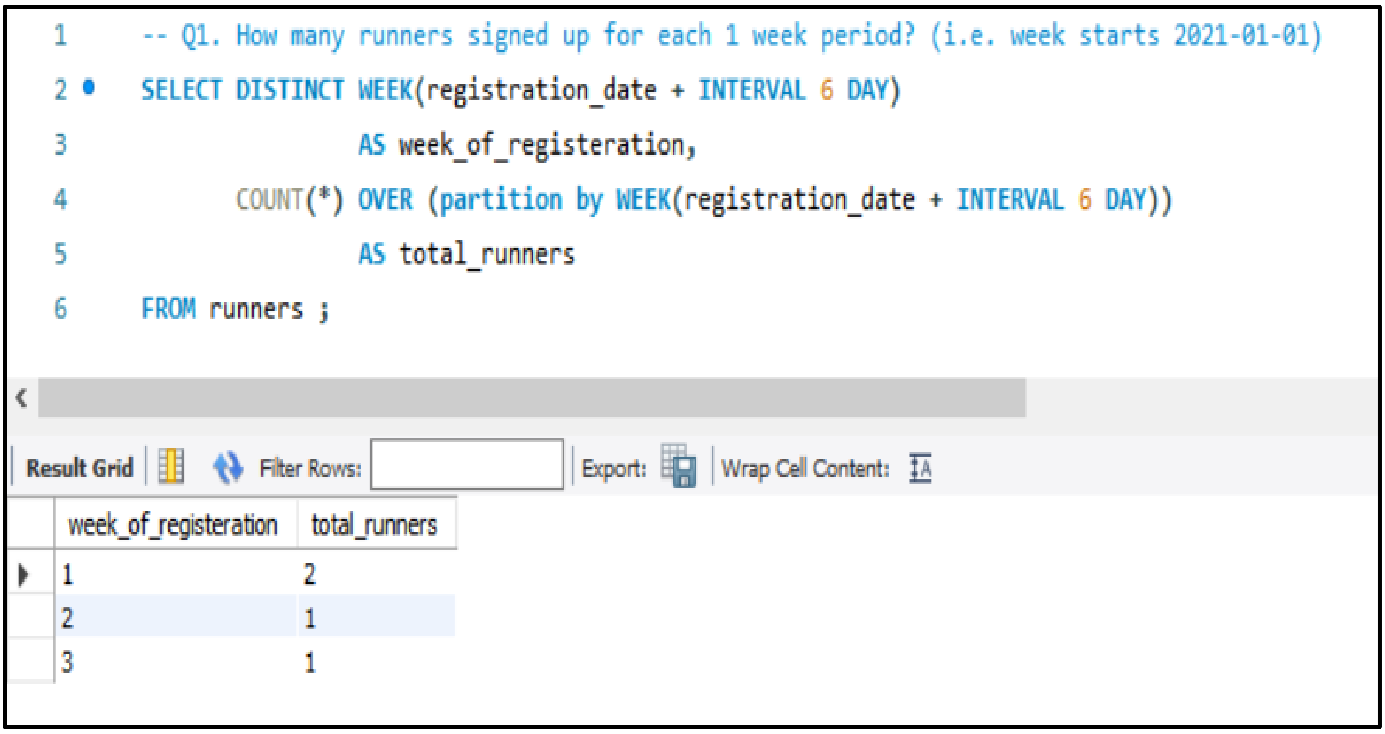


```sql
SELECT DAYNAME(order_time) as order_days,
        cOUNT(*) AS total_orders
FROM customer_orders
GROUP BY DAYNAME(order_time)
ORDER BY total_orders DESC, order_days;
```

order_days	total_orders
Saturday	5
Wednesday	5
Thursday	3
Friday	1

Runner and Customer Experience:

Q1. How many runners signed up for each 1week period? (i.e. week starts 2021-01-01)



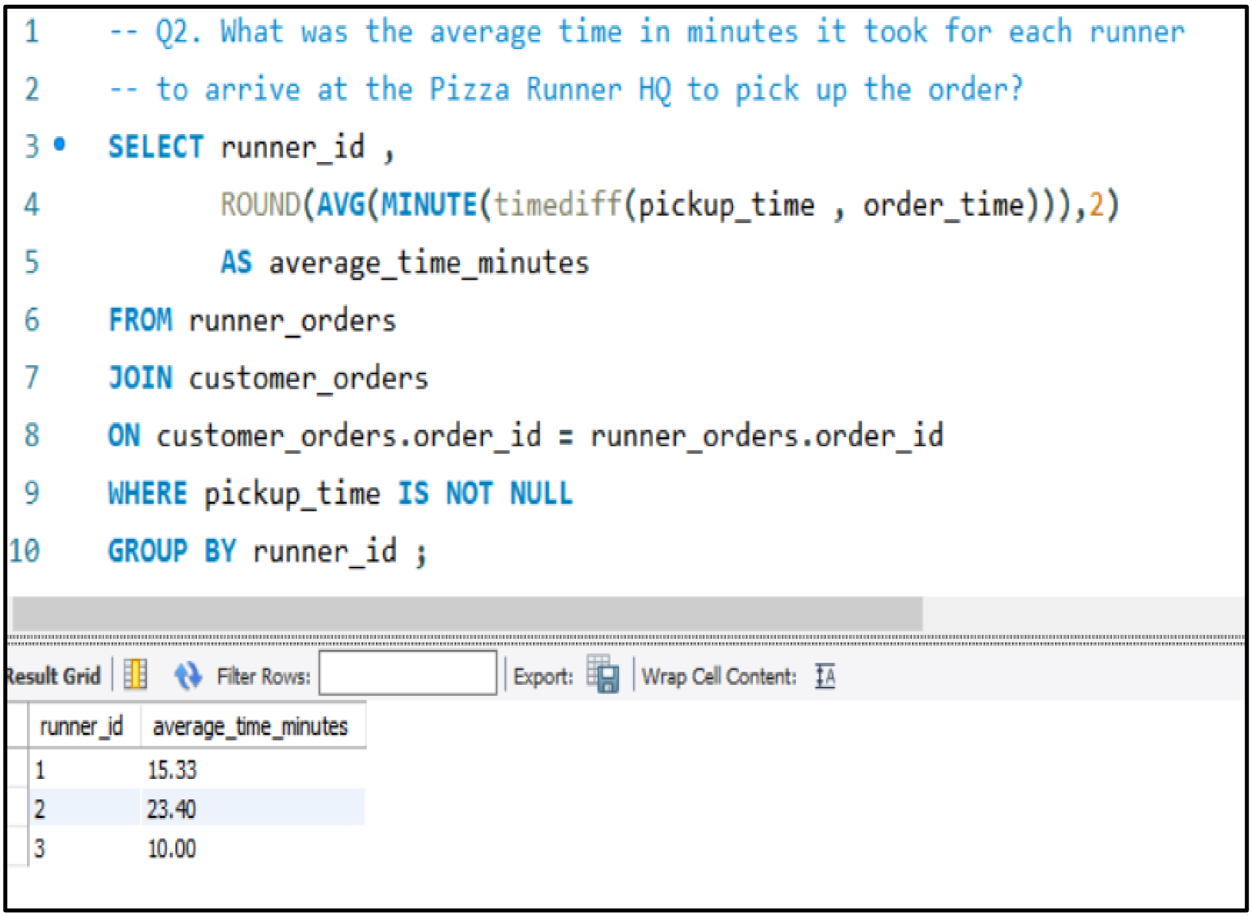
```sql
-- Q1. How many runners signed up for each 1 week period? (i.e. week starts 2021-01-01)
SELECT DISTINCT WEEK(registration_date + INTERVAL 6 DAY)
                AS week_of_registeration,
        COUNT(*) OVER (partition by WEEK(registration_date + INTERVAL 6 DAY))
                AS total_runners
FROM runners ;
```

week_of_registeration	total_runners
1	2
2	1
3	1

EXPLANATION:

- ➔ Since the week started from ‘01-01-2021’, for first week we observed 2 registrations on ‘01-01-2021’ and ‘03-01-2021’, for second week there was 1 registration on ‘08-01-2021’ and finally for third week there was 1 registration on ‘15-01-2021’

Q2. What was the average time in minutes it took for each runner to arrive at the Pizza Runner HQ to pick up the order?



```sql
-- Q2. What was the average time in minutes it took for each runner
-- to arrive at the Pizza Runner HQ to pick up the order?
SELECT runner_id ,
       ROUND(AVG(MINUTE(timediff(pickup_time , order_time))),2)
       AS average_time_minutes
FROM runner_orders
JOIN customer_orders
ON customer_orders.order_id = runner_orders.order_id
WHERE pickup_time IS NOT NULL
GROUP BY runner_id ;
```

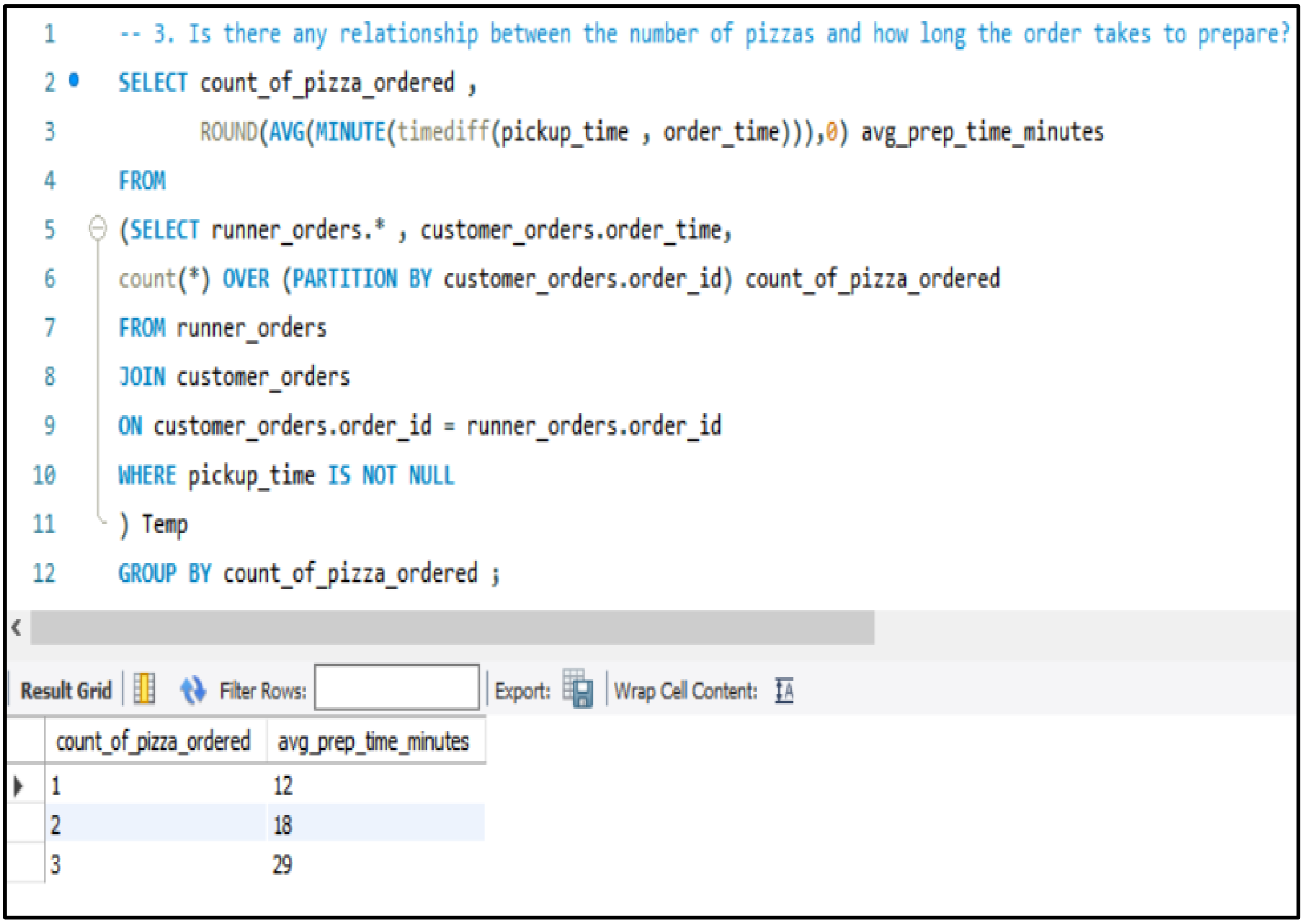
runner_id	average_time_minutes
1	15.33
2	23.40
3	10.00

Runner 1 took on average 15 minutes

Runner 2 took on average 23 minutes

Runner 3 took around 10 minutes on average (least time) to arrive at Pizza Runner HQ to pick up the order .

Q3. Is there any relationship between the number of pizzas and how long the order takes to prepare?



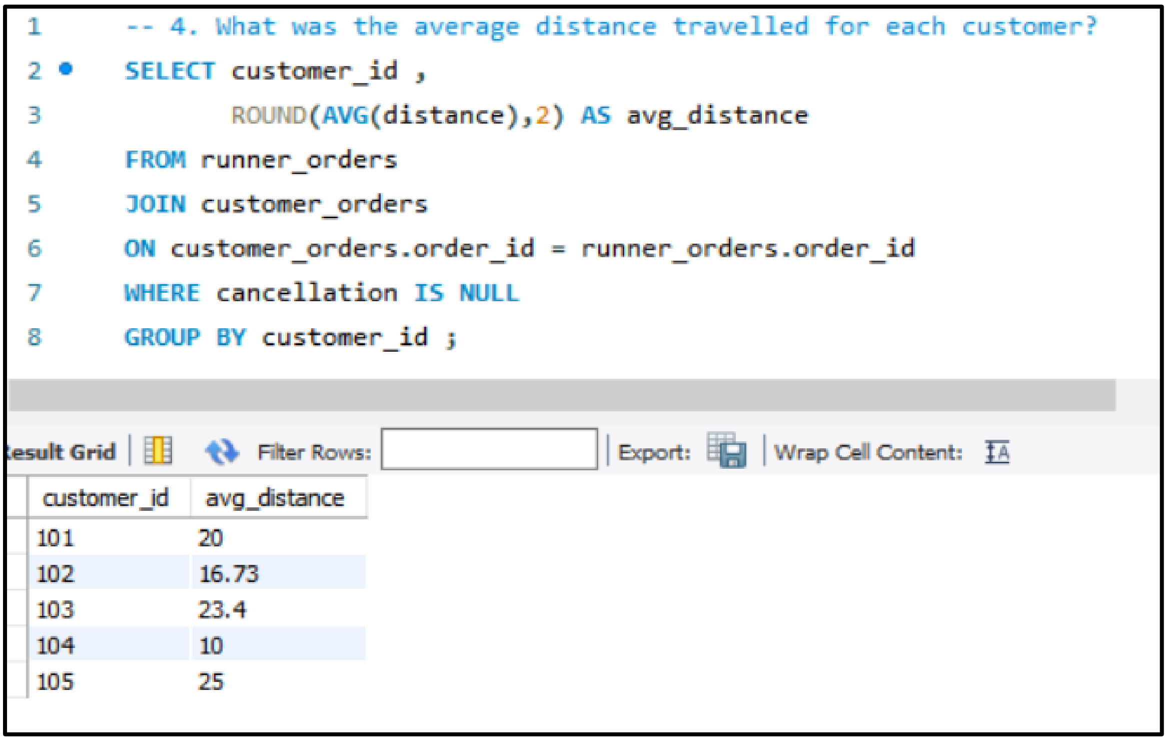
```sql
-- 3. Is there any relationship between the number of pizzas and how long the order takes to prepare?
SELECT count_of_pizza_ordered ,
        ROUND(AVG(MINUTE(timediff(pickup_time , order_time))),0) avg_prep_time_minutes
FROM
(SELECT runner_orders.* , customer_orders.order_time,
count(*) OVER (PARTITION BY customer_orders.order_id) count_of_pizza_ordered
FROM runner_orders
JOIN customer_orders
ON customer_orders.order_id = runner_orders.order_id
WHERE pickup_time IS NOT NULL
) Temp
GROUP BY count_of_pizza_ordered ;
```

count_of_pizza_ordered	avg_prep_time_minutes
1	12
2	18
3	29

RELATIONSHIP

-> More the quantity of pizza ordered, more will be the preparation time. Orders with a single pizza can be prepared with an average time of 12 minutes while orders having 3 pizzas take about around half an hour for preparation.

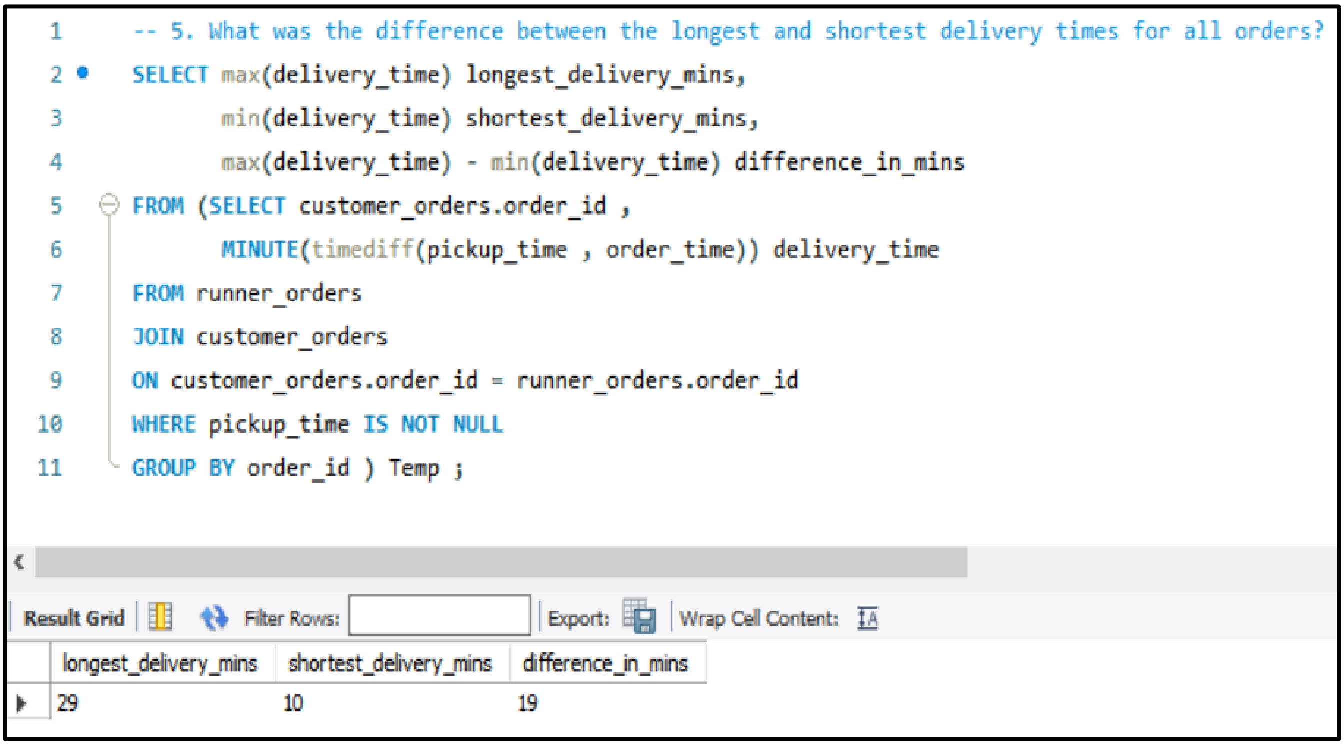
Q4. What was the average distance travelled for each customer?



```sql
-- 4. What was the average distance travelled for each customer?
SELECT customer_id ,
        ROUND(AVG(distance),2) AS avg_distance
FROM runner_orders
JOIN customer_orders
ON customer_orders.order_id = runner_orders.order_id
WHERE cancellation IS NULL
GROUP BY customer_id ;
```

customer_id	avg_distance
101	20
102	16.73
103	23.4
104	10
105	25

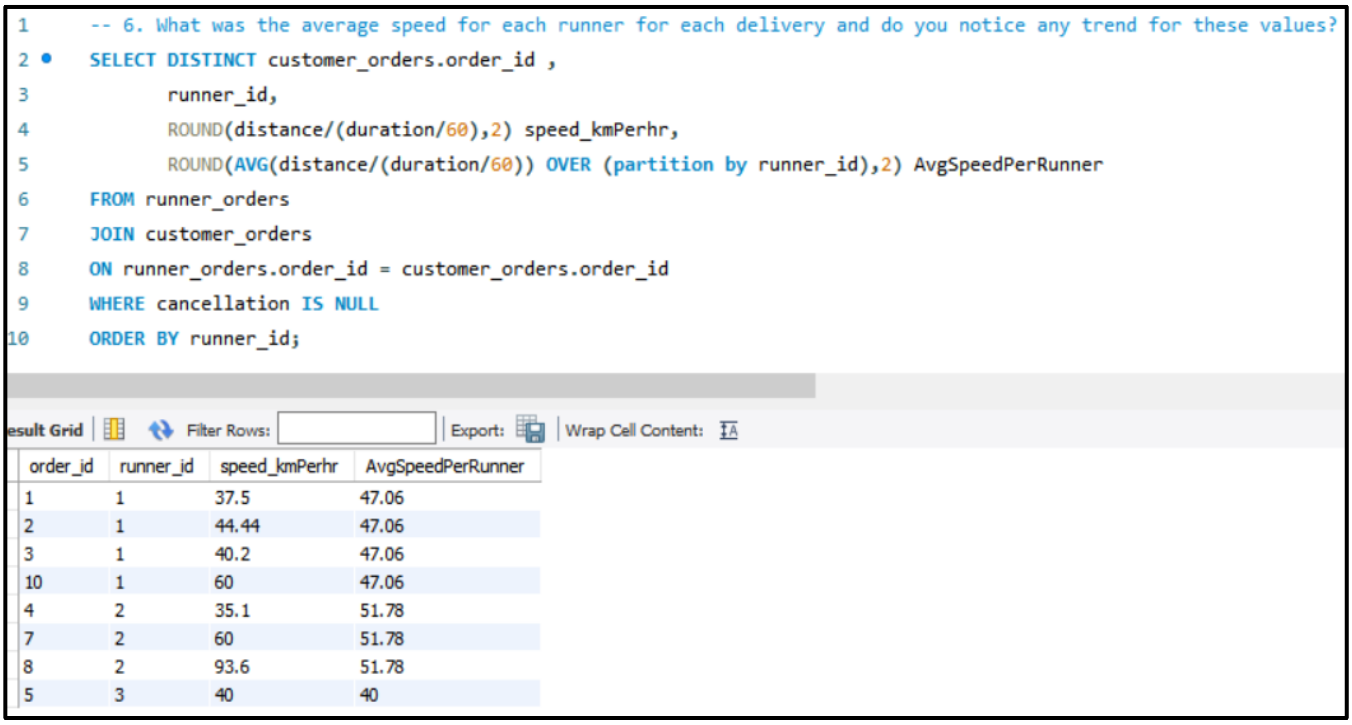
Q5. What was the difference between the longest and shortest delivery times for all orders?



```sql
-- 5. What was the difference between the longest and shortest delivery times for all orders?
SELECT max(delivery_time) longest_delivery_mins,
       min(delivery_time) shortest_delivery_mins,
       max(delivery_time) - min(delivery_time) difference_in_mins
FROM (SELECT customer_orders.order_id ,
       MINUTE(timediff(pickup_time , order_time)) delivery_time
FROM runner_orders
JOIN customer_orders
ON customer_orders.order_id = runner_orders.order_id
WHERE pickup_time IS NOT NULL
GROUP BY order_id ) Temp ;
```

longest_delivery_mins	shortest_delivery_mins	difference_in_mins
29	10	19

Q6. What was the average speed for each runner for each delivery and do you notice any trend for these values?

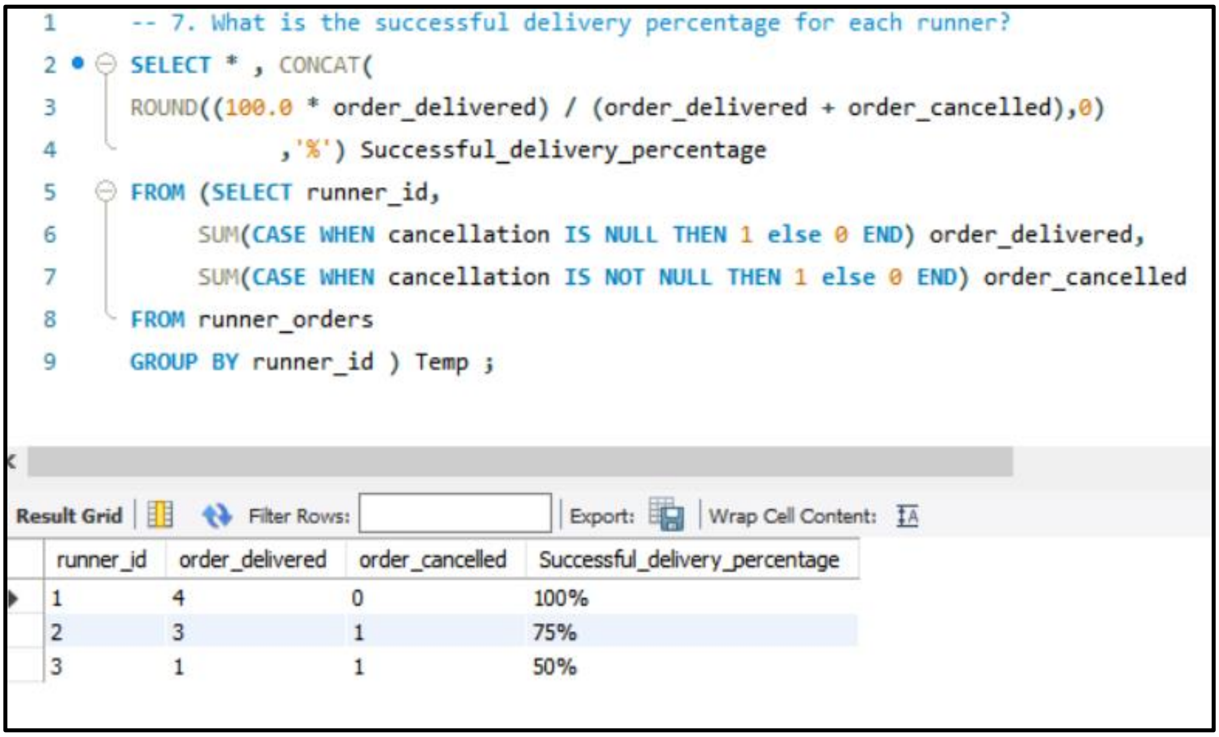


```sql
-- 6. What was the average speed for each runner for each delivery and do you notice any trend for these values?
SELECT DISTINCT customer_orders.order_id ,
        runner_id,
        ROUND(distance/(duration/60),2) speed_kmPerhr,
        ROUND(AVG(distance/(duration/60)) OVER (partition by runner_id),2) AvgSpeedPerRunner
FROM runner_orders
JOIN customer_orders
ON runner_orders.order_id = customer_orders.order_id
WHERE cancellation IS NULL
ORDER BY runner_id;
```

order_id	runner_id	speed_kmPerhr	AvgSpeedPerRunner
1	1	37.5	47.06
2	1	44.44	47.06
3	1	40.2	47.06
10	1	60	47.06
4	2	35.1	51.78
7	2	60	51.78
8	2	93.6	51.78
5	3	40	40

TREND: For runner id 3 having least average speed, we observe least deliveries. Comparatively runner id 1 and 2 completed more deliveries because they have higher average speed

7. What is the successful delivery percentage for each runner?

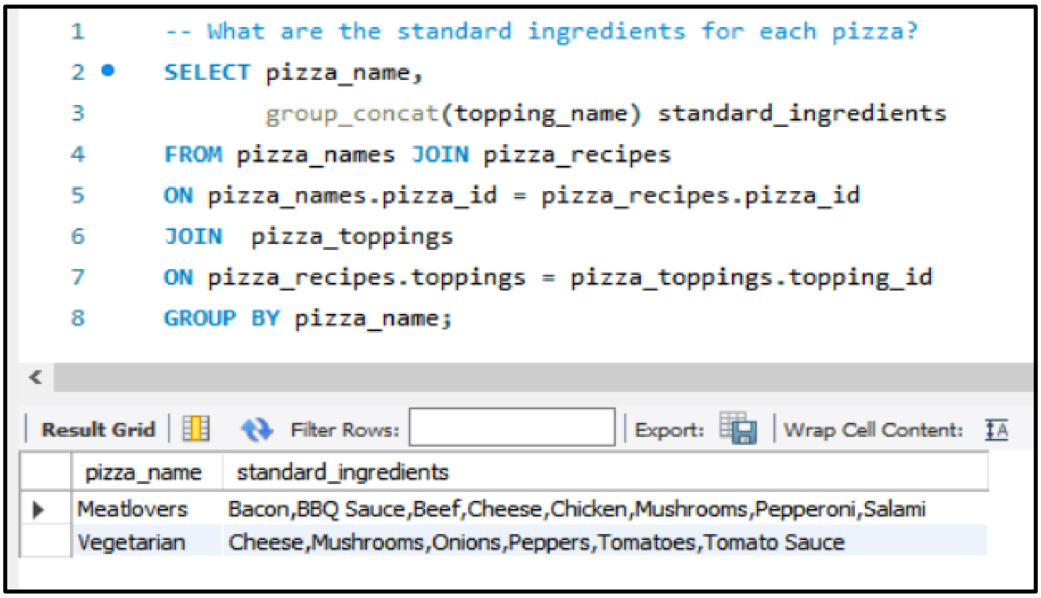


```sql
-- 7. What is the successful delivery percentage for each runner?
SELECT * , CONCAT(
ROUND((100.0 * order_delivered) / (order_delivered + order_cancelled),0)
            ,'%') Successful_delivery_percentage
FROM (SELECT runner_id,
        SUM(CASE WHEN cancellation IS NULL THEN 1 else 0 END) order_delivered,
        SUM(CASE WHEN cancellation IS NOT NULL THEN 1 else 0 END) order_cancelled
FROM runner_orders
GROUP BY runner_id ) Temp ;
```

runner_id	order_delivered	order_cancelled	Successful_delivery_percentage
1	4	0	100%
2	3	1	75%
3	1	1	50%

C. Ingredient Optimisation

Q1. What are the standard ingredients for each pizza?

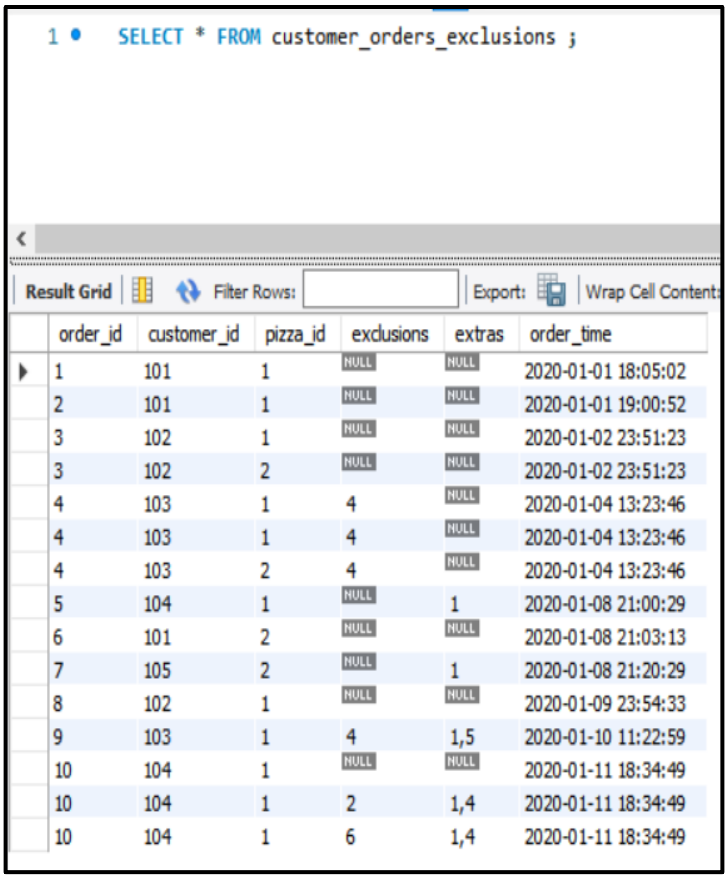


```sql
-- What are the standard ingredients for each pizza?
SELECT pizza_name,
       group_concat(topping_name) standard_ingredients
FROM pizza_names JOIN pizza_recipes
ON pizza_names.pizza_id = pizza_recipes.pizza_id
JOIN  pizza_toppings
ON pizza_recipes.toppings = pizza_toppings.topping_id
GROUP BY pizza_name;
```

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

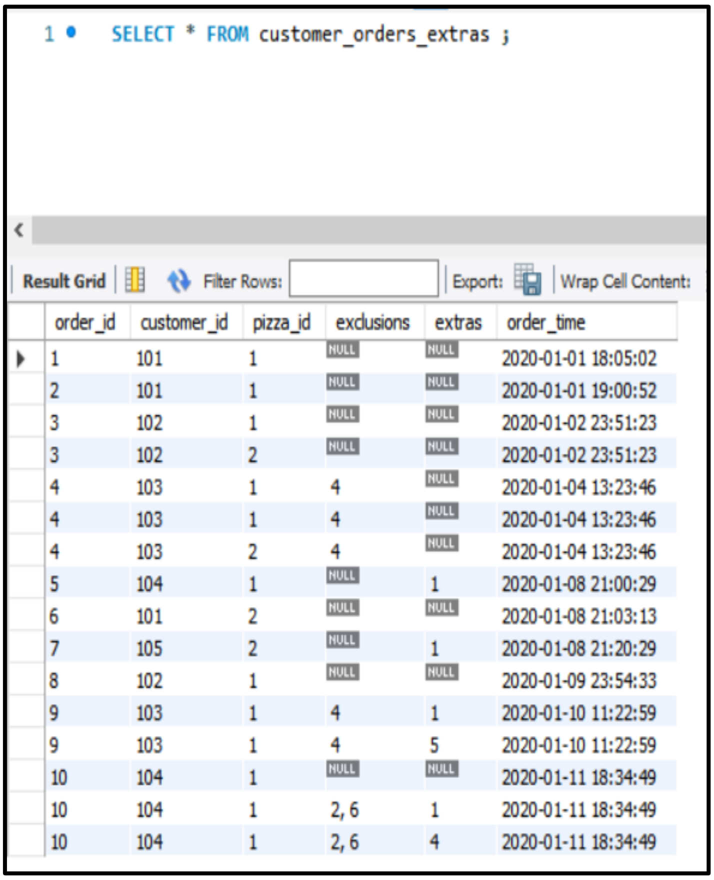
Q2. What was the most commonly added extra?

To solve this, I first normalized customer orders table.



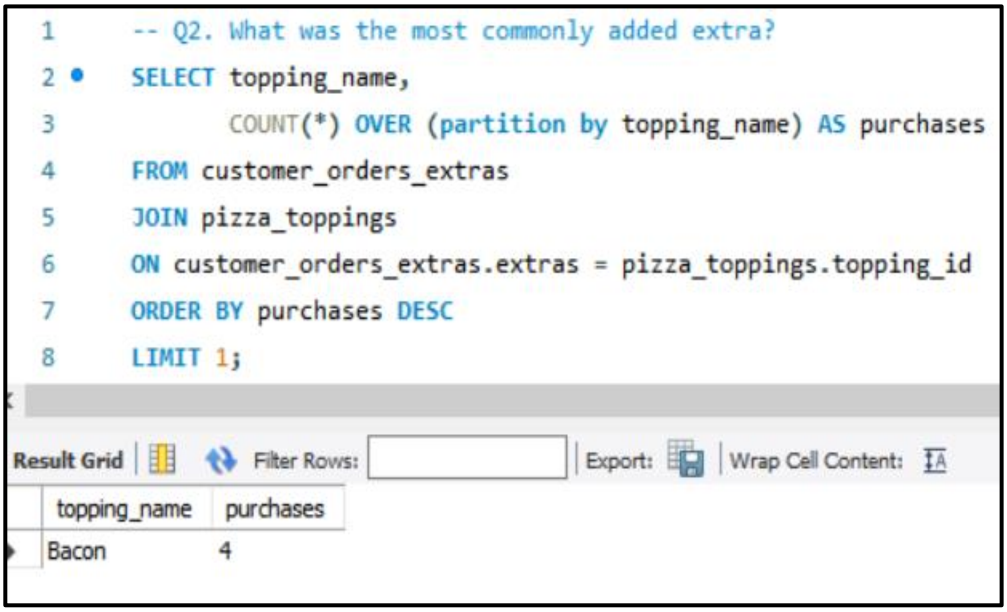
```sql
SELECT * FROM customer_orders_exclusions ;
```

order_id	customer_id	pizza_id	exclusions	extras	order_time
1	101	1	NULL	NULL	2020-01-01 18:05:02
2	101	1	NULL	NULL	2020-01-01 19:00:52
3	102	1	NULL	NULL	2020-01-02 23:51:23
3	102	2	NULL	NULL	2020-01-02 23:51:23
4	103	1	4	NULL	2020-01-04 13:23:46
4	103	1	4	NULL	2020-01-04 13:23:46
4	103	2	4	NULL	2020-01-04 13:23:46
5	104	1	NULL	1	2020-01-08 21:00:29
6	101	2	NULL	NULL	2020-01-08 21:03:13
7	105	2	NULL	1	2020-01-08 21:20:29
8	102	1	NULL	NULL	2020-01-09 23:54:33
9	103	1	4	1,5	2020-01-10 11:22:59
10	104	1	NULL	NULL	2020-01-11 18:34:49
10	104	1	2	1,4	2020-01-11 18:34:49
10	104	1	6	1,4	2020-01-11 18:34:49



```sql
SELECT * FROM customer_orders_extras ;
```

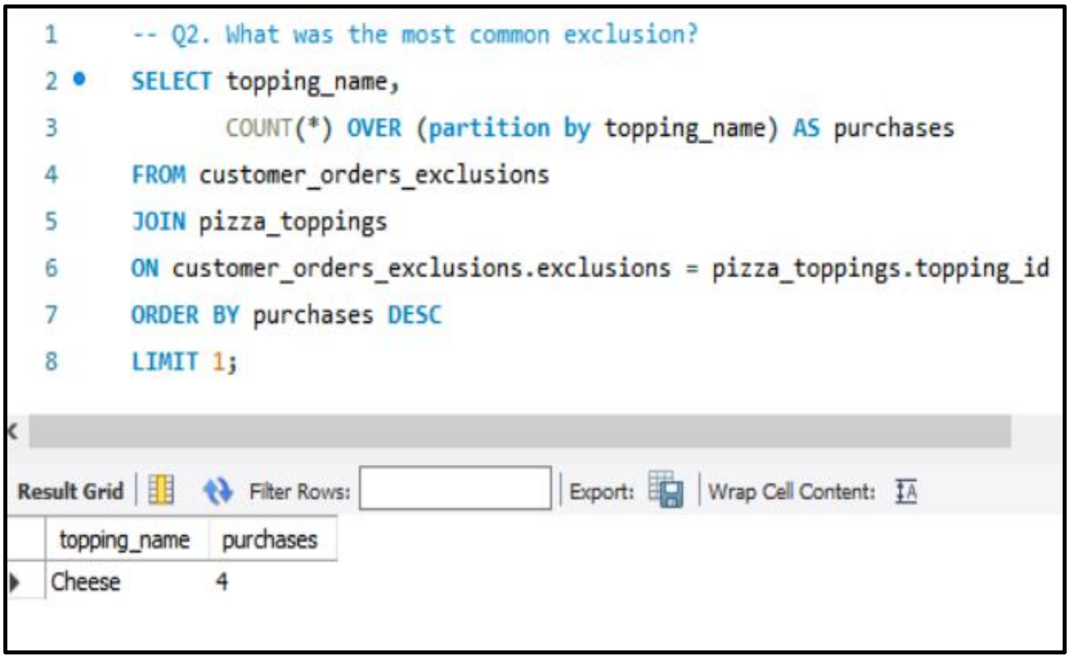
order_id	customer_id	pizza_id	exclusions	extras	order_time
1	101	1	NULL	NULL	2020-01-01 18:05:02
2	101	1	NULL	NULL	2020-01-01 19:00:52
3	102	1	NULL	NULL	2020-01-02 23:51:23
3	102	2	NULL	NULL	2020-01-02 23:51:23
4	103	1	4	NULL	2020-01-04 13:23:46
4	103	1	4	NULL	2020-01-04 13:23:46
4	103	2	4	NULL	2020-01-04 13:23:46
5	104	1	NULL	1	2020-01-08 21:00:29
6	101	2	NULL	NULL	2020-01-08 21:03:13
7	105	2	NULL	1	2020-01-08 21:20:29
8	102	1	NULL	NULL	2020-01-09 23:54:33
9	103	1	4	1	2020-01-10 11:22:59
9	103	1	4	5	2020-01-10 11:22:59
10	104	1	NULL	NULL	2020-01-11 18:34:49
10	104	1	2, 6	1	2020-01-11 18:34:49
10	104	1	2, 6	4	2020-01-11 18:34:49



```sql
-- Q2. What was the most commonly added extra?
SELECT topping_name,
        COUNT(*) OVER (partition by topping_name) AS purchases
FROM customer_orders_extras
JOIN pizza_toppings
ON customer_orders_extras.extras = pizza_toppings.topping_id
ORDER BY purchases DESC
LIMIT 1;
```

```
topping_name   purchases
Bacon          4
```

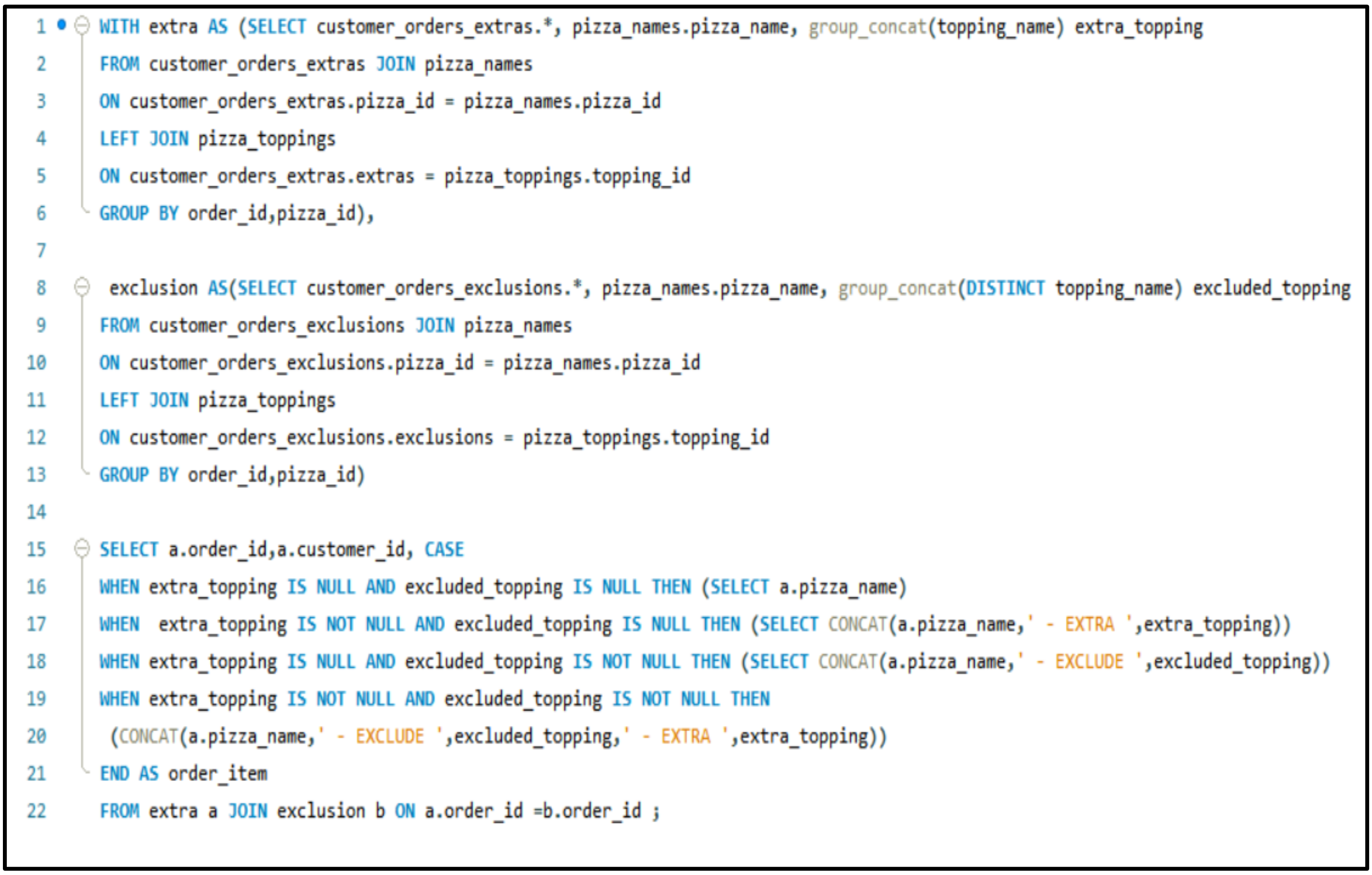
Q3. What was the most common exclusion?



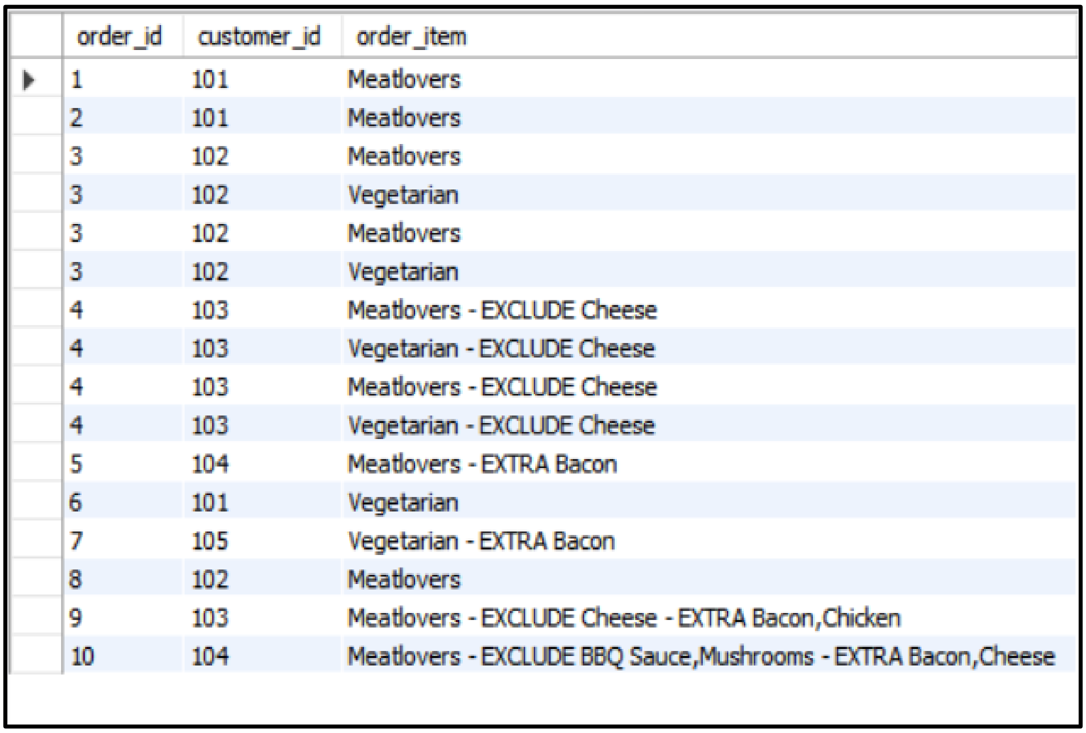
```sql
-- Q2. What was the most common exclusion?
SELECT topping_name,
        COUNT(*) OVER (partition by topping_name) AS purchases
FROM customer_orders_exclusions
JOIN pizza_toppings
ON customer_orders_exclusions.exclusions = pizza_toppings.topping_id
ORDER BY purchases DESC
LIMIT 1;
```

```
topping_name   purchases
Cheese         4
```

4. Generate an order item for each record in the customer orders table in the format of one of the following: Meat Lovers , Meat Lovers - Exclude Beef , Meat Lovers - Extra Bacon

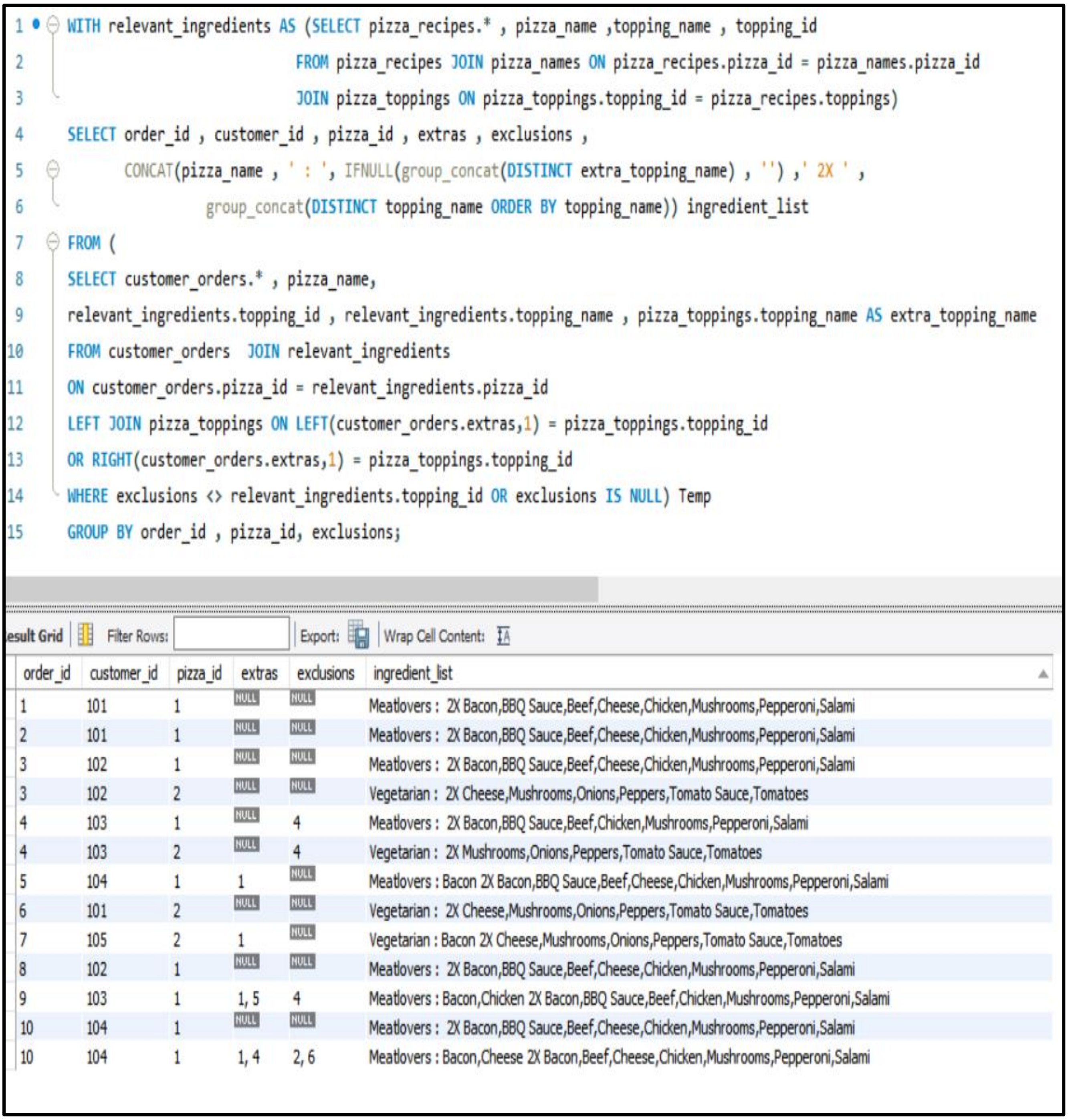


```sql
WITH extra AS (SELECT customer_orders_extras.*, pizza_names.pizza_name, group_concat(topping_name) extra_topping
FROM customer_orders_extras JOIN pizza_names
ON customer_orders_extras.pizza_id = pizza_names.pizza_id
LEFT JOIN pizza_toppings
ON customer_orders_extras.extras = pizza_toppings.topping_id
GROUP BY order_id,pizza_id),

 exclusion AS(SELECT customer_orders_exclusions.*, pizza_names.pizza_name, group_concat(DISTINCT topping_name) excluded_topping
FROM customer_orders_exclusions JOIN pizza_names
ON customer_orders_exclusions.pizza_id = pizza_names.pizza_id
LEFT JOIN pizza_toppings
ON customer_orders_exclusions.exclusions = pizza_toppings.topping_id
GROUP BY order_id,pizza_id)

SELECT a.order_id,a.customer_id, CASE
WHEN extra_topping IS NULL AND excluded_topping IS NULL THEN (SELECT a.pizza_name)
WHEN  extra_topping IS NOT NULL AND excluded_topping IS NULL THEN (SELECT CONCAT(a.pizza_name,' - EXTRA ',extra_topping))
WHEN extra_topping IS NULL AND excluded_topping IS NOT NULL THEN (SELECT CONCAT(a.pizza_name,' - EXCLUDE ',excluded_topping))
WHEN extra_topping IS NOT NULL AND excluded_topping IS NOT NULL THEN
 (CONCAT(a.pizza_name,' - EXCLUDE ',excluded_topping,' - EXTRA ',extra_topping))
END AS order_item
FROM extra a JOIN exclusion b ON a.order_id =b.order_id ;
```



order_id	customer_id	order_item
1	101	Meatlovers
2	101	Meatlovers
3	102	Meatlovers
3	102	Vegetarian
3	102	Meatlovers
3	102	Vegetarian
4	103	Meatlovers - EXCLUDE Cheese
4	103	Vegetarian - EXCLUDE Cheese
4	103	Meatlovers - EXCLUDE Cheese
4	103	Vegetarian - EXCLUDE Cheese
5	104	Meatlovers - EXTRA Bacon
6	101	Vegetarian
7	105	Vegetarian - EXTRA Bacon
8	102	Meatlovers
9	103	Meatlovers - EXCLUDE Cheese - EXTRA Bacon,Chicken
10	104	Meatlovers - EXCLUDE BBQ Sauce,Mushrooms - EXTRA Bacon,Cheese

Q5. 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



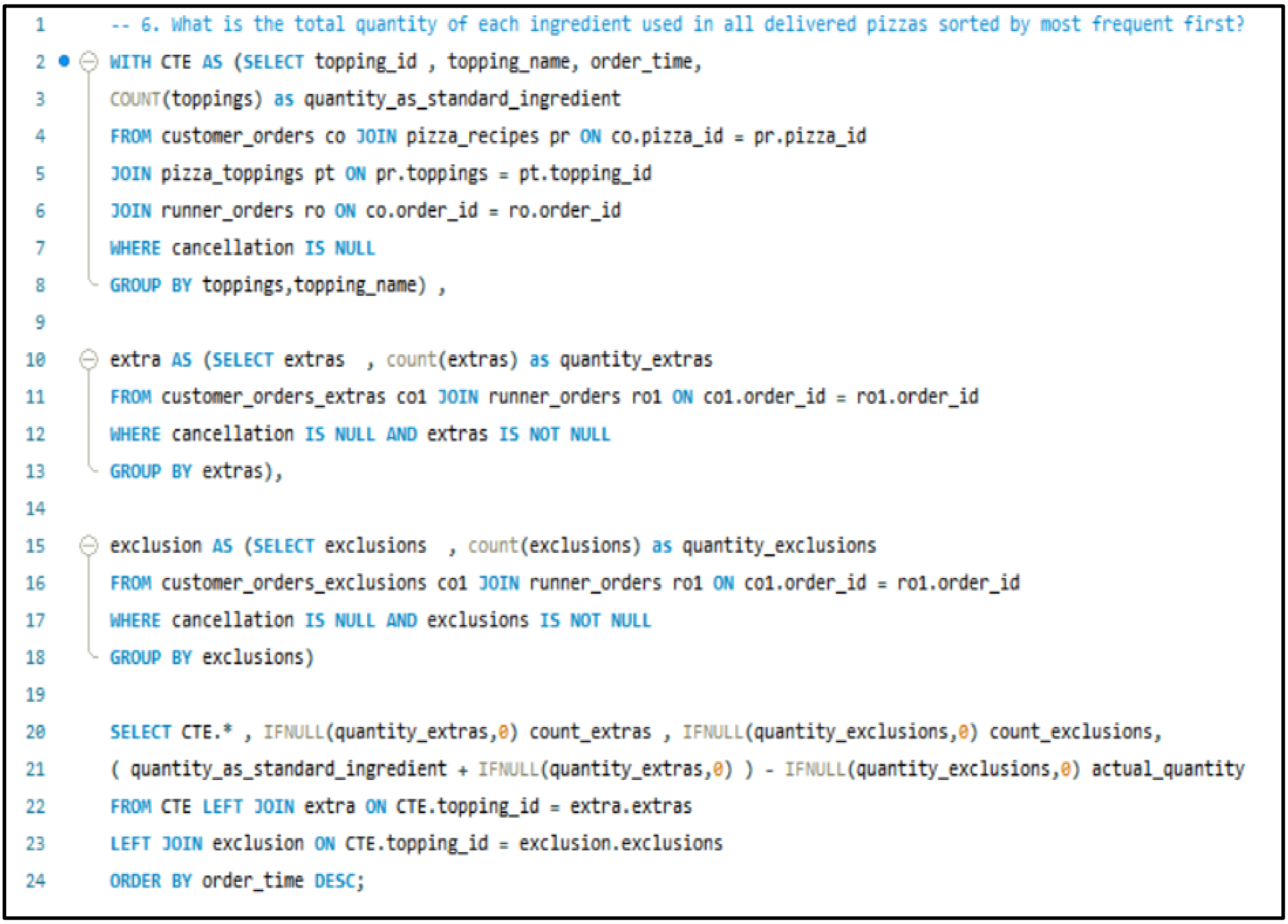
```sql
WITH relevant_ingredients AS (SELECT pizza_recipes.* , pizza_name ,topping_name , topping_id
                              FROM pizza_recipes JOIN pizza_names ON pizza_recipes.pizza_id = pizza_names.pizza_id
                              JOIN pizza_toppings ON pizza_toppings.topping_id = pizza_recipes.toppings)
SELECT order_id , customer_id , pizza_id , extras , exclusions ,
       CONCAT(pizza_name , ' : ', IFNULL(group_concat(DISTINCT extra_topping_name) , '') ,' 2X ' ,
              group_concat(DISTINCT topping_name ORDER BY topping_name)) ingredient_list
FROM (
SELECT customer_orders.* , pizza_name,
relevant_ingredients.topping_id , relevant_ingredients.topping_name , pizza_toppings.topping_name AS extra_topping_name
FROM customer_orders  JOIN relevant_ingredients
ON customer_orders.pizza_id = relevant_ingredients.pizza_id
LEFT JOIN pizza_toppings ON LEFT(customer_orders.extras,1) = pizza_toppings.topping_id
OR RIGHT(customer_orders.extras,1) = pizza_toppings.topping_id
WHERE exclusions <> relevant_ingredients.topping_id OR exclusions IS NULL) Temp
GROUP BY order_id , pizza_id, exclusions;
```

order_id	customer_id	pizza_id	extras	exclusions	ingredient_list
1	101	1	NULL	NULL	Meatlovers : 2X Bacon,BBQ Sauce,Beef,Cheese,Chicken,Mushrooms,Pepperoni,Salami
2	101	1	NULL	NULL	Meatlovers : 2X Bacon,BBQ Sauce,Beef,Cheese,Chicken,Mushrooms,Pepperoni,Salami
3	102	1	NULL	NULL	Meatlovers : 2X Bacon,BBQ Sauce,Beef,Cheese,Chicken,Mushrooms,Pepperoni,Salami
3	102	2	NULL	NULL	Vegetarian : 2X Cheese,Mushrooms,Onions,Peppers,Tomato Sauce,Tomatoes
4	103	1	NULL	4	Meatlovers : 2X Bacon,BBQ Sauce,Beef,Chicken,Mushrooms,Pepperoni,Salami
4	103	2	NULL	4	Vegetarian : 2X Mushrooms,Onions,Peppers,Tomato Sauce,Tomatoes
5	104	1	1	NULL	Meatlovers : Bacon 2X Bacon,BBQ Sauce,Beef,Cheese,Chicken,Mushrooms,Pepperoni,Salami
6	101	2	NULL	NULL	Vegetarian : 2X Cheese,Mushrooms,Onions,Peppers,Tomato Sauce,Tomatoes
7	105	2	1	NULL	Vegetarian : Bacon 2X Cheese,Mushrooms,Onions,Peppers,Tomato Sauce,Tomatoes
8	102	1	NULL	NULL	Meatlovers : 2X Bacon,BBQ Sauce,Beef,Cheese,Chicken,Mushrooms,Pepperoni,Salami
9	103	1	1, 5	4	Meatlovers : Bacon,Chicken 2X Bacon,BBQ Sauce,Beef,Chicken,Mushrooms,Pepperoni,Salami
10	104	1	NULL	NULL	Meatlovers : 2X Bacon,BBQ Sauce,Beef,Cheese,Chicken,Mushrooms,Pepperoni,Salami
10	104	1	1, 4	2, 6	Meatlovers : Bacon,Cheese 2X Bacon,Beef,Cheese,Chicken,Mushrooms,Pepperoni,Salami

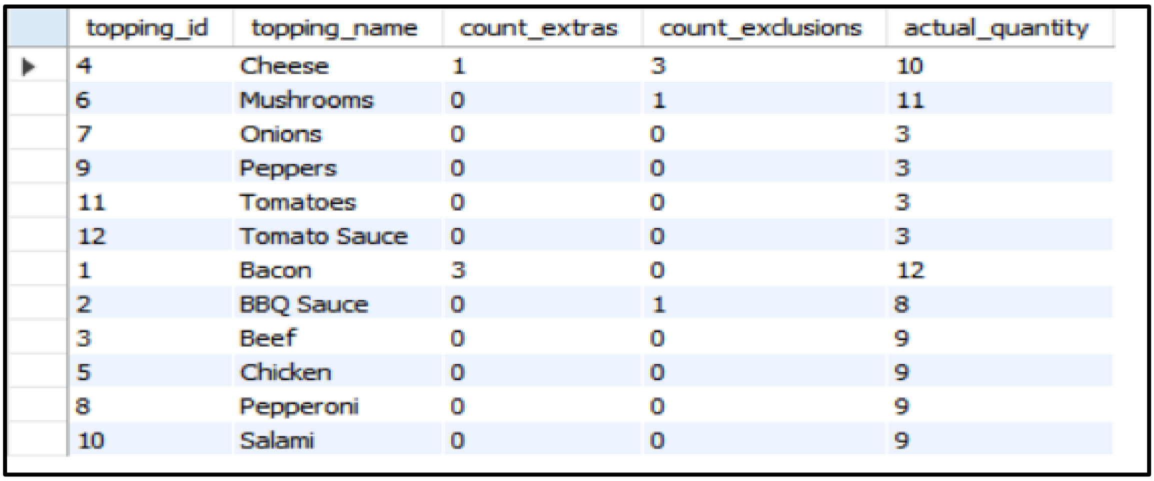
Here the output is like->

pizza name: extras (if any) 2X relevant ingredients (excluding exclusions if any)

Q6. What is the total quantity of each ingredient used in all delivered pizzas sorted by most frequent first?



```sql
-- 6. What is the total quantity of each ingredient used in all delivered pizzas sorted by most frequent first?
WITH CTE AS (SELECT topping_id , topping_name, order_time,
COUNT(toppings) as quantity_as_standard_ingredient
FROM customer_orders co JOIN pizza_recipes pr ON co.pizza_id = pr.pizza_id
JOIN pizza_toppings pt ON pr.toppings = pt.topping_id
JOIN runner_orders ro ON co.order_id = ro.order_id
WHERE cancellation IS NULL
GROUP BY toppings,topping_name) ,

extra AS (SELECT extras  , count(extras) as quantity_extras
FROM customer_orders_extras co1 JOIN runner_orders ro1 ON co1.order_id = ro1.order_id
WHERE cancellation IS NULL AND extras IS NOT NULL
GROUP BY extras),

exclusion AS (SELECT exclusions  , count(exclusions) as quantity_exclusions
FROM customer_orders_exclusions co1 JOIN runner_orders ro1 ON co1.order_id = ro1.order_id
WHERE cancellation IS NULL AND exclusions IS NOT NULL
GROUP BY exclusions)

SELECT CTE.* , IFNULL(quantity_extras,0) count_extras , IFNULL(quantity_exclusions,0) count_exclusions,
( quantity_as_standard_ingredient + IFNULL(quantity_extras,0) ) - IFNULL(quantity_exclusions,0) actual_quantity
FROM CTE LEFT JOIN extra ON CTE.topping_id = extra.extras
LEFT JOIN exclusion ON CTE.topping_id = exclusion.exclusions
ORDER BY order_time DESC;
```



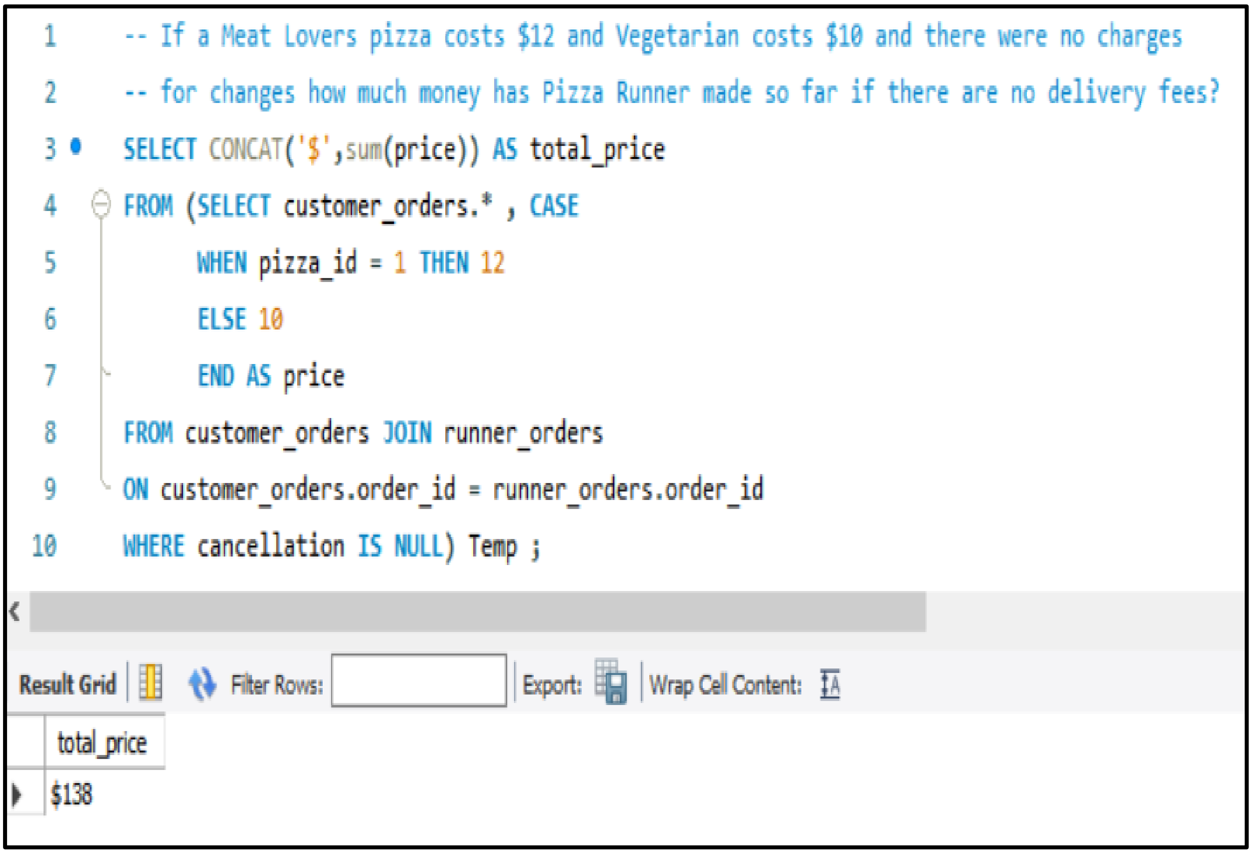
topping_id	topping_name	count_extras	count_exclusions	actual_quantity
4	Cheese	1	3	10
6	Mushrooms	0	1	11
7	Onions	0	0	3
9	Peppers	0	0	3
11	Tomatoes	0	0	3
12	Tomato Sauce	0	0	3
1	Bacon	3	0	12
2	BBQ Sauce	0	1	8
3	Beef	0	0	9
5	Chicken	0	0	9
8	Pepperoni	0	0	9
10	Salami	0	0	9

Total quantity of each ingredient used is calculated as:

(Total no. of times that ingredient was a standard ingredient for all sold pizzas + no. of times it was added as an extra ingredient in all sold pizzas) – no. of times it was excluded from standard ingredients for all sold pizzas.

D. Pricing and Ratings

Q1. 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

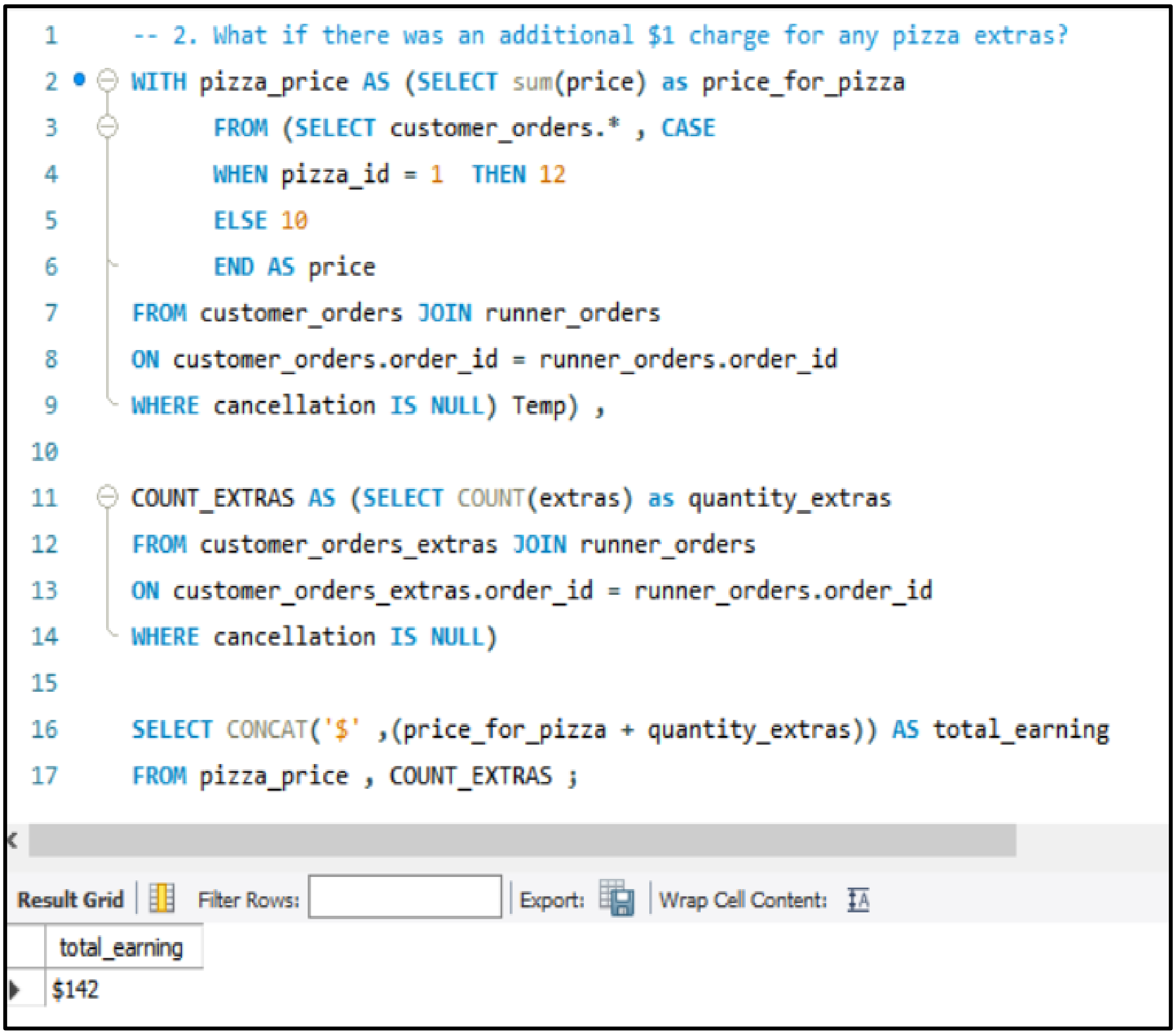


```sql
-- If a Meat Lovers pizza costs $12 and Vegetarian costs $10 and there were no charges
-- for changes how much money has Pizza Runner made so far if there are no delivery fees?
SELECT CONCAT('$',sum(price)) AS total_price
FROM (SELECT customer_orders.* , CASE
        WHEN pizza_id = 1 THEN 12
        ELSE 10
        END AS price
FROM customer_orders JOIN runner_orders
ON customer_orders.order_id = runner_orders.order_id
WHERE cancellation IS NULL) Temp ;
```

```
total_price
$138
```

Q2. What if there was an additional $1 charge for any pizza extras?

Like Q1 calculated the base prize for pizza through temporary table pizza price. Next created another temp table COUNT_EXTRAS to count the extras ordered for successful deliveries and added that to the base price because each extra cost $1.

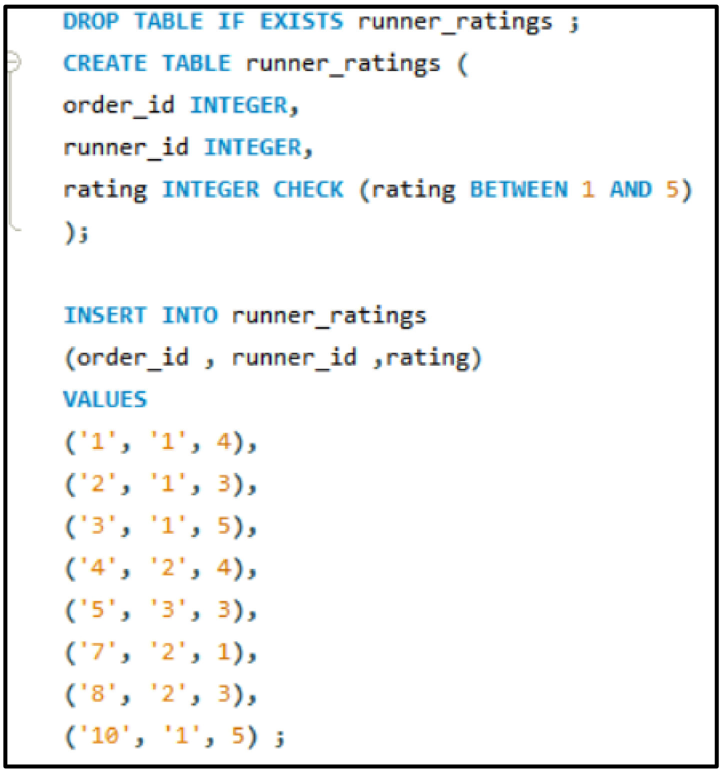


```
-- 2. What if there was an additional $1 charge for any pizza extras?
WITH pizza_price AS (SELECT sum(price) as price_for_pizza
        FROM (SELECT customer_orders.* , CASE
        WHEN pizza_id = 1  THEN 12
        ELSE 10
        END AS price
FROM customer_orders JOIN runner_orders
ON customer_orders.order_id = runner_orders.order_id
WHERE cancellation IS NULL) Temp) ,

COUNT_EXTRAS AS (SELECT COUNT(extras) as quantity_extras
FROM customer_orders_extras JOIN runner_orders
ON customer_orders_extras.order_id = runner_orders.order_id
WHERE cancellation IS NULL)

SELECT CONCAT('$' ,(price_for_pizza + quantity_extras)) AS total_earning
FROM pizza_price , COUNT_EXTRAS ;
```

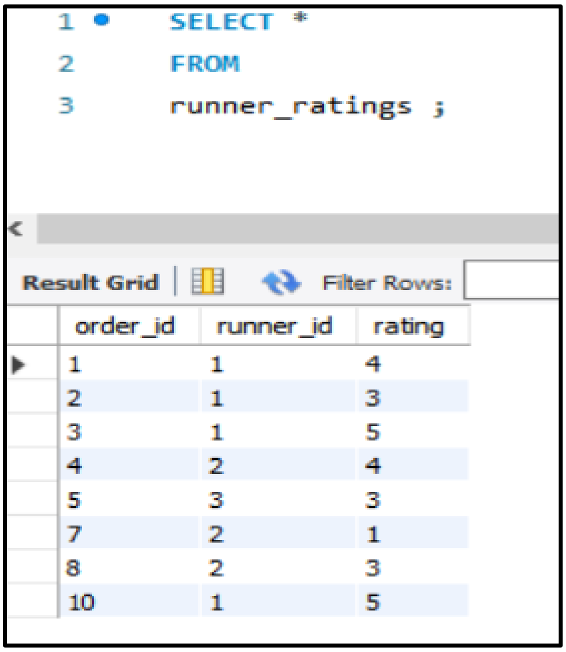
```
total_earning
$142
```

Q3. 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.

NOTE: While generating this table, order id 6 and 9 can't be included because these orders were cancelled so there can be no runner rating for it.



```sql
DROP TABLE IF EXISTS runner_ratings ;
CREATE TABLE runner_ratings (
order_id INTEGER,
runner_id INTEGER,
rating INTEGER CHECK (rating BETWEEN 1 AND 5)
);

INSERT INTO runner_ratings
(order_id , runner_id ,rating)
VALUES
('1', '1', 4),
('2', '1', 3),
('3', '1', 5),
('4', '2', 4),
('5', '3', 3),
('7', '2', 1),
('8', '2', 3),
('10', '1', 5) ;
```

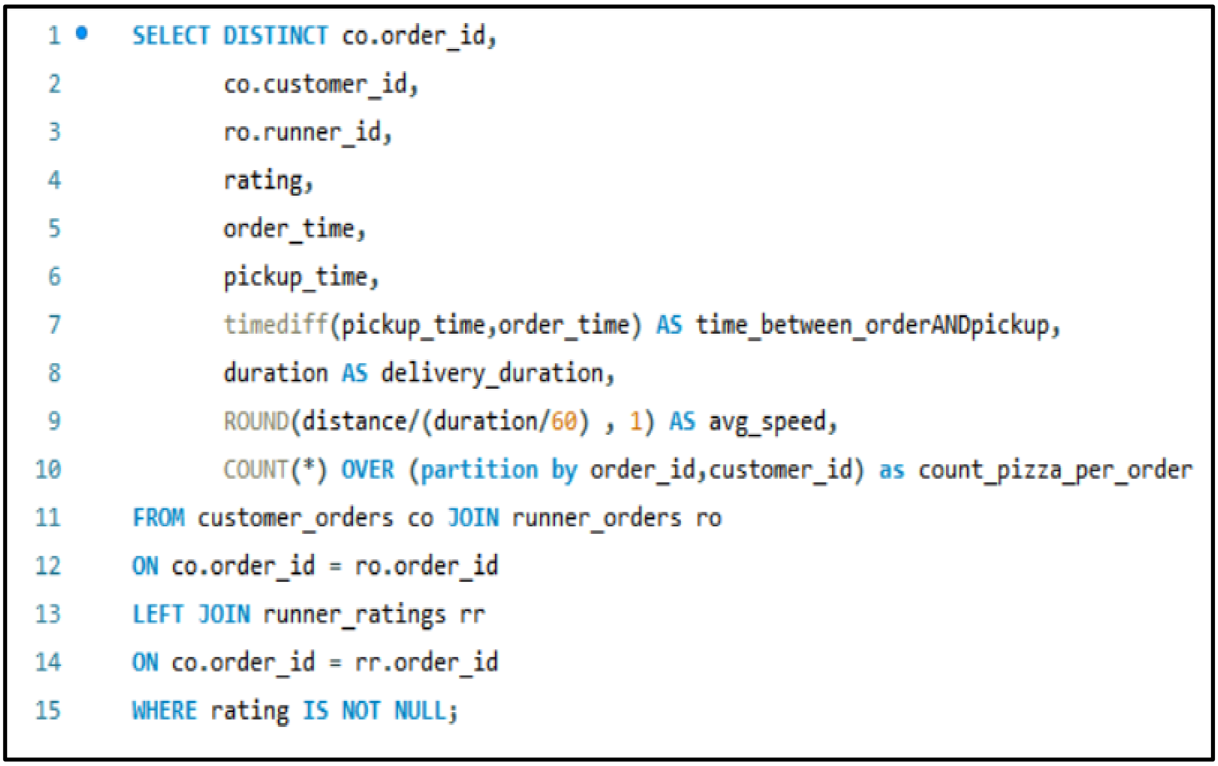


```sql
SELECT *
FROM
runner_ratings ;
```

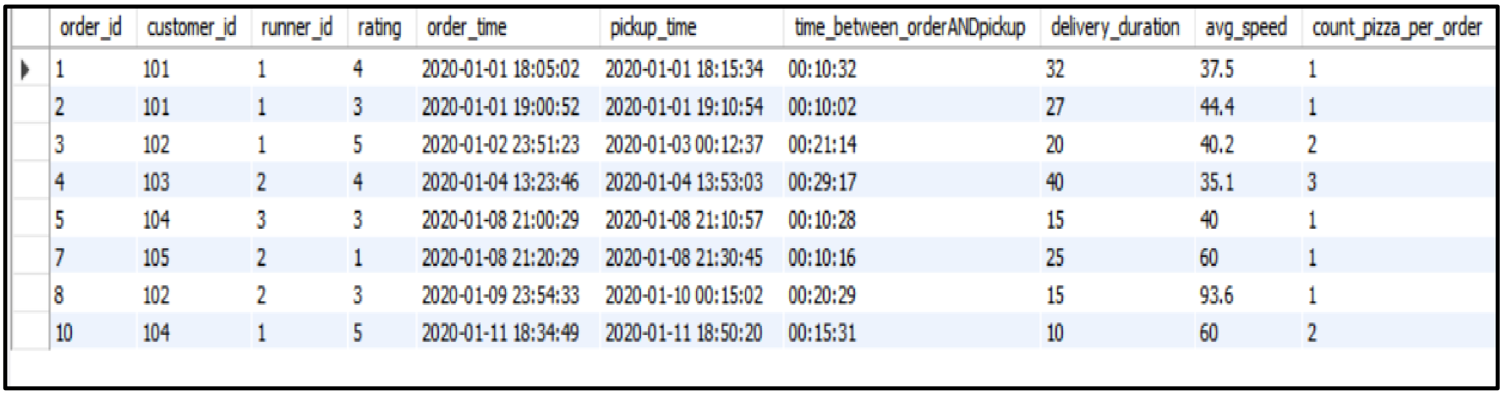
order_id	runner_id	rating
1	1	4
2	1	3
3	1	5
4	2	4
5	3	3
7	2	1
8	2	3
10	1	5

Q4. 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

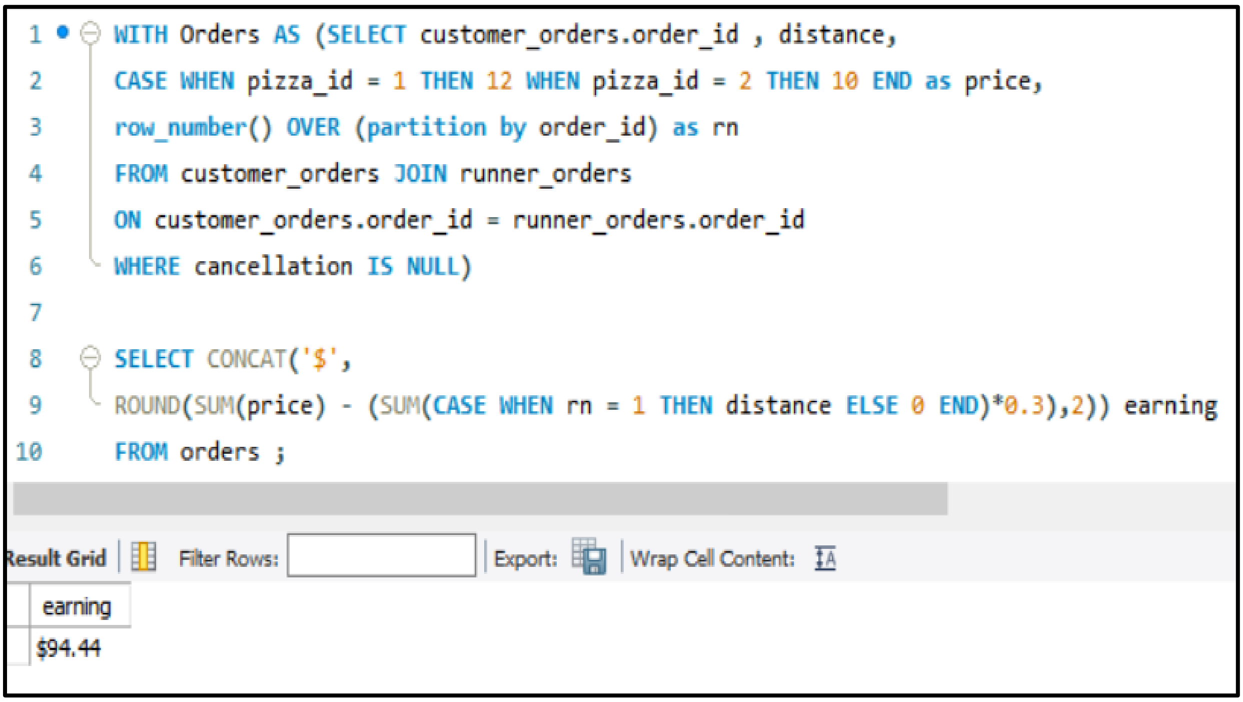


```
SELECT DISTINCT co.order_id,
        co.customer_id,
        ro.runner_id,
        rating,
        order_time,
        pickup_time,
        timediff(pickup_time,order_time) AS time_between_orderANDpickup,
        duration AS delivery_duration,
        ROUND(distance/(duration/60) , 1) AS avg_speed,
        COUNT(*) OVER (partition by order_id,customer_id) as count_pizza_per_order
FROM customer_orders co JOIN runner_orders ro
ON co.order_id = ro.order_id
LEFT JOIN runner_ratings rr
ON co.order_id = rr.order_id
WHERE rating IS NOT NULL;
```



order_id	customer_id	runner_id	rating	order_time	pickup_time	time_between_orderANDpickup	delivery_duration	avg_speed	count_pizza_per_order
1	101	1	4	2020-01-01 18:05:02	2020-01-01 18:15:34	00:10:32	32	37.5	1
2	101	1	3	2020-01-01 19:00:52	2020-01-01 19:10:54	00:10:02	27	44.4	1
3	102	1	5	2020-01-02 23:51:23	2020-01-03 00:12:37	00:21:14	20	40.2	2
4	103	2	4	2020-01-04 13:23:46	2020-01-04 13:53:03	00:29:17	40	35.1	3
5	104	3	3	2020-01-08 21:00:29	2020-01-08 21:10:57	00:10:28	15	40	1
7	105	2	1	2020-01-08 21:20:29	2020-01-08 21:30:45	00:10:16	25	60	1
8	102	2	3	2020-01-09 23:54:33	2020-01-10 00:15:02	00:20:29	15	93.6	1
10	104	1	5	2020-01-11 18:34:49	2020-01-11 18:50:20	00:15:31	10	60	2

Q5. 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 travelled, how much money does Pizza Runner have left over after these deliveries?



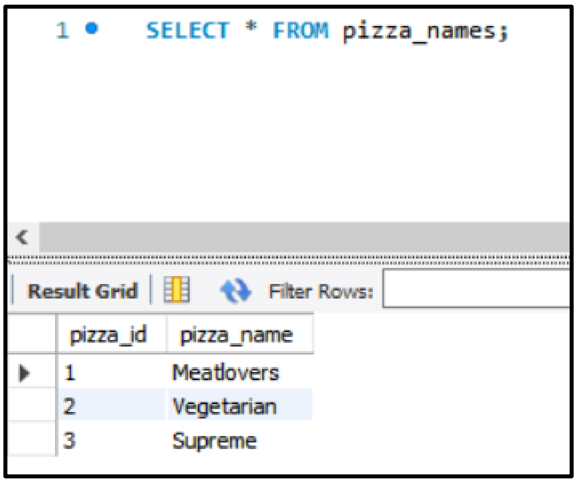
```
WITH Orders AS (SELECT customer_orders.order_id , distance,
CASE WHEN pizza_id = 1 THEN 12 WHEN pizza_id = 2 THEN 10 END as price,
row_number() OVER (partition by order_id) as rn
FROM customer_orders JOIN runner_orders
ON customer_orders.order_id = runner_orders.order_id
WHERE cancellation IS NULL)

SELECT CONCAT('$',
ROUND(SUM(price) - (SUM(CASE WHEN rn = 1 THEN distance ELSE 0 END)*0.3),2)) earning
FROM orders ;
```

earning
$94.44

E. Bonus Questions

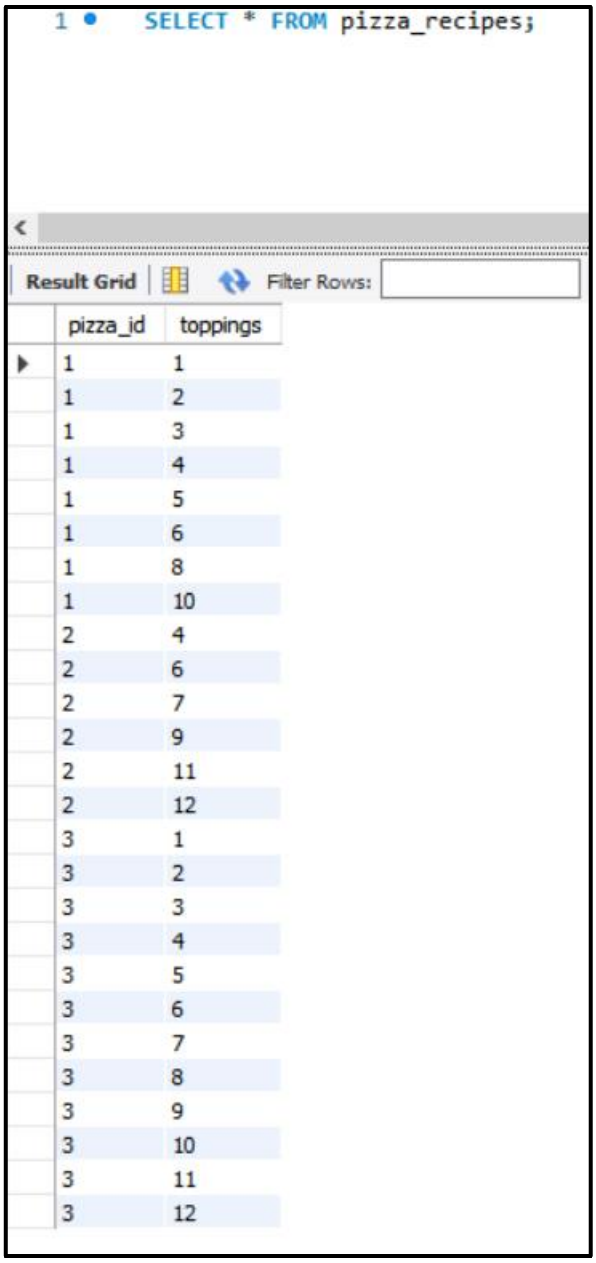
Q1. If Danny wants to expand his range of pizzas - how would this impact the existing data design? Write an `INSERT` statement to demonstrate what would happen if a new `Supreme` pizza with all the toppings was added to the Pizza Runner menu?

➔ If Supreme pizza is to be added in the database, we need to use Insert statement for table pizza names and pizza recipes both. And after inserting the values, this is how the two tables looks like:



```
SELECT * FROM pizza_names;
```

pizza_id	pizza_name
1	Meatlovers
2	Vegetarian
3	Supreme



```
SELECT * FROM pizza_recipes;
```

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