DATAWITHDANNY.COM

START YOUR SQL ENGINES

PIZZA RUNNER

CASE STUDY #2

8 WEEK SQL CHALLENGE

8WEEKSQLCHALLENGE.COM

NANDINI AGARWAL

INTRODUCTION

Danny was scrolling through his Instagram feed when something really caught his eye - “80s Retro Styling and Pizza Is The Future!”

Danny was sold on the idea, but he knew that pizza alone was not going to help him get seed funding to expand his new Pizza Empire - so he had one more genius idea to combine with it - he was going to Uberize it - and so Pizza Runner was launched!

Danny started by recruiting “runners” to deliver fresh pizza from Pizza Runner Headquarters (otherwise known as Danny’s house) and also maxed out his credit card to pay freelance developers to build a mobile app to accept orders from customers.

CHALLENGE

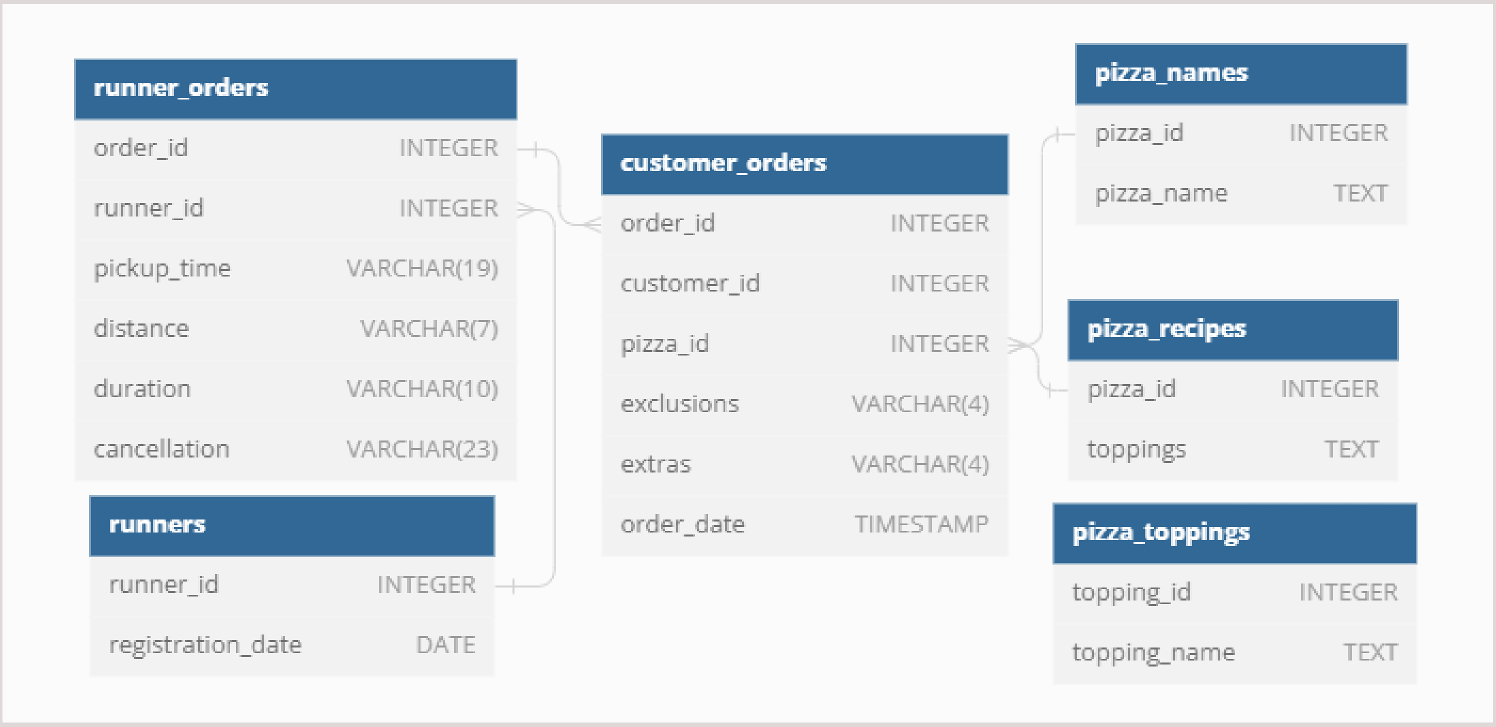
This case study has LOTS of questions - they are broken up by area of focus including:

A. Pizza Metrics
B. Runner and Customer Experience
C. Ingredient Optimisation
D. Pricing and Ratings

Tables:

- runner_orders
- runner
- customer_orders
- pizza_names
- pizza_recipes
- pizza_toppings

Entity Relationship Diagram:



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

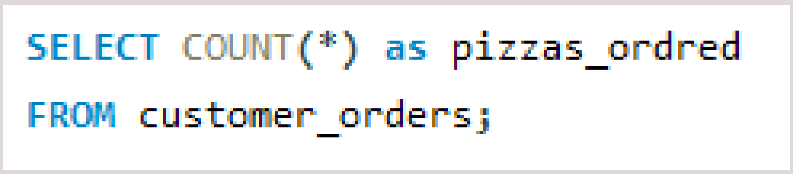
pizza_id	INTEGER
toppings	TEXT

pizza_toppings

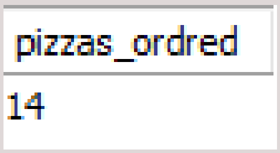
topping_id	INTEGER
topping_name	TEXT

A. PIZZA METRICS

1. How many pizzas were ordered?

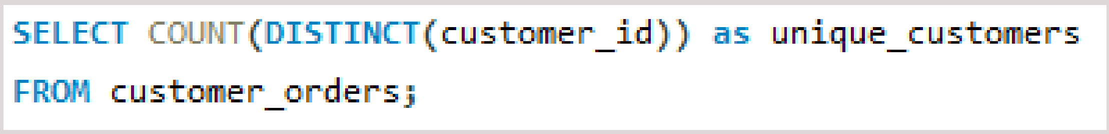


```
SELECT COUNT(*) as pizzas_ordred
FROM customer_orders;
```

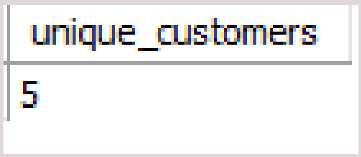


pizzas_ordred
14

2.How many unique customer orders were made?

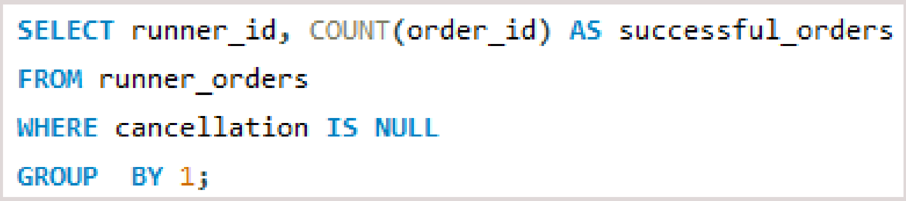


```
SELECT COUNT(DISTINCT(customer_id)) as unique_customers
FROM customer_orders;
```

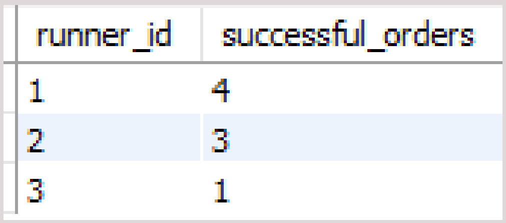


unique_customers
5

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

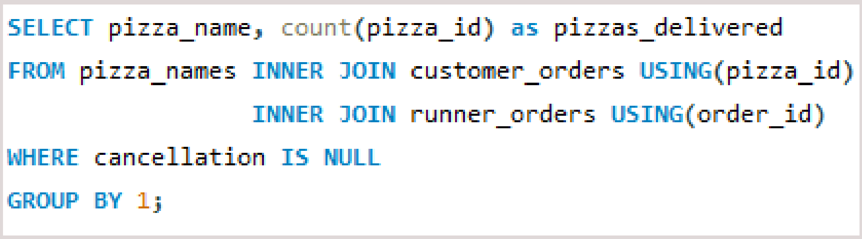


```
SELECT runner_id, COUNT(order_id) AS successful_orders
FROM runner_orders
WHERE cancellation IS NULL
GROUP  BY 1;
```

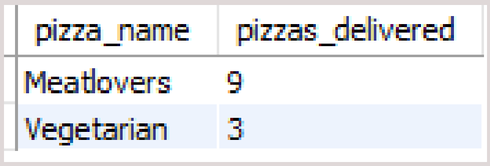


runner_id	successful_orders
1	4
2	3
3	1

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

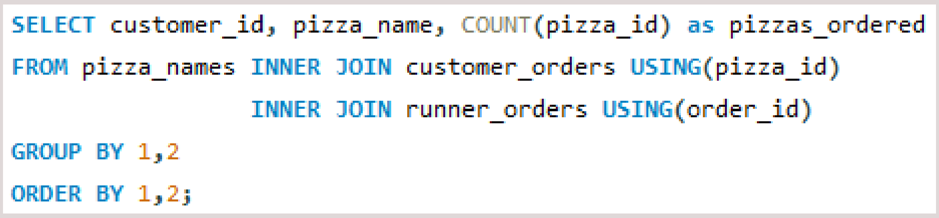


```sql
SELECT pizza_name, count(pizza_id) as pizzas_delivered
FROM pizza_names INNER JOIN customer_orders USING(pizza_id)
                 INNER JOIN runner_orders USING(order_id)
WHERE cancellation IS NULL
GROUP BY 1;
```

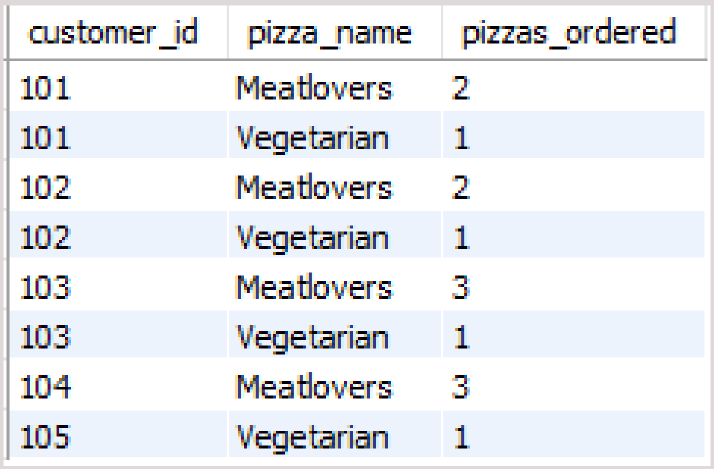


pizza_name	pizzas_delivered
Meatlovers	9
Vegetarian	3

5.How many Vegetarian and Meatlovers were ordered by each customer?

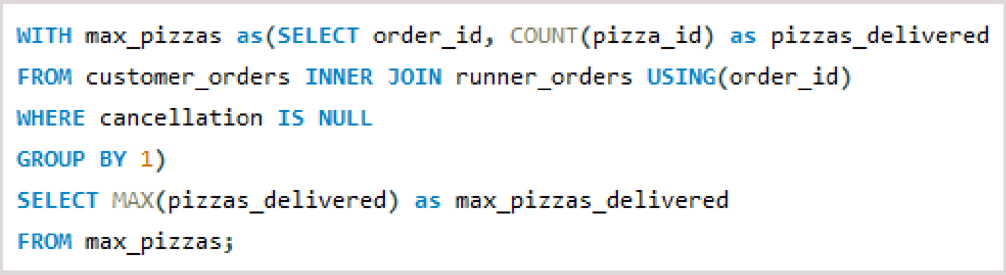


```sql
SELECT customer_id, pizza_name, COUNT(pizza_id) as pizzas_ordered
FROM pizza_names INNER JOIN customer_orders USING(pizza_id)
                 INNER JOIN runner_orders USING(order_id)
GROUP BY 1,2
ORDER BY 1,2;
```

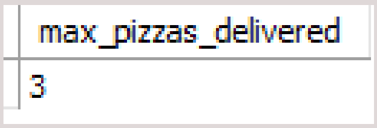


customer_id	pizza_name	pizzas_ordered
101	Meatlovers	2
101	Vegetarian	1
102	Meatlovers	2
102	Vegetarian	1
103	Meatlovers	3
103	Vegetarian	1
104	Meatlovers	3
105	Vegetarian	1

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

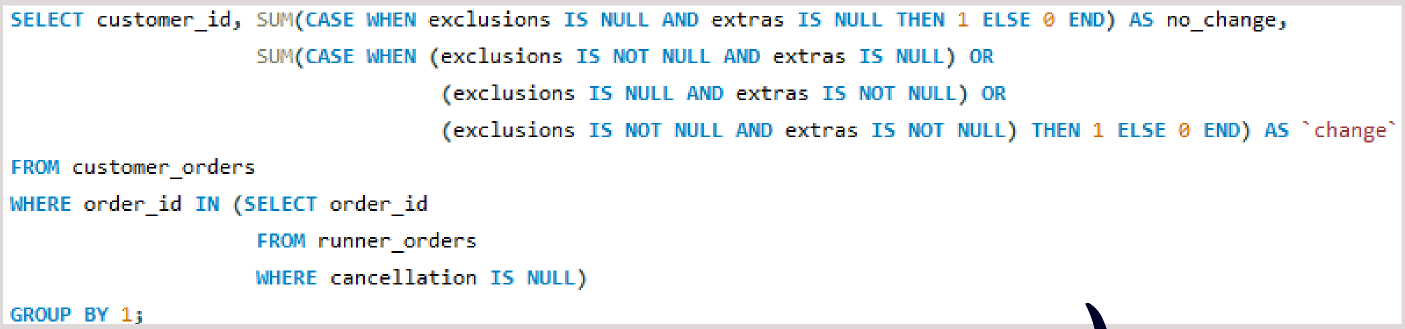


```sql
WITH max_pizzas as(SELECT order_id, COUNT(pizza_id) as pizzas_delivered
FROM customer_orders INNER JOIN runner_orders USING(order_id)
WHERE cancellation IS NULL
GROUP BY 1)
SELECT MAX(pizzas_delivered) as max_pizzas_delivered
FROM max_pizzas;
```

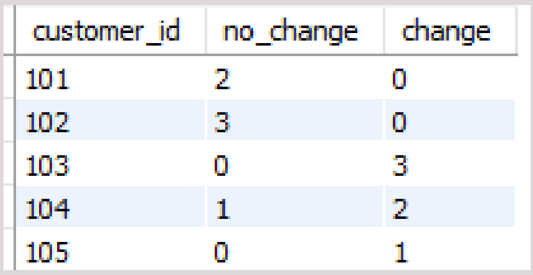


max_pizzas_delivered
3

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

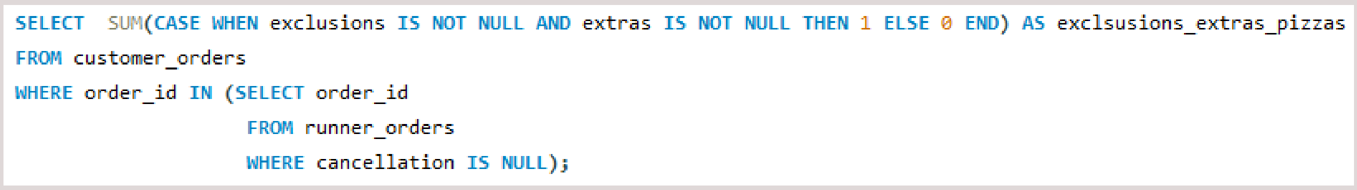


```sql
SELECT customer_id, SUM(CASE WHEN exclusions IS NULL AND extras IS NULL THEN 1 ELSE 0 END) AS no_change,
                    SUM(CASE WHEN (exclusions IS NOT NULL AND extras IS NULL) OR
                                  (exclusions IS NULL AND extras IS NOT NULL) OR
                                  (exclusions IS NOT NULL AND extras IS NOT NULL) THEN 1 ELSE 0 END) AS `change`
FROM customer_orders
WHERE order_id IN (SELECT order_id
                   FROM runner_orders
                   WHERE cancellation IS NULL)
GROUP BY 1;
```

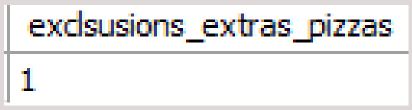


customer_id	no_change	change
101	2	0
102	3	0
103	0	3
104	1	2
105	0	1

8.**How many pizzas were delivered that had both exclusions and extras?**

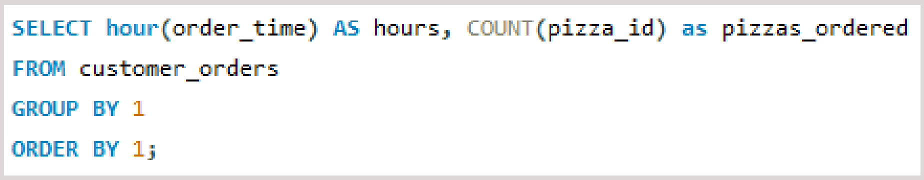


```
SELECT  SUM(CASE WHEN exclusions IS NOT NULL AND extras IS NOT NULL THEN 1 ELSE 0 END) AS exclsusions_extras_pizzas
FROM customer_orders
WHERE order_id IN (SELECT order_id
                   FROM runner_orders
                   WHERE cancellation IS NULL);
```

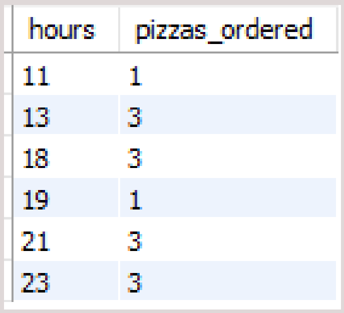


exclsusions_extras_pizzas
1

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

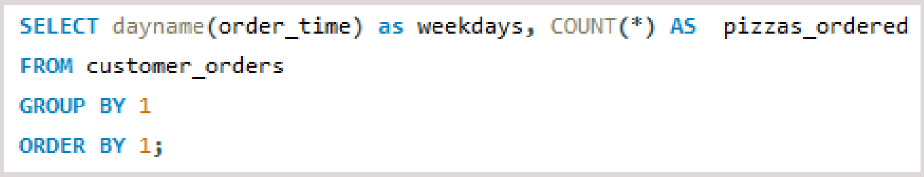


```
SELECT hour(order_time) AS hours, COUNT(pizza_id) as pizzas_ordered
FROM customer_orders
GROUP BY 1
ORDER BY 1;
```

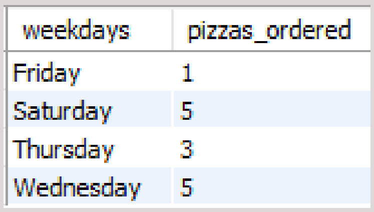


hours	pizzas_ordered
11	1
13	3
18	3
19	1
21	3
23	3

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



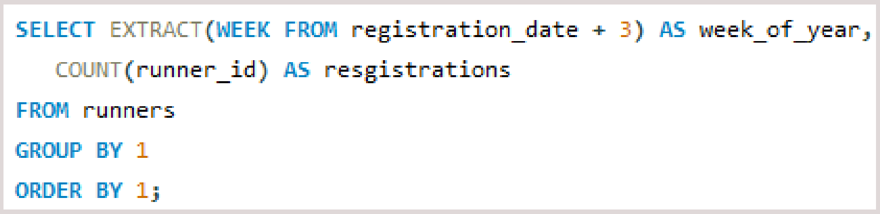
```
SELECT dayname(order_time) as weekdays, COUNT(*) AS  pizzas_ordered
FROM customer_orders
GROUP BY 1
ORDER BY 1;
```



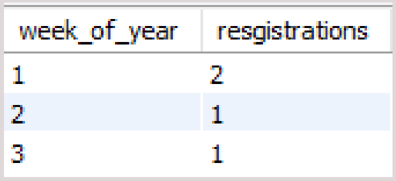
weekdays	pizzas_ordered
Friday	1
Saturday	5
Thursday	3
Wednesday	5

B. RUNNER AND CUSTOMER EXPERIENCE

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

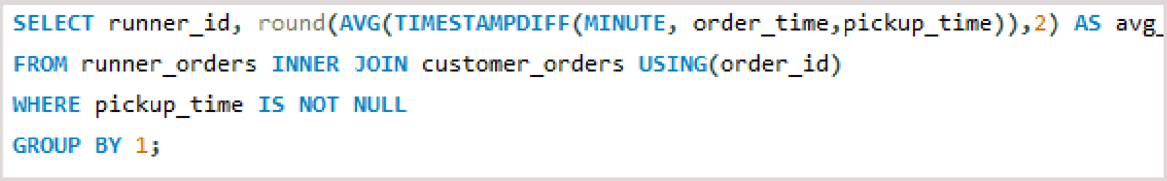


```sql
SELECT EXTRACT(WEEK FROM registration_date + 3) AS week_of_year,
   COUNT(runner_id) AS resgistrations
FROM runners
GROUP BY 1
ORDER BY 1;
```

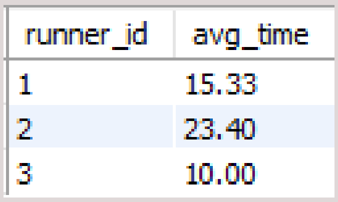


week_of_year	resgistrations
1	2
2	1
3	1

2. **What was the average time in minutes it took for each runner to arrive at the Pizza Runner HQ to pickup the order?**

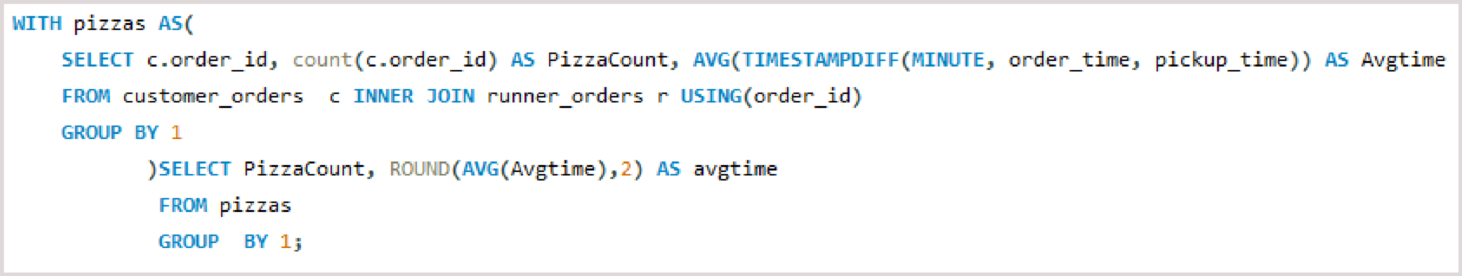


```sql
SELECT runner_id, round(AVG(TIMESTAMPDIFF(MINUTE, order_time,pickup_time)),2) AS avg_
FROM runner_orders INNER JOIN customer_orders USING(order_id)
WHERE pickup_time IS NOT NULL
GROUP BY 1;
```

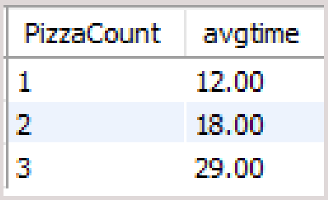


runner_id	avg_time
1	15.33
2	23.40
3	10.00

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

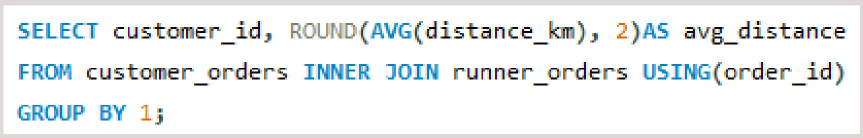


```sql
WITH pizzas AS(
    SELECT c.order_id, count(c.order_id) AS PizzaCount, AVG(TIMESTAMPDIFF(MINUTE, order_time, pickup_time)) AS Avgtime
    FROM customer_orders  c INNER JOIN runner_orders r USING(order_id)
    GROUP BY 1
          )SELECT PizzaCount, ROUND(AVG(Avgtime),2) AS avgtime
           FROM pizzas
           GROUP  BY 1;
```

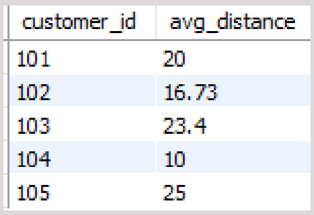


PizzaCount	avgtime
1	12.00
2	18.00
3	29.00

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

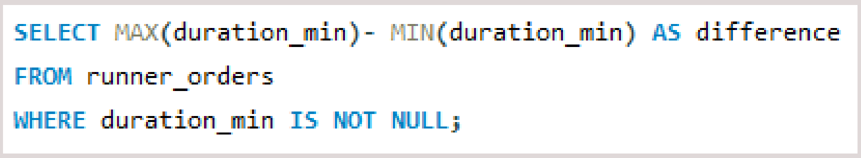


```sql
SELECT customer_id, ROUND(AVG(distance_km), 2)AS avg_distance
FROM customer_orders INNER JOIN runner_orders USING(order_id)
GROUP BY 1;
```

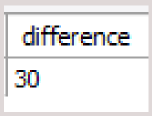


customer_id	avg_distance
101	20
102	16.73
103	23.4
104	10
105	25

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

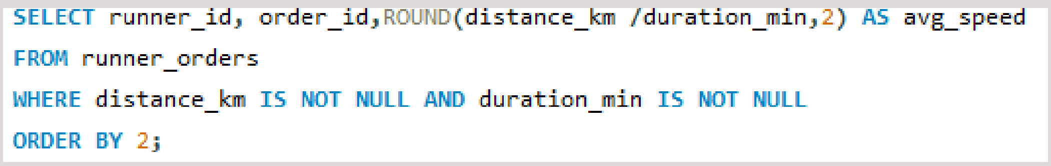


```sql
SELECT MAX(duration_min)- MIN(duration_min) AS difference
FROM runner_orders
WHERE duration_min IS NOT NULL;
```

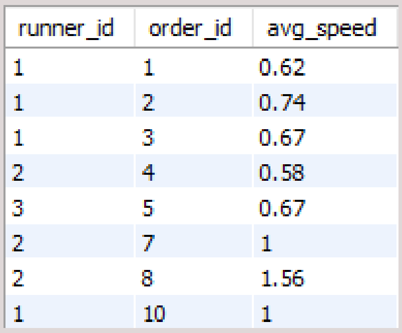


difference
30

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

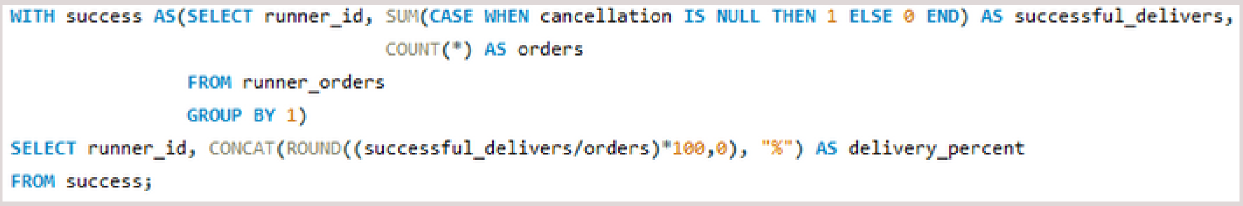


```sql
SELECT runner_id, order_id,ROUND(distance_km /duration_min,2) AS avg_speed
FROM runner_orders
WHERE distance_km IS NOT NULL AND duration_min IS NOT NULL
ORDER BY 2;
```

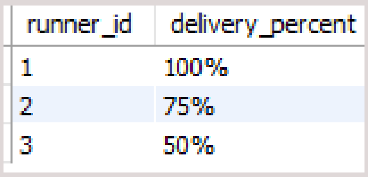


runner_id	order_id	avg_speed
1	1	0.62
1	2	0.74
1	3	0.67
2	4	0.58
3	5	0.67
2	7	1
2	8	1.56
1	10	1

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



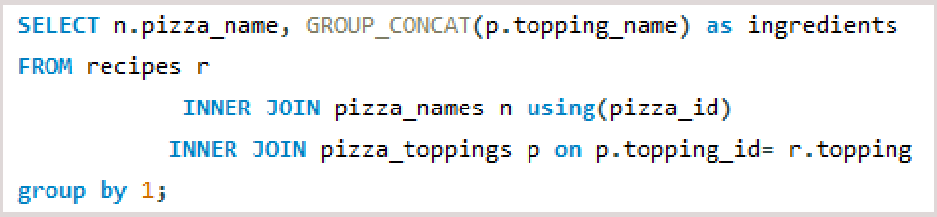
```
WITH success AS(SELECT runner_id, SUM(CASE WHEN cancellation IS NULL THEN 1 ELSE 0 END) AS successful_delivers,
                       COUNT(*) AS orders
                FROM runner_orders
                GROUP BY 1)
SELECT runner_id, CONCAT(ROUND((successful_delivers/orders)*100,0), "%") AS delivery_percent
FROM success;
```



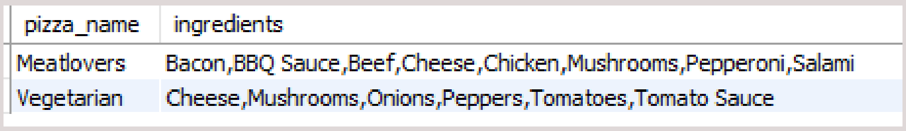
runner_id	delivery_percent
1	100%
2	75%
3	50%

C. Ingredient Optimisation

1. What are the standard ingredients for each pizza?

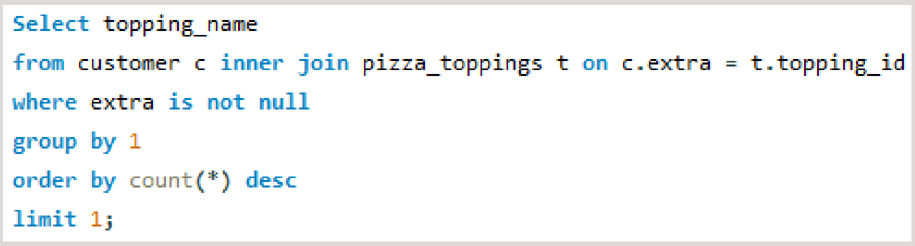


```
SELECT n.pizza_name, GROUP_CONCAT(p.topping_name) as ingredients
FROM recipes r
            INNER JOIN pizza_names n using(pizza_id)
           INNER JOIN pizza_toppings p on p.topping_id= r.topping
group by 1;
```

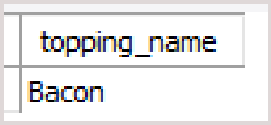


pizza_name	ingredients
Meatlovers	Bacon,BBQ Sauce,Beef,Cheese,Chicken,Mushrooms,Pepperoni,Salami
Vegetarian	Cheese,Mushrooms,Onions,Peppers,Tomatoes,Tomato Sauce

2.What was the most commonly added extra?

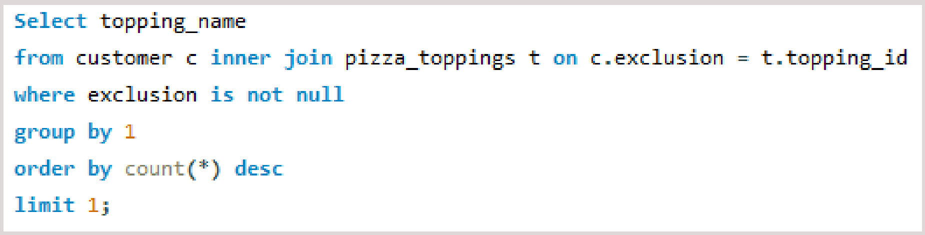


```
Select topping_name
from customer c inner join pizza_toppings t on c.extra = t.topping_id
where extra is not null
group by 1
order by count(*) desc
limit 1;
```

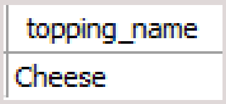


topping_name
Bacon

3.What was the most common exclusion?



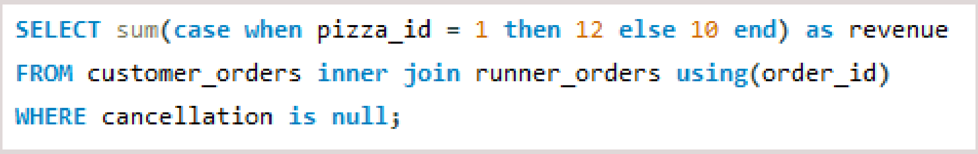
```
Select topping_name
from customer c inner join pizza_toppings t on c.exclusion = t.topping_id
where exclusion is not null
group by 1
order by count(*) desc
limit 1;
```



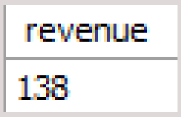
topping_name
Cheese

D. 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?

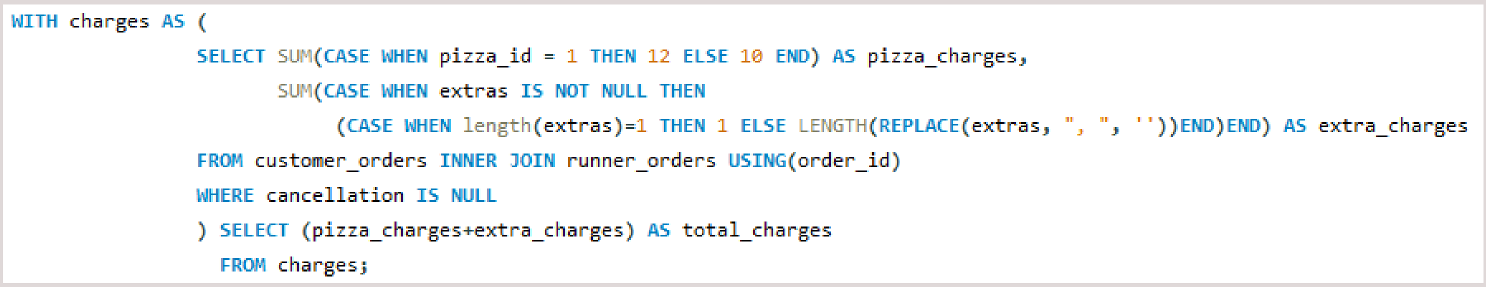


```sql
SELECT sum(case when pizza_id = 1 then 12 else 10 end) as revenue
FROM customer_orders inner join runner_orders using(order_id)
WHERE cancellation is null;
```

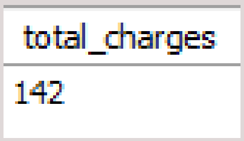


revenue
138

2.What if there was an additional $1 charge for any pizza extras? Add cheese is $1 extra

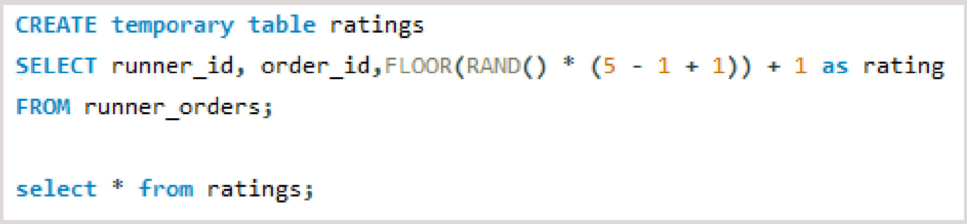


```sql
WITH charges AS (
                SELECT SUM(CASE WHEN pizza_id = 1 THEN 12 ELSE 10 END) AS pizza_charges,
                       SUM(CASE WHEN extras IS NOT NULL THEN
                            (CASE WHEN length(extras)=1 THEN 1 ELSE LENGTH(REPLACE(extras, ", ", ''))END)END) AS extra_charges
                FROM customer_orders INNER JOIN runner_orders USING(order_id)
                WHERE cancellation IS NULL
                ) SELECT (pizza_charges+extra_charges) AS total_charges
                  FROM charges;
```



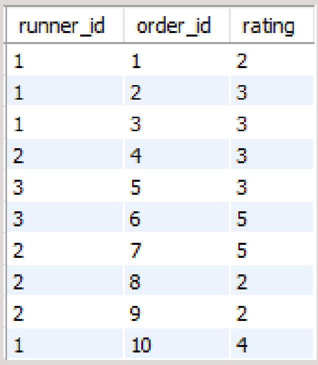
total_charges
142

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.



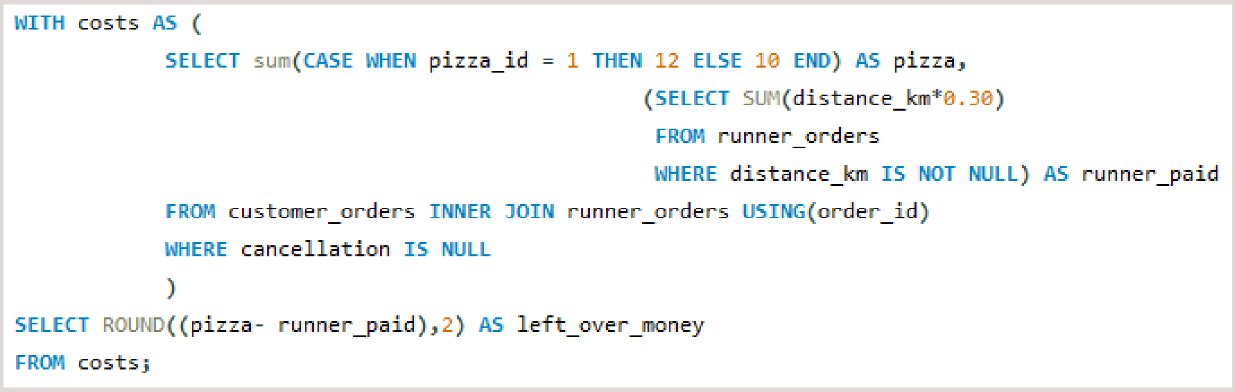
```sql
CREATE temporary table ratings
SELECT runner_id, order_id,FLOOR(RAND() * (5 - 1 + 1)) + 1 as rating
FROM runner_orders;

select * from ratings;
```

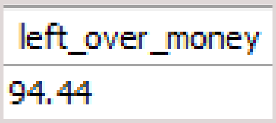


runner_id	order_id	rating
1	1	2
1	2	3
1	3	3
2	4	3
3	5	3
3	6	5
2	7	5
2	8	2
2	9	2
1	10	4

4.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?



```sql
WITH costs AS (
            SELECT sum(CASE WHEN pizza_id = 1 THEN 12 ELSE 10 END) AS pizza,
                                                (SELECT SUM(distance_km*0.30)
                                                 FROM runner_orders
                                                 WHERE distance_km IS NOT NULL) AS runner_paid
            FROM customer_orders INNER JOIN runner_orders USING(order_id)
            WHERE cancellation IS NULL
            )
SELECT ROUND((pizza- runner_paid),2) AS left_over_money
FROM costs;
```



left_over_money
94.44

Thank You