DATAWITHDANNY.COM

START YOUR SQL ENGINES

PIZZA RUNNER

CASE STUDY #2

8 WEEK SQL CHALLENGE

8WEEKSQLCHALLENGE.COM

https://8weeksqlchallenge.com/case-study-2/

Introduction

Did you know that over **115 million kilograms** of pizza is consumed daily worldwide??? (Well according to Wikipedia anyway…)

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.

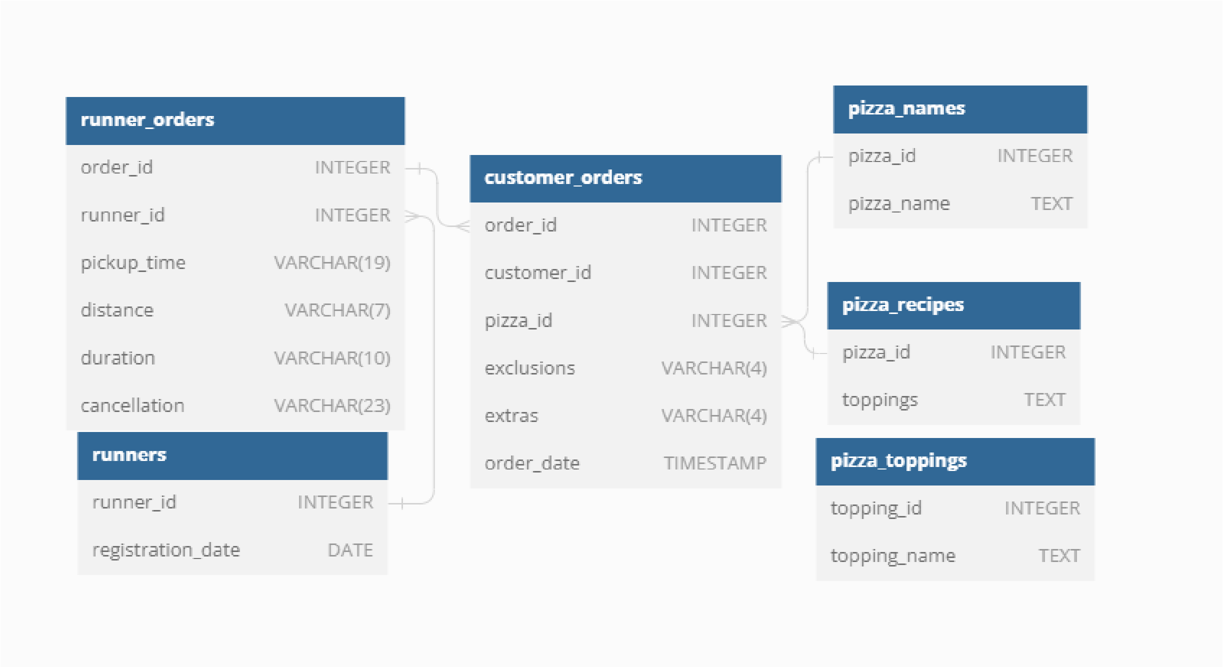
Problem Statement

Because Danny had a few years of experience as a data scientist - he was very aware that data collection was going to be critical for his business' growth.

He has prepared for us an entity relationship diagram of his database design but requires further assistance to clean his data and apply some basic calculations so he can better direct his runners and optimize Pizza Runner's operations.

All datasets exist within the `pizza_runner` database schema - be sure to include this reference within your SQL scripts as you start exploring the data and answering the case study questions.

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

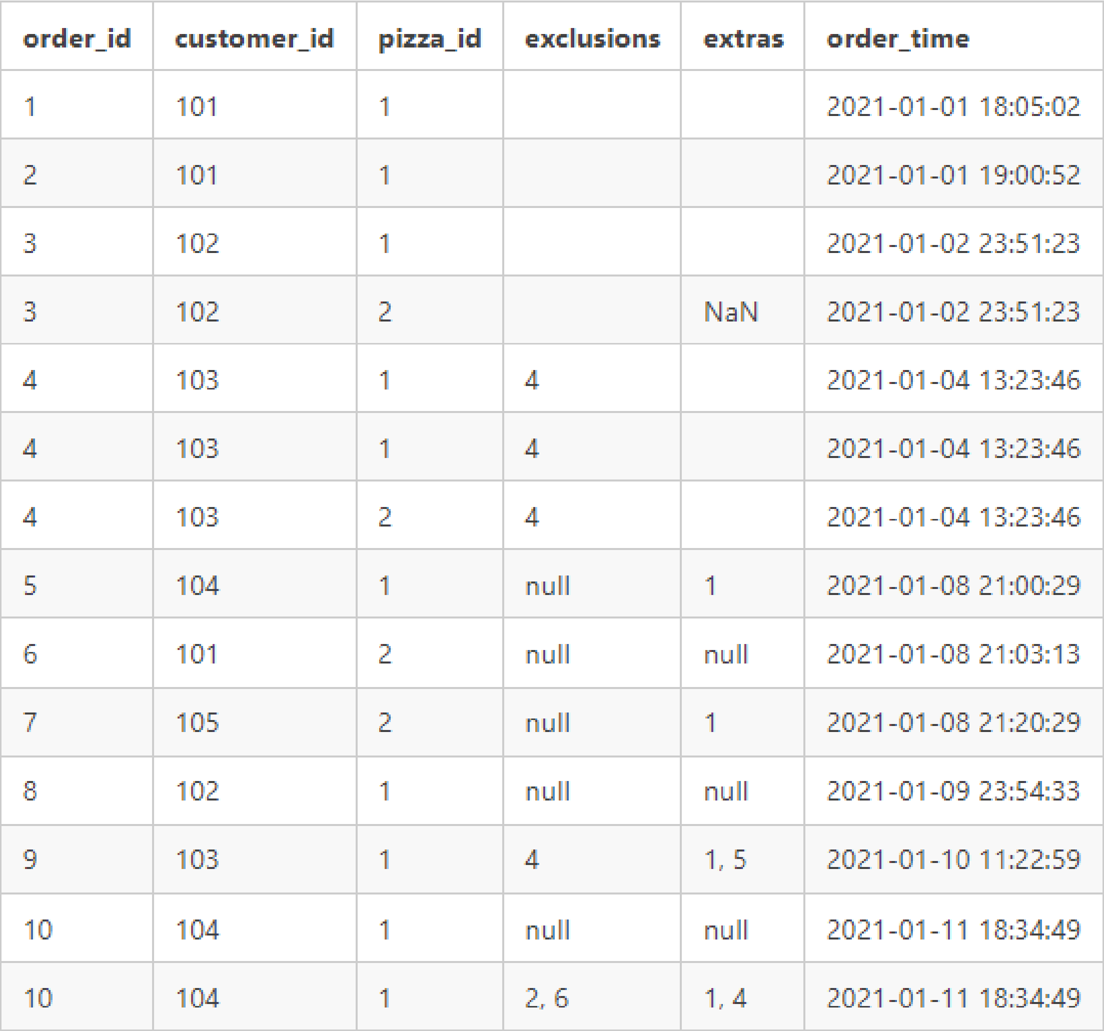
Table 1: runners

The `runners` table shows the `registration_date` for each new runner

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

Table 2: customer_orders

Customer pizza orders are captured in the customer_orders table with 1 row for each individual pizza that is part of the order.
The pizza_id relates to the type of pizza which was ordered whilst the exclusions are the ingredient_id values which should be removed from the pizza and the extras are the ingredient_id values which need to be added to the pizza.
Note that customers can order multiple pizzas in a single order with varying exclusions and extras values even if the pizza is the same type!
The exclusions and extras columns will need to be cleaned up before using them in your queries.



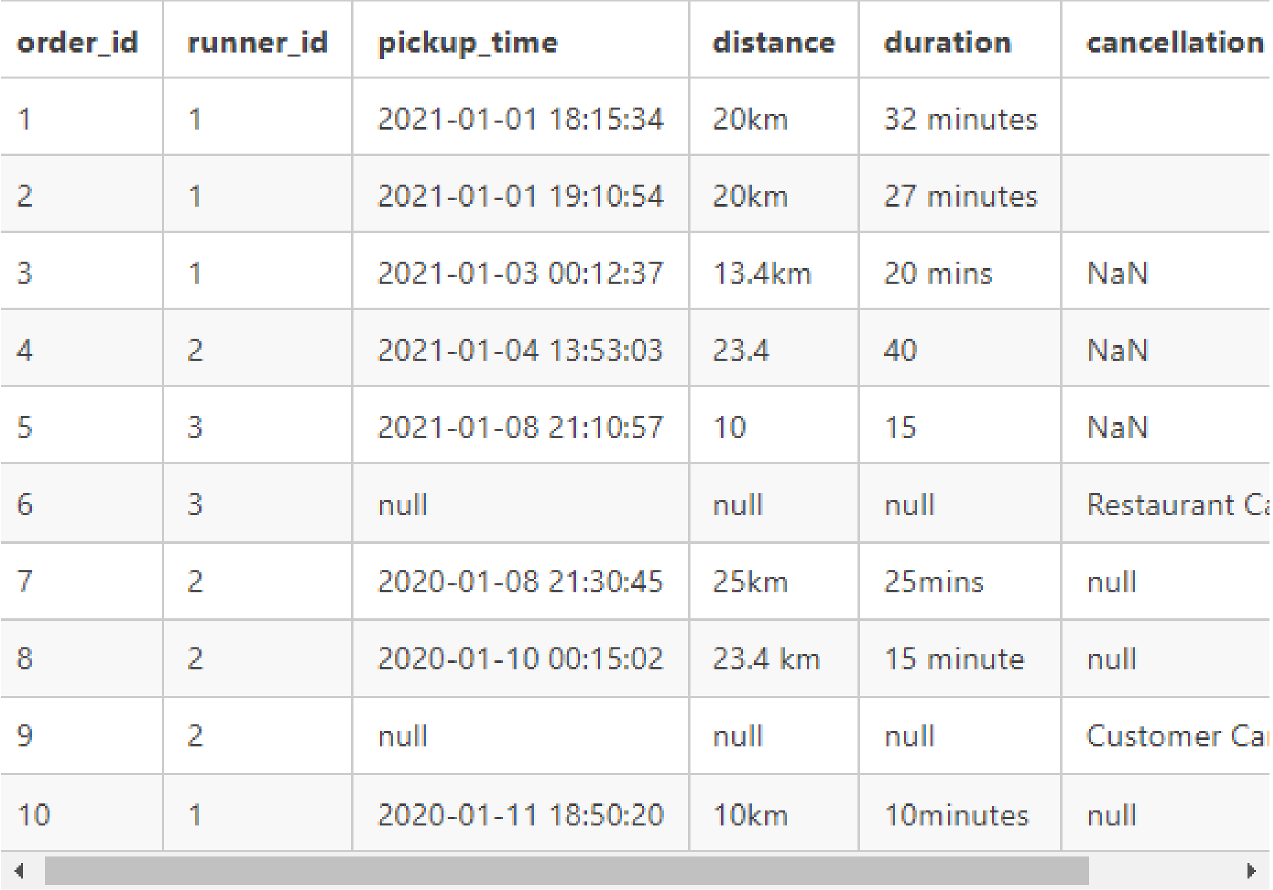
order_id	customer_id	pizza_id	exclusions	extras	order_time
1	101	1			2021-01-01 18:05:02
2	101	1			2021-01-01 19:00:52
3	102	1			2021-01-02 23:51:23
3	102	2		NaN	2021-01-02 23:51:23
4	103	1	4		2021-01-04 13:23:46
4	103	1	4		2021-01-04 13:23:46
4	103	2	4		2021-01-04 13:23:46
5	104	1	null	1	2021-01-08 21:00:29
6	101	2	null	null	2021-01-08 21:03:13
7	105	2	null	1	2021-01-08 21:20:29
8	102	1	null	null	2021-01-09 23:54:33
9	103	1	4	1, 5	2021-01-10 11:22:59
10	104	1	null	null	2021-01-11 18:34:49
10	104	1	2, 6	1, 4	2021-01-11 18:34:49

Table 3: runner_orders

After each orders are received through the system - they are assigned to a runner - however not all orders are fully completed and can be cancelled by the restaurant or the customer.

The pickup_time is the timestamp at which the runner arrives at the Pizza Runner headquarters to pick up the freshly cooked pizzas. The distance and duration fields are related to how far and long the runner had to travel to deliver the order to the respective customer.

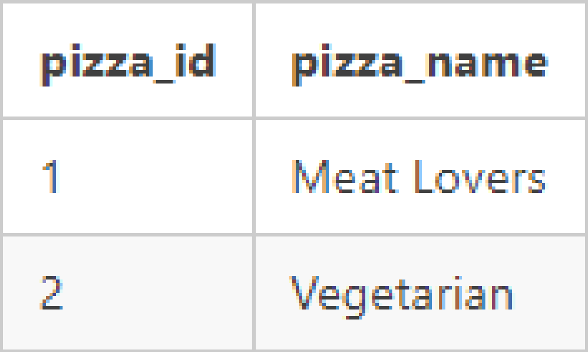
There are some known data issues with this table so be careful when using this in your queries - make sure to check the data types for each column in the schema SQL!



order_id	runner_id	pickup_time	distance	duration	cancellation
1	1	2021-01-01 18:15:34	20km	32 minutes	
2	1	2021-01-01 19:10:54	20km	27 minutes	
3	1	2021-01-03 00:12:37	13.4km	20 mins	NaN
4	2	2021-01-04 13:53:03	23.4	40	NaN
5	3	2021-01-08 21:10:57	10	15	NaN
6	3	null	null	null	Restaurant Ca
7	2	2020-01-08 21:30:45	25km	25mins	null
8	2	2020-01-10 00:15:02	23.4 km	15 minute	null
9	2	null	null	null	Customer Ca
10	1	2020-01-11 18:50:20	10km	10minutes	null

Table 4: pizza_names

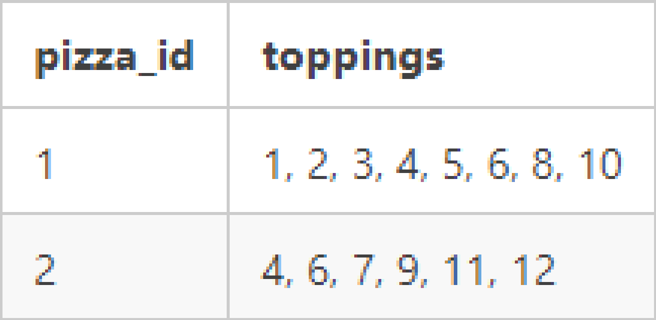
At the moment - Pizza Runner only has 2 pizzas available the Meat Lovers or Vegetarian!



pizza_id	pizza_name
1	Meat Lovers
2	Vegetarian

Table 5: pizza_recipes

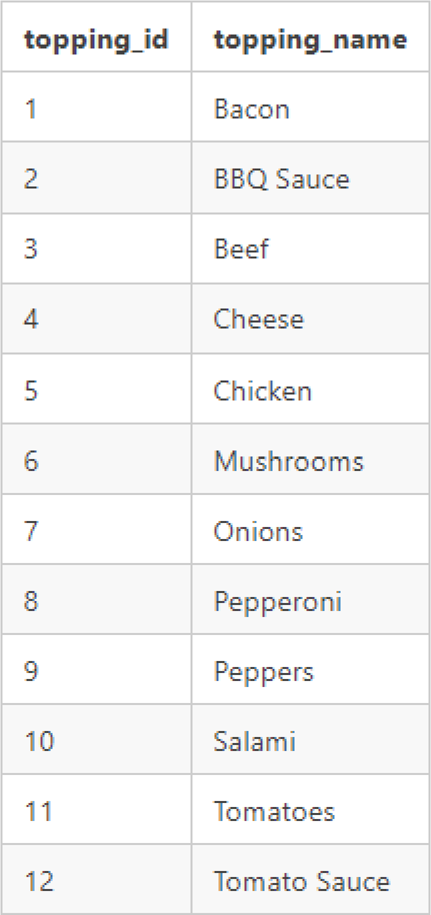
Each pizza_id has a standard set of toppings which are used as part of the pizza recipe.



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

Table 6: pizza_toppings

This table contains all of the topping_name values with their corresponding topping_id value



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

Case Study Questions

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

- Pizza Metrics
- Runner and Customer Experience
- Ingredient Optimisation
- Pricing and Ratings
- Bonus DML Challenges (DML = Data Manipulation Language)

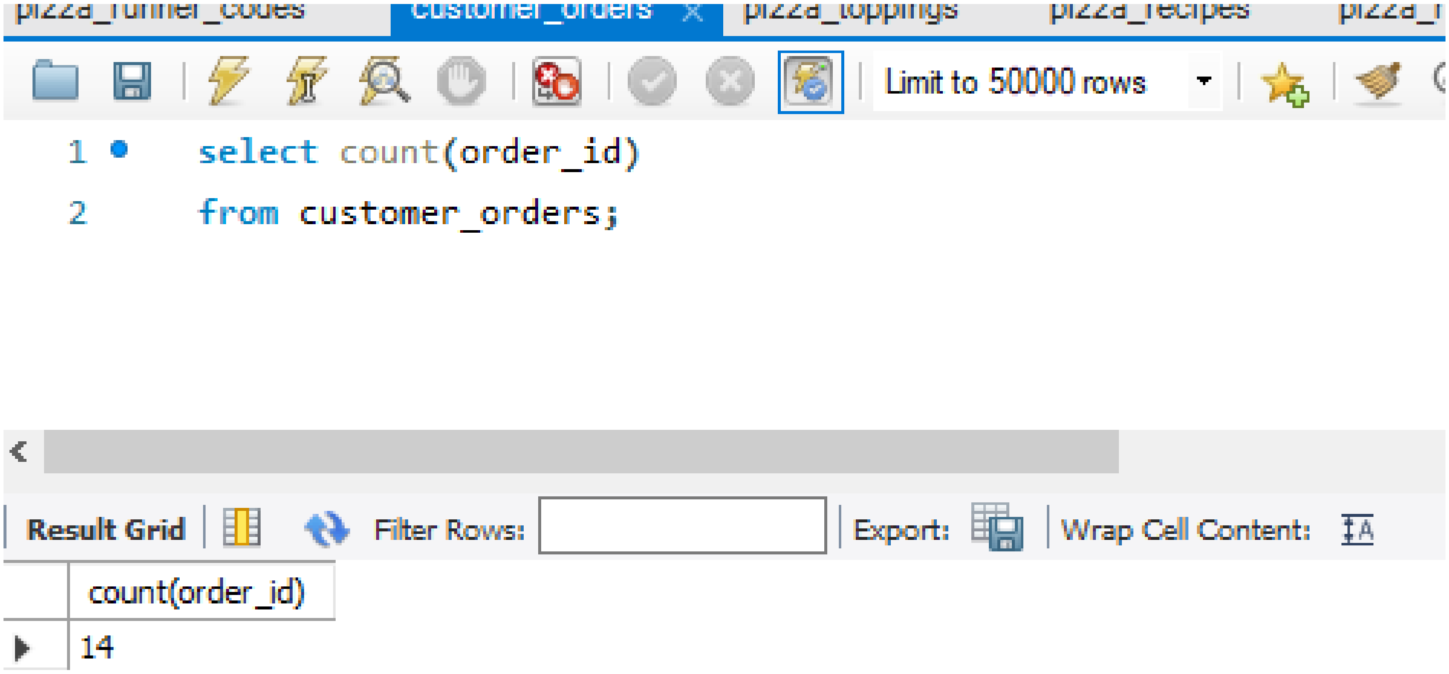
Each of the following case study questions can be answered using a single SQL statement.

Again, there are many questions in this case study - please feel free to pick and choose which ones you'd like to try!

Before you start writing your SQL queries however - you might want to investigate the data, you may want to do something with some of those null values and data types in the customer_orders and runner_orders tables!

A. Pizza Metrics

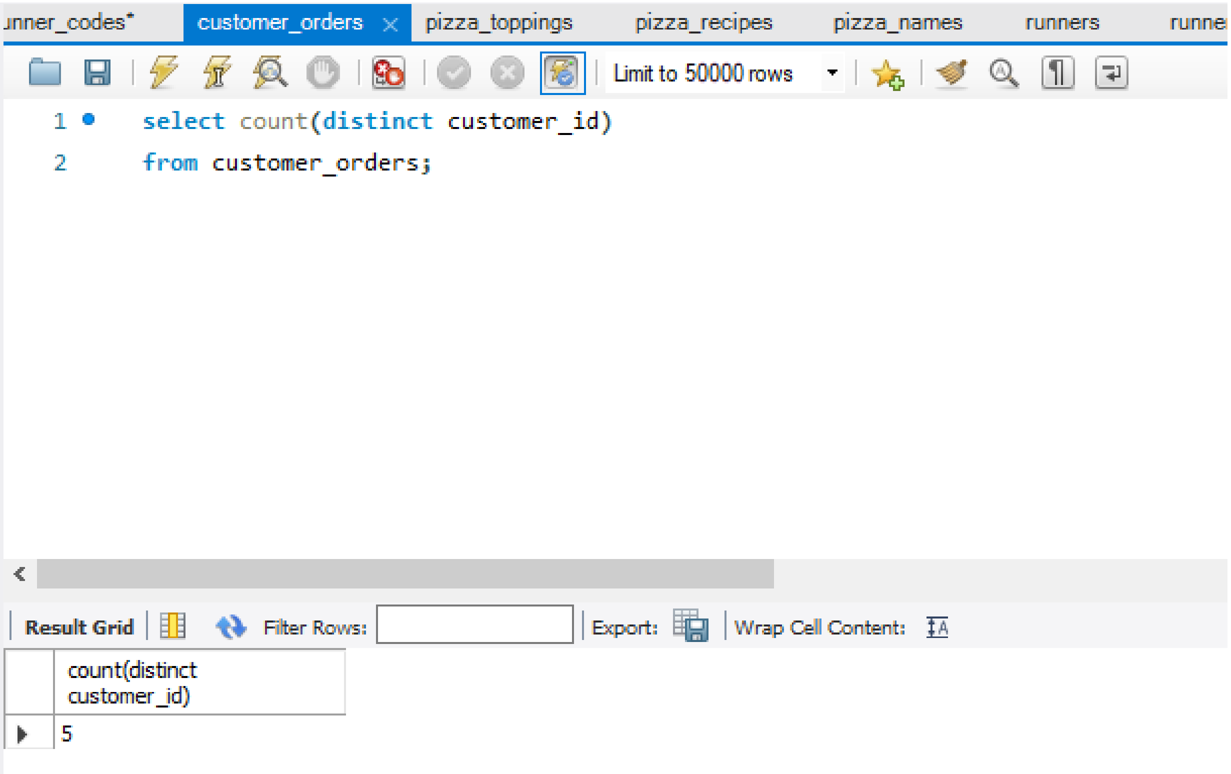
1. How many pizzas were ordered?



```
select count(order_id)
from customer_orders;
```

```
count(order_id)
14
```

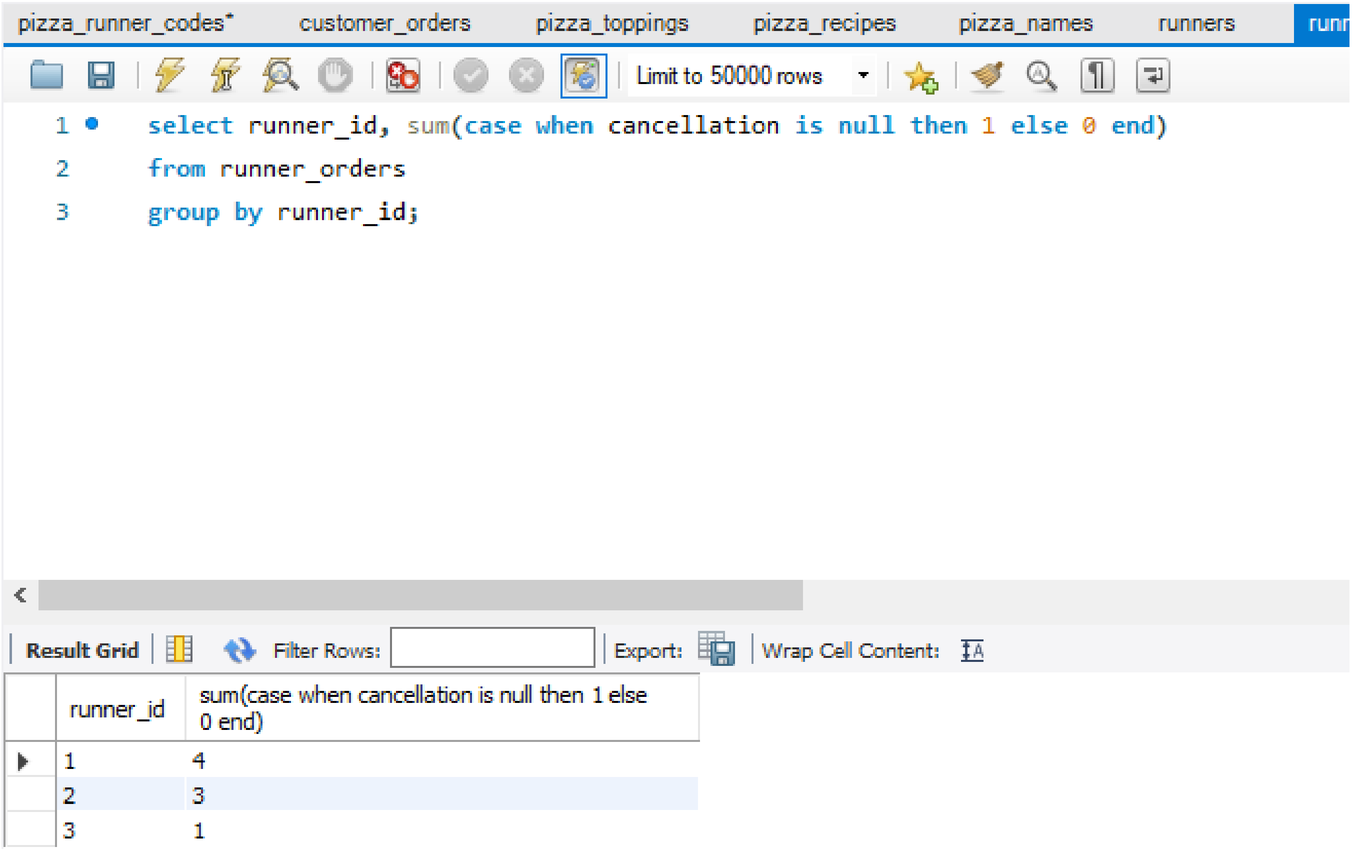
2. How many unique customer orders were made?



```
select count(distinct customer_id)
from customer_orders;
```

```
count(distinct
customer_id)
5
```

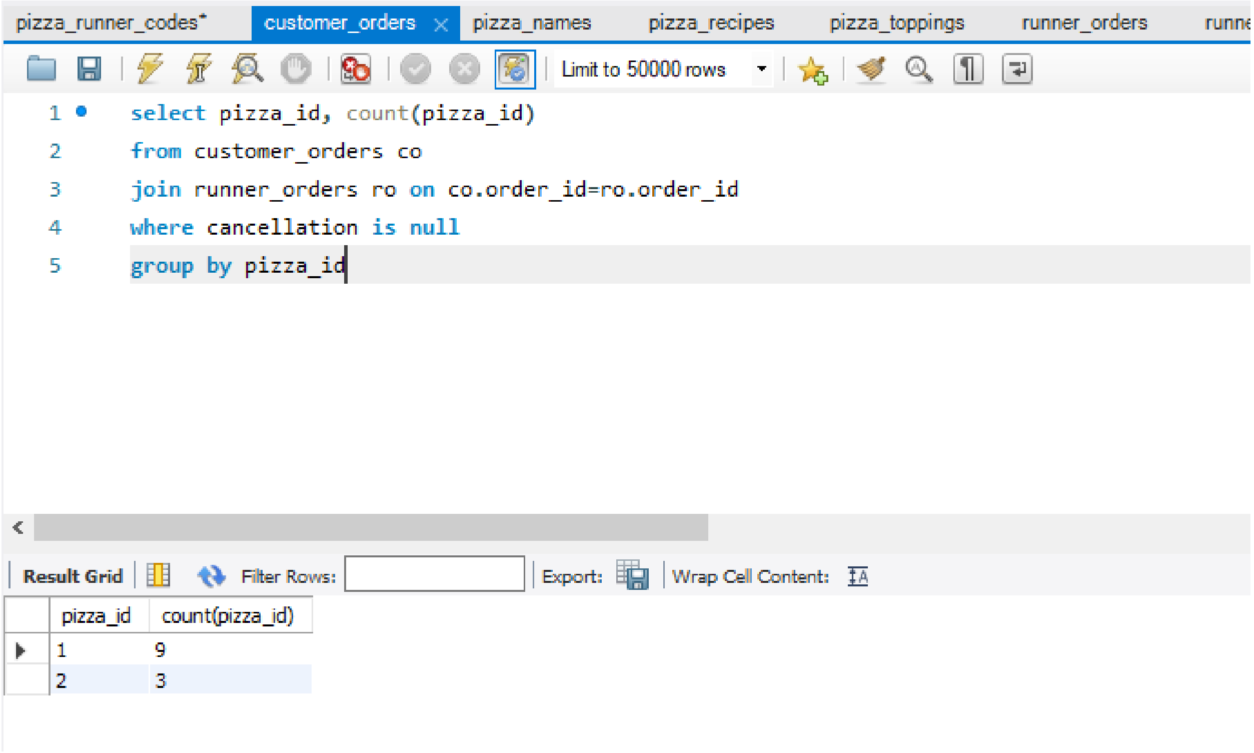
3. How many successful orders were delivered by each runner?



```
select runner_id, sum(case when cancellation is null then 1 else 0 end)
from runner_orders
group by runner_id;
```

runner_id	sum(case when cancellation is null then 1 else 0 end)
1	4
2	3
3	1

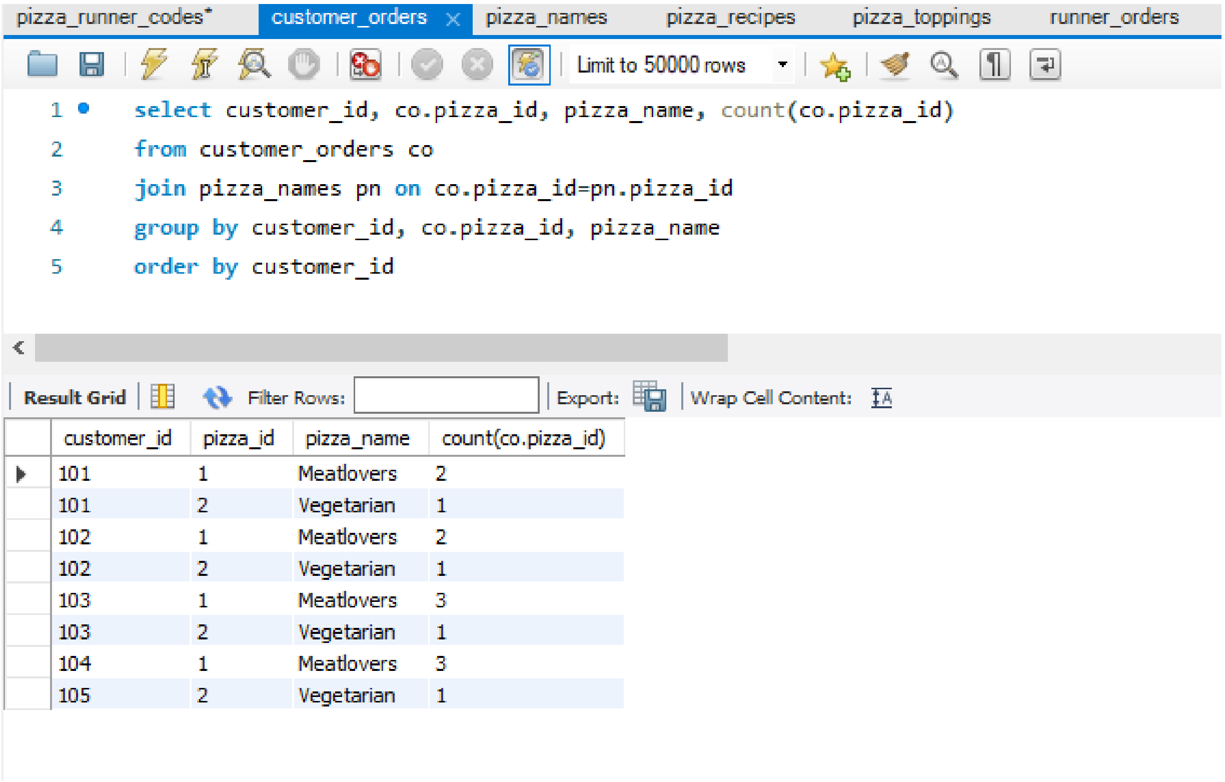
4. How many of each type of pizza was delivered?



```
select pizza_id, count(pizza_id)
from customer_orders co
join runner_orders ro on co.order_id=ro.order_id
where cancellation is null
group by pizza_id
```

pizza_id	count(pizza_id)
1	9
2	3

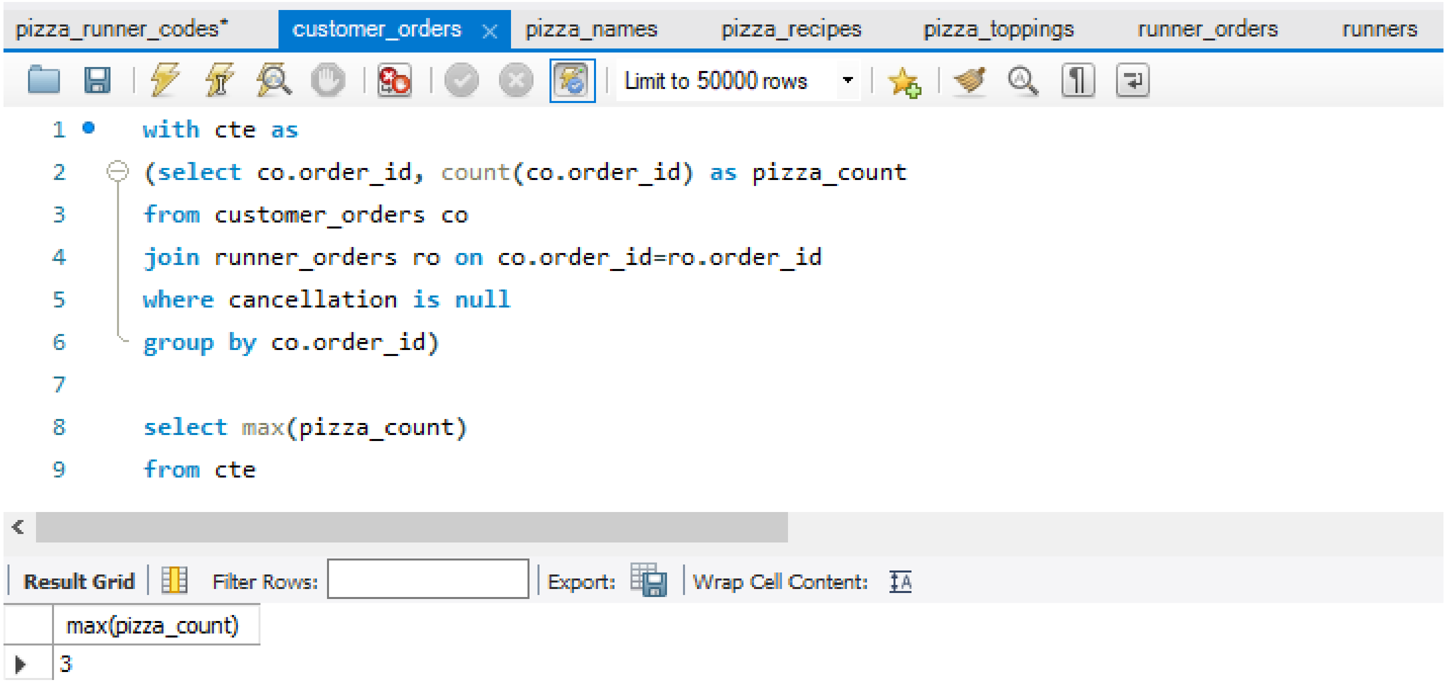
5. How many Vegetarian and Meatlovers were ordered by each customer?



```
select customer_id, co.pizza_id, pizza_name, count(co.pizza_id)
from customer_orders co
join pizza_names pn on co.pizza_id=pn.pizza_id
group by customer_id, co.pizza_id, pizza_name
order by customer_id
```

customer_id	pizza_id	pizza_name	count(co.pizza_id)
101	1	Meatlovers	2
101	2	Vegetarian	1
102	1	Meatlovers	2
102	2	Vegetarian	1
103	1	Meatlovers	3
103	2	Vegetarian	1
104	1	Meatlovers	3
105	2	Vegetarian	1

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

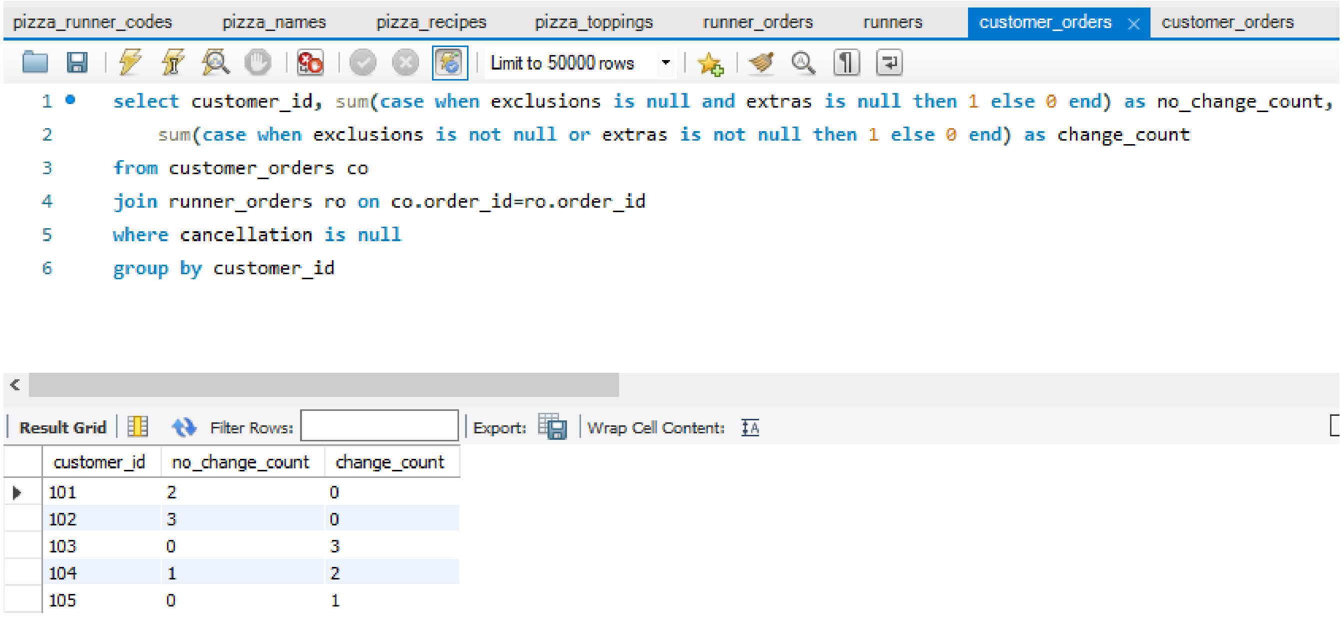


```
with cte as
(select co.order_id, count(co.order_id) as pizza_count
from customer_orders co
join runner_orders ro on co.order_id=ro.order_id
where cancellation is null
group by co.order_id)

select max(pizza_count)
from cte
```

max(pizza_count)
3

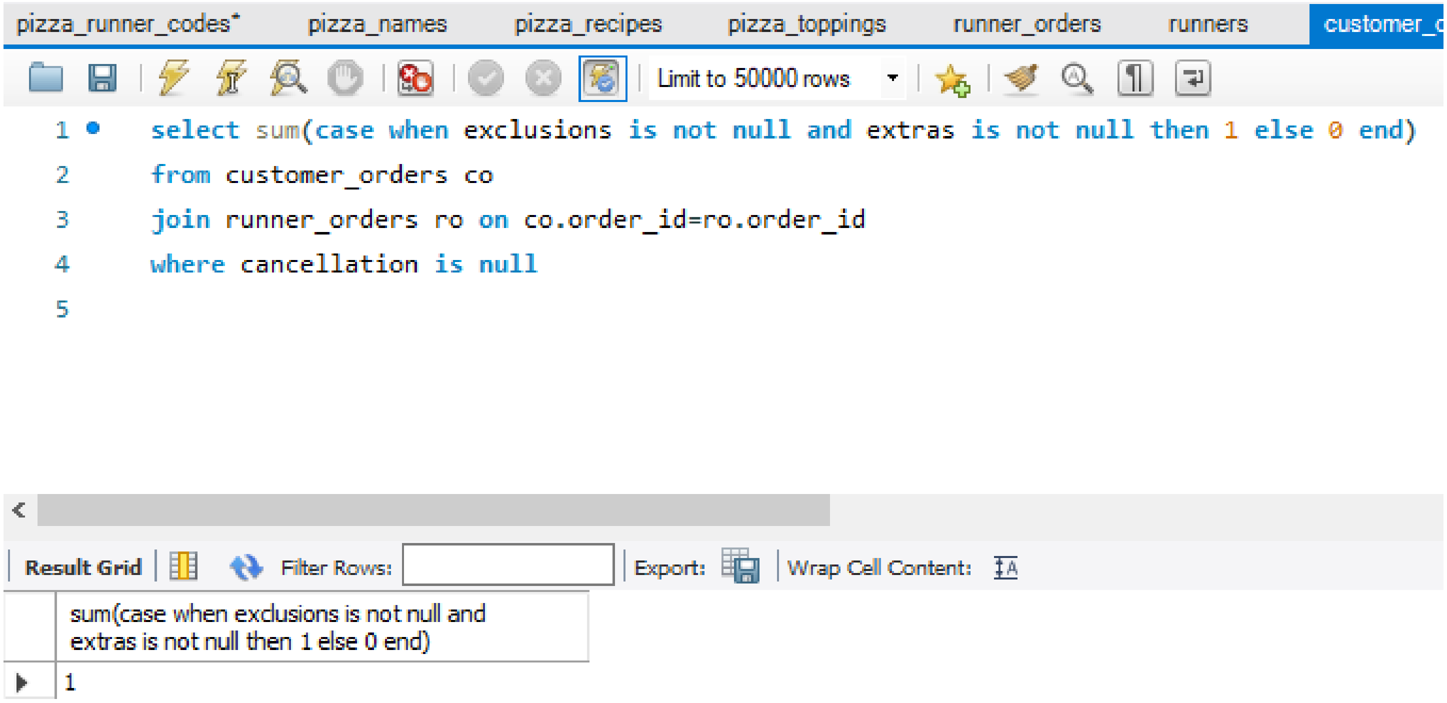
7. For each customer, how many delivered pizzas had at least 1 change and how many had no changes?



```sql
select customer_id, sum(case when exclusions is null and extras is null then 1 else 0 end) as no_change_count,
    sum(case when exclusions is not null or extras is not null then 1 else 0 end) as change_count
from customer_orders co
join runner_orders ro on co.order_id=ro.order_id
where cancellation is null
group by customer_id
```

customer_id	no_change_count	change_count
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?

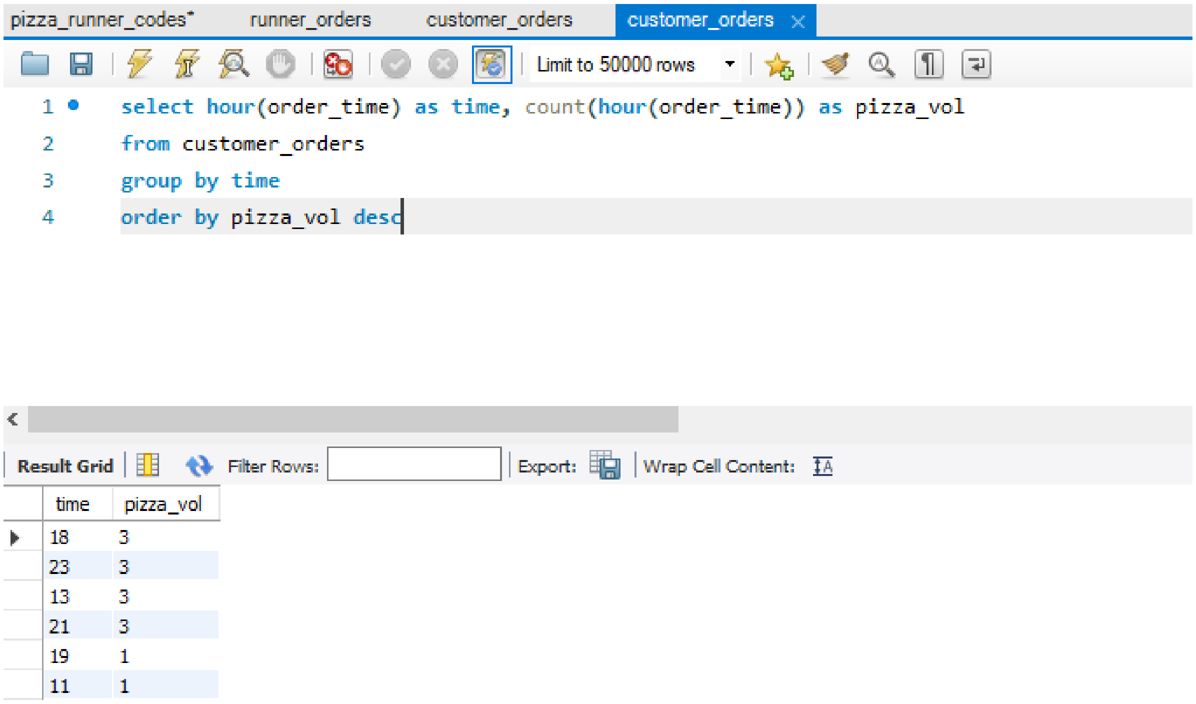


```sql
select sum(case when exclusions is not null and extras is not null then 1 else 0 end)
from customer_orders co
join runner_orders ro on co.order_id=ro.order_id
where cancellation is null

```

sum(case when exclusions is not null and extras is not null then 1 else 0 end)
1

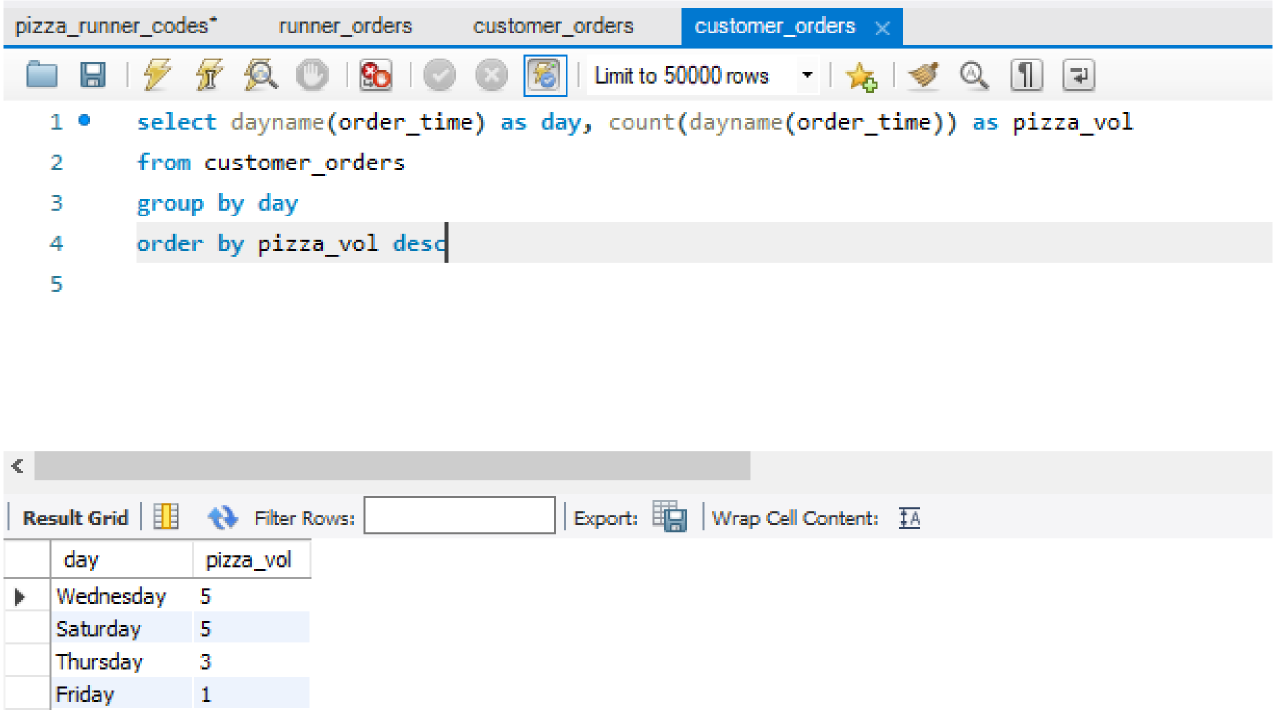
9. What was the total volume of pizzas ordered for each hour of the day?



```sql
select hour(order_time) as time, count(hour(order_time)) as pizza_vol
from customer_orders
group by time
order by pizza_vol desc
```

time	pizza_vol
18	3
23	3
13	3
21	3
19	1
11	1

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



```sql
select dayname(order_time) as day, count(dayname(order_time)) as pizza_vol
from customer_orders
group by day
order by pizza_vol desc
```

day	pizza_vol
Wednesday	5
Saturday	5
Thursday	3
Friday	1

Insights

The following topics are completely covered in this case study:

- Common Table Expressions
- Group By and Aggregates
- Table Joins
- Case When clause
- Subqueries
- Dayname and hour function
- Data cleaning

The following insights can be gathered for this case study:

- Meat Lovers is the most demanding pizza.
- Customer_id 103 is the most frequent customer.
- Runner_id 1 has most delivered pizza.
- 1pm, 6pm, 9pm and 11pm are the most busy hours for pizza sale.
- Wednesdays and Saturdays are the most voluminous sale days.