

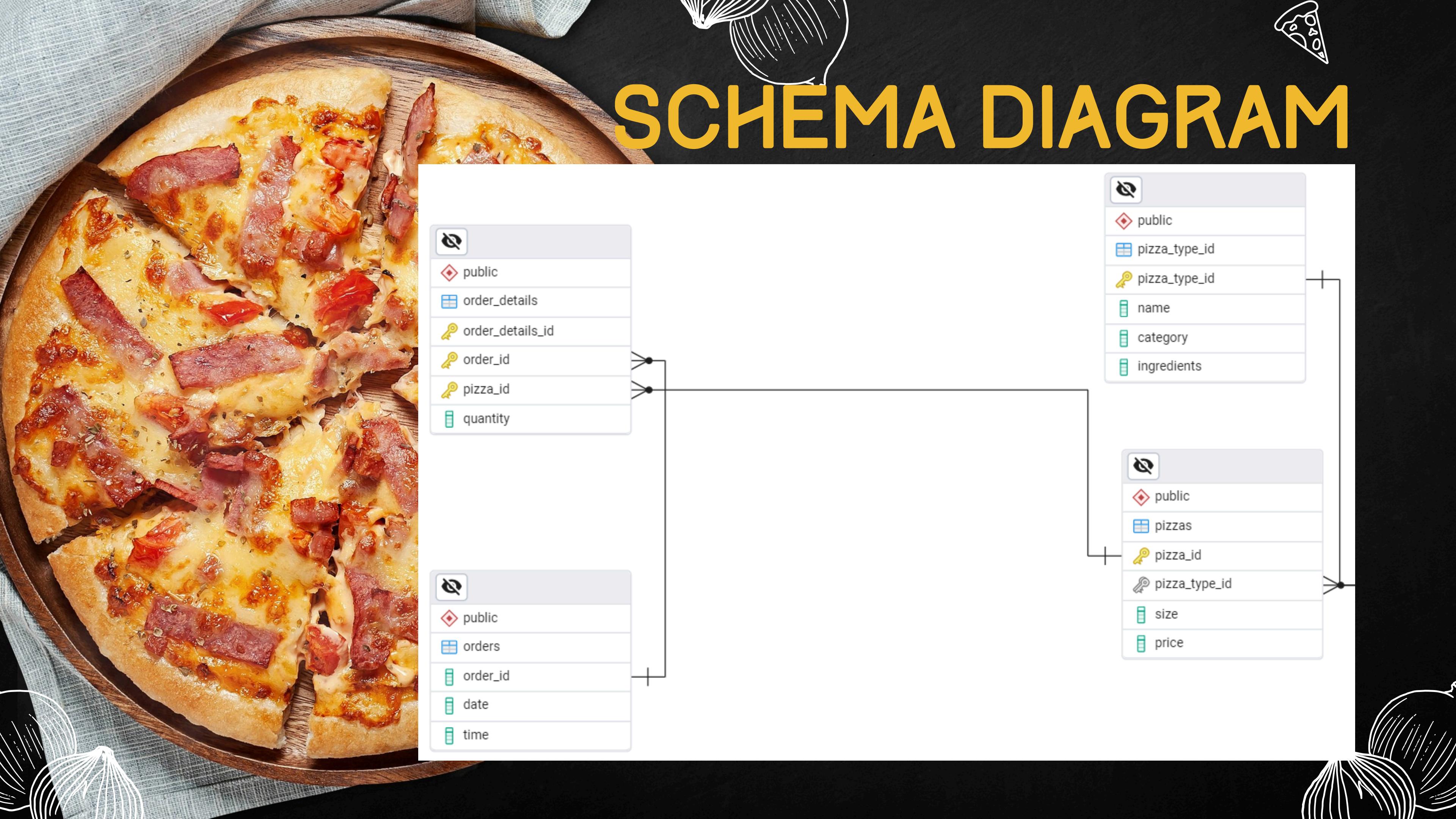
# HELLO

My name is Mangesh Sambare, and in this project I have used SQL queries to solve questions related to pizza sales.



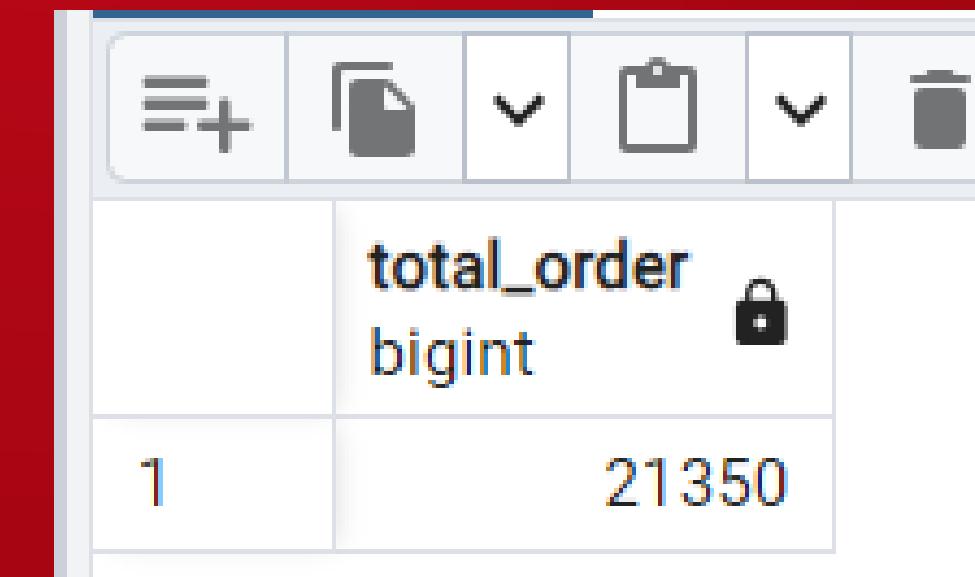


# SCHEMA DIAGRAM



# 1) RETRIEVE THE TOTAL NUMBER OF ORDERS PLACED.

```
9  
10  select  
11      count(order_id) as total_order  
12  from  
13      orders;
```

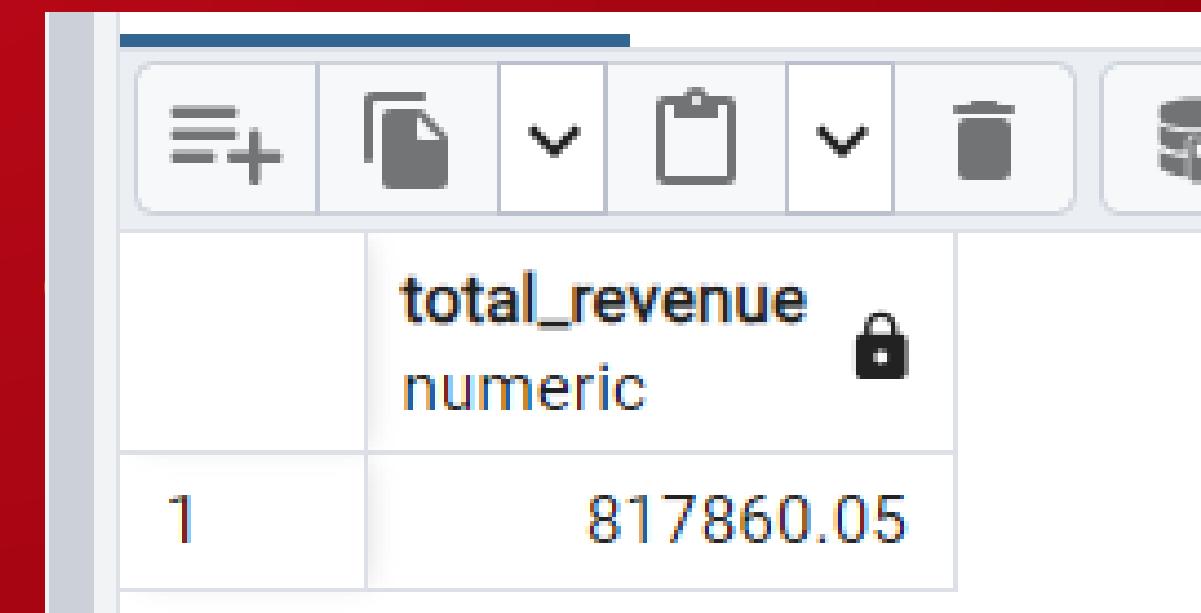


The screenshot shows a MySQL Workbench interface with a toolbar at the top and a results grid below. The results grid has two columns: 'total\_order' and 'bigint'. The value '1' is in the first row under 'total\_order', and the value '21350' is in the same row under 'bigint'. A lock icon is present in the 'bigint' column header.

total_order	bigint
1	21350

## 2) CALCULATE THE TOTAL REVENUE GENERATED FROM PIZZA SALES..

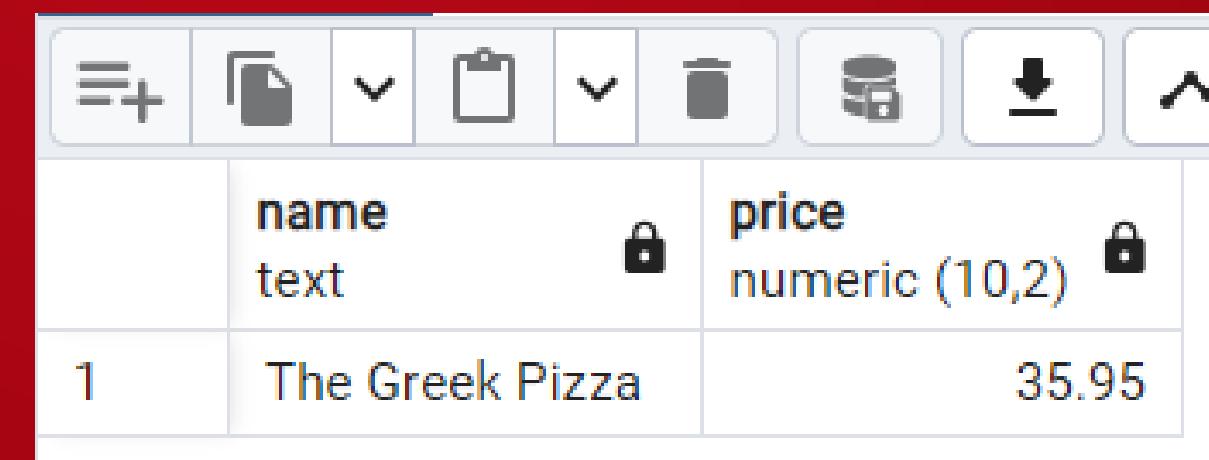
```
17  
18  select  
19      round(sum(order_details.quantity * pizzas.price),2) as total_revenue  
20  from  
21      order_details join pizzas  
22  on pizzas.pizza_id = order_details.pizza_id
```



	total_revenue
1	817860.05

### 3) IDENTIFY THE HIGHEST-PRICED PIZZA..

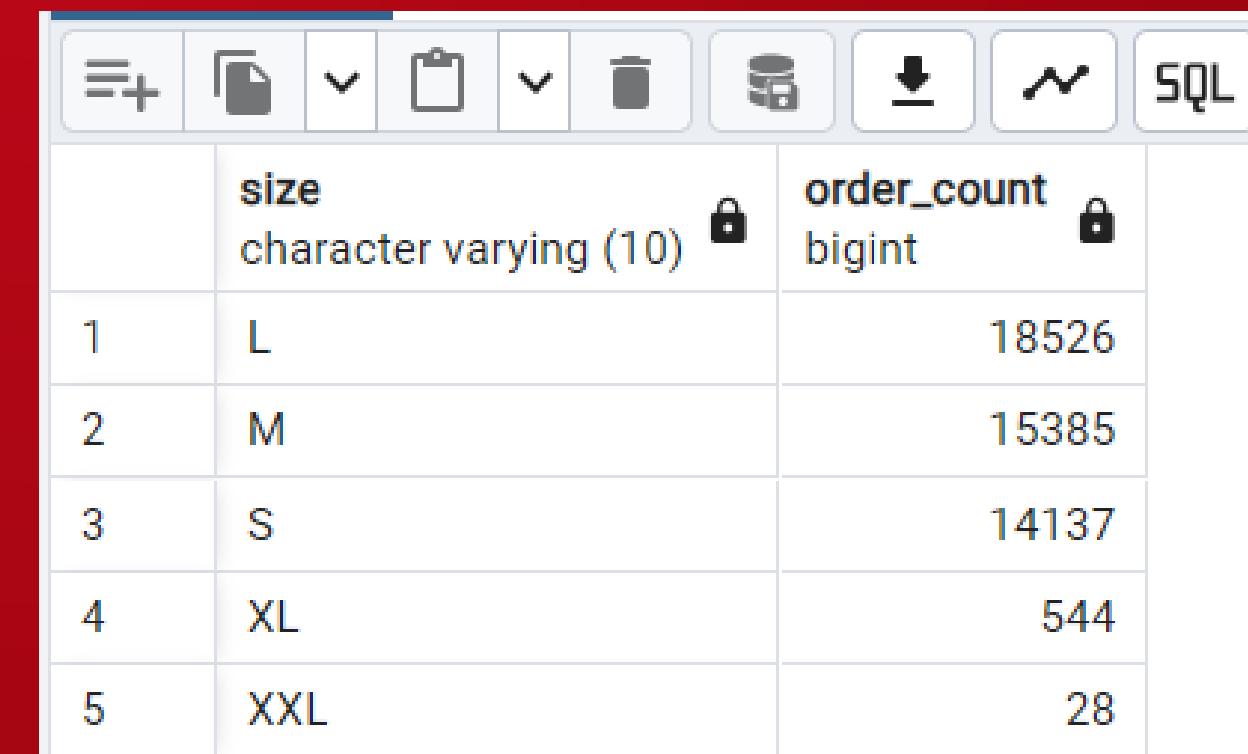
```
27 select
28     pizza_types.name,pizzas.price
29 from
30     pizza_types join pizzas
31 on pizza_types.pizza_type_id = pizzas.pizza_type_id
32 order by
33     pizzas.price desc limit 1;
```



	name text	price numeric (10,2)
1	The Greek Pizza	35.95

## 4) IDENTIFY THE MOST COMMON PIZZA SIZE ORDERED..

```
37  select
38      pizzas.size, count(order_details.order_details_id) as order_count
39  from
40      pizzas join order_details
41  on pizzas.pizza_id = order_details.pizza_id
42  group by pizzas.size order by order_count desc;
```



	size character varying (10)	order_count bigint
1	L	18526
2	M	15385
3	S	14137
4	XL	544
5	XXL	28

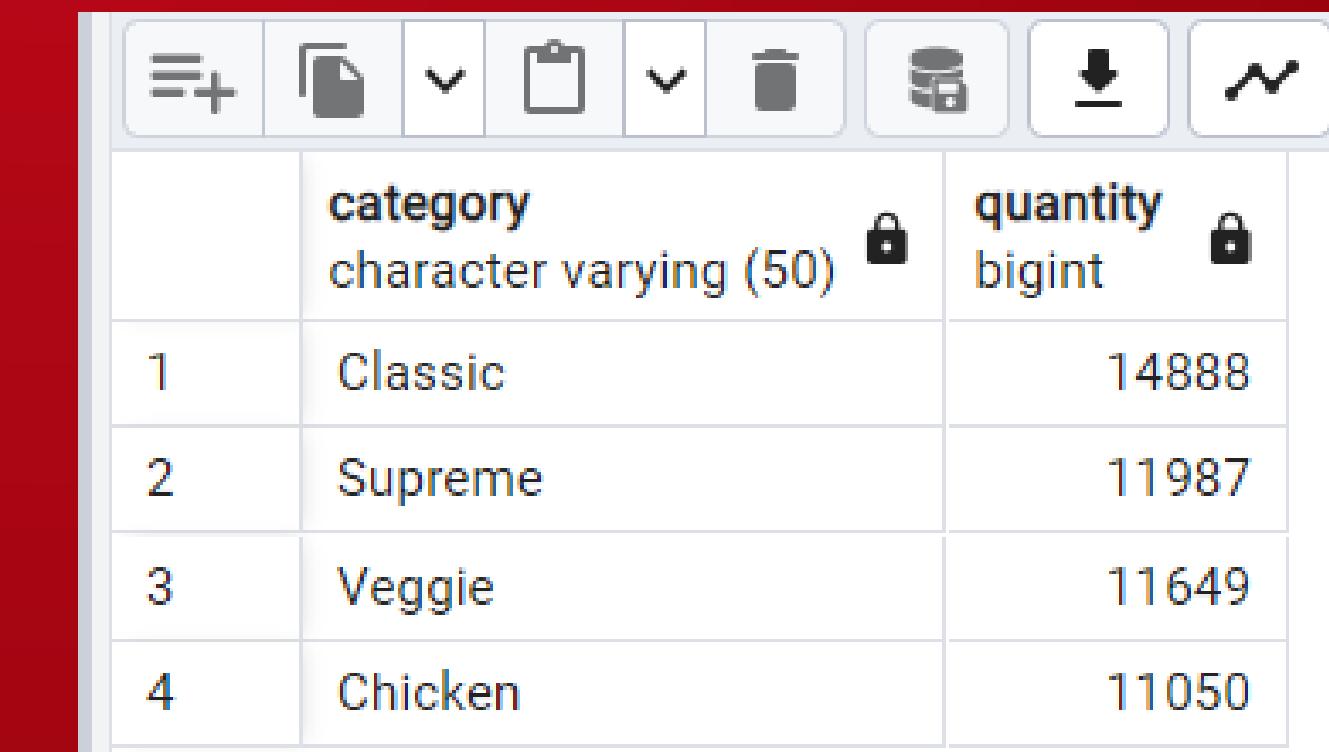
## 5) LIST THE TOP 5 MOST ORDERED PIZZA TYPES ALONG WITH THEIR QUANTITIES..

```
46 select
47     pizza_types.name , sum(order_details.quantity) as quantity
48 from pizza_types join pizzas
49 on pizza_types.pizza_type_id = pizzas.pizza_type_id
50 join order_details
51 on order_details.pizza_id = pizzas.pizza_id
52 group by pizza_types.name order by quantity desc limit 5;
```

	name text	quantity bigint
1	The Classic Deluxe Pizza	2453
2	The Barbecue Chicken Pizza	2432
3	The Hawaiian Pizza	2422
4	The Pepperoni Pizza	2418
5	The Thai Chicken Pizza	2371

## 6) JOIN THE NECESSARY TABLES TO FIND THE TOTAL QUANTITY OF EACH PIZZA CATEGORY ORDERED.

```
11 ✓ select
12     pizza_types.category, sum(order_details.quantity) as quantity
13 from pizza_types join pizzas
14 on pizza_types.pizza_type_id = pizzas.pizza_type_id
15 join order_details
16 on order_details.pizza_id = pizzas.pizza_id
17 group by pizza_types.category order by quantity desc limit 5;
```



	category	quantity
1	Classic	14888
2	Supreme	11987
3	Veggie	11649
4	Chicken	11050

## 7) DETERMINE THE DISTRIBUTION OF ORDERS BY HOUR OF THE DAY..

```
23 select  
24     extract(hour from time) as order_by_hours, count(order_id) as order_count  
25 from orders  
26 group by extract(hour from time) order by order_by_hours asc;  
27
```

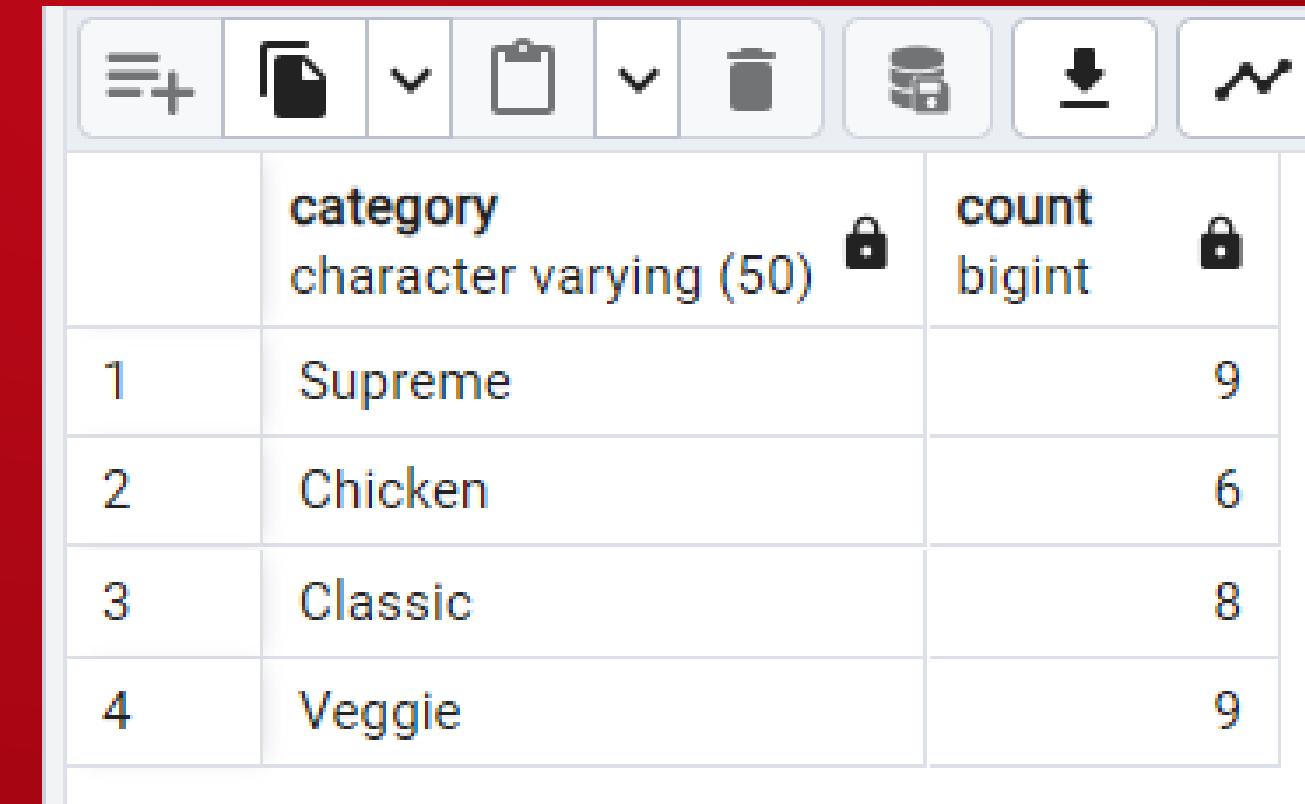


The screenshot shows a database query results interface with a toolbar at the top containing various icons for file operations. The results are presented in a table with two columns: 'order\_by\_hours' (numeric) and 'order\_count' (bigint). The data shows the number of orders per hour from 1 to 23.

	order_by_hours numeric	order_count bigint
1	9	1
2	10	8
3	11	1231
4	12	2520
5	13	2455
6	14	1472
7	15	1468
8	16	1920
9	17	2336
10	18	2399
11	19	2009
12	20	1642
13	21	1198
14	22	663
15	23	28

## 8) JOIN RELEVANT TABLES TO FIND THE CATEGORY-WISE DISTRIBUTION OF PIZZAS.

```
32 ✓ select category, count(name) from pizza_types  
33   group by category;
```



	category	count	
	character varying (50)	bigint	
1	Supreme	9	
2	Chicken	6	
3	Classic	8	
4	Veggie	9	

## 9) GROUP THE ORDERS BY DATE AND CALCULATE THE AVERAGE NUMBER OF PIZZAS ORDERED PER DAY.

```
38 v select
39      round(avg(quantity),0) as avg_pizza_order_by_perday
40  from
41    (select
42      orders.date, sum(order_details.quantity) as quantity
43    from orders join order_details
44    on orders.order_id = order_details.order_id
45    group by orders.date)
46  as order_quantity
47
```

Data Output    Messages    Notifications

	avg_pizza_order_by_perday
1	138

## 10) DETERMINE THE TOP 3 MOST ORDERED PIZZA TYPES BASED ON REVENUE.

```
51 select
52     pizza_types.name , sum(order_details.quantity * pizzas.price) as revenue
53 from pizza_types join pizzas
54 on pizzas.pizza_type_id = pizza_types.pizza_type_id
55 join order_details
56 on order_details.pizza_id = pizzas.pizza_id
57 group by pizza_types.name order by revenue desc limit 3;
```

Data Output    Messages    Notifications

≡+ ↻ ⌂ ↻ ⏷ ↻ ⏸ ↻ ⏹ ↻ ⏵ ↻ ⏴ ↻ SQL

	name	revenue
	text	numeric
1	The Thai Chicken Pizza	43434.25
2	The Barbecue Chicken Pizza	42768.00
3	The California Chicken Pizza	41409.50

## 11) CALCULATE THE PERCENTAGE CONTRIBUTION OF EACH PIZZA TYPE TO TOTAL REVENUE.

```
10  select pizza_types.category,  
11      round(sum(order_details.quantity * pizzas.price)/  
12      (select round(sum(order_details.quantity *pizzas.price),2)  
13      as total_sales  
14  from order_details  
15  join pizzas  
16  on pizzas.pizza_id =order_details.pizza_id)*100,2) as revenue  
17  from pizza_types join pizzas  
18  on pizza_types.pizza_type_id = pizzas.pizza_type_id  
19  join order_details  
20  on order_details.pizza_id = pizzas.pizza_id  
21  group by pizza_types.category order by revenue desc;
```

	category character varying (50)	revenue numeric
1	Classic	26.91
2	Supreme	25.46
3	Chicken	23.96
4	Veggie	23.68

## 12) ANALYZE THE CUMULATIVE REVENUE GENERATED OVER TIME.

```
28 ✓ select date, sum(revenue) over(order by date) as cum_revenue  
29 from  
30   (select  
31     orders.date, sum(order_details.quantity * pizzas.price) as revenue  
32   from order_details join pizzas  
33   on order_details.pizza_id = pizzas.pizza_id  
34   join orders  
35   on orders.order_id = order_details.order_id  
36   group by orders.date)  
37 as sales;
```

	date	cum_revenue
	date	numeric
1	2015-01-01	2713.85
2	2015-01-02	5445.75
3	2015-01-03	8108.15
4	2015-01-04	9863.60
5	2015-01-05	11929.55
6	2015-01-06	14358.50
7	2015-01-07	16560.70
8	2015-01-08	19399.05
9	2015-01-09	21526.40
10	2015-01-10	23990.35
11	2015-01-11	25862.65
12	2015-01-12	27781.70
13	2015-01-13	29831.30
14	2015-01-14	32358.70
15	2015-01-15	34343.50
16	2015-01-16	36937.65
17	2015-01-17	39001.75
18	2015-01-18	40978.60
19	2015-01-19	43365.75
20	2015-01-20	45763.65
21	2015-01-21	47904.20

# 13) DETERMINE THE TOP 3 MOST ORDERED PIZZA TYPES BASED ON REVENUE FOR EACH PIZZA CATEGORY.

```
43 v select name, revenue
44   from (select category, name, revenue,
45           rank() over(partition by category order by revenue desc)
46           as rn
47         from
48           (select
49             pizza_types.category , pizza_types.name,
50             sum((order_details.quantity)*pizzas.price) as revenue
51             from pizza_types join pizzas
52             on pizza_types.pizza_type_id = pizzas.pizza_type_id
53             join order_details
54             on order_details.pizza_id = pizzas.pizza_id
55             group by pizza_types.category, pizza_types.name)
56           as a)
57     as b
58   where rn <=3;
```

	name	revenue
1	The Thai Chicken Pizza	43434.25
2	The Barbecue Chicken Pizza	42768.00
3	The California Chicken Pizza	41409.50
4	The Classic Deluxe Pizza	38180.50
5	The Hawaiian Pizza	32273.25
6	The Pepperoni Pizza	30161.75
7	The Spicy Italian Pizza	34831.25
8	The Italian Supreme Pizza	33476.75
9	The Sicilian Pizza	30940.50
10	The Four Cheese Pizza	32265.70
11	The Mexicana Pizza	26780.75
12	The Five Cheese Pizza	26066.50



# THANK YOU

Mangesh Sambare

🌐 sambaremangesh123@gmail.com

📍 Nagpur, Maharashtra