

In this presentation, I'll be showcasing the solutions to 13 questions that I executed it in MySQL. Each question and its corresponding solution are presented in the slides.

This project involves analyzing the Pizza Sales data of a restaurant. I hope you find the insights and results as engaging as I did.

## Highlights:

- Objective: To analyze and derive insights from the restaurant's Pizza Sales data.
- Scope: Addressing 13 different queries to extract meaningful information from the database.
- Tools Used: MySQL for database querying and PowerPoint for presenting the results.

Please follow along and enjoy the journey through my first SQL project!

# Retrieve the total number of orders placed.

Solution :

```
3 •   SELECT
4           COUNT(order_id) AS total_orders
5 FROM
6     orders;
```

Kaushal\_Verma

Output:

The screenshot shows a MySQL query editor interface. At the top, there is a code editor window containing the SQL query. Below it is a toolbar with zoom controls (100%, 1:1), a result grid icon, a refresh icon, a search bar labeled "Filter Rows: Search", and an export icon. The main area displays the results of the query in a table format.

total_orders
21350

# Calculate the total revenue generated from pizza sales.

Solution :

```
3 •   SELECT
4       ROUND(SUM(od.quantity * pi.price), 2) AS total_revenue
5   FROM
6       order_details od
7   JOIN
8       pizzas pi ON pi.pizza_id = od.pizza_id
```

Kaushal\_Verma

100% 1:3

**Result Grid** Filter Rows: Search Export:

total_revenue
817860.05

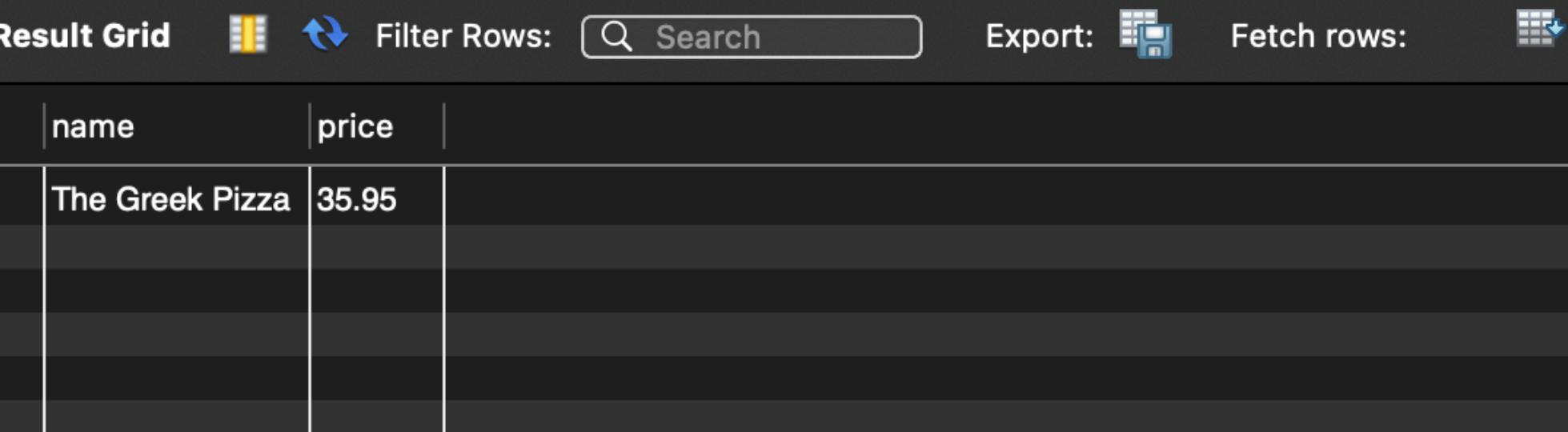
Output:

# Identify the highest-priced pizza.

Solution :

```
3 •   SELECT
4       pizza_types.name, pizzas.price
5   FROM
6       pizza_types
7   JOIN
8       pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
9   ORDER BY pizzas.price DESC
10  LIMIT 1;
```

Kaushal\_Verma



The screenshot shows a MySQL command-line interface window. At the top, there is a code editor containing the SQL query. Below the code editor is a toolbar with various icons. The main area displays the results of the query in a table format. The table has two columns: 'name' and 'price'. The single row returned is 'The Greek Pizza' with a price of '35.95'.

name	price
The Greek Pizza	35.95

Output:

# Identify the most common pizza size ordered.

Solution :

```
3 •   SELECT
4       pizzas.size, COUNT(order_details.pizza_id) AS order_count
5   FROM
6       order_details
7       JOIN
8           pizzas ON order_details.pizza_id = pizzas.pizza_id
9   GROUP BY pizzas.size
10  ORDER BY order_count DESC
11  LIMIT 1;
```

Kaushal\_Verma

The screenshot shows a database query interface with the following details:

- SQL Query:** The code provided in the previous block is executed.
- Progress Bar:** Shows 0% completion.
- Timestamp:** 8:11.
- Result Grid:** A table with two columns: "size" and "order\_count".
- Data:** One row is present: "L" in the "size" column and "18526" in the "order\_count" column.
- Toolbar:** Includes "result Grid", "Filter Rows:", "Search" (with a magnifying glass icon), "Export:" (with a file icon), "Fetch rows:" (with a grid icon), and other standard database icons.

Output:

# List the top 5 most ordered pizza types along with their quantities.

Solution :

```
3 •   SELECT
4     pizza_types.name, SUM(order_details.quantity) AS quantity
5   FROM
6     pizza_types
7       JOIN
8       pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
9       JOIN
10      order_details ON order_details.pizza_id = pizzas.pizza_id
11    GROUP BY pizza_types.name
12    ORDER BY quantity DESC
13    LIMIT 5;
14
15
```

Result Grid   Filter Rows:  Search   Export: Fetch rows:

name	quantity
The Classic Deluxe Pizza	2453
The Barbecue Chicken Pizza	2432
The Hawaiian Pizza	2422
The Pepperoni Pizza	2418
The Thai Chicken Pizza	2371

Kaushal\_Verma

Output:

# Join the necessary tables to find the total quantity of each pizza category ordered.

Solution :

```
4 • SELECT
5     pizza_types.category,
6     SUM(order_details.quantity) AS quantity
7 FROM
8     pizza_types
9     JOIN
10    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
11    JOIN
12    order_details ON order_details.pizza_id = pizzas.pizza_id
13 GROUP BY pizza_types.category
14 ORDER BY quantity DESC;
15
```

100% 23:14

Result Grid Filter Rows: Search Export:

category	quantity
Classic	14888
Supreme	11987
Veggie	11649
Chicken	11050

• Kaushal\_Verma

Output:

# Determine the distribution of orders by hour of the day.

Solution :

Output:

```
3 •   SELECT
4       HOUR(order_time) AS hour, COUNT(order_id) AS order_count
5   FROM
6       orders
7   GROUP BY HOUR(order_time);
8
```

100% 26:7

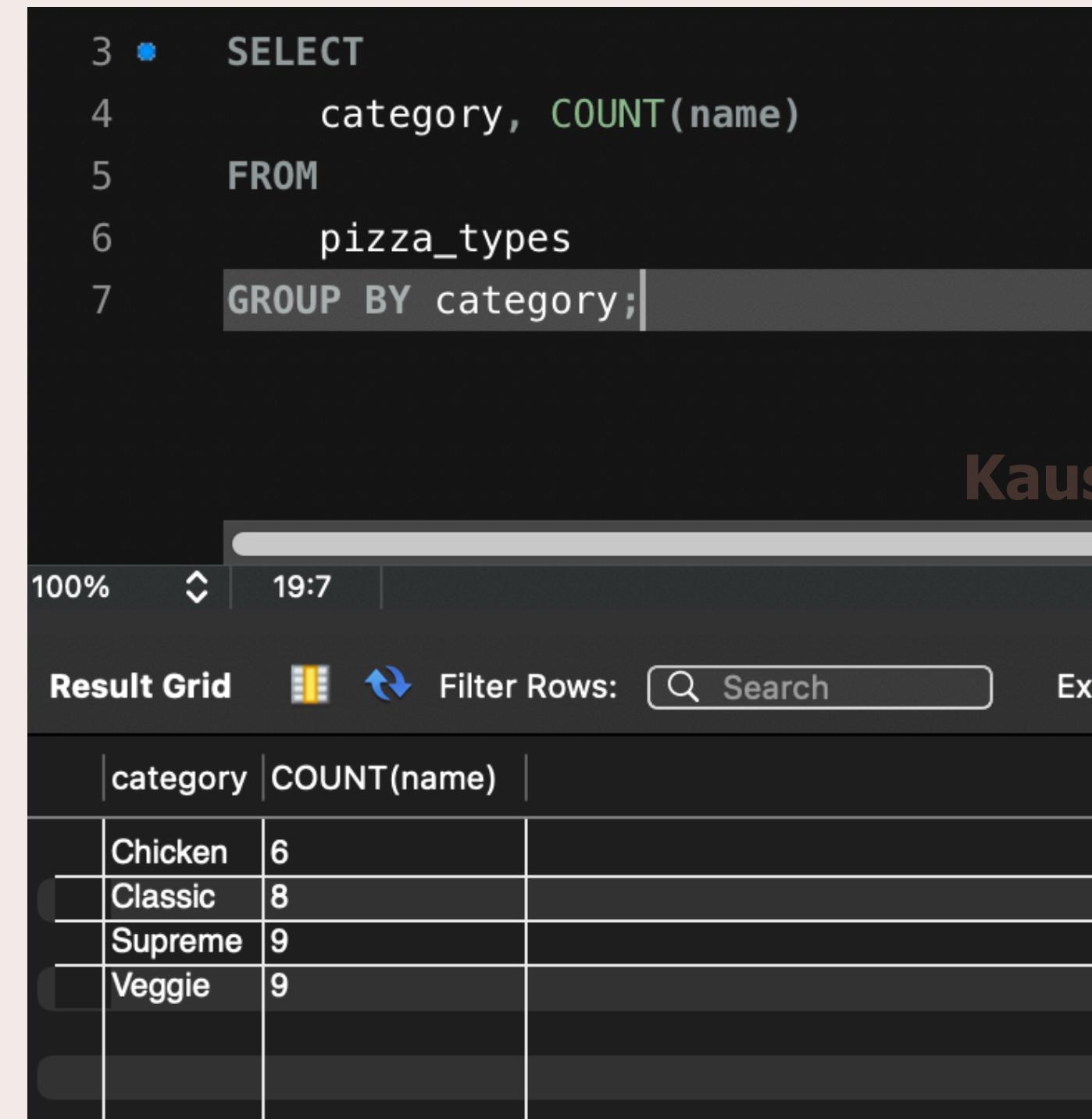
Result Grid Filter Rows: Search Export:

hour	order_count
11	1231
12	2520
13	2455
14	1472
15	1468
16	1920
17	2336
18	2399
19	2009
20	1642
21	1198
22	663
23	28
10	8
9	1

Kaushal\_Verma

Join relevant tables to find the category-wise distribution of pizzas.

Solution :



A screenshot of a SQL editor interface. The code area at the top shows a query:

```
3 •   SELECT
4       category, COUNT(name)
5   FROM
6       pizza_types
7   GROUP BY category;
```

The interface includes a watermark "Kaushal\_Verma" in the center. Below the code, there are controls for zoom (100%) and time (19:7). At the bottom, there are buttons for "Result Grid" and "Filter Rows:", and a search bar. The results grid below the controls displays the following data:

category	COUNT(name)
Chicken	6
Classic	8
Supreme	9
Veggie	9

Output:

# Group the orders by date and calculate the average number of pizzas ordered per day.

Solution :

```
3 •   SELECT
4     ROUND(AVG(quantity), 0) as avg_pizzas_ordered_per_day
5   FROM
6   (SELECT
7     orders.order_date, SUM(order_details.quantity) AS quantity
8   FROM
9     orders
10  JOIN order_details ON orders.order_id = order_details.order_id
11  GROUP BY orders.order_date) AS order_quantity;
12
```

00% 58:4

Result Grid Filter Rows: Search Export:

avg_pizzas_ordered_per_day
138

Kaushal\_Verma

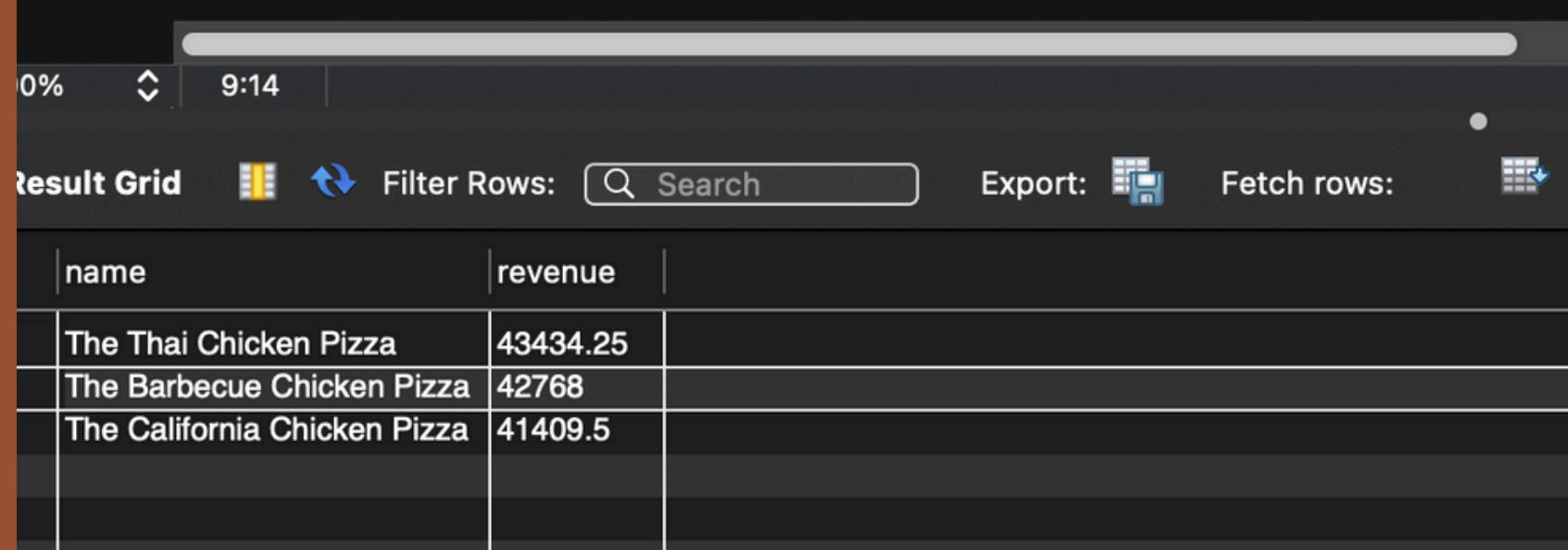
Output:

# Determine the top 3 most ordered pizza types based on revenue.

Solution :

```
3 •   SELECT
4     pizza_types.name,
5     SUM(order_details.quantity * pizzas.price) AS revenue
6   FROM
7     pizza_types
8       JOIN
9     pizzas ON pizzas.pizza_type_id = pizza_types.pizza_type_id
10      JOIN
11    order_details ON order_details.pizza_id = pizzas.pizza_id
12  GROUP BY pizza_types.name
13 ORDER BY revenue DESC
14 LIMIT 3;
```

Kaushal\_Verma



The screenshot shows a MySQL command-line interface window. At the top, there is a code editor containing the SQL query. Below the code editor is a toolbar with various icons. The main area displays the results of the query in a table format. The table has two columns: 'name' and 'revenue'. The results show three rows of data: 'The Thai Chicken Pizza' with a revenue of 43434.25, 'The Barbecue Chicken Pizza' with a revenue of 42768, and 'The California Chicken Pizza' with a revenue of 41409.5. The table has a dark header row and light gray body rows.

name	revenue
The Thai Chicken Pizza	43434.25
The Barbecue Chicken Pizza	42768
The California Chicken Pizza	41409.5

Output:

# Calculate the percentage contribution of each pizza type to total revenue.

Solution :

```
3 *   SELECT
4     pizza_types.category,
5     ROUND(SUM(order_details.quantity*pizzas.price) / (SELECT
6       ROUND(SUM(order_details.quantity*pizzas.price),
7         2) AS total_sales
8
9     FROM
10    order_details
11    JOIN
12      pizzas ON pizzas.pizza_id = order_details.pizza_id) * 100,
13      2) AS revenue
14
15   FROM
16     pizza_types
17     JOIN
18       pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
19     JOIN
20       order_details ON order_details.pizza_id = pizzas.pizza_id
21   GROUP BY pizza_types.category
22   ORDER BY revenue DESC;
```

Kaushal\_Verma

00% 24:20

Result Grid Filter Rows: Search Export:

	category	revenue
	Classic	26.91
	Supreme	25.46
	Chicken	23.96
	Veggie	23.68

Output:

# Analyze the cumulative revenue generated over time.

Solution :

```
3 •   select order_date,
4     sum(revenue) over(order by order_date) as cum_revenue
5   from
6   (select orders.order_date, sum(order_details.quantity * pizzas.price) as revenue
7    from order_details join pizzas
8    on order_details.pizza_id = pizzas.pizza_id
9    join orders
10   on orders.order_id = order_details.order_id
11   group by orders.order_date) as sales;
```

0% 37:11

Result Grid Filter Rows: Search Export:

	order_date	cum_revenue
	2015-01-01	2713.8500000000004
	2015-01-02	5445.75
	2015-01-03	8108.15
	2015-01-04	9863.6
	2015-01-05	11929.55
	2015-01-06	14358.5
	2015-01-07	16560.7
	2015-01-08	19399.05
	2015-01-09	21526.4
	2015-01-10	23990.350000000002
	2015-01-11	25862.65
	2015-01-12	27781.7
	2015-01-13	29831.300000000003
	2015-01-14	32358.700000000004
	2015-01-15	34343.50000000001
	2015-01-16	36937.650000000001

Output:

Kaushal\_Verma

# Determine the top 3 most ordered pizza types based on revenue for each pizza category.

Solution :

```
3 •  select name, revenue from
4   (select category, name, revenue,
5    rank() over(partition by category order by revenue desc) as rn
6    from
7   (select pizza_types.category, pizza_types.name,
8    sum((order_details.quantity) * pizzas.price) as revenue
9    from pizza_types join pizzas
10   on pizza_types.pizza_type_id = pizzas.pizza_type_id
11   join order_details
12   on order_details.pizza_id = pizzas.pizza_id
13   group by pizza_types.category, pizza_types.name) as a) as b
14  where rn <=3;
```

14:14

Result Grid Filter Rows: Search Export:

• Kaushal\_Verma

name	revenue
The Thai Chicken Pizza	43434.25
The Barbecue Chicken Pizza	42768
The California Chicken Pizza	41409.5
The Classic Deluxe Pizza	38180.5
The Hawaiian Pizza	32273.25
The Pepperoni Pizza	30161.75
The Spicy Italian Pizza	34831.25
The Italian Supreme Pizza	33476.75
The Sicilian Pizza	30940.5
The Four Cheese Pizza	32265.70000000065
The Mexicana Pizza	26780.75
The Five Cheese Pizza	26066.5

Output:

THANK YOU SO MUCH  
FOR SPENDING  
SOME TIME.

Kaushal\_Verma