

Delicious Pizza for Everyone!

PIZZA HOUSE



Hello!

We are a pizza house serving delicious pizza via the food truck way. we are ready to go around to deliver and serve pizza for you lovers!

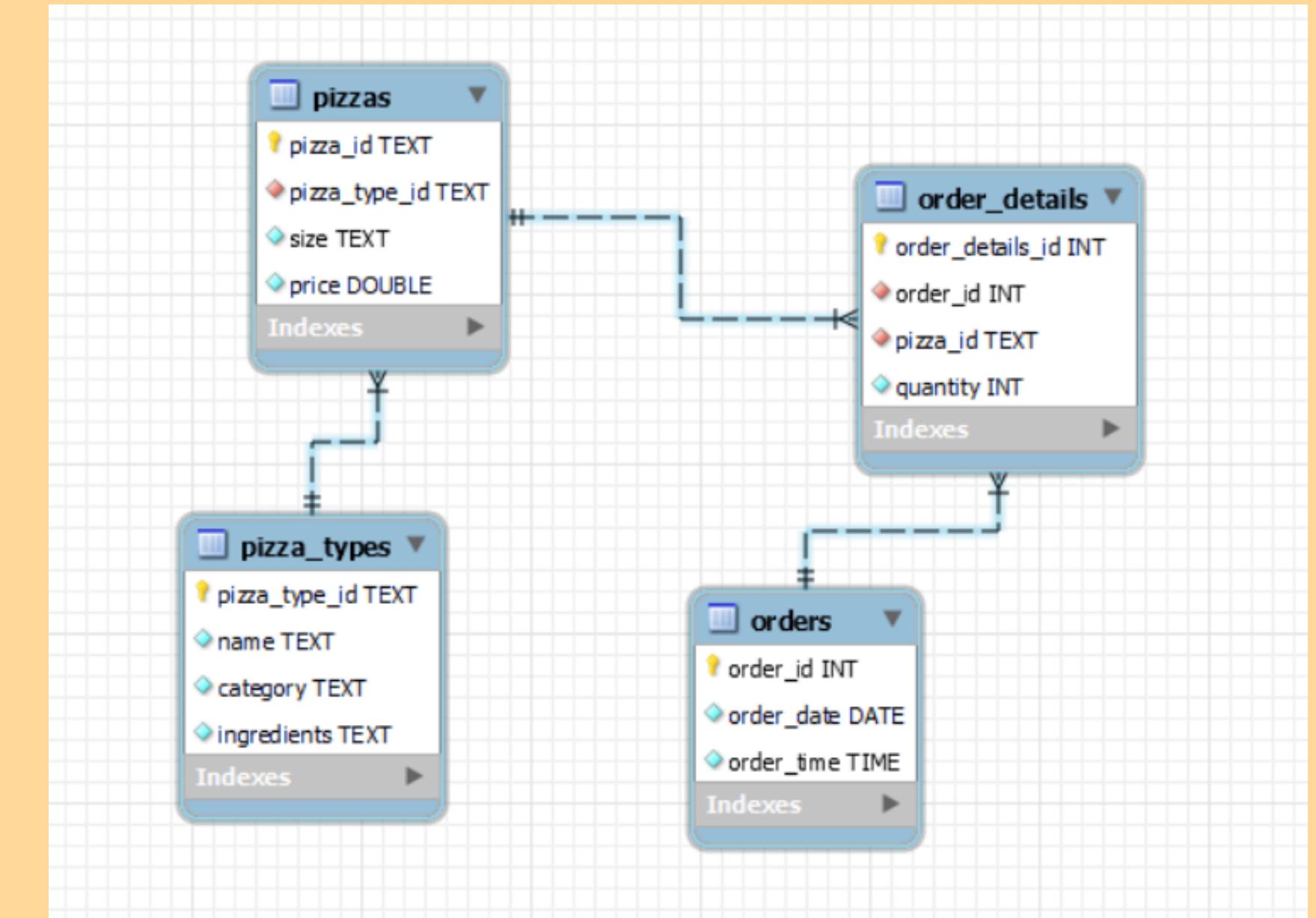




Chef Nail Train

hi, i am the main chef and owner of adri
pizza house. pizza house is a family-
owned secret recipe pizza that has been
passed down for generations with
incredible delicacy!

Table schema



Q1. Retrieve the total number of orders placed.

```
select count(order_id) as total_order from orders;
```

| Result Grid | |
|-------------|-------------|
| | total_order |
| ▶ | 21350 |

Q2.Calculate the total revenue generated from pizza sales.

The screenshot shows a MySQL Workbench interface with a query editor and a result grid. The query is:

```
1 select round((sum(order_details.quantity * pizzas.price)),2)as total_sales
2 from order_details join pizzas
3 on pizzas.pizza_id= order_details.pizza_id
4
```

The result grid displays one row with the column 'total_sales' containing the value '817860.05'.

| total_sales |
|-------------|
| 817860.05 |

Q3. Identify the highest-priced pizza.

The screenshot shows a MySQL Workbench interface with the following details:

- Tab Bar:** acme_pos_transactions (selected), acmeitem, acmeregion, acmeregister, acmestore
- Toolbar:** Includes icons for file operations, search, and connection management.
- Query Editor:** Contains the following SQL code:

```
1 • select pizza_types.name,pizzas.price from pizza_types join pizzas  
2   on pizza_types.pizza_type_id =pizzas.pizza_type_id  
3   order by pizzas.price desc limit 1;  
4
```
- Result Grid:** Shows the results of the query in a tabular format.

| name | price |
|-----------------|-------|
| The Greek Pizza | 35.95 |
- Bottom Bar:** Includes buttons for Result Grid, Filter Rows, Export, Wrap Cell Content, and Fetch rows.

Q4. Identify the most common pizza size ordered.

The screenshot shows a MySQL Workbench interface. The top pane contains the following SQL query:

```
1 • select pizzas.size, count(order_details.order_details_id) as order_count
2   from pizzas join order_details on
3     pizzas.pizza_id = order_details.pizza_id
4   group by pizzas.size order by order_count desc;
5
```

The bottom pane displays the results in a "Result Grid" table:

| | size | order_count |
|---|------|-------------|
| ▶ | L | 18526 |
| | M | 15385 |
| | S | 14137 |
| | XL | 544 |
| | XXL | 28 |

Q5.List the top 5 most ordered pizza types along with their quantities.

The screenshot shows a MySQL query editor interface. At the top, there is a toolbar with various icons. Below the toolbar, the SQL query is displayed:

```
1 • select pizza_types.name ,sum(order_details.quantity) as quantity
2   from pizza_types join pizzas
3     on pizza_types.pizza_type_id=pizzas.pizza_type_id
4   join order_details
5     on order_details.pizza_id=pizzas.pizza_id
6   group by pizza_types.name order by quantity desc limit 5;
7
```

Below the query, there is a "Result Grid" section. It contains a table with two columns: "name" and "quantity". The data is as follows:

| | 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 |

Q6. Join the necessary tables to find the total quantity of each pizza category ordered.

The screenshot shows a MySQL query editor window. At the top, there is a toolbar with various icons. Below the toolbar, the SQL query is displayed:

```
1 • select pizza_types.category,sum(order_details.quantity) as quantity
2   from pizza_types join pizzas
3     on pizza_types.pizza_type_id = pizzas.pizza_type_id
4   join order_details
5     on order_details.pizza_id = pizzas.pizza_id
6   group by pizza_types.category order by quantity desc;
```

Below the query, the result grid is shown:

| category | quantity |
|----------|----------|
| Classic | 14888 |
| Supreme | 11987 |
| Veggie | 11649 |
| Chicken | 11050 |

Q7.Determine the distribution of orders by hour of the day.

The screenshot shows a MySQL Workbench interface with the following details:

Query Editor:

```
1 select hour(order_time),count(order_id) from orders
2 group by hour(order_time);
3
```

Result Grid:

| | hour(order_time) | count(order_id) |
|---|------------------|-----------------|
| ▶ | 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 |

Q8.Join relevant tables to find the category-wise distribution of pizzas.

The screenshot shows a MySQL Workbench interface with the following details:

- Tables:** acme_pos_transactions (selected), acmeitem, acmeregion, acmeregister, acmestore
- Toolbar:** Includes icons for file operations, search, and other database functions.
- Query Editor:** A SQL query is entered:

```
1 • select category ,count(name) from pizza_types group by category;
```
- Result Grid:** The query results are displayed in a grid format:

| category | count(name) |
|----------|-------------|
| Chicken | 6 |
| Classic | 8 |
| Supreme | 9 |
| Veggie | 9 |

Q9. Group the orders by date and calculate the average number of pizzas ordered per day.

The screenshot shows a MySQL Workbench interface with the following details:

- Query Editor:** The tab "acme_pos_transactions" is selected. The query is:1 • select round(avg(quantity),0) from (select orders.order_date,sum(order_details.quantity)as quantity
2 from orders join order_details
3 on orders.order_id =order_details.order_id
4 group by orders.order_date) as order_quantity;
5
- Result Grid:** The result of the query is displayed in a grid:

| round(avg(quantity),0) |
|------------------------|
| 138 |

Q10.Determine the top 3 most ordered pizza types based on revenue.

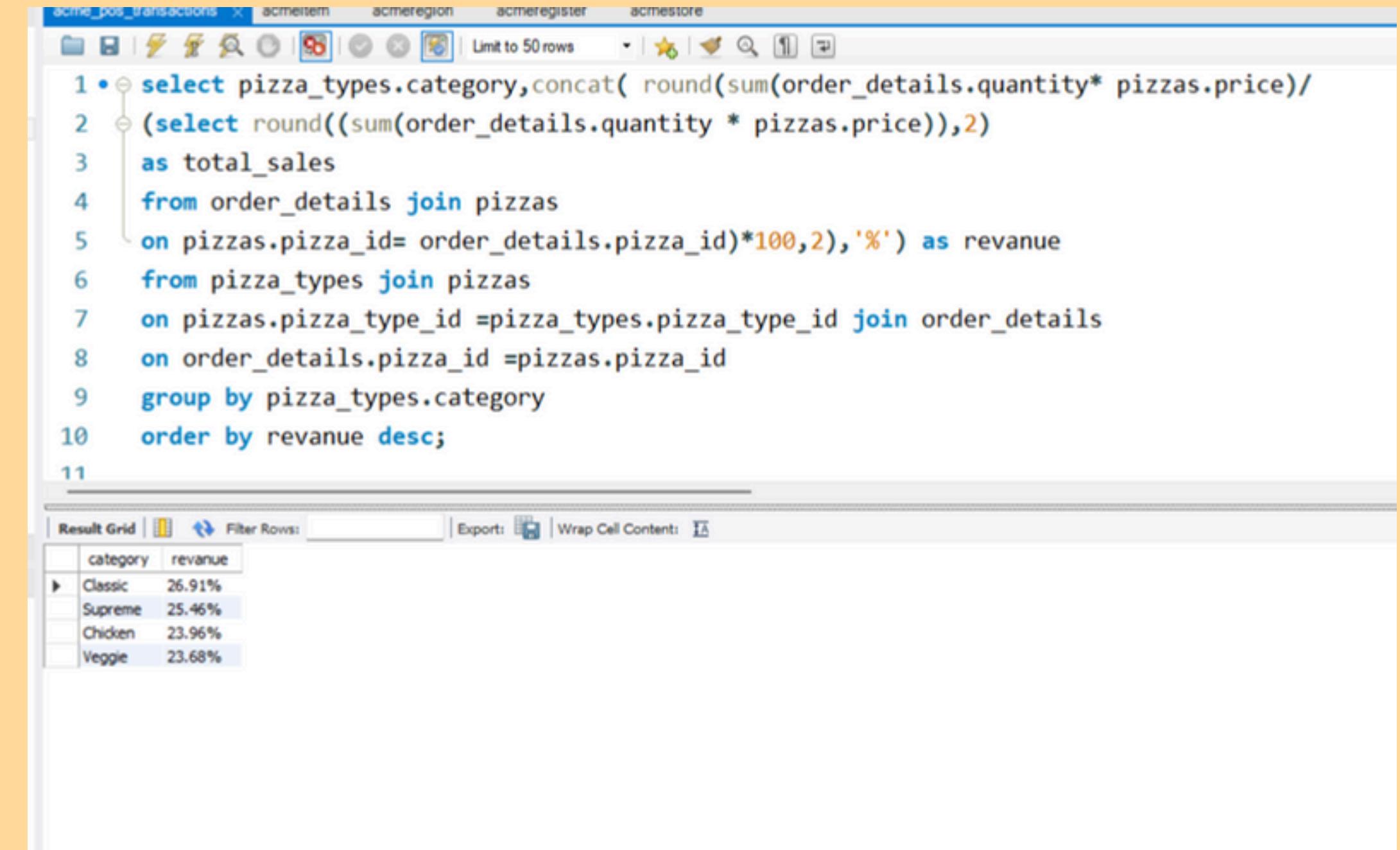
The screenshot shows a MySQL query editor window with the following content:

```
1 • select pizza_types.name ,sum(order_details.quantity * pizzas.price) as revenue
2   from pizza_types join pizzas
3     on pizza_types.pizza_type_id = pizzas.pizza_type_id
4   join order_details
5     on order_details.pizza_id =pizzas.pizza_id
6   group by pizza_types.name order by revenue desc limit 3;
7
```

Below the code, the Result Grid displays the following data:

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

Q11. Calculate the percentage contribution of each pizza type to total revenue.



The screenshot shows a MySQL query editor window with the following content:

```
acme_pos_transactions x acmestoreitem acmeregion acmeregister acmestore
1 • select pizza_types.category,concat( round(sum(order_details.quantity*pizzas.price)/
2   (select round((sum(order_details.quantity * pizzas.price)),2)
3     as total_sales
4   from order_details join pizzas
5   on pizzas.pizza_id= order_details.pizza_id)*100,2),'%') as revanue
6   from pizza_types join pizzas
7   on pizzas.pizza_type_id =pizza_types.pizza_type_id join order_details
8   on order_details.pizza_id =pizzas.pizza_id
9   group by pizza_types.category
10  order by revanue desc;
11
```

Result Grid | Filter Rows: _____ | Export: | Wrap Cell Content: |

| category | revanue |
|----------|---------|
| Classic | 26.91% |
| Supreme | 25.46% |
| Chicken | 23.96% |
| Veggie | 23.68% |

Q12. Analyze the cumulative revenue generated over time.

```
1 • select order_date,sum(revenue) over (order by order_date ) as cumulative_revenue from
2   (select orders.order_date ,sum(order_details.quantity * pizzas.price) as revenue from
3    order_details join pizzas on order_details.pizza_id =pizzas.pizza_id join orders
4    on orders.order_id = order_details.order_id group by orders.order_date) as sales;
5
```

| order_date | cumulative_revenue |
|------------|--------------------|
| 2015-01-01 | 2713.850000000004 |
| 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 | 23990.35000000002 |
| 2015-01-11 | 25862.65 |
| 2015-01-12 | 27781.7 |
| 2015-01-13 | 29831.30000000003 |
| 2015-01-14 | 32358.70000000004 |
| 2015-01-15 | 34343.50000000001 |
| 2015-01-16 | 36937.65000000001 |
| 2015-01-17 | 39001.75000000001 |
| 2015-01-18 | 40978.60000000006 |
| 2015-01-19 | 43365.75000000001 |
| 2015-01-20 | 45763.65000000001 |

Q13.Determine the top 3 most ordered pizza types based on revenue for each pizza category.

The screenshot shows a MySQL Workbench interface with a query editor and a result grid.

Query Editor:

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

Result Grid:

| name | revenue |
|------------------------------|-------------------|
| The Thai Chicken Pizza | 42434.25 |
| 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 |

Our New Taste



SPICY CHEESE PIZZA

Our new variant this time is cheese. for you cheese lovers, you can enjoy warm cheese in abundance with pizza bread plus chili flakes on top.

Where we are?

we have 15 trucks that will drive around the housing and parks.



So don't worry because you will always find us in your city.

Order from home

Currently pizza boxcar can also be ordered through the application and the phone. you just need to pick and order from home, we will deliver your order and make sure it stays warm.



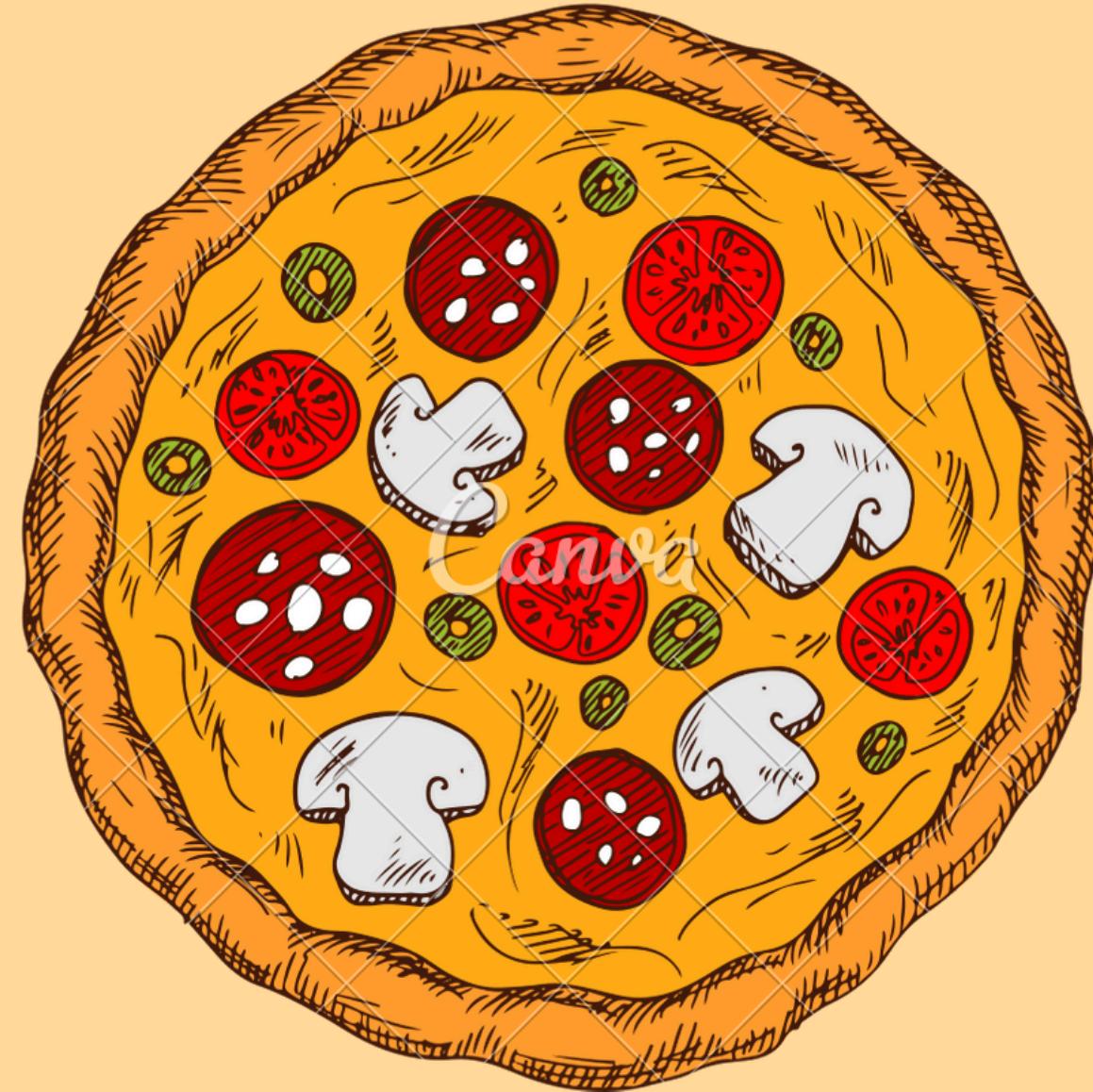
Call us for Order



Phone Number:
+123-456-7890 / +123-456-7890

social media:
@pizza_hub

Website:
www.pizzahub.com



Pizza Hub Present

**THANK
YOU**