

SQL Project

Pizza Sales

MySQL®

Who I Am

Hi, I'm Nithin, an aspiring data analyst with a strong interest in turning data into actionable insights. I have hands-on experience working with SQL, including a project where I analyzed pizza sales data to uncover trends, optimize performance, and support data-driven decision-making. I'm eager to continue building my analytical skills and contribute to real-world business problems.



I have worked on SQL questions across three levels: basic, intermediate, and advanced.

Basic:

- Retrieve the total number of orders placed.
- Calculate the total revenue generated from pizza sales.
- Identify the highest-priced pizza.
- Identify the most common pizza size ordered.
- List the top 5 most ordered pizza types along with their quantities.

Intermediate:

- Join the necessary tables to find the total quantity of each pizza category ordered.
- Determine the distribution of orders by hour of the day.
- Join relevant tables to find the category-wise distribution of pizzas.
- Group the orders by date and calculate the average number of pizzas ordered per day.
- Determine the top 3 most ordered pizza types based on revenue.

Advanced:

- Calculate the percentage contribution of each pizza type to total revenue.
- Analyze the cumulative revenue generated over time.
- Determine the top 3 most ordered pizza types based on revenue for each pizza category.

Q1. Retrieve the total number of orders placed.

The screenshot shows the MySQL Workbench interface. The SQL editor window contains the following code:

```
1 -- Retrieve the total number of orders placed.  
2  
3 • Select count(order_id) as total_orders from orders;
```

The results are displayed in the Result Grid:

| total_orders |
|--------------|
| 21350 |

The interface includes various toolbars and panels, and a status bar at the bottom indicating "Result 1" and "Output". A vertical toolbar on the right side shows icons for Result Grid, Form Editor, and Read Only mode, with "Result Grid" currently selected.

Q2. Calculate the total revenue generated from pizza sales?

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

- Query Editor:** Displays the SQL query:

```
1 -- calculate the total revenue generated from pizza sales
2
3 • SELECT
4     ROUND(SUM(order_details.quantity * pizzas.price),
5           2) AS total_sales
6
7 FROM
8     order_details
9     JOIN
10    pizzas ON pizzas.pizza_id = order_details.pizza_id;
```
- Result Grid:** Shows the output of the query:

| total_sales |
|-------------|
| 817860.05 |
- Toolbar:** Includes standard database management icons for file operations, search, and refresh.
- Status Bar:** Shows "Limit to 10000 rows".
- Right Panel:** Contains a vertical toolbar with icons for "Result Grid" (selected), "Form Editor", and "Read Only".

Q3. Identify the highest priced pizza?

The screenshot shows the MySQL Workbench interface. At the top, there's a toolbar with various icons. Below it is a query editor window containing the following SQL code:

```
1 -- identify the highest - priced pizza.
2
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;
```

Below the query editor is a results grid titled "Result Grid". It displays one row of data:

| | name | price |
|---|-----------------|-------|
| ▶ | The Greek Pizza | 35.95 |

At the bottom right of the results grid, there's a sidebar with icons for "Result Grid", "Form Editor", and "Read Only".

Q4. Identify the most common pizza ordered?

The screenshot shows a MySQL query editor interface with the following details:

- Toolbar:** Includes icons for file operations, search, and refresh.
- Query Editor:** Displays the following SQL code:

```
1 -- identify the most common pizza size ordered
2
3 • SELECT
4     pizzas.size,
5     COUNT(order_details.order_details_id) AS order_count
6 FROM
7     pizzas
8     JOIN
9         order_details ON pizzas.pizza_id = order_details.pizza_id
10 GROUP BY pizzas.size
11 ORDER BY order_count DESC;
```
- Result Grid:** Shows the output of the query in a tabular format.

| size | order_count |
|------|-------------|
| L | 18526 |
| M | 15385 |
| S | 14137 |
| XL | 544 |
| XXL | 28 |
- Right Panel:** Contains a sidebar with icons for "Result Grid" (selected), "Form Editor", and "Read Only".

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

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

Query Editor:

```
1 -- list the top 5 most ordered pizza types along with their quantities
2
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;
```

Results Grid:

| 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 |

Toolbar: The toolbar includes icons for file operations, search, and refresh, followed by a "Limit to 10000 rows" dropdown and various status indicators.

Status Bar: The status bar at the bottom right indicates "Result 1" and "Read Only".

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

```
1  -- join the necessary table to find the total quantity of each pizza category ordered.  
2  
3 • SELECT  
4      pizza_types.category,  
5      SUM(order_details.quantity) AS quantity  
6  FROM  
7      pizza_types  
8      JOIN  
9      pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id  
10     JOIN  
11     order_details ON order_details.pizza_id = pizzas.pizza_id  
12  GROUP BY pizza_types.category  
13  ORDER BY quantity DESC;
```

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

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

Result 1 x

Output: _____

Result Grid | Form Editor | Read Only

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

The screenshot shows a database interface with a query editor and a results grid. The query is:

```
1 -- determine the distribution of orders by hour of the day
2
3 • SELECT
4     HOUR(order_time) AS hour, COUNT(order_id) AS order_count
5 FROM
6     orders
7 GROUP BY HOUR(order_time);
```

The results grid displays the following data:

| 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 |

Below the results grid, there is a message "Result 1" and an "Output" section. On the right side of the interface, there is a vertical toolbar with icons for Result Grid, Form Editor, Field Types, and Query Stats. A blue arrow points from the "Result Grid" icon on the toolbar to the "Result Grid" icon in the top bar. A "Read Only" status indicator is also present.

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

The screenshot shows a database interface with a query editor and a result grid. The query editor contains the following SQL code:

```
1 -- join relevant tables to find the category wise distribution of pizza
2
3 • select category, count(name) from pizza_types
4 group by category;
```

The result grid displays the following data:

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

The interface includes various toolbar icons for file operations, search, and export. On the right side, there is a vertical sidebar with icons for Result Grid, Form Editor, Field Types, and Query Stats. A large blue arrow points downwards towards the result grid. At the bottom right, there is a circular icon with a large letter 'R' and a 'Read Only' label.

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

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

Query Editor:

```
1 -- group the orders by date and calculate the average number of pizzas ordered per day
2
3 • SELECT
4     ROUND(AVG(quantity), 0)
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;
```

Results Grid:

| ROUND(AVG(quantity), 0) |
|-------------------------|
| 138 |

Toolbar: The toolbar includes icons for file operations, search, and refresh, along with a "Limit to 10000 rows" dropdown.

Status Bar: Shows "Result 1" and "Output".

Right Panel: A sidebar with tabs for "Result Grid" (selected), "Form Editor", and "Field Types".

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

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

Query Editor:

```
1 -- determine the top 3 most ordered pizza based on revenue
2
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;
```

Result Grid:

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

Output: Result 1

Right Panel:

- Result Grid (selected)
- Form Editor
- Read Only

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

The screenshot shows a MySQL query editor interface. The top section displays the SQL query:

```
-- calculate the percentage contribution of each pizza type to total revenue
SELECT
    pizza_types.category,
    ROUND(SUM(order_details.quantity * pizzas.price) / (SELECT
        ROUND(SUM(order_details.quantity * pizzas.price),
        2) AS total_sales
    FROM
        order_details
    JOIN
        pizzas ON pizzas.pizza_id = order_details.pizza_id) * 100,
    2) AS revenue
FROM
    pizza_types
JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
JOIN
    order_details ON order_details.pizza_id = pizzas.pizza_id
GROUP BY pizza_types.category
ORDER BY revenue DESC;
```

The bottom section shows the results in a "Result Grid":

| category | revenue |
|----------|---------|
| Classic | 26.91 |
| Supreme | 25.46 |
| Chicken | 23.96 |
| Veggie | 23.68 |

Additional UI elements include a "Read Only" status indicator and a large decorative graphic featuring the letters "SQL" and "R".

Q12. Analyze the cumulative revenue generated over time.

The screenshot shows a database interface with a query editor and a results grid. The query is as follows:

```
1 -- analyze the cumulative revenue generated over time
2
3 • select order_date,
4     sum(revenue) over(order by order_date) as cum_revenue
5   from
6   (select orders.order_date,
7     sum(order_details.quantity * pizzas.price) as revenue
8   from order_details join pizzas
9     on order_details.pizza_id = pizzas.pizza_id
10  join orders
11    on orders.order_id = order_details.order_id
12  group by orders.order_date) as sales;
```

The results grid displays the following data:

| 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 |

The interface includes various toolbar icons and a sidebar with options like 'Result Grid' and 'Form Editor'.

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

```
1  -- Determine the top 3 most ordered pizza types based on revenue for each pizza category.
2
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;
```

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

| | 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 Margherita Pizza | 26066.5 |

Result 1 x

Output

Result Grid | Form Editor | Field Types | Read Only

The End

Thank You

MugSQL®