PIZZA SALES PERFORMANCE ANALYSIS

Strategic Data Insights using Advanced MySQL

🎯 Executive Summary: Unlocking Business Value

This document details a comprehensive data analysis project focused on transforming raw transactional data from a fictional pizza chain into actionable business intelligence. The core objective was to move beyond simple reporting to uncover specific, high-value insights that can drive revenue growth and operational efficiency through the masterful application of SQL.

💡 Project Goal

To design, query, and analyse a large-scale sales dataset using **Advanced SQL (MySQL)** to quantify business performance, model growth trends, and generate strategic recommendations for product and operations teams.

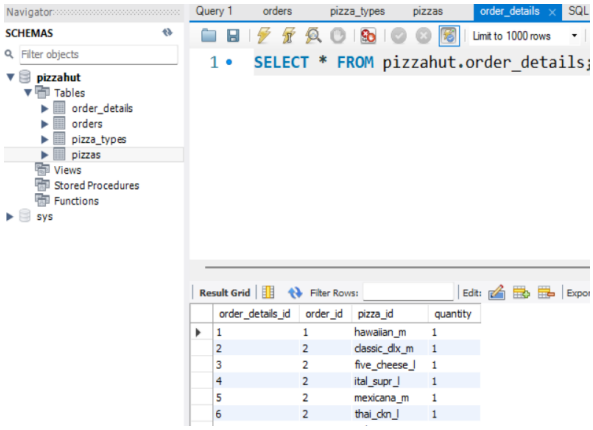
🚀 Key Results & Impact Snapshot

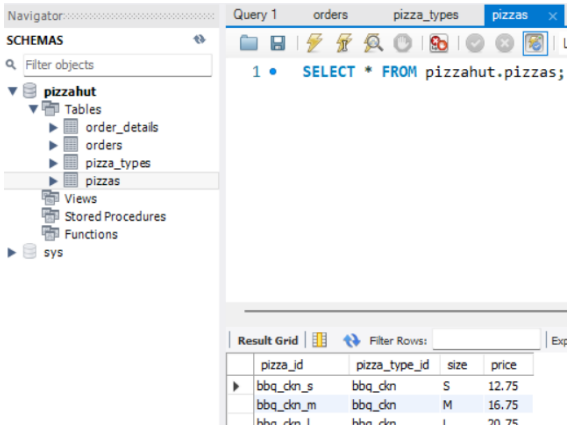
- **Revenue Growth Modelling:** Successfully analysed the **Cumulative Revenue trend** over time, providing crucial visualization for business forecasting.
- **Strategic Menu Focus:** Pinpointed the **Top 3 Revenue Drivers** within *every* pizza category, essential for inventory management and menu profitability optimization.
- **Operational Efficiency:** Mapped the **Hourly Order Distribution** to identify peak periods, directly informing precise staff scheduling and resource allocation.
- **Technical Mastery:** Verified expertise in **Advanced SQL techniques** including **Window Functions, CTEs, and Complex Joins** to solve challenging, real-world business questions.

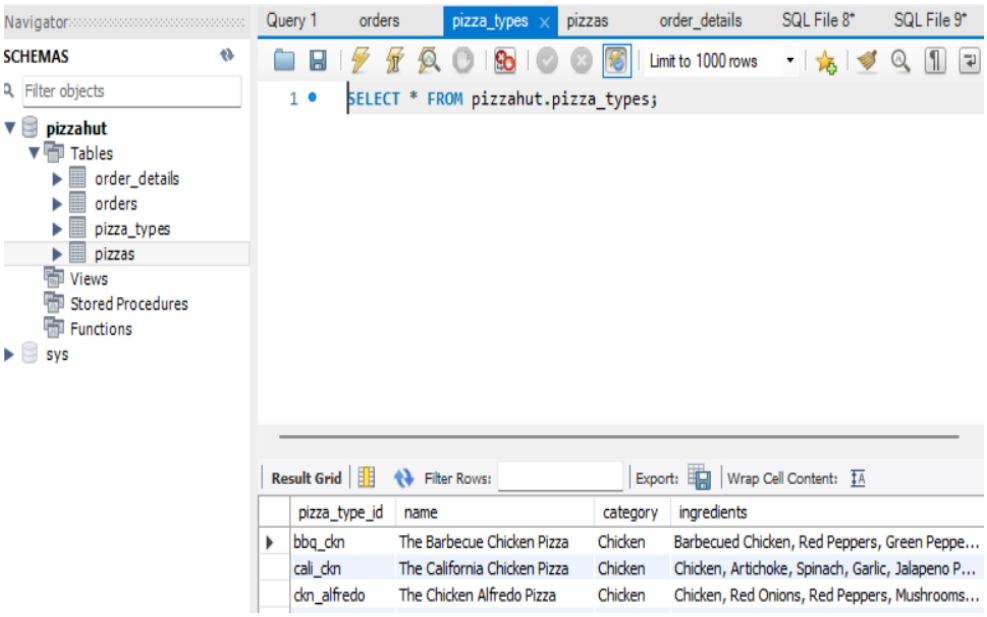
🛠️ Technical Environment

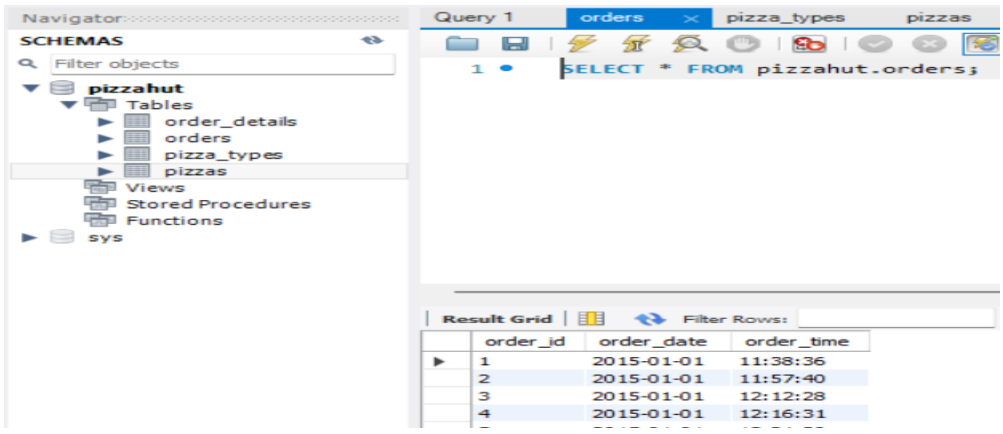
Category	Tool / Method Used
Database Platform	MySQL Workbench / MySQL Server
Core Techniques	Advanced SQL, Data Modelling, Time-Series Analysis

So, the dataset we used for this project has 4 tables in it followed as below.









Tasks Done in this project:

Basic:

1.Retrieve the total number of orders placed.

Code:

```
#Retrieve the total no.of orders placed
select count(order_id) as total_orders from orders;
```

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

total_orders
21350

2. Calculate the total revenue generated from pizza sales.

Code:

```
#calculate total revenue generated from pizza sales.
select round(sum(order_details.quantity*pizzas.price),2) as toatl_sales
From order_details join pizzas
On pizzas.pizza_id=order_details.pizza_id
```

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

toatl_sales
817860.05

3. Identify the highest-priced pizza.

Code:

```
1   #Identify the highest priced pizza.
2 • Select pizza_types.name,pizzas.price
3   from pizza_types Join pizzas
4   on pizza_types.pizza_type_id=pizzas.pizza_type_id
5   order by pizzas.price Desc LIMIT 1;
6
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

name	price
The Greek Pizza	35.95

4. Identify the most common pizza size ordered.

Code:

```
1   # Identify the most common pizza size ordered.
2 • SELECT
3       pizzas.size,
4       COUNT(order_details.order_details_id) AS order_count
5   FROM pizzas JOIN
6       order_details ON pizzas.pizza_id = order_details.pizza_id
7   GROUP BY pizzas.size
8   ORDER BY order_count DESC;
```

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

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

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

Code:

```
SELECT
    pizza_types.name,
    SUM(order_details.quantity) AS quantity -- Corrected spelling
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.name
ORDER BY quantity DESC -- Now correctly referencing the alias 'quantity'
LIMIT 5;
```

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

Intermediate Level:

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

Code:

```
SELECT
    pizza_types.category,
    SUM(order_details.quantity) AS quantity
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 quantity DESC;
```

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

category	quantity
Classic	14888
Supreme	11987
Veggie	11649
Chicken	11050

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

Code:

```
SELect hour(order_time) as hour,
count(order_id) as order_count
FROM orders
Group by hour(order_time);
```

hour	order_count
11	1231
12	2520
13	2455
14	1472
15	1468
16	1920

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

Code:

```
Select category,
count(name)
from pizza_types
group by category;
```

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

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

Code:

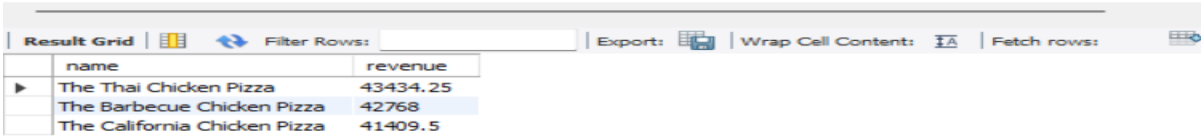
```
Select round(avg(quantity),0)
FROM (select orders.order_date,
SUM(order_details.quantity)as quantity
FROM orders
JOIN order_details
ON orders.order_id=order_details.order_id
GROUP BY orders.order_date)
as order_quantity;
```

round(avg(quantity),0)
138

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

Code:

```
SELECT
    pizza_types.name,
    SUM(order_details.quantity * pizzas.price) AS revenue
FROM pizza_types
JOIN  pizzas ON pizzas.pizza_type_id = pizza_types.pizza_type_id
JOIN  order_details ON order_details.pizza_id = pizzas.pizza_id
GROUP BY pizza_types.name
ORDER BY revenue DESC
LIMIT 3;
```



Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

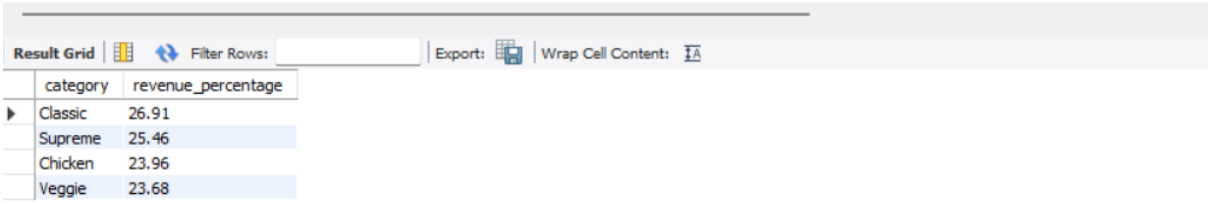
name	revenue
The Thai Chicken Pizza	43434.25
The Barbecue Chicken Pizza	42768
The California Chicken Pizza	41409.5

Advanced Level:

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

Code:

```
SELECT
    pizza_types.category,
    ROUND(SUM(order_details.quantity * pizzas.price) / (SELECT
                SUM(order_details.quantity * pizzas.price)
FROM order_details JOIN  pizzas ON pizzas.pizza_id = order_details.pizza_id) * 100,2) AS revenue_percentage
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_percentage DESC;
```



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

category	revenue_percentage
Classic	26.91
Supreme	25.46
Chicken	23.96
Veggie	23.68

2. Analyze the cumulative revenue generated over time.

Code:

```
Select order_date,SUM(revenue)
over(order by order_date) as cum_revenue
FROM (Select orders.order_date,
SUM(order_details.quantity*pizzas.price)as revenue
From order_details join pizzas on
order_details.pizza_id=pizzas.pizza_id
JOIn orders On
orders.order_id=order_details.order_id
GROUP BY orders.order_date) AS sales;
```

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

order_date	cum_revenue
2015-01-01	2713.8500000000004
2015-01-02	5445.75
2015-01-03	8108.15
2015-01-04	9863.6

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

Code:

```
Select name,revenue from
(Select category,name,revenue,rank() over(partition by category
order by revenue desc)as rn
FROM (Select pizza_types.category,pizza_types.name,
SUM((order_details.quantity)*pizzas.price)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,pizza_types.name)as a)as b
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