

Title	Pizzahut Sales Analysis using SQL
Prepared by	Krishna Sanjay Jadhav
Course	MBA III SEM – Business Analytics
University	Sanjivani University

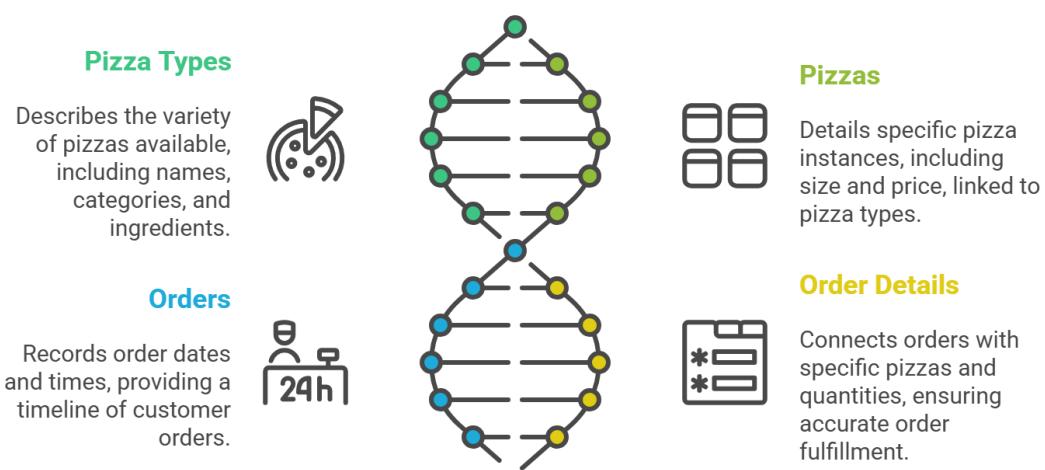
1. Introduction

This report presents insights derived from a pizzahut sales database using SQL. The queries aim to address business questions related to order volume, revenue, pizza types, and customer behavior patterns. The analysis is divided into basic, intermediate, and advanced levels to showcase progression in analytical complexity.

2. Dataset Overview

Table Name	Description
orders	Contains order IDs and timestamps.
order_details	Tracks items and quantity in each order.
pizzas	Contains pizza size, price, and IDs.
pizza_types	Describes each pizza's name, category, and ingredients.

Database Structure for Pizza Orders



3. Analytical Questions and Screenshots

A. Basic-Level Analysis

1. Total number of orders placed.

The screenshot shows the SQL Server Management Studio interface. The left pane displays the Navigator with the Schemas section expanded, showing the krishnadb and pizzahut schemas. The pizzahut schema contains tables like order_details, orders, pizza_types, and pizzas. The right pane shows a query window with the following SQL code:

```
1 -- Retrieve the total number of orders placed.  
2 • select count(order_id) from orders;
```

The results grid below the query shows a single row with the value 1163.

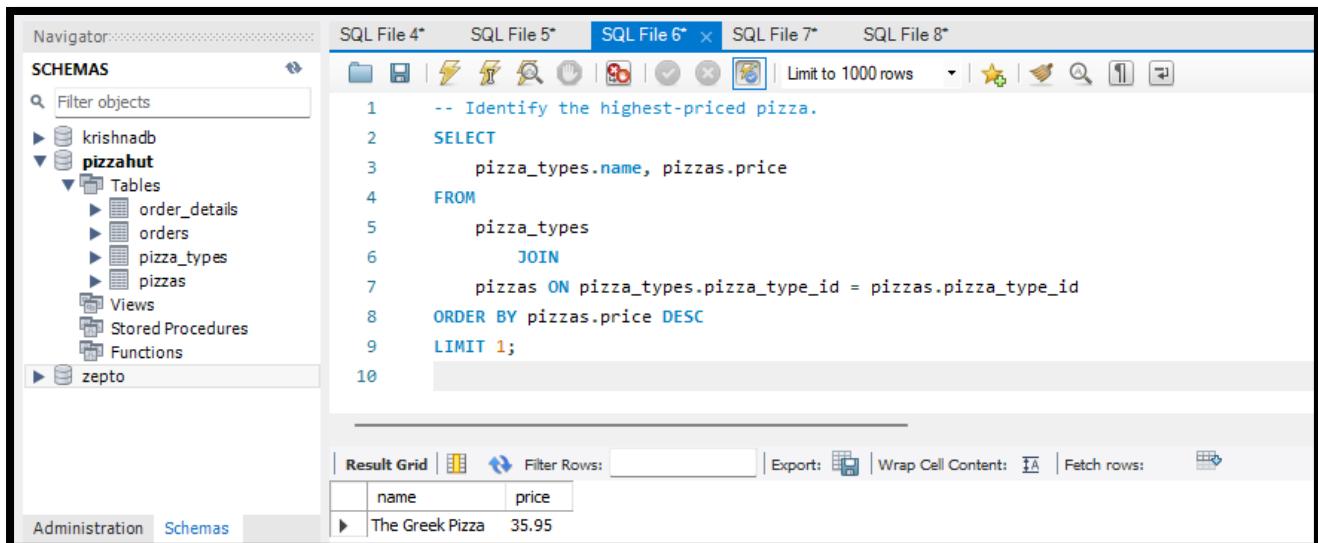
2. Total revenue generated from pizza sales.

The screenshot shows the SQL Server Management Studio interface. The left pane displays the Navigator with the Schemas section expanded, showing the krishnadb and pizzahut schemas. The pizzahut schema contains tables like order_details, orders, pizza_types, and pizzas. The right pane shows a query window with the following SQL code:

```
1 -- Calculate the total revenue generated from pizza sales.  
2 • select  
3 round(sum( order_details.quantity * Pizzas.price),2)as TOTAL_SALES  
4 from order_details join pizzas  
5 on pizzas.pizza_id= order_details.pizza_id
```

The results grid below the query shows a single row with the value 396540.2.

3. Highest-priced pizza.



The screenshot shows the SQL Workbench interface. The left pane displays the Navigator with the Schemas section expanded, showing the krishnadb and pizzahut schemas. The pizzahut schema contains tables like order_details, orders, pizza_types, pizzas, Views, Stored Procedures, and Functions. The right pane shows the SQL Editor with the following query:

```

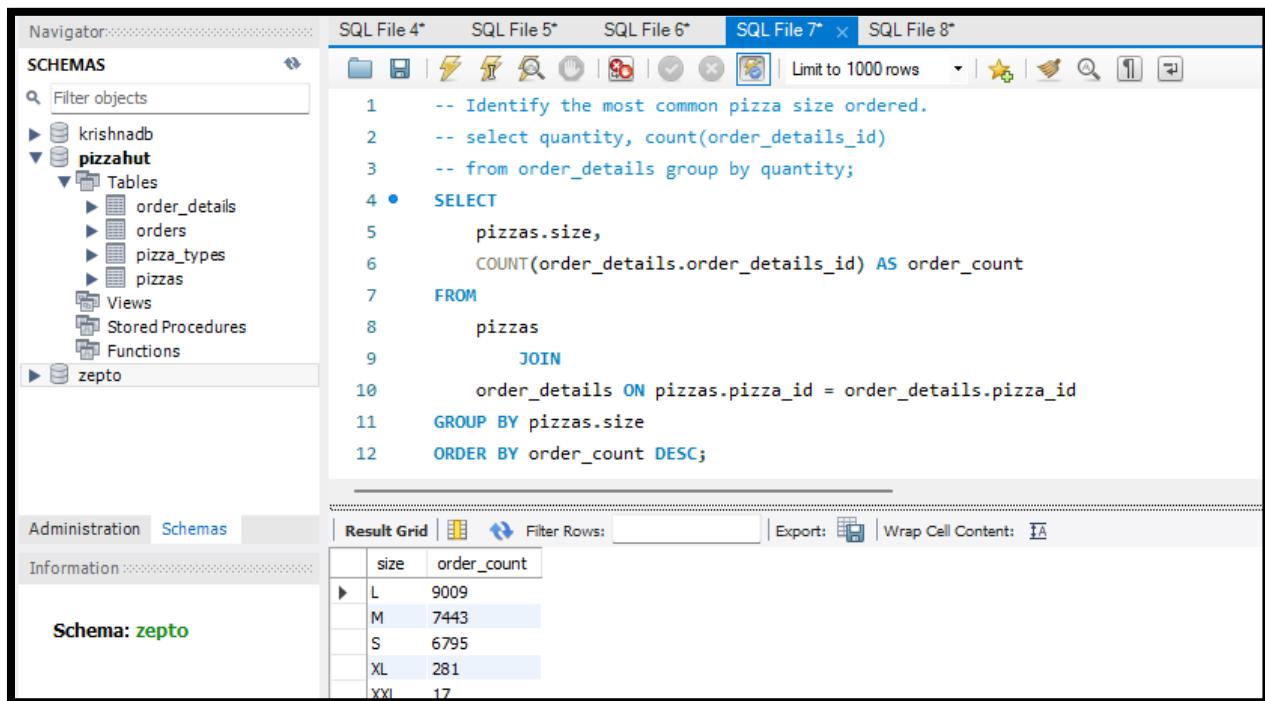
1 -- Identify the highest-priced pizza.
2 SELECT
3     pizza_types.name, pizzas.price
4 FROM
5     pizza_types
6     JOIN
7     pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
8 ORDER BY pizzas.price DESC
9 LIMIT 1;
10

```

Below the editor is the Result Grid, which displays a single row of data:

name	price
The Greek Pizza	35.95

4. Most common pizza size ordered.



The screenshot shows the SQL Workbench interface. The left pane displays the Navigator with the Schemas section expanded, showing the krishnadb and pizzahut schemas. The pizzahut schema contains tables like order_details, orders, pizza_types, pizzas, Views, Stored Procedures, and Functions. The right pane shows the SQL Editor with the following query:

```

1 -- Identify the most common pizza size ordered.
2 -- select quantity, count(order_details_id)
3 -- from order_details group by quantity;
4 • SELECT
5     pizzas.size,
6     COUNT(order_details.order_details_id) AS order_count
7 FROM
8     pizzas
9     JOIN
10    order_details ON pizzas.pizza_id = order_details.pizza_id
11 GROUP BY pizzas.size
12 ORDER BY order_count DESC;

```

Below the editor is the Result Grid, which displays the following data:

size	order_count
L	9009
M	7443
S	6795
XL	281
XXL	17

5. Top 3 most ordered pizza types with quantities.

The screenshot shows the MySQL Workbench interface. The left pane displays the Navigator with the SCHEMAS section open, showing databases krishnadb and pizzahut, and tables order_details, orders, pizza_types, and pizzas under pizzahut. The right pane contains the SQL Editor with the following query:

```
1 -- List the top 3 most ordered pizza types along with their quantities.
2 SELECT
3     pizza_types.name AS pizza_type,
4     SUM(order_details.quantity) AS total_quantity
5 FROM
6     order_details
7     JOIN
8     pizzas ON order_details.pizza_id = pizzas.pizza_id
9     JOIN
10    pizza_types ON pizzas.pizza_type_id = pizza_types.pizza_type_id
11 GROUP BY pizza_types.name
12 ORDER BY total_quantity DESC
13 LIMIT 3;
```

The Result Grid below the SQL editor shows the output:

pizza_type	total_quantity
The Barbecue Chicken Pizza	1211
The Hawaiian Pizza	1159
The Pepperoni Pizza	1157

B. Intermediate-Level Analysis

1. total quantity of each pizza category category ordered.

The screenshot shows the SQL Server Management Studio interface. The left pane displays the Navigator and Schemas. The right pane shows the SQL Editor with the following query:

```

1 -- total quantity of each pizza category category ordered.
2 • SELECT
3     pizza_types.category,
4     SUM(order_details.quantity) AS total_quantity
5     FROM order_details
6     JOIN pizzas ON order_details.pizza_id = pizzas.pizza_id
7     JOIN pizza_types ON pizzas.pizza_type_id = pizza_types.pizza_type_id
8     GROUP BY pizza_types.category
9     ORDER BY total_quantity DESC;
10

```

The Result Grid below the query shows the following data:

category	total_quantity
Classic	7127
Supreme	5842
Veggie	5718
Chicken	5313

2. Order distribution by hour of day

The screenshot shows the SQL Server Management Studio interface. The left pane displays the Navigator and Schemas. The right pane shows the SQL Editor with the following query:

```

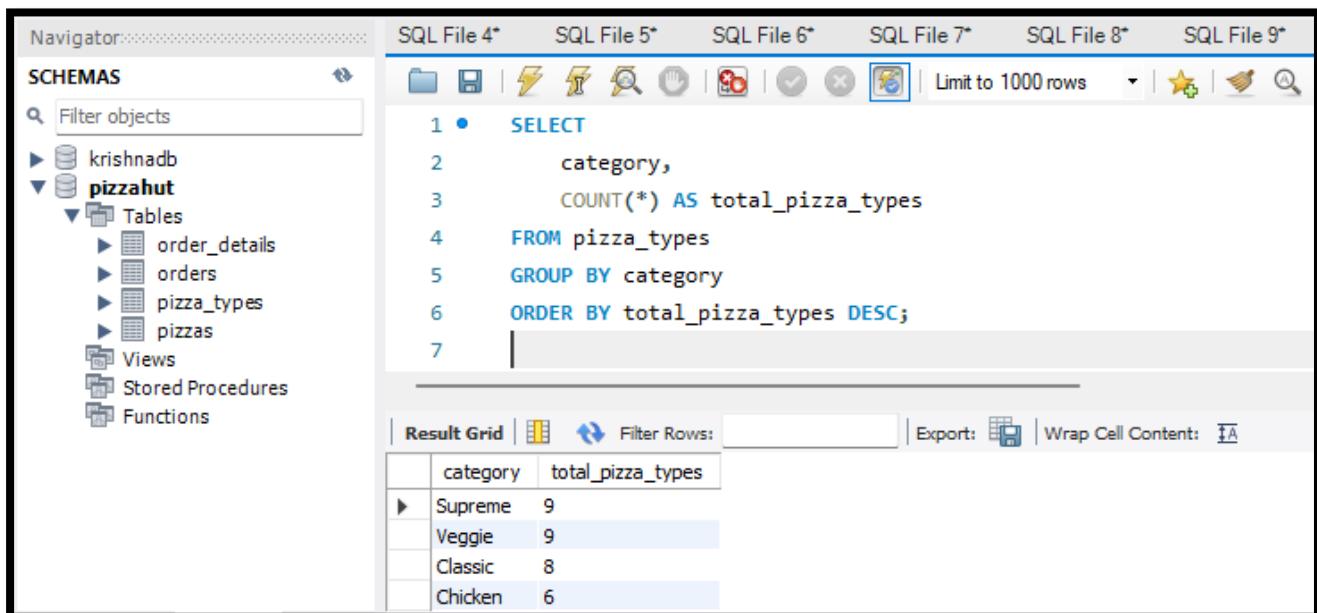
1 • SELECT
2     EXTRACT(HOUR FROM order_time) AS order_hour,
3     COUNT(order_id) AS total_orders
4     FROM orders
5     GROUP BY order_hour
6     ORDER BY order_hour;
7

```

The Result Grid below the query shows the following data:

order_hour	total_orders
11	65
12	144
13	125
14	106
15	85
16	99
17	130
18	127
19	111
20	80
21	57
22	34

3. Category-wise distribution of pizzas



The screenshot shows a SQL interface with a Navigator pane on the left displaying database schemas and tables. The main area contains a SQL query and its results.

```

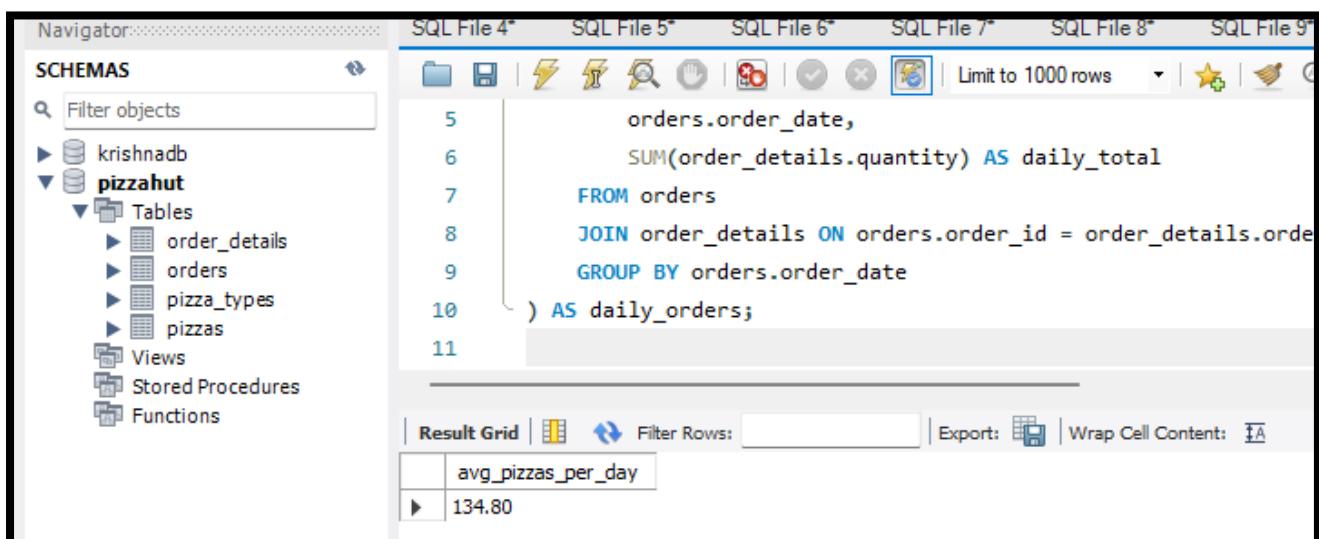
1 •   SELECT
2       category,
3       COUNT(*) AS total_pizza_types
4   FROM pizza_types
5   GROUP BY category
6   ORDER BY total_pizza_types DESC;
7

```

Result Grid

category	total_pizza_types
Supreme	9
Veggie	9
Classic	8
Chicken	6

4. Avg. number of pizzas ordered per day



The screenshot shows a SQL interface with a Navigator pane on the left displaying database schemas and tables. The main area contains a SQL query and its results.

```

5           orders.order_date,
6           SUM(order_details.quantity) AS daily_total
7   FROM orders
8   JOIN order_details ON orders.order_id = order_details.order_id
9   GROUP BY orders.order_date
10      ) AS daily_orders;
11

```

Result Grid

avg_pizzas_per_day
134.80

5. Top 3 pizza types by revenue

The screenshot shows the SQL Workbench interface. On the left, the Navigator pane displays the schema structure under 'SCHEMAS'. The 'pizzahut' schema is expanded, showing tables like 'order_details', 'orders', 'pizza_types', and 'pizzas'. The main area shows a SQL editor with the following query:

```

4   FROM order_details
5   JOIN pizzas ON order_details.pizza_id = pizzas.pizza_id
6   JOIN pizza_types ON pizzas.pizza_type_id = pizza_types.pizza_type_id
7   GROUP BY pizza_types.name
8   ORDER BY total_revenue DESC
9   LIMIT 3;
10

```

Below the query is a 'Result Grid' showing the results:

pizza_type	total_revenue
The Barbecue Chicken Pizza	21368.25
The Thai Chicken Pizza	20251.75
The California Chicken Pizza	19999

C. Advanced-Level Analysis

1. % contribution of each pizza type to total revenue

The screenshot shows the SQL Workbench interface. On the left, the Navigator pane displays the schema structure under 'SCHEMAS'. The 'pizzahut' schema is expanded, showing tables like 'order_details', 'orders', 'pizza_types', and 'pizzas'. The main area shows a SQL editor with the following complex query:

```

1 •  SELECT
2     pizza_types.name AS pizza_type,
3     ROUND(SUM(order_details.quantity * pizzas.price), 2) AS revenue,
4     ROUND(
5         100.0 * SUM(order_details.quantity * pizzas.price) /
6         (SELECT SUM(order_details.quantity * pizzas.price)
7          FROM order_details
8          JOIN pizzas ON order_details.pizza_id = pizzas.pizza_id),
9     ) AS percent_contribution
10    FROM order_details
11    JOIN pizzas ON order_details.pizza_id = pizzas.pizza_id

```

Below the query is a 'Result Grid' showing the results:

pizza_type	revenue	percent_contribution
The California Chicken Pizza	19999	5.04
The Classic Deluxe Pizza	17749.5	4.48
The Spicy Italian Pizza	16930	4.27
The Italian Supreme Pizza	16627.5	4.19
The Southwest Chicken Pizza	16324.5	4.12
The Four Cheese Pizza	15984.8	4.03
The Hawaiian Pizza	15418.5	3.89
The Sicilian Pizza	14996	3.78
The Pepperoni Pizza	14432.25	3.64
The Greek Pizza	14190.15	3.58
The Five Cheese Pizza	12987	3.28
The Mexicana Pizza	12733.5	3.21
The Pepper Salami Pizza	12614.5	3.18

2. Cumulative revenue over time

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

- Schemas:** krishnadb, pizzahut (selected).
- Tables:** order_details, orders, pizza_types, pizzas.
- Query:**

```

1 •  SELECT
2     orders.order_date,
3     ROUND(SUM(order_details.quantity * pizzas.price), 2) AS daily_revenue,
4     ROUND(SUM(SUM(order_details.quantity * pizzas.price)) OVER (ORDER BY orders.order_date), 2) AS cumulative_revenue
5   FROM orders
6   JOIN order_details ON orders.order_id = order_details.order_id
7   JOIN pizzas ON order_details.pizza_id = pizzas.pizza_id
8   GROUP BY orders.order_date
9   ORDER BY orders.order_date;

```
- Result Grid:**

order_date	daily_revenue	cumulative_revenue
2015-01-01	2713.85	2713.85
2015-01-02	2731.9	5445.75
2015-01-03	2662.4	8108.15
2015-01-04	1755.45	9863.6
2015-01-05	2065.95	11929.55
2015-01-06	2428.95	14358.5
2015-01-07	2202.2	16560.7
2015-01-08	2838.35	19399.05
2015-01-09	2127.35	21526.4
2015-01-10	2463.95	23990.35
2015-01-11	1872.3	25862.65
2015-01-12	1919.05	27781.7
2015-01-13	2049.6	29831.3
2015-01-14	2527.4	32358.7
- Right Panel:** Result Grid, Form Editor, Field Types, Query Stats.

3. Top 3 pizza types by revenue within each category

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

- Schemas:** krishnadb, pizzahut (selected).
- Tables:** order_details, orders, pizza_types, pizzas.
- Query:**

```

7     RANK() OVER (PARTITION BY pizza_types.category ORDER BY SUM(order_details.quantity * pizzas.price) DESC) AS rank_in_category
8
9   FROM order_details
10  JOIN pizzas ON order_details.pizza_id = pizzas.pizza_id
11  JOIN pizza_types ON pizzas.pizza_type_id = pizza_types.pizza_type_id
12  GROUP BY pizza_types.category, pizza_types.name
13 ) AS ranked_pizzas
14 WHERE rank_in_category <= 3
15 ORDER BY category, revenue DESC;

```
- Result Grid:**

category	pizza_type	revenue	rank_in_category
Chicken	The Barbecue Chicken Pizza	21368.25	1
Chicken	The Thai Chicken Pizza	20251.75	2
Chicken	The California Chicken Pizza	19999	3
Classic	The Classic Deluxe Pizza	17749.5	1
Classic	The Hawaiian Pizza	15418.5	2
Classic	The Pepperoni Pizza	14432.25	3
Supreme	The Spicy Italian Pizza	16930	1
Supreme	The Italian Supreme Pizza	16627.5	2
Supreme	The Sicilian Pizza	14996	3
Veggie	The Four Cheese Pizza	15984.8	1
Veggie	The Five Cheese Pizza	12987	2
Veggie	The Mexicana Pizza	12733.5	3
- Right Panel:** Res Grd, Form Edit, Field Typ, Qu.

4. Conclusion

1. The most ordered pizza type is: **BBQ Chicken Pizza**
2. The highest revenue is generated by: **Non-Veg category**
3. Most orders are placed during: **12 PM to 2 PM**
4. Revenue contribution is highly skewed toward a few top-performing pizzas like BBQ Chicken and Veggie Deluxe.