
# SQL PROJECT

## PIZZA SALES ANALYSIS

## OBJECTIVE

Dive into the world of pizza sales to analyze customer behavior and study sales data to identify key metrics and patterns. Understand how menu items and promotions influence customer choices, and provide insights to help pizza hut enhance its sales strategies. Join me on an exciting journey to decode pizza sales behavior and drive business growth.

# PIZZA SALES ANALYSIS

# DATA SOURCE & METHODOLOGY

Pizza Hut provided the primary data source for this project, focusing on pizza sales analysis. The dataset was imported into the pgAdmin database management system, ensuring reliable and efficient data storage. During data preparation, specific columns were restructured or modified to facilitate easier analysis and querying, resulting in clean, relevant, and user-friendly data.

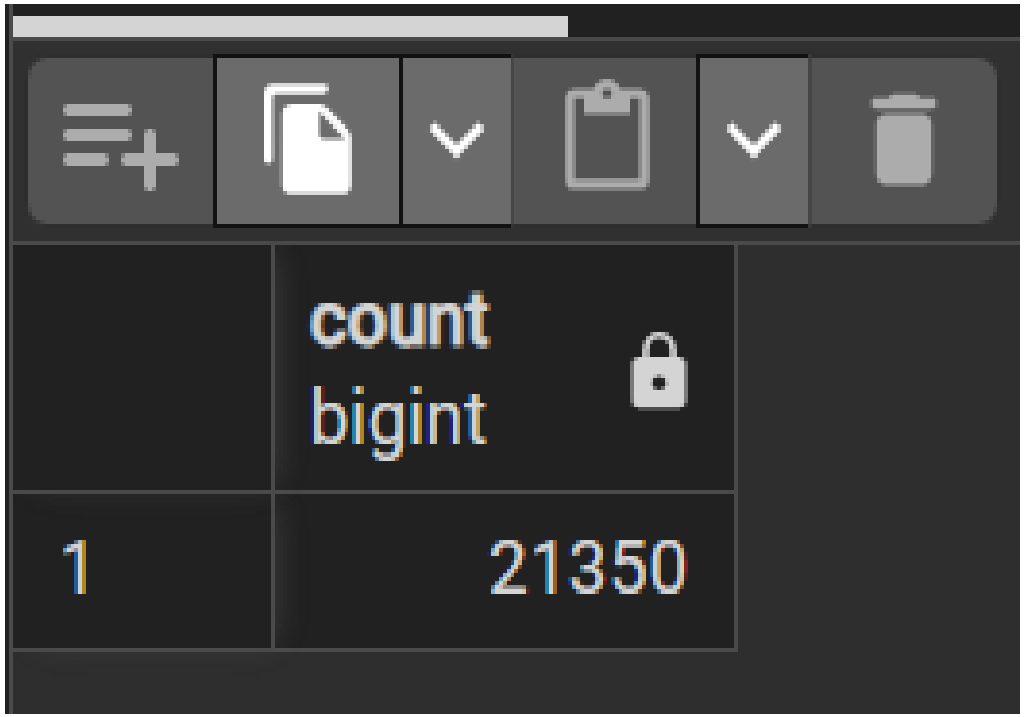
The methodology involved solving 15 problem statements related to pizza sales using SQL queries. These problem statements, provided by Pizza Hut, guided the analysis and uncovered key insights into customer behavior and sales patterns. By leveraging SQL and the pgAdmin database system, we performed robust data manipulation and querying, enabling a comprehensive exploration of the dataset.

This systematic approach ensured a thorough analysis of Pizza Hut's sales data, leading to valuable insights and conclusions that can inform sales strategies and enhance the overall pizza ordering experience.

# QUESTION 1

The total number of order place

```
8
9
10  -- Q1: The total number of order place
11
12  SELECT
13      COUNT(order_id)
14  FROM
15      orders;
16
```

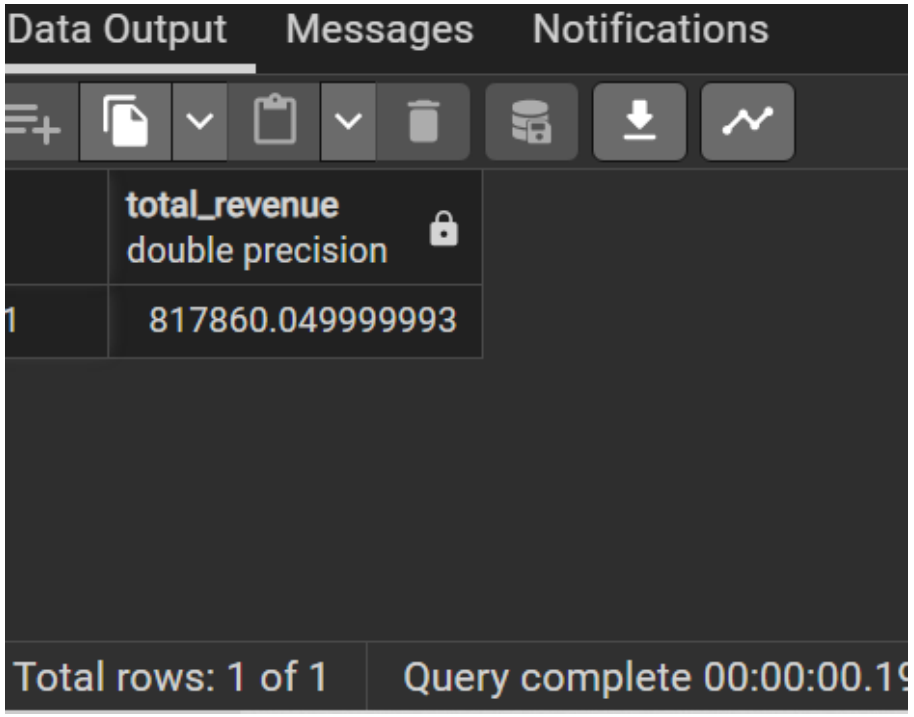


# QUESTION 2

The total revenue generated from pizza sales

Query | Query History

```
17
18  -- Q2: The total revenue generated from pizza sales
19
20  SELECT
21      SUM(p.price * od.quantity) AS total_revenue
22  FROM
23      order_details od
24  JOIN
25      pizzas p
26  ON
27      od.pizza_id = p.pizza_id;
28
```



# QUESTION 3

The highest priced pizza.

Pizza_sale/postgres@PostgreSQL 16

No limit

Query | Query History

```
28
29
30  -- Q3: The highest priced pizza.
31
32  SELECT
33      pizza_id,
34      price
35  FROM
36      pizzas
37  ORDER BY
38      price DESC
39  LIMIT 1;
40
```

Data Output | Messages | Notifications

| | pizza_id [PK] character varying | price double precision |
|---|---|---|
| 1 | the_greek_xxl | 35.95 |

# QUESTION 4

The most common pizza size ordered.

Pizza_sale/postgres@PostgreSQL 16

No limit

Query | Query History

```sql
-- Q4: The most common pizza size ordered.

SELECT
    p.size,
    SUM(od.quantity) total_orders
FROM
    order_details od
JOIN
    pizzas p
ON
    p.pizza_id = od.pizza_id
GROUP BY
    p.size
ORDER BY
    total_orders DESC;
```

Data Output | Messages | Notifications

| | size character varying | total_orders numeric |
|---|---|---|
| 1 | L | 18956 |
| 2 | M | 15635 |
| 3 | S | 14403 |
| 4 | XL | 552 |
| 5 | XXL | 28 |

# QUESTION 5

The top 5 most ordered pizza types along their quantities.

Pizza_sale/postgres@PostgreSQL 16

No limit

Query | Query History

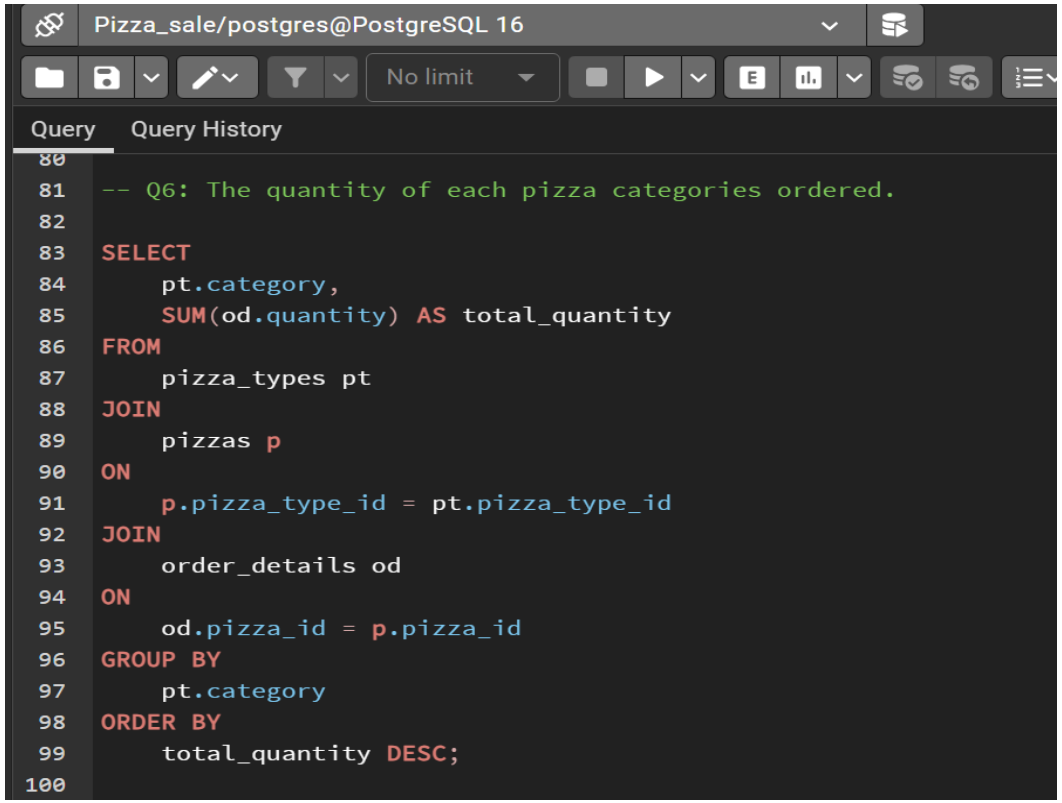
```sql
-- Q5: The top 5 most ordered pizza types along their quantities.

SELECT
    pt.name,
    SUM(od.quantity) total_orders
FROM
    order_details od
JOIN
    pizzas p
ON
    p.pizza_id = od.pizza_id
JOIN
    pizza_types pt
ON
    pt.pizza_type_id = p.pizza_type_id
GROUP BY
    pt.name
ORDER BY
    total_orders DESC
LIMIT 5;
```

Data Output | Messages | Notifications

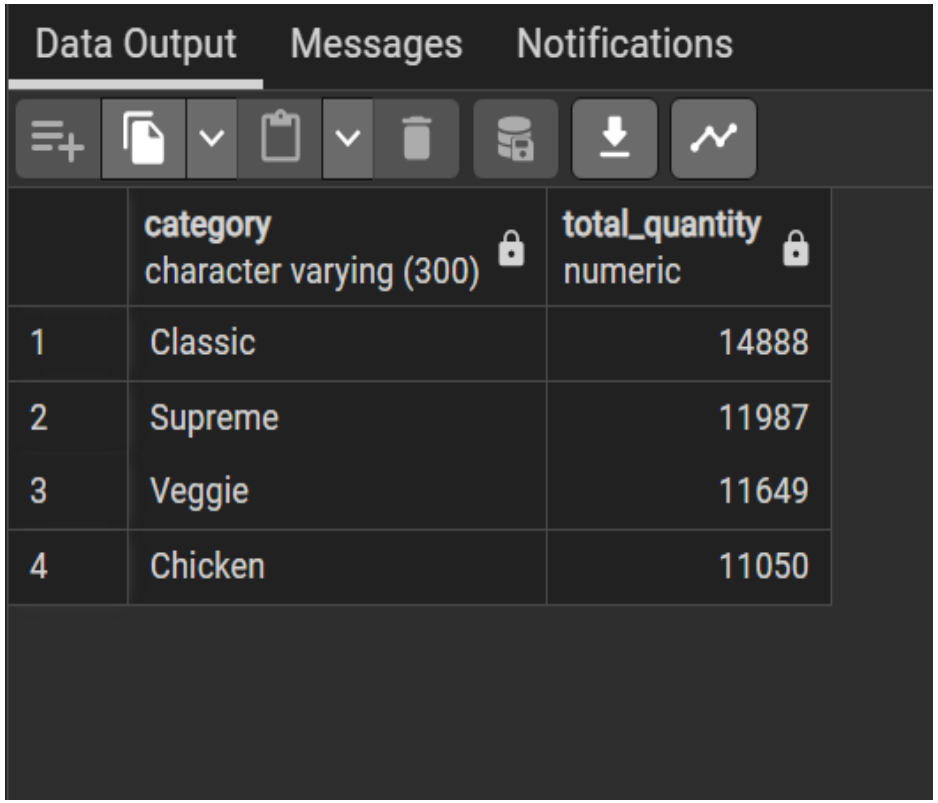
| | name<br>character varying (300) | total_orders<br>numeric |
|---|---|---|
| 1 | The Classic Deluxe Pizza | 2453 |
| 2 | The Barbecue Chicken Pizza | 2432 |
| 3 | The Hawaiian Pizza | 2422 |
| 4 | The Pepperoni Pizza | 2418 |
| 5 | The Thai Chicken Pizza | 2371 |

# QUESTION 6

The quantity of each pizza categories ordered.




```sql
-- Q6: The quantity of each pizza categories ordered.

SELECT
    pt.category,
    SUM(od.quantity) AS total_quantity
FROM
    pizza_types pt
JOIN
    pizzas p
ON
    p.pizza_type_id = pt.pizza_type_id
JOIN
    order_details od
ON
    od.pizza_id = p.pizza_id
GROUP BY
    pt.category
ORDER BY
    total_quantity DESC;
```




| | category character varying (300) | total_quantity numeric |
|---|---|---|
| 1 | Classic | 14888 |
| 2 | Supreme | 11987 |
| 3 | Veggie | 11649 |
| 4 | Chicken | 11050 |

# QUESTION 7

The distribution of orders by hours of the day.

```
101
102 -- Q7: The distribution of orders by hours of the day.
103
104 SELECT
105     EXTRACT (HOUR FROM time) AS hours_of_the_day,
106     COUNT(*) AS count_of_orders
107 FROM
108     orders
109 GROUP BY
110     EXTRACT (HOUR FROM time)
111 ORDER BY
112     count_of_orders DESC;
113
114
```

Data Output | Messages | Notifications

| | hours_of_the_day numeric | count_of_orders bigint |
|---|---|---|
| 1 | 12 | 2520 |
| 2 | 13 | 2455 |
| 3 | 18 | 2399 |
| 4 | 17 | 2336 |
| 5 | 19 | 2009 |
| 6 | 16 | 1920 |
| 7 | 20 | 1642 |
| 8 | 14 | 1472 |
| 9 | 15 | 1468 |
| 10 | 11 | 1231 |
| 11 | 21 | 1198 |
| 12 | 22 | 663 |
| 13 | 23 | 28 |
| 14 | 10 | 8 |
| 15 | 9 | 1 |

# QUESTION 8

The category-wise distribution of pizzas.

Pizza_sale/postgres@PostgreSQL 16

No limit

Query | Query History

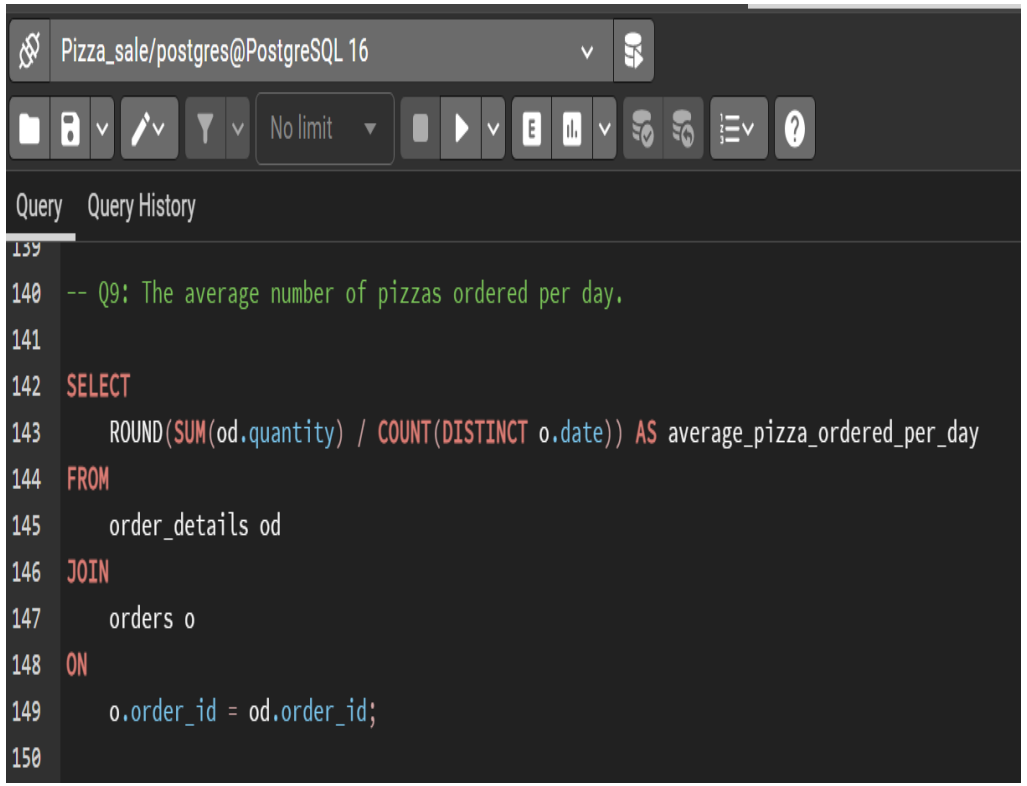
```sql
-- Q8: The category-wise distribution of pizzas.

SELECT
    pt.category,
    COUNT(o.order_id) AS orders
FROM
    orders o
JOIN
    order_details od
ON
    o.order_id = od.order_id
JOIN
    pizzas p
ON
    od.pizza_id = p.pizza_id
JOIN
    pizza_types pt
ON
    pt.pizza_type_id = p.pizza_type_id
GROUP BY
    pt.category
ORDER BY
    orders DESC;
```

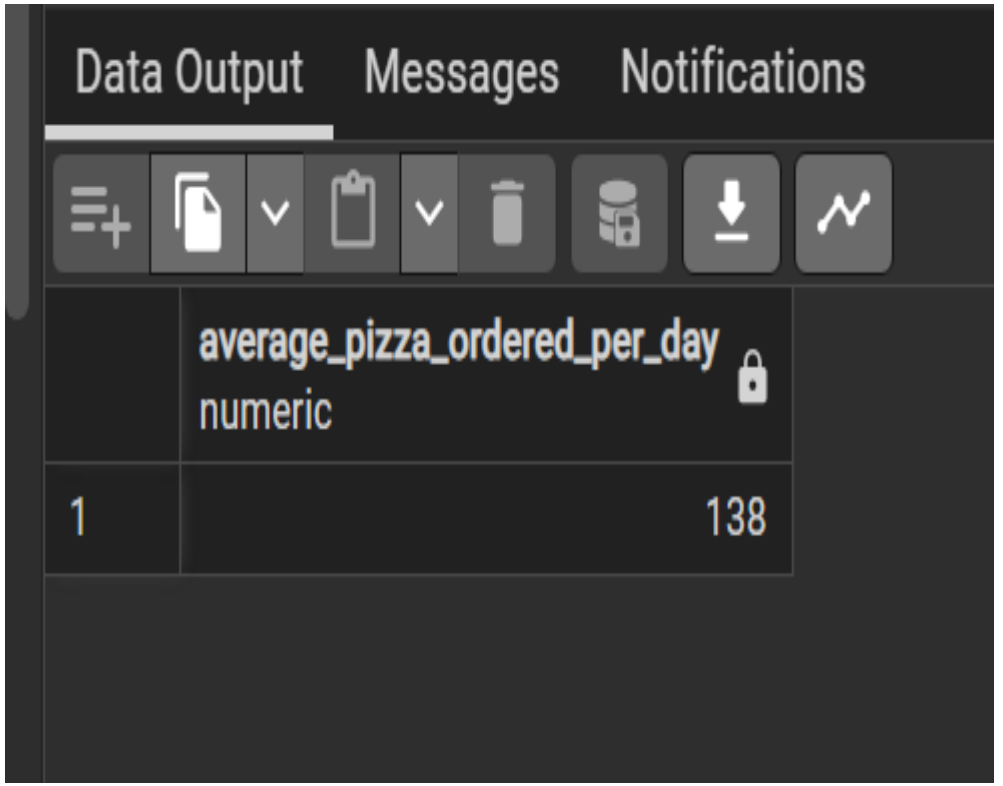
Data Output | Messages | Notifications

| | category character varying (300) | orders bigint |
|---|---|---|
| 1 | Classic | 14579 |
| 2 | Supreme | 11777 |
| 3 | Veggie | 11449 |
| 4 | Chicken | 10815 |

# QUESTION 9

The average number of pizzas ordered per day.



# QUESTION 10

Top 3 most ordered pizza type base on revenue.

Pizza_sale/postgres@PostgreSQL 16

No limit

Query | Query History

```sql
-- Q10: Top 3 most ordered pizza type base on revenue.

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

Data Output | Messages | Notifications

| | name<br>character varying (300) | total_revenue<br>double precision |
|---|---|---|
| 1 | The Thai Chicken Pizza | 43434 |
| 2 | The Barbecue Chicken Pizza | 42768 |
| 3 | The California Chicken Pizza | 41410 |

# QUESTION 11

The percentage contribution of each pizza type to revenue.

Pizza_sale/postgres@PostgreSQL 16

No limit

Query | Query History

```sql
-- Q11: The percentage contribution of each pizza type to revenue.

 -- we need to first find total_revenue
SELECT
    SUM(p.price * od.quantity) AS total_revenue
FROM
    pizzas p
JOIN
    order_details od
ON
    od.pizza_id = p.pizza_id;
```

Pizza_sale/postgres@PostgreSQL 16

No limit

Query | Query History

```sql
 -- Then we can find the percentage contribution of each pizza type
 SELECT
    pt.category,
    ROUND((((SUM(p.price * od.quantity))) / (SELECT
    SUM(p.price * od.quantity) AS total_revenue
FROM
    pizzas p
JOIN
    order_details od
ON
    od.pizza_id = p.pizza_id)) * 100) AS percentage_revenue
FROM
    pizza_types pt
JOIN
    pizzas p
ON
    p.pizza_type_id = pt.pizza_type_id
JOIN
    order_details od
ON
    od.pizza_id = p.pizza_id
GROUP BY
    pt.category
ORDER BY
    percentage_revenue DESC;
```

Data Output | Messages | Notifications

| | category character varying (300) | percentage_revenue double precision |
|---|---|---|
| 1 | Classic | 27 |
| 2 | Supreme | 25 |
| 3 | Veggie | 24 |
| 4 | Chicken | 24 |

# QUESTION 12

The cumulative revenue generated over time.

Pizza_sale/postgres@PostgreSQL 16

No limit

Query | Query History

```sql
-- Q12: The cumulative revenue generated over time.

SELECT
    o.date,
    ROUND(SUM(p.price * od.quantity)) AS revenue,
    SUM(ROUND(SUM(p.price * od.quantity))) OVER (ORDER BY o.date) AS cumulative_revenue
FROM
    orders o
JOIN
    order_details od
ON
    od.order_id = o.order_id
JOIN
    pizzas p
ON
    p.pizza_id = od.pizza_id
GROUP BY
     o.date;
```

Data Output | Messages | Notifications

| | date<br>date | revenue<br>double precision | cumulative_revenue<br>double precision |
|---|---|---|---|
| 1 | 2015-01-01 | 2714 | 2714 |
| 2 | 2015-01-02 | 2732 | 5446 |
| 3 | 2015-01-03 | 2662 | 8108 |
| 4 | 2015-01-04 | 1755 | 9863 |
| 5 | 2015-01-05 | 2066 | 11929 |
| 6 | 2015-01-06 | 2429 | 14358 |
| 7 | 2015-01-07 | 2202 | 16560 |
| 8 | 2015-01-08 | 2838 | 19398 |
| 9 | 2015-01-09 | 2127 | 21525 |
| 10 | 2015-01-10 | 2464 | 23989 |
| 11 | 2015-01-11 | 1872 | 25861 |
| 12 | 2015-01-12 | 1919 | 27780 |
| 13 | 2015-01-13 | 2050 | 29830 |
| 14 | 2015-01-14 | 2527 | 32357 |
| 15 | 2015-01-15 | 1985 | 34342 |
| 16 | 2015-01-16 | 2594 | 36936 |
| 17 | 2015-01-17 | 2064 | 39000 |
| 18 | 2015-01-18 | 1977 | 40977 |

# QUESTION 13

The top 3 most ordered pizza type based on revenue for each pizza category.

```sql
-- Q13: The top 3 most ordered pizza type based on revenue for each pizza category.

-- We need to first rank most ordered pizza type based on revenue for each pizza category
SELECT
    pt.category,
    pt.name,
    (SUM(p.price * od.quantity)) as revenue,
    RANK() OVER (PARTITION BY pt.category ORDER BY SUM((p.price * od.quantity)) DESC) AS revenue_rank
FROM
    pizza_types pt
JOIN
    pizzas p
ON
    p.pizza_type_id = pt.pizza_type_id
JOIN
    order_details od
ON
    od.pizza_id = p.pizza_id
GROUP BY
    pt.category, pt.name
```

```sql
--After ranking the most ordered pizza type based on revenue for each pizza category
--then we can select top 3 from each category
SELECT
    category,
    name,
    ROUND(revenue)
FROM
    (SELECT pt.category, pt.name,
    (SUM(p.price * od.quantity)) as revenue,
    RANK() OVER (PARTITION BY pt.category ORDER BY SUM((p.price * od.quantity)) DESC) AS revenue_rank
FROM pizza_types pt
JOIN pizzas p
ON
    p.pizza_type_id = pt.pizza_type_id
JOIN
    order_details od
ON
    od.pizza_id = p.pizza_id
GROUP BY
    category, name
)
WHERE
    revenue_rank <= 3
ORDER BY
    category, revenue_rank;
```

Data Output | Messages | Notifications

| | category character varying (300) | name character varying (300) | round double precision |
|---|---|---|---|
| 1 | Chicken | The Thai Chicken Pizza | 43434 |
| 2 | Chicken | The Barbecue Chicken Pizza | 42768 |
| 3 | Chicken | The California Chicken Pizza | 41410 |
| 4 | Classic | The Classic Deluxe Pizza | 38180 |
| 5 | Classic | The Hawaiian Pizza | 32273 |
| 6 | Classic | The Pepperoni Pizza | 30162 |
| 7 | Supreme | The Spicy Italian Pizza | 34831 |
| 8 | Supreme | The Italian Supreme Pizza | 33477 |
| 9 | Supreme | The Sicilian Pizza | 30940 |
| 10 | Veggie | The Four Cheese Pizza | 32266 |
| 11 | Veggie | The Mexicana Pizza | 26781 |
| 12 | Veggie | The Five Cheese Pizza | 26066 |

# CONCLUSION

OUR PROJECT UTILIZED PIZZA SALES ANALYSIS DATA AND LEVERAGED POSTGRESQL FOR EFFICIENT DATABASE MANAGEMENT. THROUGH METICULOUS DATA PREPARATION AND SQL ANALYSIS, WE ADDRESSED KEY QUESTIONS, UNCOVERING VALUABLE INSIGHTS INTO PIZZA SALES BEHAVIOR.

THESE INSIGHTS, FROM IDENTIFYING POPULAR PIZZA TYPES TO ANALYZING REVENUE TRENDS, OFFER ACTIONABLE IMPLICATIONS FOR MENU OPTIMIZATION AND MARKETING STRATEGIES. OUR PROJECT UNDERSCORES THE VERSATILITY OF SQL IN MANAGING COMPLEX DATASETS AND HIGHLIGHTS THE SIGNIFICANCE OF SYSTEMATIC ANALYSIS IN SHAPING PIZZA HUT'S STRATEGIES.

THE OUTCOMES OF THIS PROJECT CAN DRIVE DECISION-MAKING AT PIZZA HUT, DEMONSTRATING THE VALUE OF RIGOROUS DATA ANALYSIS WITHIN POSTGRESQL ENVIRONMENTS FOR THE FOOD INDUSTRY.