# Pizza Hut Sales Project

# welcome to
# Pizza Hut Sales Project

Just wrapped up an exciting SQL project: a deep dive into Pizza Hut sales data! I explored some new techniques and uncovered some interesting insights. Really enjoyed working on this and excited to share it with you all. Check it out!

# VISSION & MISSION

## VISSION

To transform raw Pizza Hut sales data into actionable insights that directly empower strategic decision-making, optimize operational efficiency, and drive sustainable revenue growth across the organization

## MISSION

To perform in-depth SQL analysis on Pizza Hut sales transactions and customer data, identifying preferences, purchasing behaviors, and engagement trends to recommend tailored product promotions, personalized marketing campaigns, and service enhancements that improve customer satisfaction and loyalty.

# MEET OUR CREATOR

ANIKET MISHRA

# 1. Retrieve the total number of orders placed.

```sql
SELECT
    COUNT(order_id) AS total_number_of_orders_placed
FROM
    orders;
```

| total_number_of_orders_placed |
| --- |
| 21350 |

# 2. CALCULATE THE TOTAL REVENUE GENERATED FROM PIZZA SALES.

```sql
SELECT
    ROUND(SUM(order_details.quantity * pizzas.price),
            2) AS Generated_revenue
FROM
    order_details
        JOIN
    pizzas ON pizzas.pizza_id = order_details.pizza_id;
```




Result Grid | Filter

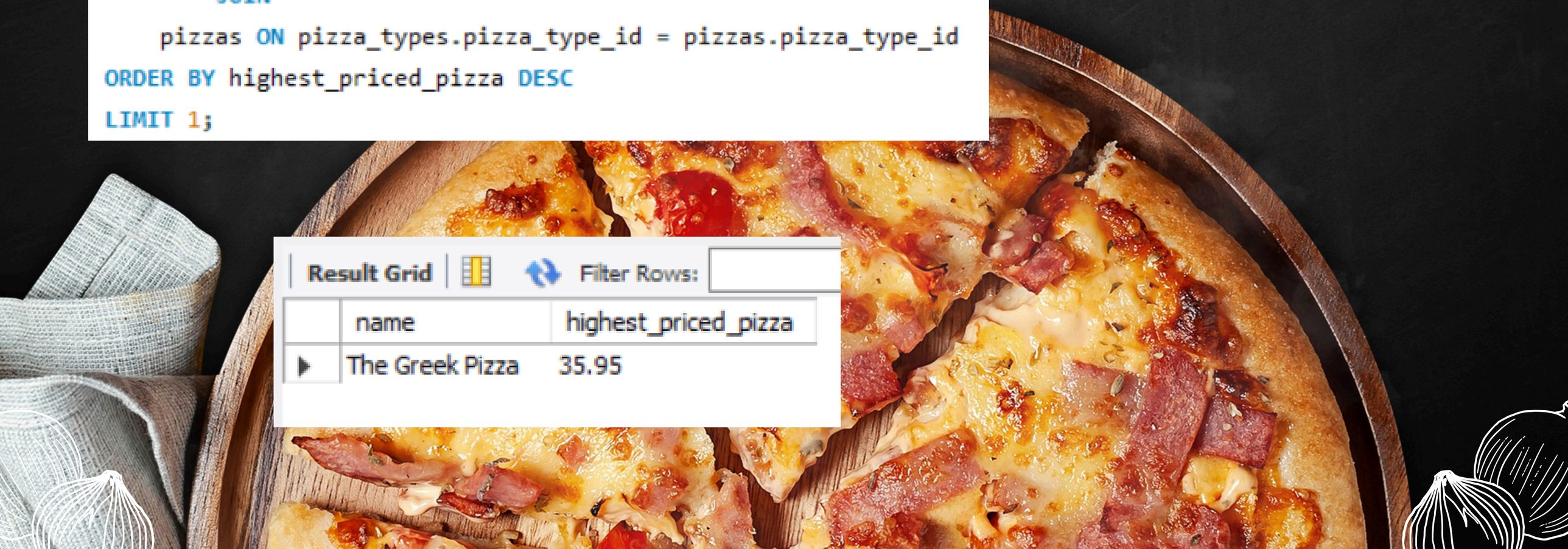
| | Generated_revenue |
|---|---|
| ▶ | 817860.05 |

# 3. Identify the highest-priced pizza.

```sql
SELECT
    pizza_types.name, pizzas.price AS highest_priced_pizza
FROM
    pizza_types
        JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
ORDER BY highest_priced_pizza DESC
LIMIT 1;
```

Result Grid | Filter Rows:

| name | highest_priced_pizza |
|---|---|
| The Greek Pizza | 35.95 |



# 4. Identify the most common pizza size ordered.

```sql
SELECT
    pizzas.size,
    COUNT(order_details.order_details_id) AS common_pizza_size_ordered
FROM
    pizzas
        JOIN
    order_details ON pizzas.pizza_id = order_details.pizza_id
GROUP BY pizzas.size
ORDER BY common_pizza_size_ordered DESC
LIMIT 1;
```

Result Grid | Filter Rows:

| size | common_pizza_size_ordered |
|---|---|
| L | 18526 |

# 5. LIST THE TOP 5 MOST ORDERED PIZZA TYPES ALONG WITH THEIR QUANTITIES.

```sql
SELECT
    pizza_types.name,
    SUM(order_details.quantity) AS top_5_most_ordered_pizza
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 top_5_most_ordered_pizza DESC
LIMIT 5;
```

Result Grid | Filter Rows: | Export:

| name | top_5_most_ordered_pizza |
|---|---|
| The Classic Deluxe Pizza | 2453 |
| The Barbecue Chicken Pizza | 2432 |
| The Hawaiian Pizza | 2422 |
| The Pepperoni Pizza | 2418 |
| The Thai Chicken Pizza | 2371 |

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

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

Result Grid | Filter Rows:

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

# 7. Determine the distribution of orders by hour of the day.

```
SELECT
    HOUR(order_time) AS order_time,
    COUNT(order_id) AS total_order
FROM
    orders
GROUP BY HOUR(order_time);
```

| order_time | total_order |
|---|---|
| 11 | 1231 |
| 12 | 2520 |
| 13 | 2455 |
| 14 | 1472 |
| 15 | 1468 |

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

```sql
SELECT
    category,count(name) AS Total_count
FROM
    pizza_types
        JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
GROUP BY Category;
```

Result Grid | Filter Ro

| | category | Total_count |
|---|---|---|
| ▶ | Chicken | 18 |
| | Classic | 26 |
| | Supreme | 25 |
| | Veggie | 27 |

# 9. Group the orders by date and calculatethe average number of pizzas ordered per day.

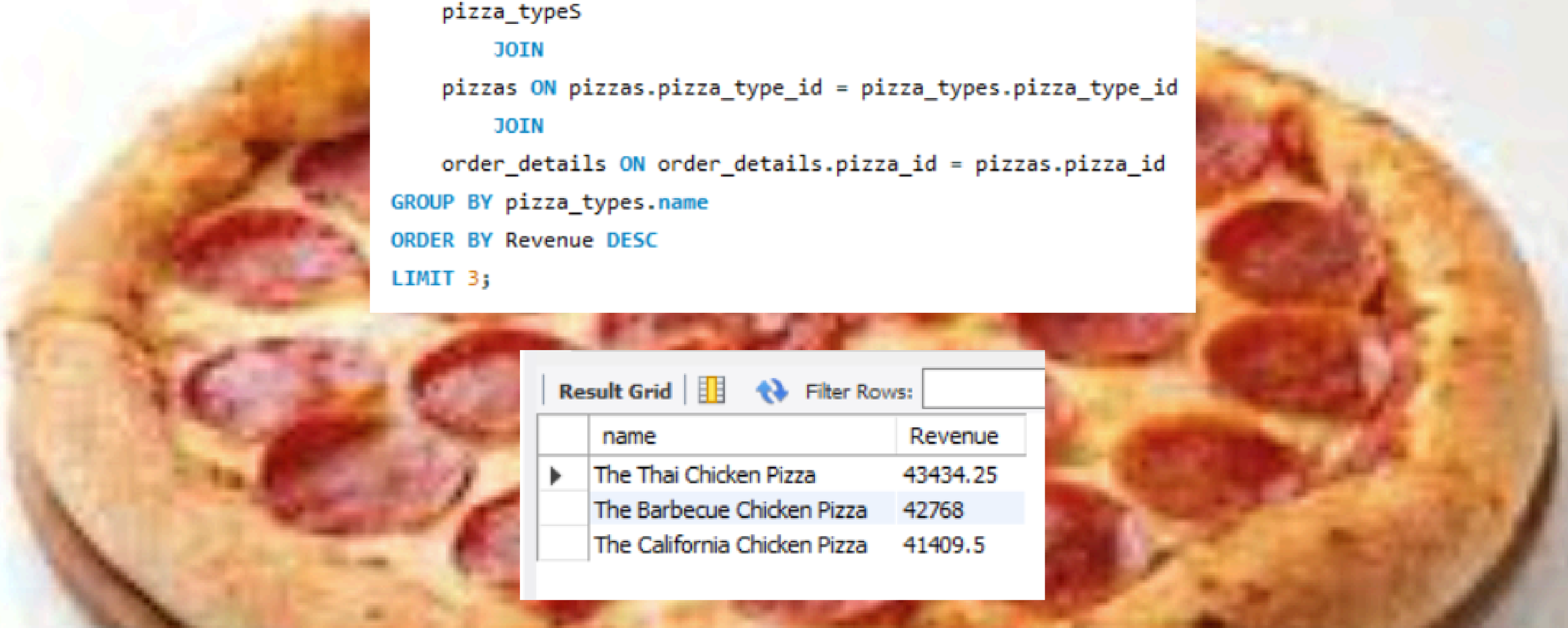
```sql
SELECT
    ROUND(AVG(quantiy_order)) AS Average_pizza_order_per_day
FROM
    (SELECT
        orders.order_date,
            SUM(order_details.quantity) AS quantiy_order
    FROM
        orders
    JOIN order_details ON orders.order_id = order_details.order_id
    GROUP BY orders.order_date) AS Order_quantity;
```

Result Grid | Filter Rows:

| Average_pizza_order_per_day |
| --- |
| 138 |

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

```sql
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:

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

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

```sql
SELECT
    pizza_types.category,
    ROUND(SUM((order_details.quantity * pizzas.price) / (SELECT
                    ROUND(SUM(order_details.quantity * pizzas.price),
                                2) AS Generated_revenue
                FROM
                    order_details
                        JOIN
                    pizzas ON pizzas.pizza_id = order_details.pizza_id) *
            2) 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.category
ORDER BY Revenue DESC;
```

Result Grid

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

# 12. Analyze the cumulative revenue generated over time.

```sql
SELECT order_date,sum(Revenue) OVER(ORDER BY order_date) as cumulative_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;
```

| order_date | cumulative_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 |

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

```
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 V
WHERE RN>3;
```

| name | Revenue |
|---|---|
| The Southwest Chicken Pizza | 34705.75 |
| The Chicken Alfredo Pizza | 16900.25 |
| The Chicken Pesto Pizza | 16701.75 |
| The Greek Pizza | 28454.100000000013 |
| The Italian Capocollo Pizza | 25094 |

# THANK YOU!