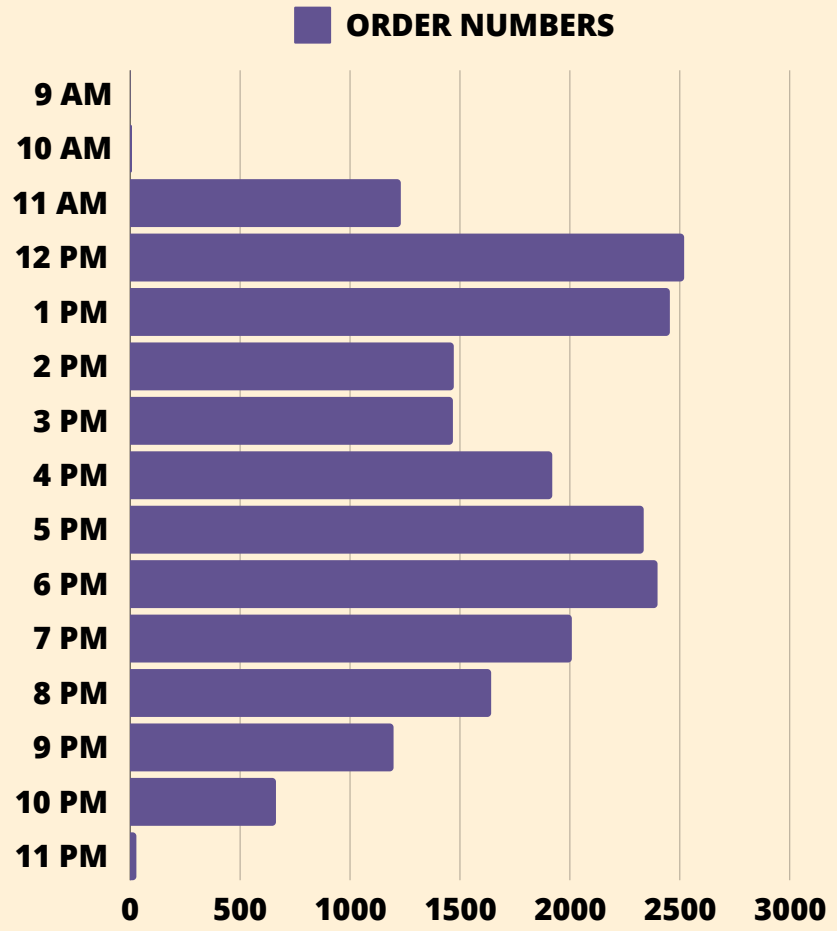
PIZZA
HUT.CO

# SQL PROJECT SHOWCASE

## ANALYZING PIZZA SALES DATA




**PRESENTED BY**
MANAS DUBEY

# Introduction

- **Objective**: The main goal is to analyze extensive pizza order data to uncover trends, assess performance across different pizza categories, and derive insights that can help in strategic decision-making and marketing efforts.

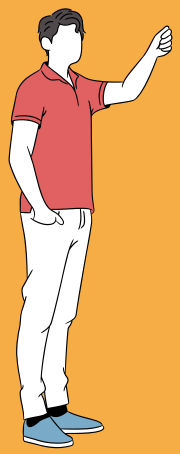
- **Data Sources**: Our analysis is based on four key tables:
  a. **Order Details**: Contains specifics of each order including order ID, pizza ID, and quantity.
  b. **Orders**: Records the date and time of each order.
  c. **Pizzas**: Details about each pizza including size and price.
  d. **Pizza Types**: Information about pizza categories, names, and ingredients.

# Key SQL Queries and Insights

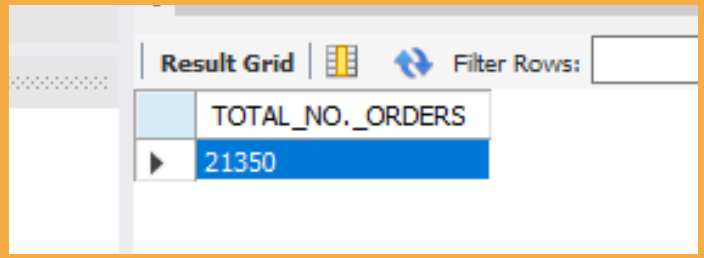
## Basic Analysis:

- **Retrieving the total number of orders placed.**

```sql
-- Retrieve the total number of orders placed.--
SELECT
    COUNT(ORDER_ID) AS 'TOTAL_NO._ORDERS'
FROM
    pizzahut_db.orders
```



*The dataset comprises a total of 21,350 orders, indicating a robust sample for our analysis.*



- **Calculating the total revenue generated from pizza sales.**

```sql
-- Calculate the total revenue generated from pizza sales.
SELECT
    ROUND(SUM(PRICE * QUANTITY), 1) AS REVENUE_GENERATED
FROM
    PIZZAS P
        JOIN
    ORDERS_DETAILS OD ON P.PIZZA_ID = OD.PIZZA_ID
```

*Pizza sales sizzled to a whopping $817,860 in revenue!*

Result Grid | Filter Rows:

| REVENUE_GENERATED |
| --- |
| 817860 |

- **Identifying the highest-priced pizza.**

```sql
-- Identify the highest-priced pizza.
SELECT
    PT.NAME, P.PRICE AS MAX_PRICE
FROM
    PIZZAS P
        JOIN
    PIZZA_TYPES PT ON P.pizza_type_id = PT.pizza_type_id
ORDER BY PRICE DESC
LIMIT 1;
```

*The Greek Pizza steals the spotlight with its lavish $35.98 price tag!*

Result Grid | Filter Rows:

| NAME | MAX_PRICE |
|---|---|
| The Greek Pizza | 35.95 |

- **Identifying the most common pizza size ordered.**

```sql
-- Identify the most common pizza size ordered.
SELECT
    P.size, COUNT(OD.ORDERS_DETAILS_ID) AS frequency
FROM
    pizzas P
        JOIN
    ORDERS_DETAILS OD ON P.pizza_id = OD.pizza_id
GROUP BY size
ORDER BY frequency DESC;
```

*Large pizzas dominate the scene with a whopping 18,526 orders, leading the size chart by a substantial margin!*

Result Grid | Filter Rows:

| size | frequency |
|---|---|
| L | 18526 |
| M | 15385 |
| S | 14137 |
| XL | 544 |
| XXL | 28 |

- **Listing the top 5 most ordered pizza types along with their quantities.**

```sql
-- List the top 5 most ordered pizza types along with their quantities.
SELECT
    PT.NAME, COUNT(QUANTITY) AS TOTAL_QUANTITY
FROM
    PIZZAS P
        JOIN
    PIZZA_TYPES PT ON P.PIZZA_TYPE_ID = PT.PIZZA_TYPE_ID
        JOIN
    ORDERS_DETAILS OD ON P.PIZZA_ID = OD.PIZZA_ID
GROUP BY PT.NAME
ORDER BY TOTAL_QUANTITY DESC
LIMIT 5;
```

*Classic Deluxe leads at 2,416 orders, closely followed by Barbecue Chicken, Hawaiian, Pepperoni, and Thai Chicken!*

Result Grid | Filter Rows: | Export:

| NAME | TOTAL_QUANTITY |
|---|---|
| The Classic Deluxe Pizza | 2416 |
| The Barbecue Chicken Pizza | 2372 |
| The Hawaiian Pizza | 2370 |
| The Pepperoni Pizza | 2369 |
| The Thai Chicken Pizza | 2315 |

# Intermediate Analysis:

- **Finding the total quantity of each pizza category ordered.**

```sql
-- Join the necessary tables to find the total quantity of each pizza category ordered.
SELECT
    PT.CATEGORY, SUM(QUANTITY) AS TOTAL_QUANTITY
FROM
    PIZZAS P
        JOIN
    PIZZA_TYPES PT ON P.PIZZA_TYPE_ID = PT.PIZZA_TYPE_ID
        JOIN
    ORDERS_DETAILS OD ON P.PIZZA_ID = OD.PIZZA_ID
GROUP BY PT.CATEGORY
ORDER BY TOTAL_QUANTITY DESC;
```

*Classic leads with 14,888, followed by Supreme, Veggie, and Chicken!*

Result Grid | Filter Rows:

| CATEGORY | TOTAL_QUANTITY |
|---|---|
| Classic | 14888 |
| Supreme | 11987 |
| Veggie | 11649 |
| Chicken | 11050 |

- **Determine the distribution of orders by hour of the day.**

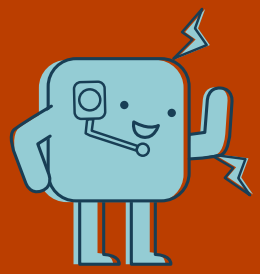
```sql
-- Determine the distribution of orders by hour of the day.
SELECT
    HOUR(ORDER_TIME) as "Hours(24 hrs.)"
    ,COUNT(ORDER_ID) as Order_count
FROM
    ORDERS
GROUP BY HOUR(ORDER_TIME)
ORDER BY HOUR(ORDER_TIME) asc
```

| Hours(24 hrs.) | Order_count |
|---|---|
| 9 | 1 |
| 10 | 8 |
| 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 |

*Orders peak at noon with 2,520, closely followed by 1 PM with 2,455 and 6 PM with 2,336.*

- **Category-wise distribution of pizzas.**

```sql
-- Join relevant tables to find the category-wise distribution of pizzas.
SELECT
    CATEGORY, COUNT(NAME)
FROM
    PIZZA_TYPES
GROUP BY category;
```



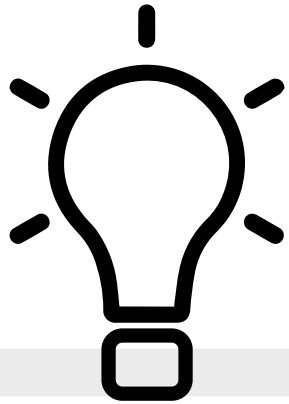
*Chicken struts with 6, Classic vibes with 8, Supreme and Veggie rock out at 9 each!*

Result Grid | Filter Rows:

| CATEGORY | count(NAME) |
|---|---|
| Chicken | 6 |
| Classic | 8 |
| Supreme | 9 |
| Veggie | 9 |

- **Group the orders by date and calculate the average daily pizza order quantity.**

```sql
-- Group the orders by date and calculate the average number of pizzas ordered per day.
WITH SALES_PER_DAY AS (
    SELECT
        DAY(O.ORDER_DATE) AS DayOfMonth,
        SUM(OD.QUANTITY) AS TotalPizzasSold,
        COUNT(DISTINCT DATE(O.ORDER_DATE)) AS NumberOfDays,
        SUM(OD.QUANTITY) / COUNT(DISTINCT DATE(O.ORDER_DATE)) AS AveragePerDay
    FROM
        orders O
    JOIN
        orders_details OD ON O.ORDER_ID = OD.ORDER_ID
    GROUP BY
        DAY(O.ORDER_DATE)
)
SELECT ROUND(AVG(AveragePerDay)) AS AVERAGE_SALES_QUANTITY_PER_DAY
FROM SALES_PER_DAY
```



*Chicken struts with 6, Classic vibes with 8, Supreme and Veggie rock out at 9 each!*

Result Grid | Filter Rows:

| AVERAGE_SALES_QUANTITY_PER_DAY |
| --- |
| 138 |

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

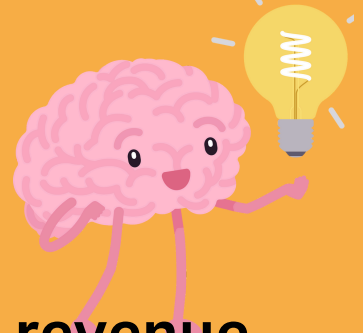
```sql
-- Determine the top 3 most ordered pizza types based on revenue.
SELECT
    PT.NAME,PT.CATEGORY, ROUND(SUM(P.PRICE)) AS TOTAL_REVENUE
FROM
    PIZZAS P
        JOIN
    PIZZA_TYPES PT ON P.PIZZA_TYPE_ID = PT.PIZZA_TYPE_ID
        JOIN
    ORDERS_DETAILS OD ON P.PIZZA_ID = OD.PIZZA_ID
GROUP BY PT.NAME,PT.CATEGORY
ORDER BY TOTAL_REVENUE DESC
LIMIT 3;
```

*Thai Chicken at $42,332, BBQ Chicken hot on its heels at $41,683, and California Chicken close with $40,166!*

Result Grid | Filter Rows: | Export:

| NAME | CATEGORY | TOTAL_REVENUE |
|---|---|---|
| The Thai Chicken Pizza | Chicken | 42332 |
| The Barbecue Chicken Pizza | Chicken | 41683 |
| The California Chicken Pizza | Chicken | 40166 |

# Advanced Analysis:



- **Percentage contribution of each pizza type to total revenue.**

```sql
WITH PRICE_DISTRIBUTION AS (
    SELECT
        PT.CATEGORY,
        SUM(P.PRICE) AS PRICE
    FROM
        PIZZAS P
    JOIN
        PIZZA_TYPES PT ON P.pizza_type_id = PT.pizza_type_id
    GROUP BY
         PT.CATEGORY)

SELECT
    CATEGORY,
    ROUND((PRICE * 100.0 / SUM(PRICE) OVER ()), 1) AS "PRICE_DST%"
FROM
    PRICE_DISTRIBUTION
ORDER BY
    "PRICE_DST%" ASC;
```

*Pie charts of profits: Chicken pizzas carve out 19.1%, Classics lead with 26.9%, Supremes slice up 26.6%, and Veggies sprout to 27.4% of total revenue!*

Result Grid | Filter Rows:

| CATEGORY | PRICE_DST% |
|---|---|
| Chicken | 19.1 |
| Classic | 26.9 |
| Supreme | 26.6 |
| Veggie | 27.4 |

- **Analyze the cumulative revenue generated over time.**

```sql
#--Analyze the cumulative revenue generated over time.#
WITH CTE1 AS(
SELECT O.ORDER_DATE,ROUND(SUM(OD.QUANTITY*P.PRICE)) AS REVENUE
FROM ORDERS_DETAILS OD
JOIN PIZZAS P
ON OD.PIZZA_ID=P.PIZZA_ID
JOIN ORDERS O
ON O.ORDER_ID=OD.ORDER_ID
GROUP BY O.ORDER_DATE)

SELECT ORDER_DATE,REVENUE,(SUM(REVENUE) OVER(ORDER BY ORDER_DATE)) AS CUM_REVENUE
FROM CTE1
```

*Revenue rollercoaster ride ends with a grand total of $817,862 on December 31st*

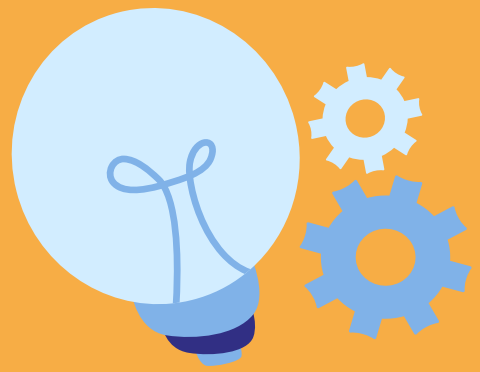
Result Grid | Filter Rows:

| ORDER_DATE | REVENUE | CUM_REVENUE |
|---|---|---|
| 2015-01-22 | 2497 | 50300 |
| 2015-01-23 | 2424 | 52724 |
| 2015-01-24 | 2289 | 55013 |
| 2015-01-25 | 1618 | 56631 |
| 2015-01-26 | 1884 | 58515 |
| 2015-01-27 | 2528 | 61043 |
| 2015-01-28 | 2016 | 63059 |
| 2015-01-29 | 2045 | 65104 |
| 2015-01-30 | 2270 | 67374 |
| 2015-01-31 | 2418 | 69792 |

Result 4

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

```sql
WITH Revenue_Ranking AS (
    SELECT PT.CATEGORY,PT.NAME,P.PIZZA_TYPE_ID,
        ROUND(SUM(QUANTITY * PRICE)) AS TOTAL_REVENUE,
        ROW_NUMBER() OVER (PARTITION BY PT.CATEGORY
        ORDER BY SUM(QUANTITY * PRICE) DESC) AS Ranking
    FROM
        PIZZAS P
    JOIN
        PIZZA_TYPES PT ON P.PIZZA_TYPE_ID = PT.PIZZA_TYPE_ID
    JOIN
        ORDERS_DETAILS OD ON P.PIZZA_ID = OD.PIZZA_ID
    GROUP BY
        PT.CATEGORY, P.PIZZA_TYPE_ID,PT.NAME
)
SELECT CATEGORY,NAME,TOTAL_REVENUE
FROM Revenue_Ranking
WHERE Ranking <= 3;
```



Result Grid | Filter Rows: | Expor

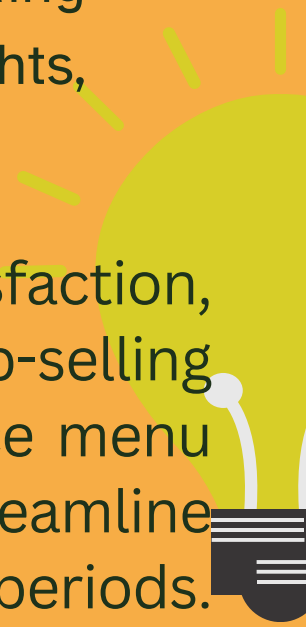
| CATEGORY | NAME | TOTAL_R |
|---|---|---|
| Chicken | The Thai Chicken Pizza | 43434 |
| Chicken | The Barbecue Chicken Pizza | 42768 |
| Chicken | The California Chicken Pizza | 41410 |
| Classic | The Classic Deluxe Pizza | 38180 |
| Classic | The Hawaiian Pizza | 32273 |
| Classic | The Pepperoni Pizza | 30162 |
| Supreme | The Spicy Italian Pizza | 34831 |
| Supreme | The Italian Supreme Pizza | 33477 |
| Supreme | The Sicilian Pizza | 30940 |
| Veggie | The Four Cheese Pizza | 32266 |
| Veggie | The Mexicana Pizza | 26781 |
| Veggie | The Five Cheese Pizza | 26066 |

*Top earners vary across categories, showcasing diverse customer preferences.*

# Findings and Business Implications

**Major Insights**: Our analysis revealed sizzling sales totaling $817,860, with top earners including the Greek Pizza at $35.98, large pizzas dominating orders, and the Classic Deluxe leading as the most ordered pizza type, among other savory insights, guiding future strategies for continued success.

**Recommendations:**To optimize profits and customer satisfaction, it's recommended to focus promotional efforts on top-selling pizza types, particularly during peak sales hours, enhance menu variety in the Veggie and Supreme categories, and streamline operations to improve efficiency during high-demand periods. Implementing customer loyalty programs and continuously adapting to market trends based on feedback will also be crucial for sustained growth.



**Thank You!**
**For your attention and time.**