Home About Contact

PIZZA SALES DATA ANALYSIS USING SQL

CREATED BY: GURKIRAT SINGH
CONTACT: GURKRAT201005@GMAIL.COM
FUTURE GOAL: DATA SCIENTIST

- WHERE EVERY SLICE TELLS A STORY

Pizzahut

INTRODUCTION

Contact

About the Project

- This project focuses on analyzing pizza sales using SQL queries.
- Dataset contains 21,000+ customer orders, making it a real-world business scenario.
- SQL was used for querying, aggregation, and extracting insights.

Why this Dataset?

- Large volume of data → helps test SQL efficiency.
- Covers real-world sales: pizzas, sizes, revenue, and customer preferences.
- Practical use case for business insights and decision-making.

Pizzahut

Objective of Project

Main Objectives-

- Explore and analyze pizza sales data through SQL.
- Implement SQL queries including joins, aggregates, and ranking functions.
- Identify customer preferences and frequently ordered pizza sizes.
- Examine revenue patterns and sales performance.
- Derive meaningful business insights to support decision-making.
- Demonstrate the practical application of SQL in solving real-world business problems.
- Build a foundation for future data visualization and predictive analysis

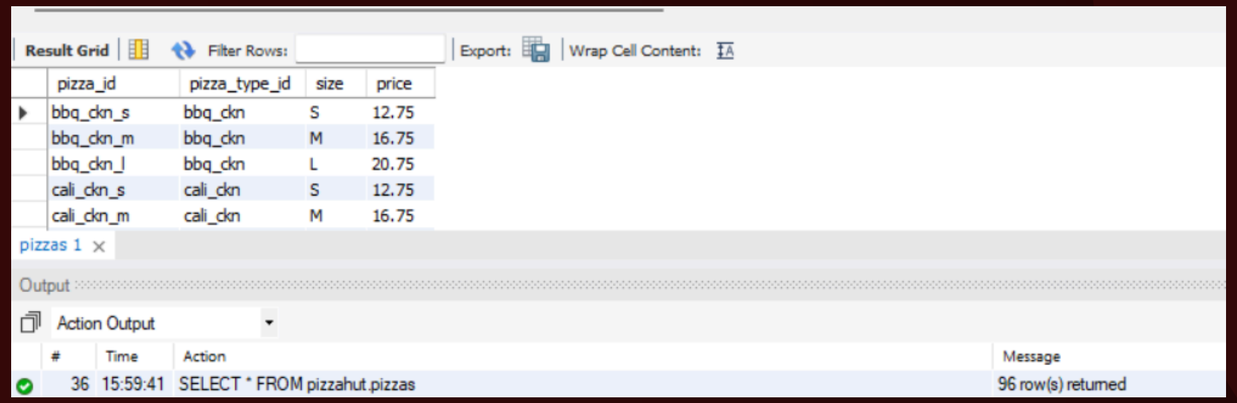
gurkirat201005@gmail.com

Database Overview

Database Name: pizzahut

Tables Used:

- pizzas → details of each pizza (id, type, size, price).
- pizza_types → pizza categories and names.
- orders → customer order information (order id, date, time).
- order_details → quantity of each pizza per order.



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

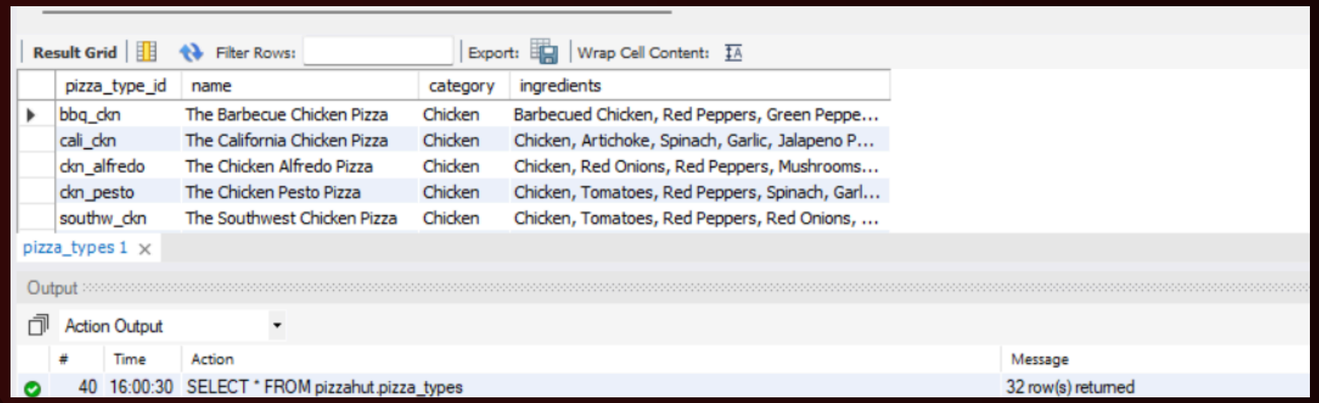
pizza_id	pizza_type_id	size	price
bbq_ckn_s	bbq_ckn	S	12.75
bbq_ckn_m	bbq_ckn	M	16.75
bbq_ckn_l	bbq_ckn	L	20.75
cali_ckn_s	cali_ckn	S	12.75
cali_ckn_m	cali_ckn	M	16.75

pizzas 1

Output

Action Output

#	Time	Action	Message
36	15:59:41	SELECT * FROM pizzahut.pizzas	96 row(s) returned



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

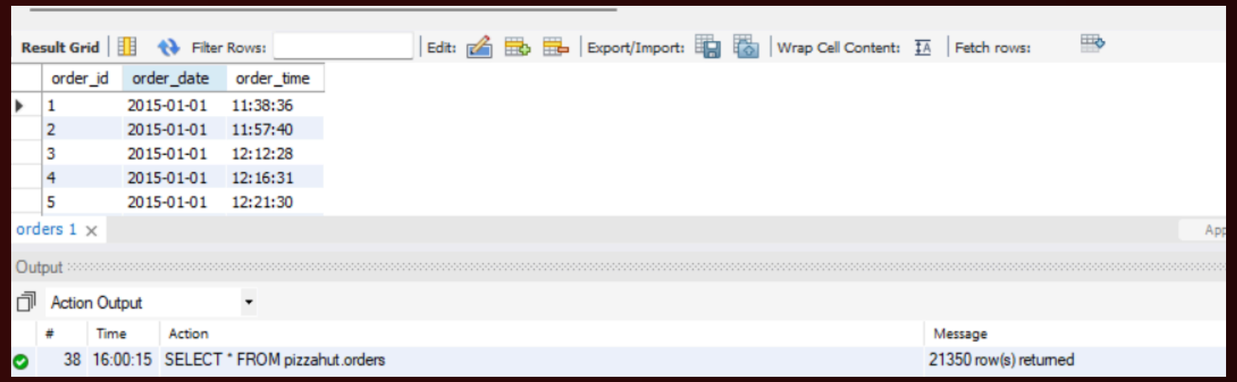
pizza_type_id	name	category	ingredients
bbq_ckn	The Barbecue Chicken Pizza	Chicken	Barbecued Chicken, Red Peppers, Green Peppe...
cali_ckn	The California Chicken Pizza	Chicken	Chicken, Artichoke, Spinach, Garlic, Jalapeno P...
ckn_alfredo	The Chicken Alfredo Pizza	Chicken	Chicken, Red Onions, Red Peppers, Mushrooms...
ckn_pesto	The Chicken Pesto Pizza	Chicken	Chicken, Tomatoes, Red Peppers, Spinach, Garl...
southw_ckn	The Southwest Chicken Pizza	Chicken	Chicken, Tomatoes, Red Peppers, Red Onions, ...

pizza_types 1

Output

Action Output

#	Time	Action	Message
40	16:00:30	SELECT * FROM pizzahut.pizza_types	32 row(s) returned



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

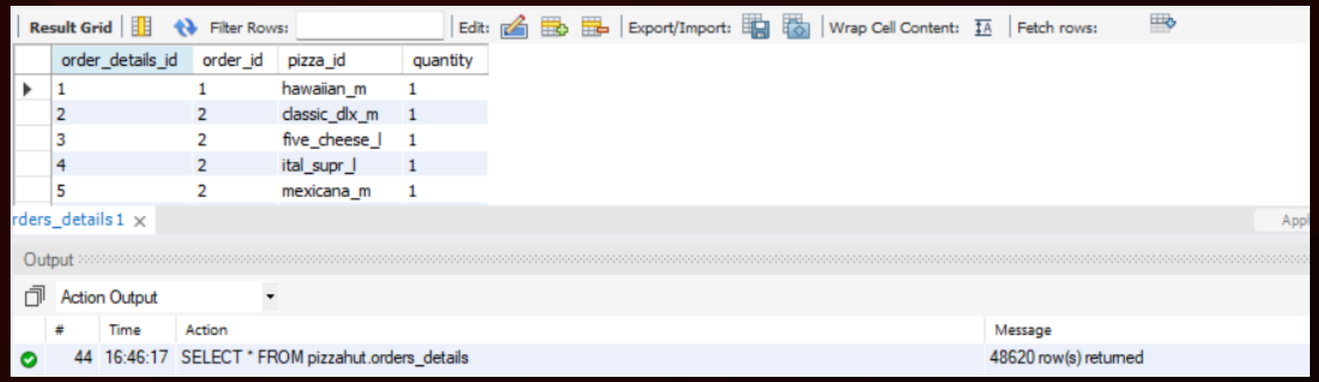
order_id	order_date	order_time
1	2015-01-01	11:38:36
2	2015-01-01	11:57:40
3	2015-01-01	12:12:28
4	2015-01-01	12:16:31
5	2015-01-01	12:21:30

orders 1

Output

Action Output

#	Time	Action	Message
38	16:00:15	SELECT * FROM pizzahut.orders	21350 row(s) returned



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

order_details_id	order_id	pizza_id	quantity
1	1	hawaiian_m	1
2	2	classic_dlx_m	1
3	2	five_cheese_l	1
4	2	ital_supr_l	1
5	2	mexicana_m	1

rders_details 1

Output

Action Output

#	Time	Action	Message
44	16:46:17	SELECT * FROM pizzahut.orders_details	48620 row(s) returned

gurkirat201005@gmail.com

SQL Queries (Basic)

Queries Performed:

1) Retrieve the total number of orders placed.

```sql
-- Retrieve the total number of orders placed.
use pizzahut;
SELECT
    COUNT(order_id) AS total_orders
FROM
    orders;
```

2) Calculate the total revenue generated from pizza sales.

```sql
-- Calculate the total revenue generated from pizza sales
SELECT
    ROUND(SUM(orders_details.quantity * pizzas.price),
            2) AS total_sales
FROM
    orders_details
        JOIN
    pizzas ON pizzas.pizza_id = orders_details.pizza_id;
```

3) Identify the highest-priced pizza.

```sql
-- Identify the highest-priced pizza.
SELECT
    pizza_types.name, pizzas.price
FROM
    pizza_types
        JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
ORDER BY pizzas.price DESC
LIMIT 1;
```

4) Find the most common pizza size ordered.

```sql
-- Identify the most common pizza size ordered.
select quantity ,count(order_details_id) from orders_details group by quantity;
SELECT
    pizzas.size, COUNT(orders_details.order_details_id)
FROM
    pizzas
        JOIN
    orders_details ON pizzas.pizza_id = orders_details.pizza_id
GROUP BY pizzas.size;
```

5) List the Top 5 most ordered pizza types with their quantities.

```sql
-- List the top 5 most ordered pizza types along with their quantities.
SELECT
    pizza_types.name,
    SUM(orders_details.quantity) AS total_quantity
FROM
    pizza_types
        JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
        JOIN
    orders_details ON orders_details.pizza_id = pizzas.pizza_id
GROUP BY pizza_types.name
ORDER BY total_quantity DESC
LIMIT 5;
```

gurkirat201005@gmail.com

SQL Queries (Intermediate)

Queries Performed:

1) Find the total quantity ordered for each pizza category.

```sql
-- Join the necessary tables to find the total quantity of each pizza category ordered.
use pizzahut;
SELECT
    pizza_types.category,
    SUM(orders_details.quantity) AS quantity
FROM
    pizza_types
        JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
        JOIN
    orders_details ON orders_details.pizza_id = pizzas.pizza_id
GROUP BY pizza_types.category
ORDER BY quantity DESC;
```

2) 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 hour, COUNT(order_id) AS order_count
FROM
    orders
GROUP BY hour;
```

3) Show the 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;
```

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

```sql
-- Group the orders by date and calculate the average number of pizzas ordered per day.
SELECT
    AVG(quantity)
FROM
    (SELECT
        orders.order_date, SUM(orders_details.quantity) AS quantity
    FROM
        orders
    JOIN orders_details ON orders.order_id = orders_details.order_id
    GROUP BY orders.order_date) AS order_quantity;
```

5) Identify the Top 3 pizza types based on revenue.

```sql
-- Determine the top 3 most ordered pizza types based on revenue.
SELECT
    pizza_types.name,
    SUM(orders_details.quantity * pizzas.price) AS revenue
FROM
    pizza_types
        JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
        JOIN
    orders_details ON orders_details.pizza_id = pizzas.pizza_id
GROUP BY pizza_types.name
ORDER BY revenue DESC
LIMIT 3;
```

gurkirat201005@gmail.com

SQL Queries (Advance)

Queries Performed:

1) Calculate the percentage revenue contribution of each pizza type.

```sql
-- Calculate the percentage contribution of each pizza type to total revenue.
SELECT
    pizza_types.category,
    ROUND(SUM(orders_details.quantity * pizzas.price) /
         (SELECT SUM(orders_details.quantity * pizzas.price)
          FROM orders_details
          JOIN pizzas ON pizzas.pizza_id = orders_details.pizza_id) * 100, 2) AS revenue_percentage
FROM
    pizza_types
JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
JOIN
    orders_details ON orders_details.pizza_id = pizzas.pizza_id
GROUP BY
    pizza_types.category
ORDER BY
    revenue_percentage DESC;
```

2) Analyze the cumulative revenue growth over time.

```sql
-- Analyze the cumulative revenue generated over time.
select order_date,sum(revenue) over (order by order_date) as cum_revenue from
(select orders.order_date,sum(orders_details.quantity*pizzas.price) as revenue
from orders_details join pizzas on orders_details.pizza_id =pizzas.pizza_id
join orders on orders.order_id = orders_details.order_id group by orders.order_date ) as sales;
```

3)Determine the Top 3 pizza types by revenue within each category.

```sql
-- 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((orders_details.quantity)*pizzas.price) as revenue from pizza_types join
pizzas on pizza_types.pizza_type_id=pizzas.pizza_type_id join orders_details on orders_details.pizza_id=pizzas.pizza_id
 group by pizza_types.category,pizza_types.name) as a) as b where rn<=3;
```

gurkirat201005@gmail.com

Pizza hut

Results & Observations

Home About Contact

Key Findings from Analysis:

- Total Orders: 21,350
- Most Common Size: Large (L)
- Highest Revenue Pizza: BBQ Chicken Large
- Top-Selling Pizzas: (list shown from query results)
- Revenue Trends: Peak sales observed on weekends & evenings .

Observations:

- Customers prefer large pizzas more than other sizes.
- Certain pizza types consistently generate higher revenue.
- Sales patterns indicate potential for targeted promotions (e.g., weekend offers).

gurkirat201005@gmail.com

Pizza hut

Conclusion

Project Achievements:

- Successfully analyzed a large pizza sales dataset using SQL.
- Applied joins, aggregate functions, and window functions effectively.
- Identified customer preferences, best-selling pizzas, and revenue trends.
- Gained hands-on experience in real-world data analysis.

SQL is a powerful tool to extract valuable business insights from raw data.

gurkirat201005@gmail.com

Pizza hut

Home About Contact

THANK YOU