PIZZA SALES ANALYSIS

SQL PROJECT

WELCOME TO

PIZZA SALES ANALYSIS

🍕 Pizza Sales Analysis – SQL Project

This project uses MySQL to create and analyze a Pizza Hut sales database. It demonstrates SQL concepts such as database creation, table design, joins, aggregations, window functions, and analytical queries

SQL PROJECT

PIZZA SALES ANALYSIS

VISSION

📂 Database Structure
Database: pizzahut
Tables:
orders → stores order ID, order date, and time.
order_details → stores details of each order (pizza, quantity).
pizzas → contains information about pizza size, type, and price.
pizza_types → contains pizza category and name.

PIZZA SALES ANALYSIS

QUERIES COVERED

The project includes basic, intermediate, and advanced SQL queries:

✅ Basic Queries

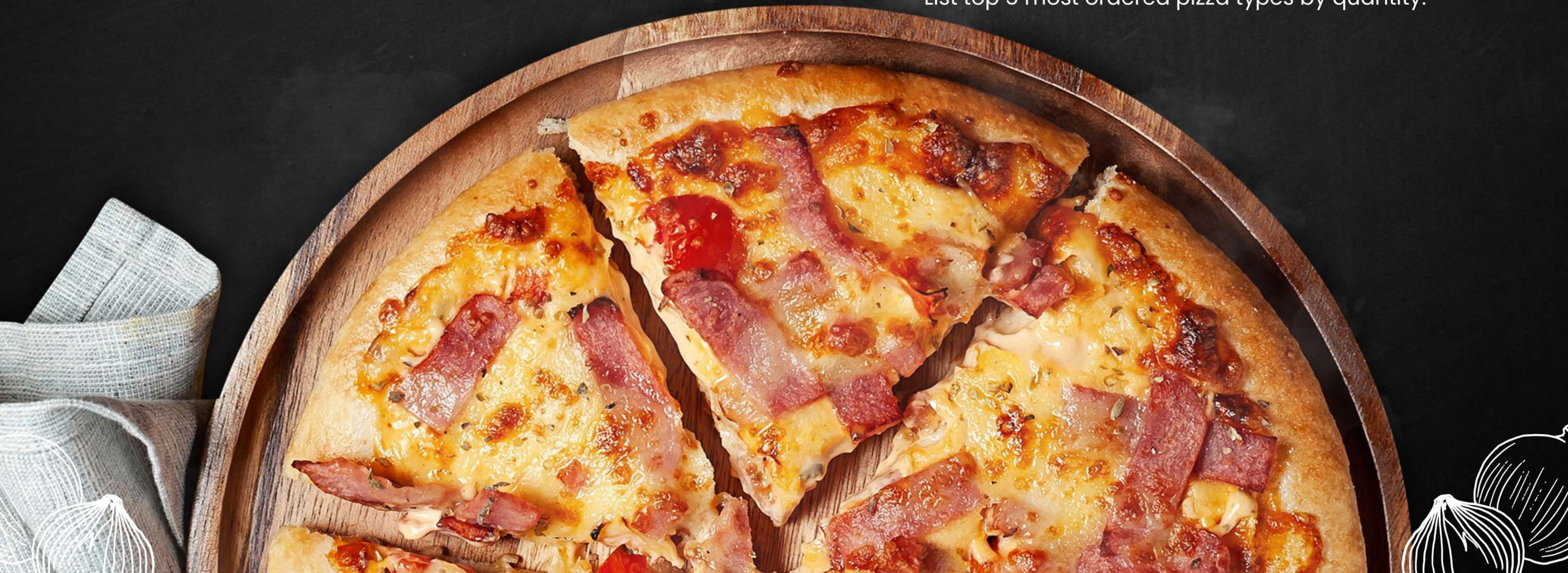
Count total number of orders placed.

Calculate total revenue from pizza sales.

Find the highest-priced pizza.

Identify the most common pizza size ordered.

List top 5 most ordered pizza types by quantity.



LARANA PIZZA

INTERMEDIATE QUERIES

Find total quantity of pizzas sold by category.
Distribution of orders by hour of the day.
Category-wise distribution of pizzas.
Average number of pizzas ordered per day.
Top 3 pizzas based on revenue.

PIZZA SALES ANALYSIS

🛠️ SQL Concepts Used
DDL (CREATE TABLE)
DML (SELECT, JOIN, GROUP BY, ORDER BY, LIMIT)
Aggregate functions → SUM(), COUNT(), AVG(), ROUND()
Date/Time functions → HOUR()
Window functions → RANK(), cumulative SUM() OVER()
Subqueries (inline views for average orders and percentage revenue)

CREATE DATABASE

CREATE TABLE 1

```
CREATE DATABASE pizzahut;
use pizzahut;

CREATE TABLE orders(
        order_id int not null,
        order_date date not null,
        order_time time not null,
        primary key(order_id));


SELECT * FROM orders;
```

CREATE TABLE 2

```sql
CREATE TABLE order_details(
        order_details_id INT NOT NULL,
        order_id INT NOT NULL,
        pizza_id TEXT NOT NULL ,
        quantity INT NOT NULL,
        PRIMARY KEY(order_details_id));


SELECT * FROM order_details;

SELECT * FROM orders;

SELECT * FROM pizza_types;

SELECT * FROM pizzas;
```

QUESTION 1

```sql
-- Basic:
-- Q1) Retrieve the total number of orders placed.

SELECT
    COUNT(order_id)
FROM
    orders;
```

OUTPUT

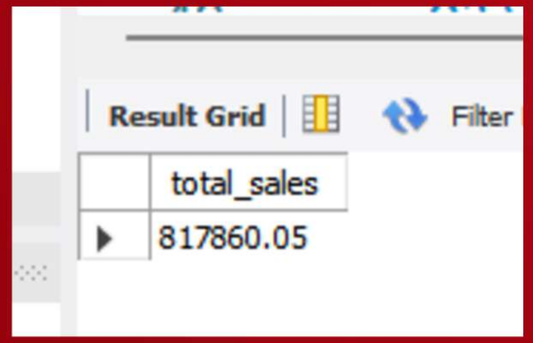
Result Grid | Filter Rows:

	COUNT(order_id)
▶	21350

QUESTION 2

```
37
38    -- 2) Calculate the total revenue generated from pizza sales.
39
40 •  SELECT
41        ROUND(SUM(order_details.quantity * pizzas.price),
42                2) AS total_sales
43    FROM
44        order_details
45            JOIN
46        pizzas ON order_details.pizza_id = pizzas.pizza_id;
47
```

OUTPUT



QUESTION 3

```sql
-- Q3) 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 price DESC
LIMIT 1;
```

OUTPUT

Result Grid | Filter Rows:

name	price
The Greek Pizza	35.95

QUESTION 4

```sql
-- Q4) Identify the most common pizza size ordered.

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

OUTPUT

Result Grid | Filter Rows:

size	order_count
L	18526
M	15385
S	14137
XL	544
XXL	28

QUESTION 5

```sql
-- Q5)  List the top 5 most ordered pizza types along with their quantities.

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

OUTPUT

Result Grid | Filter Rows:

name	quantity
The Classic Deluxe Pizza	2453
The Barbecue Chicken Pizza	2432
The Hawaiian Pizza	2422
The Pepperoni Pizza	2418
The Thai Chicken Pizza	2371

QUESTION 6

```sql
-- * Intermediate * :
-- Q6) Join the necessary tables to find the total quantity of each pizza category ordered.

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

OUTPUT

Result Grid | Filter Rows:

category	quantity
Classic	14888
Supreme	11987
Veggie	11649
Chicken	11050

QUESTION 7

```sql
-- Q7)  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(order_time);
```

OUTPUT

hour	order_count
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
10	8
9	1

QUESTION 8

```sql
-- Q8) Join relevant tables to find the category-wise distribution of pizzas.

SELECT
    category, COUNT(name) AS total_count
FROM
    pizza_types
GROUP BY category;
```

OUTPUT

Result Grid | Filter Rows:

category	total_count
Chicken	6
Classic	8
Supreme	9
Veggie	9

QUESTION 9

```sql
-- Q9) Group the orders by date and calculate the average number of pizzas ordered per day.

SELECT
    ROUND(AVG(quantity), 0) as avg__pizza_order_per_day
FROM
    (SELECT
        orders.order_date, SUM(order_details.quantity) AS quantity
    FROM
        orders
    JOIN order_details ON orders.order_id = order_details.order_id
    GROUP BY orders.order_date) AS order_quantity;
```

OUTPUT

Result Grid | Filter Rows:

	avg__pizza_order_per_day
▶	138

QUESTION 10

```sql
-- Q10) Determine the top 3 most ordered pizza types based on revenue.


SELECT
    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.name
ORDER BY revenue DESC
LIMIT 3;
```

OUTPUT

Result Grid | Filter Rows:

name	revenue
The Thai Chicken Pizza	43434.25
The Barbecue Chicken Pizza	42768
The California Chicken Pizza	41409.5

QUESTION 11

```sql
-- * Advanced: *
-- Q11) Calculate the percentage contribution of each pizza type to total revenue.

select pizza_types.category , round(sum(order_details.quantity * pizzas.price) / (SELECT
round(sum(order_details.quantity * pizzas.price), 2) as total_sales
FROM order_details
join pizzas
on order_details.pizza_id = pizzas.pizza_id) * 100,2)
as revenue
from pizza_types join pizzas
on pizzas.pizza_type_id = pizza_types.pizza_type_id
join order_details
on pizzas.pizza_id = order_details.pizza_id
group by pizza_types.category order by revenue desc;
```

OUTPUT

Result Grid | Filter Rows:

category	revenue
Classic	26.91
Supreme	25.46
Chicken	23.96
Veggie	23.68

QUESTION 12

```sql
-- Q12) 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(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 order_date ) as saleas;
```

OUTPUT

Result Grid | Filter Rows:

order_date	cum_revenue
2015-01-01	2713.850000000004
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
2015-01-07	16560.7
2015-01-08	19399.05
2015-01-09	21526.4
2015-01-10	23990.350000000002
2015-01-11	25862.65
2015-01-12	27781.7
2015-01-13	29831.300000000003

QUESTION 13

```sql
-- Q13) 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 b where rn <= 3;
```

OUTPUT

Result Grid | Filter Rows:

name	revenue
The Thai Chicken Pizza	43434.25
The Barbecue Chicken Pizza	42768
The California Chicken Pizza	41409.5
The Classic Deluxe Pizza	38180.5
The Hawaiian Pizza	32273.25
The Pepperoni Pizza	30161.75
The Spicy Italian Pizza	34831.25
The Italian Supreme Pizza	33476.75
The Sicilian Pizza	30940.5

OUR

🎯LEARNING OUTCOMES

Understand relational database design.
Perform data analysis in SQL.
Apply window functions for advanced insights.
Practice real-world business queries like sales tracking, product performance, and revenue analysis.

SQL PROJRCT

THANK YOU!