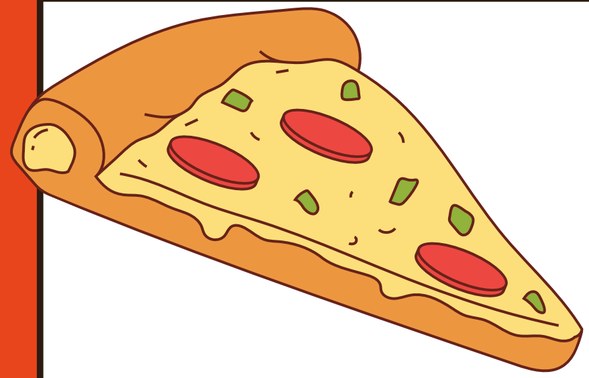
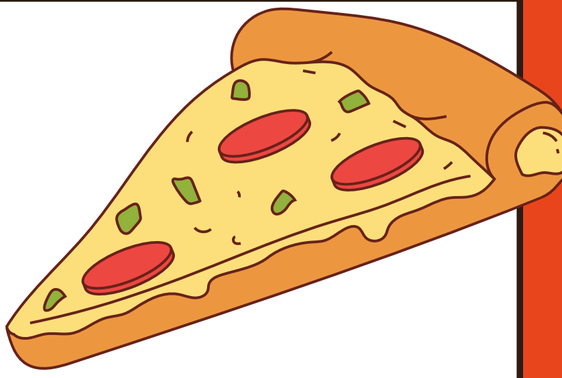
SQL PROJECT ON PIZZA SALES



HELLO !



I am Tamanna Afrin, a student with experience in data analysis and SQL.

This project's primary objective was to examine pizza sales data in order to identify important trends and insights and to assist the company in better understanding sales performance, client preferences, and possible areas for growth.

SCHEMAS

Table: orders

Columns:

Order_id	int PK
Order_date	date
Order_time	time

Table: order_details

Columns:

Order_details_id	int PK
Order_id	int
Pizza_id	text
Quantity	int

Table: pizza_types

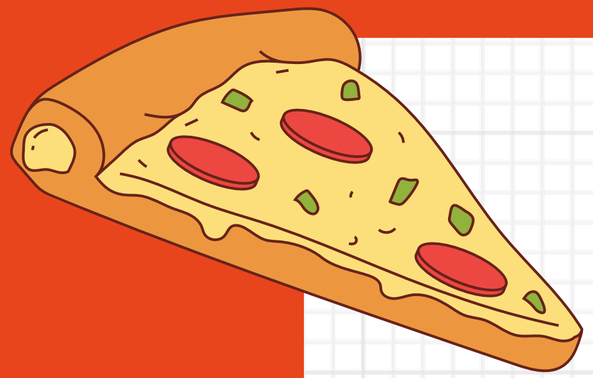
Columns:

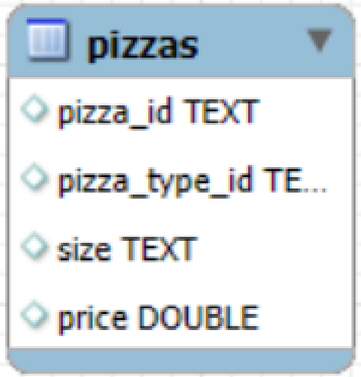
pizza_type_id	text
name	text
category	text
ingredients	text

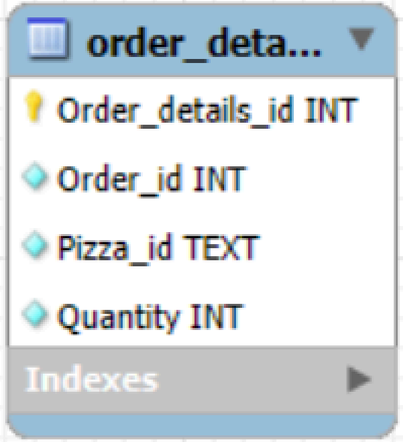
Table: pizzas

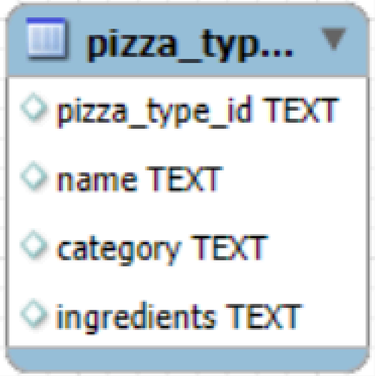
Columns:

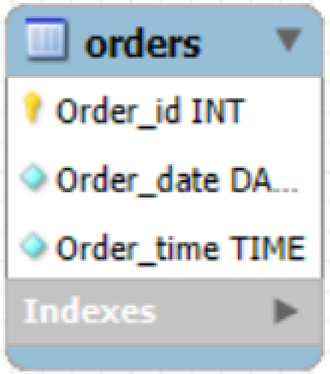
pizza_id	text
pizza_type_id	text
size	text
price	double




pizzas
pizza_id TEXT
pizza_type_id TE...
size TEXT
price DOUBLE

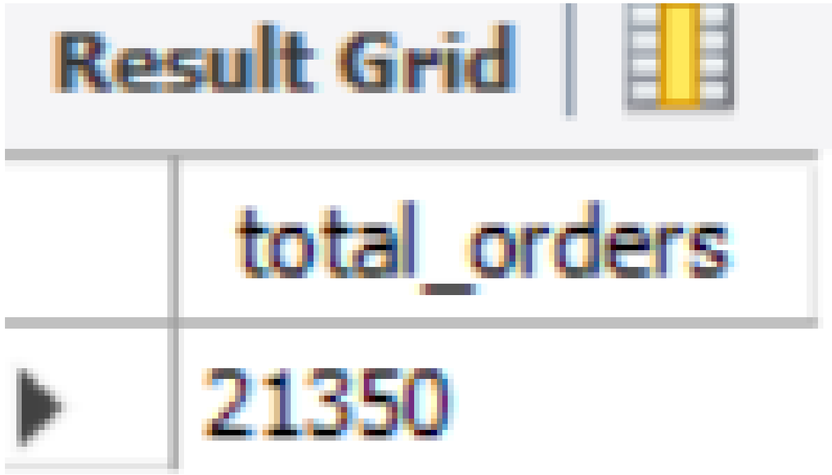

order_deta...
Order_details_id INT
Order_id INT
Pizza_id TEXT
Quantity INT
Indexes


pizza_typ...
pizza_type_id TEXT
name TEXT
category TEXT
ingredients TEXT


orders
Order_id INT
Order_date DA...
Order_time TIME
Indexes

Retrieve the total number of orders placed.

```
SELECT
    *
FROM
    Orders;
SELECT
    COUNT(Order_id) AS total_orders
FROM
    orders;
```



Calculate the total revenue generated from pizza sales.

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

Result Grid

total_sales
817860.05

Identify the highest-priced pizza.

```sql
SELECT * FROM pizzas;
SELECT * FROM pizza_types;


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;
```

Result Grid | Filter Rows:

	name	price
▶	The Greek Pizza	35.95

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;
```

Result Grid | Filter Rows:

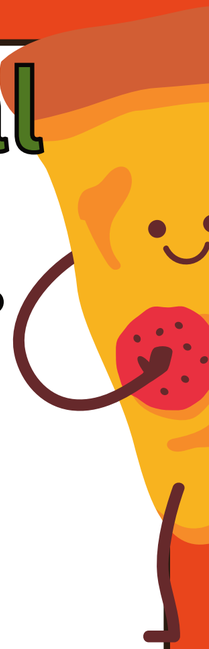
size	order_count
L	18526
M	15385
S	14137
XL	544
XXL	28

List the top 5 most ordered pizza types along with their quantities.

```sql
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;
```

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

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



```sql
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;
```

	category	quantity
▶	Classic	14888
	Supreme	11987
	Veggie	11649
	Chicken	11050

Determine the distribution of orders by hour of the day.

```sql
SELECT * FROM orders;


SELECT
    HOUR(order_time) AS hour, COUNT(Order_id)
FROM
    orders
GROUP BY HOUR(Order_time)
```

hour	COUNT(Order_id)
11	1231
12	2520
13	2455
14	1472
15	1468
16	1920
17	2336
18	2399
19	2009
20	1642
21	1198

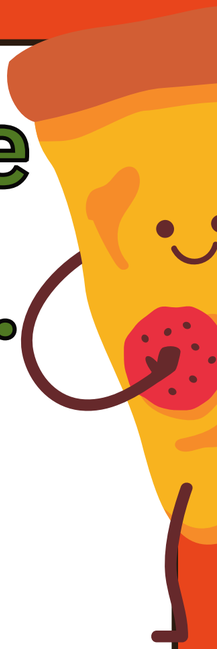
Join relevant tables to find the category-wise distribution of pizzas.

```
SELECT * FROM pizza_types;


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

category	COUNT(name)
Chicken	6
Classic	8
Supreme	9
Veggie	9

Group the orders by date and calculate the average number of pizzas ordered per day.



```sql
SELECT
    AVG(quantity)
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;
```

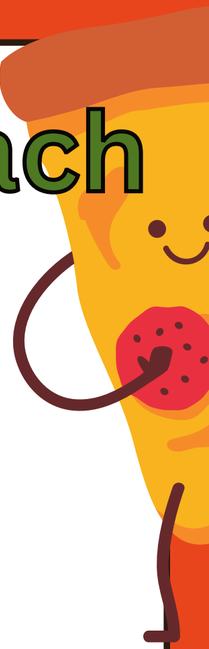
AVG (quantity)
138.4749

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;
```

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

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 total_sales
FROM
    order_details
        JOIN
    pizzas ON pizzas.pizza_id = order_details.pizza_id) *100,2) 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 order by revenue desc;
```

category	revenue
Classic	26.91
Supreme	25.46
Chicken	23.96
Veggie	23.68

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 orders.Order_date) AS sales;
```

order_date	cum_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
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

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

```sql
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;
```

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

THANK YOU :-)