MySQL project on pizza sales

INTRODUCTION

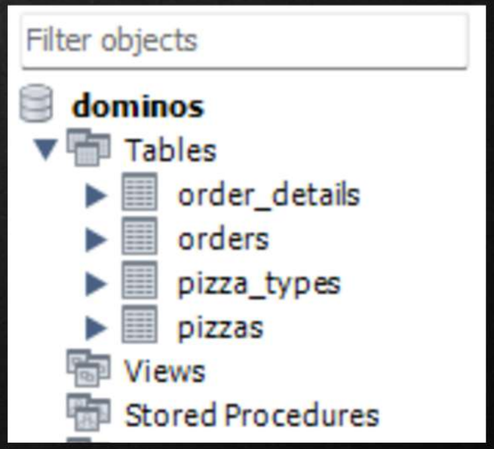
- Hi, my name is Khushi Tyagi, and in this project, I have utilized SQL queries to analyze a pizza sales dataset.
- The analysis covers a range of questions aimed at understanding key metrics and trends in pizza orders.
- This includes:
- Basic sales metrics (e.g., total orders, revenue).
- Intermediate insights (e.g., order distribution by time, category analysis).
- Advanced analysis (e.g., cumulative revenue, top-performing pizza types).
- The objective is to provide valuable business insights that can help optimize pizza sales and customer preferences.

Database Overview

- The pizza sales database consists of multiple interconnected tables that capture details about orders, pizzas, and customer preferences.

Tables in the Database:

- Orders: Contains information about each order placed, including order ID, order date, and order time.
- Order Details: Stores information about the specific pizzas ordered ID, including quantity and pizza ID and order ID.
- Pizzas: Lists all available pizzas, including pizza ID, size,pizza type ID and price.
- Pizza Types: Describes the type ID, category of each pizza (e.g., Vegetarian, Meat, Classic),ingredients and name of the pizzas.



- In this project, I performed a series of SQL queries to analyze the pizza sales data, starting from basic analysis to more intermediate and advanced insights.
- The queries addressed key business questions, helping to uncover important trends and patterns in the data.
- In the following slides, I'll walk you through the SQL queries, highlighting the progression from basic data retrieval and aggregation to intermediate analysis using joins, grouping, and calculations.

RETRIEVE THE TOTAL NUMBER OF ORDERS PLACED.

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

Result Grid

	total_ordres
▶	21350

CALCULATE THE TOTAL REVENUE GENERATED FROM PIZZA SALES.

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

Result Grid

	Revenue
▶	817860.05

DENTIFY THE HIGHEST-PRICED PIZZA.

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

Result Grid | Filter Ro

	name	price
▶	The Greek Pizza	35.95

IDENTIFY THE MOST COMMON PIZZA SIZE ORDERED.

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

sult Grid

size	size_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
        INNER JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
        JOIN
    order_details ON pizzas.pizza_id = order_details.pizza_id
GROUP BY pizza_types.name
ORDER BY quantity DESC
LIMIT 5;
```

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

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
        INNER JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
        INNER JOIN
    order_details ON order_details.pizza_id = pizzas.pizza_id
GROUP BY category
ORDER BY quantity DESC;
```

Result Grid | Filter

category	quantity
Classic	14888
Supreme	11987
Veggie	11649
Chicken	11050

DETERMINE THE DISTRIBUTION OF ORDERS BY HOUR OF THE DAY.

```sql
SELECT
    HOUR(order_time) AS hours, COUNT(order_id) AS order_count
FROM
    orders
GROUP BY hours
ORDER BY order_count DESC;
```

Result Grid | Filter

hours	order_count
12	2520
13	2455
18	2399
17	2336
19	2009
16	1920
20	1642
14	1472

Result Grid | Filt

hours	order_count
15	1468
11	1231
21	1198
22	663
23	28
10	8
9	1

JOIN RELEVANT TABLES TO FIND THE CATEGORY-WISE DISTRIBUTION OF PIZZAS.

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

Result Grid | Filter Rows:

count(pizza_type_id)	category
6	Chicken
8	Classic
9	Supreme
9	Veggie

GROUP THE ORDERS BY DATE AND CALCULATE THE AVERAGE NUMBER OF PIZZAS ORDERED PE DAY.

```sql
SELECT
    AVG(quantity) AS average_quantity
FROM
    (SELECT
        orders.order_date, SUM(order_details.quantity) AS quantity
    FROM
        orders
    INNER JOIN order_details ON orders.order_id = order_details.order_id
    GROUP BY orders.order_date) AS order_quantity;
```

Result Grid

	average_quantity
▶	138.4749

DETERMINE THE TOP 3 MOST ORDERED PIZZA TYPES BASED ON REVENUE.

```sql
SELECT
    pizza_types.name,
    SUM(pizzas.price * order_details.quantity) AS revenue
FROM
    pizzas
        JOIN
    order_details ON pizzas.pizza_id = order_details.pizza_id
        JOIN
    pizza_types ON pizzas.pizza_type_id = pizza_types.pizza_type_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

CALCULATE THE PERCENTAGE CONTRIBUTION OF EACH PIZZA TYPE TO TOTAL REVENUE.

```
    pizza_types.category,
    ROUND((SUM(order_details.quantity * pizzas.price) / (SELECT
                    ROUND(SUM(pizzas.price * order_details.quantity),
                                2) AS Revenue
                FROM
                    pizzas
                        INNER JOIN
                    order_details ON pizzas.pizza_id = order_details.pizza_id)) * 100,
            2) AS total_revenue
FROM
    pizza_types
        INNER JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
        INNER JOIN
    order_details ON pizzas.pizza_id = order_details.pizza_id
```

Result Grid | Filter Ro

category	total_revenue
Classic	26.91
Veggie	23.68
Supreme	25.46
Chicken	23.96

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 pizzas inner join order_details
on pizzas.pizza_id= order_details.pizza_id
inner join orders on order_details.order_id= orders.order_id
group by orders.order_date) as sales;
```

Result Grid | Filter Rows:

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
2015-01-13	29831.300000000003

DETERMINE THE TOP 3 MOST ORDERED PIZZA TYPES BASED ON REVENUE FOR EACH PIZZA CATEGORY.

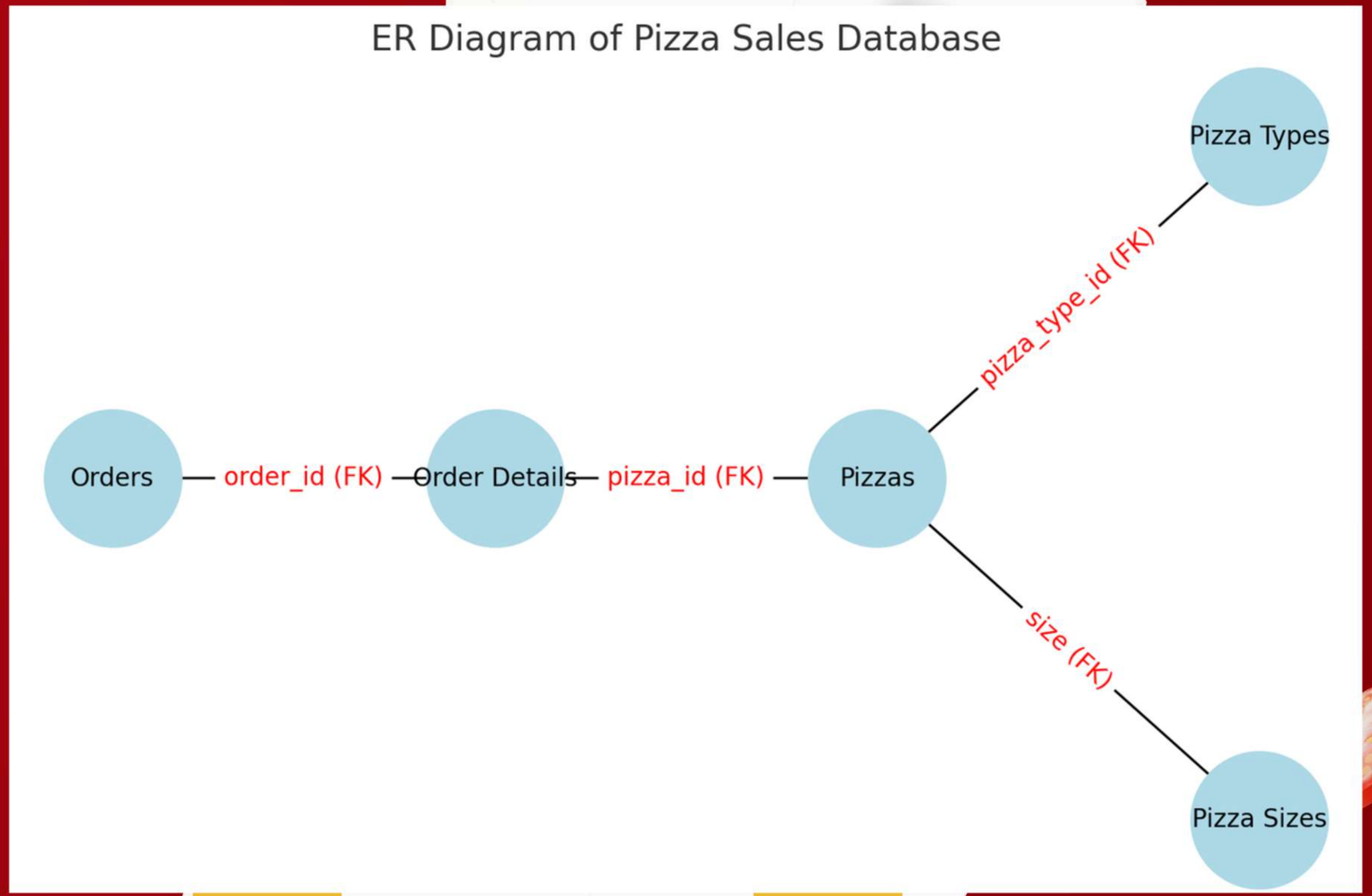
```sql
select name,revenue from
(select category, name,revenue,
rank () over (partition by category order by revenue) as rn
from
(select pizza_types.category, pizza_types.name,
sum(order_details.quantity* pizzas.price) as revenue from pizza_types
inner join pizzas on pizza_types.pizza_type_id= pizzas.pizza_type_id
inner join order_details
on pizzas.pizza_id= order_details.pizza_id group by pizza_types.category ,
pizza_types.name order by revenue desc) as a) as b
where rn <= 3;
```

Result Grid | Filter Rows: | Export:

name	revenue
The Chicken Pesto Pizza	16701.75
The Chicken Alfredo Pizza	16900.25
The Southwest Chicken Pizza	34705.75
The Pepperoni, Mushroom, and Peppers Pizza	18834.5
The Big Meat Pizza	22968
The Napolitana Pizza	24087
The Brie Carre Pizza	11588.4999999999
The Spinach Supreme Pizza	15277.75
The Calabrese Pizza	15934.25
The Green Garden Pizza	13955.75
The Mediterranean Pizza	15360.5
The Spinach Pesto Pizza	15596

ER (ENTITY-RELATIONSHIP) DIAGRAM OF PIZZA SALES DATABASE

- ER diagram shows the structure of the pizza sales database and how the tables are connected:
- Orders Table: Stores order details like order_id and order_date.
- Order Details Table: Links each order with the pizzas ordered using order_id and includes quantity.
- Pizzas Table: Contains details of each pizza, such as pizza_name, size, price, and pizza_type_id.
- Pizza Types Table: Categorizes pizzas (e.g., Vegetarian, Meat) using pizza_type_id.
- Pizza Sizes Table: Lists available pizza sizes (Small, Medium, Large).



- FK- Foreign Key

CONCLUSION

- Total Orders and Revenue: Determined the overall performance of pizza sales.
- Popular Pizza Types and Sizes: Identified customer preferences, highlighting the most frequently ordered pizzas and sizes.
- Category and Revenue Analysis: Evaluated the contribution of each pizza category to total sales, helping identify top-performing items.
- Order Patterns: Analyzed the distribution of orders by hour and date, providing insights into peak sales times.

★★★★★

"Analyzing pizza sales data taught me one important lesson: When in doubt, follow the crust! 🍕 Turns out, data doesn't lie—customers always find the *cheesiest* way to happiness."

mysql project for pizza sales

THANK YOU!