PIZZA SALES PROJECT

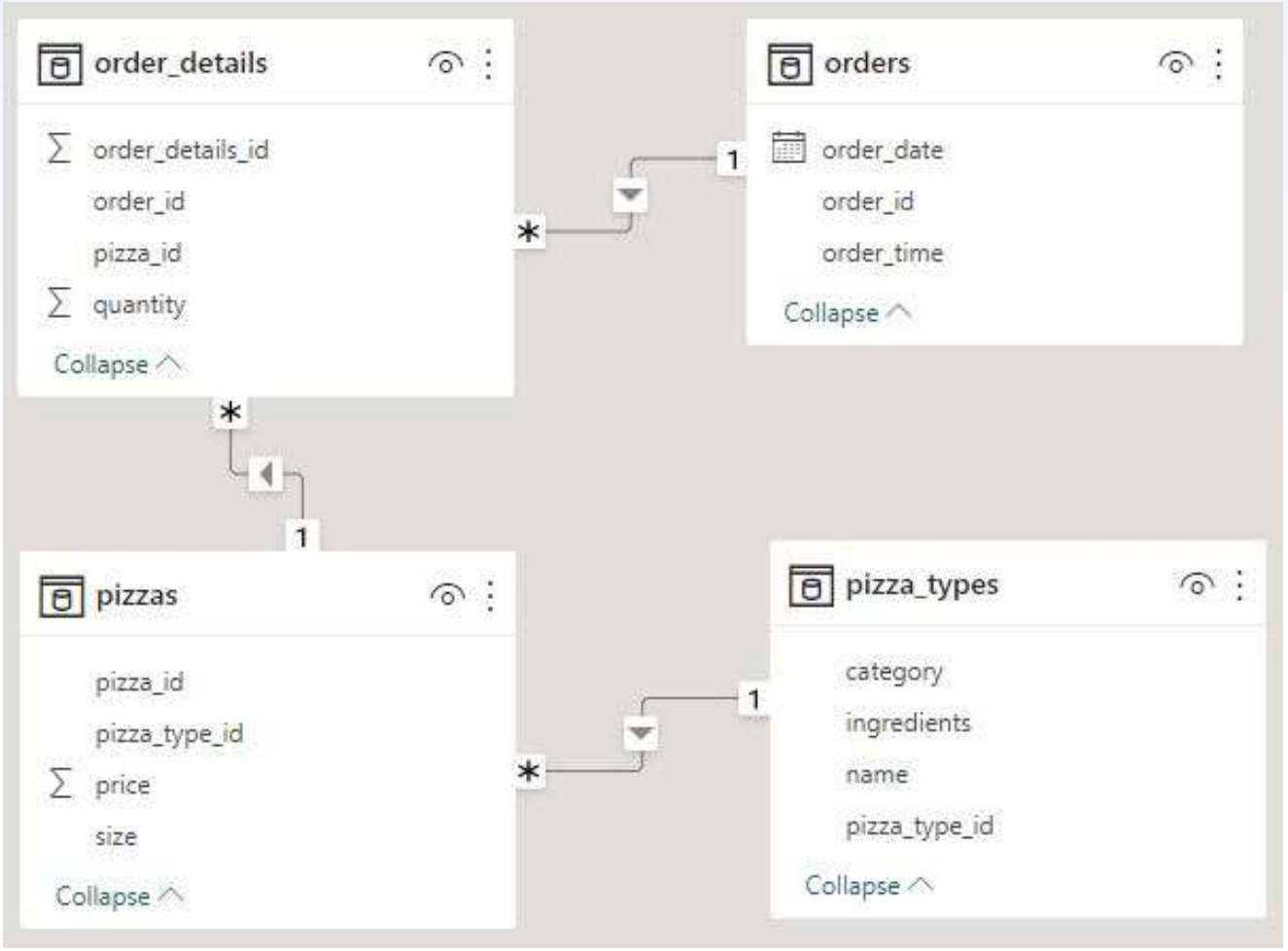
ABOUT MYSELF

Hello everyone
My name is KANIKA BHATT.
In this project I had utilize SQL queries to solve the questions that were related to PIZZA SALES

Here in this project I had used MySql database. MySQL is a widely-used open-source relational database management system (RDBMS) that is renowned for its speed, reliability, and ease of use.

Schema of PIZZA SALES

Established relationships between tables using primary keys and foreign keys



Questions

Basic:

- *Retrieve the total number of orders placed.*
- *Calculate the total revenue generated from pizza sales.*
- *Identify the most common pizza size ordered.*
- *List the top 5 most ordered pizza types along with their quantities.*

Intermediate:

- *Join the necessary tables to find the total quantity of each pizza category ordered.*
- *Determine the distribution of orders by hour of the day.*
- *Join relevant tables to find the category-wise distribution of pizzas.*
- *Group the orders by date and calculate the average number of pizzas ordered per day.*
- *Determine the top 3 most ordered pizza types based on revenue.*

Advanced:

- *Calculate the percentage contribution of each pizza type to total revenue.*
- *Analyze the cumulative revenue generated over time.*
- *Determine the top 3 most ordered pizza types based on revenue for each pizza category.*

SQL queries to retrieve specific data

Retrieve the total number of orders placed

```
SELECT
    COUNT(order_id) AS Total_orders
FROM
    orders
```

Result Grid

	Total_orders
▶	21350

Calculate the total revenue generated from pizza sales

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

Result Grid | Filter

	Total_revenue
▶	817860.05

Identify the most common pizza size ordered

```sql
SELECT
    pizzas.size,
    COUNT(order_details.order_details_id) AS Most_ordered
FROM
    pizzas
        JOIN
    order_details ON order_details.pizza_id = pizzas.pizza_id
GROUP BY 1
ORDER BY 2 DESC
LIMIT 1
```

Result Grid | Filter Rows:

	size	Most_ordered
▶	L	18526

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

```sql
SELECT
    pizza_types.name,
    SUM(order_details.quantity) AS Total_quantity
FROM
    order_details
        JOIN
    pizzas ON pizzas.pizza_id = order_details.pizza_id
        JOIN
    pizza_types ON pizza_types.pizza_type_id = pizzas.pizza_type_id
GROUP BY 1
ORDER BY 2 DESC
LIMIT 5
```

Result Grid | Filter Rows:

name	Total_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 Total_quantity
FROM
    order_details
        JOIN
    pizzas ON pizzas.pizza_id = order_details.pizza_id
        JOIN
    pizza_types ON pizza_types.pizza_type_id = pizzas.pizza_type_id
GROUP BY 1
ORDER BY 2 DESC
```

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
SELECT
    HOUR(orders.order_time) AS Hour,
    COUNT(orders.order_time) AS Order_count
FROM
    orders
GROUP BY 1
```

Result Grid | Filter Rows:

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

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

```
SELECT
    pizza_types.category, COUNT(pizza_types.name) as Pizza_type_count
FROM
    pizza_types
GROUP BY 1
```

Result Grid | Filter Rows:

category	Pizza_type_count
Chicken	6
Classic	8
Supreme	9
Veggie	9

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

```sql
with order_quantity as(
SELECT
    orders.order_date,
    SUM(order_details.quantity) AS Total_quantity
FROM
    order_details
        JOIN
    orders ON order_details.order_id = orders.order_id
GROUP BY 1
)

SELECT
    AVG(Total_quantity)
FROM
    order_quantity
```

Result Grid | Filter Rows:

	AVG(Total_quantity)
▶	138.4749

Determine the top 3 most ordered pizza types based on revenue

```sql
SELECT
    SUM(pizzas.price * order_details.quantity) AS Revenue,
    pizza_types.name
FROM
    pizzas
        JOIN
    order_details ON pizzas.pizza_id = order_details.pizza_id
        JOIN
    pizza_types ON pizza_types.pizza_type_id = pizzas.pizza_type_id
GROUP BY 2
ORDER BY 1 DESC
LIMIT 3
```

Result Grid | Filter Rows:

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

Calculate the percentage contribution of each pizza type to total revenue

```sql
with amount as (
SELECT
    pizza_types.category as Category,
    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 pizza_types.pizza_type_id = pizzas.pizza_type_id
group by 1),
Total_amount as(
SELECT
    ROUND(SUM(pizzas.price * order_details.quantity),2) AS total_revenue
FROM
    order_details
        JOIN
    pizzas ON pizzas.pizza_id = order_details.pizza_id)
SELECT
    r.Category,
    ROUND(r.Revenue / tr.total_revenue * 100, 2) AS Percent
FROM
    amount AS r,
    Total_amount AS tr
ORDER BY 2 DESC
```

Result Grid | Filter Rows:

	Category	Percent
▶	Classic	26.91
	Supreme	25.46
	Chicken	23.96
	Veggie	23.68

Analyze the cumulative revenue generated over time

```sql
with total_revenue as(
SELECT
    orders.order_date as Daily_date,
    round(SUM(order_details.quantity * pizzas.price),0) AS revenue
FROM
    pizzas
        JOIN
    order_details ON order_details.pizza_id = pizzas.pizza_id
        JOIN
    orders ON orders.order_id = order_details.order_id
GROUP BY 1
)

select
    Daily_date,
    sum(revenue) over (order by Daily_date) as Cumulative_revenue
from total_revenue
```

Result Grid | Filter Rows:

Daily_date	Cumulative_revenue
2015-01-01	2714
2015-01-02	5446
2015-01-03	8108
2015-01-04	9863
2015-01-05	11929
2015-01-06	14358
2015-01-07	16560
2015-01-08	19398
2015-01-09	21525
2015-01-10	23989
2015-01-11	25861
2015-01-12	27780
2015-01-13	29830
2015-01-14	32357
2015-01-15	34342
2015-01-16	36936
2015-01-17	39000
2015-01-18	40977
2015-01-19	43364
2015-01-20	45762

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

```sql
with total_revenue as(
SELECT
     SUM(pizzas.price * order_details.quantity) AS Revenue,
     pizza_types.name as Pizza_name,
     pizza_types.category as Category
FROM
    pizzas
        JOIN
    order_details ON pizzas.pizza_id = order_details.pizza_id
        JOIN
    pizza_types ON pizza_types.pizza_type_id = pizzas.pizza_type_id
GROUP BY 2,3
order by 1 desc
),
ranking as(
select
    Category,
    Pizza_name,
    revenue,
    rank() over(partition by Category order by Revenue desc) as Rank_no
from total_revenue
)
select Category, Pizza_name, revenue from ranking
where Rank_No <=3
```

Result Grid | Filter Rows: | Export: | Wrap

Category	Pizza_name	revenue
Chicken	The Thai Chicken Pizza	43434.25
Chicken	The Barbecue Chicken Pizza	42768
Chicken	The California Chicken Pizza	41409.5
Classic	The Classic Deluxe Pizza	38180.5
Classic	The Hawaiian Pizza	32273.25
Classic	The Pepperoni Pizza	30161.75
Supreme	The Spicy Italian Pizza	34831.25
Supreme	The Italian Supreme Pizza	33476.75
Supreme	The Sicilian Pizza	30940.5
Veggie	The Four Cheese Pizza	32265.70100402832
Veggie	The Mexicana Pizza	26780.75
Veggie	The Five Cheese Pizza	26066.5

By leveraging MySQL’s capabilities for data storage, retrieval and management, the pizza sales project aims to streamline operations, enhance customer service, and improve decision-making through detailed analytics

THANK YOU