PIZZA SALES ANALYSIS USING MYSQL

INTRODUCTION

THIS PROJECT FOCUSES ON ANALYZING PIZZA SALES DATA USING MYSQL TO UNDERSTAND CUSTOMER BEHAVIOR, SALES TRENDS, AND BUSINESS PERFORMANCE.
THE DATASET CONTAINS INFORMATION ABOUT ORDERS, PIZZAS, CATEGORIES, SIZES, PRICES, AND ORDER TIME.
BY WRITING SQL QUERIES, MEANINGFUL INSIGHTS WERE EXTRACTED TO HELP THE BUSINESS IMPROVE REVENUE AND DECISION-MAKING.

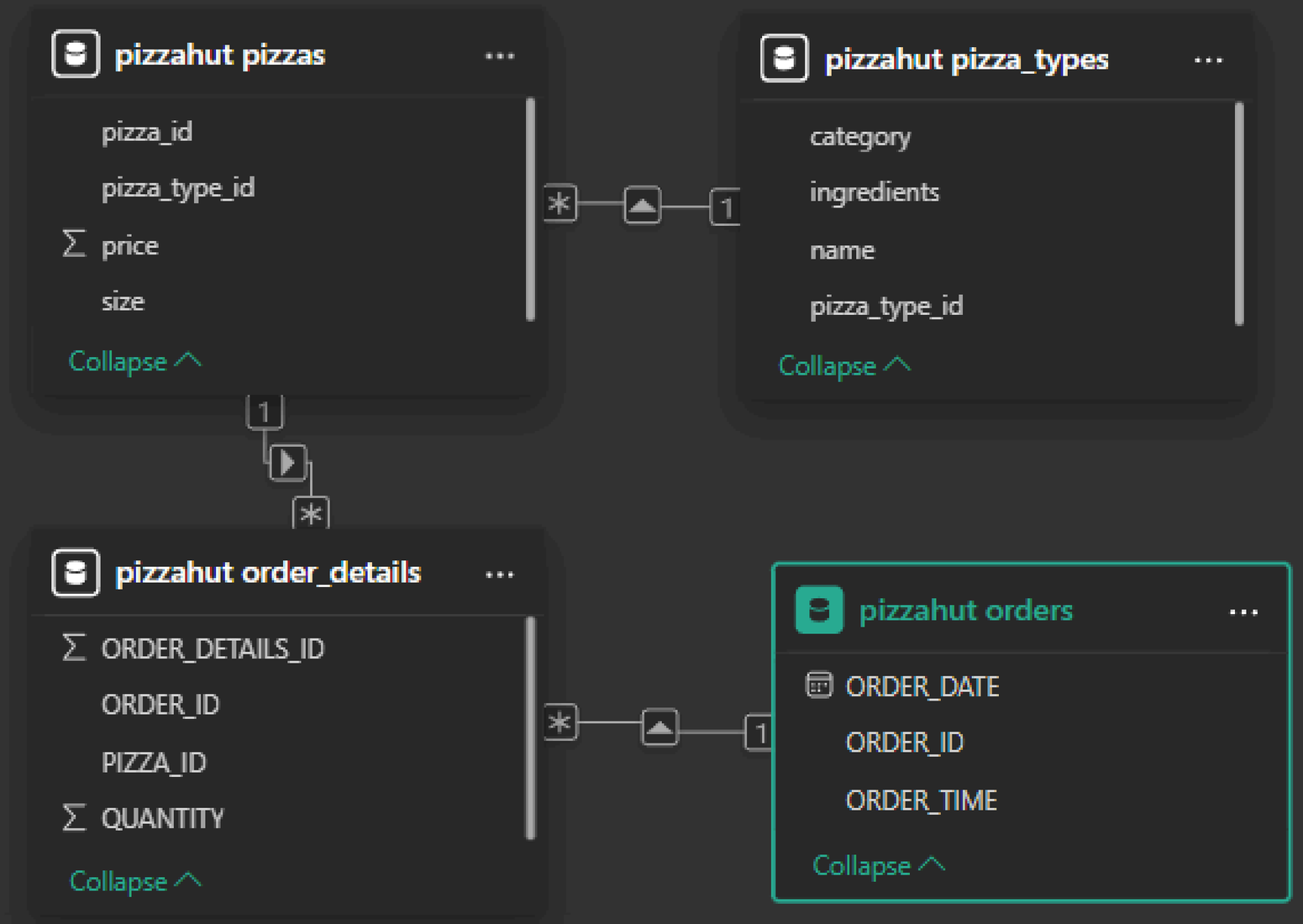
THE ANALYSIS ANSWERS IMPORTANT BUSINESS QUESTIONS SUCH AS:

- TOTAL ORDERS PLACED
- TOTAL REVENUE GENERATED
- MOST POPULAR PIZZA TYPES AND SIZES
- SALES DISTRIBUTION BY TIME AND DATE
- CATEGORY-WISE AND REVENUE-BASED PERFORMANCE

THIS PROJECT DEMONSTRATES PRACTICAL SQL SKILLS INCLUDING:

- DATA RETRIEVAL
- JOINS
- AGGREGATE FUNCTIONS
- GROUPING AND FILTERING BUSINESS-ORIENTED ANALYSIS

DATABASE SCHEMA & TABLE RELATIONSHIP



RETRIEVE THE TOTAL NUMBER OF ORDERS PLACED.

```sql
SELECT
    COUNT(ORDER_ID) AS TOTAL_ORDERS
FROM
    ORDERS;
```

TOTAL_ORDERS
21350

CALCULATE THE TOTAL REVENUE GENERATED FROM PIZZA SALES.

Result Grid

	TOTAL_SALES
▶	817860.05

```sql
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;
```

IDENTIFY THE HIGHEST-PRICED PIZZA.

```sql
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 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 order_count
FROM
    pizzas
        JOIN
    order_details ON pizzas.pizza_id = order_details.PIZZA_ID
GROUP BY pizzas.size
ORDER BY order_count DESC
LIMIT 1;
```

Result Grid

size	order_count
L	18526

LIST THE TOP 5 MOST ORDERED PIZZA TYPES ALONG WITH THEIR QUANTITIES.

```sql
SELECT
    pizza_types.name, SUM(order_details.QUANTITY) AS pizza_count
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 pizza_count DESC
LIMIT 5;
```

Result Grid | Filter Rows:

name	pizza_count
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;
```

Result Grid

category	quantity
Classic	14888
Veggie	11649
Supreme	11987
Chicken	11050

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

```sql
SELECT
    ROUND(AVG(quantity), 0)
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;
```

Result Grid

	round(avg(
▶	138

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

Result Grid

	category	revenue
▶	Classic	26.91
	Supreme	25.46
	Chicken	23.96
	Veggie	23.68

```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;
```

ANALYZE THE CUMULATIVE REVENUE GENERATED OVER TIME.

```sql
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

ANALYSIS BY RAHUL KUMAR

THANK YOU