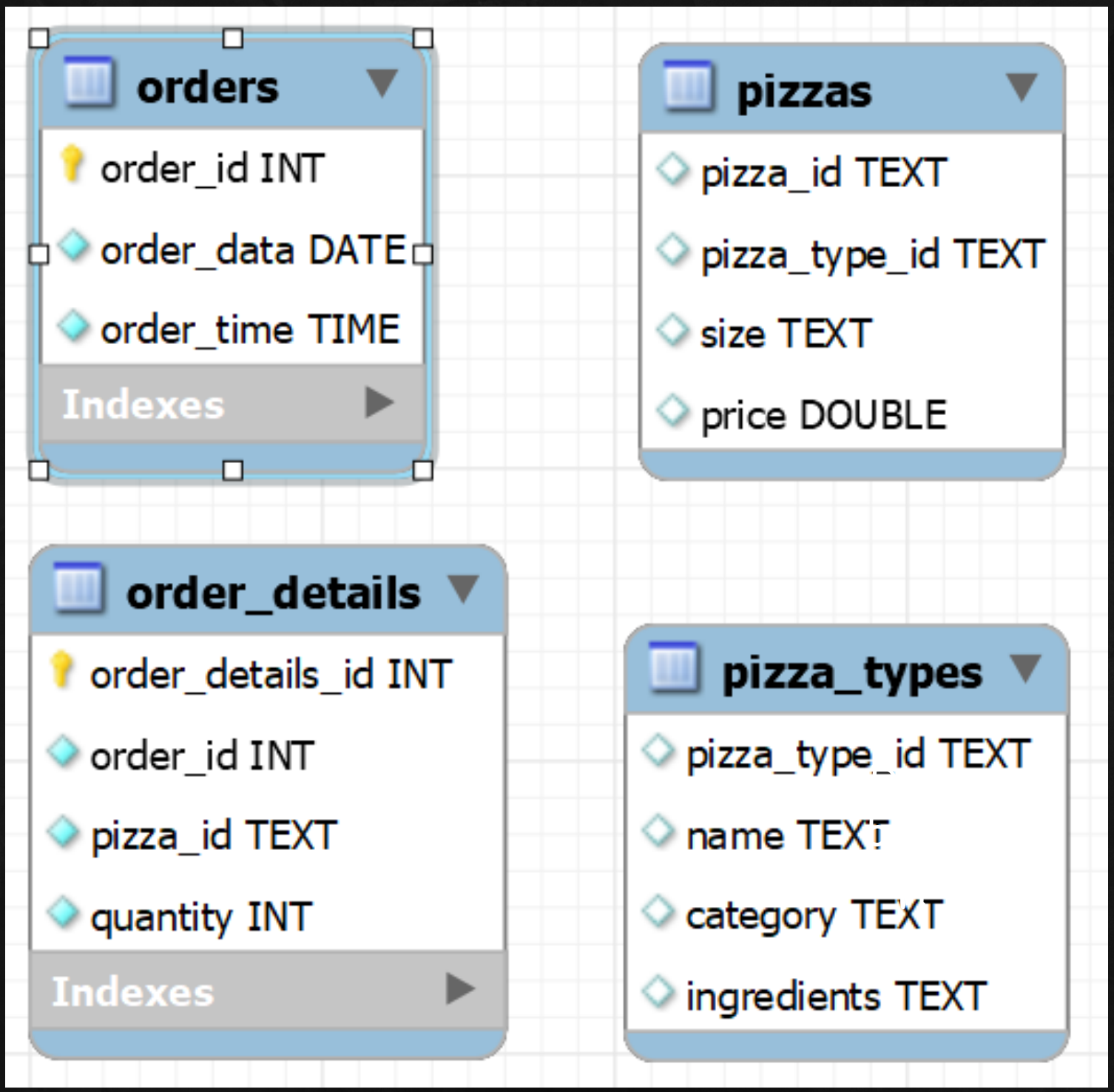
MySQL Analysis PIZZAHUT

I have done this project to understand what is DATA SCIENCE AND THE job of a DATA SCIENTIST

ER DIAGRAM


orders
order_id INT
order_data DATE
order_time TIME
Indexes
pizzas
pizza_id TEXT
pizza_type_id TEXT
size TEXT
price DOUBLE
order_details
order_details_id INT
order_id INT
pizza_id TEXT
quantity INT
Indexes
pizza_types
pizza_type_id TEXT
name TEXT
category TEXT
ingredients TEXT

Questions provided by Ayushi0214

Basic:

- Retrieve the total number of orders placed.
- Calculate the total revenue generated from pizza sales.
- Identify the highest-priced pizza.
- 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.

```
-- Retrieve the total number of orders placed.
SELECT COUNT(*) AS total_orders
FROM orders;

-- Calculate the total revenue generated from pizza
sales.
SELECT ROUND (SUM(pizzas.price *
order_details.quantity),1) AS total_revenue
FROM order_details
JOIN pizzas ON order_details.pizza_id =
pizzas.pizza_id;

-- Identify the highest-priced pizza.
SELECT *
FROM pizzas
WHERE price = (SELECT MAX(price) FROM
pizzas);
```

```
-- Identify the most common pizza size ordered.
SELECT size, COUNT(*) AS total_orders
FROM order_details
JOIN pizzas ON order_details.pizza_id =
pizzas.pizza_id
GROUP BY size
ORDER BY total_orders DESC;

-- List the top 5 most ordered pizza types along with
their quantities.
SELECT pt.name AS pizza_type, SUM(od.quantity)
AS total_quantity
FROM order_details od
JOIN pizzas p ON od.pizza_id = p.pizza_id
JOIN pizza_types pt ON p.pizza_type_id =
pt.pizza_type_id
GROUP BY pt.name
ORDER BY total_quantity DESC
LIMIT 5;
```

```
-- Join the necessary tables to find the total quantity of each pizza category ordered.
SELECT pt.category, SUM(od.quantity) AS total_quantity
FROM order_details od
JOIN pizzas p ON od.pizza_id = p.pizza_id
JOIN pizza_types pt ON p.pizza_type_id = pt.pizza_type_id
GROUP BY pt.category;

-- Determine the distribution of orders by hour of the day.
SELECT HOUR(order_time) AS hour_of_day, COUNT(*) AS order_count
FROM orders
GROUP BY HOUR(order_time)
ORDER BY hour_of_day;

-- Join relevant tables to find the category-wise distribution of pizzas.
SELECT category, COUNT(*) AS pizza_count
FROM pizza_types
GROUP BY category;
```

```
-- Group the orders by date and calculate the
average number of pizzas ordered per day.
SELECT AVG (quantity)
FROM
 (SELECT orders.order_data,
SUM(order_details.quantity) AS quantity
 FROM orders
 JOIN order_details ON orders.order_id =
order_details.order_id
 GROUP BY orders.order_data) AS order_quantity;

-- Determine the top 3 most ordered pizza types
based on revenue.
SELECT pt.name AS pizza_type, SUM(p.price *
od.quantity) AS total_revenue
FROM order_details od
JOIN pizzas p ON od.pizza_id = p.pizza_id
JOIN pizza_types pt ON p.pizza_type_id =
pt.pizza_type_id
GROUP BY pt.name
ORDER BY total_revenue DESC
LIMIT 3;
```

```
-- Calculate the percentage contribution of each
pizza type to total revenue.
SELECT pt.name AS pizza_type,
       SUM(p.price * od.quantity) AS total_revenue,
       SUM(p.price * od.quantity) / (SELECT
SUM(p.price * od.quantity) FROM order_details od
JOIN pizzas p ON od.pizza_id = p.pizza_id) * 100
       AS revenue_percentage
FROM order_details od
JOIN pizzas p ON od.pizza_id = p.pizza_id
JOIN pizza_types pt ON p.pizza_type_id =
pt.pizza_type_id
GROUP BY pt.name;
```

```sql
-- Analyze the cumulative revenue generated over time.
SELECT order_date,
      ROUND(SUM(total_revenue),2) AS cumulative_revenue
FROM (
    SELECT DATE(o.order_data) AS order_date,
          SUM(p.price * od.quantity) AS total_revenue
    FROM order_details od
    JOIN orders o ON od.order_id = o.order_id
    JOIN pizzas p ON od.pizza_id = p.pizza_id
    GROUP BY order_date
) AS daily_revenue
GROUP BY order_date
ORDER BY order_date;
```

```
-- Determine the top 3 most ordered pizza types based on revenue for each pizza category.
SELECT category, pizza_type, total_revenue,
RANK() OVER(partition by category order by total_revenue desc) AS RN
FROM (
 SELECT pt.category,
 pt.name AS pizza_type,
 SUM(p.price * od.quantity) AS total_revenue,
 ROW_NUMBER() OVER(PARTITION BY pt.category ORDER BY SUM(p.price * od.quantity) DESC) AS ranks
 FROM order_details od
 JOIN pizzas p ON od.pizza_id = p.pizza_id
 JOIN pizza_types pt ON p.pizza_type_id = pt.pizza_type_id
 GROUP BY pt.category, pt.name
) AS ranked_pizzas
WHERE ranks <= 3;
```

My analysis questions

Basic:

- Retrieve the total number of orders placed for each month.
- Calculate the average revenue per order.
- Identify the pizza type with the highest total quantity ordered.
- Identify the most common pizza size ordered for each pizza type.
- List the top 5 highest revenue-generating pizza types along with their revenue.

Intermediate:

- Determine the distribution of orders by hour of the day.
- Find the distribution of orders by day of the week.
- Determine the category-wise distribution of pizzas for each size.
- Calculate the average revenue per day.
- Identify the top 3 highest revenue-generating pizza types for each pizza category.

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.
- Calculate the total revenue generated by each pizza size.
- Identify the top 3 highest revenue-generating pizza types for each pizza category, considering only orders made after a specific date.

```sql
-- Retrieve the total number of orders placed for each month.
SELECT MONTH(order_data) AS month, COUNT(*) AS total_orders
FROM orders
GROUP BY MONTH(order_data);

-- Calculate the average revenue per order.
SELECT AVG(total_revenue) AS avg_revenue_per_order
FROM (
    SELECT SUM(p.price * od.quantity) AS total_revenue
    FROM order_details od
    JOIN pizzas p ON od.pizza_id = p.pizza_id
    GROUP BY od.order_id
) AS order_revenues;
```

```
-- Identify the pizza type with the highest total
quantity ordered.
SELECT pt.name AS pizza_type, SUM(od.quantity)
AS total_quantity
FROM order_details od
JOIN pizzas p ON od.pizza_id = p.pizza_id
JOIN pizza_types pt ON p.pizza_type_id =
pt.pizza_type_id
GROUP BY pt.name
ORDER BY total_quantity DESC
LIMIT 1;

-- Identify the most common pizza size ordered for
each pizza type.
SELECT pt.name AS pizza_type, p.size, COUNT(*)
AS total_orders
FROM order_details od
JOIN pizzas p ON od.pizza_id = p.pizza_id
JOIN pizza_types pt ON p.pizza_type_id =
pt.pizza_type_id
GROUP BY pt.name, p.size
ORDER BY pt.name, total_orders DESC;
```

```
-- List the top 5 highest revenue-generating pizza
types along with their revenue.
SELECT pt.name AS pizza_type, SUM(p.price *
od.quantity) AS total_revenue
FROM order_details od
JOIN pizzas p ON od.pizza_id = p.pizza_id
JOIN pizza_types pt ON p.pizza_type_id =
pt.pizza_type_id
GROUP BY pt.name
ORDER BY total_revenue DESC
LIMIT 5;
```

```
-- Determine the distribution of orders by hour of the
day.
SELECT pt.category, SUM(p.price * od.quantity) AS
total_revenue
FROM order_details od
JOIN pizzas p ON od.pizza_id = p.pizza_id
JOIN pizza_types pt ON p.pizza_type_id =
pt.pizza_type_id
GROUP BY pt.category;

-- Find the distribution of orders by day of the week.
SELECT DAYOFWEEK(order_date) AS
day_of_week, COUNT(*) AS order_count
FROM orders
GROUP BY day_of_week
ORDER BY day_of_week;
```

```
-- Determine the category-wise distribution of pizzas for each size.
SELECT pt.category, p.size, COUNT(*) AS pizza_count
FROM pizzas p
JOIN pizza_types pt ON p.pizza_type_id = pt.pizza_type_id
GROUP BY pt.category, p.size;

-- Calculate the average revenue per day.
SELECT order_date, AVG(total_revenue) AS avg_revenue_per_day
FROM (
    SELECT DATE(o.order_date) AS order_date, SUM(p.price * od.quantity) AS total_revenue
    FROM order_details od
    JOIN orders o ON od.order_id = o.order_id
    JOIN pizzas p ON od.pizza_id = p.pizza_id
    GROUP BY order_date
) AS daily_revenue
GROUP BY order_date;
```

```
-- Identify the top 3 highest revenue-generating pizza
types for each pizza category.
SELECT category, pizza_type, total_revenue
FROM (
    SELECT pt.category,
        pt.name AS pizza_type,
        SUM(p.price * od.quantity) AS total_revenue,
        ROW_NUMBER() OVER(PARTITION BY
pt.category ORDER BY SUM(p.price * od.quantity)
DESC) AS ranks
    FROM order_details od
    JOIN pizzas p ON od.pizza_id = p.pizza_id
    JOIN pizza_types pt ON p.pizza_type_id =
pt.pizza_type_id
    GROUP BY pt.category, pt.name
) AS ranked_pizzas
WHERE ranks <= 3;
```

```
-- Calculate the percentage contribution of each
pizza type to total revenue.
SELECT pt.name AS pizza_type,
       SUM(p.price * od.quantity) AS total_revenue,
       (SUM(p.price * od.quantity) / (SELECT
SUM(p.price * od.quantity) FROM order_details od
JOIN pizzas p ON od.pizza_id = p.pizza_id)) * 100
AS revenue_percentage
FROM order_details od
JOIN pizzas p ON od.pizza_id = p.pizza_id
JOIN pizza_types pt ON p.pizza_type_id =
pt.pizza_type_id
GROUP BY pt.name;
```

```
-- Analyze the cumulative revenue generated over
time.
SELECT order_date,
 SUM(total_revenue) OVER (ORDER BY
order_date) AS cumulative_revenue
FROM (
 SELECT DATE(o.order_date) AS order_date,
 SUM(p.price * od.quantity) AS total_revenue
 FROM order_details od
 JOIN orders o ON od.order_id = o.order_id
 JOIN pizzas p ON od.pizza_id = p.pizza_id
 GROUP BY order_date
) AS daily_revenue;
```

```
-- Determine the top 3 most ordered pizza types
based on revenue for each pizza category.
WITH ranked_pizzas AS (
    SELECT pt.category,
        pt.name AS pizza_type,
        SUM(p.price * od.quantity) AS total_revenue,
        ROW_NUMBER() OVER(PARTITION BY
pt.category ORDER BY SUM(p.price * od.quantity)
DESC) AS ranks
    FROM order_details od
    JOIN pizzas p ON od.pizza_id = p.pizza_id
    JOIN pizza_types pt ON p.pizza_type_id =
pt.pizza_type_id
    GROUP BY pt.category, pt.name
)
SELECT category, pizza_type, total_revenue
FROM ranked_pizzas
WHERE ranks <= 3;
```

```
-- Calculate the total revenue generated by each
pizza size.
SELECT size,
       SUM(p.price * od.quantity) AS total_revenue
FROM order_details od
JOIN pizzas p ON od.pizza_id = p.pizza_id
GROUP BY size;
```

```
-- Identify the top 3 highest revenue-generating pizza types for each pizza category, considering only orders made after a specific date.
WITH ranked_pizzas AS (
    SELECT pt.category,
        pt.name AS pizza_type,
        SUM(p.price * od.quantity) AS total_revenue,
        ROW_NUMBER() OVER(PARTITION BY pt.category ORDER BY SUM(p.price * od.quantity) DESC) AS rank
    FROM order_details od
    JOIN pizzas p ON od.pizza_id = p.pizza_id
    JOIN pizza_types pt ON p.pizza_type_id = pt.pizza_type_id
    WHERE o.order_date > '2023-01-01' -- Specify your desired date here
    GROUP BY pt.category, pt.name
)
SELECT category, pizza_type, total_revenue
FROM ranked_pizzas
WHERE rank <= 3;
```

Usama
Computer Engineer

Exploring the field of data science

usama200101010@gmail.com
03099604550

Skills

- **OOP**
- **Data Structure**
- **Database**
- **Python**
- **Data Science**
- **MySQL**
- **JAVA**
- **C++**
- **C**