PIZZA SALES ANALYSIS SQL QUERIES

1. Retrieve the total number of orders placed.

```
SELECT
COUNT(order_id) AS total_orders
FROM
orders;
```

2. Calculate the total revenue generated from pizza sales.

```
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
```

3. Identify the highest-priced pizza.

```
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;
```

PIZZA SALES ANALYSIS SQL QUERIES

4. 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;
```

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

```
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;
```

PIZZA SALES ANALYSIS SQL QUERIES

6. 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
      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;
```

7. Determine the distribution of orders by hour of the day.

```
SELECT
   HOUR(order_time) AS hour, COUNT(order_id) AS order_count
FROM
   orders
GROUP BY HOUR(order_time);
```

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

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

PIZZA SALES ANALYSIS SQL QUERIES

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

```
SELECT
   ROUND(AVG(quantity), 0) as avg_pizza_ordered_per_day
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;
```

10. Determine the top 3 most ordered pizza types based on revenue.

```
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;
```

PIZZA SALES ANALYSIS SQL QUERIES

11. **Calculate the percentage contribution of each pizza type to total revenue**.

```
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;
```

PIZZA SALES ANALYSIS SQL QUERIES

12. 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;
```

PIZZA SALES ANALYSIS SQL QUERIES

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

```
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 ;
```