```sql
-- Q.1 Retrieve the total number of orders placed.
SELECT
    COUNT(order_id)
FROM
    orders;

-- Q.2 Calculate the total revenue generated from pizza sales.
SELECT
    ROUND(SUM(quantity * price), 1) AS Total_revenue
FROM
    order_details o
        JOIN
    pizzas pi ON o.pizza_id = pi.pizza_id;

-- Q.3 Identify the highest-priced pizza.
SELECT
    name, category, ingredients, price
FROM
    pizza_types pt
        JOIN
    pizzas pi ON pt.pizza_type_id = pi.pizza_type_id
ORDER BY price DESC
LIMIT 1;

-- Q.4 Identify the most common pizza size ordered.

SELECT
    size, SUM(quantity) AS total_quantity
FROM
    pizza_types pt
        JOIN
    pizzas pi ON pt.pizza_type_id = pi.pizza_type_id
        JOIN
    order_details o ON pi.pizza_id = o.pizza_id
GROUP BY size
ORDER BY total_quantity DESC
LIMIT 1;

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

SELECT
    name, SUM(quantity) AS total_quantity
FROM
    pizza_types pt
        JOIN
    pizzas pi ON pt.pizza_type_id = pi.pizza_type_id
        JOIN
    order_details o ON pi.pizza_id = o.pizza_id
```

```sql
GROUP BY name
ORDER BY total_quantity DESC
LIMIT 5;

-- Q.6 Join the necessary tables to find the total quantity of each pizza category
  ordered.

SELECT
    category, SUM(quantity) AS total_quantity
FROM
    pizza_types pt
        JOIN
    pizzas pi ON pt.pizza_type_id = pi.pizza_type_id
        JOIN
    order_details o ON pi.pizza_id = o.pizza_id
GROUP BY category
ORDER BY total_quantity DESC
LIMIT 5;


-- Q.6  Determine the distribution of orders by hour of the day.

SELECT
    HOUR(time) AS order_hour, COUNT(*) AS order_count
FROM
    orders
GROUP BY order_hour
ORDER BY order_count DESC;


-- Q.7 Group the orders by date and calculate the average number of pizzas ordered
  per day.
SELECT
    date, ROUND(AVG(order_id)) AS Average_orders
FROM
    orders
GROUP BY date;


-- Q.8 Determine the top 3 most ordered pizza types based on revenue.
SELECT
    name, SUM(quantity * price) AS rev
FROM
    order_details
        LEFT JOIN
    pizzas ON order_details.pizza_id = pizzas.pizza_id
        LEFT JOIN
    pizza_types ON pizza_types.pizza_type_id = pizzas.pizza_type_id
GROUP BY pizza_types.name
```

```sql
ORDER BY rev DESC
LIMIT 3;

-- Q.9 Calculate the percentage contribution of each pizza type to total revenue.
SELECT
    name,
    (SUM(quantity * price) / (SELECT
            ROUND(SUM(order_details.quantity * pizzas.price),
                        2)
        FROM
            order_details
                LEFT JOIN
            pizzas ON order_details.pizza_id = pizzas.pizza_id)) * 100 AS perc_rev
FROM
    order_details
        LEFT JOIN
    pizzas ON order_details.pizza_id = pizzas.pizza_id
        LEFT JOIN
    pizza_types ON pizza_types.pizza_type_id = pizzas.pizza_type_id
GROUP BY pizza_types.name
ORDER BY perc_rev DESC;

-- Q.10 Analyze the cumulative revenue generated over time.
select time, sum(quantity*price) over(order by time) as cum_sum
from order_details left join orders on order_details.order_id=orders.order_id left
  join pizzas on order_details.pizza_id=pizzas.pizza_id;

-- Q.11 Analyze the cumulative revenue generated over date.
select date, sum(sum_r) over (order by date) as cum_rev
from
(select date, sum(quantity*price) as sum_r
from order_details left join orders on order_details.order_id=orders.order_id
left join pizzas on order_details.pizza_id=pizzas.pizza_id
group by date) as sum_rev;

-- Q.12 Determine the top 3 most ordered pizza types based on revenue for each
  pizza category.
select category, name, rev, ranks
from
(select category, name, rev, rank() over(partition by category order by rev desc)
  as ranks
from
(select category, name,  sum(quantity*price) as rev
from order_details left join pizzas on order_details.pizza_id=pizzas.pizza_id
left join pizza_types on pizzas.pizza_type_id=pizza_types.pizza_type_id
group by pizza_types.category, pizza_types.name) as tb) as tr
where ranks <= 3;

-- Q13. Most Frequent Order Date
```

```sql
SELECT
    date, COUNT(order_id) AS TotalOrders
FROM
    orders
GROUP BY date
ORDER BY TotalOrders DESC
LIMIT 1;
```