Q1. Retrieve the total number of orders placed.

```
SELECT
   COUNT(Order_ID) AS Total_Number
FROM
   order_details;
```

Q2. Calculate the total revenue generated from pizza sales.

```
SELECT
   ROUND(SUM(quantity * price), 0) AS Revenue
FROM
   order_details
      JOIN
   pizzas ON order_details.Pizza_ID = pizzas.pizza_id;
```

Q3. Identify the highest-priced pizza.

```
SELECT
   Name, Price
FROM
   pizza_types
      JOIN
   Pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
ORDER BY Price DESC
LIMIT 1;
```

Q4. Identify the most common pizza size ordered.

```
SELECT
   size, COUNT(quantity) AS Order_Pizza
FROM
   order_details
      JOIN
   Pizzas ON order_details.Pizza_ID = pizzas.pizza_id
GROUP BY size
ORDER BY Order_Pizza DESC;
```

Q5. List the top 5 most ordered pizza types along with their quantities.

```
SELECT
   name, SUM(quantity) AS Quantity
FROM
   order_details
      JOIN
   pizzas ON order_details.Pizza_ID = pizzas.pizza_id
      JOIN
   pizza_types ON pizzas.pizza_type_id = pizza_types.pizza_type_id
GROUP BY name
ORDER BY Quantity DESC
LIMIT 5;
```

Q6. Join the necessary tables to find the total quantity of each pizza category ordered.

```
SELECT
   category, SUM(quantity) AS Total_Quantity
FROM
   order_details
      JOIN
   pizzas ON order_details.Pizza_ID = pizzas.pizza_id
      JOIN
   pizza_types ON pizza_types.pizza_type_id = pizzas.pizza_type_id
GROUP BY category;
```

Q7. Determine the distribution of orders by hour of the day.

```
SELECT
   HOUR(order_time) AS Hour, COUNT(order_id) AS Orders
FROM
   orders
GROUP BY Hour
ORDER BY Hour ASC;
```

Q8. Join relevant tables to find the category-wise distribution of pizzas.

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

Q9. Group the orders by date and calculate the average number of pizzas ordered per day.

```
SELECT
   ROUND(AVG(Orders), 0)
FROM
   (SELECT
      Order_Date, SUM(quantity) AS Orders
   FROM
      orders
   JOIN order_details ON orders.order_id = order_details.order_id
   GROUP BY Order_Date) AS Total_orders;
```

Q10. Determine the top 3 most ordered pizza types based on revenue.

```
SELECT
   Name, SUM(quantity * 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  name
   ORDER BY revenue
   DESC LIMIT 3;
```

Q11. Calculate the percentage contribution of each pizza type to total revenue.

```
SELECT
  category, round(SUM(quantity * price) /
  (SELECT
          SUM(quantity * price) AS Revenue
  FROM
   order_details
     JOIN
   pizzas ON order_details.Pizza_ID = pizzas.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  category
   ORDER BY revenue;
```

Q12. Analyze the cumulative revenue generated over time.

```
SELECT
  Order_Date, sum(Revenue) OVER (ORDER BY Order_Date) AS Cum_Revenue
FROM
(SELECT
   order_date, ROUND(SUM(quantity * price),0)AS Revenue
FROM
   pizzas
     JOIN
   order_details ON pizzas.pizza_id = order_details.pizza_id
     JOIN
   orders ON orders.order_id = order_details.Order_ID
GROUP BY order_date) as sales;
```

Q13. Determine the top 3 most ordered pizza types based on revenue for each pizza category.

```
  SELECT
   name, T_Revenue
   FROM
(SELECT
   category, name, T_Revenue,
RANK() OVER (PARTITION BY Category ORDER BY T_Revenue DESC) AS Revenue
FROM
(SELECT
   category, name, SUM(quantity * price) AS T_Revenue
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 category, name)
   AS a) AS B WHERE Revenue <=3;
```