1. Which product has the highest price? Only return a single row.

```sql
--1) Which product has the highest price? Only return a single row.
SELECT *
FROM   products
WHERE  price = (SELECT Max(price)
                FROM   products);


```

Query #1 **Execution time: 11ms**

product_id	product_name	price
13	Product M	70.00

2. Which customer has made the most orders?

```sql
--2) Which customer has made the most orders?
WITH cte
     AS (SELECT customer_id,
                first_name,
                last_name,
                Count(order_id)                    AS total_orders,
                Rank()
                  OVER (
                    ORDER BY Count(order_id) DESC) ranking
         FROM   customers
                JOIN orders using(customer_id)
         GROUP  BY 1,
                   2,
                   3)
SELECT customer_id,
       first_name,
       last_name,
       total_orders
FROM   cte
WHERE  ranking = 1;
```

Query #2 **Execution time: 1ms**

customer_id	first_name	last_name	total_orders
2	Jane	Smith	2
3	Bob	Johnson	2
1	John	Doe	2

3. What's the total revenue per product?

```sql
--3) What's the total revenue per product?
SELECT product_id,
       product_name,
       Sum(quantity * price) AS total_revenue
FROM   order_items
       JOIN products USING(product_id)
GROUP  BY 1,
          2
ORDER  BY 1;
```

Query #3 **Execution time: 0ms**

product_id	product_name	total_revenue
1	Product A	50.00
2	Product B	135.00
3	Product C	160.00
4	Product D	75.00
5	Product E	90.00
6	Product F	210.00
7	Product G	120.00

4. Find the day with the highest revenue.

```
44 --4) Find the day with the highest revenue.
45 SELECT Extract(day FROM order_date) AS "Day",
46        Sum(quantity * price)        AS "Revenue"
47 FROM   orders
48        JOIN order_items USING(order_id)
49        JOIN products USING(product_id)
50 GROUP  BY 1
51 ORDER  BY 2 DESC
52 LIMIT  1;
53
```

Query #4 **Execution time: 1ms**

Day	Revenue
16	340.00

5. Find the first order (by date) for each customer.

```sql
--5) Find the first order (by date) for each customer.
WITH cte
     AS (SELECT customer_id,
                first_name,
                last_name,
                order_id,
                order_date,
                product_id,
                Dense_rank()
                  OVER (
                    partition BY customer_id
                    ORDER BY order_date) AS order_rank
         FROM   orders
                JOIN customers using(customer_id)
                JOIN order_items using(order_id))
SELECT customer_id,
       first_name,
       last_name,
       order_id,
       order_date,
       product_id
FROM   cte
WHERE  order_rank = 1;

```

Query #5 **Execution time: 1ms**

customer_id	first_name	last_name	order_id	order_date	product_id
1	John	Doe	1	2023-05-01T00:00:00.000Z	2
1	John	Doe	1	2023-05-01T00:00:00.000Z	1
2	Jane	Smith	2	2023-05-02T00:00:00.000Z	2
2	Jane	Smith	2	2023-05-02T00:00:00.000Z	3

6. Find the top 3 customers who have ordered the most distinct products

```sql
--6) Find the top 3 customers who have ordered the most distinct products
WITH cte
     AS (SELECT customer_id,
                Count(DISTINCT product_id)                    AS
                distinct_product_count,
                Rank()
                  OVER (
                    ORDER BY Count(DISTINCT product_id) DESC) AS ranking
         FROM   orders
                JOIN order_items using(order_id)
         GROUP  BY 1)
SELECT customer_id,
       distinct_product_count
FROM   cte
WHERE  ranking <= 3;

```

Query #6 **Execution time: 0ms**

customer_id	distinct_product_count
1	3
2	3
3	3

7. Which product has been bought the least in terms of quantity?

```sql
--7) Which product has been bought the least in terms of quantity?
WITH cte
     AS (SELECT product_id,
                product_name,
                Sum(quantity)              AS qty_bought,
                Rank()
                  OVER (
                    ORDER BY Sum(quantity)) AS ranking
         FROM   order_items
                JOIN products using(product_id)
         GROUP  BY 1,
                   2)
SELECT product_id,
       product_name,
       qty_bought
FROM   cte
WHERE  ranking = 1;
```

Query #7 **Execution time: 1ms**

product_id	product_name	qty_bought
7	Product G	3
5	Product E	3
4	Product D	3
11	Product K	3

8. What is the median order total?

```
118 --8) What is the median order total?
119 WITH cte
120      AS (SELECT order_id,
121                 SUM(quantity * price) AS total_price
122          FROM   order_items
123                 join products USING(product_id)
124          GROUP  BY 1)
125 SELECT Percentile_cont(0.5)
126          within GROUP (ORDER BY total_price) AS median_order_total
127 FROM   cte;
128
```

Query #8 **Execution time: 0ms**

median_order_total
112.5

9. For each order, determine if it was ‘Expensive’ (total over 300), ‘Affordable’ (total over 100), or ‘Cheap’.

```sql
--9) For each order, determine if it was ‘Expensive’ (total over 300),
‘Affordable’ (total over 100), or ‘Cheap’.
WITH cte
     AS (SELECT order_id,
                Sum(quantity * price) AS order_total
         FROM   order_items
                JOIN products using(product_id)
         GROUP  BY 1)
SELECT order_id,
       order_total,
       CASE
         WHEN order_total > 300 THEN 'Expensive'
         WHEN order_total > 100
              AND order_total <= 300 THEN 'Affordable'
         ELSE 'Cheap'
       END category
FROM   cte;

```

Query #9 **Execution time: 1ms**

order_id	order_total	category
11	275.00	Affordable
9	140.00	Affordable
15	225.00	Affordable
3	50.00	Cheap
5	50.00	Cheap

10. Find customers who have ordered the product with the highest price.

```sql
--10) Find customers who have ordered the product with the highest price.
SELECT customer_id,
       first_name,
       last_name,
       product_id,
       price AS highest_price
FROM   customers
       JOIN orders USING(customer_id)
       JOIN order_items USING(order_id)
       JOIN products USING(product_id)
WHERE  price = (SELECT Max(price)
                FROM   products);

```

Query #10 **Execution time: 0ms**

customer_id	first_name	last_name	product_id	highest_price
8	Ivy	Jones	13	70.00
13	Sophia	Thomas	13	70.00