Schema (PostgreSQL v15)

Case Study Questions, Answer Queries and Output

Query #1: *Which product has the highest price? Only return a single row.*

```
SELECT * FROM products
ORDER BY price DESC
LIMIT 1;
```

product_id	product_name	price
13	Product M	70.00

Query #2: *Which customer has made the most orders?*

```
WITH tmp AS (SELECT customer_id, COUNT(order_id) as total_orders,
             DENSE_RANK() OVER (ORDER BY COUNT(order_id) DESC) rnk
FROM orders
GROUP BY customer_id)
SELECT customer_id, total_orders FROM tmp
WHERE rnk = 1
ORDER BY customer_id;
```

customer_id	total_orders
1	2
2	2
3	2

Query #3: *What's the total revenue per product?*

```
WITH prod_total AS (SELECT product_id, SUM(quantity) as total_quantity
                    FROM order_items
                    GROUP BY product_id
                    ORDER BY product_id)
SELECT p.product_id,
CASE
WHEN pt.total_quantity IS NOT NULL THEN pt.total_quantity * p.price
ELSE 0
END as total_revenue
FROM products as p
LEFT JOIN prod_total as pt
```

```
ON p.product_id = pt.product_id
ORDER BY product_id;
```

product_id	total_revenue
1	50.00
2	135.00
3	160.00
4	75.00
5	90.00
6	210.00
7	120.00
8	135.00
9	150.00
10	330.00
11	180.00
12	195.00
13	420.00

Query #4: *Find the day with the highest revenue.*

```
-- Option 1
SELECT order_date, day_revenue FROM (WITH
                                    order_revenue AS
                                    (SELECT o.order_id, SUM(o.quantity * p.price)
as order_revenue
                                    FROM order_items as o
                                    LEFT JOIN products as p
                                    ON o.product_id = p.product_id
                                    GROUP BY o.order_id)
SELECT od.order_date, SUM(ore.order_revenue) as day_revenue,
DENSE_RANK() OVER (ORDER BY SUM(ore.order_revenue) DESC) as rnk
FROM orders as od
LEFT JOIN order_revenue as ore
ON od.order_id = ore.order_id
GROUP BY od.order_date) tmp
WHERE rnk = 1;
```

order_date	day_revenue

order_date	day_revenue
2023-05-16T00:00:00.000Z	340.00

```
-- Option 2
WITH ord_cost as (SELECT oi.order_id, SUM(oi.quantity*p.price) as order_total
                FROM order_items as oi
                LEFT JOIN products as p
                ON oi.product_id = p.product_id
                GROUP BY oi.order_id)
SELECT day, sum FROM (SELECT to_char(o.order_date, 'day') as day,
SUM(oc.order_total) as sum, DENSE_RANK() OVER (ORDER BY SUM(oc.order_total) DESC)
as rnk
FROM orders as o
LEFT JOIN ord_cost as oc
ON o.order_id = oc.order_id
GROUP BY 1
ORDER BY 2 DESC)tmp WHERE rnk = 1;


-- Option 3
SELECT  to_char(o.order_date, 'day') as Day, sum(p.price*oi.quantity) as Revenue
FROM products p
join order_items oi
on p.product_id=oi.product_id
join orders o
on o.order_id=oi.order_id
group by 1
order by 2 DESC
LIMIT 1
```

Day	Revenue
tuesday	555.00

Query #5: *Find the first order (by date) for each customer.*

```
SELECT customer_id, MIN(order_date) as first_order
               FROM orders
GROUP BY customer_id
ORDER BY customer_id;
```

customer_id	first_order
1	2023-05-01T00:00:00.000Z

customer_id	first_order
2	2023-05-02T00:00:00.000Z
3	2023-05-03T00:00:00.000Z
4	2023-05-07T00:00:00.000Z
5	2023-05-08T00:00:00.000Z
6	2023-05-09T00:00:00.000Z
7	2023-05-10T00:00:00.000Z
8	2023-05-11T00:00:00.000Z
9	2023-05-12T00:00:00.000Z
10	2023-05-13T00:00:00.000Z
11	2023-05-14T00:00:00.000Z
12	2023-05-15T00:00:00.000Z
13	2023-05-16T00:00:00.000Z

Query #6: *Find the top 3 customers who have ordered the most distinct products*

```
SELECT o.customer_id, COUNT(DISTINCT oi.product_id) as total_distinct_prods
FROM orders as o
LEFT JOIN order_items as oi
ON o.order_id = oi.order_id
GROUP BY customer_id
ORDER BY total_distinct_prods DESC
LIMIT 3;
```

customer_id	total_distinct_prods
2	3
3	3
1	3

Query #7: *Which product has been bought the least in terms of quantity?*

```
SELECT product_id, total_quantity FROM (SELECT product_id, SUM(quantity) as
total_quantity,
DENSE_RANK() OVER (ORDER BY  SUM(quantity) ASC) as rnk
from order_items
GROUP BY product_id) tmp
```

```
WHERE rnk = 1
ORDER BY product_id;
```

product_id	total_quantity
4	3
5	3
7	3
8	3
9	3
11	3
12	3

Query #8: *What is the median order total?*

```
WITH order_sum AS (SELECT o.order_id, SUM(o.quantity * p.price) as order_total
FROM order_items as o
LEFT JOIN products as p
ON o.product_id = p.product_id
GROUP BY o.order_id)
SELECT PERCENTILE_CONT(0.5) WITHIN GROUP(ORDER BY order_total) as
median_order_total
FROM order_sum;
```

median_order_total
112.5

Query #9: *For each order, determine if it was 'Expensive' (total over 300), 'Affordable' (total over 100), or 'Cheap'.*

```
WITH order_sum AS (SELECT oi.order_id, SUM(p.price * oi.quantity) as order_total
FROM order_items as oi
LEFT JOIN products as p
ON oi.product_id = p.product_id
GROUP BY oi.order_id)
SELECT *,
CASE
WHEN order_total > 300 THEN 'Expensive'
WHEN order_total > 100 AND order_total <= 300 THEN 'Affordable'
WHEN order_total <= 100 THEN 'Cheap'
END as order_type
```

```
FROM order_sum
ORDER BY order_id;
```

order_id	order_total	order_type
1	35.00	Cheap
2	75.00	Cheap
3	50.00	Cheap
4	80.00	Cheap
5	50.00	Cheap
6	55.00	Cheap
7	85.00	Cheap
8	145.00	Affordable
9	140.00	Affordable
10	285.00	Affordable
11	275.00	Affordable
12	80.00	Cheap
13	185.00	Affordable
14	145.00	Affordable
15	225.00	Affordable
16	340.00	Expensive

Query #10: *Find customers who have ordered the product with the highest price.*

```
WITH high_val_prod as (SELECT product_id, DENSE_RANK() OVER (ORDER BY price DESC)
as rnk
FROM products)
SELECT o.customer_id , c.first_name, c.last_name, c.email
FROM customers as c
INNER JOIN orders as o
ON c.customer_id = o.customer_id
LEFT JOIN order_items as oi
ON o.order_id = oi.order_id
WHERE oi.product_id IN (SELECT product_id
FROM high_val_prod
WHERE rnk =1
);
```

customer_id	first_name	last_name	email
8	Ivy	Jones	ivyjones@email.com
13	Sophia	Thomas	sophiathomas@email.com

View on DB Fiddle