SQL CASE STUDY
TINY SHOP SALES

PRESENTED BY - SHIKHAR CHOPRA

OVERVIEW

This case study uses PostgreSQL. To successfully answer all the questions you should have been exposed to the following areas of SQL:

Basic aggregations

CASE WHEN statements

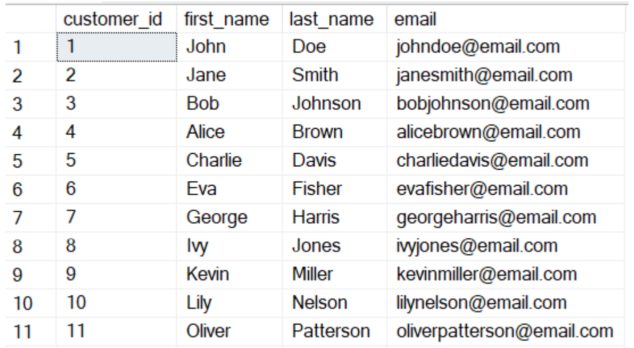
Window Functions

Joins

Date time functions

CTEs

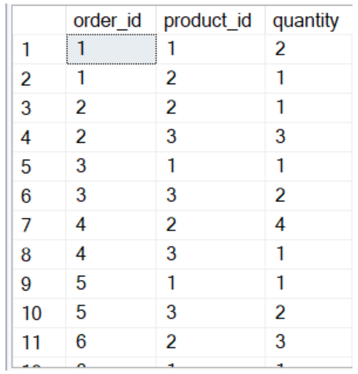
DATA



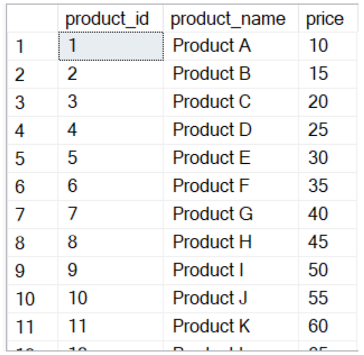
	customer_id	first_name	last_name	email
1	1	John	Doe	johndoe@email.com
2	2	Jane	Smith	janesmith@email.com
3	3	Bob	Johnson	bobjohnson@email.com
4	4	Alice	Brown	alicebrown@email.com
5	5	Charlie	Davis	charliedavis@email.com
6	6	Eva	Fisher	evafisher@email.com
7	7	George	Harris	georgeharris@email.com
8	8	Ivy	Jones	ivyjones@email.com
9	9	Kevin	Miller	kevinmiller@email.com
10	10	Lily	Nelson	lilynelson@email.com
11	11	Oliver	Patterson	oliverpatterson@email.com

Customers Table

Order_items Table



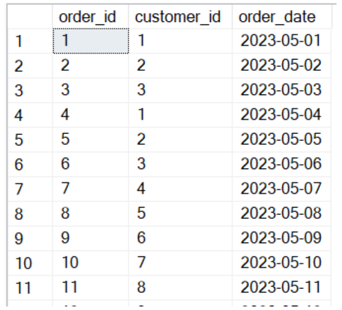
	order_id	product_id	quantity
1	1	1	2
2	1	2	1
3	2	2	1
4	2	3	3
5	3	1	1
6	3	3	2
7	4	2	4
8	4	3	1
9	5	1	1
10	5	3	2
11	6	2	3



	product_id	product_name	price
1	1	Product A	10
2	2	Product B	15
3	3	Product C	20
4	4	Product D	25
5	5	Product E	30
6	6	Product F	35
7	7	Product G	40
8	8	Product H	45
9	9	Product I	50
10	10	Product J	55
11	11	Product K	60

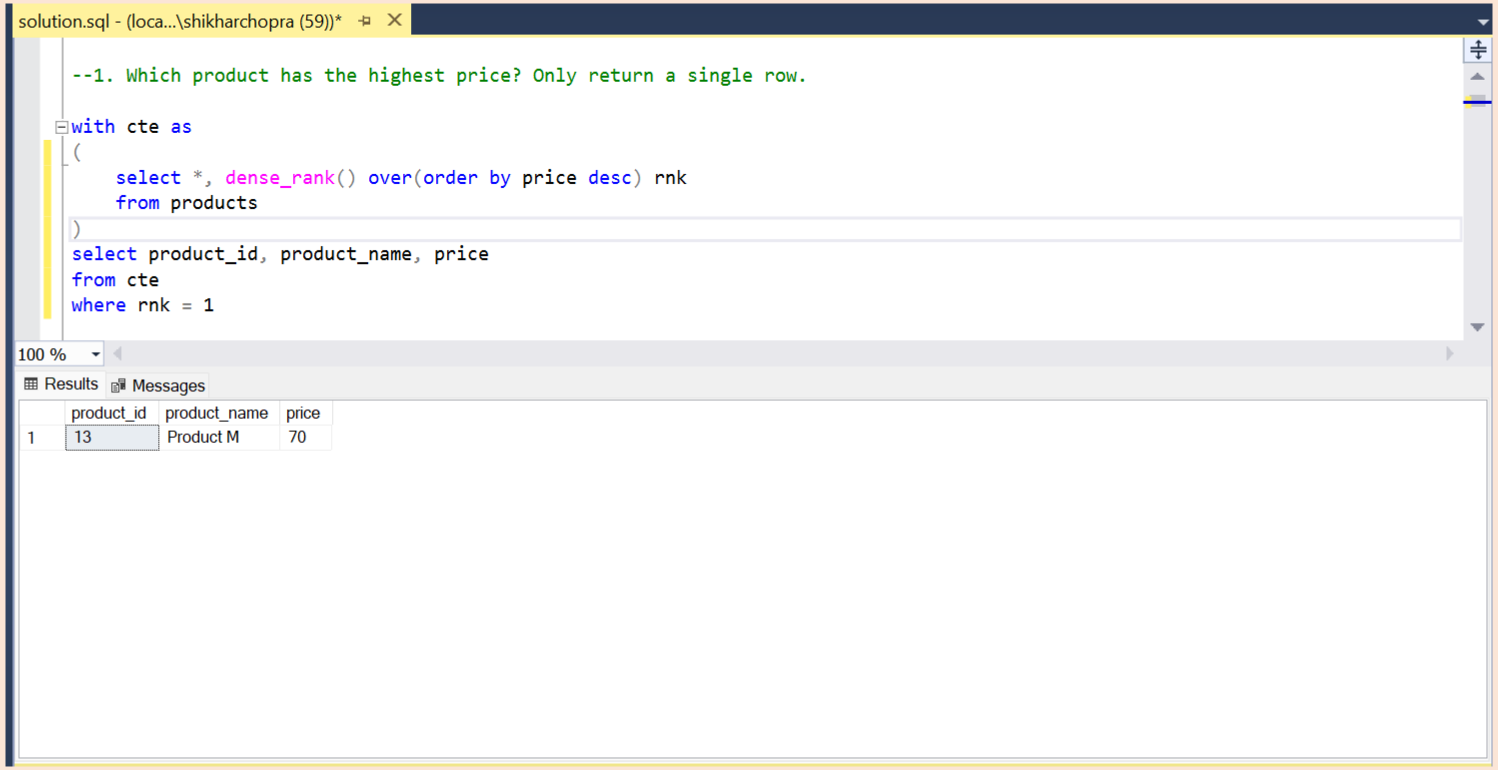
Products Table

Orders Table



	order_id	customer_id	order_date
1	1	1	2023-05-01
2	2	2	2023-05-02
3	3	3	2023-05-03
4	4	1	2023-05-04
5	5	2	2023-05-05
6	6	3	2023-05-06
7	7	4	2023-05-07
8	8	5	2023-05-08
9	9	6	2023-05-09
10	10	7	2023-05-10
11	11	8	2023-05-11

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



solution.sql - (loca...\shikharchopra (59))*

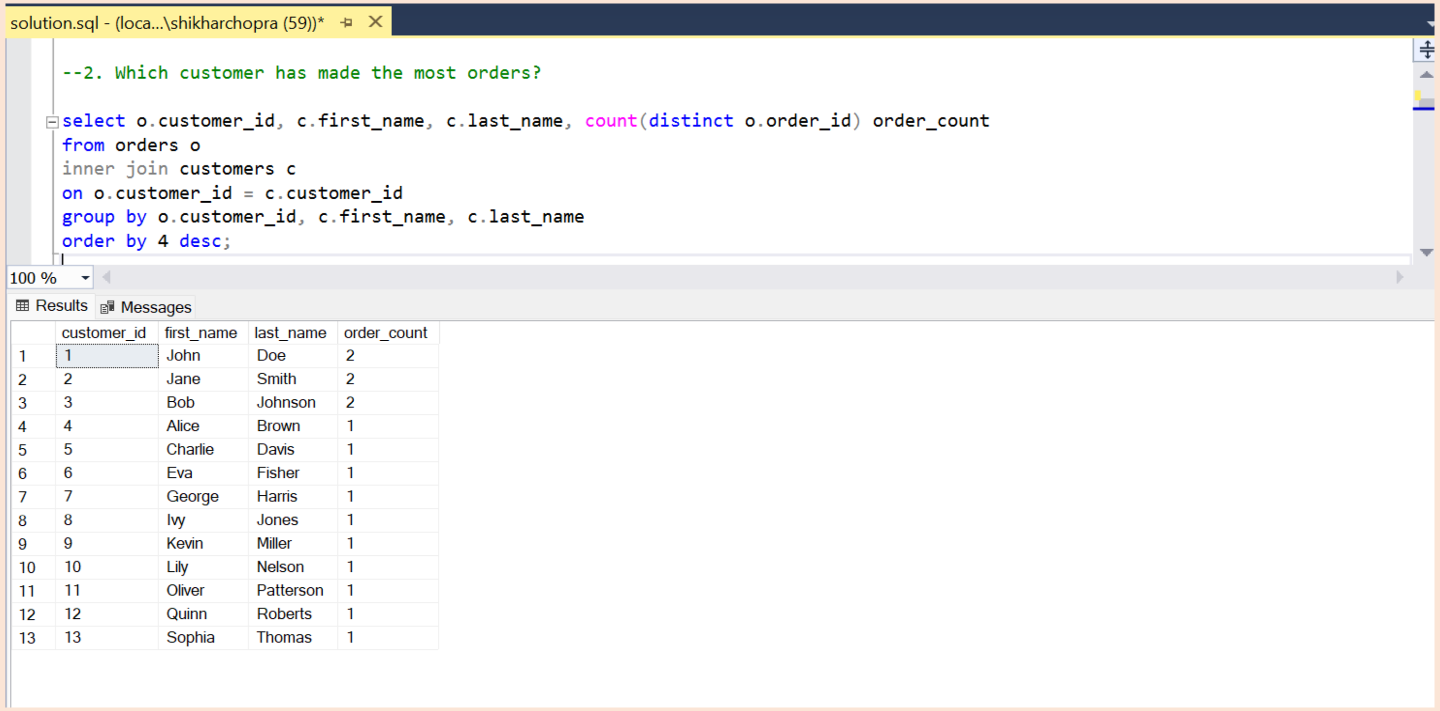
```sql
--1. Which product has the highest price? Only return a single row.

with cte as
(
    select *, dense_rank() over(order by price desc) rnk
    from products
)
select product_id, product_name, price
from cte
where rnk = 1
```

100 %

Results | Messages

```
   product_id  product_name  price
1  13          Product M     70
```

2. Which customer has made the most orders?



solution.sql - (loca...\shikharchopra (59))*

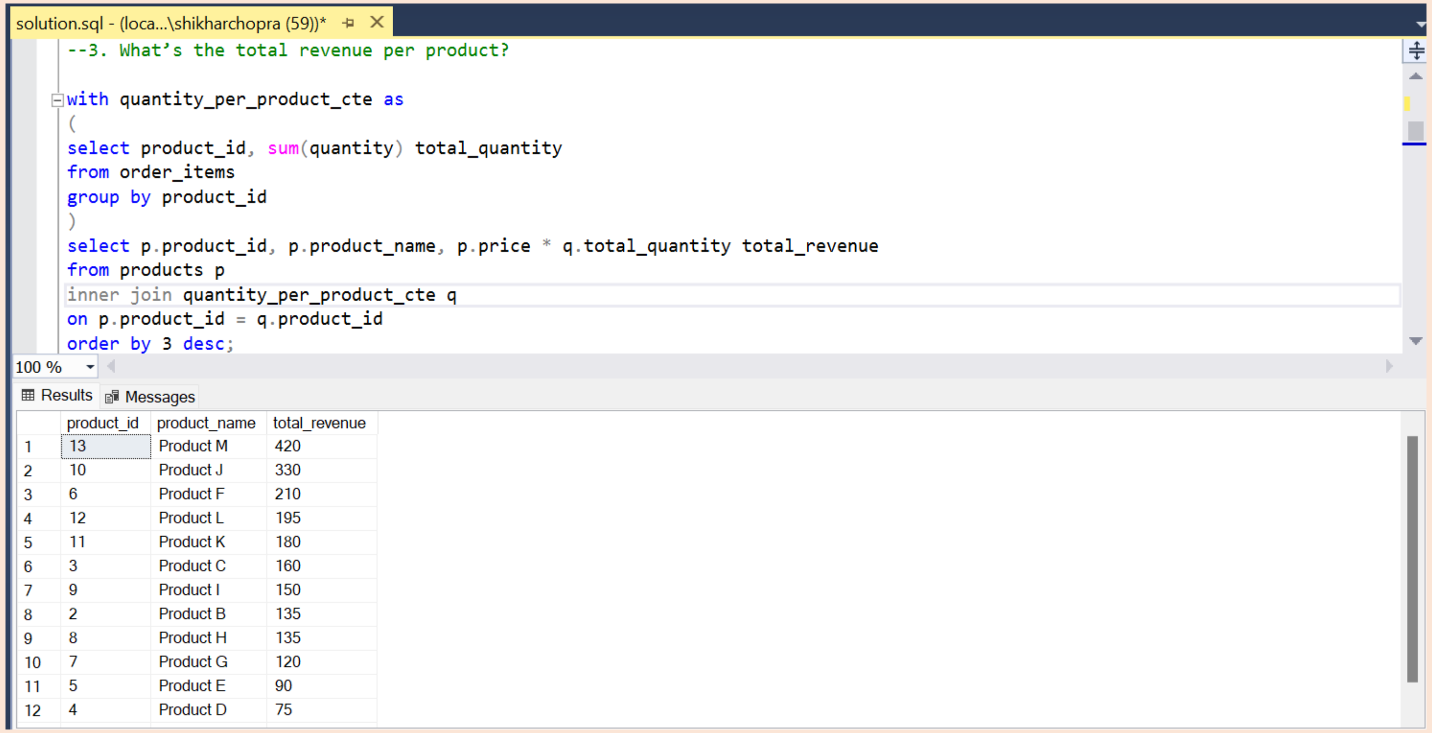
```
--2. Which customer has made the most orders?

select o.customer_id, c.first_name, c.last_name, count(distinct o.order_id) order_count
from orders o
inner join customers c
on o.customer_id = c.customer_id
group by o.customer_id, c.first_name, c.last_name
order by 4 desc;
```

100 %

Results | Messages

	customer_id	first_name	last_name	order_count
1	1	John	Doe	2
2	2	Jane	Smith	2
3	3	Bob	Johnson	2
4	4	Alice	Brown	1
5	5	Charlie	Davis	1
6	6	Eva	Fisher	1
7	7	George	Harris	1
8	8	Ivy	Jones	1
9	9	Kevin	Miller	1
10	10	Lily	Nelson	1
11	11	Oliver	Patterson	1
12	12	Quinn	Roberts	1
13	13	Sophia	Thomas	1

3. What’s the total revenue per product?


solution.sql - (loca...\shikharchopra (59))*

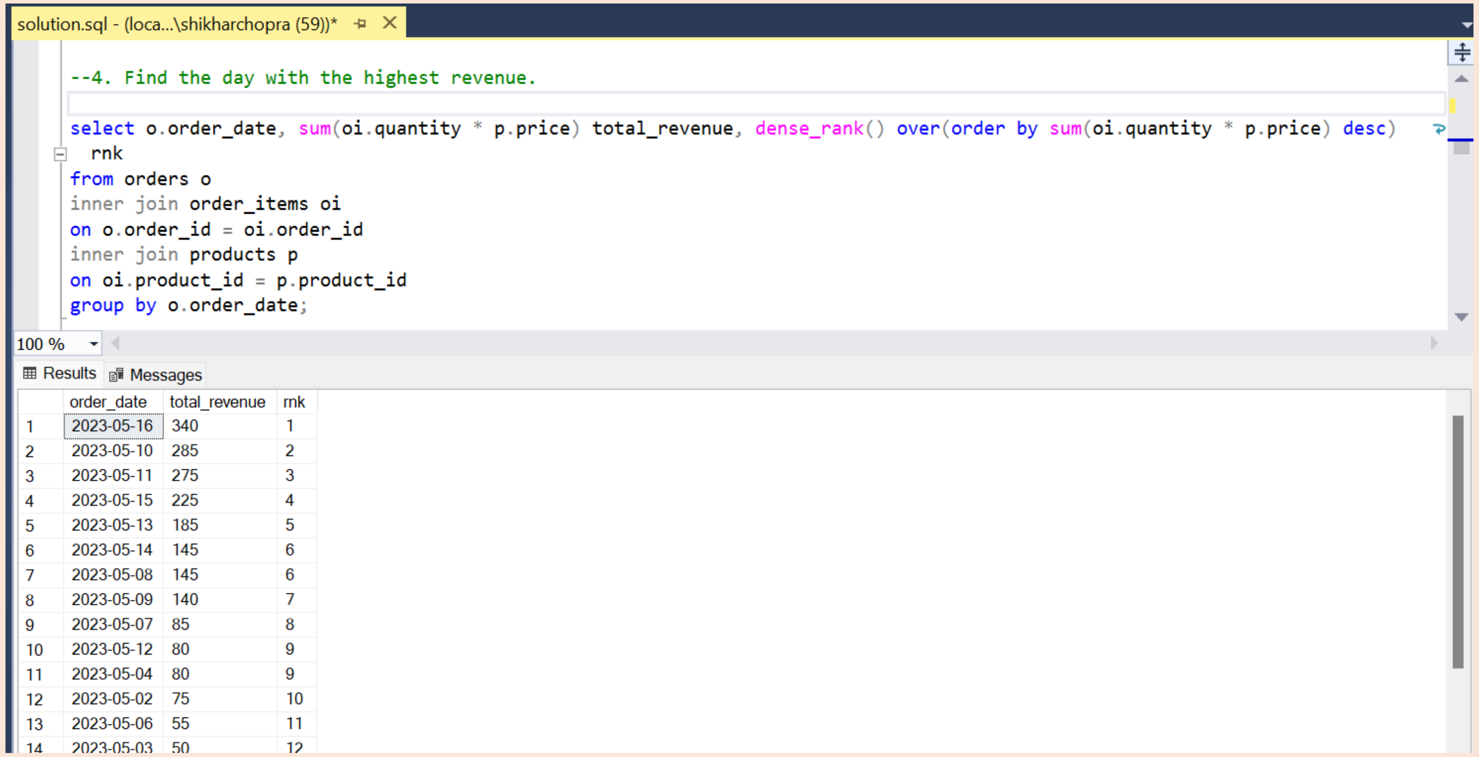
```
--3. What’s the total revenue per product?

with quantity_per_product_cte as
(
select product_id, sum(quantity) total_quantity
from order_items
group by product_id
)
select p.product_id, p.product_name, p.price * q.total_quantity total_revenue
from products p
inner join quantity_per_product_cte q
on p.product_id = q.product_id
order by 3 desc;
```

100 %

Results Messages

```
    product_id  product_name  total_revenue
1   13          Product M     420
2   10          Product J     330
3   6           Product F     210
4   12          Product L     195
5   11          Product K     180
6   3           Product C     160
7   9           Product I     150
8   2           Product B     135
9   8           Product H     135
10  7           Product G     120
11  5           Product E     90
12  4           Product D     75
```

4. Find the day with the highest revenue.



solution.sql - (loca...\shikharchopra (59))*

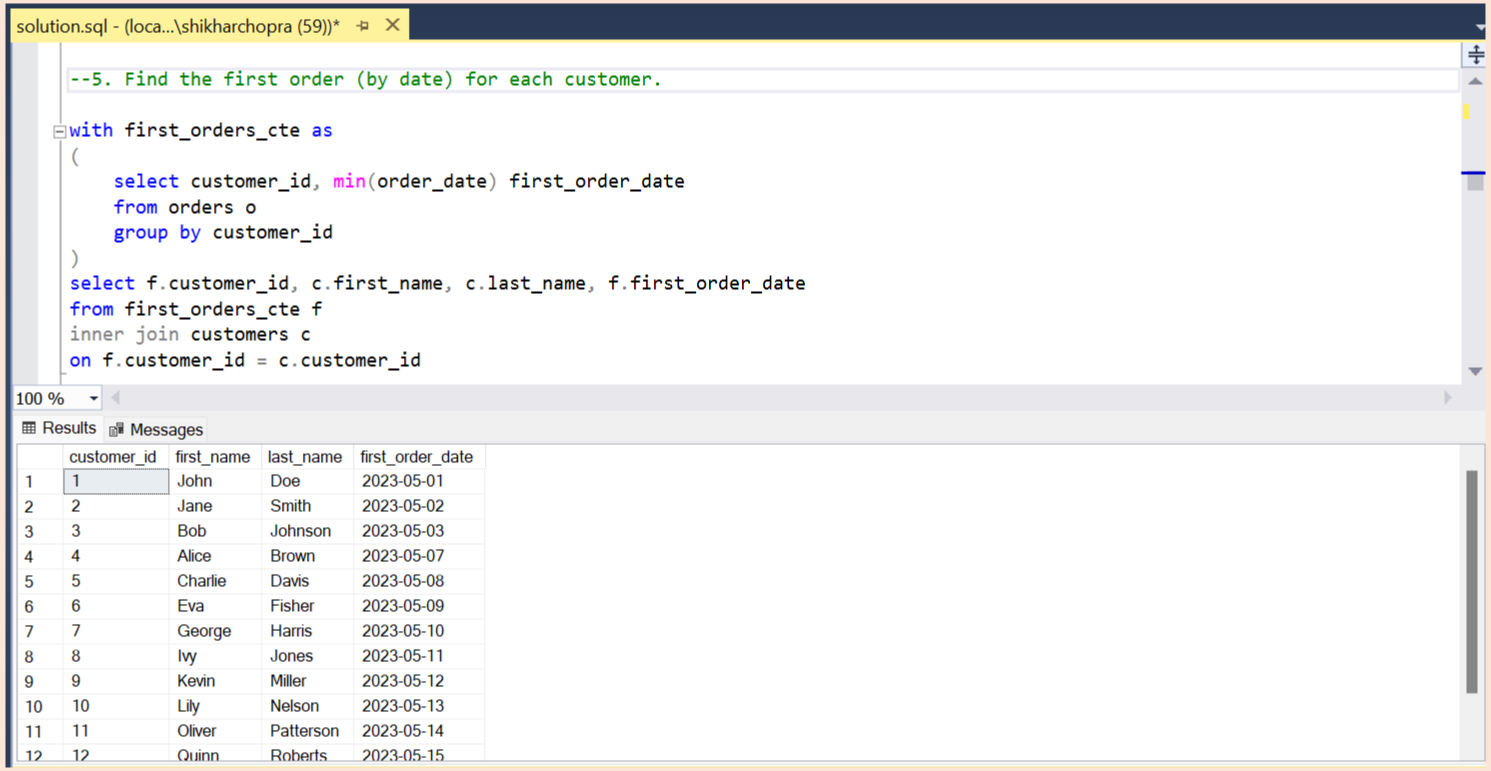
```sql
--4. Find the day with the highest revenue.

select o.order_date, sum(oi.quantity * p.price) total_revenue, dense_rank() over(order by sum(oi.quantity * p.price) desc)
 rnk
from orders o
inner join order_items oi
on o.order_id = oi.order_id
inner join products p
on oi.product_id = p.product_id
group by o.order_date;
```

100 %

Results Messages

	order_date	total_revenue	rnk
1	2023-05-16	340	1
2	2023-05-10	285	2
3	2023-05-11	275	3
4	2023-05-15	225	4
5	2023-05-13	185	5
6	2023-05-14	145	6
7	2023-05-08	145	6
8	2023-05-09	140	7
9	2023-05-07	85	8
10	2023-05-12	80	9
11	2023-05-04	80	9
12	2023-05-02	75	10
13	2023-05-06	55	11
14	2023-05-03	50	12

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



solution.sql - (loca...\shikharchopra (59))*

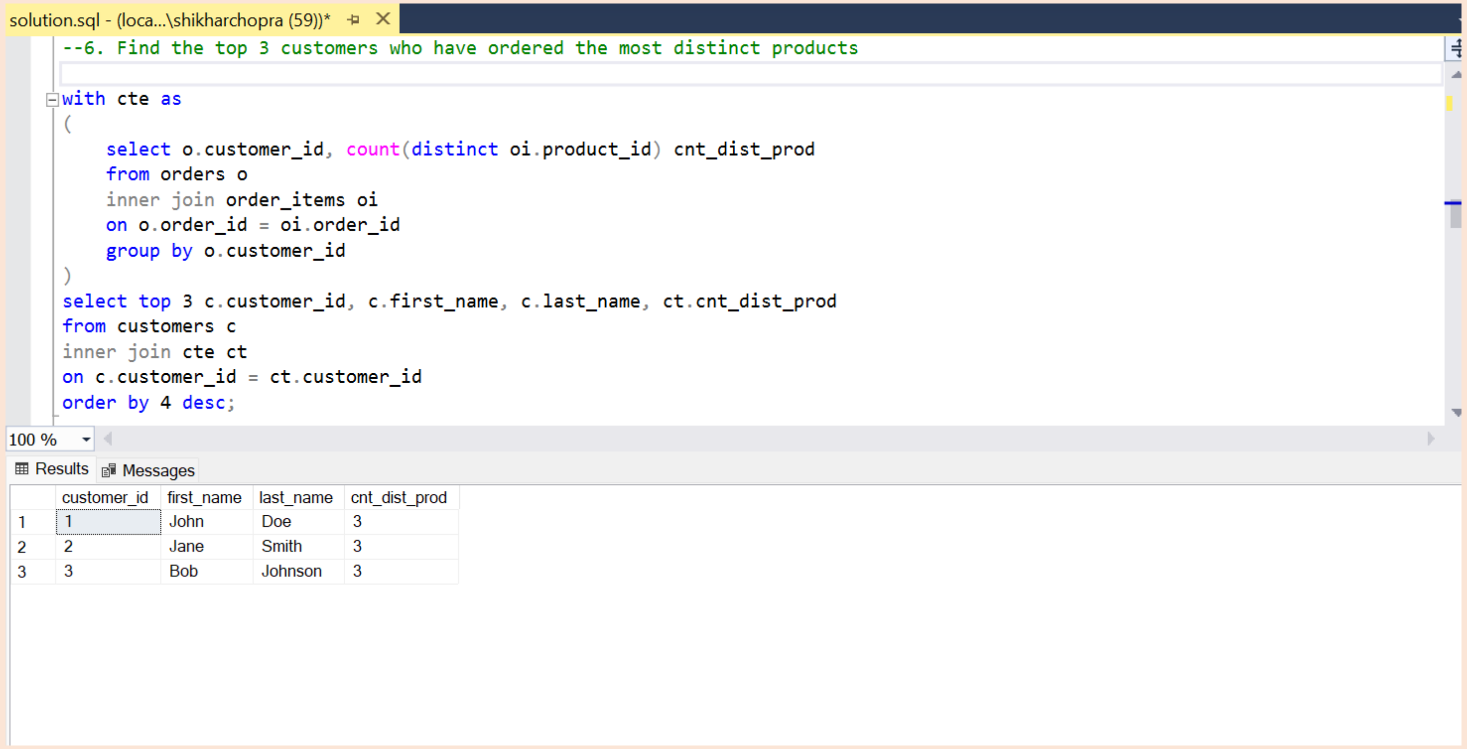
```
--5. Find the first order (by date) for each customer.

with first_orders_cte as
(
    select customer_id, min(order_date) first_order_date
    from orders o
    group by customer_id
)
select f.customer_id, c.first_name, c.last_name, f.first_order_date
from first_orders_cte f
inner join customers c
on f.customer_id = c.customer_id
```

100 %

Results Messages

	customer_id	first_name	last_name	first_order_date
1	1	John	Doe	2023-05-01
2	2	Jane	Smith	2023-05-02
3	3	Bob	Johnson	2023-05-03
4	4	Alice	Brown	2023-05-07
5	5	Charlie	Davis	2023-05-08
6	6	Eva	Fisher	2023-05-09
7	7	George	Harris	2023-05-10
8	8	Ivy	Jones	2023-05-11
9	9	Kevin	Miller	2023-05-12
10	10	Lily	Nelson	2023-05-13
11	11	Oliver	Patterson	2023-05-14
12	12	Quinn	Roberts	2023-05-15

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

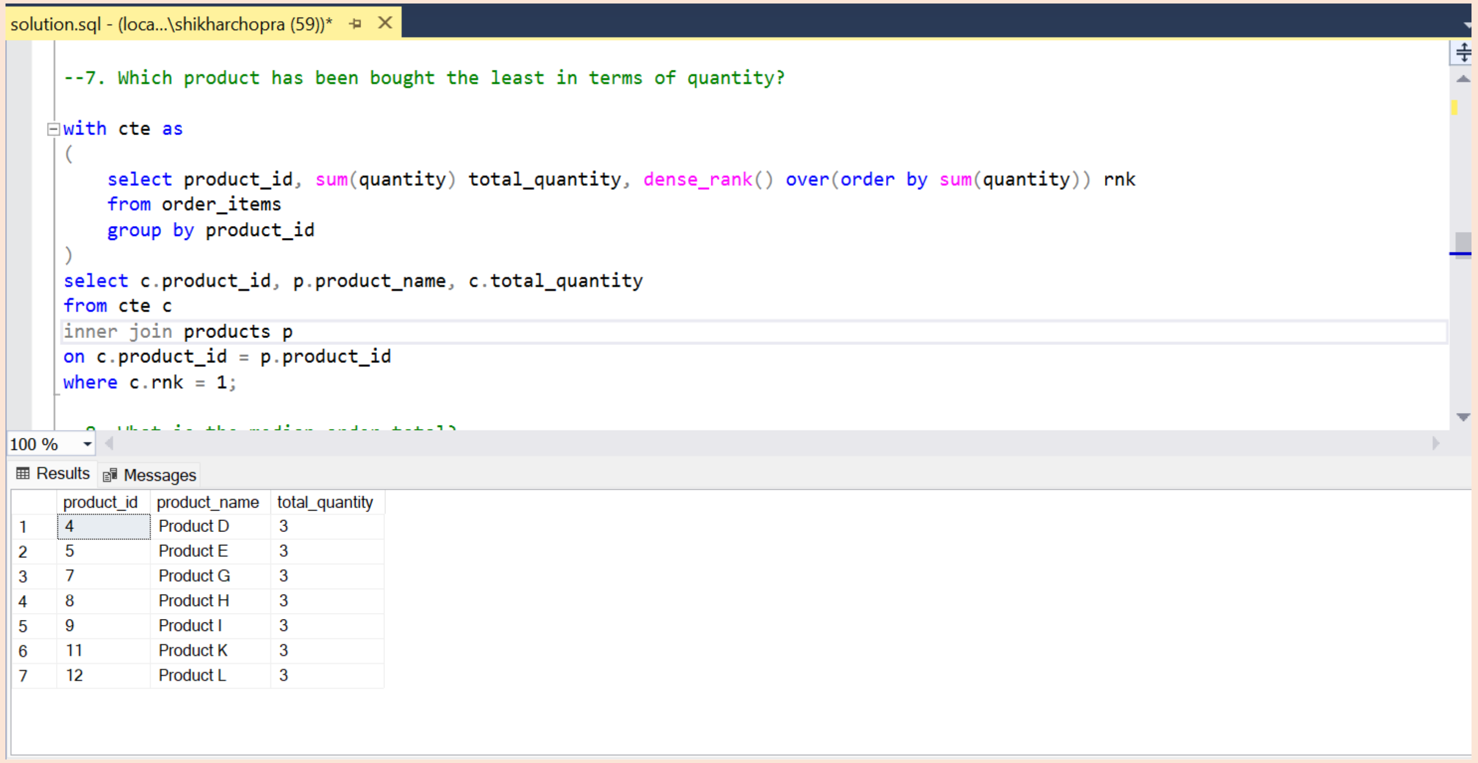


solution.sql - (loca...\shikharchopra (59))*

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

with cte as
(
    select o.customer_id, count(distinct oi.product_id) cnt_dist_prod
    from orders o
    inner join order_items oi
    on o.order_id = oi.order_id
    group by o.customer_id
)
select top 3 c.customer_id, c.first_name, c.last_name, ct.cnt_dist_prod
from customers c
inner join cte ct
on c.customer_id = ct.customer_id
order by 4 desc;
```

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

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


solution.sql - (loca...\shikharchopra (59))*

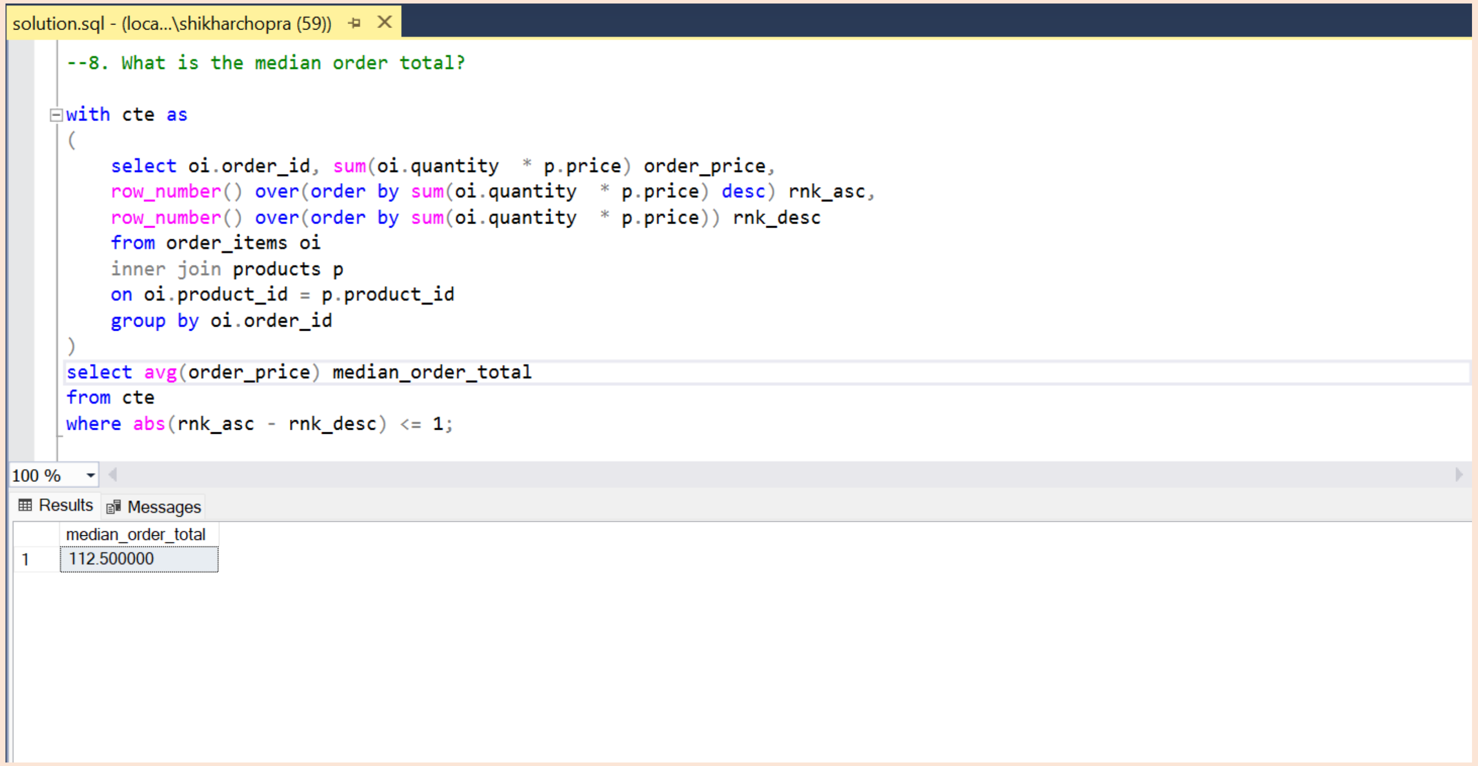
```
--7. Which product has been bought the least in terms of quantity?

with cte as
(
    select product_id, sum(quantity) total_quantity, dense_rank() over(order by sum(quantity)) rnk
    from order_items
    group by product_id
)
select c.product_id, p.product_name, c.total_quantity
from cte c
inner join products p
on c.product_id = p.product_id
where c.rnk = 1;
```

100 %

Results | Messages

	product_id	product_name	total_quantity
1	4	Product D	3
2	5	Product E	3
3	7	Product G	3
4	8	Product H	3
5	9	Product I	3
6	11	Product K	3
7	12	Product L	3

8. What is the median order total?



solution.sql - (loca...\shikharchopra (59))

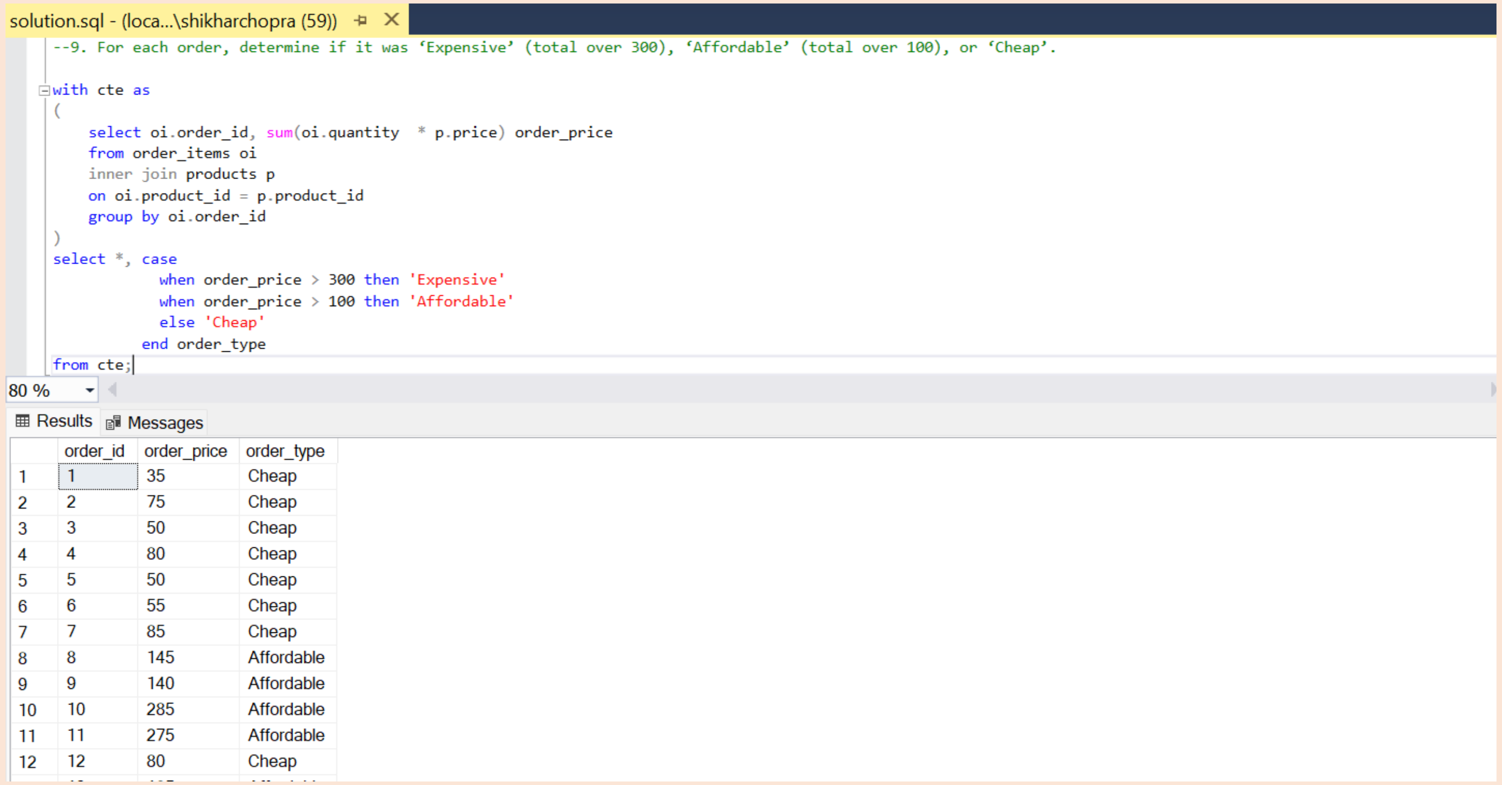
```
--8. What is the median order total?

with cte as
(
    select oi.order_id, sum(oi.quantity  * p.price) order_price,
    row_number() over(order by sum(oi.quantity  * p.price) desc) rnk_asc,
    row_number() over(order by sum(oi.quantity  * p.price)) rnk_desc
    from order_items oi
    inner join products p
    on oi.product_id = p.product_id
    group by oi.order_id
)
select avg(order_price) median_order_total
from cte
where abs(rnk_asc - rnk_desc) <= 1;
```

100 %

Results | Messages

	median_order_total
1	112.500000

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



solution.sql - (loca...\shikharchopra (59))

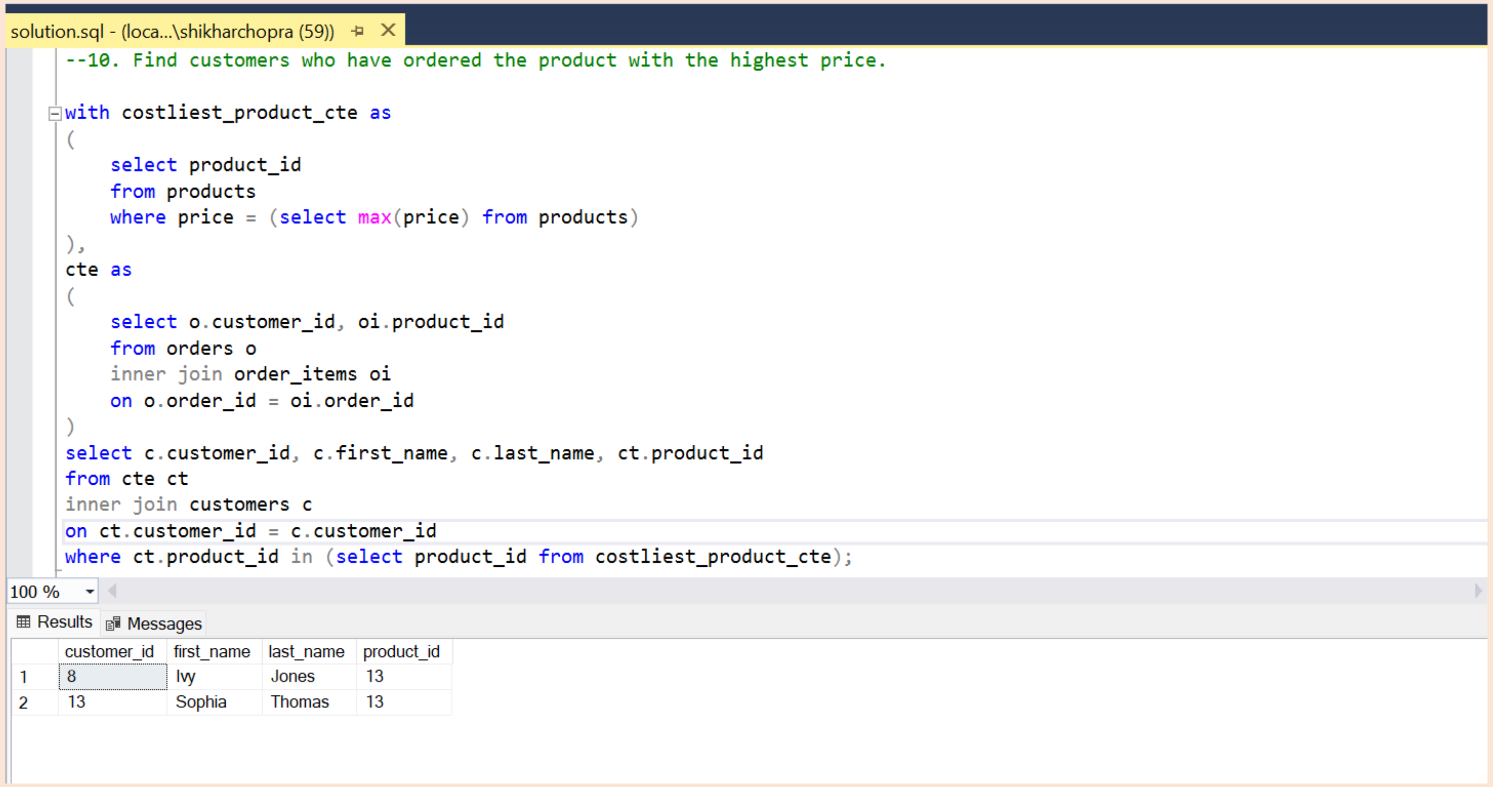
```sql
--9. For each order, determine if it was ‘Expensive’ (total over 300), ‘Affordable’ (total over 100), or ‘Cheap’.

with cte as
(
    select oi.order_id, sum(oi.quantity  * p.price) order_price
    from order_items oi
    inner join products p
    on oi.product_id = p.product_id
    group by oi.order_id
)
select *, case
            when order_price > 300 then 'Expensive'
            when order_price > 100 then 'Affordable'
            else 'Cheap'
         end order_type
from cte;
```

80 %

Results | Messages

	order_id	order_price	order_type
1	1	35	Cheap
2	2	75	Cheap
3	3	50	Cheap
4	4	80	Cheap
5	5	50	Cheap
6	6	55	Cheap
7	7	85	Cheap
8	8	145	Affordable
9	9	140	Affordable
10	10	285	Affordable
11	11	275	Affordable
12	12	80	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.

with costliest_product_cte as
(
    select product_id
    from products
    where price = (select max(price) from products)
),
cte as
(
    select o.customer_id, oi.product_id
    from orders o
    inner join order_items oi
    on o.order_id = oi.order_id
)
select c.customer_id, c.first_name, c.last_name, ct.product_id
from cte ct
inner join customers c
on ct.customer_id = c.customer_id
where ct.product_id in (select product_id from costliest_product_cte);
```

	customer_id	first_name	last_name	product_id
1	8	Ivy	Jones	13
2	13	Sophia	Thomas	13

INSIGHTS

- Product M is the costliest product
- Jon Doe, Jane Smith, Bob Johnson have made the most orders (2)
- Highest revenue was generated on 16/05/23 ($360)
- Jon Doe, Jane Smith, Bob Johnson have ordered the most distinct products (3)
- The median order total is $112.50

Note – Assumed currency as USD ($) for the case study