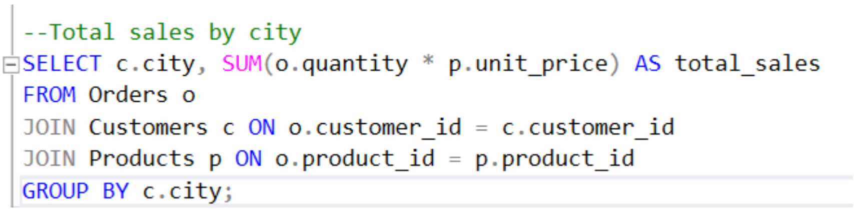
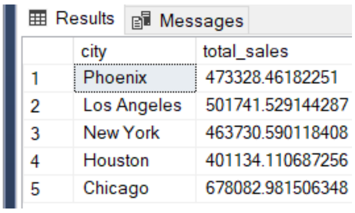
TASK 4

1

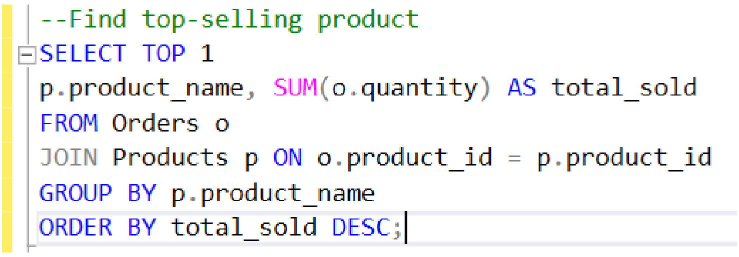


```sql
--Total sales by city
SELECT c.city, SUM(o.quantity * p.unit_price) AS total_sales
FROM Orders o
JOIN Customers c ON o.customer_id = c.customer_id
JOIN Products p ON o.product_id = p.product_id
GROUP BY c.city;
```

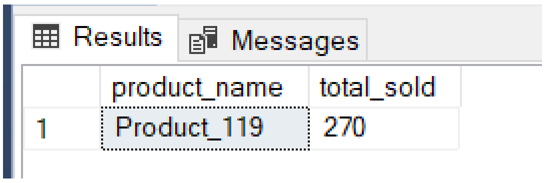


	city	total_sales
1	Phoenix	473328.46182251
2	Los Angeles	501741.529144287
3	New York	463730.590118408
4	Houston	401134.110687256
5	Chicago	678082.981506348

2

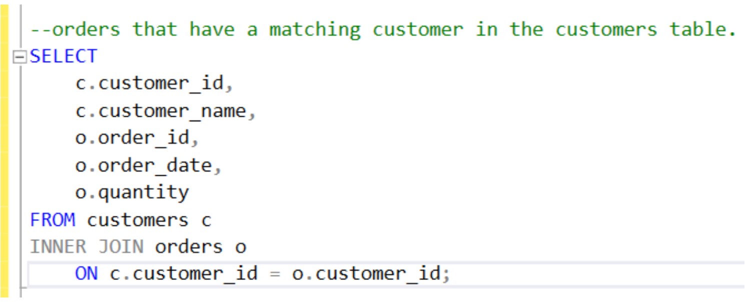


```sql
--Find top-selling product
SELECT TOP 1
p.product_name, SUM(o.quantity) AS total_sold
FROM Orders o
JOIN Products p ON o.product_id = p.product_id
GROUP BY p.product_name
ORDER BY total_sold DESC;
```

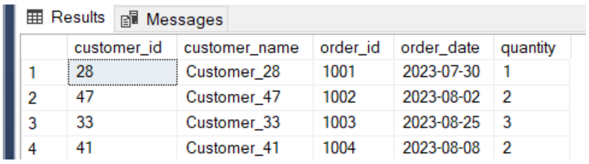


	product_name	total_sold
1	Product_119	270

3



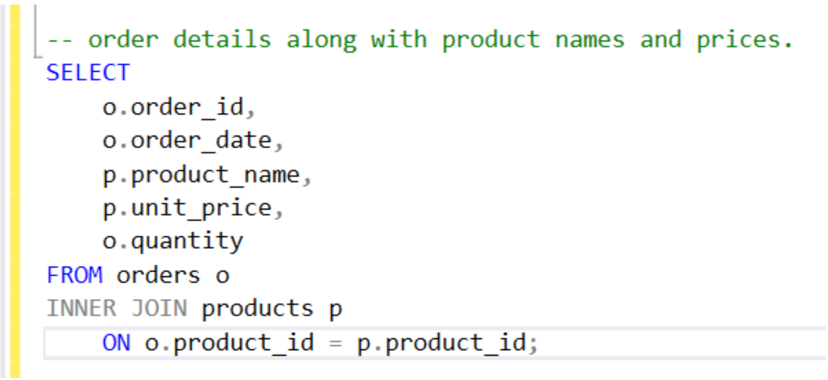
```sql
--orders that have a matching customer in the customers table.
SELECT
    c.customer_id,
    c.customer_name,
    o.order_id,
    o.order_date,
    o.quantity
FROM customers c
INNER JOIN orders o
    ON c.customer_id = o.customer_id;
```



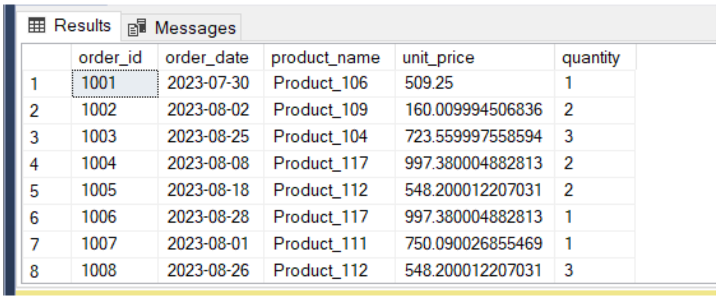
	customer_id	customer_name	order_id	order_date	quantity
1	28	Customer_28	1001	2023-07-30	1
2	47	Customer_47	1002	2023-08-02	2
3	33	Customer_33	1003	2023-08-25	3
4	41	Customer_41	1004	2023-08-08	2

TASK 4

4

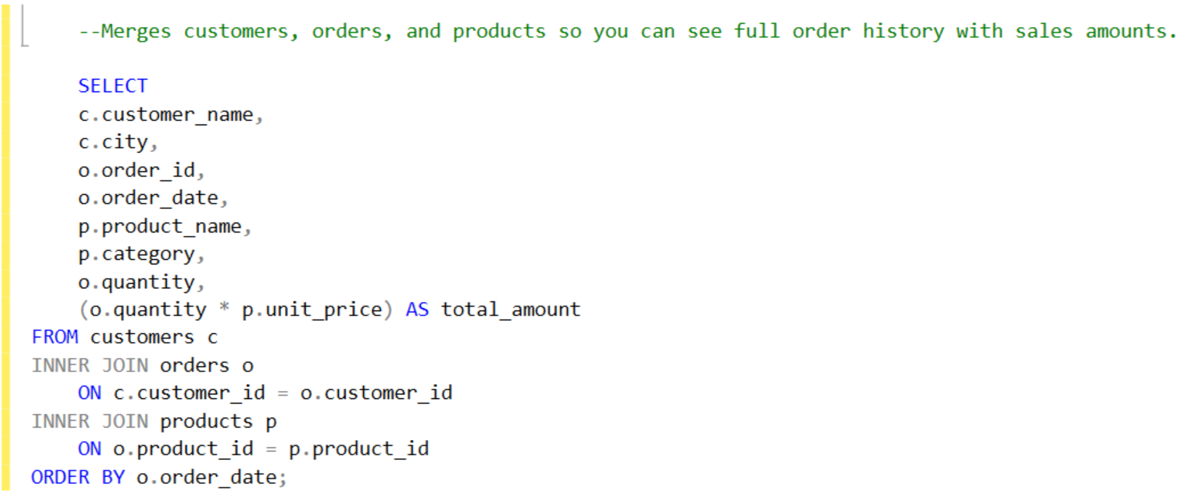


```
-- order details along with product names and prices.
SELECT
    o.order_id,
    o.order_date,
    p.product_name,
    p.unit_price,
    o.quantity
FROM orders o
INNER JOIN products p
    ON o.product_id = p.product_id;
```

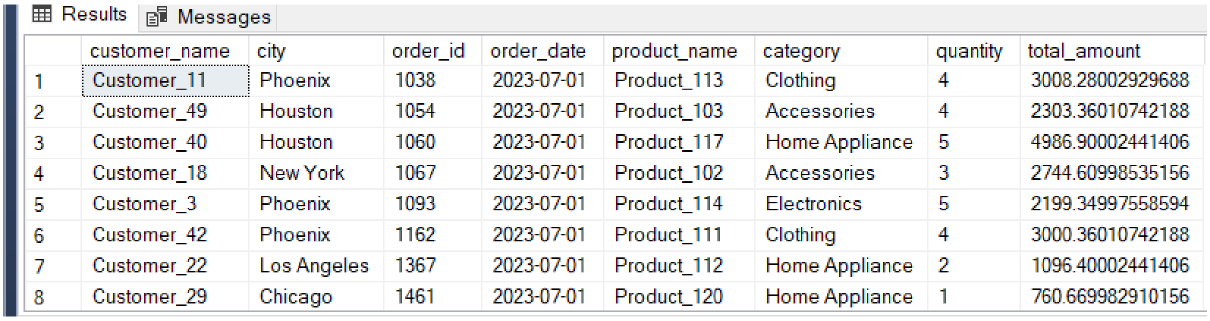


	order_id	order_date	product_name	unit_price	quantity
1	1001	2023-07-30	Product_106	509.25	1
2	1002	2023-08-02	Product_109	160.009994506836	2
3	1003	2023-08-25	Product_104	723.559997558594	3
4	1004	2023-08-08	Product_117	997.380004882813	2
5	1005	2023-08-18	Product_112	548.200012207031	2
6	1006	2023-08-28	Product_117	997.380004882813	1
7	1007	2023-08-01	Product_111	750.090026855469	1
8	1008	2023-08-26	Product_112	548.200012207031	3

5



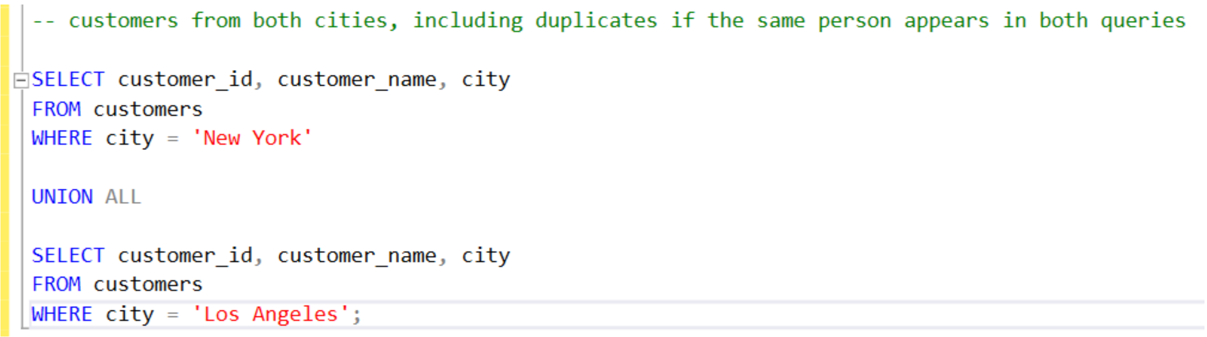
```
    --Merges customers, orders, and products so you can see full order history with sales amounts.

    SELECT
    c.customer_name,
    c.city,
    o.order_id,
    o.order_date,
    p.product_name,
    p.category,
    o.quantity,
    (o.quantity * p.unit_price) AS total_amount
FROM customers c
INNER JOIN orders o
    ON c.customer_id = o.customer_id
INNER JOIN products p
    ON o.product_id = p.product_id
ORDER BY o.order_date;
```



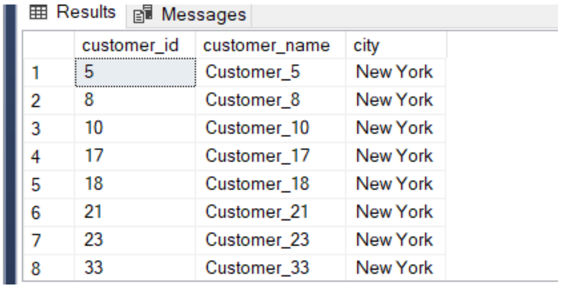
	customer_name	city	order_id	order_date	product_name	category	quantity	total_amount
1	Customer_11	Phoenix	1038	2023-07-01	Product_113	Clothing	4	3008.28002929688
2	Customer_49	Houston	1054	2023-07-01	Product_103	Accessories	4	2303.36010742188
3	Customer_40	Houston	1060	2023-07-01	Product_117	Home Appliance	5	4986.90002441406
4	Customer_18	New York	1067	2023-07-01	Product_102	Accessories	3	2744.60998535156
5	Customer_3	Phoenix	1093	2023-07-01	Product_114	Electronics	5	2199.34997558594
6	Customer_42	Phoenix	1162	2023-07-01	Product_111	Clothing	4	3000.36010742188
7	Customer_22	Los Angeles	1367	2023-07-01	Product_112	Home Appliance	2	1096.40002441406
8	Customer_29	Chicago	1461	2023-07-01	Product_120	Home Appliance	1	760.669982910156

TASK 4

6

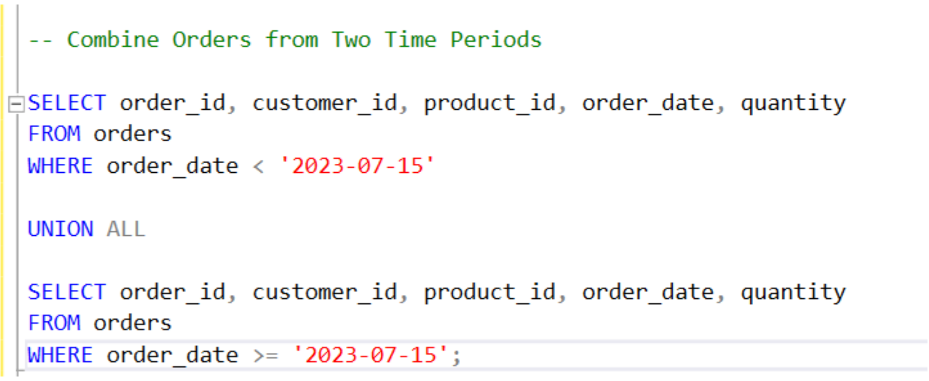


```
-- customers from both cities, including duplicates if the same person appears in both queries

SELECT customer_id, customer_name, city
FROM customers
WHERE city = 'New York'

UNION ALL

SELECT customer_id, customer_name, city
FROM customers
WHERE city = 'Los Angeles';
```

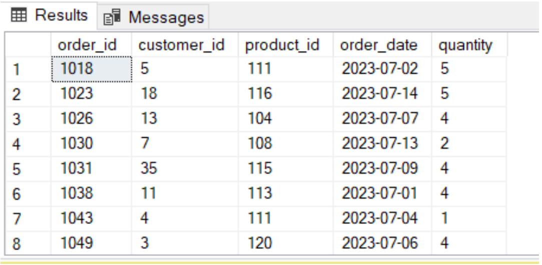


	customer_id	customer_name	city
1	5	Customer_5	New York
2	8	Customer_8	New York
3	10	Customer_10	New York
4	17	Customer_17	New York
5	18	Customer_18	New York
6	21	Customer_21	New York
7	23	Customer_23	New York
8	33	Customer_33	New York

7



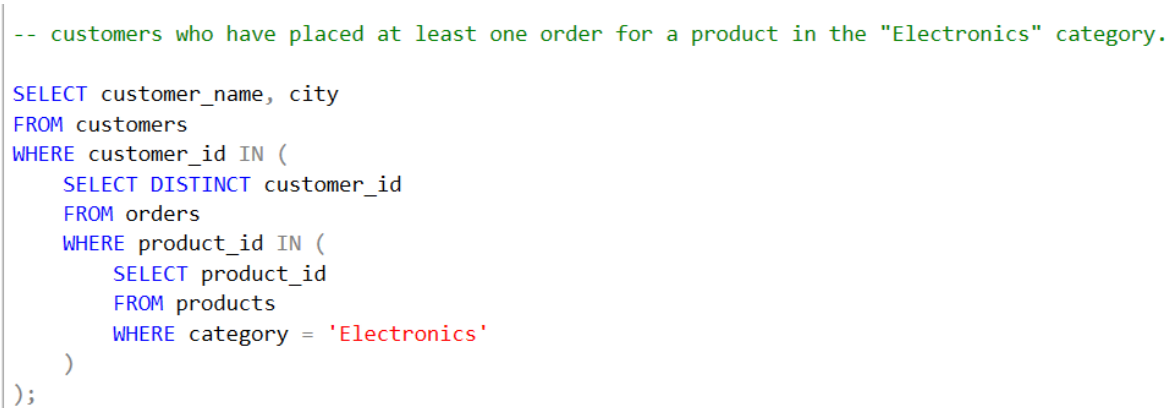
```
-- Combine Orders from Two Time Periods

SELECT order_id, customer_id, product_id, order_date, quantity
FROM orders
WHERE order_date < '2023-07-15'

UNION ALL

SELECT order_id, customer_id, product_id, order_date, quantity
FROM orders
WHERE order_date >= '2023-07-15';
```



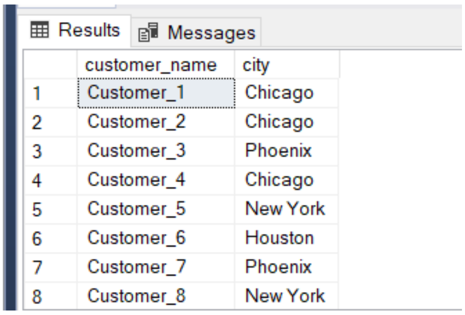
	order_id	customer_id	product_id	order_date	quantity
1	1018	5	111	2023-07-02	5
2	1023	18	116	2023-07-14	5
3	1026	13	104	2023-07-07	4
4	1030	7	108	2023-07-13	2
5	1031	35	115	2023-07-09	4
6	1038	11	113	2023-07-01	4
7	1043	4	111	2023-07-04	1
8	1049	3	120	2023-07-06	4

TASK 4

8

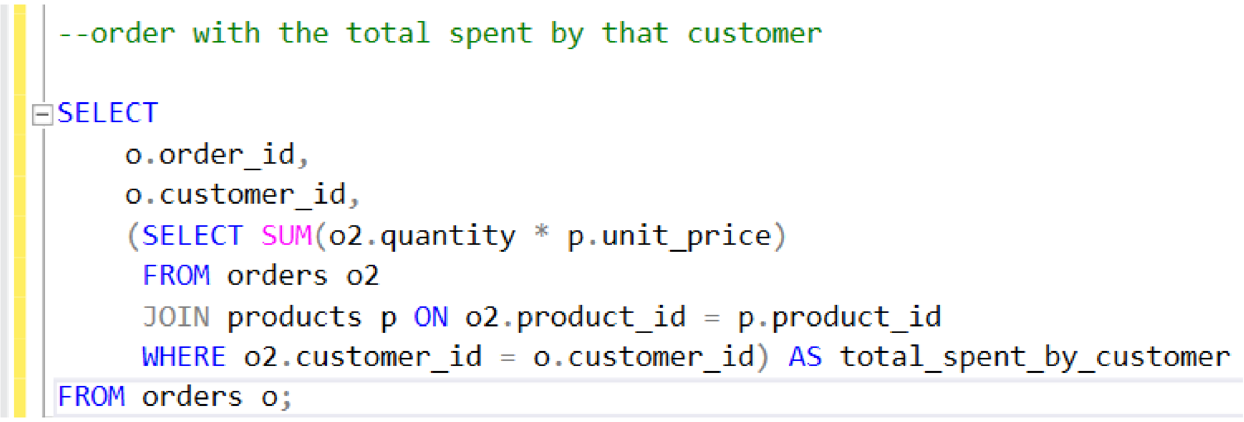


```sql
-- customers who have placed at least one order for a product in the "Electronics" category.

SELECT customer_name, city
FROM customers
WHERE customer_id IN (
    SELECT DISTINCT customer_id
    FROM orders
    WHERE product_id IN (
        SELECT product_id
        FROM products
        WHERE category = 'Electronics'
    )
);
```

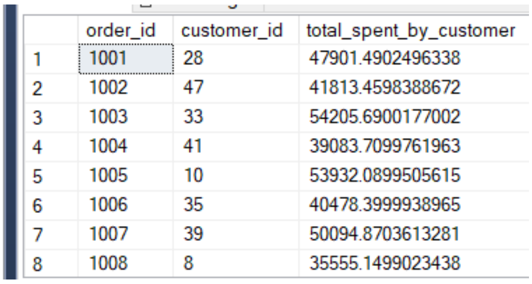


	customer_name	city
1	Customer_1	Chicago
2	Customer_2	Chicago
3	Customer_3	Phoenix
4	Customer_4	Chicago
5	Customer_5	New York
6	Customer_6	Houston
7	Customer_7	Phoenix
8	Customer_8	New York

9



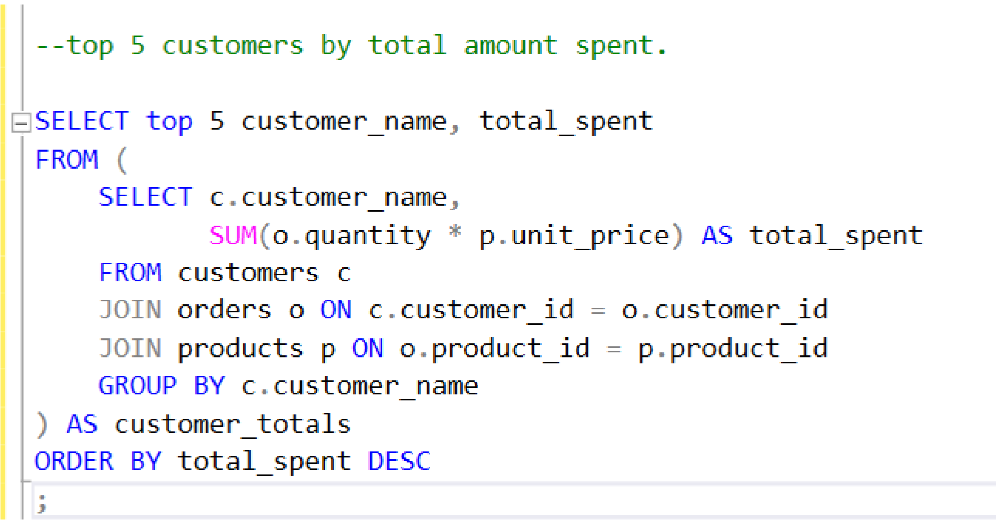
```sql
--order with the total spent by that customer

SELECT
    o.order_id,
    o.customer_id,
    (SELECT SUM(o2.quantity * p.unit_price)
     FROM orders o2
     JOIN products p ON o2.product_id = p.product_id
     WHERE o2.customer_id = o.customer_id) AS total_spent_by_customer
FROM orders o;
```



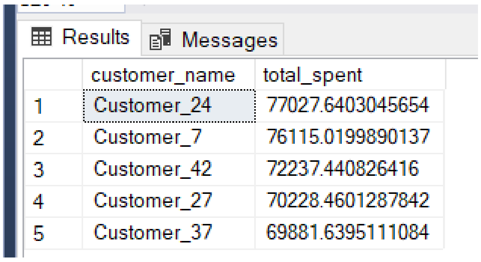
	order_id	customer_id	total_spent_by_customer
1	1001	28	47901.4902496338
2	1002	47	41813.4598388672
3	1003	33	54205.6900177002
4	1004	41	39083.7099761963
5	1005	10	53932.0899505615
6	1006	35	40478.3999938965
7	1007	39	50094.8703613281
8	1008	8	35555.1499023438

TASK 4

10

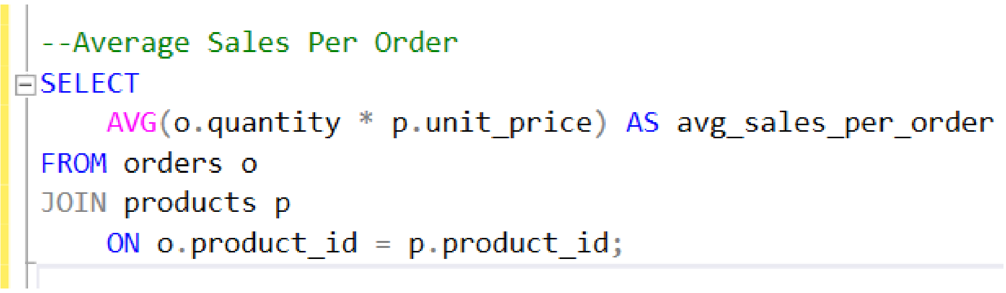


```
--top 5 customers by total amount spent.

SELECT top 5 customer_name, total_spent
FROM (
    SELECT c.customer_name,
           SUM(o.quantity * p.unit_price) AS total_spent
    FROM customers c
    JOIN orders o ON c.customer_id = o.customer_id
    JOIN products p ON o.product_id = p.product_id
    GROUP BY c.customer_name
) AS customer_totals
ORDER BY total_spent DESC
;
```

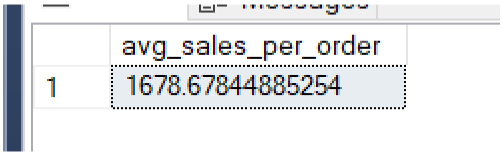


	customer_name	total_spent
1	Customer_24	77027.6403045654
2	Customer_7	76115.0199890137
3	Customer_42	72237.440826416
4	Customer_27	70228.4601287842
5	Customer_37	69881.6395111084

11



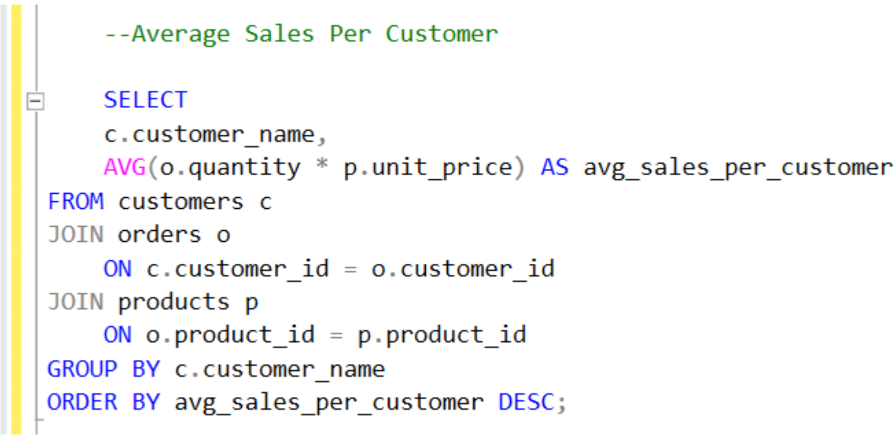
```
--Average Sales Per Order
SELECT
    AVG(o.quantity * p.unit_price) AS avg_sales_per_order
FROM orders o
JOIN products p
    ON o.product_id = p.product_id;
```



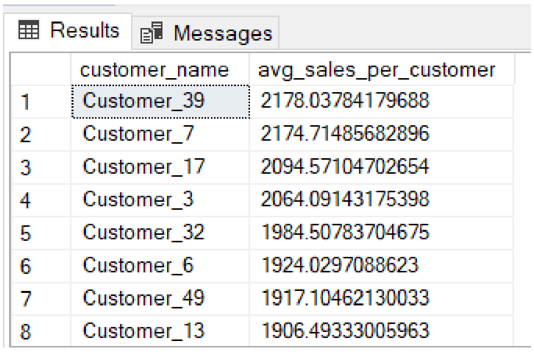
	avg_sales_per_order
1	1678.67844885254

12

TASK 4

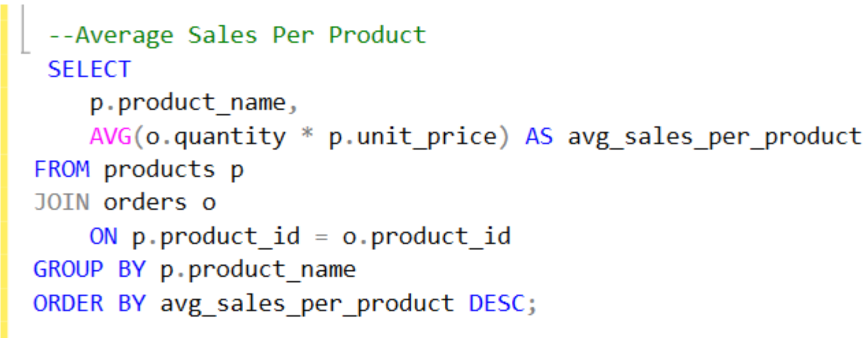


```sql
    --Average Sales Per Customer

    SELECT
    c.customer_name,
    AVG(o.quantity * p.unit_price) AS avg_sales_per_customer
FROM customers c
JOIN orders o
    ON c.customer_id = o.customer_id
JOIN products p
    ON o.product_id = p.product_id
GROUP BY c.customer_name
ORDER BY avg_sales_per_customer DESC;
```

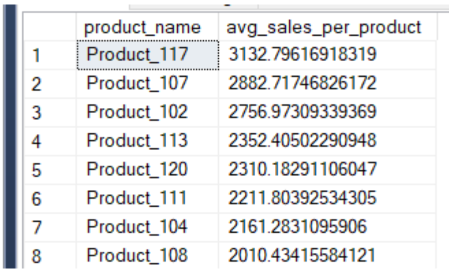


	customer_name	avg_sales_per_customer
1	Customer_39	2178.03784179688
2	Customer_7	2174.71485682896
3	Customer_17	2094.57104702654
4	Customer_3	2064.09143175398
5	Customer_32	1984.50783704675
6	Customer_6	1924.0297088623
7	Customer_49	1917.10462130033
8	Customer_13	1906.49333005963

13



```sql
 --Average Sales Per Product
 SELECT
    p.product_name,
    AVG(o.quantity * p.unit_price) AS avg_sales_per_product
FROM products p
JOIN orders o
    ON p.product_id = o.product_id
GROUP BY p.product_name
ORDER BY avg_sales_per_product DESC;
```



	product_name	avg_sales_per_product
1	Product_117	3132.79616918319
2	Product_107	2882.71746826172
3	Product_102	2756.97309339369
4	Product_113	2352.40502290948
5	Product_120	2310.18291106047
6	Product_111	2211.80392534305
7	Product_104	2161.2831095906
8	Product_108	2010.43415584121

TASK 4