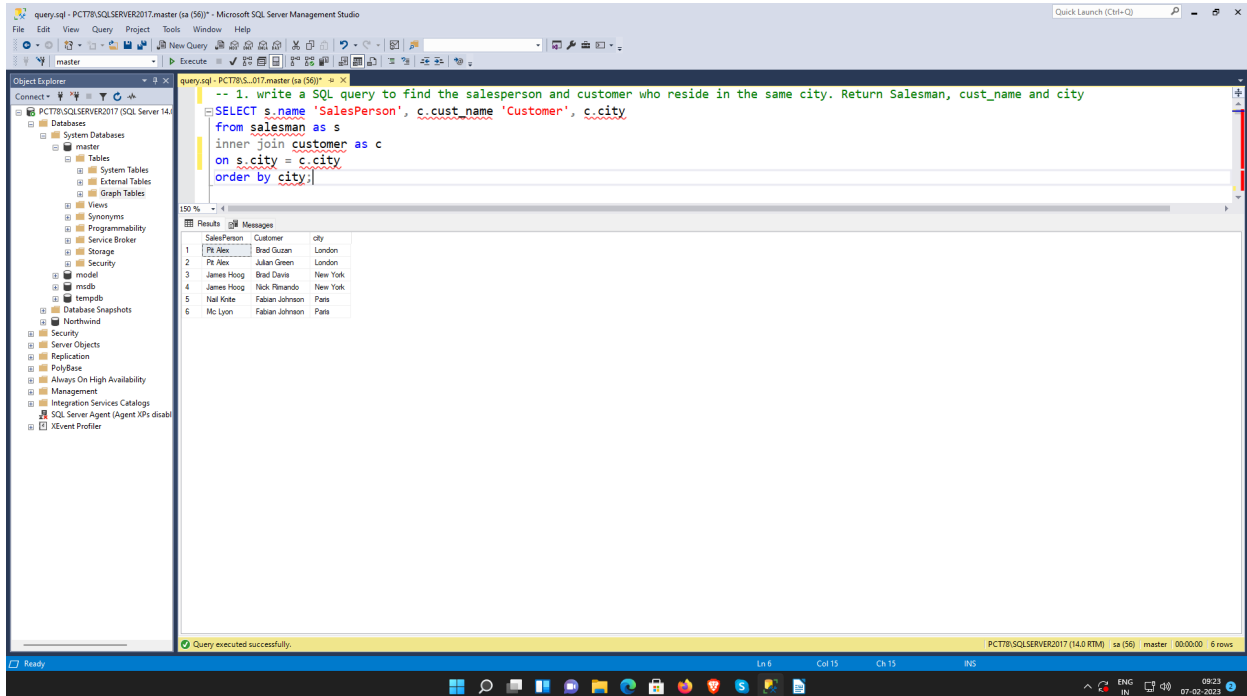


Assignment 2

-- 1. write a SQL query to find the salesperson and customer who reside in the same city. Return Salesman, cust_name and city



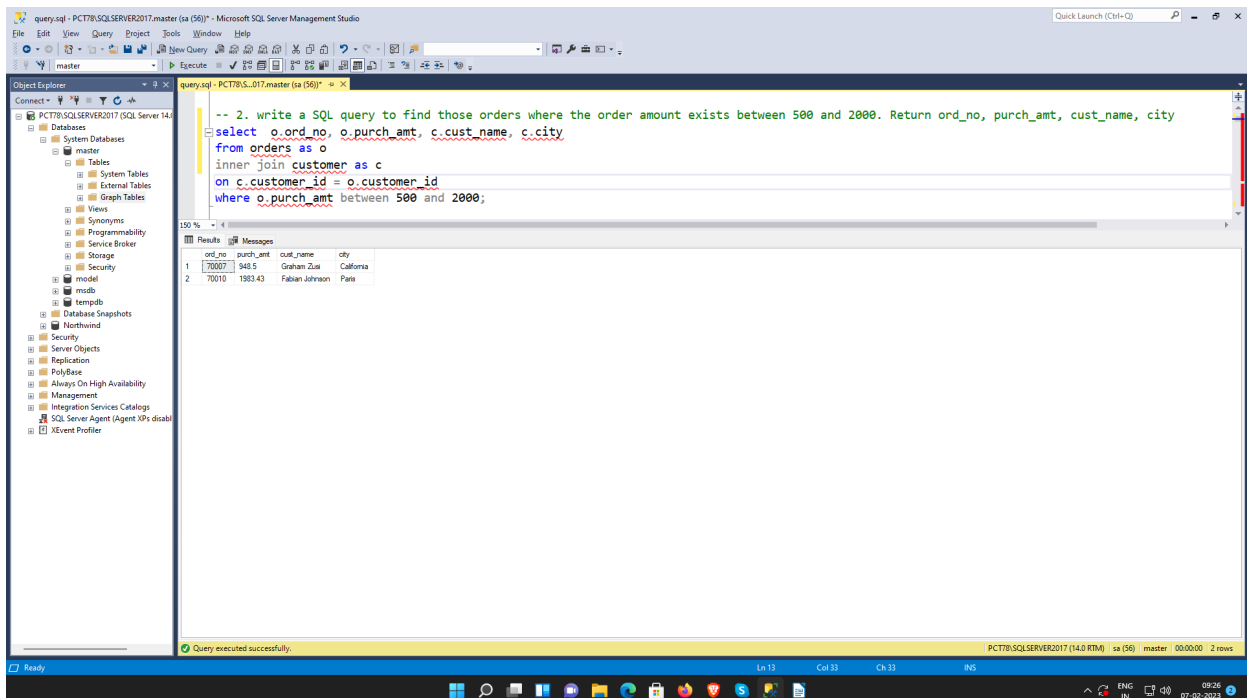
The screenshot shows the Microsoft SQL Server Enterprise Manager interface. The left pane displays the 'Object Explorer' with the 'master' database selected. The right pane shows a SQL query window with the following query:

```
-- 1. write a SQL query to find the salesperson and customer who reside in the same city. Return Salesman, cust_name and city
SELECT s.name 'SalesPerson', c.cust_name 'Customer', c.city
from salesman as s
inner join customer as c
on s.city = c.city
order by city;
```

The query results are displayed in a table with 6 rows and 3 columns: SalesPerson, Customer, and city.

SalesPerson	Customer	city
Pt Alex	Brad Guzan	London
Pt Alex	Adrian Green	London
James Hoog	Brad Davis	New York
James Hoog	Nick Rimando	New York
Nail Kite	Fabian Johnson	Paris
Mc Lyon	Fabian Johnson	Paris

-- 2. write a SQL query to find those orders where the order amount exists between 500 and 2000. Return ord_no, purch_amt, cust_name, city



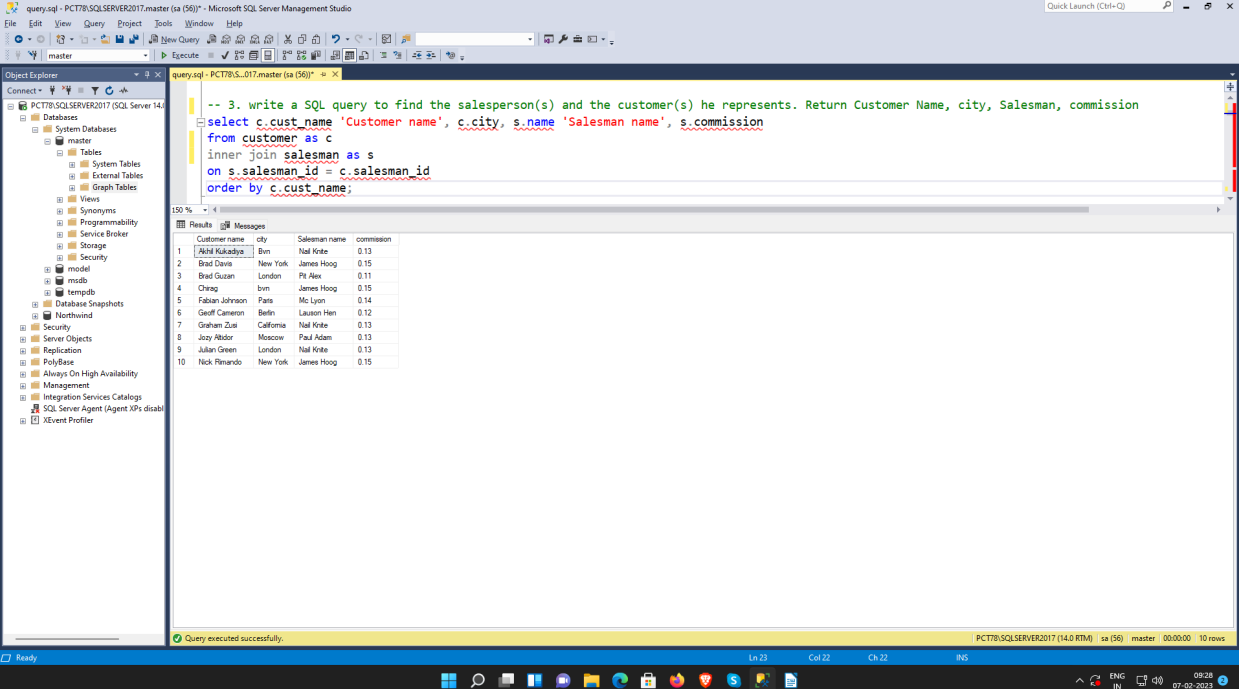
The screenshot shows the Microsoft SQL Server Enterprise Manager interface. The left pane displays the 'Object Explorer' with the 'master' database selected. The right pane shows a SQL query window with the following query:

```
-- 2. write a SQL query to find those orders where the order amount exists between 500 and 2000. Return ord_no, purch_amt, cust_name, city
select o.ord_no, o.purch_amt, c.cust_name, c.city
from orders as o
inner join customer as c
on c.customer_id = o.customer_id
where o.purch_amt between 500 and 2000;
```

The query results are displayed in a table with 2 rows and 4 columns: ord_no, purch_amt, cust_name, and city.

ord_no	purch_amt	cust_name	city
70007	148.5	Graham Zusi	California
70010	1983.43	Fabian Johnson	Paris

-- 3. write a SQL query to find the salesperson(s) and the customer(s) he represents. Return Customer Name, city, Salesman, commission



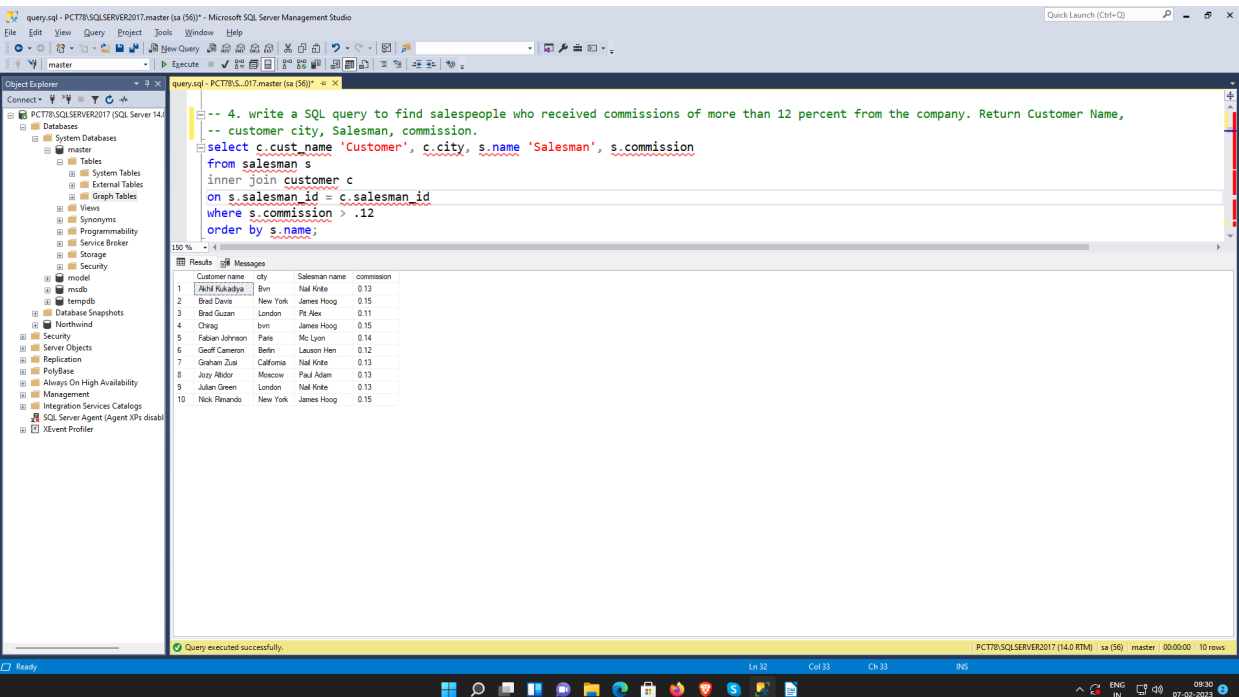
The screenshot shows the Microsoft SQL Server Enterprise Manager interface. The left pane displays the 'Object Explorer' with the 'master' database selected. The right pane shows a SQL query window with the following query:

```
-- 3. write a SQL query to find the salesperson(s) and the customer(s) he represents. Return Customer Name, city, Salesman, commission
select c.cust_name 'Customer name', c.city, s.name 'Salesman name', s.commission
from customer as c
inner join salesman as s
on s.salesman_id = c.salesman_id
order by c.cust_name;
```

The query results are displayed in a table with 10 rows and 4 columns: Customer name, city, Salesman name, and commission.

Customer name	city	Salesman name	commission
Akhil Kakadiya	Bvn	Nal Krite	0.13
Brad Davis	New York	James Hoog	0.15
Brad Guzan	London	Pit Alex	0.11
Chrag	bvn	James Hoog	0.15
Fabian Johnson	Paris	Mc Lyon	0.14
Geoff Cameron	Berlin	Leuson Hen	0.12
Graham Zusi	California	Nal Krite	0.13
Jozzy Abdo	Moscow	Paul Adam	0.13
Julian Green	London	Nal Krite	0.13
Nick Pinardo	New York	James Hoog	0.15

-- 4. write a SQL query to find salespeople who received commissions of more than 12 percent from the company. Return Customer Name, customer city, Salesman, commission.



The screenshot shows the Microsoft SQL Server Enterprise Manager interface. The left pane displays the 'Object Explorer' with the 'master' database selected. The right pane shows a SQL query window with the following query:

```
-- 4. write a SQL query to find salespeople who received commissions of more than 12 percent from the company. Return Customer Name,
-- customer city, Salesman, commission.
select c.cust_name 'Customer', c.city, s.name 'Salesman', s.commission
from salesman s
inner join customer c
on s.salesman_id = c.salesman_id
where s.commission > .12
order by s.name;
```

The query results are displayed in a table with 10 rows and 4 columns: Customer name, city, Salesman name, and commission.

Customer name	city	Salesman name	commission
Akhil Kakadiya	Bvn	Nal Krite	0.13
Brad Davis	New York	James Hoog	0.15
Brad Guzan	London	Pit Alex	0.11
Chrag	bvn	James Hoog	0.15
Fabian Johnson	Paris	Mc Lyon	0.14
Geoff Cameron	Berlin	Leuson Hen	0.12
Graham Zusi	California	Nal Krite	0.13
Jozzy Abdo	Moscow	Paul Adam	0.13
Julian Green	London	Nal Krite	0.13
Nick Pinardo	New York	James Hoog	0.15

-- 5. write a SQL query to locate those salespeople who do not live in the same city where their customers live and have received a commission of more than 12% from the company. Return Customer Name, customer city, Salesman, salesman city, commission

The screenshot shows the Microsoft SQL Server Enterprise Manager interface. The Object Explorer on the left displays the database structure. The central query window contains the following SQL query:

```
-- 5. write a SQL query to locate those salespeople who do not live in the same city where their customers live and have received a commission
--of more than 12% from the company. Return Customer Name, customer city, Salesman, salesman city, commission
select c.cust_name 'Customer name', c.city 'Customer city', s.name 'Salesman name', s.city 'Salesman city', s.commission
from salesman as s
inner join customer as c
on c.salesman_id = s.salesman_id
where (s.city <> c.city) and (s.commission > .12);
```

The Results pane shows the following data:

Customer name	Customer city	Salesman name	Salesman city	commission
Johny Aldor	Moscow	Paul Adam	Rome	0.13
Graham Zusi	California	Nal Kite	Paris	0.13
Chrag	Ivry	James Hoog	New York	0.15
Julian Green	London	Nal Kite	Paris	0.13
Ahli Kakadya	Bvn	Nal Kite	Paris	0.13

-- 6. write a SQL query to find the details of an order. Return ord_no, ord_date, purch_amt, Customer Name, grade, Salesman, commission

The screenshot shows the Microsoft SQL Server Enterprise Manager interface. The Object Explorer on the left displays the database structure. The central query window contains the following SQL query:

```
-- 6. write a SQL query to find the details of an order. Return ord_no, ord_date, purch_amt, Customer Name, grade, Salesman, commission
select o.ord_no, o.ord_date, o.purch_amt, c.cust_name 'Customer name', c.grade, s.name 'Salesman name', s.commission
from orders o
inner join customer c on c.customer_id = o.customer_id
inner join salesman s on s.salesman_id = o.salesman_id
order by o.purch_amt;
```

The Results pane shows the following data:

ord_no	ord_date	purch_amt	Customer name	grade	Salesman name	commission
70002	2012-10-05	65.26	Nick Rimando	100	James Hoog	0.15
70011	2012-08-17	75.29	Joey Aldor	200	Paul Adam	0.13
70004	2012-08-17	110.5	Geoff Cameron	100	Lauson Hen	0.12
70001	2012-10-05	150.5	Graham Zusi	200	Nal Kite	0.13
70012	2012-06-27	250.45	Julian Green	300	Nal Kite	0.13
70009	2012-09-10	270.65	Brad Guzan	NULL	PI Alex	0.11
70007	2012-09-10	548.5	Graham Zusi	200	Nal Kite	0.13
70010	2012-10-10	1983.43	Fabian Johnson	300	Mo Lyon	0.14
70005	2012-07-27	2400.6	Brad Davis	200	James Hoog	0.15
70003	2012-10-10	2400.4	Geoff Cameron	100	Lauson Hen	0.12
70013	2012-04-28	3045.6	Nick Rimando	100	James Hoog	0.15
70008	2012-09-10	5760	Nick Rimando	100	James Hoog	0.15

-- 7. Write a SQL statement to join the tables salesman, customer and orders so that the same column of each table appears once and only the relational rows are returned.

The screenshot shows the Microsoft SQL Server Enterprise Manager interface. The left pane displays the 'Object Explorer' with the 'PCT787SQLSERVER2017' database selected. The right pane shows a SQL query window with the following query:

```
-- 7. Write a SQL statement to join the tables salesman, customer and orders so that the same column of each table appears once and only the relational rows are returned.
select o.ord_no, o.purch_amt, o.ord_date, c.customer_id, c.cust_name 'Customer name', c.city 'Customer city', c.grade,
s.salesman_id, s.name 'Salesman name', s.city 'Salesman city', s.commission
from orders o
inner join customer c on c.customer_id = o.customer_id
inner join salesman s on s.salesman_id = o.salesman_id;
```

The 'Results' pane shows the output of the query, which is a table with 12 columns: ord_no, purch_amt, ord_date, customer_id, Customer name, Customer city, grade, salesman_id, Salesman name, Salesman city, and commission. The results are sorted by ord_no in ascending order.

ord_no	purch_amt	ord_date	customer_id	Customer name	Customer city	grade	salesman_id	Salesman name	Salesman city	commission	
1	70001	150.5	2012-10-05	3005	Graham Zusi	California	200	5002	Nail Kotte	Paris	0.13
2	70002	65.26	2012-10-05	3002	Nick Rimando	New York	100	5001	James Hoog	New York	0.15
3	70003	2400.4	2012-10-10	3009	Geoff Cameron	Berlin	100	5003	Lauson Hen	San Jose	0.12
4	70004	110.5	2012-08-17	3009	Geoff Cameron	Berlin	100	5003	Lauson Hen	San Jose	0.12
5	70005	2400.6	2012-07-27	3007	Brad Davis	New York	200	5001	James Hoog	New York	0.15
6	70007	948.5	2012-09-10	3005	Graham Zusi	California	200	5002	Nail Kotte	Paris	0.13
7	70008	5760	2012-09-10	3002	Nick Rimando	New York	100	5001	James Hoog	New York	0.15
8	70009	270.65	2012-09-10	3001	Brad Guzan	London	NULL	5005	Pt Alex	London	0.11
9	70010	1983.43	2012-10-10	3004	Fabian Johnson	Paris	300	5006	Mc Lyon	Paris	0.14
10	70011	75.29	2012-08-17	3003	Jozzy Aldor	Moscow	200	5007	Paul Adam	Rome	0.13
11	70012	250.45	2012-08-27	3008	Julian Green	London	300	5002	Nail Kotte	Paris	0.13
12	70013	3045.6	2012-04-25	3002	Nick Rimando	New York	100	5001	James Hoog	New York	0.15

-- 8. write a SQL query to display the customer name, customer city, grade, salesman, salesman city. The results should be sorted by ascending customer_id.

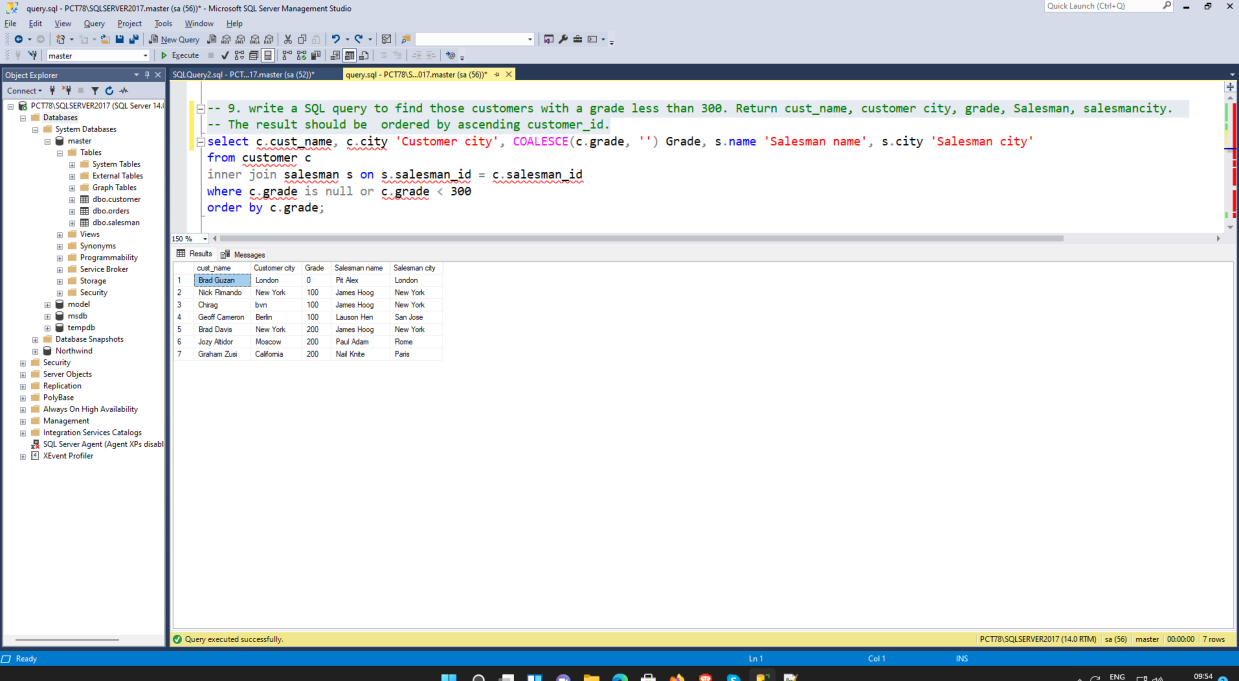
The screenshot shows the Microsoft SQL Server Enterprise Manager interface. The left pane displays the 'Object Explorer' with the 'PCT787SQLSERVER2017' database selected. The right pane shows a SQL query window with the following query:

```
-- 8. write a SQL query to display the customer name, customer city, grade, salesman, salesman city. The results should be sorted by ascending customer_id.
select c.customer_id, c.cust_name 'Customer name', c.city 'Customer city',
ISNULL(c.grade, '') 'Grade', s.salesman_id, s.name 'Salesman name', s.city 'Salesman city'
from customer c
inner join salesman s on s.salesman_id = c.salesman_id
order by c.customer_id;
```

The 'Results' pane shows the output of the query, which is a table with 7 columns: customer_id, Customer name, Customer city, Grade, salesman_id, Salesman name, and Salesman city. The results are sorted by customer_id in ascending order.

customer_id	Customer name	Customer city	Grade	salesman_id	Salesman name	Salesman city	
1	3001	Brad Guzan	London	0	5005	Pt Alex	London
2	3002	Nick Rimando	New York	100	5001	James Hoog	New York
3	3003	Jozzy Aldor	Moscow	200	5007	Paul Adam	Rome
4	3004	Fabian Johnson	Paris	300	5006	Mc Lyon	Paris
5	3005	Graham Zusi	California	200	5002	Nail Kotte	Paris
6	3006	Chirag Bvni	Ivni	100	5001	James Hoog	New York
7	3007	Brad Davis	New York	200	5001	James Hoog	New York
8	3008	Julian Green	London	300	5002	Nail Kotte	Paris
9	3009	Geoff Cameron	Berlin	100	5003	Lauson Hen	San Jose
10	3010	Akhil Kukadia	Bvn	300	5002	Nail Kotte	Paris

-- 9. write a SQL query to find those customers with a grade less than 300. Return cust_name, customer city, grade, Salesman, salesmancity. The result should be ordered by ascending customer_id.



The screenshot shows the Microsoft SQL Server Enterprise Manager interface. The left pane displays the 'Object Explorer' with the 'master' database selected. The central pane shows a SQL query window with the following text:

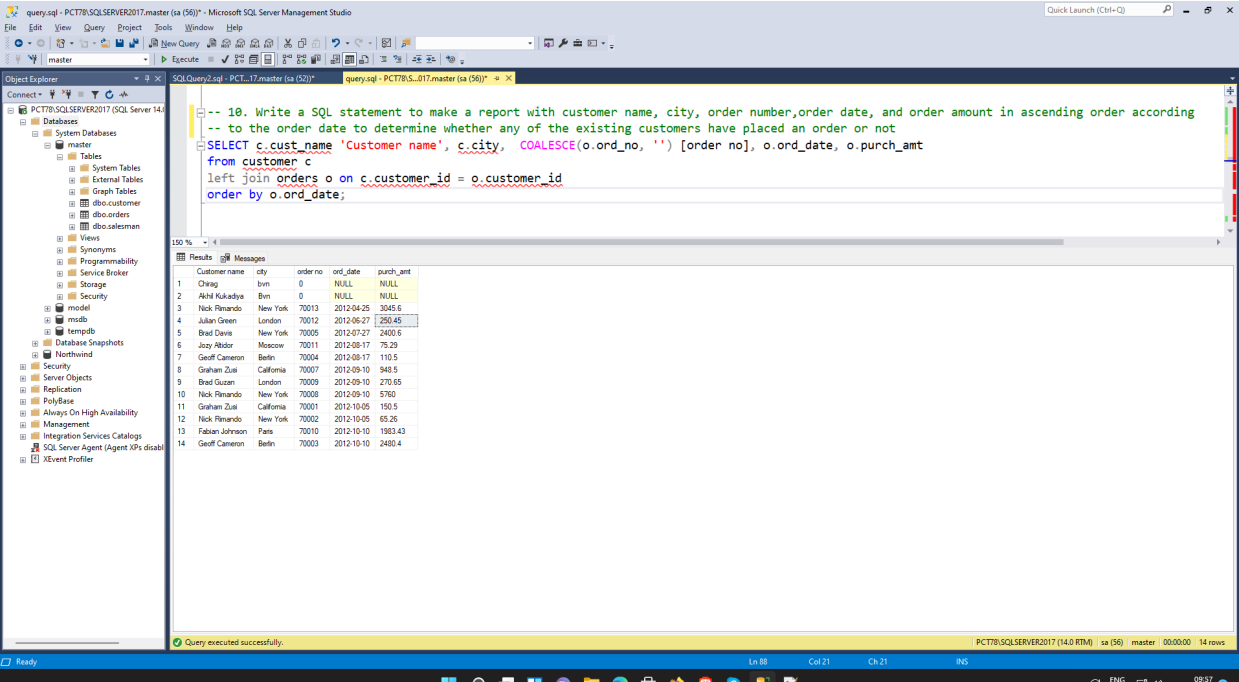
```
-- 9. write a SQL query to find those customers with a grade less than 300. Return cust_name, customer city, grade, Salesman, salesmancity.
-- The result should be ordered by ascending customer_id.
select c.cust_name, c.city 'Customer city', COALESCE(c.grade, '') Grade, s.name 'Salesman name', s.city 'Salesman city'
from customer c
inner join salesman s on s.salesman_id = c.salesman_id
where c.grade is null or c.grade < 300
order by c.grade;
```

The right pane shows the 'Results' tab with the following data:

	cust_name	Customer city	Grade	Salesman name	Salesman city
1	Brad Guzan	London	0	Pit Alex	London
2	Nick Pinardo	New York	100	James Hoog	New York
3	Ching	Ibm	100	James Hoog	New York
4	Geoff Cameron	Berlin	100	Lauson Hen	San Jose
5	Brad Davis	New York	200	James Hoog	New York
6	Jozzy Aldor	Moscow	200	Paul Adam	Rome
7	Graham Zusi	California	200	Nat Note	Pars

The status bar at the bottom indicates 'Query executed successfully.' and 'PCT78:SQLSERVER2017 (14.0 RTM) sa (56) master 00:00:00 7 rows'.

-- 10. Write a SQL statement to make a report with customer name, city, order number, order date, and order amount in ascending order according to the order date to determine whether any of the existing customers have placed an order or not



The screenshot shows the Microsoft SQL Server Enterprise Manager interface. The left pane displays the 'Object Explorer' with the 'master' database selected. The central pane shows a SQL query window with the following text:

```
-- 10. Write a SQL statement to make a report with customer name, city, order number, order date, and order amount in ascending order according
-- to the order date to determine whether any of the existing customers have placed an order or not
SELECT c.cust_name 'Customer name', c.city, COALESCE(o.ord_no, '') [order no], o.ord_date, o.purch_amt
from customer c
left join orders o on c.customer_id = o.customer_id
order by o.ord_date;
```

The right pane shows the 'Results' tab with the following data:

	Customer name	city	order no	ord_date	purch_amt
1	Ching	Ibm	0	NULL	NULL
2	Akhil Kikadiya	Bvn	0	NULL	NULL
3	Nick Pinardo	New York	70013	2012-04-25	3045.6
4	Julian Green	London	70012	2012-05-27	250.45
5	Brad Davis	New York	70005	2012-07-27	2400.6
6	Jozzy Aldor	Moscow	70011	2012-08-17	75.29
7	Geoff Cameron	Berlin	70004	2012-08-17	110.5
8	Graham Zusi	California	70007	2012-09-10	948.5
9	Brad Guzan	London	70009	2012-09-10	270.65
10	Nick Pinardo	New York	70008	2012-09-10	5760
11	Graham Zusi	California	70001	2012-10-05	150.5
12	Nick Pinardo	New York	70002	2012-10-05	65.26
13	Fabian Johnson	Pars	70010	2012-10-10	1983.43
14	Geoff Cameron	Berlin	70003	2012-10-10	2480.4

The status bar at the bottom indicates 'Query executed successfully.' and 'PCT78:SQLSERVER2017 (14.0 RTM) sa (56) master 00:00:00 14 rows'.

-- 11. Write a SQL statement to generate a report with customer name, city, order number, order date, order amount, salesperson name, and commission to determine if any of the existing customers have not placed orders or if they have placed orders through their salesman or by themselves

The screenshot shows the Microsoft SQL Server Enterprise Manager interface. The query window displays the following SQL statement:

```
-- 11. Write a SQL statement to generate a report with customer name, city, order number, order date, order amount, salesperson name, and commission to determine if any of the existing customers have not placed orders or if they have placed orders through their salesman or by themselves
select c.cust_name, c.city [customer city], o.ord_no, o.ord_date, o.purch_amt, s.name [salesman name], s.commission
from customer c
left join orders o on o.customer_id = c.customer_id
left join salesman s on s.salesman_id = o.salesman_id
order by c.cust_name;
```

The Results pane shows the following data:

cust_name	customer city	ord_no	ord_date	purch_amt	salesman name	commission
1 Akhil Kakadia	Bvn	NULL	NULL	NULL	NULL	NULL
2 Brad Davis	New York	70005	2012-07-27	2400.6	James Hoog	0.15
3 Brad Guzan	London	70009	2012-09-10	270.65	Pt Alex	0.11
4 Cheng	bvn	NULL	NULL	NULL	NULL	NULL
5 Fabian Johnson	Paris	70010	2012-10-10	1983.43	Mc Lyon	0.14
6 Geoff Cameron	Berlin	70003	2012-10-10	2480.4	Lauson Hen	0.12
7 Geoff Cameron	Berlin	70004	2012-08-17	110.5	Lauson Hen	0.12
8 Graham Zusi	California	70001	2012-10-05	150.5	Nail Kite	0.13
9 Graham Zusi	California	70007	2012-09-10	948.5	Nail Kite	0.13
10 Jozy Alder	Moscow	70011	2012-08-17	75.29	Paul Adam	0.13
11 Julian Green	London	70012	2012-08-27	201.45	Nail Kite	0.13
12 Nick Rimando	New York	70002	2012-10-05	65.26	James Hoog	0.15
13 Nick Rimando	New York	70008	2012-09-10	5760	James Hoog	0.15
14 Nick Rimando	New York	70013	2012-04-25	3045.6	James Hoog	0.15

-- 12. Write a SQL statement to generate a list in ascending order of salespersons who work either for one or more customers or have not yet joined any of the customers

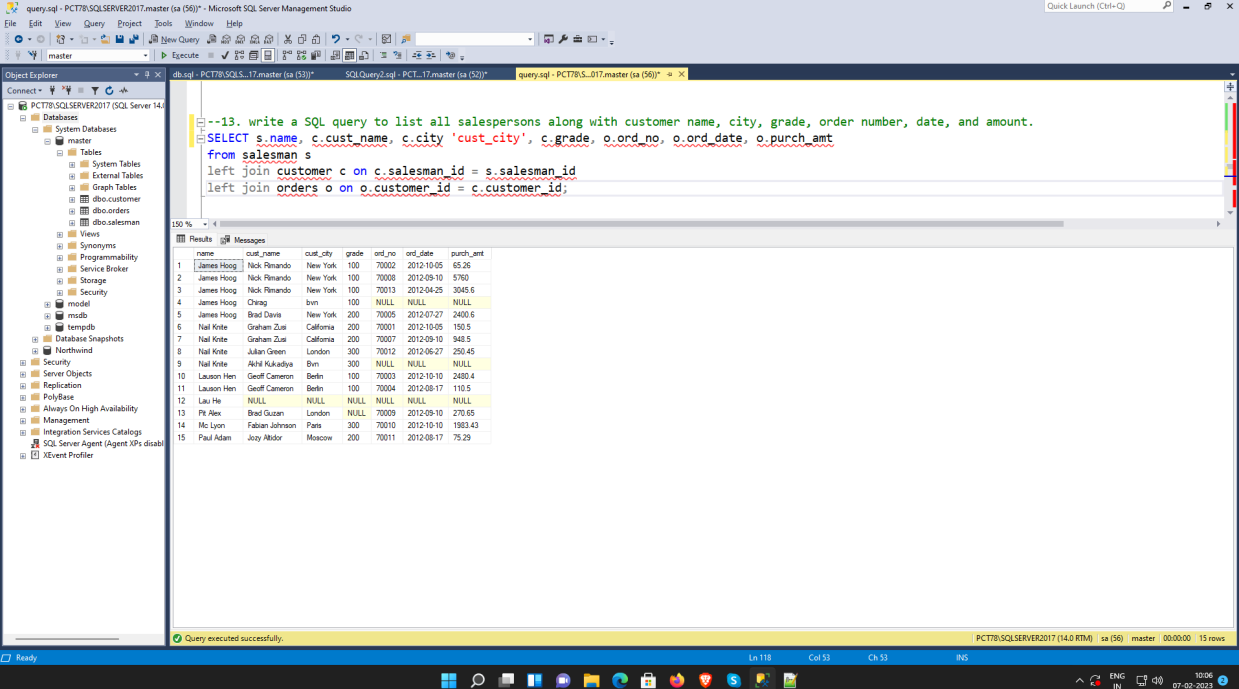
The screenshot shows the Microsoft SQL Server Enterprise Manager interface. The query window displays the following SQL statement:

```
-- 12. Write a SQL statement to generate a list in ascending order of salespersons who work either for one or more customers or have not yet joined any of the customers
select s.name 'Salesman', s.city, c.cust_name [customer], c.city
from salesman s
left join customer c
on c.salesman_id = s.salesman_id
order by s.salesman_id;
```

The Results pane shows the following data:

Salesman	city	customer	city
1 James Hoog	New York	Nick Rimando	New York
2 James Hoog	New York	Cheng	bvn
3 James Hoog	New York	Brad Davis	New York
4 Nail Kite	Paris	Graham Zusi	California
5 Nail Kite	Paris	Julian Green	London
6 Nail Kite	Paris	Akhil Kakadia	Bvn
7 Lauson Hen	San Jose	Geoff Cameron	Berlin
8 Lau He	San Jose	NULL	NULL
9 Pt Alex	London	Brad Guzan	London
10 Mc Lyon	Paris	Fabian Johnson	Paris
11 Paul Adam	Rome	Jozy Alder	Moscow

--13. write a SQL query to list all salespersons along with customer name, city, grade, order number, date, and amount.



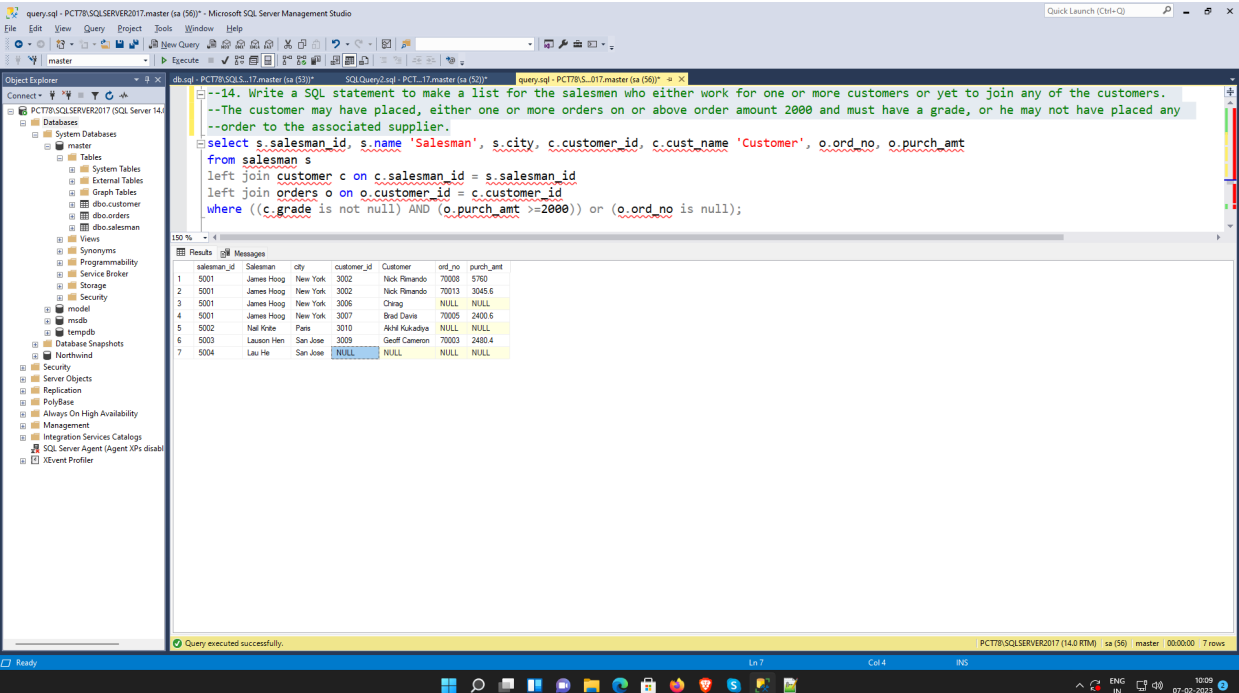
The screenshot shows the Microsoft SQL Server Enterprise Manager interface. The left pane displays the 'Object Explorer' with the 'master' database selected. The right pane shows a SQL query window with the following query:

```
--13. write a SQL query to list all salespersons along with customer name, city, grade, order number, date, and amount.
SELECT s.name, c.cust_name, c.city 'cust_city', c.grade, o.ord_no, o.ord_date, o.purch_amt
FROM salesman s
LEFT JOIN customer c ON c.salesman_id = s.salesman_id
LEFT JOIN orders o ON o.customer_id = c.customer_id;
```

The query results are displayed in a table with 15 rows and 7 columns:

	name	cust_name	cust_city	grade	ord_no	ord_date	purch_amt
1	James Hoog	Nick Rimando	New York	100	70002	2012-10-05	65.26
2	James Hoog	Nick Rimando	New York	100	70008	2012-09-10	5760
3	James Hoog	Nick Rimando	New York	100	70013	2012-04-25	3045.6
4	James Hoog	Chrag	bvn	100	NULL	NULL	NULL
5	James Hoog	Brad Davis	New York	200	70005	2012-07-07	2400.6
6	Nail Krite	Graham Zusi	California	200	70001	2012-10-05	150.5
7	Nail Krite	Graham Zusi	California	200	70007	2012-09-10	948.5
8	Nail Krite	Julian Green	London	300	70012	2012-06-27	250.45
9	Nail Krite	Ahli Kukadya	Bvn	300	NULL	NULL	NULL
10	Lauson Hen	Geoff Cameron	Berlin	100	70003	2012-10-10	2480.4
11	Lauson Hen	Geoff Cameron	Berlin	100	70004	2012-08-17	110.5
12	Lau He	NULL	NULL	NULL	NULL	NULL	NULL
13	PL Alex	Brad Guzan	London	NULL	70009	2012-09-10	270.65
14	He Lynn	Fabian Johnson	Paris	300	70010	2012-10-10	1863.43
15	Paul Adam	Jozzy Aldor	Moscow	200	70011	2012-08-17	75.29

--14. Write a SQL statement to make a list for the salesmen who either work for one or more customers or yet to join any of the customers. The customer may have placed, either one or more orders on or above order amount 2000 and must have a grade, or he may not have placed any order to the associated supplier.



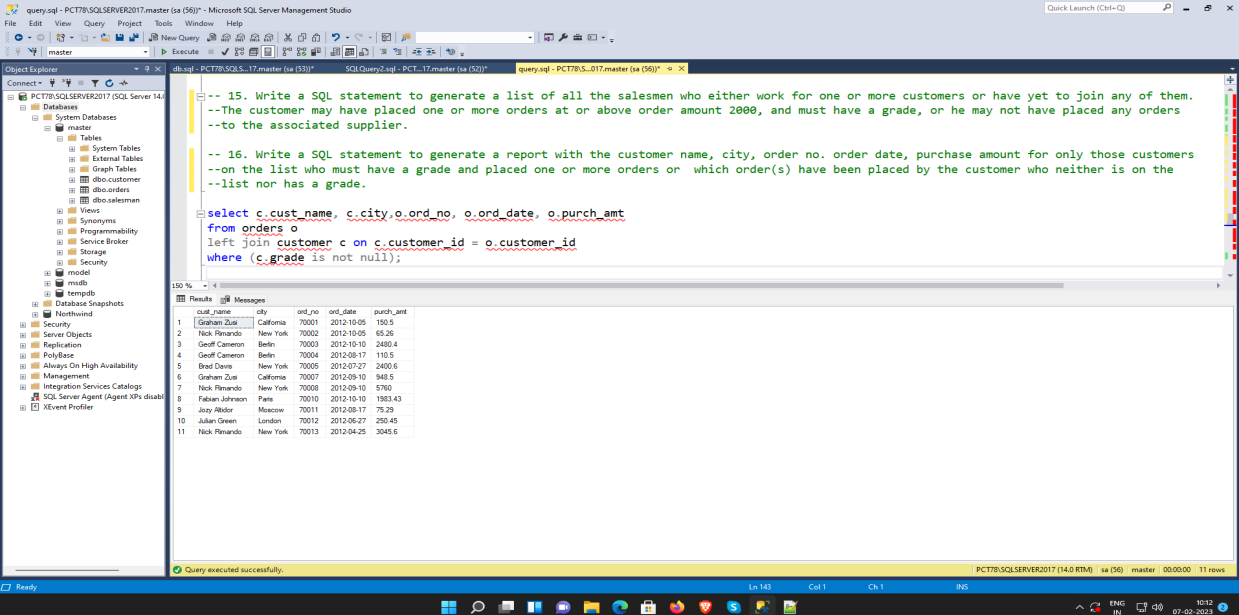
The screenshot shows the Microsoft SQL Server Enterprise Manager interface. The left pane displays the 'Object Explorer' with the 'master' database selected. The right pane shows a SQL query window with the following query:

```
--14. Write a SQL statement to make a list for the salesmen who either work for one or more customers or yet to join any of the customers.
--The customer may have placed, either one or more orders on or above order amount 2000 and must have a grade, or he may not have placed any
--order to the associated supplier.
select s.salesman_id, s.name 'Salesman', s.city, c.customer_id, c.cust_name 'Customer', o.ord_no, o.purch_amt
from salesman s
left join customer c on c.salesman_id = s.salesman_id
left join orders o on o.customer_id = c.customer_id
where ((c.grade is not null) AND (o.purch_amt >=2000)) or (o.ord_no is null);
```

The query results are displayed in a table with 7 rows and 7 columns:

	salesman_id	Salesman	city	customer_id	Customer	ord_no	purch_amt
1	5001	James Hoog	New York	3002	Nick Rimando	70008	5760
2	5001	James Hoog	New York	3002	Nick Rimando	70013	3045.6
3	5001	James Hoog	New York	3006	Chrag	NULL	NULL
4	5001	James Hoog	New York	3007	Brad Davis	70005	2400.6
5	5002	Nail Krite	Paris	3010	Ahli Kukadya	NULL	NULL
6	5003	Lauson Hen	San Jose	3009	Geoff Cameron	70003	2480.4
7	5004	Lau He	San Jose	NULL	NULL	NULL	NULL

- 15. Write a SQL statement to generate a list of all the salesmen who either work for one or more customers or have yet to join any of them.
 --The customer may have placed one or more orders at or above order amount 2000, and must have a grade, or he may not have placed any orders
 --to the associated supplier.
- 16. Write a SQL statement to generate a report with the customer name, city, order no. order date, purchase amount for only those customers
 --on the list who must have a grade and placed one or more orders or which order(s) have been placed by the customer who neither is on the
 --list nor has a grade.



The screenshot shows the Microsoft SQL Server Enterprise Manager interface. The left pane displays the 'Object Explorer' with the 'master' database selected. The right pane shows a SQL query window with the following text:

```
-- 15. Write a SQL statement to generate a list of all the salesmen who either work for one or more customers or have yet to join any of them.
--The customer may have placed one or more orders at or above order amount 2000, and must have a grade, or he may not have placed any orders
--to the associated supplier.

-- 16. Write a SQL statement to generate a report with the customer name, city, order no. order date, purchase amount for only those customers
--on the list who must have a grade and placed one or more orders or which order(s) have been placed by the customer who neither is on the
--list nor has a grade.

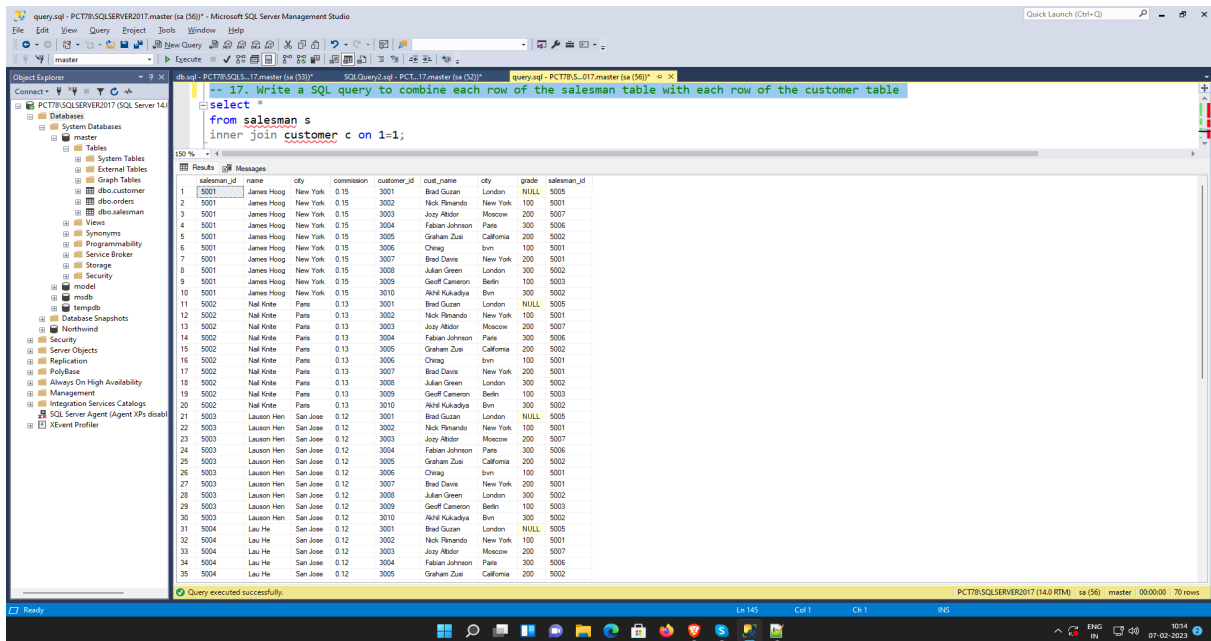
select c.cust_name, c.city, o.ord_no, o.ord_date, o.purch_amt
from orders o
left join customer c on c.customer_id = o.customer_id
where (c.grade is not null);
```

Below the query, the 'Results' pane displays the following data:

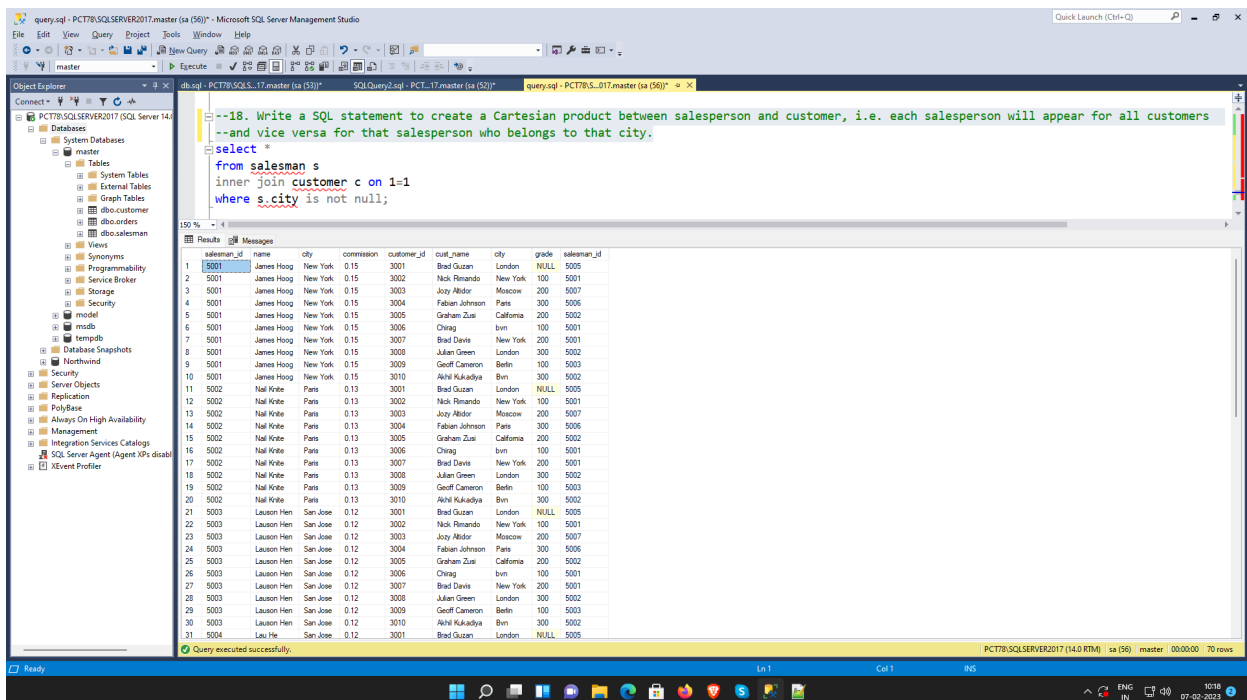
	cust_name	city	ord_no	ord_date	purch_amt
1	Graham Zusi	California	70001	2012-10-05	150.5
2	Nick Rimando	New York	70002	2012-10-05	65.25
3	Geoff Cameron	Berlin	70003	2012-10-10	2480.4
4	Geoff Cameron	Berlin	70004	2012-09-17	110.5
5	Bred Davis	New York	70005	2012-07-27	2400.6
6	Graham Zusi	California	70007	2012-09-10	945.5
7	Nick Rimando	New York	70008	2012-09-10	5760
8	Fabian Johnson	Paris	70010	2012-10-10	1983.43
9	Jens Rieder	Moscow	70011	2012-08-17	75.29
10	Julian Green	London	70012	2012-06-27	250.45
11	Nick Rimando	New York	70013	2012-04-25	3045.5

The status bar at the bottom indicates 'Query executed successfully.' and '11 rows'.

- 17. Write a SQL query to combine each row of the salesman table with each row of the customer table



--18. Write a SQL statement to create a Cartesian product between salesperson and customer, i.e. each salesperson will appear for all customers and vice versa for that salesperson who belongs to that city.



-- 19. Write a SQL statement to create a Cartesian product between salesperson and customer, i.e. each salesperson will appear for every customer and vice versa for those salesmen who belong to a city and customers who require a grade

The screenshot shows the Microsoft SQL Server Management Studio interface. The query editor contains the following SQL statement:

```
-- 19. Write a SQL statement to create a Cartesian product between salesperson and
-- customer, i.e. each salesperson will appear for every customer and vice versa for
-- those salesmen who belong to a city and customers who require a grade
select *
from salesperson s
inner join customer c on 1=1
where (s.city is not null) and (c.grade is not null);
```

The Results pane displays the output of the query, showing a Cartesian product of salespersons and customers. The table has columns: salesperson_id, name, city, commission, customer_id, cust_name, city, grade, and salesman_id. The results show 29 rows of data.

salesperson_id	name	city	commission	customer_id	cust_name	city	grade	salesman_id
1	James Hoog	New York	0.15	3002	Nick Rimando	New York	100	5001
2	James Hoog	New York	0.15	3003	Joy Abidor	Moscow	200	5007
3	James Hoog	New York	0.15	3004	Fabian Johnson	Paris	300	5006
4	James Hoog	New York	0.15	3005	Graham Zue	California	200	5002
5	James Hoog	New York	0.15	3006	Chang	Ivni	100	5001
6	James Hoog	New York	0.15	3007	Brad Davis	New York	200	5001
7	James Hoog	New York	0.15	3008	Julian Green	London	300	5002
8	James Hoog	New York	0.15	3009	Geoff Cameron	Berlin	100	5003
9	James Hoog	New York	0.15	3010	Akhil Kukadrya	Bvn	300	5002
10	Nail Krite	Paris	0.13	3002	Nick Rimando	New York	100	5001
11	Nail Krite	Paris	0.13	3003	Joy Abidor	Moscow	200	5007
12	Nail Krite	Paris	0.13	3004	Fabian Johnson	Paris	300	5006
13	Nail Krite	Paris	0.13	3005	Graham Zue	California	200	5002
14	Nail Krite	Paris	0.13	3006	Chang	Ivni	100	5001
15	Nail Krite	Paris	0.13	3007	Brad Davis	New York	200	5001
16	Nail Krite	Paris	0.13	3008	Julian Green	London	300	5002
17	Nail Krite	Paris	0.13	3009	Geoff Cameron	Berlin	100	5003
18	Nail Krite	Paris	0.13	3010	Akhil Kukadrya	Bvn	300	5002
19	Lauson Hen	San Jose	0.12	3002	Nick Rimando	New York	100	5001
20	Lauson Hen	San Jose	0.12	3003	Joy Abidor	Moscow	200	5007
21	Lauson Hen	San Jose	0.12	3004	Fabian Johnson	Paris	300	5006
22	Lauson Hen	San Jose	0.12	3005	Graham Zue	California	200	5002
23	Lauson Hen	San Jose	0.12	3006	Chang	Ivni	100	5001
24	Lauson Hen	San Jose	0.12	3007	Brad Davis	New York	200	5001
25	Lauson Hen	San Jose	0.12	3008	Julian Green	London	300	5002
26	Lauson Hen	San Jose	0.12	3009	Geoff Cameron	Berlin	100	5003
27	Lauson Hen	San Jose	0.12	3010	Akhil Kukadrya	Bvn	300	5002
28	Lau He	San Jose	0.12	3002	Nick Rimando	New York	100	5001
29	Lau He	San Jose	0.12	3003	Joy Abidor	Moscow	200	5007

-- 20. Write a SQL statement to make a Cartesian product between salesman and customer i.e. each salesman will appear for all customers and vice versa for those salesmen who must belong to a city which is not the same as his customer and the customers should have their own grade

The screenshot shows the Microsoft SQL Server Management Studio interface. The query editor contains the following SQL statement:

```
-- 20. Write a SQL statement to make a Cartesian product between salesman and customer i.e. each salesman will appear for all customers and
-- vice versa for those salesmen who must belong to a city which is not the same as his customer and the customers should have their own grade
select *
from salesman s
inner join customer c on 1=1
where (s.city is not null) AND (s.city <> c.city) AND (c.grade is not null);
```

The Results pane displays the output of the query, showing a Cartesian product of salesmen and customers with additional filters. The table has columns: salesman_id, name, city, commission, customer_id, cust_name, city, grade, and salesman_id. The results show 29 rows of data.

salesman_id	name	city	commission	customer_id	cust_name	city	grade	salesman_id
1	James Hoog	New York	0.15	3003	Joy Abidor	Moscow	200	5007
2	James Hoog	New York	0.15	3004	Fabian Johnson	Paris	300	5006
3	James Hoog	New York	0.15	3005	Graham Zue	California	200	5002
4	James Hoog	New York	0.15	3006	Chang	Ivni	100	5001
5	James Hoog	New York	0.15	3008	Julian Green	London	300	5002
6	James Hoog	New York	0.15	3009	Geoff Cameron	Berlin	100	5003
7	James Hoog	New York	0.15	3010	Akhil Kukadrya	Bvn	300	5002
8	Nail Krite	Paris	0.13	3002	Nick Rimando	New York	100	5001
9	Nail Krite	Paris	0.13	3003	Joy Abidor	Moscow	200	5007
10	Nail Krite	Paris	0.13	3005	Graham Zue	California	200	5002
11	Nail Krite	Paris	0.13	3006	Chang	Ivni	100	5001
12	Nail Krite	Paris	0.13	3007	Brad Davis	New York	200	5001
13	Nail Krite	Paris	0.13	3008	Julian Green	London	300	5002
14	Nail Krite	Paris	0.13	3009	Geoff Cameron	Berlin	100	5003
15	Nail Krite	Paris	0.13	3010	Akhil Kukadrya	Bvn	300	5002
16	Lauson Hen	San Jose	0.12	3002	Nick Rimando	New York	100	5001
17	Lauson Hen	San Jose	0.12	3003	Joy Abidor	Moscow	200	5007
18	Lauson Hen	San Jose	0.12	3004	Fabian Johnson	Paris	300	5006
19	Lauson Hen	San Jose	0.12	3005	Graham Zue	California	200	5002
20	Lauson Hen	San Jose	0.12	3006	Chang	Ivni	100	5001
21	Lauson Hen	San Jose	0.12	3007	Brad Davis	New York	200	5001
22	Lauson Hen	San Jose	0.12	3008	Julian Green	London	300	5002
23	Lauson Hen	San Jose	0.12	3009	Geoff Cameron	Berlin	100	5003
24	Lauson Hen	San Jose	0.12	3010	Akhil Kukadrya	Bvn	300	5002
25	Lau He	San Jose	0.12	3002	Nick Rimando	New York	100	5001
26	Lau He	San Jose	0.12	3003	Joy Abidor	Moscow	200	5007
27	Lau He	San Jose	0.12	3004	Fabian Johnson	Paris	300	5006
28	Lau He	San Jose	0.12	3005	Graham Zue	California	200	5002
29	Lau He	San Jose	0.12	3006	Chang	Ivni	100	5001

