**Assignment:2**

**First i have created database sales**

**create database sales**

**Then I have created three tables and added entries using following queries.**

**create table**

**salesman(salesman\_id int primary key, name varchar(50), city varchar(30),commision float)**

**insert into salesman values(5001, 'James Hoog', 'New York', 0.15)**

**insert into salesman values(5002, 'Nail Knite', 'Paris', 0.13)**

**insert into salesman values(5005, 'Pit Alex', 'London', 0.11)**

**insert into salesman values(5006, 'Mc Lyon', 'Paris', 0.14)**

**insert into salesman values(5007, 'Paul Adam', 'Rome', 0.13)**

**insert into salesman values(5003, 'Lauson Hen', 'San Jose', 0.12)**

**create table customer(customer\_id int primary key, cust\_name varchar(50), city varchar(30),grade int,salesman\_id int foreign key REFERENCES salesman(salesman\_id))**

**insert into customer values(3002, 'Nick Rimando', 'New York', 100, 5001)**

**insert into customer values(3007, 'Brad Davis', 'New York', 200, 5001)**

**insert into customer values(3005, 'Graham Zusi', 'California', 200, 5002)**

**insert into customer values(3008, 'Julian Green', 'London', 300, 5002)**

**insert into customer values(3004, 'Fabian Johnson', 'Paris', 300, 5006)**

**insert into customer values(3009, 'Geoff Cameron', 'Berlin', 100, 5003)**

**insert into Customer values(3003, 'Jozy Altidor', 'Moscow', 200, 5007)**

**insert into Customer values(3001, 'Brad Guzan', 'London', NULL, 5005)**

**create table orders(**

**order\_no numeric(5) primary key, purch\_amt decimal(8,2), order\_date date,**

**customer\_id int foreign key references customer(customer\_id), salesman\_id int foreign key references salesman(salesman\_id))**

**insert into orders values(70001, 150.5, '2012-10-05', 3005, 5002)**

**insert into orders values(70009, 270.65, '2012-09-10', 3001, 5005)**

**insert into orders values(70002, 65.26, '2012-10-05', 3002, 5001)**

**insert into orders values(70004, 110.5, '2012-08-17', 3009, 5003)**

**insert into orders values(70007, 948.5, '2012-09-10', 3005, 5002)**

**insert into orders values(70005, 2400.6, '2012-07-27', 3007, 5001)**

**insert into orders values(70008, 5760, '2012-09-10', 3002, 5001)**

**insert into orders values(70010, 1983.43, '2012-10-10', 3004, 5006)**

**insert into orders values(70003, 2480.4, '2012-10-10', 3009, 5003)**

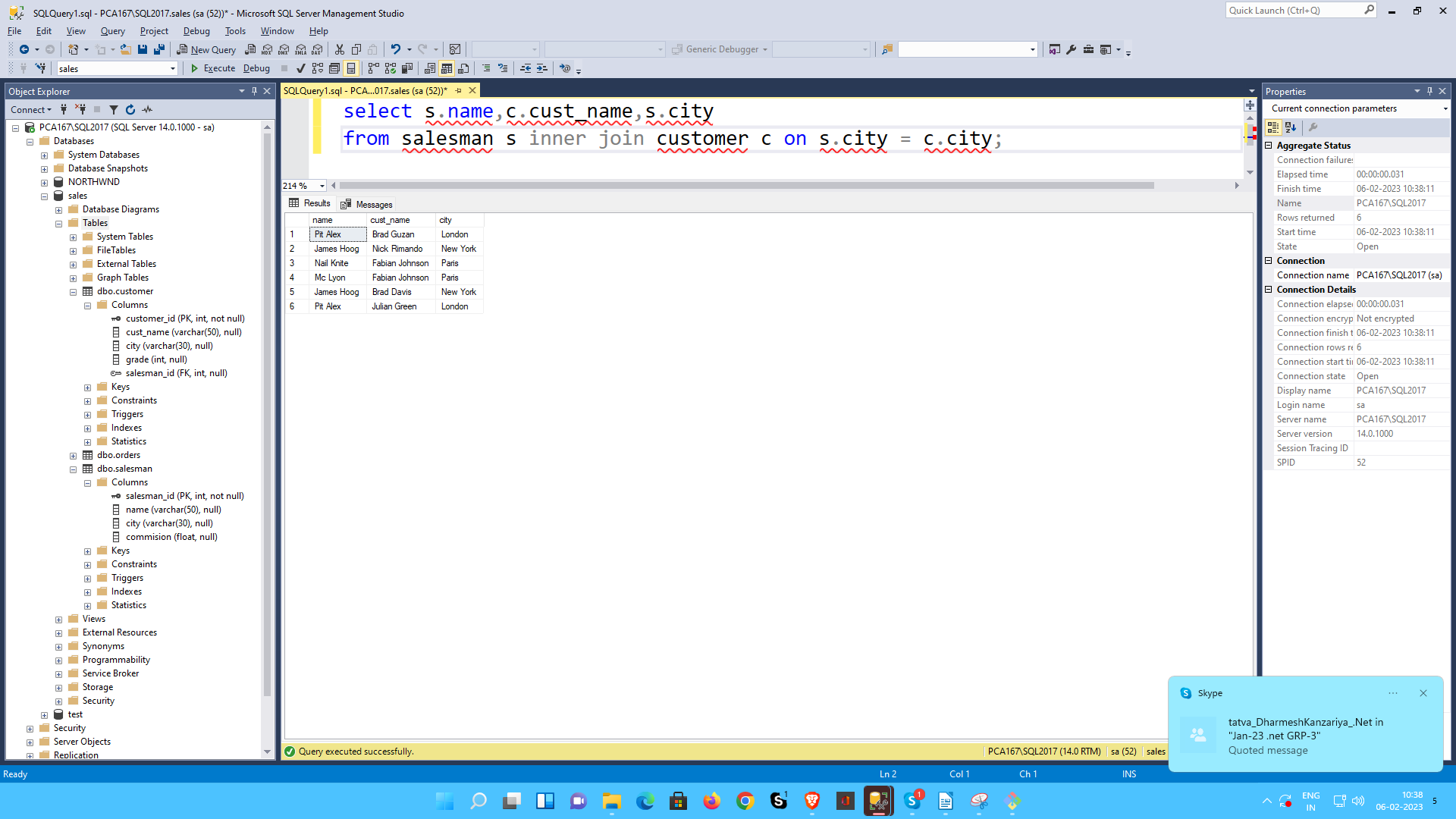
**insert into orders values(70012, 250.45, '2012-06-27', 3008, 5002)**

**insert into orders values(70011, 75.29, '2012-08-17', 3003, 5007)**

**insert into orders values(70013, 3045.6, '2012-04-25', 3002, 5001)**

**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,c.cust\_name,s.city from salesman s inner join customer c on s.city = c.city;**

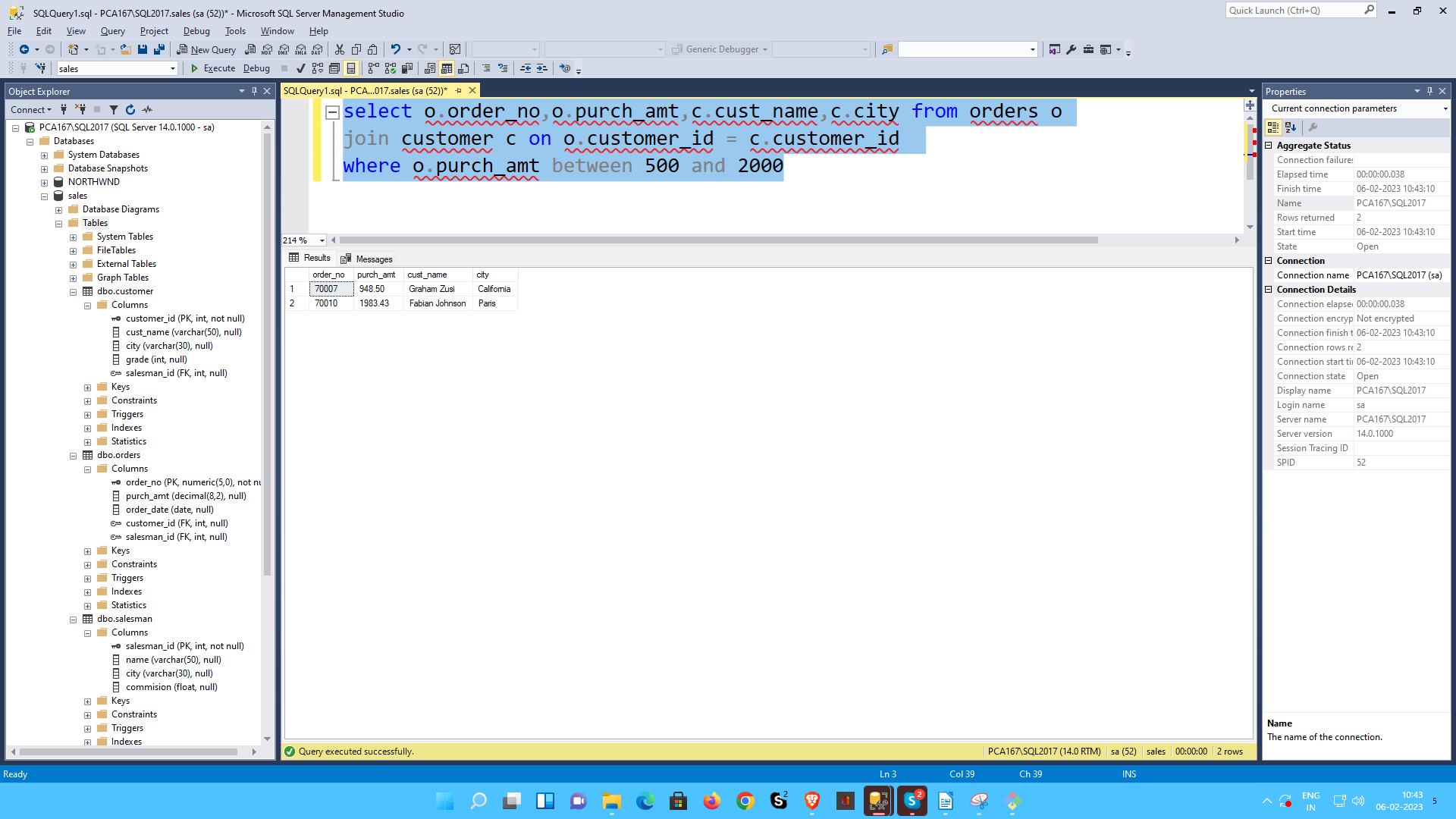
****

**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.order\_no,o.purch\_amt,c.cust\_name,c.city from orders o**

**join customer c on o.customer\_id = c.customer\_id**

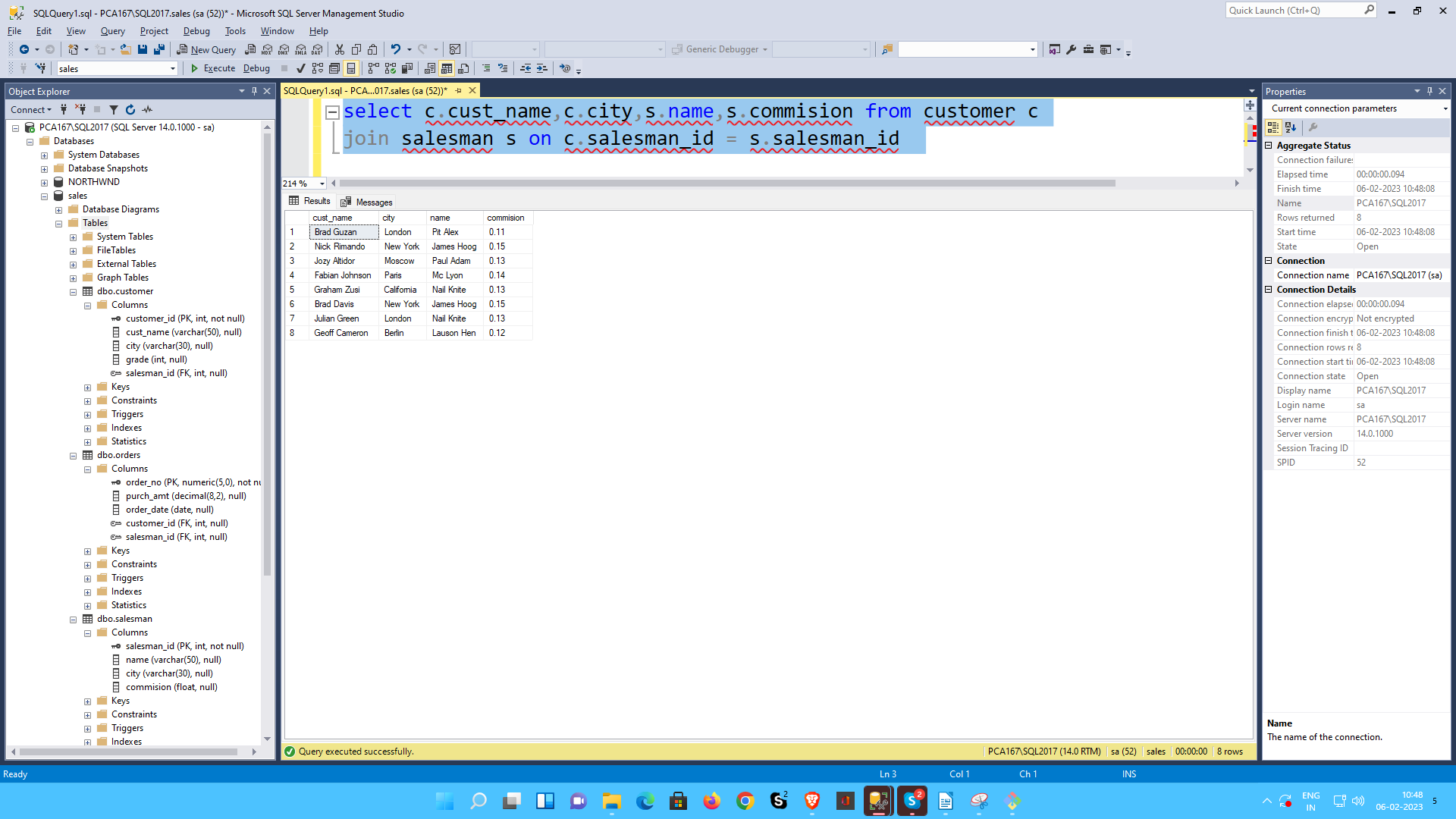
**where o.purch\_amt between 500 and 2000**

****

**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,c.city,s.name,s.commision from customer c**

**join salesman s on c.salesman\_id = s.salesman\_id**

****

**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,c.city,s.name AS "salespeople",s.commision**

**from customer c join salesman s on c.salesman\_id=s.salesman\_id**

**where s.commision>0.12**

****

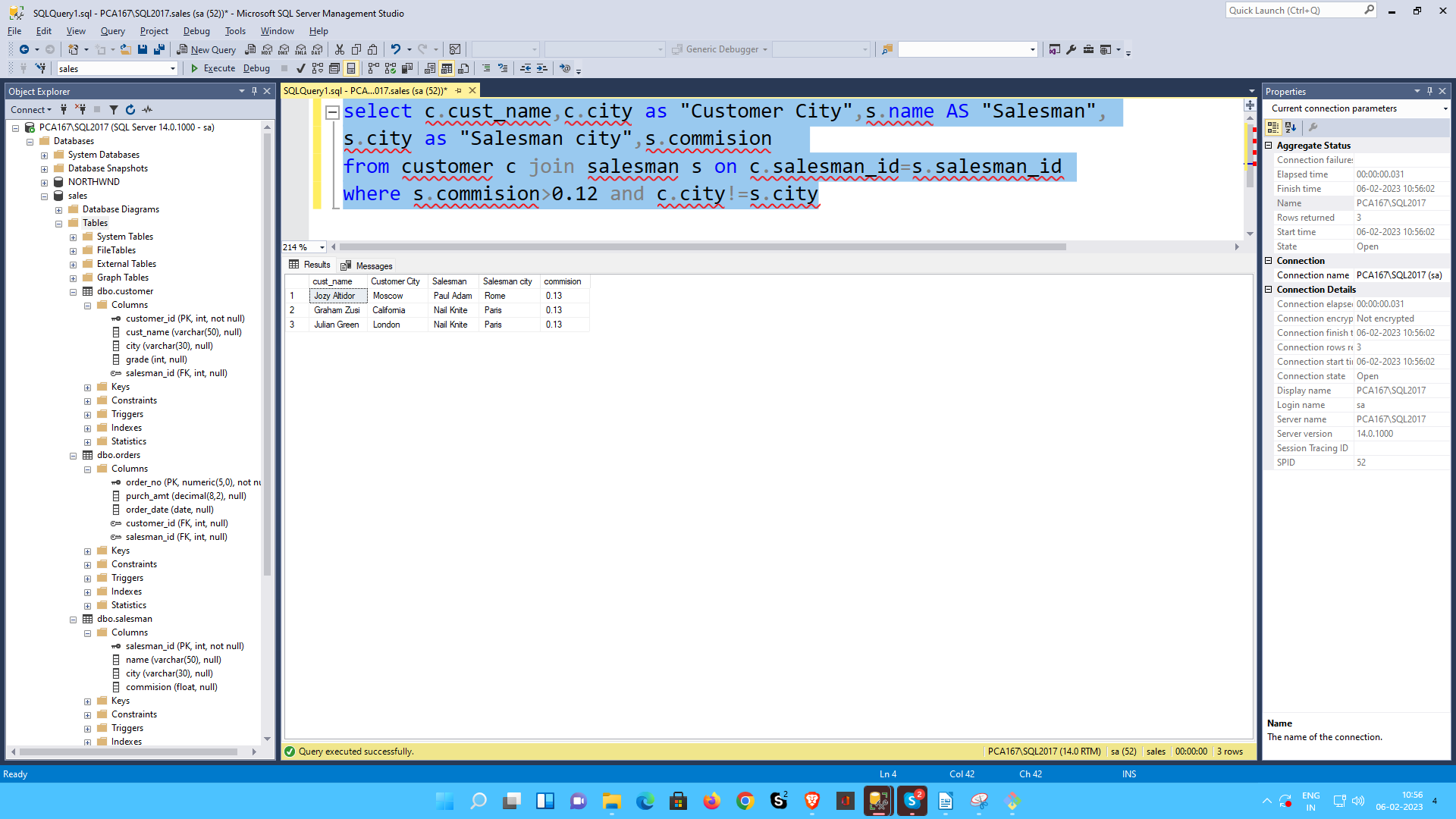
**5. write a SQL query to locate those salespeople who do not live in the same city where their customers live and have received a commission of more than 12% from the company. Return Customer Name, customer city, Salesman, salesman city, commission**

**select c.cust\_name,c.city as "Customer City",s.name AS "Salesman",**

**s.city as "Salesman city",s.commision**

**from customer c join salesman s on c.salesman\_id=s.salesman\_id**

**where s.commision>0.12 and c.city!=s.city**

****

**6. write a SQL query to find the details of an order. Return ord\_no, ord\_date, purch\_amt, Customer Name, grade, Salesman, commission**

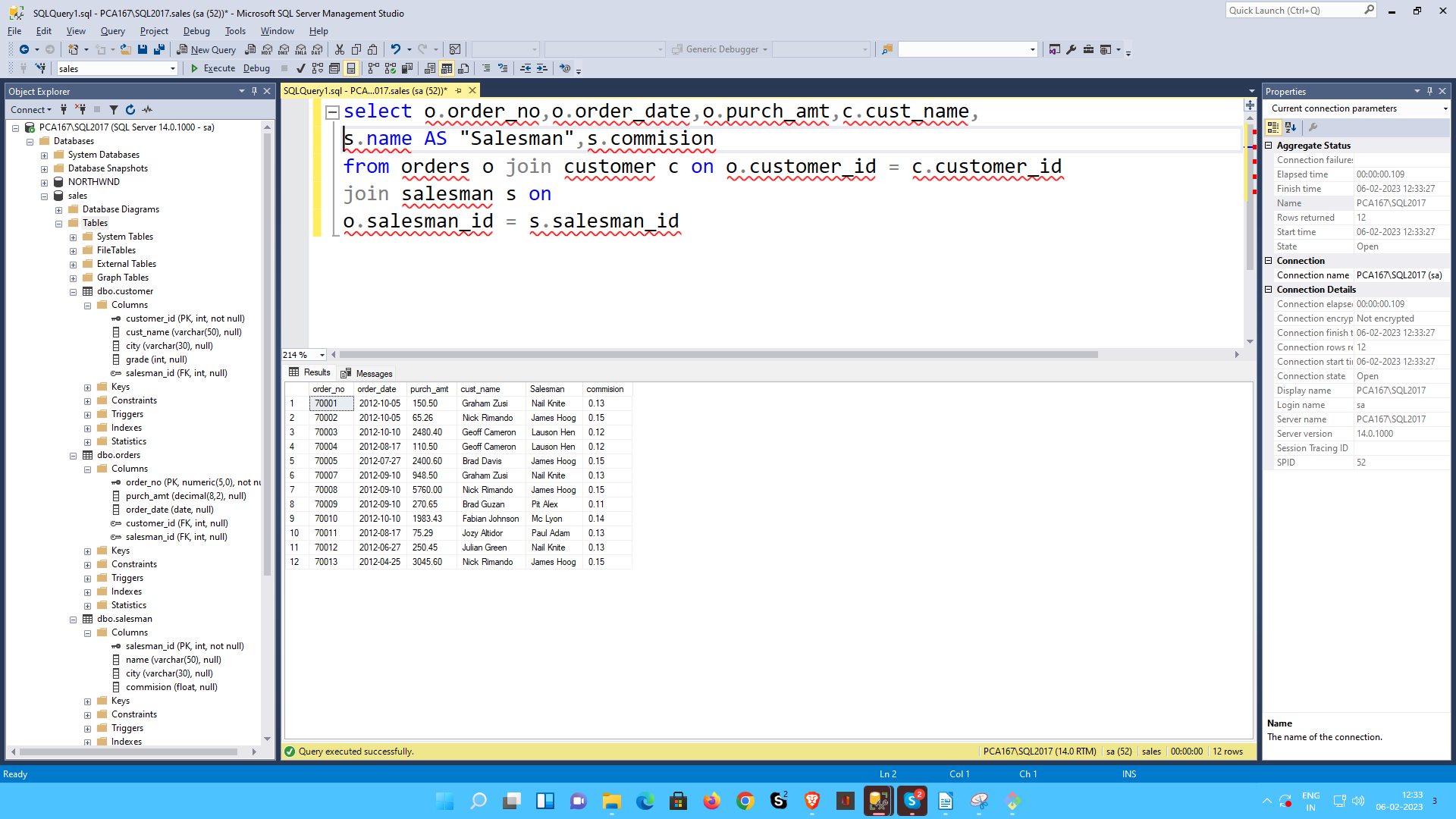
**select o.order\_no,o.order\_date,o.purch\_amt,c.cust\_name,**

**s.name AS "Salesman",s.commision**

**from orders o join customer c on o.customer\_id = c.customer\_id**

**join salesman s on**

**o.salesman\_id = s.salesman\_id**

****

**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.**

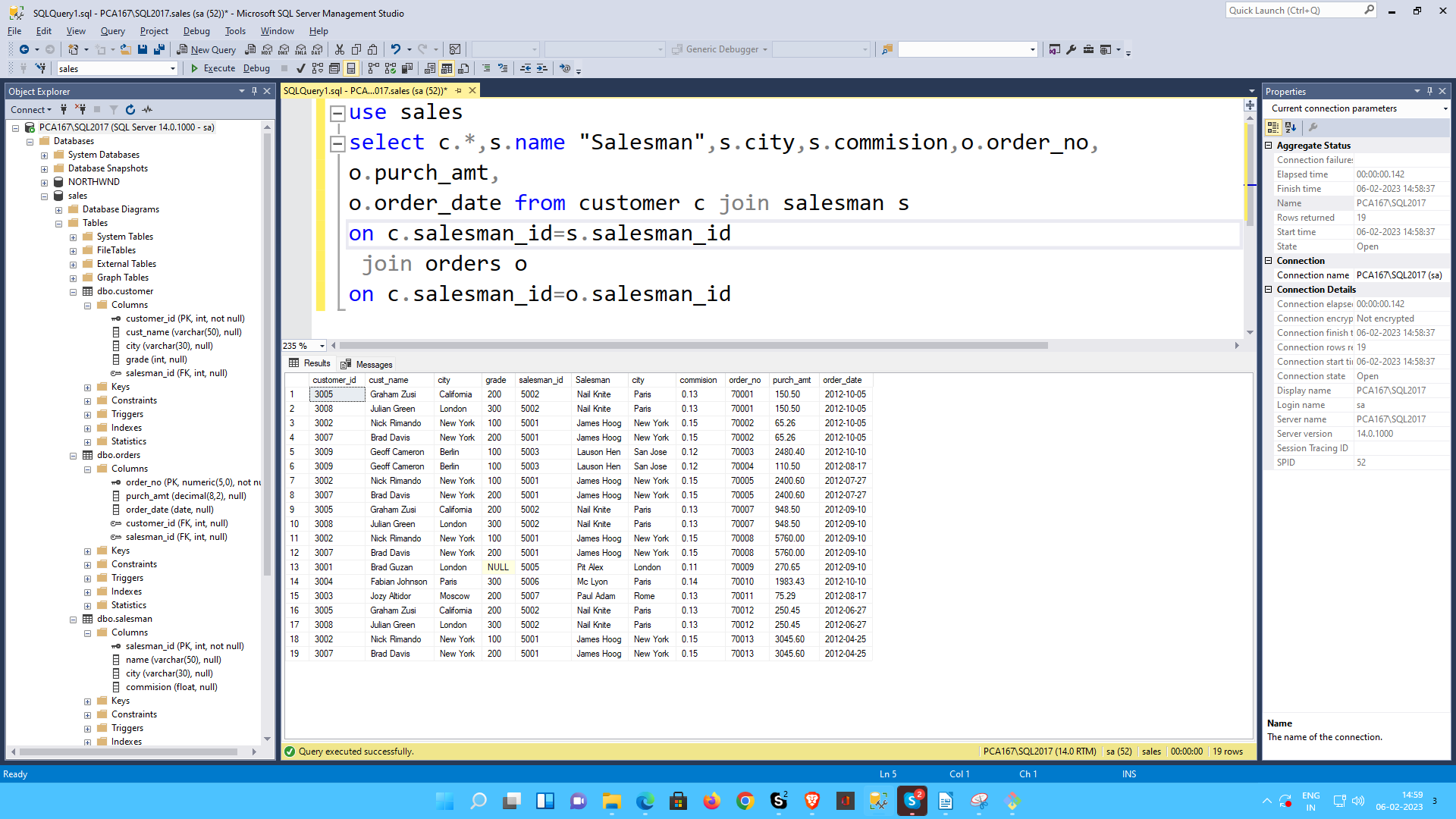
**use sales**

**select c.\*,s.name "Salesman",s.city,s.commision,o.order\_no,o.purch\_amt,**

**o.order\_date from customer c join salesman s on c.salesman\_id=s.salesman\_id**

**join orders o**

**on c.salesman\_id=o.salesman\_id**

****

**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.**

**use sales**

**select c.cust\_name,c.city,c.grade,**

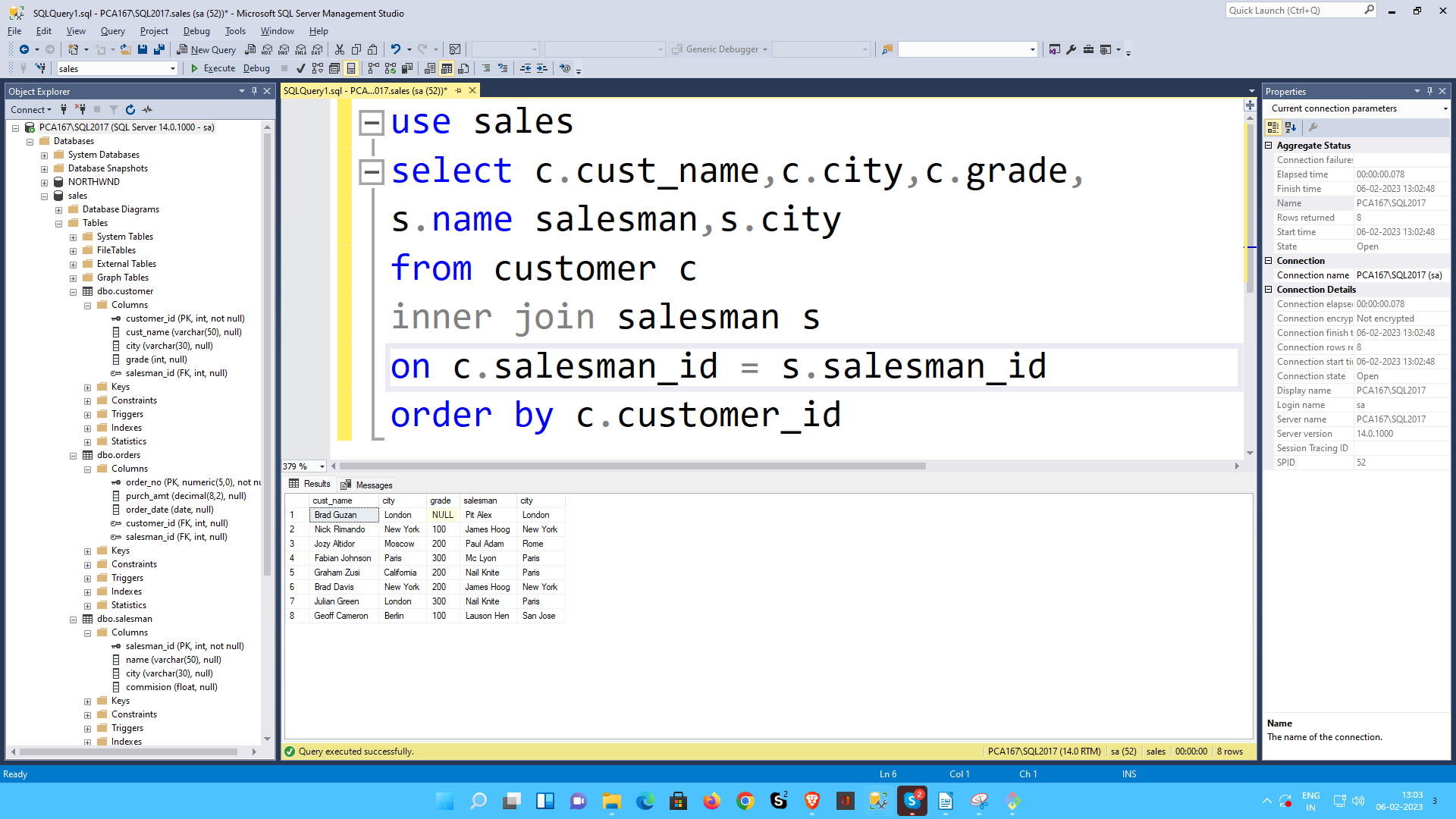
**s.name salesman,s.city**

**from customer c**

**inner join salesman s**

**on c.salesman\_id = s.salesman\_id**

**order by c.customer\_id**

****

**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.**

**use sales**

**select c.cust\_name,c.city,c.grade,**

**s.name salesman,s.city**

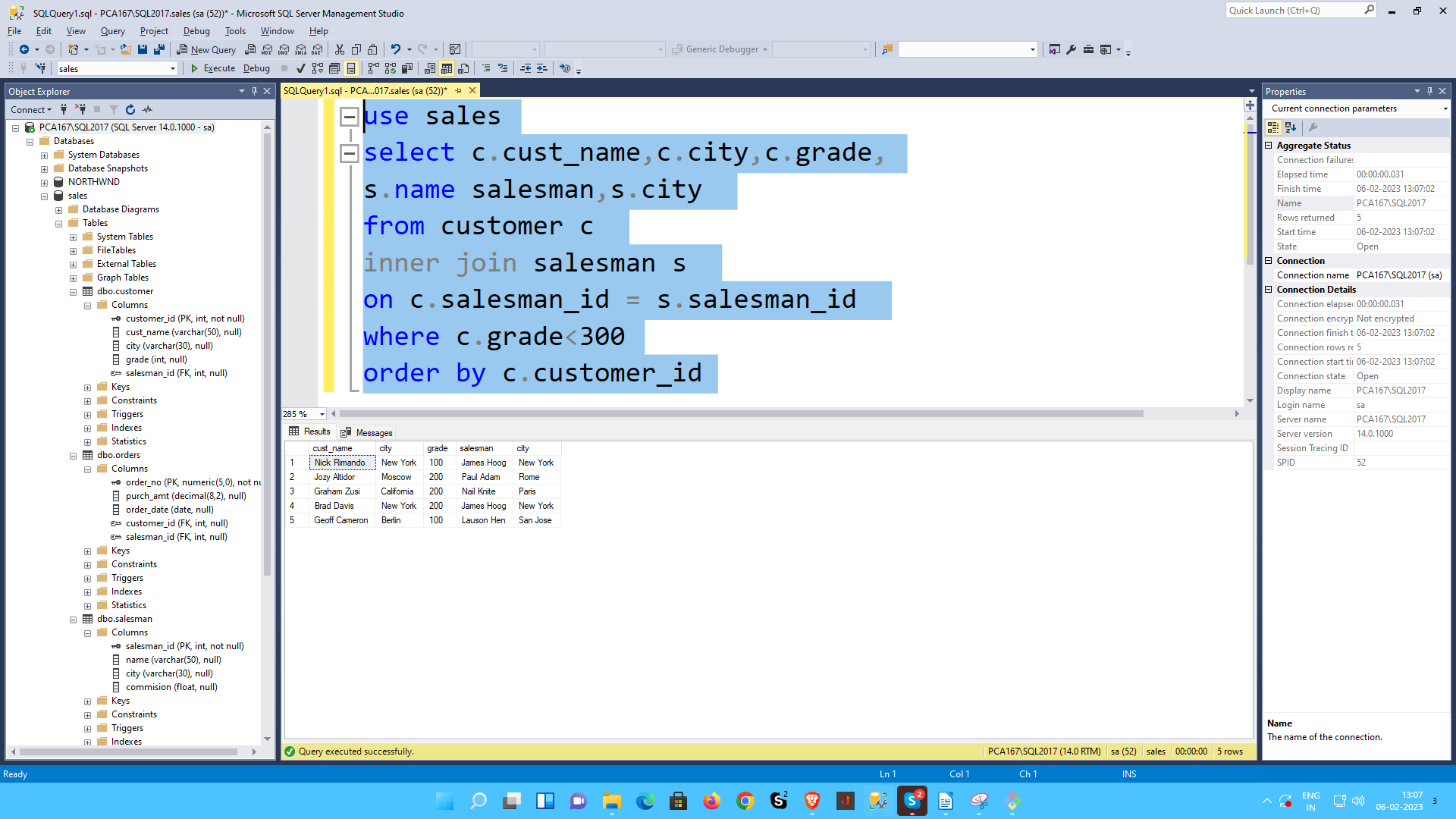
**from customer c**

**inner join salesman s**

**on c.salesman\_id = s.salesman\_id**

**where c.grade<300**

**order by c.customer\_id**

****

**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**

**use sales**

**select c.cust\_name,c.city,c.grade,**

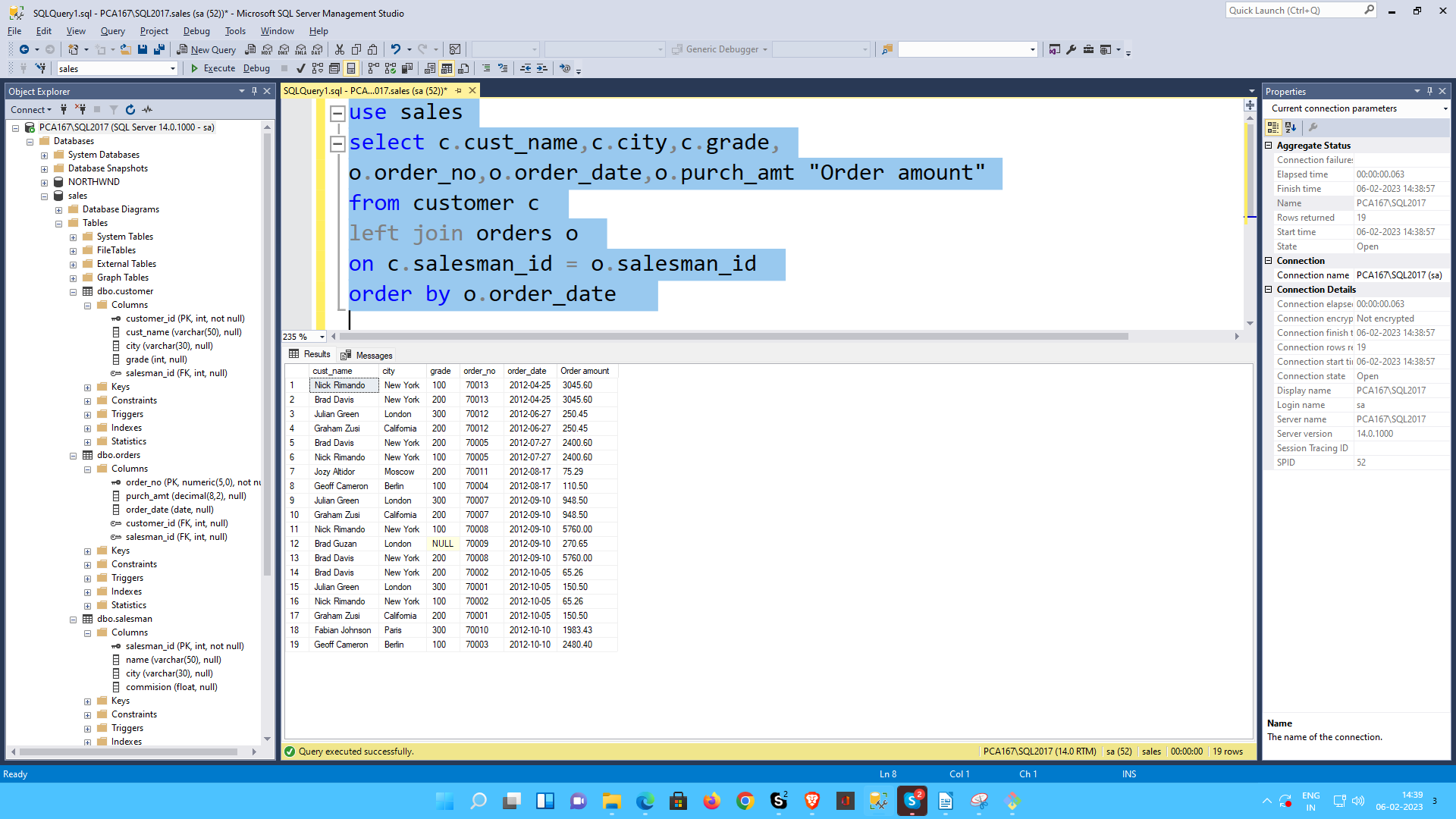
**o.order\_no,o.order\_date,o.purch\_amt "Order amount"**

**from customer c**

**left join orders o**

**on c.salesman\_id = o.salesman\_id**

**order by o.order\_date**

****

**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,o.order\_no,o.purch\_amt,**

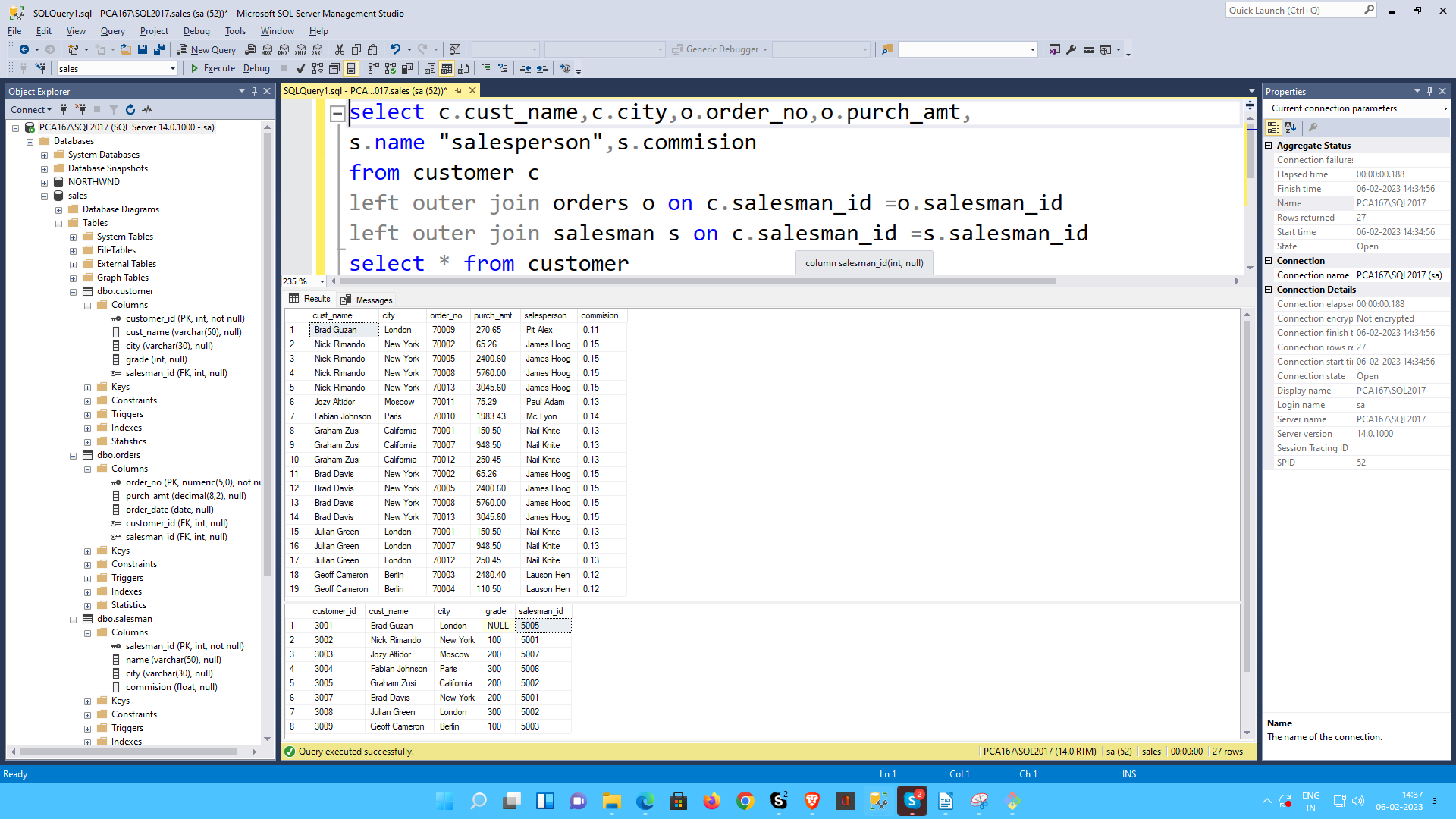
**s.name "salesperson",s.commision**

**from customer c**

**left outer join orders o on c.salesman\_id =o.salesman\_id**

**left outer join salesman s on c.salesman\_id =s.salesman\_id**

**select \* from customer**

****

**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**

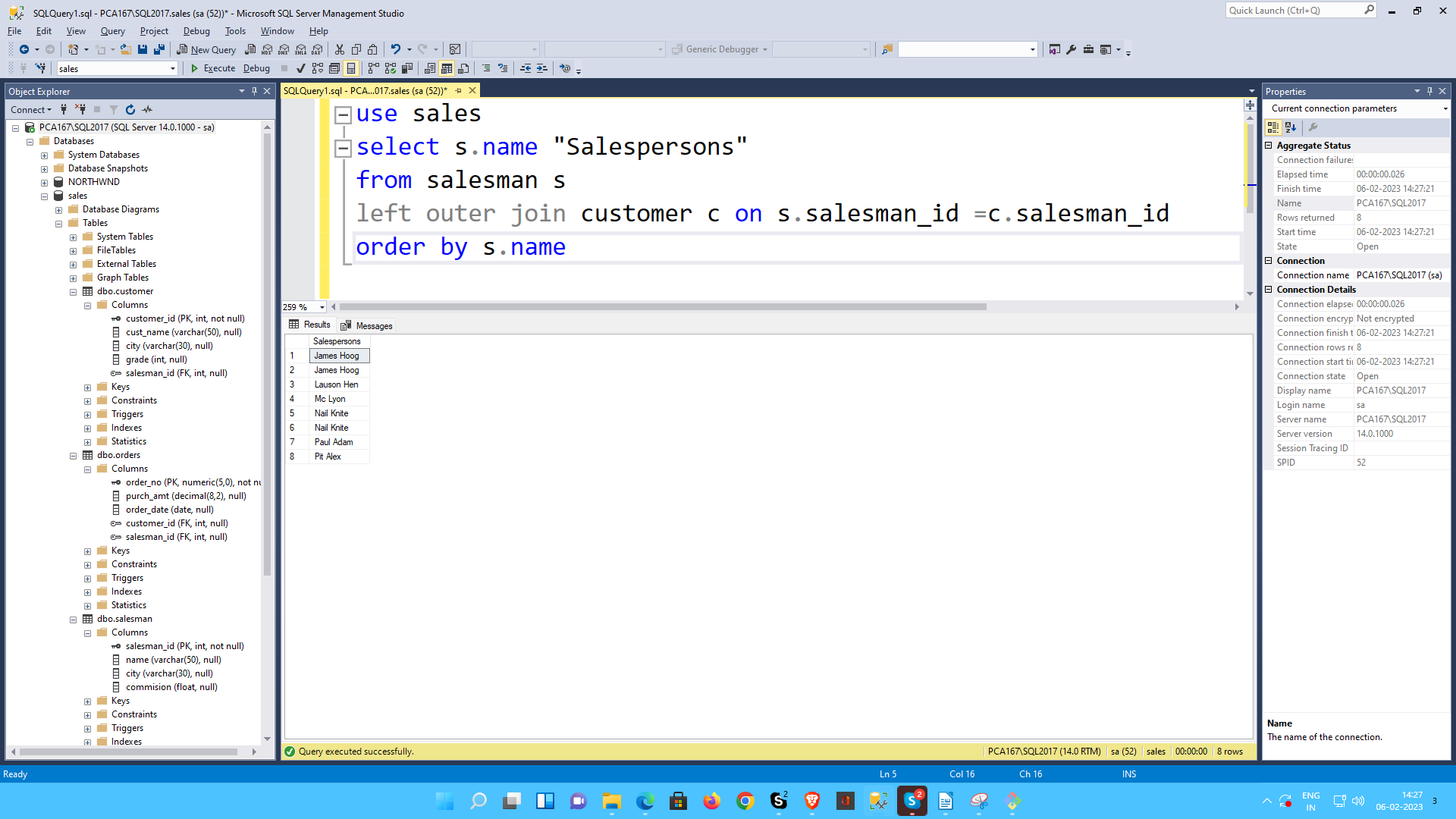
**use sales**

**select s.name "Salespersons"**

**from salesman s**

**left outer join customer c on s.salesman\_id =c.salesman\_id**

**order by s.name**

****

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

**use sales**

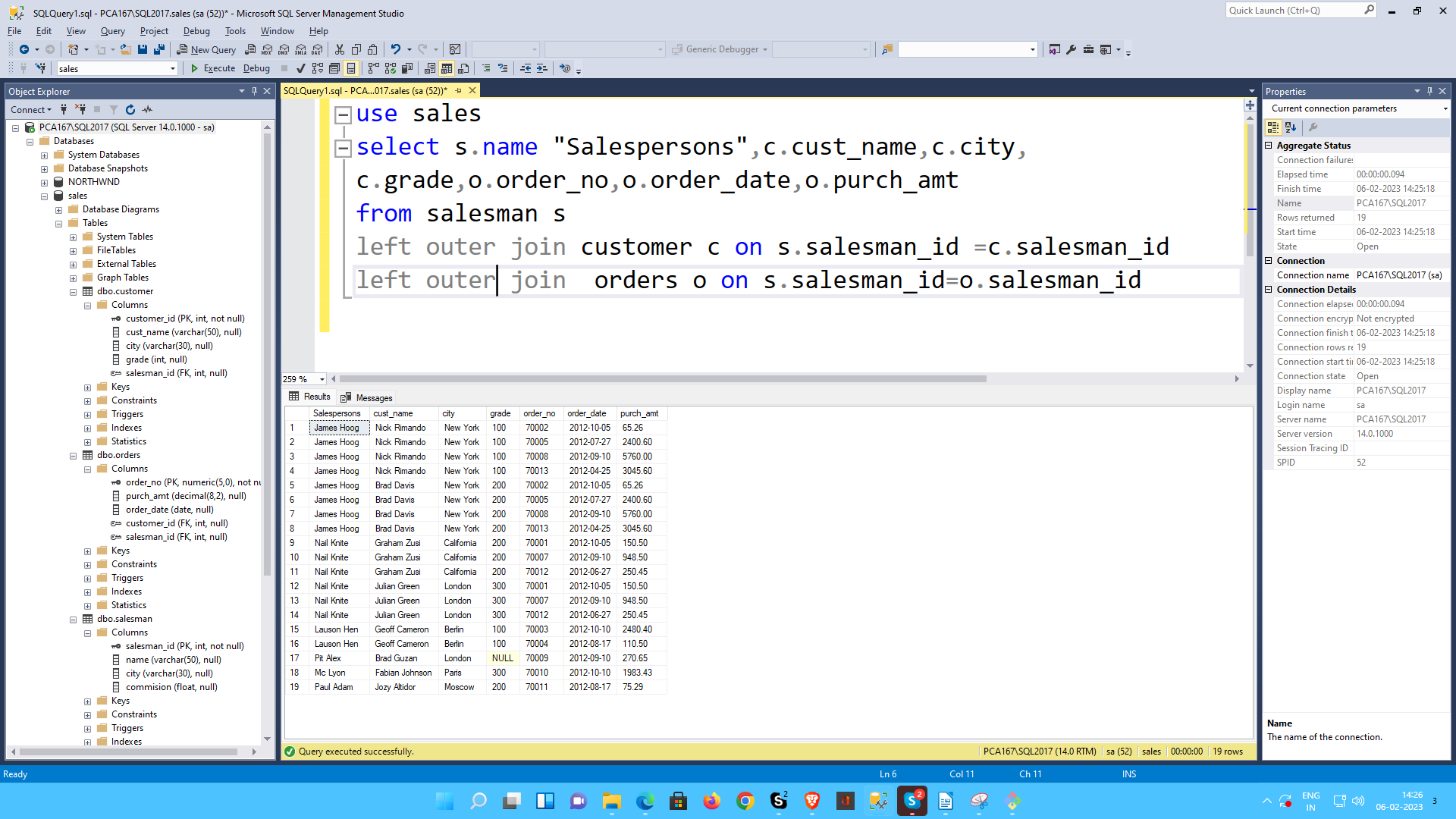
**select s.name "Salespersons",c.cust\_name,c.city,**

**c.grade,o.order\_no,o.order\_date,o.purch\_amt**

**from salesman s**

**left outer join customer c on s.salesman\_id =c.salesman\_id**

**left outer join orders o on s.salesman\_id=o.salesman\_id**

****

**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.**

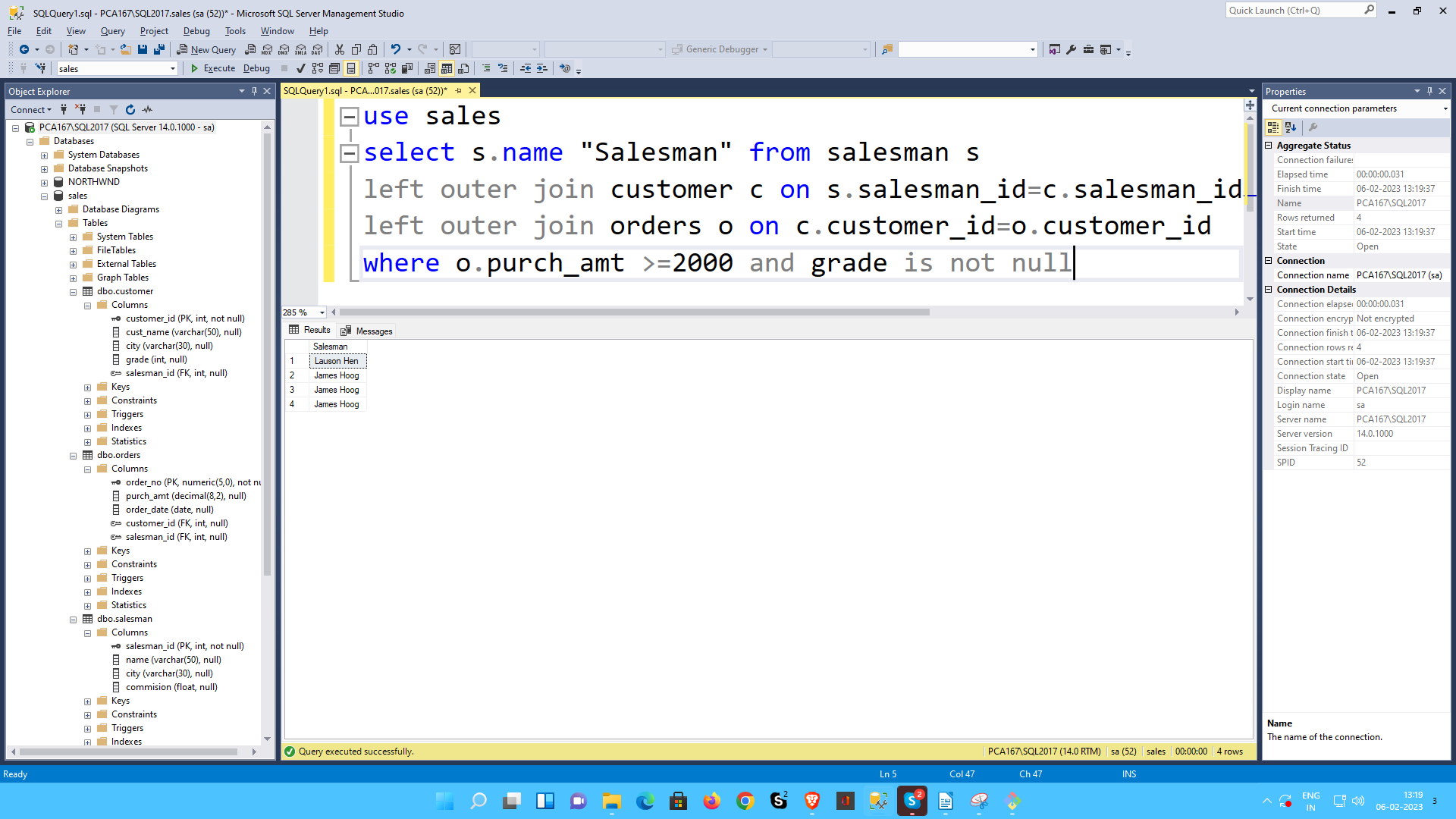
**use sales**

**select s.name "Salesman" from salesman s**

**left outer join customer c on s.salesman\_id=c.salesman\_id**

**left outer join orders o on c.customer\_id=o.customer\_id**

**where o.purch\_amt >=2000 and grade is not 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.**

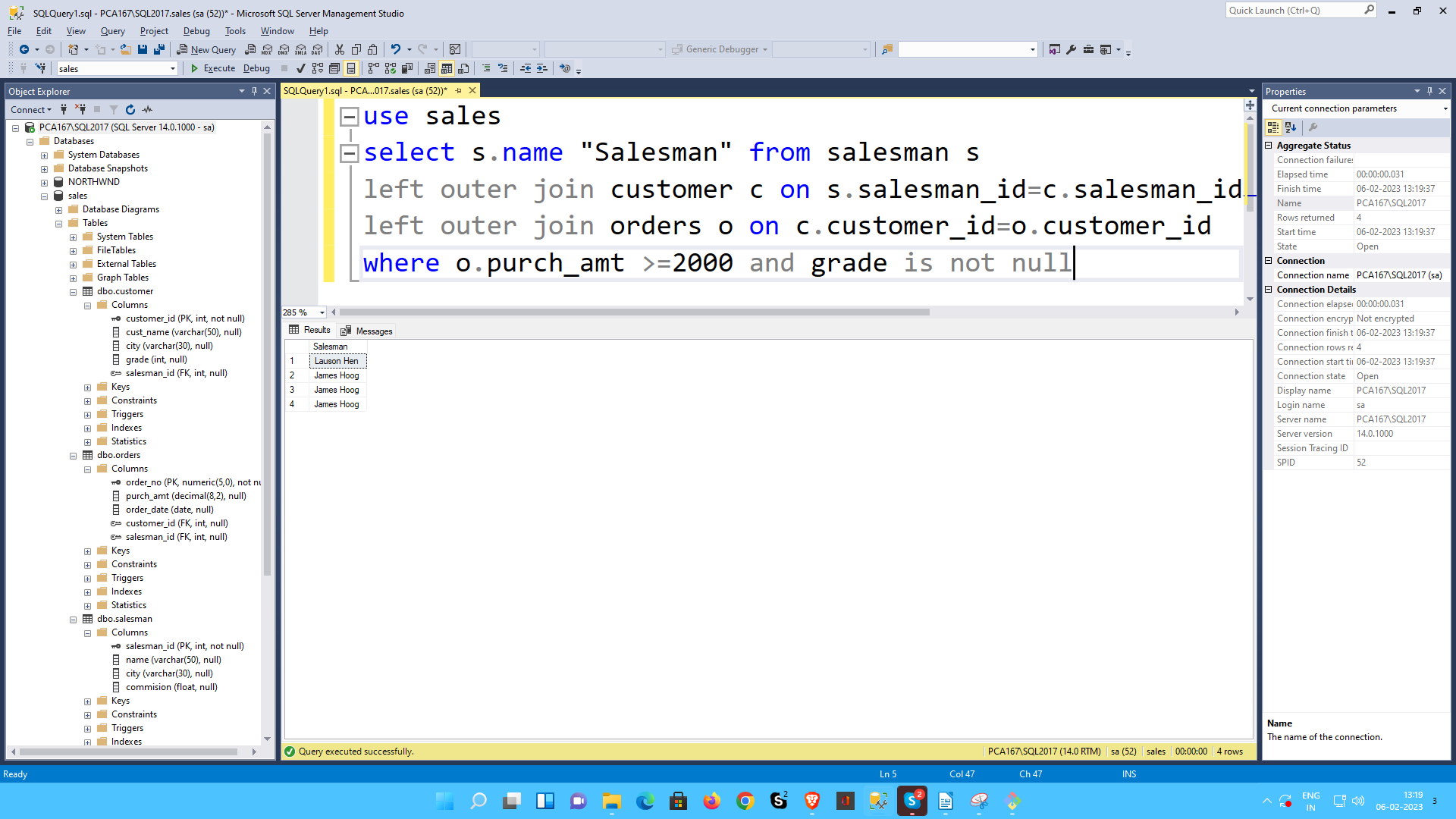
**use sales**

**select s.name "Salesman" from salesman s**

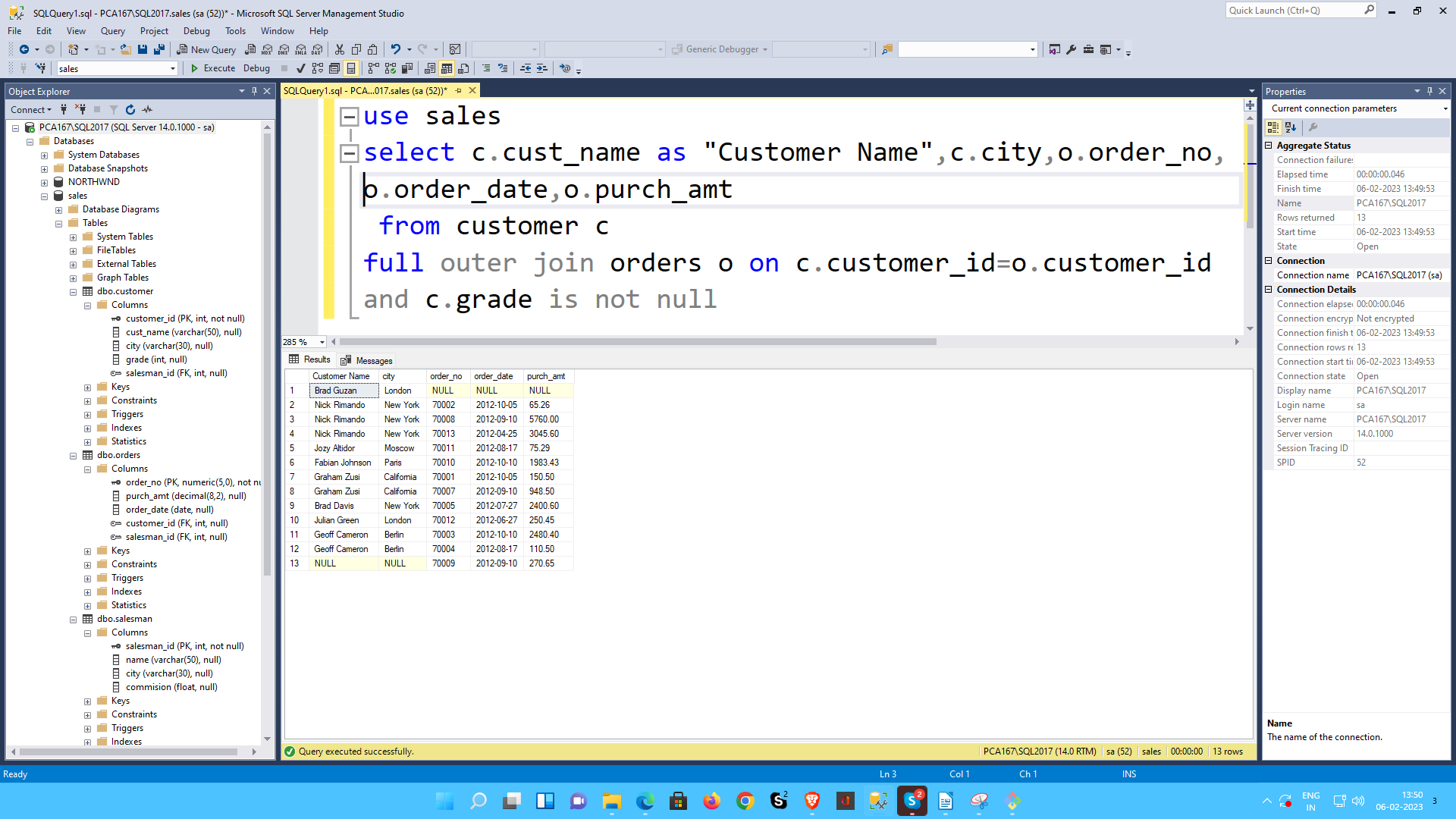
**left outer join customer c on s.salesman\_id=c.salesman\_id**

**left outer join orders o on c.customer\_id=o.customer\_id**

**where o.purch\_amt >=2000 and grade is not null**

****

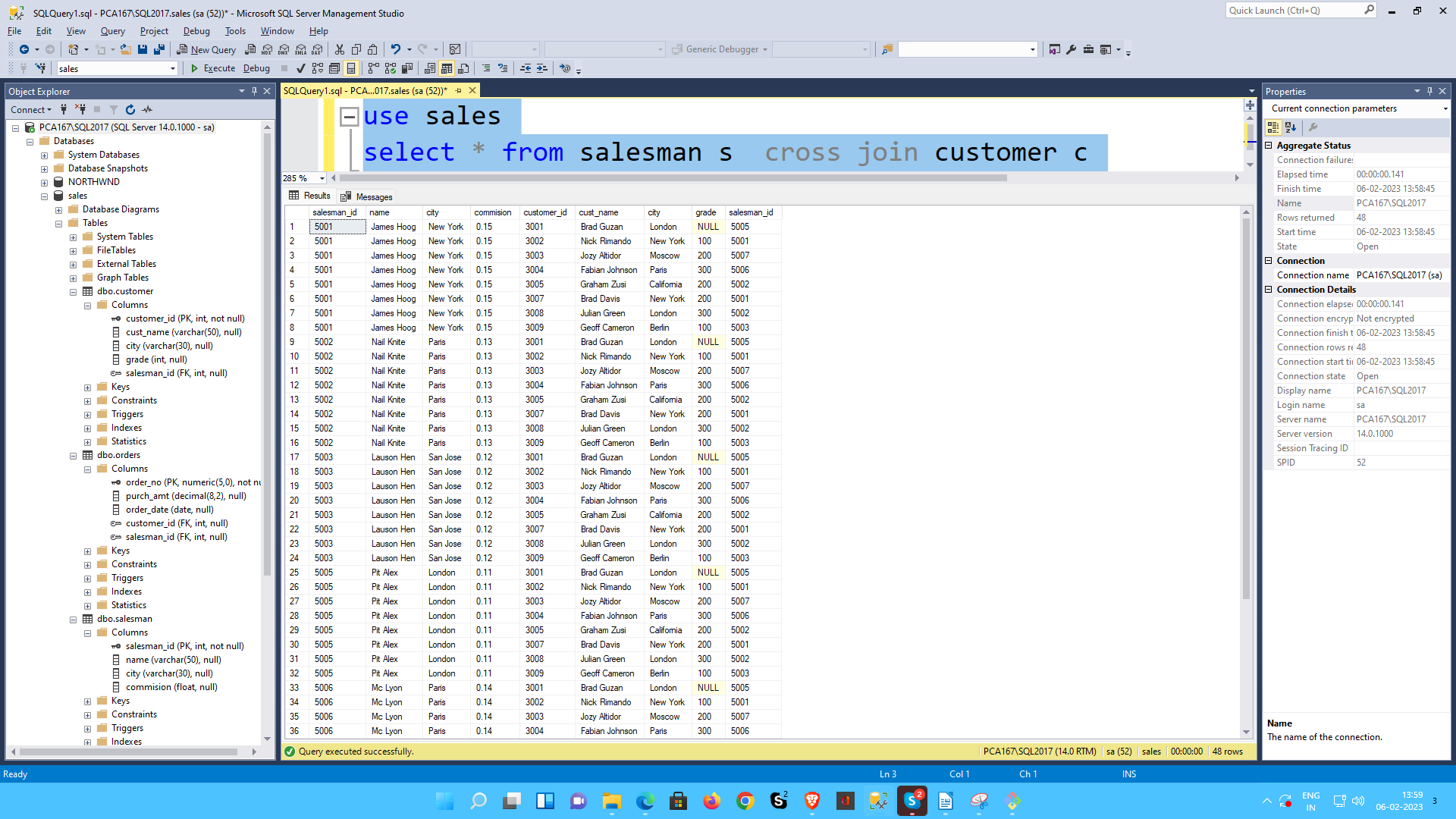
**16. Write a SQL statement to generate a report with the customer name, city, order no. order date, purchase amount for only those customers on the list who must have a grade and placed one or more orders or which order(s) have been placed by the customer who neither is on the list nor has a grade.**

****

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

**use sales**

**select \* from salesman s cross join customer c**

****

**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.**

**use sales**

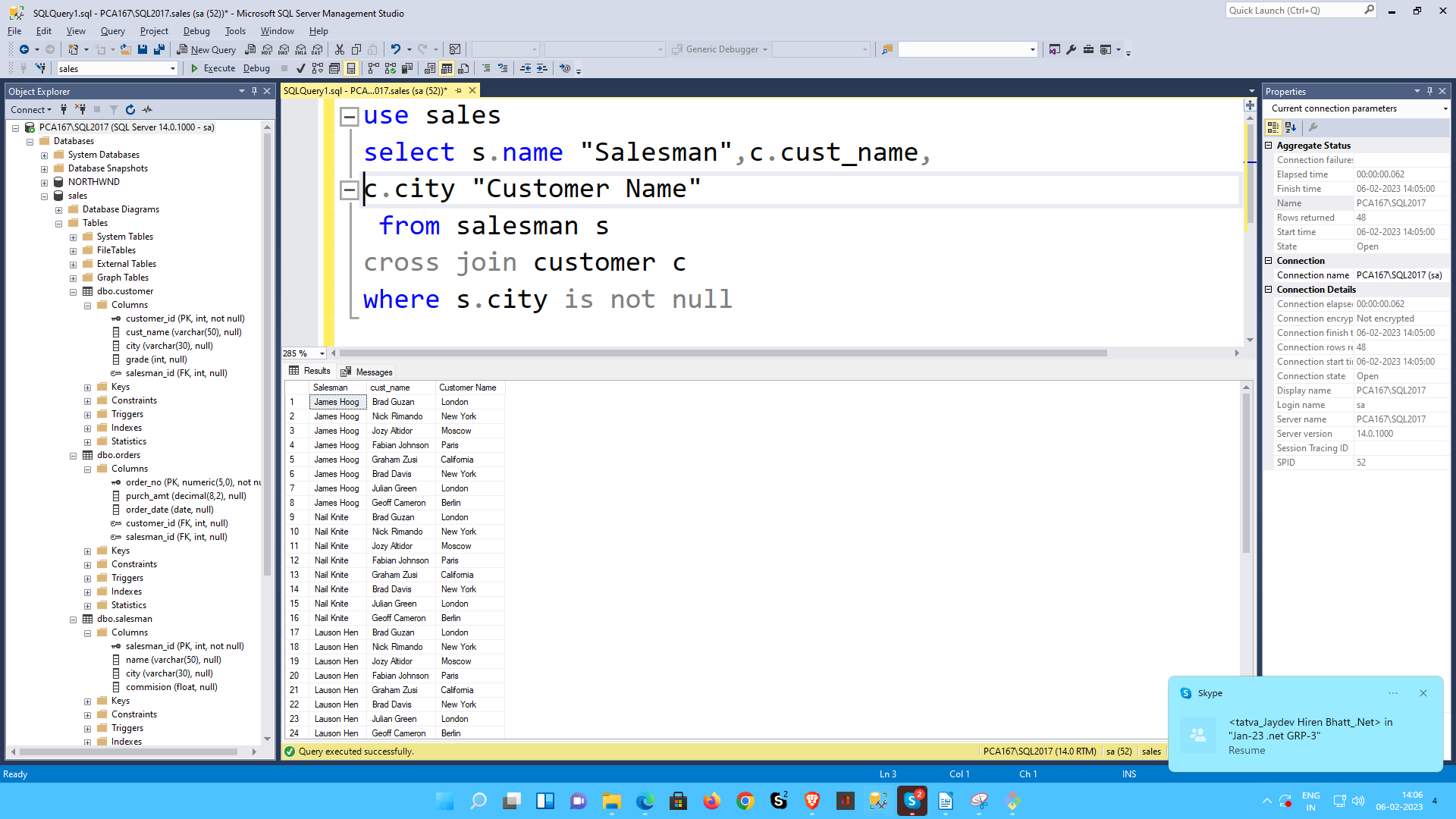
**select s.name "Salesman",c.cust\_name,**

**c.city "Customer Name"**

**from salesman s**

**cross join customer c**

**where s.city is not null**

****

**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.**

**use sales**

**select s.name "Salesman",c.cust\_name,**

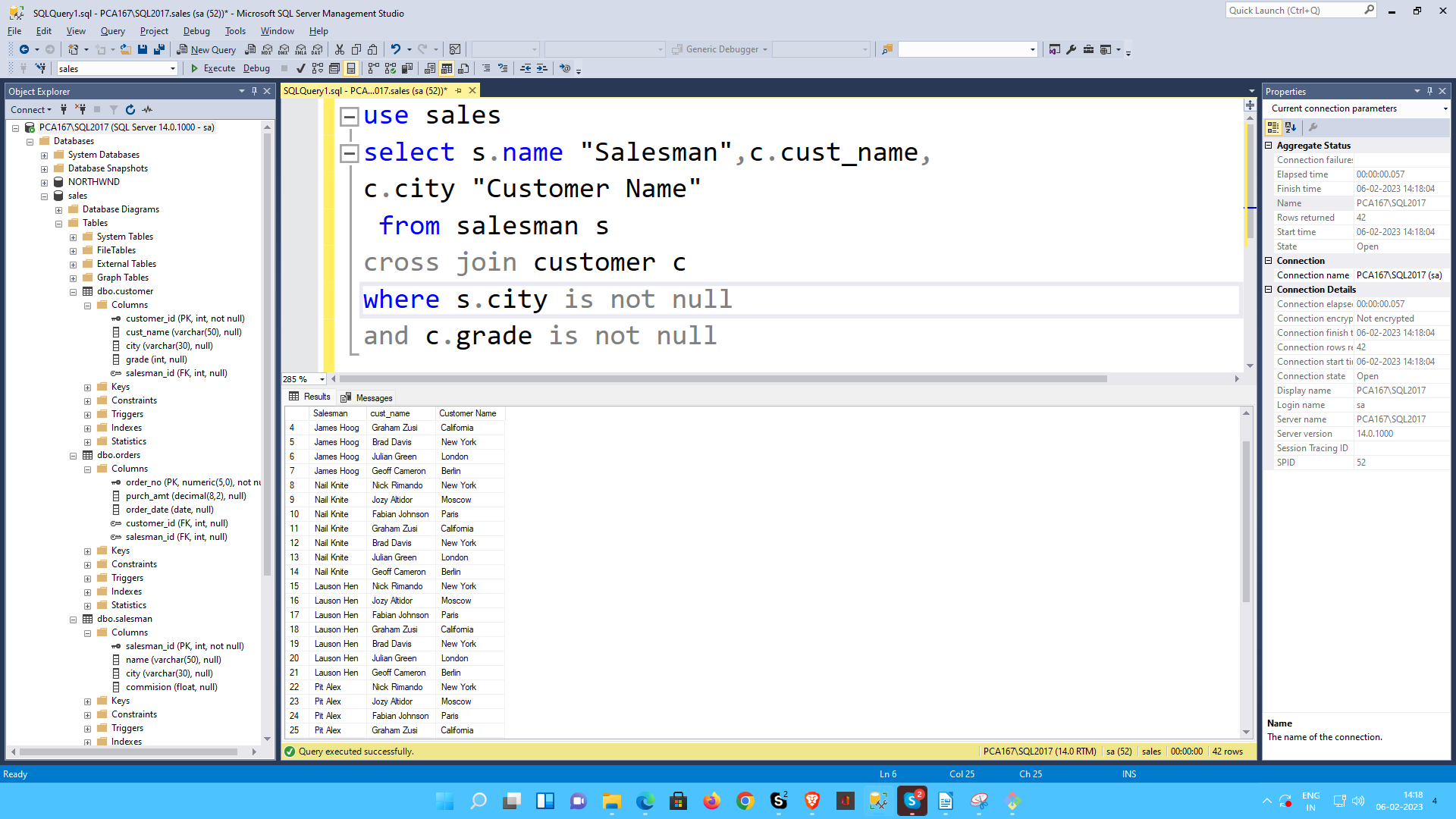
**c.city "Customer Name"**

**from salesman s**

**cross join customer c**

**where s.city is not null**

**and c.grade is not null**

****

**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**

**use sales**

**select s.name "Salesman",c.cust\_name,**

**c.city "Customer Name"**

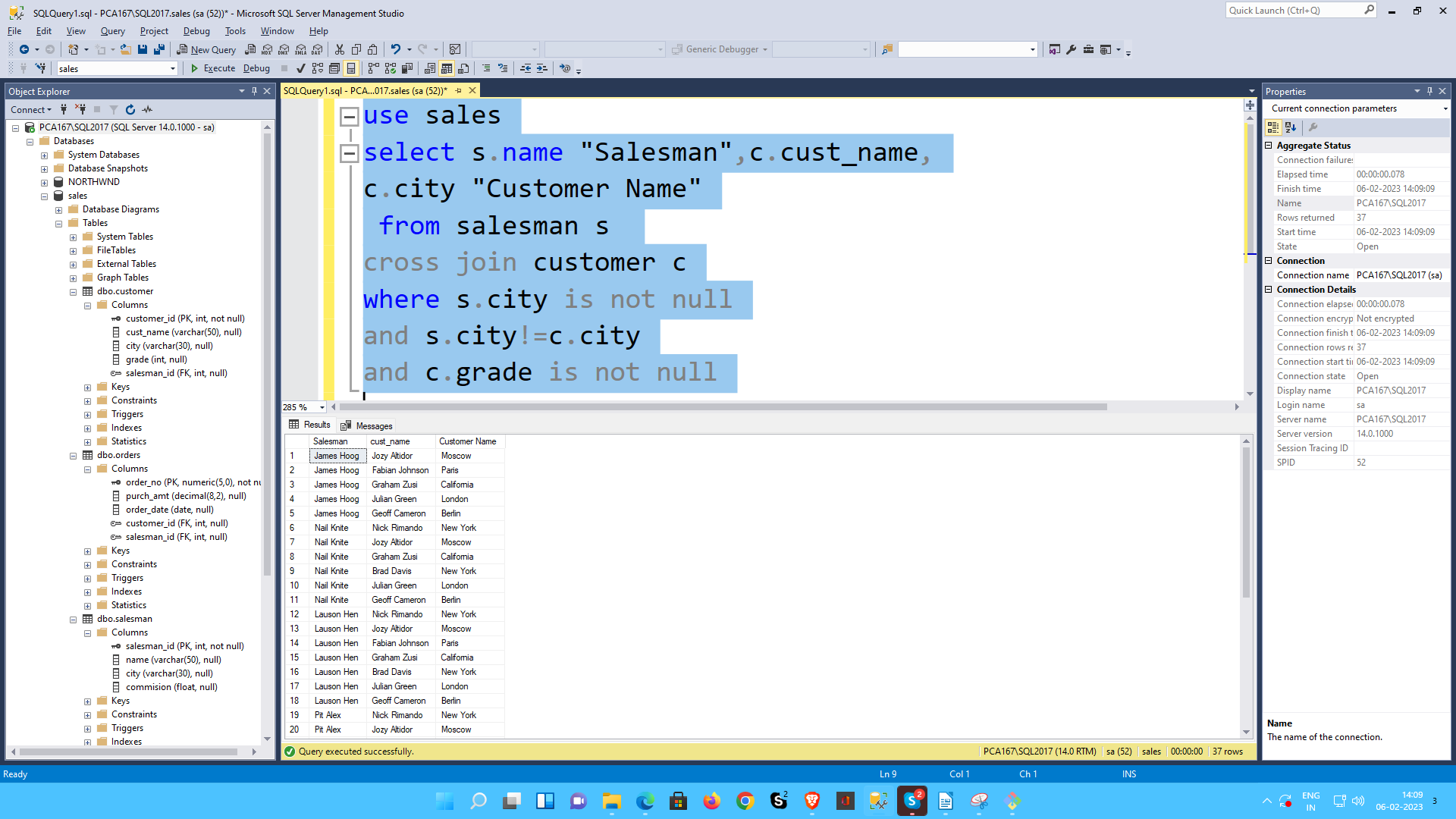
**from salesman s**

**cross join customer c**

**where s.city is not null**

**and s.city!=c.city**

**and c.grade is not null**

****