**SQL Assignment-2**

**Retrieve data using join with where clause**

CREATE TABLE Salesman

(

salesman\_id INT IDENTITY (5001,1) PRIMARY KEY,

name VARCHAR(20),

city VARCHAR(20),

commission decimal(2,2)

);

CREATE TABLE Customer

(

customer\_id INT IDENTITY (3001,1) PRIMARY KEY,

cust\_name VARCHAR(20),

city VARCHAR(20),

grade INT,

salesman\_id INT

);

CREATE TABLE Orders

(

ord\_no INT IDENTITY (70001,1) PRIMARY KEY,

purch\_amt DECIMAL(5,2),

ord\_date DATE,

customer\_id INT,

salesman\_id INT

);

INSERT INTO Salesman VALUES('John' , 'Baroda','0.15');

INSERT INTO Salesman VALUES('Ben Stocks' , 'Surat','0.13');

INSERT INTO Salesman VALUES('Kohli' , 'Mumbai','0.12');

INSERT INTO Salesman VALUES('Pit Alex' , 'London','0.11');

INSERT INTO Salesman VALUES('Malan' , 'Bhavnagar','0.14');

INSERT INTO Salesman VALUES('Paul Adom' ,'Rome','0.13');

INSERT INTO Customer VALUES('Yash','Damnagar','100','5001');

INSERT INTO Customer VALUES('Harshil','Surat','200','5001');

INSERT INTO Customer VALUES('Shrey','Bhavnagar','300','5002');

INSERT INTO Customer VALUES('Sagar','Baroda','100','5003');

INSERT INTO Customer VALUES('Bhavesh','Mumbai','300','5004');

INSERT INTO Customer VALUES('Gaurav','Patan','200','5003');

INSERT INTO Customer VALUES('Vivek','Gandhinagar','100','5005');

INSERT INTO Customer VALUES('Chirag','Rajkot','200','5006');

INSERT INTO Orders VALUES('150.5','2022-02-06','3005','5002');

INSERT INTO Orders VALUES('270.65','2022-02-07','3001','5001');

INSERT INTO Orders VALUES('450.75','2022-08-06','3002','5005');

INSERT INTO Orders VALUES('134.15','2022-09-09','3001','5004');

INSERT INTO Orders VALUES('167.25','2022-09-06','3002','5003');

INSERT INTO Orders VALUES('356.35','2022-08-06','3005','5006');

INSERT INTO Orders VALUES('178.95','2022-10-06','3004','5002');

INSERT INTO Orders VALUES('350.55','2022-02-06','3006','5001');

INSERT INTO Orders VALUES('190.65','2022-07-07','3008','5004');

INSERT INTO Orders VALUES('107.75','2022-02-10','3005','5003');

1) write a SQL query to find the salesperson and customer who reside in the same city. Return Salesman, cust\_name and city

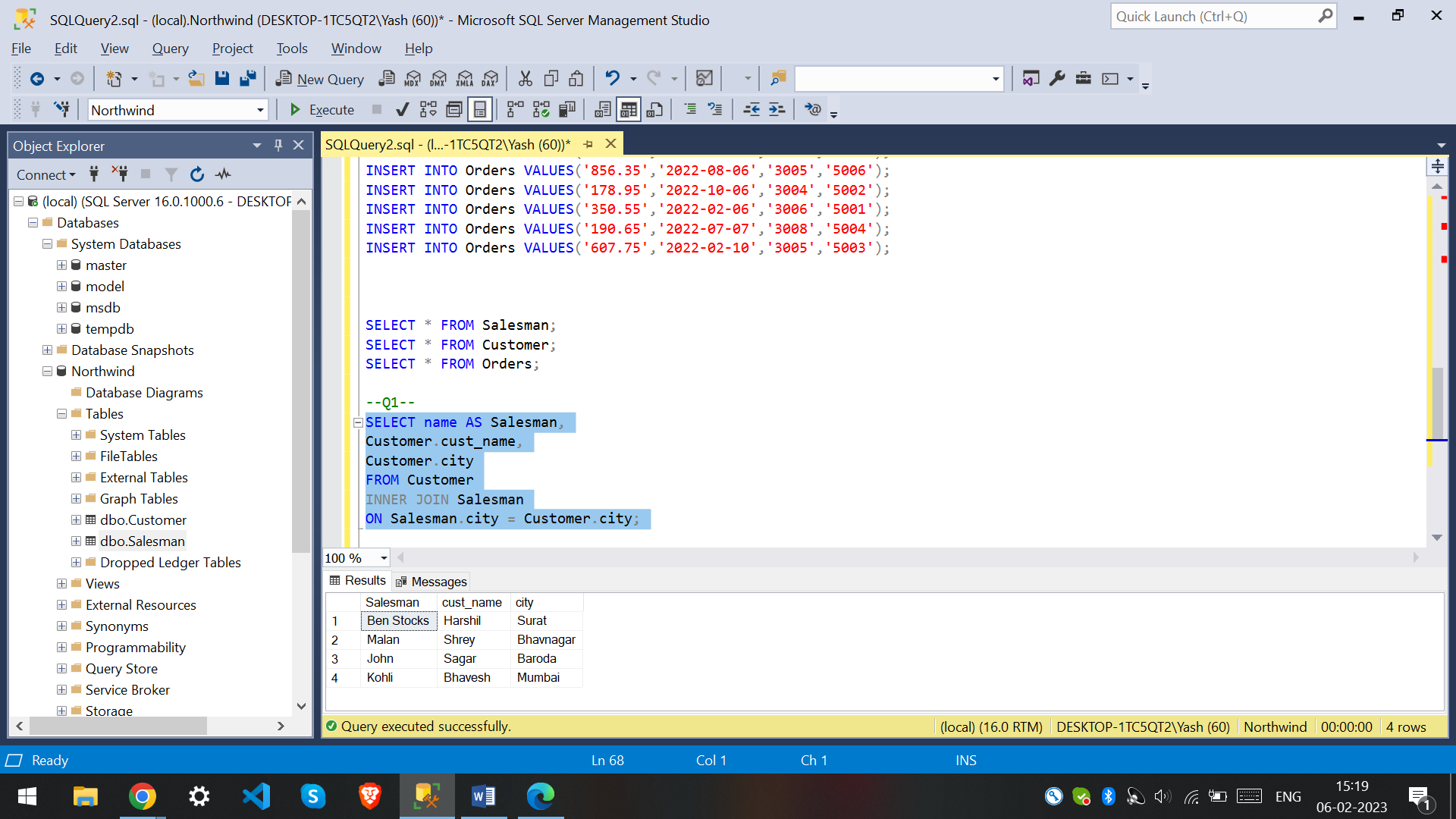
SELECT name AS Salesman,

Customer.cust\_name,

Customer.city

FROM Customer,Salesman

WHERE Salesman.city = Customer.city;



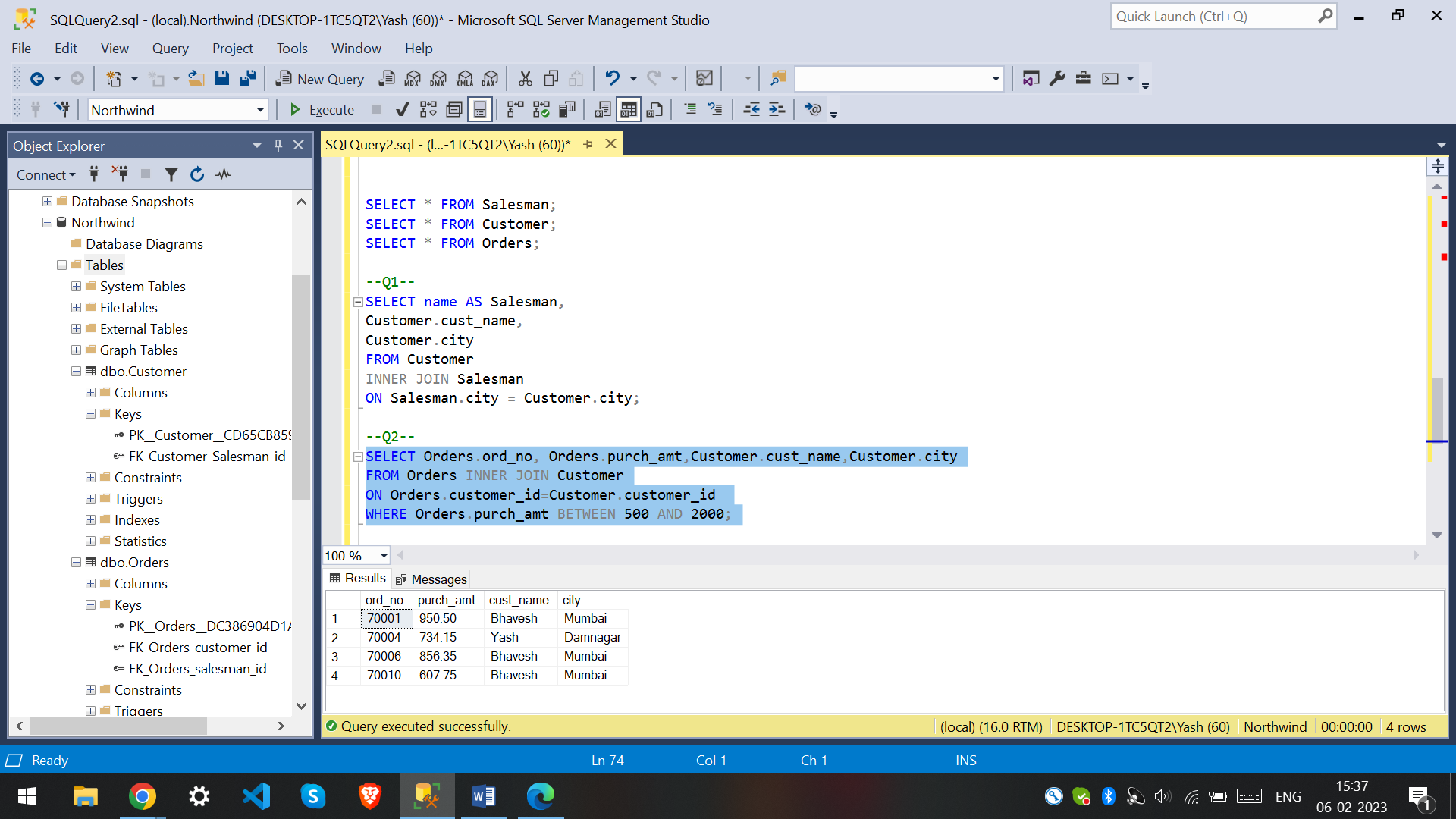
2) write a SQL query to find those orders where the order amount exists between 500 and 2000. Return ord\_no, purch\_amt, cust\_name, city.

SELECT Orders.ord\_no, Orders.purch\_amt,Customer.cust\_name,Customer.city

FROM Orders INNER JOIN Customer

ON Orders.customer\_id=Customer.customer\_id

WHERE Orders.purch\_amt BETWEEN 500 AND 2000;

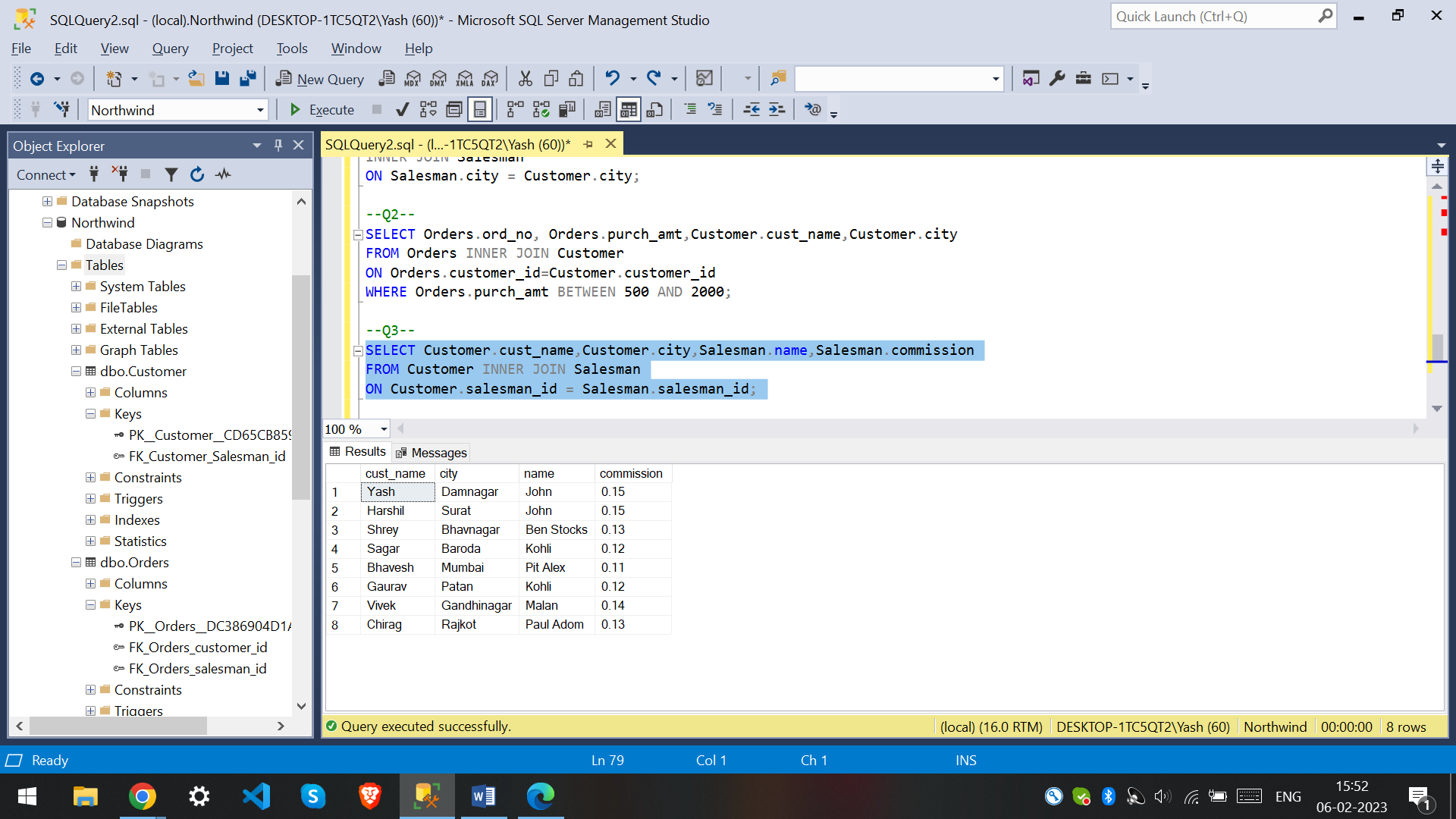


3. write a SQL query to find the salesperson(s) and the customer(s) he represents. Return Customer Name, city, Salesman, commission

SELECT Customer.cust\_name,Customer.city,Salesman.name,Salesman.commission

FROM Customer INNER JOIN Salesman

ON Customer.salesman\_id = Salesman.salesman\_id;



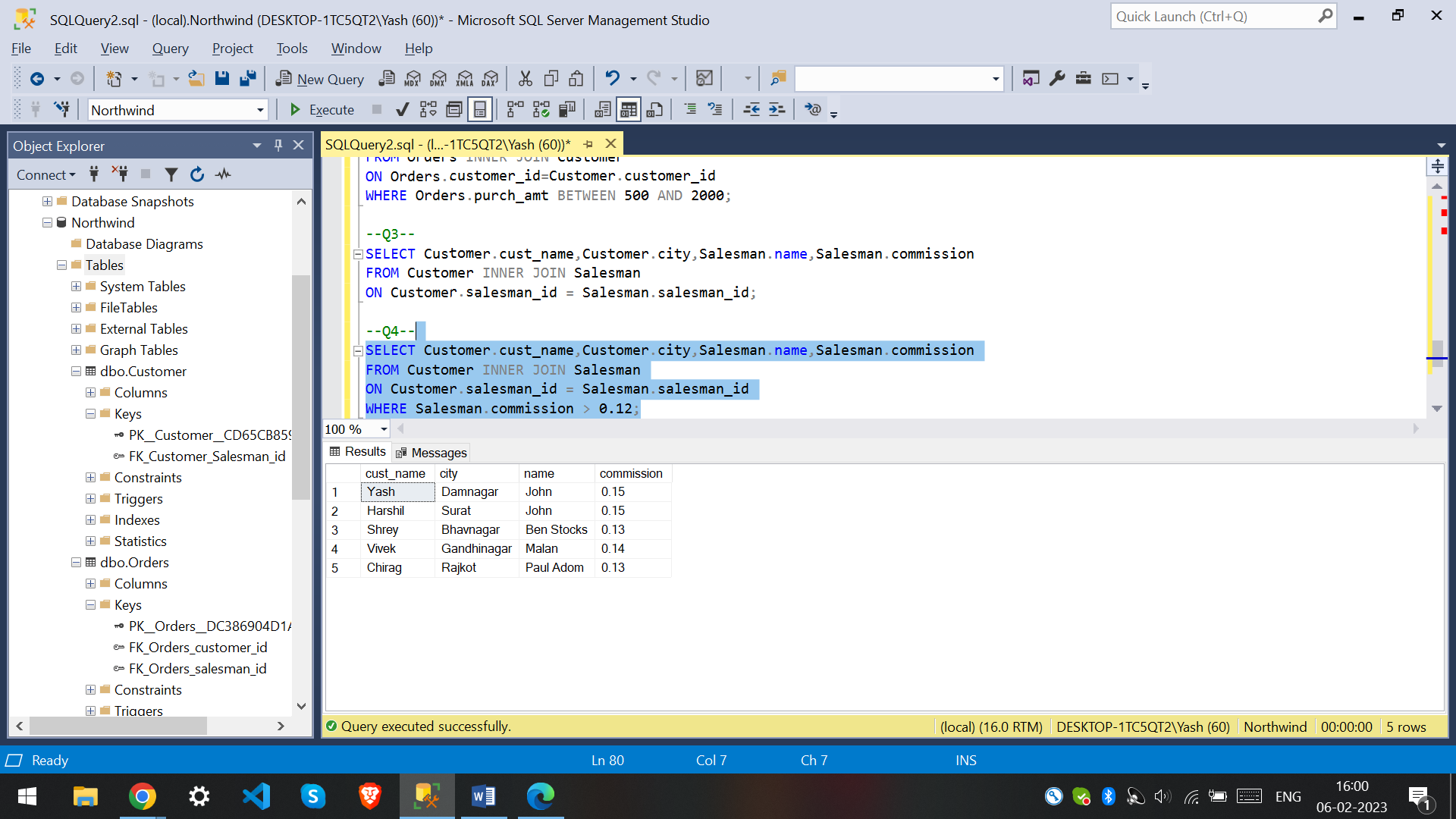
4. write a SQL query to find salespeople who received commissions of more than 12 percent from the company. Return Customer Name, customer city, Salesman, commission.

SELECT Customer.cust\_name,Customer.city,Salesman.name,Salesman.commission

FROM Customer INNER JOIN Salesman

ON Customer.salesman\_id = Salesman.salesman\_id

WHERE Salesman.commission > 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 Customer.cust\_name,Customer.city,Salesman.name,Salesman.city,Salesman.commission

FROM Customer INNER JOIN Salesman

ON Customer.salesman\_id = Salesman.salesman\_id

WHERE Salesman.commission > 0.12 AND Customer.city <> Salesman.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

SELECTOrders.ord\_no,Orders.ord\_date,Orders.purch\_amt,Customer.cust\_name,Customer.grade,Salesman.name,Salesman.commission

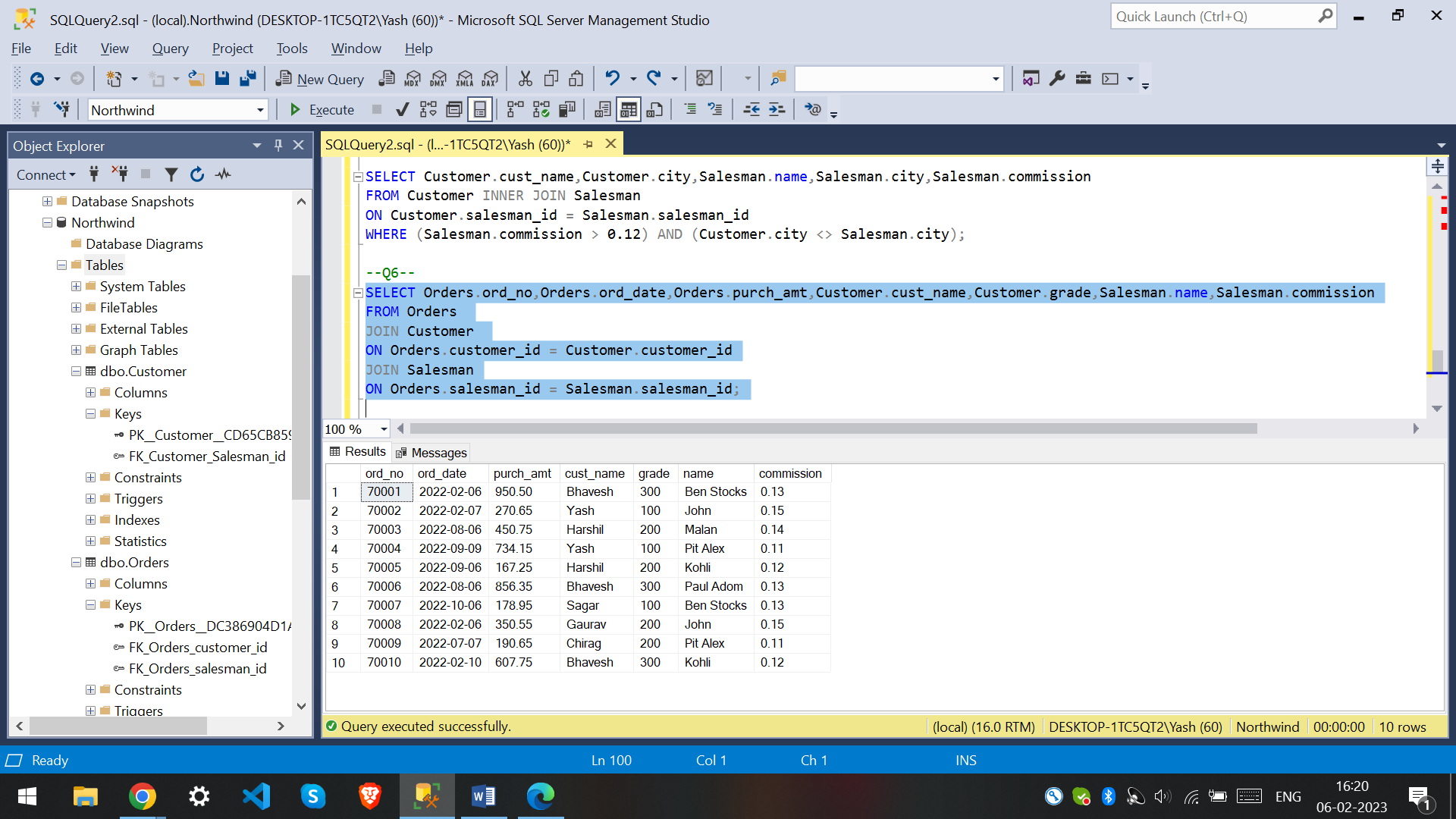
FROM Orders

JOIN Customer

ON Orders.customer\_id = Customer.customer\_id

JOIN Salesman

ON Orders.salesman\_id = Salesman.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.

--Q7--

SELECT S.salesman\_id , S.name AS salesman\_name,

C.customer\_id, C.cust\_name AS customer\_name, C.grade,

O.ord\_no, O.purch\_amt

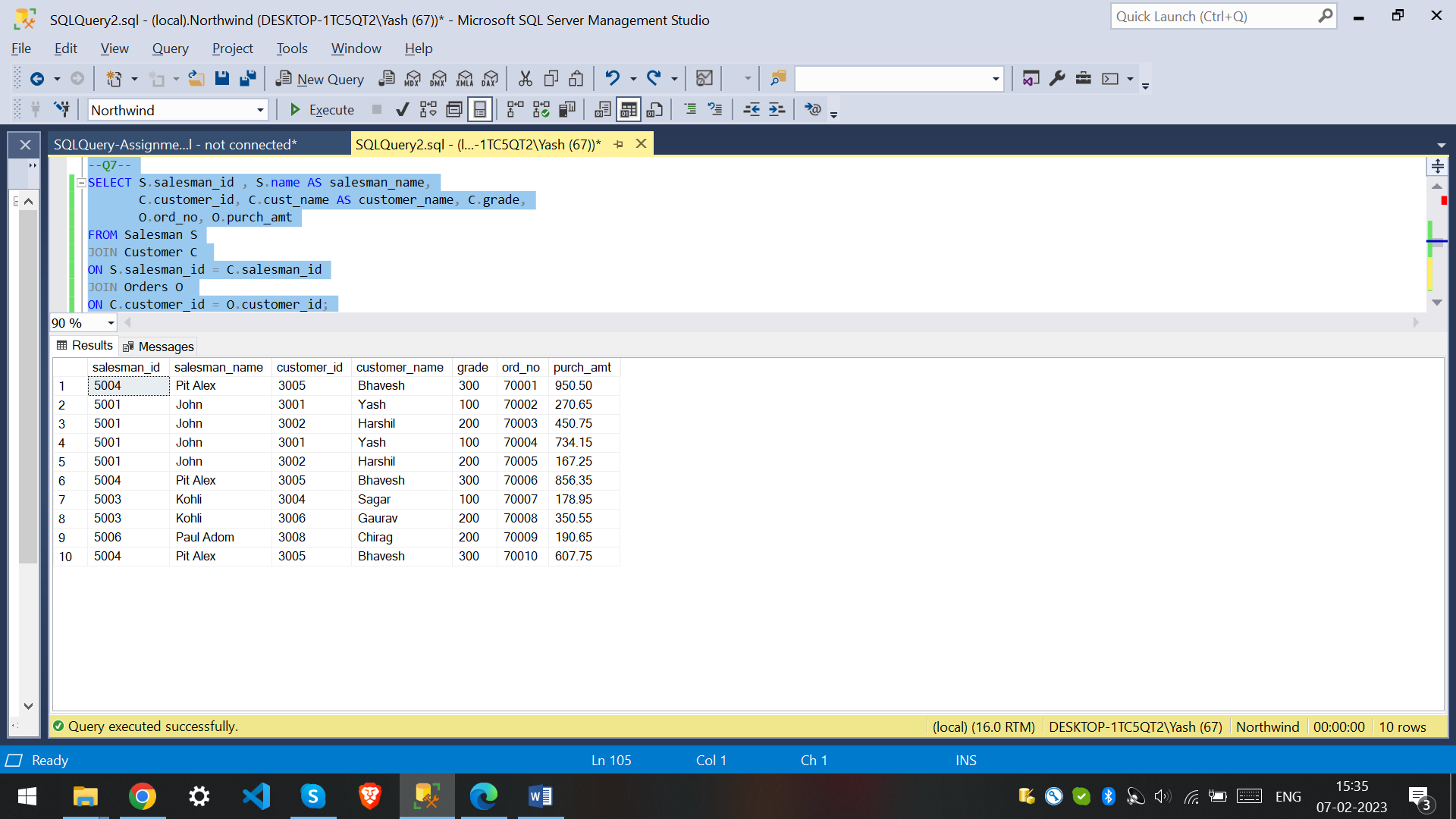
FROM Salesman S

JOIN Customer C

ON S.salesman\_id = C.salesman\_id

JOIN Orders O

ON C.customer\_id = O.customer\_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.

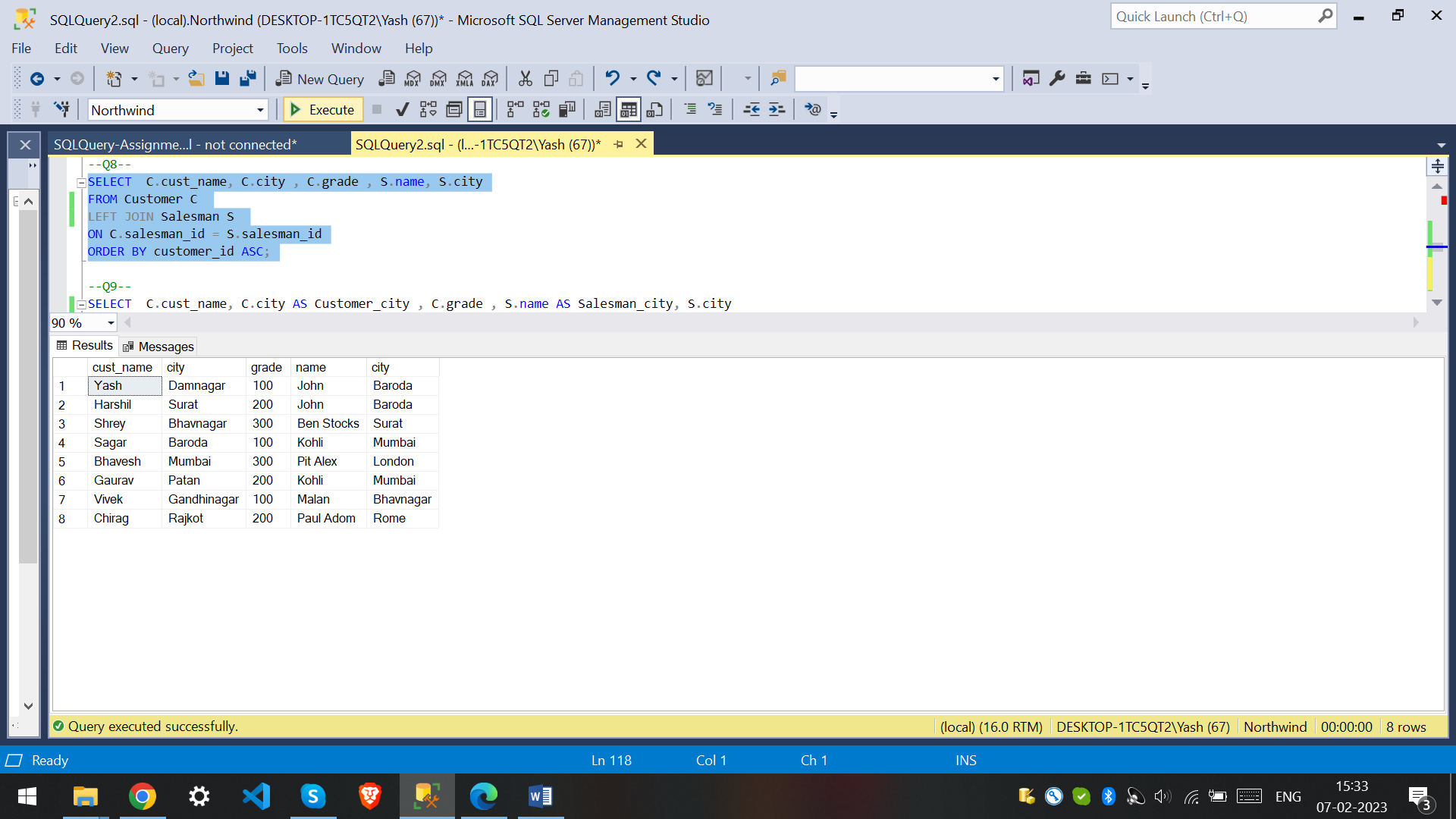
SELECT C.cust\_name, C.city , C.grade , S.name, S.city

FROM Customer C

LEFT JOIN Salesman S

ON C.salesman\_id = S.salesman\_id

ORDER BY customer\_id ASC;



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 AS Customer\_city , C.grade , S.name AS Salesman\_city, S.city

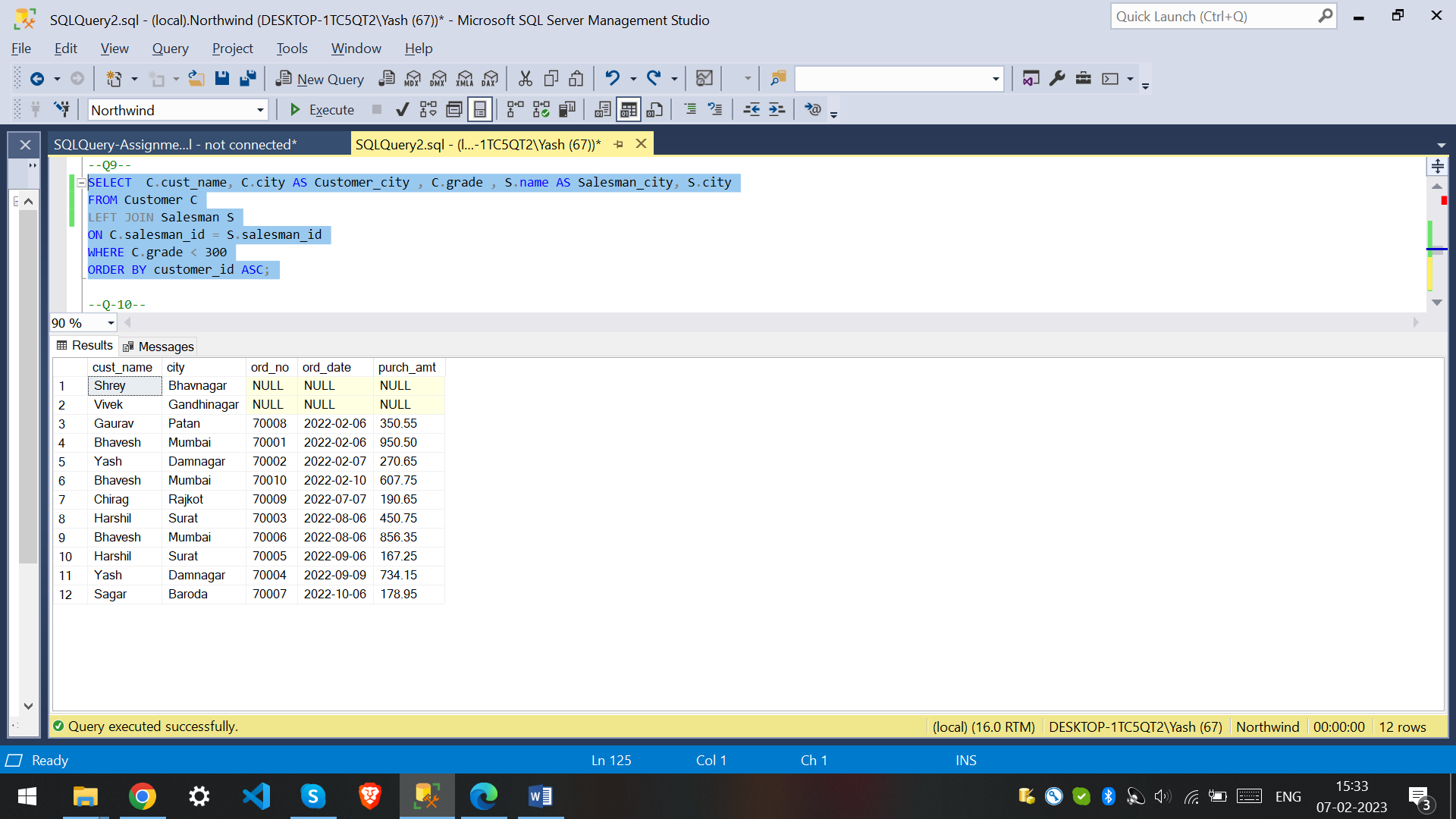
FROM Customer C

LEFT JOIN Salesman S

ON C.salesman\_id = S.salesman\_id

WHERE C.grade < 300

ORDER BY customer\_id ASC;



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

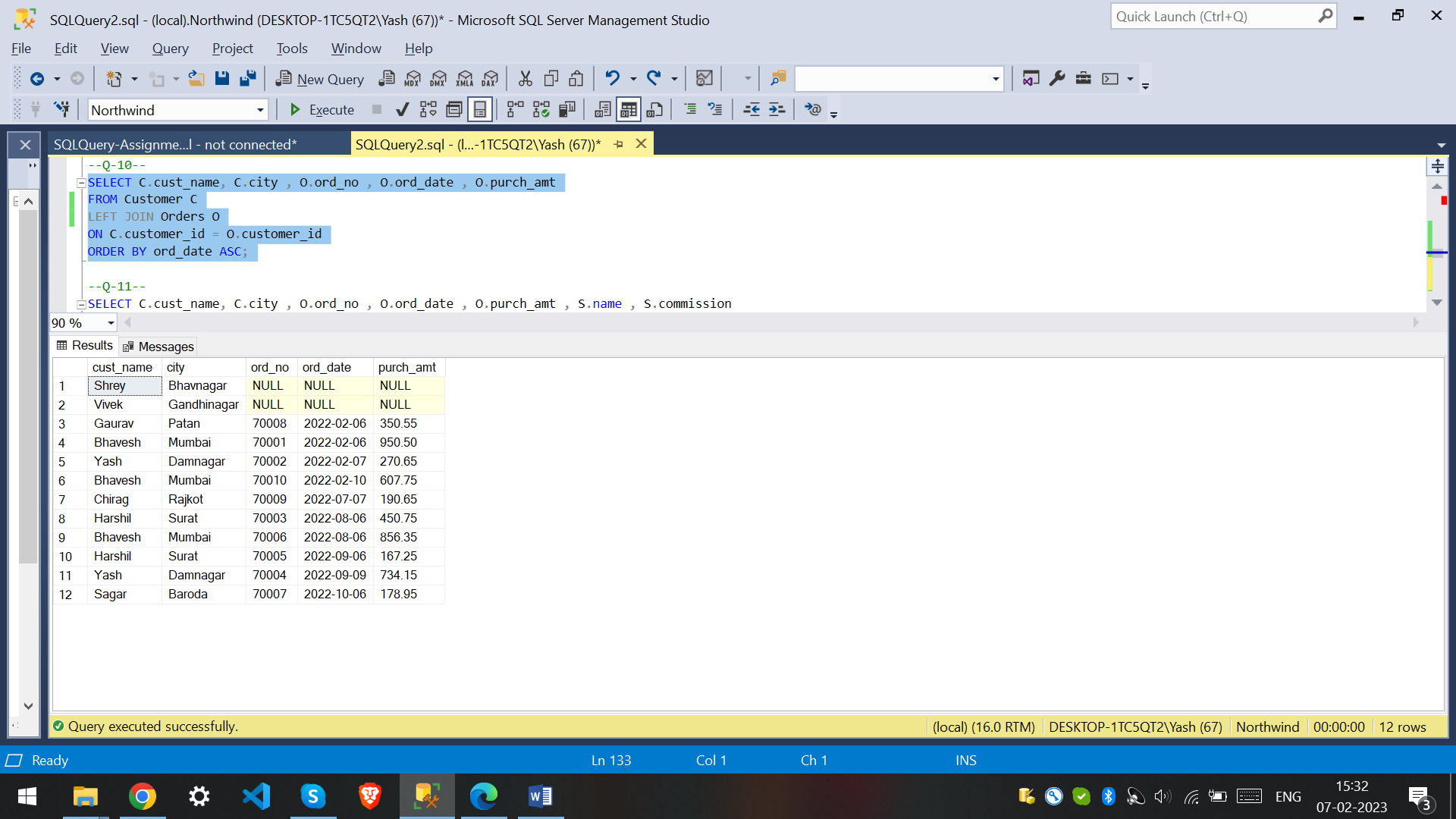
SELECT C.cust\_name, C.city , O.ord\_no , O.ord\_date , O.purch\_amt

FROM Customer C

LEFT JOIN Orders O

ON C.customer\_id = O.customer\_id

ORDER BY ord\_date ASC;



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.ord\_no , O.ord\_date , O.purch\_amt , S.name , S.commission

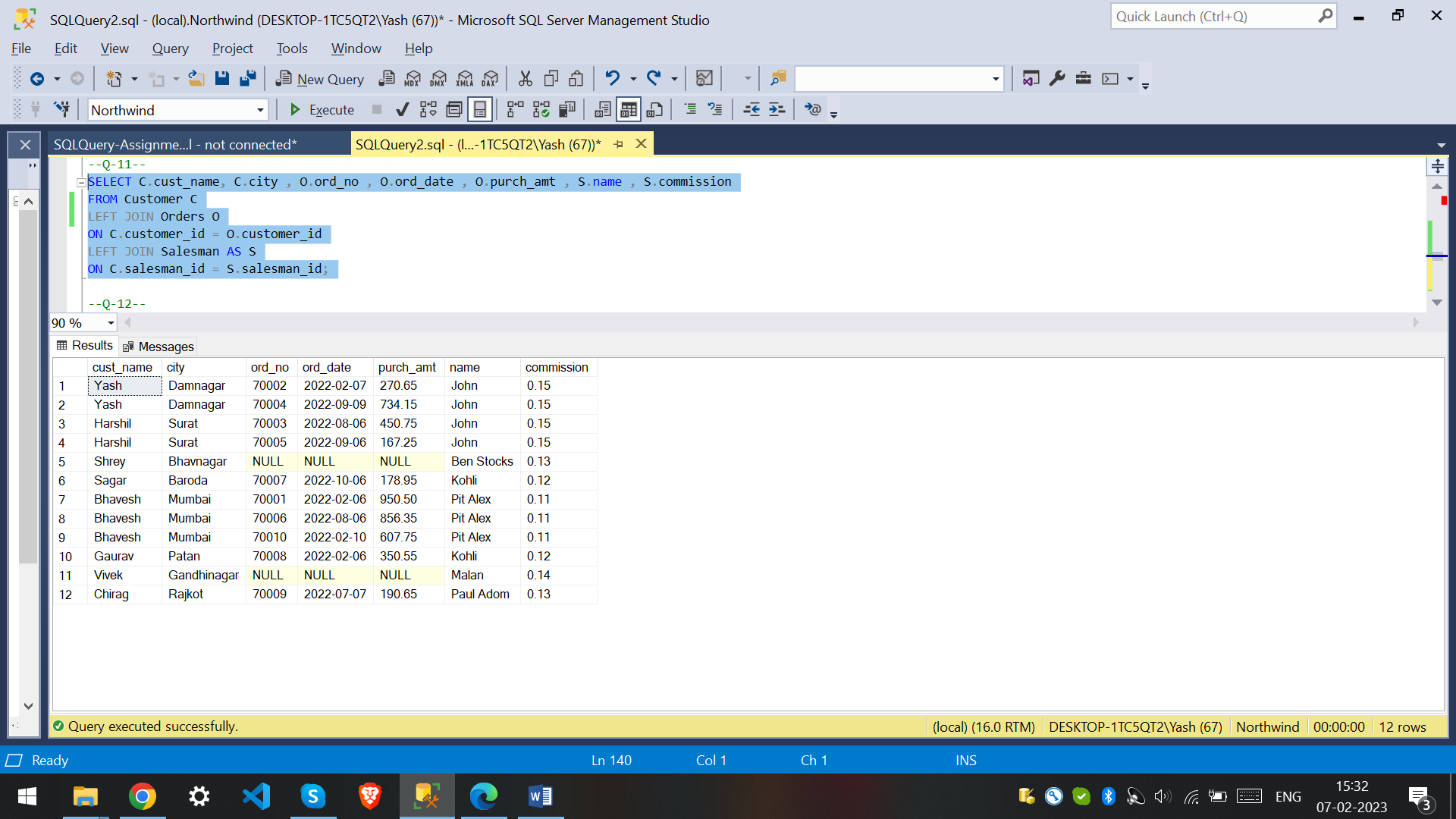
FROM Customer C

LEFT JOIN Orders O

ON C.customer\_id = O.customer\_id

LEFT JOIN Salesman AS S

ON C.salesman\_id = S.salesman\_id;



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,count(C.customer\_id) AS total\_customers

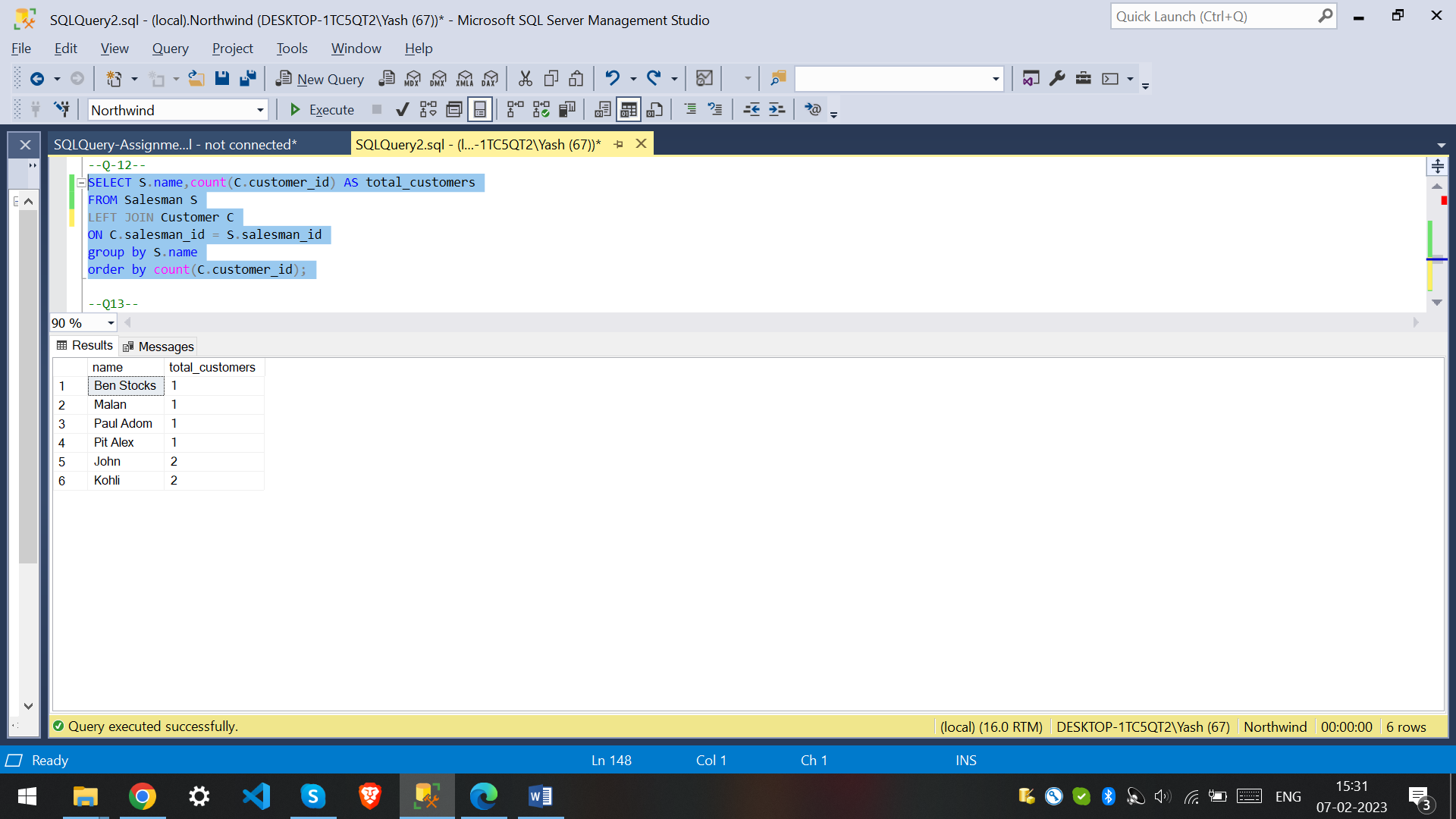
FROM Salesman S

LEFT JOIN Customer C

ON C.salesman\_id = S.salesman\_id

group by S.name

order by count(C.customer\_id);



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 , C.grade , O.ord\_no , O.ord\_date , O.purch\_amt

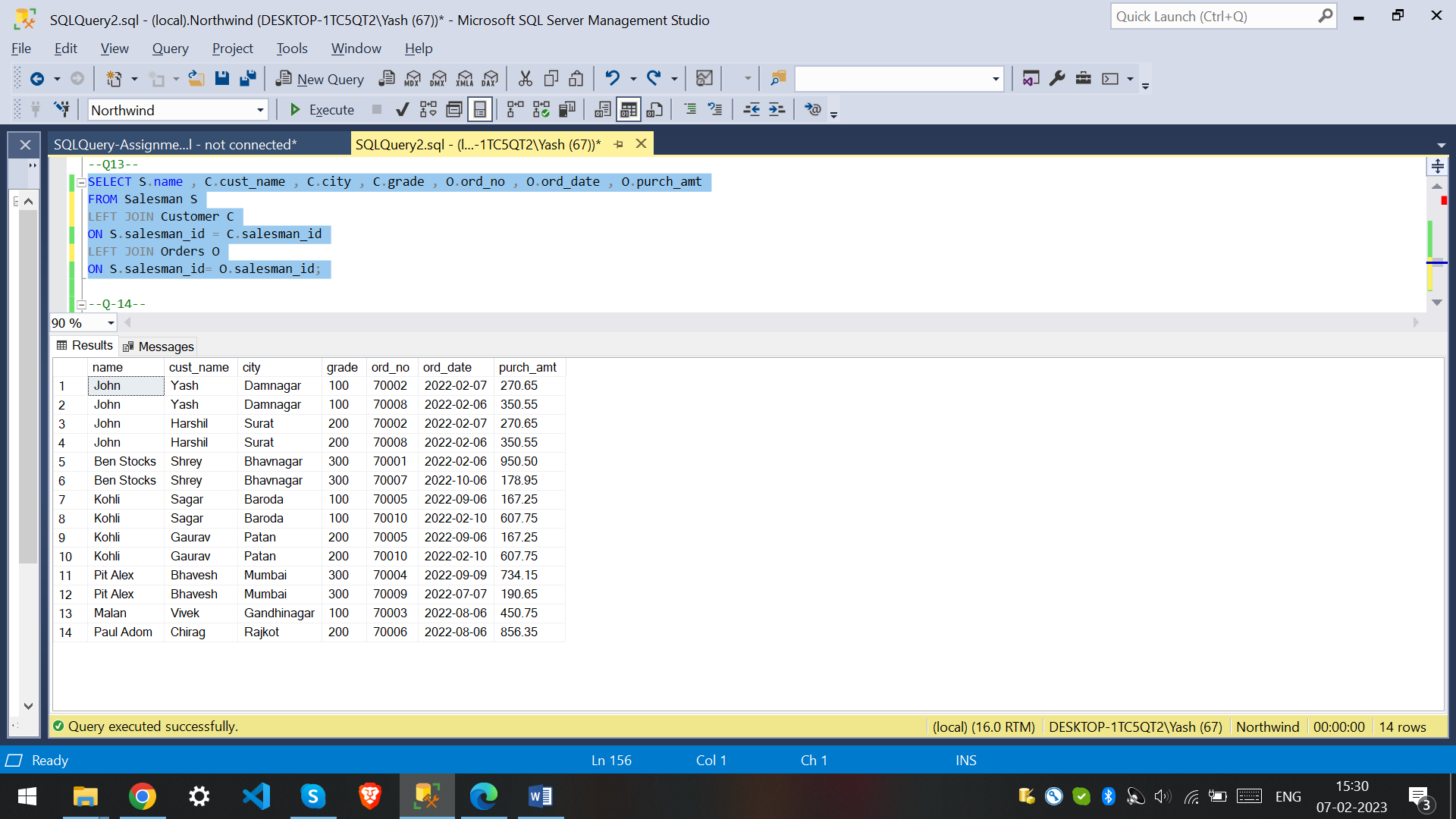
FROM Salesman S

LEFT JOIN Customer C

ON S.salesman\_id = C.salesman\_id

LEFT 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.

SELECT S.salesman\_id,S.name

FROM Salesman S

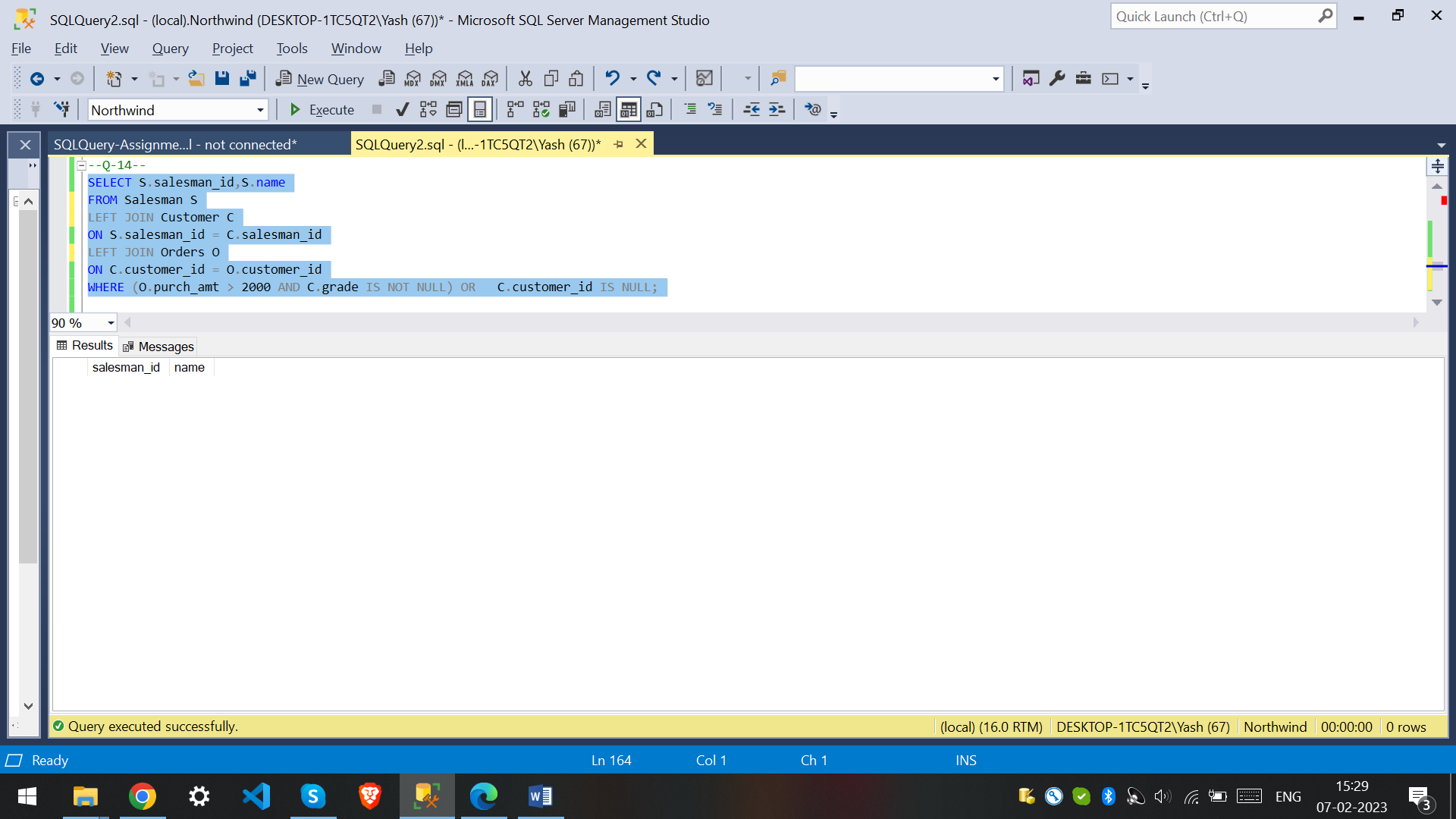
LEFT JOIN Customer C

ON S.salesman\_id = C.salesman\_id

LEFT JOIN Orders O

ON C.customer\_id = O.customer\_id

WHERE (O.purch\_amt > 2000 AND C.grade IS NOT NULL) OR C.customer\_id IS 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.

SELECT S.salesman\_id,S.name

FROM Salesman S

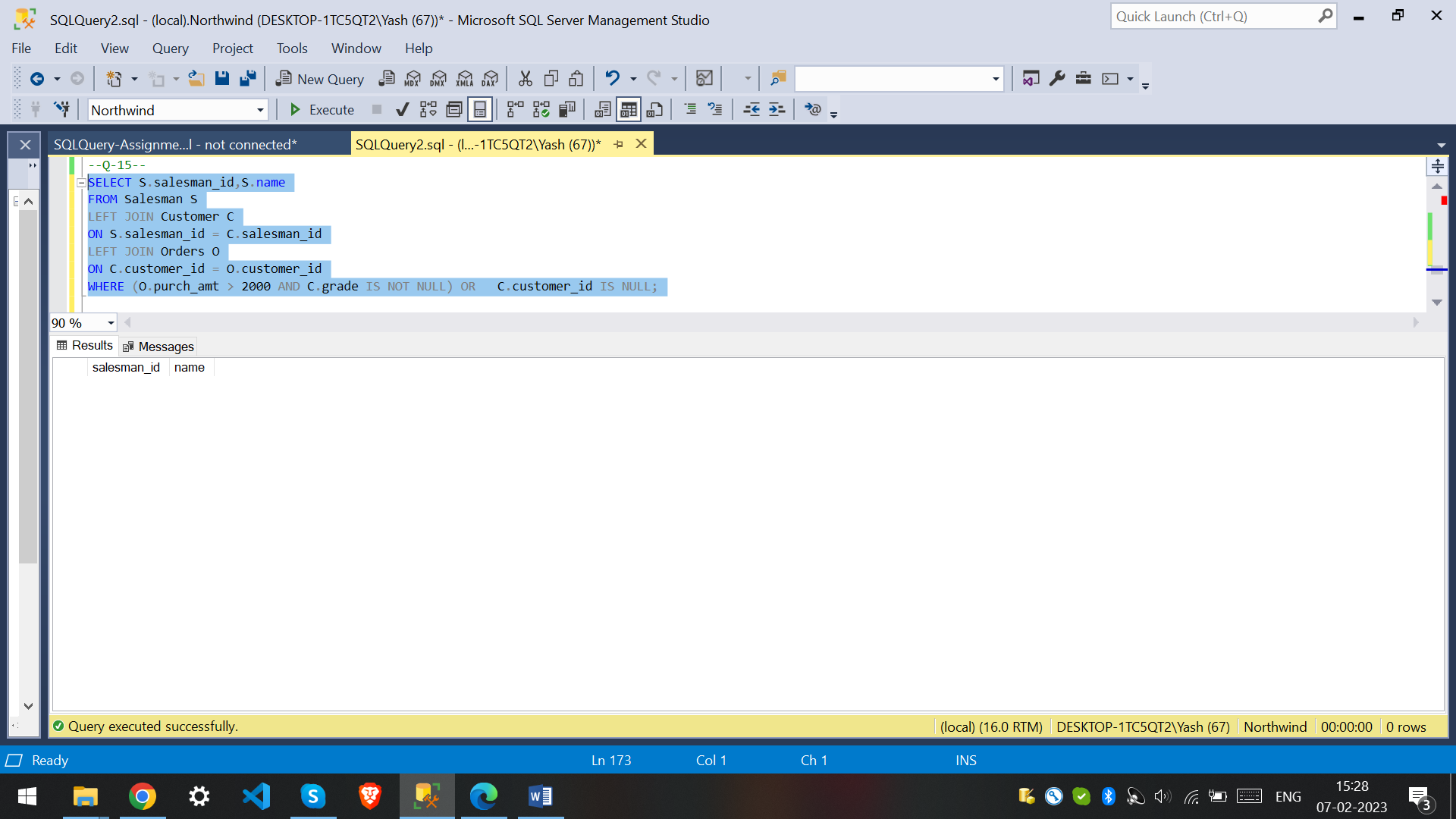
LEFT JOIN Customer C

ON S.salesman\_id = C.salesman\_id

LEFT JOIN Orders O

ON C.customer\_id = O.customer\_id

WHERE (O.purch\_amt > 2000 AND C.grade IS NOT NULL) OR C.customer\_id IS 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.

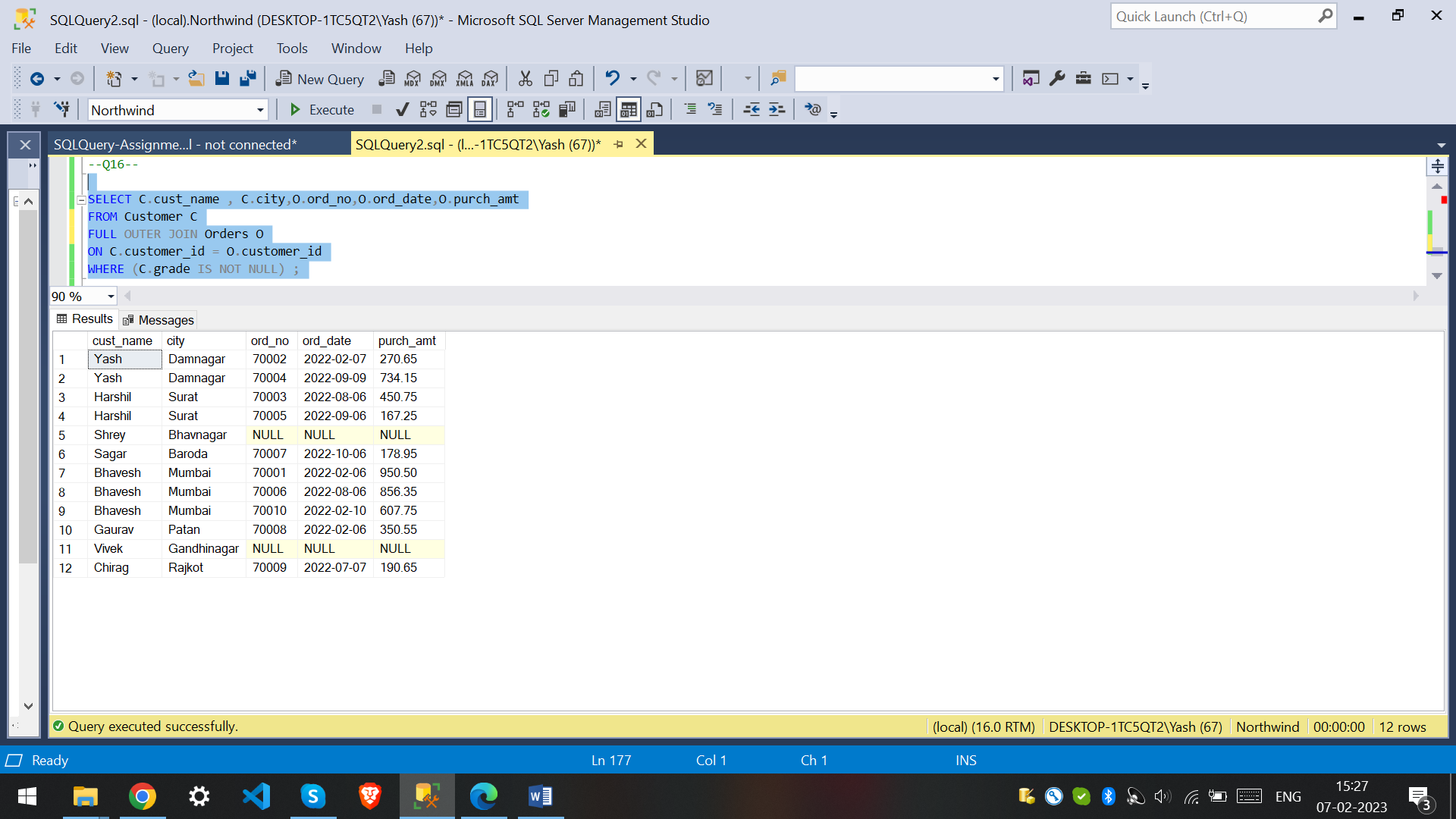
SELECT C.cust\_name , C.city,O.ord\_no,O.ord\_date,O.purch\_amt

FROM Customer C

FULL OUTER JOIN Orders O

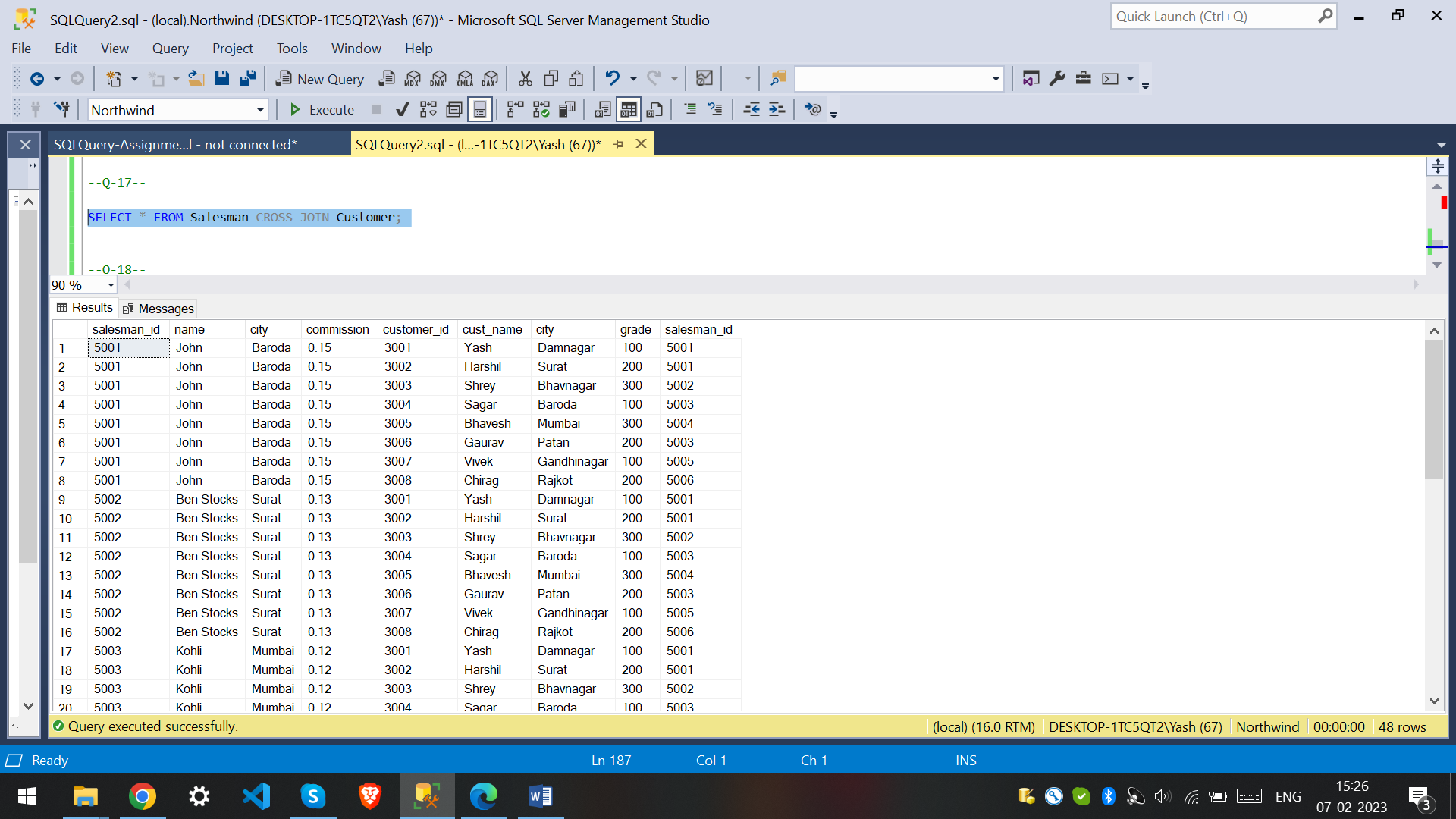
ON C.customer\_id = O.customer\_id

WHERE (C.grade IS NOT NULL) ;



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

SELECT \* FROM Salesman CROSS JOIN Customer;

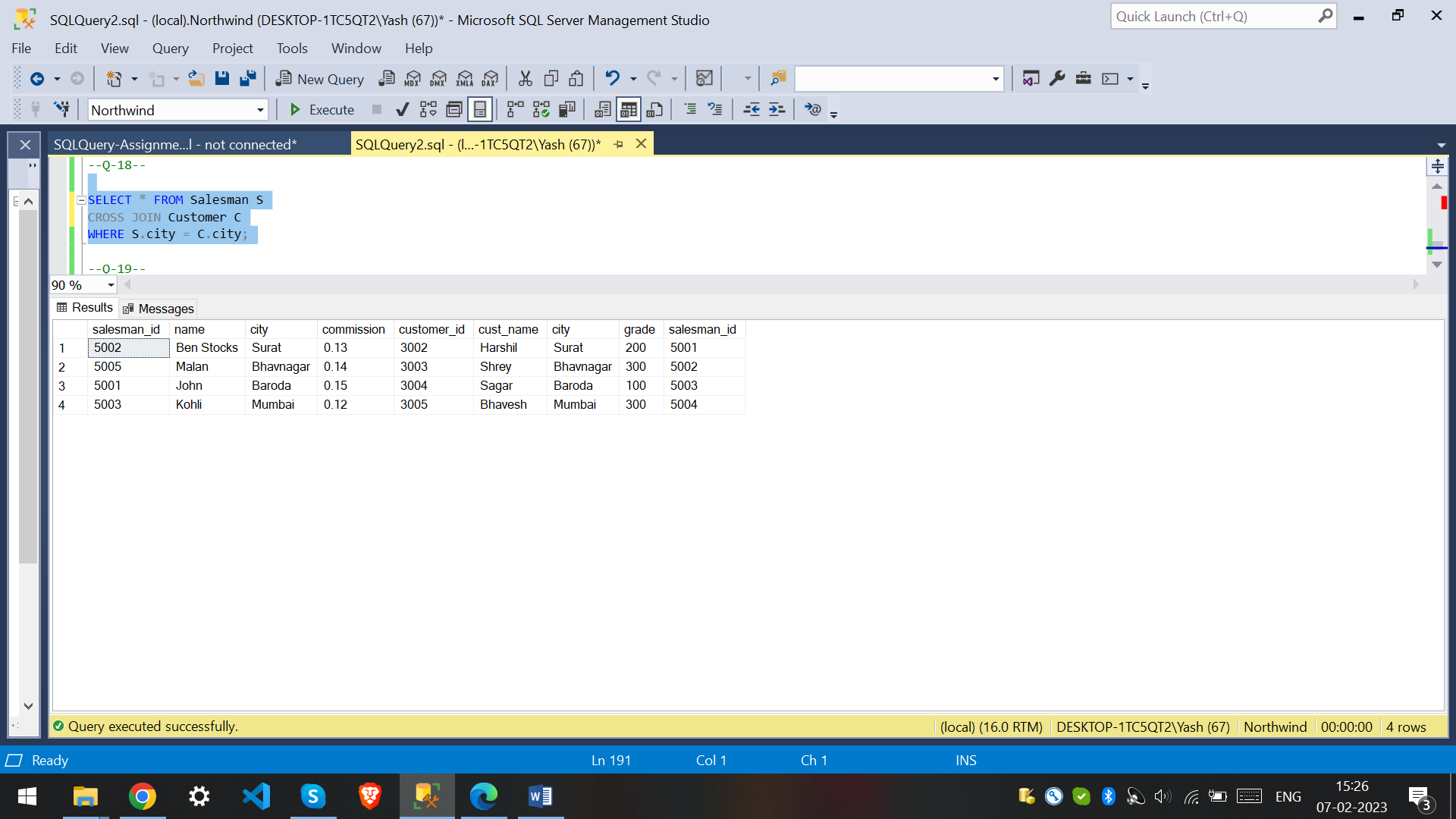


18. Write a SQL statement to create a Cartesian product between salesperson and customer, i.e. each salesperson will appear for all customers and vice versa for that salesperson who belongs to that city

SELECT \* FROM Salesman S

CROSS JOIN Customer C

WHERE S.city = C.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

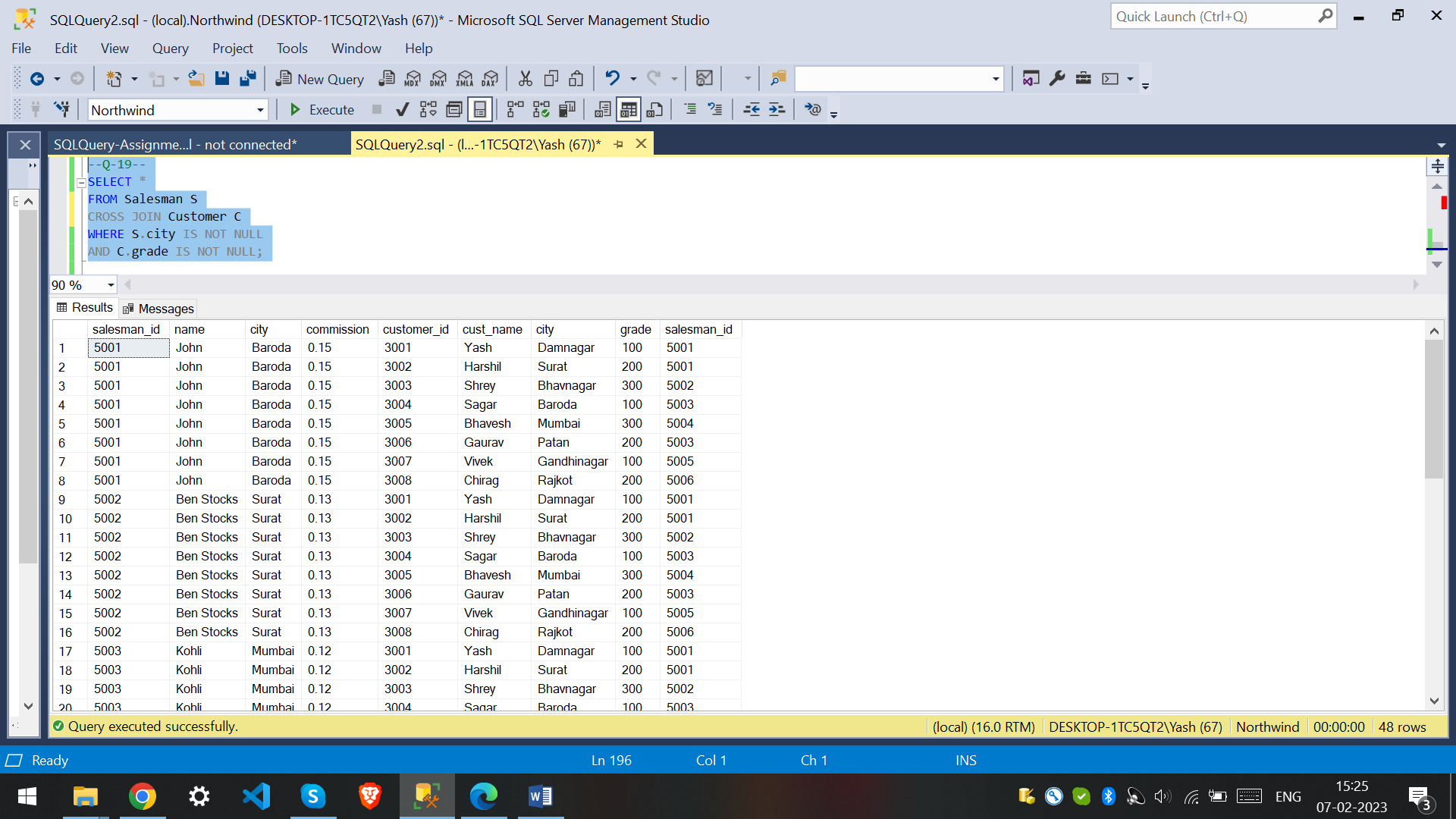
SELECT \*

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

SELECT \*

FROM Salesman S

CROSS JOIN Customer C

WHERE S.city IS NOT NULL

AND S.city <> C.city AND C.grade IS NOT NULL;

