**Assignment 2**

CREATE TABLE salesman(

salesman\_id int not null primary key,

name varchar(50) not null,

city varchar(30) not null,

commission float not null

);

CREATE TABLE customer(

customer\_id int not null primary key,

cust\_name varchar(50) not null,

city varchar(30) not null,

grade int not null,

salesman\_id int not null FOREIGN KEY REFERENCES salesman(salesman\_id)

);

CREATE TABLE orders(

ord\_no int not null primary key,

purch\_amt decimal(10,2) not null,

ord\_date date not null,

customer\_id int not null FOREIGN KEY REFERENCES customer(customer\_id),

salesman\_id int not null FOREIGN KEY REFERENCES salesman(salesman\_id)

);

--Insert Data

INSERT INTO salesman VALUES

(5001 , 'James Hoog' , 'New York' , 0.15),

(5002 , 'Nail Knite' , 'Paris' , 0.13),

(5005 , 'Pit Alex' , 'London' , 0.11),

(5006 , 'Mc Lyon' , 'Paris' , 0.14),

(5007 , 'Paul Adam' , 'Rome' , 0.13),

(5003 , 'Lauson Hen' , 'San Jose' , 0.12);

INSERT INTO customer VALUES

(3001 , 'Brad Guzan' , 'London' , 300 , 5005),

(3002 , 'Nick Rimando' , 'New York' , 100 , 5001),

(3003 , 'Jozy Altidor' , 'Moscow' , 200 , 5007),

(3004 , 'Fabian Johnson' , 'Paris' , 300 , 5006),

(3005 , 'Graham Zusi' , 'California' , 200 , 5002),

(3006 , 'Brad Davis' , 'New York' , 200 , 5001),

(3007 , 'Julian Green' , 'London' , 300 , 5002),

(3008 , 'Geoff Cameron' , 'Berlin' , 100 , 5003);

INSERT INTO orders VALUES

(70001 , 150.5 , '2012-10-05' , 3005 , 5002 ),

(70002 , 65.26 , '2012-10-05' , 3002 , 5001 ),

(70003 , 2480.4 , '2012-10-10' , 3008 , 5003 ),

(70004 , 110.5 , '2012-08-17' , 3008 , 5003 ),

(70005 , 2400.6 , '2012-07-27' , 3006 , 5001 ),

(70007 , 948.5 , '2012-09-10' , 3005 , 5002 ),

(70009 , 270.65 , '2012-09-10' , 3001 , 5005 ),

(70008 , 5760 , '2012-09-10' , 3002 , 5001 ),

(70010 , 1983.43 , '2012-10-10' , 3004 , 5006 ),

(70011 , 75.29 , '2012-08-17' , 3003 , 5007 ),

(70012 , 250.45 , '2012-06-27' , 3007 , 5002 ),

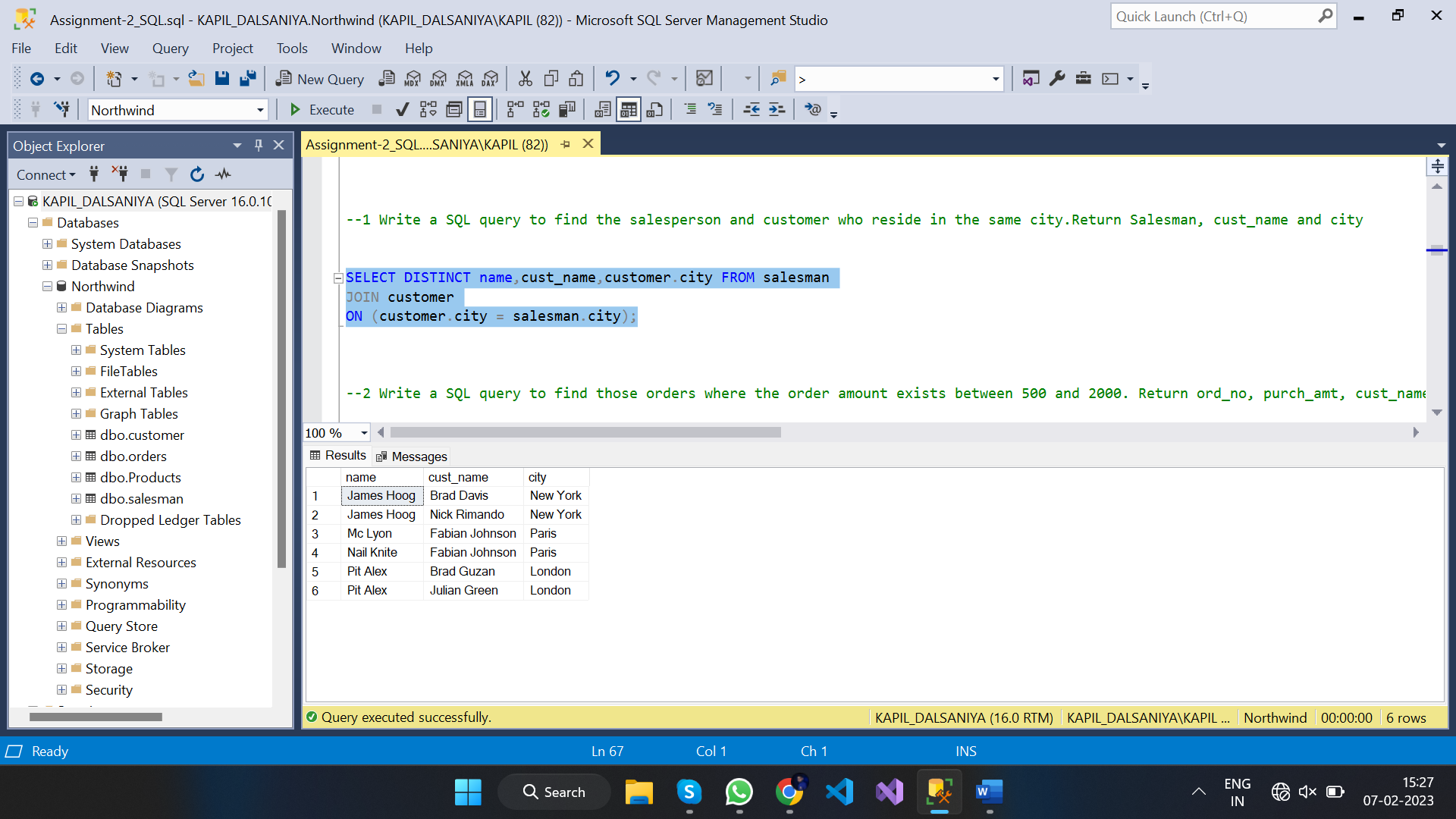
(70013 , 3045.6 , '2012-04-25' , 3002 , 5001 );

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

SELECT DISTINCT name,cust\_name,customer.city FROM salesman

JOIN customer

ON (customer.city = salesman.city);



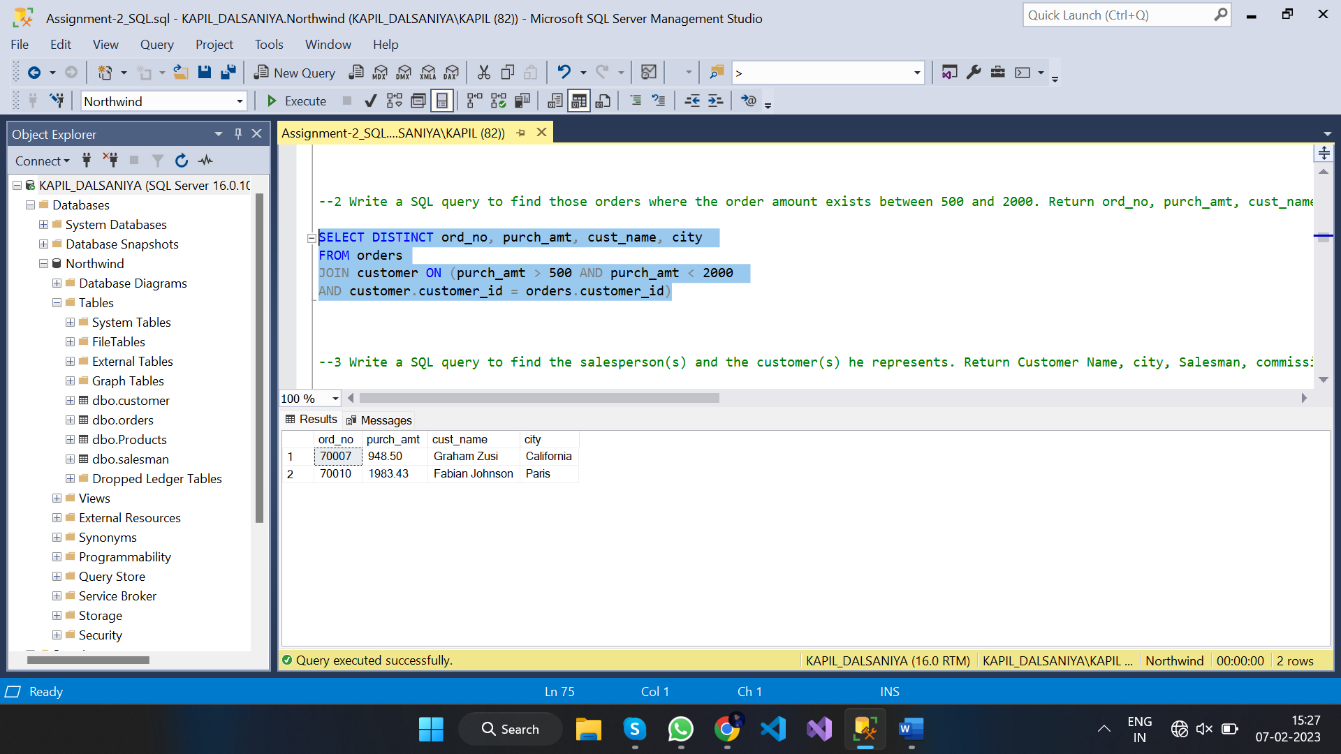
1. --2 Write a SQL query to find those orders where the order amount exists between 500 and 2000. Return ord\_no, purch\_amt, cust\_name, city

SELECT DISTINCT ord\_no, purch\_amt, cust\_name, city

FROM orders

JOIN customer ON (purch\_amt > 500 AND purch\_amt < 2000

AND customer.customer\_id = orders.customer\_id)



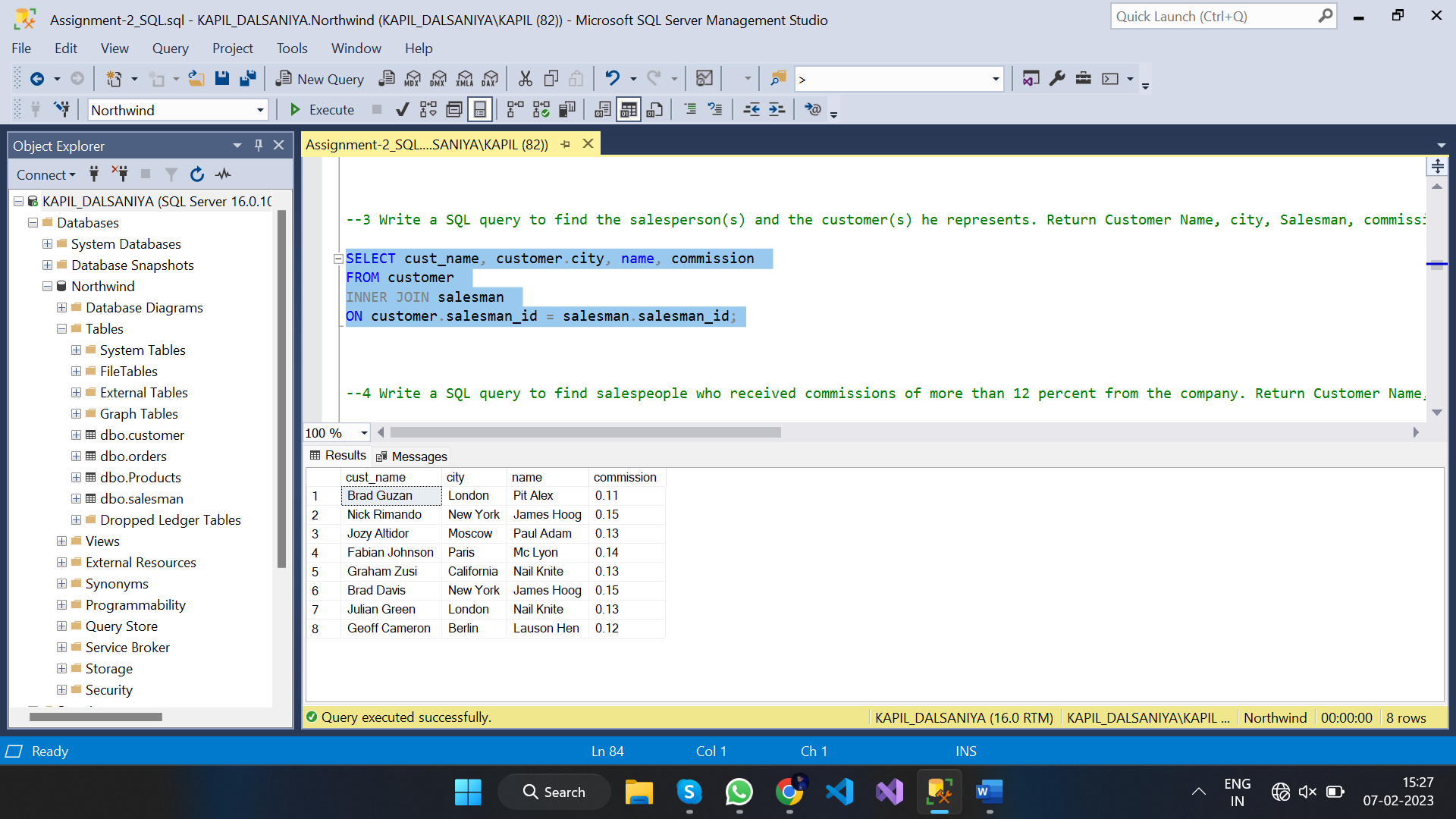
1. --3 Write a SQL query to find the salesperson(s) and the customer(s) he represents. Return Customer Name, city, Salesman, commission

SELECT cust\_name, customer.city, name, commission

FROM customer

INNER JOIN salesman

ON customer.salesman\_id = salesman.salesman\_id;



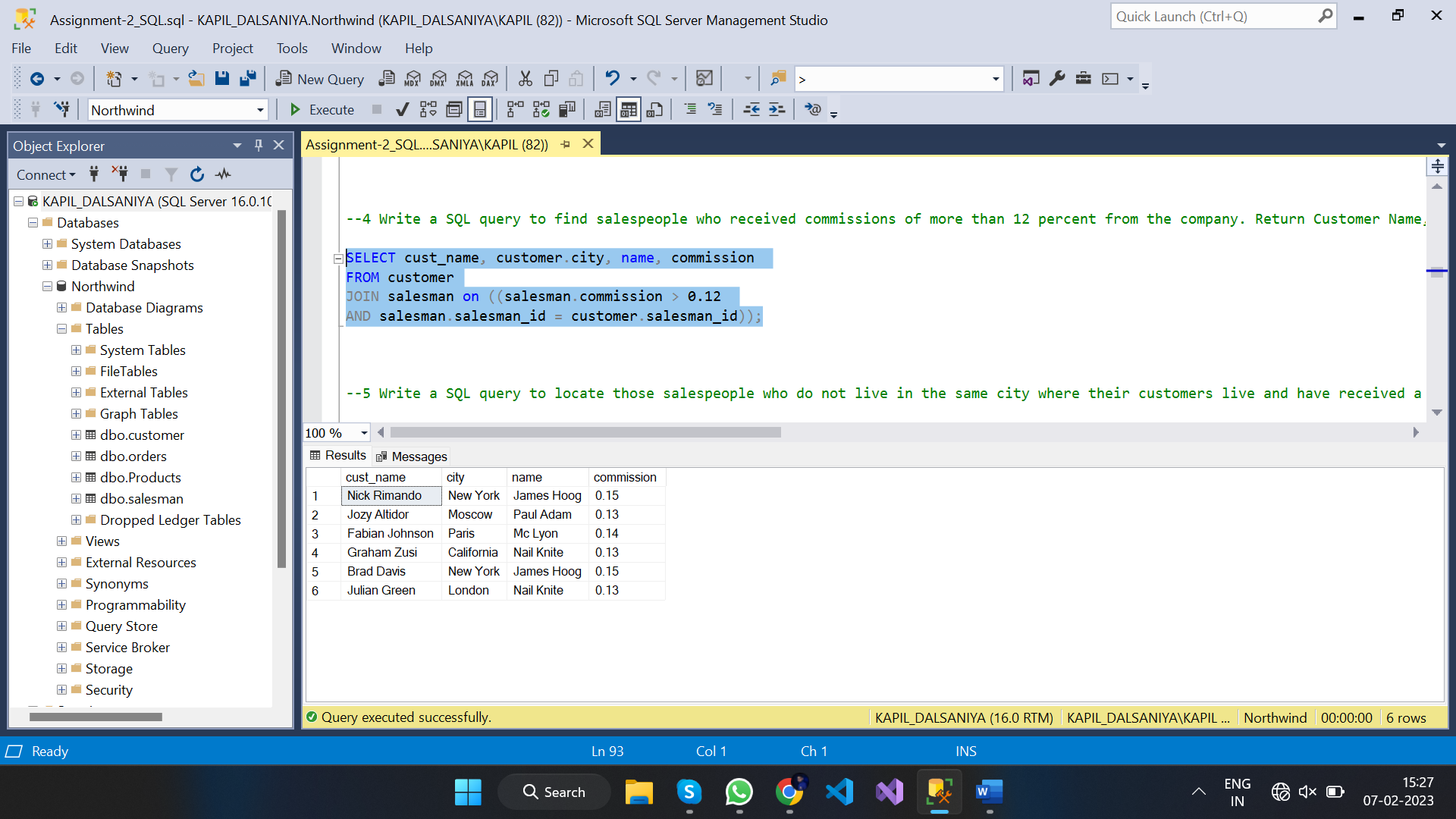
1. --4 Write a SQL query to find salespeople who received commissions of more than 12 percent from the company. Return Customer Name, customer city, Salesman, commission

SELECT cust\_name, customer.city, name, commission

FROM customer

JOIN salesman on ((salesman.commission > 0.12

AND salesman.salesman\_id = customer.salesman\_id));



1. --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 cust\_name, customer.city AS customer\_city,

name, salesman.city AS sales\_city, commission

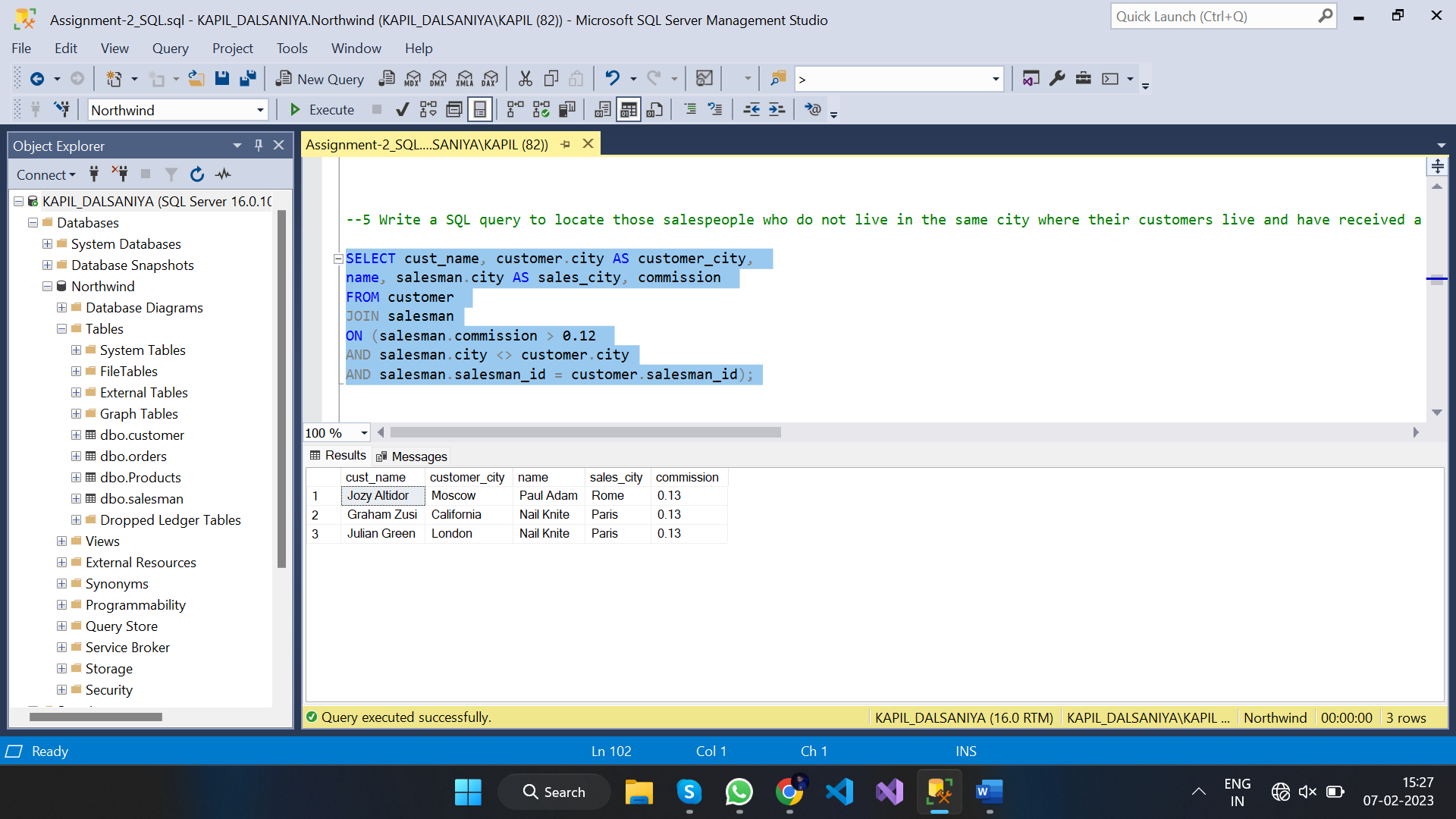
FROM customer

JOIN salesman

ON (salesman.commission > 0.12

AND salesman.city <> customer.city

AND salesman.salesman\_id = customer.salesman\_id);



1. --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 ord\_no, ord\_date, purch\_amt, cust\_name, grade, name AS salesman, commission

FROM orders

JOIN customer ON orders.customer\_id = customer.customer\_id

JOIN salesman ON customer.salesman\_id = salesman.salesman\_id;



1. --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 ord\_no, purch\_amt, ord\_date,

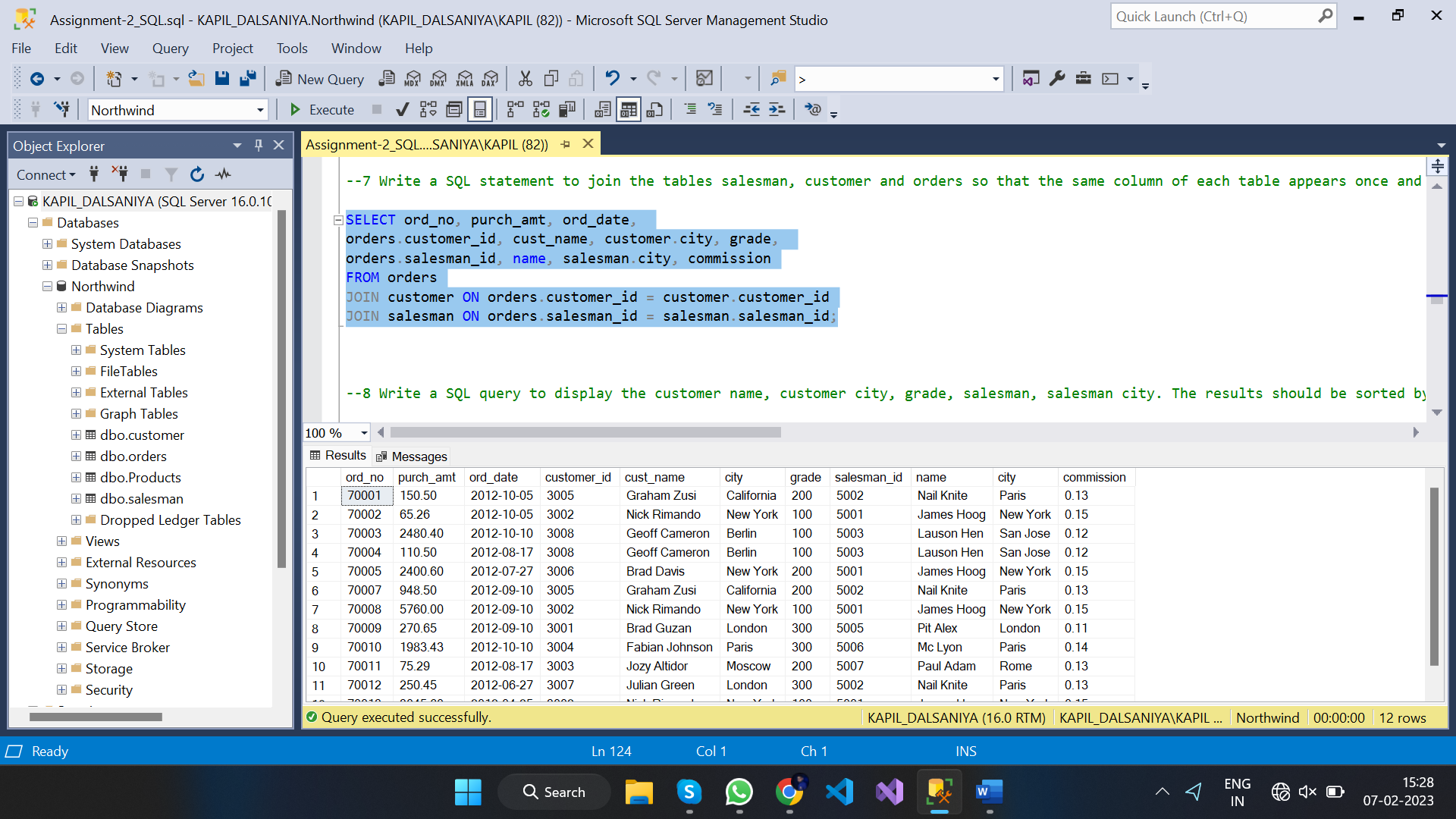
orders.customer\_id, cust\_name, customer.city, grade,

orders.salesman\_id, name, salesman.city, commission

FROM orders

JOIN customer ON orders.customer\_id = customer.customer\_id

JOIN salesman ON orders.salesman\_id = salesman.salesman\_id;



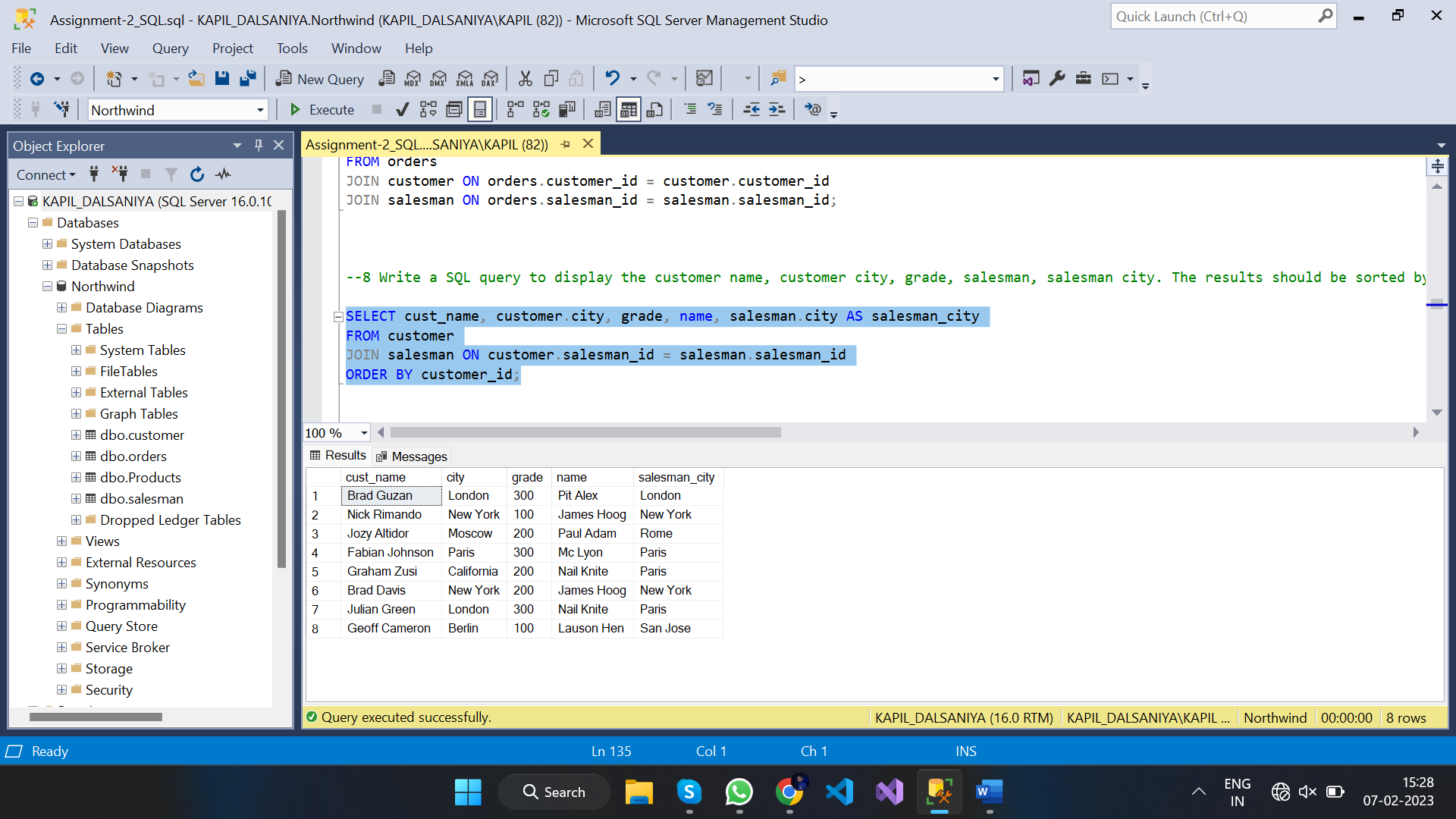
1. --8 Write a SQL query to display the customer name, customer city, grade, salesman, salesman city. The results should be sorted by ascending customer\_id -

SELECT cust\_name, customer.city, grade, name, salesman.city AS salesman\_city

FROM customer

JOIN salesman ON customer.salesman\_id = salesman.salesman\_id

ORDER BY customer\_id;



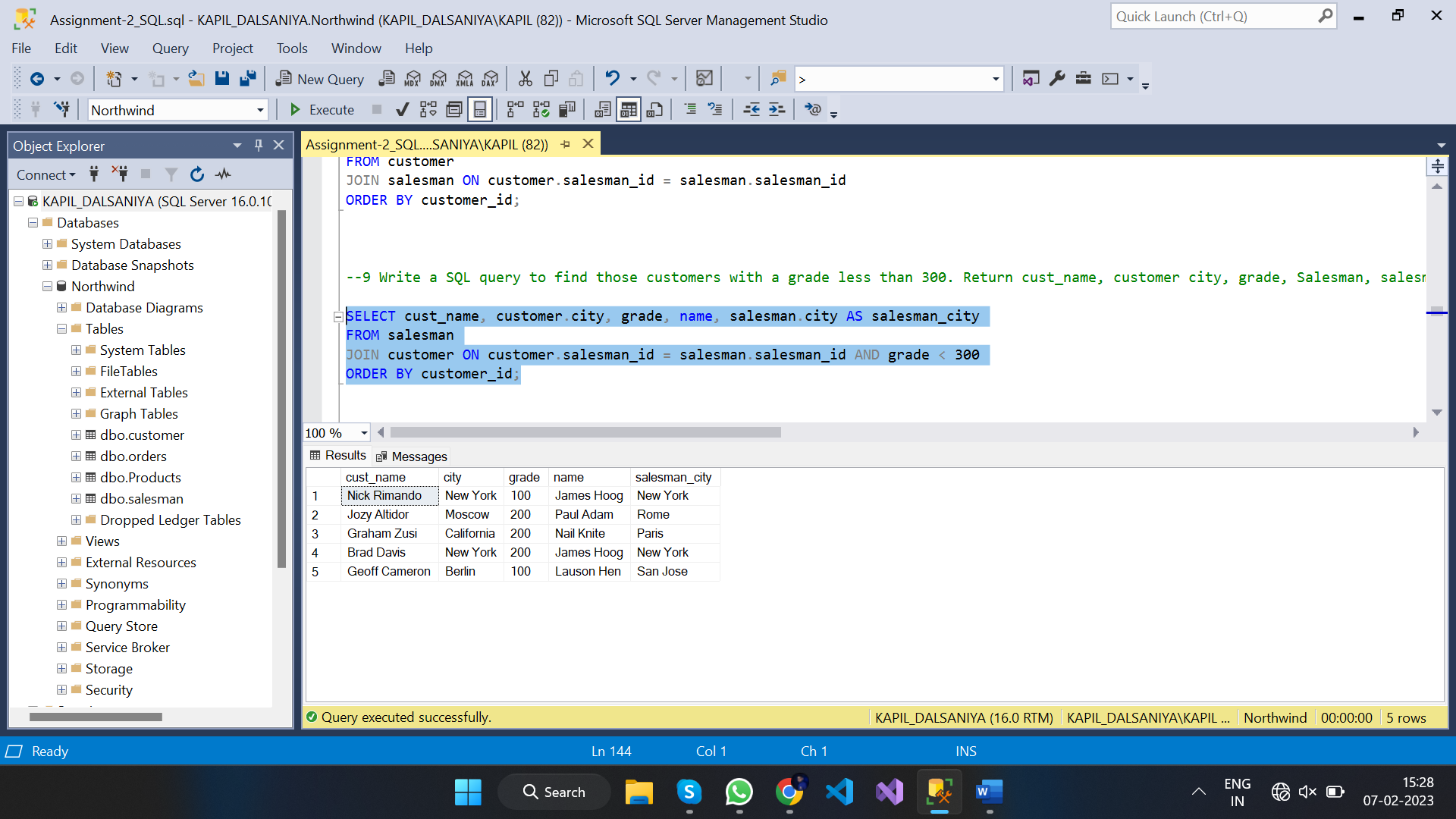
1. --9 Write a SQL query to find those customers with a grade less than 300. Return cust\_name, customer city, grade, Salesman, salesmancity. The result should be ordered by ascending customer\_id

SELECT cust\_name, customer.city, grade, name, salesman.city AS salesman\_city

FROM salesman

JOIN customer ON customer.salesman\_id = salesman.salesman\_id AND grade < 300

ORDER BY customer\_id;



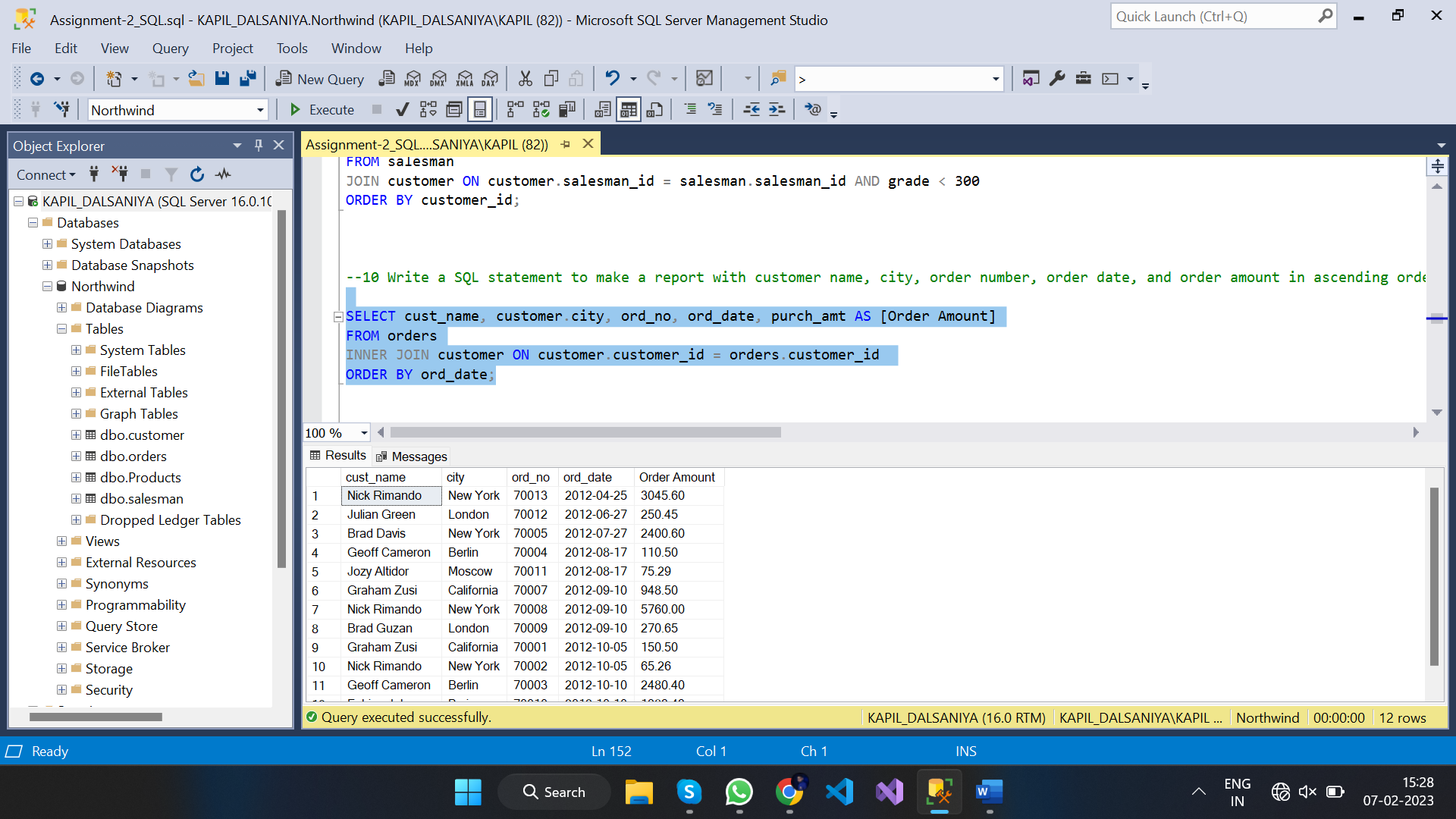
1. --10 Write a SQL statement to make a report with customer name, city, order number, order date, and order amount in ascending order according to the order date to determine whether any of the existing customers have placed an order or not

SELECT cust\_name, customer.city, ord\_no, ord\_date, purch\_amt AS [Order Amount]

FROM orders

INNER JOIN customer ON customer.customer\_id = orders.customer\_id

ORDER BY ord\_date;



1. --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

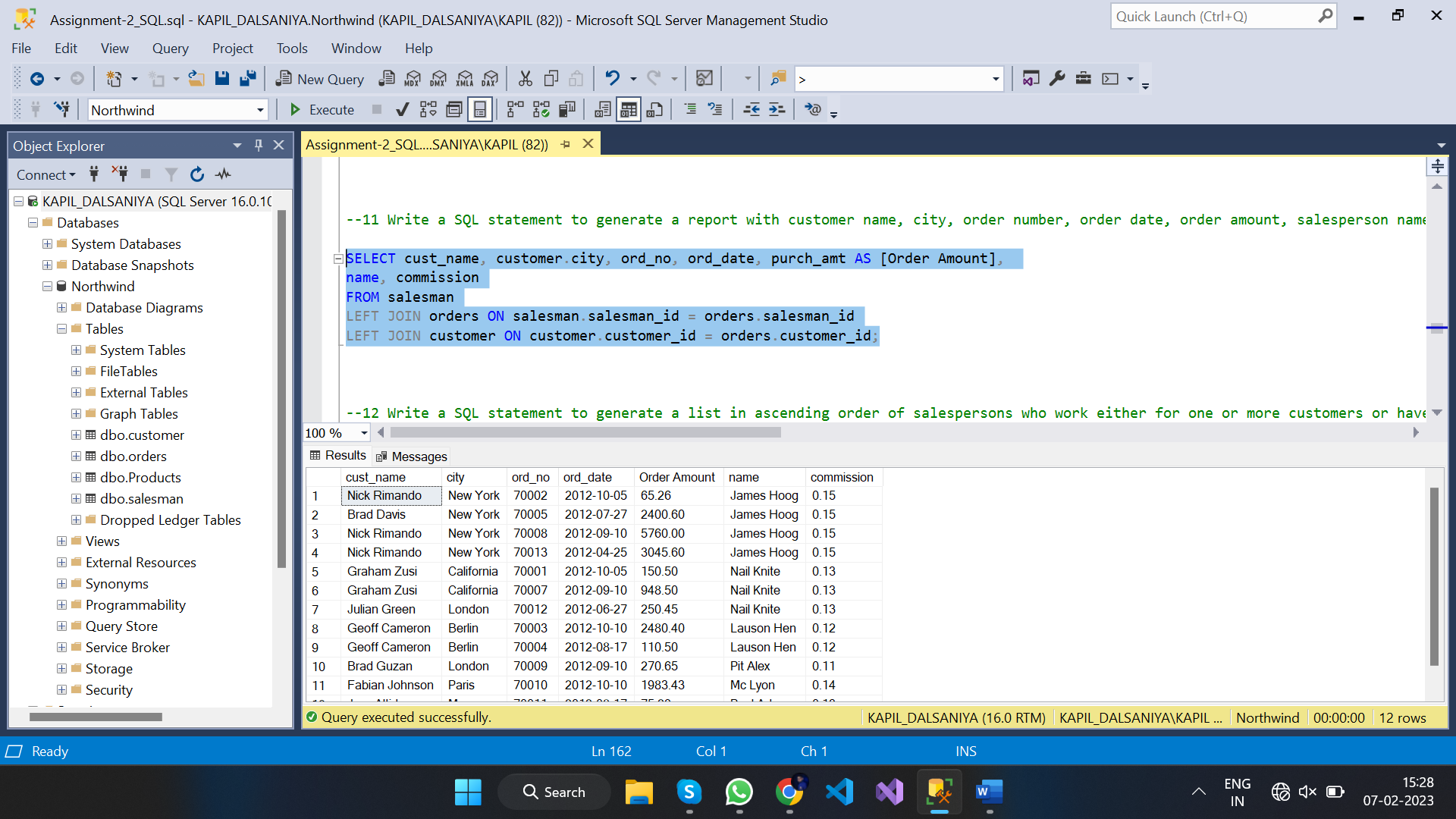
SELECT cust\_name, customer.city, ord\_no, ord\_date, purch\_amt AS [Order Amount],

name, commission

FROM salesman

LEFT JOIN orders ON salesman.salesman\_id = orders.salesman\_id

LEFT JOIN customer ON customer.customer\_id = orders.customer\_id;



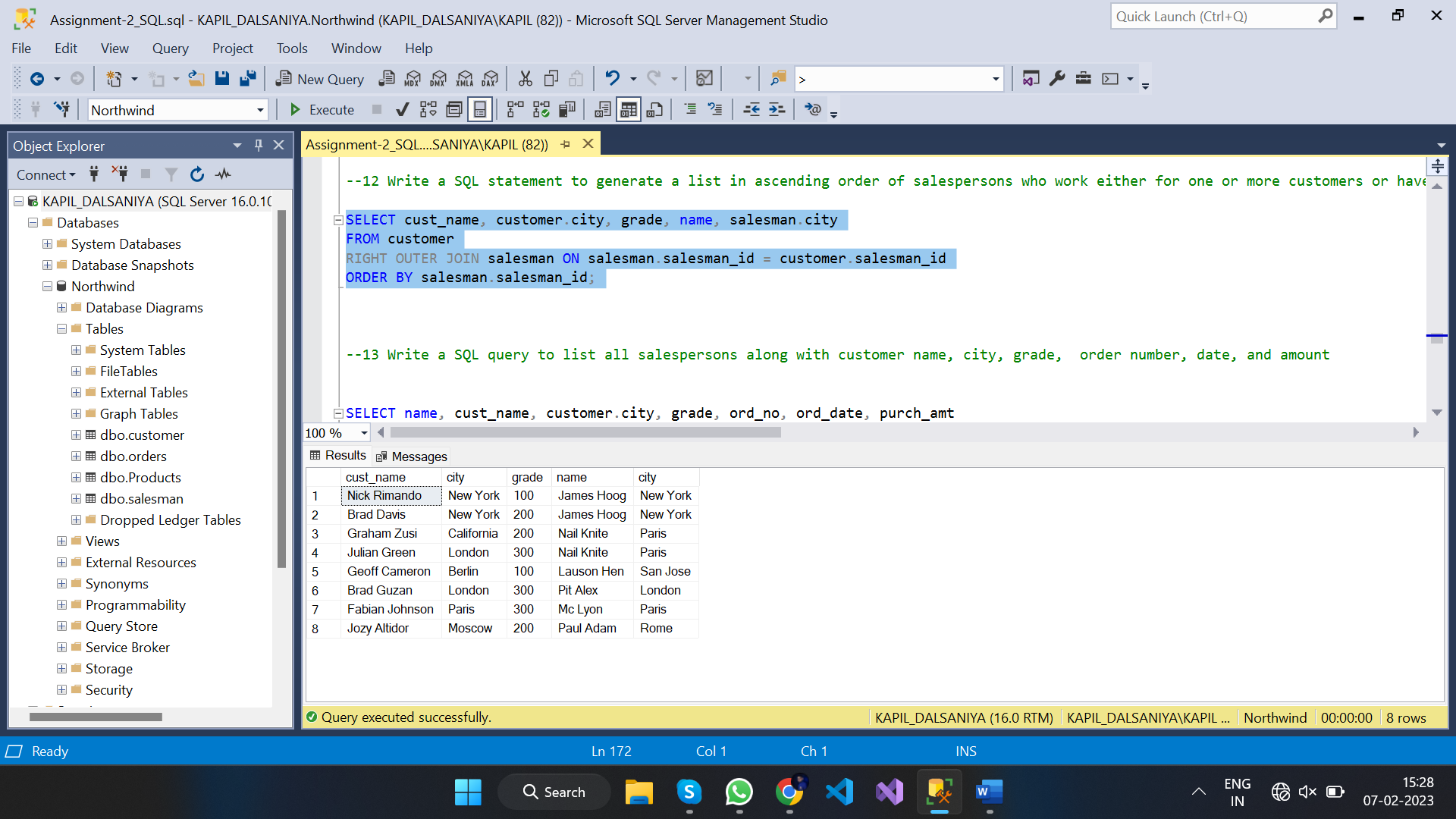
1. --12 Write a SQL statement to generate a list in ascending order of salespersons who work either for one or more customers or have not yet joined any of the customers

SELECT cust\_name, customer.city, grade, name, salesman.city

FROM customer

RIGHT OUTER JOIN salesman ON salesman.salesman\_id = customer.salesman\_id

ORDER BY salesman.salesman\_id;



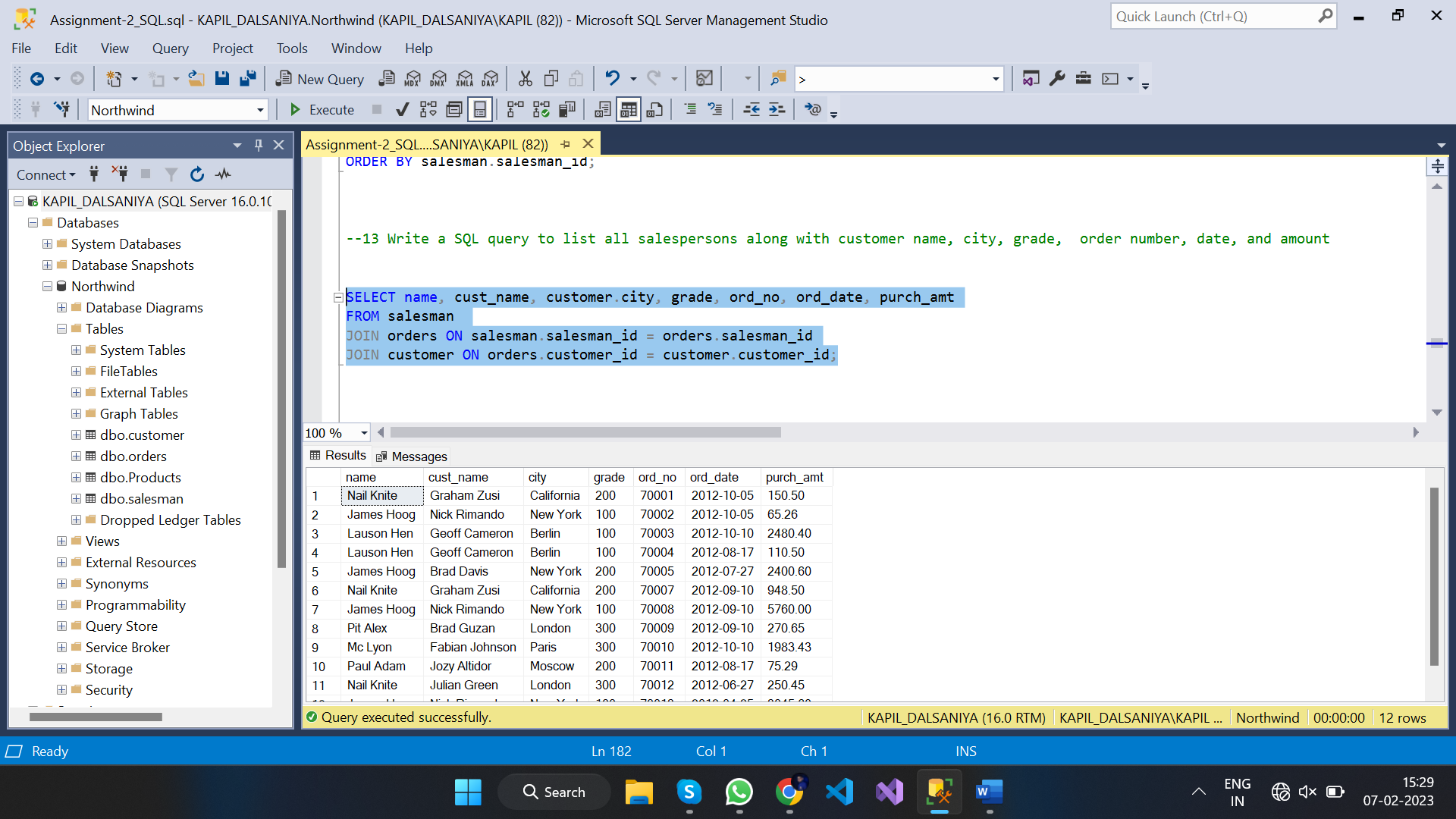
1. --13 Write a SQL query to list all salespersons along with customer name, city, grade, order number, date, and amount

SELECT name, cust\_name, customer.city, grade, ord\_no, ord\_date, purch\_amt

FROM salesman

JOIN orders ON salesman.salesman\_id = orders.salesman\_id

JOIN customer ON orders.customer\_id = customer.customer\_id;



1. --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

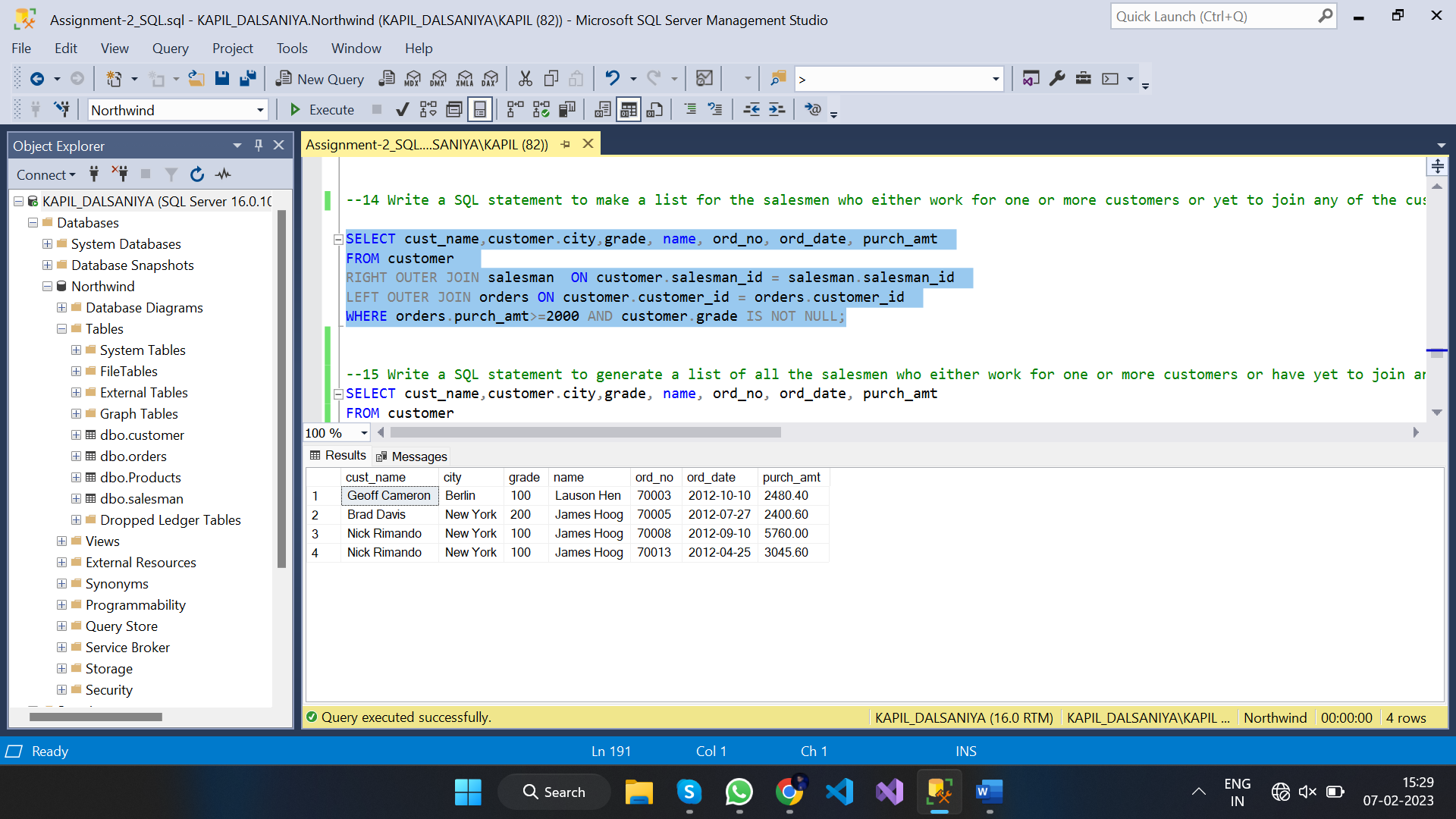
SELECT cust\_name,customer.city,grade, name, ord\_no, ord\_date, purch\_amt

FROM customer

RIGHT OUTER JOIN salesman ON customer.salesman\_id = salesman.salesman\_id

LEFT OUTER JOIN orders ON customer.customer\_id = orders.customer\_id

WHERE orders.purch\_amt>=2000 AND customer.grade IS NOT NULL;



1. --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.

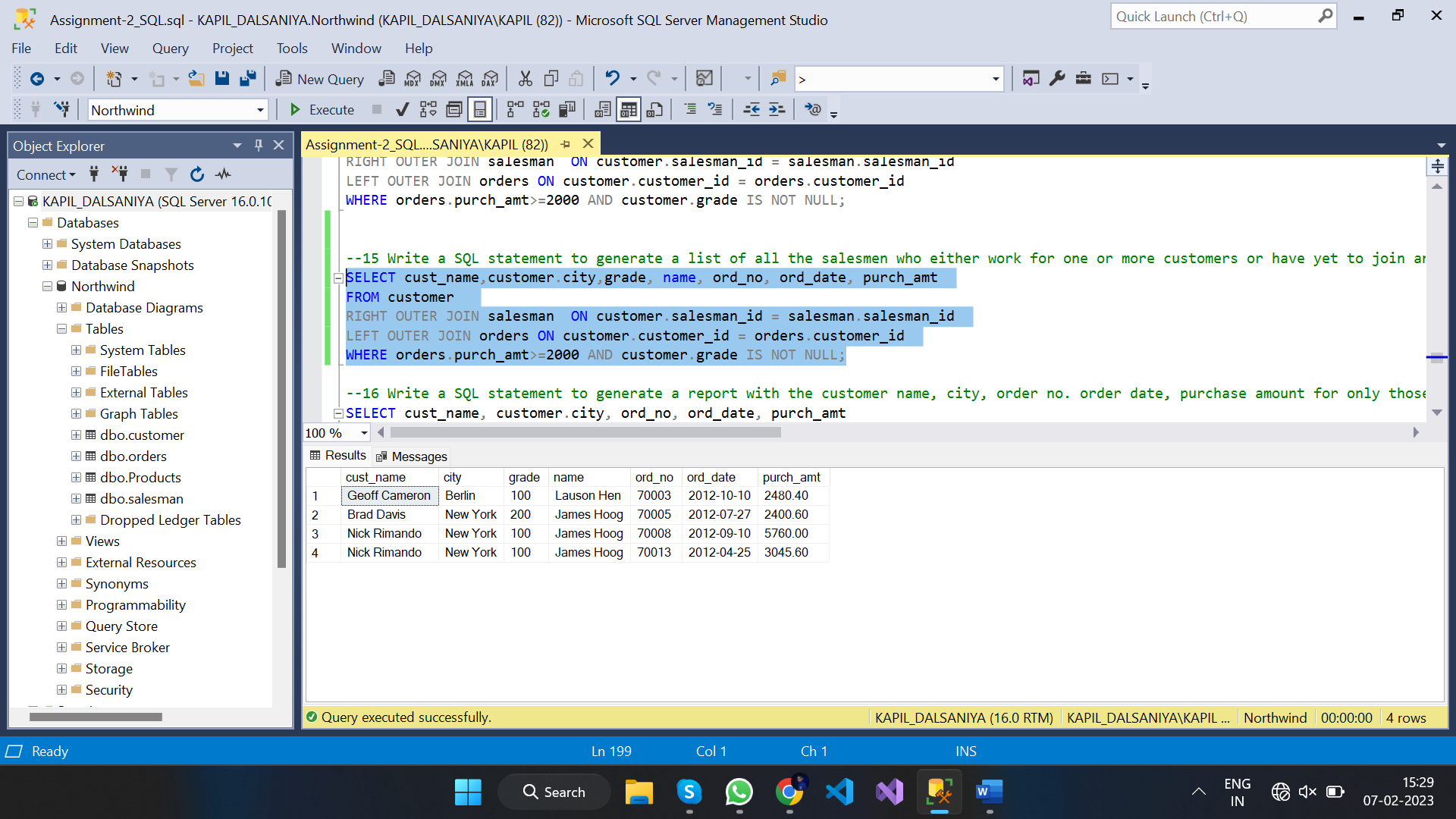
SELECT cust\_name,customer.city,grade, name, ord\_no, ord\_date, purch\_amt

FROM customer

RIGHT OUTER JOIN salesman ON customer.salesman\_id = salesman.salesman\_id

LEFT OUTER JOIN orders ON customer.customer\_id = orders.customer\_id

WHERE orders.purch\_amt>=2000 AND customer.grade IS NOT NULL;



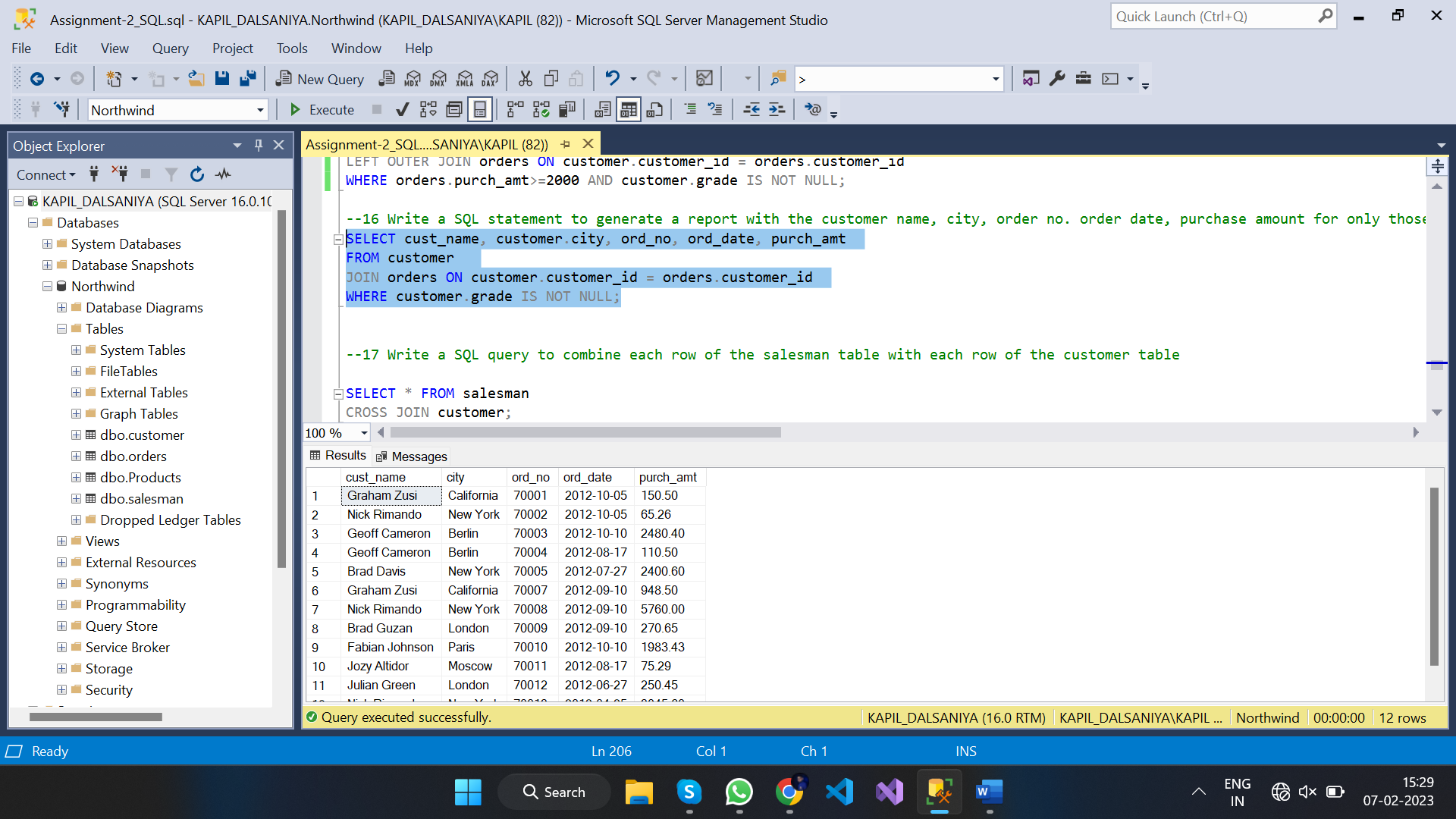
1. --16 Write a SQL statement to generate a report with the customer name, city, order no. order date, purchase amount for only those customers on the list who must have a grade and placed one or more orders or which order(s) have been placed by the customer who neither is on the list nor has a grade

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

FROM customer

JOIN orders ON customer.customer\_id = orders.customer\_id

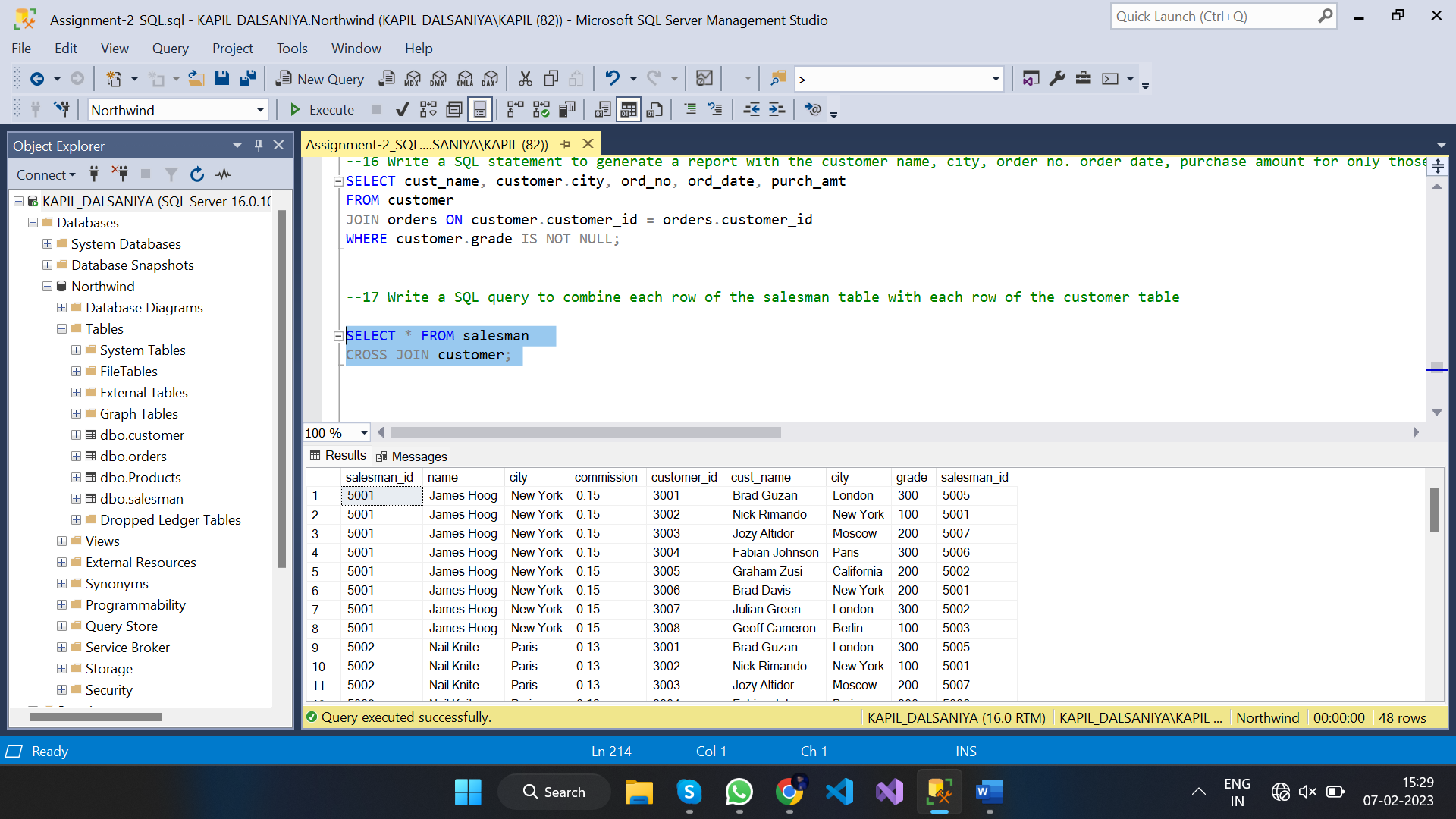
WHERE customer.grade IS NOT NULL;



1. --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;



1. --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

CROSS JOIN customer

WHERE salesman.city = customer.city;



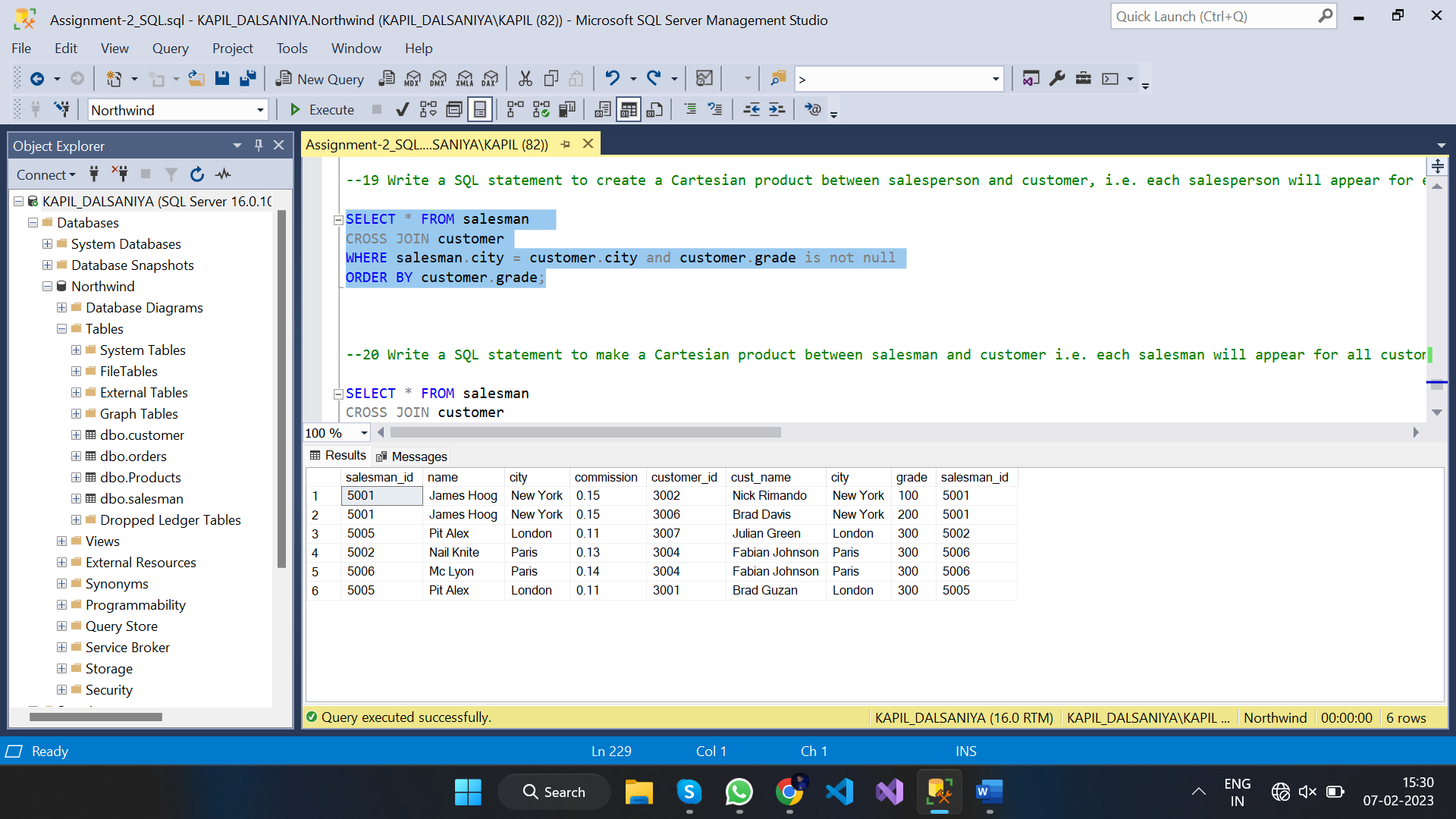
1. --19 Write a SQL statement to create a Cartesian product between salesperson and customer, i.e. each salesperson will appear for every customer and vice versa for those salesmen who belong to a city and customers who require a grade

SELECT \* FROM salesman

CROSS JOIN customer

WHERE salesman.city = customer.city and customer.grade is not null

ORDER BY customer.grade;



1. --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 thecustomers should have their own grade

SELECT \* FROM salesman

CROSS JOIN customer

WHERE salesman.city <> customer.city and customer.grade is not null

ORDER BY customer.grade;

