**Assignment 2**

**Creating tables**

**Query:**

/\*\* Creating Tables \*\*/

/\* Create 'salesman' Table\*/

CREATE TABLE salesman(

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

name VARCHAR(50) NOT NULL,

city VARCHAR(30) NOT NULL,

commission DECIMAL(4, 2) NOT NULL,

)

/\* Create 'customer' Table\*/

CREATE TABLE customer(

customer\_id INT PRIMARY KEY,

cust\_name VARCHAR(50) NOT NULL,

city VARCHAR(30) NOT NULL,

grade INT NULL,

salesman\_id INT NOT NULL FOREIGN KEY REFERENCES salesman(salesman\_id)

)

/\* Create 'orders' Table\*/

CREATE TABLE orders(

ord\_no INT PRIMARY KEY IDENTITY(5000, 1),

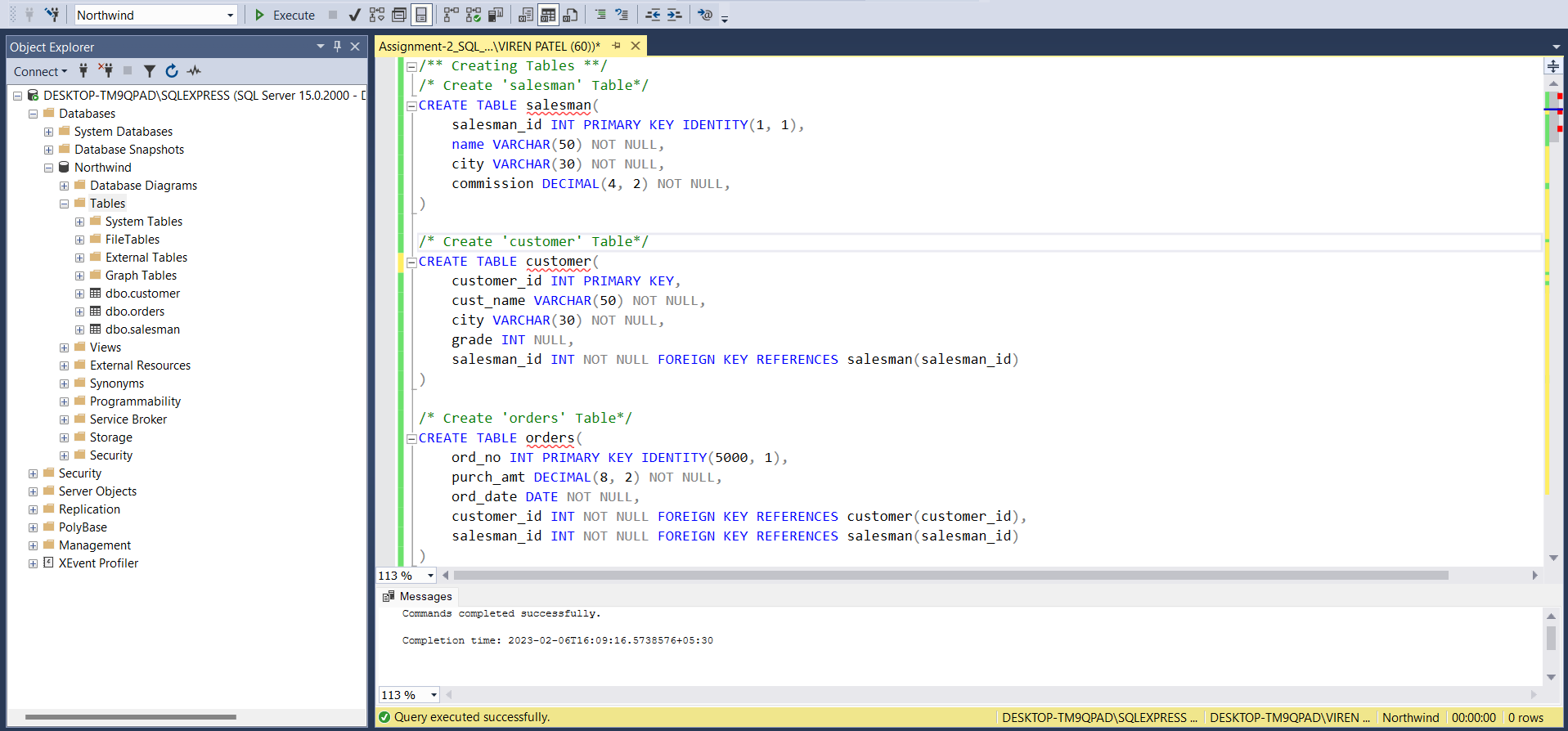
purch\_amt DECIMAL(8, 2) NOT NULL,

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)

)



**Inserting data**

**Query:**

/\*\* Inserting data into created tables \*\*/

/\* salesman \*/

INSERT INTO salesman (name, city, commission)

VALUES ('Viren Laniya', 'New York', 0.15), ('Bhargav Sharma', 'Paris', 0.13),

('Shubham gray','London',0.11), ('Mc Lyon', 'Paris', 0.14), ('Paul Adam', 'Rome', 0.13),

('Lauson Hen', 'San Jose', 0.12), ('Lionel Messi', 'Argentina', 0.15)

/\* customer \*/

INSERT INTO customer (customer\_id, cust\_name, city, grade, salesman\_id)

VALUES (3002, 'Kushal Mehta', 'New York', 100, 1), (3007, 'Salman Malik', 'New York', 200, 1),

(3005, 'Graham Zusi', 'California', 200, 2), (3008, 'Armaan Khan', 'London', 300, 2),

(3004, 'Fabian Johnson', 'Paris', 300, 4), (3009, 'Geoff Cameron', 'Berlin', 100, 6),

(3003, 'Aarun Gaitonde', 'Moscow', 200, 5), (3001, 'Amir mahaan', 'London', 0, 3)

/\* orders \*/

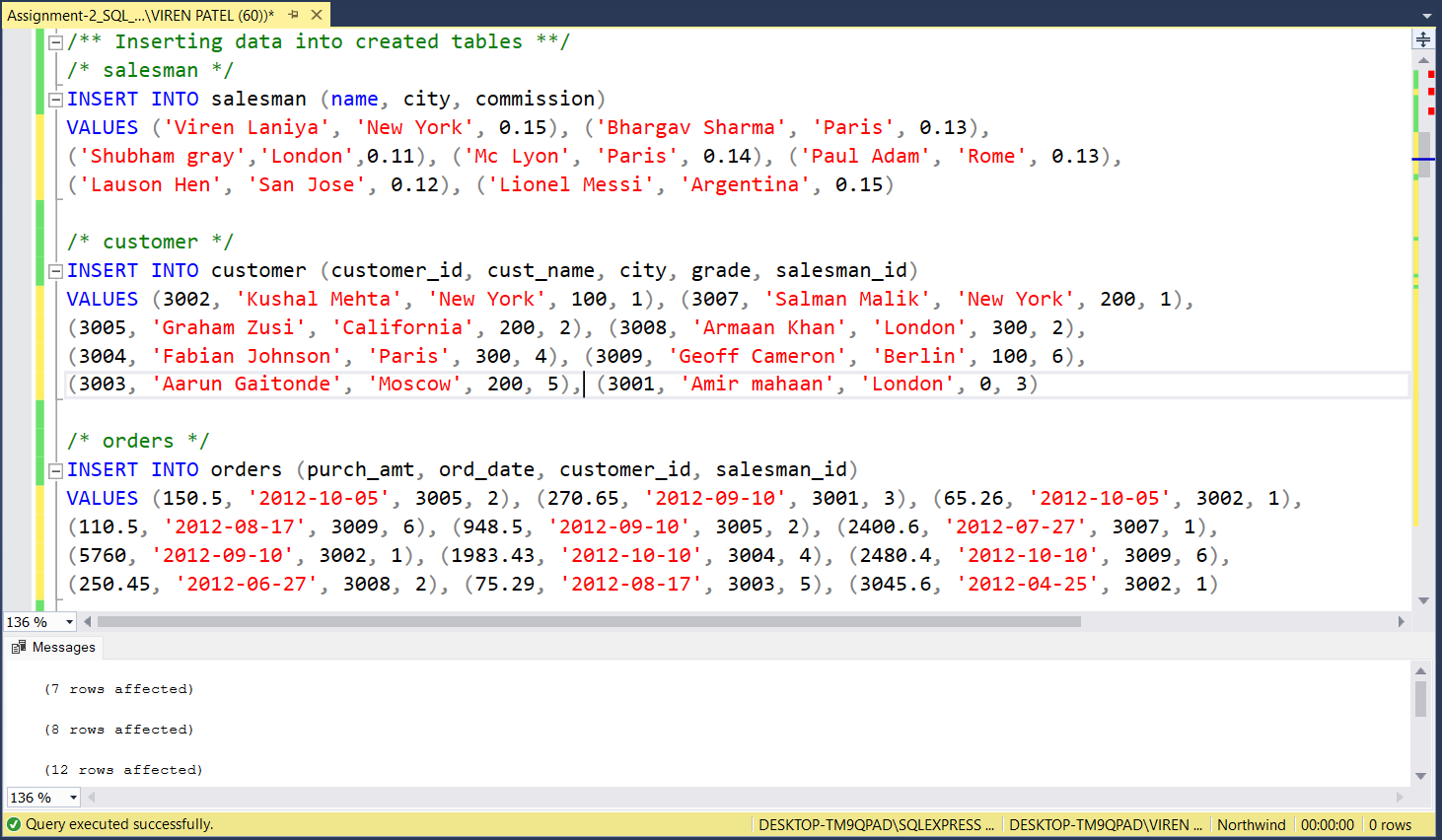
INSERT INTO orders (purch\_amt, ord\_date, customer\_id, salesman\_id)

VALUES (150.5, '2012-10-05', 3005, 2), (270.65, '2012-09-10', 3001, 3), (65.26, '2012-10-05', 3002, 1),

(110.5, '2012-08-17', 3009, 6), (948.5, '2012-09-10', 3005, 2), (2400.6, '2012-07-27', 3007, 1),

(5760, '2012-09-10', 3002, 1), (1983.43, '2012-10-10', 3004, 4), (2480.4, '2012-10-10', 3009, 6),

(250.45, '2012-06-27', 3008, 2), (75.29, '2012-08-17', 3003, 5), (3045.6, '2012-04-25', 3002, 1)



1. **write a SQL query to find the salesperson and customer who reside in the same city.**

**Return Salesman, cust\_name and city**

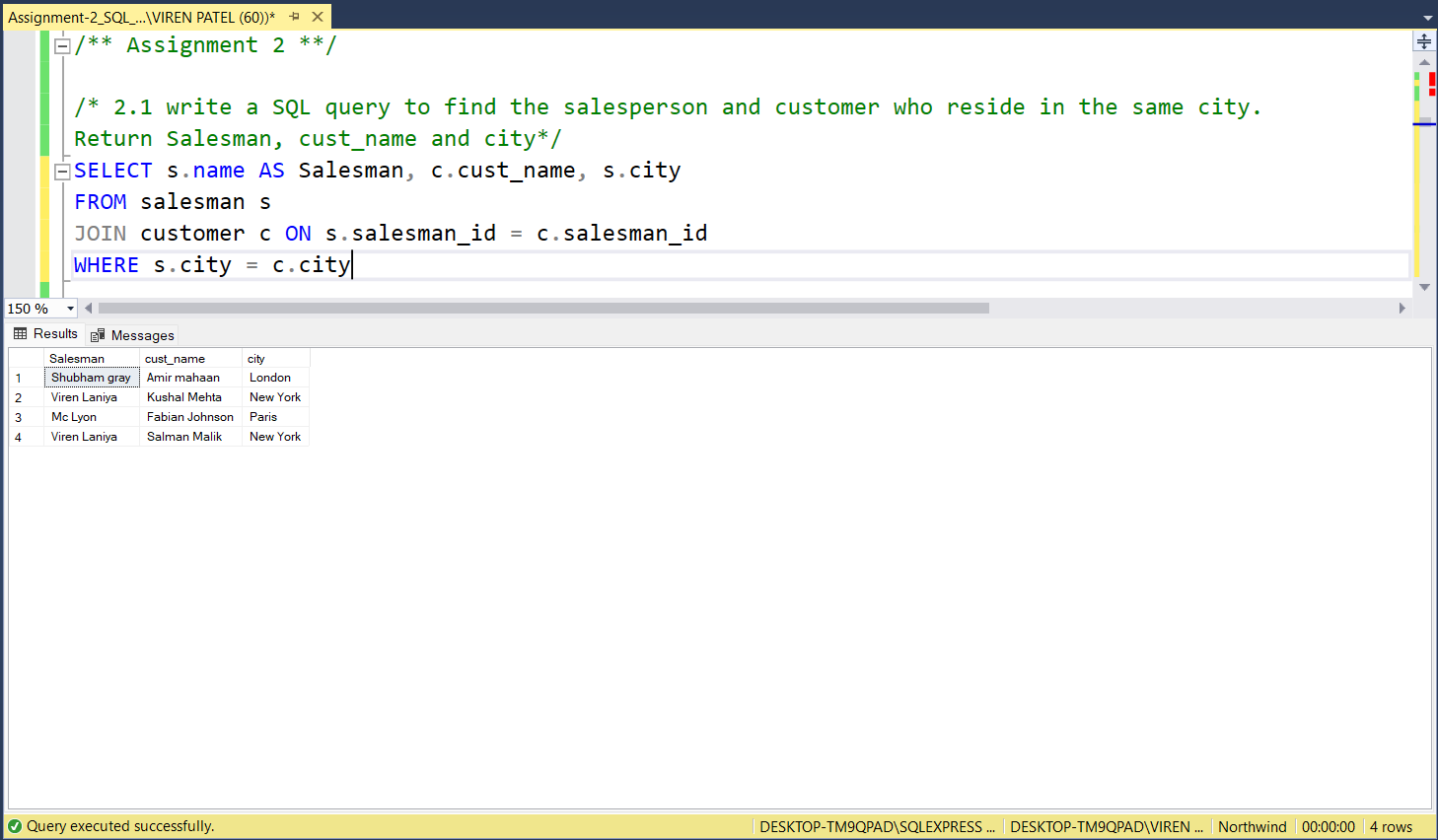
**Query:**

SELECT s.name AS Salesman, c.cust\_name, s.city

FROM salesman s

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

WHERE s.city = c.city



1. **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**

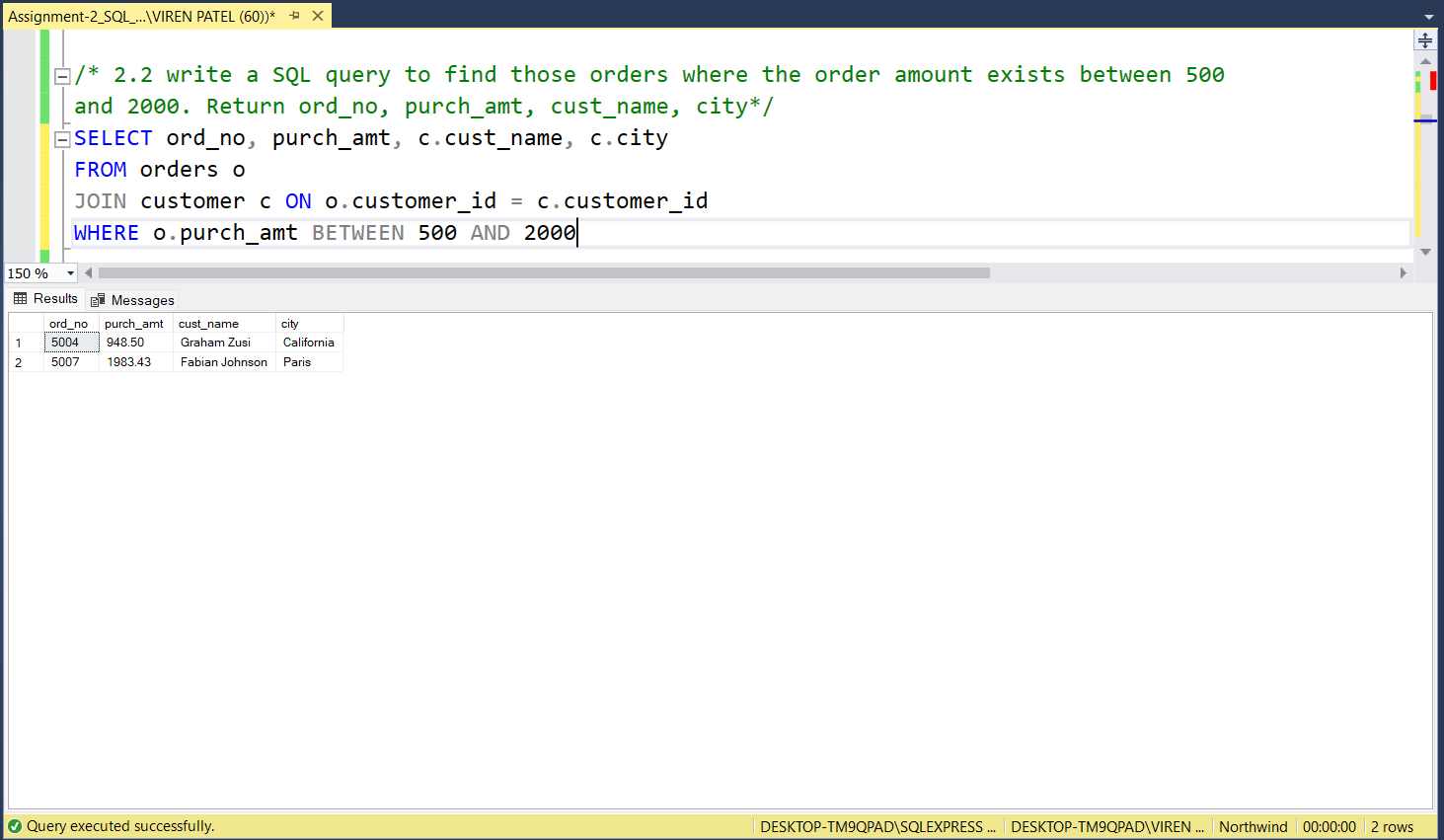
**Query:**

SELECT ord\_no, 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



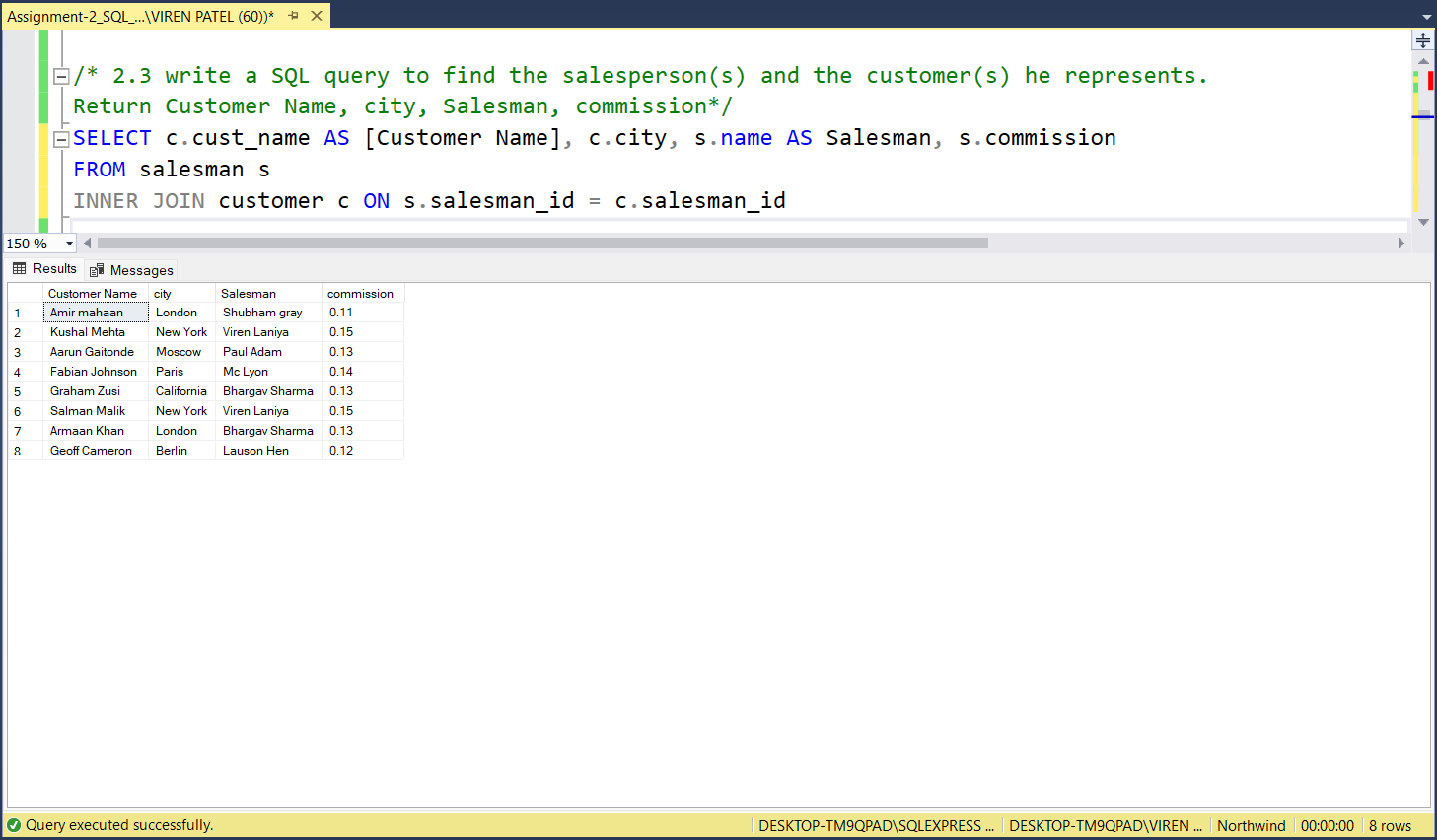
1. **write a SQL query to find the salesperson(s) and the customer(s) he represents. Return Customer Name, city, Salesman, commission**

**Query:**

SELECT c.cust\_name AS [Customer Name], c.city, s.name AS Salesman, s.commission

FROM salesman s

INNER JOIN customer c ON s.salesman\_id = c.salesman\_id



1. **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.**

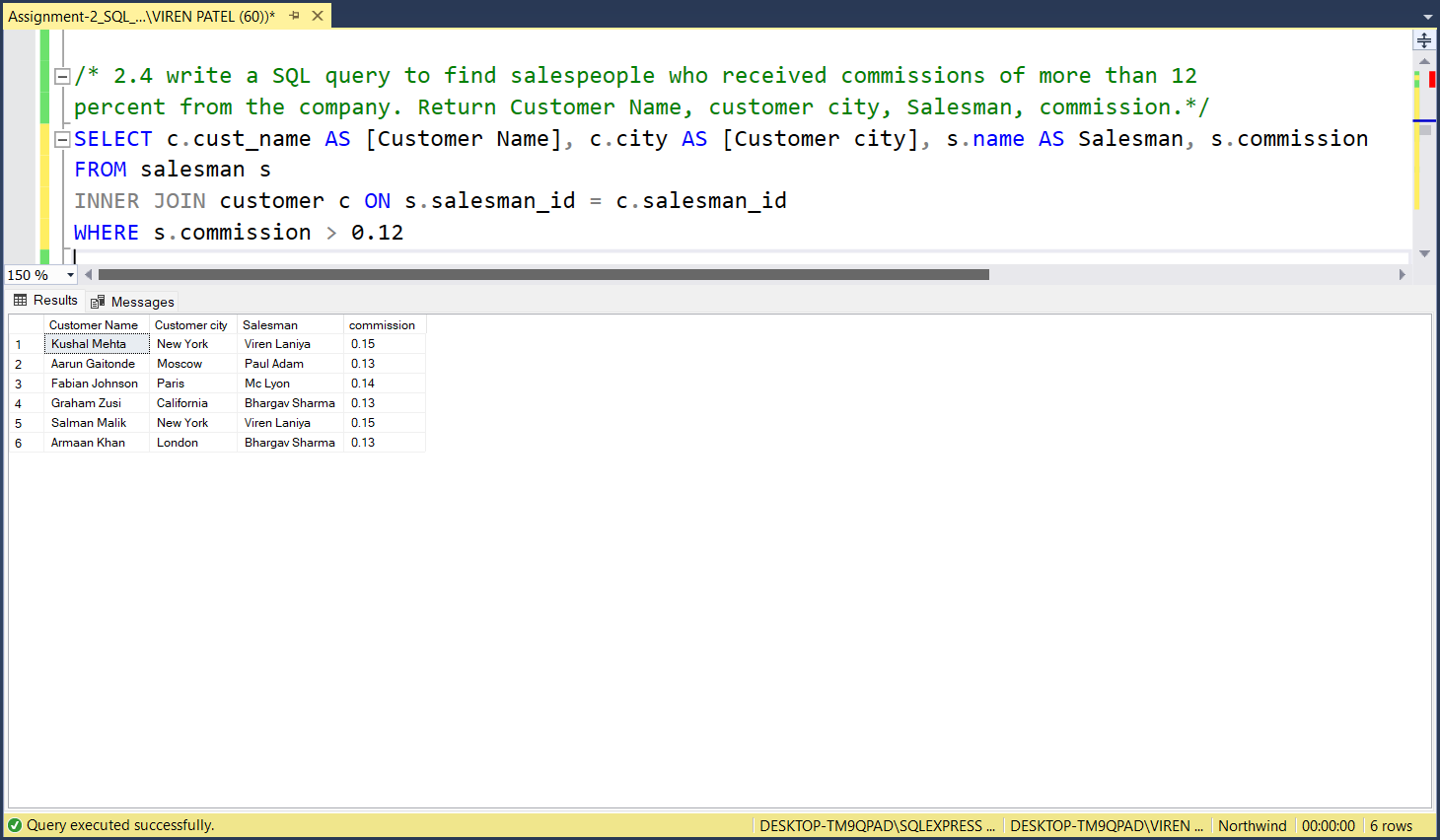
**Query:**

SELECT c.cust\_name AS [Customer Name], c.city AS [Customer city], s.name AS Salesman, s.commission

FROM salesman s

INNER JOIN customer c ON s.salesman\_id = c.salesman\_id

WHERE s.commission > 0.12



1. **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**

**Query:**

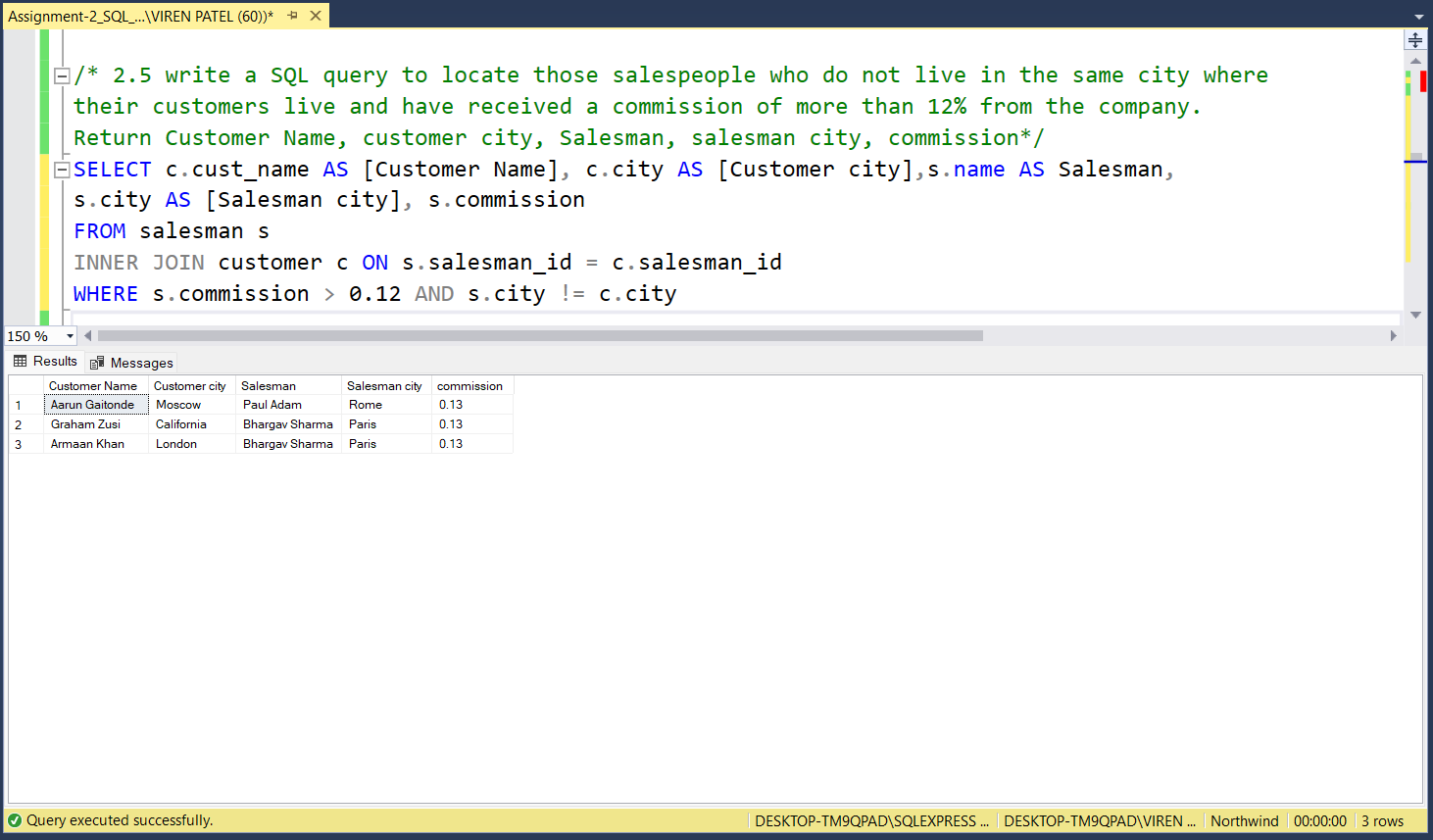
SELECT c.cust\_name AS [Customer Name], c.city AS [Customer city],s.name AS Salesman,

s.city AS [Salesman city], s.commission

FROM salesman s

INNER JOIN customer c ON s.salesman\_id = c.salesman\_id

WHERE s.commission > 0.12 AND s.city != c.city



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

**Query:**

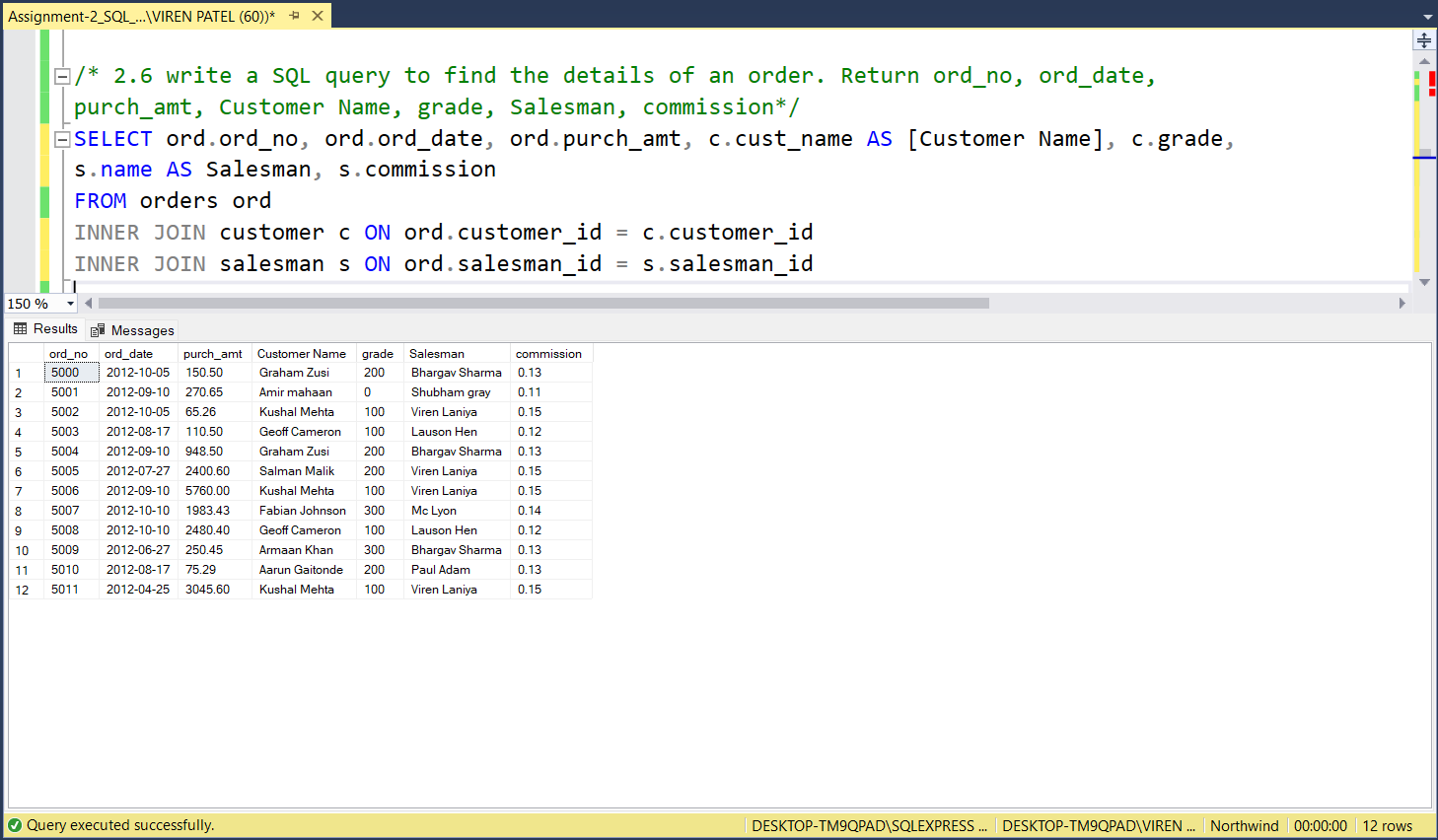
SELECT ord.ord\_no, ord.ord\_date, ord.purch\_amt, c.cust\_name AS [Customer Name], c.grade,

s.name AS Salesman, s.commission

FROM orders ord

INNER JOIN customer c ON ord.customer\_id = c.customer\_id

INNER JOIN salesman s ON ord.salesman\_id = s.salesman\_id



1. **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.**

**Query:**

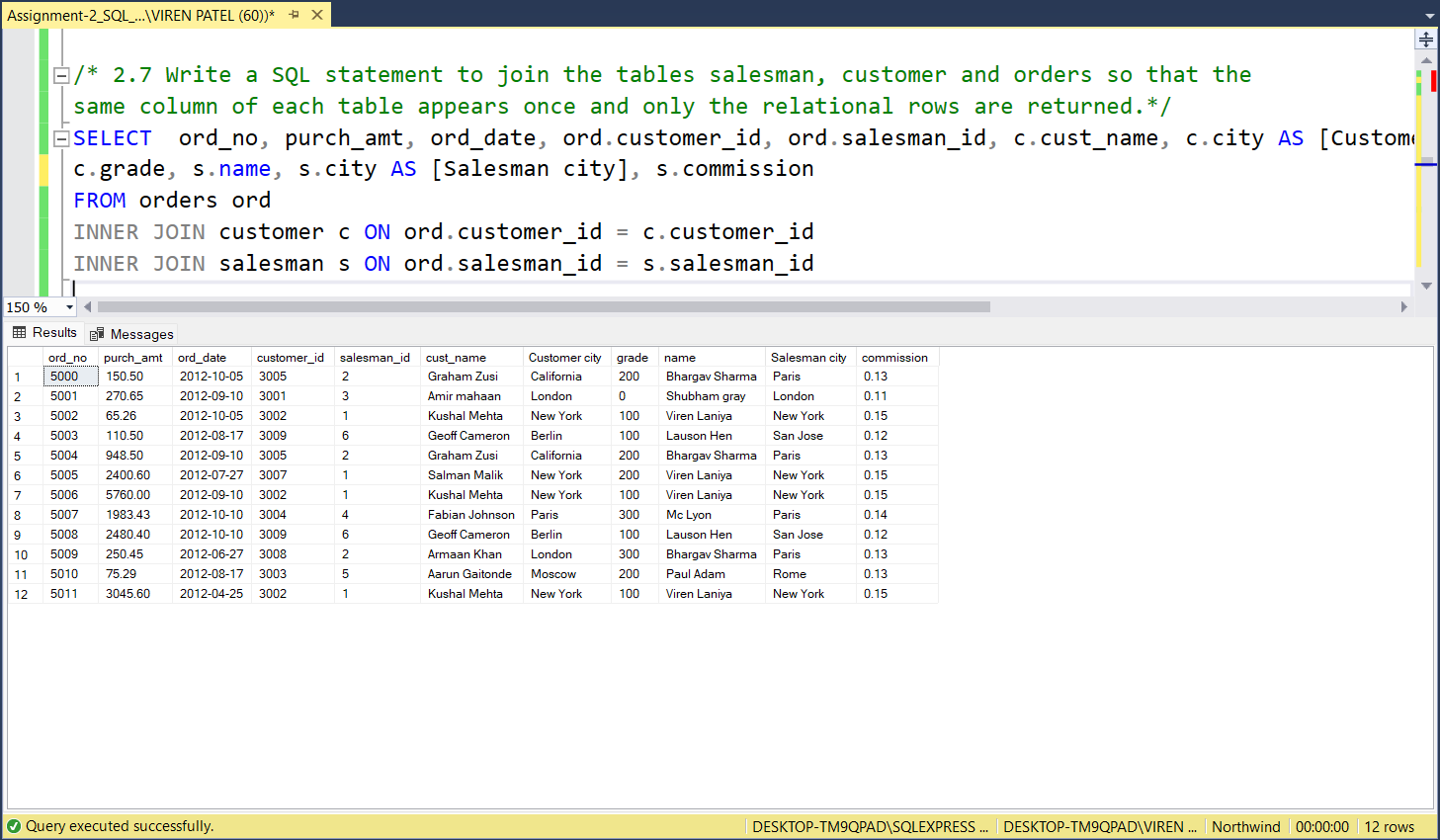
SELECT ord\_no, purch\_amt, ord\_date, ord.customer\_id, ord.salesman\_id, c.cust\_name, c.city AS [Customer city],

c.grade, s.name, s.city AS [Salesman city], s.commission

FROM orders ord

INNER JOIN customer c ON ord.customer\_id = c.customer\_id

INNER JOIN salesman s ON ord.salesman\_id = s.salesman\_id



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

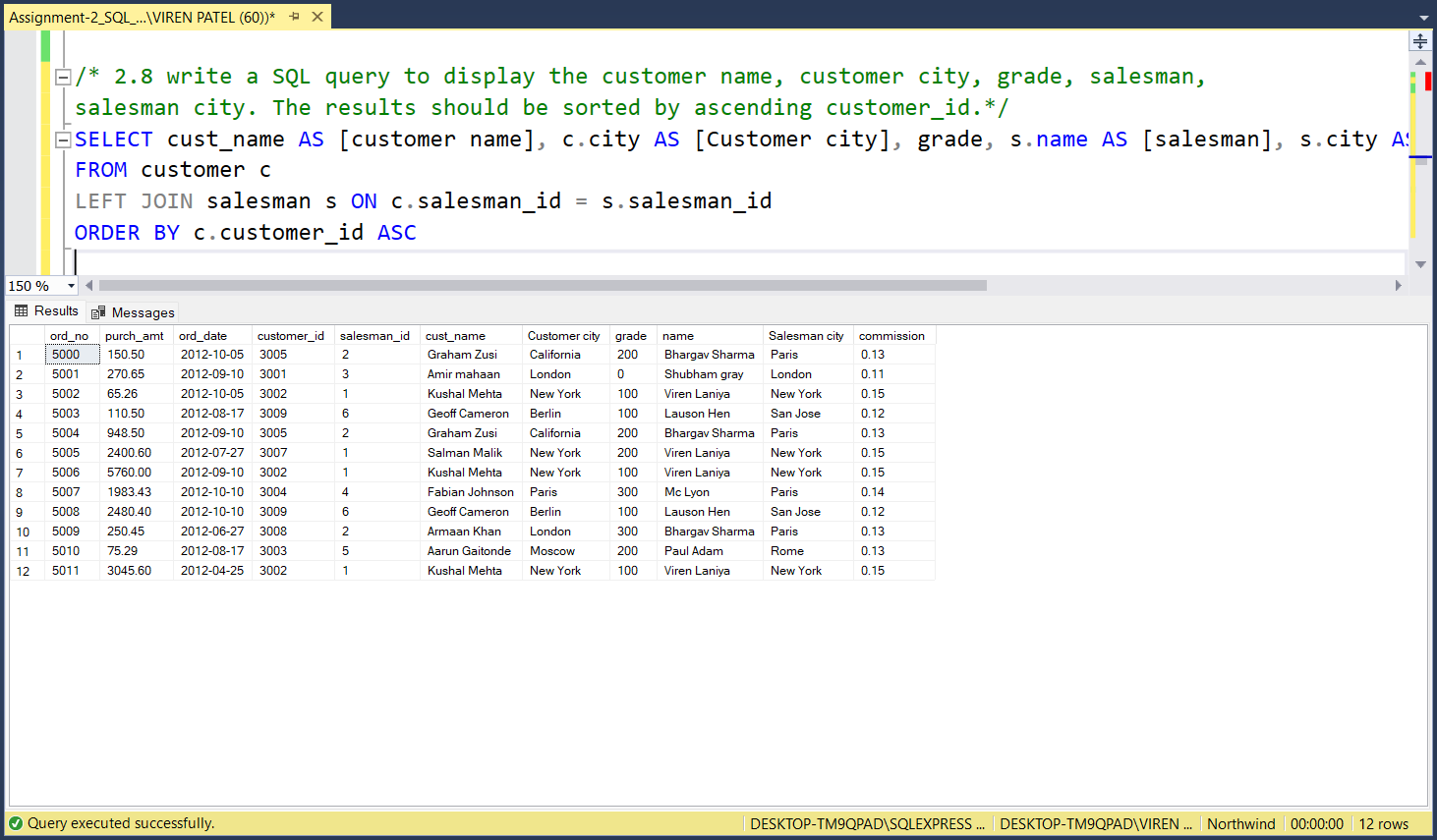
**Query:**

SELECT cust\_name AS [customer name], c.city AS [Customer city], grade, s.name AS [salesman], s.city AS [Salesman city]

FROM customer c

LEFT JOIN salesman s ON c.salesman\_id = s.salesman\_id

ORDER BY c.customer\_id ASC



1. **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.**

**Query:**

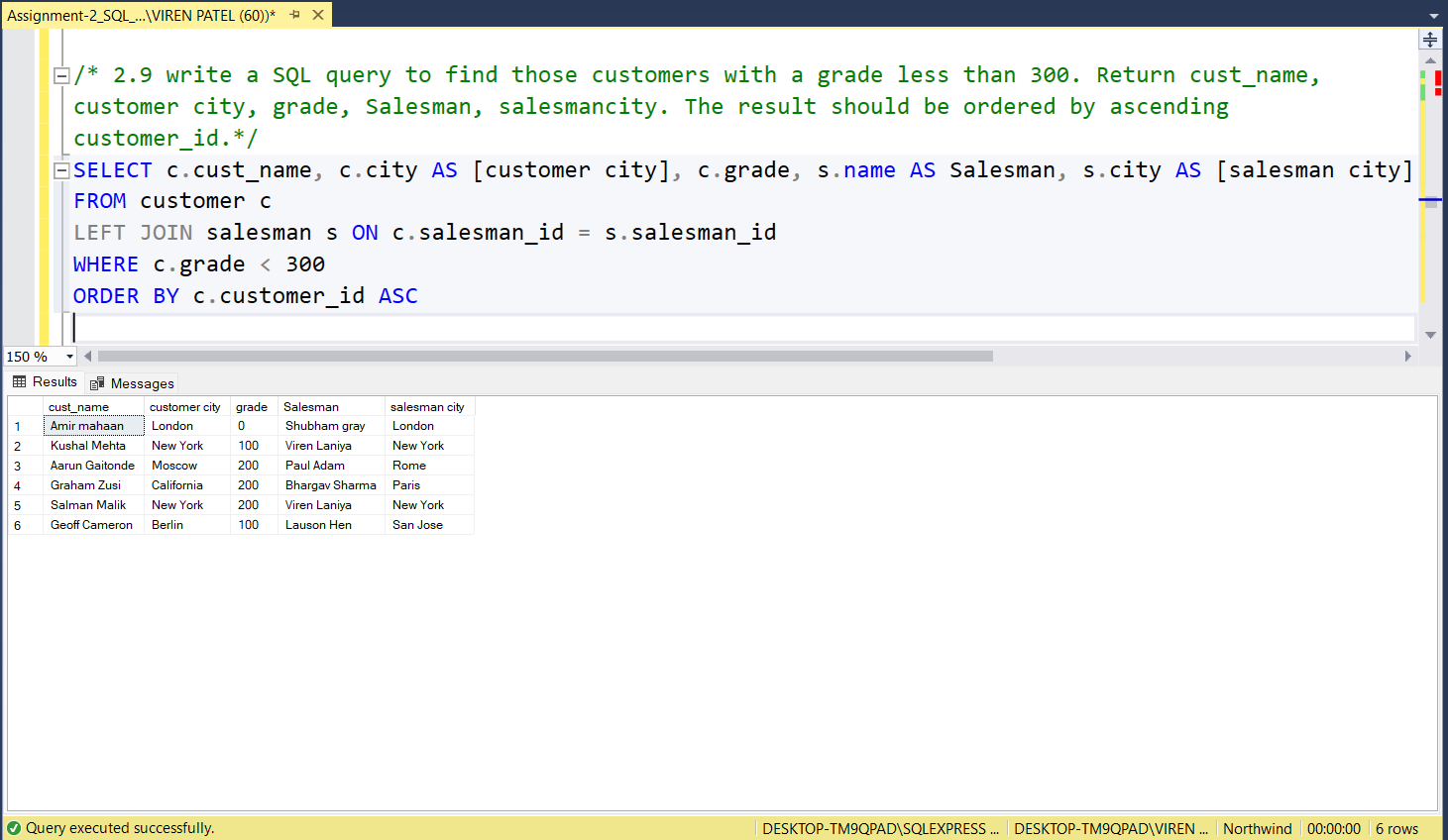
SELECT c.cust\_name, c.city AS [customer city], c.grade, s.name AS Salesman, s.city AS [salesman city]

FROM customer c

LEFT JOIN salesman s ON c.salesman\_id = s.salesman\_id

WHERE c.grade < 300

ORDER BY c.customer\_id ASC



1. **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**

**Query:**

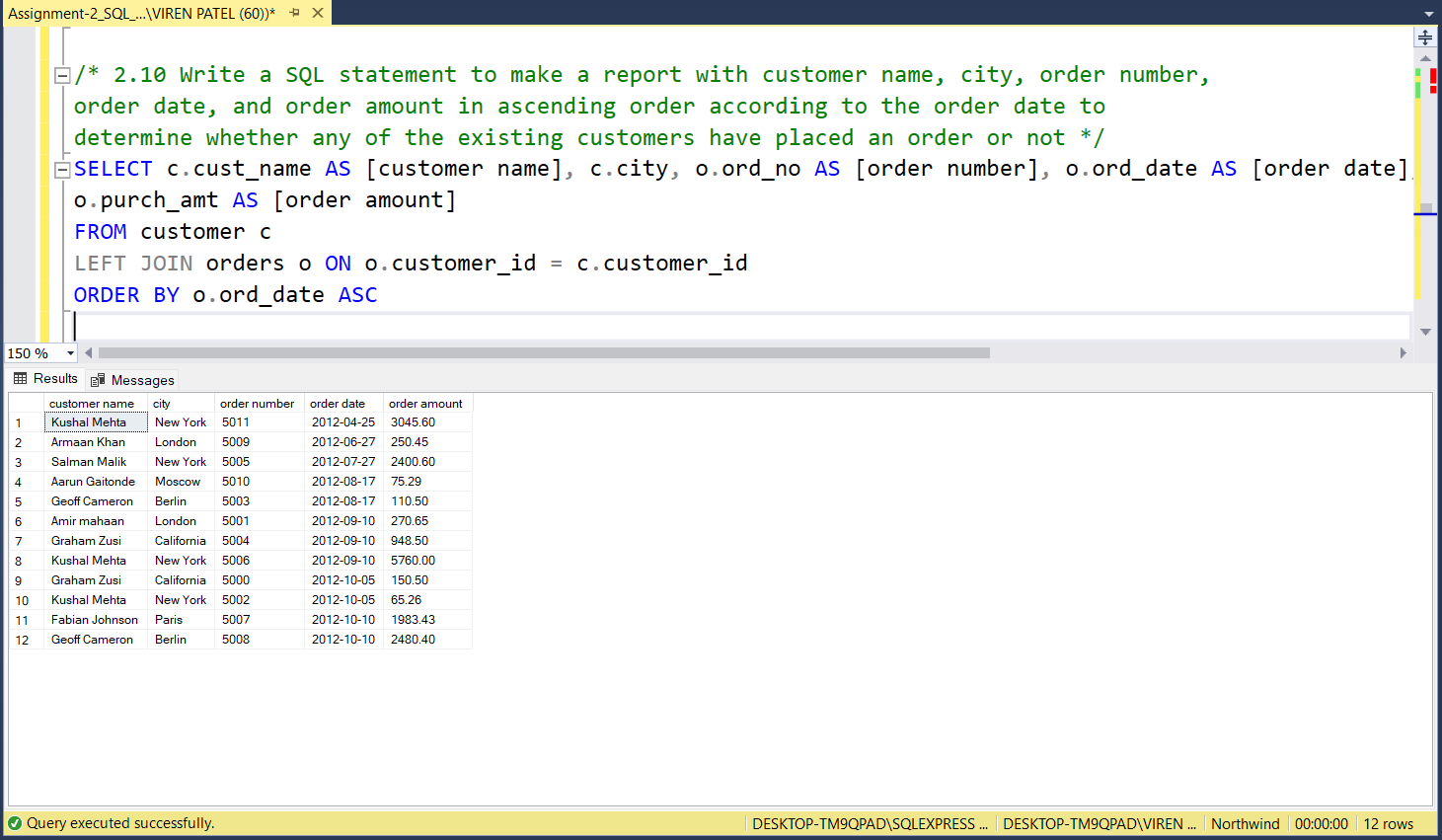
SELECT c.cust\_name AS [customer name], c.city, o.ord\_no AS [order number], o.ord\_date AS [order date],

o.purch\_amt AS [order amount]

FROM customer c

LEFT JOIN orders o ON o.customer\_id = c.customer\_id

ORDER BY o.ord\_date ASC



1. **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**

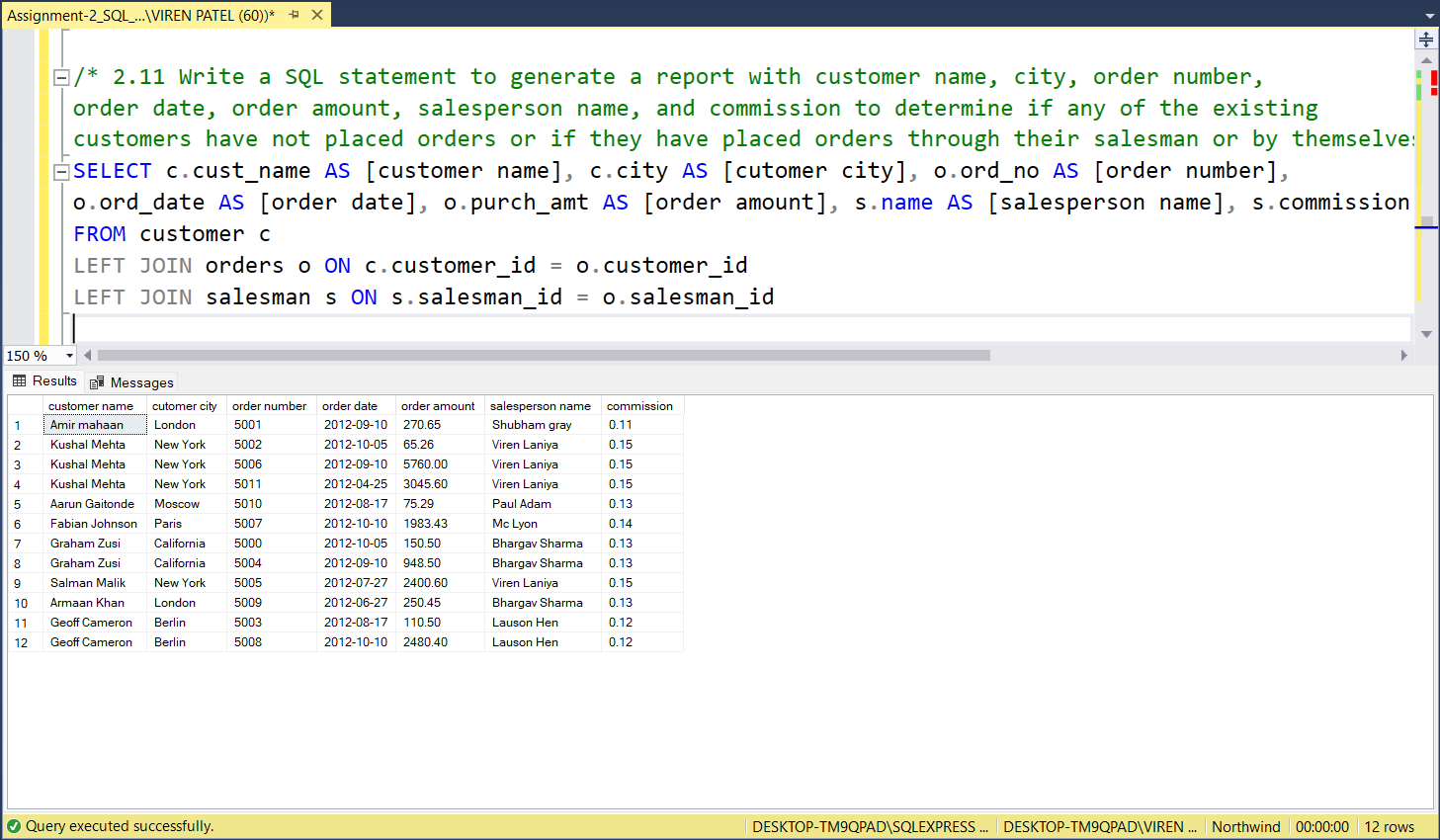
**Query:**

SELECT c.cust\_name AS [customer name], c.city AS [cutomer city], o.ord\_no AS [order number], o.ord\_date AS [order date], o.purch\_amt AS [order amount], s.name AS [salesperson name], s.commission

FROM customer c

LEFT JOIN orders o ON c.customer\_id = o.customer\_id

LEFT JOIN salesman s ON s.salesman\_id = o.salesman\_id

****

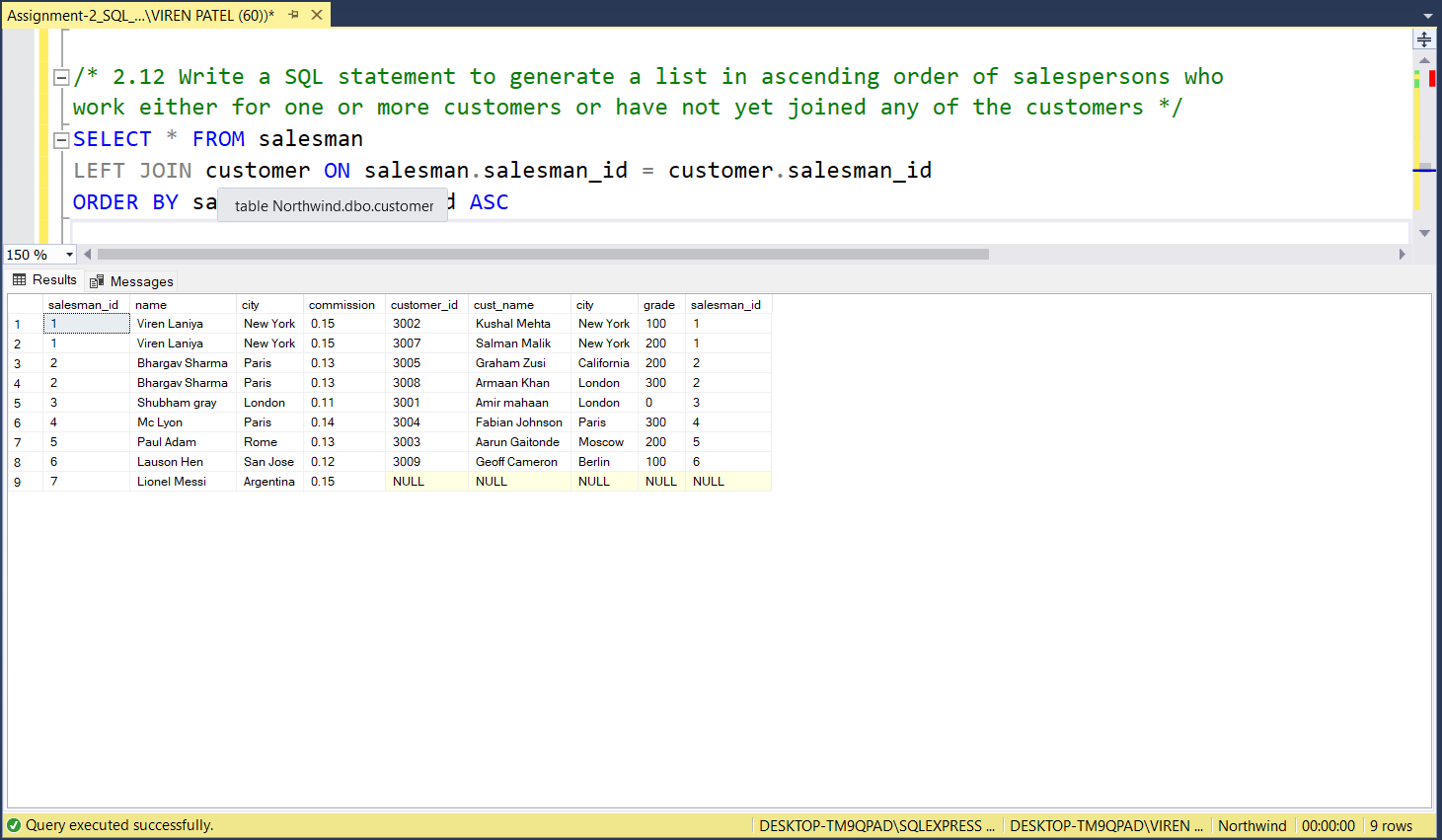
1. **Write a SQL statement to generate a list in ascending order of salespersons who work either for one or more customers or have not yet joined any of the customers**

**Query:**

SELECT \* FROM salesman

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

ORDER BY salesman.salesman\_id ASC



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

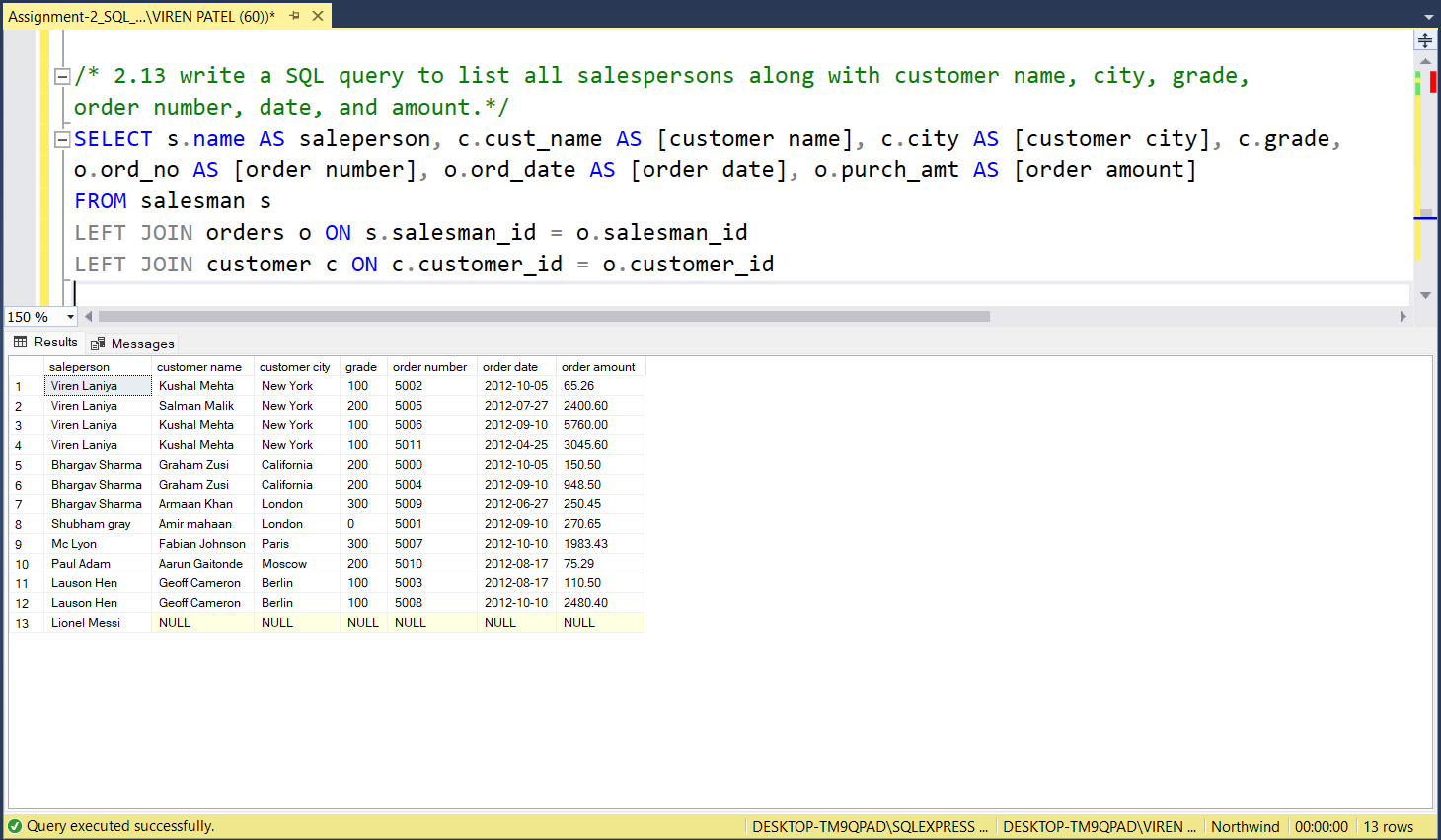
**Query:**

SELECT s.name AS saleperson, c.cust\_name AS [customer name], c.city AS [customer city], c.grade, o.ord\_no AS [order number], o.ord\_date AS [order date], o.purch\_amt AS [order amount]

FROM salesman s

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

LEFT JOIN customer c ON c.customer\_id = o.customer\_id



1. **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.**

**Query:**

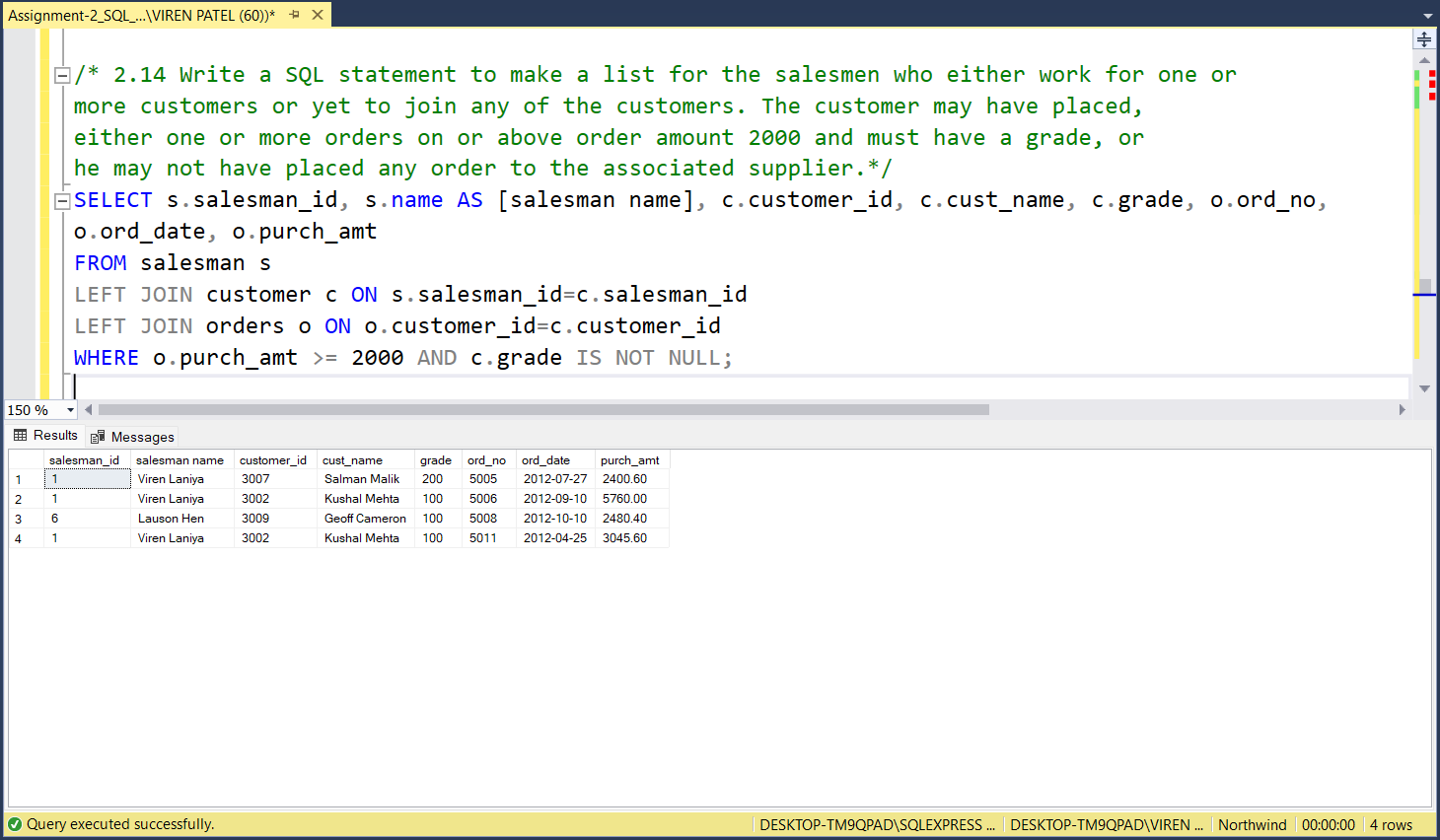
SELECT s.salesman\_id, s.name AS [salesman name], c.customer\_id, c.cust\_name, c.grade, o.ord\_no, o.ord\_date, o.purch\_amt

FROM salesman s

LEFT JOIN customer c ON s.salesman\_id=c.salesman\_id

LEFT JOIN orders o ON o.customer\_id=c.customer\_id

WHERE o.purch\_amt >= 2000 AND c.grade IS NOT NULL;



1. **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.**

**Query:**

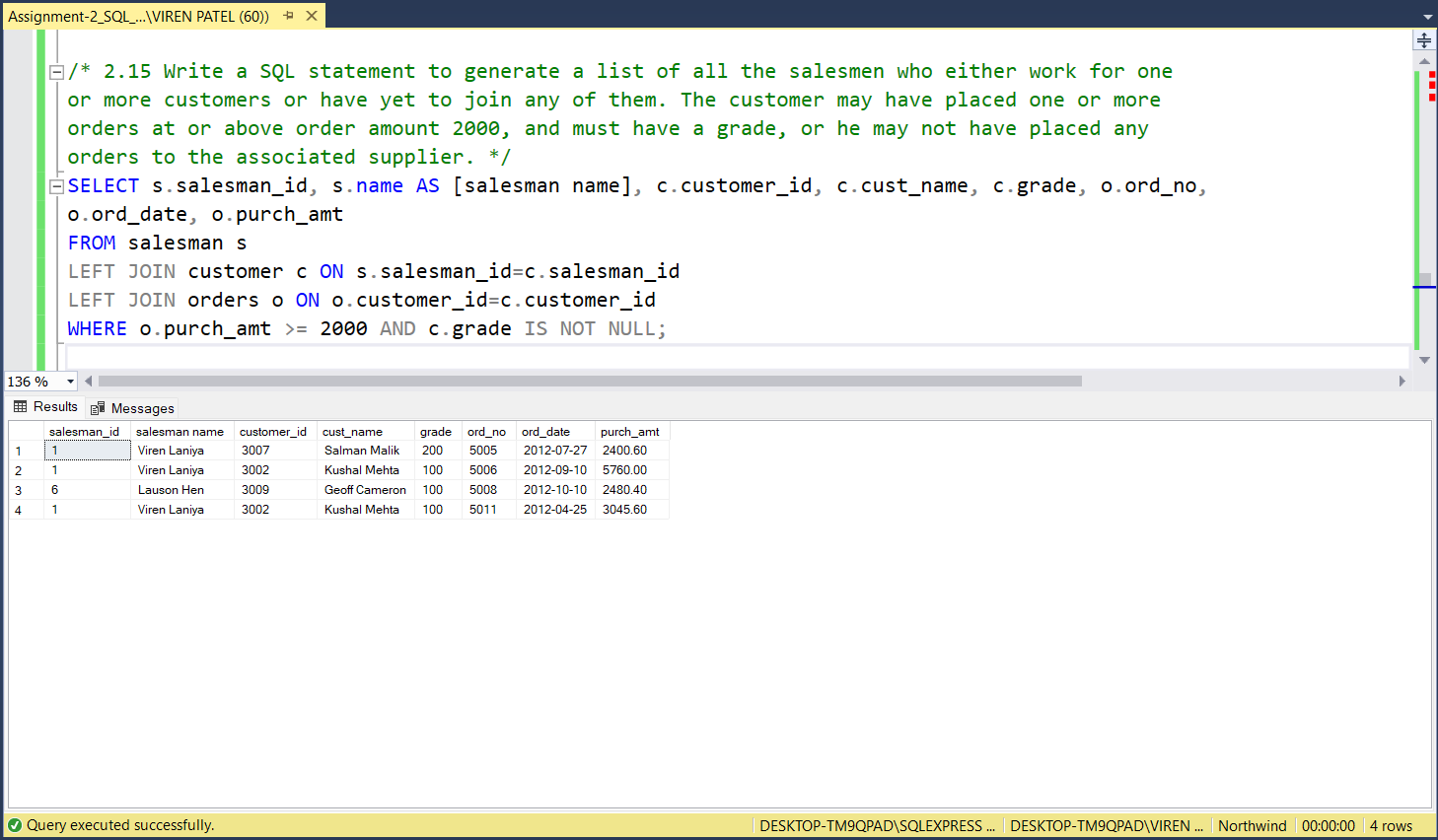
SELECT s.salesman\_id, s.name AS [salesman name], c.customer\_id, c.cust\_name, c.grade, o.ord\_no, o.ord\_date, o.purch\_amt

FROM salesman s

LEFT JOIN customer c ON s.salesman\_id=c.salesman\_id

LEFT JOIN orders o ON o.customer\_id=c.customer\_id

WHERE o.purch\_amt >= 2000 AND c.grade IS NOT NULL;



1. **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.**

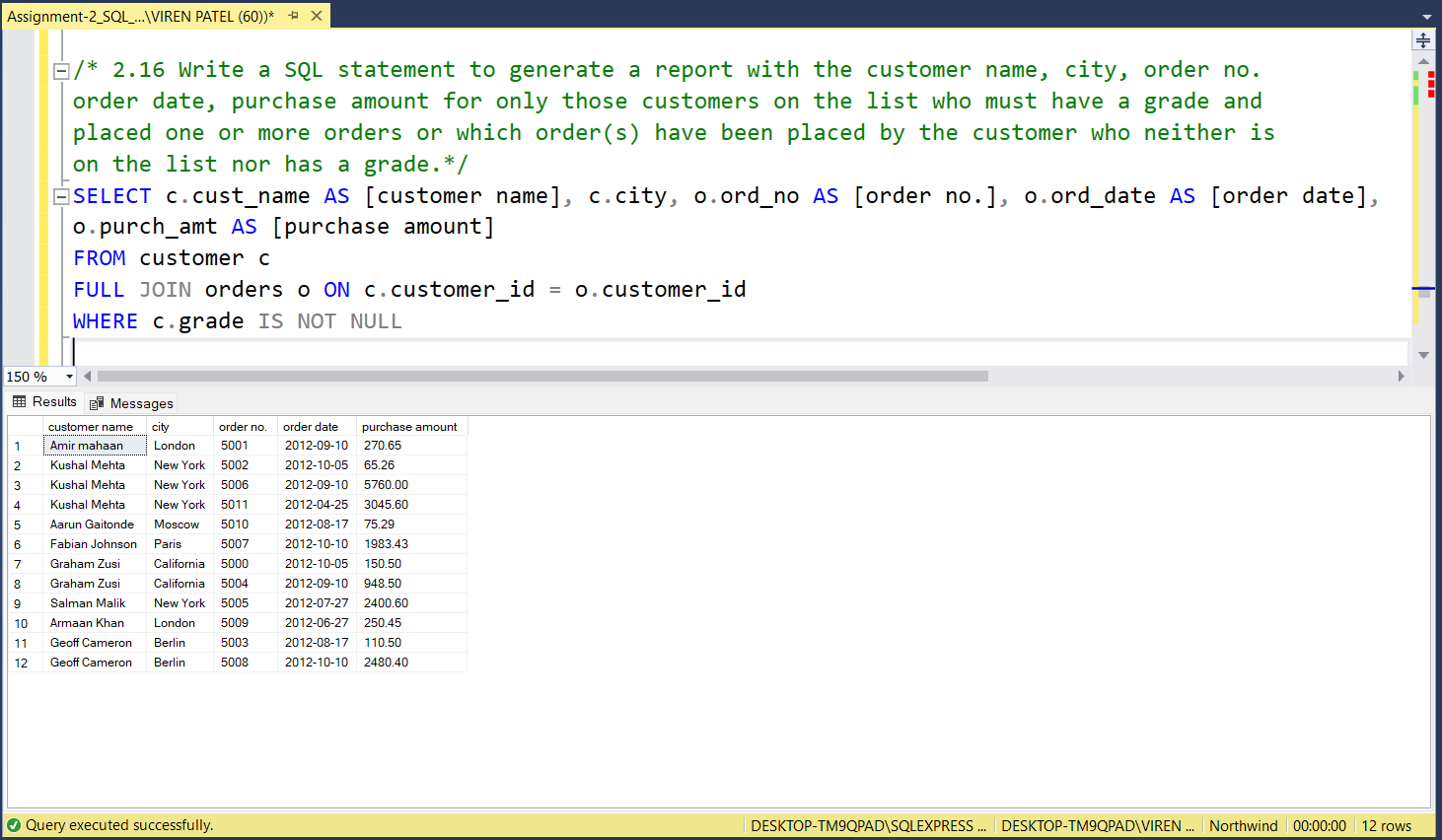
**Query:**

SELECT c.cust\_name AS [customer name], c.city, o.ord\_no AS [order no.], o.ord\_date AS [order date], o.purch\_amt AS [purchase amount]

FROM customer c

FULL JOIN orders o ON c.customer\_id = o.customer\_id

WHERE c.grade IS NOT NULL

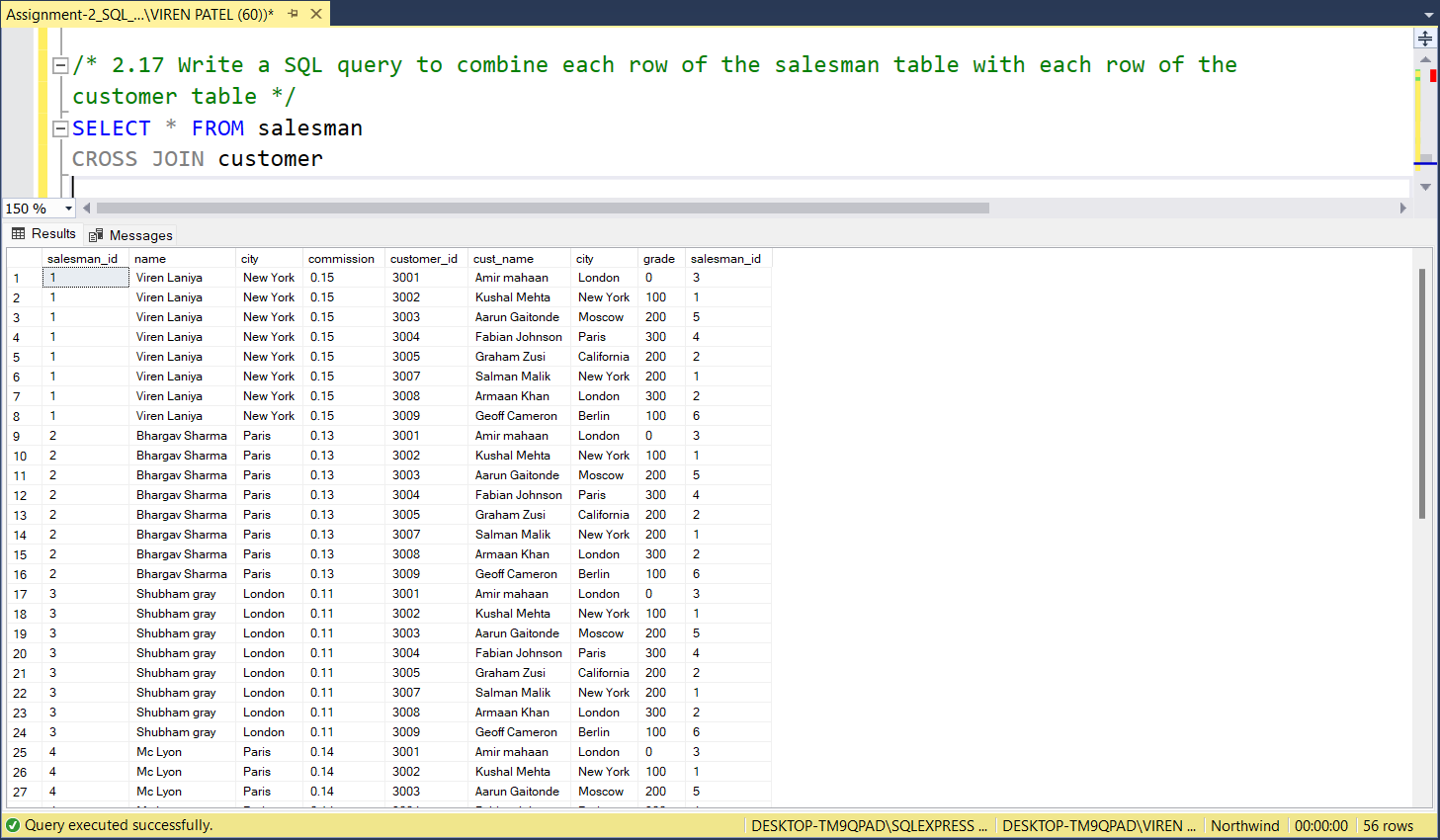


1. **Write a SQL query to combine each row of the salesman table with each row of the customer table**

**Query:**

SELECT \* FROM salesman

CROSS JOIN customer



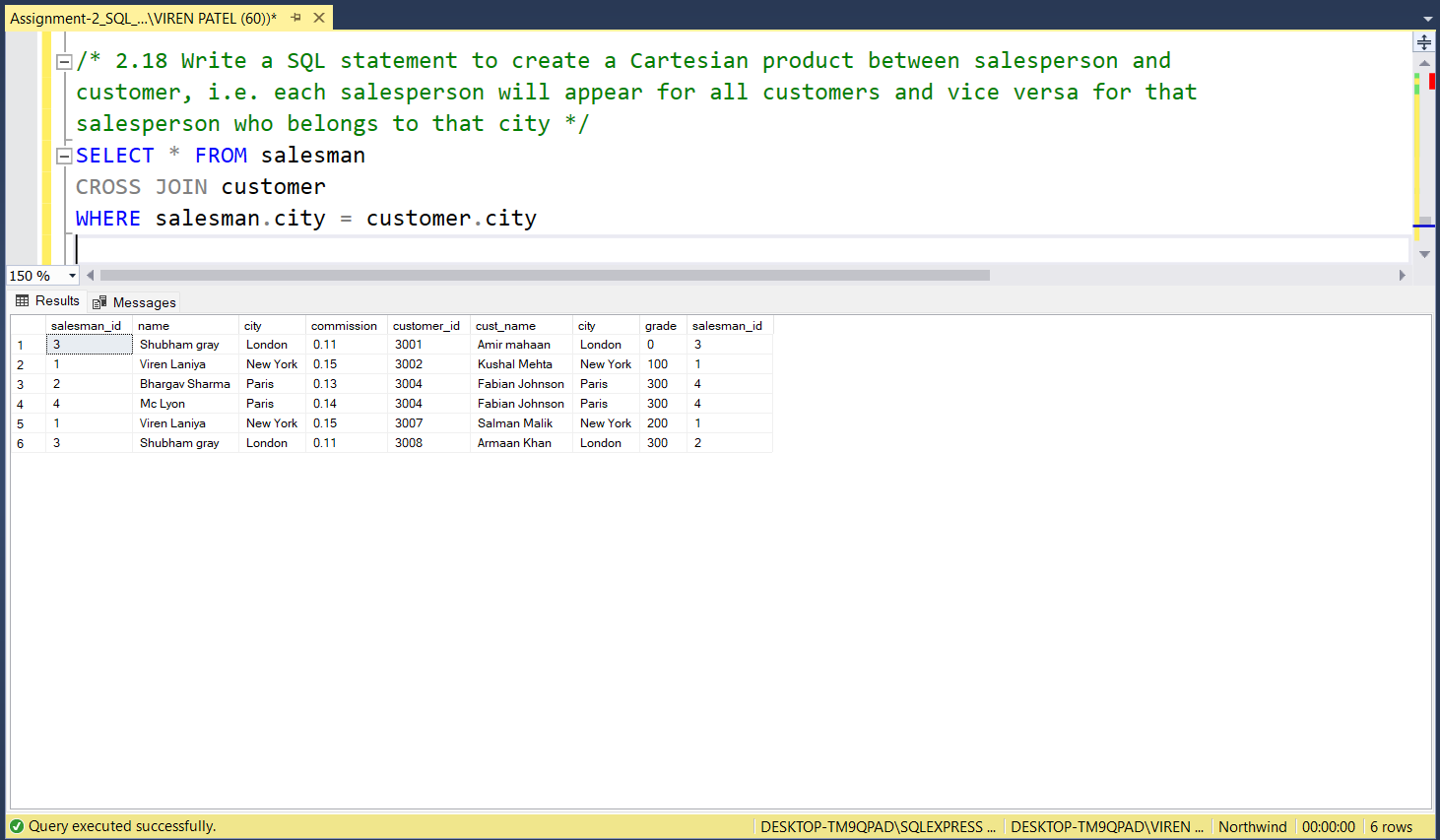
1. **Write a SQL statement to create a Cartesian product between salesperson and customer, i.e. each salesperson will appear for all customers and vice versa for that salesperson who belongs to that city**

**Query:**

SELECT \* FROM salesman

CROSS JOIN customer

WHERE salesman.city = customer.city



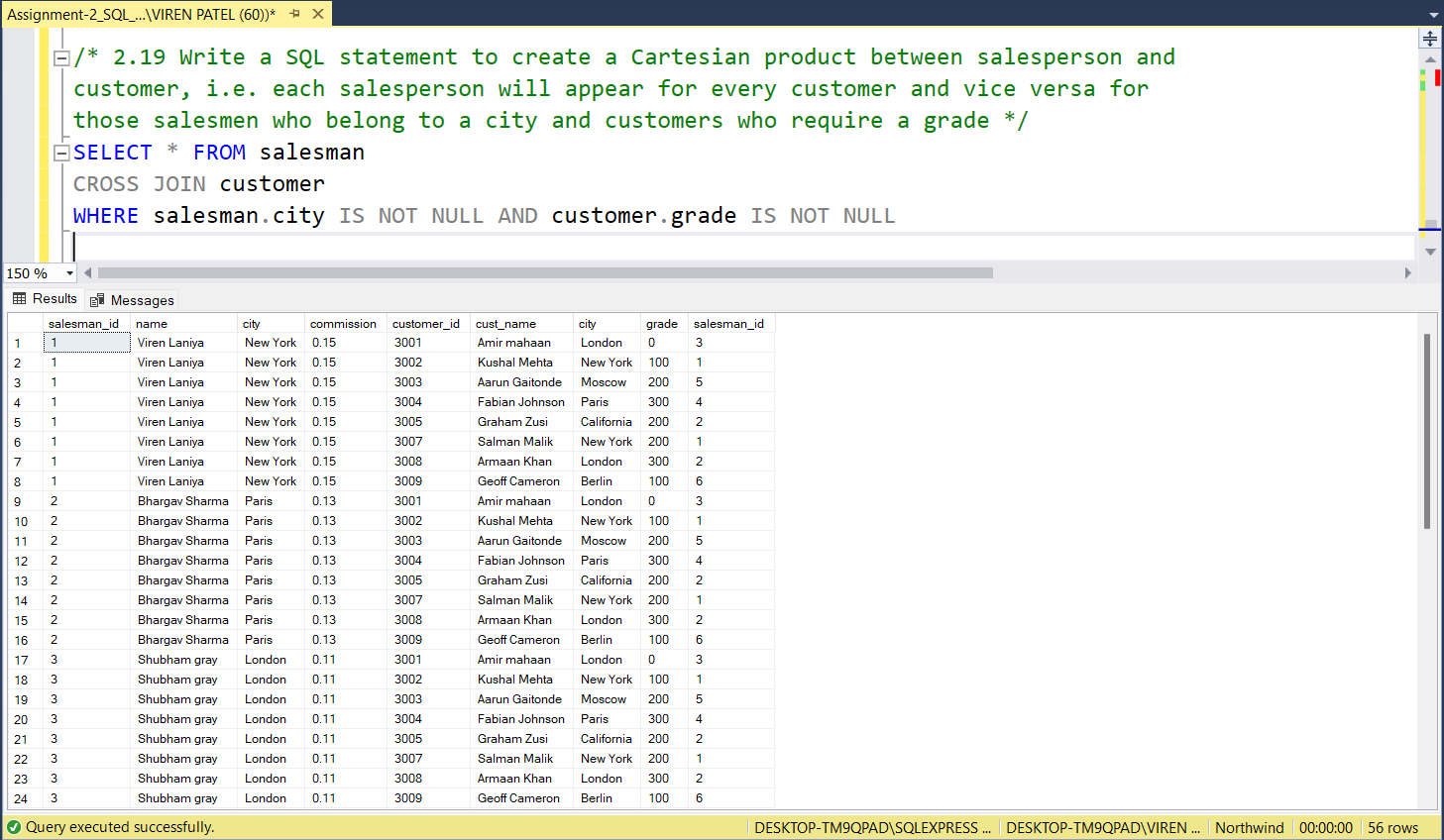
1. **Write a SQL statement to create a Cartesian product between salesperson and customer, i.e. each salesperson will appear for every customer and vice versa for those salesmen who belong to a city and customers who require a grade**

**Query:**

SELECT \* FROM salesman

CROSS JOIN customer

WHERE salesman.city IS NOT NULL AND customer.grade IS NOT NULL



1. **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**

**Query:**

SELECT \* FROM salesman

CROSS JOIN customer

WHERE (salesman.city IS NOT NULL) AND (salesman.city != customer.city) AND (customer.grade IS NOT NULL)

