**Assignment-2**

-- Create Tables

CREATE TABLE Salesman(

salesman\_id int primary key NOT NULL,

name varchar(50) NOT NULL,

city varchar(50) NOT NULL,

commission decimal(18,2) NOT NULL

);



CREATE TABLE Customer(

customer\_id int PRIMARY KEY NOT NULL,

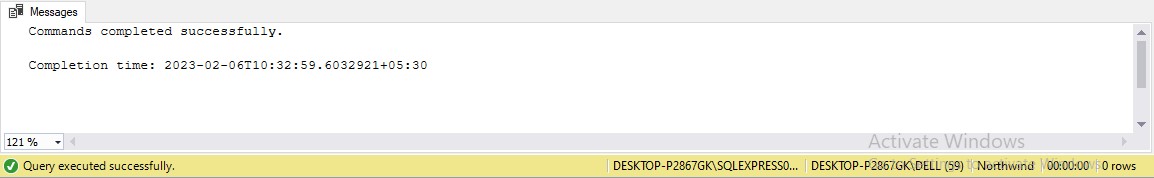
cust\_name varchar(50) NOT NULL,

city varchar(50) NOT NULL,

grade int,

salesman\_id int FOREIGN KEY REFERENCES Salesman(salesman\_id) NOT NULL

);



CREATE TABLE Orders(

ord\_no int PRIMARY KEY NOT NULL,

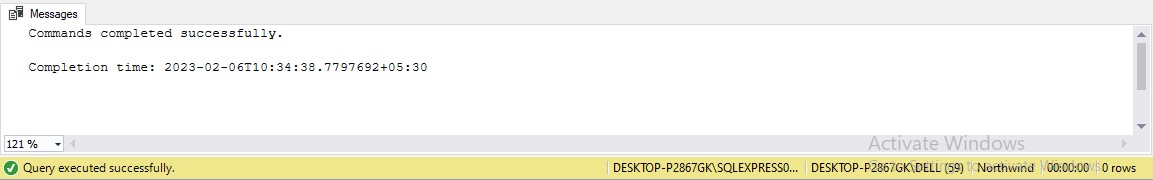
purch\_amt decimal(18,2) NOT NULL,

ord\_date date NOT NULL,

customer\_id int FOREIGN KEY REFERENCES Customer(customer\_id) NOT NULL,

salesman\_id int FOREIGN KEY REFERENCES Salesman(salesman\_id) NOT NULL

);



-- Insert Data into Table

INSERT INTO Salesman VALUES(101,'Meet Patel','Ahmedabad',0.15),

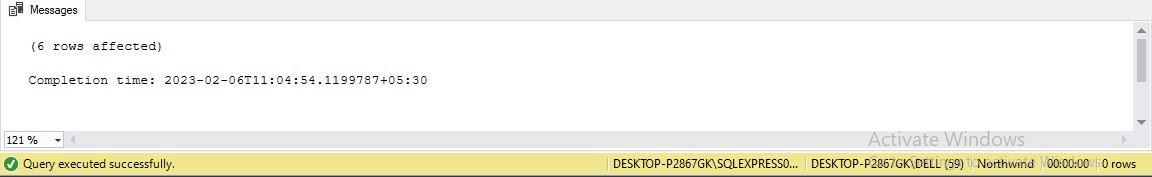
(102,'Suchi Mehta','Baroda',0.13),

(105,'Mayur Shah','Rajkot',0.11),

(106,'Tirth Patel','Surat',0.12),

(107,'Foram Sharma','Junagadh',0.14),

(103,'Ansh Desai','Bharuch',0.13);



-- Insert Data into Table

INSERT INTO Customer VALUES(302,'Bhumi Shah','Ahmedabad',100,101),

(307,'Aryan Patel','Ahmedabad',200,101),

(305,'Aarvi Birla','Surendranagar',200,102),

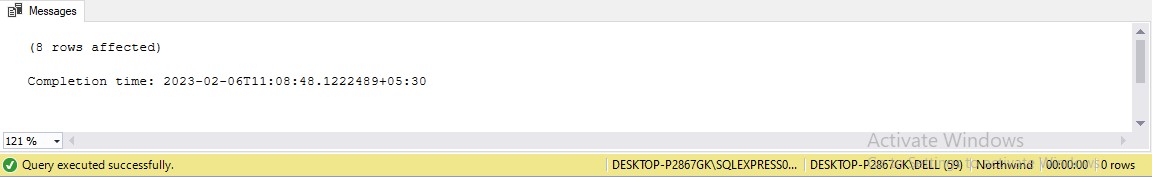
(308,'Ruhi Mehta','Rajkot',300,102),

(304,'Abhi Sharma','Surat',300,106),

(309,'Ved Patel','Bhavnagar',100,103),

(303,'Mansi Desai','Anand',200,107),

(301,'Bhautik Birla ','Rajkot',NULL,105);



-- Insert Data into Table

INSERT INTO Orders VALUES(5001,150.5,'2012-10-5',305,102),

(5009,270.65,'2012-09-10',301,105),

(5002,65.26,'2012-10-05',302,101),

(5004,110.5,'2012-08-17',308,103),

(5007,948.5,'2012-09-10',305,102),

(5005,2400.6,'2012-07-27',307,101),

(5008,5760,'2012-09-10',302,101),

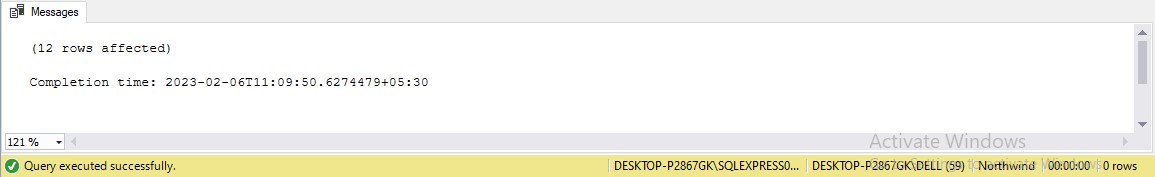
(5010,1983.43,'2012-10-10',304,106),

(5003,2480.4,'2012-10-10',308,103),

(5012,250.45,'2012-06-27',307,102),

(5011,75.29,'2012-08-17',303,106),

(5013,3045.6,'2012-04-25',302,101);



-- 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, c.city

FROM Salesman s INNER JOIN Customer c

ON s.city = c.city



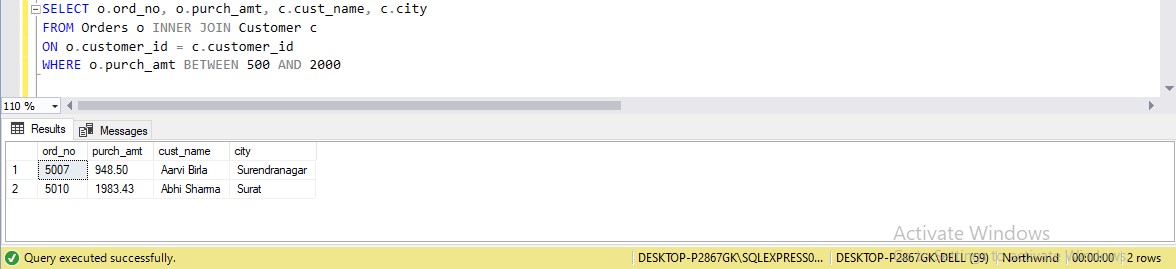
-- 2. write a SQL query to find those orders where the order amount exists between 500 and 2000. Return ord\_no, purch\_amt, cust\_name, city.

SELECT o.ord\_no, o.purch\_amt, c.cust\_name, c.city

FROM Orders o INNER JOIN Customer c

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

WHERE o.purch\_amt BETWEEN 500 AND 2000

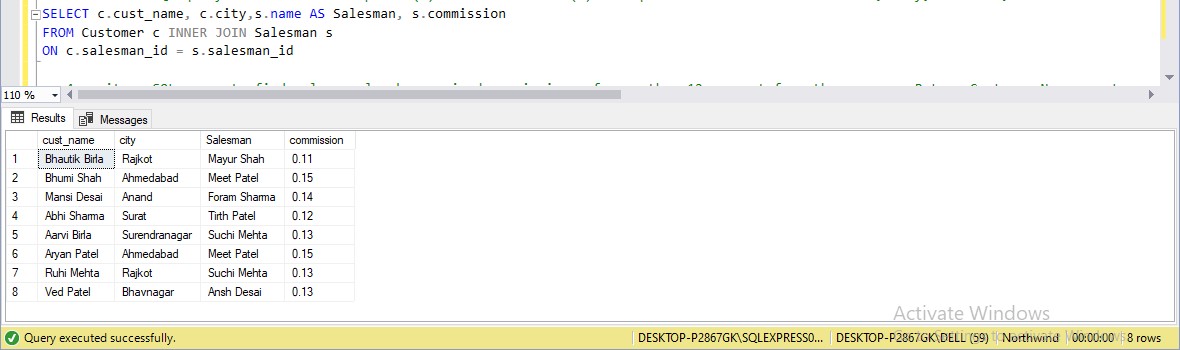


-- 3. write a SQL query to find the salesperson(s) and the customer(s) he represents. Return Customer Name, city, Salesman, commission.

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

FROM Customer c INNER JOIN Salesman s

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



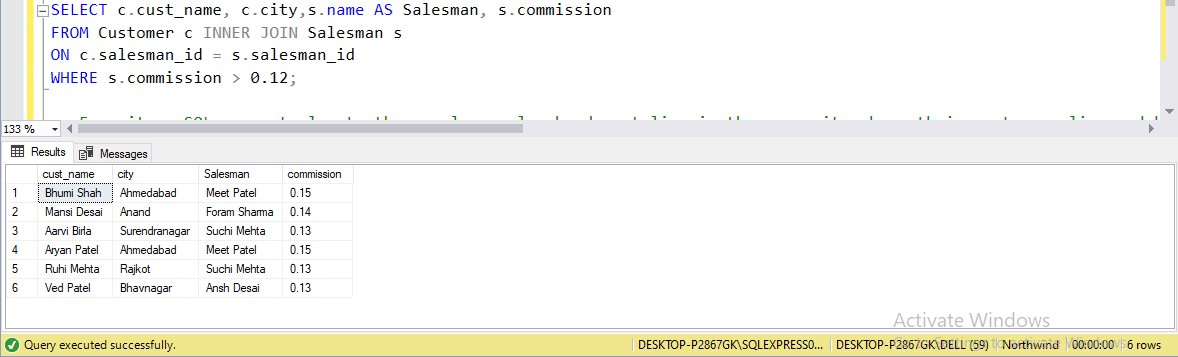
-- 4. write a SQL query to find salespeople who received commissions of more than 12 percent from the company. Return Customer Name, customer city, Salesman, commission.

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

FROM Customer c INNER JOIN Salesman s

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

WHERE s.commission > 0.12;



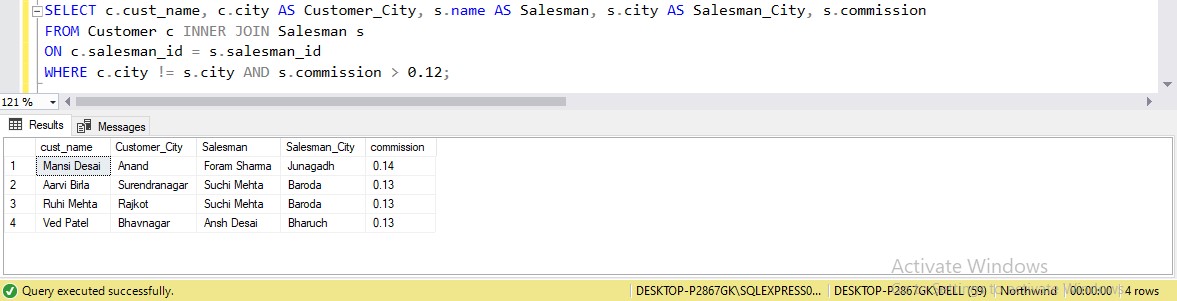
-- 5. write a SQL query to locate those salespeople who do not live in the same city where their customers live and have received a commission of more than 12% from the Company. Return Customer Name, customer city, Salesman, salesman city, commission.

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

FROM Customer c INNER JOIN Salesman s

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

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



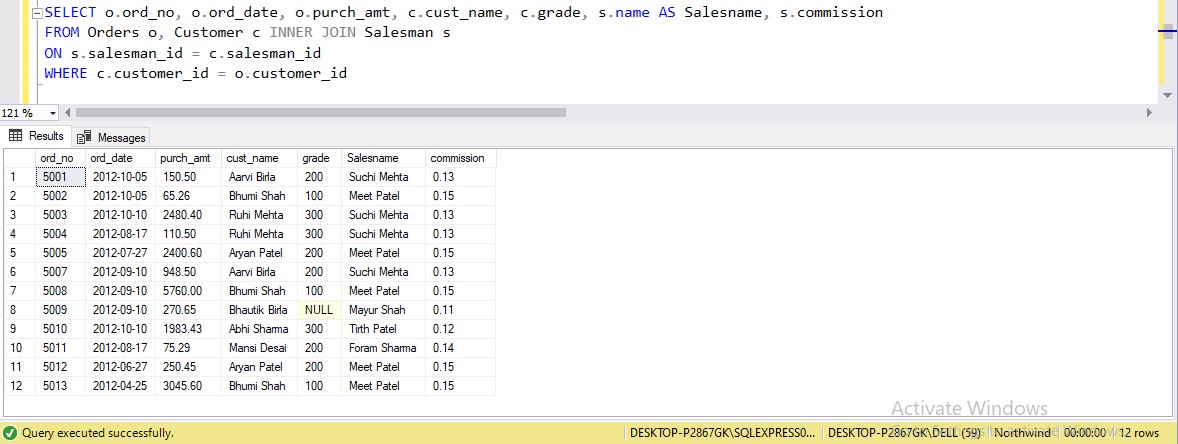
-- 6. write a SQL query to find the details of an order. Return ord\_no, ord\_date, purch\_amt, Customer Name, grade, Salesman, commission.

SELECT o.ord\_no, o.ord\_date, o.purch\_amt, c.cust\_name, c.grade, s.name AS Salesname, s.commission

FROM Orders o, Customer c INNER JOIN Salesman s

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

WHERE c.customer\_id = o.customer\_id



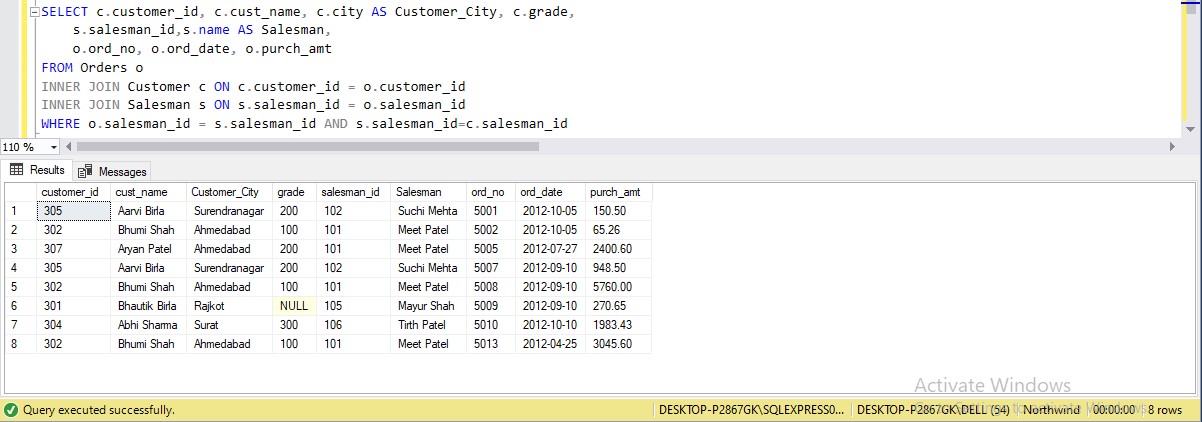
-- 7. Write a SQL statement to join the tables salesman, customer and orders so that the same column of each table appears once and only the relational rows are returned.

SELECT \* 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

WHERE o.salesman\_id = s.salesman\_id AND s.salesman\_id=c.salesman\_id



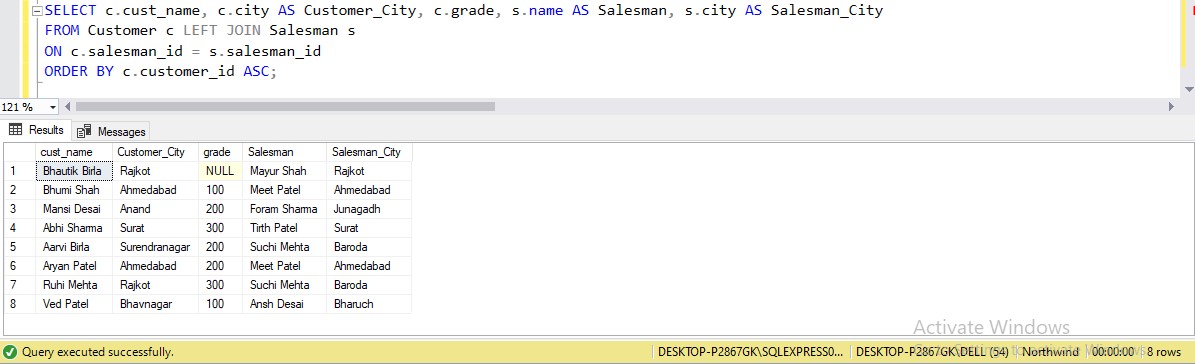
-- 8. write a SQL query to display the customer name, customer city, grade, salesman, salesman city. The results should be sorted by ascending customer\_id.

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

ORDER BY c.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.

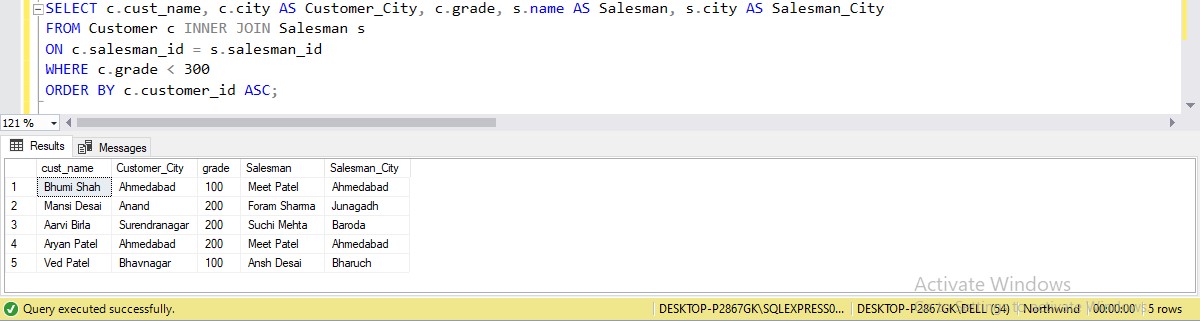
SELECT c.cust\_name, c.city AS Customer\_City, c.grade, s.name AS Salesman, s.city AS Salesman\_City

FROM Customer c INNER JOIN Salesman s

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

WHERE c.grade < 300

ORDER BY c.customer\_id ASC;



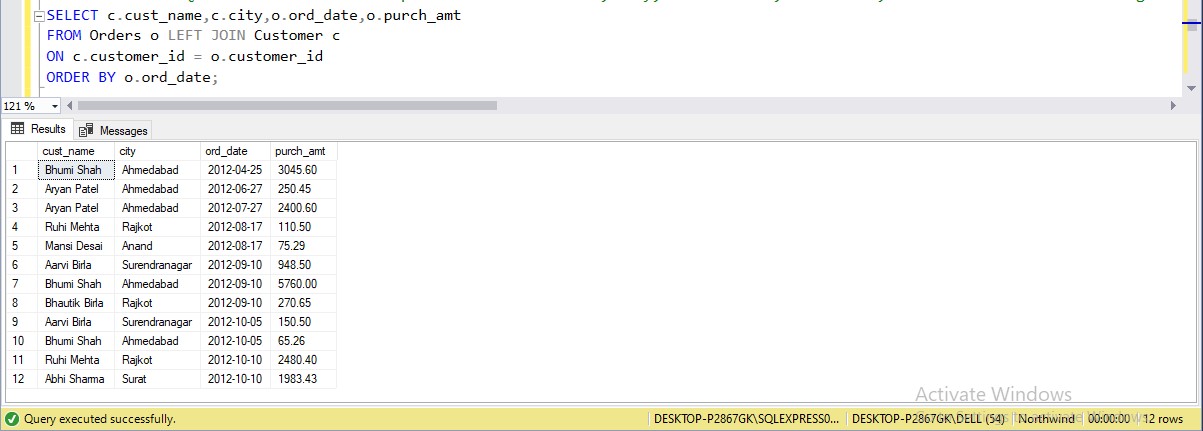
-- 10. Write a SQL statement to make a report with customer name, city, order number, order date, and order amount in ascending order according to the order date to determine whether any of the existing customers have placed an order or not.

SELECT c.cust\_name,c.city,o.ord\_date,o.purch\_amt

FROM Orders o LEFT JOIN Customer c

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

ORDER BY o.ord\_date;



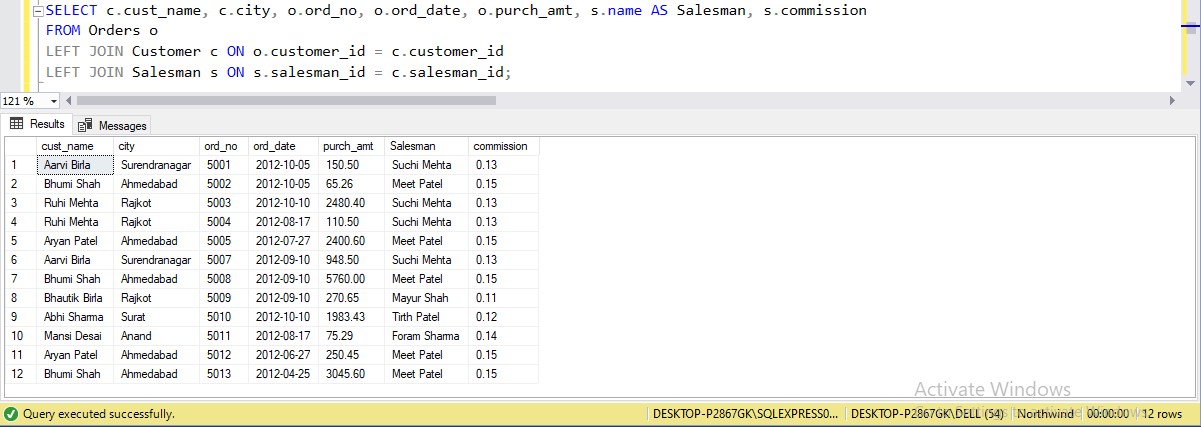
-- 11. Write a SQL statement to generate a report with customer name, city, order number, order date, order amount, salesperson name, and commission to determine if any of the existing customers have not placed orders or if they have placed orders through their salesman or by themselves.

SELECT c.cust\_name, c.city, o.ord\_no, o.ord\_date, o.purch\_amt, s.name AS Salesman, s.commission

FROM Orders o

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

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



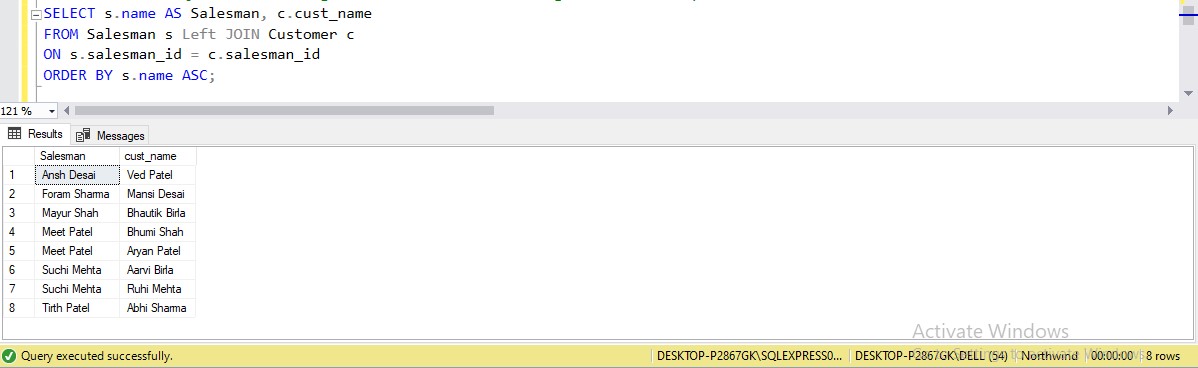
-- 12. Write a SQL statement to generate a list in ascending order of salespersons who work either for one or more customers or have not yet joined any of the customers.

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

FROM Salesman s Left JOIN Customer c

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

ORDER BY s.name ASC;

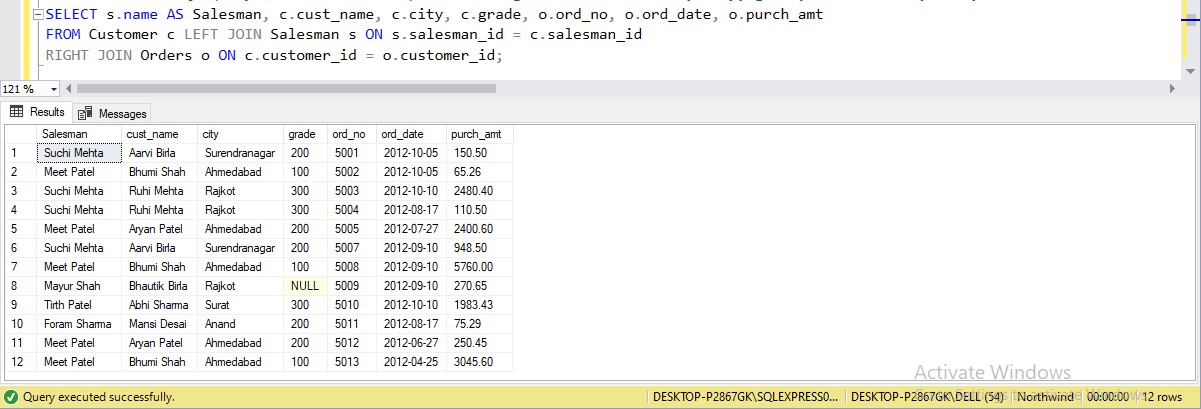


-- 13. write a SQL query to list all salespersons along with customer name, city, grade, order number, date, and amount.

SELECT s.name AS Salesman, c.cust\_name, c.city, c.grade, o.ord\_no, o.ord\_date, o.purch\_amt

FROM Customer c LEFT JOIN Salesman s ON s.salesman\_id = c.salesman\_id

RIGHT JOIN Orders o ON c.customer\_id = o.customer\_id;



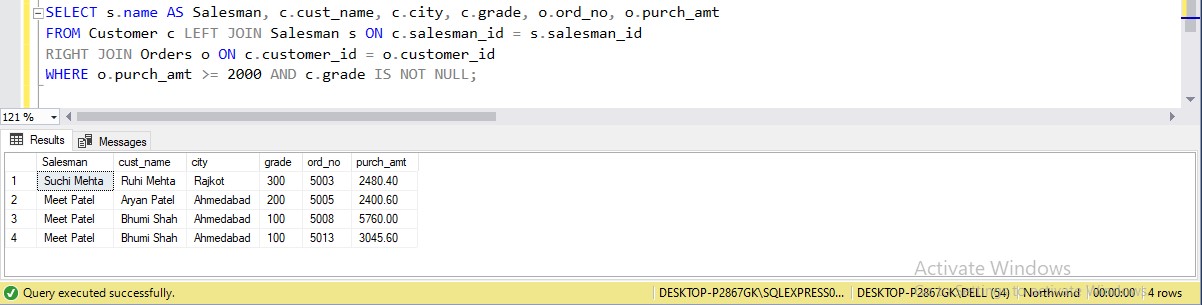
-- 14. Write a SQL statement to make a list for the salesmen who either work for one or more customers or yet to join any of the customers. The customer may have placed, either one or more orders on or above order amount 2000 and must have a grade, or he may not have placed any order to the associated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

SELECT s.name AS Salesman, c.cust\_name, c.city, c.grade, o.ord\_no, o.purch\_amt

FROM Customer c LEFT JOIN Salesman s ON c.salesman\_id = s.salesman\_id

RIGHT JOIN Orders o ON c.customer\_id = o.customer\_id

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



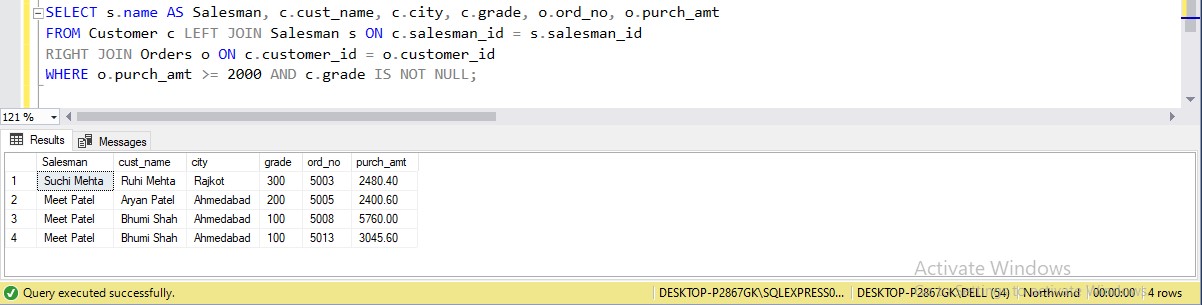
-- 15. Write a SQL statement to generate a list of all the salesmen who either work for one or more customers or have yet to join any of them. The customer may have placed one or more orders at or above order amount 2000, and must have a grade, or he may not have placed any orders to the associated supplier.

SELECT s.name AS Salesman, c.cust\_name, c.city, c.grade, o.ord\_no, o.purch\_amt

FROM Customer c LEFT JOIN Salesman s ON c.salesman\_id = s.salesman\_id

RIGHT JOIN Orders o ON c.customer\_id = o.customer\_id

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



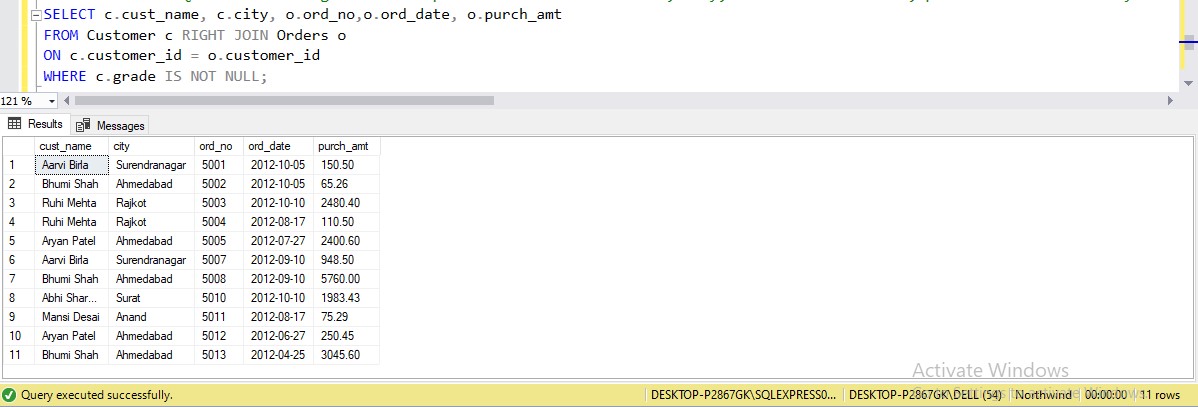
-- 16. Write a SQL statement to generate a report with the customer name, city, order no. order date, purchase amount for only those customers on the list who must have a grade and placed one or more orders or which order(s) have been placed by the customer who neither is on the list nor has a grade.

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

FROM Customer c RIGHT JOIN Orders o

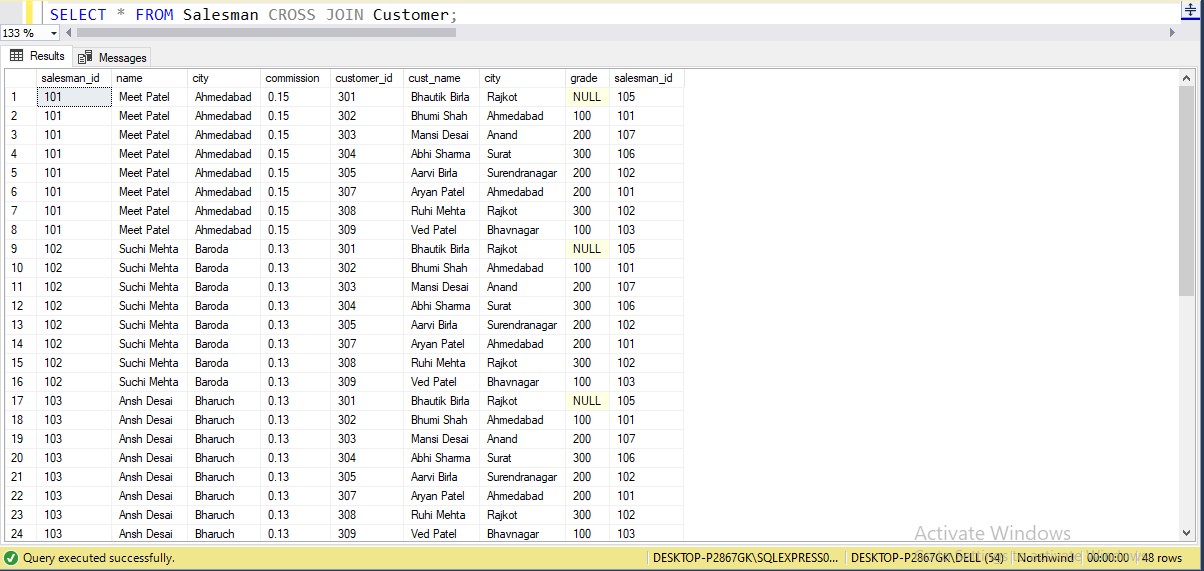
ON c.customer\_id = o.customer\_id

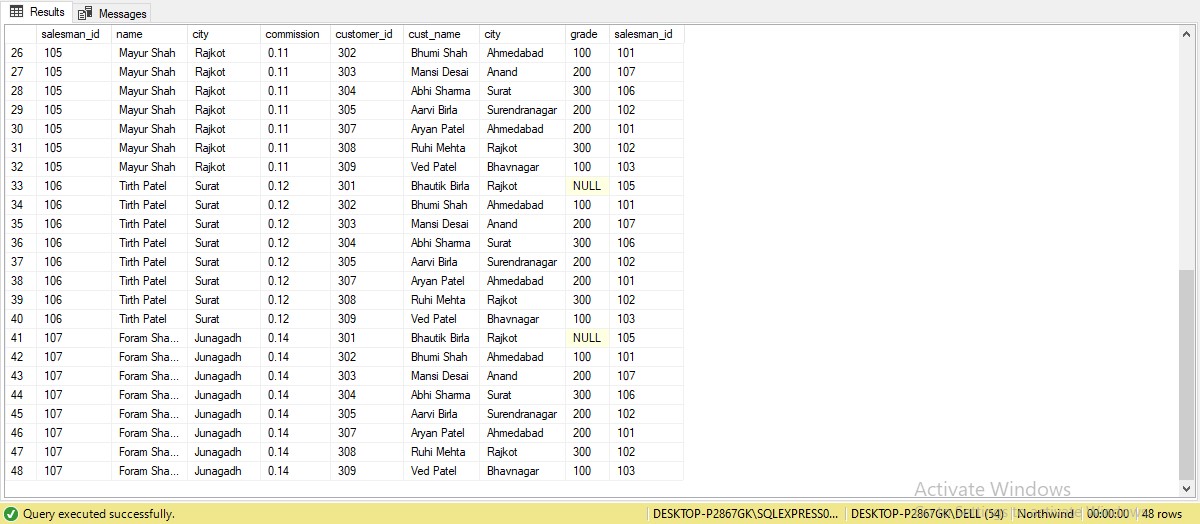
WHERE c.grade IS NOT NULL;



-- 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;

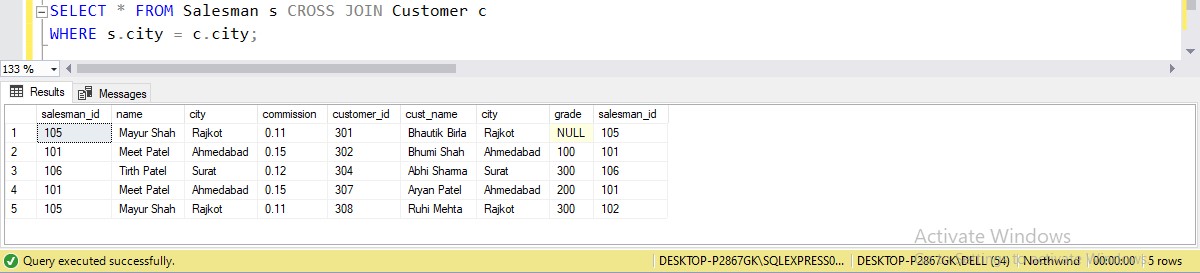




-- 18. Write a SQL statement to create a Cartesian product between salesperson and customer, i.e. each salesperson will appear for all customers and vice versa for that salesperson who belongs to that city.

SELECT \* FROM Salesman s CROSS JOIN Customer c

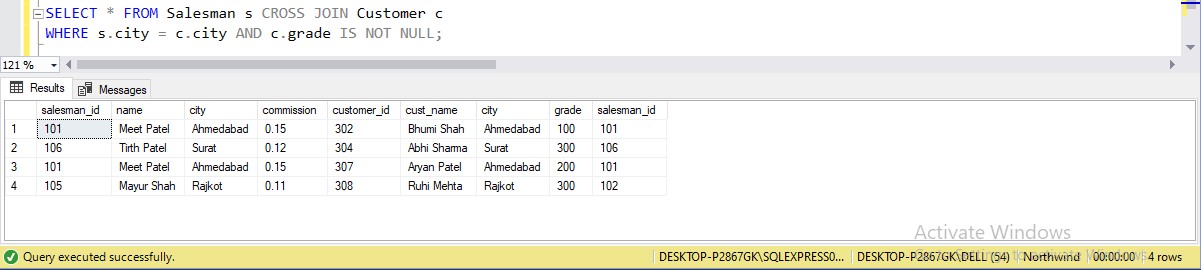
WHERE s.city = c.city;



-- 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 = c.city 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 != c.city AND c.grade IS NOT NULL;



