**SQL - Assignnmet-2**

**NAME: PATEL DEV VIPULKUMAR**

* **Create Table:**

**QUERY**:

* **Salesman Table**

CREATE TABLE salesman(

salesman\_id int NOT NULL PRIMARY KEY,

name nvarchar(50) NOT NULL,

city nvarchar(50),

commission int,

)

* **Customer Table**

CREATE TABLE customer(

customer\_id int NOT NULL PRIMARY KEY,

cust\_name nvarchar(50),

city nvarchar(50),

grade int,

salesman\_id int,

);

ALTER TABLE customer

ADD FOREIGN KEY (salesman\_id) REFERENCES salesman(salesman\_id);

* **Order Table**

**CREATE** TABLE orders(

ord\_no int NOT NULL PRIMARY KEY,

purch\_amt money NOT NULL,

ord\_date date NOT NULL,

customer\_id int NOT NULL,

salesman\_id int NOT NULL,

FOREIGN KEY (customer\_id) REFERENCES customer(customer\_id),

FOREIGN KEY (salesman\_id) REFERENCES salesman(salesman\_id),

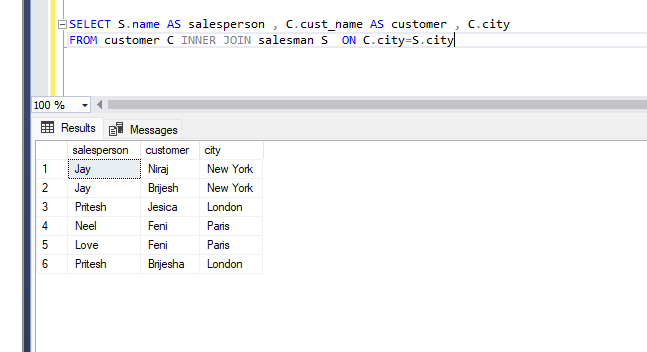
);

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

**QUERY:1**

SELECT S.name AS salesperson , C.cust\_name AS customer , C.city

FROM customer C INNER JOIN salesman S ON C.city=S.city



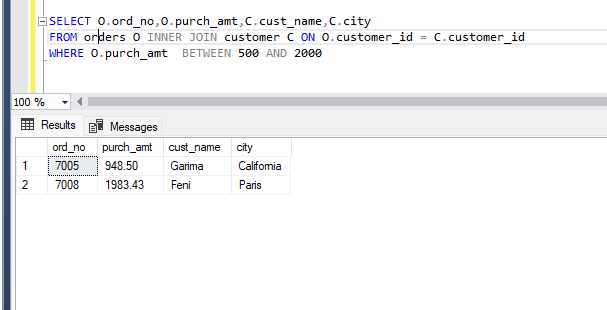
* **write a SQL query to find those orders where the order amount exists between 500 and 2000. Return ord\_no, purch\_amt, cust\_name, city.**

**QUERY:2**

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

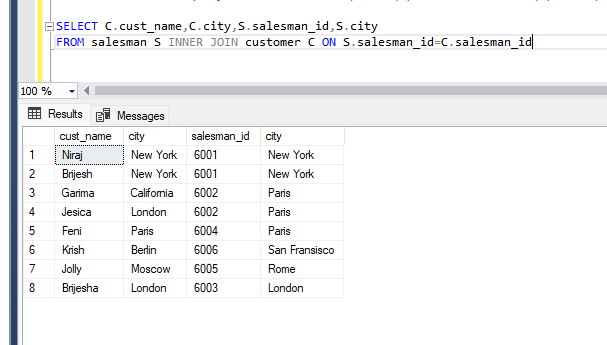


* **write a SQL query to find the salesperson(s) and the customer(s) he represents.Return Customer Name, city, Salesman, commission**

**QUERY:3**

SELECT C.cust\_name,C.city,S.salesman\_id,S.city

FROM salesman S INNER JOIN customer C ON S.salesman\_id=C.salesman\_id



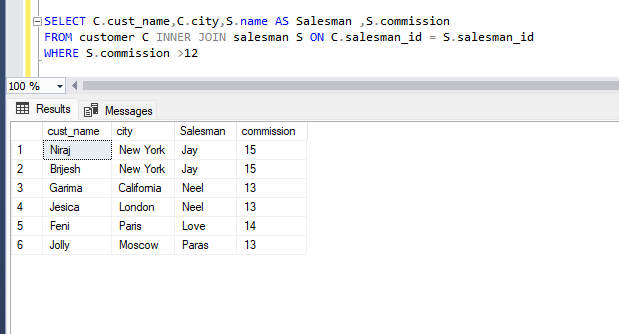
* **write a SQL query to find salespeople who received commissions of more than 12 percent from the company. Return Customer Name, customer city, Salesman,commission.**

**QUERY:4**

**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 >12



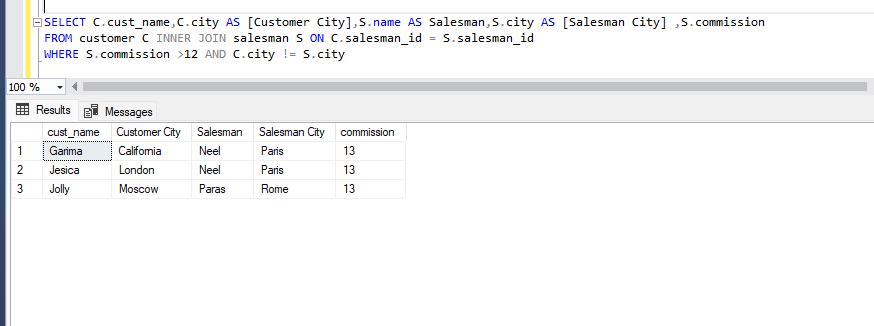
* **write a SQL query to locate those salespeople who do not live in the same city where their customers live and have received a commission of more than 12% from the company. Return Customer Name, customer city, Salesman, salesman city,commission.**

**QUERY:5**

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 S.commission >12 AND C.city != S.city

****

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

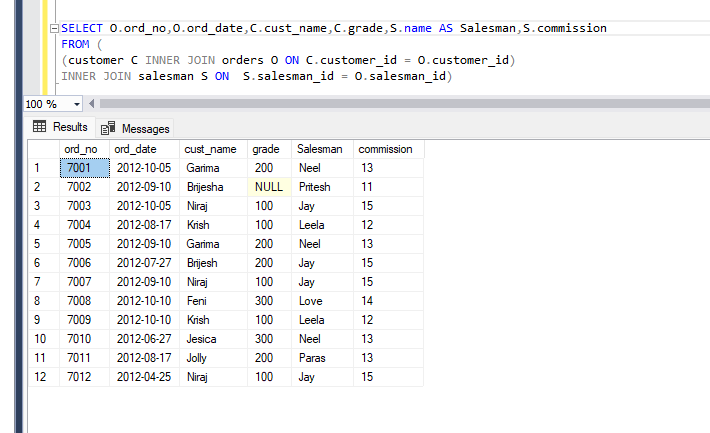
**QUERY:6**

SELECT O.ord\_no,O.ord\_date,C.cust\_name,C.grade,S.name AS Salesman,S.commission

FROM (

(customer C INNER JOIN orders O ON C.customer\_id = O.customer\_id)

INNER JOIN salesman S ON S.salesman\_id = O.salesman\_id)



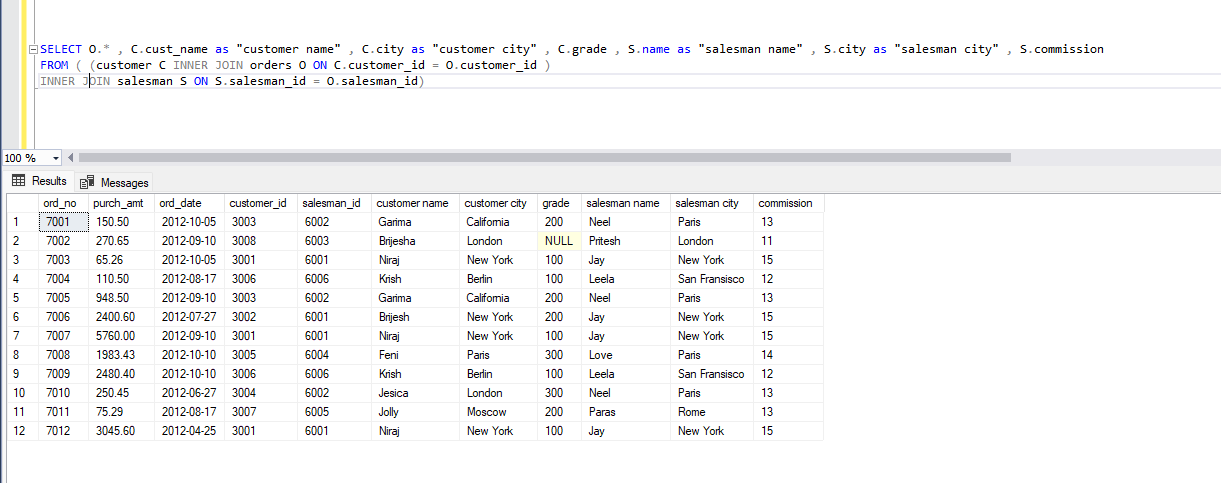
* **Write a SQL statement to join the tables salesman, customer and orders so that the same column of each table appears once and only the relational rows are returned.**

**QUERY: 7**

SELECT O.\* , C.cust\_name as "customer name" , C.city as "customer city" , C.grade , S.name as "salesman name" , S.city as "salesman city" , S.commission

FROM ( (customer C INNER JOIN orders O ON C.customer\_id = O.customer\_id )

INNER JOIN salesman S ON S.salesman\_id = O.salesman\_id)



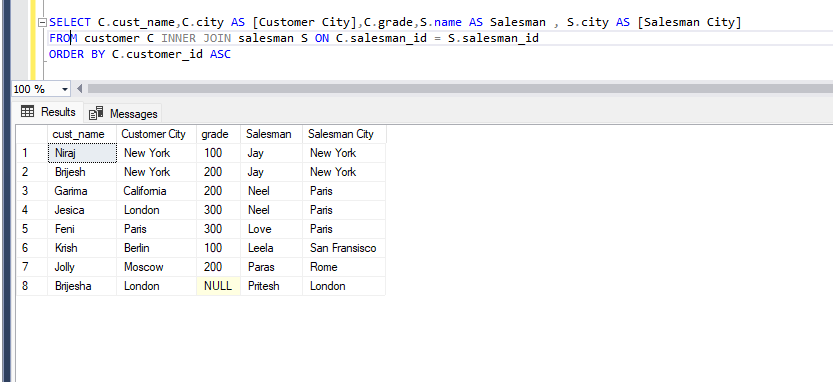
* **write a SQL query to display the customer name, customer city, grade, salesman, salesman city. The results should be sorted by ascending customer\_id.**

**QUERY:8**

**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

ORDER BY C.customer\_id ASC

****

* **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.**

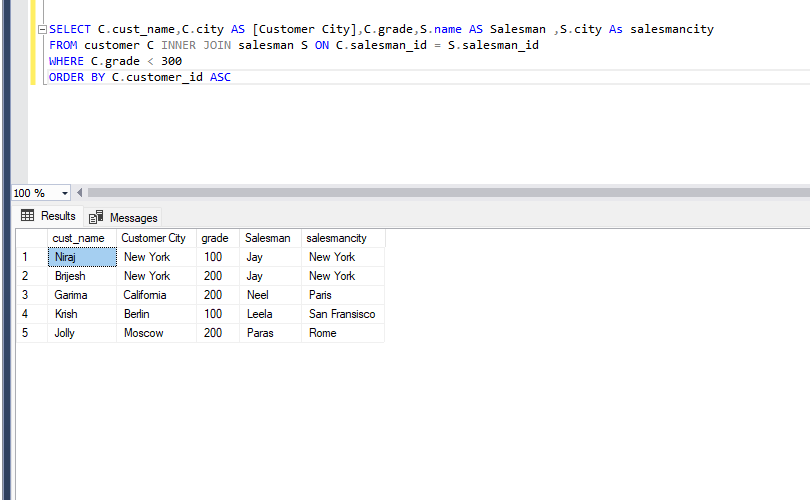
**QUERY:9**

SELECT C.cust\_name,C.city AS [Customer City],C.grade,S.name AS Salesman ,S.city As salesmancity

FROM customer C INNER JOIN salesman S ON C.salesman\_id = S.salesman\_id

WHERE C.grade < 300

ORDER BY C.customer\_id ASC



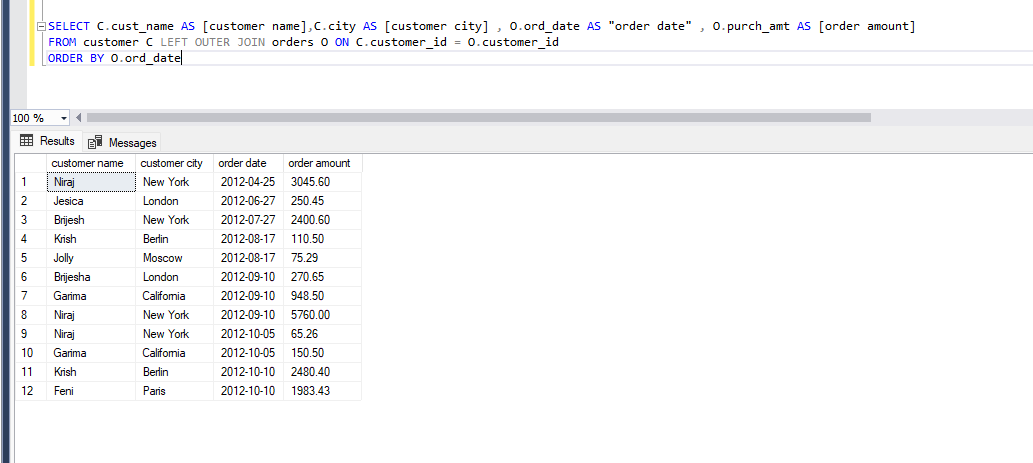
* **Write a SQL statement to make a report with customer name, city, order number, order date, and order amount in ascending order according to the order date to determine whether any of the existing customers have placed an order or not**

**QUERY:10**

SELECT C.cust\_name AS [customer name],C.city AS [customer city] , O.ord\_date AS "order date" , O.purch\_amt AS [order amount]

FROM customer C LEFT OUTER JOIN orders O ON C.customer\_id = O.customer\_id

ORDER BY O.ord\_date

****

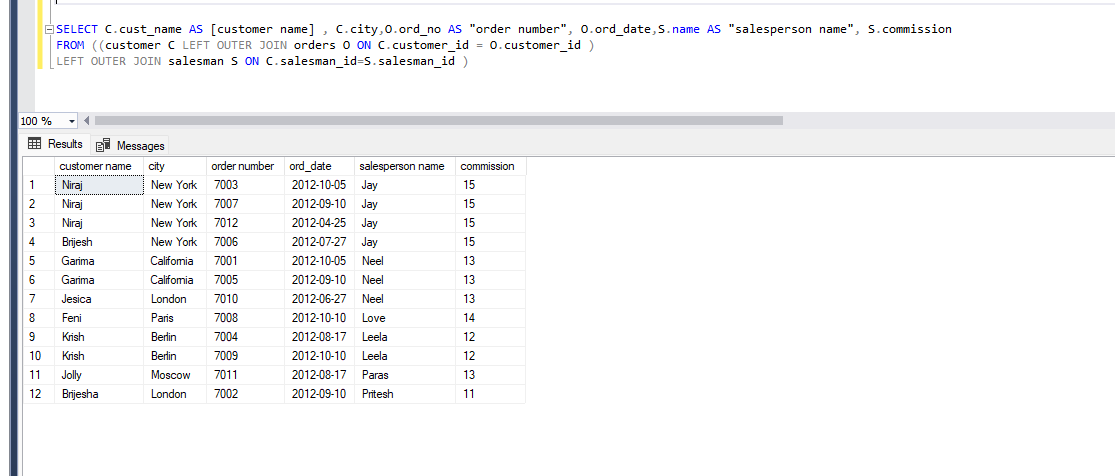
* **Write a SQL statement to generate a report with customer name, city, order number, order date, order amount, salesperson name, and commission to determine if any of the existing customers have not placed orders or if they have placed orders through their salesman or by themselves**

**QUERY:11**

SELECT C.cust\_name AS [customer name] , C.city,O.ord\_no AS "order number", O.ord\_date,S.name AS "salesperson name", S.commission

FROM ((customer C LEFT OUTER JOIN orders O ON C.customer\_id = O.customer\_id )

LEFT OUTER JOIN salesman S ON C.salesman\_id=S.salesman\_id )

****

* **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**

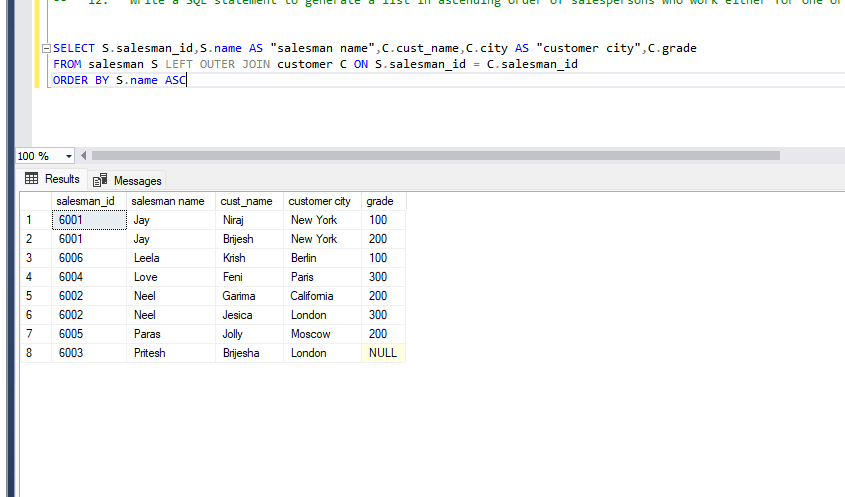
**QUERY:12**

**SELECT** S.salesman\_id,S.name AS "salesman name",C.cust\_name,C.city AS "customer city",C.grade

FROM salesman S LEFT OUTER JOIN customer C

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

ORDER BY S.name ASC

****

* **write a SQL query to list all salespersons along with customer name, city, grade,**

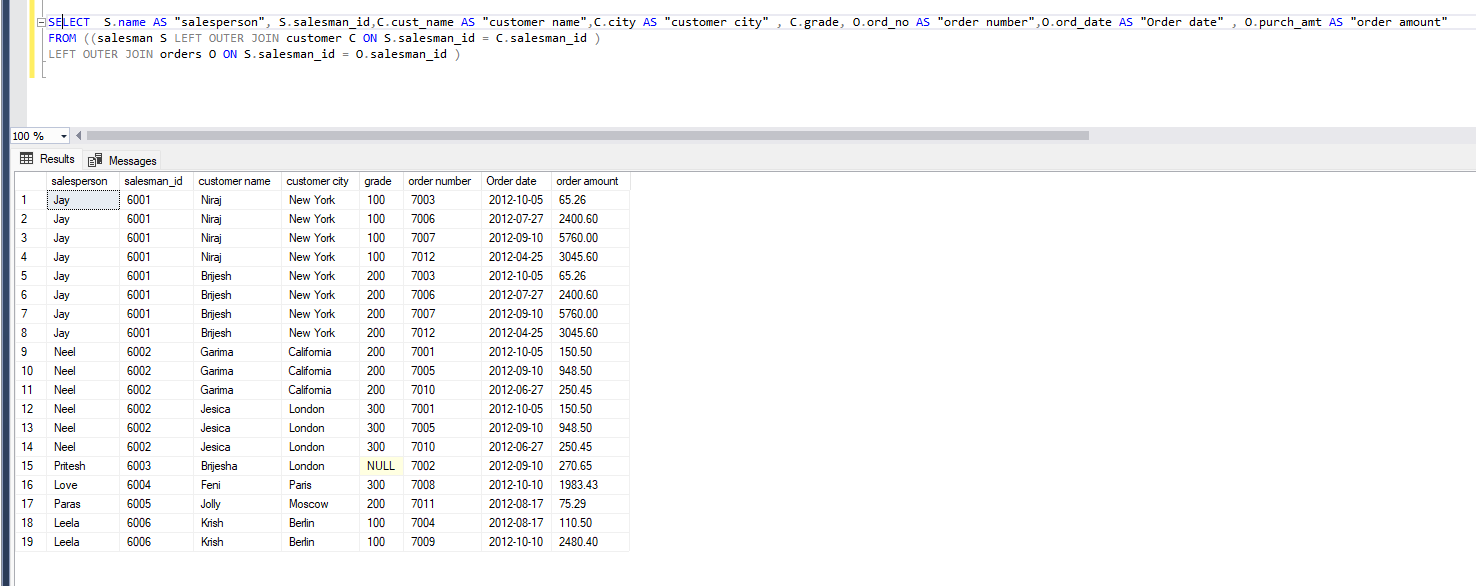
**order number, date, and amount.**

**QUERY:13**

**SELECT** S.name AS "salesperson", S.salesman\_id,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 OUTER JOIN customer C ON S.salesman\_id = C.salesman\_id )

LEFT OUTER JOIN orders O ON S.salesman\_id = O.salesman\_id )

****

* **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.**

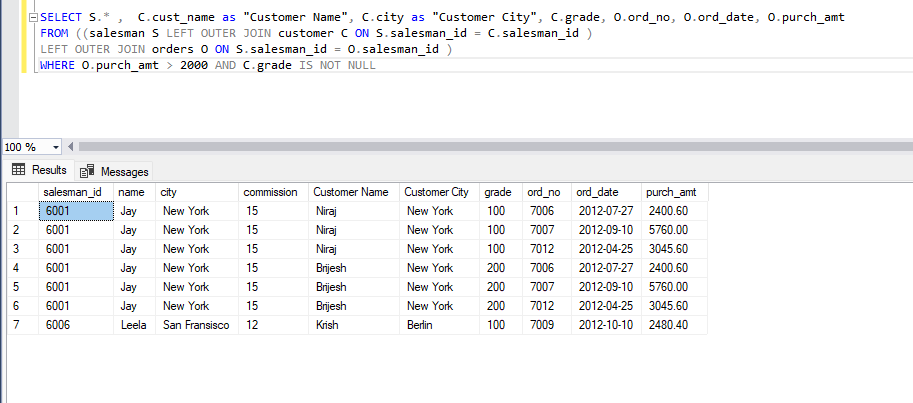
**QUERY:14**

**SELECT** S.\* , C.cust\_name as "Customer Name", C.city as "Customer City", C.grade, O.ord\_no, O.ord\_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 )

WHERE O.purch\_amt > 2000 AND C.grade IS NOT NULL



* **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.**

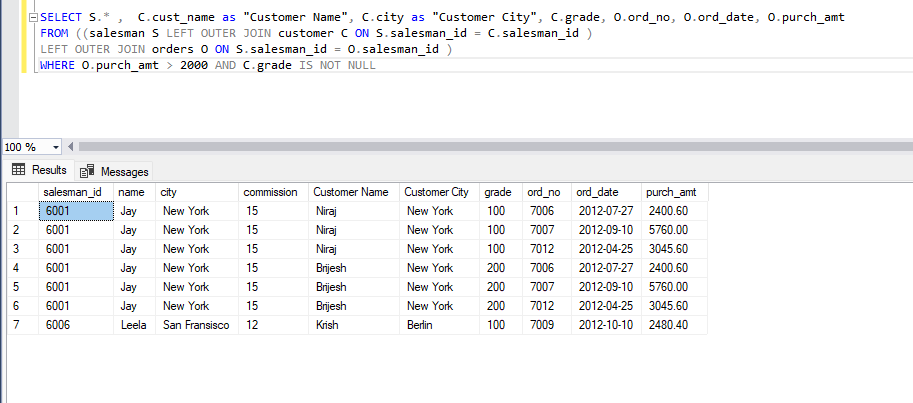
**QUERY:15**

SELECT S.\* , C.cust\_name as "Customer Name", C.city as "Customer City", C.grade, O.ord\_no, O.ord\_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 )

WHERE O.purch\_amt > 2000 AND C.grade IS NOT NULL



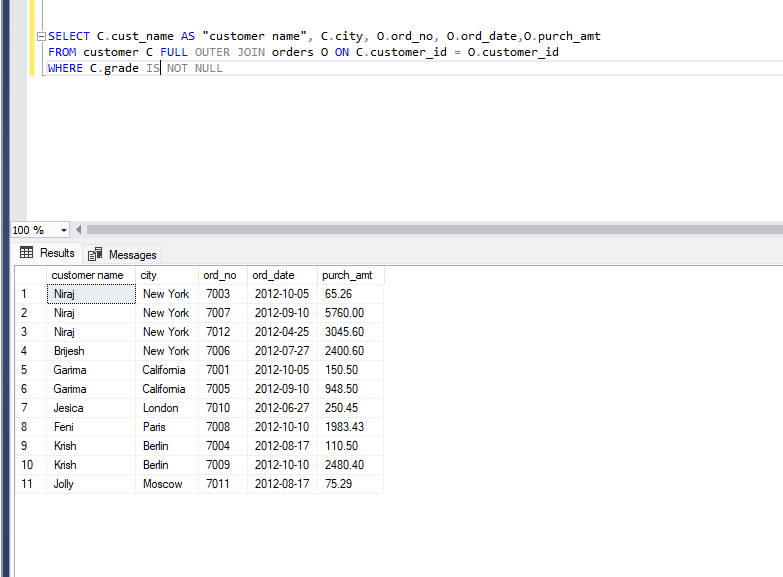
* **Write a SQL statement to generate a report with the customer name, city, order no. order date, purchase amount for only those customers on the list who must have a grade and placed one or more orders or which order(s) have been placed by the customer who neither is on the list nor has a grade.**

**QUERY:16**

SELECT C.cust\_name AS "customer 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

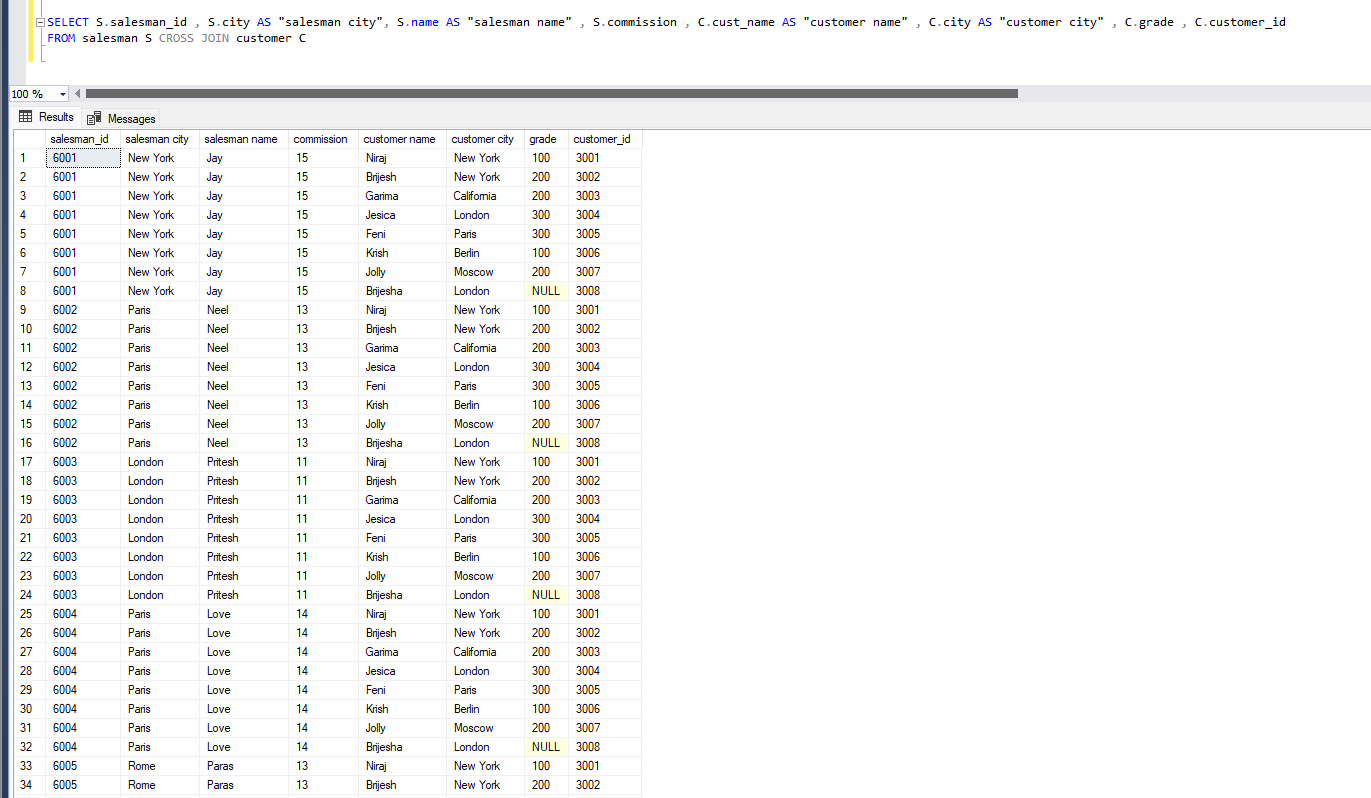


* **Write a SQL query to combine each row of the salesman table with each row of the customer table.**

**QUERY:17**

SELECT S.salesman\_id , S.city AS "salesman city", S.name AS "salesman name" , S.commission , C.cust\_name AS "customer name" , C.city AS "customer city" , C.grade , C.customer\_id

FROM salesman S CROSS JOIN customer C





* **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**

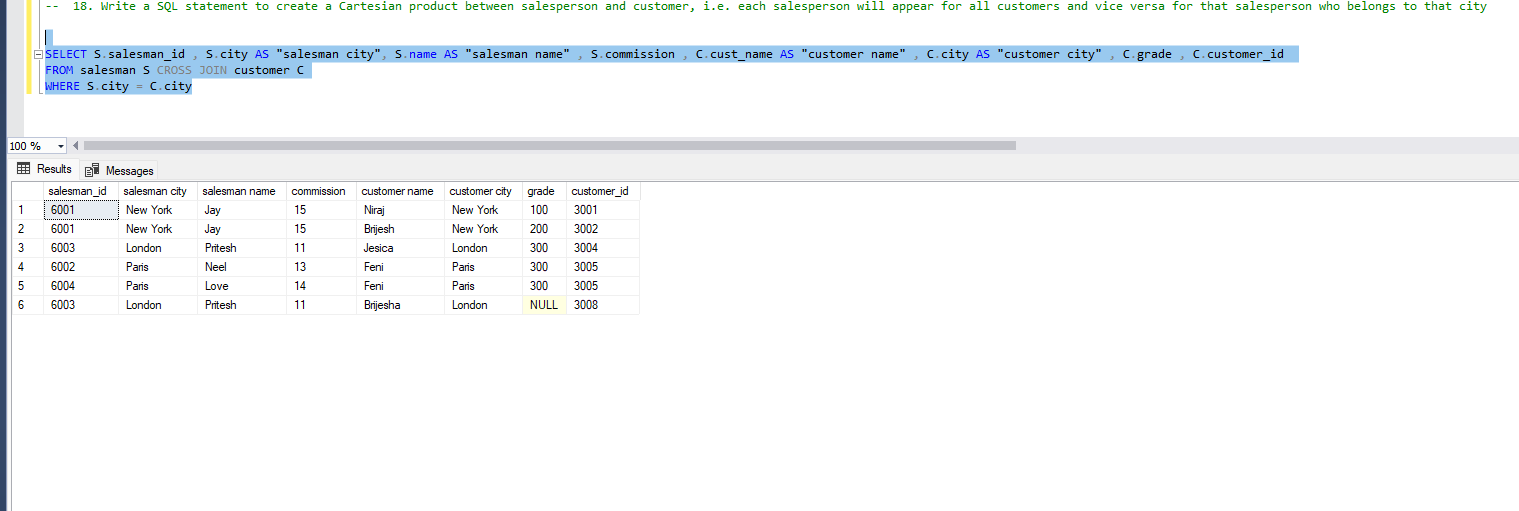
**salesperson who belongs to that city**

**QUERY:18**

SELECT S.salesman\_id , S.city AS "salesman city", S.name AS "salesman name" , S.commission , C.cust\_name AS "customer name" , C.city AS "customer city" , C.grade , C.customer\_id

FROM salesman S CROSS JOIN customer C

WHERE S.city = C.city

****

* **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**

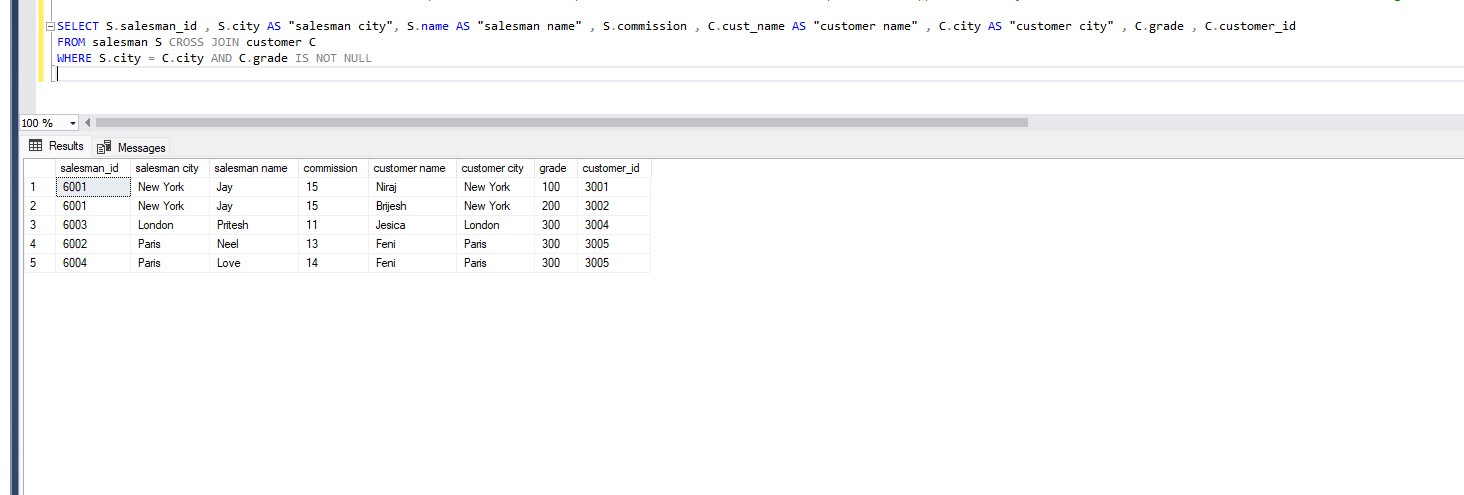
**those salesmen who belong to a city and customers who require a grade**

**QUERY:19**

SELECT S.salesman\_id , S.city AS "salesman city", S.name AS "salesman name" , S.commission , C.cust\_name AS "customer name" , C.city AS "customer city" , C.grade , C.customer\_id

FROM salesman S CROSS JOIN customer C

WHERE S.city = C.city AND C.grade IS NOT NULL

****

* **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:20**

SELECT S.salesman\_id , S.city AS "salesman city", S.name AS "salesman name" , S.commission , C.cust\_name AS "customer name" , C.city AS "customer city" , C.grade , C.customer\_id

FROM salesman S CROSS JOIN customer C

WHERE S.city != C.city AND C.grade IS NOT NULL

