**ASSIGMENT-3**

**DATE:7-2-23**

CREATE database AS2

CREATE TABLE salesman(

salesman\_id int NOT NULL IDENTITY(1,1) PRIMARY KEY,

name NVARCHAR(50) not null,

city NVARCHAR(20) not NULL,

commission decimal(3,2) CONSTRAINT chk\_commission check(commission between 0 and 1) NOT NULL

)

CREATE TABLE customer(

customer\_id int NOT NULL PRIMARY KEY,

cust\_name NVARCHAR(50) NOT NULL,

city NVARCHAR(20) NOT NULL,

grade int NOT NULL,

salesman\_id int not NULL,

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

)

CREATE TABLE orders(

ord\_no int NOT NULL IDENTITY(7001,1) PRIMARY KEY,

purch\_amt DECIMAL(6,2) NOT NULL,

ord\_date date NOT NULL,

customer\_id INT NOT NULL,

salesman\_id INT NOT NULL,

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

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

)

INSERT into salesman VALUES( 'Vikas' , 'Bhopal', '0.1')

,( 'Ram' , 'Vastrapur', '0.2')

,( 'Jayesh' , 'Maninagar', '0.15')

,( 'Suresh' , 'Narol', '0.25')

,( 'Rakesh' , 'Gota', '0.17')

,( 'Aman' , 'Satellite', '0.12')

,( 'Fenil' , 'Sola', '0.1')

,( 'Rakesh' , 'Narol', '0.2')

,( 'Neel' , 'Gota', '0.22')

,( 'Piyush' , 'Sola', '0.12')

,( 'Lucky' , 'Sharkhej' , '0.30')

INSERT into customer VALUES( 101 , 'Darshil' , 'Bhopal', '100' , '1' )

,( 103 , 'Ravi' , 'Shakhej', '200' ,'11')

,( 104 , 'Jayesh' , 'Gota', '400', '5')

,( 105 , 'Manish' , 'Sola', '300','10')

,( 106 , 'Yash' , 'Sola', '280', '10')

,( 121 , 'Raj' , 'Narol', '100' , '4')

,( 125 , 'Riya' , 'Gota', '220', '8')

,( 153 , 'Jay' , 'Vastrapur', '100' , '2')

,( 204 , 'Jay' , 'Gota', '250', '8')

,( 221 , 'Nidhi' , 'Vastrapur', '500' ,'2')

INSERT into customer VALUES( 299 , 'Alice' , 'Sola', '230' ,'10')

,( 306 , 'Lee' , 'Shakhej', '250' ,'11')

,( 402 , 'Mihir' , 'Vastrapur', '400' ,'2')

,( 405 , 'Meet' , 'Vastrapur', '430' ,'2')

,( 425 , 'Malav' , 'Gota', '400' ,'5')

INSERT into orders VALUES('1999', '2022-11-21', '103' ,'11'),

('2566', '2023-01-13', '121' ,'4'),

('2000', '2022-05-07', '221' ,'2'),

('850', '2022-03-01', '106' ,'10'),

('7599', '2022-12-31', '425' ,'2'),

('1999', '2023-01-01', '204' ,'8'),

('3200', '2022-08-05', '405' ,'2'),

('2000', '2023-02-01', '306' ,'11'),

('1099', '2022-07-27', '405' ,'6'),

('1299', '2022-11-21', '402' ,'6'),

('7999', '2022-12-23', '106' ,'1'),

('9999', '2019-09-09', '153' ,'9'),

('2222', '2022-02-22', '221' ,'2'),

('2345', '2020-1-23', '125' ,'6'),

('3333', '2021-03-3', '306' ,'3'),

('999', '2022-12-21', '204' ,'9'),

('7777', '2021-07-07', '221' ,'7'),

('1999', '2022-09-09', '425' ,'9'),

('2500', '2022-11-21', '425' ,'5'),

('2999', '2022-12-31', '103' ,'6')

SELECT\*FROM salesman

SELECT\*FROM customer

SELECT\*FROM orders

drop TABLE salesman

drop TABLE customer

drop TABLE orders

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

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

from salesman inner join customer on salesman.city=customer.city



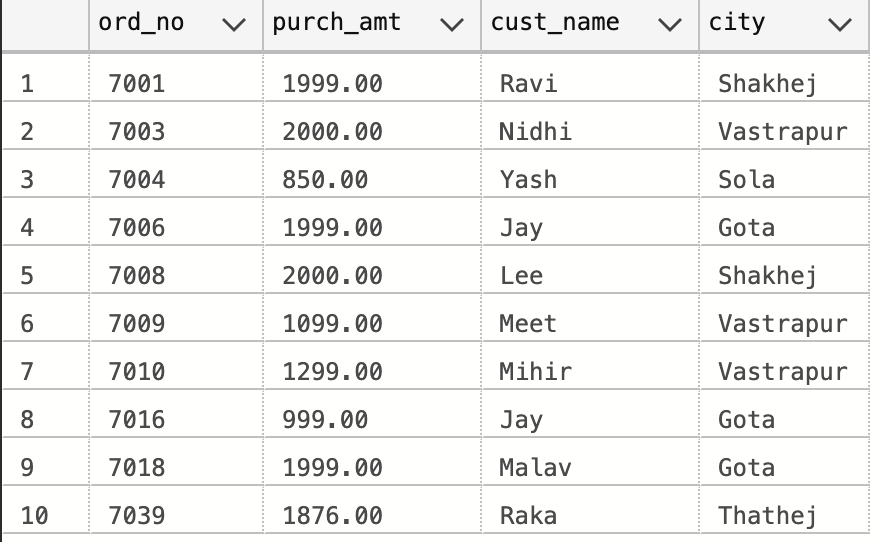
›

-- 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 customer inner join orders on orders.customer\_id=customer.customer\_id

and 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 salesman inner join customer on salesman.salesman\_id=customer.salesman\_id

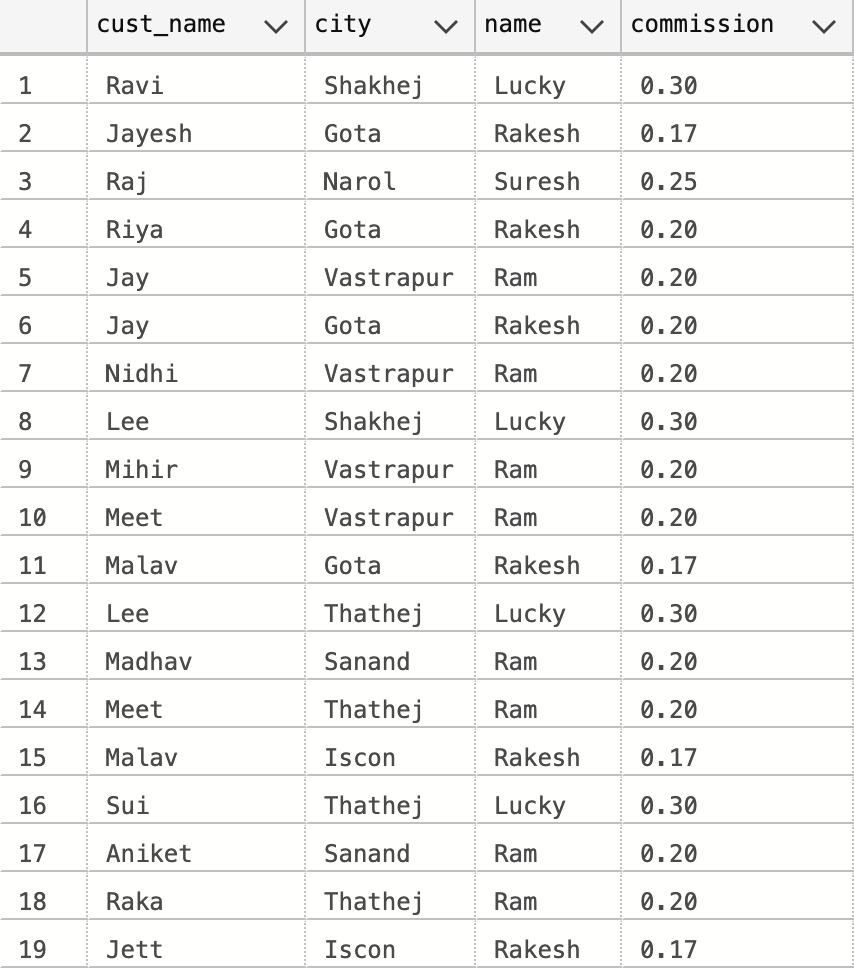


-- 4. write a SQL query to find salespeople who received commissions of more than 12 percent from the company. Return Customer Name, customer city, Salesman, commission.

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

FROM salesman INNER JOIN customer on salesman.salesman\_id=customer.salesman\_id

AND salesman.commission >0.12



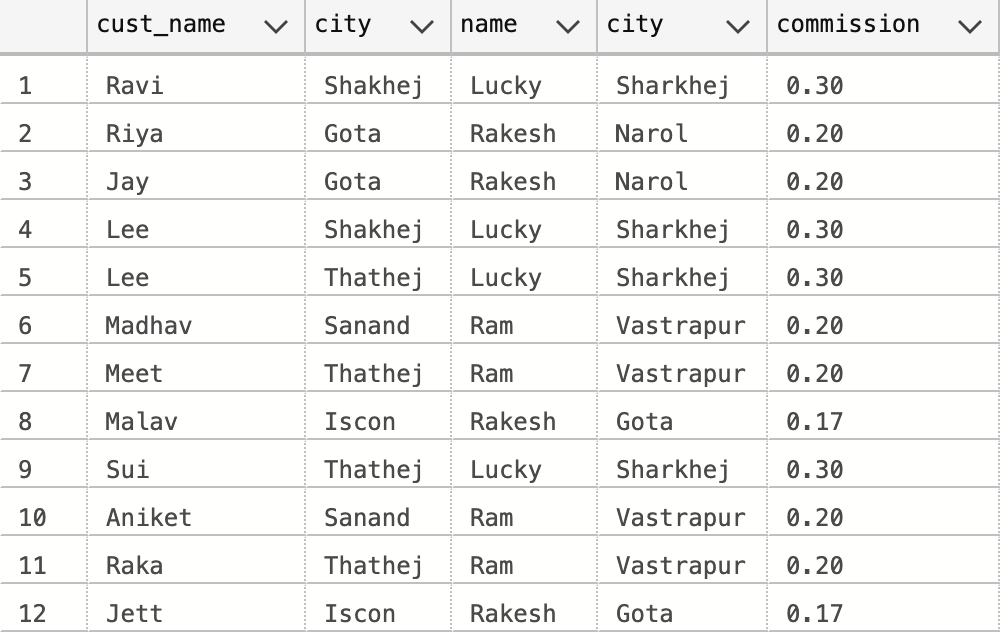
-- 5. write a SQL query to locate those salespeople who do not live in the same city where their customers live and have received a commission of more than 12% from the company. Return Customer Name, customer city, Salesman, salesman city, commission

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

FROM salesman INNER JOIN customer on salesman.city!=customer.city

AND salesman.salesman\_id=customer.salesman\_id

AND salesman.commission >0.12

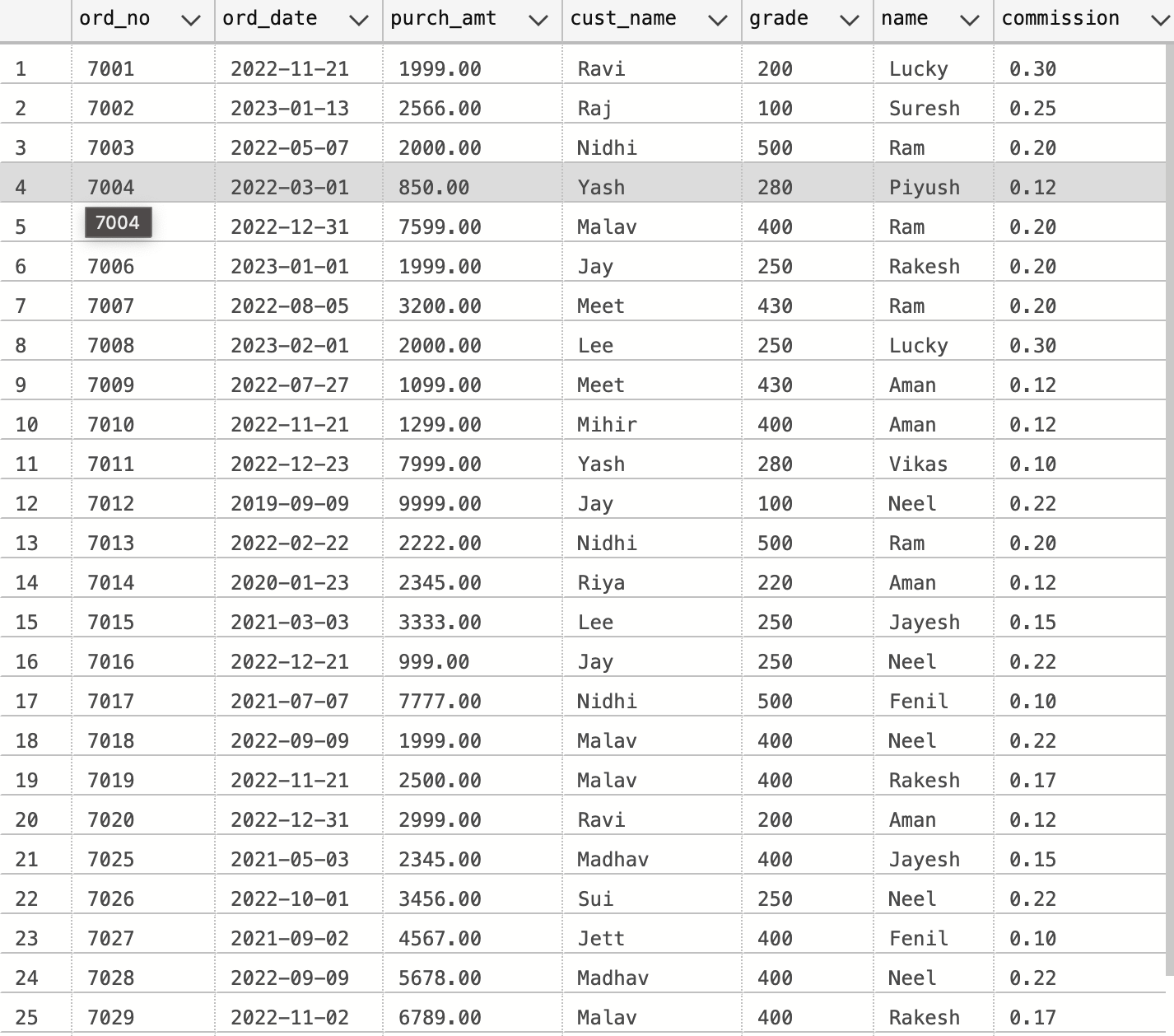


-- 6. write a SQL query to find the details of an order. Return ord\_no, ord\_date, purch\_amt, Customer Name, grade, Salesman, commission

SELECT orders.ord\_no,orders.ord\_date,orders.purch\_amt,customer.cust\_name,customer.grade,salesman.name,salesman.commission

From orders INNER JOIN customer on orders.customer\_id = customer.customer\_id(

INNER JOIN salesman on salesman.salesman\_id=orders.salesman\_id)

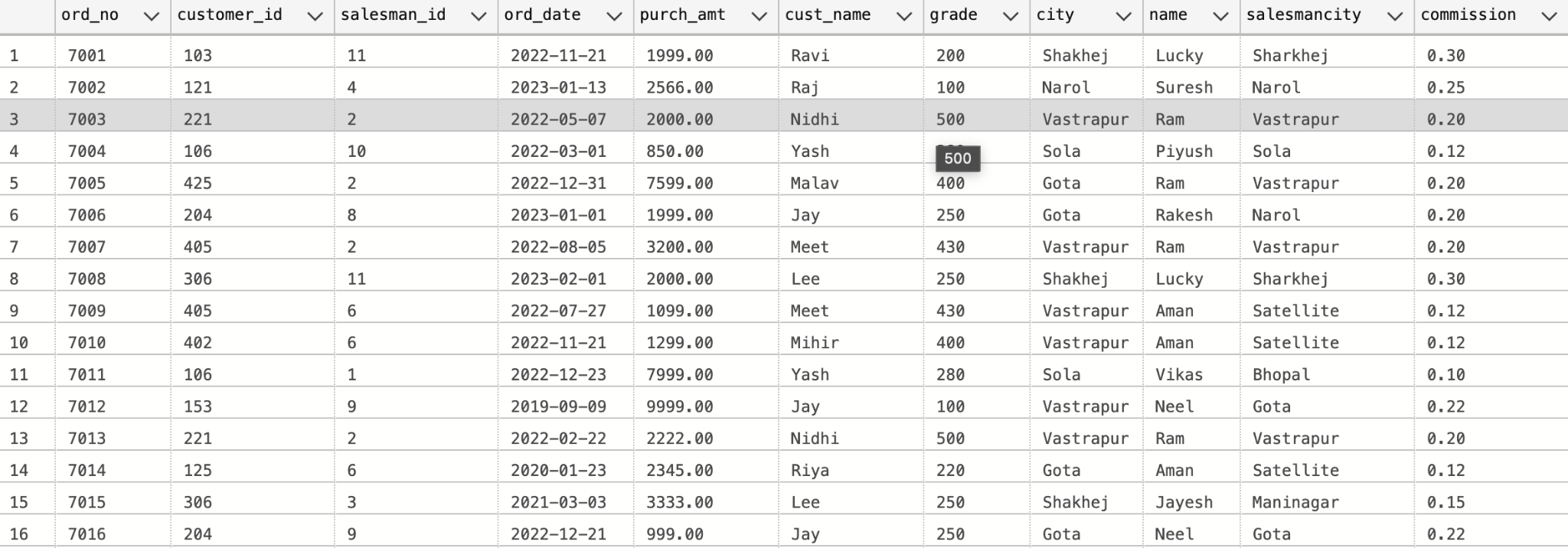


-- 7. Write a SQL statement to join the tables salesman, customer and orders so that the same column of each table appears once and only the relational rows are returned.

SELECT orders.ord\_no,customer.customer\_id,salesman.salesman\_id,orders.ord\_date,orders.purch\_amt,customer.cust\_name,customer.grade,salesman.name,salesman.commission

From orders LEFT JOIN customer on orders.customer\_id = customer.customer\_id

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

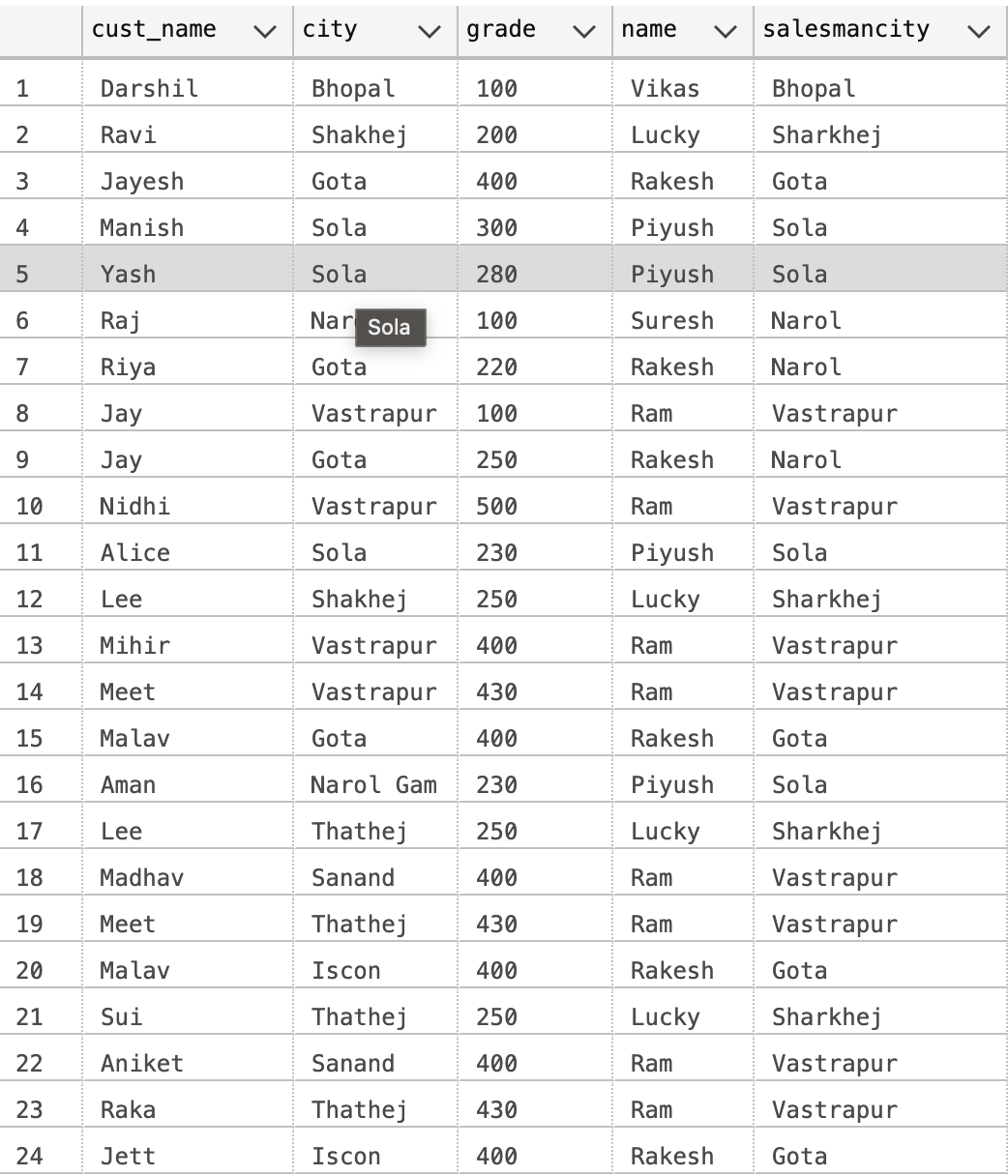


-- 8. write a SQL query to display the customer name, customer city, grade, salesman, salesman city. The results should be sorted by ascending customer\_id.

SELECT customer.cust\_name,customer.city,customer.grade,salesman.name,salesman.city as 'salesmancity'

FROM customer JOIN salesman on salesman.salesman\_id=customer.salesman\_id

ORDER BY customer.customer\_id ASC



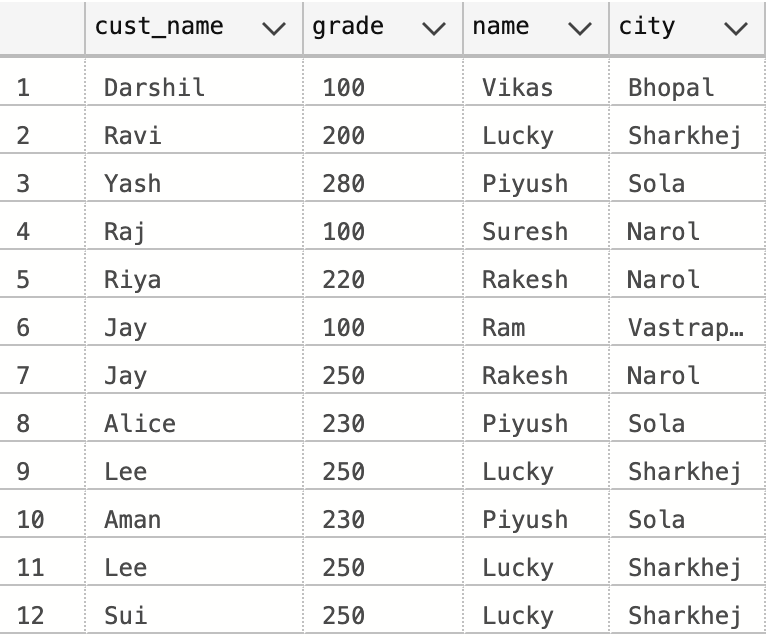
-- 9. write a SQL query to find those customers with a grade less than 300. Return cust\_name, customer city, grade, Salesman, salesmancity. The result should be ordered by ascending customer\_id.

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

From customer INNER JOIN salesman

on salesman.salesman\_id=customer.salesman\_id

AND customer.grade<300



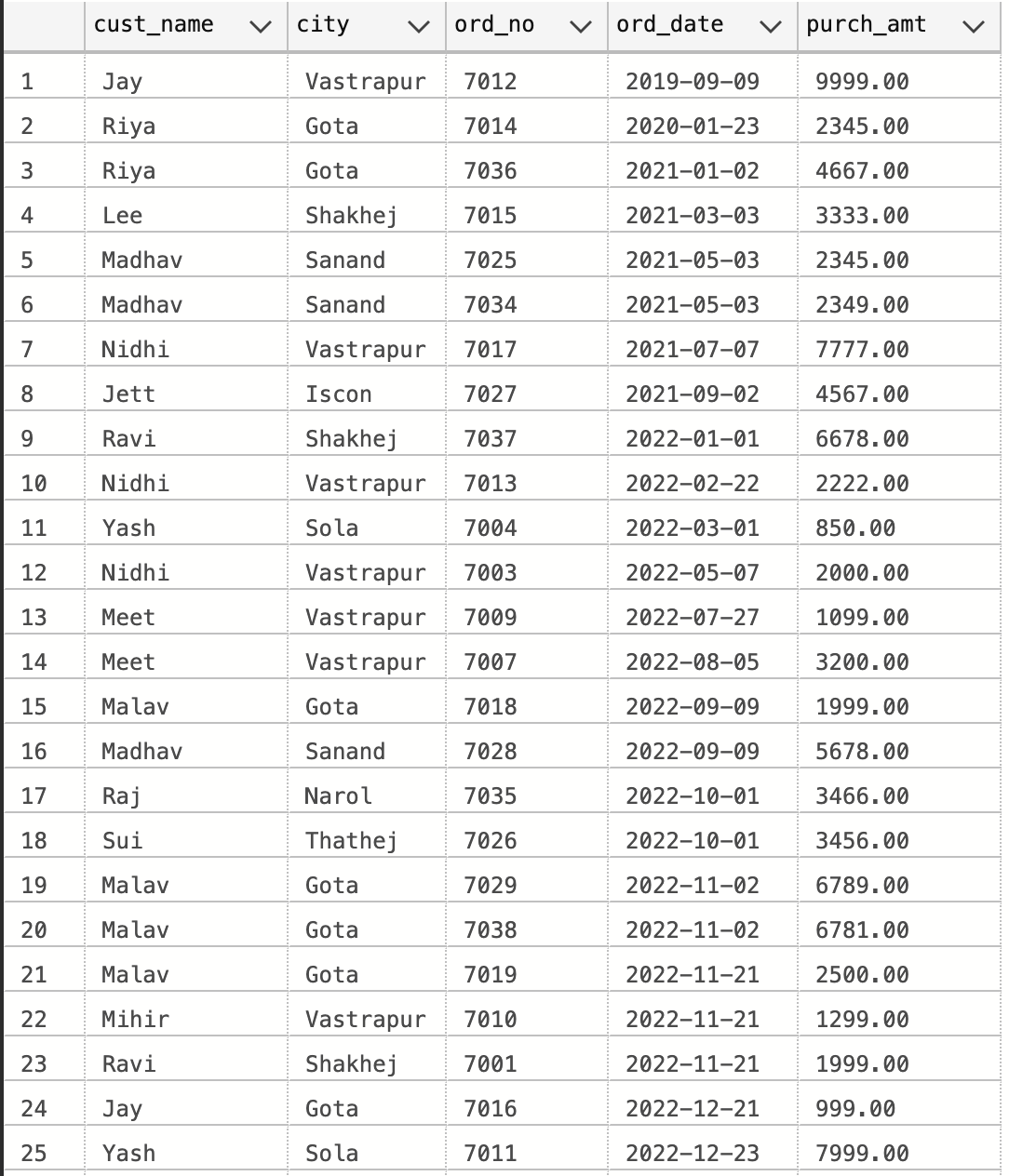
-- 10. Write a SQL statement to make a report with customer name, city, order number, order date, and order amount in ascending order according to the order date to determine whether any of the existing customers have placed an order or not

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

FROM customer INNER JOIN orders

on customer.customer\_id = orders.customer\_id

ORDER BY orders.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

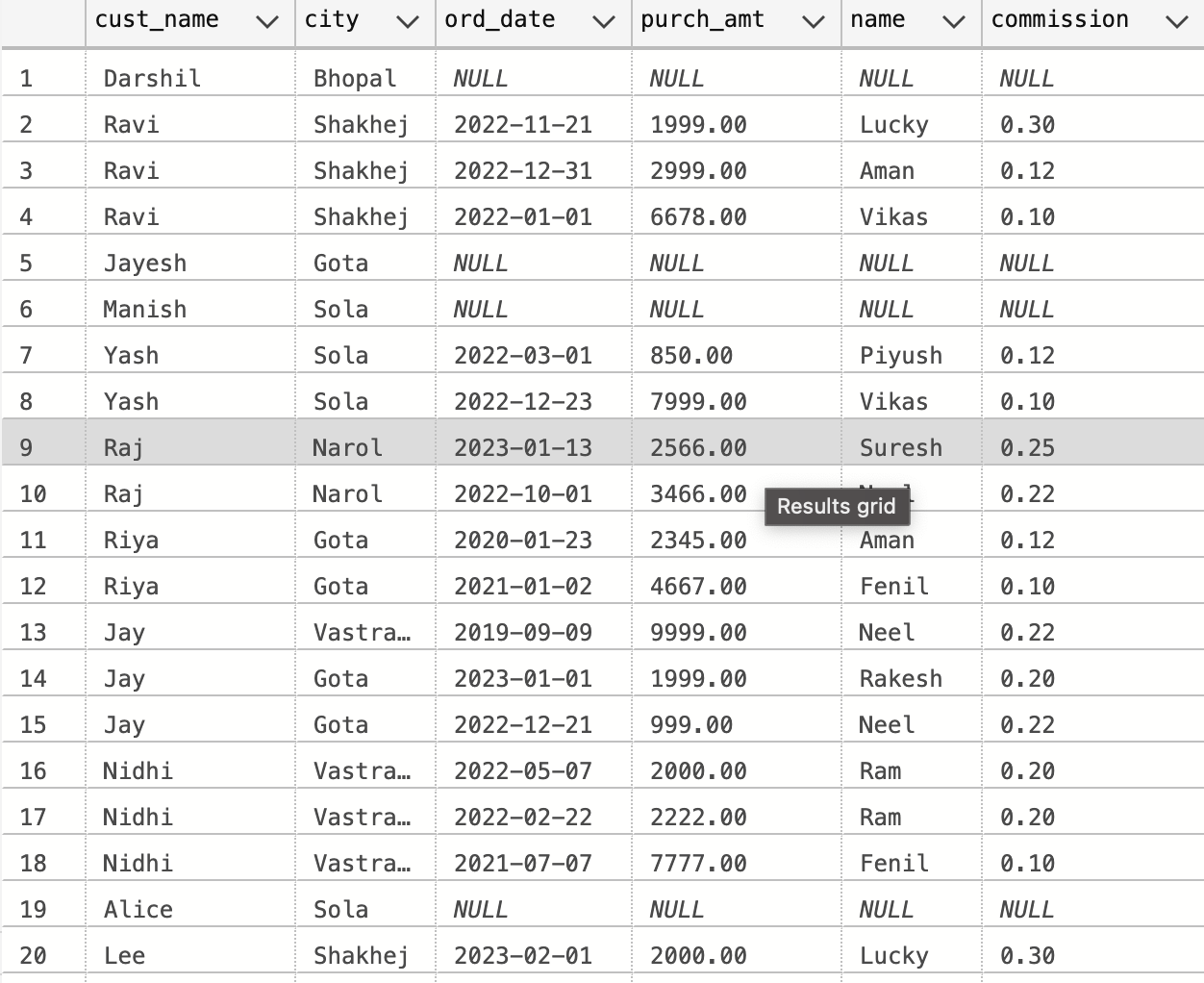
SELECT customer.cust\_name,customer.city,orders.ord\_date,orders.purch\_amt,salesman.name,salesman.commission

FROM customer LEFT JOIN orders

on orders.customer\_id = customer.customer\_id

LEFT JOIN salesman

on orders.salesman\_id = salesman.salesman\_id



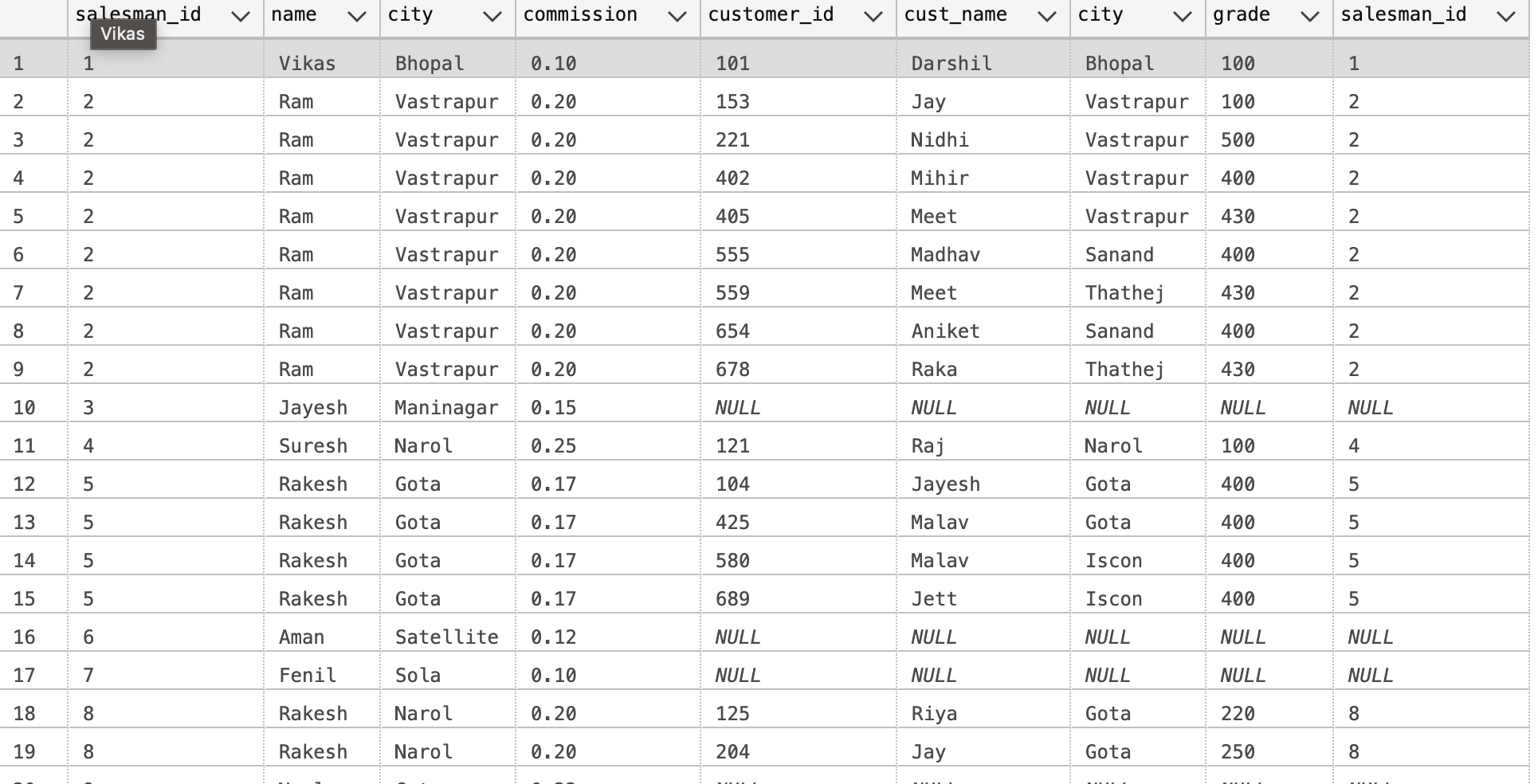
-- 12. Write a SQL statement to generate a list in ascending order of salespersons who work either for one or more customers or have not yet joined any of the customers

SELECT \* FROM

salesman LEFT JOIN customer

on salesman.salesman\_id=customer.salesman\_id

ORDER BY salesman.salesman\_id

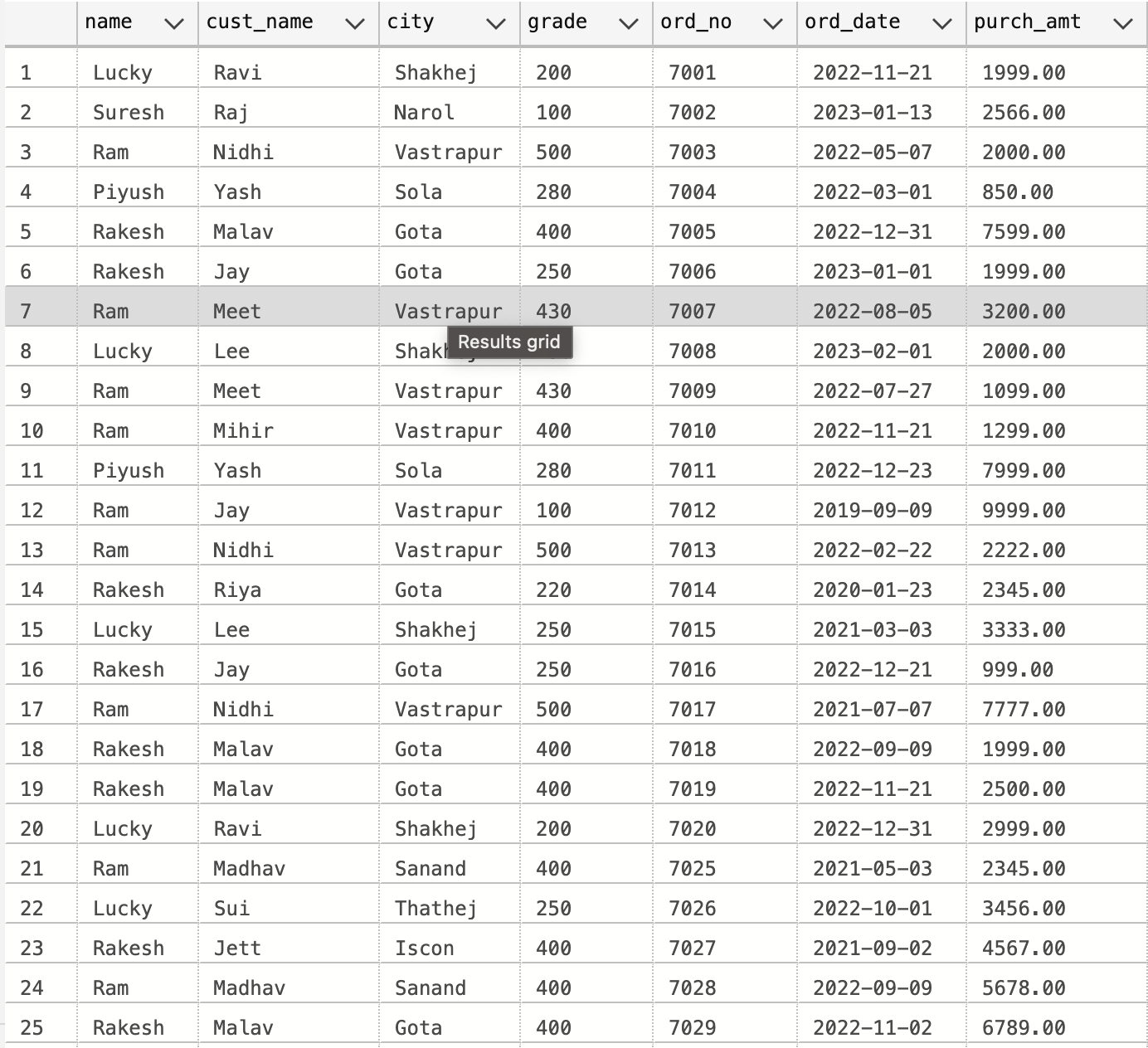


-- 13. write a SQL query to list all salespersons along with customer name, city, grade, order number, date, and amount.

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

FROM salesman INNER JOIN customer on salesman.salesman\_id = customer.salesman\_id

JOIN orders on customer.customer\_id = orders.customer\_id



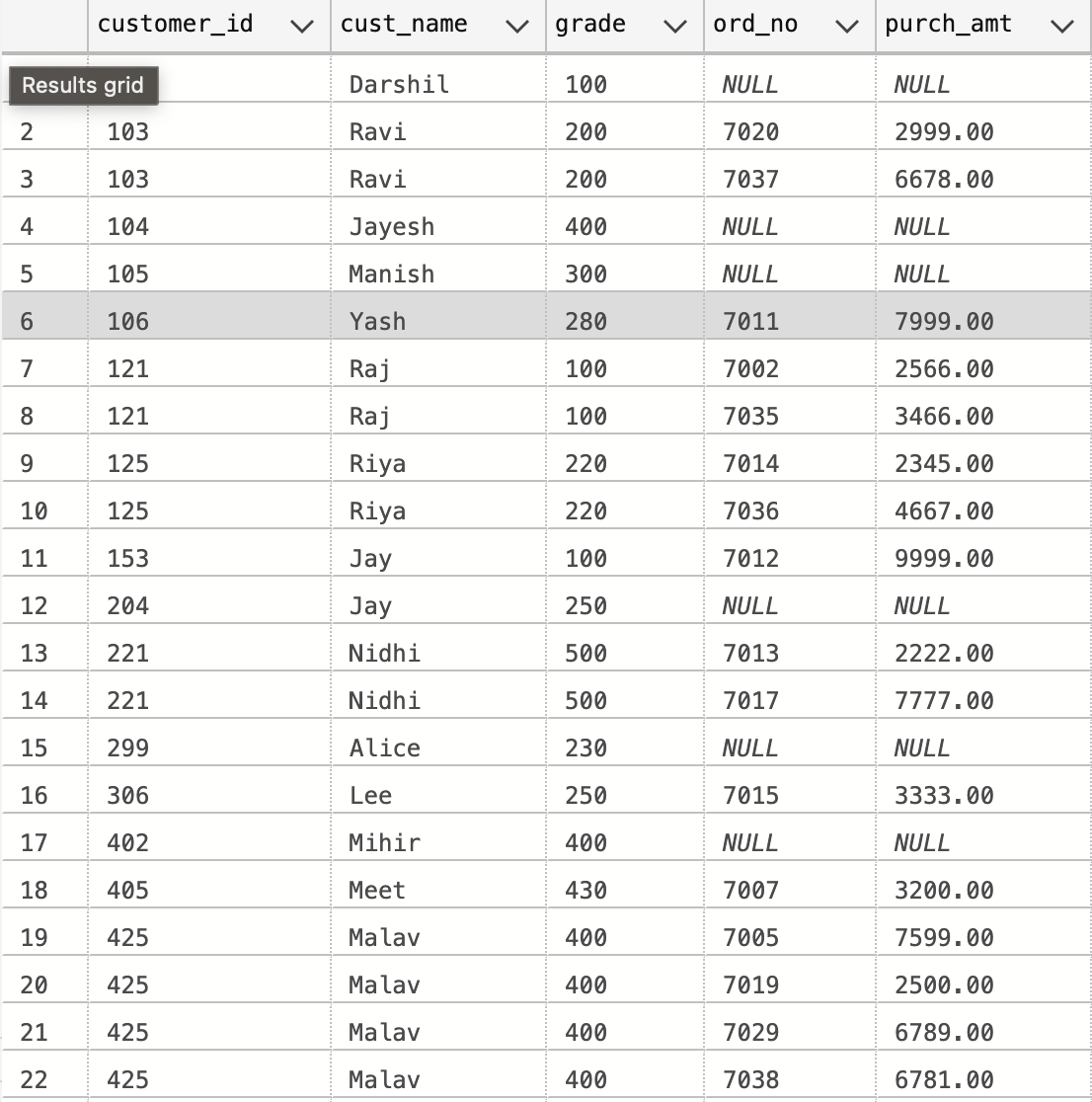
-- 14. Write a SQL statement to make a list for the salesmen who either work for one or more customers or yet to join any of the customers. The customer may have placed, either one or more orders on or above order amount 2000 and must have a grade, or he may not have placed any order to the associated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

SELECT salesman.name,customer.customer\_id,customer.cust\_name,customer.grade,orders.ord\_no,orders.purch\_amt

FROM orders LEFT JOIN salesman on orders.salesman\_id = salesman.salesman\_id

LEFT JOIN customer on customer.customer\_id=orders.customer\_id

WHERE (orders.purch\_amt>2000 AND customer.grade is NOT NULL) or orders.ord\_no 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 salesman.name,customer.customer\_id,customer.cust\_name,customer.grade,orders.ord\_no,orders.purch\_amt

FROM orders LEFT JOIN salesman on orders.salesman\_id = salesman.salesman\_id

LEFT JOIN customer on customer.customer\_id=orders.customer\_id

WHERE (orders.purch\_amt>2000 AND customer.grade is NOT NULL) or orders.ord\_no 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 customer.cust\_name, customer.city, orders.ord\_no, orders.ord\_date, orders.purch\_amt

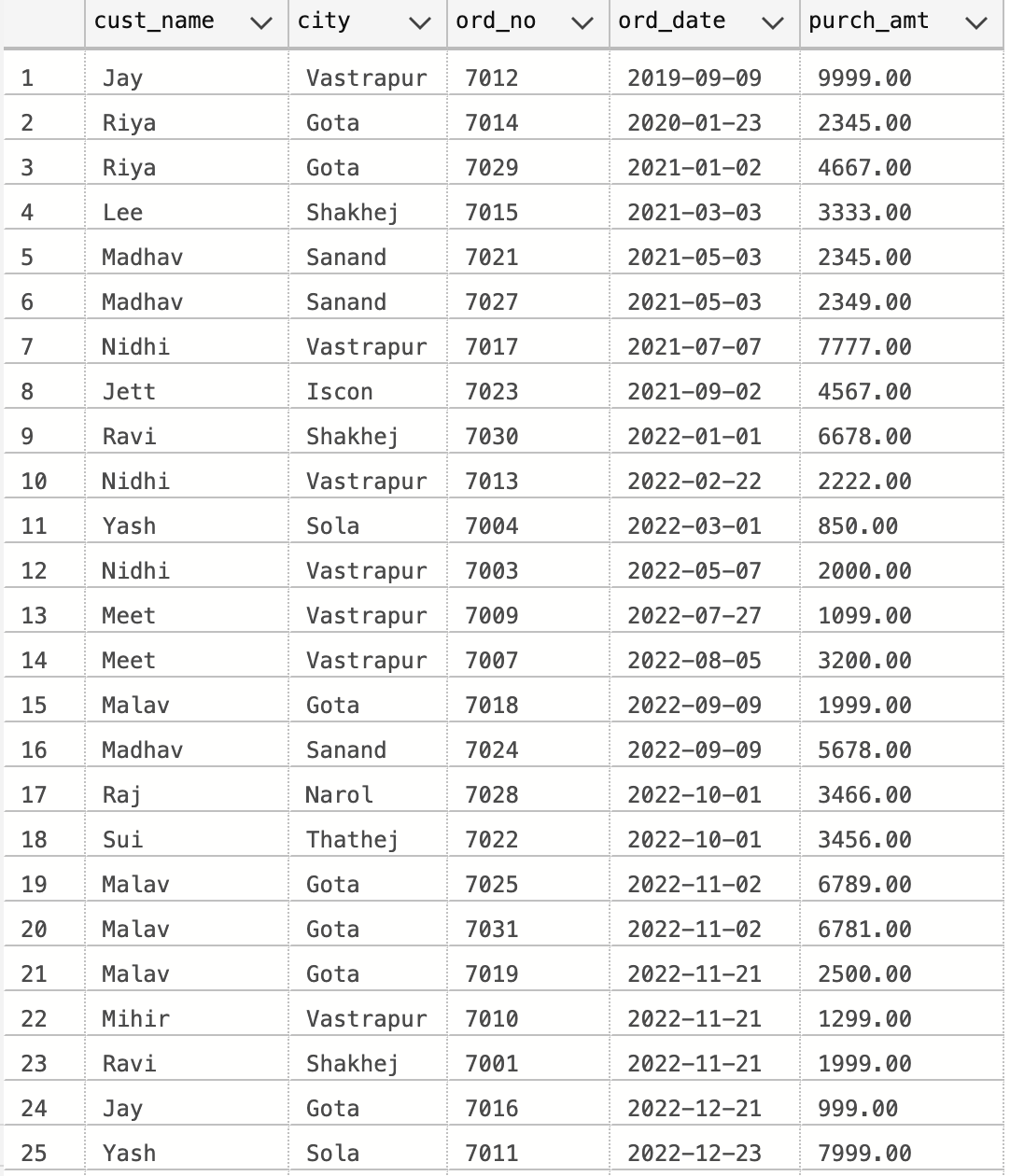
FROM customer

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

WHERE (customer.grade IS NOT NULL AND customer.customer\_id IN (SELECT customer\_id FROM orders))

OR (customer.grade IS NULL AND customer.customer\_id NOT IN (SELECT customer\_id FROM customer WHERE grade IS NOT NULL))

ORDER BY orders.ord\_date;



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

SELECT \*

FROM salesman a

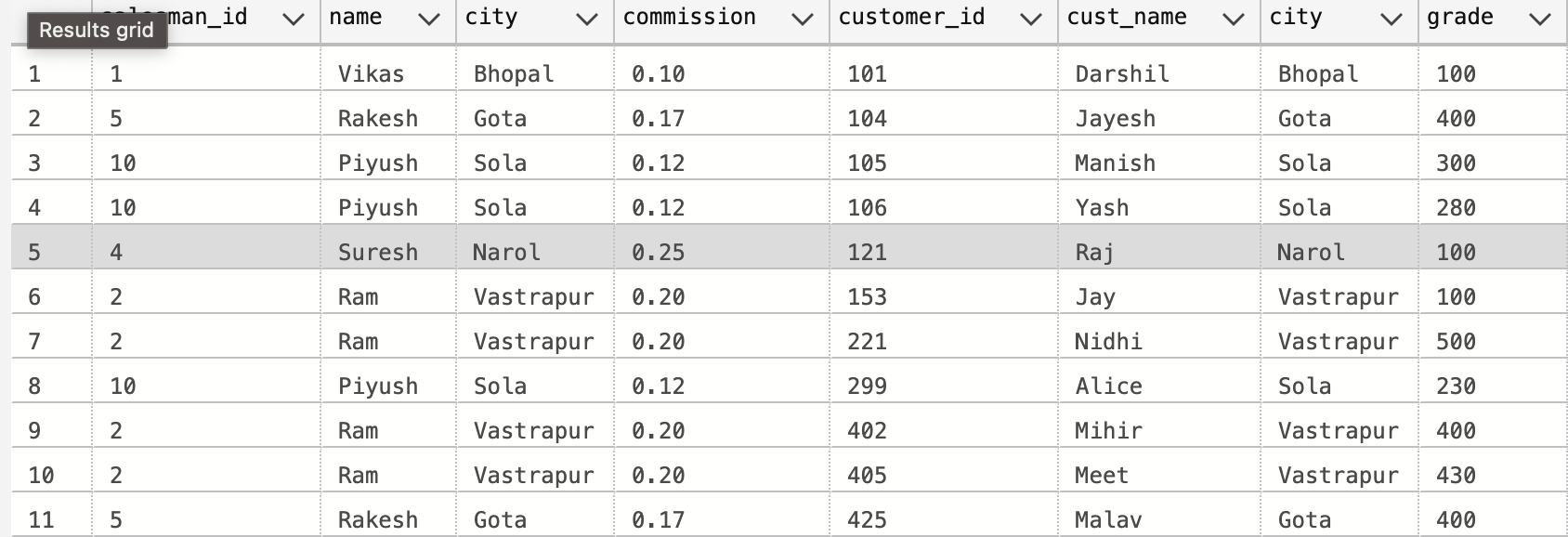
CROSS JOIN customer b;

-- 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 salesman.\*, customer.\*

FROM salesman, customer

WHERE salesman.city = customer.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

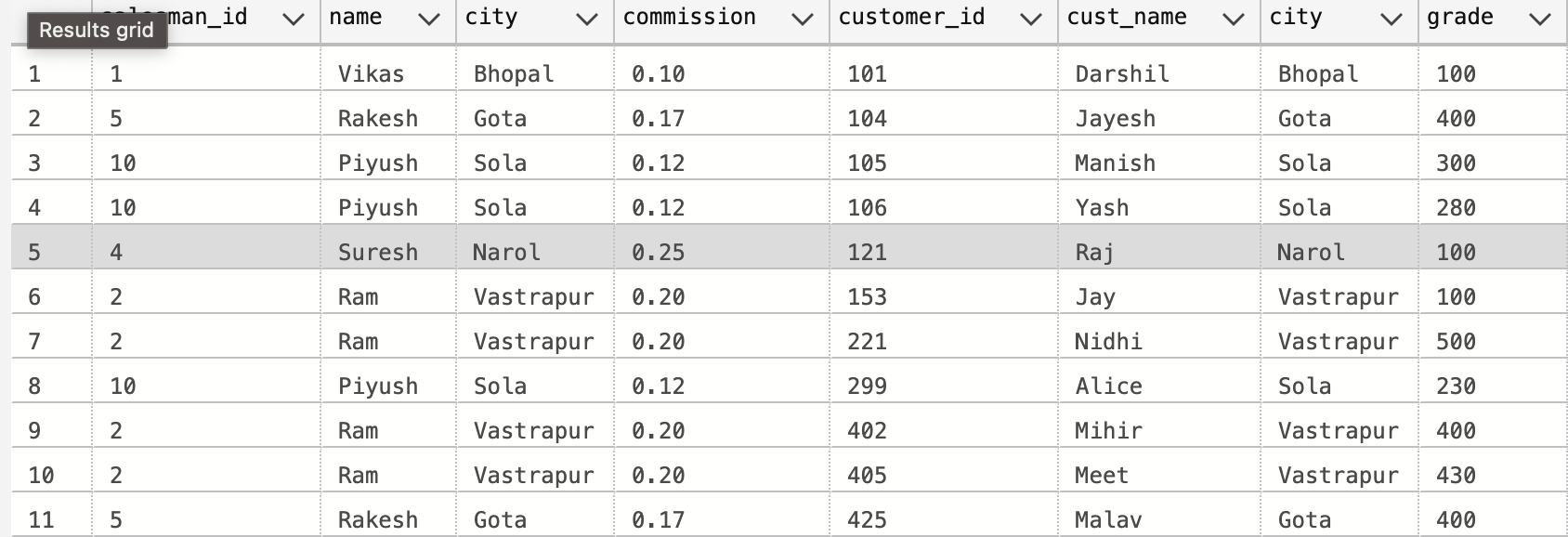
SELECT salesman.\*, customer.\*

FROM salesman, customer

WHERE salesman.city = customer.city

AND customer.grade IS NOT NULL

AND salesman.salesman\_id=customer.salesman\_id



-- 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 salesman.\*, customer.\*

FROM salesman, customer

WHERE salesman.city != customer.city

AND customer.grade IS NOT NULL AND salesman.salesman\_id=customer.salesman\_id;

