**Assignment – 2**

* Creating Database:

CREATE DATABASE sales\_db

* Creating Tables:
  1. Salesman:

==>

CREATE TABLE tblSalesman

(

salesman\_id int PRIMARY KEY,

salesman\_name VARCHAR(30) NOT NULL,

city VARCHAR(30),

commission FLOAT(2) NOT NULL CONSTRAINT CHK\_tblSalesman\_commission CHECK( commission > 0 )

)

2) Customer:

==>

CREATE TABLE tblCustomers

(

customer\_id INT PRIMARY KEY,

cust\_name VARCHAR(30) NOT NULL,

city VARCHAR(30) NOT NULL,

grade INT,

salesman\_id INT,

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

)

ALTER TABLE tblCustomers

ADD CONSTRAINT CHK\_tblCustomers\_grade CHECK(grade > 0)

3) Order:

==>

CREATE TABLE tblOrders

(

ord\_no INT PRIMARY KEY,

purch\_amount FLOAT(2) NOT NULL CONSTRAINT CHK\_tblOrders\_amount CHECK (purch\_amount > 0),

ord\_date DATE NOT NULL,

customer\_id INT,

salesman\_id INT,

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

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

)

Queries:

Q1) Write a SQL query to find the salesperson and customer who reside in the same city.

Return Salesman, cust\_name and city.

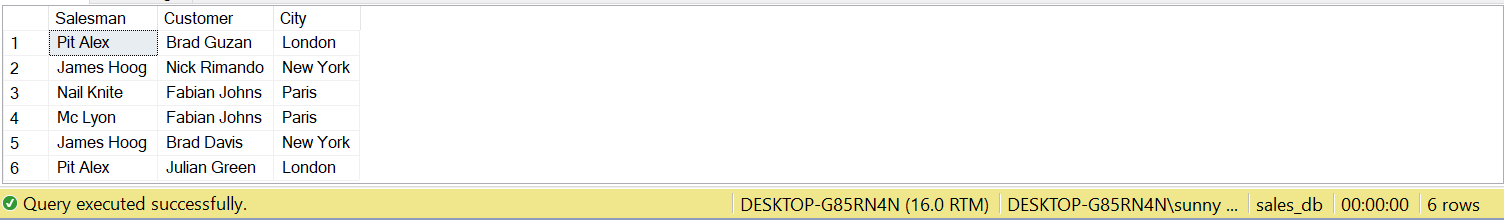
==>

SELECT salesman.salesman\_name Salesman, customer.cust\_name Customer, customer.city City

FROM tblCustomers customer JOIN tblSalesman salesman

ON customer.city = salesman.city

Output:



Q2) 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.

==>

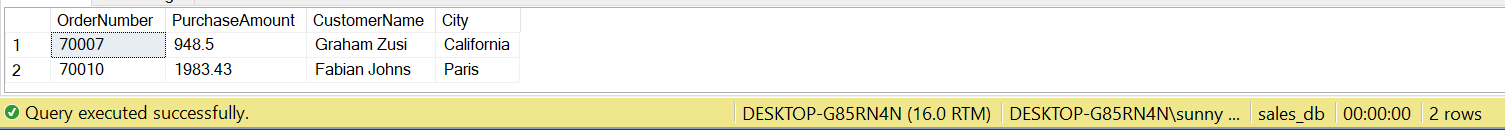
SELECT ord.ord\_no OrderNumber, ord.purch\_amount PurchaseAmount, customer.cust\_name CustomerName, customer.city City

FROM tblOrders ord JOIN tblCustomers customer

ON ord.customer\_id = customer.customer\_id

WHERE ord.purch\_amount BETWEEN 500 AND 2000

Output:



Q3) Write a SQL query to find the salesperson(s) and the customer(s) he represents. Return Customer Name, city, Salesman, commission.

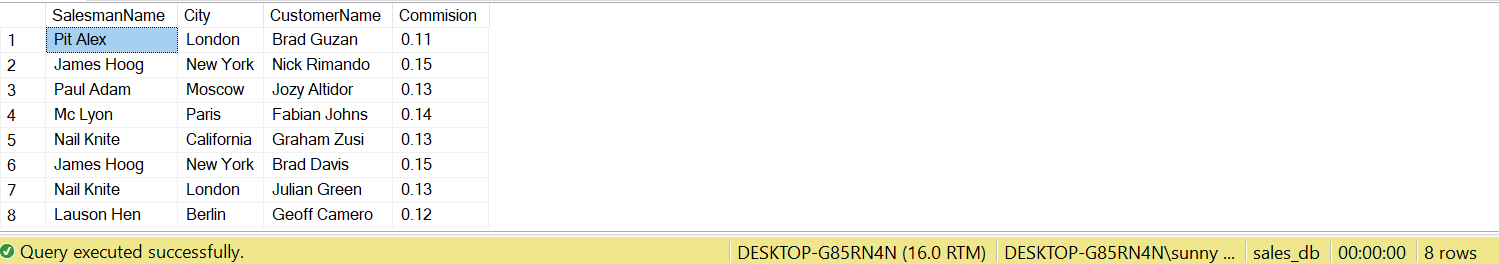
==>

SELECT salesman.salesman\_name SalesmanName, customer.city City, customer.cust\_name CustomerName, salesman.commission Commision

FROM tblSalesman salesman JOIN tblCustomers customer

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

Output:



Q4) 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 CustomerName, customer.city City, salesman.salesman\_name SalesmanName,

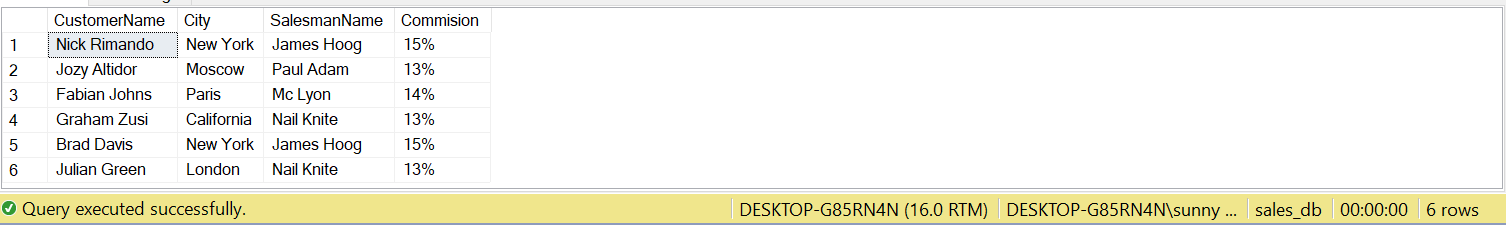
FORMAT( salesman.commission, 'P0') Commision

FROM tblSalesman salesman INNER JOIN tblCustomers customer

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

WHERE salesman.commission > 0.12

Output:



Q5) 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 CustomerName, customer.city CustomerCity, salesman.salesman\_name SalesmanName,

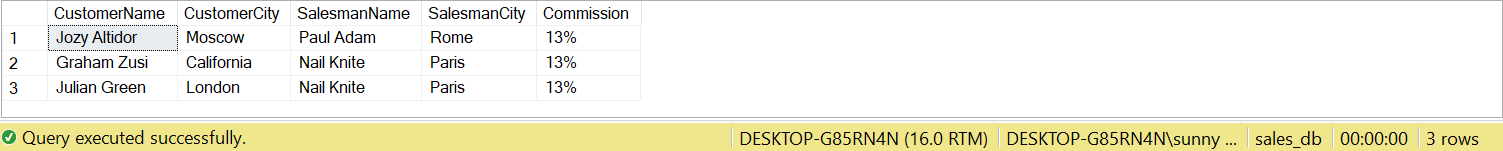
salesman.city SalesmanCity, FORMAT(salesman.commission, 'P0') Commission

FROM tblCustomers customer JOIN tblSalesman salesman

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

WHERE ( customer.city <> salesman.city AND salesman.commission > 0.12)

Output:



Q6) 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.ord\_no, ord.ord\_date, ord.purch\_amount, customer.cust\_name, grade,

salesman.salesman\_name, salesman.commission

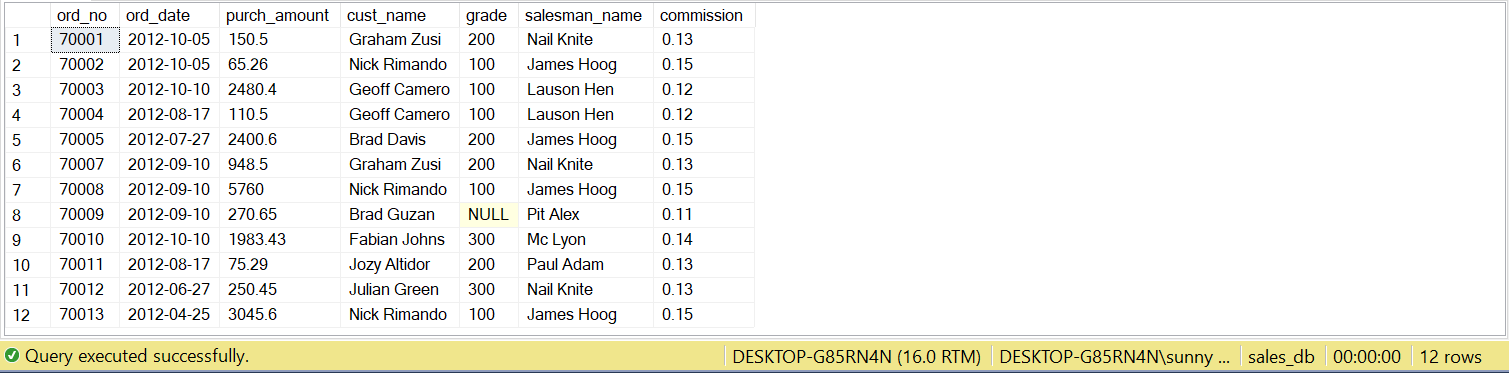
FROM tblOrders ord JOIN tblCustomers customer

ON ord.customer\_id = customer.customer\_id

INNER JOIN tblSalesman salesman

ON ord.salesman\_id = salesman.salesman\_id

Output:



Q7) 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 customer.customer\_id, customer.cust\_name CustomerName, customer.city City, customer.grade Grade,

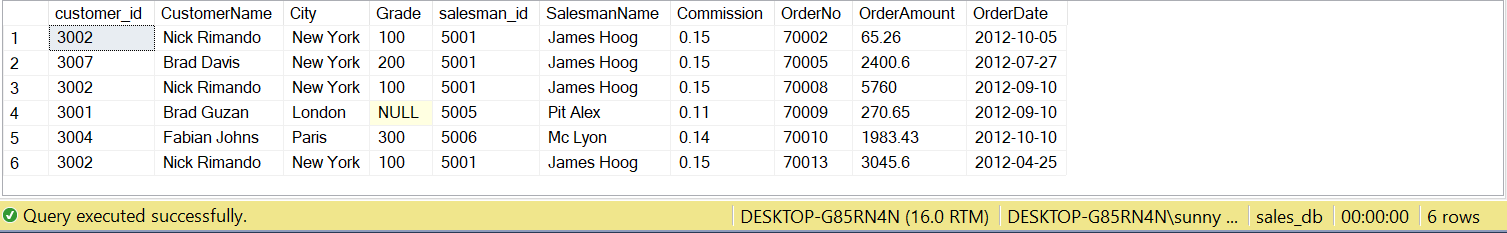
salesman.salesman\_id, salesman.salesman\_name SalesmanName, salesman.commission Commission,

ord.ord\_no OrderNo, ord.purch\_amount OrderAmount, ord.ord\_date OrderDate

FROM tblCustomers customer, tblSalesman salesman, tblOrders ord

WHERE customer.customer\_id = ord.customer\_id AND ord.salesman\_id = salesman.salesman\_id AND customer.city = salesman.city

Output:



Q8) Write a SQL query to display the customer name, customer city, grade, salesman, salesman city. The results should be sorted by ascending customer\_id.

==>

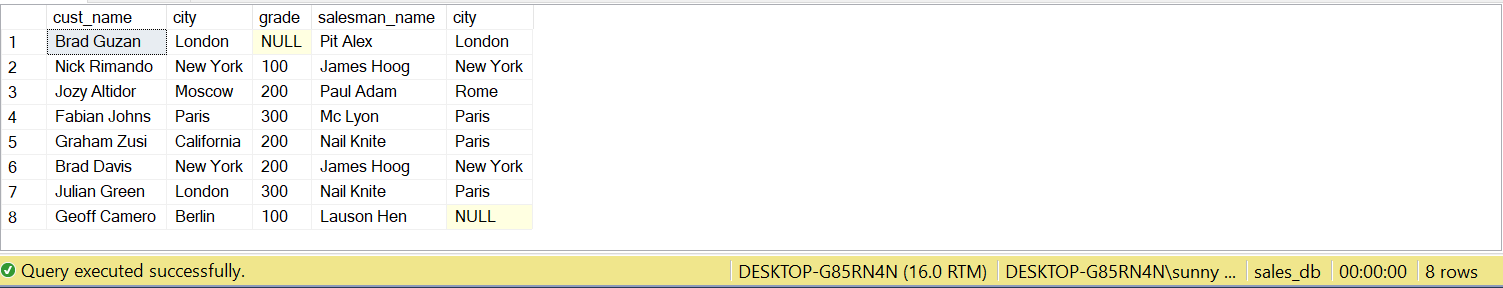
SELECT customer.cust\_name, customer.city, customer.grade, salesman.salesman\_name, salesman.city

FROM tblCustomers customer JOIN tblSalesman salesman

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

ORDER BY customer.customer\_id

Output:



Q9) Write a SQL query to find those customers with a grade less than 300. Return

cust\_name, customer city, grade, Salesman, salesman city. The result should be

ordered by ascending customer\_id.

==>

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

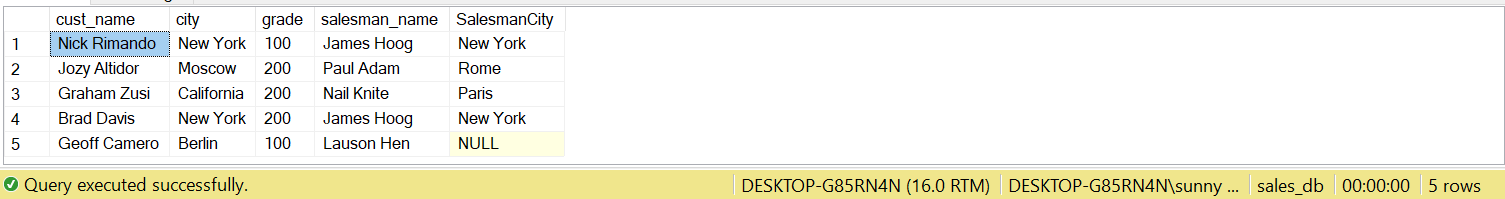
FROM tblCustomers customer JOIN tblSalesman salesman

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

WHERE customer.grade < 300

ORDER BY customer.customer\_id

Output:



Q10) 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 CustomerName, customer.city City, ord.ord\_no OrderNo, ord.ord\_date OrderDate, ord.purch\_amount OrderAmount

FROM tblCustomers customer JOIN tblOrders ord

ON customer.customer\_id = ord.customer\_id

ORDER BY ord.ord\_date

Output:



Q11) 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 CustomerName, customer.city City, ord.ord\_no OrderNo, ord.ord\_date OrderDate, ord.purch\_amount OrderAmount,

salesman.salesman\_name SalesmanName, salesman.commission Commission

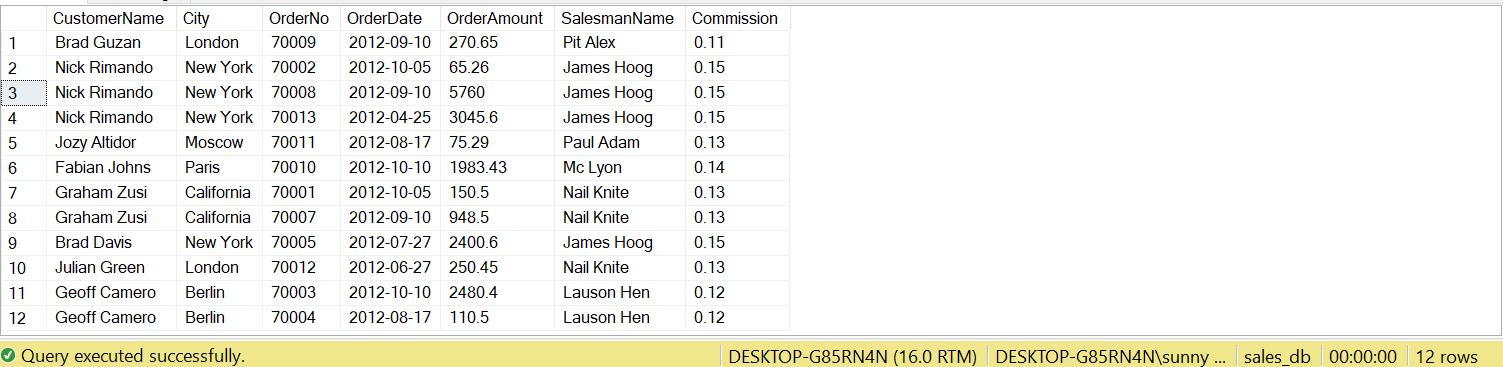
FROM tblCustomers customer LEFT JOIN tblOrders ord

ON customer.customer\_id = ord.customer\_id

LEFT JOIN tblSalesman salesman

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

Output:



Q12) 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.

==>

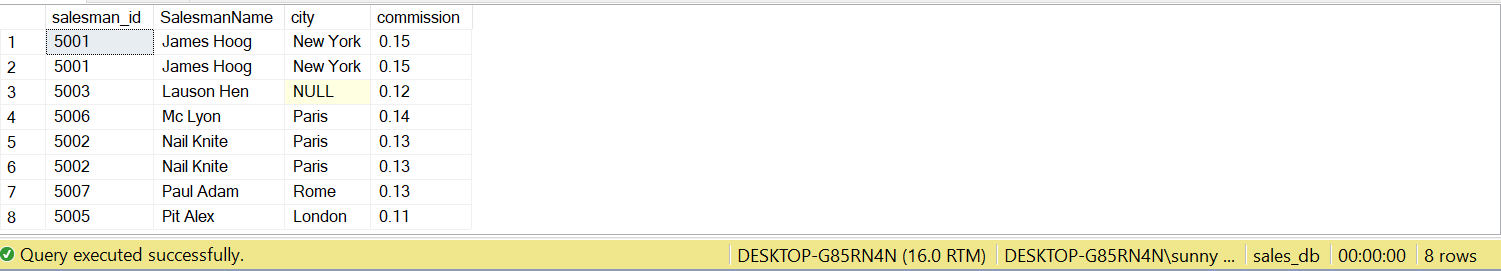
SELECT salesman.salesman\_id, salesman.salesman\_name SalesmanName, salesman.city, salesman.commission

FROM tblCustomers customer RIGHT JOIN tblSalesman salesman

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

ORDER BY salesman.salesman\_name

Output:



Q13) write a SQL query to list all salespersons along with customer name, city, grade, order number, date, and amount.

==>

SELECT salesman.salesman\_id, salesman.salesman\_name SalesmanName, customer.cust\_name CustomerName, customer.city, customer.grade

,ord.ord\_no OrderNo, ord.ord\_date OrderDate, ord.purch\_amount OrderAmount

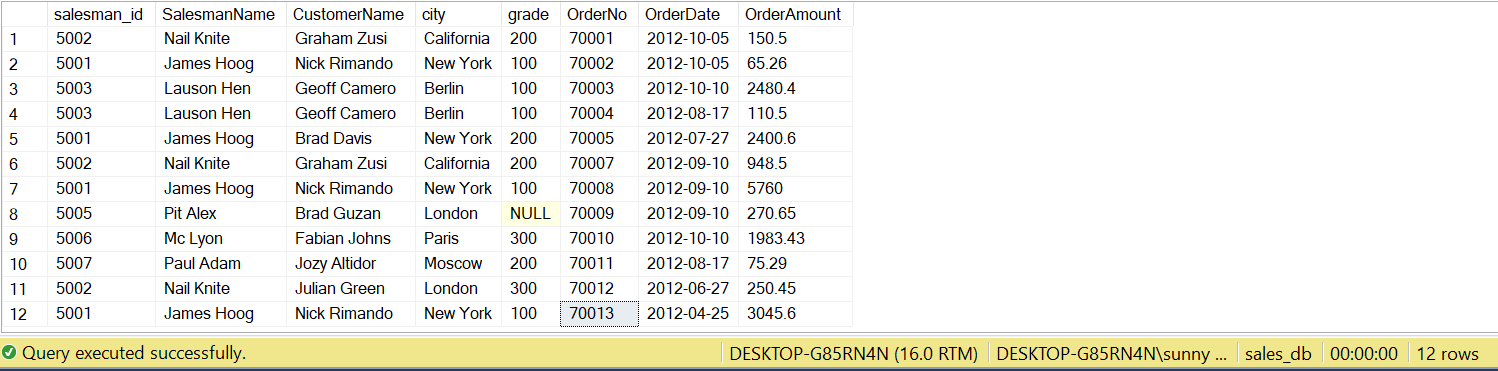
FROM tblSalesman salesman JOIN tblCustomers customer

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

JOIN tblOrders ord

ON ord.customer\_id = customer.customer\_id

Output:



Q14) 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 DISTINCT salesman.salesman\_id, salesman.salesman\_name SalesmanName

FROM tblSalesman salesman LEFT JOIN tblCustomers customer

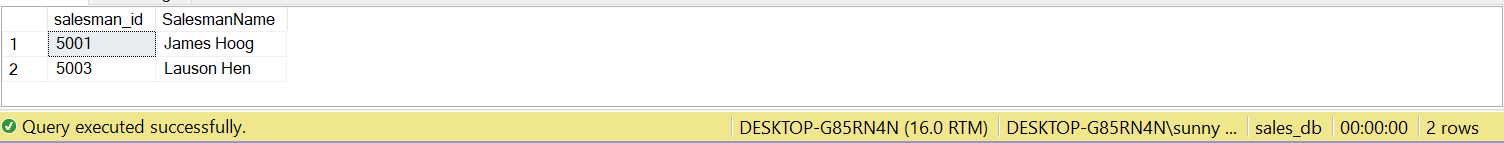
ON salesman.salesman\_id = customer.salesman\_id

LEFT JOIN tblOrders ord

ON ord.customer\_id = customer.customer\_id

WHERE ( ord.purch\_amount > 2000 AND customer.grade IS NOT NULL)

Output:



Q15) Same as Q14

Q16) 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.

==>

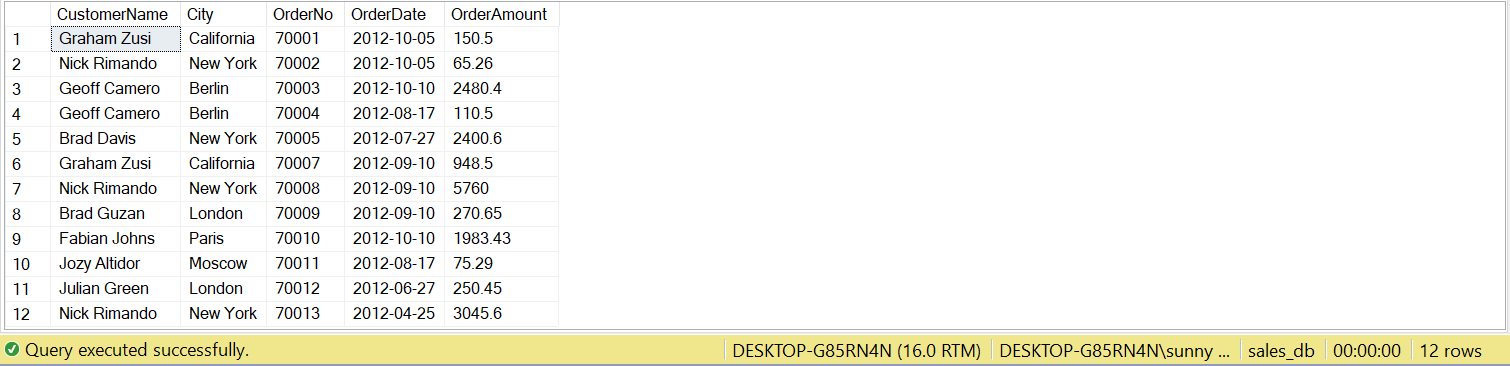
SELECT customer.cust\_name CustomerName, customer.city City, ord.ord\_no OrderNo, ord.ord\_date OrderDate,

ord.purch\_amount OrderAmount

FROM tblCustomers customer RIGHT JOIN tblOrders ord

ON customer.customer\_id = ord.customer\_id

Output:



Q17) Write a SQL query to combine each row of the salesman table with each row of the

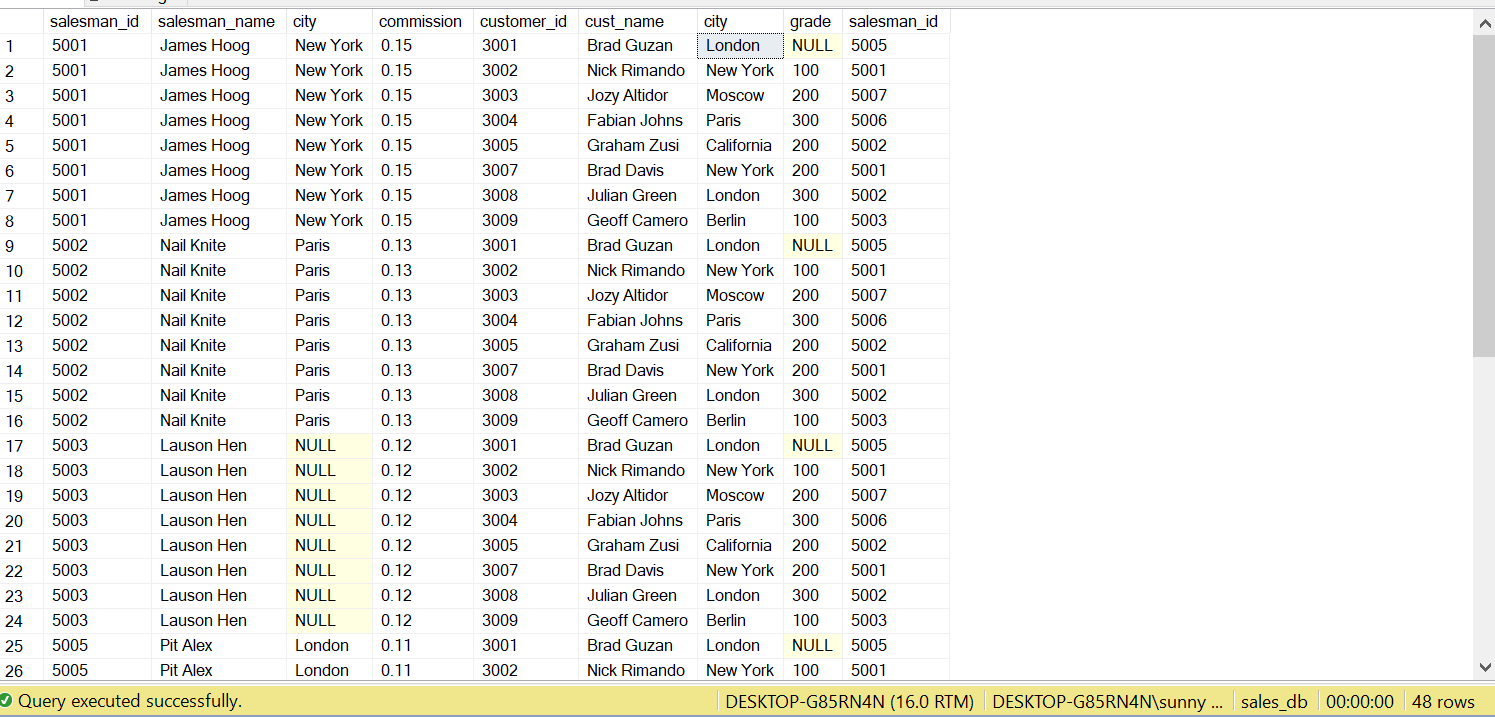
customer table.

==>

SELECT \*

FROM tblSalesman salesman CROSS JOIN tblCustomers customer

Output:



Q18) 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.

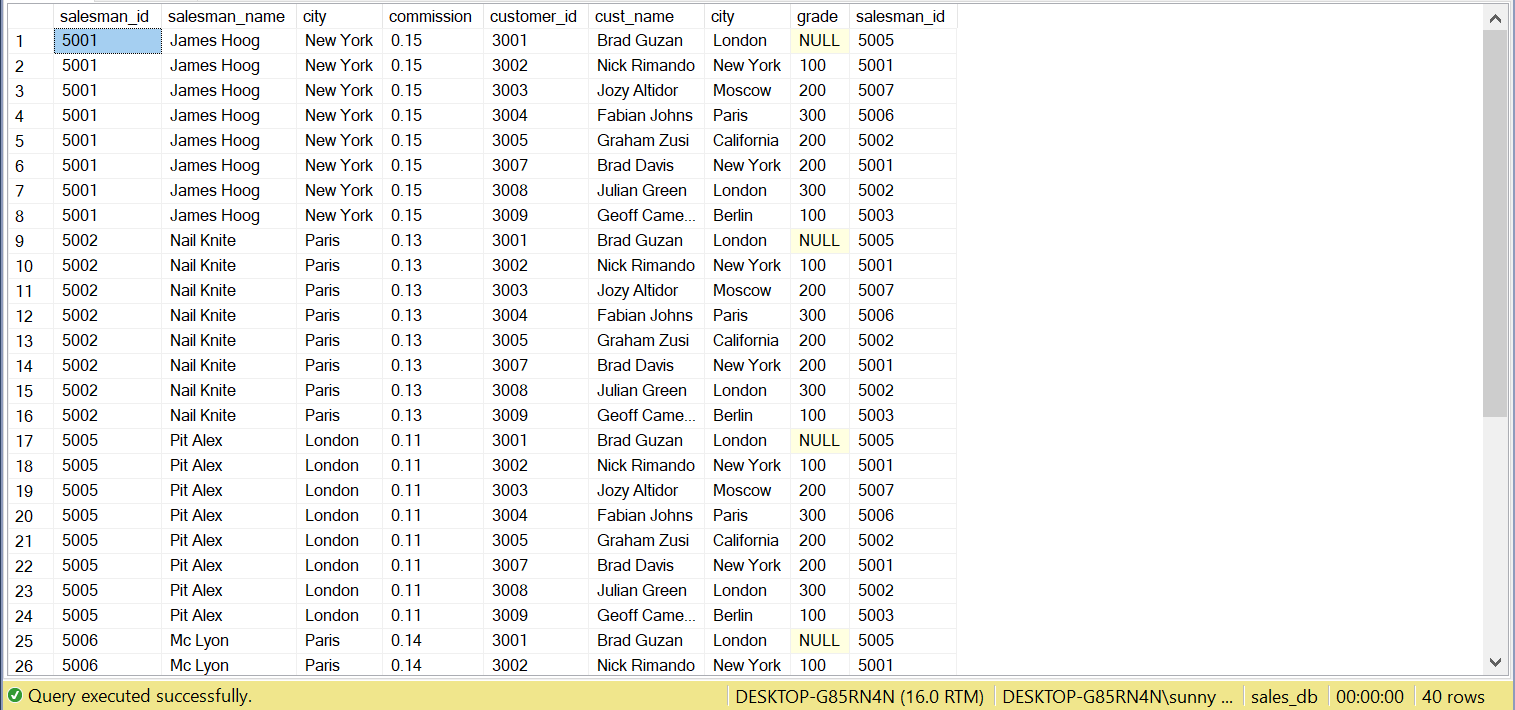
==>

SELECT \*

FROM tblSalesman salesman CROSS JOIN tblCustomers customer

WHERE salesman.city IS NOT NULL

Output:



Q19) 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 tblSalesman salesman CROSS JOIN tblCustomers customer

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

Output:



Q20) 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 tblSalesman salesman CROSS JOIN tblCustomers customer

WHERE (salesman.city IS NOT NULL AND salesman.city <> customer.city AND customer.grade IS NOT NULL)

Output:

