

### **Salesman**

```
create table salesman(  
salesman_id int primary key,  
name VARCHAR(50),  
city VARCHAR(50),  
commision decimal(4,2)  
)
```

```
insert into salesman (salesman_id, name, city, commision) values (5001 , 'James Hoog' , ' New York' , 0.15);
```

```
insert into salesman (salesman_id, name, city, commision) values ( 5002,' Nail Knit',' Paris ', 0.13);
```

```
insert into salesman (salesman_id, name, city, commision) values ( 5005 ,'Pit Alex ','London ', 0.11);
```

```
insert into salesman (salesman_id, name, city, commision) values (5006 ,'Mc Lyon ','Paris ', 0.14);
```

```
insert into salesman (salesman_id, name, city, commision) values (5007 ,'Paul Adam ','Rome ', 0.13);
```

```
insert into salesman (salesman_id, name, city, commision) values (5003 ,'Lauson Hen','San Jos', 0.12);
```

```
USE [assignment_tatvasoft]  
GO
```

```
USE [assignment_tatvasoft]  
GO
```

```
SELECT [salesman_id]  
      ,[name]  
      ,[city]  
      ,[commision]  
FROM [dbo].[salesman]  
GO
```

### **customer**

```
create table customer(  

```

```
customer_id int primary key,  
cust_name VARCHAR(50),  
city VARCHAR(50),
```

```
grade int,  
salesman_id int,
```

```
CONSTRAINT "FK_Orders_salesman" FOREIGN KEY  
(  
    "salesman_id"  
) REFERENCES "dbo"."salesman" (  
    "salesman_id"  
)  
)
```

```
insert into customer (customer_id, cust_name, city, grade,salesman_id) values (3002 , ' Nick  
Rimando ', ' New York ', 100 ,    5001);  
insert into customer (customer_id, cust_name, city, grade,salesman_id) values (3007 , ' Brad  
Davis ', ' New York ', 200 ,    5001);  
insert into customer (customer_id, cust_name, city, grade,salesman_id) values (3005 , ' Graham  
Zusi ', ' California', 200 ,    5002);  
insert into customer (customer_id, cust_name, city, grade,salesman_id) values (3008 , ' Julian  
Green ', ' London ', 300 ,    5002);  
insert into customer (customer_id, cust_name, city, grade,salesman_id) values (3004 , ' Fabian  
Johnson', ' Paris ', 300 ,    5006);  
insert into customer (customer_id, cust_name, city, grade,salesman_id) values (3009 , ' Geoff  
Cameron ', ' Berlin ', 100 ,    5003);  
insert into customer (customer_id, cust_name, city, grade,salesman_id) values (3003 , ' Jozy  
Altidor ', ' Moscow ', 200 ,    5007);  
insert into customer (customer_id, cust_name, city, grade,salesman_id) values (3001 , ' Brad  
Guzan ', ' London ', 100 ,    5005);
```

```
SELECT [customer_id]  
    ,[cust_name]  
    ,[city]  
    ,[grade]  
    ,[salesman_id]  
FROM [dbo].[customer]  
GO
```

### **Orders**

```
CREATE TABLE "Orders" (
```

```
    ord_no INT primary key,  
    purch_amt DECIMAL(6,2),  
    ord_date DATE,
```

```
customer_id INT,  
salesman_id INT
```

```
CONSTRAINT "FK_Orders_Customers" FOREIGN KEY  
(  
    "customer_id"  
) REFERENCES "dbo"."customer" (  
    "customer_id"  
)  
,  
CONSTRAINT "FK_salesman" FOREIGN KEY  
(  
    "salesman_id"  
) REFERENCES "dbo"."salesman" (  
    "salesman_id"  
)  
)  
GO
```

```
insert into orders (ord_no, purch_amt, ord_date, customer_id,salesman_id) values (70001 ,  
150.5 , '2012-10-05', 3005 , 5002);  
insert into orders (ord_no, purch_amt, ord_date, customer_id,salesman_id) values (70009 ,  
270.65 , '2012-09-10', 3001 , 5005);  
insert into orders (ord_no, purch_amt, ord_date, customer_id,salesman_id) values (70002 ,  
65.26 , '2012-10-05', 3002 , 5001);  
insert into orders (ord_no, purch_amt, ord_date, customer_id,salesman_id) values (70004 ,  
110.5 , '2012-08-17', 3009 , 5003);  
insert into orders (ord_no, purch_amt, ord_date, customer_id,salesman_id) values (70007 ,  
948.5 , '2012-09-10', 3005 , 5002);  
insert into orders (ord_no, purch_amt, ord_date, customer_id,salesman_id) values (70005 ,  
2400.6 , '2012-07-27', 3007 , 5001);  
insert into orders (ord_no, purch_amt, ord_date, customer_id,salesman_id) values (70008 ,  
5760 , '2012-09-10', 3002 , 5001);  
insert into orders (ord_no, purch_amt, ord_date, customer_id,salesman_id) values (70010 ,  
1983.43 , '2012-10-10', 3004 , 5006);  
insert into orders (ord_no, purch_amt, ord_date, customer_id,salesman_id) values (70003 ,  
2480.4 , '2012-10-10', 3009 , 5003);  
insert into orders (ord_no, purch_amt, ord_date, customer_id,salesman_id) values (70012 ,  
250.45 , '2012-06-27', 3008 , 5002);  
insert into orders (ord_no, purch_amt, ord_date, customer_id,salesman_id) values (70011 ,  
75.29 , '2012-08-17', 3003 , 5007);  
insert into orders (ord_no, purch_amt, ord_date, customer_id,salesman_id) values (70013 ,  
3045.6 , '2012-04-25', 3002 , 5001)
```

```

USE [assignment_tatvasoft]
GO
SELECT [ord_no]
      ,[purch_amt]
      ,[ord_date]
      ,[customer_id]
      ,[salesman_id]
FROM [dbo].[Orders]
GO

```

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

```

SELECT a.cust_name AS "Customer Name",
      a.city, b.name AS "Salesman", b.city as "bcity"
FROM customer a ,salesman b
where a.city=b.city;

```

1 %

Results Messages

	Customer Name	city	Salesman	bcity
1	Nick Rimando	New York	James Hoog	New York
2	Fabian Johnson	Paris	Nail Knit	Paris
3	Brad Davis	New York	James Hoog	New York

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 a.cust_name AS "Customer Name",
      a.city, b.ord_no, b.purch_amt
FROM orders b
INNER JOIN customer a
ON b.customer_id=a.customer_id
where b.purch_amt between 50.00 and 2000.00

```

91 %

Results Messages

	Customer Name	city	ord_no	purch_amt
1	Graham Zusi	California	70001	150.50
2	Nick Rimando	New York	70002	65.26
3	Geoff Cameron	Berlin	70004	110.50
4	Graham Zusi	California	70007	948.50
5	Brad Guzan	London	70009	270.65
6	Fabian Johnson	Paris	70010	1983.43
7	Jozy Altidor	Moscow	70011	75.29
8	Julian Green	London	70012	250.45

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

```
SELECT a.cust_name AS "Customer Name",  
a.city, b.name , b.commission  
FROM customer a  
INNER JOIN salesman b  
ON a.salesman_id=b.salesman_id;
```

91 %

Results Messages

	Customer Name	city	name	commission
1	Brad Guzan	London	Pit Alex	0.11
2	Nick Rimando	New York	James Hoog	0.15
3	Jozy Altidor	Moscow	Paul Adam	0.13
4	Fabian Johnson	Paris	Mc Lyon	0.14
5	Graham Zusi	California	Nail Knit	0.13
6	Brad Davis	New York	James Hoog	0.15
7	Julian Green	London	Nail Knit	0.13
8	Geoff Cameron	Berlin	Lauson Hen	0.12

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 a.cust_name AS "Customer Name",  
a.city, b.name, b.commission  
FROM customer a  
INNER JOIN salesman b  
ON a.salesman_id=b.salesman_id  
WHERE b.commission > 0.12;
```

91 %

Results Messages

	Customer Name	city	name	commission
1	Nick Rimando	New York	James Hoog	0.15
2	Jozy Altidor	Moscow	Paul Adam	0.13
3	Fabian Johnson	Paris	Mc Lyon	0.14
4	Graham Zusi	California	Nail Knit	0.13
5	Brad Davis	New York	James Hoog	0.15
6	Julian Green	London	Nail Knit	0.13

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 a.cust_name AS "Customer Name",
a.city, b.name, b.city, b.commission
FROM customer a
INNER JOIN salesman b
ON a.salesman_id=b.salesman_id
WHERE b.commission>.12
AND a.city<>b.city;

```

	Customer Name	city	name	city	commision
1	Jozy Altidor	Moscow	Paul Adam	Rome	0.13
2	Fabian Johnson	Paris	Mc Lyon	Paris	0.14
3	Graham Zusi	Califomia	Nail Knit	Paris	0.13
4	Julian Green	London	Nail Knit	Paris	0.13

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

```

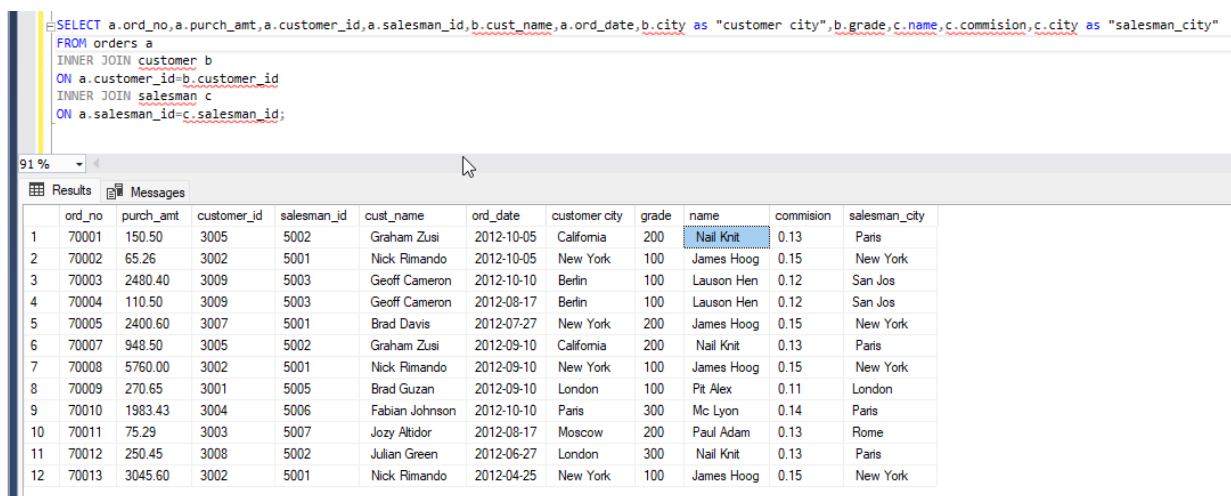
SELECT a.ord_no, a.ord_date, a.purch_amt,
b.cust_name AS "Customer Name", b.grade,
c.name, c.commission
FROM orders a
INNER JOIN customer b
ON a.customer_id=b.customer_id
INNER JOIN salesman c
ON a.salesman_id=c.salesman_id;

```

	ord_no	ord_date	purch_amt	Customer Name	grade	name	commision
1	70001	2012-10-05	150.50	Graham Zusi	200	Nail Knit	0.13
2	70002	2012-10-05	65.26	Nick Rimando	100	James Hoog	0.15
3	70003	2012-10-10	2480.40	Geoff Cameron	100	Lauson Hen	0.12
4	70004	2012-08-17	110.50	Geoff Cameron	100	Lauson Hen	0.12
5	70005	2012-07-27	2400.60	Brad Davis	200	James Hoog	0.15
6	70007	2012-09-10	948.50	Graham Zusi	200	Nail Knit	0.13
7	70008	2012-09-10	5760.00	Nick Rimando	100	James Hoog	0.15
8	70009	2012-09-10	270.65	Brad Guzan	100	Pit Alex	0.11
9	70010	2012-10-10	1983.43	Fabian Johnson	300	Mc Lyon	0.14
10	70011	2012-08-17	75.29	Jozy Altidor	200	Paul Adam	0.13
11	70012	2012-06-27	250.45	Julian Green	300	Nail Knit	0.13
12	70013	2012-04-25	3045.60	Nick Rimando	100	James Hoog	0.15

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 a.ord_no,a.purch_amt,a.customer_id,a.salesman_id,b.cust_name,a.ord_date,b.city as
"customer city",b.grade,c.name,c.commission,c.city as "salesman_city"
FROM orders a
INNER JOIN customer b
ON a.customer_id=b.customer_id
INNER JOIN salesman c
ON a.salesman_id=c.salesman_id;
```



The screenshot shows a SQL query execution interface. The query is displayed in a text area at the top, and the results are shown in a table below. The table has 12 rows and 11 columns. The columns are: ord\_no, purch\_amt, customer\_id, salesman\_id, cust\_name, ord\_date, customer city, grade, name, commission, and salesman\_city. The results are sorted by customer\_id in ascending order.

	ord_no	purch_amt	customer_id	salesman_id	cust_name	ord_date	customer city	grade	name	commission	salesman_city
1	70001	150.50	3005	5002	Graham Zusi	2012-10-05	California	200	Nail Knit	0.13	Paris
2	70002	65.26	3002	5001	Nick Rimando	2012-10-05	New York	100	James Hoog	0.15	New York
3	70003	2480.40	3009	5003	Geoff Cameron	2012-10-10	Berlin	100	Lauson Hen	0.12	San Jos
4	70004	110.50	3009	5003	Geoff Cameron	2012-08-17	Berlin	100	Lauson Hen	0.12	San Jos
5	70005	2400.60	3007	5001	Brad Davis	2012-07-27	New York	200	James Hoog	0.15	New York
6	70007	948.50	3005	5002	Graham Zusi	2012-09-10	California	200	Nail Knit	0.13	Paris
7	70008	5760.00	3002	5001	Nick Rimando	2012-09-10	New York	100	James Hoog	0.15	New York
8	70009	270.65	3001	5005	Brad Guzan	2012-09-10	London	100	Pit Alex	0.11	London
9	70010	1983.43	3004	5006	Fabian Johnson	2012-10-10	Paris	300	Mc Lyon	0.14	Paris
10	70011	75.29	3003	5007	Jozy Altidor	2012-08-17	Moscow	200	Paul Adam	0.13	Rome
11	70012	250.45	3008	5002	Julian Green	2012-06-27	London	300	Nail Knit	0.13	Paris
12	70013	3045.60	3002	5001	Nick Rimando	2012-04-25	New York	100	James Hoog	0.15	New York

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,customer.customer_id,
salesman.name as "salesman_name",salesman.city
FROM customer,salesman
where customer.salesman_id=salesman.salesman_id
ORDER BY customer.customer_id;
```



```

SELECT customer.cust_name, customer.city, customer.grade, customer.customer_id, salesman.name as "salesman_name", salesman.city
FROM customer, salesman
where customer.salesman_id=salesman.salesman_id
ORDER BY customer.customer_id;

```

91 %

Results Messages

	cust_name	city	grade	customer_id	salesman_name	city
1	Brad Guzan	London	100	3001	Pit Alex	London
2	Nick Rimando	New York	100	3002	James Hoog	New York
3	Jozy Altidor	Moscow	200	3003	Paul Adam	Rome
4	Fabian Johnson	Paris	300	3004	Mc Lyon	Paris
5	Graham Zusi	California	200	3005	Nail Knit	Paris
6	Brad Davis	New York	200	3007	James Hoog	New York
7	Julian Green	London	300	3008	Nail Knit	Paris
8	Geoff Cameron	Berlin	100	3009	Lauson Hen	San Jos

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 a.cust_name, a.city, a.grade,
b.name AS "Salesman", b.city
FROM customer a
LEFT OUTER JOIN salesman b
ON a.salesman_id=b.salesman_id
WHERE a.grade<300
ORDER BY a.customer_id;

```

91 %

Results Messages

	cust_name	city	grade	Salesman	city
1	Brad Guzan	London	100	Pit Alex	London
2	Nick Rimando	New York	100	James Hoog	New York
3	Jozy Altidor	Moscow	200	Paul Adam	Rome
4	Graham Zusi	California	200	Nail Knit	Paris
5	Brad Davis	New York	200	James Hoog	New York
6	Geoff Cameron	Berlin	100	Lauson Hen	San Jos

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 a.cust_name, a.city, b.ord_no,
       b.ord_date, b.purch_amt
FROM customer a
LEFT OUTER JOIN orders b
ON a.customer_id=b.customer_id
order by b.ord_date;
```

91 %

Results Messages

	cust_name	city	ord_no	ord_date	purch_amt
1	Nick Rimando	New York	70013	2012-04-25	3045.60
2	Julian Green	London	70012	2012-06-27	250.45
3	Brad Davis	New York	70005	2012-07-27	2400.60
4	Jozy Altidor	Moscow	70011	2012-08-17	75.29
5	Geoff Cameron	Berlin	70004	2012-08-17	110.50
6	Brad Guzan	London	70009	2012-09-10	270.65
7	Graham Zusi	California	70007	2012-09-10	948.50
8	Nick Rimando	New York	70008	2012-09-10	5760.00
9	Graham Zusi	California	70001	2012-10-05	150.50
10	Nick Rimando	New York	70002	2012-10-05	65.26
11	Fabian Johnson	Paris	70010	2012-10-10	1983.43
12	Geoff Cameron	Berlin	70003	2012-10-10	2480.40

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 a.cust_name,a.city, b.ord_no,
b.ord_date,b.purch_amt AS "Order Amount" ,c.name,c.commission
FROM customer a
LEFT OUTER JOIN orders b
ON a.customer_id=b.customer_id
LEFT OUTER JOIN salesman c
ON c.salesman_id=b.salesman_id
```

91 %

Results Messages

	cust_name	city	ord_no	ord_date	Order Amount	name	commision
1	Brad Guzan	London	70009	2012-09-10	270.65	Pit Alex	0.11
2	Nick Rimando	New York	70002	2012-10-05	65.26	James Hoog	0.15
3	Nick Rimando	New York	70008	2012-09-10	5760.00	James Hoog	0.15
4	Nick Rimando	New York	70013	2012-04-25	3045.60	James Hoog	0.15
5	Jozy Altidor	Moscow	70011	2012-08-17	75.29	Paul Adam	0.13
6	Fabian Johnson	Paris	70010	2012-10-10	1983.43	Mc Lyon	0.14
7	Graham Zusi	California	70001	2012-10-05	150.50	Nail Knit	0.13
8	Graham Zusi	California	70007	2012-09-10	948.50	Nail Knit	0.13
9	Brad Davis	New York	70005	2012-07-27	2400.60	James Hoog	0.15
10	Julian Green	London	70012	2012-06-27	250.45	Nail Knit	0.13
11	Geoff Cameron	Berlin	70003	2012-10-10	2480.40	Lauson Hen	0.12
12	Geoff Cameron	Berlin	70004	2012-08-17	110.50	Lauson Hen	0.12

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 a.cust_name, a.city, a.grade,  
       b.name AS "Salesman", b.city, b.salesman_id  
FROM customer a  
RIGHT OUTER JOIN salesman b  
ON b.salesman_id=a.salesman_id  
ORDER BY b.salesman_id;
```

91 %

Results

Messages

	cust_name	city	grade	Salesman	city	salesman_id
1	Nick Rimando	New York	100	James Hoog	New York	5001
2	Brad Davis	New York	200	James Hoog	New York	5001
3	Graham Zusi	California	200	Nail Knit	Paris	5002
4	Julian Green	London	300	Nail Knit	Paris	5002
5	Geoff Cameron	Berlin	100	Lauson Hen	San Jos	5003
6	Brad Guzan	London	100	Pit Alex	London	5005
7	Fabian Johnson	Paris	300	Mc Lyon	Paris	5006
8	Jozy Altidor	Moscow	200	Paul Adam	Rome	5007

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

```
SELECT a.cust_name, a.city, a.grade,
       b.name AS "Salesman",
       c.ord_no, c.ord_date, c.purch_amt
FROM customer a
RIGHT OUTER JOIN salesman b
ON b.salesman_id=a.salesman_id
RIGHT OUTER JOIN orders c
ON c.customer_id=a.customer_id;
```

91 %

	cust_name	city	grade	Salesman	ord_no	ord_date	purch_amt
1	Graham Zusi	California	200	Nail Knit	70001	2012-10-05	150.50
2	Nick Rimando	New York	100	James Hoog	70002	2012-10-05	65.26
3	Geoff Cameron	Berlin	100	Lauson Hen	70003	2012-10-10	2480.40
4	Geoff Cameron	Berlin	100	Lauson Hen	70004	2012-08-17	110.50
5	Brad Davis	New York	200	James Hoog	70005	2012-07-27	2400.60
6	Graham Zusi	California	200	Nail Knit	70007	2012-09-10	948.50
7	Nick Rimando	New York	100	James Hoog	70008	2012-09-10	5760.00
8	Brad Guzan	London	100	Pit Alex	70009	2012-09-10	270.65
9	Fabian Johnson	Paris	300	Mc Lyon	70010	2012-10-10	1983.43
10	Jozy Altidor	Moscow	200	Paul Adam	70011	2012-08-17	75.29
11	Julian Green	London	300	Nail Knit	70012	2012-06-27	250.45
12	Nick Rimando	New York	100	James Hoog	70013	2012-04-25	3045.60

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 a.cust_name, a.city, a.grade,
       b.name AS "Salesman",
       c.ord_no, c.ord_date, c.purch_amt
FROM customer a
RIGHT OUTER JOIN salesman b
ON b.salesman_id=a.salesman_id
LEFT OUTER JOIN orders c
ON c.customer_id=a.customer_id
WHERE c.purch_amt >= 2000
AND a.grade IS NOT NULL;

```

1 %

Results Messages

	cust_name	city	grade	Salesman	ord_no	ord_date	purch_amt
1	Geoff Cameron	Berlin	100	Lauson Hen	70003	2012-10-10	2480.40
2	Brad Davis	New York	200	James Hoog	70005	2012-07-27	2400.60
3	Nick Rimando	New York	100	James Hoog	70008	2012-09-10	5760.00
4	Nick Rimando	New York	100	James Hoog	70013	2012-04-25	3045.60

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 a.cust_name, a.city, a.grade,
       b.name AS "Salesman",
       c.ord_no, c.ord_date, c.purch_amt
FROM customer a
RIGHT OUTER JOIN salesman b
ON b.salesman_id=a.salesman_id
LEFT OUTER JOIN orders c
ON c.customer_id=a.customer_id
WHERE c.purch_amt >= 2000
AND a.grade IS NOT NULL;

```

1 %

Results Messages

	cust_name	city	grade	Salesman	ord_no	ord_date	purch_amt
1	Geoff Cameron	Berlin	100	Lauson Hen	70003	2012-10-10	2480.40
2	Brad Davis	New York	200	James Hoog	70005	2012-07-27	2400.60
3	Nick Rimando	New York	100	James Hoog	70008	2012-09-10	5760.00
4	Nick Rimando	New York	100	James Hoog	70013	2012-04-25	3045.60

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.

SQL Query:

```
SELECT a.cust_name, a.city, b.ord_no,
       b.ord_date, b.purch_amt
FROM customer a
FULL OUTER JOIN orders b
ON a.customer_id=b.customer_id
WHERE a.grade IS NOT NULL;
```

The multi-part identifier "a.city" could not be bound.

91 %

Results Messages

	cust_name	city	ord_no	ord_date	Order Amount
1	Brad Guzan	London	70009	2012-09-10	270.65
2	Nick Rimando	New York	70002	2012-10-05	65.26
3	Nick Rimando	New York	70008	2012-09-10	5760.00
4	Nick Rimando	New York	70013	2012-04-25	3045.60
5	Jozy Altidor	Moscow	70011	2012-08-17	75.29
6	Fabian Johnson	Paris	70010	2012-10-10	1983.43
7	Graham Zusi	California	70001	2012-10-05	150.50
8	Graham Zusi	California	70007	2012-09-10	948.50
9	Brad Davis	New York	70005	2012-07-27	2400.60
10	Julian Green	London	70012	2012-06-27	250.45
11	Geoff Cameron	Berlin	70003	2012-10-10	2480.40
12	Geoff Cameron	Berlin	70004	2012-08-17	110.50



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

SQL query: SELECT \* FROM salesman a CROSS JOIN customer b;

91 %

Results Messages

	salesman_id	name	city	commision	customer_id	cust_name	city	grade	salesman_id
1	5001	James Hoog	New York	0.15	3001	Brad Guzan	London	100	5005
2	5001	James Hoog	New York	0.15	3002	Nick Rimando	New York	100	5001
3	5001	James Hoog	New York	0.15	3003	Jozy Altidor	Moscow	200	5007
4	5001	James Hoog	New York	0.15	3004	Fabian Johnson	Paris	300	5006
5	5001	James Hoog	New York	0.15	3005	Graham Zusi	California	200	5002
6	5001	James Hoog	New York	0.15	3007	Brad Davis	New York	200	5001
7	5001	James Hoog	New York	0.15	3008	Julian Green	London	300	5002
8	5001	James Hoog	New York	0.15	3009	Geoff Cameron	Berlin	100	5003
9	5002	Nail Knit	Paris	0.13	3001	Brad Guzan	London	100	5005
10	5002	Nail Knit	Paris	0.13	3002	Nick Rimando	New York	100	5001
11	5002	Nail Knit	Paris	0.13	3003	Jozy Altidor	Moscow	200	5007
12	5002	Nail Knit	Paris	0.13	3004	Fabian Johnson	Paris	300	5006
13	5002	Nail Knit	Paris	0.13	3005	Graham Zusi	California	200	5002
14	5002	Nail Knit	Paris	0.13	3007	Brad Davis	New York	200	5001
15	5002	Nail Knit	Paris	0.13	3008	Julian Green	London	300	5002
16	5002	Nail Knit	Paris	0.13	3009	Geoff Cameron	Berlin	100	5003
17	5003	Lauson Hen	San Jos	0.12	3001	Brad Guzan	London	100	5005
18	5003	Lauson Hen	San Jos	0.12	3002	Nick Rimando	New York	100	5001
19	5003	Lauson Hen	San Jos	0.12	3003	Jozy Altidor	Moscow	200	5007
20	5003	Lauson Hen	San Jos	0.12	3004	Fabian Johnson	Paris	300	5006
21	5003	Lauson Hen	San Jos	0.12	3005	Graham Zusi	California	200	5002
22	5003	Lauson Hen	San Jos	0.12	3007	Brad Davis	New York	200	5001
23	5003	Lauson Hen	San Jos	0.12	3008	Julian Green	London	300	5002
24	5003	Lauson Hen	San Jos	0.12	3009	Geoff Cameron	Berlin	100	5003
25	5005	Pit Alex	London	0.11	3001	Brad Guzan	London	100	5005
26	5005	Pit Alex	London	0.11	3002	Nick Rimando	New York	100	5001
27	5005	Pit Alex	London	0.11	3003	Jozy Altidor	Moscow	200	5007
28	5005	Pit Alex	London	0.11	3004	Fabian Johnson	Paris	300	5006

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 a
CROSS JOIN customer b
WHERE a.city IS NOT NULL;
```

91 %

Results Messages

	salesman_id	name	city	commision	customer_id	cust_name	city	grade	salesman_id
8	5001	James Hoog	New York	0.15	3009	Geoff Cameron	Berlin	100	5003
9	5002	Nail Knit	Paris	0.13	3001	Brad Guzan	London	100	5005
10	5002	Nail Knit	Paris	0.13	3002	Nick Rimando	New York	100	5001
11	5002	Nail Knit	Paris	0.13	3003	Jozy Altidor	Moscow	200	5007
12	5002	Nail Knit	Paris	0.13	3004	Fabian Johnson	Paris	300	5006
13	5002	Nail Knit	Paris	0.13	3005	Graham Zusi	California	200	5002
14	5002	Nail Knit	Paris	0.13	3007	Brad Davis	New York	200	5001
15	5002	Nail Knit	Paris	0.13	3008	Julian Green	London	300	5002
16	5002	Nail Knit	Paris	0.13	3009	Geoff Cameron	Berlin	100	5003
17	5003	Lauson Hen	San Jos	0.12	3001	Brad Guzan	London	100	5005
18	5003	Lauson Hen	San Jos	0.12	3002	Nick Rimando	New York	100	5001
19	5003	Lauson Hen	San Jos	0.12	3003	Jozy Altidor	Moscow	200	5007
20	5003	Lauson Hen	San Jos	0.12	3004	Fabian Johnson	Paris	300	5006
21	5003	Lauson Hen	San Jos	0.12	3005	Graham Zusi	California	200	5002
22	5003	Lauson Hen	San Jos	0.12	3007	Brad Davis	New York	200	5001
23	5003	Lauson Hen	San Jos	0.12	3008	Julian Green	London	300	5002
24	5003	Lauson Hen	San Jos	0.12	3009	Geoff Cameron	Berlin	100	5003
25	5005	Pit Alex	London	0.11	3001	Brad Guzan	London	100	5005
26	5005	Pit Alex	London	0.11	3002	Nick Rimando	New York	100	5001
27	5005	Pit Alex	London	0.11	3003	Jozy Altidor	Moscow	200	5007
28	5005	Pit Alex	London	0.11	3004	Fabian Johnson	Paris	300	5006
29	5005	Pit Alex	London	0.11	3005	Graham Zusi	California	200	5002
30	5005	Pit Alex	London	0.11	3007	Brad Davis	New York	200	5001
31	5005	Pit Alex	London	0.11	3008	Julian Green	London	300	5002
32	5005	Pit Alex	London	0.11	3009	Geoff Cameron	Berlin	100	5003
33	5006	Mc Lyon	Paris	0.14	3001	Brad Guzan	London	100	5005
34	5006	Mc Lyon	Paris	0.14	3002	Nick Rimando	New York	100	5001
35	5006	Mc Lyon	Paris	0.14	3003	Jozy Altidor	Moscow	200	5007
36	5006	Mc Lyon	Paris	0.14	3004	Fabian Johnson	Paris	300	5006
37	5006	Mc Lyon	Paris	0.14	3005	Graham Zusi	California	200	5002
38	5006	Mc Lyon	Paris	0.14	3007	Brad Davis	New York	200	5001
39	5006	Mc Lyon	Paris	0.14	3008	Julian Green	London	300	5002
40	5006	Mc Lyon	Paris	0.14	3009	Geoff Cameron	Berlin	100	5003
41	5007	Paul Adam	Rome	0.13	3001	Brad Guzan	London	100	5005
42	5007	Paul Adam	Rome	0.13	3002	Nick Rimando	New York	100	5001

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 a
CROSS JOIN customer b
WHERE a.city IS NOT NULL
AND b.grade IS NOT NULL;

```

91 %

Results Messages

	salesman_id	name	city	commision	customer_id	cust_name	city	grade	salesman_id
1	5001	James Hoog	New York	0.15	3001	Brad Guzan	London	100	5005
2	5001	James Hoog	New York	0.15	3002	Nick Rimando	New York	100	5001
3	5001	James Hoog	New York	0.15	3003	Jozy Altidor	Moscow	200	5007
4	5001	James Hoog	New York	0.15	3004	Fabian Johnson	Paris	300	5006
5	5001	James Hoog	New York	0.15	3005	Graham Zusi	Califomia	200	5002
6	5001	James Hoog	New York	0.15	3007	Brad Davis	New York	200	5001
7	5001	James Hoog	New York	0.15	3008	Julian Green	London	300	5002
8	5001	James Hoog	New York	0.15	3009	Geoff Cameron	Berlin	100	5003
9	5002	Nail Knit	Paris	0.13	3001	Brad Guzan	London	100	5005
10	5002	Nail Knit	Paris	0.13	3002	Nick Rimando	New York	100	5001
11	5002	Nail Knit	Paris	0.13	3003	Jozy Altidor	Moscow	200	5007
12	5002	Nail Knit	Paris	0.13	3004	Fabian Johnson	Paris	300	5006
13	5002	Nail Knit	Paris	0.13	3005	Graham Zusi	Califomia	200	5002
14	5002	Nail Knit	Paris	0.13	3007	Brad Davis	New York	200	5001
15	5002	Nail Knit	Paris	0.13	3008	Julian Green	London	300	5002
16	5002	Nail Knit	Paris	0.13	3009	Geoff Cameron	Berlin	100	5003
17	5003	Lauson Hen	San Jos	0.12	3001	Brad Guzan	London	100	5005
18	5003	Lauson Hen	San Jos	0.12	3002	Nick Rimando	New York	100	5001
19	5003	Lauson Hen	San Jos	0.12	3003	Jozy Altidor	Moscow	200	5007
20	5003	Lauson Hen	San Jos	0.12	3004	Fabian Johnson	Paris	300	5006
21	5003	Lauson Hen	San Jos	0.12	3005	Graham Zusi	Califomia	200	5002
22	5003	Lauson Hen	San Jos	0.12	3007	Brad Davis	New York	200	5001
23	5003	Lauson Hen	San Jos	0.12	3008	Julian Green	London	300	5002
24	5003	Lauson Hen	San Jos	0.12	3009	Geoff Cameron	Berlin	100	5003
25	5005	Pit Alex	London	0.11	3001	Brad Guzan	London	100	5005
26	5005	Pit Alex	London	0.11	3002	Nick Rimando	New York	100	5001
27	5005	Pit Alex	London	0.11	3003	Jozy Altidor	Moscow	200	5007
28	5005	Pit Alex	London	0.11	3004	Fabian Johnson	Paris	300	5006
29	5005	Pit Alex	London	0.11	3005	Graham Zusi	Califomia	200	5002

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 a
CROSS JOIN customer b
WHERE a.city IS NOT NULL
AND b.grade IS NOT NULL
AND a.city<>b.city;
```

	salesman_id	name	city	commision	customer_id	cust_name	city	grade	salesman_id
1	5001	James Hoog	New York	0.15	3001	Brad Guzan	London	100	5005
2	5001	James Hoog	New York	0.15	3003	Jozy Altidor	Moscow	200	5007
3	5001	James Hoog	New York	0.15	3004	Fabian Johnson	Paris	300	5006
4	5001	James Hoog	New York	0.15	3005	Graham Zusi	California	200	5002
5	5001	James Hoog	New York	0.15	3008	Julian Green	London	300	5002
6	5001	James Hoog	New York	0.15	3009	Geoff Cameron	Berlin	100	5003
7	5002	Nail Knit	Paris	0.13	3001	Brad Guzan	London	100	5005
8	5002	Nail Knit	Paris	0.13	3002	Nick Rimando	New York	100	5001
9	5002	Nail Knit	Paris	0.13	3003	Jozy Altidor	Moscow	200	5007
10	5002	Nail Knit	Paris	0.13	3005	Graham Zusi	California	200	5002
11	5002	Nail Knit	Paris	0.13	3007	Brad Davis	New York	200	5001
12	5002	Nail Knit	Paris	0.13	3008	Julian Green	London	300	5002
13	5002	Nail Knit	Paris	0.13	3009	Geoff Cameron	Berlin	100	5003
14	5003	Lauson Hen	San Jos	0.12	3001	Brad Guzan	London	100	5005
15	5003	Lauson Hen	San Jos	0.12	3002	Nick Rimando	New York	100	5001
16	5003	Lauson Hen	San Jos	0.12	3003	Jozy Altidor	Moscow	200	5007
17	5003	Lauson Hen	San Jos	0.12	3004	Fabian Johnson	Paris	300	5006
18	5003	Lauson Hen	San Jos	0.12	3005	Graham Zusi	California	200	5002
19	5003	Lauson Hen	San Jos	0.12	3007	Brad Davis	New York	200	5001
20	5003	Lauson Hen	San Jos	0.12	3008	Julian Green	London	300	5002
21	5003	Lauson Hen	San Jos	0.12	3009	Geoff Cameron	Berlin	100	5003
22	5005	Pit Alex	London	0.11	3001	Brad Guzan	London	100	5005
23	5005	Pit Alex	London	0.11	3002	Nick Rimando	New York	100	5001
24	5005	Pit Alex	London	0.11	3003	Jozy Altidor	Moscow	200	5007
25	5005	Pit Alex	London	0.11	3004	Fabian Johnson	Paris	300	5006
26	5005	Pit Alex	London	0.11	3005	Graham Zusi	California	200	5002
27	5005	Pit Alex	London	0.11	3007	Brad Davis	New York	200	5001
28	5005	Pit Alex	London	0.11	3008	Julian Green	London	300	5002
29	5005	Pit Alex	London	0.11	3009	Geoff Cameron	Berlin	100	5003
30	5006	Mc Lyon	Paris	0.14	3001	Brad Guzan	London	100	5005
31	5006	Mc Lyon	Paris	0.14	3002	Nick Rimando	New York	100	5001
32	5006	Mc Lyon	Paris	0.14	3003	Jozy Altidor	Moscow	200	5007