CSLR 51 : DBMS LAB-3

Roll no. : **106119100**

Name : **Rajneesh Pandey**

Section : **CSE-B**

PROBLEM 1

1) Given three tables, perform the following queries using joins:

Customer Table :

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

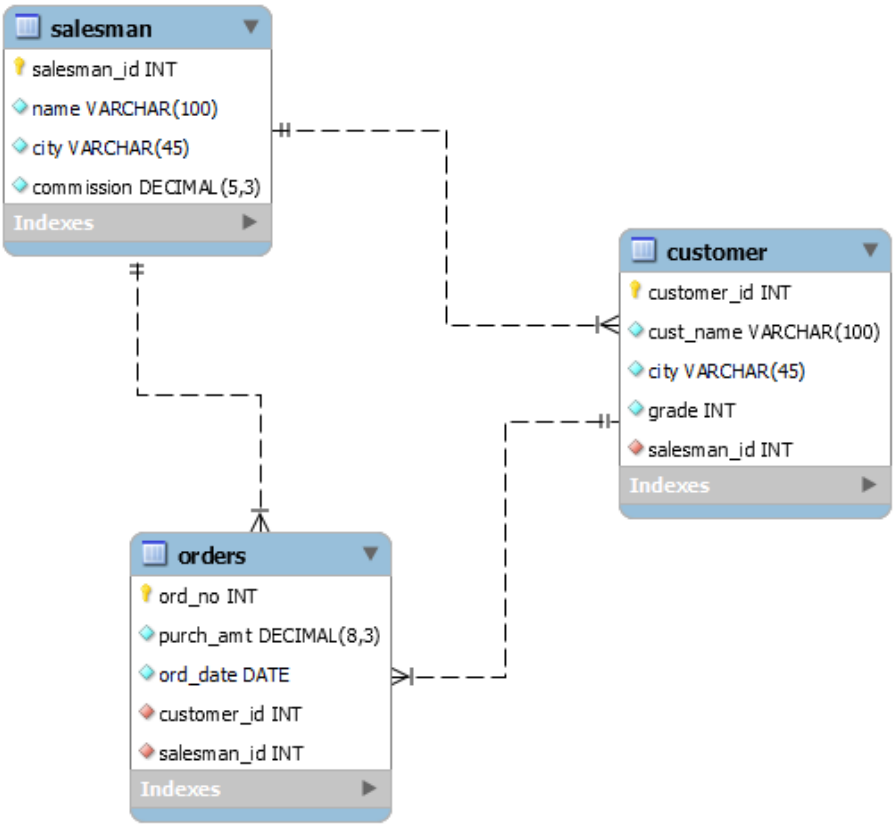
Salesman Table :

salesman_id	name	city	commission
5001	James Hoog	New York	0.15
5002	Nail Knite	Paris	0.13
5005	Pit Alex	London	0.11
5006	Mc Lyon	Paris	0.14
5007	Paul Adam	Rome	0.13
5003	Lauson Hen	San Jose	0.12

Orders Table :

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

ER Diagram :



```sql
CREATE DATABASE salesDB;

CREATE TABLE salesman (
  `salesman_id` INT NOT NULL,
  `name` VARCHAR(100) NOT NULL,
  `city` VARCHAR(45) NOT NULL,
  `commission` DECIMAL(5,3) NOT NULL,
  PRIMARY KEY (`salesman_id`)
  );

CREATE TABLE customer (
  `customer_id` INT NOT NULL,
  `cust_name` VARCHAR(100) NOT NULL,
  `city` VARCHAR(45) NOT NULL,
  `grade` INT NOT NULL,
  `salesman_id` INT NOT NULL,
  PRIMARY KEY (`customer_id`),
  INDEX `salesman_id_idx` (`salesman_id` ASC) VISIBLE,
  CONSTRAINT `salesman_id`
    FOREIGN KEY (`salesman_id`)
    REFERENCES `salesDB`.`salesman` (`salesman_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION
    );

CREATE TABLE orders(
  `ord_no` INT NOT NULL,
  `purch_amt` DECIMAL(8,3) NOT NULL,
  `ord_date` DATE NOT NULL,
  `customer_id` INT NOT NULL,
  `salesman_id` INT NOT NULL,
    PRIMARY KEY (`ord_no`),
    FOREIGN KEY (`customer_id`)
    REFERENCES `salesDB`.`customer` (`customer_id`),
    FOREIGN KEY (`salesman_id`)
    REFERENCES `salesDB`.`salesman` (`salesman_id`)
    );
```

```sql
INSERT INTO salesman (`salesman_id`, `name`, `city`, `commission`)
VALUES
       ('5001', 'James Hoog', 'New York', '0.15'),
       ('5002', 'Nail Knite', 'Paris', '0.13'),
       ('5005', 'Pit Alex', 'London', '0.11'),
       ('5006', 'Mc Lyon', 'Paris', '0.14'),
       ('5007', 'Paul Adam', 'Rome', '0.13'),
       ('5003', 'Lauson Hen', 'San Jose', '0.12');

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


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

Query :-

Show Tables :

```
SELECT * FROM salesman;
```

```
MySQL  localhost:3306 ssl  SQL > use salesDB
Default schema set to `salesDB`.
Fetching table and column names from `salesdb` for auto-completion... Press ^C to stop.
MySQL  localhost:3306 ssl  salesdb  SQL >
MySQL  localhost:3306 ssl  salesdb  SQL > SELECT * FROM salesman;
+-------------+------------+----------+------------+
| salesman_id | name       | city     | commission |
+-------------+------------+----------+------------+
|        5001 | James Hoog | New York |      0.150 |
|        5002 | Nail Knite | Paris    |      0.130 |
|        5003 | Lauson Hen | San Jose |      0.120 |
|        5005 | Pit Alex   | London   |      0.110 |
|        5006 | Mc Lyon    | Paris    |      0.140 |
|        5007 | Paul Adam  | Rome     |      0.130 |
+-------------+------------+----------+------------+
6 rows in set (0.0020 sec)
```

```
SELECT * FROM customer;
```

```
MySQL  localhost:3306 ssl  salesdb  SQL > SELECT * FROM customer;
+-------------+----------------+------------+-------+-------------+
| customer_id | cust_name      | city       | grade | salesman_id |
+-------------+----------------+------------+-------+-------------+
|        3001 | Brad Guzan     | London     |   200 |        5005 |
|        3002 | Nick Rimando   | New York   |   100 |        5001 |
|        3003 | Jozy Altidor   | Moscow     |   200 |        5007 |
|        3004 | Fabian Johnson | Paris      |   300 |        5006 |
|        3005 | Graham Zusi    | California |   200 |        5002 |
|        3007 | Brad Davis     | New York   |   200 |        5001 |
|        3008 | Julian Green   | London     |   300 |        5002 |
|        3009 | Geoff Cameroon | Berlin     |   100 |        5003 |
+-------------+----------------+------------+-------+-------------+
8 rows in set (0.0123 sec)
```

```sql
SELECT * FROM orders;
```

```
MySQL localhost:3306 ssl salesdb SQL > SELECT * FROM orders;
+--------+-----------+------------+-------------+-------------+
| ord_no | purch_amt | ord_date   | customer_id | salesman_id |
+--------+-----------+------------+-------------+-------------+
|  70001 |   150.500 | 2012-10-05 |        3005 |        5002 |
|  70002 |    65.260 | 2012-10-05 |        3002 |        5001 |
|  70003 |  2480.400 | 2012-10-10 |        3009 |        5003 |
|  70004 |   110.500 | 2012-08-17 |        3009 |        5003 |
|  70005 |  2400.600 | 2012-07-27 |        3007 |        5001 |
|  70007 |   948.500 | 2012-09-10 |        3005 |        5002 |
|  70008 |  5760.000 | 2012-09-10 |        3002 |        5001 |
|  70009 |   270.650 | 2012-09-10 |        3001 |        5005 |
|  70010 |  1983.430 | 2012-10-10 |        3004 |        5006 |
|  70011 |    75.290 | 2012-08-17 |        3003 |        5007 |
|  70012 |   250.450 | 2012-06-27 |        3008 |        5002 |
|  70013 |  3045.600 | 2012-04-25 |        3002 |        5001 |
+--------+-----------+------------+-------------+-------------+
12 rows in set (0.0089 sec)
```

a) Write a SQL query to find those salespersons who received a commission from the company more than 12%.Return Customer Name,customer city, Salesman, commission (Use Inner join)

```sql
SELECT c.cust_name AS "Customer 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;
```

```
MySQL localhost:3306 ssl salesdb SQL > SELECT c.cust_name AS "Customer 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;
+----------------+------------+------------+------------+
| Customer Name  | city       | Salesman   | commission |
+----------------+------------+------------+------------+
| Nick Rimando   | New York   | James Hoog |      0.150 |
| Brad Davis     | New York   | James Hoog |      0.150 |
| Graham Zusi    | California | Nail Knite |      0.130 |
| Julian Green   | London     | Nail Knite |      0.130 |
| Fabian Johnson | Paris      | Mc Lyon    |      0.140 |
| Jozy Altidor   | Moscow     | Paul Adam  |      0.130 |
+----------------+------------+------------+------------+
6 rows in set (0.0017 sec)
```

b) 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 find that either any of the existing customers have placed no order or placed one or more orders. (Use left outer join)

```
SELECT c.cust_name AS "Customer Name", c.city, o.ord_no, o.ord_date
, o.purch_amt
  FROM customer c
  LEFT OUTER JOIN orders o
    ON c.customer_id=o.customer_id
  ORDER BY o.ord_date;
```

```
MySQL  localhost:3306 ssl  salesdb  SQL > SELECT c.cust_name AS "Customer Name", c.city, o.ord_no, o.ord_date, o.purch_amt
                                      ->   FROM customer c
                                      ->   LEFT OUTER JOIN orders o
                                      ->     ON c.customer_id=o.customer_id
                                      ->   ORDER BY o.ord_date;
+----------------+------------+--------+------------+-----------+
| Customer Name  | city       | ord_no | ord_date   | purch_amt |
+----------------+------------+--------+------------+-----------+
| Nick Rimando   | New York   |  70013 | 2012-04-25 |  3045.600 |
| Julian Green   | London     |  70012 | 2012-06-27 |   250.450 |
| Brad Davis     | New York   |  70005 | 2012-07-27 |  2400.600 |
| Jozy Altidor   | Moscow     |  70011 | 2012-08-17 |    75.290 |
| Geoff Cameroon | Berlin     |  70004 | 2012-08-17 |   110.500 |
| Brad Guzan     | London     |  70009 | 2012-09-10 |   270.650 |
| Nick Rimando   | New York   |  70008 | 2012-09-10 |  5760.000 |
| Graham Zusi    | California |  70007 | 2012-09-10 |   948.500 |
| Nick Rimando   | New York   |  70002 | 2012-10-05 |    65.260 |
| Graham Zusi    | California |  70001 | 2012-10-05 |   150.500 |
| Fabian Johnson | Paris      |  70010 | 2012-10-10 |  1983.430 |
| Geoff Cameroon | Berlin     |  70003 | 2012-10-10 |  2480.400 |
+----------------+------------+--------+------------+-----------+
12 rows in set (0.0099 sec)
```

c) Write a SQL statement to make a report with customer name, city, order number, order date, order amount salesman name and commission tofind that either any of the existing customers have placed no order or placed one or more orders by their salesman or by own.

```
SELECT c.cust_name AS "Customer Name",c.city,o.ord_no,o.ord_date,o.
purch_amt,s.name AS "Salesman",
       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;
```

```
MySQL localhost:3306 ssl salesdb SQL > SELECT c.cust_name AS "Customer Name",c.city,o.ord_no,o.ord_date,o.purch_amt,s.name AS "Salesman",
                                       ->        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;
+----------------+------------+--------+------------+-----------+------------+------------+
| Customer Name  | city       | ord_no | ord_date   | purch_amt | Salesman   | commission |
+----------------+------------+--------+------------+-----------+------------+------------+
| Brad Guzan     | London     |  70009 | 2012-09-10 |   270.650 | Pit Alex   |      0.110 |
| Nick Rimando   | New York   |  70002 | 2012-10-05 |    65.260 | James Hoog |      0.150 |
| Nick Rimando   | New York   |  70008 | 2012-09-10 |  5760.000 | James Hoog |      0.150 |
| Nick Rimando   | New York   |  70013 | 2012-04-25 |  3045.600 | James Hoog |      0.150 |
| Jozy Altidor   | Moscow     |  70011 | 2012-08-17 |    75.290 | Paul Adam  |      0.130 |
| Fabian Johnson | Paris      |  70010 | 2012-10-10 |  1983.430 | Mc Lyon    |      0.140 |
| Graham Zusi    | California |  70001 | 2012-10-05 |   150.500 | Nail Knite |      0.130 |
| Graham Zusi    | California |  70007 | 2012-09-10 |   948.500 | Nail Knite |      0.130 |
| Brad Davis     | New York   |  70005 | 2012-07-27 |  2400.600 | James Hoog |      0.150 |
| Julian Green   | London     |  70012 | 2012-06-27 |   250.450 | Nail Knite |      0.130 |
| Geoff Cameron  | Berlin     |  70003 | 2012-10-10 |  2480.400 | Lauson Hen |      0.120 |
| Geoff Cameron  | Berlin     |  70004 | 2012-08-17 |   110.500 | Lauson Hen |      0.120 |
+----------------+------------+--------+------------+-----------+------------+------------+
12 rows in set (0.0006 sec)
```

d) Write a SQL statement to make a list in ascending order for the salesmenwho works either for one or more customer or not yet join u nder and of the customers. (Use Right outer join)

```sql
SELECT s.name AS "Salesman"
  FROM salesman s
  RIGHT OUTER JOIN customer c
    ON s.salesman_id=c.salesman_id
  ORDER BY c.salesman_id ASC;
```

```
MySQL localhost:3306 ssl salesdb SQL > SELECT s.name AS "Salesman"
                                       ->   FROM salesman s
                                       ->   LEFT OUTER JOIN customer c
                                       ->     ON s.salesman_id=c.salesman_id
                                       ->   ORDER BY c.salesman_id ASC;
+------------+
| Salesman   |
+------------+
| James Hoog |
| James Hoog |
| Nail Knite |
| Nail Knite |
| Lauson Hen |
| Pit Alex   |
| Mc Lyon    |
| Paul Adam  |
+------------+
8 rows in set (0.0006 sec)
```

e) Write a SQL query to combine each row of salesman table with eac h row of customer table. (Use cross join)

```
SELECT *
    FROM salesman a
    CROSS JOIN customer b;
```

```
MySQL  localhost:3306 ssl  salesdb  SQL > SELECT *
                                      ->     FROM salesman a
                                      ->     CROSS JOIN customer b;
+-------------+------------+----------+------------+-------------+----------------+------------+-------+-------------+
| salesman_id | name       | city     | commission | customer_id | cust_name      | city       | grade | salesman_id |
+-------------+------------+----------+------------+-------------+----------------+------------+-------+-------------+
|        5007 | Paul Adam  | Rome     |      0.130 |        3001 | Brad Guzan     | London     |   200 |        5005 |
|        5006 | Mc Lyon    | Paris    |      0.140 |        3001 | Brad Guzan     | London     |   200 |        5005 |
|        5005 | Pit Alex   | London   |      0.110 |        3001 | Brad Guzan     | London     |   200 |        5005 |
|        5003 | Lauson Hen | San Jose |      0.120 |        3001 | Brad Guzan     | London     |   200 |        5005 |
|        5002 | Nail Knite | Paris    |      0.130 |        3001 | Brad Guzan     | London     |   200 |        5005 |
|        5001 | James Hoog | New York |      0.150 |        3001 | Brad Guzan     | London     |   200 |        5005 |
|        5007 | Paul Adam  | Rome     |      0.130 |        3002 | Nick Rimando   | New York   |   100 |        5001 |
|        5006 | Mc Lyon    | Paris    |      0.140 |        3002 | Nick Rimando   | New York   |   100 |        5001 |
|        5005 | Pit Alex   | London   |      0.110 |        3002 | Nick Rimando   | New York   |   100 |        5001 |
|        5003 | Lauson Hen | San Jose |      0.120 |        3002 | Nick Rimando   | New York   |   100 |        5001 |
|        5002 | Nail Knite | Paris    |      0.130 |        3002 | Nick Rimando   | New York   |   100 |        5001 |
|        5001 | James Hoog | New York |      0.150 |        3002 | Nick Rimando   | New York   |   100 |        5001 |
|        5007 | Paul Adam  | Rome     |      0.130 |        3003 | Jozy Altidor   | Moscow     |   200 |        5007 |
|        5006 | Mc Lyon    | Paris    |      0.140 |        3003 | Jozy Altidor   | Moscow     |   200 |        5007 |
|        5005 | Pit Alex   | London   |      0.110 |        3003 | Jozy Altidor   | Moscow     |   200 |        5007 |
|        5003 | Lauson Hen | San Jose |      0.120 |        3003 | Jozy Altidor   | Moscow     |   200 |        5007 |
|        5002 | Nail Knite | Paris    |      0.130 |        3003 | Jozy Altidor   | Moscow     |   200 |        5007 |
|        5001 | James Hoog | New York |      0.150 |        3003 | Jozy Altidor   | Moscow     |   200 |        5007 |
|        5007 | Paul Adam  | Rome     |      0.130 |        3004 | Fabian Johnson | Paris      |   300 |        5006 |
|        5006 | Mc Lyon    | Paris    |      0.140 |        3004 | Fabian Johnson | Paris      |   300 |        5006 |
|        5005 | Pit Alex   | London   |      0.110 |        3004 | Fabian Johnson | Paris      |   300 |        5006 |
|        5003 | Lauson Hen | San Jose |      0.120 |        3004 | Fabian Johnson | Paris      |   300 |        5006 |
|        5002 | Nail Knite | Paris    |      0.130 |        3004 | Fabian Johnson | Paris      |   300 |        5006 |
|        5001 | James Hoog | New York |      0.150 |        3004 | Fabian Johnson | Paris      |   300 |        5006 |
|        5007 | Paul Adam  | Rome     |      0.130 |        3005 | Graham Zusi    | California |   200 |        5002 |
|        5006 | Mc Lyon    | Paris    |      0.140 |        3005 | Graham Zusi    | California |   200 |        5002 |
|        5005 | Pit Alex   | London   |      0.110 |        3005 | Graham Zusi    | California |   200 |        5002 |
|        5003 | Lauson Hen | San Jose |      0.120 |        3005 | Graham Zusi    | California |   200 |        5002 |
|        5002 | Nail Knite | Paris    |      0.130 |        3005 | Graham Zusi    | California |   200 |        5002 |
|        5001 | James Hoog | New York |      0.150 |        3005 | Graham Zusi    | California |   200 |        5002 |
|        5007 | Paul Adam  | Rome     |      0.130 |        3007 | Brad Davis     | New York   |   200 |        5001 |
|        5006 | Mc Lyon    | Paris    |      0.140 |        3007 | Brad Davis     | New York   |   200 |        5001 |
|        5005 | Pit Alex   | London   |      0.110 |        3007 | Brad Davis     | New York   |   200 |        5001 |
|        5003 | Lauson Hen | San Jose |      0.120 |        3007 | Brad Davis     | New York   |   200 |        5001 |
|        5002 | Nail Knite | Paris    |      0.130 |        3007 | Brad Davis     | New York   |   200 |        5001 |
|        5001 | James Hoog | New York |      0.150 |        3007 | Brad Davis     | New York   |   200 |        5001 |
|        5007 | Paul Adam  | Rome     |      0.130 |        3008 | Julian Green   | London     |   300 |        5002 |
|        5006 | Mc Lyon    | Paris    |      0.140 |        3008 | Julian Green   | London     |   300 |        5002 |
|        5005 | Pit Alex   | London   |      0.110 |        3008 | Julian Green   | London     |   300 |        5002 |
|        5003 | Lauson Hen | San Jose |      0.120 |        3008 | Julian Green   | London     |   300 |        5002 |
|        5002 | Nail Knite | Paris    |      0.130 |        3008 | Julian Green   | London     |   300 |        5002 |
|        5001 | James Hoog | New York |      0.150 |        3008 | Julian Green   | London     |   300 |        5002 |
|        5007 | Paul Adam  | Rome     |      0.130 |        3009 | Geoff Cameroon | Berlin     |   100 |        5003 |
|        5006 | Mc Lyon    | Paris    |      0.140 |        3009 | Geoff Cameroon | Berlin     |   100 |        5003 |
|        5005 | Pit Alex   | London   |      0.110 |        3009 | Geoff Cameroon | Berlin     |   100 |        5003 |
|        5003 | Lauson Hen | San Jose |      0.120 |        3009 | Geoff Cameroon | Berlin     |   100 |        5003 |
|        5002 | Nail Knite | Paris    |      0.130 |        3009 | Geoff Cameroon | Berlin     |   100 |        5003 |
|        5001 | James Hoog | New York |      0.150 |        3009 | Geoff Cameroon | Berlin     |   100 |        5003 |
+-------------+------------+----------+------------+-------------+----------------+------------+-------+-------------+
48 rows in set (0.0014 sec)
```

f) Write a SQL statement to make a cartesian product between salesm an and customer i.e.each salesman will appear for all customer and vice versa for that salesman who belongs to a city.(Use cross join)

```sql
SELECT s.name AS "Salesman", c.cust_name AS "Customer Name"
    FROM salesman s
    CROSS JOIN customer c;
```

```
MySQL localhost:3306 ssl salesdb SQL > SELECT s.name AS "Salesman", c.cust_name AS "Customer Name"
                                     ->     FROM salesman s
                                     ->     CROSS JOIN customer c;
+------------+----------------+
| Salesman   | Customer Name  |
+------------+----------------+
| Paul Adam  | Brad Guzan     |
| Mc Lyon    | Brad Guzan     |
| Pit Alex   | Brad Guzan     |
| Lauson Hen | Brad Guzan     |
| Nail Knite | Brad Guzan     |
| James Hoog | Brad Guzan     |
| Paul Adam  | Nick Rimando   |
| Mc Lyon    | Nick Rimando   |
| Pit Alex   | Nick Rimando   |
| Lauson Hen | Nick Rimando   |
| Nail Knite | Nick Rimando   |
| James Hoog | Nick Rimando   |
| Paul Adam  | Jozy Altidor   |
| Mc Lyon    | Jozy Altidor   |
| Pit Alex   | Jozy Altidor   |
| Lauson Hen | Jozy Altidor   |
| Nail Knite | Jozy Altidor   |
| James Hoog | Jozy Altidor   |
| Paul Adam  | Fabian Johnson |
| Mc Lyon    | Fabian Johnson |
| Pit Alex   | Fabian Johnson |
| Lauson Hen | Fabian Johnson |
| Nail Knite | Fabian Johnson |
| James Hoog | Fabian Johnson |
| Paul Adam  | Graham Zusi    |
| Mc Lyon    | Graham Zusi    |
| Pit Alex   | Graham Zusi    |
| Lauson Hen | Graham Zusi    |
| Nail Knite | Graham Zusi    |
| James Hoog | Graham Zusi    |
| Paul Adam  | Brad Davis     |
| Mc Lyon    | Brad Davis     |
| Pit Alex   | Brad Davis     |
| Lauson Hen | Brad Davis     |
| Nail Knite | Brad Davis     |
| James Hoog | Brad Davis     |
| Paul Adam  | Julian Green   |
| Mc Lyon    | Julian Green   |
| Pit Alex   | Julian Green   |
| Lauson Hen | Julian Green   |
| Nail Knite | Julian Green   |
| James Hoog | Julian Green   |
| Paul Adam  | Geoff Cameroon |
| Mc Lyon    | Geoff Cameroon |
| Pit Alex   | Geoff Cameroon |
| Lauson Hen | Geoff Cameroon |
| Nail Knite | Geoff Cameroon |
| James Hoog | Geoff Cameroon |
+------------+----------------+
48 rows in set (0.0008 sec)
```

g) 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 customer. 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.
(Use left outer join and right outer join)

```sql
SELECT s.name AS "Salesman"
  FROM salesman s
  LEFT OUTER JOIN customer c
    ON s.salesman_id=c.salesman_id
  RIGHT OUTER JOIN orders o
    ON c.customer_id=o.customer_id
  WHERE o.purch_amt >= 2000
    AND grade IS NOT NULL;
```

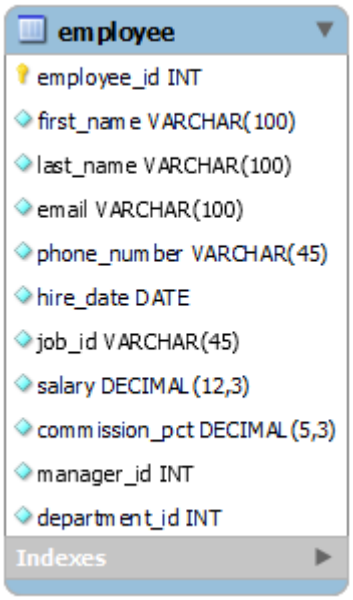
```
MySQL  localhost:3306 ssl  salesdb  SQL > SELECT s.name AS "Salesman"
                                      ->   FROM salesman s
                                      ->   LEFT OUTER JOIN customer c
                                      ->     ON s.salesman_id=c.salesman_id
                                      ->   LEFT OUTER JOIN orders o
                                      ->     ON c.customer_id=o.customer_id
                                      ->   WHERE o.purch_amt >= 2000
                                      ->     AND grade IS NOT NULL;
+------------+
| Salesman   |
+------------+
| Lauson Hen |
| James Hoog |
| James Hoog |
| James Hoog |
+------------+
4 rows in set (0.0195 sec)
```

PROBLEM 2

2) Given the Employee table, perform the following statements using nested queries :

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID
100	Steven	King	SKING	515.123.4567	2003-06-17	AD_PRES	24000.00	0.00	0	90
101	Neena	Kochhar	NKOCHHAR	515.123.4568	2005-09-21	AD_VP	17000.00	0.00	100	40
102	Lex	De Haan	LDEHAAN	515.123.4569	2001-01-13	AD_VP	17000.00	0.00	100	90
103	Alexander	Hunold	AHUNOLD	590.423.4567	2006-01-03	IT_PROG	9000.00	0.00	102	60
104	Bruce	Ernst	BERNST	590.423.4568	2007-05-21	IT_PROG	6000.00	0.00	103	60
105	David	Austin	DAUSTIN	590.423.4569	2005-06-25	IT_PROG	4800.00	0.00	103	40
106	Valli	Pataballa	VPATABAL	590.423.4560	2006-02-05	IT_PROG	4800.00	0.00	103	60
107	Diana	Lorentz	DLORENTZ	590.423.5567	2007-02-07	IT_PROG	4200.00	0.00	103	60
108	Nancy	Greenberg	NGREENBE	515.124.4569	2002-08-17	FI_MGR	12008.00	0.00	101	100
109	Daniel	Faviet	DFAVIET	515.124.4169	2002-08-16	FI_ACCOUNT	9000.00	0.00	108	100
110	John	Chen	JCHEN	515.124.4269	2005-09-28	FI_ACCOUNT	8200.00	0.00	108	40

ER Diagram :




```
CREATE DATABASE employeeDB;

CREATE TABLE `employee`(
  `employee_id` INT NOT NULL,
  `first_name` VARCHAR(100) NOT NULL,
  `last_name` VARCHAR(100) NOT NULL,
  `email` VARCHAR(100) NOT NULL,
  `phone_number` VARCHAR(45) NOT NULL,
  `hire_date` DATE NOT NULL,
  `job_id` VARCHAR(45) NOT NULL,
```

```sql
  `salary` DECIMAL(12,3) NOT NULL,
  `commission_pct` DECIMAL(5,3) NOT NULL,
  `manager_id` INT NOT NULL,
  `department_id` INT NOT NULL,
  PRIMARY KEY (`employee_id`)
  );

INSERT INTO employee
(`employee_id`, `first_name`, `last_name`, `email`, `phone_number`,
 `hire_date`, `job_id`, `salary`, `commission_pct`, `manager_id`, `
department_id`)
VALUES
        ('100', 'Steven', 'King', 'SKING', '515.123.4567', '2003-
06-17', 'AD_PRES', 24000.00, '0.00', '0', '90'),
        ('101', 'Neena', 'Kochhar', 'NKOCHHAR ', '515.123.4568 ', '
 2005-09-21 ', 'AD_VP', 17000.00, '0.00', '100', '40'),
        ('102', 'Lex', 'De Haan', 'LDEHAAN', '515.123.4569', '2001-
01-13 ', 'AD_VP', 17000.00, '0.00', '100', '90'),
        ('103', 'Alexander', 'Hunold', 'AHUNOLD', '590.423.4567', '
2006-01-03', 'IT_PROG', 9000.00, '0.00', '102', '60'),
        ('104', 'Bruce', 'Ernst', 'BERNST', '590.423.4568 ', '2007-
05-21', 'IT_PROG', 6000.00, '0.00', '103', '60'),
        ('105', 'David', 'Austin', 'DAUSTIN', '590.423.4569', '2005
-06-25', 'IT_PROG', 4800.00, '0.00', '103', '40'),
        ('106', 'Valli', 'Pataballa', 'VPATABAL', '590.423.4560', '
2006-02-05', 'IT_PROG', 4800.00, '0.00', '103', '60'),
        ('107', 'Diana', 'Lorentz', 'DLORENTZ', '590.423.5567', '20
07-02-07', 'IT_PROG', 4200.00, '0.00', '103', '60'),
        ('108', 'Nancy', 'Greenberg', 'NGREENBE', '515.124.4569', '
2002-08-17', 'FI_MGR', 12008.00, '0.00', '101', '100'),
        ('109', 'Daniel', 'Faviet', 'DFAVIET', '515.124.4169', '200
2-08-16', 'FI_ACCOUNT', 9000.00, '0.00', '108', '100'),
        ('110', 'John', 'Chen', 'JCHEN', '515.124.4269', '2005-09-
28', 'FI_ACCOUNT', 8200.00, '0.00', '108', '40');
```

-- Query :-
-- Show Tables :

```
SELECT * FROM employee;
```

```
MySQL localhost:3306 ssl employeedb SQL > SELECT * FROM employee;
+-------------+------------+-----------+----------+--------------+------------+------------+----------+----------------+------------+----------
-----+
| employee_id | first_name | last_name | email    | phone_number | hire_date  | job_id     | salary   | commission_pct | manager_id | departmen
t_id |
+-------------+------------+-----------+----------+--------------+------------+------------+----------+----------------+------------+----------
-----+
|         100 | Steven     | King      | SKING    | 515.123.4567 | 2003-06-17 | AD_PRES    | 24000.000 |          0.000 |          0 |
  90 |
|         101 | Neena      | Kochhar   | NKOCHHAR | 515.123.4568 | 2005-09-21 | AD_VP      | 17000.000 |          0.000 |        100 |
  40 |
|         102 | Lex        | De Haan   | LDEHAAN  | 515.123.4569 | 2001-01-13 | AD_VP      | 17000.000 |          0.000 |        100 |
  90 |
|         103 | Alexander  | Hunold    | AHUNOLD  | 590.423.4567 | 2006-01-03 | IT_PROG    |  9000.000 |          0.000 |        102 |
  60 |
|         104 | Bruce      | Ernst     | BERNST   | 590.423.4568 | 2007-05-21 | IT_PROG    |  6000.000 |          0.000 |        103 |
  60 |
|         105 | David      | Austin    | DAUSTIN  | 590.423.4569 | 2005-06-25 | IT_PROG    |  4800.000 |          0.000 |        103 |
  40 |
|         106 | Valli      | Pataballa | VPATABAL | 590.423.4560 | 2006-02-05 | IT_PROG    |  4800.000 |          0.000 |        103 |
  60 |
|         107 | Diana      | Lorentz   | DLORENTZ | 590.423.5567 | 2007-02-07 | IT_PROG    |  4200.000 |          0.000 |        103 |
  60 |
|         108 | Nancy      | Greenberg | NGREENBE | 515.124.4569 | 2002-08-17 | FI_MGR     | 12008.000 |          0.000 |        101 |
 100 |
|         109 | Daniel     | Faviet    | DFAVIET  | 515.124.4169 | 2002-08-16 | FI_ACCOUNT |  9000.000 |          0.000 |        108 |
 100 |
|         110 | John       | Chen      | JCHEN    | 515.124.4269 | 2005-09-28 | FI_ACCOUNT |  8200.000 |          0.000 |        108 |
  40 |
+-------------+------------+-----------+----------+--------------+------------+------------+----------+----------------+------------+----------
-----+
11 rows in set (0.0008 sec)
```

a) Write a query to display the employee name (first name and last name) and department for all employees for any existence of those e mployees whose salary is more than 3700.

```
SELECT first_name, last_name, department_id
    FROM employee
    WHERE EXISTS (SELECT *
          FROM employee
          WHERE salary > 3700.00);
```

```
MySQL localhost:3306 ssl employeedb SQL > SELECT first_name, last_name, department_id
                                        ->         FROM employee
                                        ->         WHERE EXISTS (SELECT *
                                        ->               FROM employee
                                        ->               WHERE salary > 3700.00);
+------------+-----------+---------------+
| first_name | last_name | department_id |
+------------+-----------+---------------+
| Steven     | King      |            90 |
| Neena      | Kochhar   |            40 |
| Lex        | De Haan   |            90 |
| Alexander  | Hunold    |            60 |
| Bruce      | Ernst     |            60 |
| David      | Austin    |            40 |
| Valli      | Pataballa |            60 |
| Diana      | Lorentz   |            60 |
| Nancy      | Greenberg |           100 |
| Daniel     | Faviet    |           100 |
| John       | Chen      |            40 |
+------------+-----------+---------------+
11 rows in set (0.0019 sec)
```

b) Write a query to display the department id and the total salary for those departments which contains at least one employee.

```
    SELECT department_id,
           SUM(salary) AS Total_Salary
    FROM employee
    WHERE department_id IN (SELECT DISTINCT department_id FROM empl
oyee)
    GROUP BY department_id;
```

```
MySQL  localhost:3306 ssl  employeedb  SQL > SELECT department_id,
                                        ->            SUM(salary) AS Total_Salary
                                        ->     FROM employee
                                        ->     WHERE department_id IN (SELECT DISTINCT department_id FROM employee)
                                        ->     GROUP BY department_id;
+---------------+--------------+
| department_id | Total_Salary |
+---------------+--------------+
|            90 |    41000.000 |
|            40 |    30000.000 |
|            60 |    24000.000 |
|           100 |    21008.000 |
+---------------+--------------+
4 rows in set (0.0006 sec)
```

c) Write a subquery that returns a set of rows to find all departments that do actually have one or more employees assigned to them.

```
    SELECT DISTINCT department_id FROM employee;
```

```
MySQL  localhost:3306 ssl  employeedb  SQL > SELECT DISTINCT department_id FROM employee;
+---------------+
| department_id |
+---------------+
|            90 |
|            40 |
|            60 |
|           100 |
+---------------+
4 rows in set (0.0006 sec)
```

d) Write a query in SQL to display the first and last name, salary, and department ID for all those employees who earn more than the average salary and arrange the list in descending order on salary.

```sql
SELECT first_name, last_name, salary, department_id
  FROM employee
  WHERE salary > (SELECT AVG(salary)
                  FROM employee)
  ORDER BY salary DESC;
```

```
MySQL  localhost:3306 ssl  employeedb  SQL >     SELECT first_name, last_name, salary, department_id
                                        ->       FROM employee
                                        ->       WHERE salary > (SELECT AVG(salary)
                                        ->                 FROM employee)
                                        ->       ORDER BY salary DESC;
+------------+-----------+-----------+---------------+
| first_name | last_name | salary    | department_id |
+------------+-----------+-----------+---------------+
| Steven     | King      | 24000.000 |            90 |
| Neena      | Kochhar   | 17000.000 |            40 |
| Lex        | De Haan   | 17000.000 |            90 |
| Nancy      | Greenberg | 12008.000 |           100 |
+------------+-----------+-----------+---------------+
4 rows in set (0.0006 sec)
```

e) Write a query in SQL to display the first and last name, salary, and department ID for those employees who earn more than the maximum salary of a department which ID is 40.

```sql
SELECT first_name, last_name, salary, department_id
    FROM employee
    WHERE salary > (SELECT MAX(salary)
                    FROM employee
                        WHERE department_id = 40);
```

```
MySQL  localhost:3306 ssl  employeedb  SQL >       SELECT first_name, last_name, salary, department_id
                                        ->           FROM employee
                                        ->           WHERE salary > (SELECT MAX(salary)
                                        ->                     FROM employee
                                        ->                         WHERE department_id = 40);
+------------+-----------+-----------+---------------+
| first_name | last_name | salary    | department_id |
+------------+-----------+-----------+---------------+
| Steven     | King      | 24000.000 |            90 |
+------------+-----------+-----------+---------------+
1 row in set (0.0008 sec)
```