

### PRACTICAL NO – 1

**Aim:** DDL operations on Relational Schema

Design the following schema and execute the following queries on it:

salesman				customer				
salesman_id	name	city	commission	customer_id	customer_name	city	grade	salesman_id
5001	James Hoog	New York	0.15	3002	Nick Rimando	New York	100	5001
5002	Nail Knite	Paris	0.13	3005	Graham Zusi	California	200	5002
5005	Pit Alex	London	0.11	3001	Brad Guzan	London		
5006	Mc Lyon	Paris	0.14	3004	Fabian Johns	Paris	300	5006
5003	Lauson Hen		0.12	3007	Brad Davis	New York	200	5001
5007	Paul Adam	Rome	0.13	3009	Geoff Camero	Berlin	100	
				3008	Julian Green	London	300	5002
				3003	Jozy Altidor	Moscow	200	5007

  

order				
order_no	purch_amt	order_date	customer_id	salesman_id
70001	150.5	2016-10-05	3005	5002
70009	270.65	2016-09-10	3001	
70002	65.26	2016-10-05	3002	5001
70004	110.5	2016-08-17	3009	
70007	948.5	2016-09-10	3005	5002
70005	2400.6	2016-07-27	3007	5001
70008	5760	2016-09-10	3002	5001
70010	1983.43	2016-10-10	3004	5006
70003	2480.4	2016-10-10	3009	
70012	250.45	2016-06-27	3008	5002
70011	75.29	2016-08-17	3003	5007

**Code:**

create database salesman;

use salesman

CREATE TABLE salesman(salesman\_id INT NOT NULL

AUTO\_INCREMENT PRIMARY key,

name VARCHAR(100)NOT NULL,

city VARCHAR(100)NOT NULL,

commission DECIMAL(10,2)

);

desc salesman;

**Output:**

```
mysql> create database salesman
-> ;
Query OK, 1 row affected (0.00 sec)

mysql> use salesman
Database changed
mysql> CREATE TABLE salesman(salesman_id INT NOT NULL AUTO_INCREMENT PRIMARY key,
-> name VARCHAR(100)NOT NULL,
-> city VARCHAR(100)NOT NULL,
-> commission DECIMAL(10,2)
-> );
Query OK, 0 rows affected (0.01 sec)

mysql> desc salesman
-> ;
+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| salesman_id | int(11) | NO | PRI | NULL | auto_increment |
| name | varchar(100) | NO | | NULL |
| city | varchar(100) | NO | | NULL |
| commission | decimal(10,2) | YES | | NULL |
+-----+-----+-----+-----+
4 rows in set (0.01 sec)
```

**Code:**

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

```
insert into salesman values(5002, 'Nail Knite', 'Paris', 0.13);
```

```
insert into salesman values(5005, 'Pit Alex', 'London', 0.11);
```

```
insert into salesman values(5006, 'Mc Lyon', 'Paris', 0.14);
```

```
insert into salesman values(5003, 'Lauson Hen', "", 0.12);
```

```
insert into salesman values(5007, 'Paul Adam', 'Rome', 0.13);
```

```
select * from salesman;
```

**Output:**

```
mysql> insert into salesman values(5001, 'James Hoog', 'New York', 0.15);
Query OK, 1 row affected (0.04 sec)

mysql> insert into salesman values(5002, 'Nail Knite', 'Paris', 0.13);
Query OK, 1 row affected (0.00 sec)

mysql> insert into salesman values(5005, 'Pit Alex', 'London', 0.11);
Query OK, 1 row affected (0.00 sec)

mysql> insert into salesman values(5006, 'Mc Lyon', 'Paris', 0.14);
Query OK, 1 row affected (0.00 sec)

mysql> insert into salesman values(5003, 'Lauson Hen', '', 0.12);
Query OK, 1 row affected (0.00 sec)

mysql> insert into salesman values(5007, 'Paul Adam', 'Rome', 0.13);
Query OK, 1 row affected (0.00 sec)

mysql>
mysql> select * from salesman;
+-----+-----+-----+-----+
| salesman_id | name      | city     | commission |
+-----+-----+-----+-----+
|      5001 | James Hoog | New York |      0.15 |
|      5002 | Nail Knite | Paris    |      0.13 |
|      5003 | Lauson Hen |          |      0.12 |
|      5005 | Pit Alex   | London   |      0.11 |
|      5006 | Mc Lyon   | Paris    |      0.14 |
|      5007 | Paul Adam  | Rome    |      0.13 |
+-----+-----+-----+-----+
6 rows in set (0.00 sec)
```

**Code:**

```
CREATE TABLE customer(customer_id INT NOT NULL
AUTO_INCREMENT PRIMARY key,
customer_name VARCHAR(100)NOT NULL,
city VARCHAR(100)NOT NULL,
grade INT,
salesman_id INT,
FOREIGN KEY (salesman_id) REFERENCES salesman(salesman_id)
);
```

desc customer;

**Output:**

```
mysql> CREATE TABLE customer(customer_id INT NOT NULL AUTO_INCREMENT PRIMARY key,
-> customer_name VARCHAR(100)NOT NULL,
-> city VARCHAR(100)NOT NULL,
-> grade INT,
-> salesman_id INT,
-> FOREIGN KEY (salesman_id) REFERENCES salesman(salesman_id)
-> );
Query OK, 0 rows affected (0.05 sec)

mysql> desc customer;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| customer_id | int(11) | NO | PRI | NULL | auto_increment |
| customer_name | varchar(100) | NO | | NULL | |
| city | varchar(100) | NO | | NULL | |
| grade | int(11) | YES | | NULL | |
| salesman_id | int(11) | YES | MUL | NULL | |
+-----+-----+-----+-----+-----+
5 rows in set (0.04 sec)
```

**Code:**

```
insert into customer values(3002, 'Nick Rimando', 'New York', 100, 5001);
```

```
insert into customer values(3005, 'Graham Zusi', 'California', 200, 5002);
```

```
insert into customer values(3001, 'Brad Guzan', 'Londan', 100, 5005);
```

```
insert into customer values(3004, 'Fabian Johns', 'Paris', 300, 5006);
```

```
insert into customer values(3007, 'Brad Davis', 'New York', 200, 5001);
```

```
insert into customer values(3009, 'Geoff Camero', 'Berlin', 100, 5003);
```

```
insert into customer values(3008, 'Julian Green', 'London', 300, 5002);
```

```
insert into customer values(3003, 'Jory Altidor', 'Moncow', 200, 5007);
```

```
select * from customer;
```

### **Output:**

```
mysql> insert into customer values(3002, 'Nick Rimando', 'New York', 100, 5001);
Query OK, 1 row affected (0.01 sec)

mysql> insert into customer values(3005, 'Graham Zusi', 'California', 200, 5002);
Query OK, 1 row affected (0.03 sec)

mysql> insert into customer values(3001, 'Brad Guzan', 'Londan', 100, 5005);
Query OK, 1 row affected (0.00 sec)

mysql> insert into customer values(3004, 'Fabian Johns', 'Paris', 300, 5006);
Query OK, 1 row affected (0.00 sec)

mysql> insert into customer values(3007, 'Brad Davis', 'New York', 200, 5001);
Query OK, 1 row affected (0.01 sec)

mysql> insert into customer values(3009, 'Geoff Camero', 'Berlin', 100, 5003);
Query OK, 1 row affected (0.01 sec)

mysql> insert into customer values(3008, 'Julian Green', 'London', 300, 5002);
Query OK, 1 row affected (0.03 sec)

mysql> insert into customer values(3003, 'Jory Altidor', 'Moncow', 200, 5007);
Query OK, 1 row affected (0.01 sec)

mysql> select * from customer
      -> ;
+-----+-----+-----+-----+
| customer_id | customer_name | city     | grade   | salesman_id |
+-----+-----+-----+-----+
| 3001 | Brad Guzan    | Londan   | 100    | 5005    |
| 3002 | Nick Rimando  | New York | 100    | 5001    |
| 3003 | Jory Altidor   | Moncow   | 200    | 5007    |
| 3004 | Fabian Johns  | Paris    | 300    | 5006    |
| 3005 | Graham Zusi   | California| 200    | 5002    |
| 3007 | Brad Davis     | New York | 200    | 5001    |
| 3008 | Julian Green   | London   | 300    | 5002    |
| 3009 | Geoff Camero   | Berlin   | 100    | 5003    |
+-----+-----+-----+-----+
8 rows in set (0.00 sec)
```

### **Code:**

```
CREATE TABLE orders(order_no INT NOT NULL AUTO_INCREMENT
PRIMARY key,
purch_amt DECIMAL(10,2) NOT NULL,
order_date DATE NOT NULL,
customer_id INT,
salesman_id INT,
FOREIGN KEY (customer_id) REFERENCES customer(customer_id),
```

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

);

desc orders;

**Output:**

```
mysql> CREATE TABLE orders(order_no INT NOT NULL AUTO_INCREMENT PRIMARY key,
-> purch_amt DECIMAL(10,2) NOT NULL,
-> order_date DATE NOT NULL,
-> customer_id INT,
-> salesman_id INT,
-> FOREIGN KEY (customer_id) REFERENCES customer(customer_id),
-> FOREIGN KEY (salesman_id) REFERENCES salesman(salesman_id)
-> );
Query OK, 0 rows affected (0.02 sec)

mysql> desc orders;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| order_no | int(11) | NO | PRI | NULL | auto_increment |
| purch_amt | decimal(10,2) | NO | | NULL |
| order_date | date | NO | | NULL |
| customer_id | int(11) | YES | MUL | NULL |
| salesman_id | int(11) | YES | MUL | NULL |
+-----+-----+-----+-----+-----+
5 rows in set (0.01 sec)
```

**Code:**

```
insert into orders values(70001, 150.5, '2016-10-05', 3005, 5002);

insert into orders values(70009, 270.65, '2016-09-10', 3001, NULL);

insert into orders values(70002, 65.26, '2016-10-05', 3002, 5001);

insert into orders values(70004, 110.5, '2016-08-17', 3009, NULL);

insert into orders values(70007, 948.5, '2016-09-10', 3005, 5002);

insert into orders values(70005, 2400.6, '2016-07-27', 3007, 5001);

insert into orders values(70008, 5760, '2016-09-10', 3002, 5001);

insert into orders values(70010, 1983.43, '2016-10-10', 3004, NULL);

insert into orders values(70003, 2480.4, '2016-10-10', 3009, 5006);
```

```
insert into orders values(70012, 250.45, '2016-06-27', 3008, 5002);
```

```
insert into orders values(70011, 75.29, '2016-08-17', 3003, 5007);
```

```
select * from orders;
```

### Output:

```
mysql> insert into orders values(70001, 150.5, '2016-10-05', 3005, 5002);
Query OK, 1 row affected (0.01 sec)

mysql> insert into orders values(70009, 270.65, '2016-09-10', 3001, NULL);
Query OK, 1 row affected (0.02 sec)

mysql> insert into orders values(70002, 65.26, '2016-10-05', 3002, 5001);
Query OK, 1 row affected (0.01 sec)

mysql> insert into orders values(70004, 110.5, '2016-08-17', 3009, NULL);
Query OK, 1 row affected (0.01 sec)

mysql> insert into orders values(70007, 948.5, '2016-09-10', 3005, 5002);
Query OK, 1 row affected (0.02 sec)

mysql> insert into orders values(70005, 2400.6, '2016-07-27', 3007, 5001);
Query OK, 1 row affected (0.02 sec)

mysql> insert into orders values(70008, 5760, '2016-09-10', 3002, 5001);
Query OK, 1 row affected (0.02 sec)

mysql> insert into orders values(70010, 1983.43, '2016-10-10', 3004, NULL);
Query OK, 1 row affected (0.01 sec)

mysql> insert into orders values(70003, 2480.4, '2016-10-10', 3009, 5006);
Query OK, 1 row affected (0.01 sec)

mysql> insert into orders values(70012, 250.45, '2016-06-27', 3008, 5002);
Query OK, 1 row affected (0.01 sec)
```

```
mysql> insert into orders values(70011, 75.29, '2016-08-17', 3003, 5007);
Query OK, 1 row affected (0.01 sec)
```

```
mysql> select * from orders;
```

order_no	purch_amt	order_date	customer_id	salesman_id
70001	150.50	2016-10-05	3005	5002
70002	65.26	2016-10-05	3002	5001
70003	2480.40	2016-10-10	3009	5006
70004	110.50	2016-08-17	3009	NULL
70005	2400.60	2016-07-27	3007	5001
70007	948.50	2016-09-10	3005	5002
70008	5760.00	2016-09-10	3002	5001
70009	270.65	2016-09-10	3001	NULL
70010	1983.43	2016-10-10	3004	NULL
70011	75.29	2016-08-17	3003	5007
70012	250.45	2016-06-27	3008	5002

1. Display name and commission for all the salesmen.

**Code:**

```
select name, commission FROM salesman;
```

**Output:**

```
mysql> select name, commission FROM salesman;
+-----+-----+
| name      | commission |
+-----+-----+
| James Hoog |      0.15 |
| Nail Knite |      0.13 |
| Lauson Hen |      0.12 |
| Pit Alex   |      0.11 |
| Mc Lyon    |      0.14 |
| Paul Adam  |      0.13 |
+-----+-----+
6 rows in set (0.00 sec)
```

2. Retrieve salesman id of all salesmen from orders table without any repeats.

**Code:**

```
select DISTINCT salesman_id FROM orders;
```

**Output:**

```
mysql> select DISTINCT salesman_id FROM orders;
+-----+
| salesman_id |
+-----+
|      NULL  |
|      5001   |
|      5002   |
|      5006   |
|      5007   |
+-----+
5 rows in set (0.00 sec)
```

3. Display names and city of salesman, who belongs to the city of Paris.

**Code:**

```
select name, city FROM salesman WHERE city = 'Paris';
```

**Output:**

```
mysql> select name, city FROM salesman WHERE city = 'Paris';
+-----+-----+
| name      | city   |
+-----+-----+
| Nail Knite | Paris |
| Mc Lyon    | Paris |
+-----+-----+
2 rows in set (0.00 sec)
```

4. Display all the information for those customers with a grade of 200.

**Code:**

```
select * FROM customer WHERE grade = 200;
```

**Output:**

```
mysql> select * FROM customer WHERE grade = 200;
+-----+-----+-----+-----+
| customer_id | customer_name | city       | grade | salesman_id |
+-----+-----+-----+-----+
|     3003 | Jory Altidor  | Moncow    | 200   |      5007 |
|     3005 | Graham Zusi   | California | 200   |      5002 |
|     3007 | Brad Davis     | New York  | 200   |      5001 |
+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

5. Display the order number, order date and the purchase amount for order(s) which will be delivered by the salesman with ID 5001

**Code:**

```
select order_no,order_date,purch_amt FROM orders WHERE salesman_id = 5001;
```

**Output:**

```
mysql> select order_no,order_date,purch_amt FROM orders WHERE salesman_id = 5001;
+-----+-----+-----+
| order_no | order_date | purch_amt |
+-----+-----+-----+
|    70002 | 2016-10-05 |    65.26 |
|    70005 | 2016-07-27 |  2400.60 |
|    70008 | 2016-09-10 |  5760.00 |
+-----+-----+-----+
3 rows in set (0.00 sec)
```

12. Display all the customers, who are either belongs to the city New York or not had a grade above 100.

**Code:**

```
select * FROM customer WHERE city ='New York' OR grade<=100;
```

**Output:**

```
mysql> select * FROM customer WHERE city ='New York' OR grade<=100;
+-----+-----+-----+-----+
| customer_id | customer_name | city      | grade   | salesman_id |
+-----+-----+-----+-----+
|     3001    | Brad Guzan    | London    |    100  |      5005   |
|     3002    | Nick Rimando | New York  |    100  |      5001   |
|     3007    | Brad Davis    | New York  |    200  |      5001   |
|     3009    | Geoff Camero  | Berlin    |    100  |      5003   |
+-----+-----+-----+-----+
4 rows in set (0.00 sec)
```

13. Find those salesmen with all information who gets the commission within a range of 0.12 and 0.14.

**Code:**

```
select * FROM salesman WHERE commission BETWEEN 0.12 AND 0.14;
```

**Output:**

```
mysql> select * FROM salesman WHERE commission BETWEEN 0.12 AND 0.14;
+-----+-----+-----+-----+
| salesman_id | name        | city      | commission |
+-----+-----+-----+-----+
|     5002    | Nail Knite  | Paris    |      0.13  |
|     5003    | Lauson Hen  |          |      0.12  |
|     5006    | Mc Lyon     | Paris    |      0.14  |
|     5007    | Paul Adam   | Rome    |      0.13  |
+-----+-----+-----+-----+
4 rows in set (0.00 sec)
```

14. Find all those customers with all information whose names are ending with the letter 'n'.

**Code:**

```
select * FROM customer WHERE customer_name LIKE '%n';
```

**Output:**

```
mysql> select * FROM customer WHERE customer_name LIKE '%n';
+-----+-----+-----+-----+
| customer_id | customer_name | city   | grade |
+-----+-----+-----+-----+
|      3001 | Brad Guzan    | London |    100 |
|      3008 | Julian Green   | London |    300 |
+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

15. Find those salesmen with all information whose name containing the 1st character is 'N' and the 4<sup>th</sup> character is 'I' and rests may be any character.

**Code:**

```
select * FROM salesman WHERE name LIKE 'N_I%';
```

**Output:**

```
mysql> select * FROM salesman WHERE name LIKE 'N_I%';
+-----+-----+-----+
| salesman_id | name       | city   | commission |
+-----+-----+-----+
|      5002 | Nail Knite | Paris |      0.13 |
+-----+-----+-----+
1 row in set (0.00 sec)
```

16. Find that customer with all information who does not get any grade except NULL.

**Code:**

```
select * FROM customer WHERE grade is NULL;
```

**Output:**

```
mysql> select * FROM customer WHERE grade is NULL;
Empty set (0.00 sec)
```

17. Find the total purchase amount of all orders.

**Code:**

```
select SUM(purch_amt) AS total_purchase FROM orders;
```

**Output:**

```
mysql> select SUM(purch_amt) AS total_purchase FROM orders;
+-----+
| total_purchase |
+-----+
|      14495.58 |
+-----+
1 row in set (0.02 sec)
```

18. Find the number of salesman currently listing for all of their customers.

**Code:**

```
select salesman_id, COUNT(customer_id) AS total_customers FROM customer GROUP BY salesman_id;
```

**Output:**

```
mysql> select salesman_id, COUNT(customer_id) AS total_customers FROM customer GROUP BY salesman_id;
+-----+-----+
| salesman_id | total_customers |
+-----+-----+
|      5001   |          2 |
|      5002   |          2 |
|      5003   |          1 |
|      5005   |          1 |
|      5006   |          1 |
|      5007   |          1 |
+-----+-----+
6 rows in set (0.00 sec)
```

19. Find the highest grade for each of the cities of the customers.

**Code:**

```
select city, Max(grade) As highest_grade FROM customer GROUP BY city;
```

**Output:**

```
mysql> select city, Max(grade) As highest_grade FROM customer GROUP BY city;
+-----+-----+
| city       | highest_grade |
+-----+-----+
| Berlin     |        100 |
| California |        200 |
| Londan     |        100 |
| London     |        300 |
| Moncow     |        200 |
| New York   |        200 |
| Paris      |        300 |
+-----+-----+
7 rows in set (0.00 sec)
```

20. Find the highest purchase amount ordered by each customer with their ID and highest purchase amount.

**Code:**

```
select customer_id, Max(purch_amt) AS highest_purchase FROM orders GROUP BY customer_id;
```

**Output:**

```
mysql> select customer_id, Max(purch_amt) AS highest_purchase FROM orders GROUP BY customer_id;
+-----+-----+
| customer_id | highest_purchase |
+-----+-----+
| 3001 | 270.65 |
| 3002 | 5760.00 |
| 3003 | 75.29 |
| 3004 | 1983.43 |
| 3005 | 948.50 |
| 3007 | 2400.60 |
| 3008 | 250.45 |
| 3009 | 2480.40 |
+-----+-----+
8 rows in set (0.00 sec)
```

21. Find the highest purchase amount ordered by each customer on a particular date with their ID, order date and highest purchase amount.

**Code:**

```
select customer_id, order_date, Max(purch_amt) AS highest_purchase FROM orders GROUP BY customer_id, order_date;
```

**Output:**

```
mysql> select customer_id, order_date, Max(purch_amt) AS highest_purchase FROM orders GROUP BY customer_id, order_date;
+-----+-----+-----+
| customer_id | order_date | highest_purchase |
+-----+-----+-----+
| 3001 | 2016-09-10 | 270.65 |
| 3002 | 2016-09-10 | 5760.00 |
| 3002 | 2016-10-05 | 65.26 |
| 3003 | 2016-08-17 | 75.29 |
| 3004 | 2016-10-10 | 1983.43 |
| 3005 | 2016-09-10 | 948.50 |
| 3005 | 2016-10-05 | 150.50 |
| 3007 | 2016-07-27 | 2400.60 |
| 3008 | 2016-06-27 | 250.45 |
| 3009 | 2016-08-17 | 110.50 |
| 3009 | 2016-10-10 | 2480.40 |
+-----+-----+-----+
11 rows in set (0.00 sec)
```

22. Find the highest purchase amount on a date '2012-08-17' for each salesman with their ID.

**Code:**

```
select salesman_id, MAX(purch_amt) AS highest_purchase FROM orders WHERE order_date = 2012-08-17 GROUP BY salesman_id;
```

**Output:**

```
mysql> select salesman_id, MAX(purch_amt) AS highest_purchase FROM orders WHERE order_date = 2012-08-17 GROUP BY salesman_id;
Empty set, 1 warning (0.00 sec)
```

23. Find the highest purchase amount with their customer ID and order date, for only those customers who have the highest purchase amount in a day is more than 2000.

**Code:**

```
select customer_id, order_date, MAX(purch_amt) AS highest_purchase FROM orders GROUP BY customer_id, order_date HAVING MAX(purch_amt)>2000;
```

**Output:**

```
mysql> select customer_id, order_date, MAX(purch_amt) AS highest_purchase FROM orders GROUP BY customer_id, order_date HAVING MAX(purch_amt)>2000;
+-----+-----+-----+
| customer_id | order_date | highest_purchase |
+-----+-----+-----+
| 3002 | 2016-09-10 | 5760.00 |
| 3007 | 2016-07-27 | 2400.60 |
| 3009 | 2016-10-10 | 2480.40 |
+-----+-----+-----+
3 rows in set (0.00 sec)
```

24. Write a SQL statement that counts all orders for a date August 17th, 2012.

**Code:**

```
select COUNT(*) AS total_orders FROM orders WHERE order_date = 2012-08-17;
```

**Output:**

```
mysql> select COUNT(*) AS total_orders FROM orders WHERE order_date = 2012-08-17;
+-----+
| total_orders |
+-----+
|          0   |
+-----+
1 row in set, 1 warning (0.00 sec)
```

## PRACTICAL NO – 2

**Aim:** Subquery-join operations on Relational Schema

### USING (practical 1)

**1. Count the customers with grades above Bangalore's average.**

**Code:**

```
SELECT COUNT(*)
FROM Customer
WHERE grade > (
    SELECT AVG(grade)
    FROM Customer
    WHERE city = 'Bangalore'
);
```

**Output:**

```
mysql> SELECT COUNT(*)
   -> FROM Customer
   -> WHERE grade > (
   ->   SELECT AVG(grade)
   ->   FROM Customer
   ->   WHERE city = 'Bangalores'
   -> );
+-----+
| COUNT(*) |
+-----+
|          0 |
+-----+
1 row in set (0.00 sec)
```

1. Find the name and numbers of all salesmen who had more than one customer.

**Code:**

```
SELECT S.name, S.salesman_id
FROM Salesman S
JOIN Customer C ON S.salesman_id = C.salesman_id
GROUP BY S.salesman_id, S.name
HAVING COUNT(C.customer_id) > 1;
```

**Output:**

```
mysql> SELECT S.name, S.salesman_id
-> FROM Salesman S
-> JOIN Customer C ON S.salesman_id = C.salesman_id
-> GROUP BY S.salesman_id, S.name
-> HAVING COUNT(C.customer_id) > 1;
+-----+-----+
| name      | salesman_id |
+-----+-----+
| James Hoog |          5001 |
| Nail Knite |          5002 |
+-----+-----+
2 rows in set (0.02 sec)
```

**2. List all salesmen and indicate those who have and don't have customers in their cities  
(Use UNION operation.)**

**Code:**

```
SELECT S.salesman_id, S.name, 'Has Customers' AS customer_status
FROM Salesman S
JOIN Customer C ON S.salesman_id = C.salesman_id
WHERE S.city = C.city
UNION
SELECT S.salesman_id, S.name, 'No Customers' AS customer_status
```

FROM Salesman S

LEFT JOIN Customer C ON S.salesman\_id = C.salesman\_id AND S.city = C.city  
WHERE C.customer\_id IS NULL;

### Output:

```
mysql> SELECT S.salesman_id, S.name, 'Has Customers' AS customer_status
-> FROM Salesman S
-> JOIN Customer C ON S.salesman_id = C.salesman_id
-> WHERE S.city = C.city
-> UNION
-> SELECT S.salesman_id, S.name, 'No Customers' AS customer_status
-> FROM Salesman S
-> LEFT JOIN Customer C ON S.salesman_id = C.salesman_id AND S.city = C.city
-> WHERE C.customer_id IS NULL;
+-----+-----+
| salesman_id | name      | customer_status |
+-----+-----+
|      5001   | James Hoog | Has Customers  |
|      5006   | Mc Lyon    | Has Customers  |
|      5002   | Nail Knite | No Customers   |
|      5003   | Lauson Hen | No Customers   |
|      5005   | Pit Alex    | No Customers   |
|      5007   | Paul Adam   | No Customers   |
+-----+-----+
6 rows in set (0.00 sec)
```

4. Create a view that finds the salesman who has the customer with the highest order of a day.

### Code:

```
CREATE VIEW SalesmanWithHighestOrder AS
SELECT S.salesman_id, S.name, O.order_date, MAX(O.purch_amt) AS
max_order_amount
FROM Salesman S
JOIN Customer C ON S.salesman_id = C.salesman_id
JOIN `orders` O ON C.customer_id = O.customer_id
GROUP BY S.salesman_id, S.name, O.order_date;
```

```
select * from SalesmanWithHighestOrder;
```

**Output:**

```
mysql> CREATE VIEW SalesmanWithHighestOrder AS
-> SELECT S.salesman_id, S.name, O.order_date, MAX(O.purch_amt) AS max_order_amount
-> FROM Salesman S
-> JOIN Customer C ON S.salesman_id = C.salesman_id
-> JOIN `orders` O ON C.customer_id = O.customer_id
-> GROUP BY S.salesman_id, S.name, O.order_date;
Query OK, 0 rows affected (0.01 sec)

mysql> select * from SalesmanWithHighestOrder;
+-----+-----+-----+-----+
| salesman_id | name      | order_date | max_order_amount |
+-----+-----+-----+-----+
|      5001 | James Hoog | 2016-07-27 |        2400.60 |
|      5001 | James Hoog | 2016-09-10 |        5760.00 |
|      5001 | James Hoog | 2016-10-05 |         65.26 |
|      5002 | Nail Knite | 2016-06-27 |        250.45 |
|      5002 | Nail Knite | 2016-09-10 |        948.50 |
|      5002 | Nail Knite | 2016-10-05 |       150.50 |
|      5003 | Lauson Hen | 2016-08-17 |       110.50 |
|      5003 | Lauson Hen | 2016-10-10 |       2480.40 |
|      5005 | Pit Alex    | 2016-09-10 |       270.65 |
|      5006 | Mc Lyon     | 2016-10-10 |      1983.43 |
|      5007 | Paul Adam   | 2016-08-17 |        75.29 |
+-----+-----+-----+-----+
11 rows in set (0.03 sec)
```

**5. Demonstrate the DELETE operation by removing salesman with id 1000. All his orders must also be deleted**

**Code:**

```
DELETE FROM salesman WHERE salesman_id = 1000;
```

```
SELECT * FROM Salesman;
```

```
SELECT * FROM Orders;
```

**Output:**

```
mysql> DELETE FROM salesman WHERE salesman_id = 1000;
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> SELECT * FROM Salesman;
+-----+-----+-----+-----+
| salesman_id | name      | city     | commission |
+-----+-----+-----+-----+
|      5001 | James Hoog | New York |      0.15 |
|      5002 | Nail Knite | Paris    |      0.13 |
|      5003 | Lauson Hen |          |      0.12 |
|      5005 | Pit Alex   | London   |      0.11 |
|      5006 | Mc Lyon    | Paris    |      0.14 |
|      5007 | Paul Adam  | Rome    |      0.13 |
+-----+-----+-----+-----+
6 rows in set (0.00 sec)
```

```
mysql> SELECT * FROM Orders;
+-----+-----+-----+-----+-----+
| order_no | purch_amt | order_date | customer_id | salesman_id |
+-----+-----+-----+-----+-----+
| 70001   | 150.50   | 2016-10-05 |      3005   |      5002 |
| 70002   | 65.26    | 2016-10-05 |      3002   |      5001 |
| 70003   | 2480.40  | 2016-10-10 |      3009   |      5006 |
| 70004   | 110.50   | 2016-08-17 |      3009   |      NULL  |
| 70005   | 2400.60  | 2016-07-27 |      3007   |      5001 |
| 70007   | 948.50   | 2016-09-10 |      3005   |      5002 |
| 70008   | 5760.00  | 2016-09-10 |      3002   |      5001 |
| 70009   | 270.65   | 2016-09-10 |      3001   |      NULL  |
| 70010   | 1983.43  | 2016-10-10 |      3004   |      NULL  |
| 70011   | 75.29    | 2016-08-17 |      3003   |      5007 |
| 70012   | 250.45   | 2016-06-27 |      3008   |      5002 |
+-----+-----+-----+-----+-----+
11 rows in set (0.00 sec)
```

## 2. Design ERD for the following schema and execute the following Queries on it:

Consider the schema for Movie Database:

**ACTOR (Act\_id, Act\_Name, Act\_Gender)**

**DIRECTOR (Dir\_id, Dir\_Name, Dir\_Phone)**

**MOVIES (Mov\_id, Mov\_Title, Mov\_Year, Mov\_Lang, Dir\_id)**

**MOVIE\_CAST (Act\_id, Mov\_id, Role)**

**RATING (Mov\_id, Rev\_Stars)**

**Code:**

```
CREATE TABLE ACTOR (
    ACT_ID INT (3),
    ACT_NAME VARCHAR (20),
    ACT_GENDER CHAR (1),
    PRIMARY KEY (ACT_ID));
```

**Output:**

```
mysql> CREATE TABLE ACTOR (
    ->     ACT_ID INT(3),
    ->     ACT_NAME VARCHAR(20),
    ->     ACT_GENDER CHAR(1),
    ->     PRIMARY KEY (ACT_ID)
    -> );
Query OK, 0 rows affected (0.02 sec)
```

**Code:**

```
CREATE TABLE DIRECTOR (
    DIR_ID INT (3),
    DIR_NAME VARCHAR (20),
    DIR_PHONE INT (10),
    PRIMARY KEY (DIR_ID));
```

**Output:**

```
mysql> CREATE TABLE DIRECTOR (
    -> DIR_ID INT (3),
    -> DIR_NAME VARCHAR (20),
    -> DIR_PHONE INT (10),
    -> PRIMARY KEY (DIR_ID));
Query OK, 0 rows affected (0.01 sec)
```

### Code:

```
CREATE TABLE MOVIES (
    MOV_ID INT (4),
    MOV_TITLE VARCHAR (25),
    MOV_YEAR INT (4),
    MOV_LANG VARCHAR (12),
    DIR_ID INT (3),
    PRIMARY KEY (MOV_ID),
    FOREIGN KEY (DIR_ID) REFERENCES DIRECTOR (DIR_ID));
```

### Output:

```
mysql> CREATE TABLE MOVIES (
    -> MOV_ID INT (4),
    -> MOV_TITLE VARCHAR (25),
    -> MOV_YEAR INT (4),
    -> MOV_LANG VARCHAR (12),
    -> DIR_ID INT (3),
    -> PRIMARY KEY (MOV_ID),
    -> FOREIGN KEY (DIR_ID) REFERENCES DIRECTOR (DIR_ID));
Query OK, 0 rows affected (0.05 sec)
```

### Code:

```
CREATE TABLE MOVIE_CAST (
    ACT_ID INT (3),
    MOV_ID INT (4),
```

OLE VARCHAR (10),  
PRIMARY KEY (ACT\_ID, MOV\_ID),  
FOREIGN KEY (ACT\_ID) REFERENCES ACTOR (ACT\_ID),  
FOREIGN KEY (MOV\_ID) REFERENCES MOVIES (MOV\_ID));

**Output:**

```
mysql> CREATE TABLE MOVIE_CAST (
    -> ACT_ID INT (3),
    -> MOV_ID INT (4),
    ->
    -> OLE VARCHAR (10),
    -> PRIMARY KEY (ACT_ID, MOV_ID),
    -> FOREIGN KEY (ACT_ID) REFERENCES ACTOR (ACT_ID),
    -> FOREIGN KEY (MOV_ID) REFERENCES MOVIES (MOV_ID));
Query OK, 0 rows affected (0.01 sec)
```

**Code:**

```
CREATE TABLE RATING (
    MOV_ID INT (4),
    REV_STARS VARCHAR (25),
    PRIMARY KEY (MOV_ID),
    FOREIGN KEY (MOV_ID) REFERENCES MOVIES (MOV_ID));
```

**Output:**

```
mysql> CREATE TABLE RATING (
    -> MOV_ID INT (4),
    -> REV_STARS VARCHAR (25),
    -> PRIMARY KEY (MOV_ID),
    -> FOREIGN KEY (MOV_ID) REFERENCES MOVIES (MOV_ID));
Query OK, 0 rows affected (0.01 sec)
```

**Code:**

```
INSERT INTO ACTOR VALUES (301,'ANUSHKA','F');
INSERT INTO ACTOR VALUES (302,'PRABHAS','M');
INSERT INTO ACTOR VALUES (303,'PUNITH','M');
```

INSERT INTO ACTOR VALUES (304,'JERMY','M');

**Output:**

```
mysql> INSERT INTO ACTOR VALUES (301,'ANUSHKA','F');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO ACTOR VALUES (302,'PRABHAS','M');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO ACTOR VALUES (303,'PUNITH','M');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO ACTOR VALUES (304,'JERMY','M');
Query OK, 1 row affected (0.00 sec)
```

**Code:**

INSERT INTO DIRECTOR VALUES (60,'RAJAMOULI', 875161100);

INSERT INTO DIRECTOR VALUES (61,'HITCHCOCK', 776613891);

INSERT INTO DIRECTOR VALUES (62,'FARAN', 998677653);

INSERT INTO DIRECTOR VALUES (63,'STEVEN SPIELBERG',  
898977653);

**Output:**

```
mysql> INSERT INTO DIRECTOR VALUES (60,'RAJAMOULI', 875161100);
Query OK, 1 row affected (0.03 sec)

mysql> INSERT INTO DIRECTOR VALUES (61,'HITCHCOCK', 776613891);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO DIRECTOR VALUES (62,'FARAN', 998677653);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO DIRECTOR VALUES (63,'STEVEN SPIELBERG', 898977653);
Query OK, 1 row affected (0.00 sec)
```

**Code:**

INSERT INTO MOVIES VALUES (1001,'BAHUBALI-2', 2017, 'TELAGU',  
60);

INSERT INTO MOVIES VALUES (1002,'BAHUBALI-1', 2015, 'TELAGU',  
60);

INSERT INTO MOVIES VALUES (1003,'AKASH', 2008, 'KANNADA', 61);

INSERT INTO MOVIES VALUES (1004,'WAR HORSE', 2011, 'ENGLISH', 63);

### Output:

```
mysql> INSERT INTO MOVIES VALUES (1001,'BAHUBALI-2', 2017, 'TELAGU', 60);
Query OK, 1 row affected (0.01 sec)

mysql> INSERT INTO MOVIES VALUES (1002,'BAHUBALI-1', 2015, 'TELAGU', 60);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO MOVIES VALUES (1003,'AKASH', 2008, 'KANNADA', 61);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO MOVIES VALUES (1004,'WAR HORSE', 2011, 'ENGLISH', 63);
Query OK, 1 row affected (0.01 sec)
```

### Code:

INSERT INTO MOVIE\_CAST VALUES (301, 1002, 'HEROINE');

INSERT INTO MOVIE\_CAST VALUES (301, 1001, 'HEROINE');

INSERT INTO MOVIE\_CAST VALUES (303, 1003, 'HERO');

INSERT INTO MOVIE\_CAST VALUES (303, 1002, 'GUEST');

INSERT INTO MOVIE\_CAST VALUES (304, 1004, 'HERO');

### Output:

```
mysql> INSERT INTO MOVIE_CAST VALUES (301, 1002, 'HEROINE');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO MOVIE_CAST VALUES (301, 1001, 'HEROINE');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO MOVIE_CAST VALUES (303, 1003, 'HERO');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO MOVIE_CAST VALUES (303, 1002, 'GUEST');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO MOVIE_CAST VALUES (304, 1004, 'HERO');
Query OK, 1 row affected (0.02 sec)
```

### Code:

INSERT INTO RATING VALUES (1001, 4);

INSERT INTO RATING VALUES (1002, 2);

INSERT INTO RATING VALUES (1003, 5);

INSERT INTO RATING VALUES (1004, 4);

**Output:**

```
mysql> INSERT INTO RATING VALUES (1001, 4);
Query OK, 1 row affected (0.02 sec)

mysql> INSERT INTO RATING VALUES (1002, 2);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO RATING VALUES (1003, 5);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO RATING VALUES (1004, 4);
Query OK, 1 row affected (0.01 sec)
```

Write SQL queries to

1. List the titles of all movies directed by 'Hitchcock'.

**Code:**

```
SELECT MOV_TITLE
FROM MOVIES m
JOIN DIRECTOR d ON m.DIR_ID = d.DIR_ID
WHERE d.DIR_NAME = 'HITCHCOCK';
```

**Output:**

```
mysql> SELECT MOV_TITLE
-> FROM MOVIES m
-> JOIN DIRECTOR d ON m.DIR_ID = d.DIR_ID
-> WHERE d.DIR_NAME = 'HITCHCOCK';
+-----+
| MOV_TITLE |
+-----+
| AKASH    |
+-----+
1 row in set (0.00 sec)
```

2. Find the movie names where one or more actors acted in two or more movies.

**Code:**

```
SELECT DISTINCT m.MOV_TITLE
FROM MOVIES m
JOIN MOVIE_CAST mc ON m.MOV_ID = mc.MOV_ID
WHERE mc.ACT_ID IN (
    SELECT ACT_ID
    FROM MOVIE_CAST
    GROUP BY ACT_ID
    HAVING COUNT(DISTINCT MOV_ID) >= 2
);
```

**Output:**

```
mysql> SELECT DISTINCT m.MOV_TITLE
-> FROM MOVIES m
-> JOIN MOVIE_CAST mc ON m.MOV_ID = mc.MOV_ID
-> WHERE mc.ACT_ID IN (
->     SELECT ACT_ID
->     FROM MOVIE_CAST
->     GROUP BY ACT_ID
->     HAVING COUNT(DISTINCT MOV_ID) >= 2
-> );
+-----+
| MOV_TITLE |
+-----+
| BAHUBALI-2 |
| BAHUBALI-1 |
| AKASH      |
+-----+
3 rows in set (0.03 sec)
```

- 3. List all actors who acted in a movie before 2000 and also in a movie after 2015 (use JOIN operation).**

**Code:**

```
SELECT DISTINCT a.ACT_NAME
FROM ACTOR a
JOIN MOVIE_CAST mc1 ON a.ACT_ID = mc1.ACT_ID
JOIN MOVIES m1 ON mc1.MOV_ID = m1.MOV_ID
JOIN MOVIE_CAST mc2 ON a.ACT_ID = mc2.ACT_ID
JOIN MOVIES m2 ON mc2.MOV_ID = m2.MOV_ID
WHERE m1.MOV_YEAR < 2000 AND m2.MOV_YEAR > 2015;
```

**Output:**

```
mysql> SELECT DISTINCT a.ACT_NAME
-> FROM ACTOR a
-> JOIN MOVIE_CAST mc1 ON a.ACT_ID = mc1.ACT_ID
-> JOIN MOVIES m1 ON mc1.MOV_ID = m1.MOV_ID
-> JOIN MOVIE_CAST mc2 ON a.ACT_ID = mc2.ACT_ID
-> JOIN MOVIES m2 ON mc2.MOV_ID = m2.MOV_ID
-> WHERE m1.MOV_YEAR < 2000 AND m2.MOV_YEAR > 2015;
Empty set (0.00 sec)
```

**4. Find the title of movies and number of stars for each movie that has at least one rating and find the highest number of stars that movie received. Sort the result by movie title.**

**Code:**

```
SELECT m.MOV_TITLE, r.REV_STARS, (
    SELECT MAX(r1.REV_STARS)
    FROM RATING r1
    WHERE r1.MOV_ID = m.MOV_ID
) AS MAX_STARS
FROM MOVIES m
JOIN RATING r ON m.MOV_ID = r.MOV_ID
ORDER BY m.MOV_TITLE;
```

**Output:**

```
mysql> SELECT m.MOV_TITLE, r.REV_STARS, (
    ->     SELECT MAX(r1.REV_STARS)
    ->     FROM RATING r1
    ->     WHERE r1.MOV_ID = m.MOV_ID
    -> ) AS MAX_STARS
    -> FROM MOVIES m
    -> JOIN RATING r ON m.MOV_ID = r.MOV_ID
    -> ORDER BY m.MOV_TITLE;
+-----+-----+-----+
| MOV_TITLE | REV_STARS | MAX_STARS |
+-----+-----+-----+
| AKASH     |      5    |      5    |
| BAHUBALI-1 |      2    |      2    |
| BAHUBALI-2 |      4    |      4    |
| WAR HORSE |      5    |      5    |
+-----+-----+-----+
4 rows in set (0.00 sec)
```

## 5. Update rating of all movies directed by 'Steven Spielberg' to 5.

### Code:

```
UPDATE RATING
SET REV_STARS = '5'
WHERE MOV_ID IN (
    SELECT m.MOV_ID
    FROM MOVIES m
    JOIN DIRECTOR d ON m.DIR_ID = d.DIR_ID
    WHERE d.DIR_NAME = 'STEVEN SPIELBERG'
);
```

### Output:

```
mysql> UPDATE RATING
      -> SET REV_STARS = '5'
      -> WHERE MOV_ID IN (
      ->     SELECT m.MOV_ID
      ->     FROM MOVIES m
      ->     JOIN DIRECTOR d ON m.DIR_ID = d.DIR_ID
      ->     WHERE d.DIR_NAME = 'STEVEN SPIELBERG'
      -> );
Query OK, 1 row affected (0.03 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

**3. Design ERD for the following schema and execute the following Queries on it:**

**Code:**

```
CREATE TABLE students (
    stno INT PRIMARY KEY,
    name VARCHAR(50),
    addr VARCHAR(255),
    city VARCHAR(50),
    state VARCHAR(2),
    zip VARCHAR(10)
);
```

```
CREATE TABLE INSTRUCTORS (
    empno INT PRIMARY KEY,
    name VARCHAR(50),
    rank VARCHAR(20),
    roomno VARCHAR(10),
    telno VARCHAR(15)
);
```

CREATE TABLE COURSES (

cno INT PRIMARY KEY,

cname VARCHAR(50),

cr INT,

cap INT

);

CREATE TABLE GRADES (

stno INT,

empno INT,

cno INT,

sem VARCHAR(10),

year INT,

grade INT,

PRIMARY KEY (stno),

FOREIGN KEY (stno) REFERENCES students(stno),

FOREIGN KEY (empno) REFERENCES INSTRUCTORS(empno),

FOREIGN KEY (cno) REFERENCES COURSES(cno)

);

CREATE TABLE ADVISING (

stno INT,

```
    empno INT,  
    PRIMARY KEY (stno, empno),  
    FOREIGN KEY (stno) REFERENCES students(stno),  
    FOREIGN KEY (empno) REFERENCES INSTRUCTORS(empno)  
);
```

**Output:**

```
mysql> CREATE TABLE students (  
->     stno INT PRIMARY KEY,  
->     name VARCHAR(50),  
->     addr VARCHAR(255),  
->     city VARCHAR(50),  
->     state VARCHAR(2),  
->     zip VARCHAR(10)  
-> );  
Query OK, 0 rows affected (0.04 sec)  
  
mysql>  
mysql> CREATE TABLE INSTRUCTORS (  
->     empno INT PRIMARY KEY,  
->     name VARCHAR(50),  
->     rank VARCHAR(20),  
->     roomno VARCHAR(10),  
->     telno VARCHAR(15)  
-> );  
Query OK, 0 rows affected (0.01 sec)  
  
mysql>  
mysql> CREATE TABLE COURSES (  
->     cno INT PRIMARY KEY,  
->     cname VARCHAR(50),  
->     cr INT,  
->     cap INT  
-> );  
Query OK, 0 rows affected (0.01 sec)
```

```
mysql> CREATE TABLE GRADES (
    ->     stno INT,
    ->     empno INT,
    ->     cno INT,
    ->     sem VARCHAR(10),
    ->     year INT,
    ->     grade INT,
    ->     PRIMARY KEY (stno),
    ->     FOREIGN KEY (stno) REFERENCES students(stno),
    ->     FOREIGN KEY (empno) REFERENCES INSTRUCTORS(empno),
    ->     FOREIGN KEY (cno) REFERENCES COURSES(cno)
    -> );
Query OK, 0 rows affected (0.04 sec)

mysql>
mysql> CREATE TABLE ADVISING (
    ->     stno INT,
    ->     empno INT,
    ->     PRIMARY KEY (stno, empno),
    ->     FOREIGN KEY (stno) REFERENCES students(stno),
    ->     FOREIGN KEY (empno) REFERENCES INSTRUCTORS(empno)
    -> );
Query OK, 0 rows affected (0.04 sec)
```

**Code:**

INSERT INTO COURSES (cno, cname, cr, cap)

VALUES

```
(1, 'Math101', 3, 30),
(2, 'CS210', 4, 25),
(3, 'Physics101', 3, 20);
```

INSERT INTO students (stno, name)

VALUES

```
(1, 'John Doe'),
(2, 'Jane Smith'),
(3, 'Alice Johnson');
```

INSERT INTO instructors (empno, name)

VALUES

```
(101, 'Instructor A'),  
(102, 'Instructor B'),  
(103, 'Instructor C');
```

INSERT INTO GRADES (stno, empno, cno, sem, year, grade)

VALUES

```
(1, 101, 1, 'Fall', 2021, 85),  
(2, 102, 2, 'Fall', 2021, 92),  
(3, 103, 3, 'Fall', 2021, 78);
```

INSERT INTO ADVISING (stno, empno)

VALUES

```
(1, 101),  
(2, 102),  
(3, 103);
```

**Output:**

```
mysql> INSERT INTO COURSES (cno, cname, cr, cap)  
-> VALUES  
->      (1, 'Math101', 3, 30),  
->      (2, 'CS210', 4, 25),  
->      (3, 'Physics101', 3, 20);  
Query OK, 3 rows affected (0.04 sec)  
Records: 3  Duplicates: 0  Warnings: 0
```

```
mysql> INSERT INTO students (stno, name)
-> VALUES
->      (1, 'John Doe'),
->      (2, 'Jane Smith'),
->      (3, 'Alice Johnson');
Query OK, 3 rows affected (0.01 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

```
mysql> INSERT INTO instructors (empno, name)
-> VALUES
->      (101, 'Instructor A'),
->      (102, 'Instructor B'),
->      (103, 'Instructor C');
Query OK, 3 rows affected (0.03 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> INSERT INTO GRADES (stno, empno, cno, sem, year, grade)
-> VALUES
->      (1, 101, 1, 'Fall', 2021, 85),
->      (2, 102, 2, 'Fall', 2021, 92),
->      (3, 103, 3, 'Fall', 2021, 78);
Query OK, 3 rows affected (0.02 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> INSERT INTO ADVISING (stno, empno)
-> VALUES
->      (1, 101),
->      (2, 102),
->      (3, 103);
Query OK, 3 rows affected (0.02 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

## For odd rollnumbers(any 10 )

1. Find the names of students who took some four-credit courses.

### Code:

```
SELECT DISTINCT s.name
FROM students s
JOIN grades g ON s.stno = g.stno
JOIN courses c ON g.cno = c.cno
WHERE c.cr = 4;
```

### Output:

```
mysql> SELECT DISTINCT s.name
-> FROM students s
-> JOIN grades g ON s.stno = g.stno
-> JOIN courses c ON g.cno = c.cno
-> WHERE c.cr = 4;
+-----+
| name |
+-----+
| Jane Smith |
+-----+
1 row in set (0.00 sec)
```

2. Find the names of students who took every four-credit course.

**Code:**

```
SELECT s.name
FROM students s
WHERE NOT EXISTS (
    SELECT 1
    FROM courses c
    WHERE c.cr = 4 AND NOT EXISTS (
        SELECT 1
        FROM grades g
        WHERE g.stno = s.stno AND g.cno = c.cno
    )
);
```

**Output:**

```
mysql> SELECT s.name
-> FROM students s
-> WHERE NOT EXISTS (
->   SELECT 1
->   FROM courses c
->   WHERE c.cr = 4 AND NOT EXISTS (
->     SELECT 1
->     FROM grades g
->     WHERE g.stno = s.stno AND g.cno = c.cno
->   )
-> );
+-----+
| name |
+-----+
| Jane Smith |
+-----+
1 row in set (0.00 sec)
```

**3. Find the names of students who took a course with an instructor who is also their advisor.**

**Code:**

```
SELECT DISTINCT s.name
FROM students s
JOIN grades g ON s.stno = g.stno
JOIN instructors i ON g.empno = i.empno
JOIN advising a ON s.stno = a.stno
WHERE g.empno = a.empno;
```

**Output:**

```
mysql> SELECT DISTINCT s.name
    -> FROM students s
    -> JOIN grades g ON s.stno = g.stno
    -> JOIN instructors i ON g.empno = i.empno
    -> JOIN advising a ON s.stno = a.stno
    -> WHERE g.empno = a.empno;
+-----+
| name |
+-----+
| John Doe
| Jane Smith
| Alice Johnson |
+-----+
3 rows in set (0.00 sec)
```

#### 4. Find the names of students who took cs210 and cs310.

**Code:**

```
SELECT s.name
FROM students s
WHERE EXISTS (
    SELECT 1
    FROM grades g
    JOIN courses c ON g.cno = c.cno
    WHERE s.stno = g.stno AND c cname = 'cs210'
)
AND EXISTS (
    SELECT 1
    FROM grades g
    JOIN courses c ON g.cno = c.cno
    WHERE s.stno = g.stno AND c cname = 'cs310'
);
```

**Output:**

```
mysql> SELECT s.name
-> FROM students s
-> WHERE EXISTS (
->     SELECT 1
->     FROM grades g
->     JOIN courses c ON g.cno = c.cno
->     WHERE s.stno = g.stno AND c cname = 'cs210'
-> )
-> AND EXISTS (
->     SELECT 1
->     FROM grades g
->     JOIN courses c ON g.cno = c.cno
->     WHERE s.stno = g.stno AND c cname = 'cs310'
-> );
Empty set (0.00 sec)
```

5. Find the names of all students whose advisor is not a full professor.

**Code:**

```
SELECT DISTINCT s.name
FROM students s
JOIN advising a ON s.stno = a.stno
JOIN instructors i ON a.empno = i.empno
WHERE i.rank <> 'Full Professor';
```

**Output:**

```
mysql> SELECT DISTINCT s.name
-> FROM students s
-> JOIN advising a ON s.stno = a.stno
-> JOIN instructors i ON a.empno = i.empno
-> WHERE i.rank <> 'Full Professor';
Empty set (0.00 sec)
```

6. Find instructors who taught students who are advised by another instructor who shares the same room.

**Code:**

```
SELECT DISTINCT i1.name
FROM instructors i1
JOIN grades g ON i1.empno = g.empno
JOIN advising a ON g.stno = a.stno
JOIN instructors i2 ON a.empno = i2.empno
WHERE i1.roomno = i2.roomno AND i1.empno <> i2.empno;
```

**Output:**

```
mysql> SELECT DISTINCT i1.name
-> FROM instructors i1
-> JOIN grades g ON i1.empno = g.empno
-> JOIN advising a ON g.stno = a.stno
-> JOIN instructors i2 ON a.empno = i2.empno
-> WHERE i1.roomno = i2.roomno AND i1.empno <> i2.empno;
Empty set (0.00 sec)
```

**7. Find course numbers for courses that enroll exactly two students**

**Code:**

```
SELECT g.cno
FROM grades g
GROUP BY g.cno
HAVING COUNT(DISTINCT g.stno) = 2;
```

**Output:**

```
mysql> SELECT g.cno
    -> FROM grades g
    -> GROUP BY g.cno
    -> HAVING COUNT(DISTINCT g.stno) = 2;
Empty set (0.00 sec)
```

- 8. Find the names of all students for whom no other student lives in the same city.**

**Code:**

```
SELECT s1.name
FROM students s1
WHERE NOT EXISTS (
    SELECT 1
    FROM students s2
    WHERE s1.city = s2.city AND s1.stno <> s2.stno
);
```

**Output:**

```
mysql> SELECT s1.name
    -> FROM students s1
    -> WHERE NOT EXISTS (
    ->     SELECT 1
    ->     FROM students s2
    ->     WHERE s1.city = s2.city AND s1.stno <> s2.stno
    -> );
+-----+
| name |
+-----+
| John Doe
| Jane Smith
| Alice Johnson |
+-----+
3 rows in set (0.00 sec)
```

- 9. Find course numbers of courses taken by students who live in Boston and which are taught by an associate professor.**

**Code:**

```
SELECT DISTINCT g.cno
FROM grades g
JOIN students s ON g.stno = s.stno
JOIN instructors i ON g.empno = i.empno
WHERE s.city = 'Boston' AND i.rank = 'Associate Professor';
```

**Output:**

```
mysql> SELECT DISTINCT g.cno
-> FROM grades g
-> JOIN students s ON g.stno = s.stno
-> JOIN instructors i ON g.empno = i.empno
-> WHERE s.city = 'Boston' AND i.rank = 'Associate Professor';
Empty set (0.00 sec)
```

**10. Find the telephone numbers of instructors who teach a course taken by any student who lives in Boston.**

**Code:**

```
SELECT DISTINCT i.telno
FROM instructors i
JOIN grades g ON i.empno = g.empno
JOIN students s ON g.stno = s.stno
WHERE s.city = 'Boston';
```

**Output:**

```
mysql> SELECT DISTINCT i.telno
-> FROM instructors i
-> JOIN grades g ON i.empno = g.empno
-> JOIN students s ON g.stno = s.stno
-> WHERE s.city = 'Boston';
Empty set (0.00 sec)
```

**11. Find names of students who took every course taken by Richard Pierce.**

**Code:**

```
SELECT s.name
FROM students s
WHERE NOT EXISTS (
    SELECT 1
    FROM grades g1
    JOIN students rp ON rp.name = 'Richard Pierce'
    JOIN grades g2 ON rp.stno = g2.stno
    WHERE g1.cno = g2.cno AND g1.stno <> s.stno
);
```

**Output:**

```
mysql> SELECT s.name
-> FROM students s
-> WHERE NOT EXISTS (
->     SELECT 1
->     FROM grades g1
->     JOIN students rp ON rp.name = 'Richard Pierce'
->     JOIN grades g2 ON rp.stno = g2.stno
->     WHERE g1.cno = g2.cno AND g1.stno <> s.stno
-> );
+-----+
| name      |
+-----+
| John Doe  |
| Jane Smith|
| Alice Johnson |
+-----+
3 rows in set (0.00 sec)
```

**12. Find the names of students who took only one course.**

**Code:**

```
SELECT s.name
FROM students s
JOIN grades g ON s.stno = g.stno
GROUP BY s.stno, s.name
HAVING COUNT(DISTINCT g.cno) = 1;
```

**Output:**

```
mysql> SELECT s.name
-> FROM students s
-> JOIN grades g ON s.stno = g.stno
-> GROUP BY s.stno, s.name
-> HAVING COUNT(DISTINCT g.cno) = 1;
+-----+
| name |
+-----+
| John Doe
| Jane Smith
| Alice Johnson |
+-----+
3 rows in set (0.01 sec)
```

**13. Find the names of instructors who teach no course.**

**Code:**

```
SELECT i.name
FROM instructors i
LEFT JOIN grades g ON i.empno = g.empno
WHERE g.cno IS NULL;
```

**Output:**

```
mysql> SELECT i.name
-> FROM instructors i
-> LEFT JOIN grades g ON i.empno = g.empno
-> WHERE g.cno IS NULL;
Empty set (0.00 sec)
```

**Code:**

```
SELECT i.name
FROM instructors i
JOIN grades g ON i.empno = g.empno
WHERE g.sem = 'Spring' AND g.year = 2001
GROUP BY i.empno, i.name
HAVING COUNT(DISTINCT g.cno) = 1;
```

**Output:**

```
mysql> SELECT i.name
    -> FROM instructors i
    -> JOIN grades g ON i.empno = g.empno
    -> WHERE g.sem = 'Spring' AND g.year = 2001
    -> GROUP BY i.empno, i.name
    -> HAVING COUNT(DISTINCT g.cno) = 1;
Empty set (0.00 sec)
```