NAME: NIDHI SHETTY
ROLL NO.: 01

PRACTICAL NO.: 02

Q1)USING (practical 1)

1.Count the customers with grades above New York's average

```
mysql> SELECT grade, COUNT(*)
    -> FROM customer
    -> GROUP BY grade
    -> HAVING grade >(SELECT AVG(grade)FROM customer WHERE city ='New York');
+-------+----------+
| grade | COUNT(*) |
+-------+----------+
|   200 |        3 |
|   300 |        2 |
+-------+----------+
2 rows in set (0.00 sec)
```

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

```
mysql> SELECT salesman_id, name
    -> FROM salesman a WHERE 1 < (SELECT COUNT(*) FROM customer
    -> WHERE salesman_id = a.salesman_id);
+-------------+------------+
| salesman_id | name       |
+-------------+------------+
|        5001 | James Hoog |
|        5002 | Nail Knite |
+-------------+------------+
2 rows in set (0.01 sec)
```

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

```
mysql> SELECT salesman.salesman_id, name, customer_name, comission
    -> FROM salesman, customer
    -> WHERE salesman.city = customer.city UNION
    -> SELECT salesman_id, name, 'NO MATCH', comission
    -> FROM salesman
    -> WHERE NOT city = ANY
    -> (SELECT city FROM customer)
    -> ORDER BY 2 DESC
    -> ;
+-------------+------------+----------------+-----------+
| salesman_id | name       | customer_name  | comission |
+-------------+------------+----------------+-----------+
|        5005 | Pit Alex   | Brad Guzan     |      0.11 |
|        5005 | Pit Alex   | Julian Green   |      0.11 |
|        5007 | Paul Adam  | NO MATCH       |      0.13 |
|        5002 | Nail Knite | Fabian Johnson |      0.13 |
|        5006 | Mc Lyon    | Fabian Johnson |      0.14 |
|        5003 | Lauson Hen | NO MATCH       |      0.12 |
|        5001 | James Hoog | Nick Rimando   |      0.15 |
|        5001 | James Hoog | Brad Davis     |      0.15 |
+-------------+------------+----------------+-----------+
8 rows in set (0.01 sec)
```

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

```
mysql> CREATE VIEW highsalesman AS
    -> SELECT b.ord_date, a.salesman_id, a.name
    -> FROM salesman a, orders b
    -> WHERE a.salesman_id = b.salesman_id
    -> AND b.purch_amt = (
    ->     SELECT MAX(c.purch_amt)
    ->     FROM orders c
    ->     WHERE c.ord_date = b.ord_date );
Query OK, 0 rows affected (0.01 sec)

mysql> SELECT * FROM highsalesman;
+------------+-------------+------------+
| ord_date   | salesman_id | name       |
+------------+-------------+------------+
| 2012-07-27 |        5001 | James Hoog |
| 2012-09-10 |        5001 | James Hoog |
| 2012-10-05 |        5002 | Nail Knite |
| 2012-06-27 |        5002 | Nail Knite |
+------------+-------------+------------+
4 rows in set (0.01 sec)
```

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

```
mysql> DELETE FROM orders WHERE salesman_id = 1000;
Query OK, 0 rows affected (0.01 sec)

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

Q2)

```
mysql> select * from Actor;
+--------+-----------------+------------+
| act_id | act_name        | act_gender |
+--------+-----------------+------------+
|   1001 | Tom Crusie      | M          |
|   1002 | Chris Hemsworth | M          |
|   1003 | Angelina Jolie  | F          |
|   1004 | Margot Robbie   | F          |
|   1005 | Kate Winslet    | F          |
|   1006 | Robert Downey   | M          |
+--------+-----------------+------------+
6 rows in set (0.00 sec)

mysql> select * from Director;
+--------+------------------+------------+
| dir_id | dir_name         | dir_phone  |
+--------+------------------+------------+
|   9001 | Hitchcock        | 9874562154 |
|   9002 | Steven Spielberg | 9874560054 |
|   9003 | Joseph Levitan   | 9874562178 |
|   9004 | Christopher Loyd | 9874564454 |
|   9005 | Yash Chopra      | 9874562994 |
|   9006 | Tom Jones        | 9874503154 |
+--------+------------------+------------+
6 rows in set (0.00 sec)

mysql> select * from Movie_cast;
+--------+--------+--------+
| act_id | mov_id | role   |
+--------+--------+--------+
|   1001 |    101 | Joey   |
|   1001 |    102 | Conor  |
|   1002 |    102 | Tim    |
|   1002 |    104 | Josh   |
|   1002 |    106 | Craft  |
|   1003 |    103 | Kate   |
|   1004 |    104 | Claire |
|   1005 |    105 | Roy    |
|   1005 |    106 | Jo     |
|   1006 |    105 | Sally  |
+--------+--------+--------+
10 rows in set (0.00 sec)

mysql> select * from Rating;
Empty set (0.00 sec)

mysql>  insert into Rating values(101,4),(102,3),(103,5),(104,2),(105,4),(106,3);
Query OK, 6 rows affected (0.01 sec)
Records: 6  Duplicates: 0  Warnings: 0

mysql> select * from Rating;
+--------+-----------+
| mov_id | rev_stars |
+--------+-----------+
|    101 |         4 |
|    102 |         3 |
|    103 |         5 |
|    104 |         2 |
|    105 |         4 |
|    106 |         3 |
+--------+-----------+
6 rows in set (0.00 sec)
```

NAME: NIDHI SHETTY
ROLL NO.: 01

1.List the titles of all movies directed by ‘Hitchcock’.

```
mysql> select mov_title from Movies where dir_id in (select dir_id from Director where
dir_name='Hitchcock');
+------------+
| mov_title  |
+------------+
| Iron Man   |
| Prosperity |
+------------+
2 rows in set (0.01 sec)
```

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

```
mysql> select distinct m.mov_title,c.act_id from Movies m, Movie_cast c where
m.mov_id=c.mov_id and c.act_id in (select act_id from Movie_cast group by act_id
having count(mov_id)>1);
+-----------------+--------+
| mov_title       | act_id |
+-----------------+--------+
| Iron Man        |   1001 |
| Prosperity      |   1001 |
| Prosperity      |   1002 |
| Star Wars       |   1002 |
| Thor            |   1005 |
| Captain America |   1002 |
| Captain America |   1005 |
+-----------------+--------+
7 rows in set (0.01 sec)
```

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

```
mysql> select act_name from Actor where act_id in(select a.act_id from (select act_id
from Movie_cast natural join Movies where mov_year<2000)a inner join  (select act_id
from Movie_cast natural join Movies where mov_year>2015)b on a.act_id=b.act_id);
```

```
+--------------+
| act_name     |
+--------------+
| Kate Winslet |
+--------------+
1 row in set (0.01 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.

```
mysql> select mov_title,max(rev_stars) from Movies m,Rating r where
m.mov_id=r.mov_id group by m.mov_title having count(r.rev_stars)>0;
+-----------------+----------------+
| mov_title       | max(rev_stars) |
+-----------------+----------------+
| Iron Man        |              4 |
| Prosperity      |              3 |
| Spiderman       |              5 |
| Star Wars       |              2 |
| Thor            |              4 |
| Captain America |              3 |
+-----------------+----------------+
6 rows in set (0.00 sec)
```

5.Update rating of all movies directed by ‘Steven Spielberg’ to 5.

```
mysql> update Rating set rev_stars=5 where mov_id in (select mov_id from Movies
inner join Director on Movies.dir_id=Director.dir_id and Director.dir_name='Steven
Spielberg');
Query OK, 1 row affected (0.01 sec)
Rows matched: 2  Changed: 1  Warnings: 0

mysql> select * from Rating;
+--------+-----------+
| mov_id | rev_stars |
+--------+-----------+
|    101 |         4 |
|    102 |         3 |
|    103 |         5 |
|    104 |         2 |
|    105 |         5 |
|    106 |         3 |
+--------+-----------+
6 rows in set (0.00 sec)
```

Q3)

```
mysql> create table students(stno int primary key,name varchar(20),addr
varchar(100),city varchar(20),state varchar(10),zip int );
Query OK, 0 rows affected (0.04 sec)


mysql> INSERT into students VALUES (2415,'Grogan A. Mary','8 Walnut
St','Malden','MA',02148),(2661,'Mixon Leatha','100 School
St','Brookline','MA',02146),(2890,'McLane Sandy','30 Case
Rd','Boston','MA',02122),(3442,'Novak Roland','42 Beacon
St','Nashua','NH',03060),(3566,'Pierce Richard','70 Park
St','Brookline','MA',02146),(4022,'Prior Lorraine','8 Beacon
St','Boston','MA',02125),(5544, 'Rawlings Jerry','15 Pleasant
Dr','Boston','MA',02115),(5571,'Lewis Jerry','1 Main Rd','Providence','RI',02904);
Query OK, 8 rows affected (0.01 sec)
Records: 8  Duplicates: 0  Warnings: 0


mysql> create table instructions(empno int primary key,name varchar(20),rankk
varchar(20),rommno int,telno int);
Query OK, 0 rows affected (0.05 sec)


mysql> insert into instructions values(019,'Evans
Robert','Professor',82,7122),(023,'Exxon George','Professor',90, 9101),(056,'Sawyer
Kathy','Assoc. Prof',91,5110),(126,'Davis Williams','Assoc. Prof',72,5411),(234,'Will
Samuel','Assist. Prof',90,7024);
Query OK, 5 rows affected (0.01 sec)
Records: 5  Duplicates: 0  Warnings: 0
```

NAME: NIDHI SHETTY
ROLL NO.: 01

```
mysql>
mysql> create table courses(cno Varchar(20) primary key,cname varchar(50),cr int,cap
int);
Query OK, 0 rows affected (0.02 sec)

mysql>
mysql> insert into courses values('cs110','Introduction to
Computing',4,120),('cs210','Computer Programming',4,100),('cs240','Computer
architecture',3,100),('cs310','Data Sructures',3,60),('cs350','Higher Level
Language',3,50),('cs410','Software Engineering',3,40),('cs460','Graphics',3,30);
Query OK, 7 rows affected (0.01 sec)
Records: 7  Duplicates: 0  Warnings: 0


mysql> create table Grades(stno int, foreign key (stno) references
students(stno),empno int, foreign key(empno) references instructions(empno),cno
varchar(20), foreign key(cno) references courses(cno),sem varchar(20),year int,grade
int );
Query OK, 0 rows affected (0.08 sec)


mysql> insert into grades values
    -> (1011,19,'cs110','Fall',2001,40),
    -> (2661,19,'cs110','Fall',2001,80),
    -> (3566,19,'cs110','Fall',2001,95),
    -> (5544,19,'cs110','Fall',2001,100),
    -> (1011,23,'cs110','Spring',2002,75),
    -> (4022,23,'cs110','Spring',2002,60),
    -> (3566,19,'cs240','Spring',2002,100),
    -> (5571,19,'cs240','Spring',2002,50),
    -> (2415,19,'cs240','Spring',2002,100),
    -> (3442,234,'cs410','Spring',2002,60),
    -> (5571,234,'cs410','Spring',2002,80),
    -> (1011,19,'cs210','Fall',2002,90),
    -> (2661,19,'cs210','Fall',2002,70),
    -> (3566,19,'cs210','Fall',2002,90),
    -> (5571,19,'cs210','Spring',2003,85),
    -> (4022,19,'cs210','Spring',2003,70),
    -> (5544,56,'cs240','Spring',2003,70),
    -> (1011,56,'cs240','Spring',2003,90),
    -> (4022,56,'cs240','Spring',2003,80),
    -> (2661,234,'cs310','Spring',2003,100),
    -> (4022,234,'cs310','Spring',2003,75);
Query OK, 21 rows affected (0.01 sec)
Records: 21  Duplicates: 0  Warnings: 0

mysql> create table Advising(stno int, foreign key (stno) references
students(stno),empno int, foreign key(empno) references instructions(empno));
Query OK, 0 rows affected (0.07 sec)

mysql> insert into Advising
values(1011,019),(2415,19),(2661,23),(2890,23),(3442,56),(3566,126),(4022,234),(5544
,23),(5571,234);
Query OK, 9 rows affected (0.01 sec)
Records: 9  Duplicates: 0  Warnings: 0


mysql> select * from students;
+------+-----------------+----------------+------------+-------+------+
| stno | name            | addr           | city       | state | zip  |
+------+-----------------+----------------+------------+-------+------+
| 1011 | Edwards P David | 10 Red Rd      | Newton     | MA    | 2159 |
| 2415 | Grogan A. Mary  | 8 Walnut St    | Malden     | MA    | 2148 |
| 2661 | Mixon Leatha    | 100 School St  | Brookline  | MA    | 2146 |
| 2890 | McLane Sandy    | 30 Case Rd     | Boston     | MA    | 2122 |
| 3442 | Novak Roland    | 42 Beacon St   | Nashua     | NH    | 3060 |
| 3566 | Pierce Richard  | 70 Park St     | Brookline  | MA    | 2146 |
| 4022 | Prior Lorraine  | 8 Beacon St    | Boston     | MA    | 2125 |
| 5544 | Rawlings Jerry  | 15 Pleasant Dr | Boston     | MA    | 2115 |
| 5571 | Lewis Jerry     | 1 Main Rd      | Providence | RI    | 2904 |
+------+-----------------+----------------+------------+-------+------+


mysql> select * from instructions;
+-------+----------------+--------------+--------+-------+
| empno | name           | rankk        | rommno | telno |
+-------+----------------+--------------+--------+-------+
|    19 | Evans Robert   | Professor    |     82 |  7122 |
|    23 | Exxon George   | Professor    |     90 |  9101 |
|    56 | Sawyer Kathy   | Assoc. Prof  |     91 |  5110 |
|   126 | Davis Williams | Assoc. Prof  |     72 |  5411 |
|   234 | Will Samuel    | Assist. Prof |     90 |  7024 |
+-------+----------------+--------------+--------+-------+
5 rows in set (0.00 sec)

mysql> select * from courses;
+-------+--------------------------+------+------+
| cno   | cname                    | cr   | cap  |
+-------+--------------------------+------+------+
| cs110 | Introduction to Computing |    4 |  120 |
| cs210 | Computer Programming     |    4 |  100 |
| cs240 | Computer architecture    |    3 |  100 |
| cs310 | Data Sructures           |    3 |   60 |
| cs350 | Higher Level Language    |    3 |   50 |
| cs410 | Software Engineering     |    3 |   40 |
| cs460 | Graphics                 |    3 |   30 |
+-------+--------------------------+------+------+
7 rows in set (0.00 sec)


mysql> select * from Grades;
+------+-------+-------+--------+------+-------+
| stno | empno | cno   | sem    | year | grade |
+------+-------+-------+--------+------+-------+
| 1011 |    19 | cs110 | Fall   | 2001 |    40 |
| 2661 |    19 | cs110 | Fall   | 2001 |    80 |
| 3566 |    19 | cs110 | Fall   | 2001 |    95 |
| 5544 |    19 | cs110 | Fall   | 2001 |   100 |
| 1011 |    23 | cs110 | Spring | 2002 |    75 |
| 4022 |    23 | cs110 | Spring | 2002 |    60 |
| 3566 |    19 | cs240 | Spring | 2002 |   100 |
| 5571 |    19 | cs240 | Spring | 2002 |    50 |
| 2415 |    19 | cs240 | Spring | 2002 |   100 |
| 3442 |   234 | cs410 | Spring | 2002 |    60 |
| 5571 |   234 | cs410 | Spring | 2002 |    80 |
| 1011 |    19 | cs210 | Fall   | 2002 |    90 |
| 2661 |    19 | cs210 | Fall   | 2002 |    70 |
| 3566 |    19 | cs210 | Fall   | 2002 |    90 |
| 5571 |    19 | cs210 | Spring | 2003 |    85 |
| 4022 |    19 | cs210 | Spring | 2003 |    70 |
| 5544 |    56 | cs240 | Spring | 2003 |    70 |
| 1011 |    56 | cs240 | Spring | 2003 |    90 |
| 4022 |    56 | cs240 | Spring | 2003 |    80 |
| 2661 |   234 | cs310 | Spring | 2003 |   100 |
| 4022 |   234 | cs310 | Spring | 2003 |    75 |
+------+-------+-------+--------+------+-------+


mysql> select * from Advising;
+------+-------+
| stno | empno |
+------+-------+
| 1011 |    19 |
| 2415 |    19 |
| 2661 |    23 |
| 2890 |    23 |
| 3442 |    56 |
| 3566 |   126 |
| 4022 |   234 |
| 5544 |    23 |
| 5571 |   234 |
+------+-------+
9 rows in set (0.00 sec)
```

Questions

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

```
mysql> SELECT DISTINCT students.name
    -> FROM students
    -> JOIN Grades ON students.stno = Grades.stno
    -> JOIN courses ON Grades.cno = courses.cno
    -> WHERE courses.cr = 4;
+-----------------+
| name            |
+-----------------+
| Edwards P David |
| Mixon Leatha    |
| Pierce Richard  |
| Rawlings Jerry  |
| Prior Lorraine  |
| Lewis Jerry     |
+-----------------+
6 rows in set (0.01 sec)
```

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

```
mysql> SELECT students.name
    -> FROM students
    -> JOIN Grades ON students.stno = Grades.stno
    -> JOIN courses ON Grades.cno = courses.cno
    -> WHERE courses.cr = 4
    -> GROUP BY students.stno, students.name
    -> HAVING COUNT(DISTINCT courses.cno) = (
    ->     SELECT COUNT(DISTINCT cno) FROM courses WHERE cr = 4
    -> );
+-----------------+
| name            |
+-----------------+
| Edwards P David |
| Mixon Leatha    |
| Pierce Richard  |
| Prior Lorraine  |
+-----------------+
4 rows in set (0.01 sec)
```

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

```
mysql> SELECT DISTINCT students.name
    -> FROM students
    -> JOIN Grades ON students.stno = Grades.stno
    -> JOIN courses ON Grades.cno = courses.cno
    -> JOIN instructions ON Grades.empno = instructions.empno
    -> JOIN Advising ON students.stno = Advising.stno
    -> WHERE instructions.empno = Advising.empno;
```

```
+-----------------+
| name            |
+-----------------+
| Edwards P David |
| Grogan A. Mary  |
| Prior Lorraine  |
| Lewis Jerry     |
+-----------------+
4 rows in set (0.00 sec)
```

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

```
mysql> SELECT students.name
    -> FROM students
    -> JOIN Grades ON students.stno = Grades.stno
    -> JOIN courses ON Grades.cno = courses.cno
    -> WHERE courses.cno IN ('cs210', 'cs310')
    -> GROUP BY students.stno, students.name
    -> HAVING COUNT(DISTINCT CASE WHEN courses.cno = 'cs210' THEN courses.cno END) > 0
    ->    AND COUNT(DISTINCT CASE WHEN courses.cno = 'cs310' THEN courses.cno END) > 0;
+----------------+
| name           |
+----------------+
| Mixon Leatha   |
| Prior Lorraine |
+----------------+
2 rows in set (0.01 sec)
```

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

```
mysql> SELECT DISTINCT students.name
    -> FROM students
    -> JOIN Advising ON students.stno = Advising.stno
    -> JOIN instructions ON Advising.empno = instructions.empno
    -> WHERE instructions.rankk <> 'Professor';
+----------------+
| name           |
+----------------+
| Novak Roland   |
| Pierce Richard |
| Prior Lorraine |
| Lewis Jerry    |
+----------------+
4 rows in set (0.00 sec)
```

6) Find course numbers for courses that enroll exactly two students;

```
mysql> SELECT cno
    -> FROM Grades
    -> GROUP BY cno
    -> HAVING COUNT(DISTINCT stno) = 2;
+-------+
| cno   |
+-------+
| cs310 |
| cs410 |
+-------+
2 rows in set (0.00 sec)
```

7) Find the names of all students for whom no other student lives in the same city.

```
mysql> SELECT DISTINCT s1.name
    -> FROM students s1
    -> WHERE NOT EXISTS (
    ->     SELECT 1
    ->     FROM students s2
    ->     WHERE s2.stno <> s1.stno
    ->     AND s2.city = s1.city
    -> );
+-----------------+
| name            |
+-----------------+
| Edwards P David |
| Grogan A. Mary  |
| Novak Roland    |
| Lewis Jerry     |
+-----------------+
```

4 rows in set (0.01 sec)

8) Find the names of students who took only one course.

```
mysql> SELECT name
    -> FROM (
    ->     SELECT s.name, COUNT(g.cno) AS num_courses
    ->     FROM students s
    ->     JOIN Grades g ON s.stno = g.stno
    ->     GROUP BY s.name
    -> ) AS student_courses
    -> WHERE num_courses = 1;
+----------------+
| name           |
+----------------+
| Grogan A. Mary |
| Novak Roland   |
+----------------+
2 rows in set (0.01 sec)
```

9) Find the names of instructors who teach no course.

```
mysql> SELECT DISTINCT instructions.name
    -> FROM instructions
    -> LEFT JOIN Grades ON instructions.empno = Grades.empno
    -> WHERE Grades.cno IS NULL;
+----------------+
| name           |
+----------------+
| Davis Williams |
+----------------+
1 row in set (0.00 sec)
```

10) Find the names of the instructors who taught only one course during the spring semester of 2001.

```
mysql> SELECT DISTINCT instructions.name
```

```
    -> FROM instructions
    -> JOIN Grades ON instructions.empno = Grades.empno
    -> JOIN courses ON Grades.cno = courses.cno
    -> WHERE Grades.sem = 'Spring' AND Grades.year = 2001
    -> GROUP BY instructions.empno
    -> HAVING COUNT(DISTINCT Grades.cno) = 1;
Empty set (0.00 sec)
```

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

```
mysql> SELECT DISTINCT instructions.telno
    -> FROM instructions
    -> JOIN Grades ON instructions.empno = Grades.empno
    -> JOIN students ON Grades.stno = students.stno
    -> JOIN courses ON Grades.cno = courses.cno
    -> WHERE students.city = 'Boston';
+-------+
| telno |
+-------+
|  9101 |
|  7122 |
|  5110 |
|  7024 |
+-------+

4 rows in set (0.00 sec)
```