1. Write an SQL command that displays the employee's first name if the employee joined the company before his/her manager.

```
mysql> select emp.first_name from employees emp join employees mgr on emp.manager_id = mgr.employee_id  where emp.hire_date <  mgr.h
ire_date;
```

```
| Payam       |
| James       |
| Renske      |
| Trenna      |
| Curtis      |
| Randall     |
| Peter       |
| Janette     |
| Patrick     |
| Allan       |
| Lisa        |
| Harrison    |
| Tayler      |
| William     |
| Elizabeth   |
| Ellen       |
| Alyssa      |
| Jonathon    |
| Jack        |
| Kimberely   |
| Charles     |
| Nandita     |
| Alexis      |
| Sarah       |
| Britney     |
| Alana       |
| Kevin       |
| Donald      |
| Jennifer    |
| Susan       |
| Hermann     |
| Shelley     |
+-------------+
37 rows in set (0.00 sec)
```

2. Write a SQL procedure that gives a salary raise of 10% to all the employees who have a salary in the bottom 20%.

The screen shot below is the result Before it the code run

```
mysql> select emp.employee_id,emp.salary from employees as emp  where emp.salary < 2800;
+-------------+---------+
| employee_id | salary  |
+-------------+---------+
|         118 | 2600.00 |
|         119 | 2500.00 |
|         126 | 2700.00 |
|         127 | 2400.00 |
|         128 | 2200.00 |
|         131 | 2500.00 |
|         132 | 2100.00 |
|         135 | 2400.00 |
|         136 | 2200.00 |
|         139 | 2700.00 |
|         140 | 2500.00 |
|         143 | 2600.00 |
|         144 | 2500.00 |
|         182 | 2500.00 |
|         191 | 2500.00 |
|         198 | 2600.00 |
|         199 | 2600.00 |
+-------------+---------+
17 rows in set (0.00 sec)
```

```
use db2;
delimiter $
drop procedure if exists myproc2;
create procedure myproc2(salarypct int )
BEGIN
 # this is a comment
 declare num1 int default 0;
 declare num2 int default 0;
 declare total_emp int default 0;
declare bottom_twenty int default 0;
 declare before_raise int default 0;

select COUNT(*) into total_emp  from employees;  # set total_emp to 107

set bottom_twenty =((total_emp * 20 ) / 100);   # set bottom_twenty

select max(salary) into num1 from (select salary from employees order by salary limit bottom_twenty) myalias;  #num1 = 2800

 update employees
 set salary = salary * (1 + salarypct/100)           # updates all employees salary that have less than 2800
 where salary < num1;
select salary into num2 from employees where salary < num2;

# select num1;
#select salarypct;
#select total_emp;
#select bottom_twenty;
select before_raise;
select num2;
# select id;
END$
delimiter ;
~
~
                                                                                    18,0-1        All
```

```
mysql> call myproc2(10);
```

```
mysql> select emp.employee_id, emp.salary from employees as emp where emp.salary < 2800;
+-------------+---------+
| employee_id | salary  |
+-------------+---------+
|         119 | 2750.00 |
|         127 | 2640.00 |
|         128 | 2420.00 |
|         131 | 2750.00 |
|         132 | 2310.00 |
|         135 | 2640.00 |
|         136 | 2420.00 |
|         140 | 2750.00 |
|         144 | 2750.00 |
|         182 | 2750.00 |
|         191 | 2750.00 |
+-------------+---------+
11 rows in set (0.00 sec)

mysql>
```

3. Given 3 employee_id's, write a SQL procedure that switches the salaries between the employees with the highest and lowest salary. For example, given employee_id's of 120, 121, and 122, if employee_id = 121 has the lowest salary and employee_id = 122 has the highest, then swap the salary values between these two employees. The salary for employee_id = 120 would not change.

The screen shot below is the result Before it the code run

```
mysql> select emp.employee_id, emp.salary from employees as emp;
+-------------+----------+
| employee_id | salary   |
+-------------+----------+
|         100 | 24000.00 |
|         101 | 17000.00 |
|         102 | 17000.00 |
|         103 |  9000.00 |
```

```
use db2;
delimiter $
drop procedure if exists myproc3;
create procedure myproc3(id1 int, id2 int,id3 int)
BEGIN
 declare min int default 0;
 declare minid int default 0;
  declare max int default 0;
 declare maxid int default 0;

# the one below graps min salary
select min(salary) into min from employees where
employee_id = id1 or employee_id = id2 or employee_id = id3;

# the one below graps the employee id of the min salary
select employee_id into minid from employees where salary = min and
(employee_id = id1 or employee_id = id2 or employee_id = id3);
# the one below graps max salary
select max(salary) into max from employees where
employee_id = id1 or employee_id = id2 or employee_id = id3;

# the one below graps the employee id of the max salary
select employee_id into maxid from employees where salary = max and
(employee_id = id1 or employee_id = id2 or employee_id = id3);

update employees
set salary = max
where employee_id = minid;

update employees
set salary = min
where employee_id = maxid;

END$
delimiter ;
```

```
mysql>
mysql> source test3_new;
Database changed
Query OK, 0 rows affected, 1 warning (0.01 sec)

Query OK, 0 rows affected (0.03 sec)

mysql> call myproc3(100,101,103);
Query OK, 1 row affected (0.03 sec)

mysql> select emp.employee_id, emp.salary from employees as emp;
+-------------+----------+
| employee_id | salary   |
+-------------+----------+
|         100 |  9000.00 |
|         101 | 17000.00 |
|         102 | 17000.00 |
|         103 | 24000.00 |
```

4. Write the SQL command that determines and outputs the year with the most employees being hired. Then, given this year, write another SQL command that outputs the number of employees hired for each month. “Extract” the month value from hire_date and use it with the Group By clause.

```
mysql>
mysql> select COUNT(emp.employee_id), year(emp.hire_date)
    -> from employees emp
    -> group by year(hire_date);
+------------------------+--------------------+
| COUNT(emp.employee_id) | year(emp.hire_date) |
+------------------------+--------------------+
|                      6 |               2003 |
|                     29 |               2005 |
|                      1 |               2001 |
|                     24 |               2006 |
|                     19 |               2007 |
|                      7 |               2002 |
|                     10 |               2004 |
|                     11 |               2008 |
+------------------------+--------------------+
8 rows in set (0.00 sec)
```

```
mysql> select COUNT(emp.employee_id), MONTH(emp.hire_date)
    ->  from employees emp
    -> where year(emp.hire_date) = 2005
    -> group by MONTH(hire_date);
+------------------------+----------------------+
| COUNT(emp.employee_id) | MONTH(emp.hire_date) |
+------------------------+----------------------+
|                      3 |                    9 |
|                      2 |                    6 |
|                      2 |                   12 |
|                      2 |                    7 |
|                      1 |                    4 |
|                      3 |                   10 |
|                      4 |                    8 |
|                      2 |                    2 |
|                      3 |                    1 |
|                      6 |                    3 |
|                      1 |                   11 |
+------------------------+----------------------+
11 rows in set (0.00 sec)

mysql>
```

5. Display the details of departments (i.e. department_id, department_name, manager_id, location_id) in which the max salary is greater than 10000 for employees who did a job in the past.

```
mysql>
mysql> select * from departments
    -> where department_id in (select department_id from employees
    -> group by department_id
    -> having  max(salary) > 10000);
+---------------+-----------------+------------+-------------+
| department_id | department_name | manager_id | location_id |
+---------------+-----------------+------------+-------------+
|            20 | Marketing       |        201 |        1800 |
|            30 | Purchasing      |        114 |        1700 |
|            80 | Sales           |        145 |        2500 |
|            90 | Executive       |        100 |        1700 |
|           100 | Finance         |        108 |        1700 |
|           110 | Accounting      |        205 |        1700 |
+---------------+-----------------+------------+-------------+
6 rows in set (0.00 sec)

mysql>
```

6. Write a program that accepts 2 parameters (i.e. department_id values), ID1 and ID2, then swap the city value of associated to the department_id ID1 and ID2. For example, given ID1 associated to Location_ID = 5 and Location_ID = 5 has Dallas, and also given ID2 associated to Location_ID = 9 and Location_ID = 9 has Phoenix. Then, the goal is to swap the Dallas and Phoenix values.

```
use db2;
delimiter $
drop procedure if exists myproc6;
create procedure myproc6(d_id1 decimal, d_id2 decimal)
BEGIN

# check type first by doing desc (table name) to match the type when declaring

declare loc_id1 decimal(4);
declare loc_id2 decimal(4);

declare city1 varchar(30);
declare city2 varchar(30);

# Graps the location id which is named loc_id from the departmet_id the user chose

select location_id into loc_id1 from departments where department_id = d_id1;

# Do the same thing for the other department id

select location_id into loc_id2 from departments where department_id = d_id2;

# print location_id to double check it works
#select loc_id1;
#select loc_id2;

# Graps the city which is named city1 where loc_id1 is there

select city into city1 from locations where location_id = loc_id1;

# Do the same thing for loc_id2
select city into city2 from locations where location_id = loc_id2;

# print city before it updates
select city1;
```

```
# print city before it updates
select city1;
select city2;

# Swap the city
update locations
set city = city2
where location_id = loc_id1;

update locations
set city = city1
where location_id = loc_id2;


select city into city1 from locations where location_id = loc_id1;
select city into city2 from locations where location_id = loc_id2;

# print after swap

select city1;
select city2;


END $
delimiter ;
```

```
mysql> call myproc6(20,40);
+---------+
| city1   |
+---------+
| Toronto |
+---------+
1 row in set (0.00 sec)

+--------+
| city2  |
+--------+
| London |
+--------+
1 row in set (0.01 sec)

+--------+
| city1  |
+--------+
| London |
+--------+
1 row in set (0.04 sec)

+---------+
| city2   |
+---------+
| Toronto |
+---------+
1 row in set (0.04 sec)

Query OK, 0 rows affected (0.04 sec)

mysql>
```

7. First, find 3 employees having different salary values (this can be done manually and then the employee_id values can be hardcoded within the code). Then display the employee_id followed by the salary value of the 3 employees in ascending order. Do not use the Order By clause. The code that performs the sorting must be within a function.

The screen shot below is the result Before it the code run

```
mysql> select employee_id, salary from employees;
+-------------+----------+
| employee_id | salary   |
+-------------+----------+
|         100 | 24000.00 |
|         101 | 17000.00 |
|         102 | 17000.00 |
|         103 |  9000.00 |
```

```
use db2;
delimiter $
drop procedure if exists myproc7;
create procedure myproc7(emp_id1 decimal, emp_id2 decimal, emp_id3 decimal)
BEGIN

declare sal_id1 decimal(8,2);
declare sal_id2 decimal(8,2);
declare sal_id3 decimal(8,2);

select salary into sal_id1 from employees where employee_id = emp_id1;
select salary into sal_id2 from employees where employee_id = emp_id2;
select salary into sal_id3 from employees where employee_id = emp_id3;

# find the smallest here

if ( sal_id1 < sal_id2 ) and  ( sal_id1 < sal_id3)
then
select emp_id1;
select sal_id1;
end if;
if ( sal_id2 < sal_id1 ) and ( sal_id2 < sal_id3)
then
select emp_id2;
select sal_id2;
end if;
if ( sal_id3 < sal_id1 ) and ( sal_id3 < sal_id1)
then
select emp_id3;
select sal_id3;
end if;

# find the second smallest here salary

if ( sal_id1 < sal_id2 ) and  ( sal_id1 > sal_id3)
```

Activities Terminal Sep 28 23:28

yafet@ubuntu: ~/Desktop/CINS570

yafet@ubuntu: ~/Desktop/CINS570 yafet@ubuntu: ~/Desktop/CINS570

```
if ( sal_id1 < sal_id2 ) and  ( sal_id1 > sal_id3)
then
select emp_id1;
select sal_id1;
end if;
if ( sal_id2 < sal_id1 ) and ( sal_id2 > sal_id3)
then
select emp_id2;
select sal_id2;
end if;
if ( sal_id3 < sal_id1 ) and ( sal_id3 > sal_id1)
then
select emp_id3;
select sal_id3;
end if;

 # find the largest salary here

if ( sal_id1 > sal_id2 ) and  ( sal_id1 > sal_id3)
then
select emp_id1;
select sal_id1;
end if;
if ( sal_id2 > sal_id1 ) and ( sal_id2 > sal_id3)
then
select emp_id2;
select sal_id2;
end if;
if ( sal_id3 > sal_id1 ) and ( sal_id3 > sal_id1)
then
select emp_id3;
select sal_id3;
end if;
END $
delimiter ;
                                                                    69,1          Bot
```

```
mysql> call myproc7(100,101,103);
+---------+
| emp_id3 |
+---------+
|     103 |
+---------+
1 row in set (0.00 sec)

+---------+
| sal_id3 |
+---------+
| 9000.00 |
+---------+
1 row in set (0.00 sec)

+---------+
| emp_id2 |
+---------+
|     101 |
+---------+
1 row in set (0.00 sec)

+----------+
| sal_id2  |
+----------+
| 17000.00 |
+----------+
1 row in set (0.00 sec)

+---------+
| emp_id1 |
+---------+
|     100 |
+---------+
1 row in set (0.00 sec)
```

```
+---------+
| emp_id1 |
+---------+
|     100 |
+---------+
1 row in set (0.00 sec)

+----------+
| sal_id1  |
+----------+
| 24000.00 |
+----------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

mysql>
```

8. Find out who the highest and lowest paid employees are. Next, output a simple report stating their hourly wage. Assume that the salary value in the employees table is the monthly salary value and that there are 87 hours per half month.

```
use db2;
delimiter $
drop procedure if exists myproc8;
create procedure myproc8()
BEGIN

declare min_id decimal(6);
declare max_id decimal(6);

declare min_name varchar(20);
declare max_name varchar(20);

declare min_salary decimal(8,2);
declare max_salary decimal(8,2);


declare min_hourly_wage int default 0;
declare max_hourly_wage int default 0;


# graps the employee id of person who earns the smallest salary and put into min_id
select employee_id into min_id from employees  where salary=(select Min(salary) from employees);

# graps the employee id of person who earns the highest salary and put into max_id
select employee_id into max_id from employees  where salary=(select Max(salary) from employees);


# graps the first_name  of person who earns the smallest salary and put into min_name
select first_name into min_name from employees  where salary=(select Min(salary) from employees);

# graps the first_name  of person who earns the highest salary and put into max_name
select first_name  into max_name  from employees  where salary=(select Max(salary) from employees);


# graps the salary  of person who earns the smallest salary and put into min_salary
select salary  into min_salary from employees  where salary=(select Min(salary) from employees);
                                                                                                  1,1           Top
```

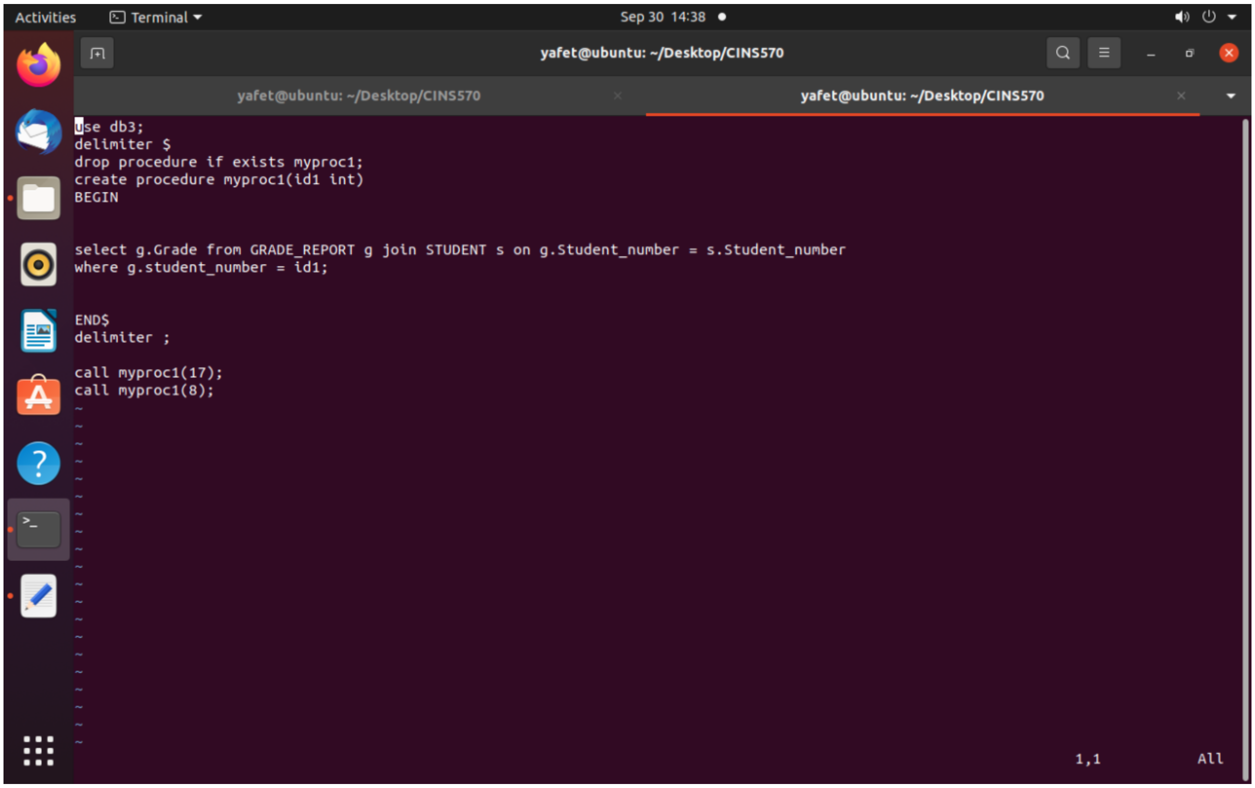
```
# graps the salary  of person who earns the smallest salary and put into min_salary
select salary  into min_salary from employees  where salary=(select Min(salary) from employees);

# graps the salary  of person who earns the highest salary and put into max_salary
select salary  into max_salary  from employees  where salary=(select Max(salary) from employees);

set min_hourly_wage= min_salary / (87*2);

select min_name;
select min_id;
select min_salary;
select min_hourly_wage;

set max_hourly_wage= max_salary / (87*2);

select max_name;
select max_id;
select max_salary;
select max_hourly_wage;

END$
delimiter ;
```

```
mysql> call myproc8();
+----------+
| min_name |
+----------+
| TJ       |
+----------+
1 row in set (0.01 sec)

+--------+
| min_id |
+--------+
|    132 |
+--------+
1 row in set (0.01 sec)

+------------+
| min_salary |
+------------+
|    2310.00 |
+------------+
1 row in set (0.01 sec)

+-----------------+
| min_hourly_wage |
+-----------------+
|              13 |
+-----------------+
1 row in set (0.01 sec)

+----------+
| max_name |
+----------+
| Tayler   |
+----------+
1 row in set (0.01 sec)

+--------+
```

```
+--------+
| max_id |
+--------+
|    170 |
+--------+
1 row in set (0.01 sec)

+------------+
| max_salary |
+------------+
|   58978.64 |
+------------+
1 row in set (0.01 sec)

+-----------------+
| max_hourly_wage |
+-----------------+
|             339 |
+-----------------+
1 row in set (0.01 sec)

Query OK, 0 rows affected (0.01 sec)

mysql>
```

9. Using your own project data, apply 4 queries similar to complexity and commands of the above 8. The goal is to practice more the commands used, plus getting to know your respective project data more.

1, Find out the grade of a student by passing the student id using procedure?



```
use db3;
delimiter $
drop procedure if exists myproc1;
create procedure myproc1(id1 int)
BEGIN


select g.Grade from GRADE_REPORT g join STUDENT s on g.Student_number = s.Student_number
where g.student_number = id1;


END$
delimiter ;

call myproc1(17);
call myproc1(8);
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
                                                                                  1,1           All
```

```
mysql>
mysql> source test_1_students;
Database changed
Query OK, 0 rows affected (0.01 sec)

Query OK, 0 rows affected (0.01 sec)

+-------+
| Grade |
+-------+
| B     |
| C     |
+-------+
2 rows in set (0.01 sec)

Query OK, 0 rows affected (0.01 sec)

+-------+
| Grade |
+-------+
| A     |
| A     |
| B     |
| A     |
+-------+
4 rows in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

mysql>
```

2, Using procedure by passing Instructor name retrieve the course number, semester, year, and number of students who took the section.

```
use db3;
delimiter $
drop procedure if exists myproc2;
create procedure myproc2(prof_name varchar(10))
BEGIN

select prof_name;

select s.Course_number,s.Semester,s.Sem_Year,G.Student_number
        from SECTION s join GRADE_REPORT G
        on s.Section_identifier = G.Section_identifier
        where s.Instructor = prof_name;

END$
delimiter ;


call myproc2('King');
call myproc2('Anderson');
call myproc2('Stone');

~
~
~
~
~
~
~
~
~
~
~
~
```

```
+-----------+
| prof_name |
+-----------+
| King      |
+-----------+
1 row in set (0.00 sec)

+---------------+----------+----------+----------------+
| Course_number | Semester | Sem_Year | Student_number |
+---------------+----------+----------+----------------+
| MATH2410      | Fall     |        7 |              8 |
+---------------+----------+----------+----------------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

+-----------+
| prof_name |
+-----------+
| Anderson  |
+-----------+
1 row in set (0.00 sec)

+---------------+----------+----------+----------------+
| Course_number | Semester | Sem_Year | Student_number |
+---------------+----------+----------+----------------+
| CS1310        | Fall     |        7 |              8 |
| CS1310        | Fall     |        8 |             17 |
+---------------+----------+----------+----------------+
2 rows in set (0.00 sec)
```

```
+------------+
| prof_name |
+------------+
| Stone      |
+------------+
1 row in set (0.00 sec)

+----------------+----------+----------+-----------------+
| Course_number | Semester | Sem_Year | Student_number |
+----------------+----------+----------+-----------------+
| CS3380         | Fall     |        8 |               8 |
+----------------+----------+----------+-----------------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

mysql>
```

3, Retrieve the name, course name, course number, credit hours, semester, year, and grade for each course completed by the student given class year and major?

```
use db3;
delimiter $
drop procedure if exists myproc3;
create procedure myproc3(maj char(4), year_class int)
BEGIN

        select S.Name, S.Major, C.Course_name, C.Course_number, C.Credit_hours, SE.Semester, SE.Sem_Year, G.Grade
        from STUDENT AS S, GRADE_REPORT AS G,SECTION AS SE, COURSE AS C
        WHERE ( S.CLASS = year_class AND S.Student_number = G.Student_number AND G.Section_identifier = SE.Section_identifier AND
        SE.Course_number = C.Course_number AND S.Major = maj);



END$
delimiter ;

# HERE MAJOR AND YEAR OF CLASS IS PASSED
call myproc3('CS',2); # A computer science Sophomore student
call myproc3('CS',1); # A computer scinence Freshman student
```

```
mysql> source test_3_students;
Database changed
Query OK, 0 rows affected (0.05 sec)

Query OK, 0 rows affected (0.10 sec)

+-------+-------+---------------------------+---------------+--------------+----------+----------+-------+
| Name  | Major | Course_name               | Course_number | Credit_hours | Semester | Sem_Year | Grade |
+-------+-------+---------------------------+---------------+--------------+----------+----------+-------+
| Brown | CS    | Discrete Mathematics      | MATH2410      |            3 | Fall     |        7 | A     |
| Brown | CS    | Intro to Computer Scinence | CS1310        |            4 | Fall     |        7 | A     |
| Brown | CS    | Data Structures           | CS3320        |            4 | Spring   |        8 | B     |
| Brown | CS    | Database                  | CS3380        |            3 | Fall     |        8 | A     |
+-------+-------+---------------------------+---------------+--------------+----------+----------+-------+
4 rows in set (0.01 sec)

Query OK, 0 rows affected (0.01 sec)

+-------+-------+---------------------------+---------------+--------------+----------+----------+-------+
| Name  | Major | Course_name               | Course_number | Credit_hours | Semester | Sem_Year | Grade |
+-------+-------+---------------------------+---------------+--------------+----------+----------+-------+
| Smith | CS    | Discrete Mathematics      | MATH2410      |            3 | Fall     |        8 | B     |
| Smith | CS    | Intro to Computer Scinence | CS1310        |            4 | Fall     |        8 | C     |
+-------+-------+---------------------------+---------------+--------------+----------+----------+-------+
2 rows in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

mysql>
```

4, Get using procedure the student's name, instructor name course name, course number of both the prerequisite and main class.

```
use db3;
delimiter $
drop procedure if exists myproc4;
create procedure myproc4()
BEGIN

        select S.Name, P.Prerequisite_number, C.Course_number, C.Course_name, SE.Instructor
        From PREREQUISITE AS P, COURSE AS C, STUDENT AS S, GRADE_REPORT AS G, SECTION AS SE
        WHERE ( P.Course_number = C.Course_number AND C.Course_number = SE.Course_number AND
        SE.Section_identifier = G.Section_identifier  AND G.Student_number = S.Student_number);


END$
delimiter ;

call myproc4();

~
~
~
~
```

```
mysql> source test_4_students;
Database changed
Query OK, 0 rows affected (0.02 sec)

Query OK, 0 rows affected (0.03 sec)

+-------+---------------------+---------------+-----------------+------------+
| Name  | Prerequisite_number | Course_number | Course_name     | Instructor |
+-------+---------------------+---------------+-----------------+------------+
| Brown | CS1310              | CS3320        | Data Structures | Knuth      |
| Brown | MATH2410            | CS3380        | Database        | Stone      |
| Brown | CS3320              | CS3380        | Database        | Stone      |
+-------+---------------------+---------------+-----------------+------------+
3 rows in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)
```