

# DBMS LAB QUERIES

ZISHAN AHMAD  
BTECH COMPUTER ENGINEERING  
SEMESTER 5, 2014

Q1. Consider the following three tables.

EMPLOYEE ( empno, name, deptno,job,hiredate, sal , comission, dob, city , phone)

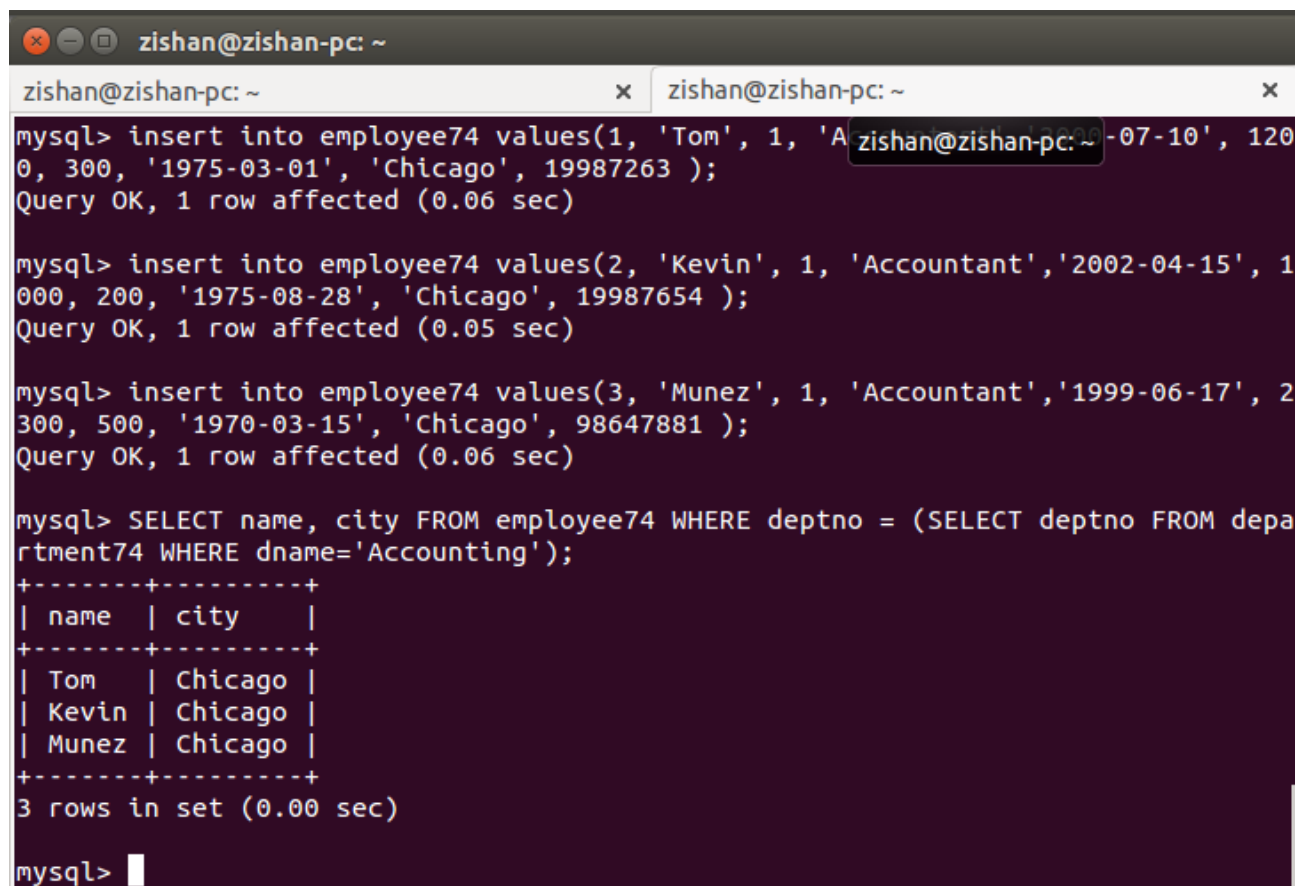
DEPARTMENT ( deptno, dname, manager,loc)

SALARY ( eno, basic,HR,DA,tax).

Write equivalent SQL for the following query. (Use foreign key to join the tables.)

1) Get the name and city of the employee working for the accounting de partment?

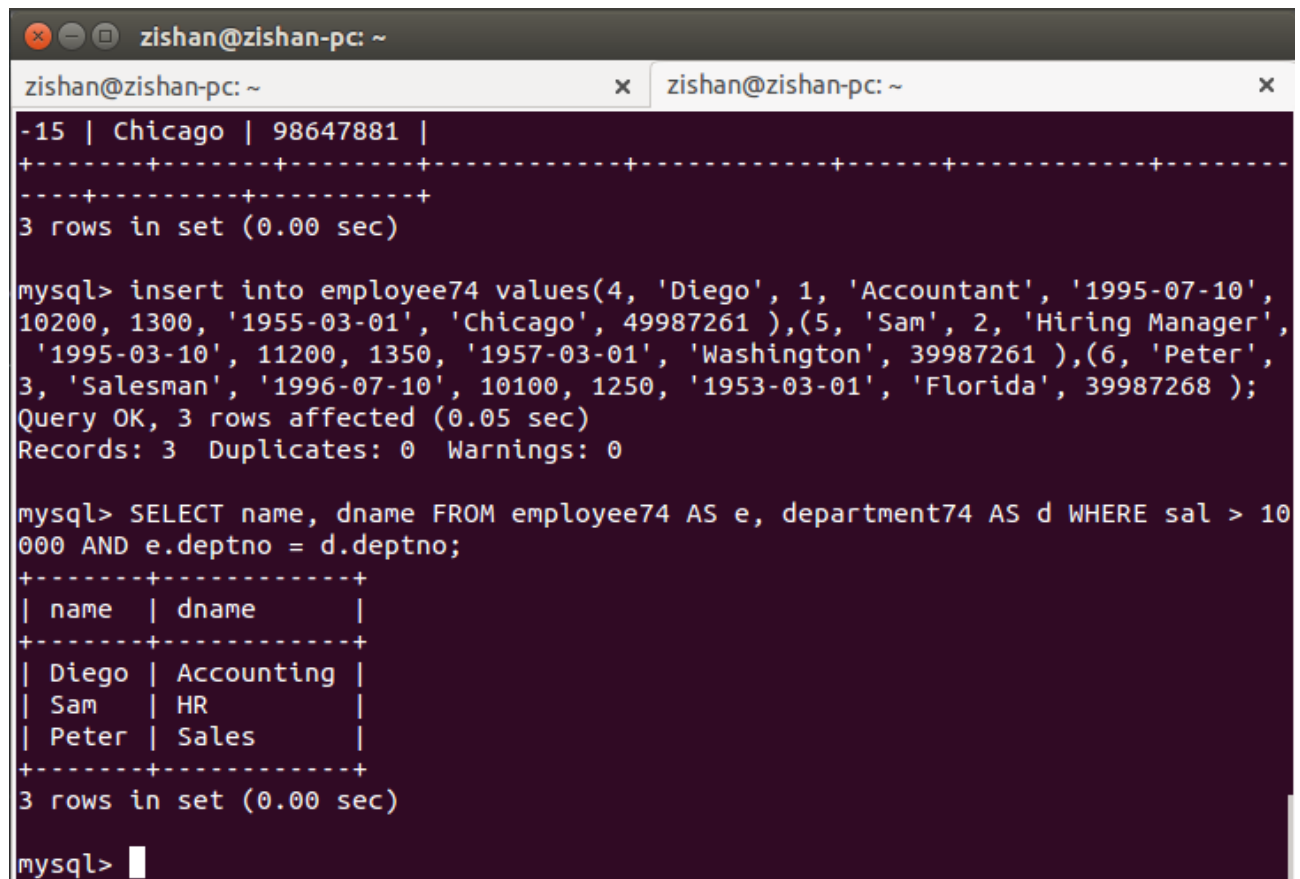
```
SELECT name, city FROM employee74 WHERE deptno = (  
    SELECT deptno FROM department74 WHERE dname='Accounting'  
);
```



```
zishan@zishan-pc: ~  
mysql> insert into employee74 values(1, 'Tom', 1, 'Accountant', '1975-03-01', 1200, 300, '1975-03-01', 'Chicago', 19987263 );  
Query OK, 1 row affected (0.06 sec)  
  
mysql> insert into employee74 values(2, 'Kevin', 1, 'Accountant', '2002-04-15', 1000, 200, '1975-08-28', 'Chicago', 19987654 );  
Query OK, 1 row affected (0.05 sec)  
  
mysql> insert into employee74 values(3, 'Munez', 1, 'Accountant', '1999-06-17', 2300, 500, '1970-03-15', 'Chicago', 98647881 );  
Query OK, 1 row affected (0.06 sec)  
  
mysql> SELECT name, city FROM employee74 WHERE deptno = (SELECT deptno FROM department74 WHERE dname='Accounting');  
+-----+-----+  
| name  | city   |  
+-----+-----+  
| Tom   | Chicago |  
| Kevin | Chicago |  
| Munez | Chicago |  
+-----+-----+  
3 rows in set (0.00 sec)  
  
mysql>
```

2) Get the name, department name of all the employees whose pay is greater than 10000.

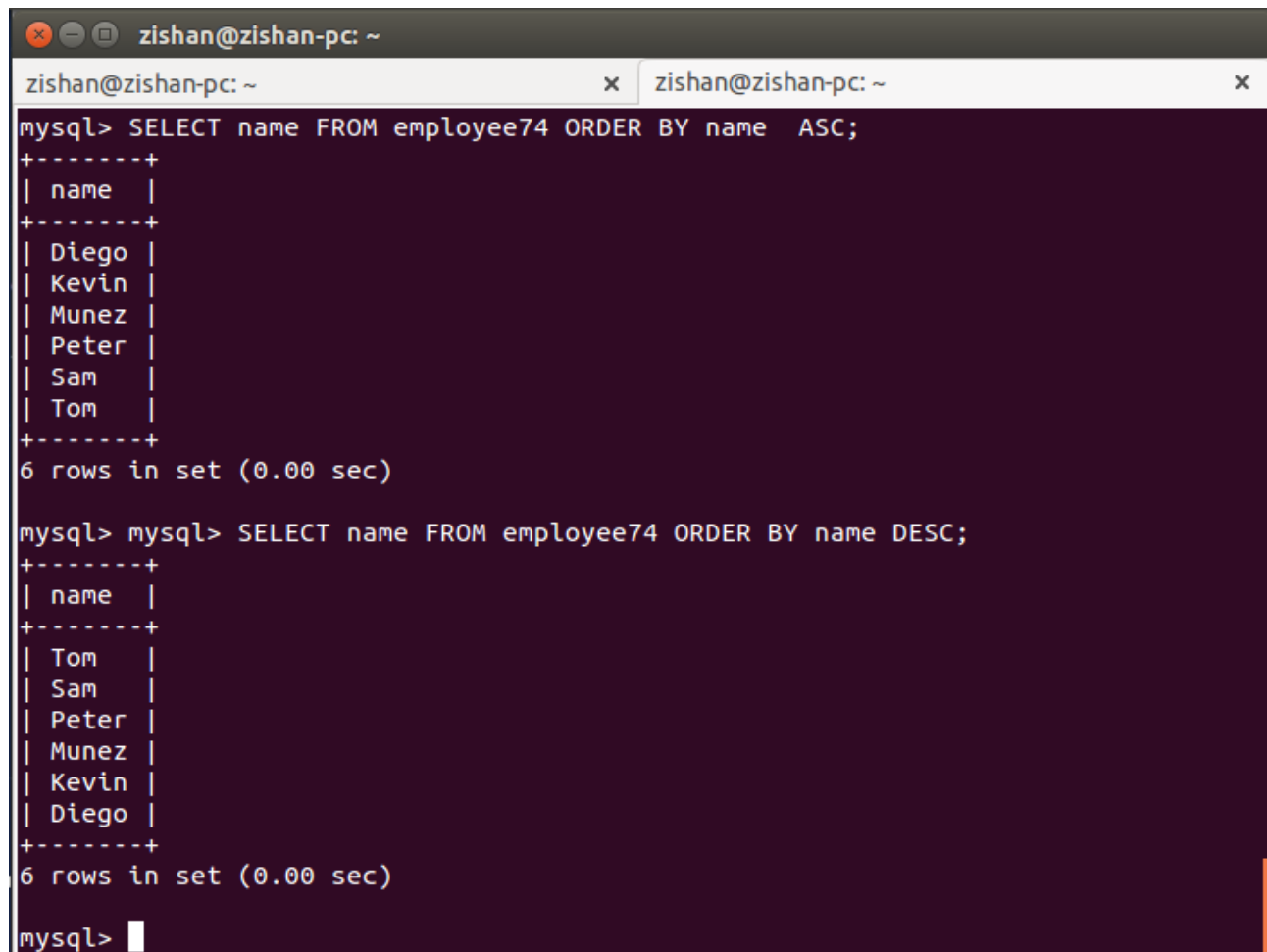
```
SELECT name, dname FROM employee74 AS e,  
       department74 AS d WHERE sal > 10000 AND  
e.deptno = d.deptno;
```



```
zishan@zishan-pc: ~  
zishan@zishan-pc: ~ x zishan@zishan-pc: ~ x  
-15 | Chicago | 98647881 |  
+-----+-----+-----+-----+-----+-----+-----+-----+  
-----+-----+  
3 rows in set (0.00 sec)  
  
mysql> insert into employee74 values(4, 'Diego', 1, 'Accountant', '1995-07-10',  
10200, 1300, '1955-03-01', 'Chicago', 49987261 ),(5, 'Sam', 2, 'Hiring Manager',  
'1995-03-10', 11200, 1350, '1957-03-01', 'Washington', 39987261 ),(6, 'Peter',  
3, 'Salesman', '1996-07-10', 10100, 1250, '1953-03-01', 'Florida', 39987268 );  
Query OK, 3 rows affected (0.05 sec)  
Records: 3 Duplicates: 0 Warnings: 0  
  
mysql> SELECT name, dname FROM employee74 AS e, department74 AS d WHERE sal > 10  
000 AND e.deptno = d.deptno;  
+-----+-----+  
| name | dname |  
+-----+-----+  
| Diego | Accounting |  
| Sam | HR |  
| Peter | Sales |  
+-----+-----+  
3 rows in set (0.00 sec)  
  
mysql> 
```

3) Get the name of the employee in ascending and descending order.

```
SELECT name FROM employee74 ORDER BY name ASC;  
SELECT name FROM employee74 ORDER BY name DESC;
```



```
zishan@zishan-pc: ~  
mysql> SELECT name FROM employee74 ORDER BY name ASC;  
+-----+  
| name |  
+-----+  
| Diego |  
| Kevin |  
| Munez |  
| Peter |  
| Sam   |  
| Tom   |  
+-----+  
6 rows in set (0.00 sec)  
  
mysql> mysql> SELECT name FROM employee74 ORDER BY name DESC;  
+-----+  
| name |  
+-----+  
| Tom   |  
| Sam   |  
| Peter |  
| Munez |  
| Kevin |  
| Diego |  
+-----+  
6 rows in set (0.00 sec)  
  
mysql> 
```

```
UPDATE employee74 SET city = 'Delhi' WHERE empno = 2;
```

5) Get the sum of the basic salary of the employees belongs to Delhi city.

```
SELECT SUM(basic) FROM salary74 WHERE eno IN (  
    SELECT empno FROM employee74 WHERE city = 'Delhi'  
);
```

```
zishan@zishan-pc: ~  
mysql> select * from salary74;  
+-----+-----+-----+-----+  
| empno | basic  | deptno | tax  |  
+-----+-----+-----+-----+  
| 2     | 1000.00 | 1      | 50.00 |  
| 6     | 10100.00 | 3      | 1200.00 |  
+-----+-----+-----+-----+  
2 rows in set (0.00 sec)  
  
mysql> select * from employee74;  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
| empno | name  | deptno | job          | hiredate | sal  | commission | dob       | city      | phone  |  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
| 1     | Tom   | 1      | Accountant  | 2000-07-10 | 1200 | 300         | 1975-03-01 | Chicago   | 19987263 |  
| 2     | Kevin | 1      | Accountant  | 2002-04-15 | 1000 | 200         | 1975-08-28 | Delhi     | 19987654 |  
| 3     | Munez | 1      | Accountant  | 1999-06-17 | 2300 | 500         | 1970-03-15 | Chicago   | 98647881 |  
| 4     | Diego | 1      | Accountant  | 1995-07-10 | 10200 | 1300        | 1955-03-01 | Chicago   | 49987261 |  
| 5     | Sam   | 2      | Hiring Manager | 1995-03-10 | 11200 | 1350        | 1957-03-01 | Washington | 39987261 |  
| 6     | Peter | 3      | Salesman    | 1996-07-10 | 10100 | 1250        | 1953-03-01 | Delhi     | 39987268 |  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
6 rows in set (0.00 sec)  
  
mysql> SELECT SUM(basic) FROM salary74 WHERE eno IN (SELECT empno FROM employee74 WHERE city = 'Delhi');  
ERROR 1054 (42S22): Unknown column 'eno' in 'IN/ALL/ANY subquery'  
mysql> SELECT SUM(basic) FROM salary74 WHERE empno IN (SELECT empno FROM employee74 WHERE city = 'Delhi');  
+-----+  
| SUM(basic) |  
+-----+  
| 11100.00 |  
+-----+  
1 row in set (0.00 sec)  
  
mysql>
```

6) Get the details of the highest income tax payee.

```
SELECT * FROM employee74 WHERE empno IN (  
    SELECT empno FROM salary74 WHERE tax = (  
        SELECT MAX(tax) FROM salary74  
    )  
);
```

```
zishan@zishan-pc: ~  
mysql> select * from salary74;  
+-----+-----+-----+-----+  
| empno | basic  | deptno | tax  |  
+-----+-----+-----+-----+  
| 2     | 1000.00 | 1      | 50.00 |  
| 6     | 10100.00 | 3      | 1200.00 |  
+-----+-----+-----+-----+  
2 rows in set (0.00 sec)  
  
mysql> select * from employee74;  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
| empno | name  | deptno | job          | hiredate | sal  | commission | dob       | city      | phone  |  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
| 1     | Tom   | 1      | Accountant   | 2000-07-10 | 1200 | 300         | 1975-03-01 | Chicago   | 19987263 |  
| 2     | Kevin | 1      | Accountant   | 2002-04-15 | 1000 | 200         | 1975-08-28 | Delhi     | 19987654 |  
| 3     | Munez | 1      | Accountant   | 1999-06-17 | 2300 | 500         | 1970-03-15 | Chicago   | 98647881 |  
| 4     | Diego | 1      | Accountant   | 1995-07-10 | 10200 | 1300        | 1955-03-01 | Chicago   | 49987261 |  
| 5     | Sam   | 2      | Hiring Manager | 1995-03-10 | 11200 | 1350        | 1957-03-01 | Washington | 39987261 |  
| 6     | Peter | 3      | Salesman     | 1996-07-10 | 10100 | 1250        | 1953-03-01 | Delhi     | 39987268 |  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
6 rows in set (0.00 sec)  
  
mysql> SELECT * FROM employee74 WHERE empno IN (SELECT empno FROM salary74 WHERE tax = (SELECT MAX(tax) FROM salary74  
));  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
| empno | name  | deptno | job          | hiredate | sal  | commission | dob       | city      | phone  |  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
| 6     | Peter | 3      | Salesman     | 1996-07-10 | 10100 | 1250        | 1953-03-01 | Delhi     | 39987268 |  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
1 row in set (0.00 sec)  
  
mysql>
```

7) Which employee is the senior most?

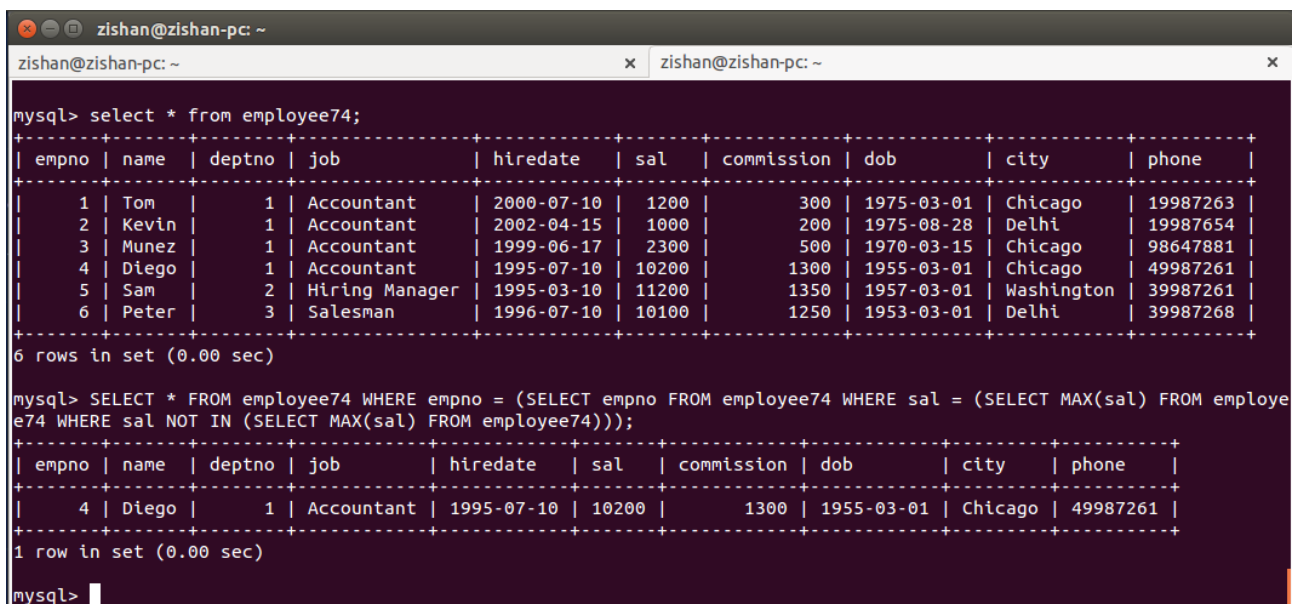
```
SELECT * FROM employee74 WHERE empno IN (  
    SELECT empno FROM employee74 WHERE hiredate = (  
        SELECT MIN(hiredate) FROM employee74  
    )  
);
```

```
zishan@zishan-pc: ~  
mysql> SELECT * FROM employee74 WHERE empno IN (SELECT empno FROM employee74 WHERE hiredate = (SELECT MIN(hiredate) FROM employee74));  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
| empno | name | deptno | job          | hiredate | sal  | commission | dob       | city       | phone  |  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
| 5      | Sam  | 2      | Hiring Manager | 1995-03-10 | 11200 | 1350      | 1957-03-01 | Washington | 39987261 |  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
1 row in set (0.00 sec)  
  
mysql> select * from employee74;  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
| empno | name | deptno | job          | hiredate | sal  | commission | dob       | city       | phone  |  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
| 1      | Tom  | 1      | Accountant   | 2000-07-10 | 1200 | 300       | 1975-03-01 | Chicago    | 19987263 |  
| 2      | Kevin | 1      | Accountant   | 2002-04-15 | 1000 | 200       | 1975-08-28 | Delhi      | 19987654 |  
| 3      | Munez | 1      | Accountant   | 1999-06-17 | 2300 | 500       | 1970-03-15 | Chicago    | 98647881 |  
| 4      | Diego | 1      | Accountant   | 1995-07-10 | 10200 | 1300      | 1955-03-01 | Chicago    | 49987261 |  
| 5      | Sam  | 2      | Hiring Manager | 1995-03-10 | 11200 | 1350      | 1957-03-01 | Washington | 39987261 |  
| 6      | Peter | 3      | Salesman     | 1996-07-10 | 10100 | 1250      | 1953-03-01 | Delhi      | 39987268 |  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
6 rows in set (0.00 sec)  
  
mysql>
```



8) Give the details of second highest salary employee (without use of < operator).

```
SELECT * FROM employee74 WHERE empno = (  
    SELECT empno FROM employee74 WHERE sal = (  
        SELECT MAX(sal) FROM employee74 WHERE sal NOT IN (  
            SELECT MAX(sal) FROM employee74  
        )  
    )  
);
```



The screenshot shows a terminal window with two tabs, both titled 'zishan@zishan-pc: ~'. The first tab displays the output of the query 'select \* from employee74;', showing a table with 6 rows. The second tab displays the output of a more complex query that finds the employee with the second highest salary, showing a single row for Diego.

```
mysql> select * from employee74;  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
| empno | name  | deptno | job          | hiredate | sal  | commission | dob       | city      | phone  |  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
| 1     | Tom   | 1      | Accountant   | 2000-07-10 | 1200 | 300        | 1975-03-01 | Chicago   | 19987263 |  
| 2     | Kevin | 1      | Accountant   | 2002-04-15 | 1000 | 200        | 1975-08-28 | Delhi     | 19987654 |  
| 3     | Munez | 1      | Accountant   | 1999-06-17 | 2300 | 500        | 1970-03-15 | Chicago   | 98647881 |  
| 4     | Diego | 1      | Accountant   | 1995-07-10 | 10200 | 1300       | 1955-03-01 | Chicago   | 49987261 |  
| 5     | Sam   | 2      | Hiring Manager | 1995-03-10 | 11200 | 1350       | 1957-03-01 | Washington | 39987261 |  
| 6     | Peter | 3      | Salesman     | 1996-07-10 | 10100 | 1250       | 1953-03-01 | Delhi     | 39987268 |  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
6 rows in set (0.00 sec)  
  
mysql> SELECT * FROM employee74 WHERE empno = (SELECT empno FROM employee74 WHERE sal = (SELECT MAX(sal) FROM employee74 WHERE sal NOT IN (SELECT MAX(sal) FROM employee74)));  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
| empno | name  | deptno | job          | hiredate | sal  | commission | dob       | city      | phone  |  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
| 4     | Diego | 1      | Accountant   | 1995-07-10 | 10200 | 1300       | 1955-03-01 | Chicago   | 49987261 |  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
1 row in set (0.00 sec)  
  
mysql>
```

9) Give the details of second highest salary employee (without use of max and limit operator).

```
SELECT * FROM employee74 AS e WHERE 2 = (  
    SELECT COUNT(DISTINCT sal) FROM employee74 WHERE e.sal <= sal  
);
```

```
zishan@zishan-pc: ~  
zishan@zishan-pc: ~ x zishan@zishan-pc: ~ x  
1 row in set (0.00 sec)  
  
mysql> select * from employee74;  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
| empno | name  | deptno | job      | hiredate | sal  | commission | dob       | city      | phone  |  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
| 1     | Tom   | 1       | Accountant | 2000-07-10 | 1200 | 300        | 1975-03-01 | Chicago  | 19987263 |  
| 2     | Kevin | 1       | Accountant | 2002-04-15 | 1000 | 200        | 1975-08-28 | Delhi    | 19987654 |  
| 3     | Munez | 1       | Accountant | 1999-06-17 | 2300 | 500        | 1970-03-15 | Chicago  | 98647881 |  
| 4     | Diego | 1       | Accountant | 1995-07-10 | 10200 | 1300       | 1955-03-01 | Chicago  | 49987261 |  
| 5     | Sam   | 2       | Hiring Manager | 1995-03-10 | 11200 | 1350       | 1957-03-01 | Washington | 39987261 |  
| 6     | Peter | 3       | Salesman   | 1996-07-10 | 10100 | 1250       | 1953-03-01 | Delhi    | 39987268 |  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
6 rows in set (0.00 sec)  
  
mysql> select * from employee74 as e where 2=(select count(distinct sal) from employee74 where e.sal <= sal);  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
| empno | name  | deptno | job      | hiredate | sal  | commission | dob       | city      | phone  |  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
| 4     | Diego | 1       | Accountant | 1995-07-10 | 10200 | 1300       | 1955-03-01 | Chicago  | 49987261 |  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
1 row in set (0.00 sec)  
  
mysql> 
```

```
SELECT * FROM employee74 ORDER BY sal DESC LIMIT 1,1;
```

11) Give the details of all employees of 5th highest salary ( or nth highest salary).

```
SELECT * FROM employee74 AS e WHERE 5 = (  
    SELECT COUNT(DISTINCT sal) FROM employee74 WHERE e.sal <= sal  
);
```

```
zishan@zishan-pc: ~  
zishan@zishan-pc: ~  
1 row in set (0.00 sec)  
  
mysql> select * from employee74;  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
| empno | name  | deptno | job          | hiredate | sal  | commission | dob       | city      | phone  |  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
| 1 | Tom  | 1 | Accountant | 2000-07-10 | 1200 | 300 | 1975-03-01 | Chicago | 19987263 |  
| 2 | Kevin | 1 | Accountant | 2002-04-15 | 1000 | 200 | 1975-08-28 | Delhi  | 19987654 |  
| 3 | Munez | 1 | Accountant | 1999-06-17 | 2300 | 500 | 1970-03-15 | Chicago | 98647881 |  
| 4 | Diego | 1 | Accountant | 1995-07-10 | 10200 | 1300 | 1955-03-01 | Chicago | 49987261 |  
| 5 | Sam  | 2 | Hiring Manager | 1995-03-10 | 11200 | 1350 | 1957-03-01 | Washington | 39987261 |  
| 6 | Peter | 3 | Salesman | 1996-07-10 | 10100 | 1250 | 1953-03-01 | Delhi  | 39987268 |  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
6 rows in set (0.00 sec)  
  
mysql> SELECT * FROM employee74 AS e WHERE 5 = (SELECT COUNT(DISTINCT sal) FROM employee74 WHERE e.sal <= sal);  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
| empno | name  | deptno | job          | hiredate | sal  | commission | dob       | city      | phone  |  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
| 1 | Tom  | 1 | Accountant | 2000-07-10 | 1200 | 300 | 1975-03-01 | Chicago | 19987263 |  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
1 row in set (0.00 sec)  
  
mysql>
```

12) How many clerks are there in the company?

`SELECT COUNT(*) FROM employee74 WHERE job = 'Clerk';`

```
zishan@zishan-pc: ~  
mysql> select * from employee74;  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
| empno | name  | deptno | job          | hiredate | sal  | commission | dob       | city       | phone  |  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
| 1     | Tom   | 1      | Accountant   | 2000-07-10 | 1200 | 300        | 1975-03-01 | Chicago    | 19987263 |  
| 2     | Kevin | 1      | Accountant   | 2002-04-15 | 1000 | 200        | 1975-08-28 | Delhi      | 19987654 |  
| 3     | Munez | 1      | Accountant   | 1999-06-17 | 2300 | 500        | 1970-03-15 | Chicago    | 98647881 |  
| 4     | Diego | 1      | Accountant   | 1995-07-10 | 10200 | 1300       | 1955-03-01 | Chicago    | 49987261 |  
| 5     | Sam   | 2      | Hiring Manager | 1995-03-10 | 11200 | 1350       | 1957-03-01 | Washington | 39987261 |  
| 6     | Peter | 3      | Salesman     | 1996-07-10 | 10100 | 1250       | 1953-03-01 | Delhi      | 39987268 |  
| 7     | Jack  | 2      | Clerk        | 1997-08-29 | 1500 | 250        | 1980-05-09 | Delhi      | 89943728 |  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
7 rows in set (0.00 sec)  
  
mysql> SELECT COUNT(*) FROM employee74 WHERE job = 'Clerk';  
+-----+  
| COUNT(*) |  
+-----+  
| 1 |  
+-----+  
1 row in set (0.00 sec)  
  
mysql>
```

13) Which department has exactly one employee as clerk?

```
SELECT d.dname FROM department74 AS d JOIN employee74 AS e
      WHERE d.deptno = e.deptno AND e.job = 'Clerk'
HAVING COUNT(*) = 1;
```

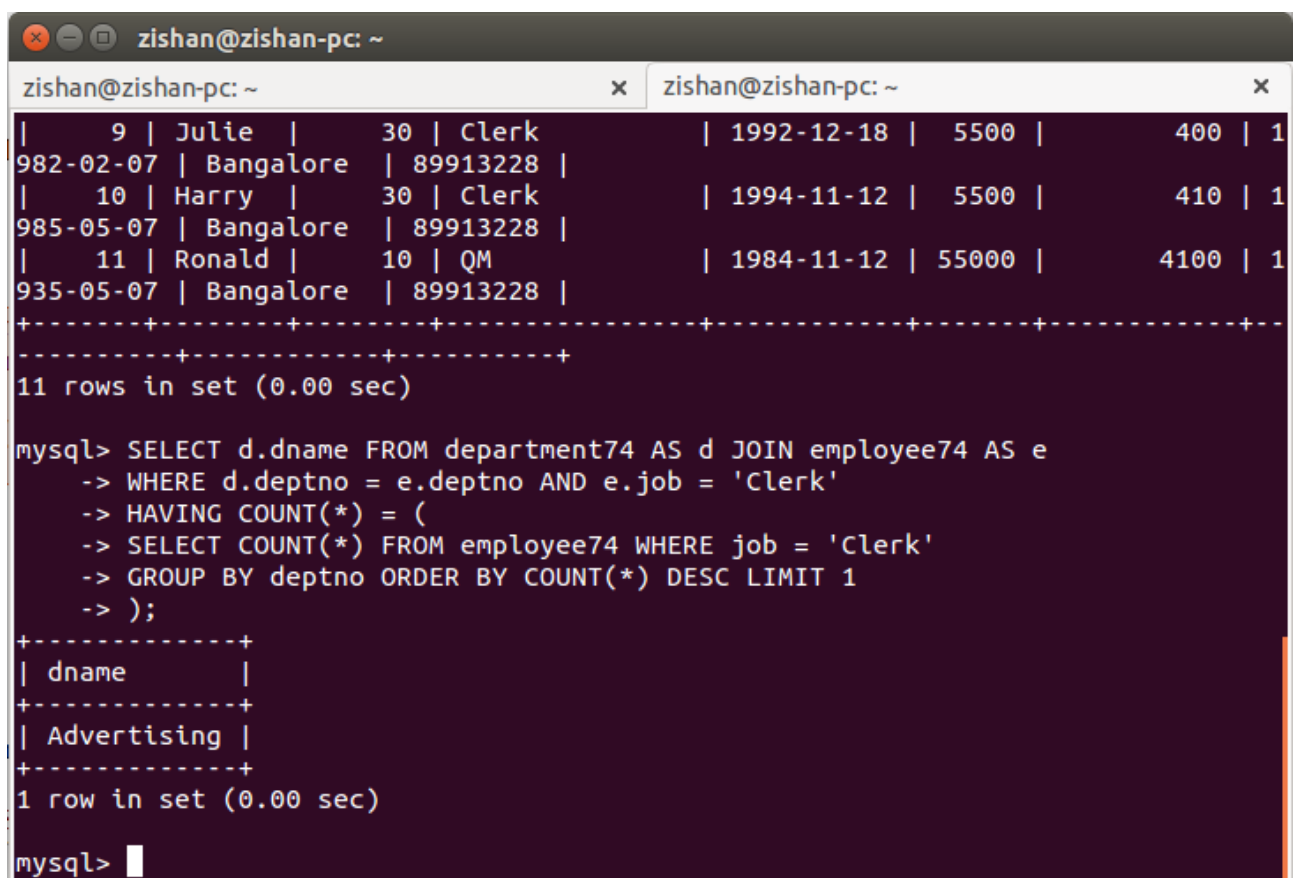
```
zishan@zishan-pc: ~
mysql> SELECT * FROM employee74;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| empno | name  | deptno | job          | hiredate | sal  | commission | dob       | city      | phone  |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1     | Tom   | 1       | Accountant   | 2000-07-10 | 1200 | 300         | 1975-03-01 | Chicago   | 19987263 |
| 2     | Kevin | 1       | Accountant   | 2002-04-15 | 1000 | 200         | 1975-08-28 | Delhi     | 19987654 |
| 3     | Munez | 1       | Accountant   | 1999-06-17 | 2300 | 500         | 1970-03-15 | Chicago   | 98647881 |
| 4     | Diego | 1       | Accountant   | 1995-07-10 | 10200 | 1300        | 1955-03-01 | Chicago   | 49987261 |
| 5     | Sam   | 2       | Hiring Manager | 1995-03-10 | 11200 | 1350        | 1957-03-01 | Washington | 39987261 |
| 6     | Peter | 3       | Salesman     | 1996-07-10 | 10100 | 1250        | 1953-03-01 | Delhi     | 39987268 |
| 7     | Jack  | 2       | Clerk        | 1997-08-29 | 1500 | 250         | 1980-05-09 | Delhi     | 89943728 |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
7 rows in set (0.00 sec)

mysql> SELECT d.dname FROM department74 AS d JOIN employee74 AS e WHERE d.deptno = e.deptno AND e.job = 'Clerk' HAVING COUNT(*) = 1;
+-----+
| dname |
+-----+
| HR     |
+-----+
1 row in set (0.01 sec)

mysql>
```

14) Which department has the highest number of clerks? Show the deptno and count.

```
SELECT d.dname FROM department74 AS d JOIN employee74 AS e
  WHERE d.deptno = e.deptno AND e.job = 'Clerk'
  HAVING COUNT(*) = (
    SELECT COUNT(*) FROM employee74 WHERE job = 'Clerk'
    GROUP BY deptno ORDER BY COUNT(*) DESC LIMIT 1
  );
```



The screenshot shows a terminal window with a dark background. At the top, there are two tabs labeled 'zishan@zishan-pc: ~'. The terminal displays a table of employee data with columns for employee ID, name, department number, job title, hire date, salary, and commission. The data is as follows:

empno	ename	deptno	job	hiredate	salary	comm
9	Julie	30	Clerk	1992-12-18	5500	400
10	Harry	30	Clerk	1994-11-12	5500	410
11	Ronald	10	QM	1984-11-12	55000	4100

Below the table, it says '11 rows in set (0.00 sec)'. Then, a MySQL prompt 'mysql>' is followed by the same SQL query as above. The output shows a single row for the 'Advertising' department.

dname
Advertising

Below the table, it says '1 row in set (0.00 sec)'. The prompt 'mysql>' is shown again at the bottom.

15) How many employees are there in each department?

```
SELECT d.dname, COUNT(*) AS 'Number of Employees'
FROM employee74 AS e, department74 AS d
WHERE e.deptno = d.deptno GROUP BY e.deptno;
```

```
zishan@zishan-pc: ~
x zishan@zishan-pc: ~ x
```

empno	name	deptno	job	hiredate	sal	commission	dob	city	phone
1	Tom	1	Accountant	2000-07-10	1200	300	1975-03-01	Chicago	19987263
2	Kevin	1	Accountant	2002-04-15	1000	200	1975-08-28	Delhi	19987654
3	Munez	1	Accountant	1999-06-17	2300	500	1970-03-15	Chicago	98647881
4	Diego	1	Accountant	1995-07-10	10200	1300	1955-03-01	Chicago	49987261
5	Sam	2	Hiring Manager	1995-03-10	11200	1350	1957-03-01	Washington	39987261
6	Peter	3	Salesman	1996-07-10	10100	1250	1953-03-01	Delhi	39987268
7	Jack	2	Clerk	1997-08-29	1500	250	1980-05-09	Delhi	89943728

```
7 rows in set (0.00 sec)
```

```
mysql> SELECT d.dname, COUNT(*) AS 'Number of Employees' FROM employee74 AS e, department74 AS d WHERE e.deptno = d.deptno GROUP BY e.deptno;
```

dname	Number of Employees
Accounting	4
HR	2
Sales	1

```
3 rows in set (0.00 sec)
```

```
mysql>
```



16) List the lowest salary for different jobs used in a company and list them in descending order.

```
SELECT job, MIN(sal) AS 'Minimum Salary' FROM employee74  
GROUP BY job ORDER BY sal DESC;
```

```
zishan@zishan-pc: ~  
mysql> SELECT empno, name, deptno, job, hiredate, sal, commission, dob, city, phone  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
| empno | name  | deptno | job          | hiredate | sal  | commission | dob       | city       | phone  |  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
| 1 | Tom  | 1 | Accountant  | 2000-07-10 | 1200 | 300 | 1975-03-01 | Chicago   | 19987263 |  
| 2 | Kevin | 1 | Accountant  | 2002-04-15 | 1000 | 200 | 1975-08-28 | Delhi     | 19987654 |  
| 3 | Munez | 1 | Accountant  | 1999-06-17 | 2300 | 500 | 1970-03-15 | Chicago   | 98647881 |  
| 4 | Diego | 1 | Accountant  | 1995-07-10 | 10200 | 1300 | 1955-03-01 | Chicago   | 49987261 |  
| 5 | Sam  | 2 | Hiring Manager | 1995-03-10 | 11200 | 1350 | 1957-03-01 | Washington | 39987261 |  
| 6 | Peter | 3 | Salesman    | 1996-07-10 | 10100 | 1250 | 1953-03-01 | Delhi     | 39987268 |  
| 7 | Jack  | 2 | Clerk       | 1997-08-29 | 1500 | 250 | 1980-05-09 | Delhi     | 89943728 |  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
7 rows in set (0.00 sec)  
  
mysql> SELECT job, MIN(sal) AS 'Minimum Salary' FROM employee74 GROUP BY job ORDER BY sal DESC;  
+-----+-----+  
| job          | Minimum Salary |  
+-----+-----+  
| Hiring Manager | 11200 |  
| Salesman      | 10100 |  
| Clerk         | 1500 |  
| Accountant     | 1000 |  
+-----+-----+  
4 rows in set (0.00 sec)  
  
mysql>
```

17) Which department average salary is the lowest among all? Show the deptno,average salary.

```
SELECT d.dname, AVG(sal) FROM department74 AS d JOIN employee74 AS e
      WHERE d.deptno = e.deptno
GROUP BY e.deptno ORDER BY AVG(sal) ASC LIMIT 1;
```

```
zishan@zishan-pc: ~
mysql> select * from employee74;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| empno | name  | deptno | job          | hiredate | sal  | commission | dob       | city       | phone    |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1     | Tom   | 1       | Accountant   | 2000-07-10 | 1200 | 300        | 1975-03-01 | Chicago    | 19987263 |
| 2     | Kevin | 1       | Accountant   | 2002-04-15 | 1000 | 200        | 1975-08-28 | Delhi      | 19987654 |
| 3     | Munez | 1       | Accountant   | 1999-06-17 | 2300 | 500        | 1970-03-15 | Chicago    | 98647881 |
| 4     | Diego | 1       | Accountant   | 1995-07-10 | 10200 | 1300       | 1955-03-01 | Chicago    | 49987261 |
| 5     | Sam   | 2       | Hiring Manager | 1995-03-10 | 11200 | 1350       | 1957-03-01 | Washington | 39987261 |
| 6     | Peter | 3       | Salesman     | 1996-07-10 | 10100 | 1250       | 1953-03-01 | Delhi      | 39987268 |
| 7     | Jack  | 2       | Clerk        | 1997-08-29 | 1500 | 250        | 1980-05-09 | Delhi      | 89943728 |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
7 rows in set (0.00 sec)

mysql> SELECT d.dname, AVG(sal) FROM department74 AS d JOIN employee74 AS e WHERE d.deptno = e.deptno GROUP BY e.deptno ORDER BY AVG(sal) ASC LIMIT 1;
+-----+-----+
| dname | AVG(sal) |
+-----+-----+
| Accounting | 3675 |
+-----+-----+
1 row in set (0.00 sec)

mysql>
```

18) List the minimum, maximum and average salary for each job.

SELECT job, MAX(sal), MIN(sal), AVG(sal) FROM employee74 GROUP BY job;

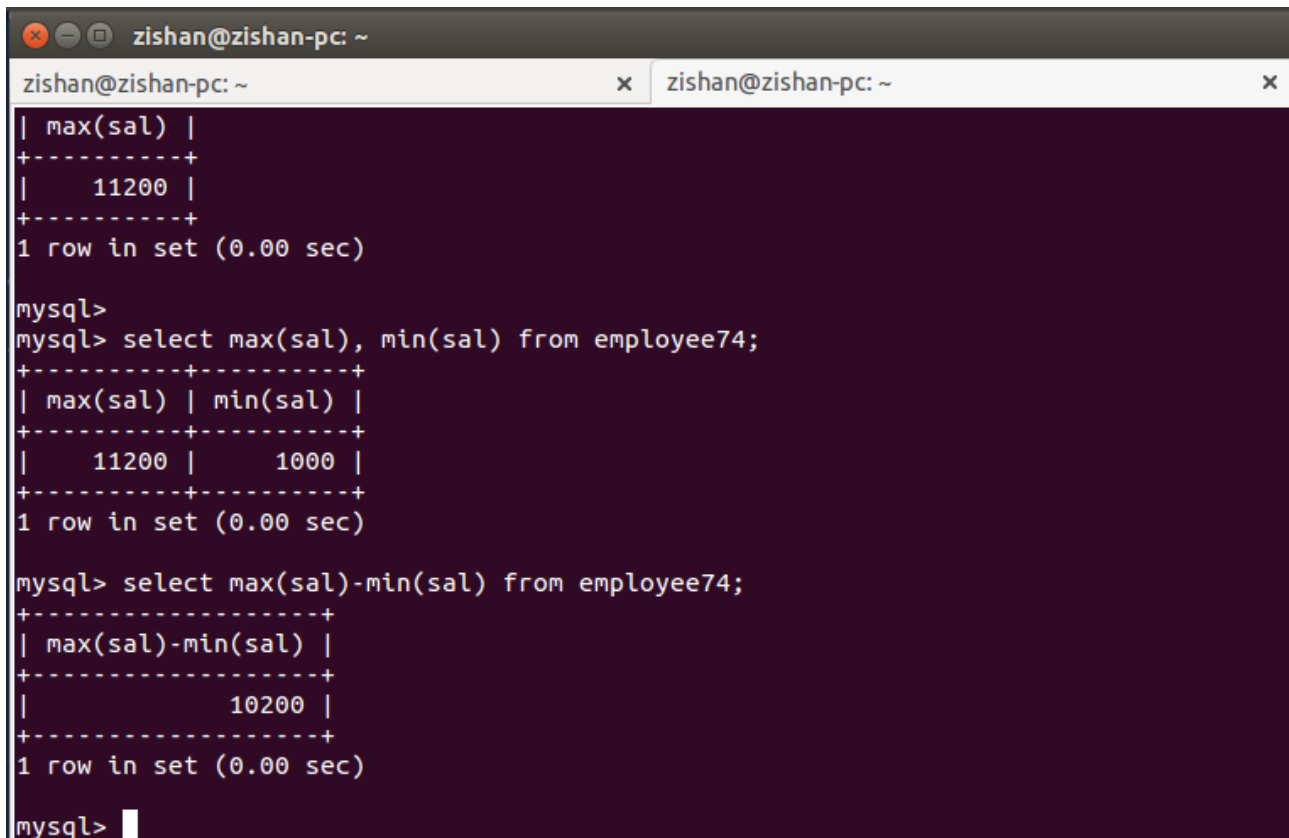
```
zishan@zishan-pc: ~
mysql> SELECT d.dname, AVG(sal) FROM department74 AS d JOIN employee74 AS e WHERE d.
ptno ORDER BY AVG(sal) ASC LIMIT 1;
+-----+-----+
| dname      | AVG(sal) |
+-----+-----+
| Accounting |    3675 |
+-----+-----+
1 row in set (0.00 sec)

mysql> SELECT job, MAX(sal), MIN(sal), AVG(sal) FROM employee74 GROUP BY job;
+-----+-----+-----+-----+
| job          | MAX(sal) | MIN(sal) | AVG(sal) |
+-----+-----+-----+-----+
| Accountant   |    10200 |     1000 |    3675 |
| Clerk        |     1500 |     1500 |    1500 |
| Hiring Manager |    11200 |    11200 |    11200 |
| Salesman     |     10100 |     10100 |    10100 |
+-----+-----+-----+-----+
4 rows in set (0.00 sec)

mysql>
```

19) Compute the difference between maximum and minimum salary.

`SELECT MAX(sal) - MIN(sal) FROM employee74;`



```
zishan@zishan-pc: ~  
zishan@zishan-pc: ~  
| max(sal) |  
+-----+  
|    11200 |  
+-----+  
1 row in set (0.00 sec)  
  
mysql>  
mysql> select max(sal), min(sal) from employee74;  
+-----+-----+  
| max(sal) | min(sal) |  
+-----+-----+  
|    11200 |     1000 |  
+-----+-----+  
1 row in set (0.00 sec)  
  
mysql> select max(sal)-min(sal) from employee74;  
+-----+  
| max(sal)-min(sal) |  
+-----+  
|           10200 |  
+-----+  
1 row in set (0.00 sec)  
  
mysql> 
```

20) List the names of the employees whose name contains LA.

```
SELECT name FROM employee74 WHERE name LIKE '%la%';
```

```
zishan@zishan-pc: ~  
zishan@zishan-pc: ~ x zishan@zishan-pc: ~ x  
3-15 | Chicago | 98647881 |  
| 4 | Diego | 1 | Accountant | 1995-07-10 | 10200 | 1300 | 1955-0  
3-01 | Chicago | 49987261 |  
| 5 | Sam | 2 | Hiring Manager | 1995-03-10 | 11200 | 1350 | 1957-0  
3-01 | Washington | 39987261 |  
| 6 | Peter | 3 | Salesman | 1996-07-10 | 10100 | 1250 | 1953-0  
3-01 | Delhi | 39987268 |  
| 7 | Jack | 2 | Clerk | 1997-08-29 | 1500 | 250 | 1980-0  
5-09 | Delhi | 89943728 |  
| 8 | Blair | 2 | Clerk | 1996-12-18 | 5500 | 400 | 1981-0  
2-07 | Bangalore | 89913228 |  
+-----+-----+-----+-----+-----+-----+-----+-----+  
-----+-----+-----+  
8 rows in set (0.00 sec)  
  
mysql> SELECT name FROM employee74 WHERE name LIKE '%la%';  
+-----+  
| name |  
+-----+  
| Blair |  
+-----+  
1 row in set (0.00 sec)  
  
mysql> 
```

21) List the names of the employees whose joining date is between 2nd April,1981 and 8<sup>th</sup> Sept,1981.

SELECT name FROM employee74 WHERE hitedate  
BETWEEN '1981-04-02' AND '1981-09-08';

```
zishan@zishan-pc: ~
mysql> select * from employee74;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| empno | name  | deptno | job          | hiredate | sal  | commission | dob       | city      | phone  |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1     | Tom   | 1       | Accountant   | 2000-07-10 | 1200 | 300         | 1975-03-01 | Chicago   | 19987263 |
| 2     | Kevin | 1       | Accountant   | 1981-06-03 | 1000 | 200         | 1975-08-28 | Delhi     | 19987654 |
| 3     | Munez | 1       | Accountant   | 1999-06-17 | 2300 | 500         | 1970-03-15 | Chicago   | 98647881 |
| 4     | Diego | 1       | Accountant   | 1995-07-10 | 10200 | 1300        | 1955-03-01 | Chicago   | 49987261 |
| 5     | Sam   | 2       | Hiring Manager | 1995-03-10 | 11200 | 1350        | 1957-03-01 | Washington | 39987261 |
| 6     | Peter | 3       | Salesman     | 1996-07-10 | 10100 | 1250        | 1953-03-01 | Delhi     | 39987268 |
| 7     | Jack  | 2       | Clerk        | 1997-08-29 | 1500 | 250         | 1980-05-09 | Delhi     | 89943728 |
| 8     | Blair | 2       | Clerk        | 1996-12-18 | 5500 | 400         | 1981-02-07 | Bangalore | 89913228 |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
8 rows in set (0.00 sec)

mysql> SELECT name FROM employee74 WHERE hiredate BETWEEN '1981-04-02' AND '1981-09-08';
+-----+
| name  |
+-----+
| Kevin |
+-----+
1 row in set (0.00 sec)

mysql>
```

22) How many different job titles exist in the employee table?

SELECT job FROM employee74 GROUP BY job;

```
zishan@zishan-pc: ~  
zishan@zishan-pc: ~ x zishan@zishan-pc: ~ x  
| 8 | Blair | 2 | Clerk | 1996-12-18 | 5500 | 400 | 1981-02-07 | Bangalore | 89913228 |  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
8 rows in set (0.00 sec)  
  
mysql> SELECT name FROM employee74 WHERE hiredate BETWEEN '1981-04-02' AND '1981-09-08';  
+-----+  
| name |  
+-----+  
| Kevin |  
+-----+  
1 row in set (0.00 sec)  
  
mysql> SELECT job FROM employee74 GROUP BY job;  
+-----+  
| job |  
+-----+  
| Accountant |  
| Clerk |  
| Hiring Manager |  
| Salesman |  
+-----+  
4 rows in set (0.00 sec)  
  
mysql>
```

23) Compute the sum of all salaries of employee working under deptno=30.

SELECT SUM(sal) FROM employee74 WHERE deptno=30;

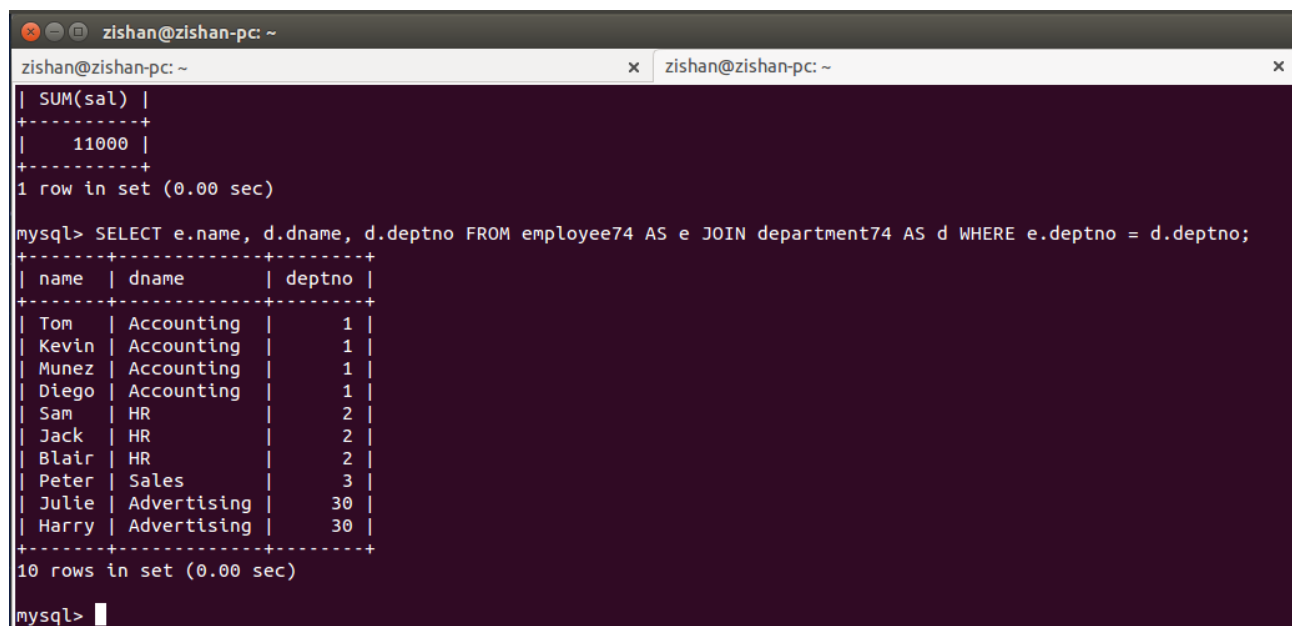
```
zishan@zishan-pc: ~  
mysql> SELECT SUM(sal) FROM employee74 WHERE deptno=30;  
+-----+  
| SUM(sal) |  
+-----+  
|    11000 |  
+-----+  
1 row in set (0.00 sec)  
  
mysql>
```

empno	name	deptno	job	hiredate	sal	commission	dob	city	phone
1	Tom	1	Accountant	2000-07-10	1200	300	1975-03-01	Chicago	19987263
2	Kevin	1	Accountant	1981-06-03	1000	200	1975-08-28	Delhi	19987654
3	Munez	1	Accountant	1999-06-17	2300	500	1970-03-15	Chicago	98647881
4	Diego	1	Accountant	1995-07-10	10200	1300	1955-03-01	Chicago	49987261
5	Sam	2	Hiring Manager	1995-03-10	11200	1350	1957-03-01	Washington	39987261
6	Peter	3	Salesman	1996-07-10	10100	1250	1953-03-01	Delhi	39987268
7	Jack	2	Clerk	1997-08-29	1500	250	1980-05-09	Delhi	89943728
8	Blair	2	Clerk	1996-12-18	5500	400	1981-02-07	Bangalore	89913228
9	Julie	30	Clerk	1992-12-18	5500	400	1982-02-07	Bangalore	89913228
10	Harry	30	Clerk	1994-11-12	5500	410	1985-05-07	Bangalore	89913228



24) For each salesman in the emp table retrieve the deptno and department name.

```
SELECT e.name, d.dname, d.deptno FROM employee74 AS e  
      JOIN department74 AS d WHERE e.deptno = d.deptno;
```



The image shows a terminal window with two tabs, both titled 'zishan@zishan-pc: ~'. The terminal displays the output of a SQL query and the execution of a new query.

```
zishan@zishan-pc: ~  
mysql> SELECT SUM(sal) FROM emp;  
+-----+  
| SUM(sal) |  
+-----+  
| 11000 |  
+-----+  
1 row in set (0.00 sec)  
  
mysql> SELECT e.name, d.dname, d.deptno FROM employee74 AS e JOIN department74 AS d WHERE e.deptno = d.deptno;  
+-----+-----+-----+  
| name | dname | deptno |  
+-----+-----+-----+  
| Tom | Accounting | 1 |  
| Kevin | Accounting | 1 |  
| Munez | Accounting | 1 |  
| Diego | Accounting | 1 |  
| Sam | HR | 2 |  
| Jack | HR | 2 |  
| Blair | HR | 2 |  
| Peter | Sales | 3 |  
| Julie | Advertising | 30 |  
| Harry | Advertising | 30 |  
+-----+-----+-----+  
10 rows in set (0.00 sec)  
  
mysql>
```

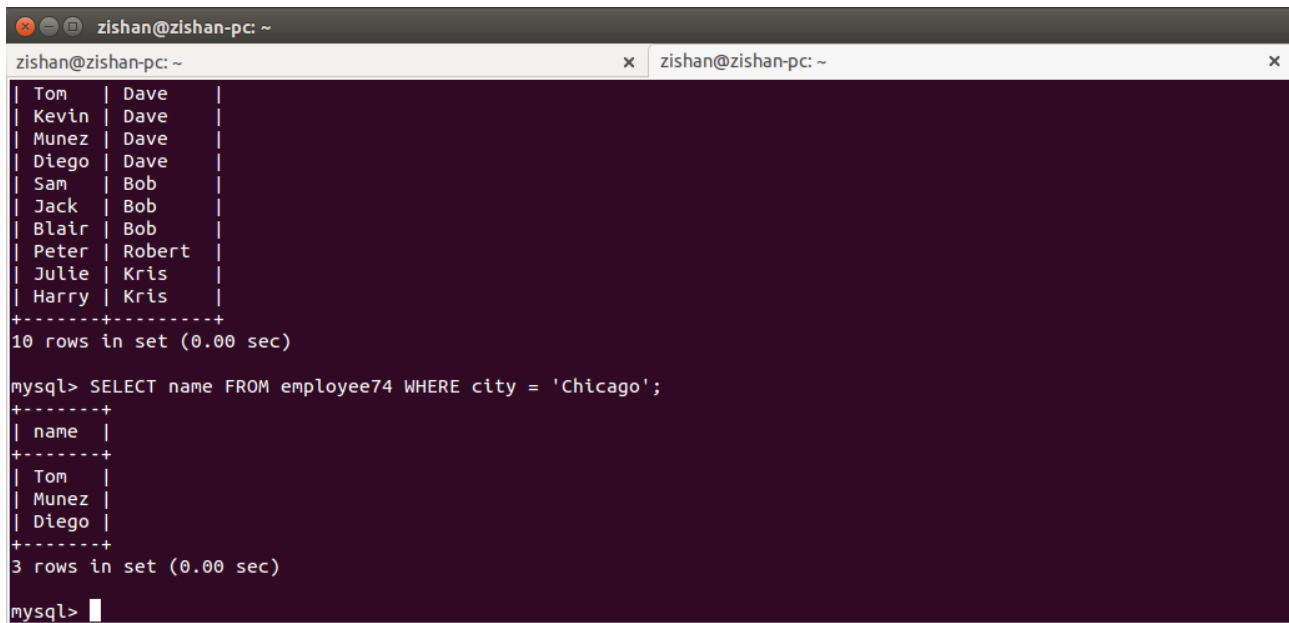
25) List the names of all the employees with their name of the manager .

SELECT e.name, d.manager FROM employee74 as e, department74 as d  
WHERE e.deptno = d.deptno;

```
zishan@zishan-pc: ~  
zishan@zishan-pc: ~  
+-----+  
| Peter | Sales | 3 |  
| Julie | Advertising | 30 |  
| Harry | Advertising | 30 |  
+-----+  
10 rows in set (0.00 sec)  
  
mysql> SELECT e.name, d.manager FROM employee74 as e, department74 as d WHERE e.deptno = d.deptno;  
+-----+  
| name | manager |  
+-----+  
| Tom | Dave |  
| Kevin | Dave |  
| Munez | Dave |  
| Diego | Dave |  
| Sam | Bob |  
| Jack | Bob |  
| Blair | Bob |  
| Peter | Robert |  
| Julie | Kris |  
| Harry | Kris |  
+-----+  
10 rows in set (0.00 sec)  
  
mysql> 
```

26) List all employees who are working in department located at CHICAGO.

`SELECT name FROM employee74 WHERE city = 'Chicago';`

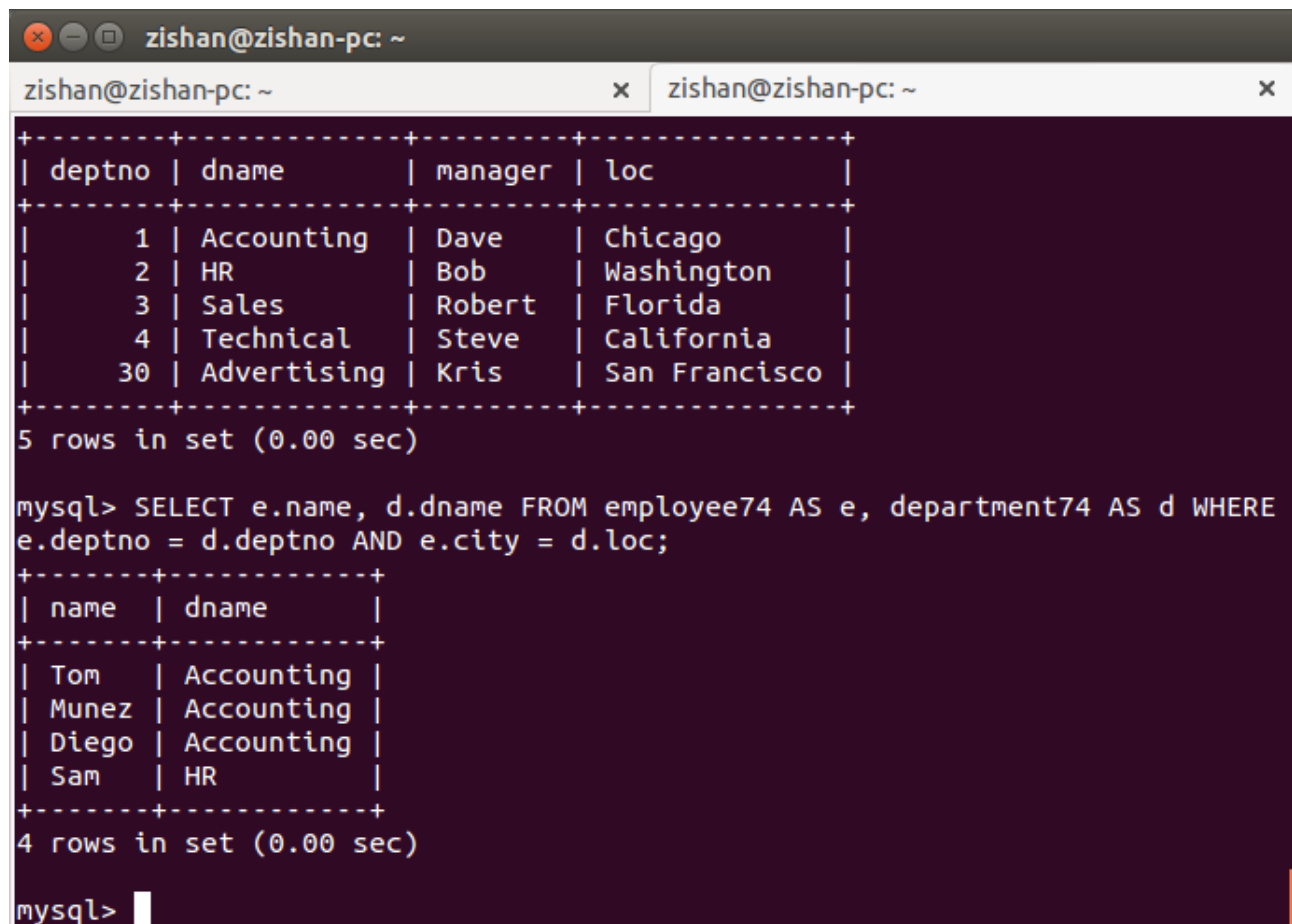


The screenshot shows a MySQL terminal window with the following content:

```
zishan@zishan-pc: ~  
zishan@zishan-pc: ~  
+-----+  
| Tom   | Dave |  
| Kevin | Dave |  
| Munez | Dave |  
| Diego | Dave |  
| Sam   | Bob  |  
| Jack  | Bob  |  
| Blair | Bob  |  
| Peter | Robert|  
| Julie | Kris |  
| Harry | Kris |  
+-----+  
10 rows in set (0.00 sec)  
  
mysql> SELECT name FROM employee74 WHERE city = 'Chicago';  
+-----+  
| name |  
+-----+  
| Tom  |  
| Munez|  
| Diego|  
+-----+  
3 rows in set (0.00 sec)  
  
mysql>
```

27) List all the employees who are working in same department as their managers.

```
SELECT e.name, d.dname FROM employee74 AS e, department74 AS d
WHERE e.deptno = d.deptno AND e.city = d.loc;
```



The image shows a terminal window with two tabs, both titled 'zishan@zishan-pc: ~'. The first tab displays a table with 5 rows of department data. The second tab shows the execution of a SQL query that filters employees based on their department and location, resulting in 4 rows.

```
zishan@zishan-pc: ~
zishan@zishan-pc: ~ x zishan@zishan-pc: ~ x
```

deptno	dname	manager	loc
1	Accounting	Dave	Chicago
2	HR	Bob	Washington
3	Sales	Robert	Florida
4	Technical	Steve	California
30	Advertising	Kris	San Francisco

5 rows in set (0.00 sec)

```
mysql> SELECT e.name, d.dname FROM employee74 AS e, department74 AS d WHERE
e.deptno = d.deptno AND e.city = d.loc;
```

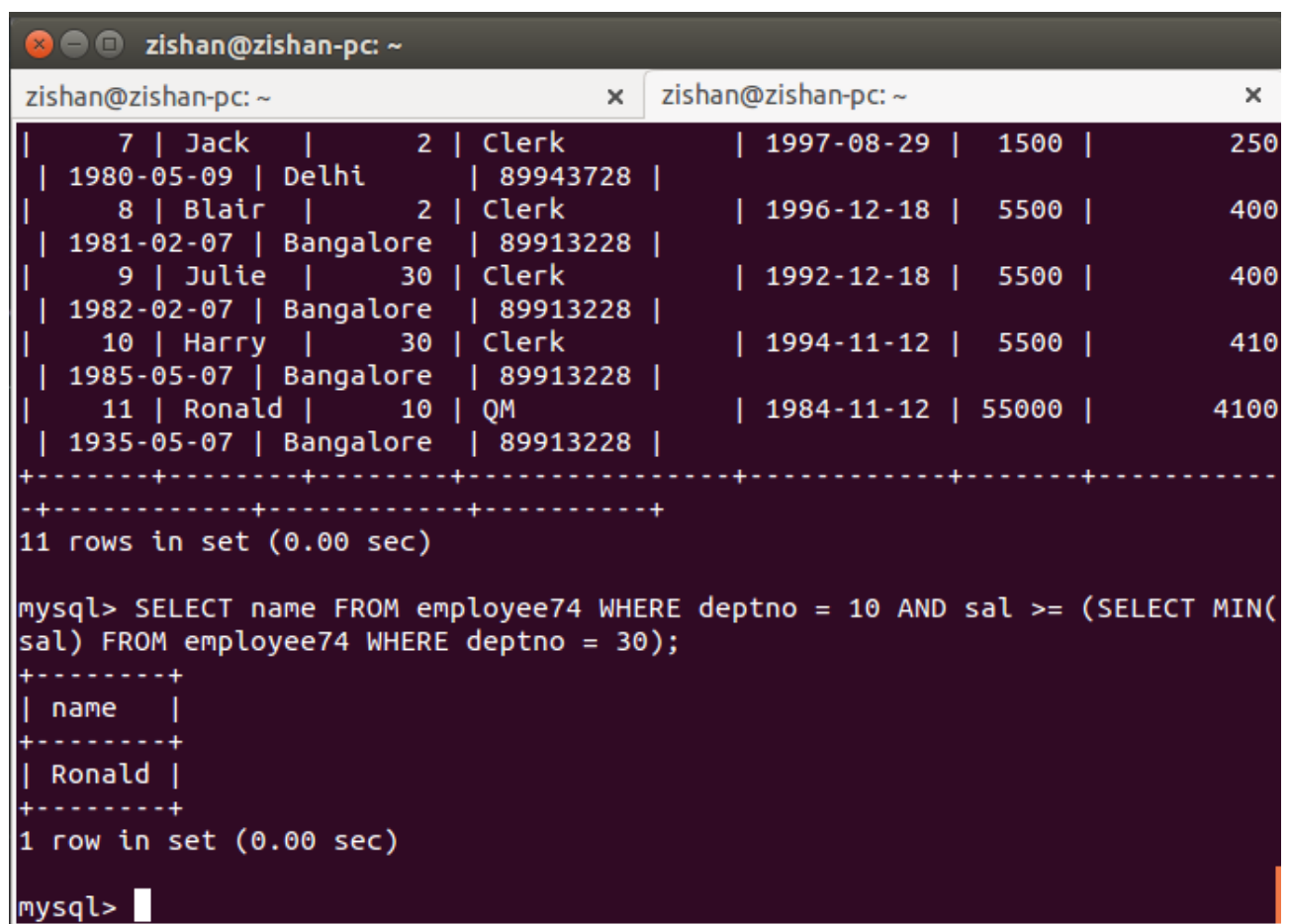
name	dname
Tom	Accounting
Munez	Accounting
Diego	Accounting
Sam	HR

4 rows in set (0.00 sec)

```
mysql>
```

28) Retrieve all the employees who are working in deptno=10 and who earn salary atleast as much as any employee working in deptno=30.

```
SELECT name FROM employee74 WHERE deptno = 10
AND sal >= (
    SELECT MIN(sal) FROM employee74 WHERE deptno = 30
);
```



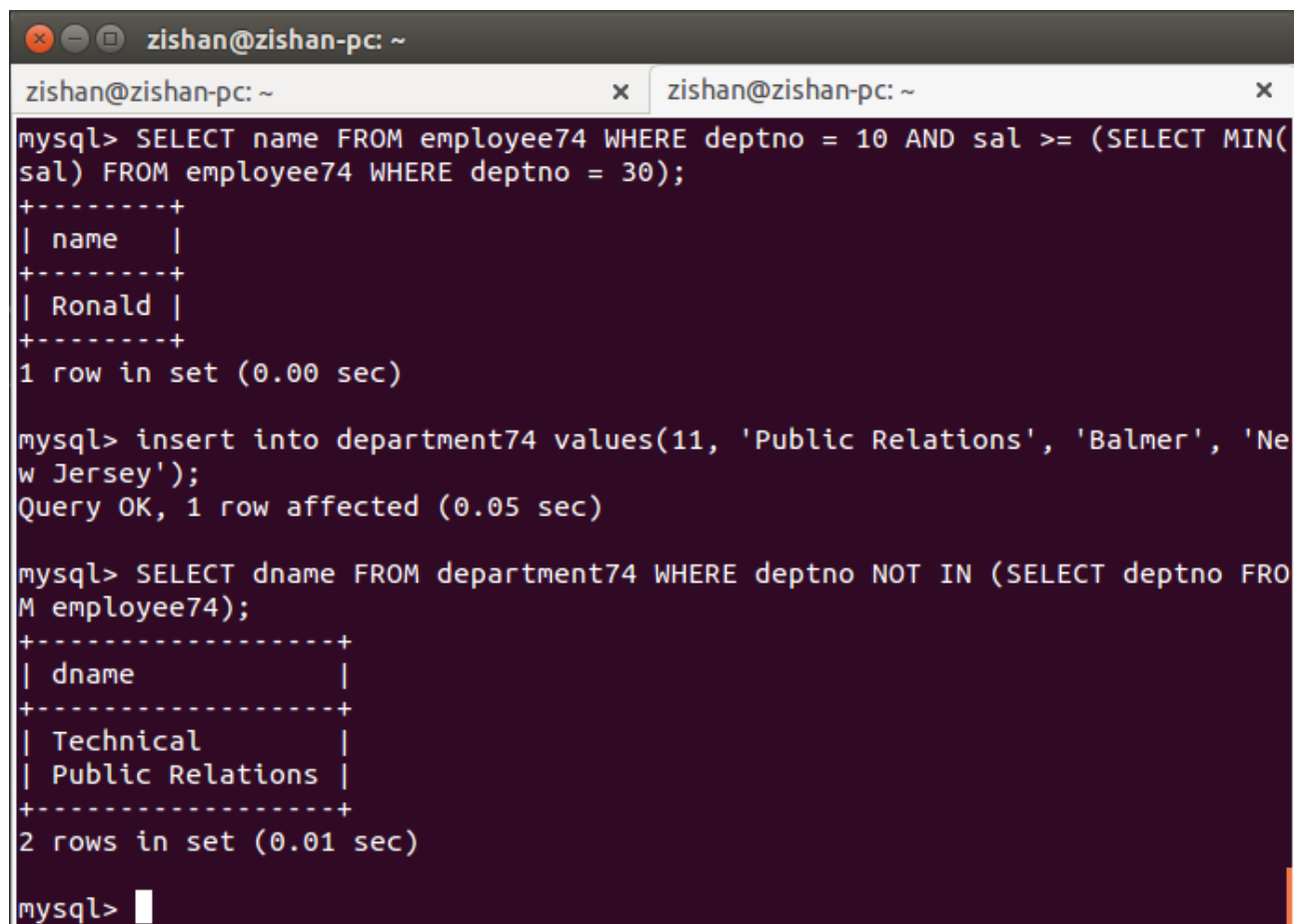
```
zishan@zishan-pc: ~
zishan@zishan-pc: ~ x zishan@zishan-pc: ~ x
| 7 | Jack | 2 | Clerk | 1997-08-29 | 1500 | 250
| 1980-05-09 | Delhi | 89943728 |
| 8 | Blair | 2 | Clerk | 1996-12-18 | 5500 | 400
| 1981-02-07 | Bangalore | 89913228 |
| 9 | Julie | 30 | Clerk | 1992-12-18 | 5500 | 400
| 1982-02-07 | Bangalore | 89913228 |
| 10 | Harry | 30 | Clerk | 1994-11-12 | 5500 | 410
| 1985-05-07 | Bangalore | 89913228 |
| 11 | Ronald | 10 | QM | 1984-11-12 | 55000 | 4100
| 1935-05-07 | Bangalore | 89913228 |
+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+
11 rows in set (0.00 sec)

mysql> SELECT name FROM employee74 WHERE deptno = 10 AND sal >= (SELECT MIN(
sal) FROM employee74 WHERE deptno = 30);
+-----+
| name |
+-----+
| Ronald |
+-----+
1 row in set (0.00 sec)

mysql>
```

29) List all the department who have no employees

```
SELECT dname FROM department74 WHERE deptno
      NOT IN (
          SELECT deptno FROM employee74
      );
```



The screenshot shows a terminal window with a dark background. The title bar indicates the user is 'zishan' on a machine named 'zishan-pc'. There are two tabs open, both showing the same path '~'. The terminal displays the following MySQL session:

```
mysql> SELECT name FROM employee74 WHERE deptno = 10 AND sal >= (SELECT MIN(sal) FROM employee74 WHERE deptno = 30);
+-----+
| name  |
+-----+
| Ronald |
+-----+
1 row in set (0.00 sec)
```

mysql> insert into department74 values(11, 'Public Relations', 'Balmer', 'New Jersey');
Query OK, 1 row affected (0.05 sec)

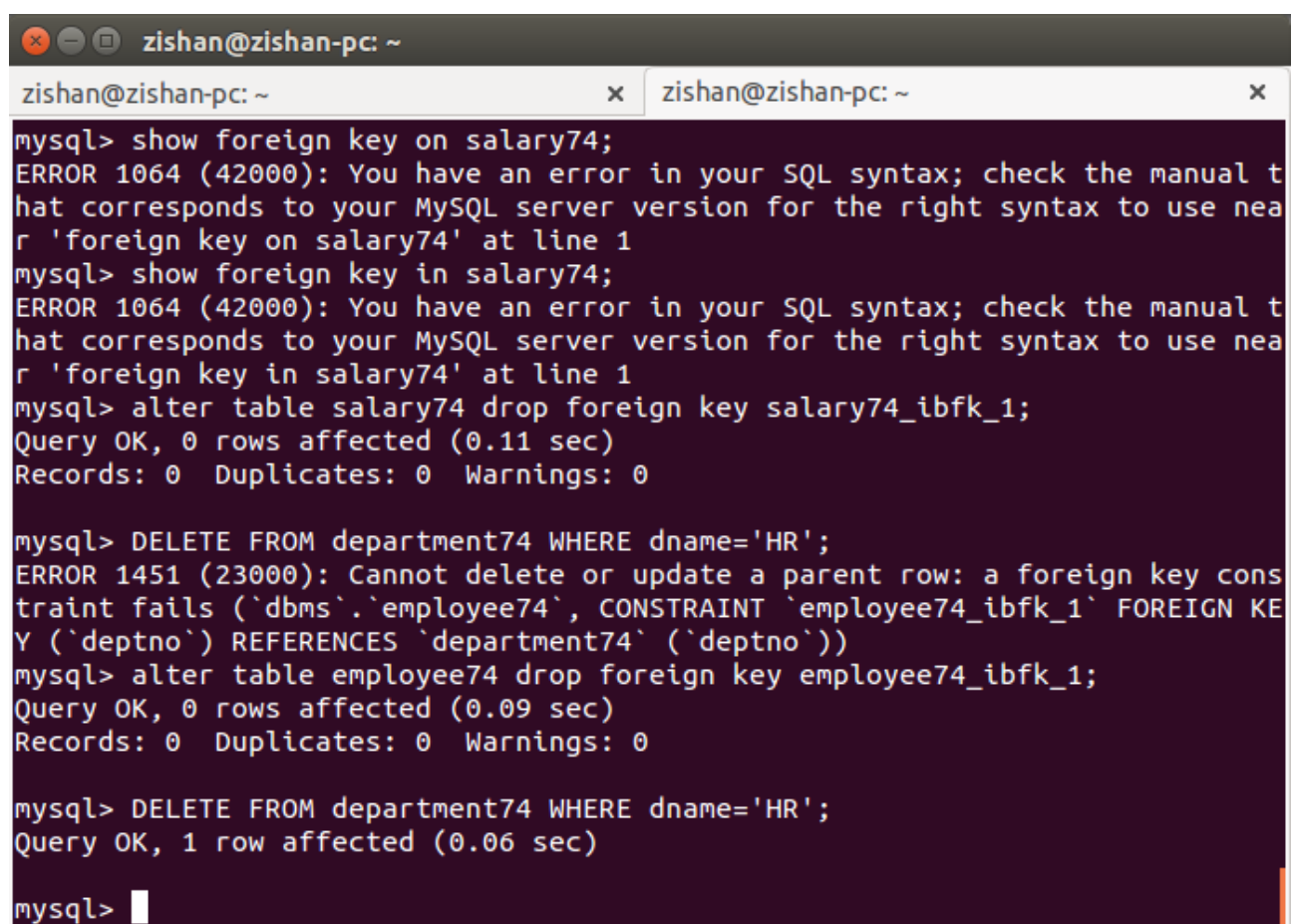
```
mysql> SELECT dname FROM department74 WHERE deptno NOT IN (SELECT deptno FROM employee74);
+-----+
| dname          |
+-----+
| Technical      |
| Public Relations |
+-----+
2 rows in set (0.01 sec)
```

mysql>

30) Delete the HR department.

```
ALTER TABLE employee74 DROP FOREIGN KEY employee74_ibfk_1;  
ALTER TABLE salary74 DROP FOREIGN KEY salary74_ibfk_1;  
DELETE FROM department74 WHERE dname='HR';
```

NOTE: Each table name end with your roll no. e.g. roll no is 11CSS23 then table name should be Employee123.

A terminal window titled 'zishan@zishan-pc: ~' with two tabs. The terminal shows a series of MySQL commands and their outputs. The first two commands, 'show foreign key on salary74;' and 'show foreign key in salary74;', both result in 'ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'foreign key on salary74' at line 1'. The third command, 'alter table salary74 drop foreign key salary74\_ibfk\_1;', succeeds with 'Query OK, 0 rows affected (0.11 sec)'. The fourth command, 'DELETE FROM department74 WHERE dname='HR';', results in 'ERROR 1451 (23000): Cannot delete or update a parent row: a foreign key constraint fails ('dbms`.`employee74`, CONSTRAINT `employee74\_ibfk\_1` FOREIGN KEY (`deptno`) REFERENCES `department74` (`deptno`))'. The fifth command, 'alter table employee74 drop foreign key employee74\_ibfk\_1;', succeeds with 'Query OK, 0 rows affected (0.09 sec)'. The sixth command, 'DELETE FROM department74 WHERE dname='HR';', succeeds with 'Query OK, 1 row affected (0.06 sec)'. The terminal ends with 'mysql>' and a cursor.

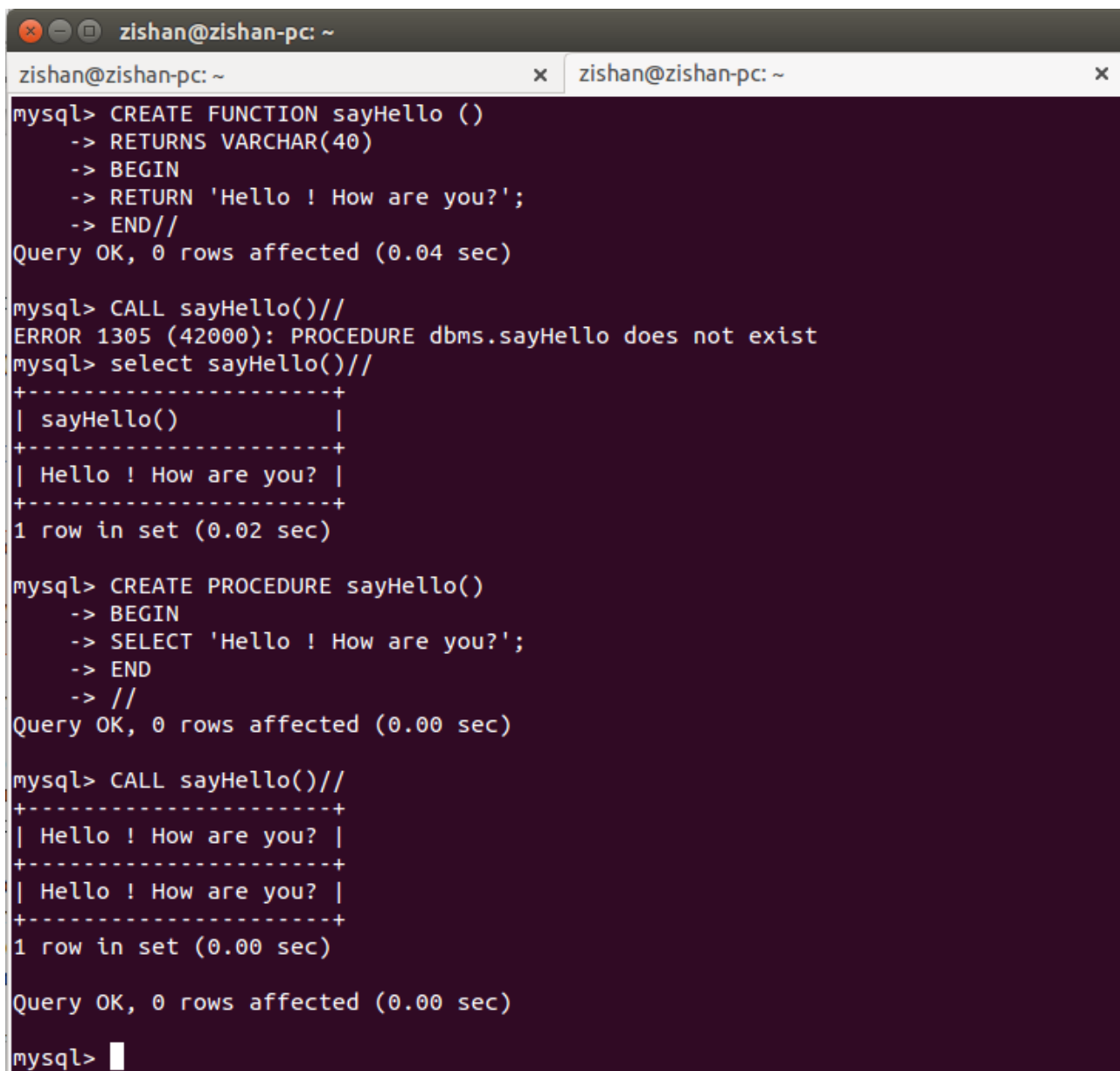
```
mysql> show foreign key on salary74;  
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that  
corresponds to your MySQL server version for the right syntax to use near  
'foreign key on salary74' at line 1  
mysql> show foreign key in salary74;  
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that  
corresponds to your MySQL server version for the right syntax to use near  
'foreign key in salary74' at line 1  
mysql> alter table salary74 drop foreign key salary74_ibfk_1;  
Query OK, 0 rows affected (0.11 sec)  
Records: 0 Duplicates: 0 Warnings: 0  
  
mysql> DELETE FROM department74 WHERE dname='HR';  
ERROR 1451 (23000): Cannot delete or update a parent row: a foreign key cons  
traint fails ('dbms`.`employee74`, CONSTRAINT `employee74_ibfk_1` FOREIGN KE  
Y (`deptno`) REFERENCES `department74` (`deptno`))  
mysql> alter table employee74 drop foreign key employee74_ibfk_1;  
Query OK, 0 rows affected (0.09 sec)  
Records: 0 Duplicates: 0 Warnings: 0  
  
mysql> DELETE FROM department74 WHERE dname='HR';  
Query OK, 1 row affected (0.06 sec)  
  
mysql> █
```

Q2.

i. Write a function and a stored procedure to print Hello ! How are you?.

```
CREATE FUNCTION sayHello ()  
RETURNS VARCHAR(40)  
BEGIN  
    RETURN 'Hello ! How are you?';  
END
```

```
CREATE PROCEDURE sayHello()  
BEGIN  
    SELECT 'Hello ! How are you?';  
END
```



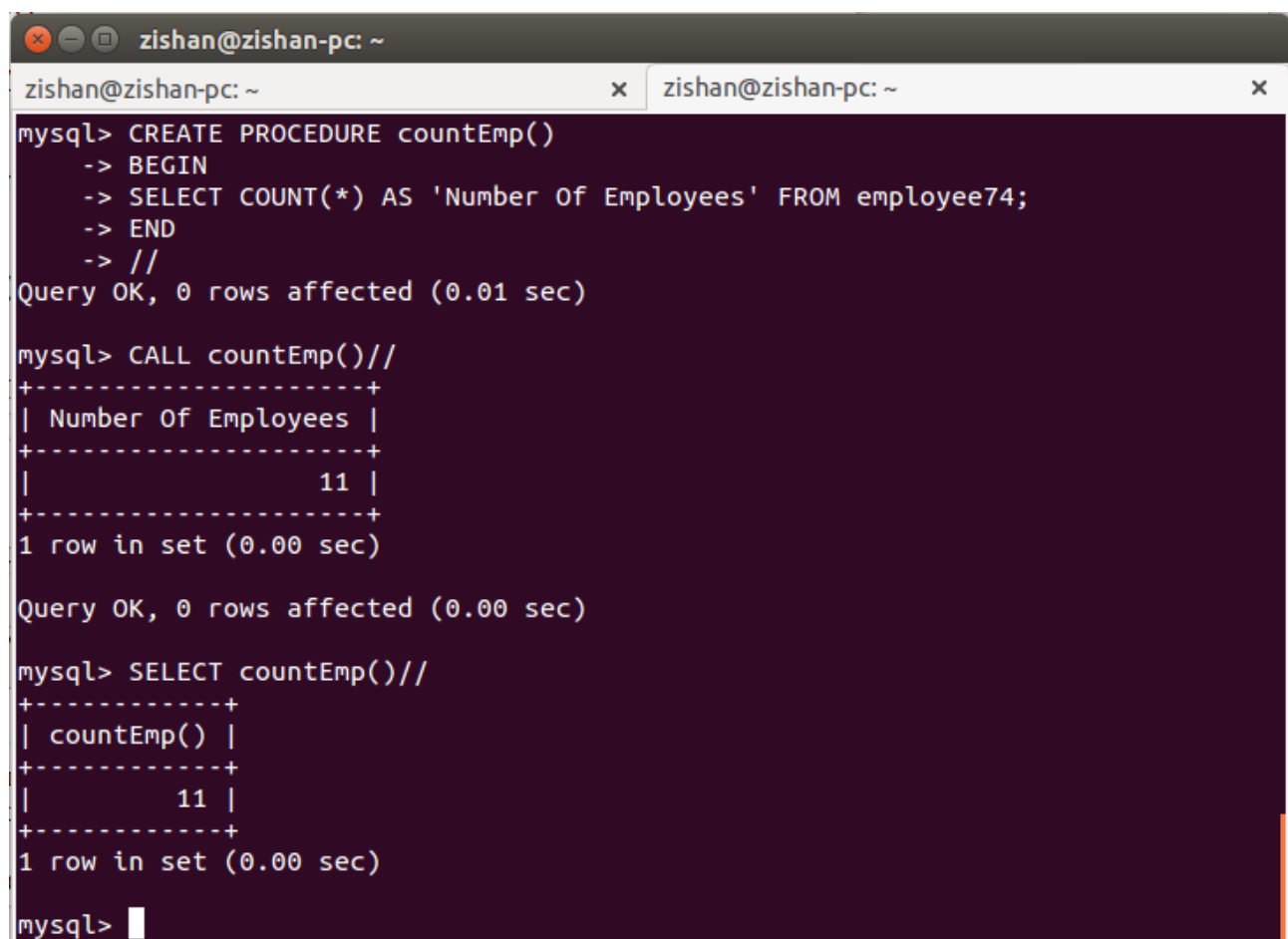
```
zishan@zishan-pc: ~  
mysql> CREATE FUNCTION sayHello ()  
-> RETURNS VARCHAR(40)  
-> BEGIN  
-> RETURN 'Hello ! How are you?';  
-> END//  
Query OK, 0 rows affected (0.04 sec)  
  
mysql> CALL sayHello();//  
ERROR 1305 (42000): PROCEDURE dbms.sayHello does not exist  
mysql> select sayHello();//  
+-----+  
| sayHello() |  
+-----+  
| Hello ! How are you? |  
+-----+  
1 row in set (0.02 sec)  
  
mysql> CREATE PROCEDURE sayHello()  
-> BEGIN  
-> SELECT 'Hello ! How are you?';  
-> END  
-> //  
Query OK, 0 rows affected (0.00 sec)  
  
mysql> CALL sayHello();//  
+-----+  
| Hello ! How are you? |  
+-----+  
| Hello ! How are you? |  
+-----+  
1 row in set (0.00 sec)  
  
Query OK, 0 rows affected (0.00 sec)  
  
mysql> 
```



ii. Write a function and a stored procedure to count the number of employees in the table employee.

```
CREATE FUNCTION countEmp()  
RETURNS INT  
BEGIN  
    DECLARE num INT DEFAULT 0;  
    SELECT COUNT(*) FROM employee74 INTO num;  
    RETURN num;  
END
```

```
CREATE PROCEDURE countEmp()  
BEGIN  
    SELECT COUNT(*) AS 'Number Of Employees' FROM employee74;  
END
```

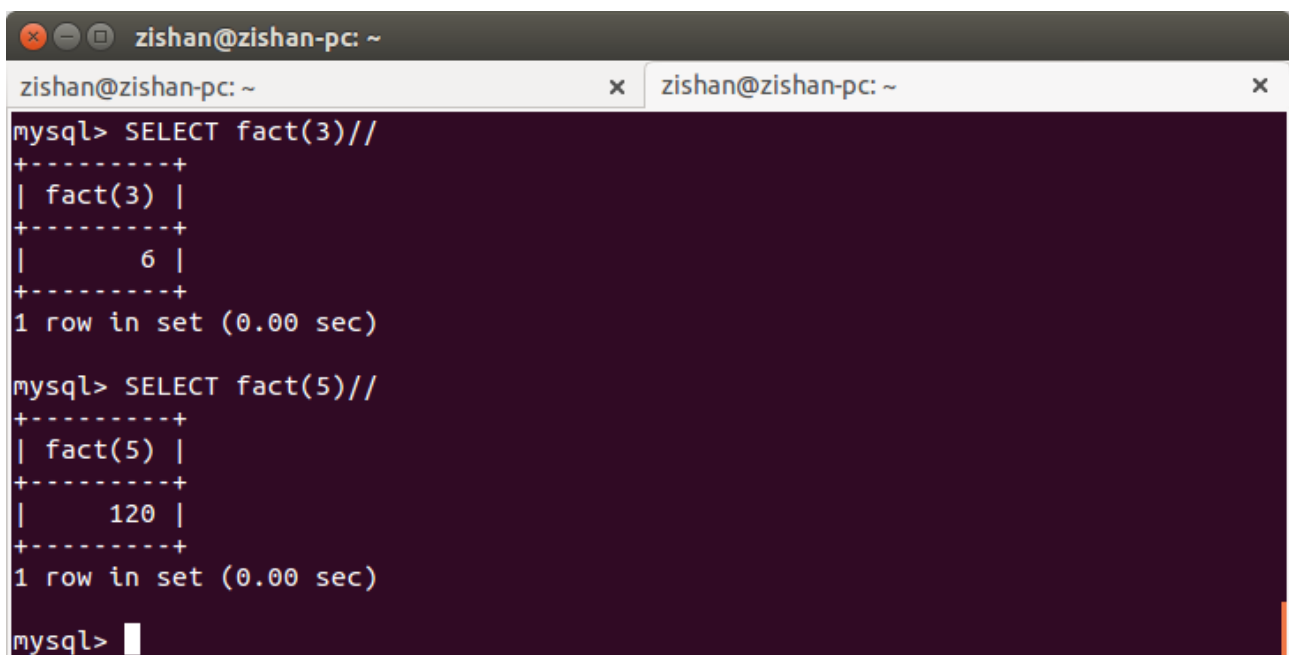


The screenshot shows a terminal window with a dark background and light text. The window title is 'zishan@zishan-pc: ~'. The terminal shows the following commands and output:

```
mysql> CREATE PROCEDURE countEmp()  
-> BEGIN  
-> SELECT COUNT(*) AS 'Number Of Employees' FROM employee74;  
-> END  
-> //  
Query OK, 0 rows affected (0.01 sec)  
  
mysql> CALL countEmp();//  
+-----+  
| Number Of Employees |  
+-----+  
|          11 |  
+-----+  
1 row in set (0.00 sec)  
  
Query OK, 0 rows affected (0.00 sec)  
  
mysql> SELECT countEmp();//  
+-----+  
| countEmp() |  
+-----+  
|          11 |  
+-----+  
1 row in set (0.00 sec)  
  
mysql> 
```

iii. Write a function and a stored procedure to calculate the factorial of the given number.

```
CREATE FUNCTION fact(num INT)
RETURNS INT
BEGIN
    IF num = 0 THEN RETURN (num+1) END IF;
    DECLARE factorial INT DEFAULT 1;
    WHILE num > 1 DO
        SET factorial = factorial * num;
        SET num = num - 1;
    END WHILE;
    RETURN factorial;
END
```



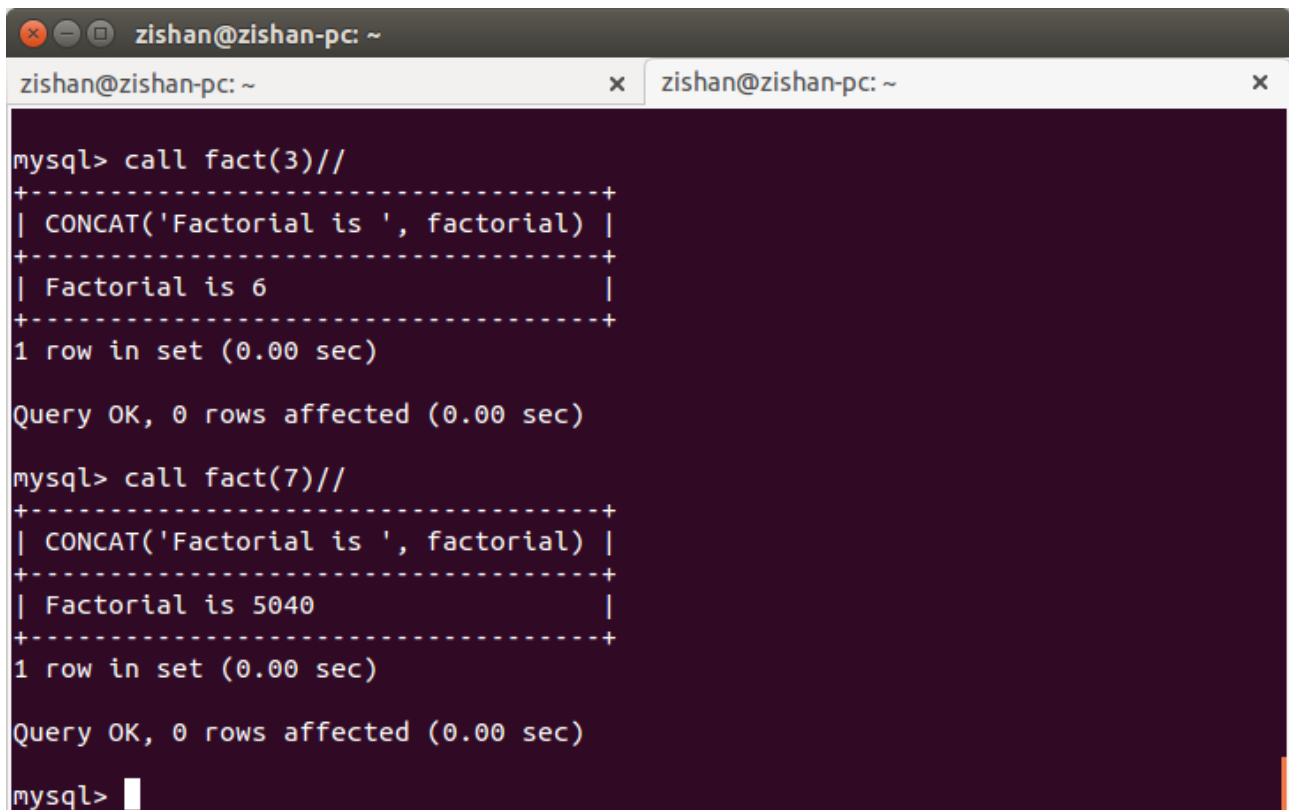
The screenshot shows a terminal window titled 'zishan@zishan-pc: ~'. Inside the terminal, two MySQL queries are executed. The first query is 'mysql> SELECT fact(3);', which returns a single row with the value 6. The second query is 'mysql> SELECT fact(5);', which returns a single row with the value 120. The terminal output is as follows:

```
mysql> SELECT fact(3);
+-----+
| fact(3) |
+-----+
|        6 |
+-----+
1 row in set (0.00 sec)

mysql> SELECT fact(5);
+-----+
| fact(5) |
+-----+
|       120 |
+-----+
1 row in set (0.00 sec)

mysql>
```

```
CREATE PROCEDURE fact (IN num INT)
BEGIN
    DECLARE factorial INT DEFAULT 1;
    IF num = 0 THEN
        SELECT 'Factorial is 1';
    ELSE
        WHILE num > 1 DO
            SET factorial = factorial * num;
            SET num = num - 1;
        END WHILE;
        SELECT CONCAT('Factorial is ', factorial);
    END IF;
END
```



A terminal window titled 'zishan@zishan-pc: ~' with two tabs. The first tab shows the execution of the 'fact' procedure for num=3. The second tab shows the execution for num=7. The output for num=3 is 'Factorial is 6' and for num=7 is 'Factorial is 5040'. The terminal has a dark purple background with white text.

```
mysql> call fact(3)//
+-----+
| CONCAT('Factorial is ', factorial) |
+-----+
| Factorial is 6                      |
+-----+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

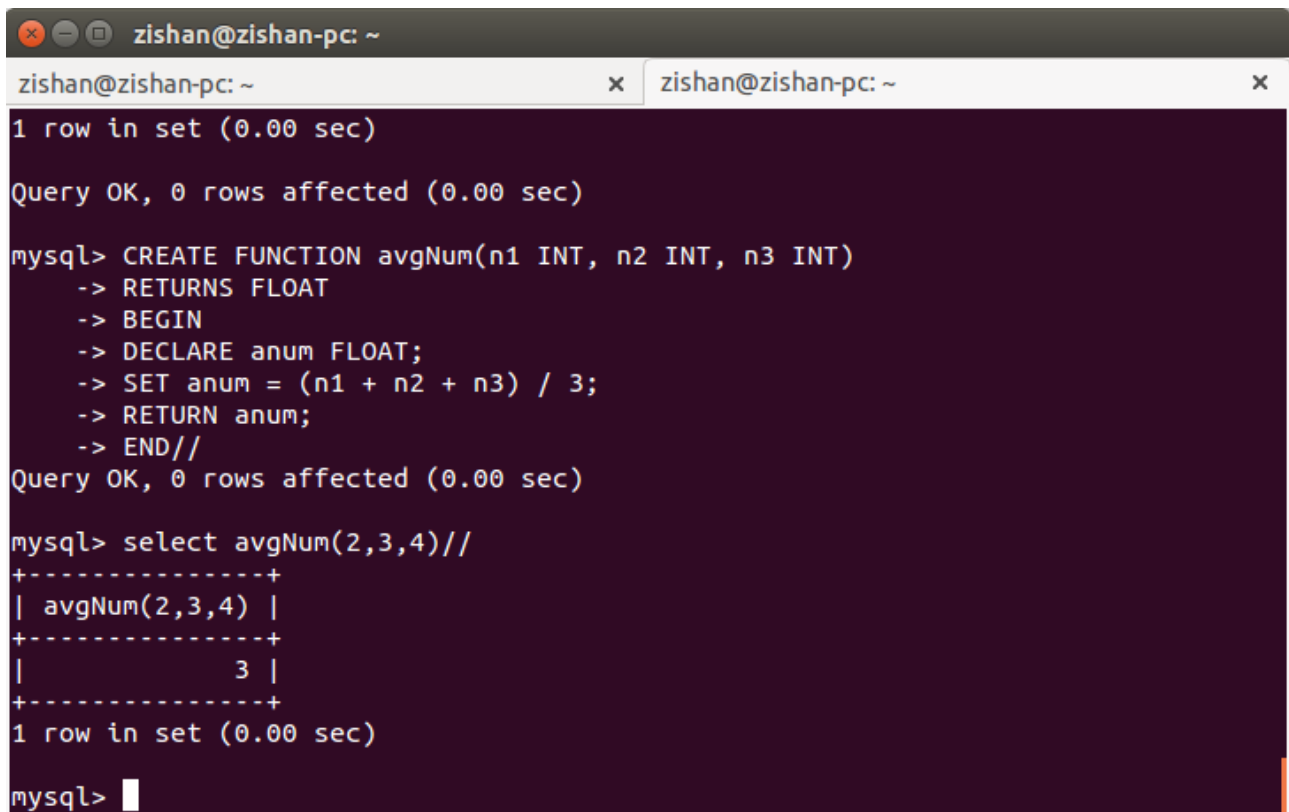
mysql> call fact(7)//
+-----+
| CONCAT('Factorial is ', factorial) |
+-----+
| Factorial is 5040                  |
+-----+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

mysql> 
```

iv. Write a function and a stored procedure to calculate the average of three numbers.

```
CREATE FUNCTION avgNum(n1 INT, n2 INT, n3 INT)
RETURNS FLOAT
BEGIN
    DECLARE anum FLOAT;
    SET anum = (n1 + n2 + n3) / 3;
    RETURN anum;
END
```



The screenshot shows a terminal window with a dark background. At the top, there are window control buttons and the title 'zishan@zishan-pc: ~'. Below the title bar, there are two tabs, both labeled 'zishan@zishan-pc: ~'. The terminal content shows the following sequence of commands and output:

```
1 row in set (0.00 sec)

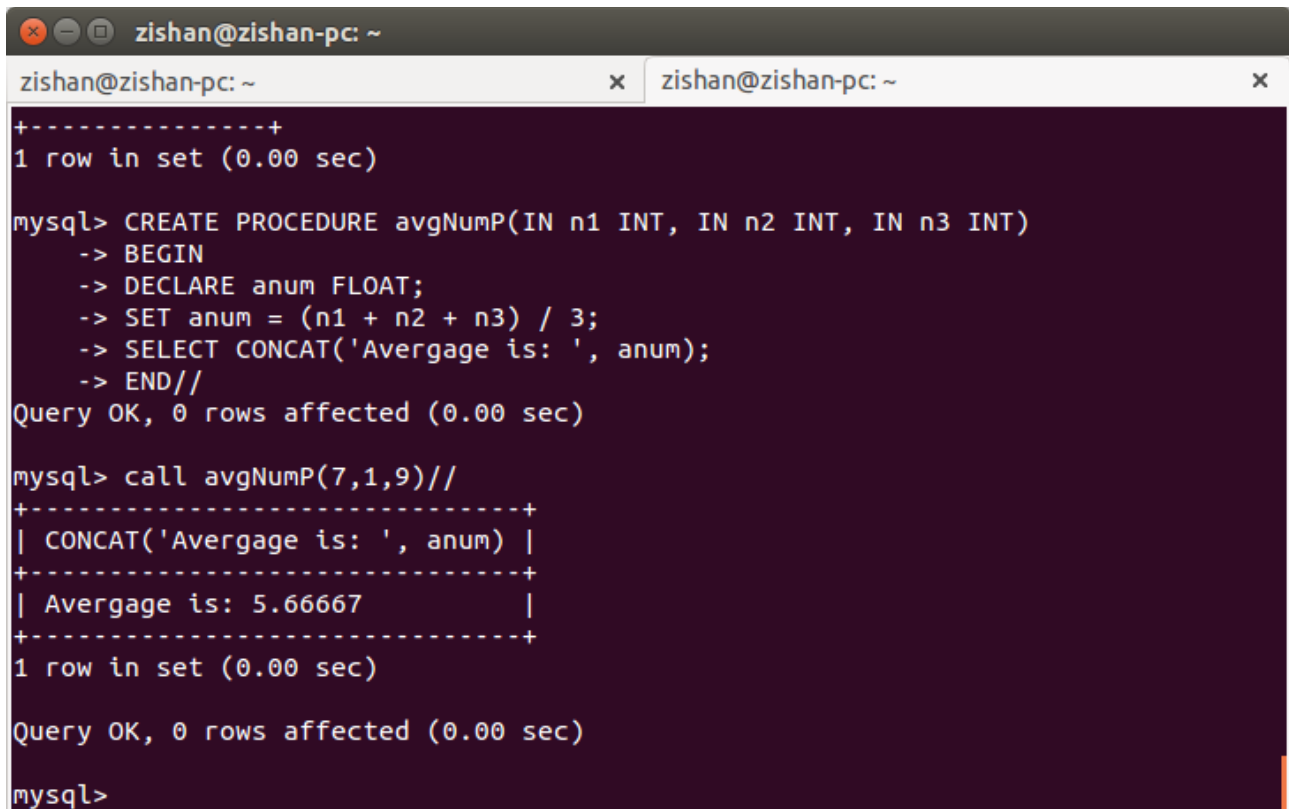
Query OK, 0 rows affected (0.00 sec)

mysql> CREATE FUNCTION avgNum(n1 INT, n2 INT, n3 INT)
      -> RETURNS FLOAT
      -> BEGIN
      -> DECLARE anum FLOAT;
      -> SET anum = (n1 + n2 + n3) / 3;
      -> RETURN anum;
      -> END//
Query OK, 0 rows affected (0.00 sec)

mysql> select avgNum(2,3,4)//
+-----+
| avgNum(2,3,4) |
+-----+
|              3 |
+-----+
1 row in set (0.00 sec)

mysql> 
```

```
CREATE PROCEDURE avgNumP(IN n1 INT, IN n2 INT, IN n3 INT)
BEGIN
    DECLARE anum FLOAT;
    SET anum = (n1 + n2 + n3) / 3;
    SELECT CONCAT('Avergage is: ', anum);
END
```

A terminal window titled 'zishan@zishan-pc: ~' with two tabs. The first tab shows the output of a previous query: '+-----+  
1 row in set (0.00 sec)'. The second tab shows the execution of a MySQL script. The script creates a procedure 'avgNumP' and then calls it with arguments (7, 1, 9). The output of the call is a table with one row: 'Avergage is: 5.66667'.

```
zishan@zishan-pc: ~
mysql> CREATE PROCEDURE avgNumP(IN n1 INT, IN n2 INT, IN n3 INT)
-> BEGIN
-> DECLARE anum FLOAT;
-> SET anum = (n1 + n2 + n3) / 3;
-> SELECT CONCAT('Avergage is: ', anum);
-> END//
Query OK, 0 rows affected (0.00 sec)

mysql> call avgNumP(7,1,9)//
+-----+
| CONCAT('Avergage is: ', anum) |
+-----+
| Avergage is: 5.66667          |
+-----+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

mysql>
```

v. Write a function and stored procedure to find fibonacci series and its sum.

```
CREATE FUNCTION fibo (num INT)
RETURNS VARCHAR(100)
BEGIN
    DECLARE n1 INT DEFAULT 0;
    DECLARE n2 INT DEFAULT 1;
    DECLARE fib INT DEFAULT 0;
    DECLARE sum INT DEFAULT 1;
    DECLARE output VARCHAR(100) DEFAULT '';
    SET output = CONCAT(n1, ' ', n2);
    SET num = num - 2;
    WHILE num > 0 DO
        SET fib = n1 + n2;
        SET sum = sum + fib;
        SET n1 = n2;
        SET n2 = fib;
        SET num = num - 1;
        SET output = CONCAT(output, ' ', n2);
    END WHILE;
    SET output = CONCAT(output, ' ;;Sum: ', sum);
    RETURN output;
END
```



The screenshot shows a MySQL terminal window with the following content:

```
zishan@zishan-pc: ~
mysql> --> END
--> //
Query OK, 0 rows affected (0.00 sec)

mysql> select fibo(4)
--> //
+-----+
| fibo(4) |
+-----+
| 0 1 1 2 ;;Sum: 4 |
+-----+
1 row in set (0.00 sec)

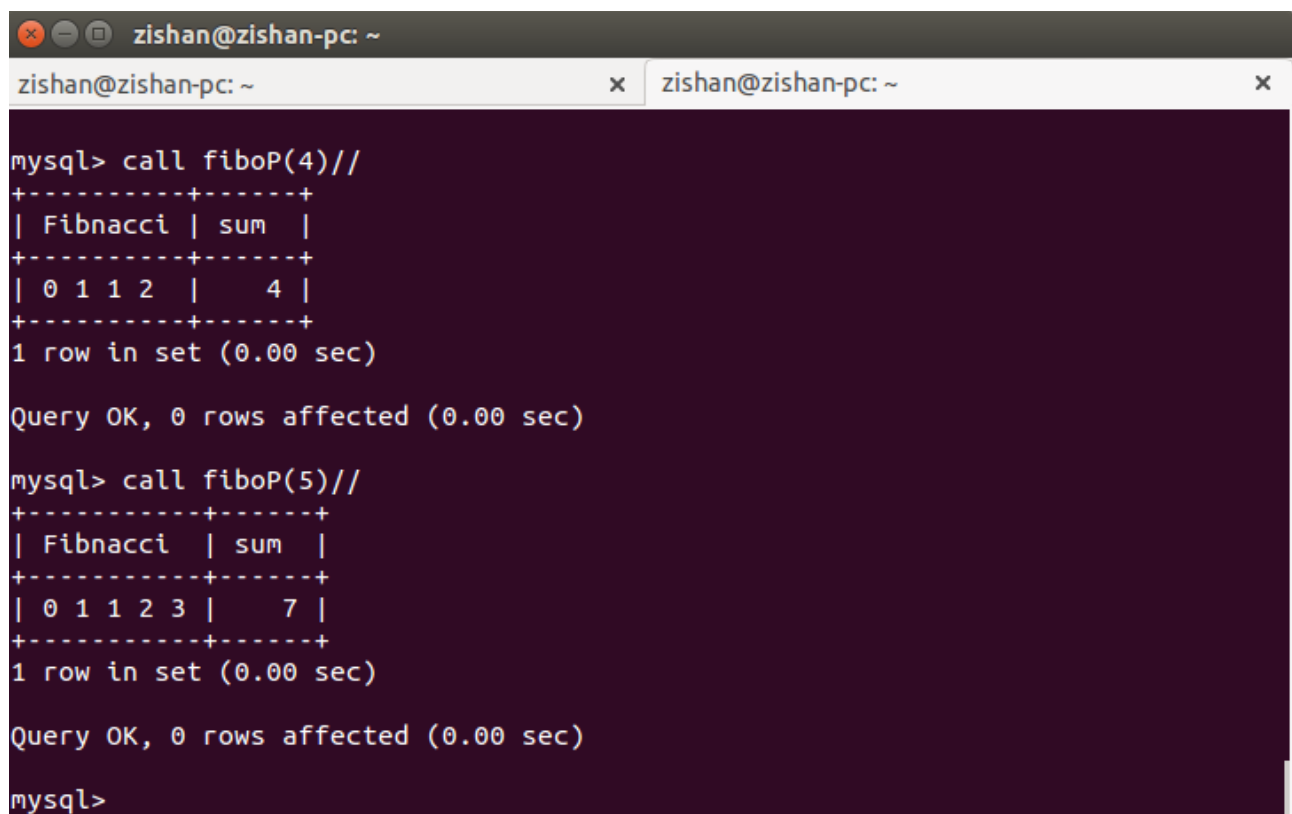
mysql> select fibo(6)//
+-----+
| fibo(6) |
+-----+
| 0 1 1 2 3 5 ;;Sum: 12 |
+-----+
1 row in set (0.01 sec)

mysql>
```

```

CREATE PROCEDURE fiboP (IN num INT)
BEGIN
    DECLARE n1 INT DEFAULT 0;
    DECLARE n2 INT DEFAULT 1;
    DECLARE fib INT DEFAULT 0;
    DECLARE sum INT DEFAULT 1;
    DECLARE output VARCHAR(100) DEFAULT '';
    SET output = CONCAT(n1, ' ', n2);
    SET num = num - 2;
    WHILE num > 0 DO
        SET fib = n1 + n2;
        SET sum = sum + fib;
        SET n1 = n2;
        SET n2 = fib;
        SET num = num - 1;
        SET output = CONCAT(output, ' ', n2);
    END WHILE;
    SELECT output AS Fibonacci, sum;
END

```



The screenshot shows a terminal window with the title 'zishan@zishan-pc: ~'. It displays two MySQL queries and their results. The first query calls 'fiboP(4)' and returns a table with 'Fibonacci' and 'sum' columns. The second query calls 'fiboP(5)' and returns a similar table. Both queries are confirmed as 'Query OK' with 0 rows affected.

```

mysql> call fiboP(4)//
+-----+-----+
| Fibonacci | sum |
+-----+-----+
| 0 1 1 2 | 4 |
+-----+-----+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

mysql> call fiboP(5)//
+-----+-----+
| Fibonacci | sum |
+-----+-----+
| 0 1 1 2 3 | 7 |
+-----+-----+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

mysql>

```

Q3. Consider the following relations

Student (snum: integer, sname: string, major: string, level: string, age: integer),

Class (name: string, meets\_at: time, room: string, fid: integer).

Enrolled (snum: integer, cname: string).

Faculty (fid: integer, fname: string, deptid: integer);

Enrolled has one record per student class pair such that the student is enrolled in the class.

Write the SQL queries. No duplicates should be printed.(use foreign key)

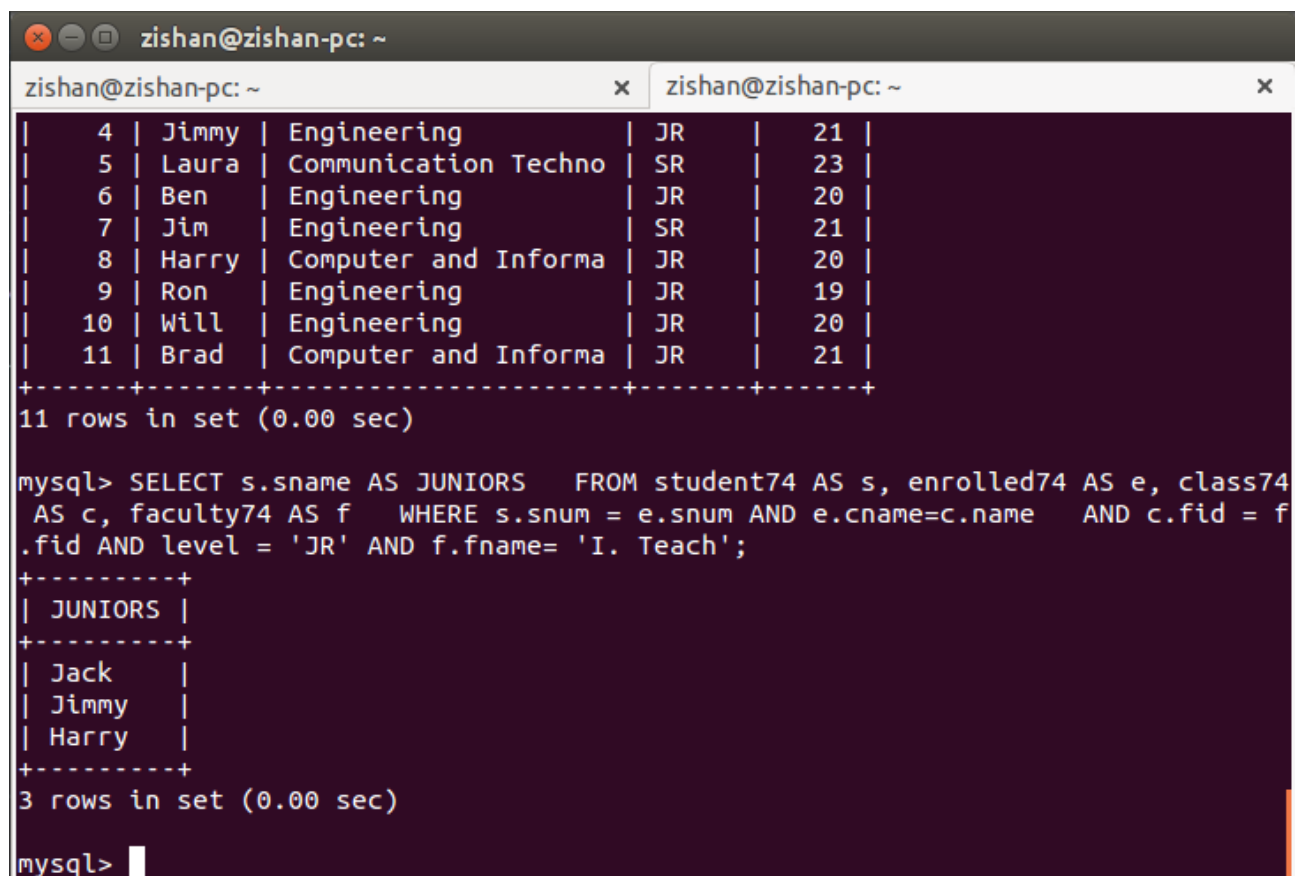
1. Find the names of all Juniors (level = JR) who are enrolled in a class taught by I. Teach.

```
SELECT s.sname AS JUNIORS
```

```
FROM student74 AS s, enrolled74 AS e, class74 AS c, faculty74 AS f
```

```
WHERE s.snum = e.snum AND e.cname=c.name
```

```
AND c.fid = f.fid AND level = 'JR' AND f.fname= 'I. Teach';
```



```
zishan@zishan-pc: ~
zishan@zishan-pc: ~ x zishan@zishan-pc: ~ x
| 4 | Jimmy | Engineering | JR | 21 |
| 5 | Laura | Communication Techno | SR | 23 |
| 6 | Ben | Engineering | JR | 20 |
| 7 | Jim | Engineering | SR | 21 |
| 8 | Harry | Computer and Informa | JR | 20 |
| 9 | Ron | Engineering | JR | 19 |
| 10 | Will | Engineering | JR | 20 |
| 11 | Brad | Computer and Informa | JR | 21 |
+-----+
11 rows in set (0.00 sec)

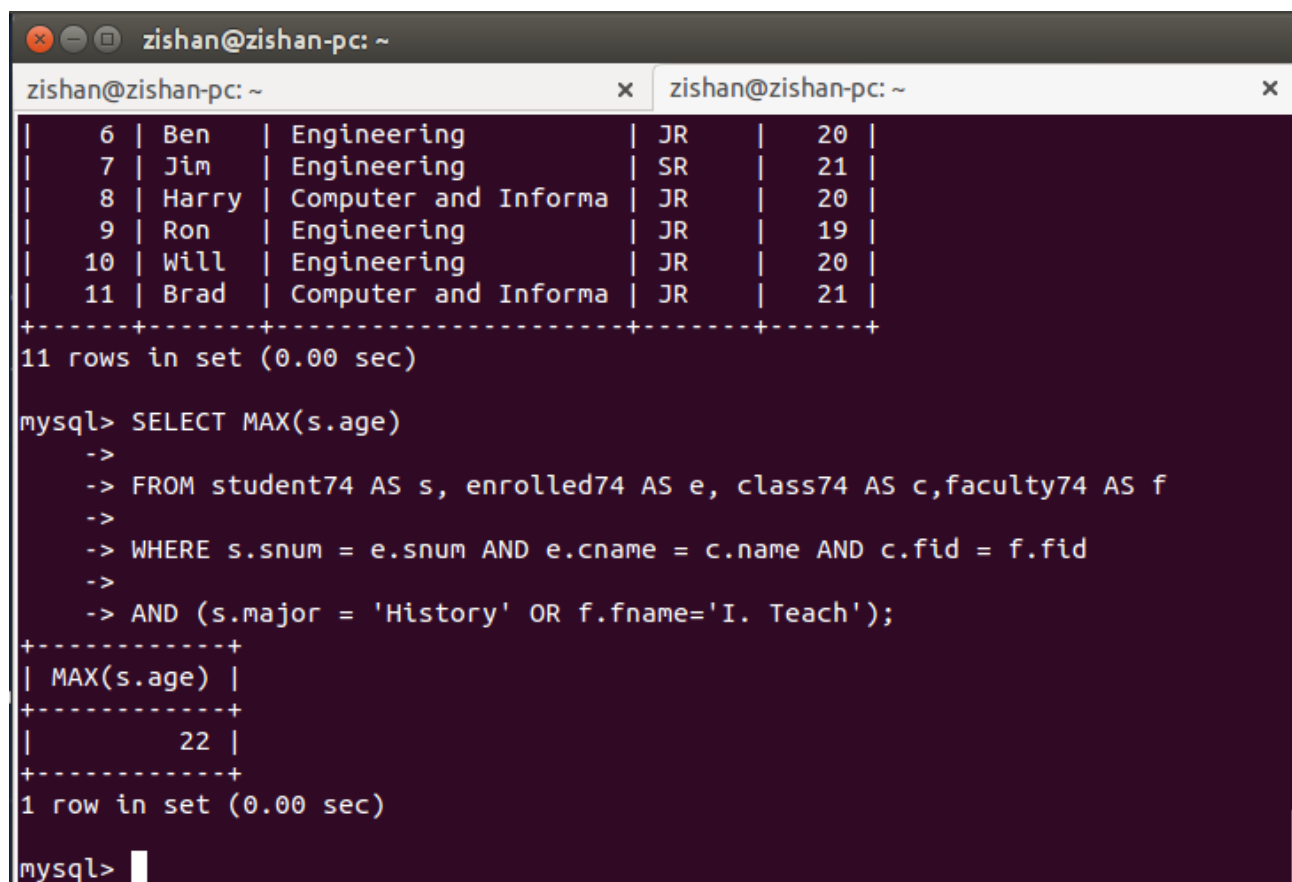
mysql> SELECT s.sname AS JUNIORS FROM student74 AS s, enrolled74 AS e, class74
AS c, faculty74 AS f WHERE s.snum = e.snum AND e.cname=c.name AND c.fid = f
.fid AND level = 'JR' AND f.fname= 'I. Teach';
+-----+
| JUNIORS |
+-----+
| Jack |
| Jimmy |
| Harry |
+-----+
3 rows in set (0.00 sec)

mysql> 
```



2. Find the age of the oldest student who is either a History major or enrolled in a course taught by I. Teach.

```
SELECT MAX(s.age)
  FROM student74 AS s, enrolled74 AS e, class74 AS c, faculty74 AS f
 WHERE s.snum = e.snum AND e.cname = c.name AND c.fid = f.fid
AND (s.major = 'History' OR f.fname='I. Teach');
```

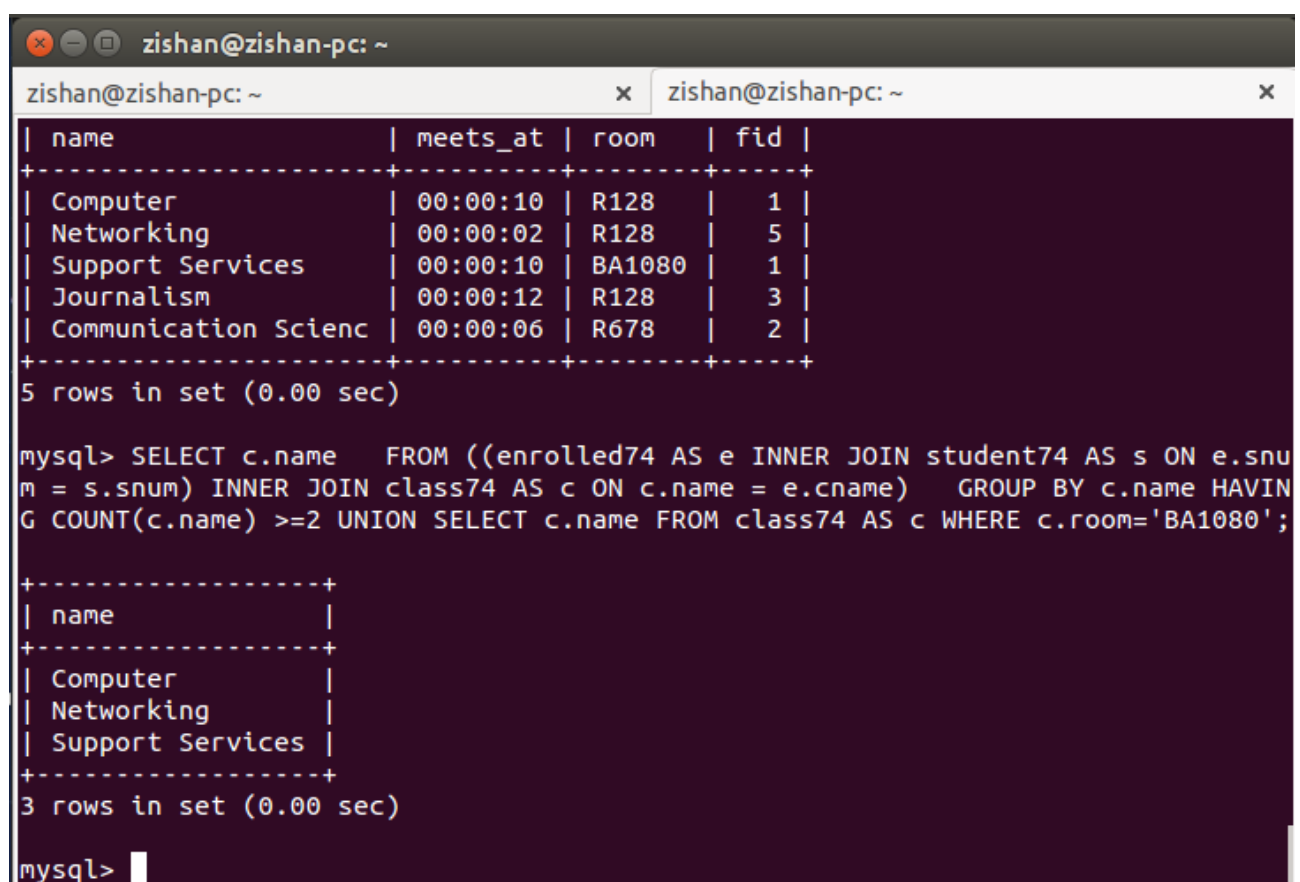


The screenshot shows a terminal window with a MySQL prompt. The first query displays a table of 11 rows with columns for student ID, name, major, advisor ID, and age. The second query finds the maximum age of students who are either History majors or enrolled in a course taught by I. Teach, returning a single row with the value 22.

```
zishan@zishan-pc: ~  
zishan@zishan-pc: ~ x zishan@zishan-pc: ~ x  
| 6 | Ben | Engineering | JR | 20 |  
| 7 | Jim | Engineering | SR | 21 |  
| 8 | Harry | Computer and Informa | JR | 20 |  
| 9 | Ron | Engineering | JR | 19 |  
| 10 | Will | Engineering | JR | 20 |  
| 11 | Brad | Computer and Informa | JR | 21 |  
+-----+  
11 rows in set (0.00 sec)  
  
mysql> SELECT MAX(s.age)  
->  
-> FROM student74 AS s, enrolled74 AS e, class74 AS c, faculty74 AS f  
->  
-> WHERE s.snum = e.snum AND e.cname = c.name AND c.fid = f.fid  
->  
-> AND (s.major = 'History' OR f.fname='I. Teach');  
+-----+  
| MAX(s.age) |  
+-----+  
| 22 |  
+-----+  
1 row in set (0.00 sec)  
  
mysql> 
```

3. Find the names of all classes that either meet in room BA1080 or have 2 or more students enrolled.

```
SELECT c.name
FROM ((enrolled74 AS e INNER JOIN student74 AS s ON e.snum = s.snum)
INNER JOIN class74 AS c ON c.name = e.cname)
GROUP BY c.name HAVING COUNT(c.name) >=2 UNION SELECT c.name
FROM class74 AS c WHERE c.room='BA1080';
```



The terminal window shows the execution of a SQL query. The first part displays the results of a query that joins the enrolled74, student74, and class74 tables, grouped by class name and filtered to show classes with 2 or more students. The results are as follows:

name	meets_at	room	fid
Computer	00:00:10	R128	1
Networking	00:00:02	R128	5
Support Services	00:00:10	BA1080	1
Journalism	00:00:12	R128	3
Communication Scienc	00:00:06	R678	2

5 rows in set (0.00 sec)

The second part shows the execution of the query specified in the problem statement, which filters for classes in room BA1080 or those with 2 or more students. The results are:

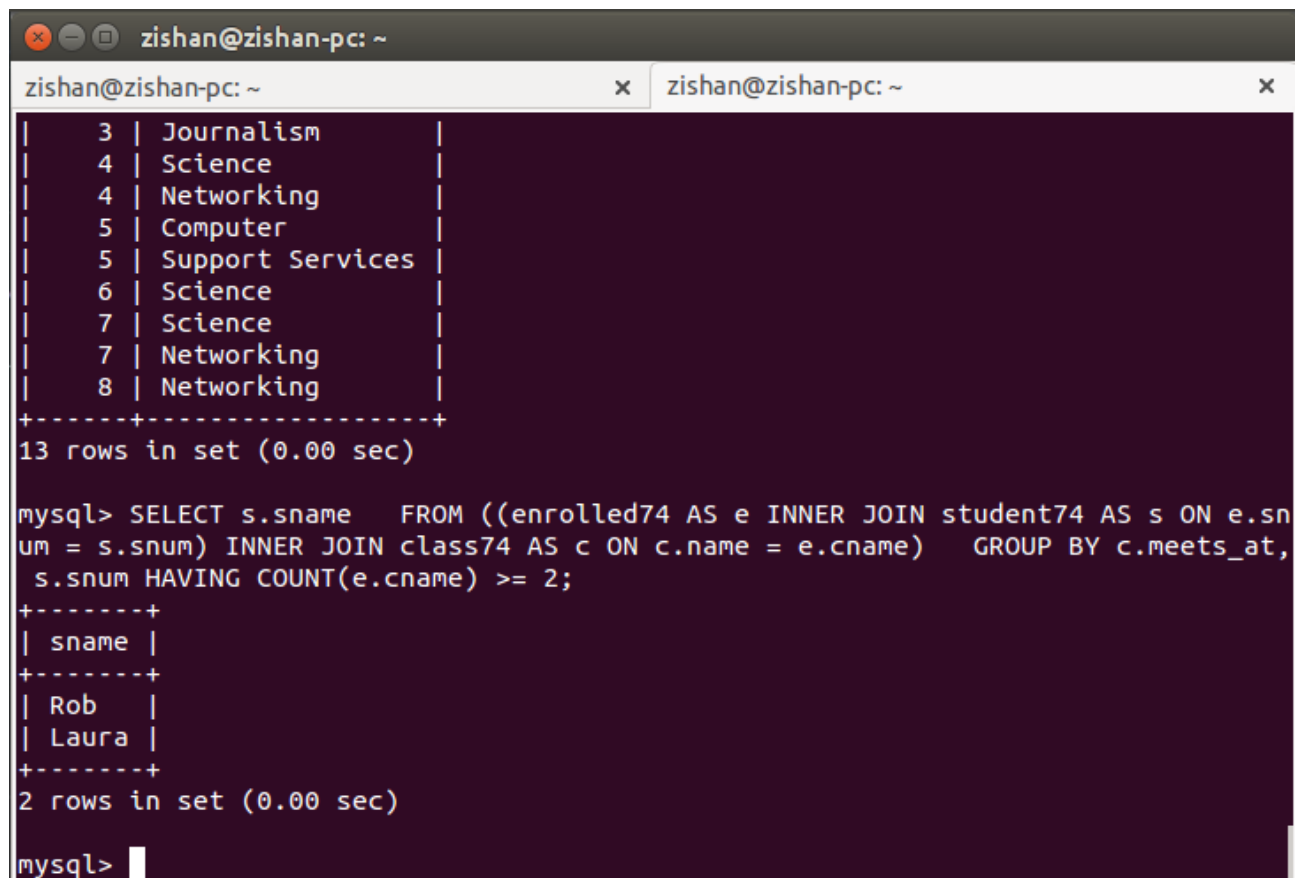
name
Computer
Networking
Support Services

3 rows in set (0.00 sec)

The terminal ends with the mysql prompt.

4. Find the names of all students who are enrolled in two classes that meet at the same time.

```
SELECT s.sname FROM (  
    (enrolled74 AS e INNER JOIN student74 AS s ON e.snum = s.snum)  
    INNER JOIN class74 AS c ON c.name = e.cname  
) GROUP BY c.meets_at, s.snum HAVING COUNT(e.cname) >= 2;
```

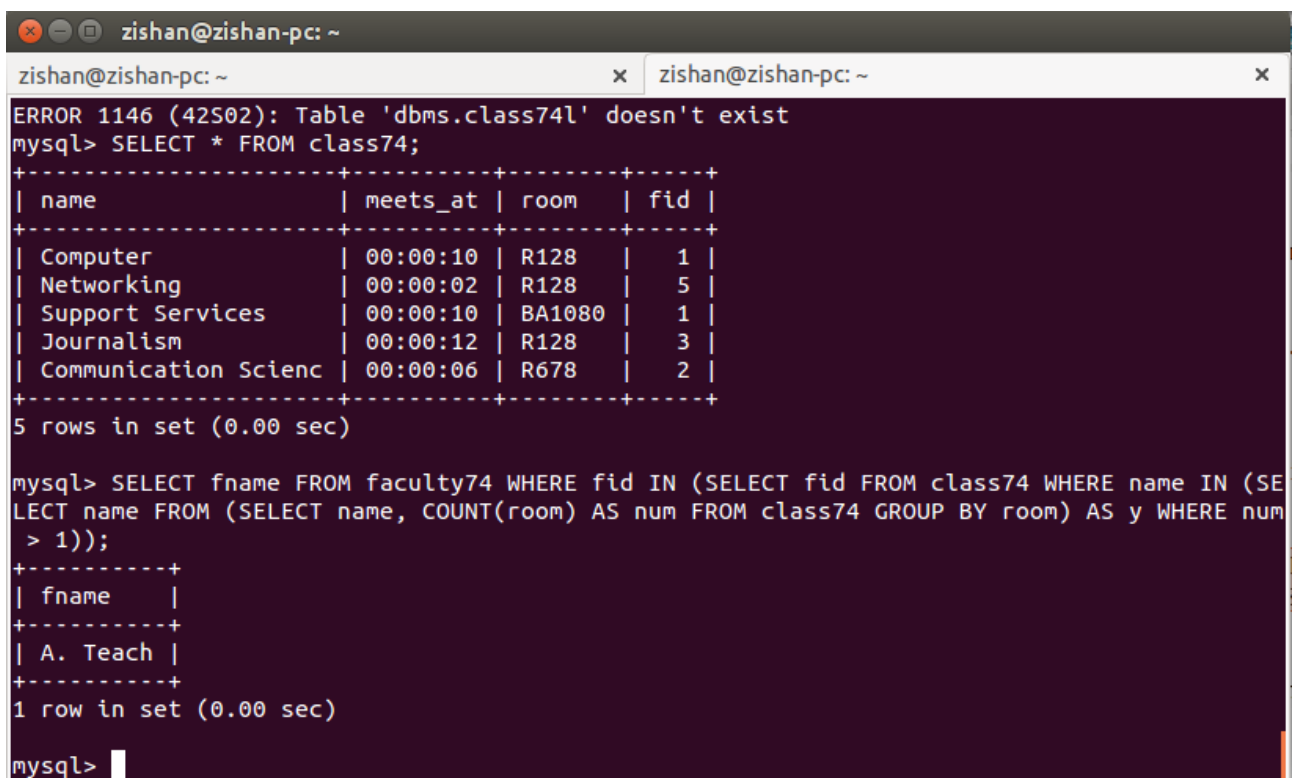


The terminal window shows the execution of a SQL query. The first query returns 13 rows of student and class information. The second query returns 2 rows of student names who are enrolled in two classes meeting at the same time.

```
zishan@zishan-pc: ~  
zishan@zishan-pc: ~ x zishan@zishan-pc: ~ x  
| 3 | Journalism |  
| 4 | Science |  
| 4 | Networking |  
| 5 | Computer |  
| 5 | Support Services |  
| 6 | Science |  
| 7 | Science |  
| 7 | Networking |  
| 8 | Networking |  
+-----+  
13 rows in set (0.00 sec)  
  
mysql> SELECT s.sname FROM ((enrolled74 AS e INNER JOIN student74 AS s ON e.snum = s.snum) INNER JOIN class74 AS c ON c.name = e.cname) GROUP BY c.meets_at, s.snum HAVING COUNT(e.cname) >= 2;  
+-----+  
| sname |  
+-----+  
| Rob |  
| Laura |  
+-----+  
2 rows in set (0.00 sec)  
  
mysql> 
```

5. Find the names of faculty members who teach in every room in which some class is taught.

```
SELECT fname FROM faculty74 WHERE fid IN (  
    SELECT fid FROM class74 WHERE name IN (  
        SELECT name FROM (  
            SELECT name, COUNT(room) AS num FROM class74  
            GROUP BY room  
        ) AS y WHERE num > 1  
    )  
);
```

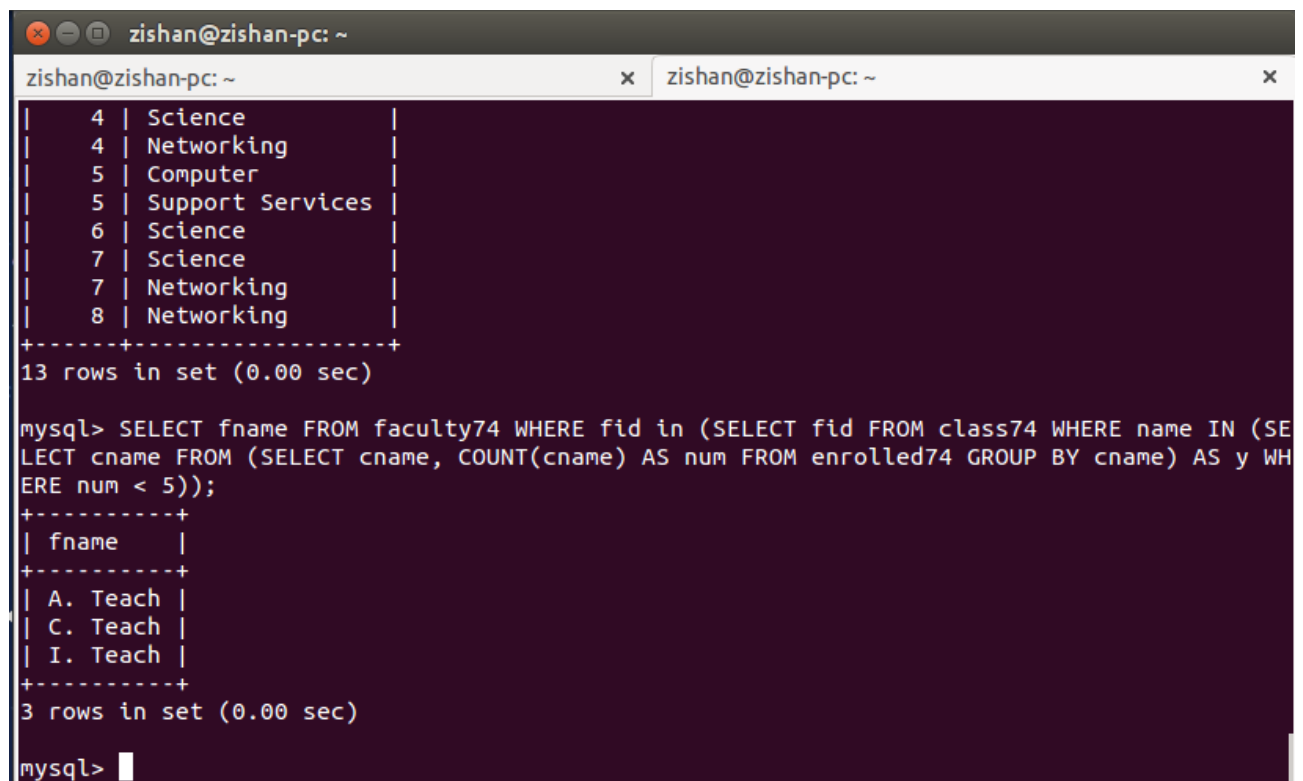


The screenshot shows a terminal window with a MySQL prompt. The first query attempts to select all data from a table named 'class74', but it results in an error: 'ERROR 1146 (42S02): Table 'dbms.class74l' doesn't exist'. The second query successfully retrieves data from 'class74', displaying a table with 5 rows. The third query successfully finds the name of the faculty member 'A. Teach' who teaches in every room where a class is held, returning 1 row.

```
zishan@zishan-pc: ~  
mysql> SELECT * FROM class74;  
+-----+-----+-----+-----+  
| name          | meets_at | room  | fid |  
+-----+-----+-----+-----+  
| Computer      | 00:00:10 | R128  | 1   |  
| Networking    | 00:00:02 | R128  | 5   |  
| Support Services | 00:00:10 | BA1080 | 1   |  
| Journalism     | 00:00:12 | R128  | 3   |  
| Communication Scienc | 00:00:06 | R678  | 2   |  
+-----+-----+-----+-----+  
5 rows in set (0.00 sec)  
  
mysql> SELECT fname FROM faculty74 WHERE fid IN (SELECT fid FROM class74 WHERE name IN (SE  
LECT name FROM (SELECT name, COUNT(room) AS num FROM class74 GROUP BY room) AS y WHERE num  
> 1));  
+-----+  
| fname |  
+-----+  
| A. Teach |  
+-----+  
1 row in set (0.00 sec)  
  
mysql>
```

6. Find the names of faculty members for whom the combined enrollment of the courses that they teach is less than five.

```
SELECT fname FROM faculty74 WHERE fid in (  
    SELECT fid FROM class74 WHERE name IN (  
        SELECT cname FROM (  
            SELECT cname, COUNT(cname) AS num FROM enrolled74  
            GROUP BY cname  
        ) AS y WHERE num < 5  
    )  
);
```



The screenshot shows a terminal window with a dark background. At the top, there are window title bars for 'zishan@zishan-pc: ~'. The terminal displays a table of course data with 13 rows. Below this, it shows the execution of a MySQL query to find faculty members with fewer than five combined enrollments. The query result shows three faculty members: A. Teach, C. Teach, and I. Teach.

```
zishan@zishan-pc: ~  
zishan@zishan-pc: ~  
+-----+  
| 4 | Science |  
+-----+  
| 4 | Networking |  
+-----+  
| 5 | Computer |  
+-----+  
| 5 | Support Services |  
+-----+  
| 6 | Science |  
+-----+  
| 7 | Science |  
+-----+  
| 7 | Networking |  
+-----+  
| 8 | Networking |  
+-----+  
13 rows in set (0.00 sec)  
  
mysql> SELECT fname FROM faculty74 WHERE fid in (SELECT fid FROM class74 WHERE name IN (SE  
LECT cname FROM (SELECT cname, COUNT(cname) AS num FROM enrolled74 GROUP BY cname) AS y WH  
ERE num < 5));  
+-----+  
| fname |  
+-----+  
| A. Teach |  
+-----+  
| C. Teach |  
+-----+  
| I. Teach |  
+-----+  
3 rows in set (0.00 sec)  
  
mysql>
```

7. For each level, print the level and the average age of student s for that level.

```
SELECT s.level , avg(age) as 'Average Age'
      FROM student74 AS s
GROUP BY s.level;
```



The image shows a terminal window with two tabs, both titled 'zishan@zishan-pc: ~'. The first tab contains a MySQL query and its result:

```
mysql> SELECT fname FROM faculty74 WHERE fid IN (SELECT fid FROM class74 WHERE name IN (SE
LECT name FROM (SELECT name, COUNT(room) AS num FROM class74 GROUP BY room) AS y WHERE num
> 1));
```

fname
A. Teach

1 row in set (0.00 sec)

The second tab contains a MySQL query and its result:

```
mysql> SELECT s.level , avg(age) as 'Average Age'
->
-> FROM student74 AS s
->
-> GROUP BY s.level;
```

level	Average Age
JR	20.2857
SR	22.0000

2 rows in set (0.00 sec)

The prompt 'mysql>' is visible at the bottom of the second tab.

8. For all levels except JR, print the level and the average age of students for that level.

```
SELECT s.level, avg(s.age) as Average FROM student74 AS s
      WHERE s.level != 'JR'
GROUP BY s.level;
```

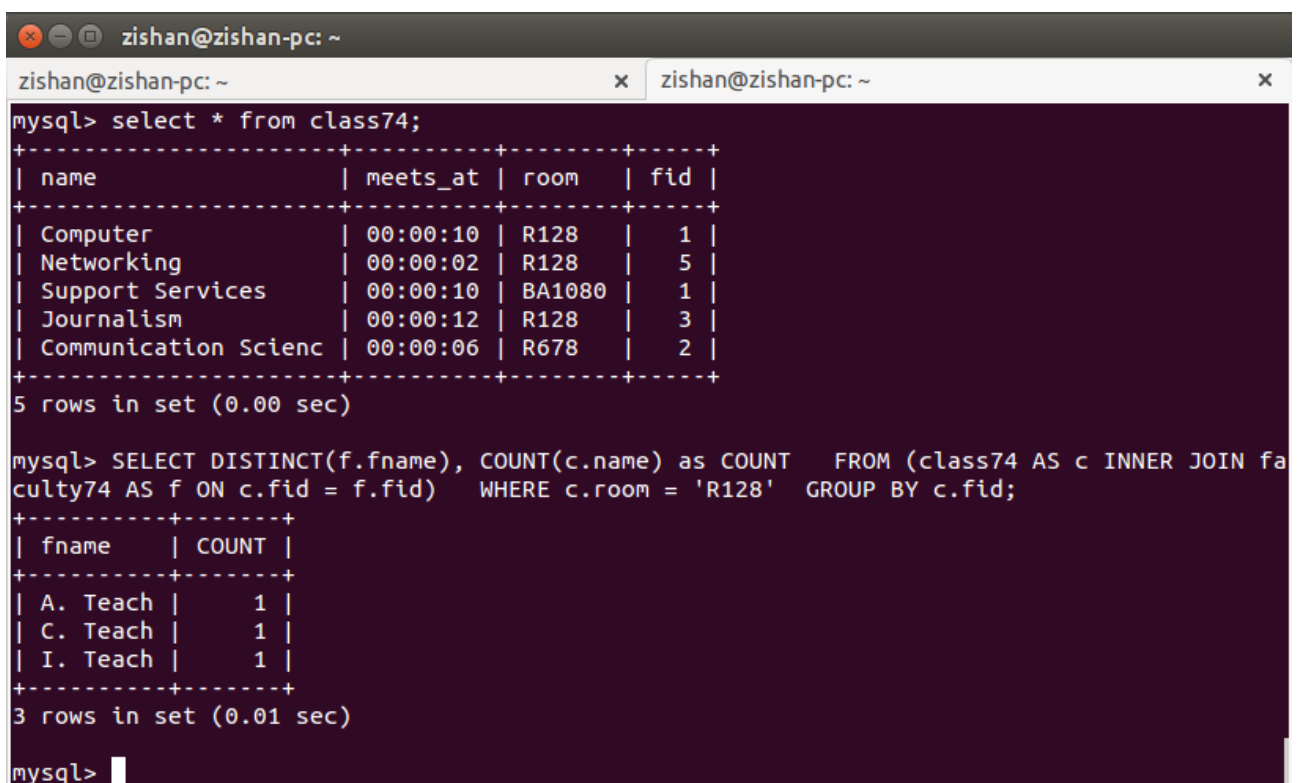
```
zishan@zishan-pc: ~
zishan@zishan-pc: ~ x zishan@zishan-pc: ~ x
| 1 | Rob | History | SR | 22 |
| 2 | Jack | Aviation | JR | 21 |
| 3 | Peter | Mathematics and Stat | SR | 22 |
| 4 | Jimmy | Engineering | JR | 21 |
| 5 | Laura | Communication Techno | SR | 23 |
| 6 | Ben | Engineering | JR | 20 |
| 7 | Jim | Engineering | SR | 21 |
| 8 | Harry | Computer and Informa | JR | 20 |
| 9 | Ron | Engineering | JR | 19 |
| 10 | Will | Engineering | JR | 20 |
| 11 | Brad | Computer and Informa | JR | 21 |
+-----+-----+
11 rows in set (0.00 sec)

mysql> SELECT s.level, avg(s.age) as Average FROM student74 AS s WHERE s.level != 'JR'
      GROUP BY s.level;
+-----+-----+
| level | Average |
+-----+-----+
| SR    | 22.0000 |
+-----+-----+
1 row in set (0.00 sec)

mysql> 
```

9. For each faculty member that has taught classes only in room R128, print the faculty members name and the total number of classes she or he has taught.

```
SELECT DISTINCT(f.fname), COUNT(c.name) as COUNT
  FROM (class74 AS c INNER JOIN faculty74 AS f ON c.fid = f.fid)
 WHERE c.room = 'R128'
GROUP BY c.fid;
```



The screenshot shows a terminal window with a dark background. The title bar indicates the user is 'zishan@zishan-pc'. The terminal displays two MySQL queries and their results. The first query selects all columns from the 'class74' table, showing 5 rows. The second query uses a JOIN to find faculty members who have taught in room R128, showing 3 rows.

```
mysql> select * from class74;
+-----+-----+-----+-----+
| name          | meets_at | room  | fid |
+-----+-----+-----+-----+
| Computer      | 00:00:10 | R128  | 1   |
| Networking    | 00:00:02 | R128  | 5   |
| Support Services | 00:00:10 | BA1080 | 1   |
| Journalism     | 00:00:12 | R128  | 3   |
| Communication Scienc | 00:00:06 | R678  | 2   |
+-----+-----+-----+-----+
5 rows in set (0.00 sec)

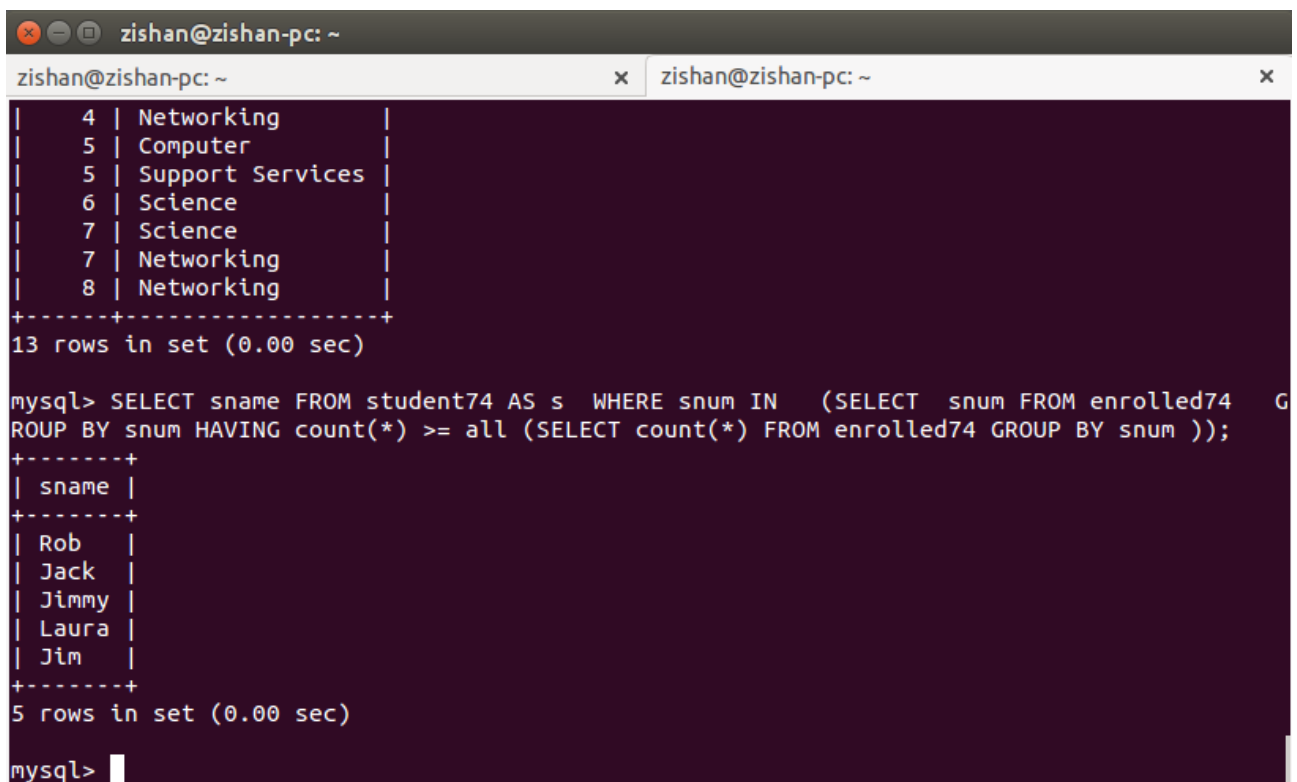
mysql> SELECT DISTINCT(f.fname), COUNT(c.name) as COUNT FROM (class74 AS c INNER JOIN fa
culty74 AS f ON c.fid = f.fid) WHERE c.room = 'R128' GROUP BY c.fid;
+-----+-----+
| fname    | COUNT |
+-----+-----+
| A. Teach | 1     |
| C. Teach | 1     |
| I. Teach | 1     |
+-----+-----+
3 rows in set (0.01 sec)

mysql>
```



10. Find the names of students enrolled in the maximum number of classes.

```
SELECT sname FROM student74 AS s WHERE snum IN (
    SELECT snum FROM enrolled74 GROUP BY snum
    HAVING count(*) >= all (
        SELECT count(*) FROM enrolled74 GROUP BY snum
    )
);
```



The screenshot shows a terminal window with a MySQL prompt. The first query displays a list of students and their enrolled classes. The second query finds students enrolled in the maximum number of classes.

```
zishan@zishan-pc: ~
zishan@zishan-pc: ~ x zishan@zishan-pc: ~ x
| 4 | Networking |
| 5 | Computer  |
| 5 | Support Services |
| 6 | Science   |
| 7 | Science   |
| 7 | Networking |
| 8 | Networking |
+-----+
13 rows in set (0.00 sec)

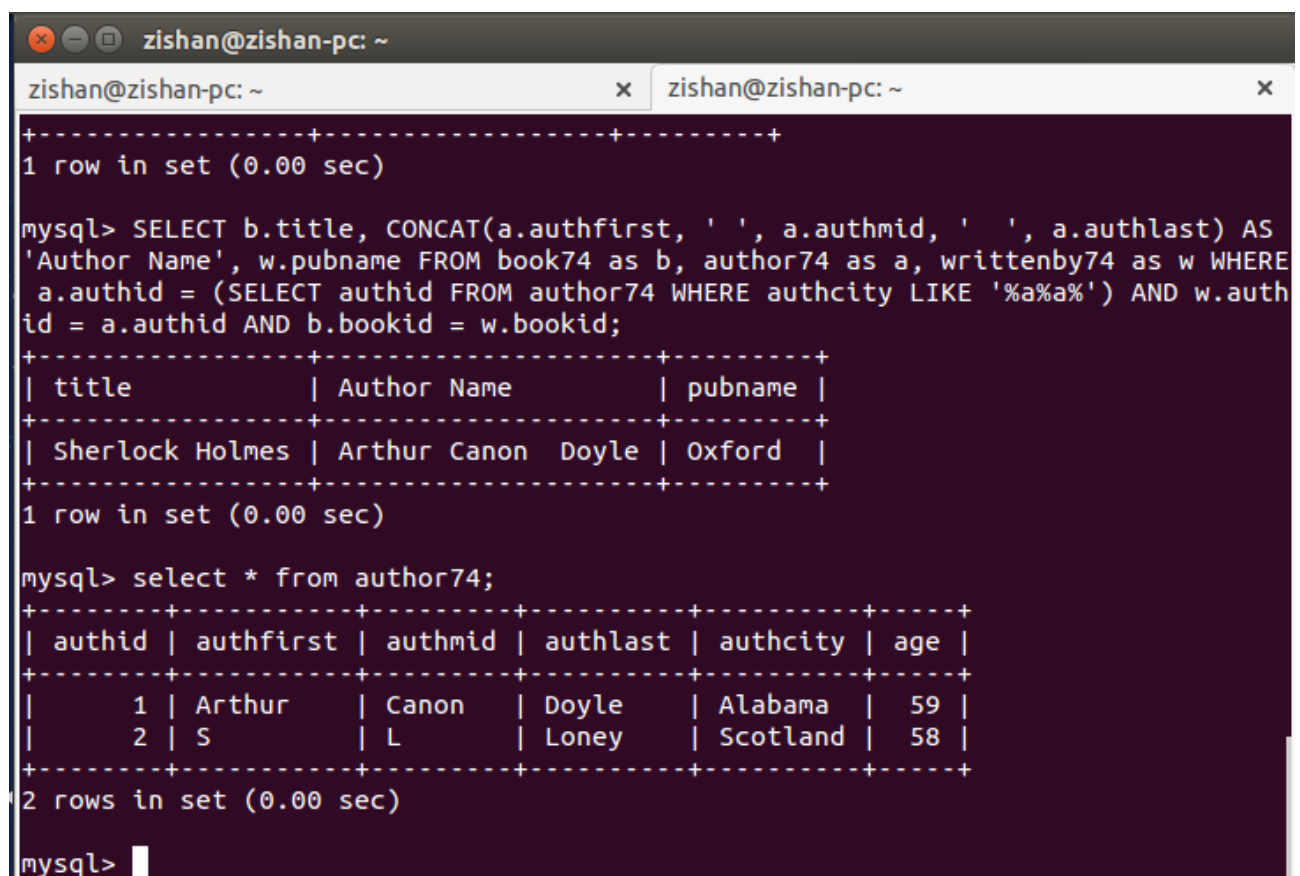
mysql> SELECT sname FROM student74 AS s WHERE snum IN (SELECT snum FROM enrolled74 GROUP BY snum HAVING count(*) >= all (SELECT count(*) FROM enrolled74 GROUP BY snum ));
+-----+
| sname |
+-----+
| Rob   |
| Jack  |
| Jimmy |
| Laura |
| Jim   |
+-----+
5 rows in set (0.00 sec)

mysql>
```

Q4. Write equivalent SQL for the following query.

1. Get the title,author name,publisher name for author whose city contain total no of a=2?

```
SELECT b.title, CONCAT(a.authfirst, ' ', a.authmid, ' ', a.authlast)
AS 'Author Name', w.pubname FROM book74 as b, author74 as a, writtenby74 as w
WHERE a.authid = (
    SELECT authid FROM author74 WHERE authcity LIKE '%a%a%'
)
AND w.authid = a.authid AND b.bookid = w.bookid;
```



The screenshot shows a terminal window with a MySQL prompt. The user has executed a query to select book titles, author names (concatenated from first, middle, and last names), and publisher names for books written by authors whose city contains the letter 'a' twice. The result shows one row: 'Sherlock Holmes' by 'Arthur Canon Doyle' published by 'Oxford'.

```
mysql> SELECT b.title, CONCAT(a.authfirst, ' ', a.authmid, ' ', a.authlast) AS
'Author Name', w.pubname FROM book74 as b, author74 as a, writtenby74 as w WHERE
a.authid = (SELECT authid FROM author74 WHERE authcity LIKE '%a%a%') AND w.auth
id = a.authid AND b.bookid = w.bookid;
```

title	Author Name	pubname
Sherlock Holmes	Arthur Canon Doyle	Oxford

1 row in set (0.00 sec)

```
mysql> select * from author74;
```

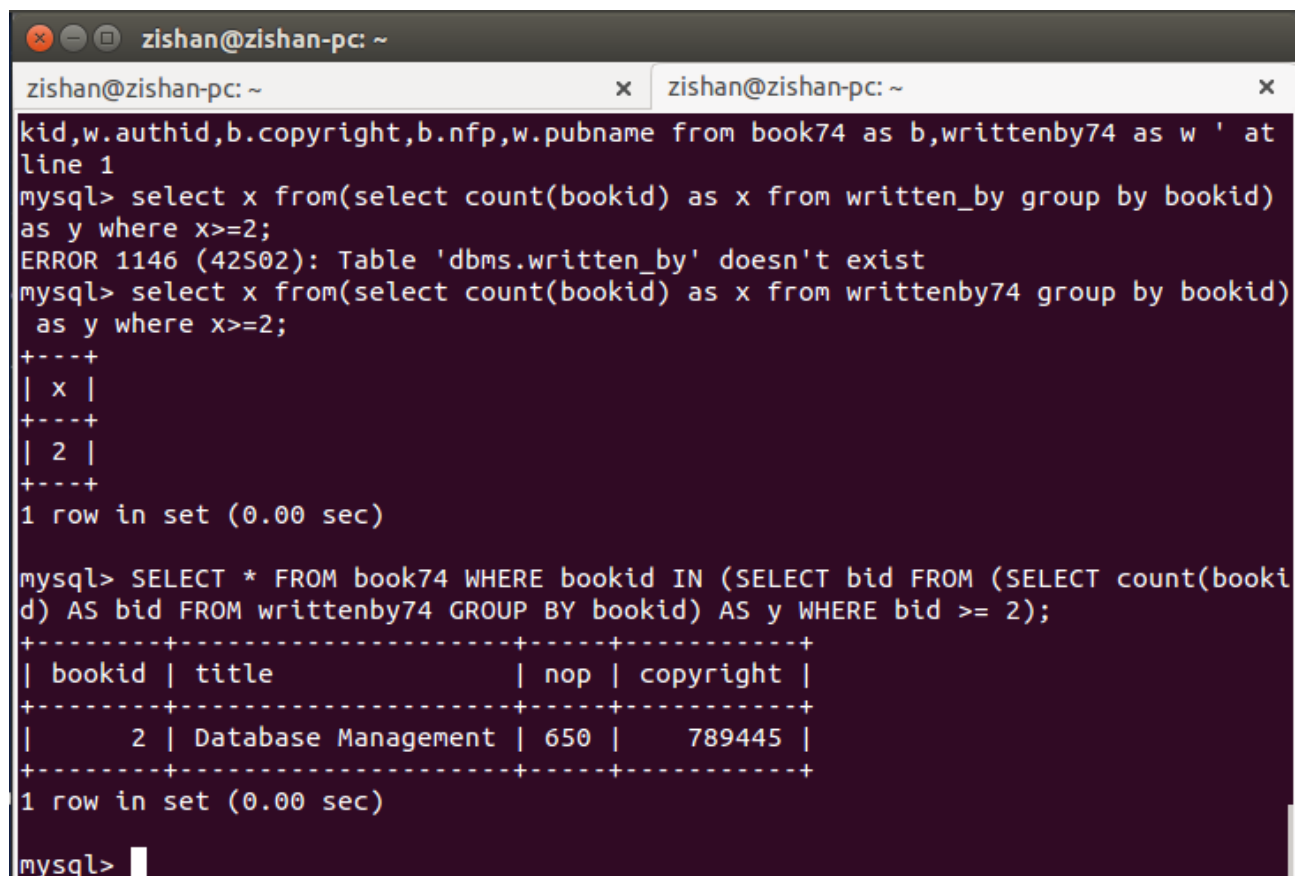
authid	authfirst	authmid	authlast	authcity	age
1	Arthur	Canon	Doyle	Alabama	59
2	S	L	Loney	Scotland	58

2 rows in set (0.00 sec)

```
mysql>
```

2. Give the details of the book which is written by at least two authors.

```
SELECT * FROM book74 WHERE bookid IN (  
    SELECT bid FROM (  
        SELECT count(bookid) AS bid FROM writtenby74 GROUP BY bookid  
    ) AS y WHERE bid >= 2  
);
```



The screenshot shows a terminal window with the title 'zishan@zishan-pc: ~'. It contains two tabs, both labeled 'zishan@zishan-pc: ~'. The terminal displays the following MySQL commands and their outputs:

```
mysql> select x from(select count(bookid) as x from written_by group by bookid)
as y where x>=2;
ERROR 1146 (42S02): Table 'dbms.written_by' doesn't exist
mysql> select x from(select count(bookid) as x from writtenby74 group by bookid)
as y where x>=2;
+----+
| x |
+----+
| 2 |
+----+
1 row in set (0.00 sec)
```

Then, the user runs the query from the problem statement:

```
mysql> SELECT * FROM book74 WHERE bookid IN (SELECT bid FROM (SELECT count(booki
d) AS bid FROM writtenby74 GROUP BY bookid) AS y WHERE bid >= 2);
```

The output is a table with 4 columns: bookid, title, nop, and copyright. It shows one row where bookid is 2, title is 'Database Management', nop is 650, and copyright is 789445.

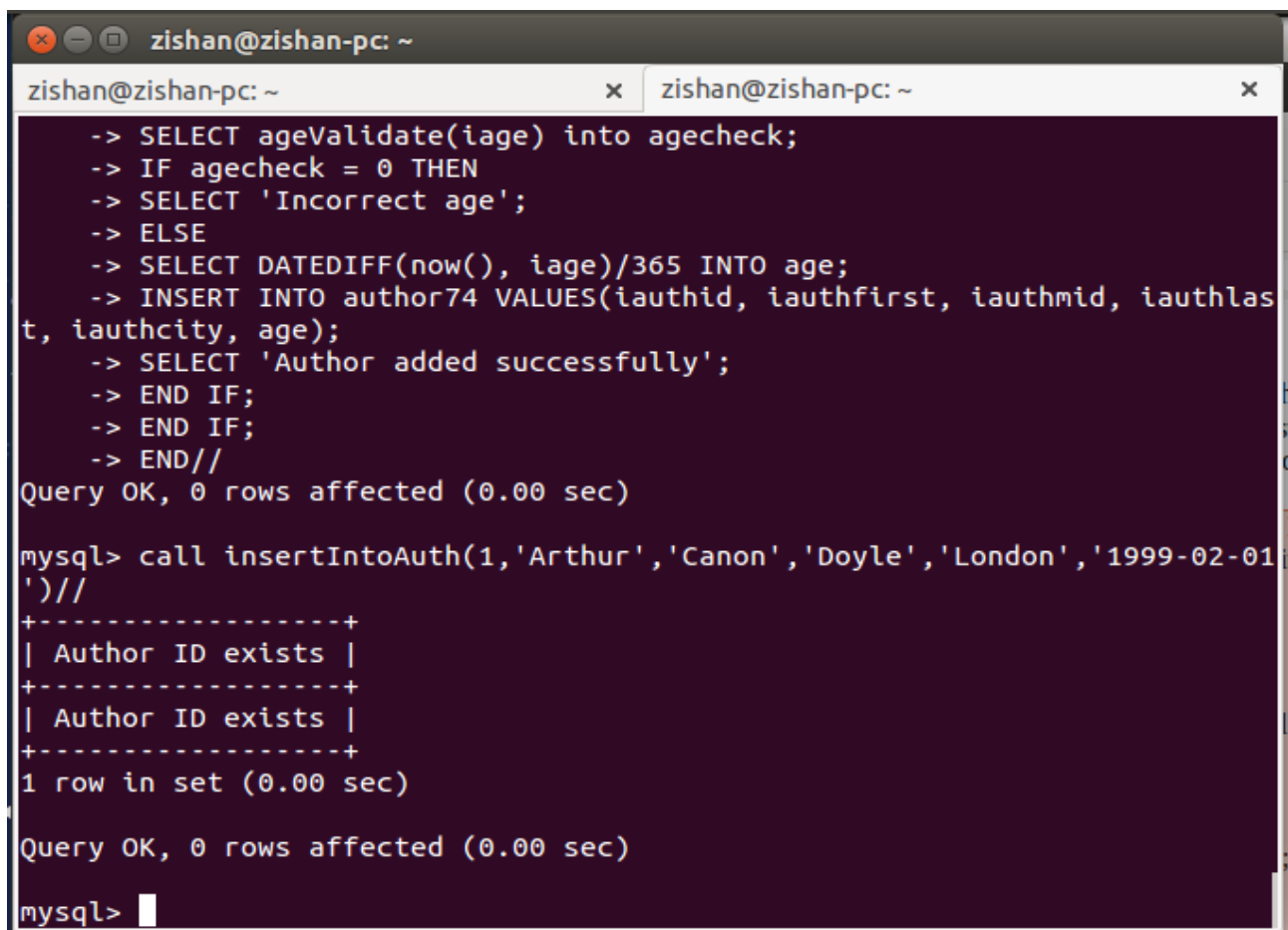
bookid	title	nop	copyright
2	Database Management	650	789445

```
1 row in set (0.00 sec)

mysql>
```

3. Write a stored procedure (SP Name : insertIntoAuth) to insert the Author information.

```
CREATE PROCEDURE insertIntoAuth(IN iauthid INT, IN iauthfirst VARCHAR(10), IN
iauthmid VARCHAR(10), IN iauthlast VARCHAR(10), IN iauthcity VARCHAR(15), IN
iage DATE)
BEGIN
    DECLARE agecheck INT DEFAULT 0;
    DECLARE idcheck INT DEFAULT 0;
    DECLARE age INT DEFAULT 0;
    SELECT authid FROM author74 WHERE authid = iauthid INTO idcheck;
    IF idcheck != 0 THEN
        SELECT 'Author ID exists';
    ELSE
        SELECT ageValidate(iage) into agecheck;
        IF agecheck = 0 THEN
            SELECT 'Incorrect age';
        ELSE
            SELECT DATEDIFF(now(), iage)/365 INTO age;
            INSERT INTO author74 VALUES(iauthid, iauthfirst, iauthmid,
iauthlast, iauthcity, age);
            SELECT 'Author added successfully';
        END IF;
    END IF;
END
```



The screenshot shows a terminal window with a dark background and light-colored text. The window title is 'zishan@zishan-pc: ~'. The terminal displays the following commands and output:

```
-> SELECT ageValidate(iage) into agecheck;
-> IF agecheck = 0 THEN
-> SELECT 'Incorrect age';
-> ELSE
-> SELECT DATEDIFF(now(), iage)/365 INTO age;
-> INSERT INTO author74 VALUES(iauthid, iauthfirst, iauthmid, iauthlast, iauthcity, age);
-> SELECT 'Author added successfully';
-> END IF;
-> END IF;
-> END//
Query OK, 0 rows affected (0.00 sec)

mysql> call insertIntoAuth(1,'Arthur','Canon','Doyle','London','1999-02-01')//
+-----+
| Author ID exists |
+-----+
| Author ID exists |
+-----+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

mysql> 
```

```
zishan@zishan-pc: ~
Query OK, 0 rows affected (0.00 sec)

mysql> call insertIntoAuth(1,'Arthur','Canon','Doyle','London','1999-02-01')//
+-----+
| Author ID exists |
+-----+
| Author ID exists |
+-----+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

mysql> call insertIntoAuth(2,'Arthur','Canon','Doyle','London','1999-02-01')//
+-----+
| Incorrect age |
+-----+
| Incorrect age |
+-----+
1 row in set (0.00 sec)

Query OK, 0 rows affected, 1 warning (0.00 sec)

mysql>
```

```
zishan@zishan-pc: ~
+-----+
1 row in set (0.00 sec)

Query OK, 0 rows affected, 1 warning (0.00 sec)

mysql> call insertIntoAuth(2,'Arthur','Canon','Doyle','London','1965-02-01')//
+-----+
| Author added successfully |
+-----+
| Author added successfully |
+-----+
1 row in set (0.04 sec)

Query OK, 0 rows affected (0.05 sec)

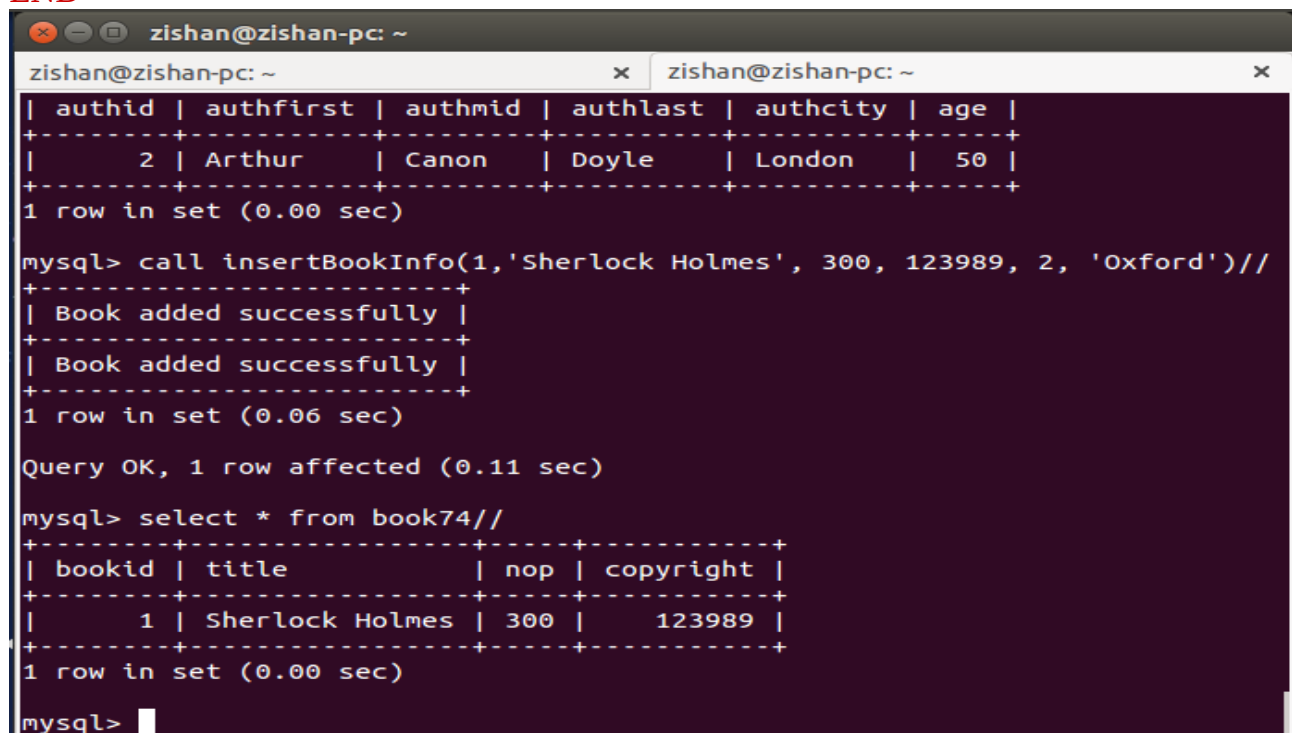
mysql> select * from author74 where authid = 2//
+-----+-----+-----+-----+-----+-----+
| authid | authfirst | authmid | authlast | authcity | age |
+-----+-----+-----+-----+-----+-----+
|      2 | Arthur   | Canon   | Doyle    | London   | 50  |
+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql>
```

4. Write a stored procedure (SP Name : insertBookInfo) to insert the book information such as bookid, title, no. of pages, copyright, authorId, Publisher Name. (Use two stored procedure and call it from one stored procedure i.e nested SP). (SP Name : insertBook, insertWBy).

```
CREATE PROCEDURE insertBookInfo (IN ibookid INT, IN itittle varchar(20), IN inop
INT, IN icopyright INT, IN iauidid INT, IN ipubname VARCHAR(20))
BEGIN
    DECLARE bidcheck INT DEFAULT 0;
    DECLARE aidcheck INT DEFAULT 0;
    SELECT bookid FROM book74 WHERE bookid = ibookid INTO bidcheck;
    IF bidcheck != 0 THEN
        SELECT 'Book ID exists';
    ELSE
        SELECT authid FROM author74 WHERE authid = iauidid INTO aidcheck;
        IF aidcheck = 0 THEN
            SELECT 'Author ID does not exist';
        ELSE
            INSERT INTO book74 VALUES(ibookid, itittle, inop, icopyright);
            SELECT 'Book added successfully';
            CALL insertWBy(ibookid, iauidid, ipubname);
        END IF;
    END IF;
END
```

```
CREATE PROCEDURE insertWBy(IN ibookid INT, IN iauidid INT, IN ipubname
VARCHAR(20))
BEGIN
    INSERT INTO writtenby74 VALUES(iauidid, ibookid, ipubname);
END
```



The screenshot shows a MySQL terminal window with the following content:

```
zishan@zishan-pc: ~
mysql> select * from author74;
+-----+-----+-----+-----+-----+-----+
| authid | authfirst | authmid | authlast | authcity | age |
+-----+-----+-----+-----+-----+-----+
|      2 | Arthur   | Canon   | Doyle    | London   | 50  |
+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql> call insertBookInfo(1,'Sherlock Holmes', 300, 123989, 2, 'Oxford')//
+-----+
| Book added successfully |
+-----+
| Book added successfully |
+-----+
1 row in set (0.06 sec)

Query OK, 1 row affected (0.11 sec)

mysql> select * from book74//
+-----+-----+-----+-----+
| bookid | title           | nop | copyright |
+-----+-----+-----+-----+
|      1 | Sherlock Holmes | 300 | 123989    |
+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql>
```

```
zishan@zishan-pc: ~
zishan@zishan-pc: ~
+-----+
1 row in set (0.00 sec)

Query OK, 0 rows affected, 1 warning (0.00 sec)

mysql> call insertIntoAuth(2,'Arthur','Canon','Doyle','London','1965-02-01')//
+-----+
| Author added successfully |
+-----+
| Author added successfully |
+-----+
1 row in set (0.04 sec)

Query OK, 0 rows affected (0.05 sec)

mysql> select * from author74 where authid = 2//
+-----+-----+-----+-----+-----+-----+
| authid | authfirst | authmid | authlast | authcity | age |
+-----+-----+-----+-----+-----+-----+
|      2 | Arthur   | Canon   | Doyle    | London   | 50 |
+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql> 
```

```
zishan@zishan-pc: ~
zishan@zishan-pc: ~
| Book added successfully |
+-----+
1 row in set (0.06 sec)

Query OK, 1 row affected (0.11 sec)

mysql> select * from book74//
+-----+-----+-----+-----+
| bookid | title           | nop | copyright |
+-----+-----+-----+-----+
|      1 | Sherlock Holmes | 300 | 123989    |
+-----+-----+-----+-----+
1 row in set (0.00 sec)

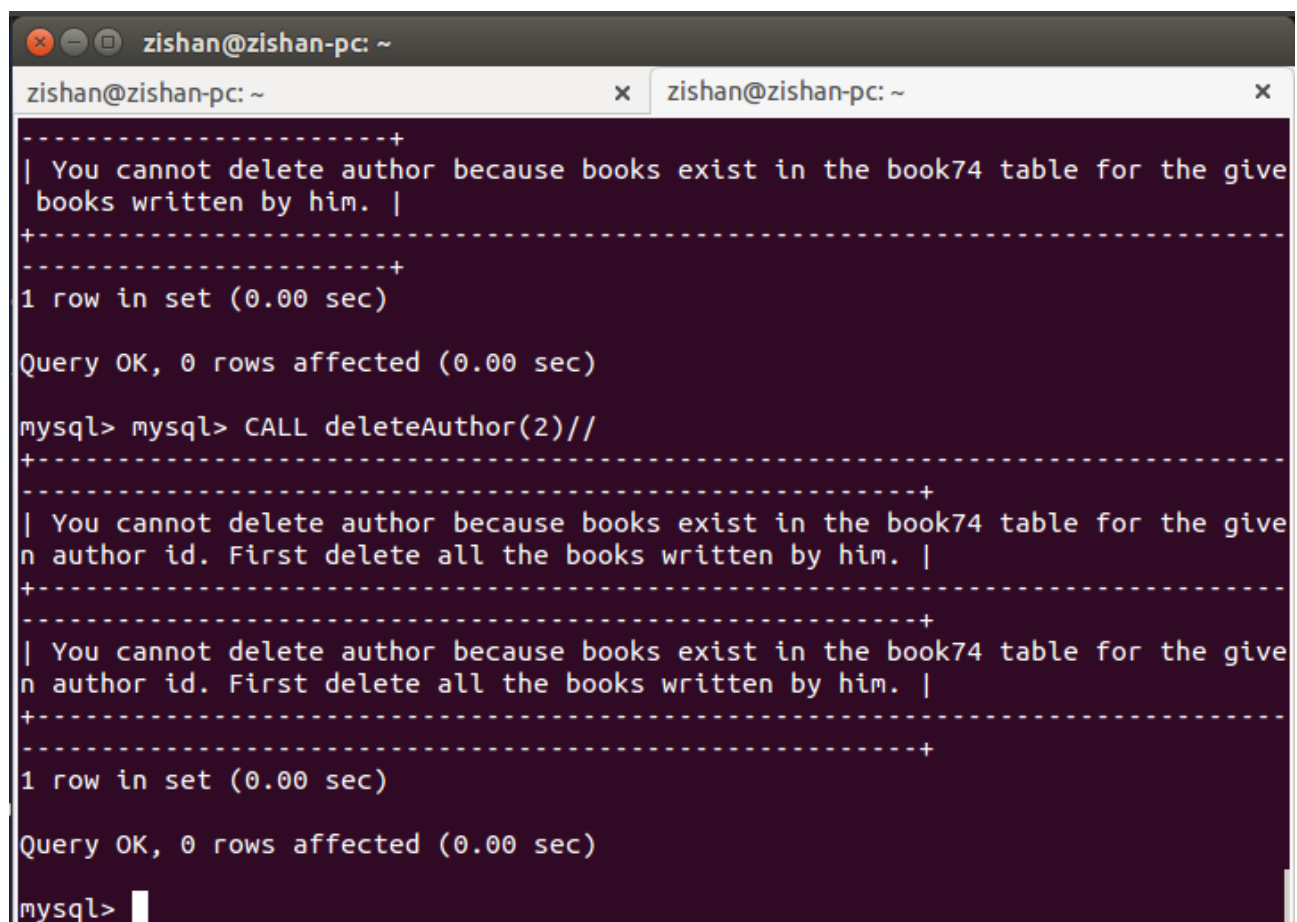
mysql> call insertBookInfo(1,'Sherlock Holmes', 300, 123989, 2, 'Oxford')//
+-----+
| Book ID exists |
+-----+
| Book ID exists |
+-----+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

mysql> 
```

5. Write a stored procedure to delete the Author information using its AuthID. (Note: If Author book (in Book Table) exists for AuthID, then it should display message as You can't delete Author because total no book exist in BookTable. First delete all the books written by him).

```
CREATE PROCEDURE deleteAuthor (IN iauthid INT)
BEGIN
    DECLARE bcheck INT DEFAULT 0;
    SELECT COUNT(*) FROM writtenby74 WHERE authid=iauthid INTO bcheck;
    IF bcheck > 0 THEN
        SELECT 'You cannot delete author because books exist in the book74 table for
the given author id. First delete all the books written by him.';
    ELSE
        DELETE FROM author74 WHERE authid = iauthid;
        SELECT 'Author deleted';
    END IF;
END
```



The screenshot shows a terminal window with a dark background and light-colored text. The window title is 'zishan@zishan-pc: ~'. There are two tabs open, both with the same title. The terminal content shows the execution of a MySQL query that returns a message: 'You cannot delete author because books exist in the book74 table for the given author id. First delete all the books written by him.' This is followed by a 'Query OK, 0 rows affected (0.00 sec)' message. Then, the user enters the command 'mysql> mysql> CALL deleteAuthor(2);'. This is followed by the same message as before, and another 'Query OK, 0 rows affected (0.00 sec)' message. The prompt 'mysql>' is visible at the bottom.

```
zishan@zishan-pc: ~
mysql> 
+-----+
| You cannot delete author because books exist in the book74 table for the give
books written by him. |
+-----+
+-----+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

mysql> mysql> CALL deleteAuthor(2);
+-----+
| You cannot delete author because books exist in the book74 table for the give
n author id. First delete all the books written by him. |
+-----+
+-----+
| You cannot delete author because books exist in the book74 table for the give
n author id. First delete all the books written by him. |
+-----+
+-----+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

mysql>
```



```
zishan@zishan-pc: ~
zishan@zishan-pc: ~ x zishan@zishan-pc: ~ x
+-----+-----+-----+
1 row in set (0.00 sec)

mysql> call deleteBook(2);
-> //
+-----+
| Book deleted |
+-----+
| Book deleted |
+-----+
1 row in set (0.05 sec)

Query OK, 0 rows affected (0.05 sec)

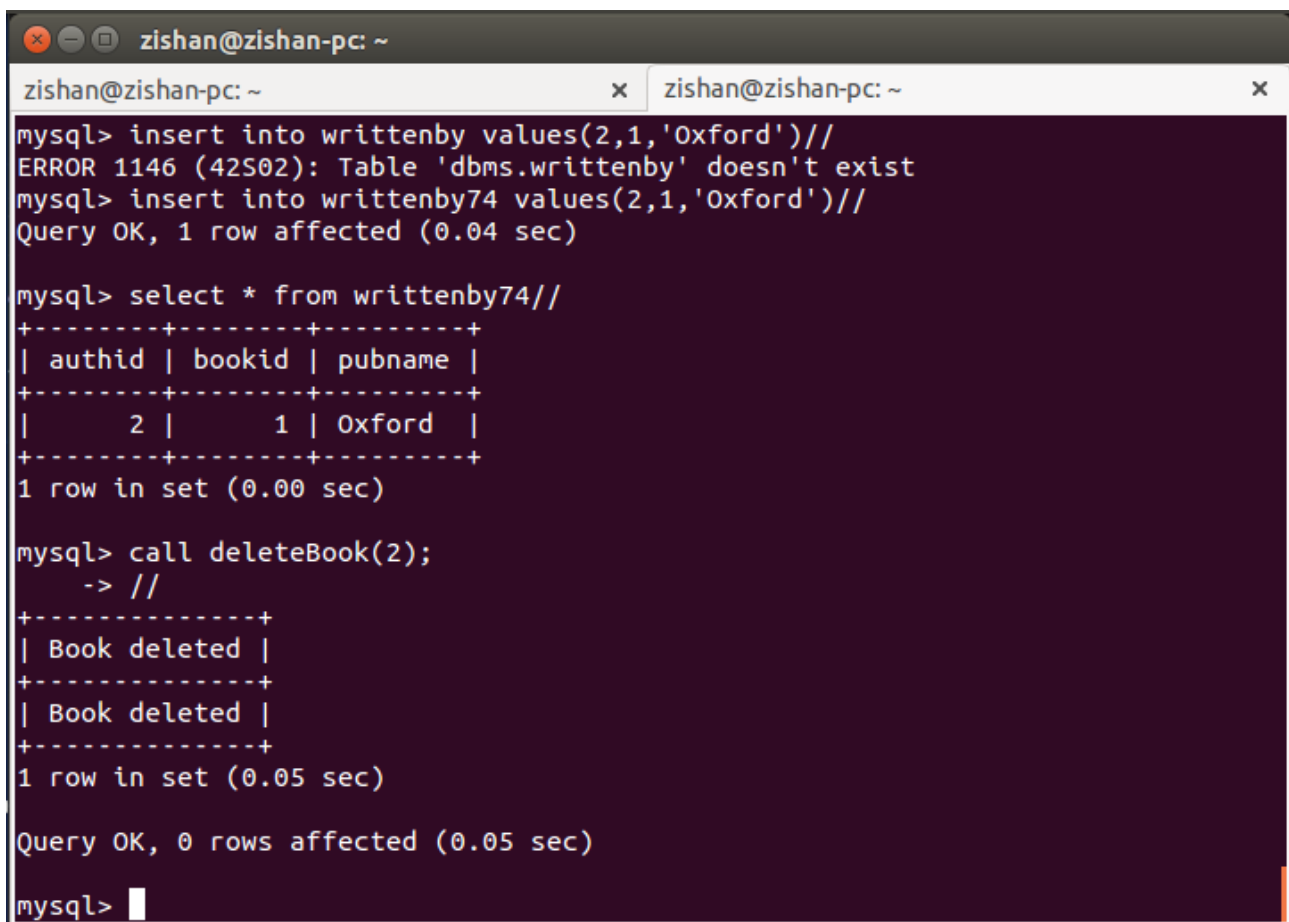
mysql> call deleteAuthor(2)//
+-----+
| Author deleted |
+-----+
| Author deleted |
+-----+
1 row in set (0.08 sec)

Query OK, 0 rows affected (0.08 sec)

mysql> 
```

6. Write a stored procedure to delete the Book using AuthID.

```
CREATE PROCEDURE deleteBook(IN iauthid INT)
BEGIN
    DECLARE bid INT;
    SELECT bookid FROM writtenby74 WHERE authid = iauthid INTO bid;
    DELETE FROM book74 WHERE bookid = bid;
    SELECT 'Book deleted';
END
```



The screenshot shows a terminal window with a MySQL prompt. The user attempts to insert a row into 'writtenby' but receives an error: 'ERROR 1146 (42S02): Table 'dbms.writtenby' doesn't exist'. They then successfully insert a row into 'writtenby74'. A subsequent 'select \* from writtenby74' query returns a single row with authid 2, bookid 1, and pubname 'Oxford'. Finally, the user calls the 'deleteBook(2)' stored procedure, which returns 'Book deleted' and reports 1 row in set.

```
mysql> insert into writtenby values(2,1,'Oxford')//
ERROR 1146 (42S02): Table 'dbms.writtenby' doesn't exist
mysql> insert into writtenby74 values(2,1,'Oxford')//
Query OK, 1 row affected (0.04 sec)

mysql> select * from writtenby74//
+-----+-----+-----+
| authid | bookid | pubname |
+-----+-----+-----+
|      2 |      1 | Oxford  |
+-----+-----+-----+
1 row in set (0.00 sec)

mysql> call deleteBook(2);
-> //
+-----+
| Book deleted |
+-----+
| Book deleted |
+-----+
1 row in set (0.05 sec)

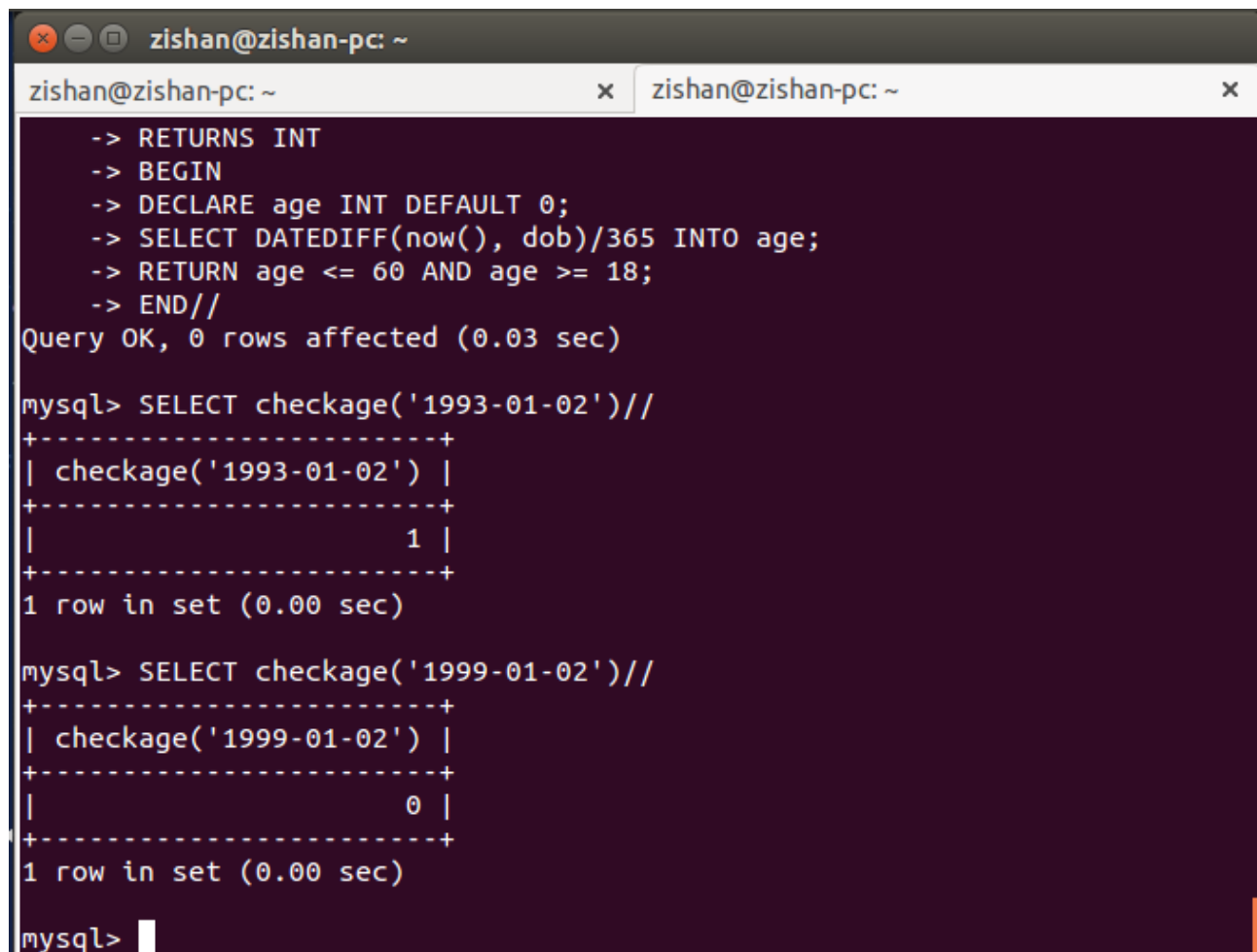
Query OK, 0 rows affected (0.05 sec)

mysql>
```

Q5. Create function that validate the age of employee. Function accept the dob of employee and

return 1 if age is lies between 18 and 60 else re turn 0

```
CREATE FUNCTION checkage(dob date)
RETURNS INT
BEGIN
    DECLARE age INT DEFAULT 0;
    SELECT DATEDIFF(now(), dob)/365 INTO age;
    RETURN age <= 60 AND age >= 18;
END
```



```
zishan@zishan-pc: ~
zishan@zishan-pc: ~ x zishan@zishan-pc: ~ x
-> RETURNS INT
-> BEGIN
-> DECLARE age INT DEFAULT 0;
-> SELECT DATEDIFF(now(), dob)/365 INTO age;
-> RETURN age <= 60 AND age >= 18;
-> END//
Query OK, 0 rows affected (0.03 sec)

mysql> SELECT checkage('1993-01-02')//
+-----+
| checkage('1993-01-02') |
+-----+
| 1 |
+-----+
1 row in set (0.00 sec)

mysql> SELECT checkage('1999-01-02')//
+-----+
| checkage('1999-01-02') |
+-----+
| 0 |
+-----+
1 row in set (0.00 sec)

mysql> 
```

Q6. Consider a following table of a database :

Book(bid, bname,authname)

1. Create triggers which create a log of every Insert ,Delete and Upda te operation on the book table record.

It should also hold the username who was operating at that time and t ime and type of operation.

NOTE: log table attributes are user, operation, pbid, pbname, pauthnam  
e, nbid, nbname, nauthname and timeofop

### INSERT TRIGGER:

CREATE TRIGGER insertlog AFTER INSERT ON book74 FOR EACH ROW  
BEGIN

INSERT INTO log74 VALUE(user(), "insert", '-', "-", "-", new.bid, new.bname,  
new.authname, now());

END

```
zishan@zishan-pc: ~
mysql> select * from book74;
Empty set (0.00 sec)

mysql> insert into book74 values(100, "Famous Five", "Enid Blyton");
Query OK, 1 row affected (0.33 sec)

mysql> select * from book74;
+-----+-----+-----+
| bid | bname      | authname |
+-----+-----+-----+
| 100 | Famous Five | Enid Blyton |
+-----+-----+-----+
1 row in set (0.00 sec)

mysql> select * from log74;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| user          | operation | pbid | pbname | pauthname | nbid | nbname      | nauthname | timeofop |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| root@localhost | insert   | 0    | -      | -          | 100 | Famous Five | Enid Blyton | 2014-11-11 18:43:29 |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql>
```

```
zishan@zishan-pc: ~
mysql> select * from log74;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| user          | operation | pbid | pbname | pauthname | nbid | nbname      | nauthname | timeofop |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| root@localhost | insert   | 0    | -      | -          | 100 | Famous Five | Enid Blyton | 2014-11-11 18:43:29 |
| root@localhost | insert   | 0    | -      | -          | 101 | Mathematics | M L Khanna  | 2014-11-11 18:51:57 |
| root@localhost | insert   | 0    | -      | -          | 102 | Database Management | Korth      | 2014-11-11 18:52:37 |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql> select * from book74;
+-----+-----+-----+
| bid | bname          | authname |
+-----+-----+-----+
| 100 | Famous Five    | Enid Blyton |
| 101 | Mathematics    | M L Khanna  |
| 102 | Database Management | Korth      |
+-----+-----+-----+
3 rows in set (0.00 sec)

mysql>
```

## UPDATE TRIGGER:

CREATE TRIGGER updatelog AFTER UPDATE ON book74 FOR EACH ROW  
BEGIN

INSERT INTO log74 VALUE(user(), 'update', old.bid, old.bname, old.authname,  
new.bid, new.bname, new.authname, now());  
END

```
zishan@zishan-pc: ~
mysql> select * from book74;
+----+
| bid | bname          | authname |
+----+
| 100 | Famous Five    | Enid Blyton |
| 101 | Maths          | M L Khanna |
| 102 | Database Management | Korth |
+----+
3 rows in set (0.00 sec)

mysql> select * from log74;
+----+
| user          | operation | pbid | pbname          | pauthname | nbid | nbname          | nauthname | timeofop |
+----+
| root@localhost | insert    | 0    | -               | -         | 100 | Famous Five    | Enid Blyton | 2014-11-11 18:43:29 |
| root@localhost | insert    | 0    | -               | -         | 101 | Mathematics    | M L Khanna | 2014-11-11 18:51:57 |
| root@localhost | insert    | 0    | -               | -         | 102 | Database Management | Korth | 2014-11-11 18:52:37 |
| root@localhost | update    | 101 | Mathematics    | M L Khanna | 101 | Maths          | M L Khanna | 2014-11-11 19:06:23 |
+----+
4 rows in set (0.00 sec)

mysql>
```

## DELETE TRIGGER:

CREATE TRIGGER deletelog BEFORE DELETE ON book74 FOR EACH ROW  
BEGIN

INSERT INTO log74 VALUE(user(), 'delete', old.bid, old.bname, old.authname,  
'-', '-', '-', now());  
END

```
zishan@zishan-pc: ~
mysql> delete from book74 where bid = 100;
Query OK, 1 row affected (0.05 sec)

mysql> select * from book74;
+----+
| bid | bname          | authname |
+----+
| 101 | Maths          | M L Khanna |
| 102 | Database Management | Korth |
+----+
2 rows in set (0.00 sec)

mysql> select * from log74;
+----+
| user          | operation | pbid | pbname          | pauthname | nbid | nbname          | nauthname | timeofop |
+----+
| root@localhost | insert    | 0    | -               | -         | 100 | Famous Five    | Enid Blyton | 2014-11-11 18:43:29 |
| root@localhost | insert    | 0    | -               | -         | 101 | Mathematics    | M L Khanna | 2014-11-11 18:51:57 |
| root@localhost | insert    | 0    | -               | -         | 102 | Database Management | Korth | 2014-11-11 18:52:37 |
| root@localhost | update    | 101 | Mathematics    | M L Khanna | 101 | Maths          | M L Khanna | 2014-11-11 19:06:23 |
| root@localhost | delete    | 100 | Famous Five    | Enid Blyton | 0    | -               | -         | 2014-11-11 19:10:12 |
+----+
5 rows in set (0.00 sec)

mysql>
```