DBMS LAB ASSIGNMENT

13bcs0003 | 5th SEM.

submitted by

abdul rauf

CREATING THE TABLES.

CREATE TABLE department\_13BCS0003 (

deptno int PRIMARY KEY,

dname varchar(20),

manager varchar(20),

loc varchar(20)

);

CREATE TABLE employee\_13BCS0003 (

empno int PRIMARY KEY,

name varchar(30),

deptno int,

job varchar(10),

hiredate date,

sal int,

commission int,

dob date,

city varchar(20),

phone int,

FOREIGN KEY (deptno) REFERENCES department\_13BCS0003 (deptno) ON UPDATE CASCADE ON DELETE CASCADE

);

CREATE TABLE salary\_13BCS0003 (

eno int,

basic int,

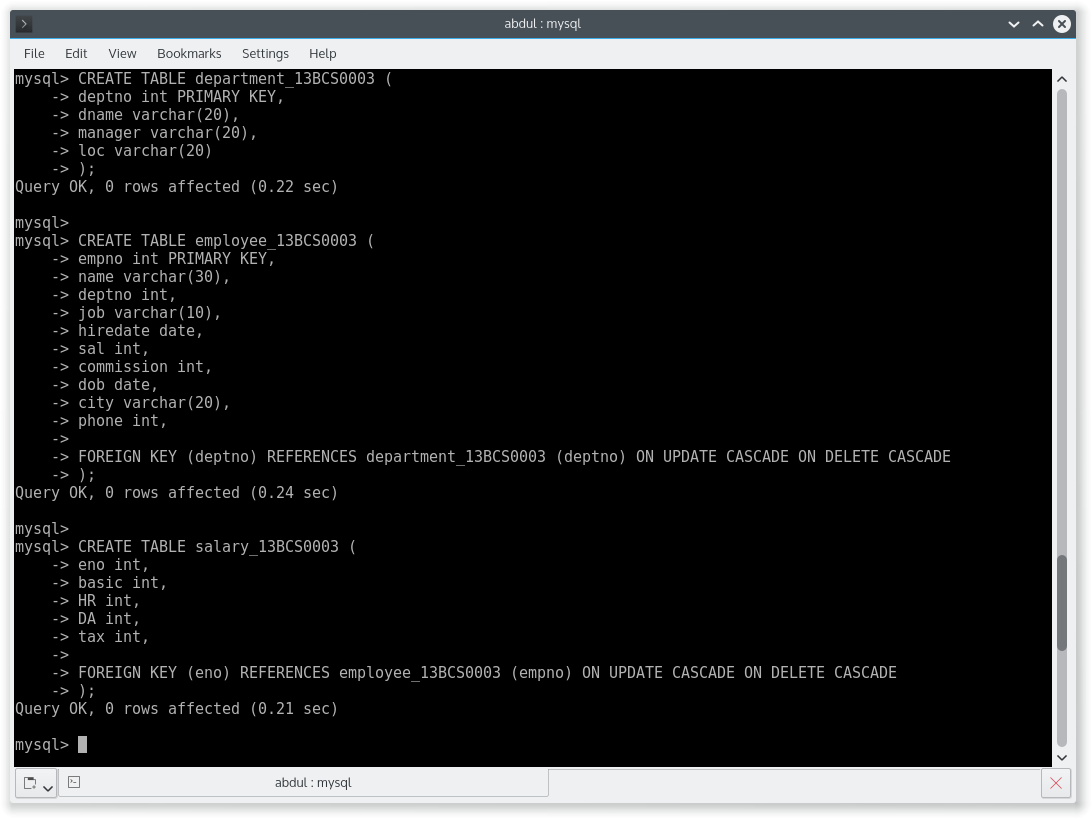
HR int,

DA int,

tax int,

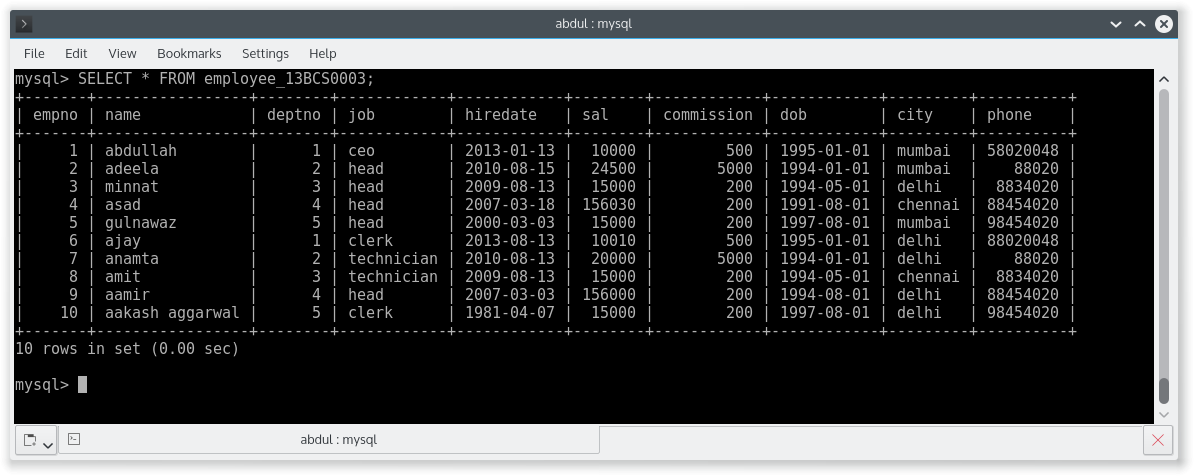
FOREIGN KEY (eno) REFERENCES employee\_13BCS0003 (empno) ON UPDATE CASCADE ON DELETE CASCADE

);

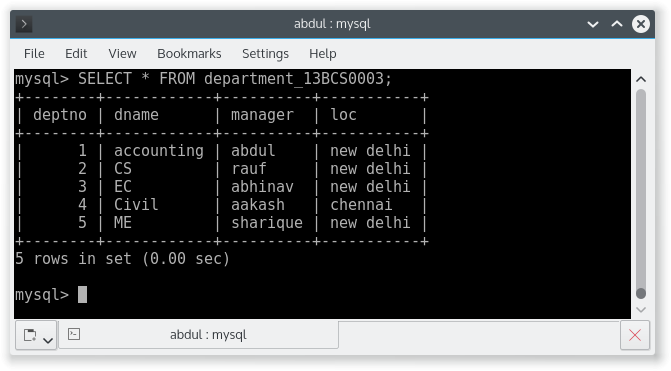


DATA IN TABLE employee\_13BCS0003

SELECT \* FROM employee\_13BCS0003;

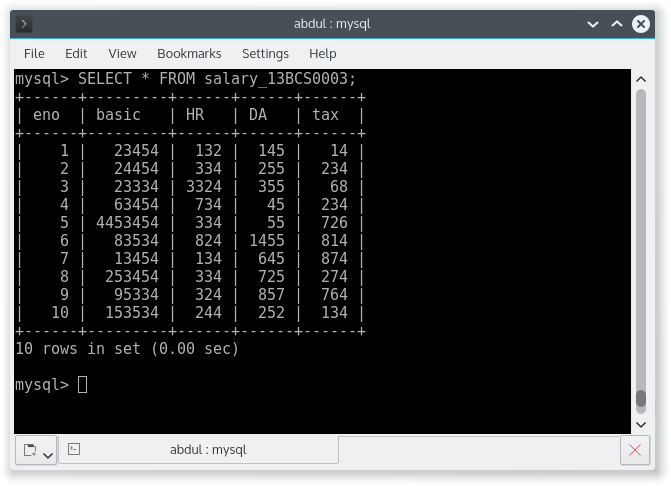


DATA IN TABLE department\_13BCS0003

SELECT \* FROM department\_13BCS0003;

DATA IN TABLE employee\_13BCS0003

SELECT \* FROM salary\_13BCS0003



Q1. Get the name and city of the employee working for the accounting department?

SELECT

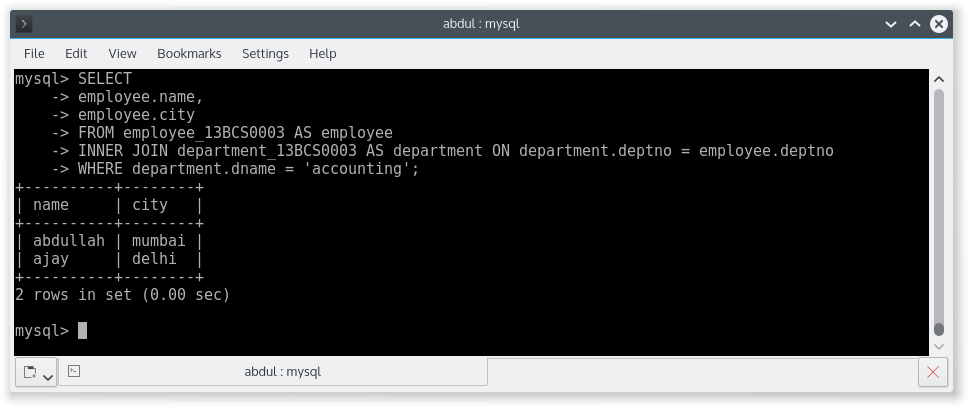
employee.name,

employee.city

FROM employee\_13BCS0003 AS employee

INNER JOIN department\_13BCS0003 AS department ON department.deptno = employee.deptno

WHERE department.dname = 'accounting';



Q2. Get the name, department, name of all the employees whose pay is greater than 1000.

SELECT

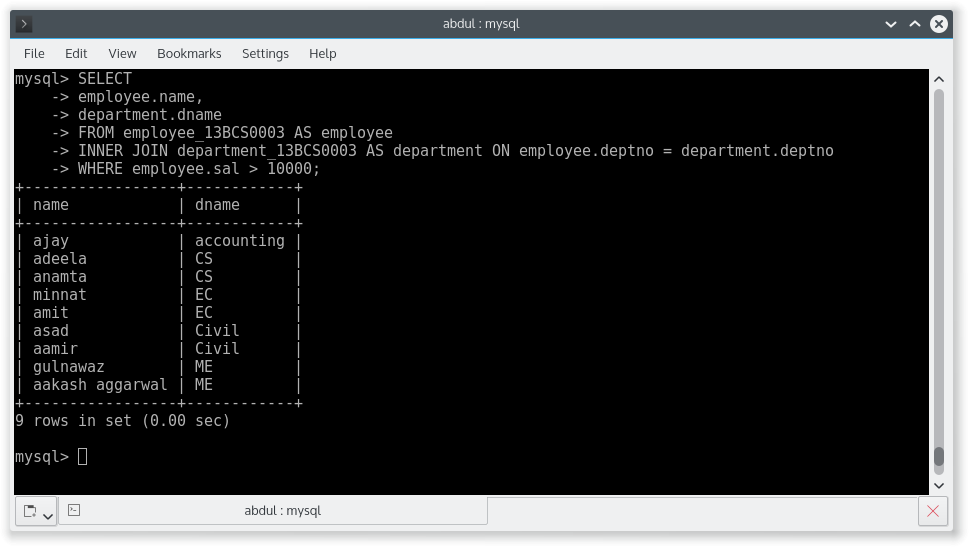
employee.name,

department.dname

FROM employee\_13BCS0003 AS employee

INNER JOIN department\_13BCS0003 AS department ON employee.deptno = department.deptno

WHERE employee.sal > 10000;



Q3. Get the name of the employee in ascending and descending order?

SELECT

employee.name

FROM employee\_13BCS0003 AS employee

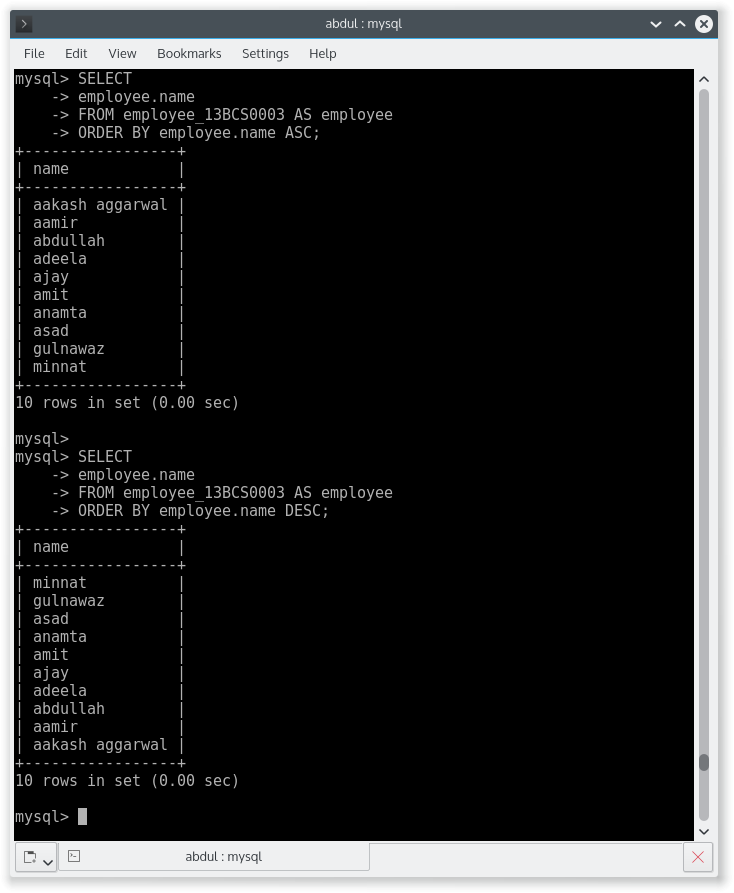
ORDER BY employee.name ASC;

SELECT

employee.name

FROM employee\_13BCS0003 AS employee

ORDER BY employee.name DESC;



Q4. Update the city of the employee no. 2 from Mumbai to Delhi.

UPDATE employee\_13BCS0003

SET city = "Delhi"

WHERE empno = 2;



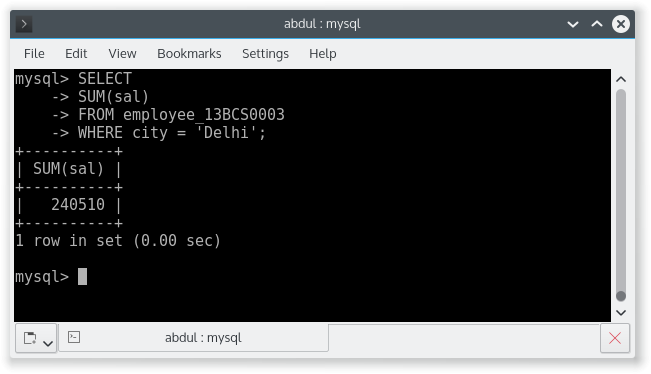
Q5. Get the sum of the basic salary of the employees which belongs to Delhi City.

SELECT

SUM(sal)

FROM employee\_13BCS0003

WHERE city = 'Delhi';



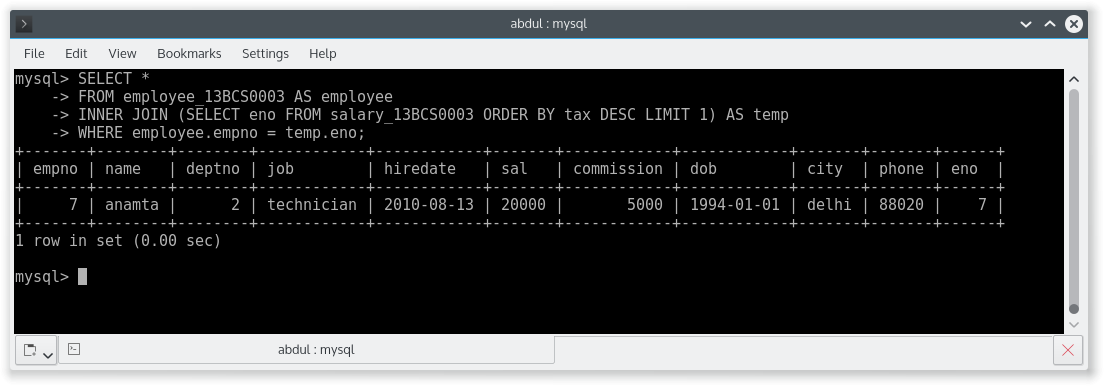
Q6. Get the details of the highest income tax payee.

SELECT \*

FROM employee\_13BCS0003 AS employee

INNER JOIN (SELECT eno FROM salary\_13BCS0003 ORDER BY tax DESC LIMIT 1) AS temp

WHERE employee.empno = temp.eno;



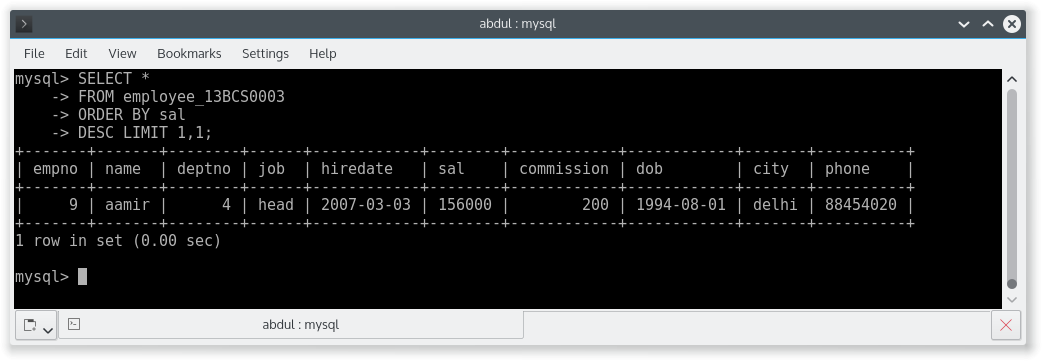
Q8. Give the details of the second highest salary employee (without use of < operator).

SELECT \*

FROM employee\_13BCS0003

ORDER BY sal

DESC LIMIT 1,1;

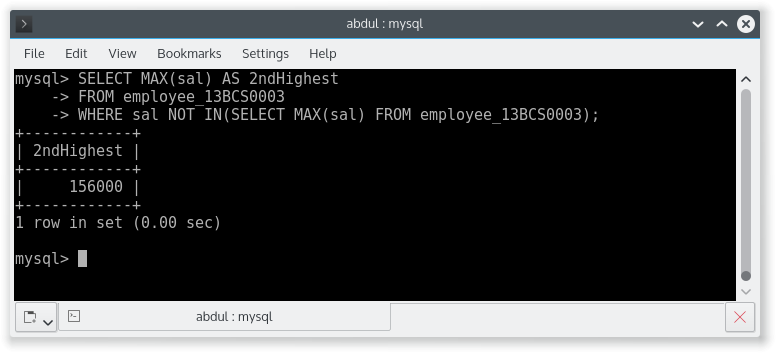


Q 9. Give the details of the second highest salary employee (without use of MAX and LIMIT operator).

SELECT MAX(sal) AS 2ndHighest

FROM employee\_13BCS0003

WHERE sal NOT IN(SELECT MAX(sal) FROM employee\_13BCS0003);

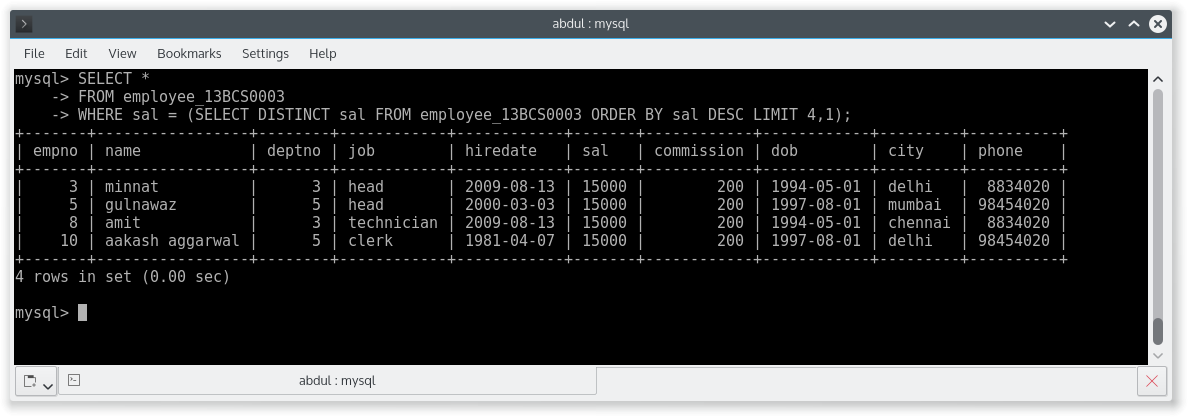


Q 11. Give the details of all employees of 5th highest salary?

SELECT \*

FROM employee\_13BCS0003

WHERE sal = (SELECT DISTINCT sal FROM employee\_13BCS0003 ORDER BY sal DESC LIMIT 4,1);

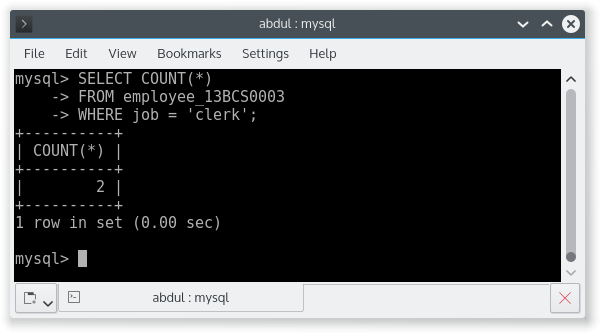


Q 12. How many clerks are there in the company?

SELECT COUNT(\*)

FROM employee\_13BCS0003

WHERE job = 'clerk';



Q 13. Which department has exactly one employee as clerk?

SELECT

department.dname

FROM department\_13BCS0003 AS department

INNER JOIN (SELECT deptno FROM employee\_13BCS0003 WHERE job='clerk' GROUP BY deptno HAVING count(\*)=1) AS temp

WHERE department.deptno = temp.deptno;



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

SELECT

deptno,

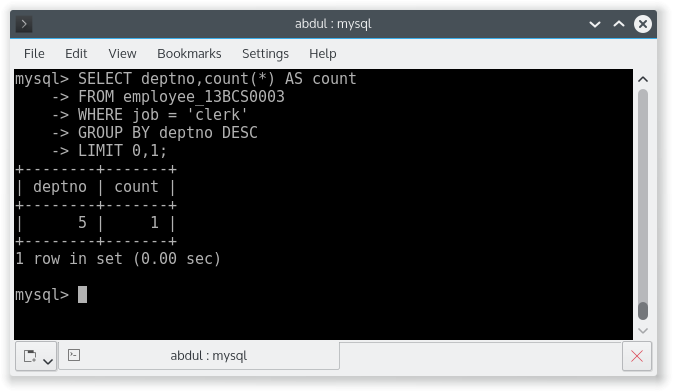
count(\*) AS count

FROM employee\_13BCS0003

WHERE job = 'clerk'

GROUP BY deptno DESC

LIMIT 0,1;



Q15. How many employees are there in each department?

SELECT

department.deptno,

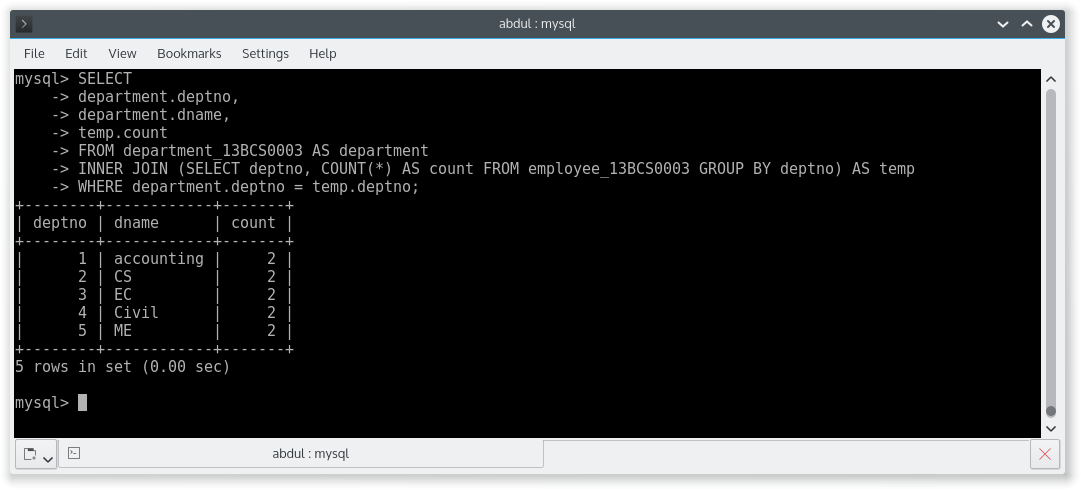
department.dname,

temp.count

FROM department\_13BCS0003 AS department

INNER JOIN (SELECT deptno, COUNT(\*) AS count FROM employee\_13BCS0003 GROUP BY deptno) AS temp

WHERE department.deptno = temp.deptno;



Q16. List the lowest salary for different jobs used in the company and list them in descending order.

SELECT

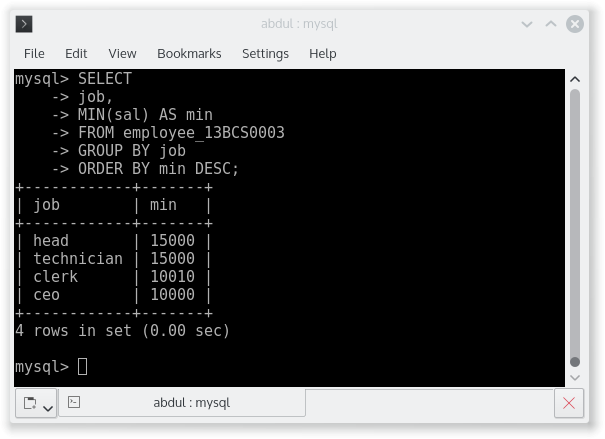
job,

MIN(sal) AS min

FROM employee\_13BCS0003

GROUP BY job

ORDER BY min DESC;



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

SELECT

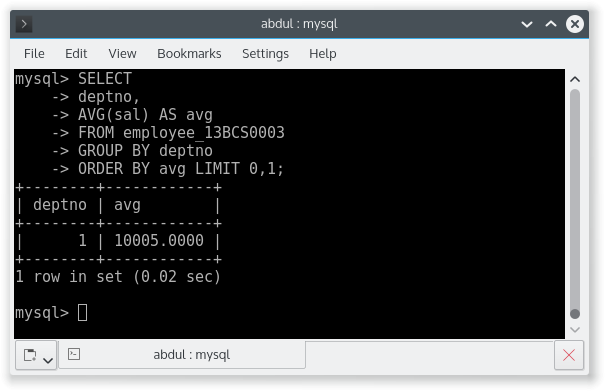
deptno,

AVG(sal) AS avg

FROM employee\_13BCS0003

GROUP BY deptno

ORDER BY avg LIMIT 0,1;



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

SELECT

job,

MIN(sal) AS min,

MAX(sal) as max,

avg(sal) as avg

FROM employee\_13BCS0003

GROUP BY job;



Q 19. Compute the difference between maximum and minimum salary.

SELECT

(MAX(sal)-MIN(sal)) AS difference

FROM employee\_13BCS0003;



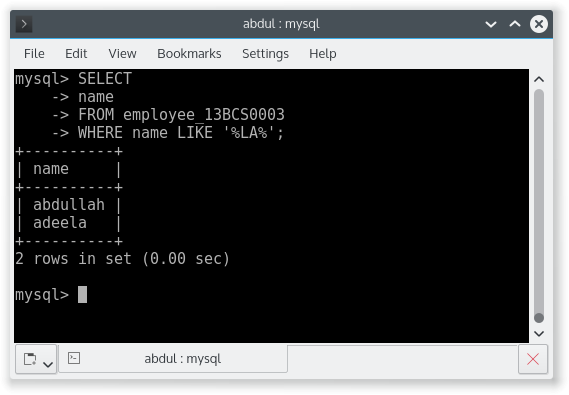
Q 20. List the names of the employees whose name contains LA.

SELECT

name

FROM employee\_13BCS0003

WHERE name LIKE '%LA%';



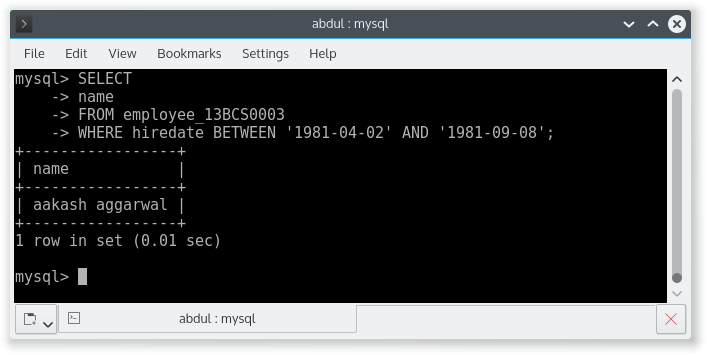
Q 21. List the names of the employees whose joining date is between 2nd April 1981 and 8th Sept 1981.

SELECT

name

FROM employee\_13BCS0003

WHERE hiredate BETWEEN '1981-04-02' AND '1981-09-08';

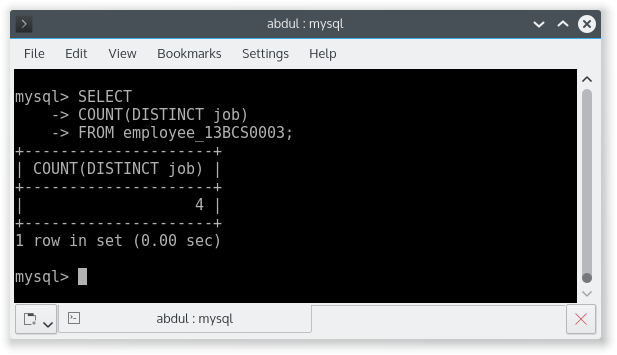


Q 22. How many different job titles exist in the employee table?

SELECT

COUNT(DISTINCT job)

FROM employee\_13BCS0003;



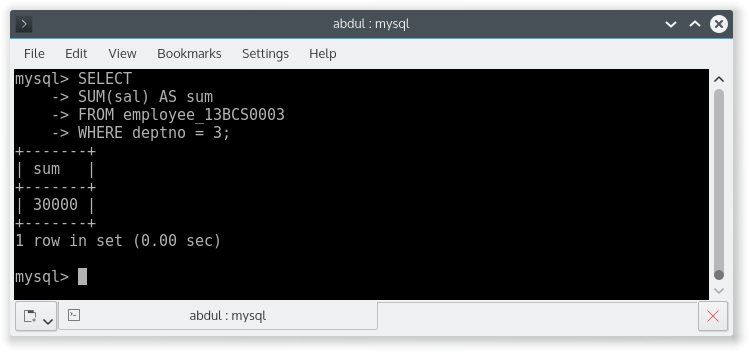
Q 23. Compute the sum of all salaries of employees working under deptno = 3?

SELECT

SUM(sal) AS sum

FROM employee\_13BCS0003

WHERE deptno = 3;



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

SELECT

employee.name,

employee.deptno,

department.dname

FROM employee\_13BCS0003 AS employee

INNER JOIN department\_13BCS0003 AS department ON employee.deptno=department.deptno;



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

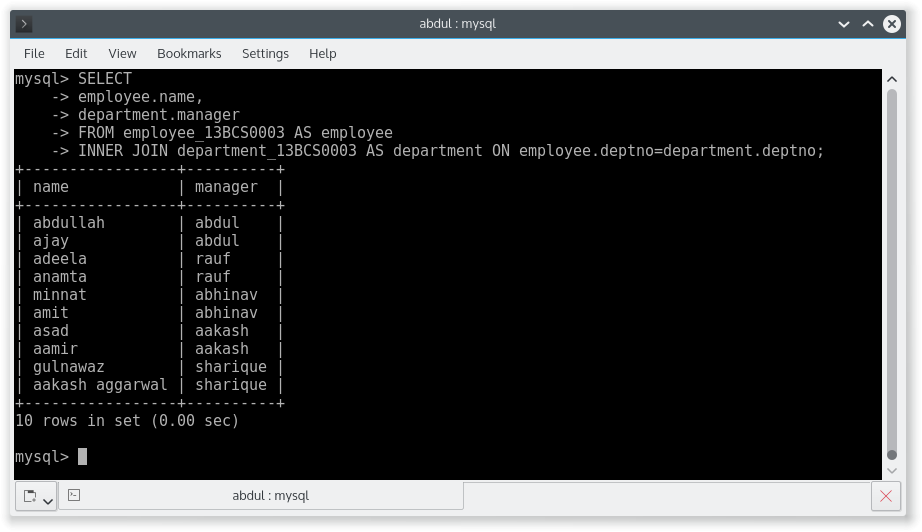
SELECT

employee.name,

department.manager

FROM employee\_13BCS0003 AS employee

INNER JOIN department\_13BCS0003 AS department ON employee.deptno=department.deptno;



Q 26. List all the employees who are working in department located at CHENNAI.

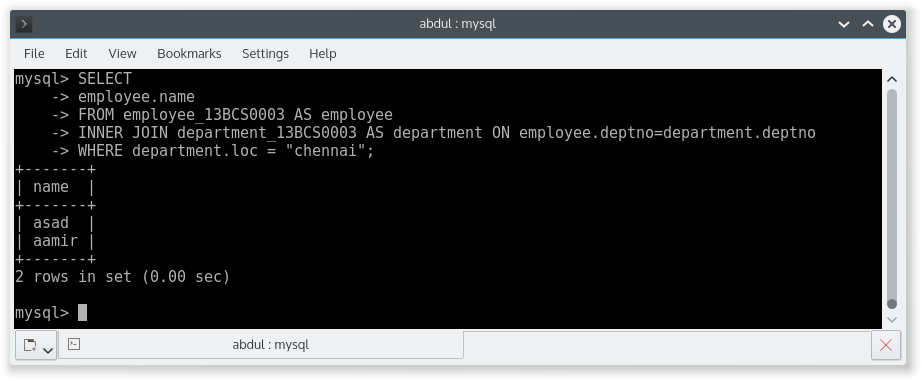
SELECT

employee.name

FROM employee\_13BCS0003 AS employee

INNER JOIN department\_13BCS0003 AS department ON employee.deptno=department.deptno

WHERE department.loc = "chennai";



Q27. List all the employees who are working in same department as their managers.

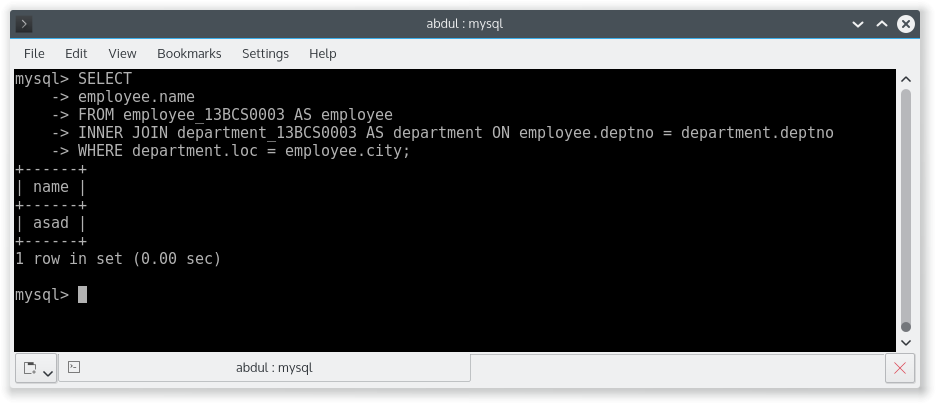
SELECT

employee.name

FROM employee\_13BCS0003 AS employee

INNER JOIN department\_13BCS0003 AS department ON employee.deptno=department.deptno

WHERE department.loc = employee.city;





Q 29. List all the departments who have no employees.

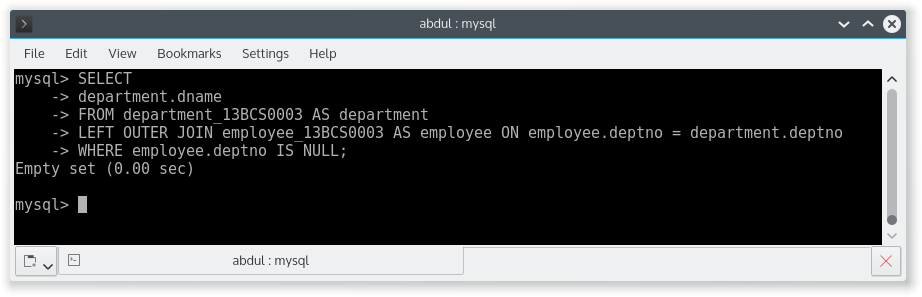
SELECT

department.dname

FROM department\_13BCS0003 AS department

LEFT OUTER JOIN employee\_13BCS0003 AS employee ON employee.deptno = department.deptno

WHERE employee.deptno IS NULL;



Q 30. Delete the EC department.

DELETE FROM department\_13BCS0003

WHERE dname = "EC";



CREATION OF TABLES.

CREATE TABLE class\_13BCS0003 (

cid int PRIMARY KEY,

cname char(20),

duration int

);

CREATE TABLE student\_13BCS0003 (

stud\_no int PRIMARY KEY,

stud\_name varchar(20),

dob date,

cid int,

FOREIGN KEY (cid) REFERENCES class\_13BCS0003(cid) ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE lab\_13BCS0003 (

mach\_no varchar(20) PRIMARY KEY,

lab\_no int,

description varchar(100)

);

CREATE TABLE allotment\_13BCS0003 (

stud\_no int,

mach\_no varchar(20),

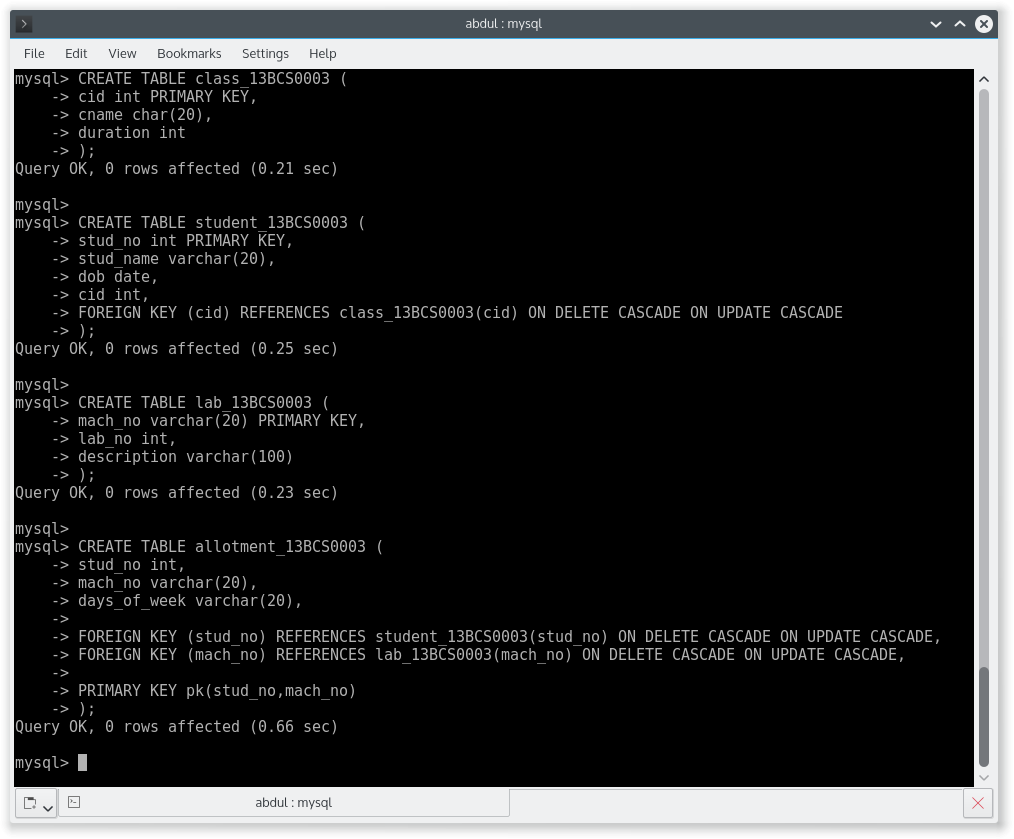
days\_of\_week varchar(20),

FOREIGN KEY (stud\_no) REFERENCES student\_13BCS0003(stud\_no) ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY (mach\_no) REFERENCES lab\_13BCS0003(mach\_no) ON DELETE CASCADE ON UPDATE CASCADE,

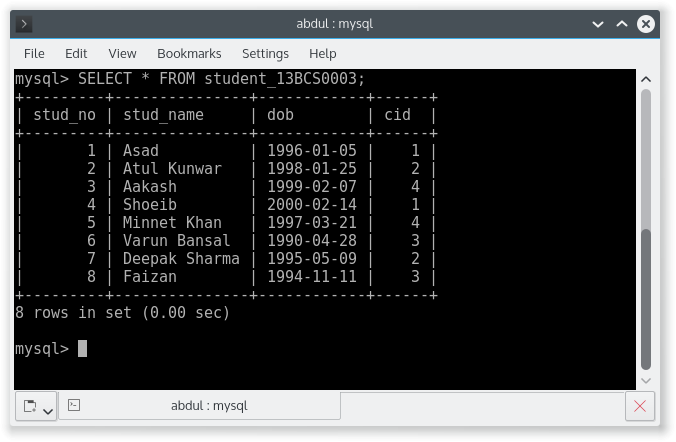
PRIMARY KEY pk(stud\_no,mach\_no)

);



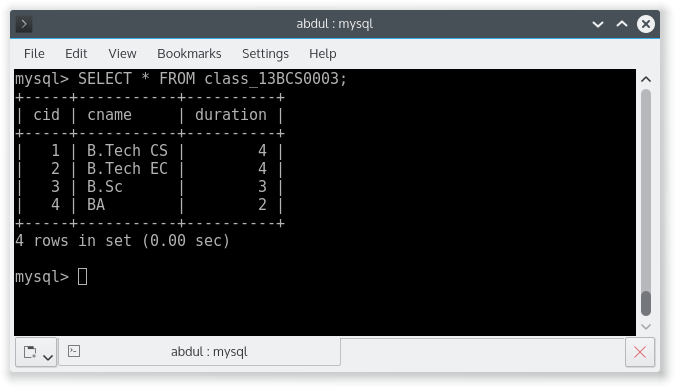
DATA IN student\_13BCS0003.

SELECT \* FROM student\_13BCS0003;



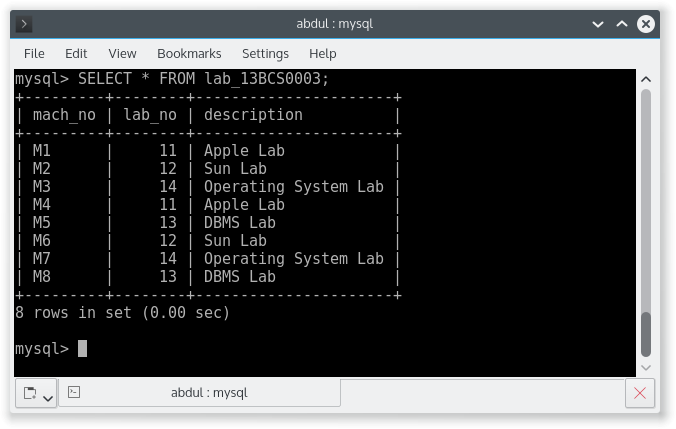
DATA IN class\_13BCS0003.

SELECT \* class\_13BCS0003;



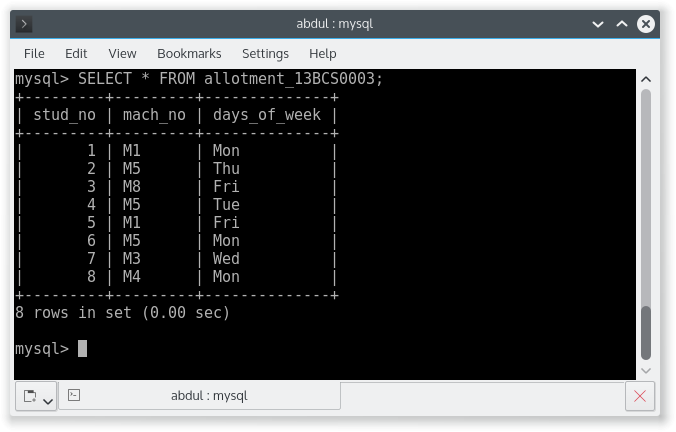
DATA IN lab\_13BCS0003.

SELECT \* FROM lab\_13BCS0003;



DATA IN allotment\_13BCS0003.

SELECT \* FROM allotment\_13BCS0003;



Q 2. List all the machines allotments with the student names, labs and the machine numbers.

SELECT student.stud\_name,

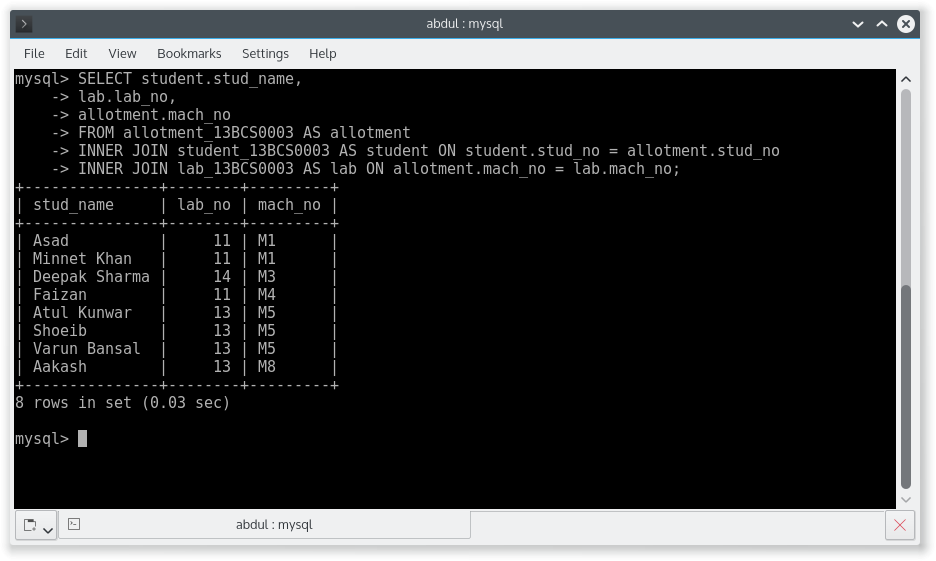
lab.lab\_no,

allotment.mach\_no

FROM allotment\_13BCS0003 AS allotment

INNER JOIN student\_13BCS0003 AS student ON student.stud\_no = allotment.stud\_no

INNER JOIN lab\_13BCS0003 AS lab ON allotment.mach\_no = lab.mach\_no;



Q 3. List the total number of lab allotment day wise.

SELECT

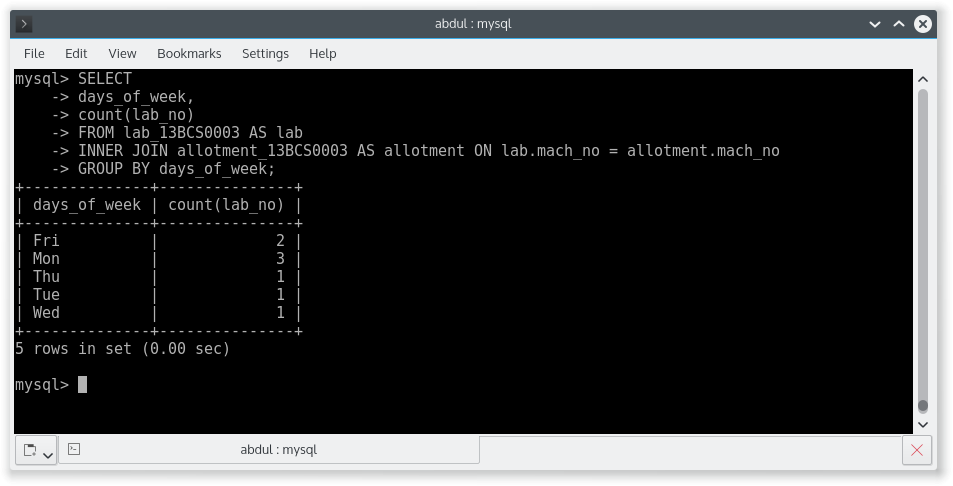
days\_of\_week,

count(lab\_no)

FROM lab\_13BCS0003 AS lab

INNER JOIN allotment\_13BCS0003 AS allotment ON lab.mach\_no = allotment.mach\_no

GROUP BY days\_of\_week;



Q 4. Give a count of how many machines have been allocated to the ‘B.Tech CS’ class or particular class.

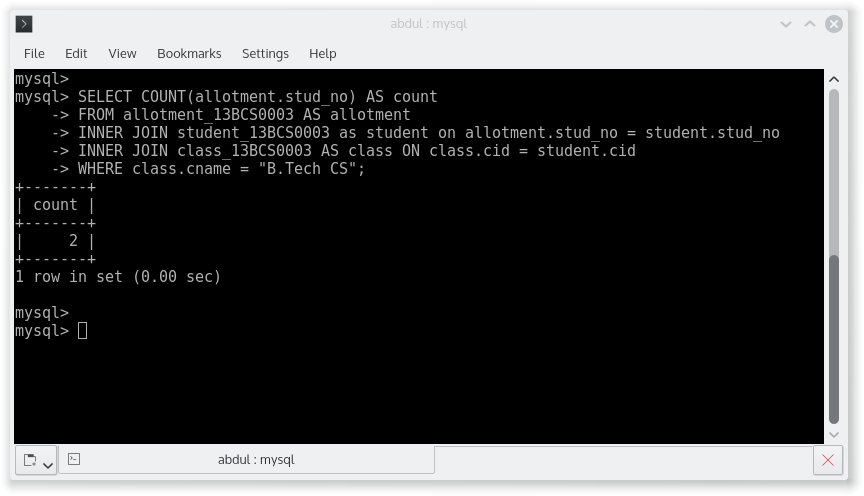
SELECT COUNT(allotment.stud\_no) AS count

FROM allotment\_13BCS0003 AS allotment

INNER JOIN student\_13BCS0003 as student on allotment.stud\_no = student.stud\_no

INNER JOIN class\_13BCS0003 AS class ON class.cid = student.cid

WHERE class.cname = "B.Tech CS";



Q 5. Give a machine allotment details of the stud\_no 5 ( or particular student) with his personal and class details.

SELECT

student.\*,

allotment.mach\_no,

allotment.days\_of\_week,

class.cname,

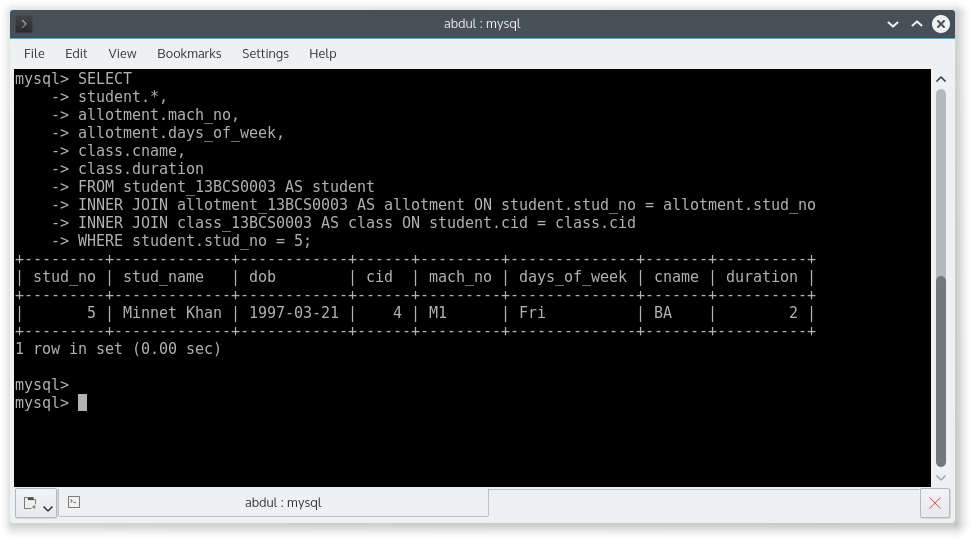
class.duration

FROM student\_13BCS0003 AS student

INNER JOIN allotment\_13BCS0003 AS allotment ON student.stud\_no = allotment.stud\_no

INNER JOIN class\_13BCS0003 AS class ON student.cid = class.cid

WHERE student.stud\_no = 5;



Q 6. Count for how many machines have been allocated lab\_no 11 for the day of the week as “Monday”.

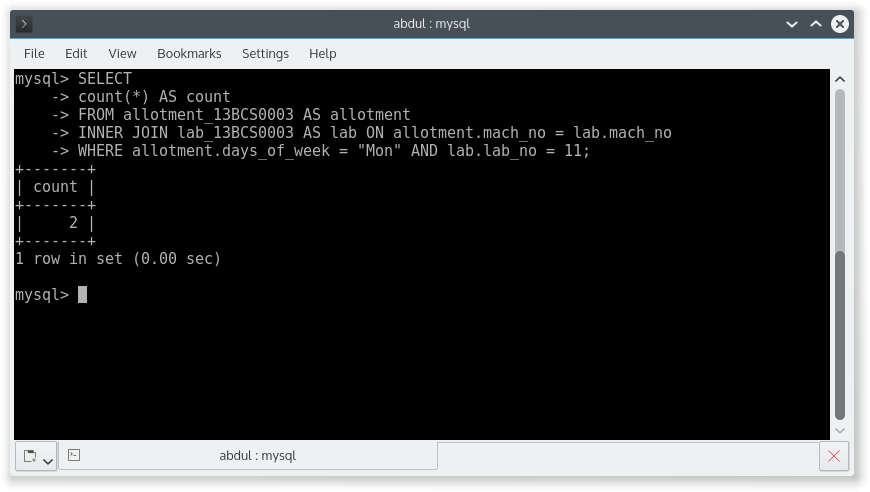
SELECT

count(\*) AS count

FROM allotment\_13BCS0003 AS allotment

INNER JOIN lab\_13BCS0003 AS lab ON allotment.mach\_no = lab.mach\_no

WHERE allotment.days\_of\_week = "Mon" AND lab.lab\_no = 11;



Q 7. How many students class wise have allocated machines in the labs.

SELECT

class.cid,

class.cname,

count(\*)

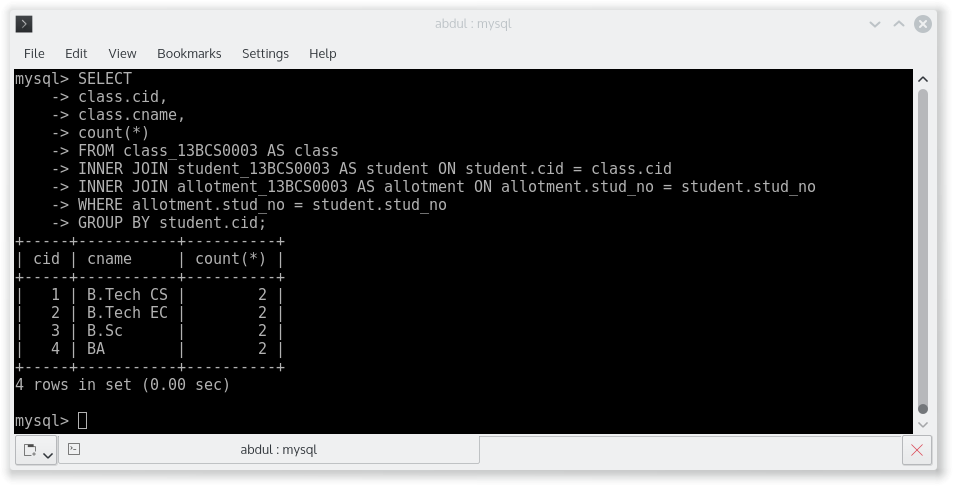
FROM class\_13BCS0003 AS class

INNER JOIN student\_13BCS0003 AS student ON student.cid = class.cid

INNER JOIN allotment\_13BCS0003 AS allotment ON allotment.stud\_no = student.stud\_no

WHERE allotment.stud\_no = student.stud\_no

GROUP BY student.cid;



Q 8. List the total number of machines allotted to students of particular lab.

SELECT

lab.lab\_no,

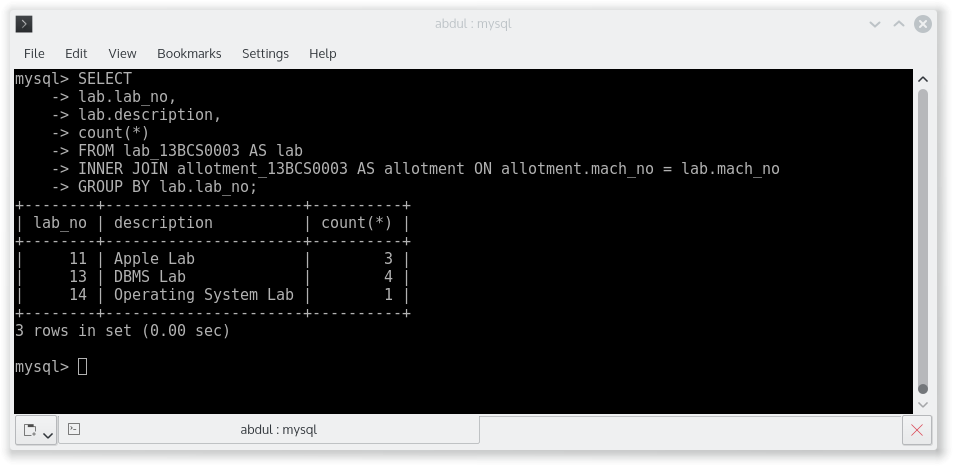
lab.description,

count(\*)

FROM lab\_13BCS0003 AS lab

INNER JOIN allotment\_13BCS0003 AS allotment ON allotment.mach\_no = lab.mach\_no

GROUP BY lab.lab\_no;



Q 9. Give the details (stuid, stuname, classname, machno, labno) of the youngest student.

SELECT

student.stud\_no,

student.stud\_name,

class.cname,

allotment.mach\_no,

lab.lab\_no

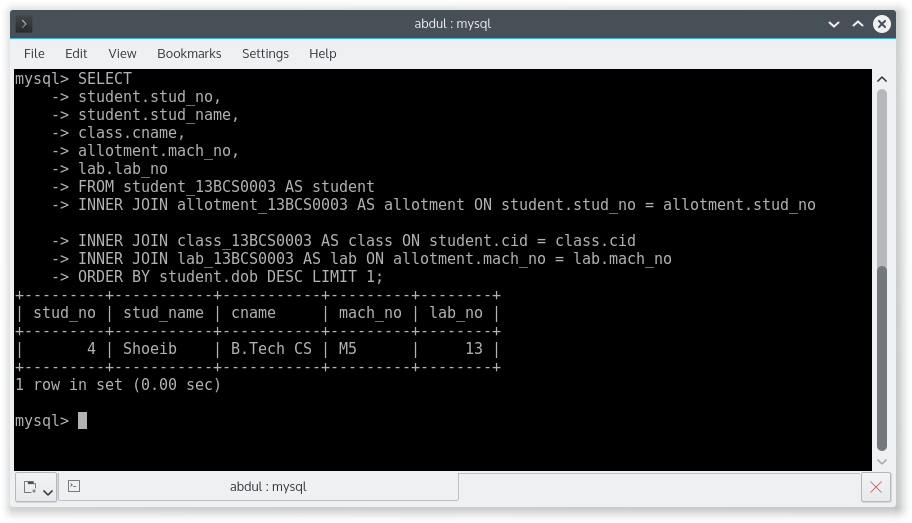
FROM student\_13BCS0003 AS student

INNER JOIN allotment\_13BCS0003 AS allotment ON student.stud\_no = allotment.stud\_no

INNER JOIN class\_13BCS0003 AS class ON student.cid = class.cid

INNER JOIN lab\_13BCS0003 AS lab ON allotment.mach\_no = lab.mach\_no

ORDER BY student.dob DESC LIMIT 1;



Q 10. Give the name of machine which is assigned to atleast two students.

SELECT

mach\_no

FROM allotment\_13BCS0003

GROUP BY mach\_no HAVING count(\*) >= 2;

CREATION OF TABLES.

CREATE TABLE product\_13BCS0003 (

pid varchar(20) PRIMARY KEY,

pname varchar(20) NOT NULL,

price int NOT NULL,

discount int,

pcompany varchar(20)

);

CREATE TABLE customer\_13BCS0003(

cid varchar(20) PRIMARY KEY,

cname varchar(20) NOT NULL,

cadd varchar(20),

ccity varchar(20)

);

CREATE TABLE orderdetail\_13BCS0003 (

oid varchar(20),

pid varchar(20),

cid varchar(20),

odate date,

qty int,

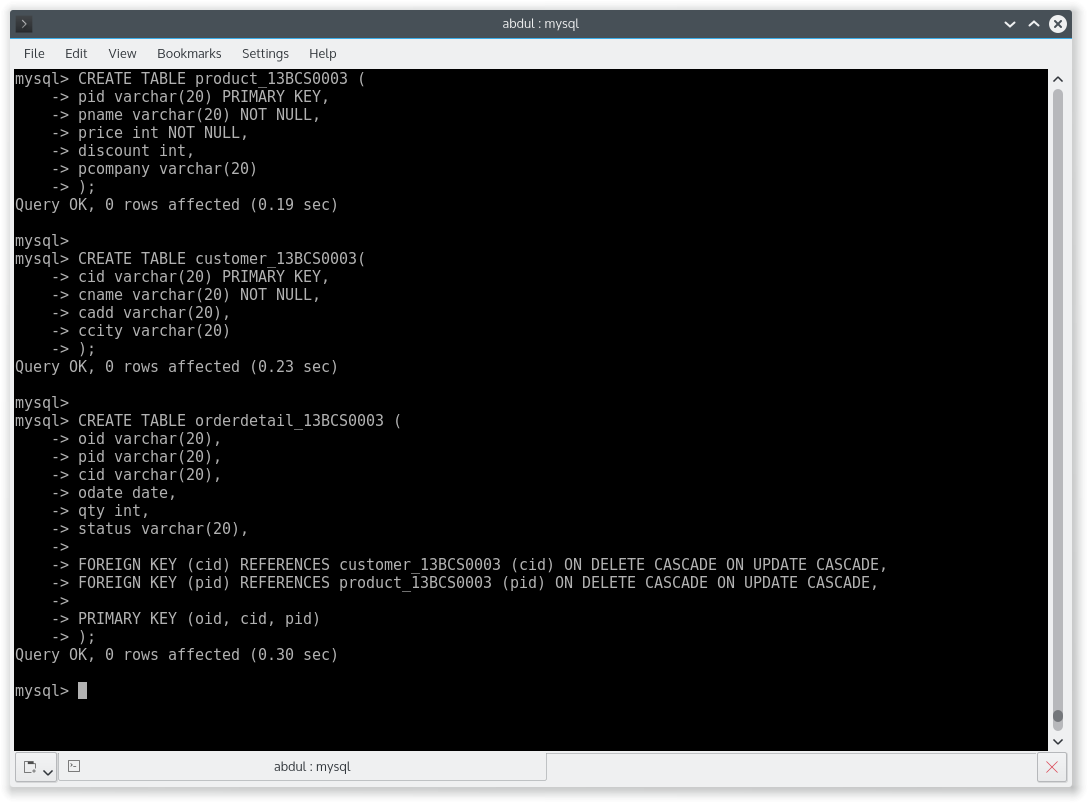
status varchar(20),

FOREIGN KEY (cid) REFERENCES customer\_13BCS0003 (cid) ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY (pid) REFERENCES product\_13BCS0003 (pid) ON DELETE CASCADE ON UPDATE CASCADE,

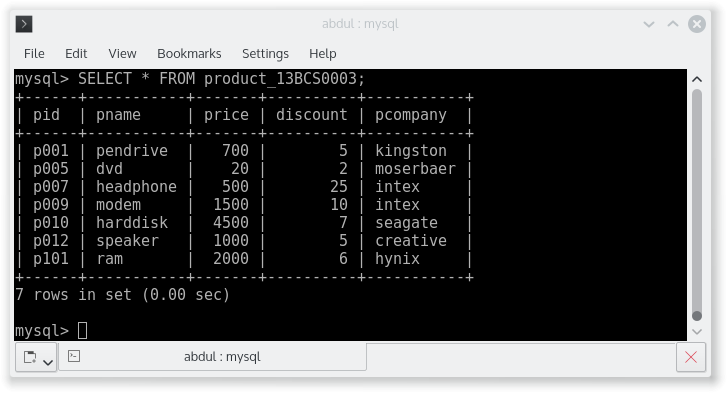
PRIMARY KEY (oid, cid, pid)

);



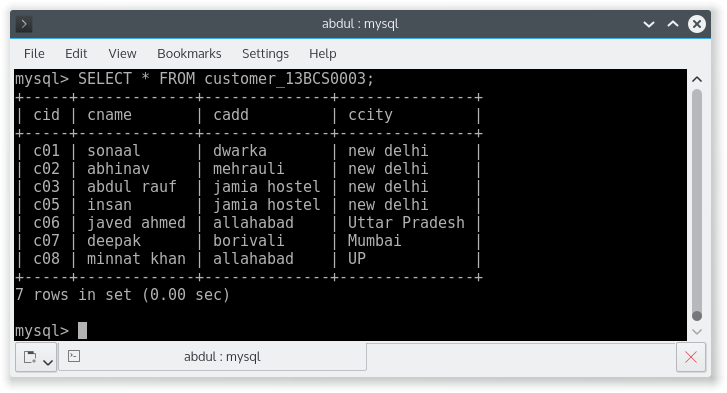
DATA IN TABLE product\_13BCS0003

SELECT \* FROM product\_13BCS0003;



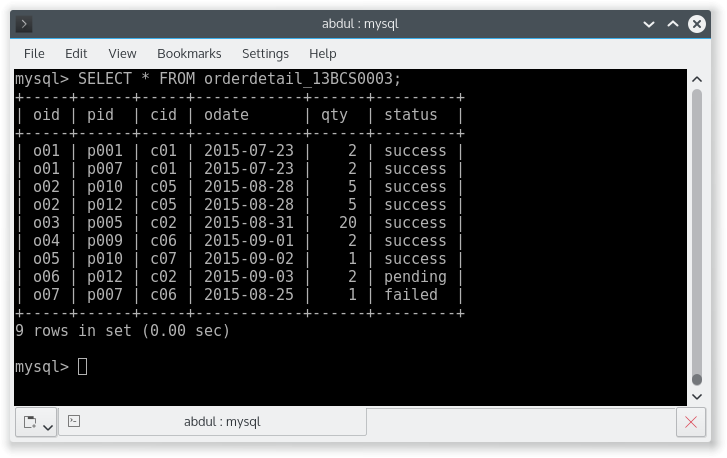
DATA IN TABLE customer\_13BCS0003

SELECT \* FROM customer\_13BCS0003;



DATA IN TABLE orderdetail\_13BCS0003

SELECT \* FROM orderdetail\_13BCS0003;



Q 2. Give the successful order details (cid,cname, pname) of particular customer.

SELECT customer.cid,

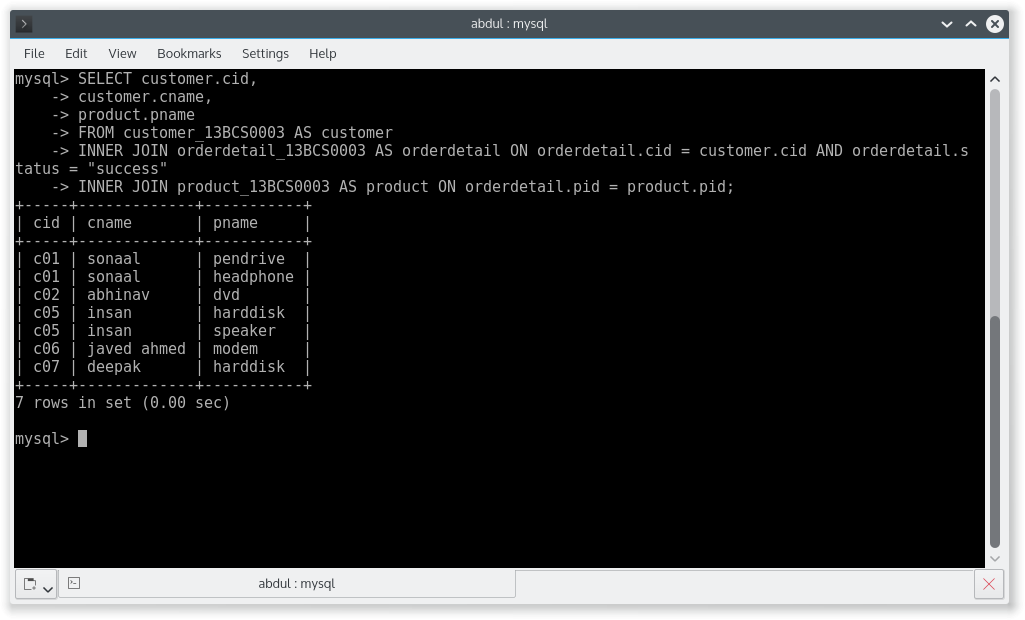
customer.cname,

product.pname

FROM customer\_13BCS0003 AS customer

INNER JOIN orderdetail\_13BCS0003 AS orderdetail ON orderdetail.cid = customer.cid AND orderdetail.status = "success"

INNER JOIN product\_13BCS0003 AS product ON orderdetail.pid = product.pid;



Q 3. Give the total discount given by shop in specified date.

SELECT

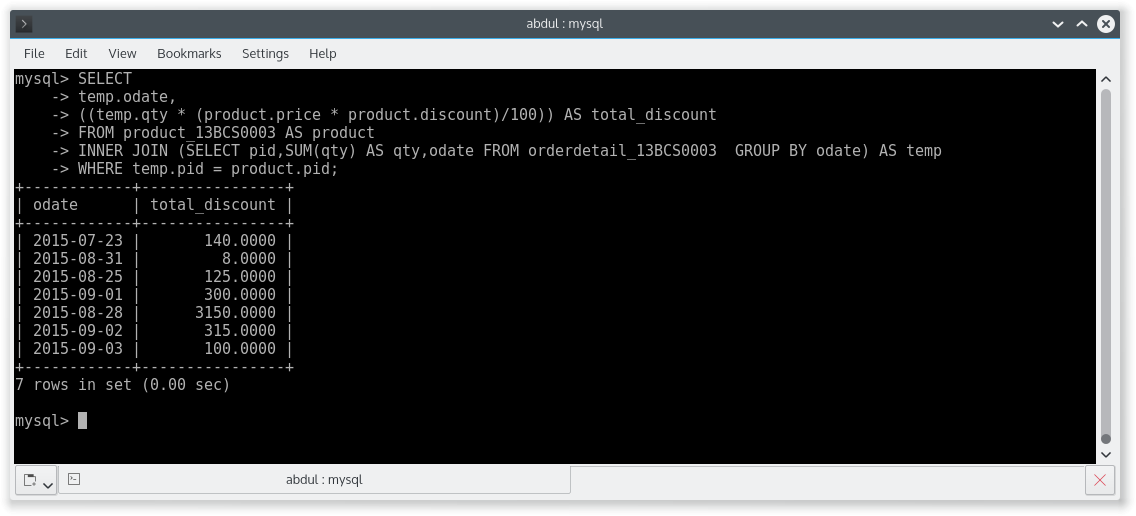
temp.odate,

((temp.qty \* (product.price \* product.discount)/100)) AS total\_discount

FROM product\_13BCS0003 AS product

INNER JOIN (SELECT pid,SUM(qty) AS qty,odate FROM orderdetail\_13BCS0003 GROUP BY odate) AS temp

WHERE temp.pid = product.pid;



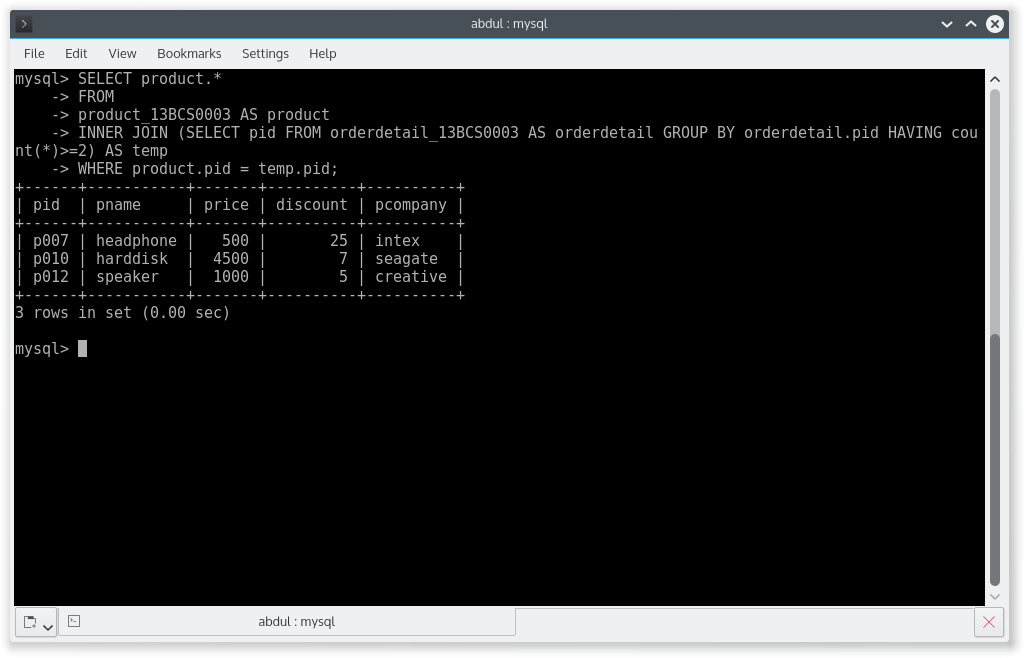
Q 4. List of all products which is purchased by atleast two customers.

SELECT product.\*

FROM product\_13BCS0003 AS product

INNER JOIN (SELECT pid FROM orderdetail\_13BCS0003 AS orderdetail GROUP BY orderdetail.pid HAVING count(\*)>=2) AS temp

WHERE product.pid = temp.pid;

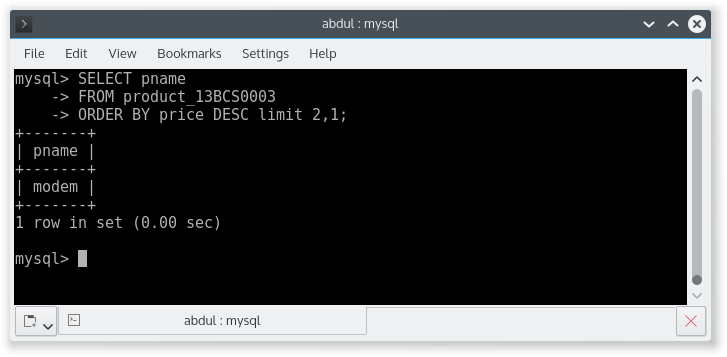


Q 5. List of all the product which have third highest price.

SELECT pname

FROM product\_13BCS0003

ORDER BY price DESC limit 2,1;



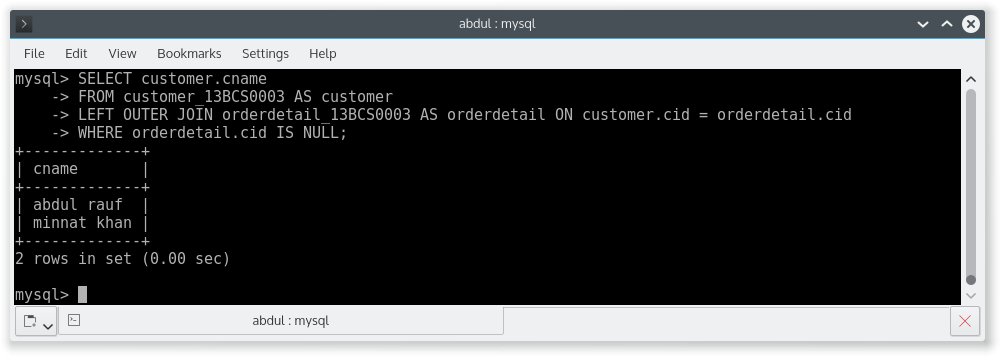
Q 6. List of customer who does not purchase any product.

SELECT customer.cname

FROM customer\_13BCS0003 AS customer

LEFT OUTER JOIN orderdetail\_13BCS0003 AS orderdetail ON customer.cid = orderdetail.cid

WHERE orderdetail.cid IS NULL;



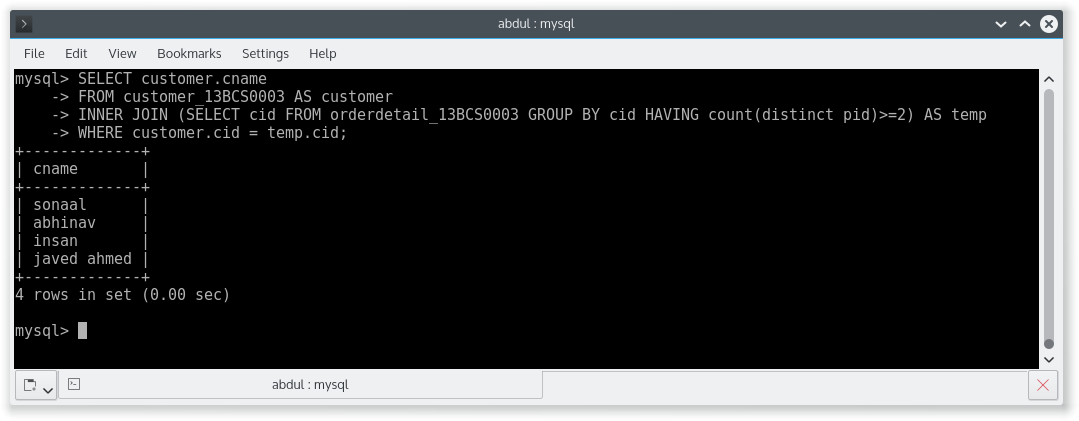
Q 7. List of customers who purchase atleast two distinct products.

SELECT customer.cname

FROM customer\_13BCS0003 AS customer

INNER JOIN (SELECT cid FROM orderdetail\_13BCS0003 GROUP BY cid HAVING count(distinct pid)>=2) AS temp

WHERE customer.cid = temp.cid;



Q 8. List of customer id and cname who purchase atleast two products.

SELECT

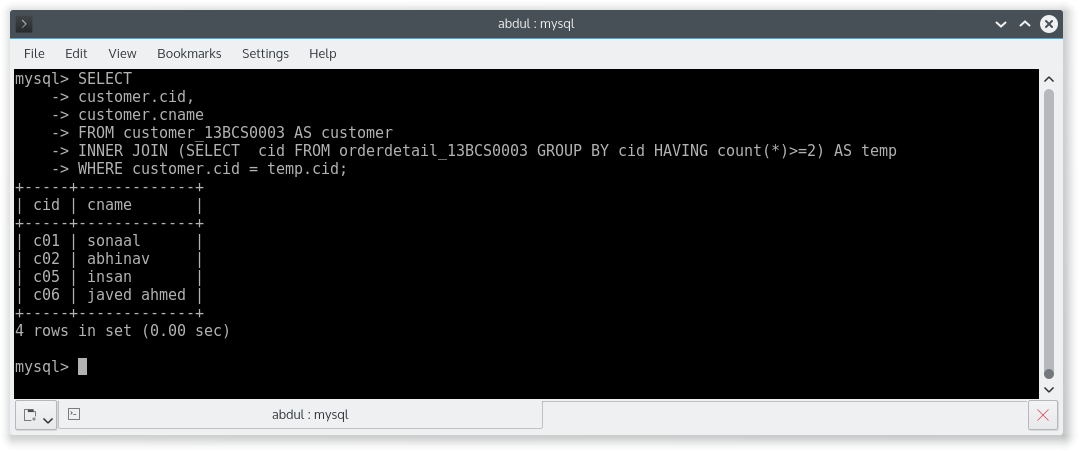
customer.cid,

customer.cname

FROM customer\_13BCS0003 AS customer

INNER JOIN (SELECT cid FROM orderdetail\_13BCS0003 GROUP BY cid HAVING count(\*)>=2) AS temp

WHERE customer.cid = temp.cid;



Q 9. List the total bill details with the quantity sold, price of the item and the final amount by customer.

SELECT

product.pname,

product.price,

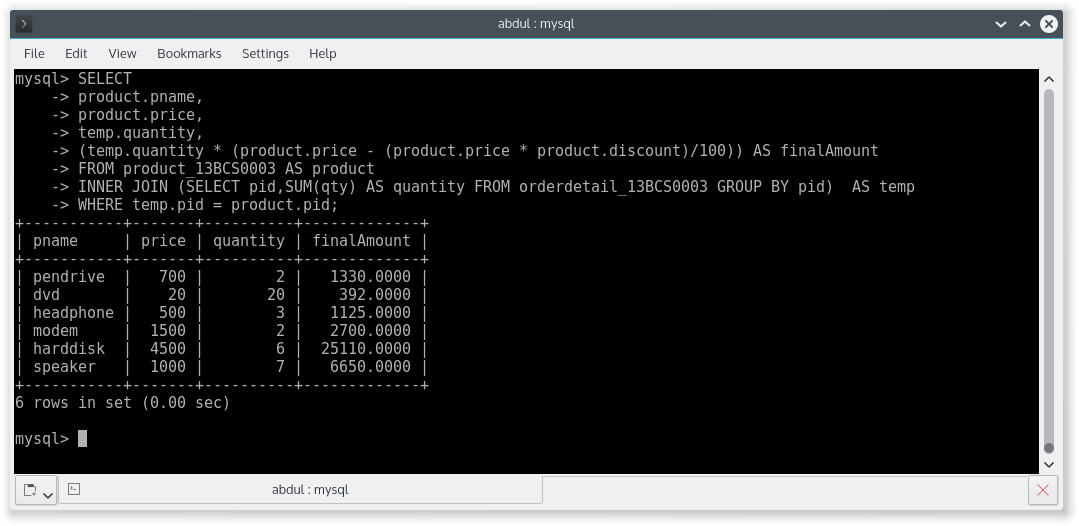
temp.quantity,

(temp.quantity \* (product.price - (product.price \* product.discount)/100)) AS finalAmount

FROM product\_13BCS0003 AS product

INNER JOIN (SELECT pid,SUM(qty) AS quantity FROM orderdetail\_13BCS0003 GROUP BY pid) AS temp

WHERE temp.pid = product.pid;



Q 10. Give a count of how many products have been bought by each customer. ( Give customer name and total product purchased )

SELECT

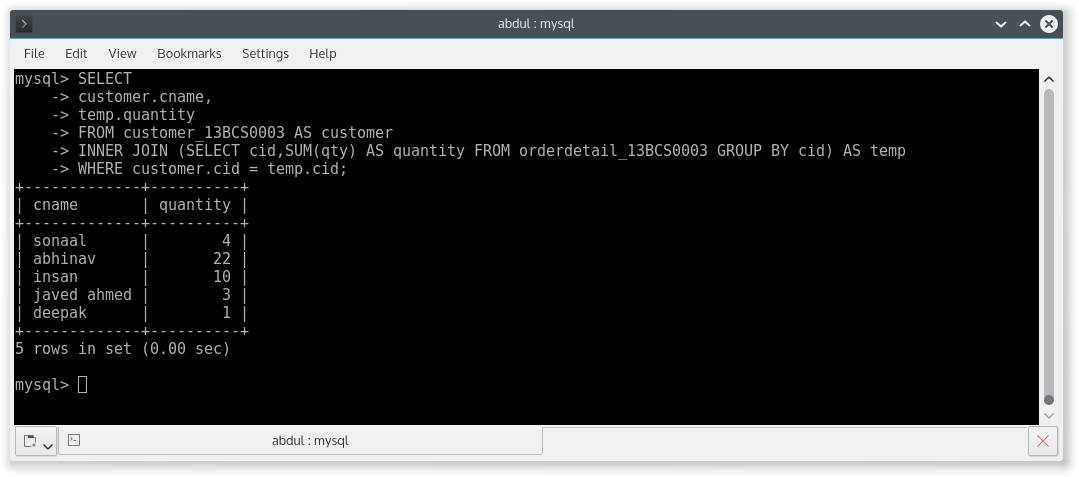
customer.cname,

temp.quantity

FROM customer\_13BCS0003 AS customer

INNER JOIN (SELECT cid,SUM(qty) AS quantity FROM orderdetail\_13BCS0003 GROUP BY cid) AS temp

WHERE customer.cid = temp.cid;



CREATION OF TABLES

CREATE TABLE book\_13BCS0003 (

bookid int PRIMARY KEY,

title varchar(30),

noofpages int,

copyright varchar(200)

);

CREATE TABLE author\_13BCS0003(

authid int PRIMARY KEY,

authFirst varchar(20),

authMid varchar(20),

authLast varchar(20),

authCity varchar(20),

age int

);

CREATE TABLE writtenby\_13BCS0003 (

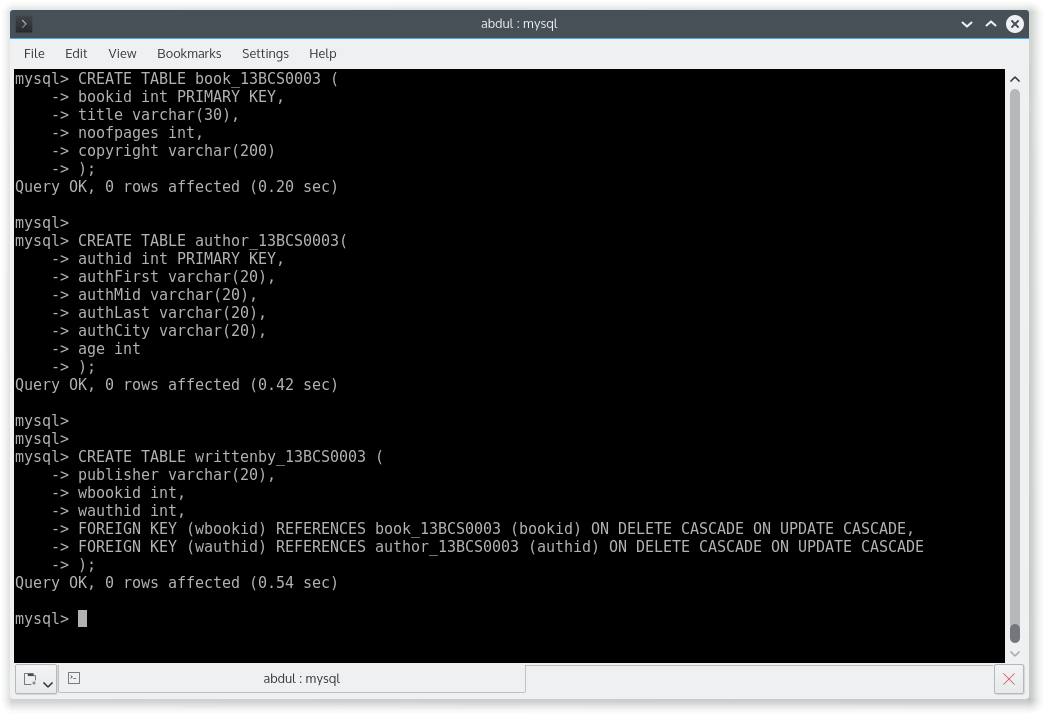
publisher varchar(20),

wbookid int,

wauthid int,

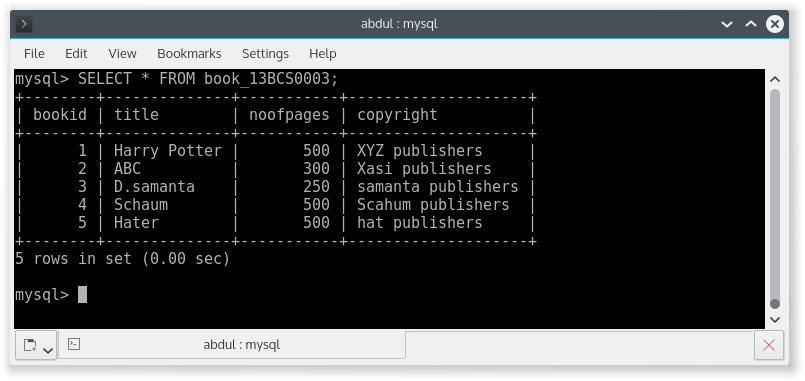
FOREIGN KEY (wbookid) REFERENCES book\_13BCS0003 (bookid) ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY (wauthid) REFERENCES author\_13BCS0003 (authid) ON DELETE CASCADE ON UPDATE CASCADE);



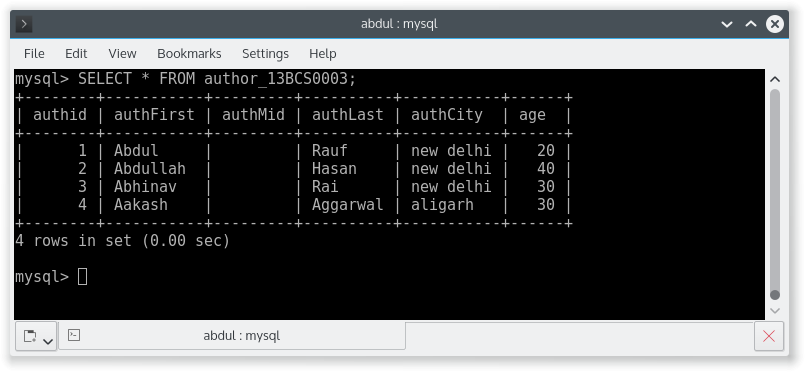
DATA IN TABLE BOOK\_13BCS0003

SELECT \* FROM book\_13BCS0003;



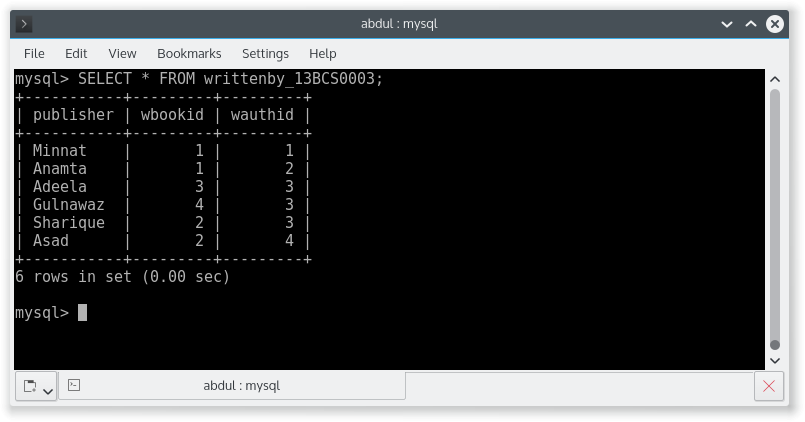
DATA IN TABLE AUTHOR\_13BCS0003

SELECT \* FROM author\_13BCS0003;



DATA IN TABLE WRITTENBY\_13BCS0003

SELECT \* FROM writtenby\_13BCS0003;



Q 1. Give the title, author name, publisher name for author whose city contains total no. of a = 2?

SELECT

author.authfirst,

book.title,

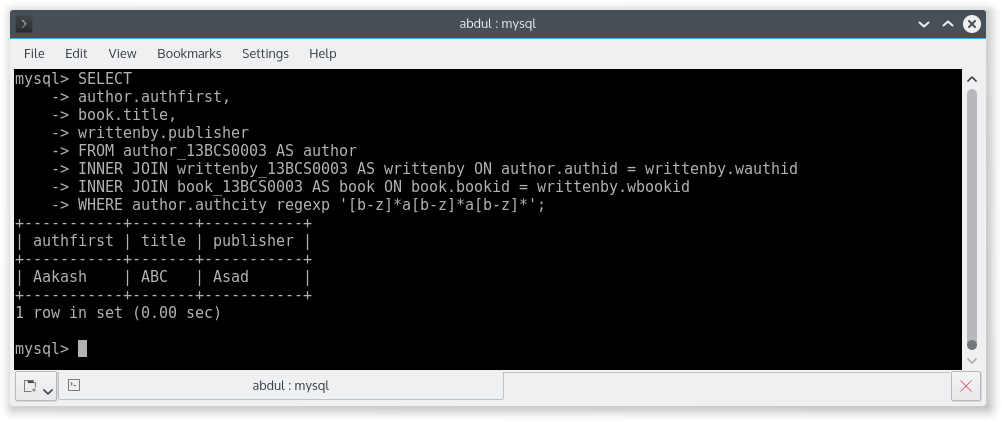
writtenby.publisher

FROM author\_13BCS0003 AS author

INNER JOIN writtenby\_13BCS0003 AS writtenby ON author.authid = writtenby.wauthid

INNER JOIN book\_13BCS0003 AS book ON book.bookid = writtenby.wbookid

WHERE author.authcity regexp '[b-z]\*a[b-z]\*a[b-z]\*';



Q 2. Give the details of the book which is written by atleast two authors.

SELECT \*

FROM book\_13BCS0003 as book

INNER JOIN (SELECT wbookid FROM writtenby\_13BCS0003 GROUP BY wbookid HAVING COUNT(wbookid)>=2) AS temp

WHERE book.bookid = temp.wbookid;



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

CREATE PROCEDURE insertIntoAuth(IN id int, IN first varchar(20),IN mid varchar(20),IN last varchar(20),IN city varchar(20), IN age int)

BEGIN

DECLARE t int;

SELECT count(\*) into t FROM author\_13BCS0003 WHERE authid = id;

IF t>0 THEN

SELECT 'id already exists';

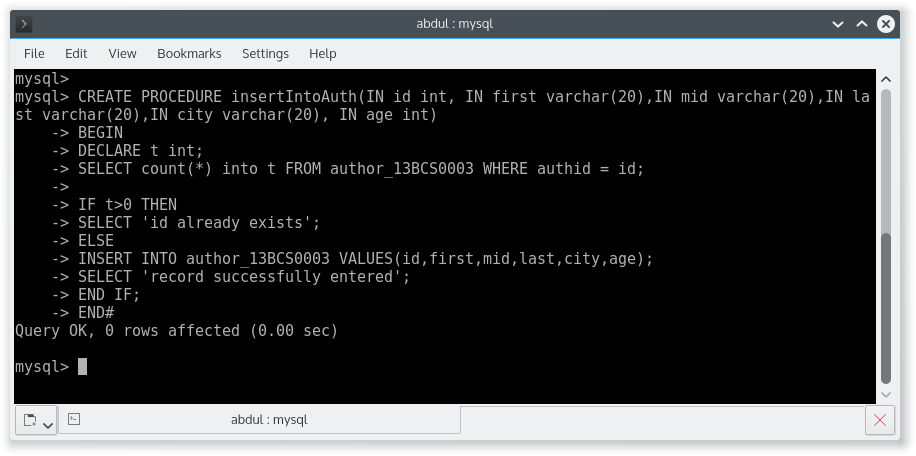
ELSE

INSERT INTO author\_13BCS0003 VALUES(id,first,mid,last,city,age);

SELECT 'record successfully entered';

END IF;

END#



Q 4. Write a stored procedure ( SP Name: InsertBookInfo ) to insert the book information.

CREATE PROCEDURE insertBook(IN id int, IN title varchar(20),IN pages int, IN copyright varchar(20))

BEGIN

DECLARE t int;

SELECT count(\*) into t FROM book\_13BCS0003 WHERE bookid = id;

IF t>0 THEN

SELECT 'id already exists';

ELSE

INSERT into book\_13BCS0003 VALUES(id,title,pages,copyright);

SELECT 'booksuccessfully added';

END IF;

END#

CREATE PROCEDURE insertWBy(IN publisher varchar(20), IN bookid int,IN authorid int)

BEGIN

INSERT into writtenby\_13BCS0003 VALUES(publisher,bookid,authorid);

SELECT 'writtenby entry successfully added';

END#

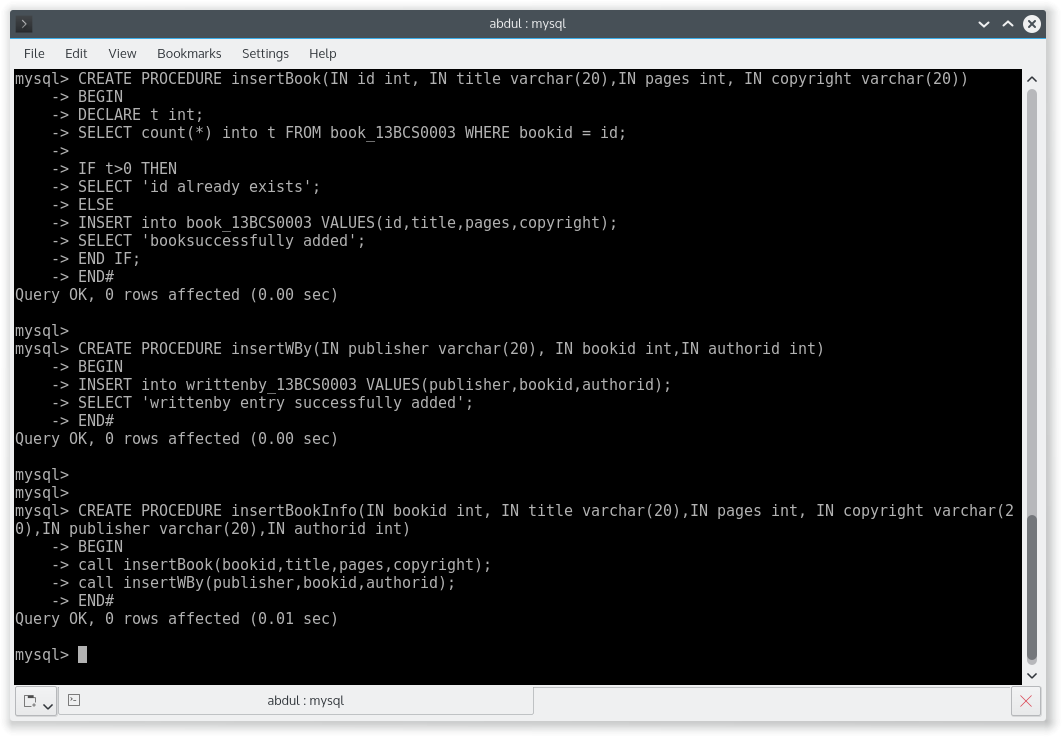
CREATE PROCEDURE insertBookInfo(IN bookid int, IN title varchar(20),IN pages int, IN copyright varchar(20),IN publisher varchar(20),IN authorid int)

BEGIN

call insertBook(bookid,title,pages,copyright);

call insertWBy(publisher,bookid,authorid);

END#



Q 5. Write a stored procedure to delete the author information using its AuthID.

CREATE PROCEDURE deleteAuthorById(IN id int)

BEGIN

DECLARE t int;

DECLARE u int;

DECLARE str varchar(255);

SELECT count(\*) INTO t FROM writtenby\_13BCS0003 WHERE writtenby\_13BCS0003.wauthid = id ;

SELECT count(\*) INTO u FROM author\_13BCS0003 WHERE author\_13BCS0003.authid = id ;

IF u = 0 THEN

SELECT 'Author not present';

ELSE

IF t>0 THEN

SET STR = concat('You cannot delete author because total ',t,' books exists IN the books table.First delete all the books written by the author. ' );

SELECT str;

ELSE

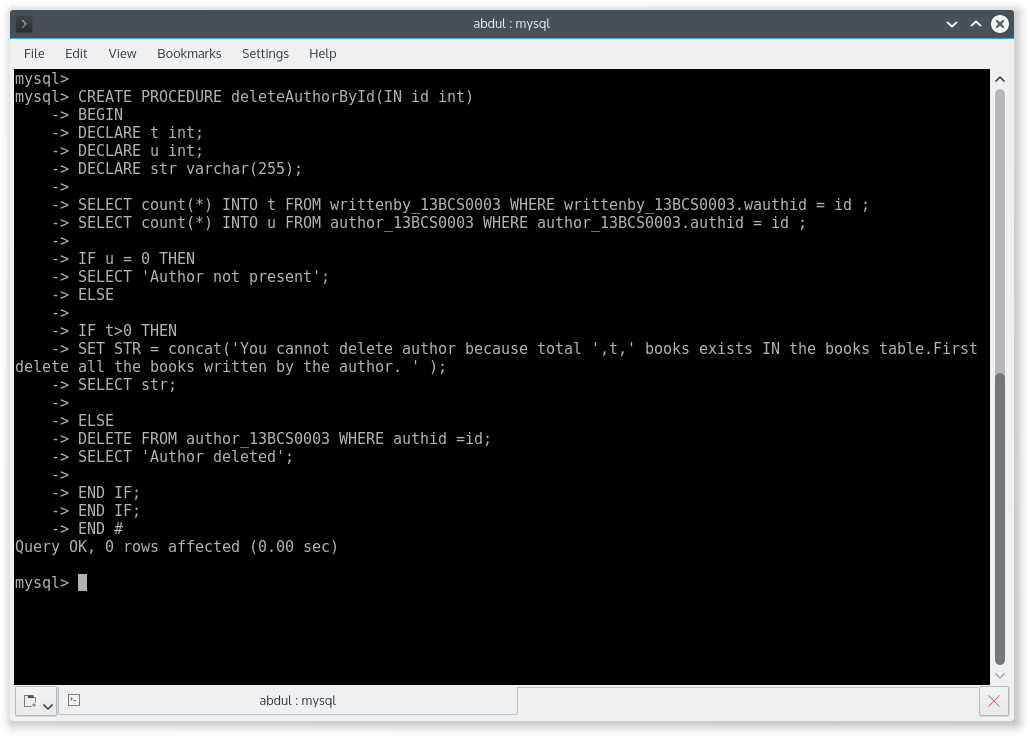
DELETE FROM author\_13BCS0003 WHERE authid =id;

SELECT 'Author deleted';

END IF;

END IF;

END #



Q 6. Write a stored procedure to delete the book using AuthID.

CREATE PROCEDURE deleteBookByAuthorID(IN id int)

BEGIN

DECLARE t int;

SELECT count(\*) INTO t FROM author\_13BCS0003 WHERE authid = id;

IF t = 0 THEN

SELECT 'Author does not Exist';

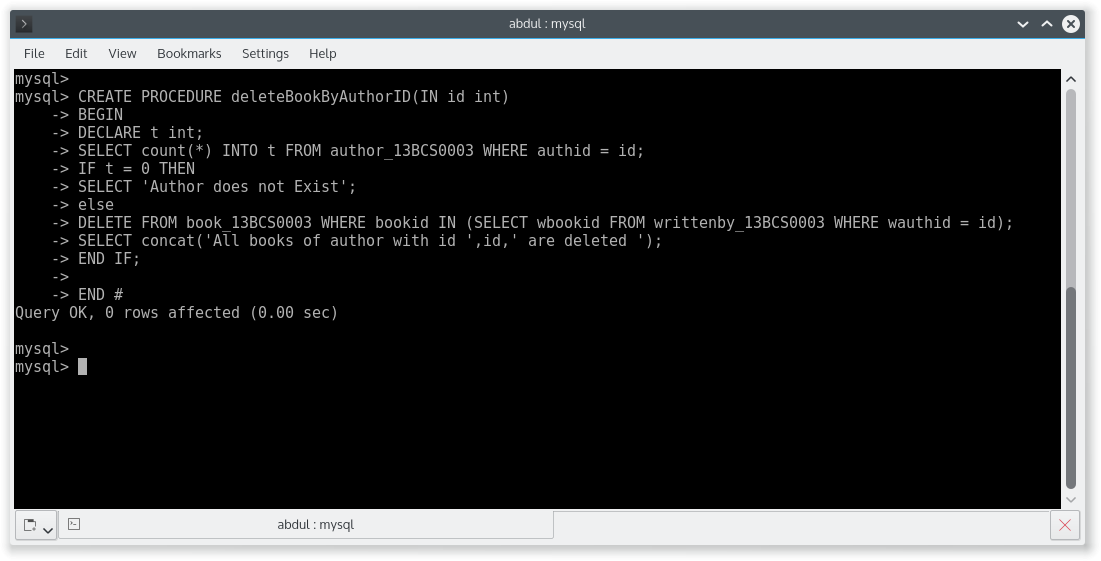
else

DELETE FROM book\_13BCS0003 WHERE bookid IN (SELECT wbookid FROM writtenby\_13BCS0003 WHERE wauthid = id);

SELECT concat('All books of author with id ',id,' are deleted ');

END IF;

END #



CREATION OF TABLES

CREATE TABLE faculty\_13BCS0003(

fid int PRIMARY KEY,

fname varchar(20),

deptid int

);

CREATE TABLE class\_13BCS0003(

name varchar(20) PRIMARY KEY,

meets\_at varchar(20),

room varchar(20),

fid int,

FOREIGN KEY (fid) REFERENCES faculty\_13BCS0003 (fid) ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE student\_13BCS0003(

snum int PRIMARY KEY,

sname varchar(20),

major varchar(20),

level varchar(20),

age int

);

CREATE TABLE enrolled\_13BCS0003 (

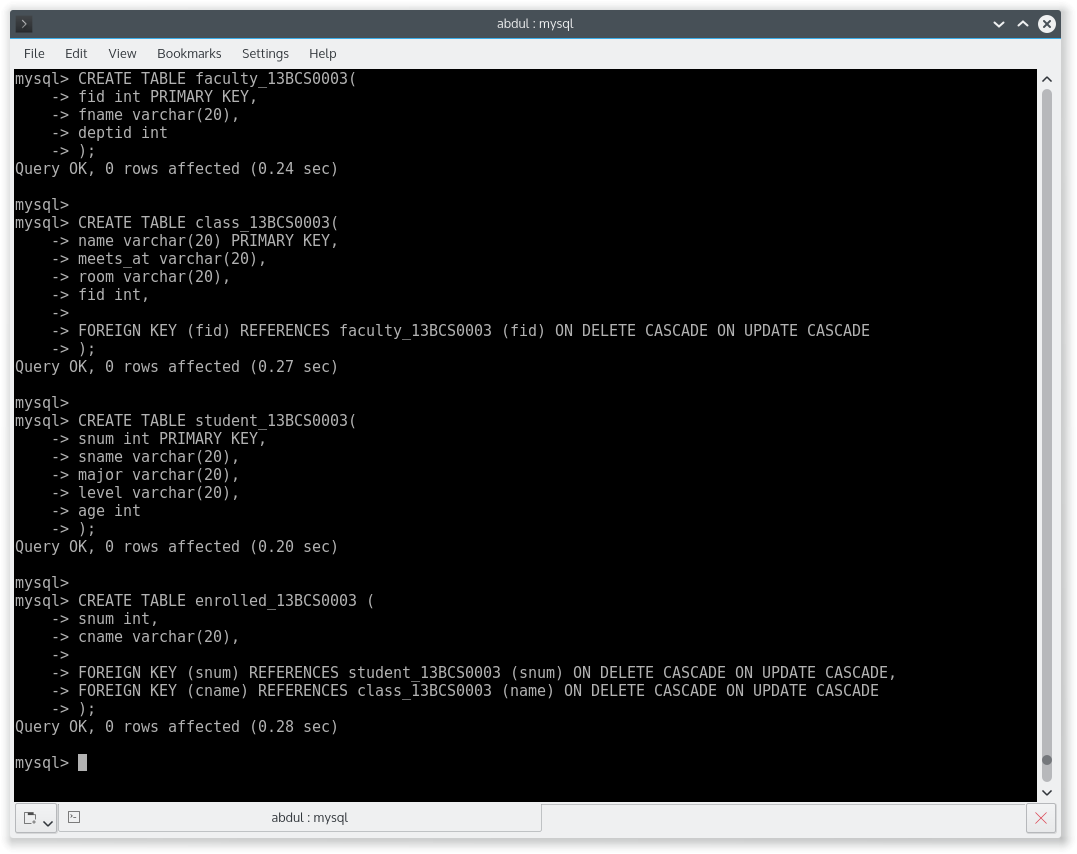
snum int,

cname varchar(20),

FOREIGN KEY (snum) REFERENCES student\_13BCS0003 (snum) ON DELETE CASCADE ON UPDATE CASCADE,

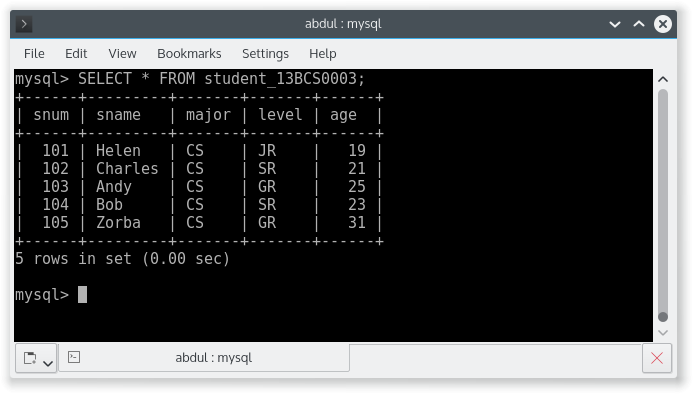
FOREIGN KEY (cname) REFERENCES class\_13BCS0003 (name) ON DELETE CASCADE ON UPDATE CASCADE

);

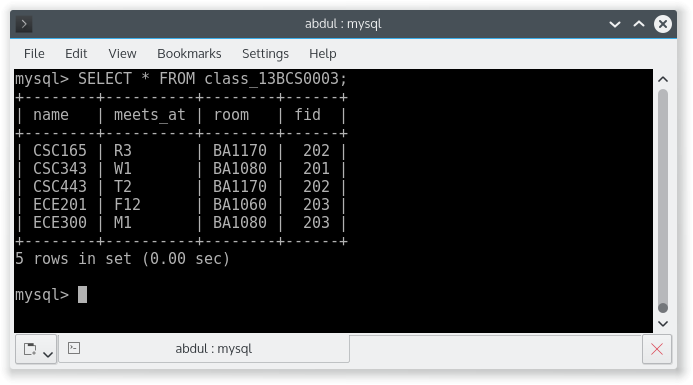


DATA IN TABLE student\_13BCS0003

SELECT \* FROM student\_13BCS0003;



DATA IN TABLE class\_13BCS0003

SELECT \* FROM class\_13BCS0003;

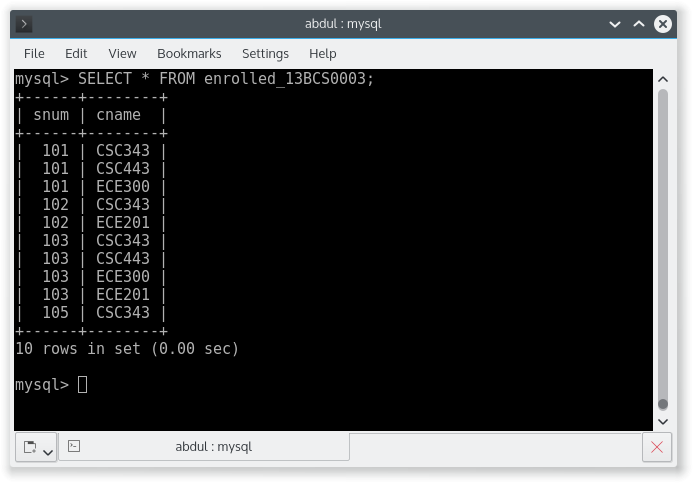
DATA IN TABLE faculty\_13BCS0003

SELECT \* FROM faculty\_13BCS0003;



DATA IN TABLE enrolled\_13BCS0003

SELECT \* FROM enrolled\_13BCS0003;



Q 1. Find the names of the juniors (level = JR) who are enrolled in class taught by X.XXXX

SELECT

sname

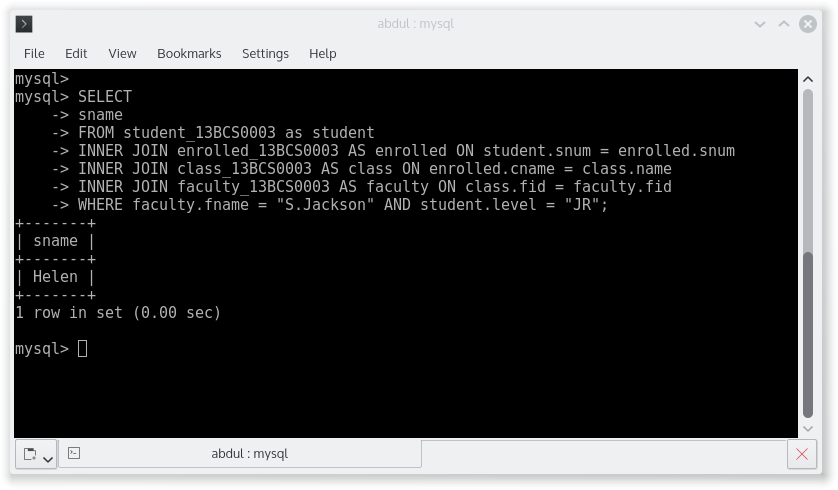
FROM student\_13BCS0003 AS student

INNER JOIN enrolled\_13BCS0003 AS enrolled ON student.snum = enrolled.snum

INNER JOIN class\_13BCS0003 AS class ON enrolled.cname = class.name

INNER JOIN faculty\_13BCS0003 AS faculty ON class.fid = faculty.fid

WHERE faculty.fname = "S.Jackson" AND student.level = "JR";



Q 2. Find the age of the oldest student who is either a CS major or enrolled in a course taught by X.XXXX

SELECT

MAX(age)

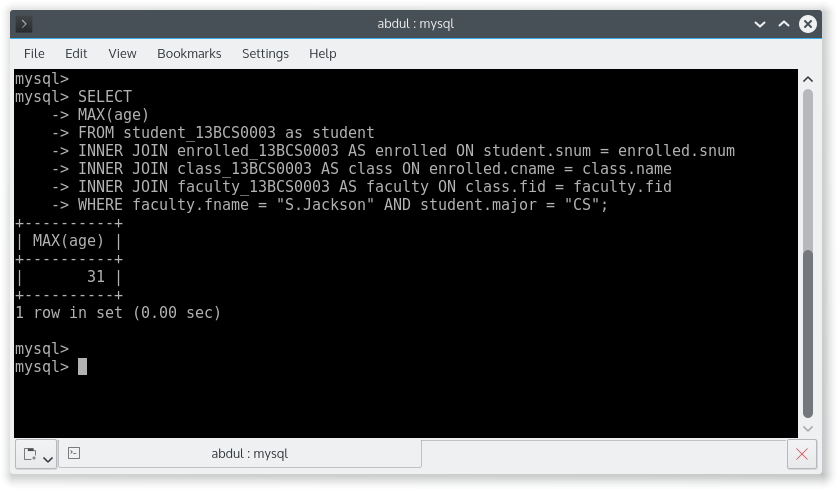
FROM student\_13BCS0003 as student

INNER JOIN enrolled\_13BCS0003 AS enrolled ON student.snum = enrolled.snum

INNER JOIN class\_13BCS0003 AS class ON enrolled.cname = class.name

INNER JOIN faculty\_13BCS0003 AS faculty ON class.fid = faculty.fid

WHERE faculty.fname = "S.Jackson" AND student.major = "CS";



Q 3. Find the names of all classes that either meet in room W1 or have five or more students enrolled.

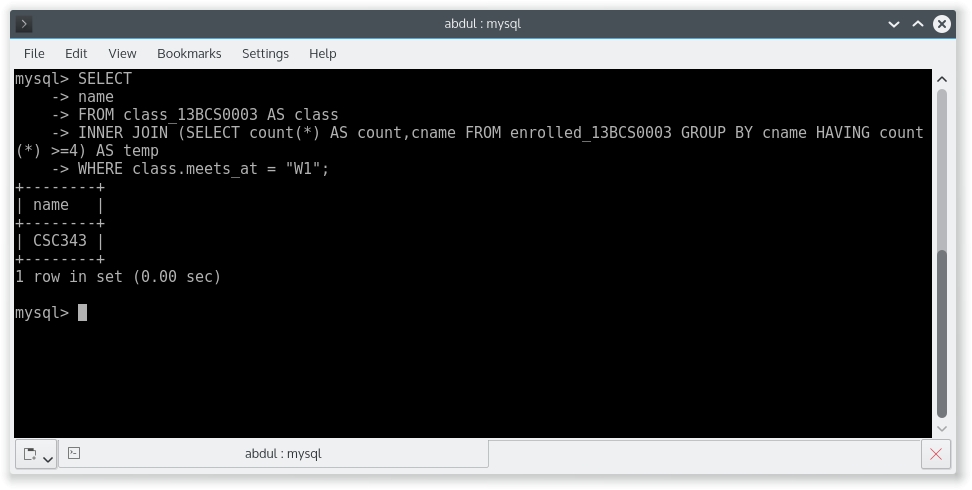
SELECT

name

FROM class\_13BCS0003 AS class

INNER JOIN (SELECT count(\*) AS count,cname FROM enrolled\_13BCS0003 GROUP BY cname HAVING count(\*) >=4) AS temp

WHERE class.meets\_at = "W1";



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

SELECT

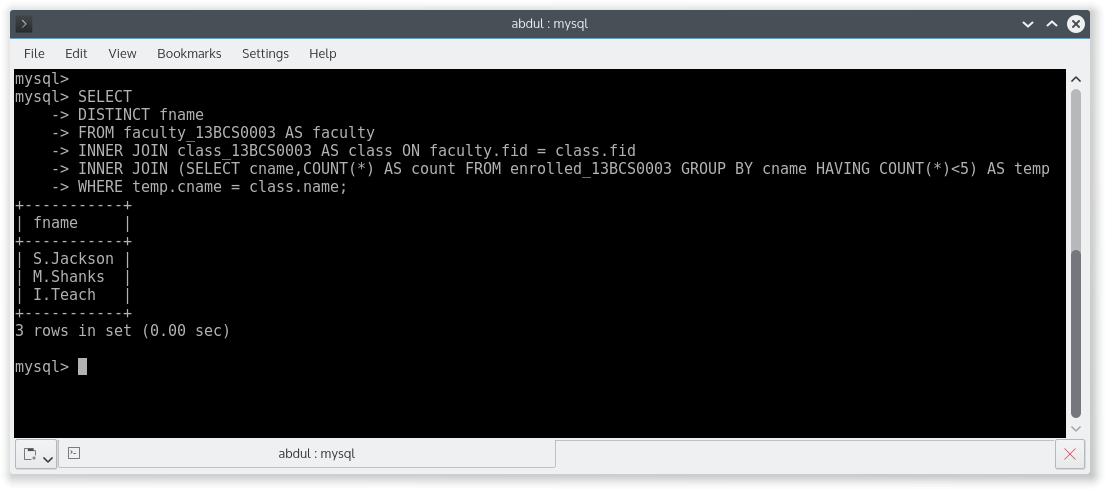
DISTINCT fname

FROM faculty\_13BCS0003 AS faculty

INNER JOIN class\_13BCS0003 AS class ON faculty.fid = class.fid

INNER JOIN (SELECT cname,COUNT(\*) AS count FROM enrolled\_13BCS0003 GROUP BY cname HAVING COUNT(\*)<5) AS temp

WHERE temp.cname = class.name;



Q 7. For each level, print the level and the average of students for that level.

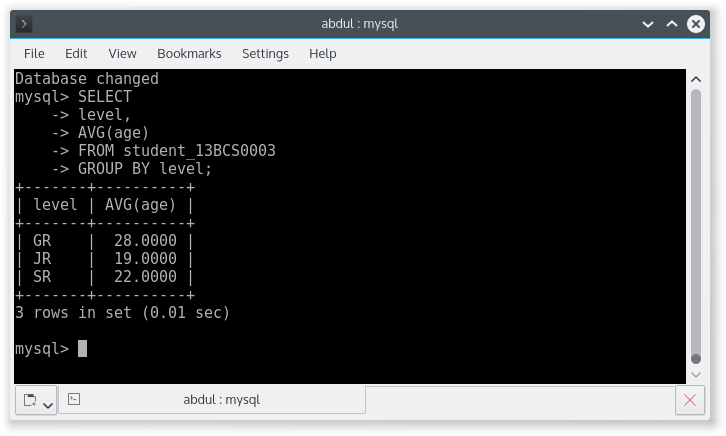
SELECT

level,

AVG(age)

FROM student\_13BCS0003

GROUP BY level;



Q 8. For each level except JR, print the level and the average age of students of that level.

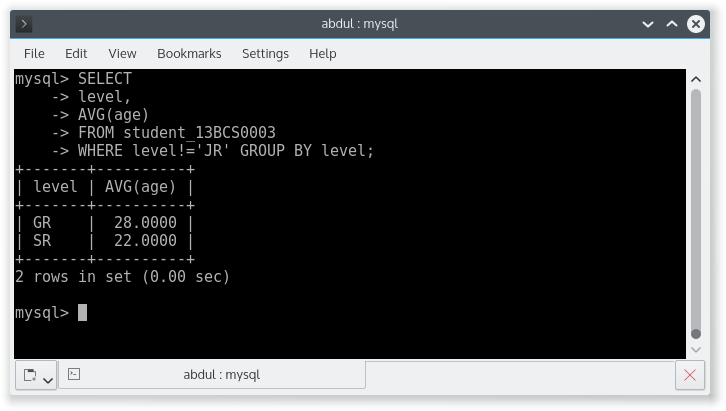
SELECT

level,

AVG(age)

FROM student\_13BCS0003

WHERE level!='JR' GROUP BY level;



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

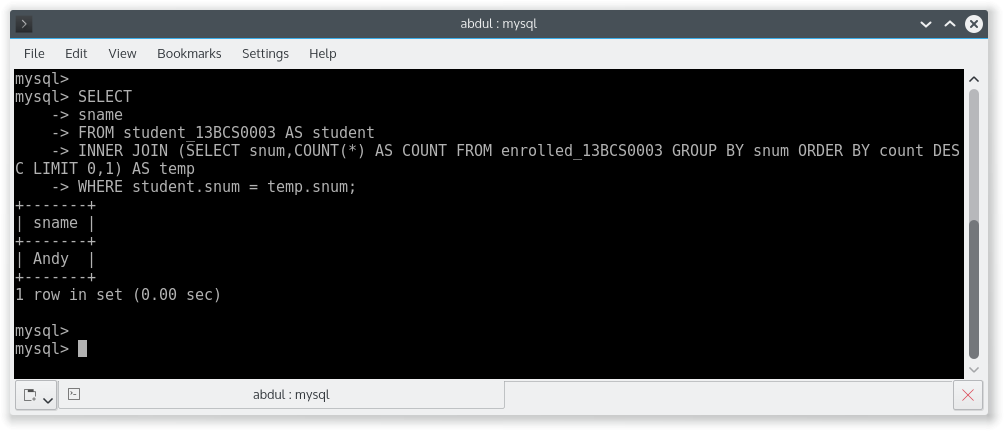
SELECT

sname

FROM student\_13BCS0003 AS student

INNER JOIN (SELECT snum,COUNT(\*) AS COUNT FROM enrolled\_13BCS0003 GROUP BY snum ORDER BY count DESC LIMIT 0,1) AS temp

WHERE student.snum = temp.snum;



FUNCTION TO CALCULATE THE AVERAGE OF THREE NUMBERS.

CREATE FUNCTION findAvgFloat(a1 int,a2 int,a3 int) RETURNS float

BEGIN

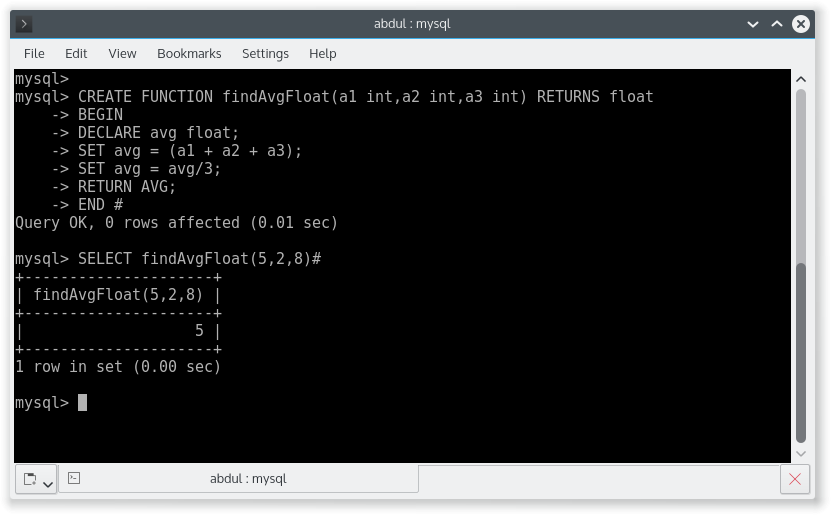
DECLARE avg float;

SET avg = (a1 + a2 + a3);

SET avg = avg/3;

RETURN AVG;

END #



FUNCTION TO CALCULATE FACTORIAL OF A NUMBER.

CREATE FUNCTION factorial (n int) RETURNS int

BEGIN

DECLARE i int DEFAULT 1;

DECLARE fact int DEFAULT 1;

myloop : LOOP

IF i>n THEN

leave myloop;

ELSE

SET fact = fact\*i;

SET i = i+1;

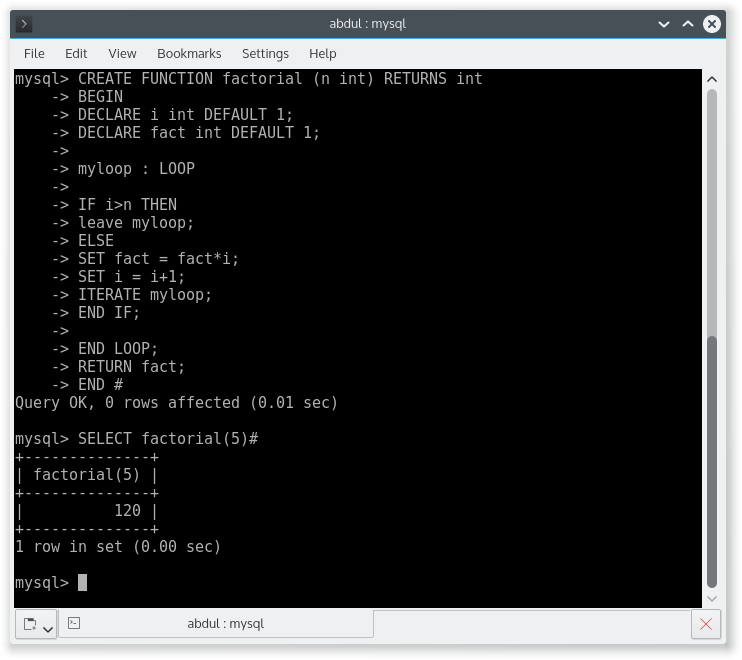
ITERATE myloop;

END IF;

END LOOP;

RETURN fact;

END #



PROCEDURE TO CALCULATE FACTORIAL OF A NUMBER.

CREATE PROCEDURE factorial (IN n int, OUT fact int)

BEGIN

DECLARE i int DEFAULT 1;

SET fact = 1;

myloop : LOOP

IF i>n THEN

leave myloop;

ELSE

SET fact = fact\*i;

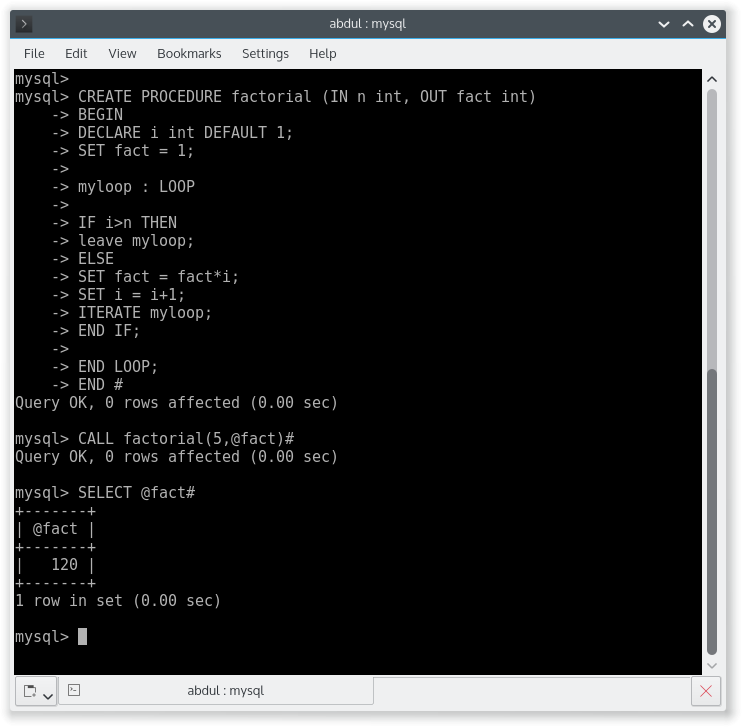
SET i = i+1;

ITERATE myloop;

END IF;

END LOOP;

END #



FUNCTION TO CALCULATE SUM OF FIBONACCI NUMBERS.

CREATE FUNCTION fibonacci (n int) RETURNS varchar(250)

BEGIN

DECLARE i int DEFAULT 2;

DECLARE a int DEFAULT 0;

DECLARE b int DEFAULT 1;

DECLARE c int DEFAULT 0;

DECLARE sum int DEFAULT 0;

DECLARE series varchar (250) DEFAULT "0, ";

myloop : LOOP

IF i>n THEN

leave myloop;

ELSE

SET c = a + b;

SET series = CONCAT(series,c,", ");

SET a = b;

SET b = c;

SET sum = sum + c;

SET i = i+1;

ITERATE myloop;

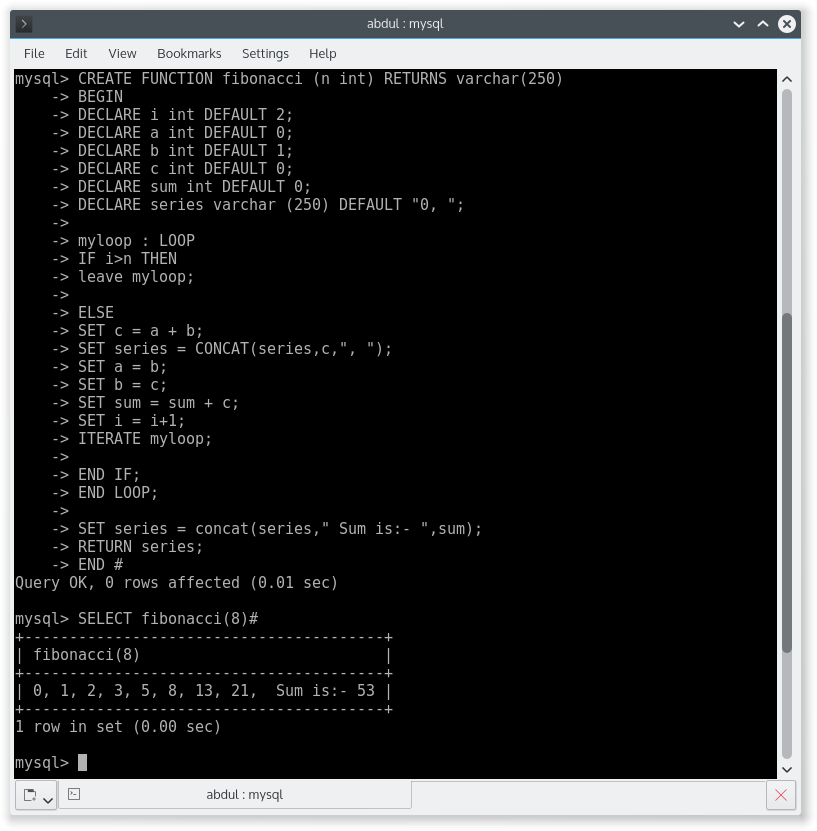
END IF;

END LOOP;

SET series = concat(series," Sum is:- ",sum);

RETURN series;

END #



PROCEDURE TO CALCULATE SUM OF FIBONACCI NUMBERS.

CREATE PROCEDURE fibonacci (IN n int, OUT series varchar(250))

BEGIN DECLARE i int DEFAULT 2;

DECLARE a int DEFAULT 0;

DECLARE b int DEFAULT 1;

DECLARE c int DEFAULT 0;

DECLARE sum int DEFAULT 0;

SET series = "0, ";

myloop : LOOP

IF i>n THEN

leave myloop;

ELSE

SET c = a+b;

SET series = CONCAT(series,c,", ");

SET a = b;

SET b = c;

SET sum = sum + c;

SET i = i+1;

ITERATE myloop;

END IF;

END LOOP;

SET series = CONCAT(series," Sum is:- ",sum);

END #

