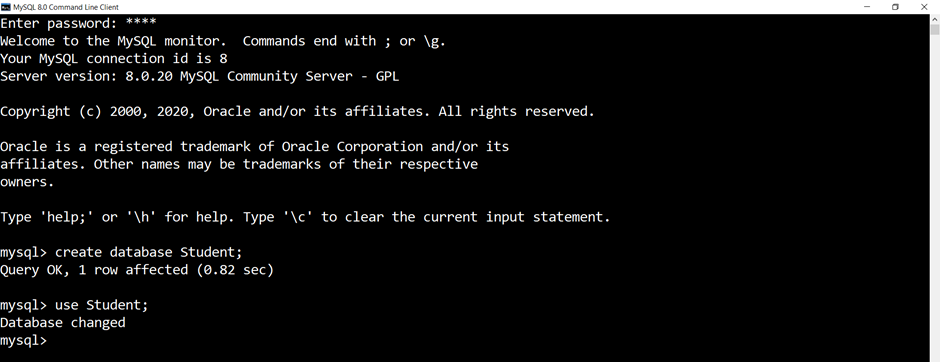
**Prior Instructions**

* **Please do read all the questions before performing any operations in the database**
* **Once you have fully gone through the questions then likewise decide the contents and table columns and follow the below instructions**

1. Create Student Database

**Solution:-**

create database Student;

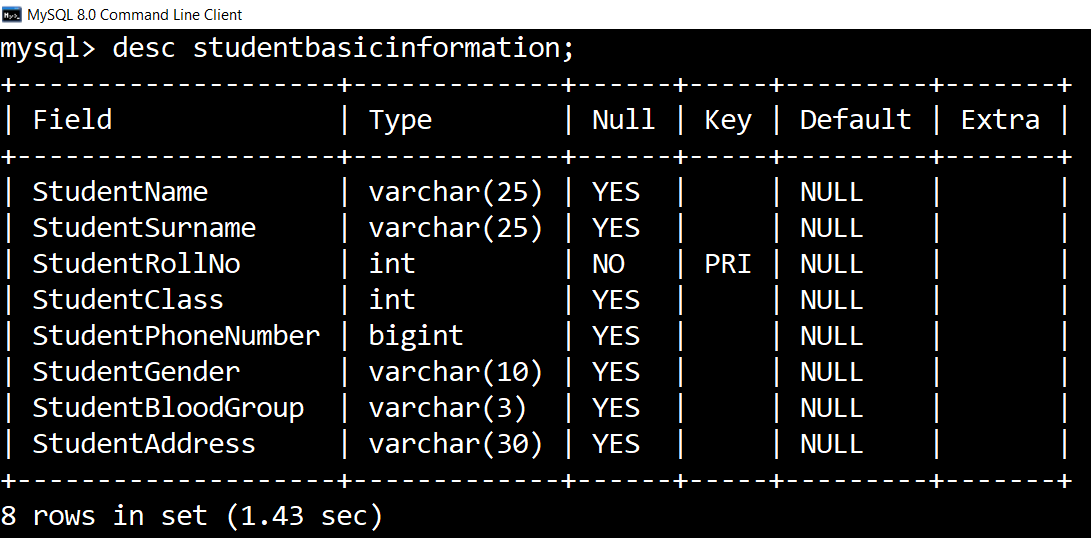


1. Create the following table under the Student Database:
   1. StudentBasicInformation
      1. Columns
         1. StudentName
         2. StudentSurname
         3. StudentRollNo
         4. StudentAddress
         5. Add more three basic columns of the name of your own
   2. StudentAdmissionPaymentDetails
      1. Columns
         1. StudentRollNo
         2. AmountPaid
         3. AmountBalance
         4. Add more four basic columns of the name of your own
   3. StudentSubjectInformation
      1. Columns
         1. SubjectOpted
         2. StudentRollNo
         3. SubjectTotalMarks
         4. SubjectObtainedMarks
         5. StudentMarksPercentage
         6. Add more one columns of the name of your own
   4. SubjectScholarshipInformation
      1. Columns
         1. StudentRollNo
         2. ScholarshipName
         3. ScholarshipDescription
         4. ScholarshipAmount
         5. ScholarshipCategory
         6. Add more two columns of the name of your own

**Solution:-**

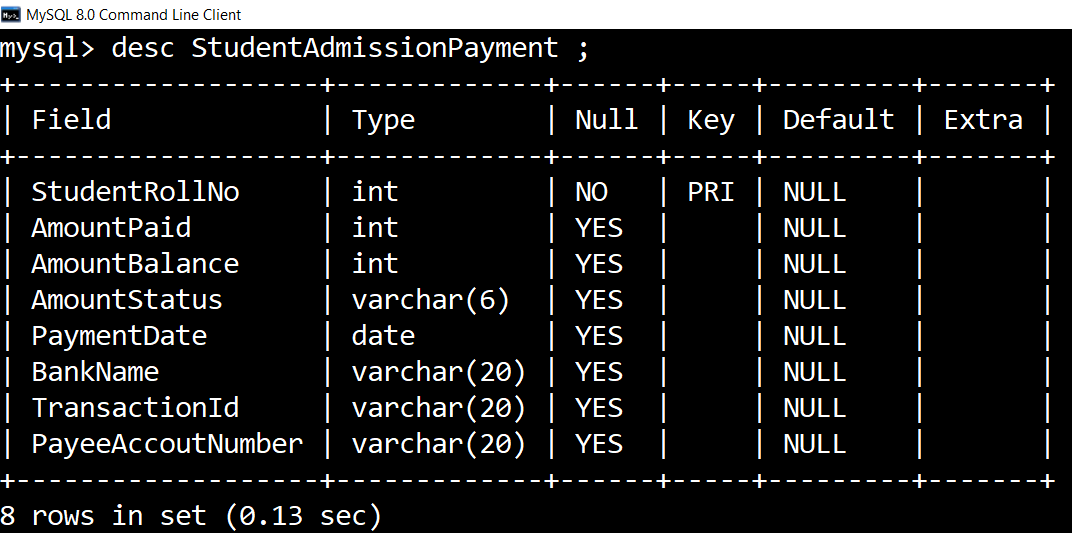
a)

create table StudentBasicInformation( StudentName varchar(25), StudentSurname varchar(25), StudentRollNo int primary key,StudentClass int,StudentPhoneNumber bigint, StudentGender varchar(10), StudentBloodGroup varchar(3), StudentAddress varchar(30));



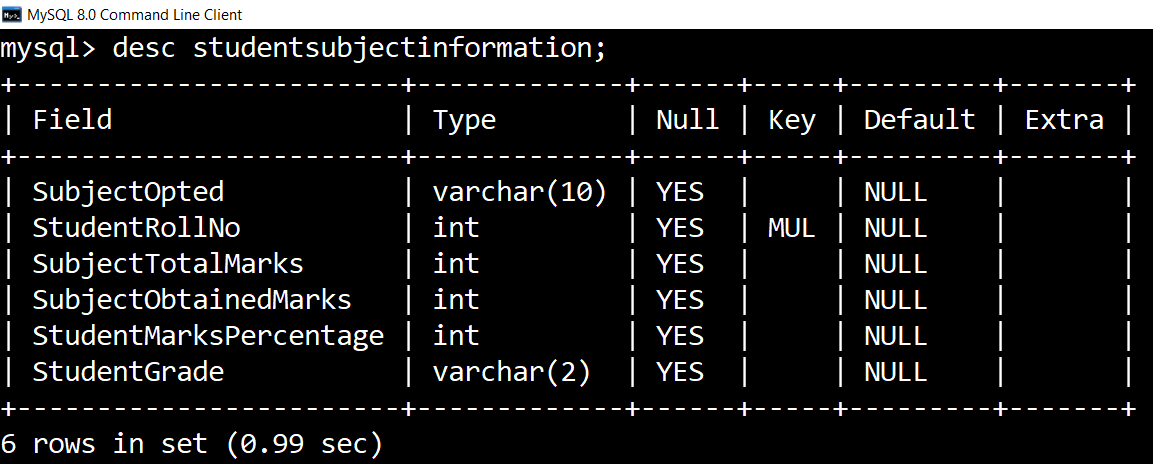
b)

create table StudentAdmissionPayment ( StudentRollNo int primary key, AmountPaid int, AmountBalance int, AmountStatus varchar(6), PaymentDate date, BankName varchar(20), TransactionId varchar(20), PayeeAccoutNumber varchar(20) ,foreign key(StudentRollNo) references StudentBasicInformation(StudentRollNo));



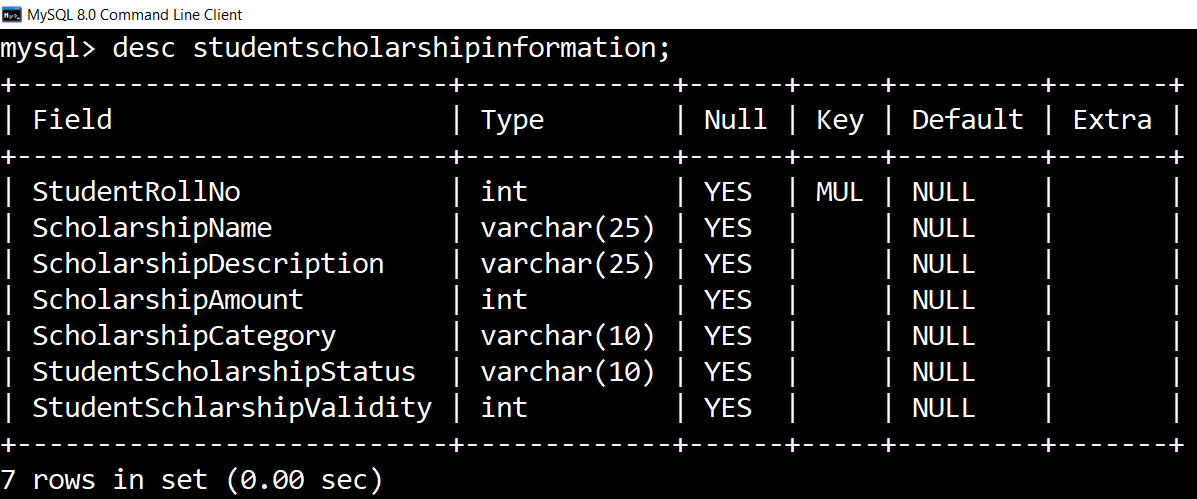
c)

create table StudentSubjectInformation ( SubjectOpted varchar(10), StudentRollNo int , SubjectTotalMarks int,SubjectObtainedMarks int, StudentMarksPercentage int, StudentGrade varchar(2),foreign key(StudentRollNo) references StudentBasicInformation(StudentRollNo));



d)

create table StudentScholarshipInformation( StudentRollNo int, ScholarshipName varchar(25),ScholarshipDescription varchar(25), ScholarshipAmount int, ScholarshipCategory varchar(10), StudentScholarshipStatus varchar(10),StudentSchlarshipValidity int,foreign key(StudentRollNo) references StudentBasicInformation(StudentRollNo));



1. Insert more than 10 records in each and every table created

**Solution:-**

**Table Name:- StudentBasicInformation**

INSERT INTO StudentBasicInformation (StudentName,StudentSurname,StudentRollNo,StudentPhoneNumber,StudentClass,StudentGender,StudentBloodGroup,StudentAddress)values("niket", "jain",1 ,9473628474, 10, "female", "ab+", "jaipur"),

("raman", "mohapatra", 2,8472627733, 12, "male", "b+", "kota"),

("archit", "dudeja", 3,9836263744, 10, "male", "b+", "jaipur"),

("rohit", "arora", 4,8372738503, 11, "male", "b+", "tonk"),

("meetali", "sahijwani",5 ,9372747382, 12, "female", "a+", "agra"),

("rishika", "mamgain", 6,8372738482, 11, "female", "b+", "haryana"),

("rishabh", "vyas", 7,8382747389, 12, "male", "ab-", "delhi" ),

("tarun", "mehta", 8,837283734, 12, "male", "b+", "dehradun"),

("tarushi", "panth",9 ,9284732881, 12, "female", "a+", "kota"),

("kritesh", "agarwal",10 ,9873627167, 12, "male", "ab+", "udaipur");

**Table Name:- StudentAdmissionPayment**

INSERT INTO StudentAdmissionPayment values

(2,8000, 0, "paid", "2000-12-24", "hdfc", "1234", "1234543245"),

(10,2000, 0, "paid", "2001-12-23", "icici", "4321", "53412432"),

(5,3000, 300, "upaid", "2000-12-30", "sbi", "2343", "4123454134"),

(4,4000, 200, "upaid", "1998-09-30", "hdfc", "5234", "541234524"),

(1,5000, 0, "paid", "2000-10-21", "indus", "52344", "4123525334"),

(9,6000, 500, "unpaid", "2000-09-21", "sbi", "4235", "431244123"),

(8,7000, 150, "unpaid", "2000-09-23", "sbi", "5234", "4231423243"),

(6,2000, 0, "paid", "2010-11-17", "icici", "42345", "342414142"),

(3,5000, 200, "unpaid", "2004-12-23", "icici", "4234", "3412432234"),

(7,6000, 250, "unpaid", "2012-12-22", "sbi", "4235", "34142523532");

**Table Name:- StudentSubjectInformation**

INSERT INTO StudentSubjectInformation values

("pcm", 3,500, 400, 80, "B"),

("commerce",2 ,500, 390, 78, "B"),

("arts", 4,500, 410, 82, "B"),

("commerce", 5,500, 380, 76, "B"),

("arts", 8,500, 210, 42, "D"),

("commerce",7,500, 400, 80, "B"),

("pcm", 6,500, 430, 86, "B"),

("pcb",1, 500, 490, 98, "A"),

("arts", 9,500, 230, 46, "D"),

("commerce", 10,500, 234, 49, "D");

**Table Name:- StudentScholarshipInformation**

INSERT INTO StudentScholarshipInformation values

(10,"xya", "fasdf", 10000, "gen", "yes", 1),

(8,"dfa", "asdf", 20000, "male", "yes", 2),

(6,"fasd", "fasd", 10000, "gen", "no", 3),

(4,"fas", "fasd", 4000, "disabled", "yes", 1),

(2,"fa", "fasd", 5000, "topper", "yes", 2),

(1,"dfs", "fasd", 9000, "domicile", "no", 3),

(5,"sdf", "fasd", 100000, "tmpai", "yes", 4),

(3,"fda", "fsd", 70000, "upes", "no", 3),

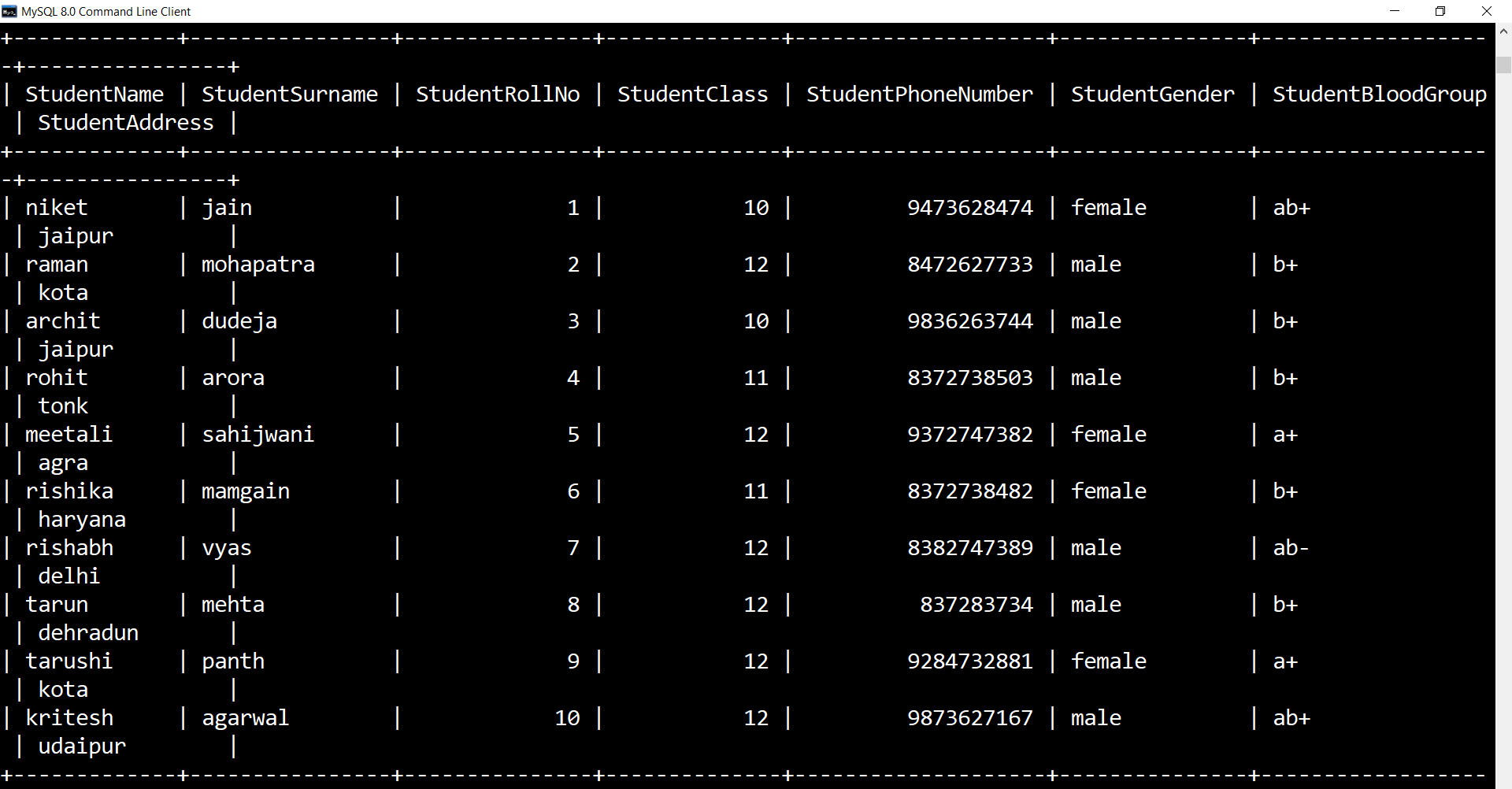
(7,"asdf", "gdsaf", 8000, "higheredu", "yes", 2),

(9,"fads", "fasd", 1000, "nonee", "yes", 1);

1. Snap of the all the tables once the insertion is completed

**Solution:-**

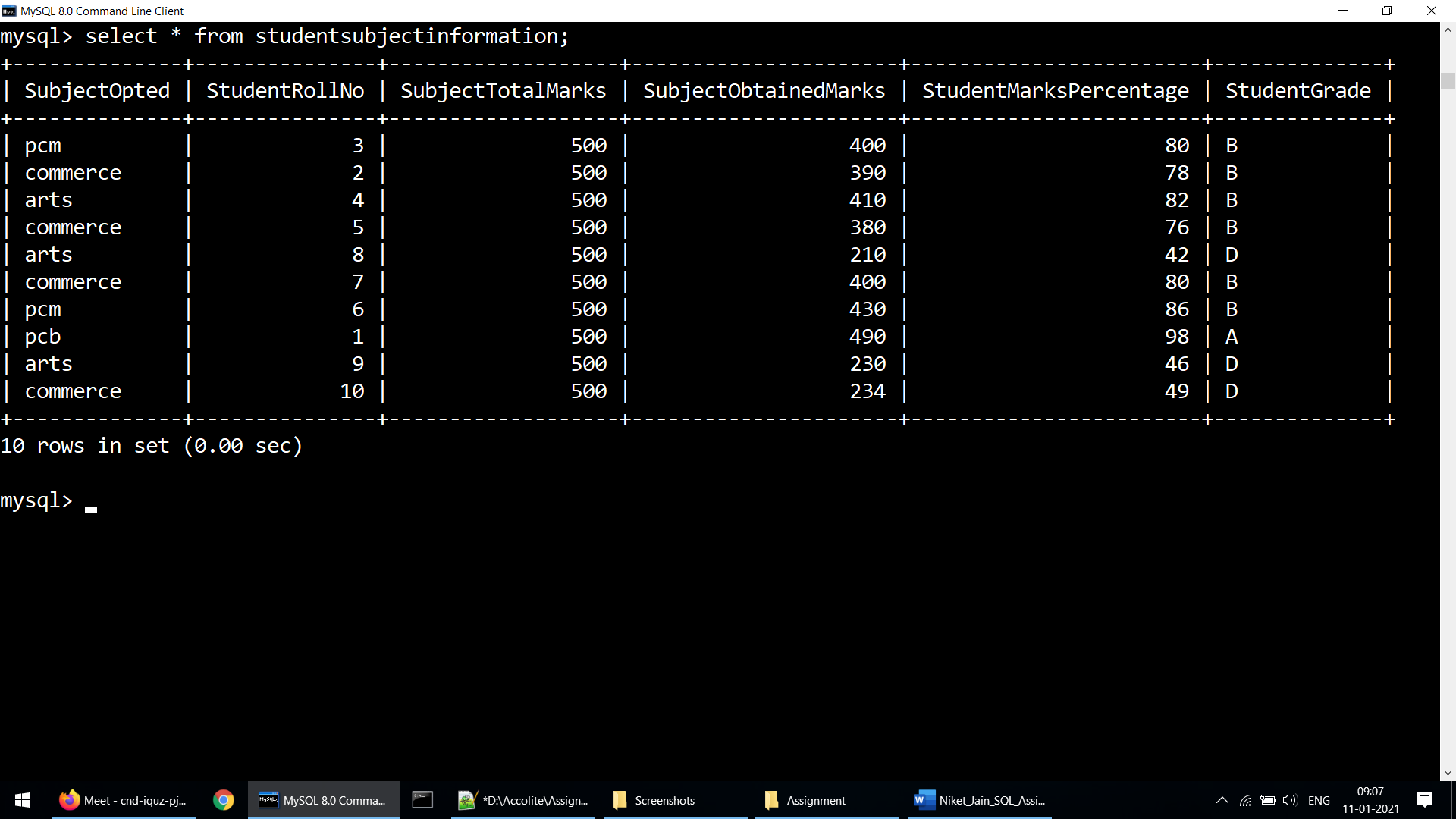
**Table Name:- StudentBasicInformation**

****

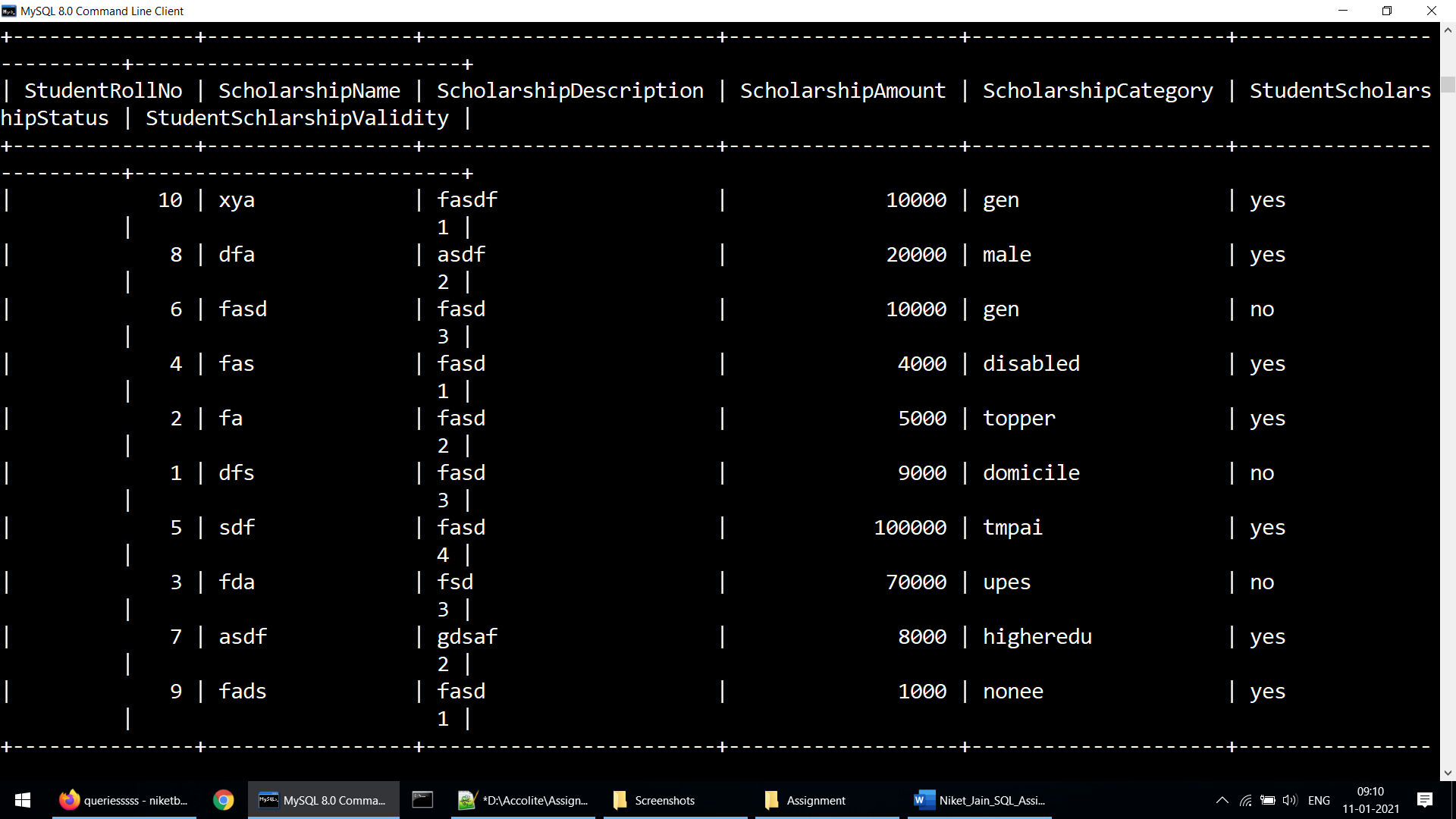
**Table Name:- StudentAdmissionPayment**

****

**Table Name:- StudentSubjectInformation**

****

**Table Name:- StudentScholarshipInformation**

****

1. Update any 5 records of your choice in any table like update the StudentAddress with some other address content and likewise so on with any records of any table of your choice

**Solution:-**

a)

update studentbasicinformation set StudentPhoneNumber=7734699715 where StudentRollNo=1;

b)

update studentscholarshipinformation set StudentScholarshipStatus="yes" where StudentRollNo=3;

c)

update studentbasicinformation set StudentAddress="dehradun" where StudentRollNo=6;

d)

update StudentSubjectInformation set SubjectOpted="pcm" where StudentRollNo=2;

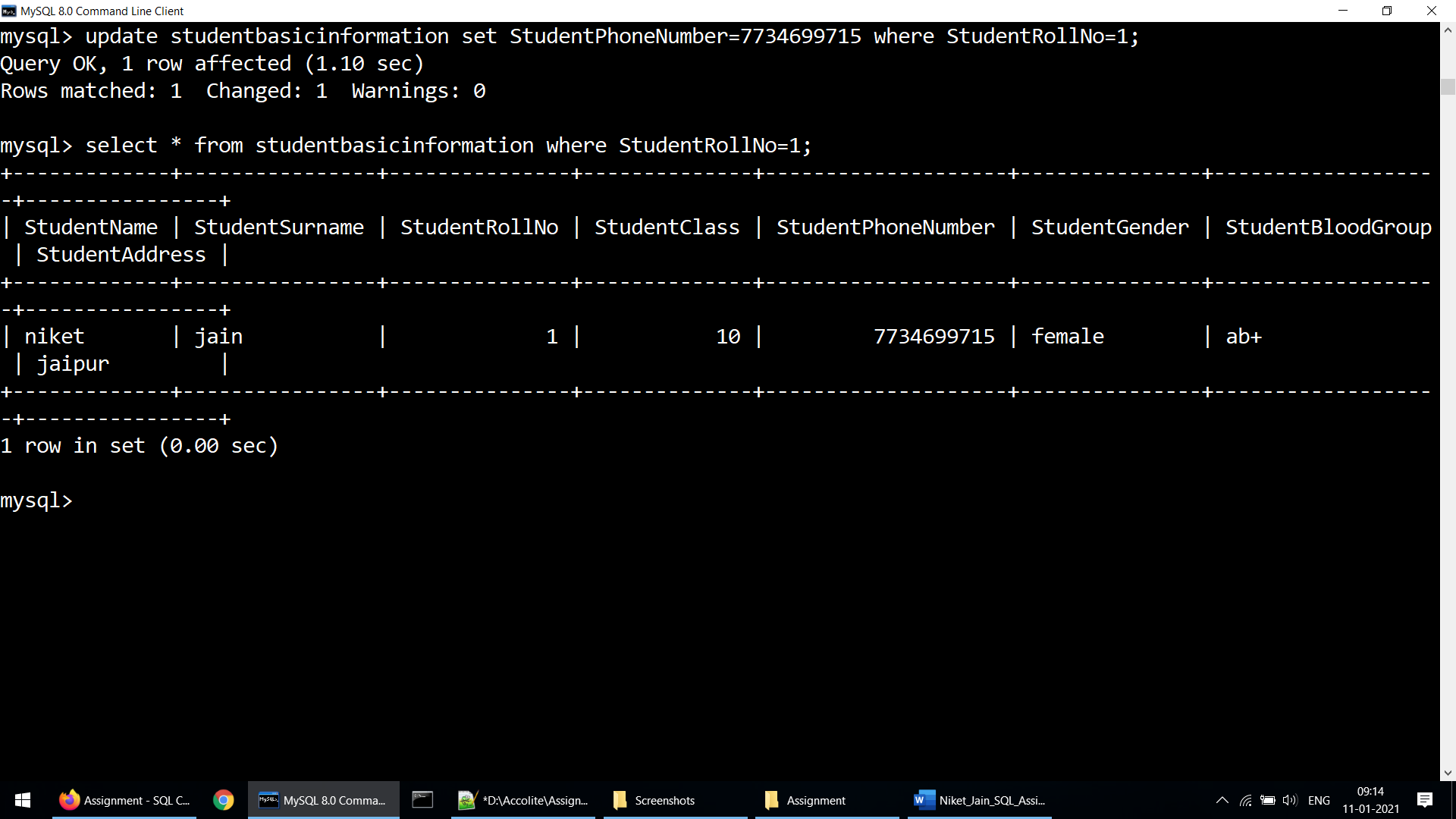
e)

update StudentAdmissionPayment set AmountStatus ="paid" where StudentRollNo=3;

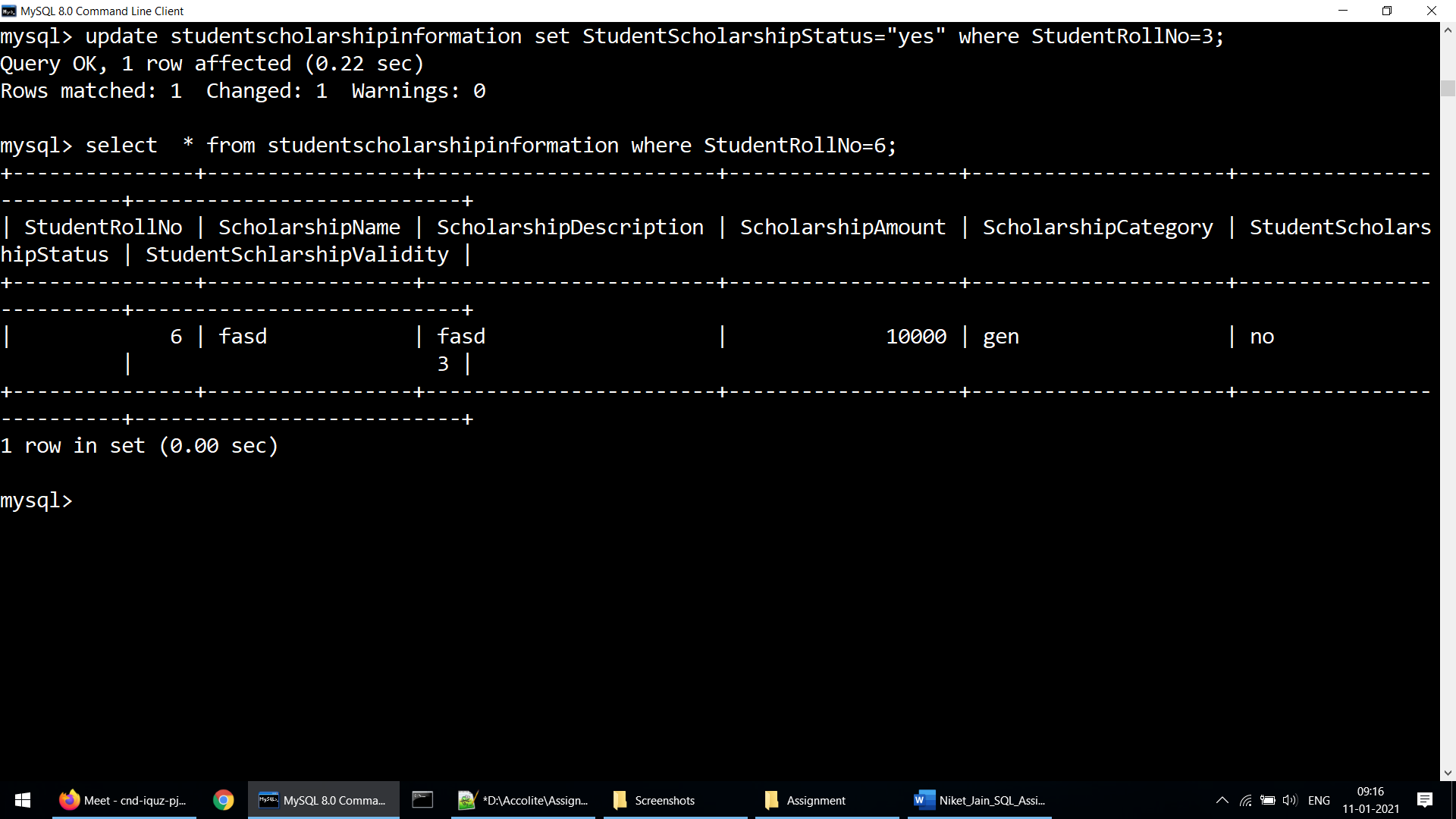
1. Snap of the all the tables post updation

**Solution:-**

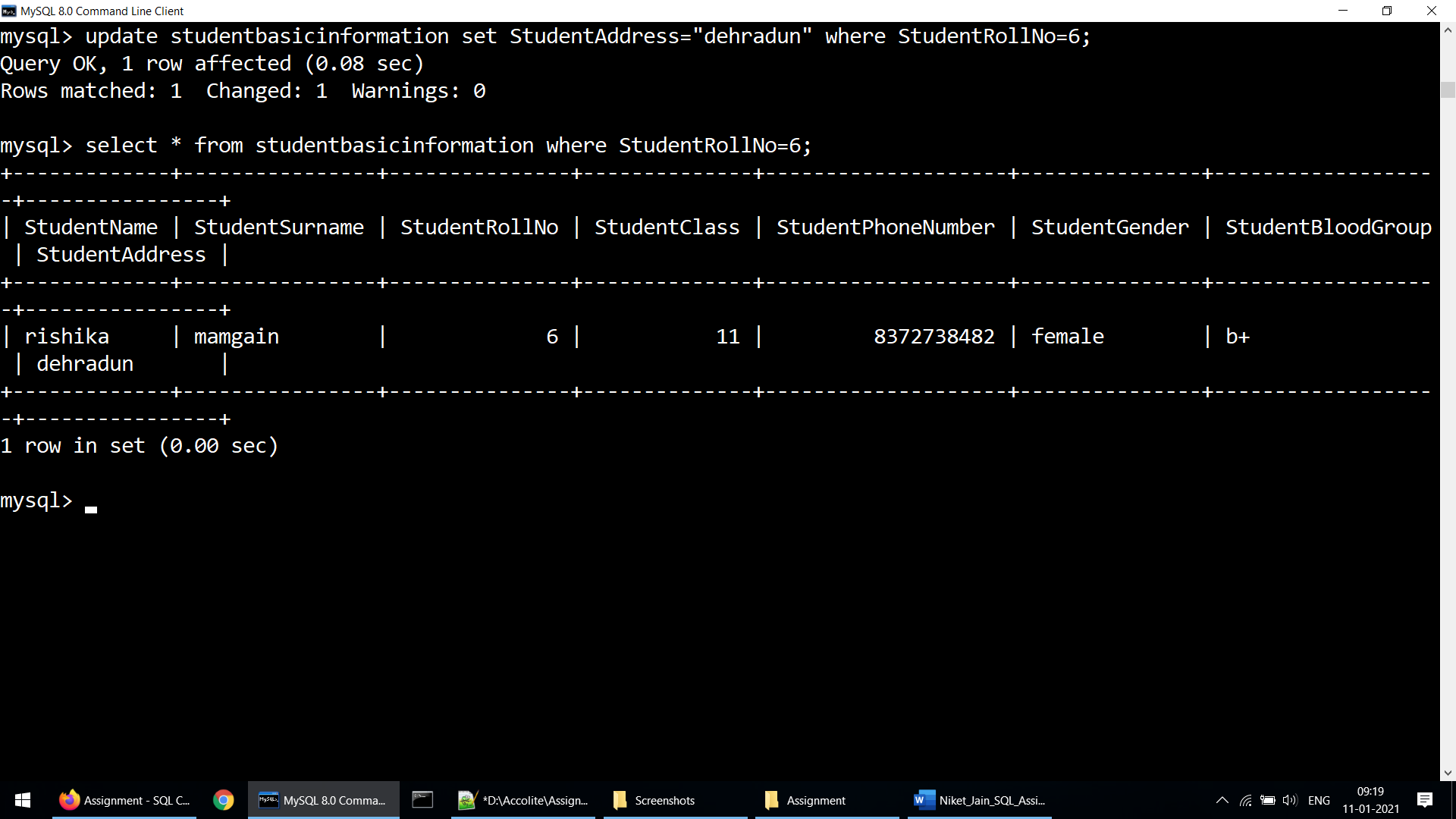
a)



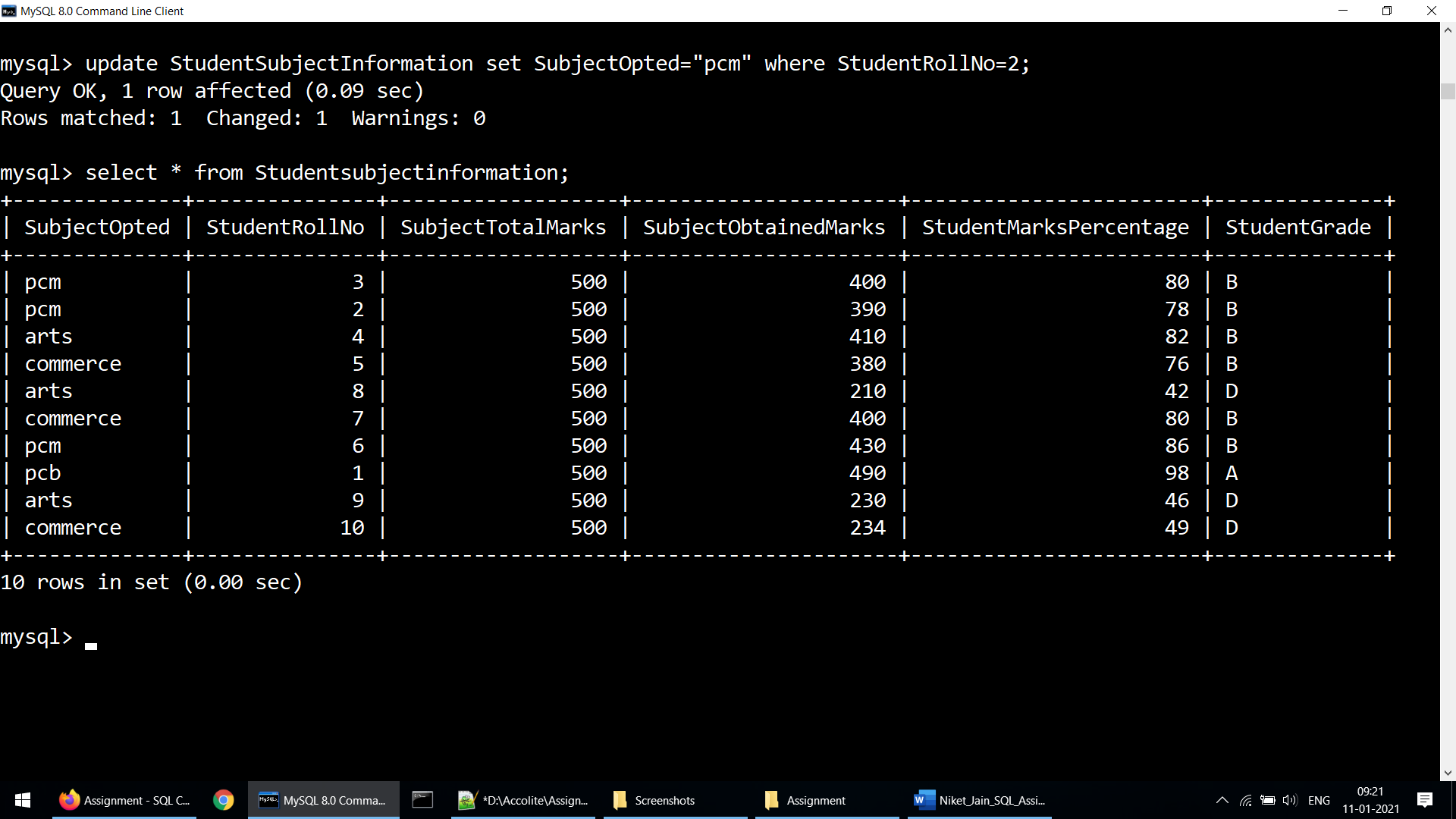
b)



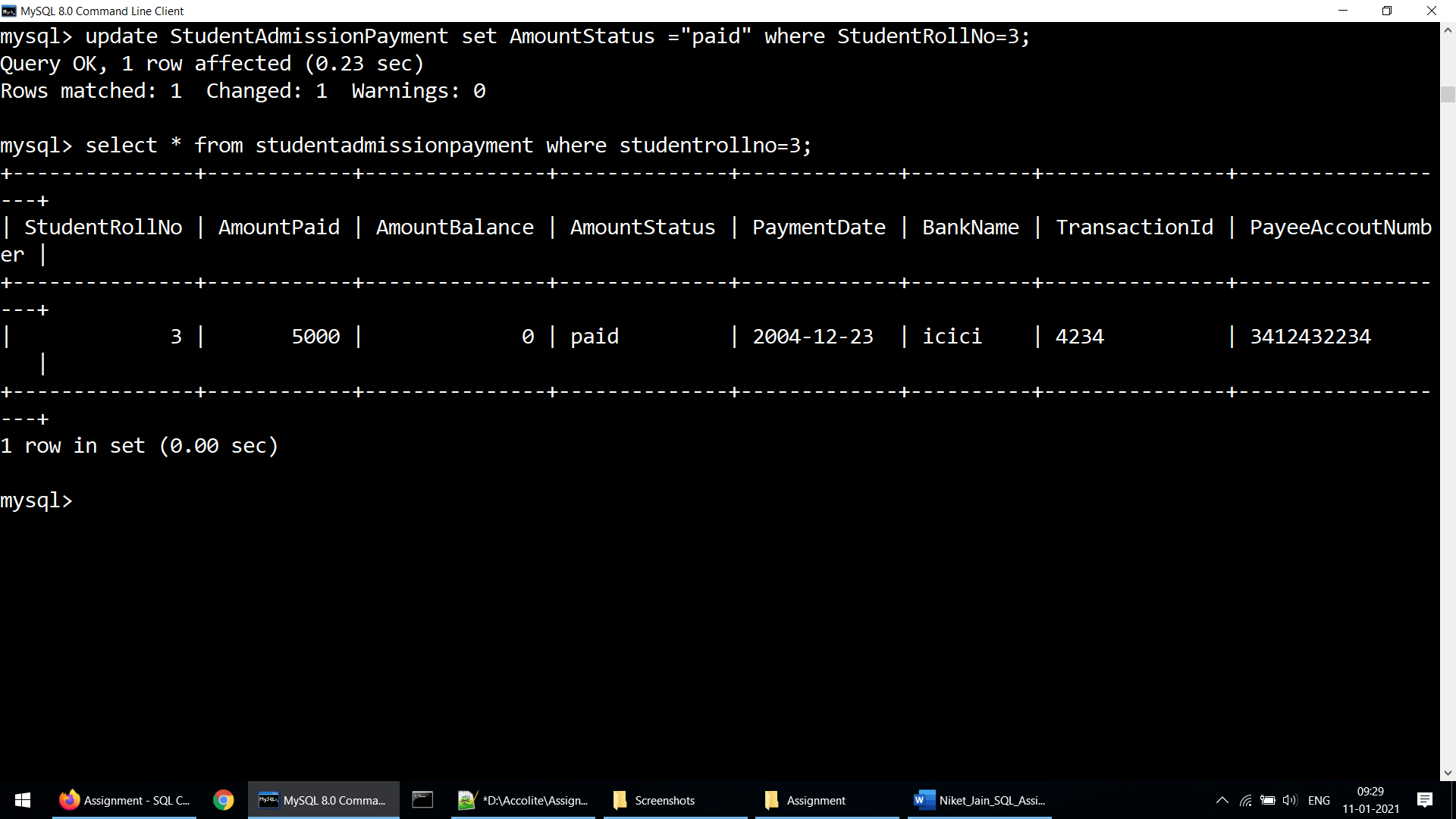
c)



d)



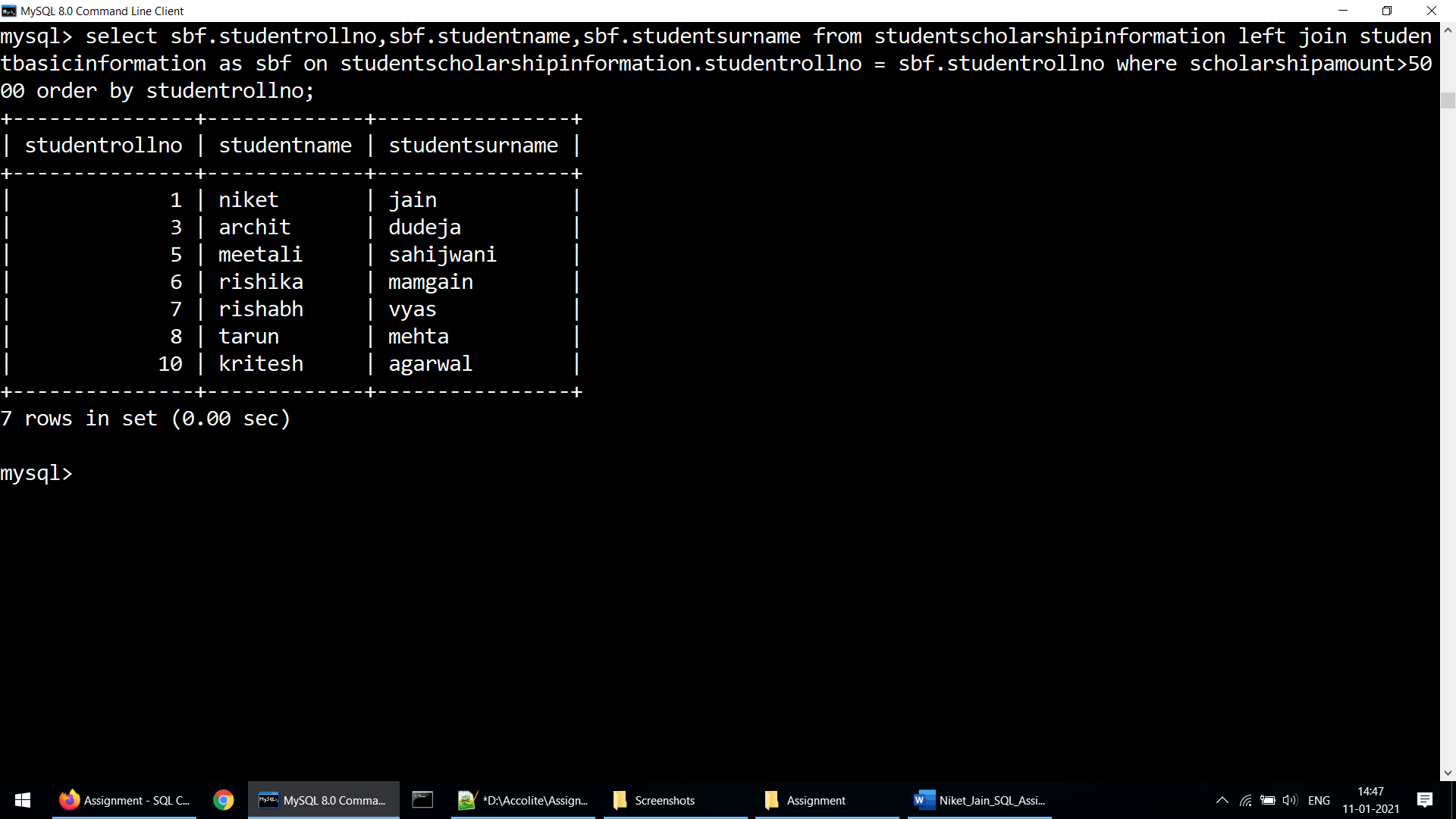
e)



1. Select the student details records who has received the scholarship more than 5000Rs/-

**Solution:-**

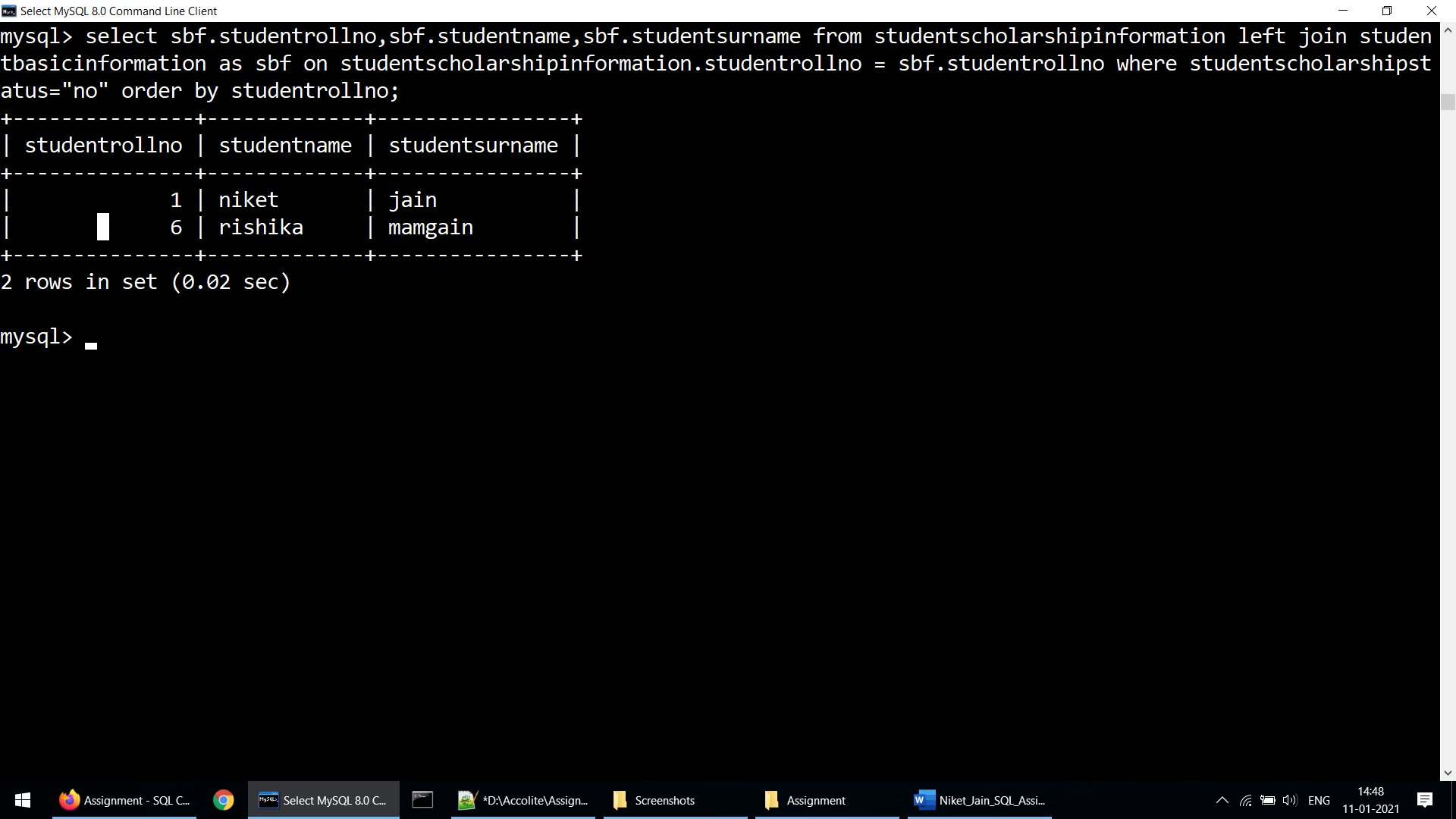
select sbf.studentrollno,sbf.studentname,sbf.studentsurname from studentscholarshipinformation inner join studentbasicinformation as sbf on studentscholarshipinformation.studentrollno = sbf.studentrollno where scholarshipamount>5000 order by studentrollno;



1. Select the students who opted for scholarship but has not got the scholarship

**Solution:-**

select sbf.studentrollno,sbf.studentname,sbf.studentsurname from studentscholarshipinformation inner join studentbasicinformation as sbf on studentscholarshipinformation.studentrollno = sbf.studentrollno where studentscholarshipstatus="no" order by studentrollno;

****

1. Fill in data for the percentage column i.e. StudentMarksPercentage in the table StudentSubjectInformation by creating and using the stored procedure created

**Solution:-**

delimiter //

create procedure insertpercentages()

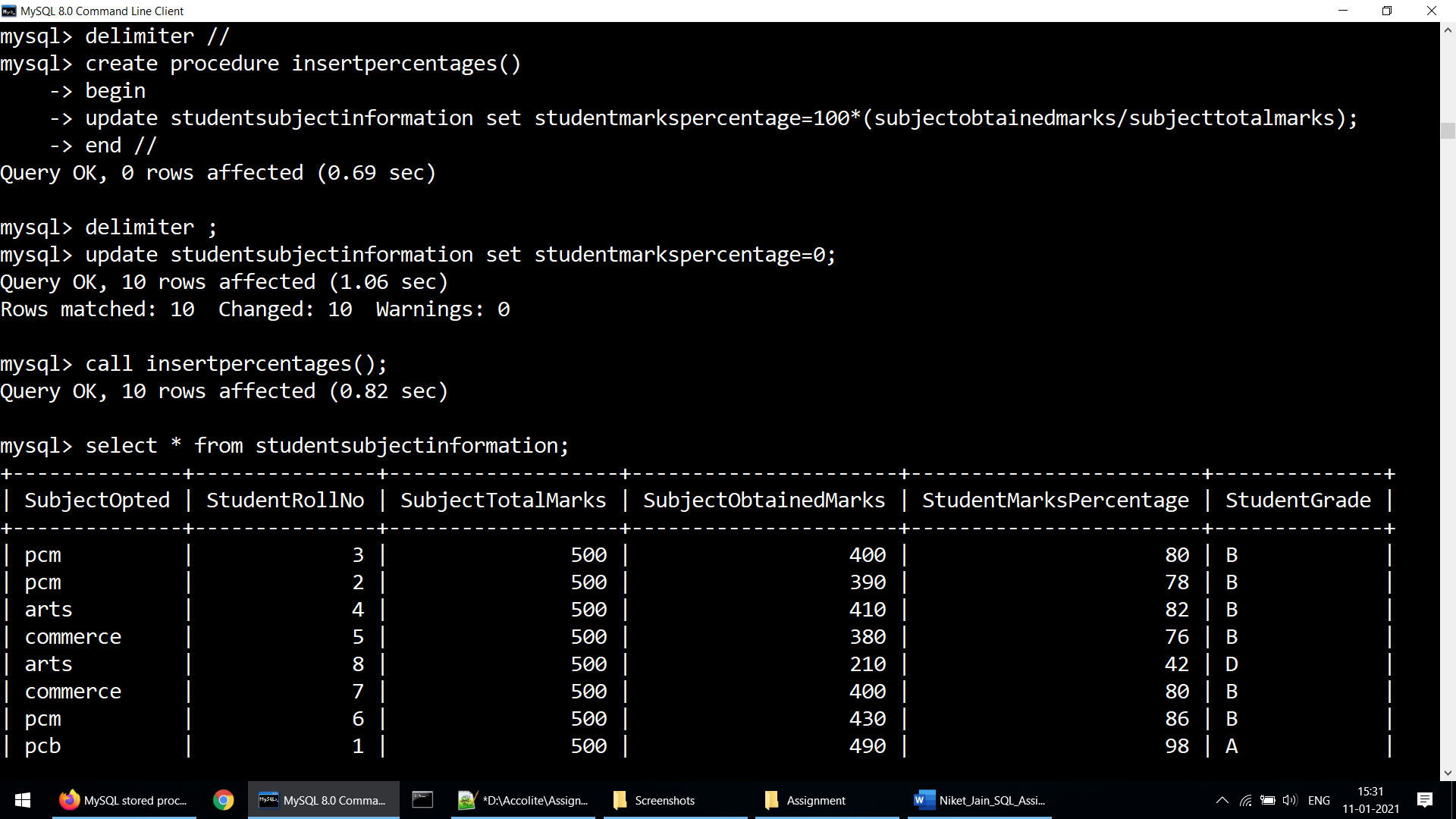
begin

update studentsubjectinformation set studentmarkspercentage=100\*(subjectobtainedmarks/subjecttotalmarks);

end //

delimiter ;

call insertpercentage();



1. Decide the category of the scholarship depending upon the marks/percentage obtained by the student and likewise update the ScholarshipCategory column, create a stored procedure in order to handle this operation

**Solution:-**

delimiter //

create procedure categoryfill()

begin

declare total int;

declare ii int;

declare rollno int;

declare per int;

select count(\*) into total from studentscholarshipinformation;

set ii=0;

while ii<total do

select studentrollno into rollno from studentscholarshipinformation limit ii,1;

select studentmarkspercentage into per from studentsubjectinformation where studentrollno=rollno;

if per >80 then

update studentscholarshipinformation set ScholarshipCategory="Category A" where studentrollno = rollno;

elseif per >50 and per <=80 then

update studentscholarshipinformation set ScholarshipCategory="Category B" where studentrollno = rollno;

elseif per>0 and per<=50 then

update studentscholarshipinformation set ScholarshipCategory="Category C" where studentrollno = rollno;

end if;

set ii = ii+1;

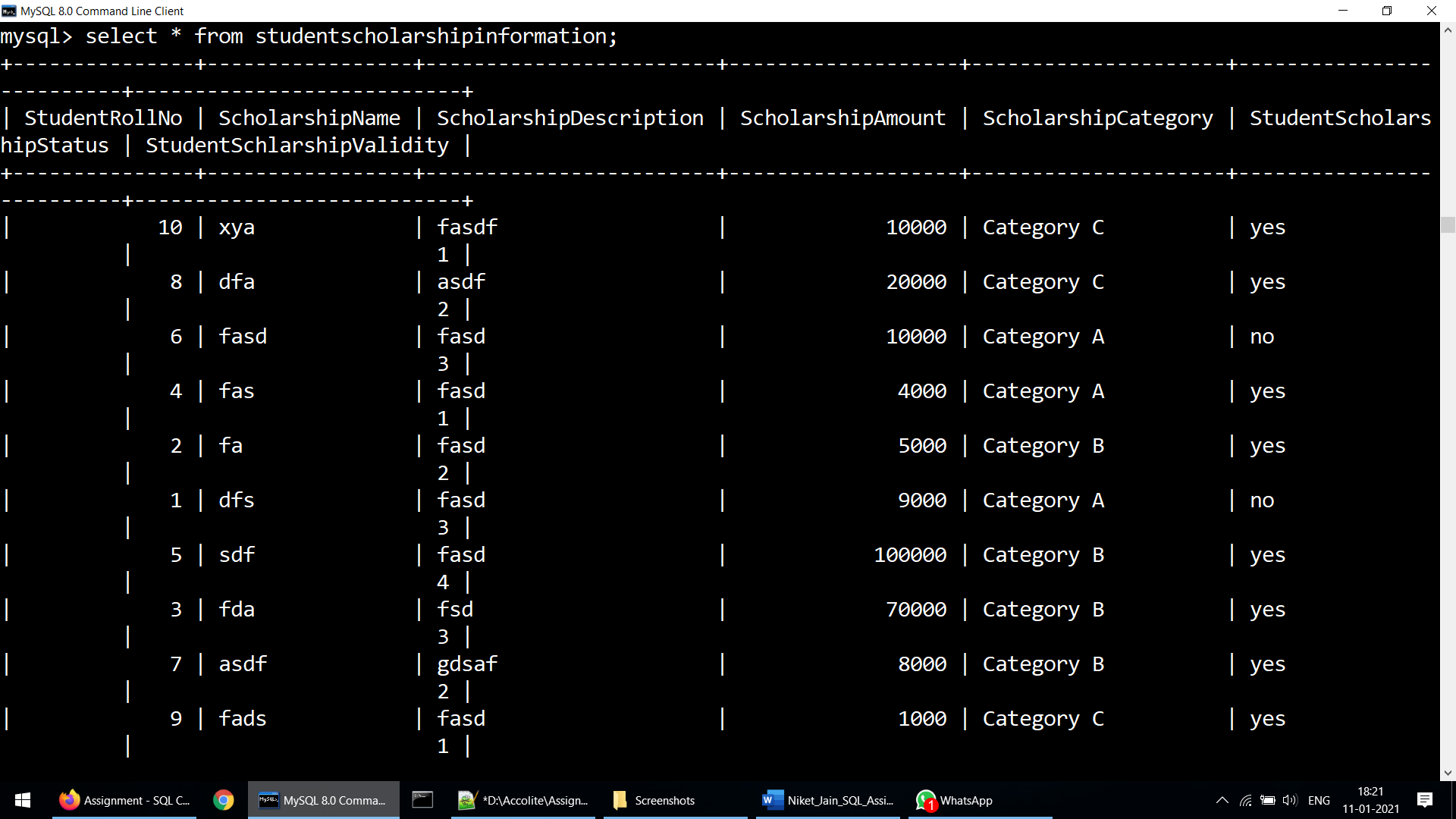
end while;

end //

delimiter ;

call categoryfill();

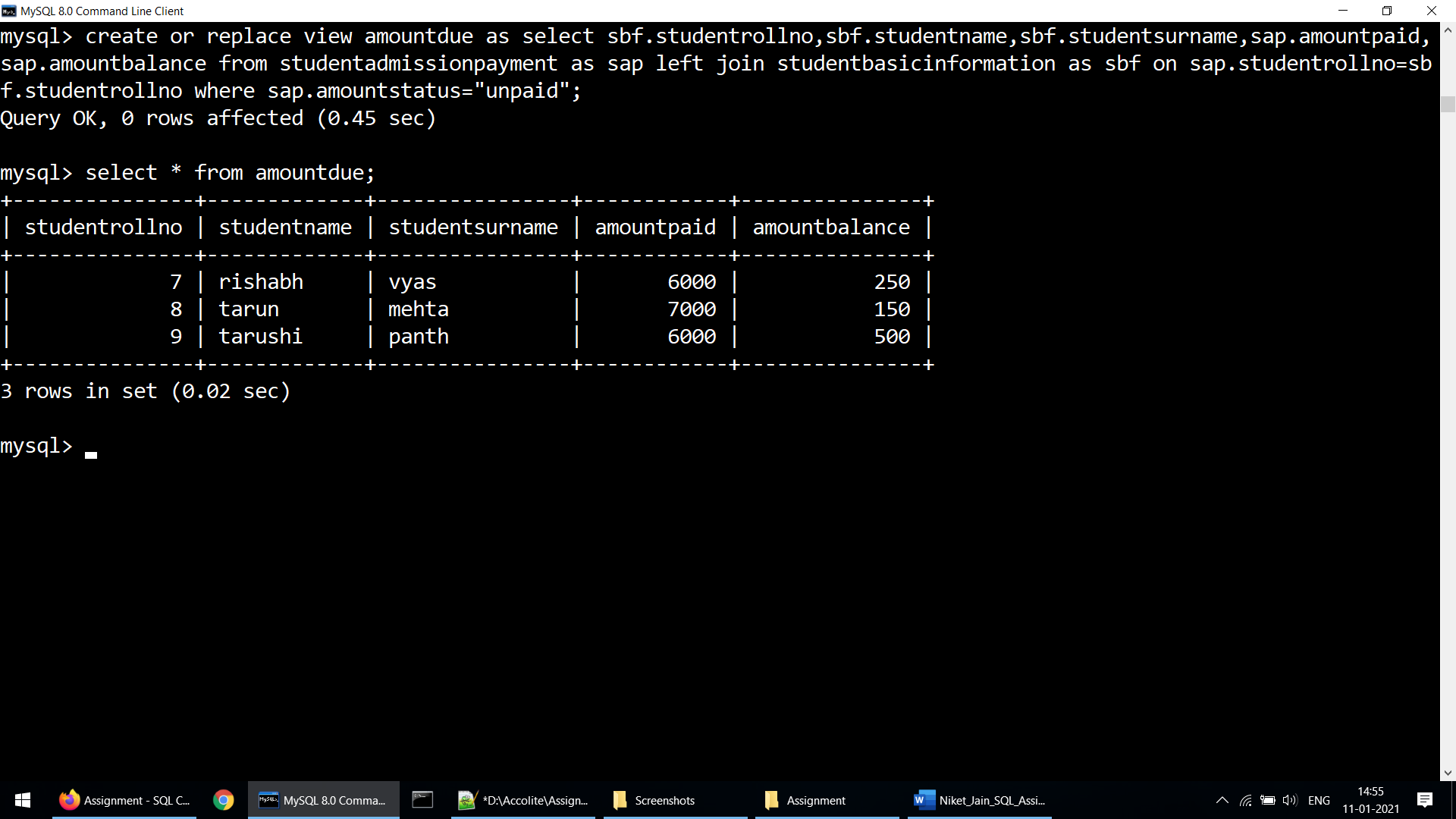
**Note:- The given screenshot is after calling procedure and then running the select query.**



1. Create the View which shows balance amount to be paid by the student along with the student detailed information (use join)

**Solution:-**

create or replace view amountdue as select sbf.studentrollno,sbf.studentname,sbf.studentsurname,sap.amountpaid,sap.amountbalance from studentadmissionpayment as sap inner join studentbasicinformation as sbf on sap.studentrollno=sbf.studentrollno where sap.amountstatus="unpaid";

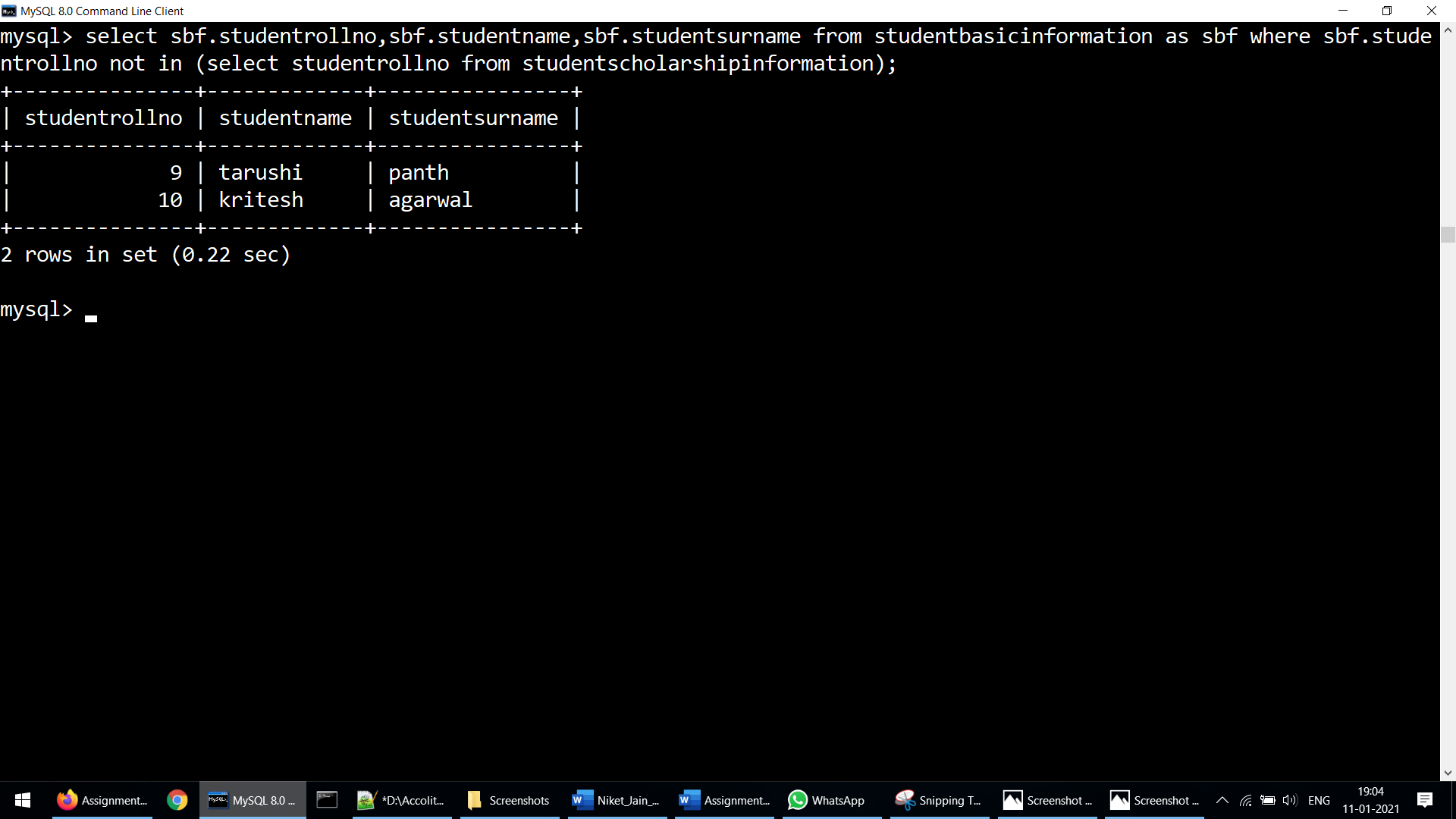


1. Get the details of the students who haven’t got any scholarship (use joins/subqueries)

**Solution:-**

select sbf.studentrollno,sbf.studentname,sbf.studentsurname from studentbasicinformation as sbf where sbf.studentrollno not in (select studentrollno from studentscholarshipinformation);

**Note:- This screenshot is taken after deleting some rows from studentscholarshipinformation table and then this select query was executed.**



1. Create Stored Procedure which will be return the amount balance to be paid by the student as per the student roll number passed through the stored procedure as the input

**Solution:-**

delimiter //

create procedure amountcheck ( in rollno int )

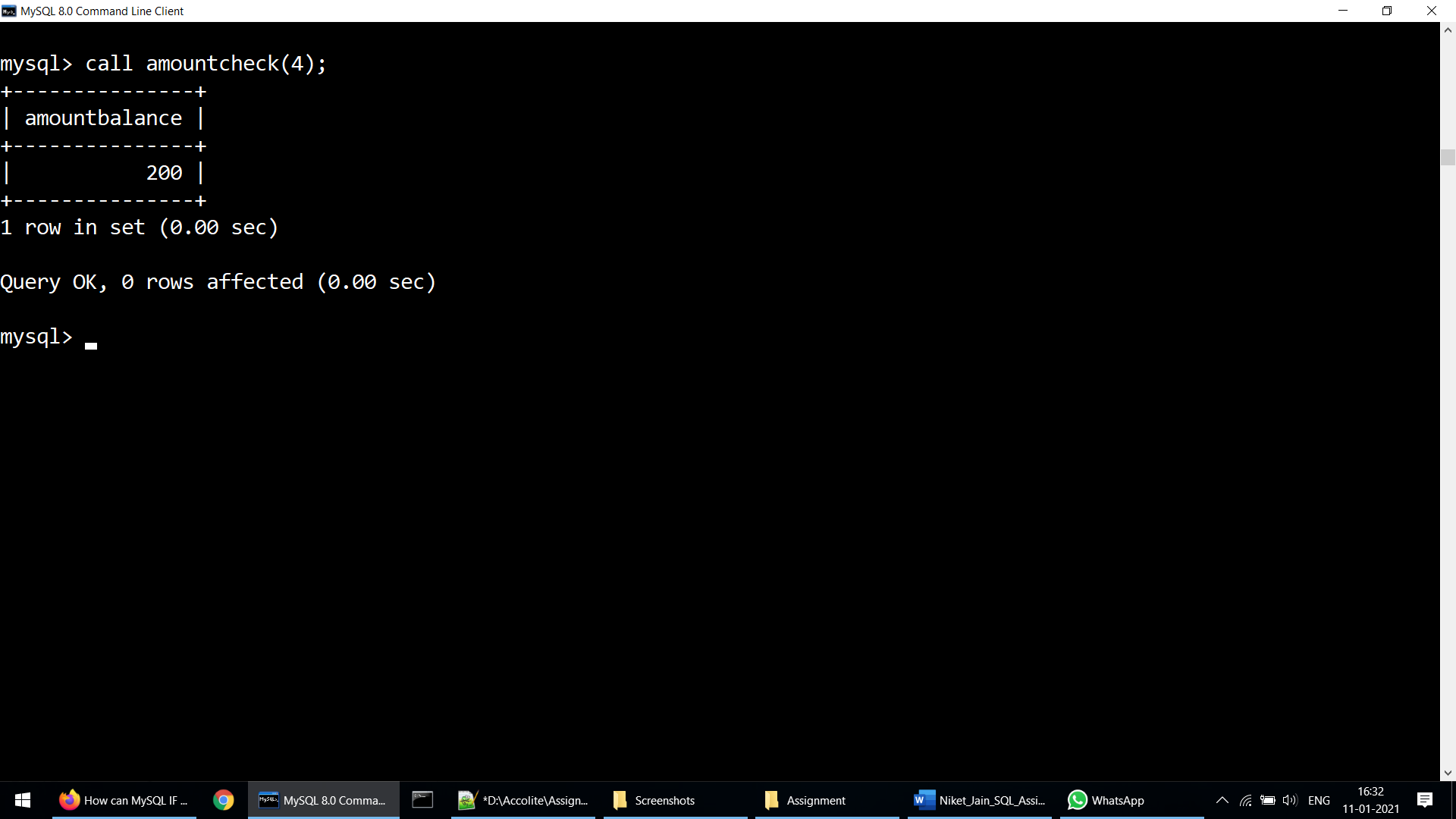
begin

select amountbalance from studentadmissionpayment where studentrollno=rollno;

end //

delimiter ;

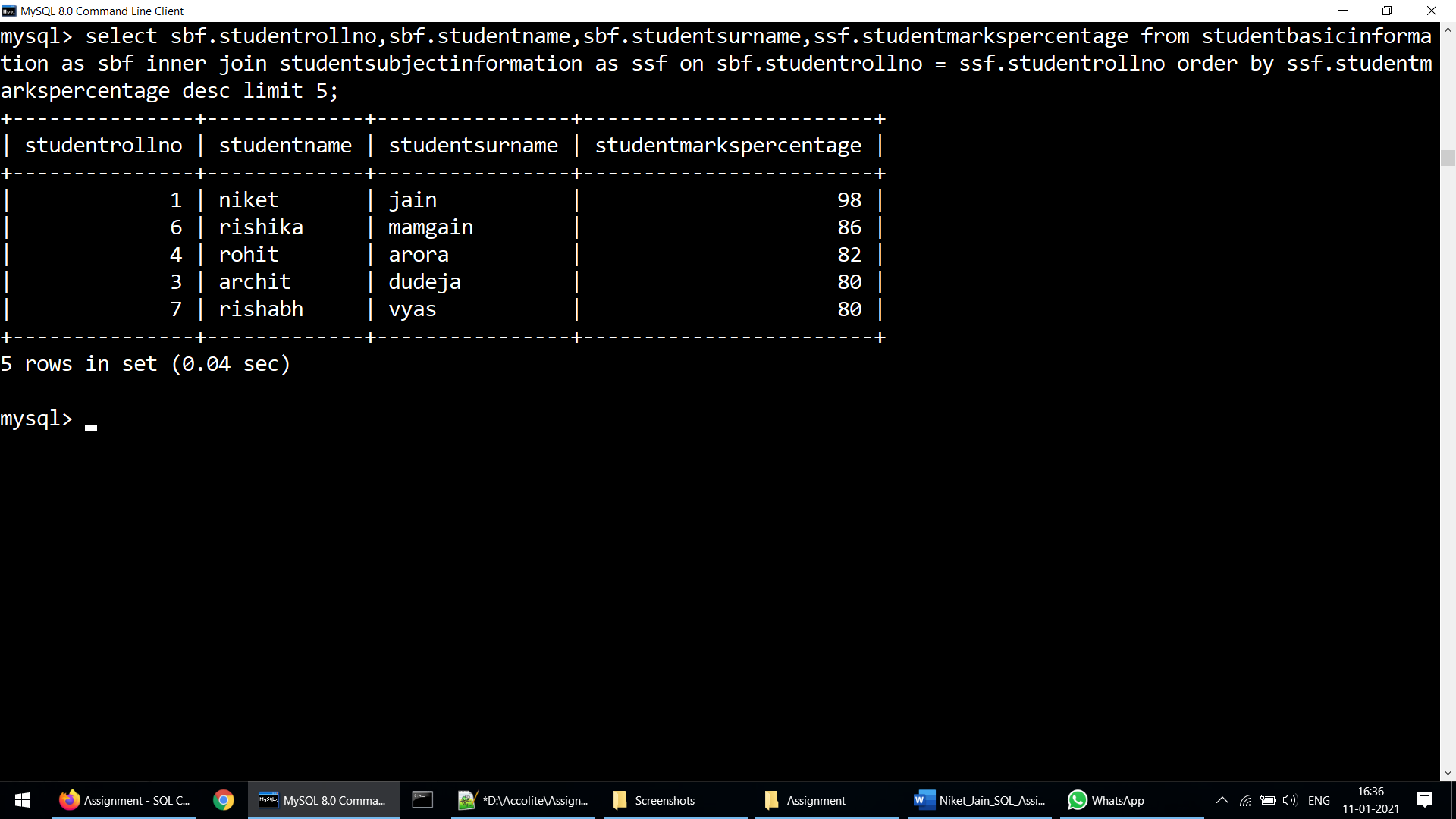
call amountcheck(RollNo);



1. Retrieve the top five student details as per the StudentMarksPercentage values (use subqueries)

**Solution:-**

sbf.studentrollno,sbf.studentname,sbf.studentsurname,ssf.studentmarkspercentage from studentbasicinformation as sbf inner join studentsubjectinformation as ssf on sbf.studentrollno = ssf.studentrollno order by ssf.studentmarkspercentage desc limit 5;



1. Try to use all the three types of join learned today in a relevant way, and explain the same why you thought of using that particular join for your selected scenarios (try to cover relevant and real time scenarios for all the three studied joins)

**Solution:-**

**a) INNER JOIN**

Real-Life sceanrio:- To retrieve information about those students who are failed in subject they opted.So that this can be informed to their guardian.

Create or replace view lowgradestudentdetails as Select sbf.studentrollno,sbf.studentname,sbf.studentsurname,sbf.studentphonenumber,sbf.studentclass,ssf.studentgrade from studentbasicinformation as sbf inner join studentsubjectinformation as ssf on sbf.studentrollno= ssf.studentrollno where ssf.studentgrade="D";

Select \* from lowgradestudentdetails;

**b) Left Outer Join**

Real-Life Scenario:- To retrieve information about students who have scholarship and who have not.

Create or replace view scholarshipdetails as select sbf.studentrollno,sbf.studentname,sbf.studentsurname,sbf.studentphonenumber,ssf.scholarshipcategory from studentbasicinformation as sbf left join studentscholarshipinformation as ssf on sbf.studentrollno= ssf.studentrollno;

Select \* from scholarshipdetails;

**c) Right Outer Join**

Real-Life Scenario:- To retrieve information about student who in last year of school and their scholarship period is not over yet so that their scholarship can be extended to their college/university.

Create or replace view scholarshipvaliditydetails as select sbf.studentrollno,sbf.studentname,sbf.studentsurname,sbf.studentphonenumber,ssf.studentschlarshipvalidity from studentbasicinformation as sbf right join studentscholarshipinformation as ssf on sbf.studentrollno= ssf.studentrollno where sbf.studentclass=12;

Select \* from scholarshipvaliditydetails;

1. Mention the differences between the delete, drop and truncate commands

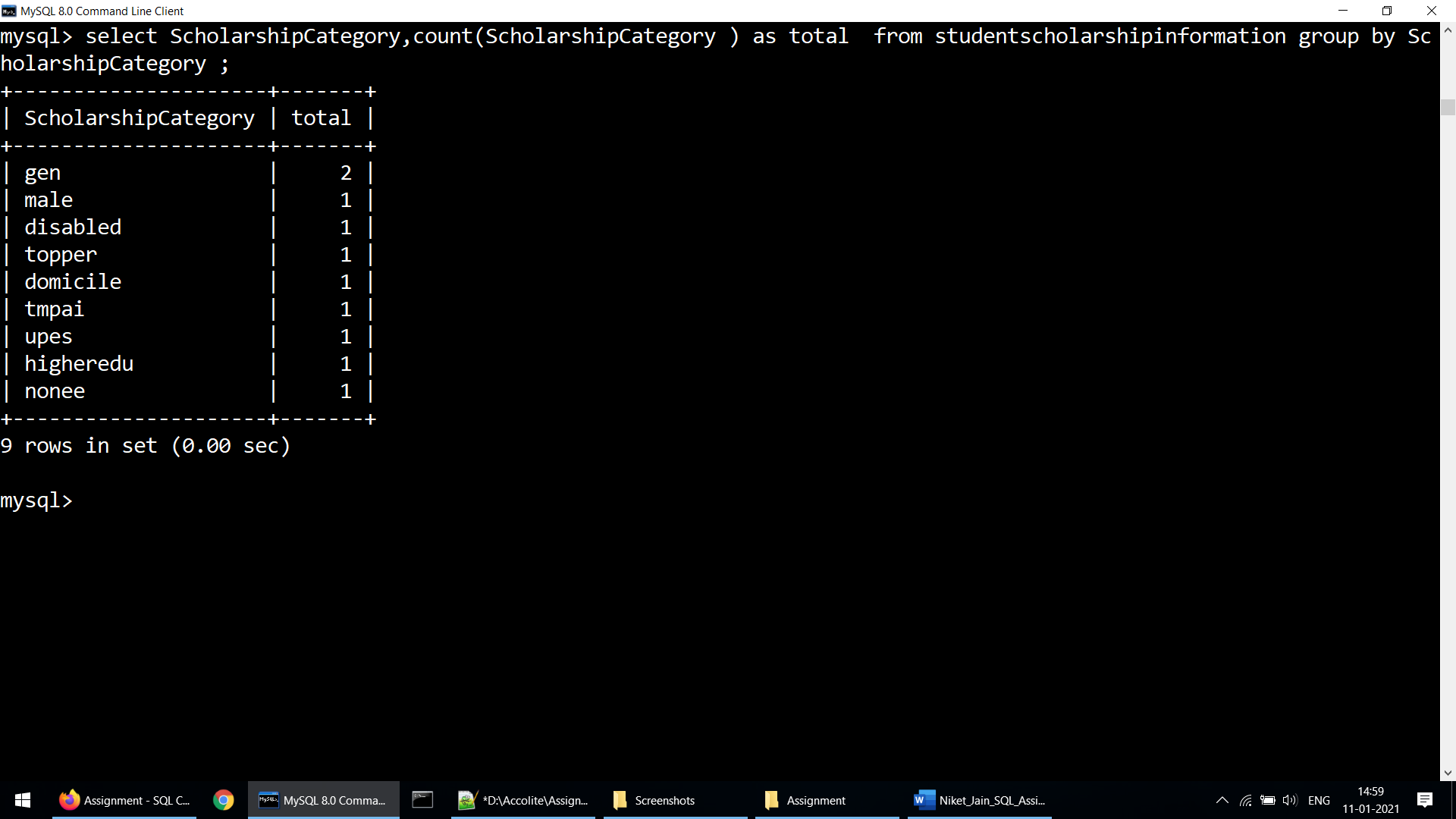
**Solution:-**

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | DELETE | DROP | TRUNCATE |
| Type | DELETE is a Data Manipulation Language (DML) command | DROP is a Data Definition Language (DDL) command | TRUNCATE is a Data Definition Language (DDL) command |
| Use | It is used to delete the data in a table, but not the table structure. | It is used to delete the data and the table structure. | Like DELETE, It is used to delete the data and not the structure. |
| Single Row | We can delete a single row using WHERE clause | We cannot delete a single row only. | We cannot delete a single row as WHERE can't be used. |
| Rollback | We can use ROLLBACK to restore | We can't restore by using ROLLBACK | We can't restore by using ROLLBACK |
| Syntax | DELETE from STUDENT;  DELETE from STUDENT where Age<18; | DROP table STUDENT; | TRUNCATE table STUDENT, |

1. Get the count of the Scholarship category which is highly been availed by the students, i.e. get the count of the total number of students corresponding to the each scholarships category

**Solution:-**

select ScholarshipCategory,count(ScholarshipCategory ) as total from studentscholarshipinformation group by ScholarshipCategory ;



1. Along with the assignment no. 17 try to retrieve the maximum used scholarship category

**Solution:-**

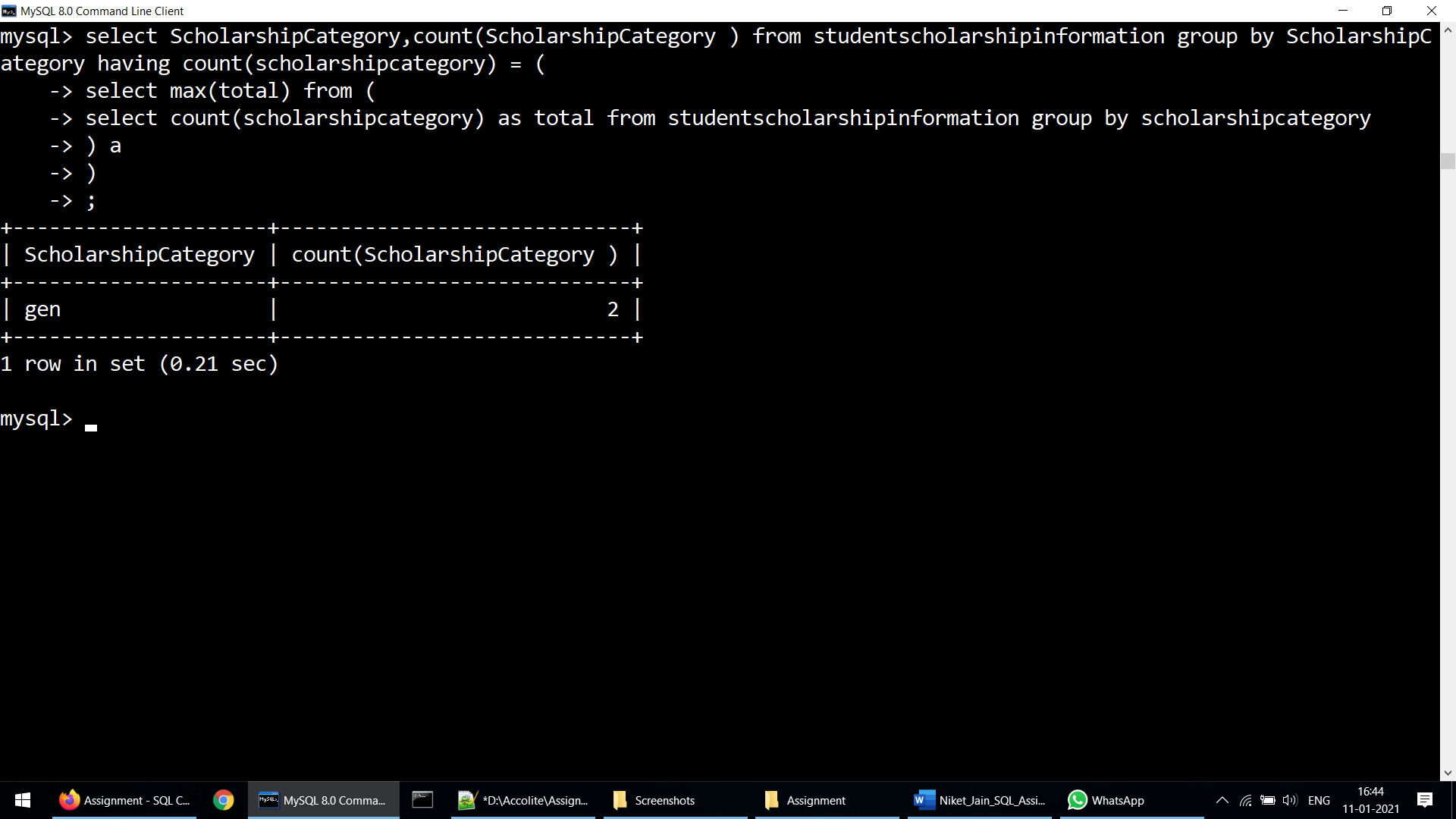
select ScholarshipCategory,count(ScholarshipCategory ) from studentscholarshipinformation group by ScholarshipCategory having count(scholarshipcategory) = (

select max(total) from (

select count(scholarshipcategory) as total from studentscholarshipinformation group by scholarshipcategory

) a

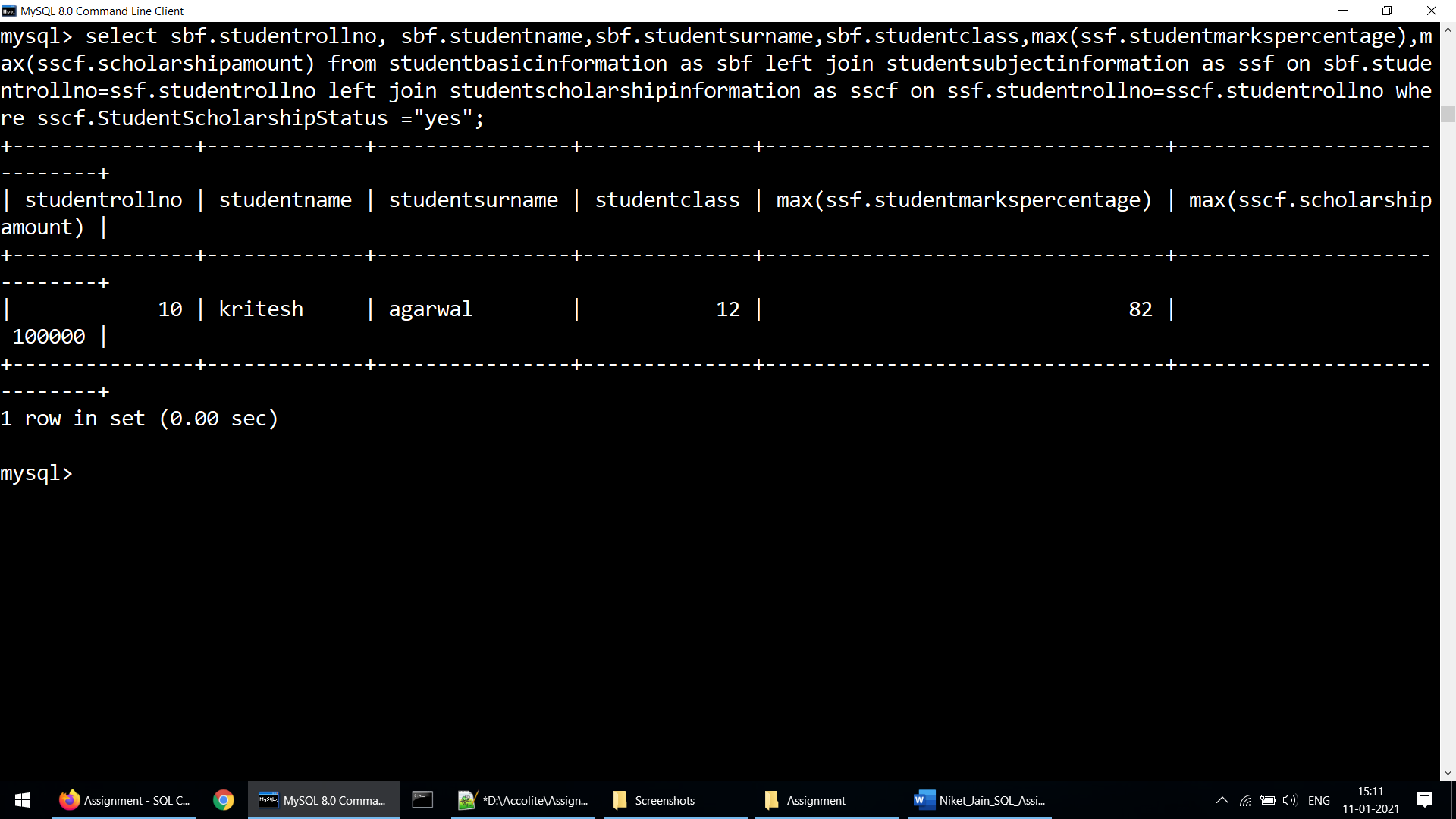
);

****

1. Retrieve the percentage of the students along with students detailed information who has scored the highest percentage along with availing the maximum scholarship amount

**Solution:-**

select sbf.studentrollno, sbf.studentname,sbf.studentsurname,sbf.studentclass,max(ssf.studentmarkspercentage),max(sscf.scholarshipamount) from studentbasicinformation as sbf left join studentsubjectinformation as ssf on sbf.studentrollno=ssf.studentrollno left join studentscholarshipinformation as sscf on ssf.studentrollno=sscf.studentrollno;



1. Difference between the Triggers, Stored Procedures, Views and Functions

**Solution:-**

**Triggers:** A trigger runs automatically when an event occurs in the Database server.

1. Triggers run implicitly when an insert, update or delete is performed on a table.

2. We cannot pass any parameters in triggers.

3. They do not return any value.

**Stored Procedure:** A stored procedure is a collection of SQL statements that can be reused.

1. Stored procedure is called explicitly.

2. We can pass parameters in a stored procedure.

3. They may or may not return any value

**Views:** A view is a virtual table which may contain fields from one or more tables.

1. A view can be used in reference with the SELECT command.

2. It does not accept any parameters.

3. It is faster than stored procedures as it displays data from a table whereas stored procedure executes a series of sql commands.

**Functions:** User defined functions are a block of code that accept parameters, do some task and return a value.

1. We can call functions from a SELECT statement.

2. We can pass parameters in User defined functions.

3. Functions always return a value.