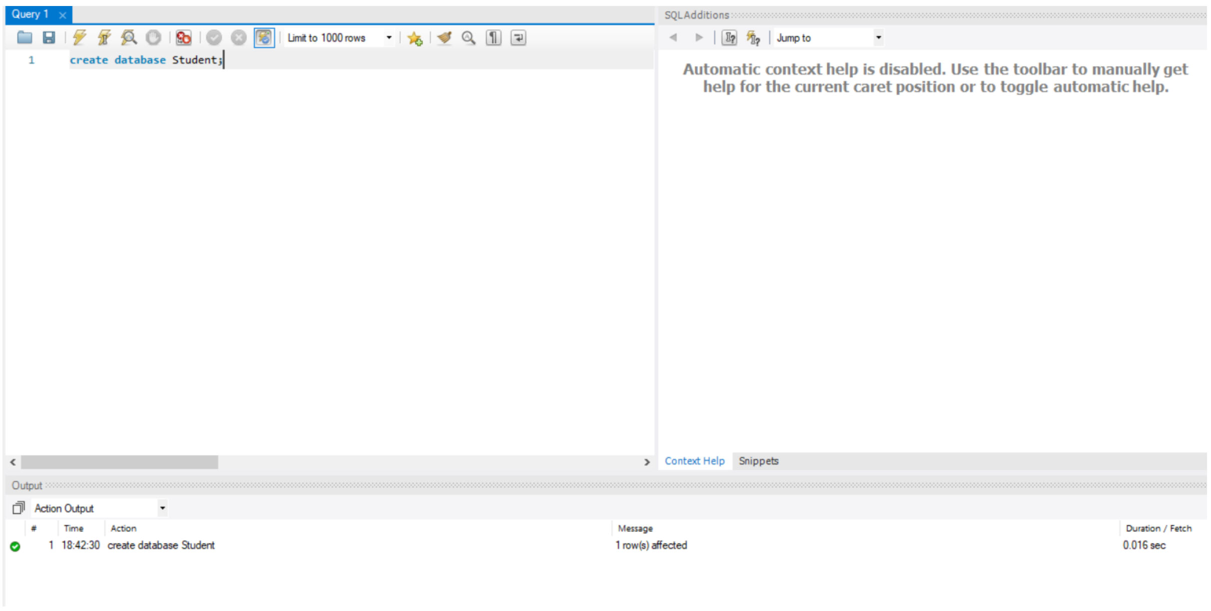
SQL Assignment – Nipun Kumar Drayan

1. **Create Student Database**

   create database Student;



```
create database Student;
```

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

```
#   Time      Action                   Message             Duration / Fetch
1   18:42:30  create database Student  1 row(s) affected   0.016 sec
```

2. **Create the following table under the Student Database:**
   a. **StudentBasicInformation**
      i. **Columns**
         1. **StudentName**
         2. **StudentSurname**
         3. **StudentRollNo**
         4. **StudentAddress**
         5. **Add more three basic columns of the name of your own**

create table StudentBasicInformation(StudentName varchar(20), StudentSurname varchar(25), StudentRollNo int primary key,StudentClass int,StudentPhoneNumber bigint, StudentGender varchar(15), StudentBloodGroup varchar(5), StudentAddress varchar(50));

   b. **StudentAdmissionPaymentDetails**
      i. **Columns**
         1. **StudentRollNo**
         2. **AmountPaid**
         3. **AmountBalance**
         4. **Add more four basic columns of the name of your own**

create table StudentAdmissionPaymentDetails (StudentRollNo int primary key, AmountPaid int, AmountBalance int, PaymentDate date, TransactionId varchar(20), AccoutNumber varchar(20) ,foreign key(StudentRollNo) referencesc StudentBasicInformation(StudentRollNo));

   c. **StudentSubjectInformation**
      i. **Columns**

1. **SubjectOpted**
2. **StudentRollNo**
3. **SubjectTotalMarks**
4. **SubjectObtainedMarks**
5. **StudentMarksPercentage**
6. **Add more one columns of the name of your own**

create table StudentSubjectInformation (SubjectOpted varchar(20), StudentRollNo int , SubjectTotalMarks int,SubjectObtainedMarks int, StudentMarksPercentage int, StudentDivsion varchar(20),foreign key(StudentRollNo) references StudentBasicInformation(StudentRollNo));

d. **SubjectScholarshipInformation**

   i. **Columns**

      1. **StudentRollNo**
      2. **ScholarshipName**
      3. **ScholarshipDescription**
      4. **ScholarshipAmount**
      5. **ScholarshipCategory**
      6. **Add more two columns of the name of your own**

create table StudentScholarshipInformation(StudentRollNo int, ScholarshipName varchar(25),ScholarshipDescription varchar(25), ScholarshipAmount int, ScholarshipCategory varchar(15), StudentScholarshipGiven varchar(20),StudentSchlarshipDuration int,foreign key(StudentRollNo) references StudentBasicInformation(StudentRollNo));

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

a) INSERT INTO StudentBasicInformation (StudentName,StudentSurname,StudentRollNo,StudentPhoneNumber,StudentClass,StudentGender,StudentBloodGroup,StudentAddress)values
("Rohan", "Goswami",15 ,9985621452, 8, "male", "B+", "Delhi"),
("Rahul", "Ojha", 8,8957451245, 12, "male", "A+", "Bihar"),
("Abhishek", "Singh", 7,945623744, 10, "male", "O-", "Bareily"),
("Vijeta", "Shekhawat", 3,999747725, 8, "female", "AB-", "Ajmer"),
("Dev", "Bharadwaj",11 ,9658214522, 8, "male", "B+", "Meerut"),
("Shwetank", "Dhruva", 7,7417657151, 12, "male", "A-", "Chattisgarh"),
("Jatin", "Jadia", 4,8569475885, 9, "male", "B-", "Bhopal"),
("Aakash", "Singh", 8,7561425865, 5, "male", "A+", "Ghaziabad"),
("Aashi", "Kumar",7 ,8860021442, 8, "female", "O+", "Mumbai"),
("Arunika", "Singhania",8 ,8695428856, 11, "female", "A+", "Hyderabad");

b) INSERT INTO StudentAdmissionPayment values
(2,15000, 1200, "Done", "2020-11-04", "Axis", "52412", "1542543245"),
(3,95000, 5200, "Done", "2020-09-13", "SBI", "45321", "85621432"),
(4,4200, 600, "Not", "2020-02-14", "Axis", "72343", "452454134"),
(6,9300, 7200, "Not", "2019-09-14", "Canara", "55234", "85124524"),
(7,7800, 820, "Done", "2019-07-12", "Axis", "52324", "4123545164"),

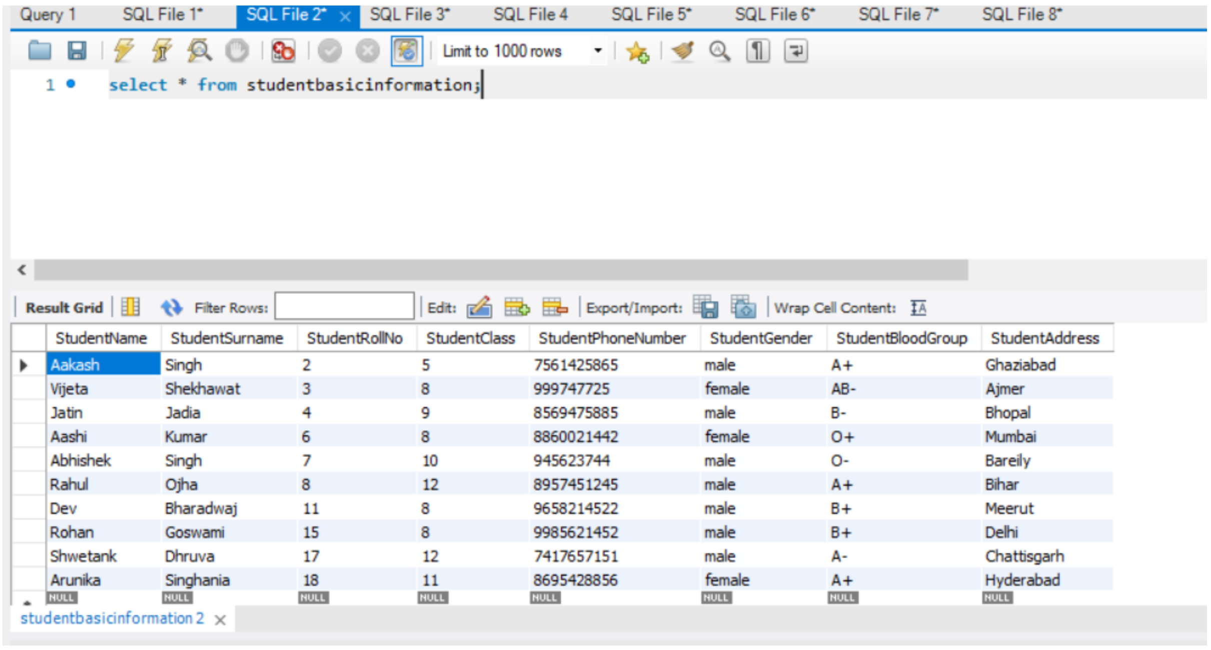
```
(8,8130, 1500, "Not", "2017-10-12", "BOB", "9235", "751244123"),
(11,3200, 2150, "Not", "2020-07-13", "Axis", "8534", "8531423243"),
(15,2900, 1540, "Done", "2019-02-01", "SBI", "42311", "95624142"),
(17,63000, 3520, "Not", "2020-01-14", "Axis", "75232", "12548732234"),
(18,75000, 850, "Done", "2014-01-01", "HDFC", "85615", "1254845532");
```

c)
```
INSERT INTO StudentSubjectInformation values
("Science", 2,1000, 800, 80, "B"),
("Commerce",3 ,1000, 990, 99, "A"),
("Science", 4,1000, 810, 81, "B"),
("Humanaties", 6,1000, 780, 78, "C"),
("Arts", 7,1000, 820, 82, "B"),
("Commerce",8,1000, 800, 80, "B"),
("Science", 11,1000, 730, 73, "C"),
("Biology",15, 1000, 980, 98, "A"),
("Computers",17,1000, 930, 93, "A"),
("Science", 18,1000, 650, 65, "D");
```

d)
```
INSERT INTO StudentScholarshipInformation values
(2,"reward", "hsdf", 3200, "OBC", "y", 1),
(3,"reward", "ysdf", 5200, "Research", "y", 1),
(4,"scheme", "jasd", 3200, "Girl", "n", 1),
(6,"funds", "fasd", 8000, "OBC", "y", 2),
(7,"funds", "fasd", 4510, "Rank", "y", 1),
(8,"reward", "fasd", 7800, "Research", "n", 4),
(11,"scheme", "fasd", 2300, "Army", "y", 3),
(15,"funds", "fsd", 5620, "Work", "n", 2),
(17,"scheme", "gdsaf", 8200, "Research", "y", 1),
(18,"funds", "fasd", 5400, "OBC", "y", 3);
```

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

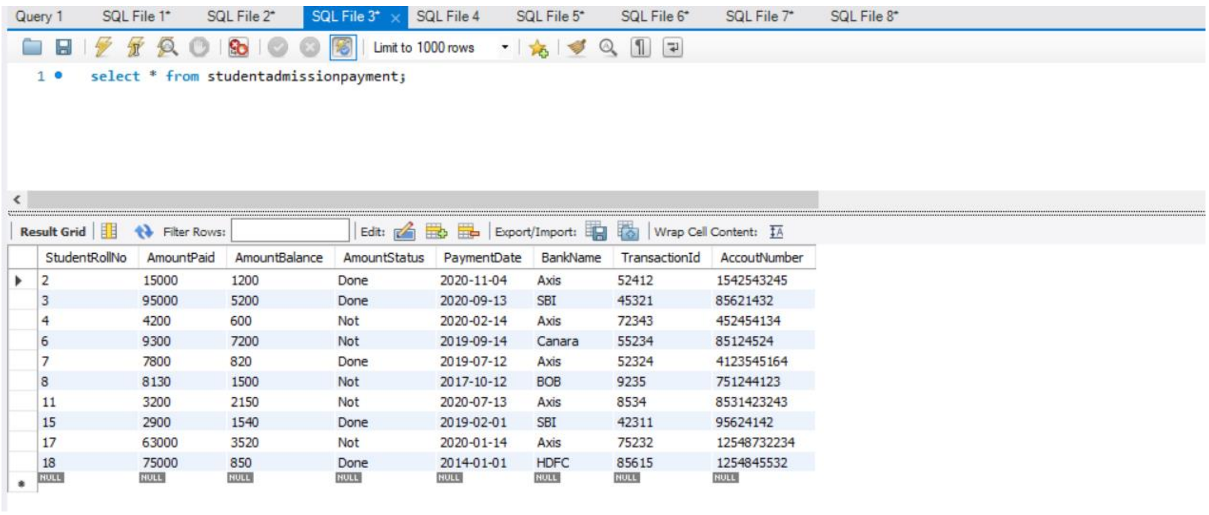
a)



```sql
select * from studentbasicinformation;
```

StudentName	StudentSurname	StudentRollNo	StudentClass	StudentPhoneNumber	StudentGender	StudentBloodGroup	StudentAddress
Aakash	Singh	2	5	7561425865	male	A+	Ghaziabad
Vijeta	Shekhawat	3	8	999747725	female	AB-	Ajmer
Jatin	Jadia	4	9	8569475885	male	B-	Bhopal
Aashi	Kumar	6	8	8860021442	female	O+	Mumbai
Abhishek	Singh	7	10	945623744	male	O-	Bareily
Rahul	Ojha	8	12	8957451245	male	A+	Bihar
Dev	Bharadwaj	11	8	9658214522	male	B+	Meerut
Rohan	Goswami	15	8	9985621452	male	B+	Delhi
Shwetank	Dhruva	17	12	7417657151	male	A-	Chattisgarh
Arunika	Singhania	18	11	8695428856	female	A+	Hyderabad
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

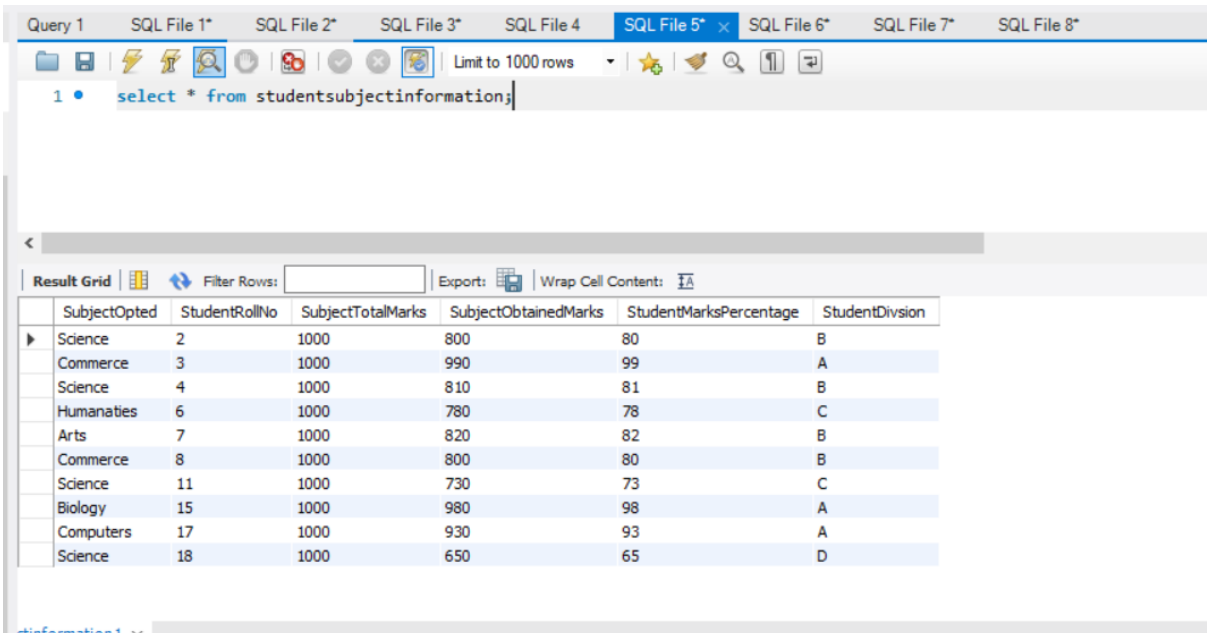
b)



```sql
select * from studentadmissionpayment;
```

StudentRollNo	AmountPaid	AmountBalance	AmountStatus	PaymentDate	BankName	TransactionId	AccoutNumber
2	15000	1200	Done	2020-11-04	Axis	52412	1542543245
3	95000	5200	Done	2020-09-13	SBI	45321	85621432
4	4200	600	Not	2020-02-14	Axis	72343	452454134
6	9300	7200	Not	2019-09-14	Canara	55234	85124524
7	7800	820	Done	2019-07-12	Axis	52324	4123545164
8	8130	1500	Not	2017-10-12	BOB	9235	751244123
11	3200	2150	Not	2020-07-13	Axis	8534	8531423243
15	2900	1540	Done	2019-02-01	SBI	42311	95624142
17	63000	3520	Not	2020-01-14	Axis	75232	12548732234
18	75000	850	Done	2014-01-01	HDFC	85615	1254845532
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

c)


Query 1 | SQL File 1* | SQL File 2* | SQL File 3* | SQL File 4 | SQL File 5* | SQL File 6* | SQL File 7* | SQL File 8*

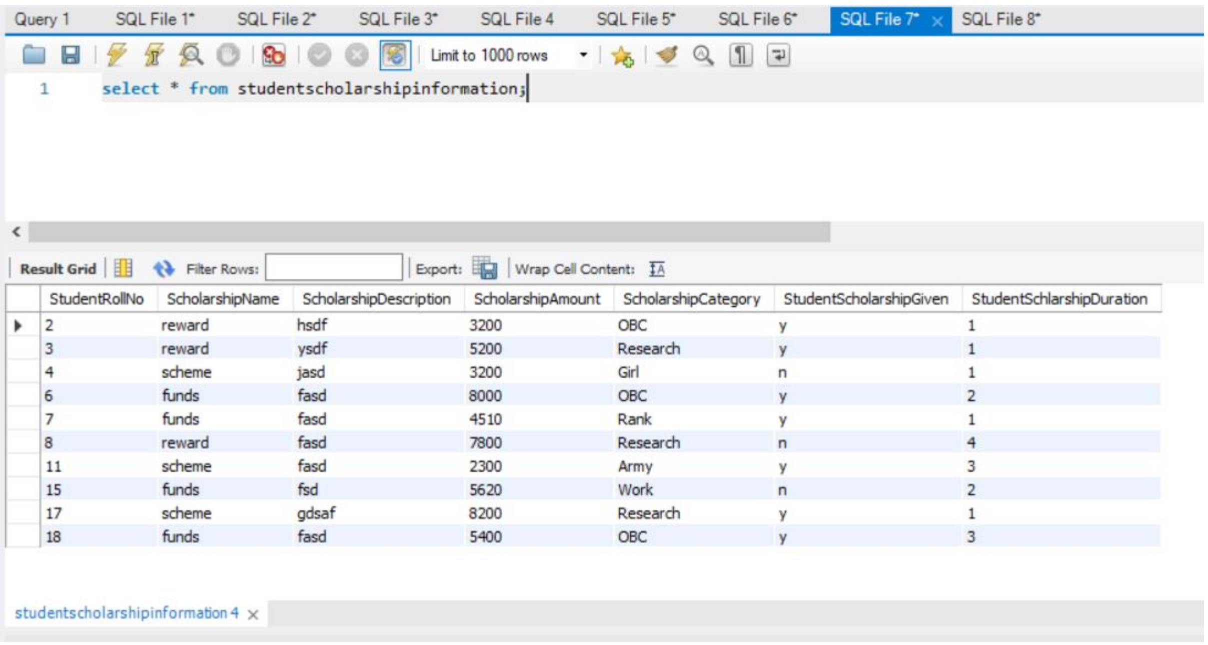
Limit to 1000 rows

```
select * from studentsubjectinformation;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

SubjectOpted	StudentRollNo	SubjectTotalMarks	SubjectObtainedMarks	StudentMarksPercentage	StudentDivsion
Science	2	1000	800	80	B
Commerce	3	1000	990	99	A
Science	4	1000	810	81	B
Humanaties	6	1000	780	78	C
Arts	7	1000	820	82	B
Commerce	8	1000	800	80	B
Science	11	1000	730	73	C
Biology	15	1000	980	98	A
Computers	17	1000	930	93	A
Science	18	1000	650	65	D

d)


Query 1 | SQL File 1* | SQL File 2* | SQL File 3* | SQL File 4 | SQL File 5* | SQL File 6* | SQL File 7* | SQL File 8*

Limit to 1000 rows

```
select * from studentscholarshipinformation;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

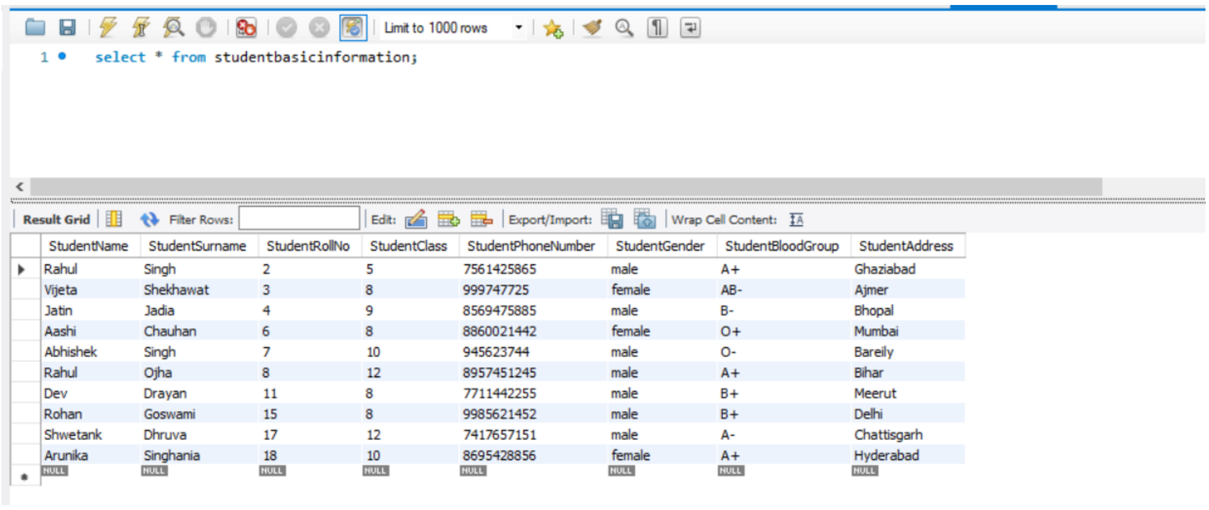
StudentRollNo	ScholarshipName	ScholarshipDescription	ScholarshipAmount	ScholarshipCategory	StudentScholarshipGiven	StudentSchlarshipDuration
2	reward	hsdf	3200	OBC	y	1
3	reward	ysdf	5200	Research	y	1
4	scheme	jasd	3200	Girl	n	1
6	funds	fasd	8000	OBC	y	2
7	funds	fasd	4510	Rank	y	1
8	reward	fasd	7800	Research	n	4
11	scheme	fasd	2300	Army	y	3
15	funds	fsd	5620	Work	n	2
17	scheme	gdsaf	8200	Research	y	1
18	funds	fasd	5400	OBC	y	3

studentscholarshipinformation 4

5. **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**

update studentbasicinformation set StudentSurname="Chauhan" where StudentRollNo=6;
update studentbasicinformation set StudentName="Rahul" where StudentRollNo=2;
update studentbasicinformation set StudentPhoneNumber=7711442255 where StudentName="Dev";
update studentbasicinformation set StudentClass=10 where StudentRollNo=18;
update studentbasicinformation set StudentSurname="Drayan" where StudentRollNo=11;

6. **Snap of the all the tables post updation**

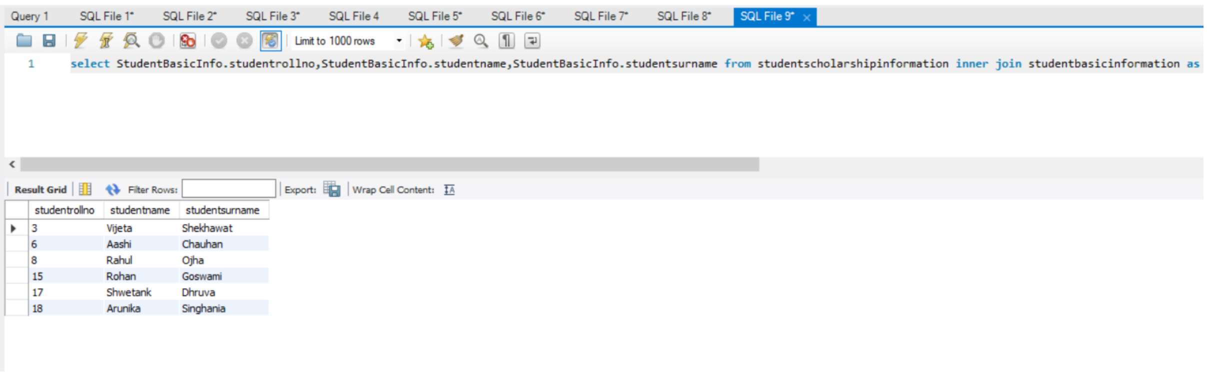


```
select * from studentbasicinformation;
```

StudentName	StudentSurname	StudentRollNo	StudentClass	StudentPhoneNumber	StudentGender	StudentBloodGroup	StudentAddress
Rahul	Singh	2	5	7561425865	male	A+	Ghaziabad
Vijeta	Shekhawat	3	8	999747725	female	AB-	Ajmer
Jatin	Jadia	4	9	8569475885	male	B-	Bhopal
Aashi	Chauhan	6	8	8860021442	female	O+	Mumbai
Abhishek	Singh	7	10	945623744	male	O-	Bareily
Rahul	Ojha	8	12	8957451245	male	A+	Bihar
Dev	Drayan	11	8	7711442255	male	B+	Meerut
Rohan	Goswami	15	8	9985621452	male	B+	Delhi
Shwetank	Dhruva	17	12	7417657151	male	A-	Chattisgarh
Arunika	Singhania	18	10	8695428856	female	A+	Hyderabad
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

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

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

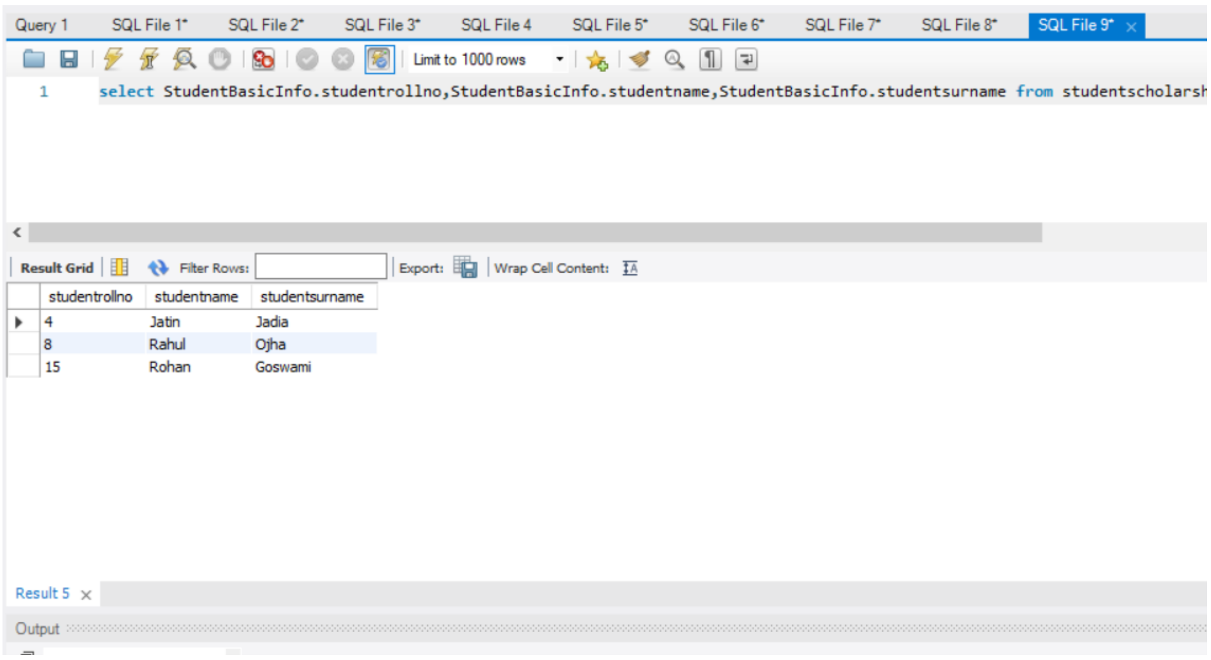


```
select StudentBasicInfo.studentrollno,StudentBasicInfo.studentname,StudentBasicInfo.studentsurname from studentscholarshipinformation inner join studentbasicinformation as
```

studentrollno	studentname	studentsurname
3	Vijeta	Shekhawat
6	Aashi	Chauhan
8	Rahul	Ojha
15	Rohan	Goswami
17	Shwetank	Dhruva
18	Arunika	Singhania

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

select StudentBasicInfo.studentrollno,StudentBasicInfo.studentname,StudentBasicInfo.studentsurname from studentscholarshipinformation inner join studentbasicinformation as StudentBasicInfo on studentscholarshipinformation.studentrollno = StudentBasicInfo.studentrollno where studentscholarshipgiven="n" order by studentrollno;



```
select StudentBasicInfo.studentrollno,StudentBasicInfo.studentname,StudentBasicInfo.studentsurname from studentscholarsh
```

studentrollno	studentname	studentsurname
4	Jatin	Jadia
8	Rahul	Ojha
15	Rohan	Goswami

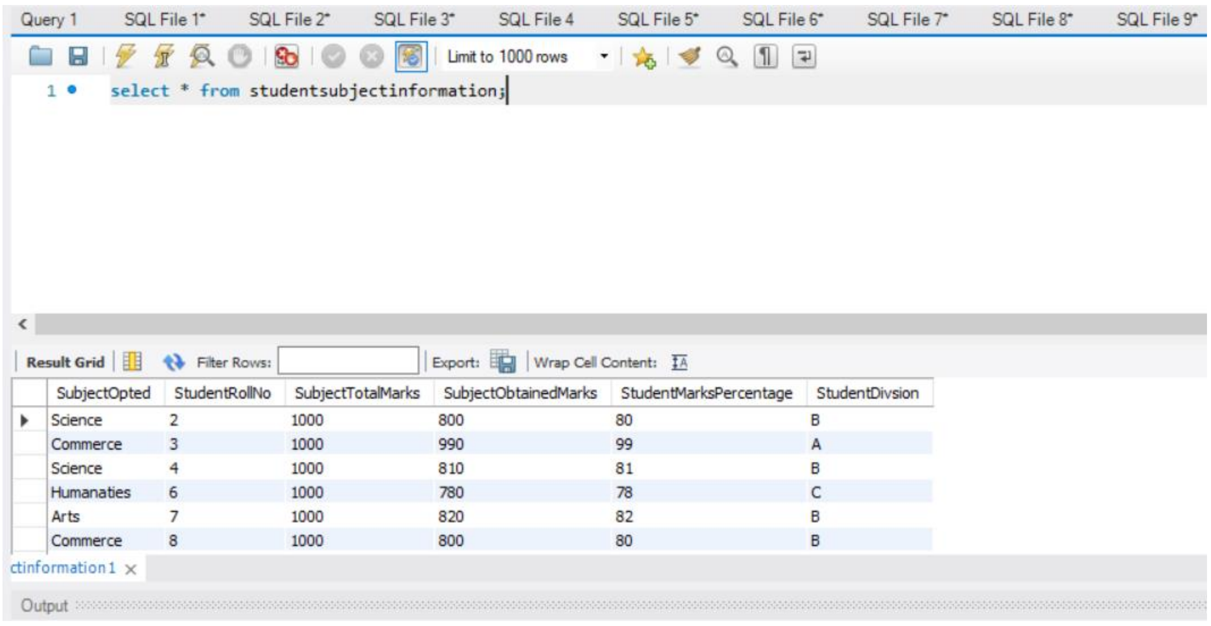
9. Fill in data for the percentage column i.e. StudentMarksPercentage in the table StudentSubjectInformation by creating and using the stored procedure created

```
delimiter //
create procedure CalculatePercen()
begin
update studentsubjectinformation set
studentmarkspercentage=100*(subjectobtainedmarks/subjecttotalmarks);
end //

delimiter ;

call CalculatePercen();

select * from studentsubjectinformation;
```



```
select * from studentsubjectinformation;
```

SubjectOpted	StudentRollNo	SubjectTotalMarks	SubjectObtainedMarks	StudentMarksPercentage	StudentDivsion
Science	2	1000	800	80	B
Commerce	3	1000	990	99	A
Science	4	1000	810	81	B
Humanaties	6	1000	780	78	C
Arts	7	1000	820	82	B
Commerce	8	1000	800	80	B

10. 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

```
delimiter //
create procedure Categorize()
begin
declare sum int;
declare counter int;
declare rollno int;
declare percentage int;
select count(*) into sum from studentscholarshipinformation;
set counter=0;
while counter<sum do
select studentrollno into rollno from studentscholarshipinformation limit counter,1;
select studentmarkspercentagecentage into percentage from studentsubjectinformation
where studentrollno=rollno;
if percentage >80 then
update studentscholarshipinformation set ScholarshipCategory="Category A" where
studentrollno = rollno;
elseif percentage >50 and percentage <=80 then
update studentscholarshipinformation set ScholarshipCategory="Category B" where
studentrollno = rollno;
elseif percentage>0 and percentage<=50 then
update studentscholarshipinformation set ScholarshipCategory="Category C" where
studentrollno = rollno;
end if;
set counter = counter+1;
end while;
end //
delimiter ;
```

```
call Categorize();
```

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

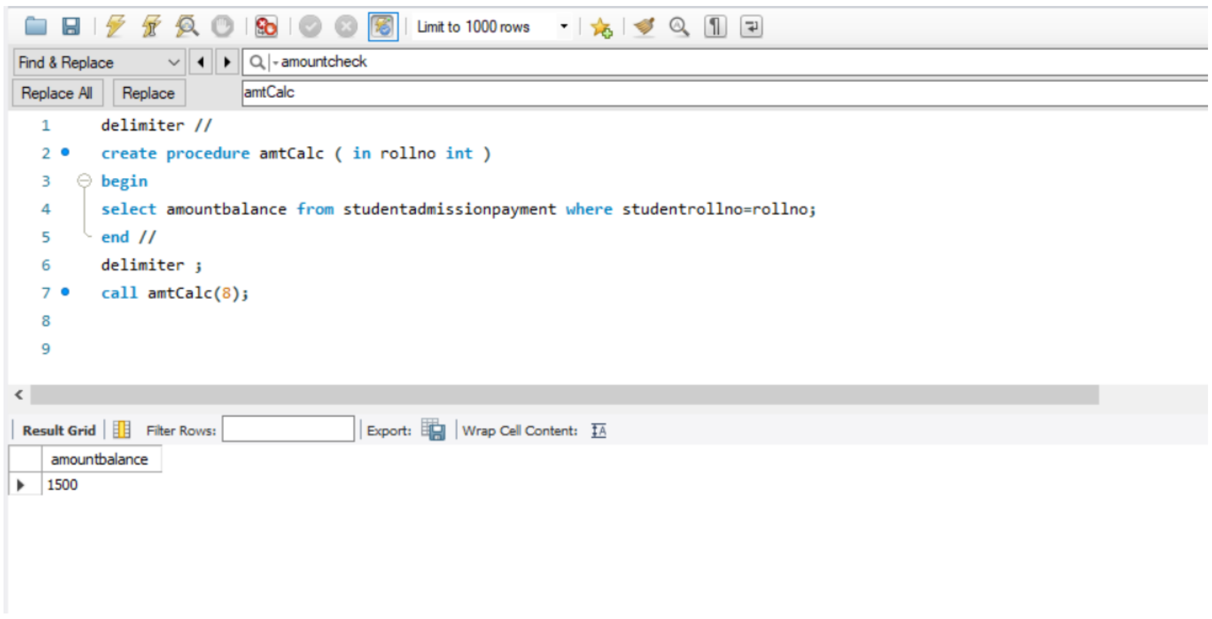
```
create or replace view Amount as select
StudentBasicInfo.studentrollno,StudentBasicInfo.studentname,StudentBasicInfo.studentsur
name,calc.amountpaid,calc.amountbalance from studentadmissionpayment as calc inner
join studentbasicinformation as StudentBasicInfo on
calc.studentrollno=StudentBasicInfo.studentrollno where calc.amountstatus="unpaid";
```

12. **Get the details of the students who haven't got any scholarship (use joins/subqueries)**

```
select
StudentBasiciInfo.studentrollno,StudentBasiciInfo.studentname,StudentBasiciInfo.studentsu
rname from studentbasicinformation as StudentBasiciInfo where
StudentBasiciInfo.studentrollno not in (select studentrollno from
studentscholarshipinformation);
```

13. **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**

```
delimiter //
create procedure amtCalc ( in rollno int )
begin
select amountbalance from studentadmissionpayment where studentrollno=rollno;
end //
delimiter ;
call amtCalc(8);
```



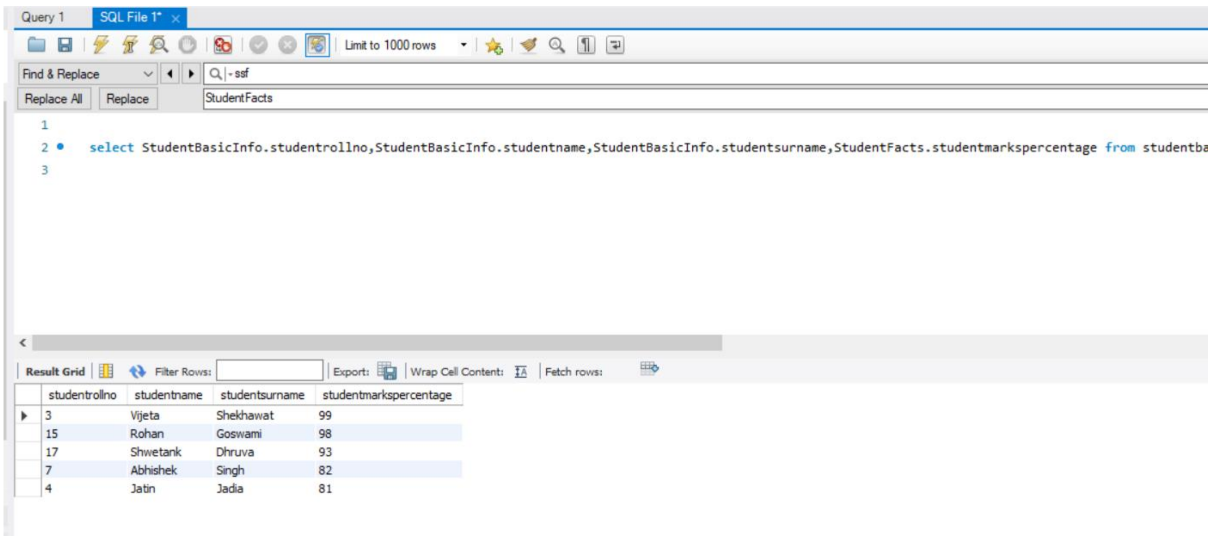
```
amountcheck
amtCalc
```

```sql
delimiter //
create procedure amtCalc ( in rollno int )
begin
select amountbalance from studentadmissionpayment where studentrollno=rollno;
end //
delimiter ;
call amtCalc(8);
```

```
amountbalance
1500
```

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

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



```
ssf
StudentFacts
```

```sql
select StudentBasicInfo.studentrollno,StudentBasicInfo.studentname,StudentBasicInfo.studentsurname,StudentFacts.studentmarkspercentage from studentba
```

```
studentrollno  studentname  studentsurname  studentmarkspercentage
3              Vijeta       Shekhawat       99
15             Rohan        Goswami         98
17             Shwetank     Dhruva          93
7              Abhishek     Singh           82
4              Jatin        Jadia           81
```

15. **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)**

    a) INNER JOIN

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

Select * from lowgradestudentdetails;
This JOIN is used so that we can find out the details of all those student and faculty information which are been connected with each other. Meaning that all those student that has been assigned with a faculty, Similarily vice versa.

b) Left Outer Join

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

Select * from scholarshipdetails;

This JOIN is used to find out the information of all the student and whether they have been given their scholarship or not. If for a student the values of scholarship are NULL then it is understood that student is not given the scholarship.

c) Right Outer Join

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

Select * from scholarshipvaliditydetails;

This JOIN is used to find out the information of all the students that whether they have paid their fees or not. If NULL values are present in payment columns then that student have not paid his/her fee.

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

● DELETE

It is used to delte the rows, we can delete row by row or all the rows in one go. The space for the records remain in the database. And we can insert the values again in that table.

- **TRUNCATE**

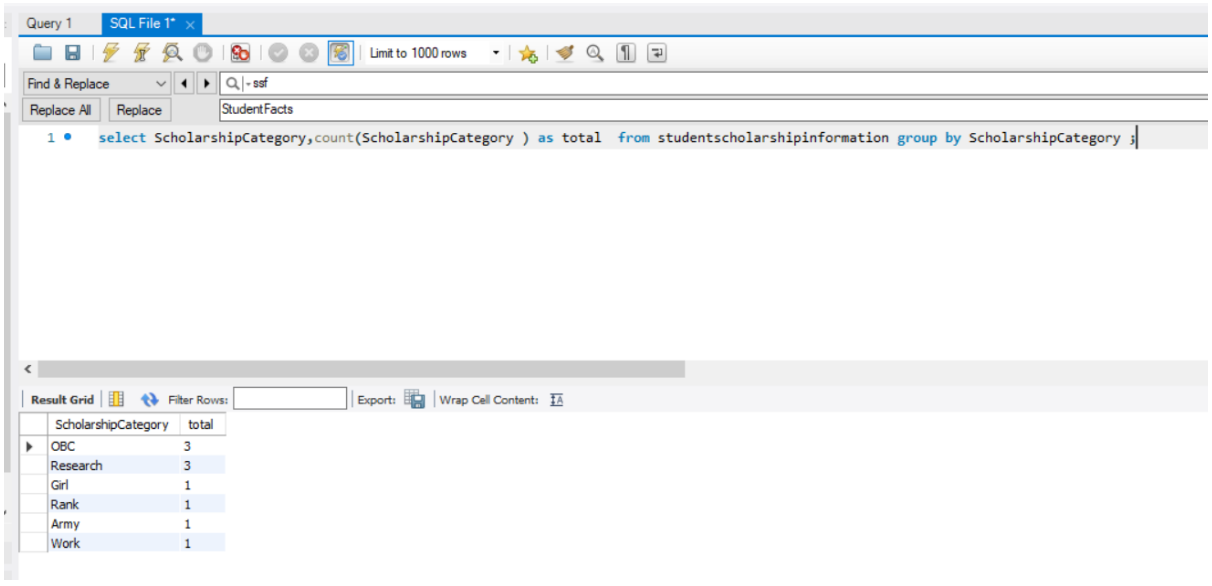
Remove all records from a table, including all spaces allocated for the records are removed and we can not insert the values again. For that we have to again do the DDL part.

- **DROP**

It is used to delete objects from the database.

17. 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

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

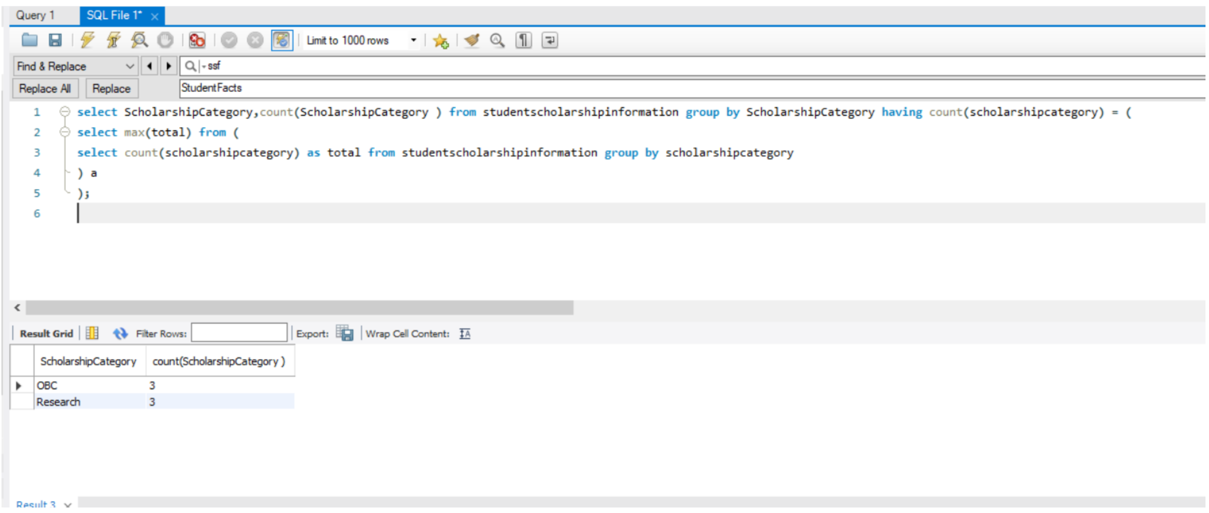


```
select ScholarshipCategory,count(ScholarshipCategory ) as total  from studentscholarshipinformation group by ScholarshipCategory ;
```

ScholarshipCategory	total
OBC	3
Research	3
Girl	1
Rank	1
Army	1
Work	1

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

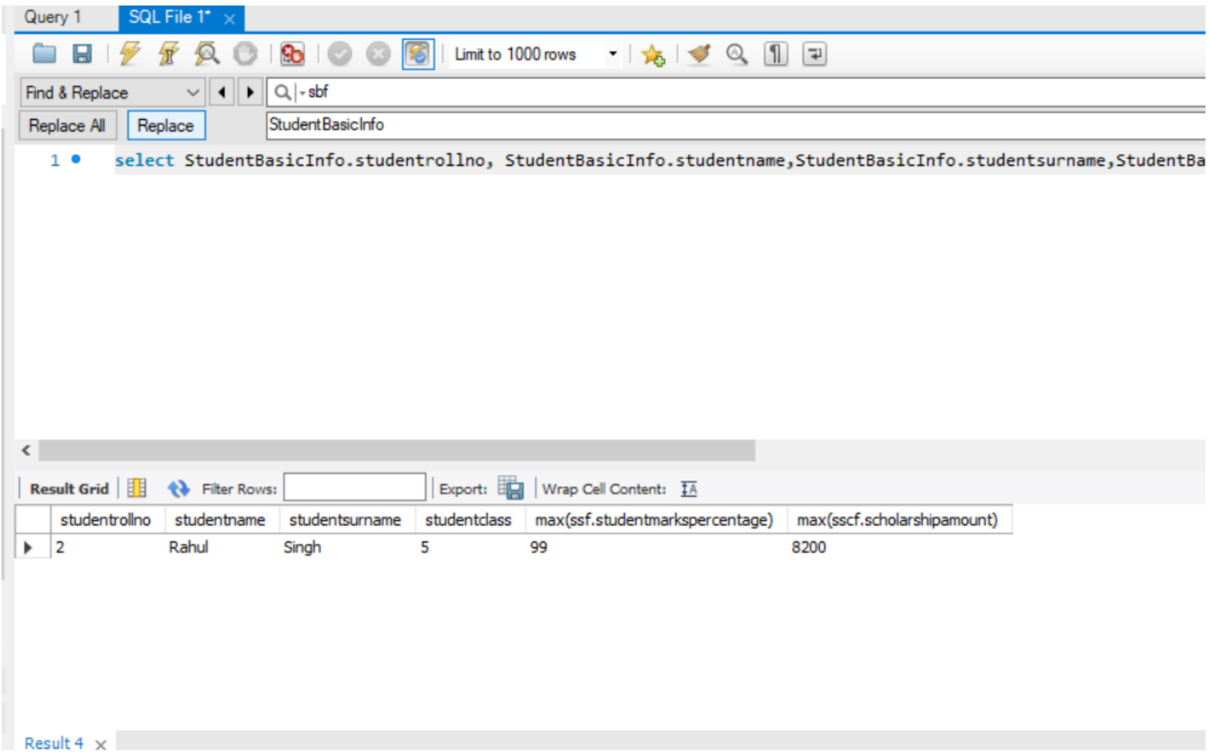
select ScholarshipCategory,count(ScholarshipCategory) from studentscholarshipinformation group by ScholarshipCategory having count(scholarshipcategory) = (
select max(total) from (
select count(scholarshipcategory) as total from studentscholarshipinformation group by scholarshipcategory
) a
);



```
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
);
```

ScholarshipCategory	count(ScholarshipCategory)
OBC	3
Research	3

19. **Retrieve the percentage of the students along with students detailed information who has scored the highest percentage along with availing the maximum scholarship amount**
    select StudentBasicInfo.studentrollno, StudentBasicInfo.studentname,StudentBasicInfo.studentsurname,StudentBasicInfo.studentclass,max(StudentFacts.studentmarkspercentage),max(sscf.scholarshipamount) from studentbasicinformation as StudentBasicInfo left join studentsubjectinformation as StudentFacts on StudentBasicInfo.studentrollno=StudentFacts.studentrollno left join studentscholarshipinformation as sscf on StudentFacts.studentrollno=sscf.studentrollno;



```
select StudentBasicInfo.studentrollno, StudentBasicInfo.studentname,StudentBasicInfo.studentsurname,StudentBa
```

studentrollno	studentname	studentsurname	studentclass	max(ssf.studentmarkspercentage)	max(sscf.scholarshipamount)
2	Rahul	Singh	5	99	8200

20. **Difference between the Triggers, Stored Procedures, Views and Functions**
    - **TRIGGERS**

    A trigger is a program which is called automatically on occuring of any type of event such as insert, update, or delete. For example, you can define a trigger that is invoked automatically before a new row is inserted into a table.
    - **STORED PROCEDURE**

It is an SQL code that is defined by the user such that it can be used again and again. So if you have an SQL query that you write over and over again, save it as a stored procedure, and then just call it to execute it.

● **VIEW**

Suppose if we have a query which is to be used again and again. And that query is computationally very expensive to be computed. So instead of executing that query again and again, we can save that query as a temporary table. Such that if have to use that query we can simply access it from that temporary table which is called view.

● **FUNCTIONS**

A function is a stored program that you can pass parameters(if required) into and then return a value. We have many in-built functions also like aggregate funtions , date functions and many more

---

Thank you.