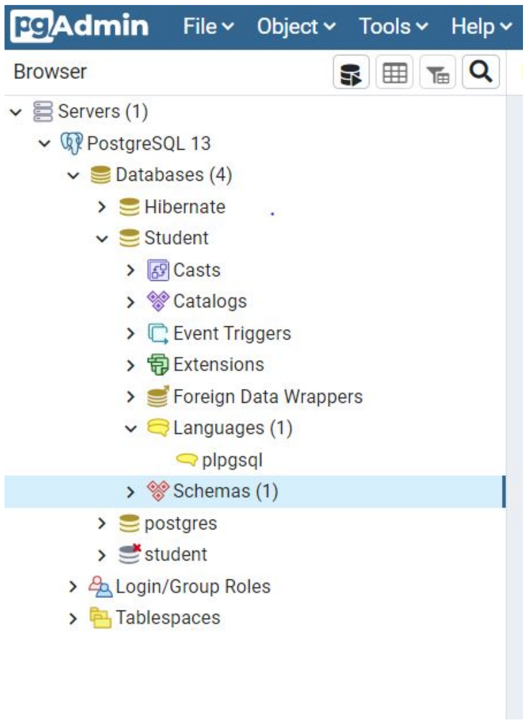
Name:- Abhishek Kumar Sah
Email id:- abhishek.sah@accolitedigital.com

Prior Instructions

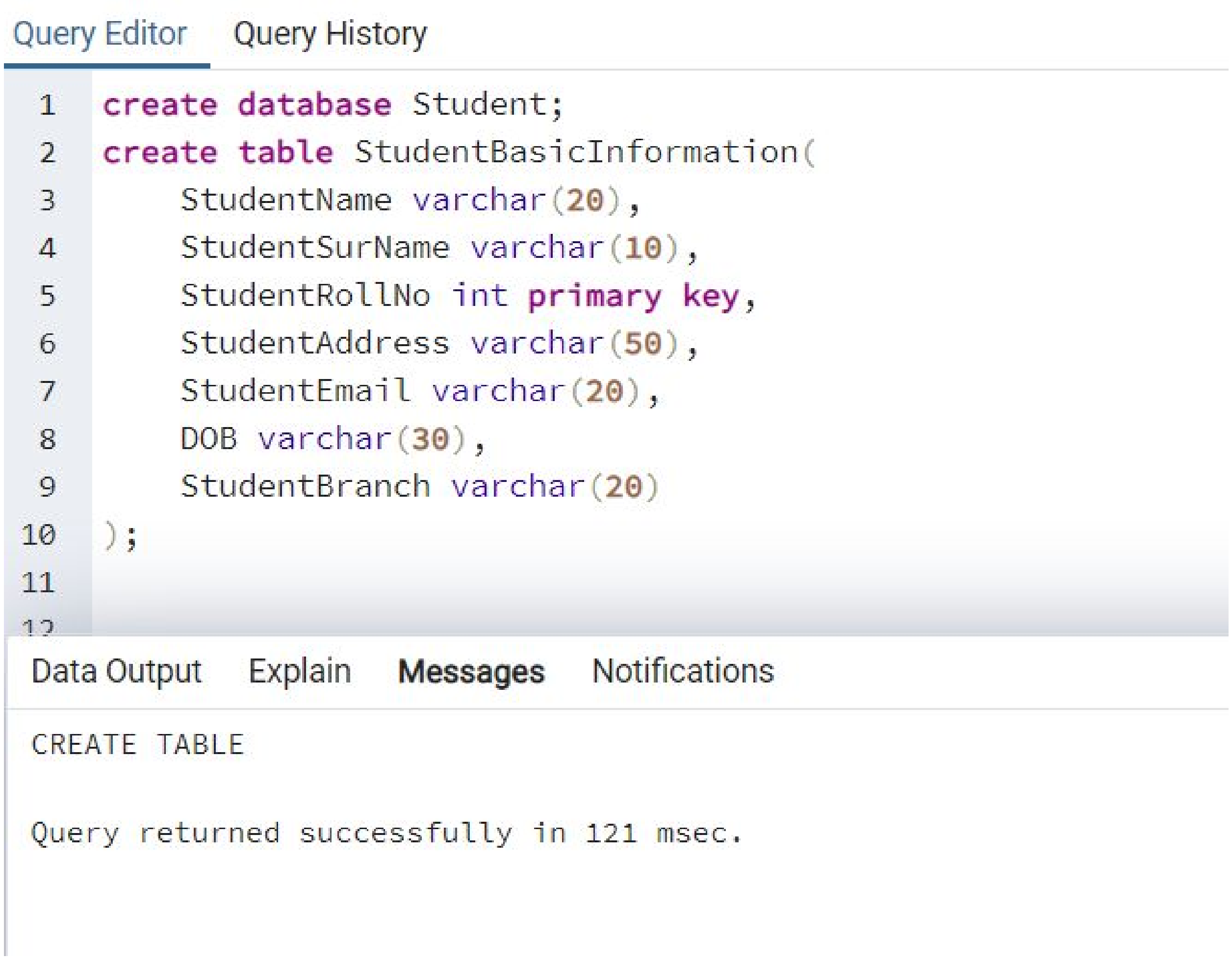
- **Please do read all the questions before performing any operations in the database**
- **Once you have fully gone through the questions then likewise decide the contents and table columns and follow the below instructions**

---

1. Create Student Database



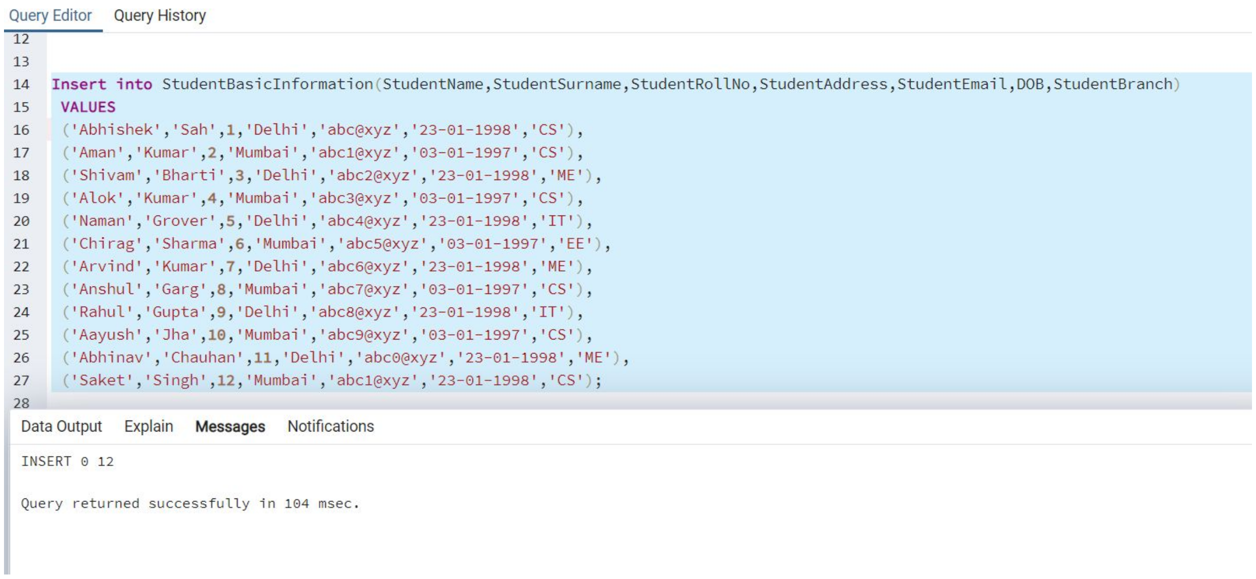
2. Create the following table under the Student Database:
    a. StudentBasicInformation
        i. Columns
            1. StudentName
            2. StudentSurname
            3. StudentRollNo
            4. StudentAddress
            5. Add more three basic columns of the name of your own



```sql
create database Student;
create table StudentBasicInformation(
    StudentName varchar(20),
    StudentSurName varchar(10),
    StudentRollNo int primary key,
    StudentAddress varchar(50),
    StudentEmail varchar(20),
    DOB varchar(30),
    StudentBranch varchar(20)
);
```

```
CREATE TABLE

Query returned successfully in 121 msec.
```

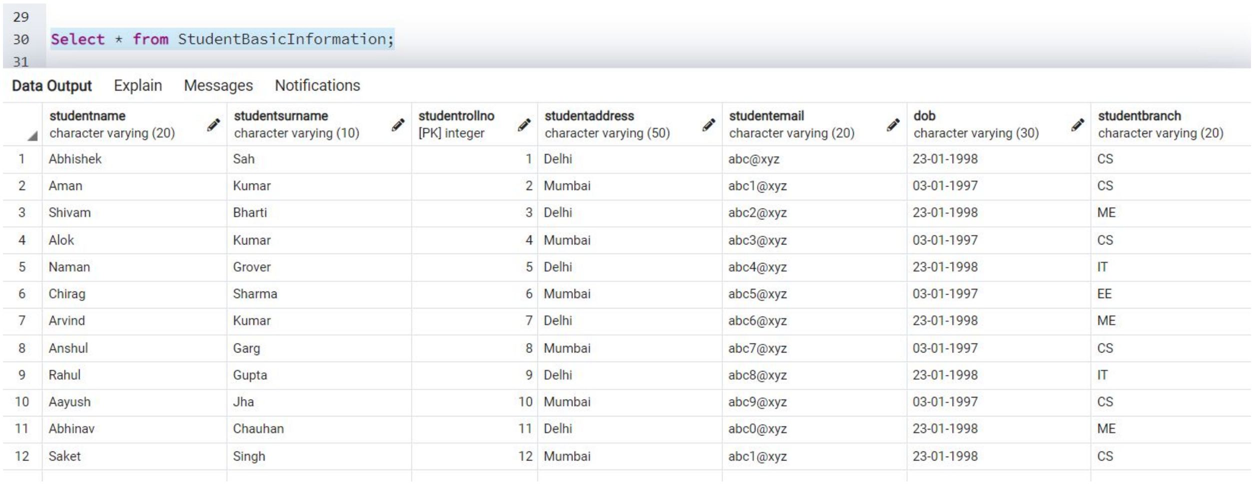


```sql
Insert into StudentBasicInformation(StudentName,StudentSurname,StudentRollNo,StudentAddress,StudentEmail,DOB,StudentBranch)
 VALUES
 ('Abhishek','Sah',1,'Delhi','abc@xyz','23-01-1998','CS'),
 ('Aman','Kumar',2,'Mumbai','abc1@xyz','03-01-1997','CS'),
 ('Shivam','Bharti',3,'Delhi','abc2@xyz','23-01-1998','ME'),
 ('Alok','Kumar',4,'Mumbai','abc3@xyz','03-01-1997','CS'),
 ('Naman','Grover',5,'Delhi','abc4@xyz','23-01-1998','IT'),
 ('Chirag','Sharma',6,'Mumbai','abc5@xyz','03-01-1997','EE'),
 ('Arvind','Kumar',7,'Delhi','abc6@xyz','23-01-1998','ME'),
 ('Anshul','Garg',8,'Mumbai','abc7@xyz','03-01-1997','CS'),
 ('Rahul','Gupta',9,'Delhi','abc8@xyz','23-01-1998','IT'),
 ('Aayush','Jha',10,'Mumbai','abc9@xyz','03-01-1997','CS'),
 ('Abhinav','Chauhan',11,'Delhi','abc0@xyz','23-01-1998','ME'),
 ('Saket','Singh',12,'Mumbai','abc1@xyz','23-01-1998','CS');
```

```
INSERT 0 12

Query returned successfully in 104 msec.
```

Table Snap:-

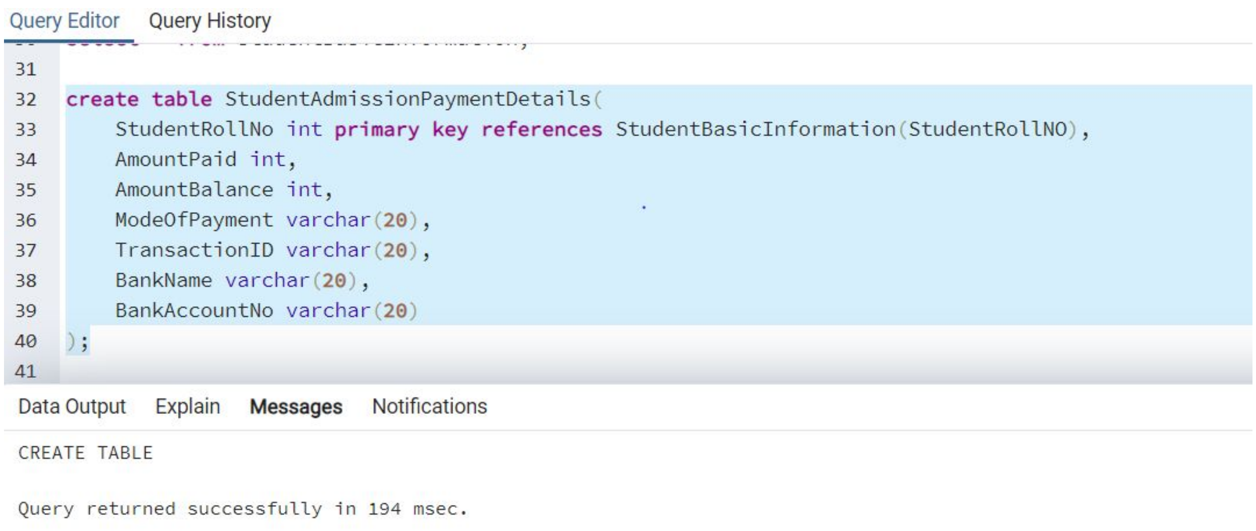


```
Select * from StudentBasicInformation;
```

Data Output Explain Messages Notifications

	studentname character varying (20)	studentsurname character varying (10)	studentrollno [PK] integer	studentaddress character varying (50)	studentemail character varying (20)	dob character varying (30)	studentbranch character varying (20)
1	Abhishek	Sah	1	Delhi	abc@xyz	23-01-1998	CS
2	Aman	Kumar	2	Mumbai	abc1@xyz	03-01-1997	CS
3	Shivam	Bharti	3	Delhi	abc2@xyz	23-01-1998	ME
4	Alok	Kumar	4	Mumbai	abc3@xyz	03-01-1997	CS
5	Naman	Grover	5	Delhi	abc4@xyz	23-01-1998	IT
6	Chirag	Sharma	6	Mumbai	abc5@xyz	03-01-1997	EE
7	Arvind	Kumar	7	Delhi	abc6@xyz	23-01-1998	ME
8	Anshul	Garg	8	Mumbai	abc7@xyz	03-01-1997	CS
9	Rahul	Gupta	9	Delhi	abc8@xyz	23-01-1998	IT
10	Aayush	Jha	10	Mumbai	abc9@xyz	03-01-1997	CS
11	Abhinav	Chauhan	11	Delhi	abc0@xyz	23-01-1998	ME
12	Saket	Singh	12	Mumbai	abc1@xyz	23-01-1998	CS

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



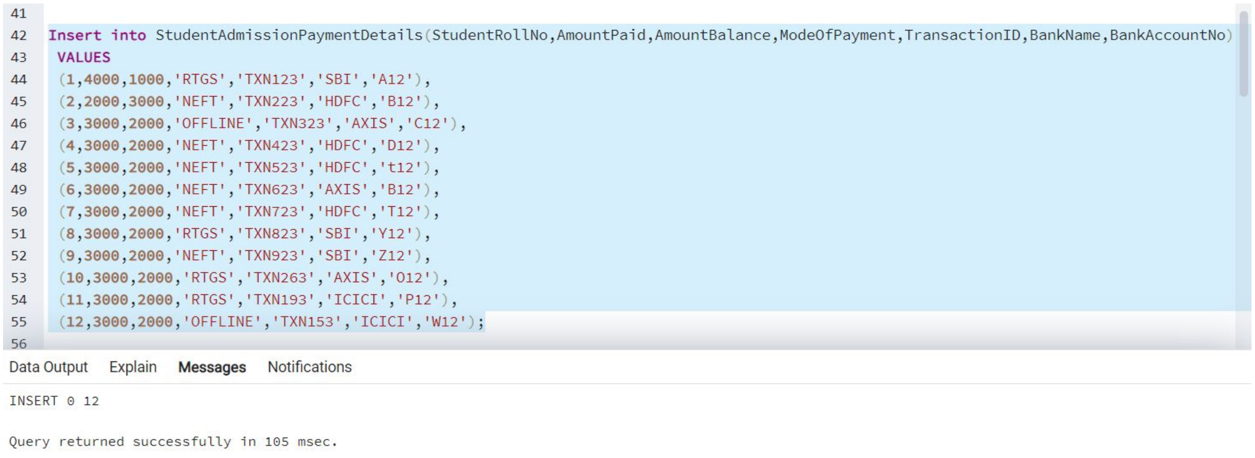
Query Editor Query History

```
create table StudentAdmissionPaymentDetails(
    StudentRollNo int primary key references StudentBasicInformation(StudentRollNO),
    AmountPaid int,
    AmountBalance int,
    ModeOfPayment varchar(20),
    TransactionID varchar(20),
    BankName varchar(20),
    BankAccountNo varchar(20)
);
```

Data Output Explain Messages Notifications

```
CREATE TABLE

Query returned successfully in 194 msec.
```



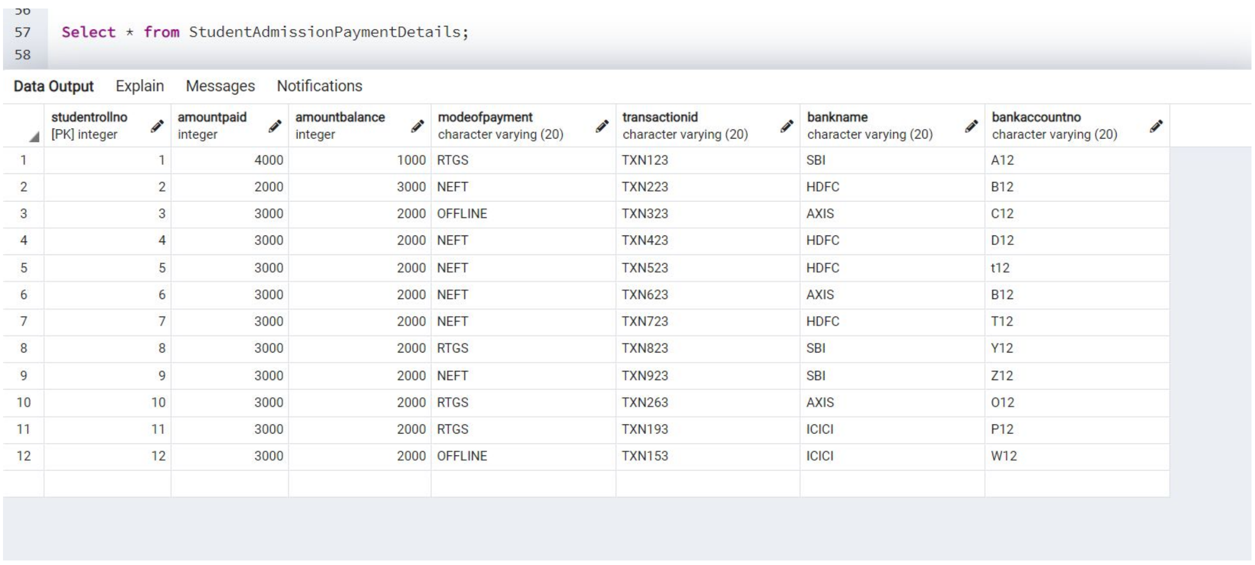
```
Insert into StudentAdmissionPaymentDetails(StudentRollNo,AmountPaid,AmountBalance,ModeOfPayment,TransactionID,BankName,BankAccountNo)
 VALUES
 (1,4000,1000,'RTGS','TXN123','SBI','A12'),
 (2,2000,3000,'NEFT','TXN223','HDFC','B12'),
 (3,3000,2000,'OFFLINE','TXN323','AXIS','C12'),
 (4,3000,2000,'NEFT','TXN423','HDFC','D12'),
 (5,3000,2000,'NEFT','TXN523','HDFC','t12'),
 (6,3000,2000,'NEFT','TXN623','AXIS','B12'),
 (7,3000,2000,'NEFT','TXN723','HDFC','T12'),
 (8,3000,2000,'RTGS','TXN823','SBI','Y12'),
 (9,3000,2000,'NEFT','TXN923','SBI','Z12'),
 (10,3000,2000,'RTGS','TXN263','AXIS','O12'),
 (11,3000,2000,'RTGS','TXN193','ICICI','P12'),
 (12,3000,2000,'OFFLINE','TXN153','ICICI','W12');
```

Data Output Explain Messages Notifications

```
INSERT 0 12

Query returned successfully in 105 msec.
```

Table Snap:-

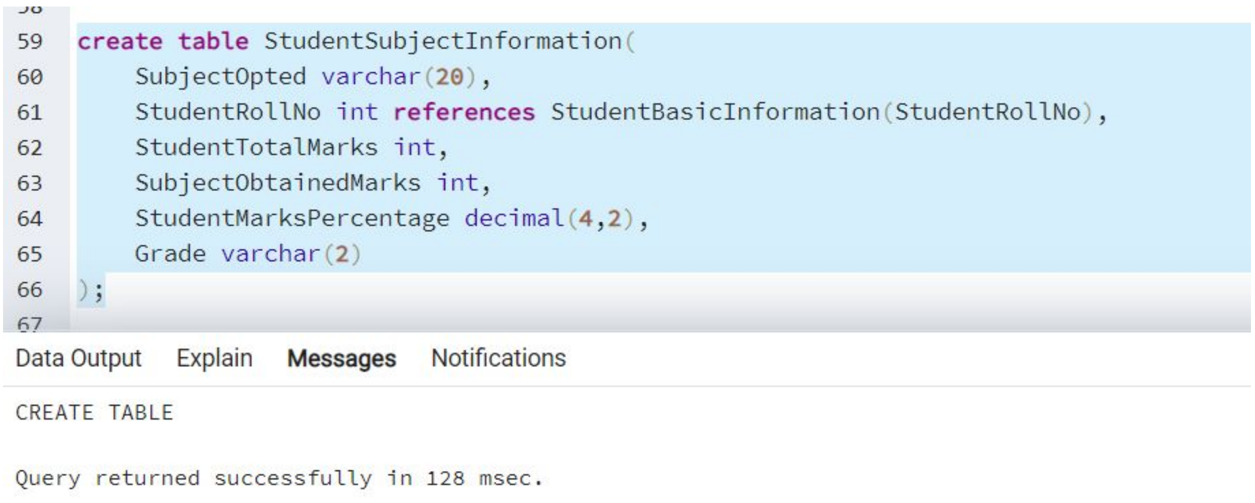


```
Select * from StudentAdmissionPaymentDetails;
```

Data Output Explain Messages Notifications

	studentrollno [PK] integer	amountpaid integer	amountbalance integer	modeofpayment character varying (20)	transactionid character varying (20)	bankname character varying (20)	bankaccountno character varying (20)
1	1	4000	1000	RTGS	TXN123	SBI	A12
2	2	2000	3000	NEFT	TXN223	HDFC	B12
3	3	3000	2000	OFFLINE	TXN323	AXIS	C12
4	4	3000	2000	NEFT	TXN423	HDFC	D12
5	5	3000	2000	NEFT	TXN523	HDFC	t12
6	6	3000	2000	NEFT	TXN623	AXIS	B12
7	7	3000	2000	NEFT	TXN723	HDFC	T12
8	8	3000	2000	RTGS	TXN823	SBI	Y12
9	9	3000	2000	NEFT	TXN923	SBI	Z12
10	10	3000	2000	RTGS	TXN263	AXIS	O12
11	11	3000	2000	RTGS	TXN193	ICICI	P12
12	12	3000	2000	OFFLINE	TXN153	ICICI	W12

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

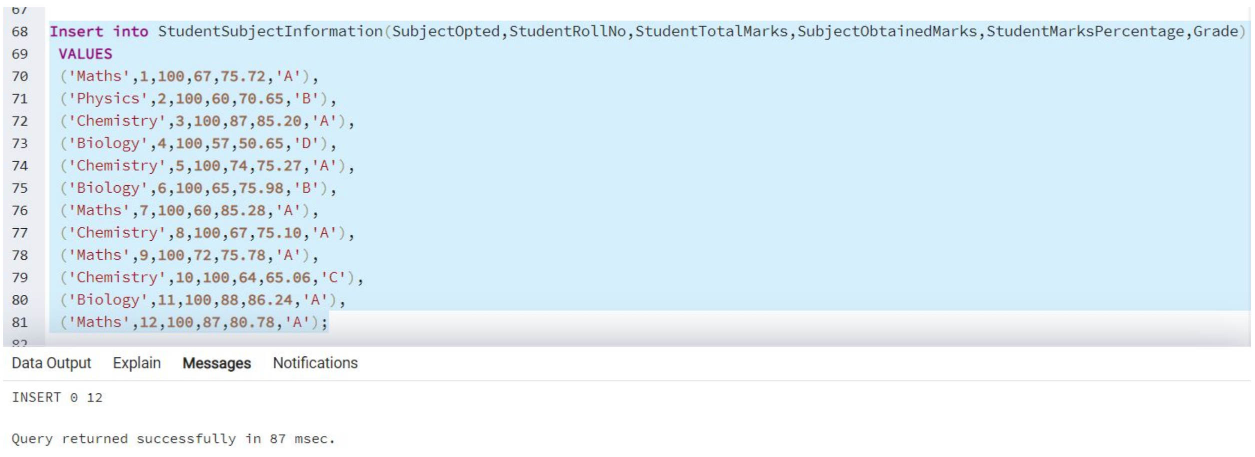


```
create table StudentSubjectInformation(
    SubjectOpted varchar(20),
    StudentRollNo int references StudentBasicInformation(StudentRollNo),
    StudentTotalMarks int,
    SubjectObtainedMarks int,
    StudentMarksPercentage decimal(4,2),
    Grade varchar(2)
);
```

Data Output Explain Messages Notifications

```
CREATE TABLE

Query returned successfully in 128 msec.
```



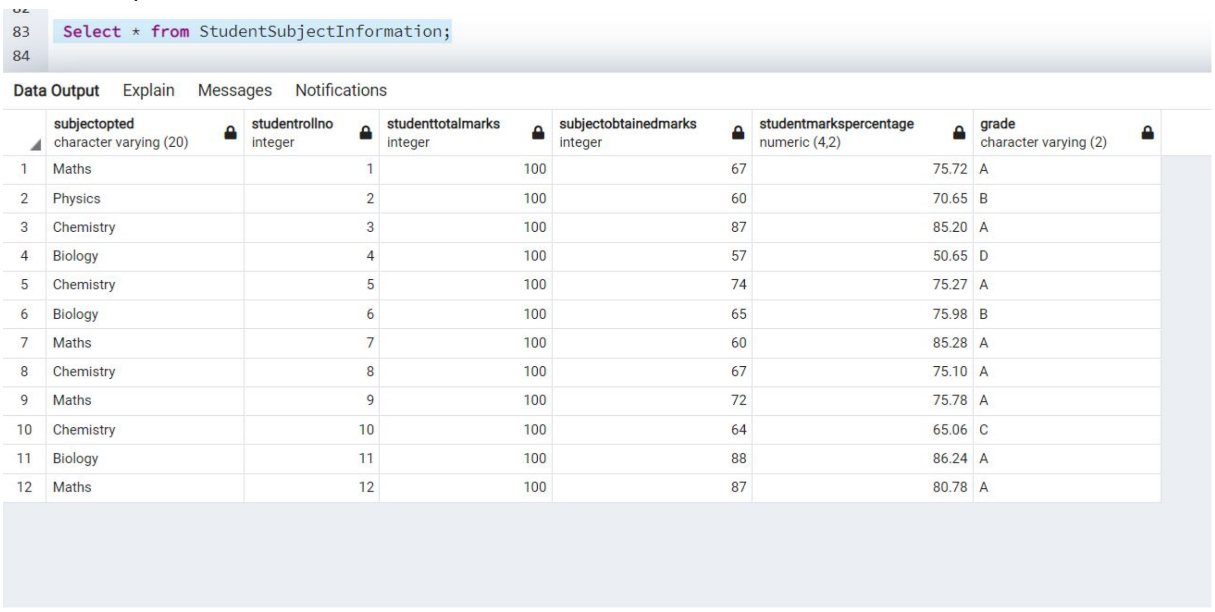
```
Insert into StudentSubjectInformation(SubjectOpted,StudentRollNo,StudentTotalMarks,SubjectObtainedMarks,StudentMarksPercentage,Grade)
 VALUES
 ('Maths',1,100,67,75.72,'A'),
 ('Physics',2,100,60,70.65,'B'),
 ('Chemistry',3,100,87,85.20,'A'),
 ('Biology',4,100,57,50.65,'D'),
 ('Chemistry',5,100,74,75.27,'A'),
 ('Biology',6,100,65,75.98,'B'),
 ('Maths',7,100,60,85.28,'A'),
 ('Chemistry',8,100,67,75.10,'A'),
 ('Maths',9,100,72,75.78,'A'),
 ('Chemistry',10,100,64,65.06,'C'),
 ('Biology',11,100,88,86.24,'A'),
 ('Maths',12,100,87,80.78,'A');
```

Data Output Explain Messages Notifications

```
INSERT 0 12

Query returned successfully in 87 msec.
```

Table Snap:-



```
Select * from StudentSubjectInformation;
```

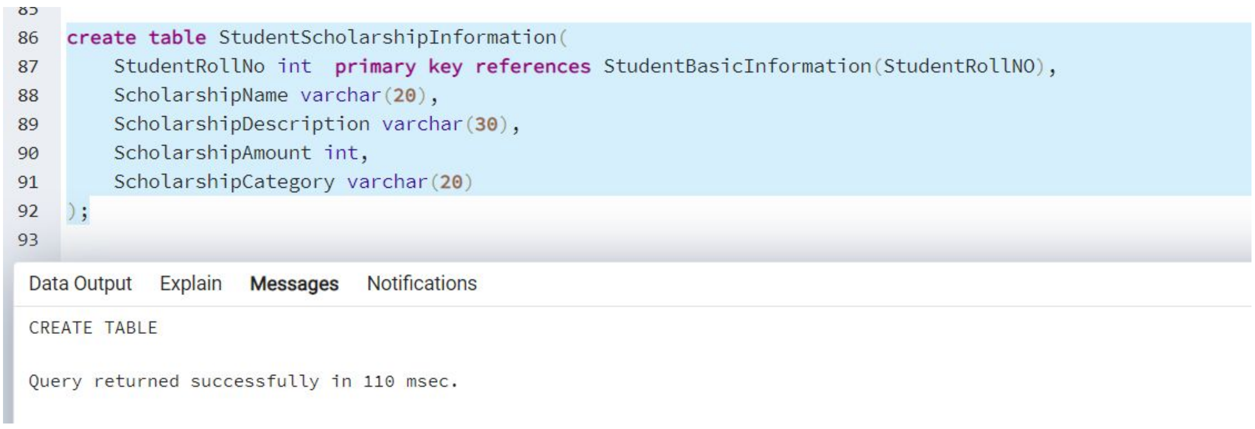
Data Output Explain Messages Notifications

	subjectopted character varying (20)	studentrollno integer	studenttotalmarks integer	subjectobtainedmarks integer	studentmarkspercentage numeric (4,2)	grade character varying (2)
1	Maths	1	100	67	75.72	A
2	Physics	2	100	60	70.65	B
3	Chemistry	3	100	87	85.20	A
4	Biology	4	100	57	50.65	D
5	Chemistry	5	100	74	75.27	A
6	Biology	6	100	65	75.98	B
7	Maths	7	100	60	85.28	A
8	Chemistry	8	100	67	75.10	A
9	Maths	9	100	72	75.78	A
10	Chemistry	10	100	64	65.06	C
11	Biology	11	100	88	86.24	A
12	Maths	12	100	87	80.78	A

d. SubjectScholarshipInformation

   i. Columns

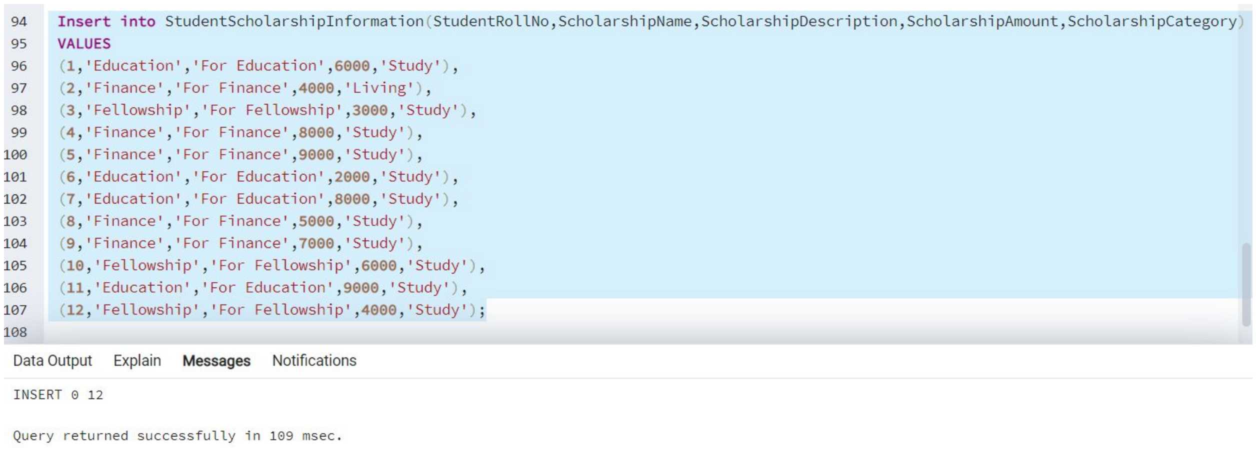
      1. StudentRollNo
      2. ScholarshipName
      3. ScholarshipDescription
      4. ScholarshipAmount
      5. ScholarshipCategory
      6. Add more two columns of the name of your own



```
create table StudentScholarshipInformation(
    StudentRollNo int  primary key references StudentBasicInformation(StudentRollNO),
    ScholarshipName varchar(20),
    ScholarshipDescription varchar(30),
    ScholarshipAmount int,
    ScholarshipCategory varchar(20)
);
```

```
CREATE TABLE

Query returned successfully in 110 msec.
```

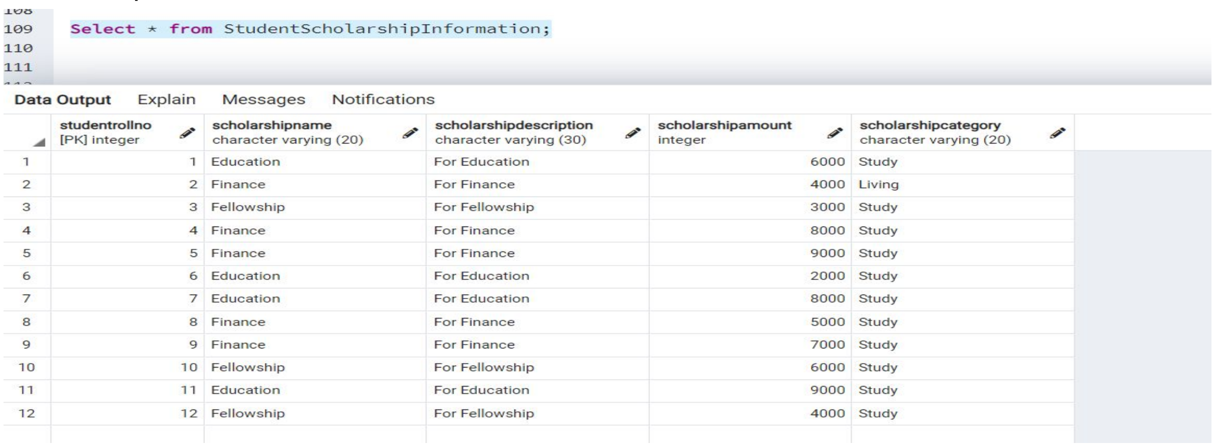


```
Insert into StudentScholarshipInformation(StudentRollNo,ScholarshipName,ScholarshipDescription,ScholarshipAmount,ScholarshipCategory)
VALUES
(1,'Education','For Education',6000,'Study'),
(2,'Finance','For Finance',4000,'Living'),
(3,'Fellowship','For Fellowship',3000,'Study'),
(4,'Finance','For Finance',8000,'Study'),
(5,'Finance','For Finance',9000,'Study'),
(6,'Education','For Education',2000,'Study'),
(7,'Education','For Education',8000,'Study'),
(8,'Finance','For Finance',5000,'Study'),
(9,'Finance','For Finance',7000,'Study'),
(10,'Fellowship','For Fellowship',6000,'Study'),
(11,'Education','For Education',9000,'Study'),
(12,'Fellowship','For Fellowship',4000,'Study');
```

```
INSERT 0 12

Query returned successfully in 109 msec.
```

Table Snap:-



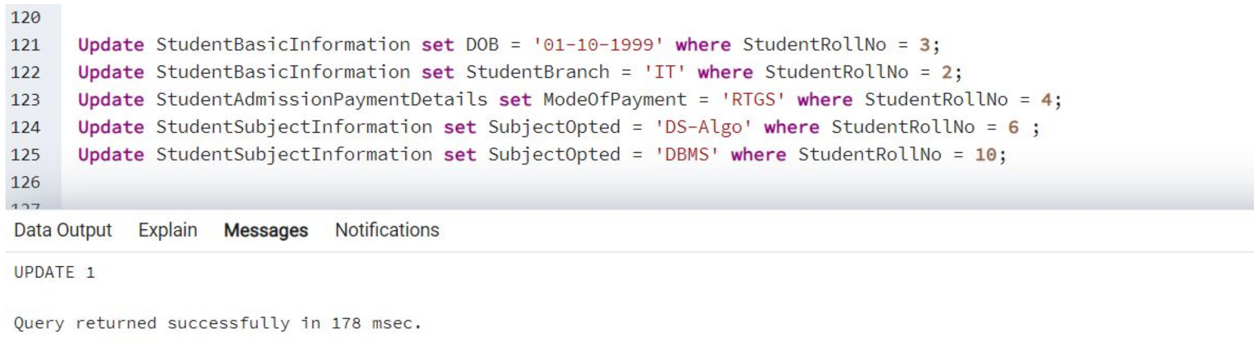
```
Select * from StudentScholarshipInformation;
```

	studentrollno [PK] integer	scholarshipname character varying (20)	scholarshipdescription character varying (30)	scholarshipamount integer	scholarshipcategory character varying (20)
1	1	Education	For Education	6000	Study
2	2	Finance	For Finance	4000	Living
3	3	Fellowship	For Fellowship	3000	Study
4	4	Finance	For Finance	8000	Study
5	5	Finance	For Finance	9000	Study
6	6	Education	For Education	2000	Study
7	7	Education	For Education	8000	Study
8	8	Finance	For Finance	5000	Study
9	9	Finance	For Finance	7000	Study
10	10	Fellowship	For Fellowship	6000	Study
11	11	Education	For Education	9000	Study
12	12	Fellowship	For Fellowship	4000	Study

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

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

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



```sql

Update StudentBasicInformation set DOB = '01-10-1999' where StudentRollNo = 3;
Update StudentBasicInformation set StudentBranch = 'IT' where StudentRollNo = 2;
Update StudentAdmissionPaymentDetails set ModeOfPayment = 'RTGS' where StudentRollNo = 4;
Update StudentSubjectInformation set SubjectOpted = 'DS-Algo' where StudentRollNo = 6 ;
Update StudentSubjectInformation set SubjectOpted = 'DBMS' where StudentRollNo = 10;

```

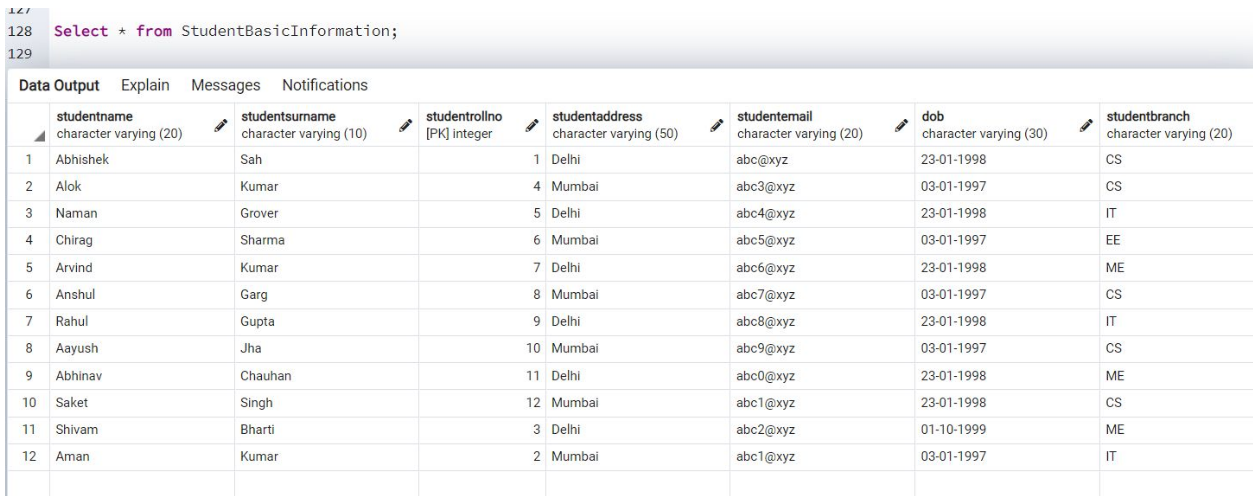
Data Output   Explain   **Messages**   Notifications

```
UPDATE 1

Query returned successfully in 178 msec.
```

6. Snap of the all the tables post updation

   StudentBasicInformation:-

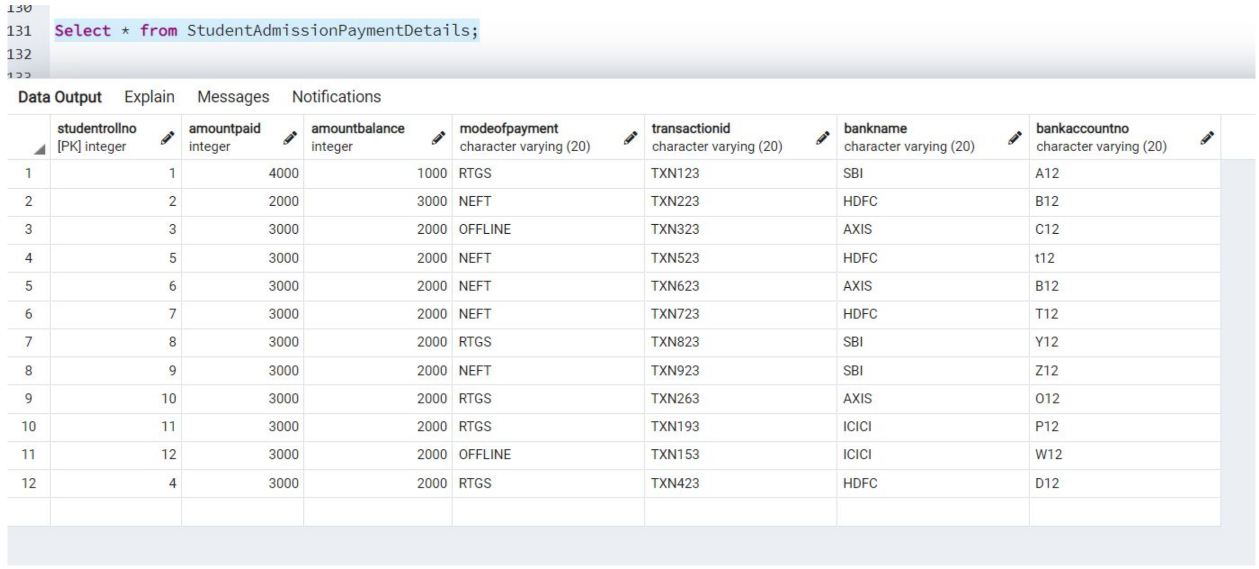


```sql
Select * from StudentBasicInformation;
```

Data Output   Explain   Messages   Notifications

	studentname character varying (20)	studentsurname character varying (10)	studentrollno [PK] integer	studentaddress character varying (50)	studentemail character varying (20)	dob character varying (30)	studentbranch character varying (20)
1	Abhishek	Sah	1	Delhi	abc@xyz	23-01-1998	CS
2	Alok	Kumar	4	Mumbai	abc3@xyz	03-01-1997	CS
3	Naman	Grover	5	Delhi	abc4@xyz	23-01-1998	IT
4	Chirag	Sharma	6	Mumbai	abc5@xyz	03-01-1997	EE
5	Arvind	Kumar	7	Delhi	abc6@xyz	23-01-1998	ME
6	Anshul	Garg	8	Mumbai	abc7@xyz	03-01-1997	CS
7	Rahul	Gupta	9	Delhi	abc8@xyz	23-01-1998	IT
8	Aayush	Jha	10	Mumbai	abc9@xyz	03-01-1997	CS
9	Abhinav	Chauhan	11	Delhi	abc0@xyz	23-01-1998	ME
10	Saket	Singh	12	Mumbai	abc1@xyz	23-01-1998	CS
11	Shivam	Bharti	3	Delhi	abc2@xyz	01-10-1999	ME
12	Aman	Kumar	2	Mumbai	abc1@xyz	03-01-1997	IT

StudentAdmissionPaymentDetails:-

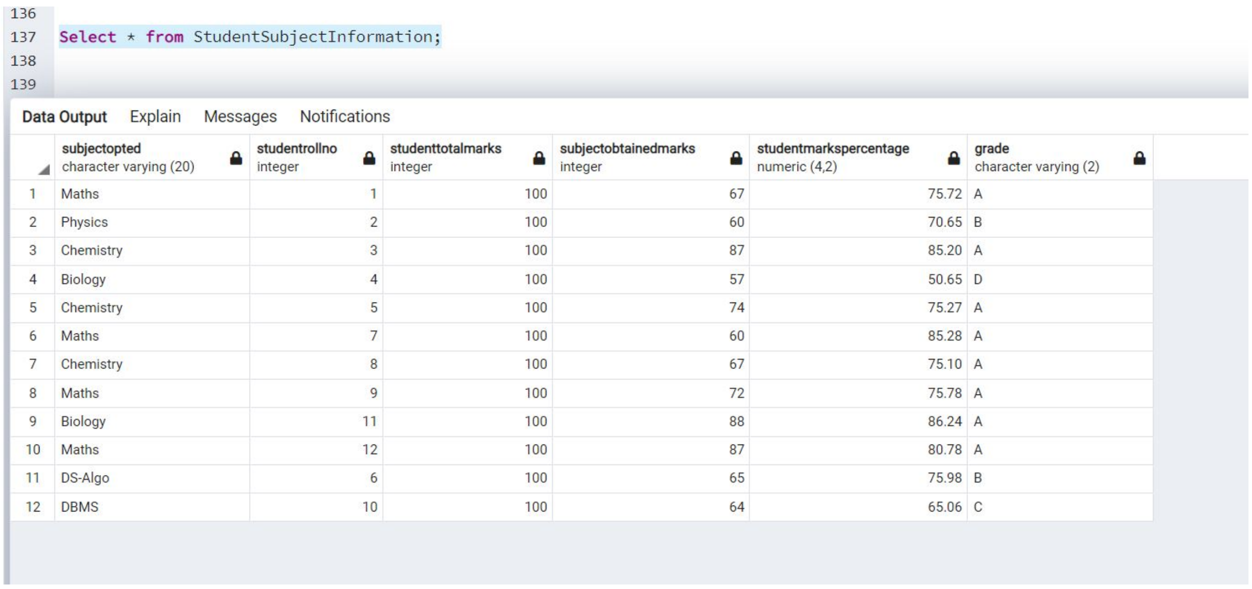


```sql
Select * from StudentAdmissionPaymentDetails;
```

Data Output Explain Messages Notifications

	studentrollno [PK] integer	amountpaid integer	amountbalance integer	modeofpayment character varying (20)	transactionid character varying (20)	bankname character varying (20)	bankaccountno character varying (20)
1	1	4000	1000	RTGS	TXN123	SBI	A12
2	2	2000	3000	NEFT	TXN223	HDFC	B12
3	3	3000	2000	OFFLINE	TXN323	AXIS	C12
4	5	3000	2000	NEFT	TXN523	HDFC	t12
5	6	3000	2000	NEFT	TXN623	AXIS	B12
6	7	3000	2000	NEFT	TXN723	HDFC	T12
7	8	3000	2000	RTGS	TXN823	SBI	Y12
8	9	3000	2000	NEFT	TXN923	SBI	Z12
9	10	3000	2000	RTGS	TXN263	AXIS	O12
10	11	3000	2000	RTGS	TXN193	ICICI	P12
11	12	3000	2000	OFFLINE	TXN153	ICICI	W12
12	4	3000	2000	RTGS	TXN423	HDFC	D12

StudentSubjectInformation:-

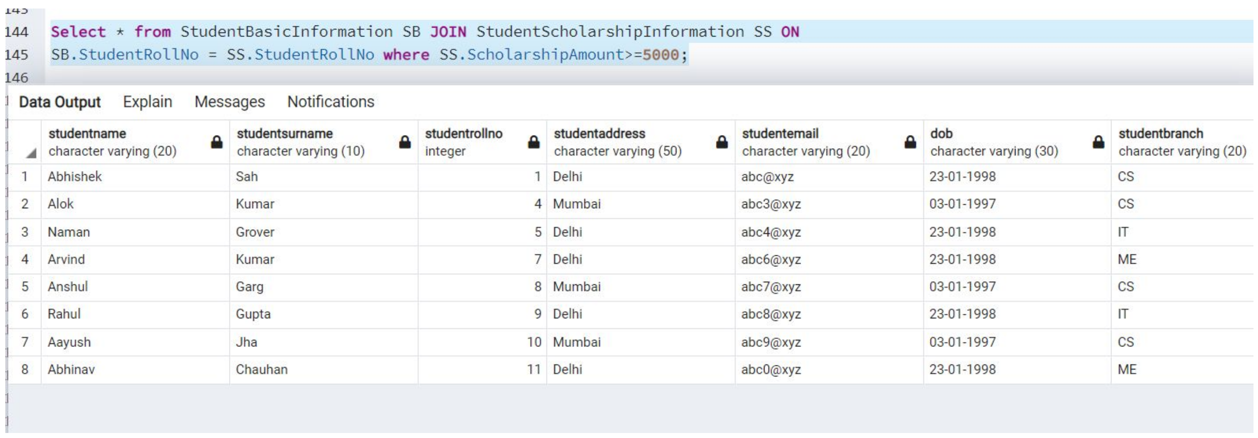


```sql
Select * from StudentSubjectInformation;
```

Data Output Explain Messages Notifications

	subjectopted character varying (20)	studentrollno integer	studenttotalmarks integer	subjectobtainedmarks integer	studentmarkspercentage numeric (4,2)	grade character varying (2)
1	Maths	1	100	67	75.72	A
2	Physics	2	100	60	70.65	B
3	Chemistry	3	100	87	85.20	A
4	Biology	4	100	57	50.65	D
5	Chemistry	5	100	74	75.27	A
6	Maths	7	100	60	85.28	A
7	Chemistry	8	100	67	75.10	A
8	Maths	9	100	72	75.78	A
9	Biology	11	100	88	86.24	A
10	Maths	12	100	87	80.78	A
11	DS-Algo	6	100	65	75.98	B
12	DBMS	10	100	64	65.06	C

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

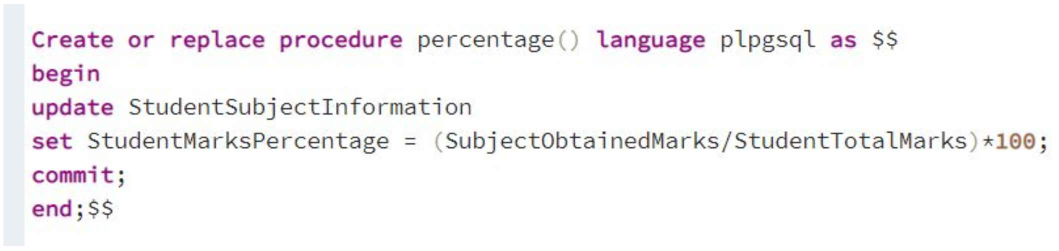


```
Select * from StudentBasicInformation SB JOIN StudentScholarshipInformation SS ON
SB.StudentRollNo = SS.StudentRollNo where SS.ScholarshipAmount>=5000;
```

Data Output | Explain | Messages | Notifications

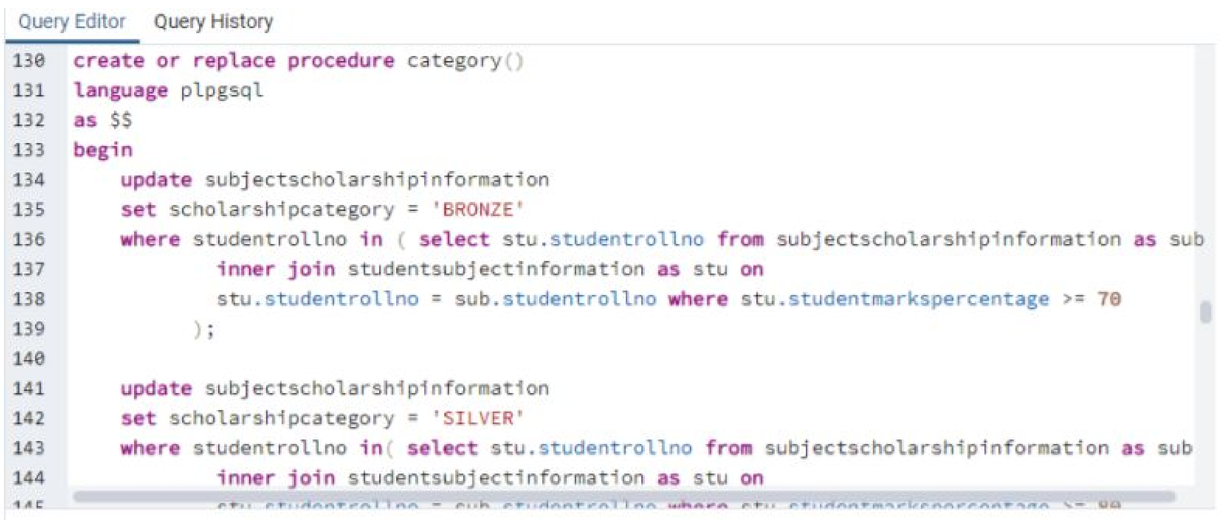
	studentname character varying (20)	studentsurname character varying (10)	studentrollno integer	studentaddress character varying (50)	studentemail character varying (20)	dob character varying (30)	studentbranch character varying (20)
1	Abhishek	Sah	1	Delhi	abc@xyz	23-01-1998	CS
2	Alok	Kumar	4	Mumbai	abc3@xyz	03-01-1997	CS
3	Naman	Grover	5	Delhi	abc4@xyz	23-01-1998	IT
4	Arvind	Kumar	7	Delhi	abc6@xyz	23-01-1998	ME
5	Anshul	Garg	8	Mumbai	abc7@xyz	03-01-1997	CS
6	Rahul	Gupta	9	Delhi	abc8@xyz	23-01-1998	IT
7	Aayush	Jha	10	Mumbai	abc9@xyz	03-01-1997	CS
8	Abhinav	Chauhan	11	Delhi	abc0@xyz	23-01-1998	ME

8. Select the students who opted for scholarship but has not got the scholarship
9. Fill in data for the percentage column i.e. StudentMarksPercentage in the table StudentSubjectInformation by creating and using the stored procedure created



```
Create or replace procedure percentage() language plpgsql as $$
begin
update StudentSubjectInformation
set StudentMarksPercentage = (SubjectObtainedMarks/StudentTotalMarks)*100;
commit;
end;$$
```

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

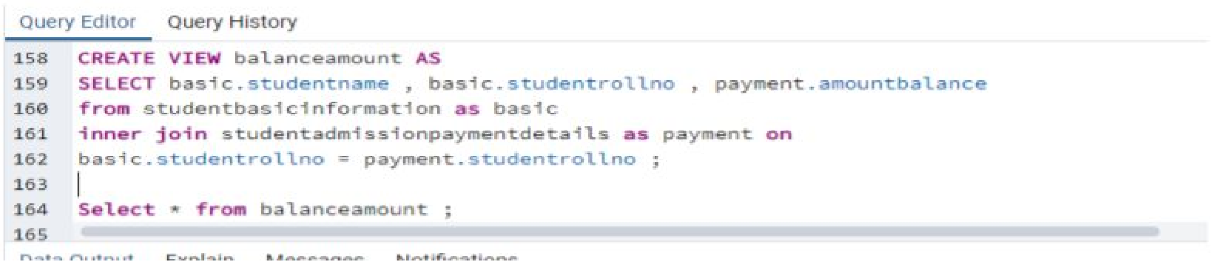


Query Editor | Query History

```
create or replace procedure category()
language plpgsql
as $$
begin
    update subjectscholarshipinformation
    set scholarshipcategory = 'BRONZE'
    where studentrollno in ( select stu.studentrollno from subjectscholarshipinformation as sub
            inner join studentsubjectinformation as stu on
            stu.studentrollno = sub.studentrollno where stu.studentmarkspercentage >= 70
          );

    update subjectscholarshipinformation
    set scholarshipcategory = 'SILVER'
    where studentrollno in( select stu.studentrollno from subjectscholarshipinformation as sub
            inner join studentsubjectinformation as stu on
            stu.studentrollno = sub.studentrollno where stu.studentmarkspercentage >= 80
```

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


Query Editor Query History

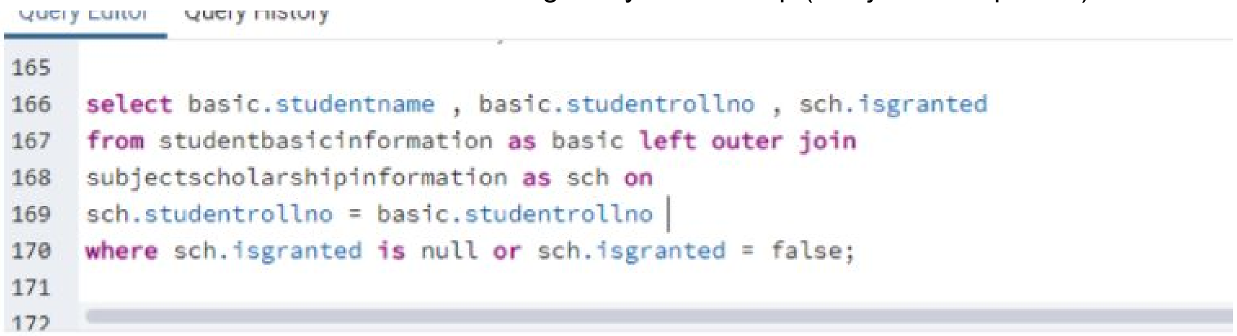
```
CREATE VIEW balanceamount AS
SELECT basic.studentname , basic.studentrollno , payment.amountbalance
from studentbasicinformation as basic
inner join studentadmissionpaymentdetails as payment on
basic.studentrollno = payment.studentrollno ;

Select * from balanceamount ;

```

Data Output Explain Messages Notifications

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

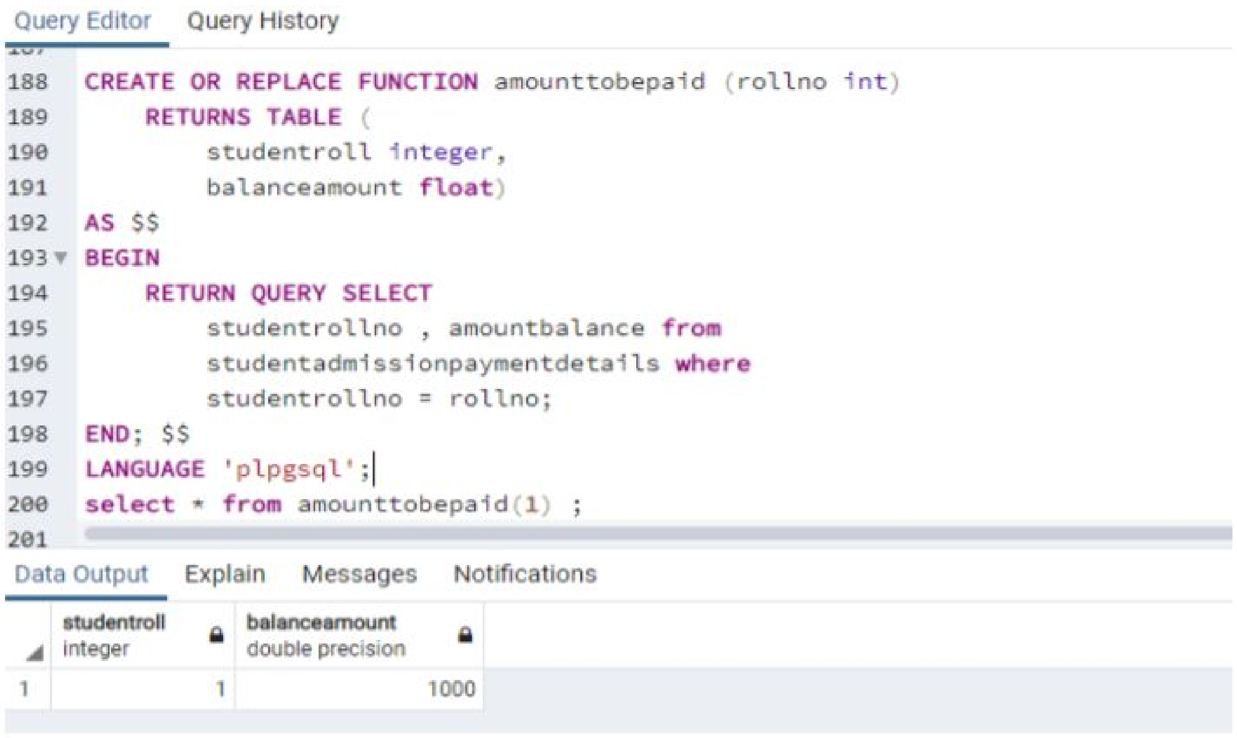

Query Editor Query History

```

select basic.studentname , basic.studentrollno , sch.isgranted
from studentbasicinformation as basic left outer join
subjectscholarshipinformation as sch on
sch.studentrollno = basic.studentrollno
where sch.isgranted is null or sch.isgranted = false;

```

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


Query Editor Query History

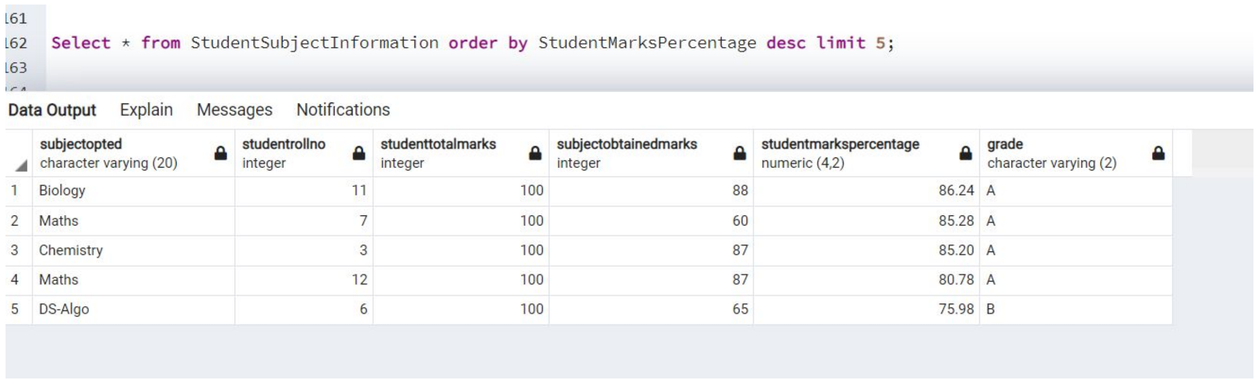
```
CREATE OR REPLACE FUNCTION amounttobepaid (rollno int)
    RETURNS TABLE (
        studentroll integer,
        balanceamount float)
AS $$
BEGIN
    RETURN QUERY SELECT
        studentrollno , amountbalance from
        studentadmissionpaymentdetails where
        studentrollno = rollno;
END; $$
LANGUAGE 'plpgsql';
select * from amounttobepaid(1) ;

```

Data Output Explain Messages Notifications

	studentroll integer	balanceamount double precision
1	1	1000

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



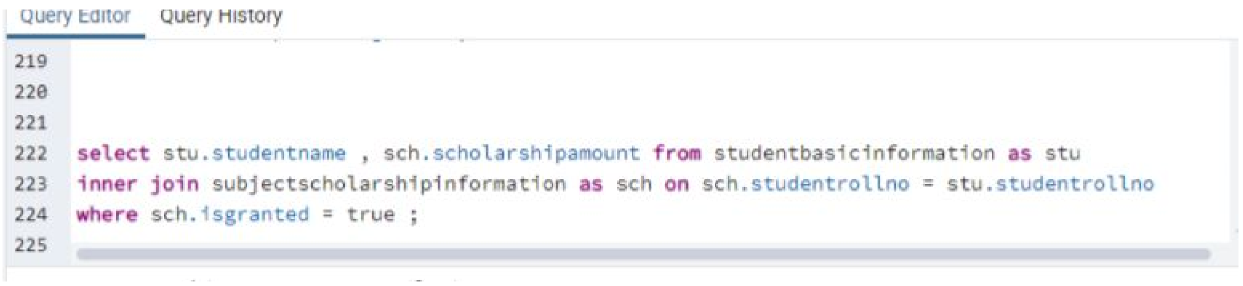
```sql
Select * from StudentSubjectInformation order by StudentMarksPercentage desc limit 5;
```

Data Output | Explain | Messages | Notifications

	subjectopted character varying (20)	studentrollno integer	studenttotalmarks integer	subjectobtainedmarks integer	studentmarkspercentage numeric (4,2)	grade character varying (2)
1	Biology	11	100	88	86.24	A
2	Maths	7	100	60	85.28	A
3	Chemistry	3	100	87	85.20	A
4	Maths	12	100	87	80.78	A
5	DS-Algo	6	100	65	75.98	B

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)

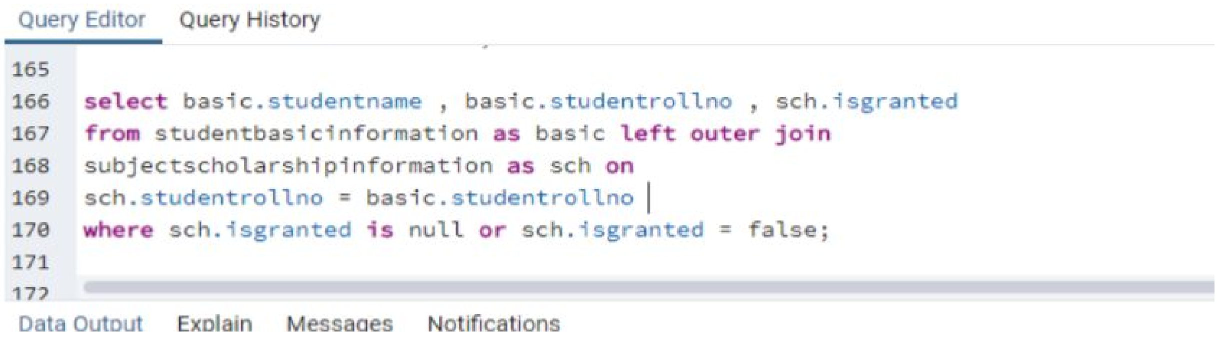
(INNER) JOIN: Returns records that have matching values in both tables



Query Editor | Query History

```sql
select stu.studentname , sch.scholarshipamount from studentbasicinformation as stu
inner join subjectscholarshipinformation as sch on sch.studentrollno = stu.studentrollno
where sch.isgranted = true ;
```

LEFT (OUTER) JOIN: Returns all records from the left table, and the matched records from the right table



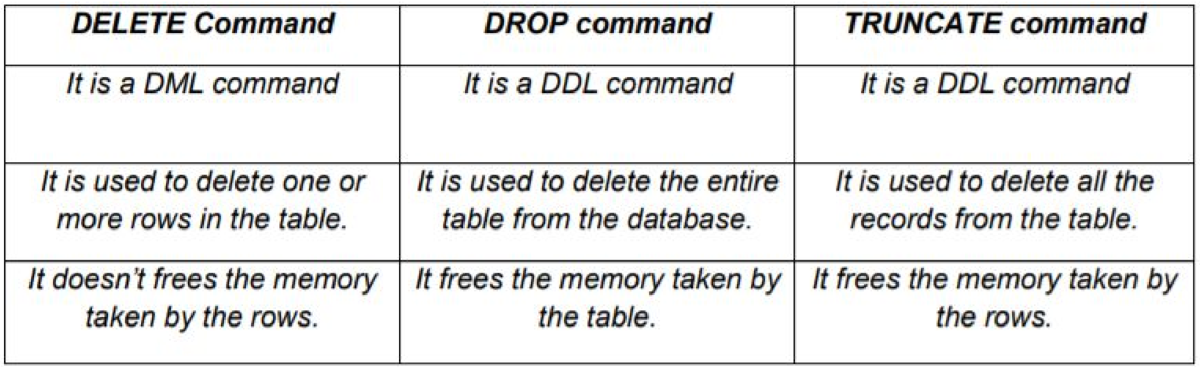
Query Editor | Query History

```sql
select basic.studentname , basic.studentrollno , sch.isgranted
from studentbasicinformation as basic left outer join
subjectscholarshipinformation as sch on
sch.studentrollno = basic.studentrollno
where sch.isgranted is null or sch.isgranted = false;
```

Data Output | Explain | Messages | Notifications

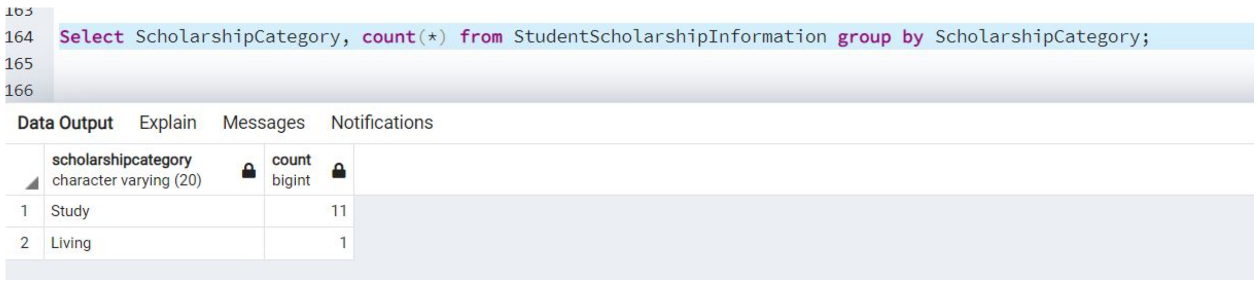
RIGHT (OUTER) JOIN: Returns all records from the right table, and the matched records from the left table

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



DELETE Command	***DROP command***	***TRUNCATE command***
It is a DML command	*It is a DDL command*	*It is a DDL command*
It is used to delete one or more rows in the table.	*It is used to delete the entire table from the database.*	*It is used to delete all the records from the table.*
It doesn't frees the memory taken by the rows.	*It frees the memory taken by the table.*	*It frees the memory taken by the rows.*

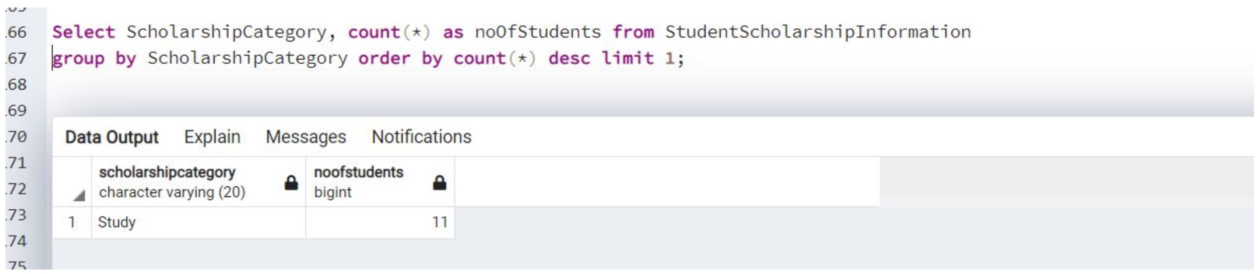
17. Get the count of the Scholarship category which is highly been availed by the students, i.e. get the count of the total number of students corresponding to the each scholarships category



```sql
Select ScholarshipCategory, count(*) from StudentScholarshipInformation group by ScholarshipCategory;
```

	scholarshipcategory character varying (20)	count bigint
1	Study	11
2	Living	1

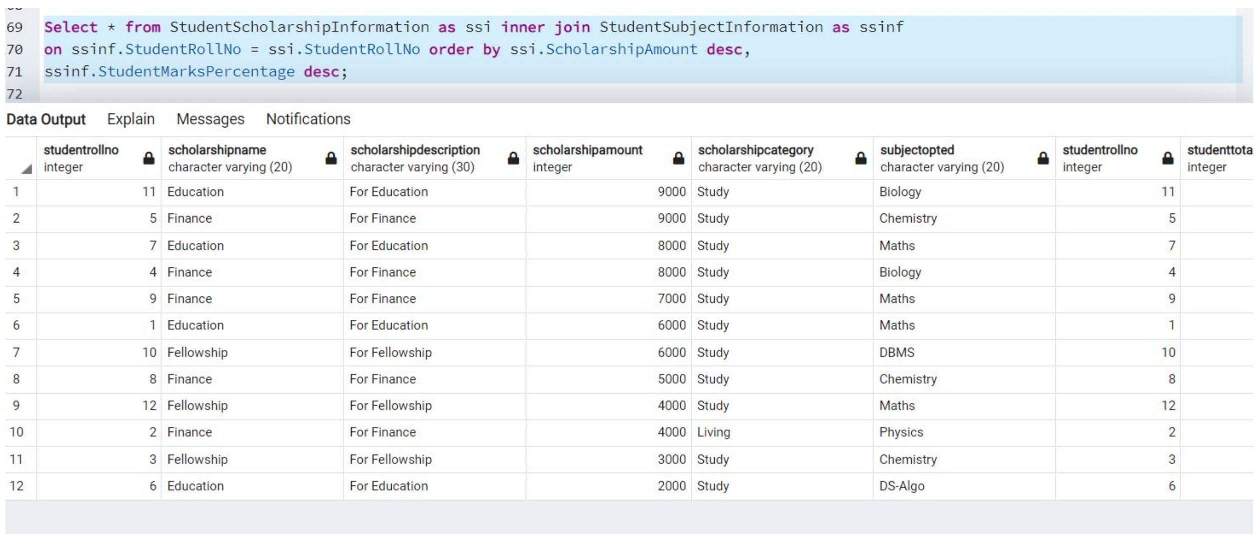
18. Along with the assignment no. 17 try to retrieve the maximum used scholarship category



```sql
Select ScholarshipCategory, count(*) as noOfStudents from StudentScholarshipInformation
group by ScholarshipCategory order by count(*) desc limit 1;
```

	scholarshipcategory character varying (20)	noofstudents bigint
1	Study	11

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



```sql
Select * from StudentScholarshipInformation as ssi inner join StudentSubjectInformation as ssinf
on ssinf.StudentRollNo = ssi.StudentRollNo order by ssi.ScholarshipAmount desc,
ssinf.StudentMarksPercentage desc;
```

Data Output | Explain | Messages | Notifications

	studentrollno integer	scholarshipname character varying (20)	scholarshipdescription character varying (30)	scholarshipamount integer	scholarshipcategory character varying (20)	subjectopted character varying (20)	studentrollno integer	studenttota integer
1	11	Education	For Education	9000	Study	Biology	11	
2	5	Finance	For Finance	9000	Study	Chemistry	5	
3	7	Education	For Education	8000	Study	Maths	7	
4	4	Finance	For Finance	8000	Study	Biology	4	
5	9	Finance	For Finance	7000	Study	Maths	9	
6	1	Education	For Education	6000	Study	Maths	1	
7	10	Fellowship	For Fellowship	6000	Study	DBMS	10	
8	8	Finance	For Finance	5000	Study	Chemistry	8	
9	12	Fellowship	For Fellowship	4000	Study	Maths	12	
10	2	Finance	For Finance	4000	Living	Physics	2	
11	3	Fellowship	For Fellowship	3000	Study	Chemistry	3	
12	6	Education	For Education	2000	Study	DS-Algo	6	

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

Triggers:

- *Trigger is a stored procedure that runs automatically when a specific event happens (update, delete, insert) .*
- *Triggers cannot take input as a parameter*
- *Triggers cannot return values.*

Stored Procedures:

- *They are the piece of code written in a block to perform a specific task when called.*
- *They can take input as a parameter.*
- *They can only return values as an OUT parameter.*

Functions:

- *They are same as stored procedures but can return values and can be used in an expression.*

Views:

- *Views are pseudo-tables that can be made from other tables by selecting any number of rows and columns from the table.*
- *They are usually made to retrieve frequent used data from the table, so that time to execute the query in the whole big table is reduced.*