

**DONE BY~**  
**HARSHIT KESAR**  
**Email- [harsshit.kesar@accolitedigital.com](mailto:harsshit.kesar@accolitedigital.com)**  
**Phone No- 7827246763**

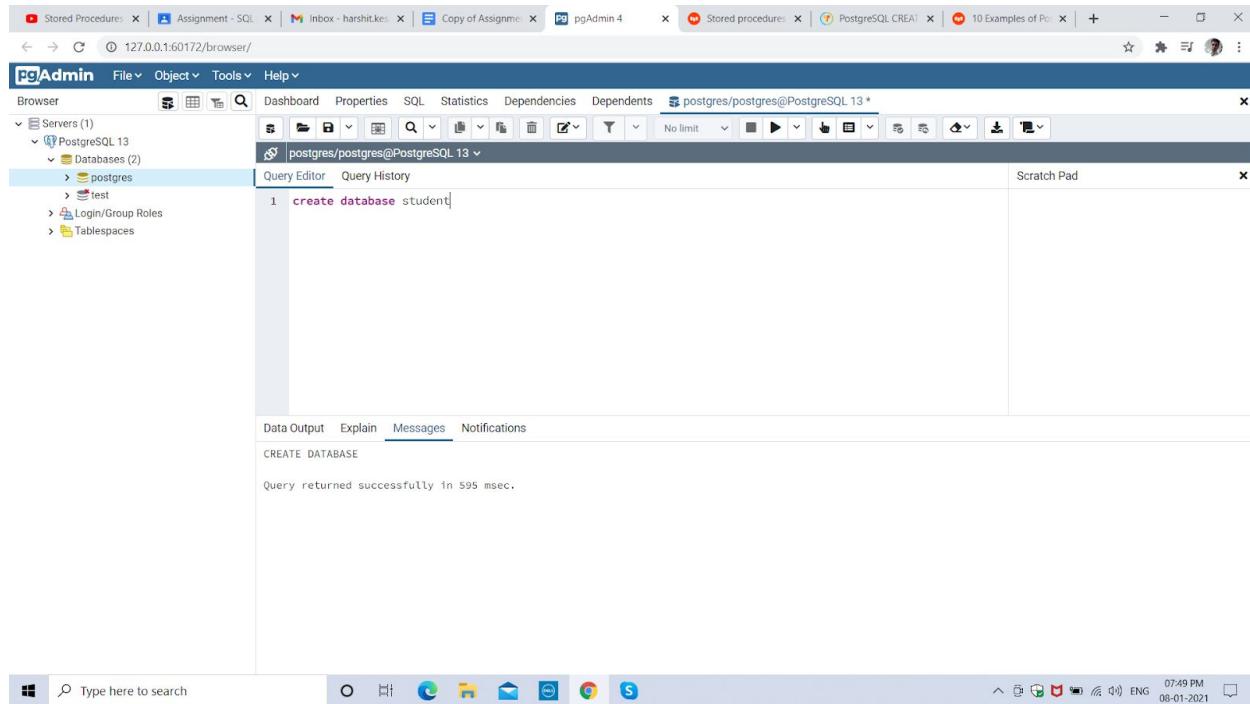
## Prior Instructions

- 
- 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

**Query:** create database student

**Screen Shot:**



### 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**

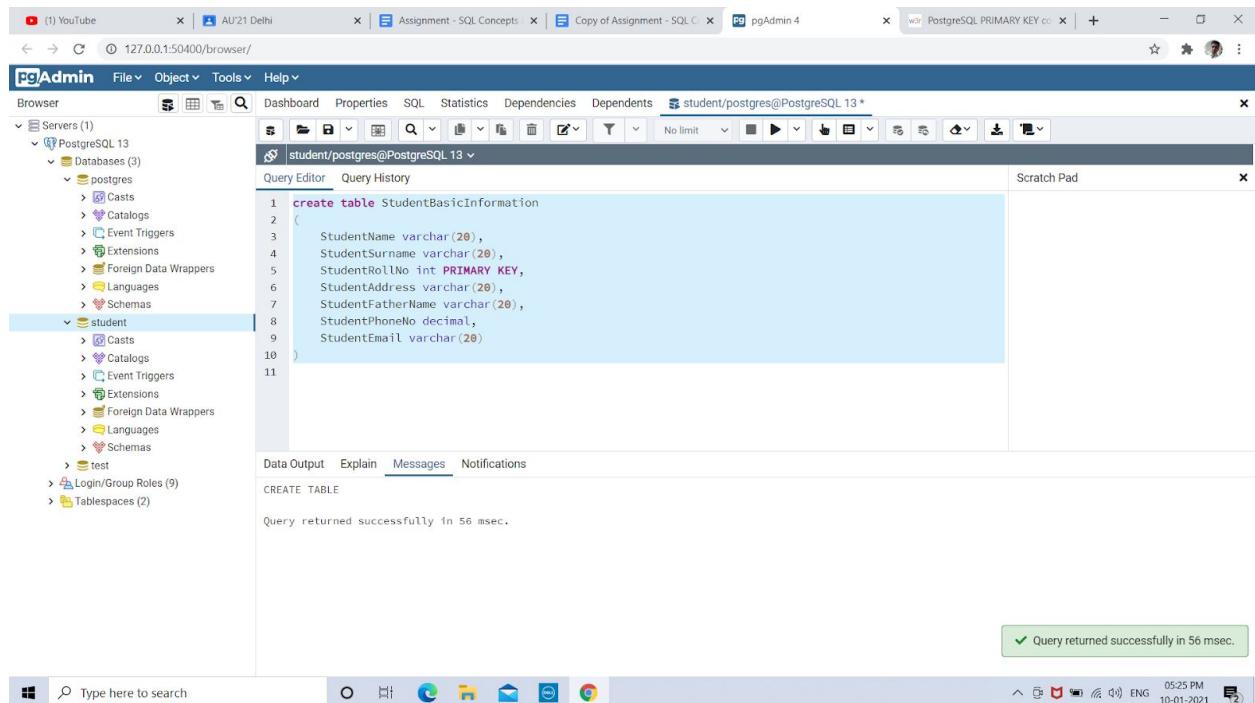
**Query:** create table StudentBasicInformation

(

```
StudentName varchar(20),
StudentSurname varchar(20),
StudentRollNo int PRIMARY KEY,
StudentAddress varchar(20),
StudentFatherName varchar(20),
StudentPhoneNo decimal,
StudentEmail varchar(20)
```

)

#### **Screen Shot:**



#### **b. StudentAdmissionPaymentDetails**

##### **i. Columns**

- 1. StudentRollNo**
- 2. AmountPaid**
- 3. AmountBalance**
- 4. Add more four basic columns of the name of your own**

**Query:** create table StudentAdmissionPaymentDetails

(

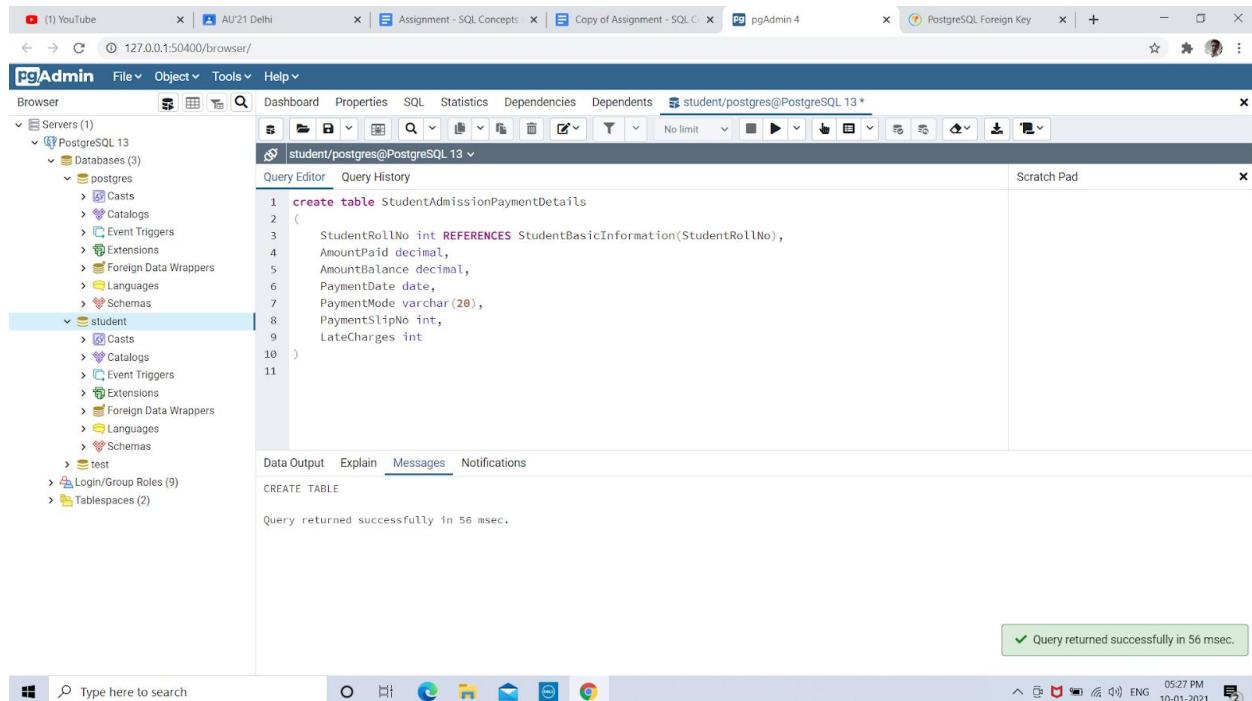
```
StudentRollNo int REFERENCES StudentBasicInformation(StudentRollNo),
```

```

        AmountPaid decimal,
        AmountBalance decimal,
        PaymentDate date,
        PaymentMode varchar(20),
        PaymentSlipNo int,
        LateCharges int
)

```

### Screen Shot:



### 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

**Query:** create table StudentSubjectInformation

(

```

SubjectOpted varchar(20),
StudentRollNo int REFERENCES StudentBasicInformation(StudentRollNo),
SubjectTotalMarks int,
SubjectObtainedMarks int,

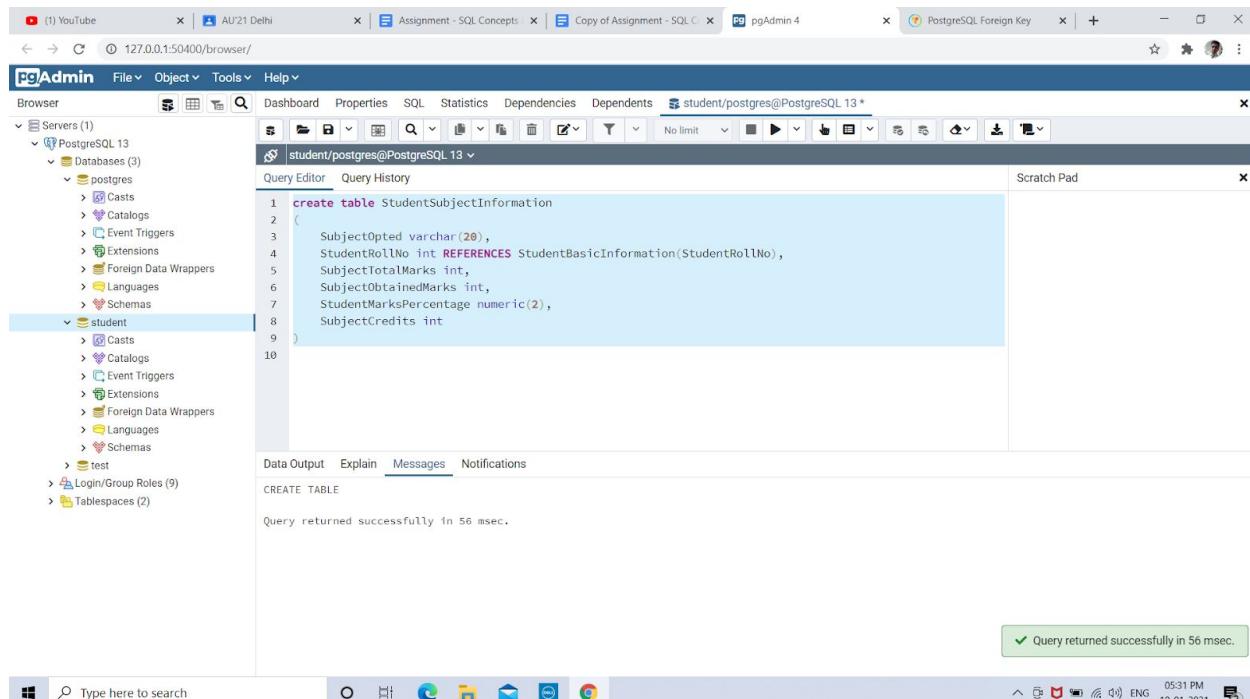
```

```

StudentMarksPercentage numeric(2),
SubjectCredits int
)

```

## Screen Shot:



## 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**

**Query:** Create table SubjectScholarshipInformation

(

```

StudentRollNo int REFERENCES StudentBasicInformation(StudentRollNo),
ScholarshipName varchar(20),
ScholarshipDescription varchar(50),
ScholarshipAmount int,
ScholarshipCategory varchar(20),
ScholarshipDate date,
ScholarshipGivenBy varchar(20)

```

)

## Screen Shot:

The screenshot shows the pgAdmin 4 interface. On the left, the 'Servers' tree view shows 'PostgreSQL 13' with 'Databases (3)' expanded, revealing 'postgres', 'student', and 'test'. The 'student' database is selected. In the center, the 'Query Editor' tab is active, displaying the following SQL code:

```
1 Create table SubjectScholarshipInformation
2 (
3     StudentRollNo int REFERENCES StudentBasicInformation(StudentRollNo),
4     ScholarshipName varchar(20),
5     ScholarshipDescription varchar(50),
6     ScholarshipAmount int,
7     ScholarshipCategory varchar(20),
8     ScholarshipDate date,
9     ScholarshipGivenBy varchar(20)
10 )
```

Below the code, the 'Messages' tab shows the message: 'CREATE TABLE'. At the bottom right of the pgAdmin window, a green status bar indicates: '✓ Query returned successfully in 57 msec.'

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

Query:

## Screen Shot:

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

TABLE 1:

Query: insert into StudentBasicInformation

(StudentName, StudentSurname, StudentRollNo, StudentAddress, StudentFatherName, StudentPhoneNo, StudentEmail)

values ('Harshit', 'Kesar', 1, 'Delhi, India', 'Raju', 7827246763, 'harshit@gmail.com');

insert into StudentBasicInformation

(StudentName, StudentSurname, StudentRollNo, StudentAddress, StudentFatherName, StudentPhoneNo, StudentEmail)

values ('abc', 'yz', 2, 'Mumbai, India', 'f2', 9827246763, 'abc@gmail.com');

insert into StudentBasicInformation

(StudentName, StudentSurname, StudentRollNo, StudentAddress, StudentFatherName, StudentPhoneNo, StudentEmail)

values ('def', 'vwx', 3, 'Delhi, India', 'f3', 8827246763, 'def@gmail.com');

insert into StudentBasicInformation

(StudentName, StudentSurname, StudentRollNo, StudentAddress, StudentFatherName, StudentPhoneNo, StudentEmail)

```
values ('ghi', 'stu', 4, 'Bihar, India', 'f4', 7825246763, 'ghi@gmail.com');
insert into StudentBasicInformation
(StudentName, StudentSurname, StudentRollNo, StudentAddress, StudentFatherName,
StudentPhoneNo, StudentEmail)
values ('jkl', 'pqr', 5, 'Banglore, India', 'f5', 7827247863, 'jkl@gmail.com');
insert into StudentBasicInformation
(StudentName, StudentSurname, StudentRollNo, StudentAddress, StudentFatherName,
StudentPhoneNo, StudentEmail)
values ('mno', 'mno', 6, 'Pune, India', 'f6', 7822224676, 'mno@gmail.com');
insert into StudentBasicInformation
(StudentName, StudentSurname, StudentRollNo, StudentAddress, StudentFatherName,
StudentPhoneNo, StudentEmail)
values ('pqr', 'jkl', 7, 'Chennai, India', 'f7', 7825546763, 'pqr@gmail.com');
insert into StudentBasicInformation
(StudentName, StudentSurname, StudentRollNo, StudentAddress, StudentFatherName,
StudentPhoneNo, StudentEmail)
values ('stu', 'ghi', 8, 'Jammu, India', 'f8', 7829946763, 'stu@gmail.com');
insert into StudentBasicInformation
(StudentName, StudentSurname, StudentRollNo, StudentAddress, StudentFatherName,
StudentPhoneNo, StudentEmail)
values ('vwx', 'def', 9, 'Gujrat, India', 'f9', 7822246763, 'vwx@gmail.com');
insert into StudentBasicInformation
(StudentName, StudentSurname, StudentRollNo, StudentAddress, StudentFatherName,
StudentPhoneNo, StudentEmail)
values ('yz', 'abc', 10, 'Delhi, India', 'f10', 7827856763, 'yz@gmail.com');
```

## Screen Shot:

The screenshot shows the pgAdmin 4 interface with a query editor window titled "PostgreSQL Foreign Key". The query being run is:

```
1 select * from studentbasicinformation;
```

The results are displayed in a table format:

	studentname	studentsurname	studentrollno	studentaddress	studentfirstname	studentphoneno	studentemail
1	Harshit	Kesar	1	Delhi, India	Raju	7827245763	harshit@gmail.com
2	abc	yz	2	Mumbai, India	f2	9827246763	abc@gmail.com
3	def	vvx	3	Delhi, India	f3	8827246763	def@gmail.com
4	ghi	stu	4	Bihar, India	f4	7825246763	ghi@gmail.com
5	jkl	pqr	5	Banglore, India	f5	7827247863	jkl@gmail.com
6	mno	mno	6	Pune, India	f6	7822224676	mno@gmail.com
7	pqr	jkl	7	Chennai, India	f7	7825546763	pqr@gmail.com
8	stu	ghi	8	Jammu, India	f8	7829946763	stu@gmail.com
9	vvx	def	9	Gujrat, India	f9	7822246763	vvx@gmail.com
10	yz	abc	10	Delhi, India	f10	7827856763	yz@gmail.com

A message at the bottom right indicates: "Successfully run. Total query runtime: 363 msec. 10 rows affected."

**TABLE 2:**

**Query:** insert into StudentAdmissionPaymentDetails

(StudentRollNo, AmountPaid, AmountBalance, PaymentDate, PaymentMode, PaymentSlipNo, LateCharges)

values (2, 50000, 25000, '25-12-2020', 'UPI', 123, 2500);

insert into StudentAdmissionPaymentDetails

(StudentRollNo, AmountPaid, AmountBalance, PaymentDate, PaymentMode, PaymentSlipNo, LateCharges)

values (1, 55000, 20000, '20-08-2019', 'Paytm', 124, 250);

insert into StudentAdmissionPaymentDetails

(StudentRollNo, AmountPaid, AmountBalance, PaymentDate, PaymentMode, PaymentSlipNo, LateCharges)

values (3, 40000, 35000, '02-01-2020', 'AXIS BANK', 125, 1700);

insert into StudentAdmissionPaymentDetails

(StudentRollNo, AmountPaid, AmountBalance, PaymentDate, PaymentMode, PaymentSlipNo, LateCharges)

values (4, 30000, 45000, '04-02-2020', 'PNB', 126, 2100);

insert into StudentAdmissionPaymentDetails

(StudentRollNo, AmountPaid, AmountBalance, PaymentDate, PaymentMode, PaymentSlipNo, LateCharges)

values (5, 35000, 30000, '06-03-2020', 'CANARA BANK', 127, 0);

```

insert into StudentAdmissionPaymentDetails
(StudentRollNo, AmountPaid, AmountBalance, PaymentDate, PaymentMode, PaymentSlipNo,
LateCharges)
values (6, 48000, 24000, '08-04-2020', 'WALLET', 128, 25);
insert into StudentAdmissionPaymentDetails
(StudentRollNo, AmountPaid, AmountBalance, PaymentDate, PaymentMode, PaymentSlipNo,
LateCharges)
values (7, 16000, 2000, '10-05-2020', 'SBI', 129, 50);
insert into StudentAdmissionPaymentDetails
(StudentRollNo, AmountPaid, AmountBalance, PaymentDate, PaymentMode, PaymentSlipNo,
LateCharges)
values (8, 8000, 20000, '15-06-2020', 'RBI', 130, 20);
insert into StudentAdmissionPaymentDetails
(StudentRollNo, AmountPaid, AmountBalance, PaymentDate, PaymentMode, PaymentSlipNo,
LateCharges)
values (9, 51000, 8000, '18-07-2020', 'DENABANK', 131, 250);
insert into StudentAdmissionPaymentDetails
(StudentRollNo, AmountPaid, AmountBalance, PaymentDate, PaymentMode, PaymentSlipNo,
LateCharges)
values (10, 17000, 19000, '22-09-2020', 'BOB', 132, 500);

```

### Screen Shot:

The screenshot shows the pgAdmin 4 interface. On the left, the schema browser displays several tables under the 'studentadmissionpaymentdetails' schema, including columns like studentrollno, amountpaid, amountbalance, paymentdate, paymentmode, paymentslipno, and latecharges. The main pane shows a query editor with the following SQL command:

```
1 select * from studentadmissionpaymentdetails
```

The results pane displays the following data:

	studentrollno	amountpaid	amountbalance	paymentdate	paymentmode	paymentslipno	latecharges
1	2	50000	25000	2020-12-25	UPI	123	2500
2	1	55000	20000	2019-08-20	Paytm	124	250
3	3	40000	35000	2020-01-02	AXIS BANK	125	1700
4	4	30000	45000	2020-02-04	PNB	126	2100
5	5	35000	30000	2020-03-06	CANARA BANK	127	0
6	6	48000	24000	2020-04-08	WALLET	128	25
7	7	16000	2000	2020-05-10	SBI	129	50
8	8	8000	20000	2020-06-15	RBI	130	20
9	9	51000	8000	2020-07-18	DENABANK	131	250
10	10	17000	19000	2020-09-22	BOB	132	500

A status bar at the bottom right of the pgAdmin window indicates: Successfully run. Total query runtime: 84 msec. 10 rows affected.

**TABLE 3:**

**Query:** insert into StudentSubjectInformation  
(SubjectOpted, StudentRollNo, SubjectTotalMarks, SubjectObtainedMarks, SubjectCredits)  
values('Cpp', 1, 100, 98, 4);  
insert into StudentSubjectInformation  
(SubjectOpted, StudentRollNo, SubjectTotalMarks, SubjectObtainedMarks, SubjectCredits)  
values('Java', 2, 100, 55, 4);  
insert into StudentSubjectInformation  
(SubjectOpted, StudentRollNo, SubjectTotalMarks, SubjectObtainedMarks, SubjectCredits)  
values('Python', 3, 100, 45, 4);  
insert into StudentSubjectInformation  
(SubjectOpted, StudentRollNo, SubjectTotalMarks, SubjectObtainedMarks, SubjectCredits)  
values('Javascript', 4, 100, 65, 4);  
insert into StudentSubjectInformation  
(SubjectOpted, StudentRollNo, SubjectTotalMarks, SubjectObtainedMarks, SubjectCredits)  
values('Maven', 5, 100, 75, 4);  
insert into StudentSubjectInformation  
(SubjectOpted, StudentRollNo, SubjectTotalMarks, SubjectObtainedMarks, SubjectCredits)  
values('MySQL', 6, 100, 65, 4);  
insert into StudentSubjectInformation  
(SubjectOpted, StudentRollNo, SubjectTotalMarks, SubjectObtainedMarks, SubjectCredits)  
values('DBMS', 7, 100, 89, 4);  
insert into StudentSubjectInformation  
(SubjectOpted, StudentRollNo, SubjectTotalMarks, SubjectObtainedMarks, SubjectCredits)  
values('NoSQL', 8, 100, 77, 4);  
insert into StudentSubjectInformation  
(SubjectOpted, StudentRollNo, SubjectTotalMarks, SubjectObtainedMarks, SubjectCredits)  
values('OOPS', 9, 100, 63, 4);  
insert into StudentSubjectInformation  
(SubjectOpted, StudentRollNo, SubjectTotalMarks, SubjectObtainedMarks, SubjectCredits)  
values('ECO', 10, 100, 86, 2);

## Screen Shot:

The screenshot shows the pgAdmin 4 interface. On the left, the object browser displays the database schema with 'studentadmissionpaymentdetails' selected. The central area contains a query editor with the following SQL code:

```

1 select * from studentsubjectinformation;
2

```

Below the query editor is a data grid titled 'Data Output' showing the results of the query. The columns are:

subjectopted	studentrollno	subjecttotalmarks	subjectobtainedmarks	studentmarkspercentage	subjectcredits
Cpp	1	100	98	[null]	4
Java	2	100	55	[null]	4
Python	3	100	45	[null]	4
Javascript	4	100	65	[null]	4
Maven	5	100	75	[null]	4
MySQL	6	100	65	[null]	4
DBMS	7	100	89	[null]	4
NoSQL	8	100	77	[null]	4
OOPS	9	100	63	[null]	4
ECO	10	100	86	[null]	2

A message at the bottom right of the data grid area says 'Successfully run. Total query runtime: 84 msec. 10 rows affected.'

**TABLE 4:**

**Query:** insert into SubjectScholarshipInformation  
 (StudentRollNo, ScholarshipName, ScholarshipDescription, ScholarshipAmount,  
 ScholarshipDate, ScholarshipGivenBy)  
 values(1, 'JRF', 'PhD Candidate must be', 32000, '10-10-2020', 'Delhi Govt');  
 insert into SubjectScholarshipInformation  
 (StudentRollNo, ScholarshipName, ScholarshipDescription, ScholarshipAmount,  
 ScholarshipDate, ScholarshipGivenBy)  
 values(2, 'SRF', 'Any Category', 4000, '10-10-2020', 'State Govt');  
 insert into SubjectScholarshipInformation  
 (StudentRollNo, ScholarshipName, ScholarshipDescription, ScholarshipAmount,  
 ScholarshipDate, ScholarshipGivenBy)  
 values(3, 'SC', 'Must belong to SC', 3000, '10-10-2020', 'Mumbai Govt');  
 insert into SubjectScholarshipInformation  
 (StudentRollNo, ScholarshipName, ScholarshipDescription, ScholarshipAmount,  
 ScholarshipDate, ScholarshipGivenBy)  
 values(4, 'OBC', 'Must belong to OBC', 2000, '10-10-2020', 'Central Govt');  
 insert into SubjectScholarshipInformation  
 (StudentRollNo, ScholarshipName, ScholarshipDescription, ScholarshipAmount,  
 ScholarshipDate, ScholarshipGivenBy)  
 values(5, 'ABC', 'Must be abc', 3200, '10-10-2020', 'Delhi Govt');

```

insert into SubjectScholarshipInformation
(StudentRollNo, ScholarshipName, ScholarshipDescription, ScholarshipAmount,
ScholarshipDate, ScholarshipGivenBy)
values(6, 'XYZ', 'Must be xyz', 320, '10-10-2020', 'Delhi Govt');

insert into SubjectScholarshipInformation
(StudentRollNo, ScholarshipName, ScholarshipDescription, ScholarshipAmount,
ScholarshipDate, ScholarshipGivenBy)
values(7, 'PQR', 'Must be pqr', 30, '10-10-2020', 'Delhi Govt');

insert into SubjectScholarshipInformation
(StudentRollNo, ScholarshipName, ScholarshipDescription, ScholarshipAmount,
ScholarshipDate, ScholarshipGivenBy)
values(8, 'BASIC', 'PhD Candidate must be', 50000, '10-10-2020', 'Delhi Govt');

insert into SubjectScholarshipInformation
(StudentRollNo, ScholarshipName, ScholarshipDescription, ScholarshipAmount,
ScholarshipDate, ScholarshipGivenBy)
values(9, 'ADVANCED', 'PhD Candidate must be', 50500, '10-10-2020', 'Delhi Govt');

insert into SubjectScholarshipInformation
(StudentRollNo, ScholarshipName, ScholarshipDescription, ScholarshipAmount,
ScholarshipDate, ScholarshipGivenBy)
values(10, 'HIGHER', 'PhD Candidate must be', 72000, '10-10-2020', 'Delhi Govt');

```

## Screen Shot:

The screenshot shows the pgAdmin 4 interface with a query editor window. The query is:

```
select * from subjectscholarshipinformation;
```

The results are displayed in a table:

	studentrollno	scholarshipname	scholarshipdescription	scholarshipamount	scholarshipcategory	scholarshipdate	scholarshipgivenby
1	1	JRF	PhD Candidate must be	32000	[null]	2020-10-10	Delhi Govt
2	2	SRF	Any Category	4000	[null]	2020-10-10	State Govt
3	3	SC	Must belong to SC	3000	[null]	2020-10-10	Mumbai Govt
4	4	OBC	Must belong to OBC	2000	[null]	2020-10-10	Central Govt
5	5	ABC	Must be abc	3200	[null]	2020-10-10	Delhi Govt
6	6	XYZ	Must be xyz	320	[null]	2020-10-10	Delhi Govt
7	7	PQR	Must be pqr	30	[null]	2020-10-10	Delhi Govt
8	8	BASIC	PhD Candidate must be	50000	[null]	2020-10-10	Delhi Govt
9	9	ADVANCED	PhD Candidate must be	50500	[null]	2020-10-10	Delhi Govt
10	10	HIGHER	PhD Candidate must be	72000	[null]	2020-10-10	Delhi Govt

A message at the bottom right of the pgAdmin window says: "Successfully run. Total query runtime: 87 msec. 10 rows affected."

**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**

**Query:** update StudentBasicInformation set StudentFatherName = 'father7' where StudentRollNo = 7;

**Screen Shot:**

The screenshot shows the pgAdmin 4 interface with multiple tabs open. The main window displays a query editor with the following SQL code:

```
1 select * from studentbasicinformation;
```

The results pane shows a table with 10 rows of data. The columns are:

studentname	studentsurname	studentrollno	studentaddress	studentfathername	studentphoneno	studentemail
1 Harshit	Kesar	1	Delhi, India	Raju	7827246763	harshit@gmail.com
2 abc	yz	2	Mumbai, India	f2	9827246763	abc@gmail.com
3 def	vwx	3	Delhi, India	f3	8827246763	def@gmail.com
4 ghi	stu	4	Bihar, India	f4	7825246763	ghi@gmail.com
5 jkl	pqr	5	Banglore, India	f5	7827247863	jkl@gmail.com
6 mno	mno	6	Pune, India	f6	7822224676	mno@gmail.com
7 stu	ghi	8	Jammu, India	f8	7829946763	stu@gmail.com
8 vwx	def	9	Gujrat, India	f9	7822246763	vwx@gmail.com
9 yz	abc	10	Delhi, India	f10	7827856763	yz@gmail.com
10 pqr	jkl	7	Chennai, India	father7	7825546763	pqr@gmail.com

A message at the bottom right of the results pane says "Successfully run. Total query runtime: 84 msec. 10 rows affected."

**Query:** update StudentAdmissionPaymentDetails set PaymentMode = 'BankOfBaroda' where PaymentMode = 'BOB';

## Screen Shot:

The screenshot shows the pgAdmin 4 interface with a query editor window. The query is:

```
1 select * from studentadmissionpaymentdetails;
```

The results show 10 rows of data:

	studentrollno	amountpaid	amountbalance	paymentdate	paymentmode	paymentslipno	latecharges
1	2	50000	25000	2020-12-25	UPI	123	2500
2	1	55000	20000	2019-08-20	Paytm	124	250
3	3	40000	35000	2020-01-02	AXIS BANK	125	1700
4	4	30000	45000	2020-02-04	PNB	126	2100
5	5	35000	30000	2020-03-06	CANARA BANK	127	0
6	6	48000	24000	2020-04-08	WALLET	128	25
7	7	16000	2000	2020-05-10	SBI	129	50
8	8	8000	20000	2020-06-15	RBI	130	20
9	9	51000	8000	2020-07-18	DENABANK	131	250
10	10	17000	19000	2020-09-22	BankOfBaroda	132	500

Message bar: ✓ Successfully run. Total query runtime: 97 msec. 10 rows affected.

**Query:** update StudentAdmissionPaymentDetails set PaymentMode = 'CASH' where PaymentMode = 'WALLET';  
**Screen Shot:**

The screenshot shows the pgAdmin 4 interface with a query editor window. The query is:

```
1 select * from studentadmissionpaymentdetails;
```

The results show 10 rows of data, identical to the previous screenshot:

	studentrollno	amountpaid	amountbalance	paymentdate	paymentmode	paymentslipno	latecharges
1	2	50000	25000	2020-12-25	UPI	123	2500
2	1	55000	20000	2019-08-20	Paytm	124	250
3	3	40000	35000	2020-01-02	AXIS BANK	125	1700
4	4	30000	45000	2020-02-04	PNB	126	2100
5	5	35000	30000	2020-03-06	CANARA BANK	127	0
6	7	16000	2000	2020-05-10	SBI	129	50
7	8	8000	20000	2020-06-15	RBI	130	20
8	9	51000	8000	2020-07-18	DENABANK	131	250
9	10	17000	19000	2020-09-22	BankOfBaroda	132	500
10	6	48000	24000	2020-04-08	CASH	128	25

Message bar: ✓ Successfully run. Total query runtime: 84 msec. 10 rows affected.

**Query:** update StudentAdmissionPaymentDetails set PaymentMode = 'ATM' where PaymentMode = 'RBI';

## Screen Shot:

The screenshot shows the pgAdmin 4 interface with a query editor window. The query being run is:

```
1 select * FROM studentadmissionpaymentdetails;
```

The results are displayed in a Data Output tab, showing 10 rows of data from the studentadmissionpaymentdetails table. The columns and their values are:

studentrollno	amountpaid	amountbalance	paymentdate	paymentmode	paymentslipno	latecharges	
1	2	50000	25000	2020-12-25	UPI	123	2500
2	1	55000	20000	2019-08-20	Paytm	124	250
3	3	40000	35000	2020-01-02	AXIS BANK	125	1700
4	4	30000	45000	2020-02-04	PNB	126	2100
5	5	35000	30000	2020-03-06	CANARA BANK	127	0
6	7	16000	2000	2020-05-10	SBI	129	50
7	9	51000	8000	2020-07-18	DENABANK	131	250
8	10	17000	19000	2020-09-22	BankOfBaroda	132	500
9	6	48000	24000	2020-04-08	CASH	128	25
10	8	8000	20000	2020-06-15	ATM	130	20

A message at the bottom right indicates: "Successfully run. Total query runtime: 94 msec. 10 rows affected."

**Query:** update SubjectScholarshipInformation set ScholarshipAmount = 4500 where StudentRollNo = 7;

## Screen Shot:

The screenshot shows the pgAdmin 4 interface with a query editor window. The query being run is:

```
1 select * from subjectscholarshipinformation;
```

The results are displayed in a Data Output tab, showing 10 rows of data from the subjectscholarshipinformation table. The columns and their values are:

studentrollno	scholarshipname	scholarshipdescription	scholarshipamount	scholarshipcategory	scholarshipdate	scholarshipgivenby
1	JRF	PhD Candidate must be	32000	[null]	2020-10-10	Delhi Govt
2	SRF	Any Category	4000	[null]	2020-10-10	State Govt
3	SC	Must belong to SC	3000	[null]	2020-10-10	Mumbai Govt
4	OBC	Must belong to OBC	2000	[null]	2020-10-10	Central Govt
5	ABC	Must be abc	3200	[null]	2020-10-10	Delhi Govt
6	XYZ	Must be xyz	320	[null]	2020-10-10	Delhi Govt
7	BASIC	PhD Candidate must be	50000	[null]	2020-10-10	Delhi Govt
8	ADVANCED	PhD Candidate must be	50500	[null]	2020-10-10	Delhi Govt
9	HIGHER	PhD Candidate must be	72000	[null]	2020-10-10	Delhi Govt
10	PQR	Must be pqr	4500	[null]	2020-10-10	Delhi Govt

A message at the bottom right indicates: "Successfully run. Total query runtime: 83 msec. 10 rows affected."

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

**Query:**

## Screen Shot:

pgAdmin 4

File Object Tools Help

Tables (4)

studentadmissionpaymentdetails

Columns (7)

studentname studentsurname studentrollno studentaddress studentfirstname studentphoneno studentemail

	studentname	studentsurname	studentrollno	studentaddress	studentfirstname	studentphoneno	studentemail
1	Harshit	Kesar	1	Delhi, India	Raju	7827246763	harshit@gmail.com
2	abc	yz	2	Mumbai, India	f2	9827246763	abc@gmail.com
3	def	vwx	3	Delhi, India	f3	8827246763	def@gmail.com
4	ghi	stu	4	Bihar, India	f4	7825246763	ghi@gmail.com
5	jkl	pqr	5	Banglore, India	f5	7827247863	jkl@gmail.com
6	mno	mno	6	Pune, India	f6	7822224676	mno@gmail.com
7	stu	ghi	8	Jammu, India	f8	7829946763	stu@gmail.com
8	vwx	def	9	Gujrat, India	f9	7822246763	vwx@gmail.com
9	yz	abc	10	Delhi, India	f10	7827856763	yz@gmail.com
10	pqr	jkl	7	Chennai, India	father?	7825546763	pqr@gmail.com

Successfully run. Total query runtime: 78 msec. 10 rows affected.

student/postgres@PostgreSQL 13

Query Editor

```
1 SELECT * FROM studentbasicinformation;
```

Data Output Explain Messages Notifications

studentrollno scholarshipname scholarshipdescription scholarshipamount scholarshipcategory scholarshipdate scholarshipgivenby

studentrollno	scholarshipname	scholarshipdescription	scholarshipamount	scholarshipcategory	scholarshipdate	scholarshipgivenby
1	ABC	High Achiever Scholarship	50000	Science	2020-01-01	Harshit
2	DEF	Merit Scholarship	40000	Mathematics	2020-01-01	abc
3	GHI	Community Service Scholarship	35000	Humanities	2020-01-01	def
4	JKL	Sports Scholarship	30000	Sports	2020-01-01	ghi
5	MNO	Arts Scholarship	30000	Arts	2020-01-01	jkl
6	PQR	Science Scholarship	25000	Science	2020-01-01	mno
7	STU	Mathematics Scholarship	20000	Mathematics	2020-01-01	stu
8	VWX	Humanities Scholarship	15000	Humanities	2020-01-01	vwx
9	YZ	Sports Scholarship	15000	Sports	2020-01-01	yz
10	JKL	Arts Scholarship	10000	Arts	2020-01-01	jkl

Successfully run. Total query runtime: 84 msec. 10 rows affected.

student/postgres@PostgreSQL 13

Query Editor

```
1 SELECT * FROM studentadmissionpaymentdetails;
```

Data Output Explain Messages Notifications

studentrollno amountpaid amountbalance paymentdate paymentmode paymentslipno latecharges

studentrollno	amountpaid	amountbalance	paymentdate	paymentmode	paymentslipno	latecharges	
1	2	50000	25000	2020-12-25	UPI	123	2500
2	1	55000	20000	2019-08-20	Paytm	124	250
3	3	40000	35000	2020-01-02	AXIS BANK	125	1700
4	4	30000	45000	2020-02-04	PNB	126	2100
5	5	35000	30000	2020-03-06	CANARA BANK	127	0
6	7	16000	2000	2020-05-10	SBI	129	50
7	9	51000	8000	2020-07-18	DENABANK	131	250
8	10	17000	19000	2020-09-22	BankOfBaroda	132	500
9	6	48000	24000	2020-04-08	CASH	128	25
10	8	8000	20000	2020-06-15	ATM	130	20

Successfully run. Total query runtime: 84 msec. 10 rows affected.

Screenshot of pgAdmin 4 showing the execution of a SQL query to select student subject information.

**Query:**

```
1 SELECT * FROM studentsubjectinformation;
```

**Result:**

subjectopted	studentrollno	subjecttotalmarks	subjectobtainedmarks	studentmarkspercentage	subjectcredits
C++	1	100	98	[null]	4
Java	2	100	55	[null]	4
Python	3	100	45	[null]	4
Javascript	4	100	65	[null]	4
Maven	5	100	75	[null]	4
MySQL	6	100	65	[null]	4
DBMS	7	100	89	[null]	4
NoSQL	8	100	77	[null]	4
OOPS	9	100	63	[null]	4
ECO	10	100	86	[null]	2

**Status:** Successfully run. Total query runtime: 95 msec. 10 rows affected.

Screenshot of pgAdmin 4 showing the execution of a SQL query to select scholarship information.

**Query:**

```
1 SELECT * FROM subjectscholarshipinformation;
```

**Result:**

studentrollno	scholarshipname	scholarshipdescription	scholarshipamount	scholarshipcategory	scholarshipdate	scholarshipscholarshipby
1	JRF	PhD Candidate must be	32000	[null]	2020-10-10	Delhi Govt
2	SRF	Any Category	4000	[null]	2020-10-10	State Govt
3	SC	Must belong to SC	3000	[null]	2020-10-10	Mumbai Govt
4	OBC	Must belong to OBC	2000	[null]	2020-10-10	Central Govt
5	ABC	Must be abc	3200	[null]	2020-10-10	Delhi Govt
6	XYZ	Must be xyz	320	[null]	2020-10-10	Delhi Govt
7	BASIC	PhD Candidate must be	50000	[null]	2020-10-10	Delhi Govt
8	ADVANCED	PhD Candidate must be	50500	[null]	2020-10-10	Delhi Govt
9	HIGHER	PhD Candidate must be	72000	[null]	2020-10-10	Delhi Govt
10	PQR	Must be pqr	4500	[null]	2020-10-10	Delhi Govt

**Status:** Successfully run. Total query runtime: 126 msec. 10 rows affected.

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

**Query:** select \* from SubjectScholarshipInformation where ScholarshipAmount > 5000;

## Screen Shot:

```

1 select * from SubjectScholarshipInformation where ScholarshipAmount > 5000;
2

```

studentrollno	scholarshipname	scholarshipdescription	scholarshipamount	scholarshipcategory	scholarshipdate	scholarshipgivenby
1	JRF	PhD Candidate must be	32000 [null]	2020-10-10	Delhi Govt	
8	BASIC	PhD Candidate must be	50000 [null]	2020-10-10	Delhi Govt	
9	ADVANCED	PhD Candidate must be	50500 [null]	2020-10-10	Delhi Govt	
10	HIGHER	PhD Candidate must be	72000 [null]	2020-10-10	Delhi Govt	

Successfully run. Total query runtime: 97 msec. 4 rows affected.

## 8. Select the students who opted for scholarship but have not got the scholarship.

**Query:** select \* from StudentBasicInformation where StudentRollNo in(select StudentRollNo from SubjectScholarshipInformation where ScholarshipAmount=0);

## Screen Shot:

```

1 select * from StudentBasicInformation where StudentRollNo in
2 (select StudentRollNo from SubjectScholarshipInformation where ScholarshipAmount=0);

```

studentname	studentsurname	studentrollno	studentaddress	studentfirstname	studentphonen	studentemail
-------------	----------------	---------------	----------------	------------------	---------------	--------------

Successfully run. Total query runtime: 89 msec. 0 rows affected.

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

**Query:** CREATE OR REPLACE PROCEDURE public.StudentMarksPercentage

(RollNo IN numeric, TotalMarks IN numeric, ObtainedMarks IN numeric, MarksPercentage INOUT numeric)

LANGUAGE 'plpgsql'

AS \$BODY\$

DECLARE

BEGIN

MarksPercentage = (ObtainedMarks / TotalMarks) \* 100;

UPDATE StudentSubjectInformation SET StudentMarksPercentage = MarksPercentage  
where StudentRollNo = RollNo;

END;

\$BODY\$

**Screen Shot:**

```

CREATE OR REPLACE PROCEDURE public.StudentMarksPercentage
(RollNo IN numeric, TotalMarks IN numeric, ObtainedMarks IN numeric, MarksPercentage INOUT numeric)
LANGUAGE 'plpgsql'
AS $BODY$
DECLARE
BEGIN
    MarksPercentage = (ObtainedMarks / TotalMarks) * 100;
    UPDATE StudentSubjectInformation SET StudentMarksPercentage = MarksPercentage where StudentRollNo = RollNo;
END;
$BODY$

```

subjectid	subjectname	studentrollno	subjecttotalmarks	subjectobtainedmarks	studentmarkspercentage	subjectcredits
1	Cpp	1	100	98	98	4
2	Java	2	100	55	55	4
3	Python	3	100	45	45	4
4	Javascript	4	100	65	65	4
5	Maven	5	100	75	75	4
6	MySQL	6	100	65	65	4
7	DBMS	7	100	89	89	4
8	NoSQL	8	100	77	77	4
9	OOPS	9	100	63	63	4
10	ECO	10	100	86		

Successfully run. Total query runtime: 84 msec. 10 rows affected.

**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:** CREATE OR REPLACE PROCEDURE public.ScholarshipCategory(RollNo IN numeric, MarksPercentage INOUT numeric)

LANGUAGE 'plpgsql'

```

AS $BODY$
DECLARE
BEGIN
    SELECT StudentMarksPercentage INTO MarksPercentage from
StudentSubjectInformation where StudentRollNo = RollNo;

    IF MarksPercentage >= 80 THEN
        UPDATE SubjectScholarshipInformation SET ScholarshipCategory =
'PostMetric' where StudentRollNo = RollNo;
    ELSE
        UPDATE SubjectScholarshipInformation SET ScholarshipCategory = 'PreMetric'
where StudentRollNo = RollNo;
    END IF;
END;
$BODY$

```

### Screen Shot:

```

CREATE OR REPLACE PROCEDURE public.ScholarshipCategory(RollNo IN numeric, MarksPercentage INOUT numeric)
LANGUAGE 'plpgsql'
AS $BODY$
DECLARE
BEGIN
    SELECT StudentMarksPercentage INTO MarksPercentage from StudentSubjectInformation where StudentRollNo = RollNo;
    IF MarksPercentage >= 80 THEN
        UPDATE SubjectScholarshipInformation SET ScholarshipCategory = 'PostMetric' where StudentRollNo = RollNo;
    ELSE
        UPDATE SubjectScholarshipInformation SET ScholarshipCategory = 'PreMetric';
    END IF;
END;
$BODY$

```

studentrollno	scholarshipname	scholarshipdescription	scholarshipamount	scholarshipcategory	scholarshipdate	scholarshipgivenby
1	JRF	PhD Candidate must be	32000	PostMetric	2020-10-10	Delhi Govt
2	SRF	Any Category	4000	PreMetric	2020-10-10	State Govt
3	SC	Must belong to SC	3000	PreMetric	2020-10-10	Mumbai Govt
4	OBC	Must belong to OBC	2000	PreMetric	2020-10-10	Central Govt
5	ABC	Must be abc	3200	PreMetric	2020-10-10	Delhi Govt
6	XYZ	Must be xyz	320	PreMetric	2020-10-10	Delhi Govt
7	PQR	Must be pqr	4500	PostMetric	2020-10-10	Delhi Govt
8	BASIC	PhD Candidate must be	50000	PreMetric	2020-10-10	Delhi Govt
9	ADVANCED	PhD Candidate must be	50500	PreMetric	2020-10-10	Delhi Govt
10	HIGHER	PhD Candidate must be	72000	PostMetric	2020-10-10	Delhi Govt

Successfully run. Total query runtime: 87 msec. 10 rows affected.

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

**Query:** CREATE MATERIALIZED VIEW StudentBalance  
AS SELECT StudentName, StudentSurname, SBInfo.StudentRollNo, StudentAddress,  
StudentFatherName, StudentPhoneNo, StudentEmail, AmountBalance

FROM (StudentBasicInformation SBInfo JOIN StudentAdmissionPaymentDetails SAPD ON  
(SBInfo.StudentRollNo = SAPD.StudentRollNo));

SELECT \* FROM StudentBalance;

### Screen Shot:

The screenshot shows the pgAdmin 4 interface with multiple tabs open. The main window displays a query editor with the following SQL code:

```
1 select * from studentbalance;
```

The results pane shows a table with the following data:

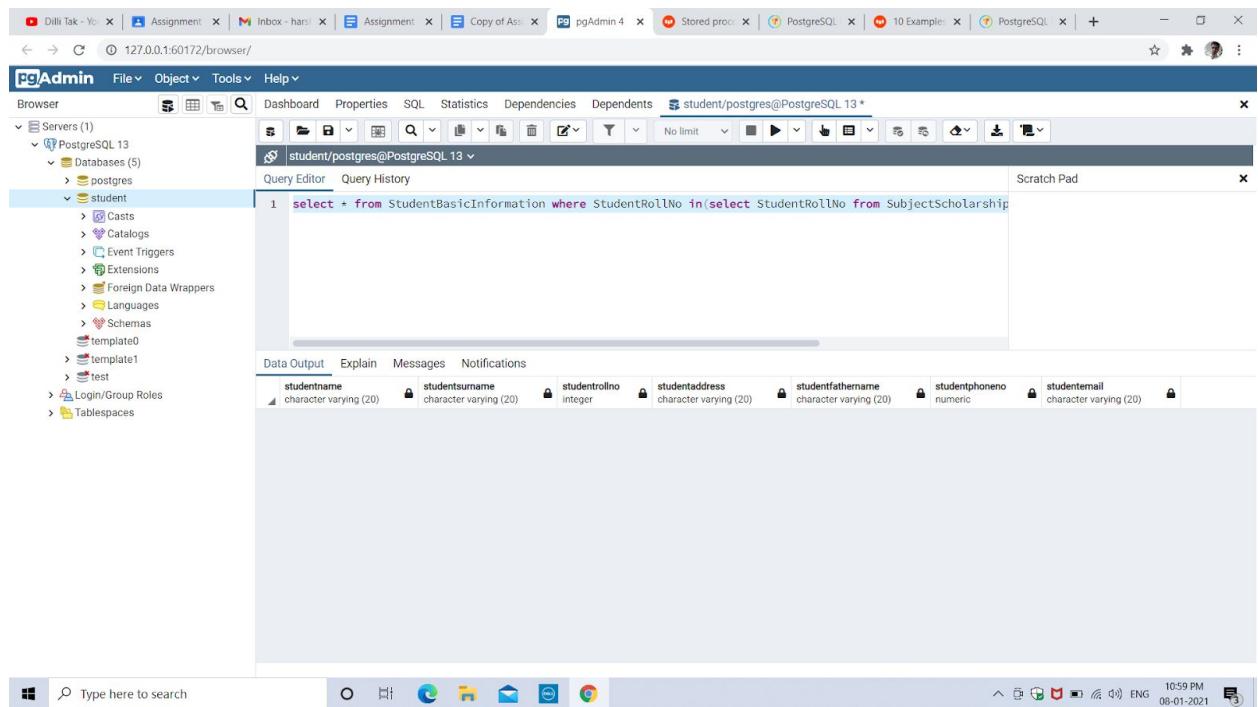
	studentname	studentsurname	studentrollno	studentaddress	studentfathername	studentphoneno	studentemail	amountbalance
1	abc	yz		Mumbai, India	Raju	9827246763	abc@gmail.com	
2	Harshit	Kesar		Delhi, India	Raju	7827246763	harshit@gmail.com	
3	def	vwx		Delhi, India	f3	8827246763	def@gmail.com	
4	ghi	stu		Bihar, India	f4	7825246763	ghi@gmail.com	
5	jkl	pqr		Banglore, India	f5	7827247863	jkl@gmail.com	
6	pqr	jkł		Chennai, India	father7	7825546763	pqr@gmail.com	
7	vwx	def		Gujrat, India	f9	78222456763	vwx@gmail.com	
8	yz	abe		Delhi, India	f10	7827856763	yz@gmail.com	
9	mno	mno		Pune, India	f6	7822224676	mno@gmail.com	
10	stu	ghi		Jammu, India	f8	7829946763	stu@gmail.com	

A message at the bottom right of the results pane says "Successfully run. Total query runtime: 84 msec. 10 rows affected."

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

**Query:** select \* from StudentBasicInformation where StudentRollNo in(select StudentRollNo from SubjectScholarshipInformation where ScholarshipAmount=0);

## Screen Shot:



**13. Create a Stored Procedure which will 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:** CREATE OR REPLACE PROCEDURE public.CheckBalanceByRollNo(RollNo IN numeric, Balance INOUT numeric)

LANGUAGE 'plpgsql'

AS \$BODY\$

DECLARE

BEGIN

    SELECT AmountBalance INTO Balance from StudentAdmissionPaymentDetails where RollNo = StudentRollNo;

END;

\$BODY\$

CALL public.CheckBalanceByRollNo(1, -1)

CALL public.CheckBalanceByRollNo(7, -1)

## Screen Shot:

The screenshots show the pgAdmin 4 interface with three tabs open in the browser:

- Assignment - SQL Concepts
- Assignment - SQL Concepts
- Copy of Assignment - SQL Concepts

The pgAdmin window shows the following details:

- Servers (1)**: PostgreSQL 13
- Databases (3)**: postgres, student, test
- student** database selected.
- Query Editor** tab: Shows the creation of a stored procedure named `CheckBalanceByRollNo`. The code is as follows:

```
1 CREATE OR REPLACE PROCEDURE public.CheckBalanceByRollNo(RollNo IN numeric, Balance INOUT numeric)
2 LANGUAGE 'plpgsql'
3 AS $BODY$
4 DECLARE
5 BEGIN
6     SELECT AmountBalance INTO Balance from StudentAdmissionPaymentDetails where RollNo = StudentRollNo;
7 END;
8 $BODY$
```

- Data Output** tab: Shows the result of running the stored procedure with parameters (1, -1). The output is:

balance
20000

- Notifications** tab: Shows a success message: "Successfully run. Total query runtime: 42 msec. 1 rows affected."

The second screenshot shows the same setup but with parameters (7, -1), resulting in a balance of 2000.

The third screenshot shows the same setup but with parameters (1, -1), resulting in a balance of 20000.

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

**Query:** `SELECT * FROM StudentBasicInformation WHERE StudentRollNo IN (SELECT StudentRollNo FROM StudentSubjectInformation ORDER BY StudentMarksPercentage DESC LIMIT 5);`

## Screen Shot:

The screenshot shows the pgAdmin 4 interface. On the left, the sidebar displays the 'Servers' section with 'PostgreSQL 13' selected, showing databases like 'postgres', 'student', and 'test'. The main area is a 'Query Editor' window with the following SQL code:

```
1 SELECT * FROM StudentBasicInformation WHERE StudentRollNo IN(
2     SELECT StudentRollNo FROM StudentSubjectInformation ORDER BY StudentMarksPercentage DESC LIMIT 5);
```

Below the code, the 'Data Output' tab is selected, displaying a table with 10 rows of student information. The columns are: studentname, studentsurname, studentrollno, studentaddress, studentfathername, studentphoneno, and studentemail. The data is as follows:

	studentname	studentsurname	studentrollno	studentaddress	studentfathername	studentphoneno	studentemail
1	Harshit	Kesar		1 Delhi, India	Raju	7827246763	harshit@gmail.com
2	jkL	pqr		5 Bangalore, India	f5	7827247863	jkL@gmail.com
3	stu	ghi		8 Jammu, India	f8	7829946763	stu@gmail.com
4	yz	abc		10 Delhi, India	f10	7827856763	yz@gmail.com
5	pqr	jkL		7 Chennai, India	father7	7825546763	pqr@gmail.com

A green success message at the bottom right of the data output area states: 'Successfully run. Total query runtime: 57 msec. 5 rows affected.'

**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).**

### 1. NATURAL JOIN :-

A natural join is a join that creates an implicit join based on the same column names in the joined tables.

Here, we can join the tables `StudentBasicInformation` with the `StudentSubjectInformation` as both of the tables contain the field `StudentRollNo` with the exact same name in both the tables. So, it automatically joins both tables on the basis of `StudentRollNo` field and generates 10 records as we could get on merging on using INNER JOIN query. For more information just see the SCREENSHOT below.

```

1 SELECT * FROM studentbasicinformation NATURAL JOIN studentsubjectinformation;

```

studentrollno	studentname	studentsurname	studentaddress	studentfathername	studentphono	studentemail	subjectopted
1	Harshit	Kesar	Delhi, India	Raju	7827246763	harshit@gmail.com	Cpp
2	abc	yz	Mumbai, India	f2	9827246763	abc@gmail.com	Java
3	def	vwx	Delhi, India	f3	8827246763	def@gmail.com	Python
4	ghi	stu	Bihar, India	f4	7825246763	ghi@gmail.com	Javascript
5	jkl	pqr	Banglore, India	f5	7827247863	jkl@gmail.com	Maven
6	mno	mno	Pune, India	f6	7822224676	mno@gmail.com	MySQL
7	pqr	jkl	Chennai, India	father7	7825546763	pqr@gmail.com	DBMS
8	stu	ghi	Jammu, India	f8	7829946763	stu@gmail.com	NoSQL
9	vwx	def	Gujrat, India	f9	7822246763	vwx@gmail.com	OOPS
10	yz	abc	Delhi, India	f10	7827856763	yz@gmail.com	ECO

Successfully run. Total query runtime: 80 msec. 10 rows affected.

2. INNER JOIN : Here, we have to explicitly define the fields on which you want to merge 2 or more tables.

Remember here you will get all the fields of all the tables. Whereas in case of NATURAL JOIN you get the StudentRollNo field at once but here you get two times. For more information see the SCREENSHOT below.

```

1 SELECT * FROM studentbasicinformation AS t1 INNER JOIN studentsubjectinformation AS t2 ON t1.StudentRollNo = t2.StudentRollNo;

```

studentname	studentsurname	studentrollno	studentaddress	studentfathername	studentphono	studentemail	subjectopted
Harshit	Kesar	1	Delhi, India	Raju	7827246763	harshit@gmail.com	Cpp
abc	yz	2	Mumbai, India	f2	9827246763	abc@gmail.com	Java
def	vwx	3	Delhi, India	f3	8827246763	def@gmail.com	Python
ghi	stu	4	Bihar, India	f4	7825246763	ghi@gmail.com	Javascript
jkl	pqr	5	Banglore, India	f5	7827247863	jkl@gmail.com	Maven
mno	mno	6	Pune, India	f6	7822224676	mno@gmail.com	MySQL
pqr	jkl	7	Chennai, India	father7	7825546763	pqr@gmail.com	DBMS
stu	ghi	8	Jammu, India	f8	7829946763	stu@gmail.com	NoSQL
vwx	def	9	Gujrat, India	f9	7822246763	vwx@gmail.com	OOPS
yz	abc	10	Delhi, India	f10	7827856763	yz@gmail.com	ECO

Successfully run. Total query runtime: 96 msec. 10 rows affected.

3. LEFT OUTER JOIN : - To select rows from the left table that do not have matching rows in the right table, you use the left join with a **where** clause.

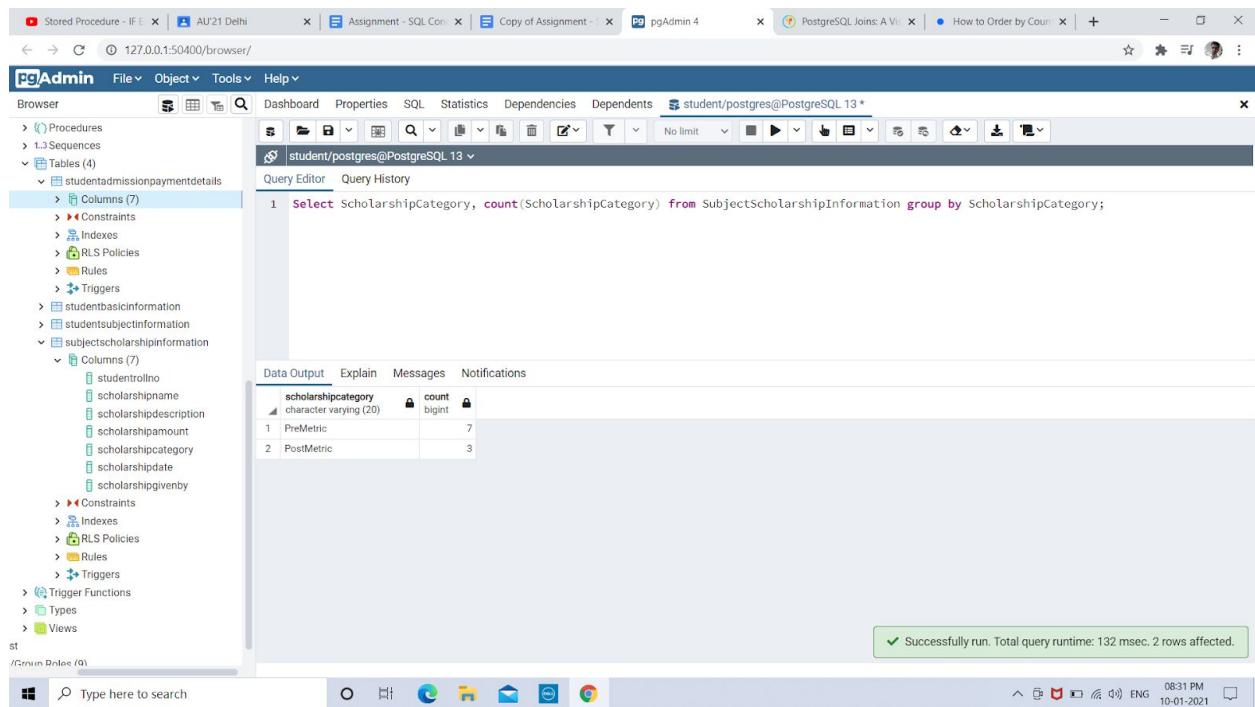
#### **16. Mention the differences between the delete, drop and truncate commands.**

Delete	Drop	Truncate
DML Command	DDL Command	DDL Command
It is used to delete the one or more tuples of a table.	It is used to drop the whole table.	It is used to delete all the rows of a relation (table) in one go. With the help of the “TRUNCATE” command we can’t delete the single row as here WHERE clause is not used.
DELETE FROM table_name; (or) DELETE FROM table_name WHERE condition;	DROP TABLE table_name;	TRUNCATE table_name;
It is comparatively slower than TRUNCATE cmd.	By using this command the existence of the whole table is finished or lost.	It is comparatively faster than delete command as it deletes all the rows fastly.
Here we can use the “ROLLBACK” command to restore the tuple.	Here we can’t restore the table by using the “ROLLBACK” command.	Here we can’t restore the tuples of the table by using the “ROLLBACK” command.

#### **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.**

**Query:** Select ScholarshipCategory, count(ScholarshipCategory) from SubjectScholarshipInformation group by ScholarshipCategory;

## Screen Shot:



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

**Query:** Select t1.ScholarshipCategory, MAX(t1.StudentCount) from  
(Select ScholarshipCategory, count(ScholarshipCategory) as StudentCount  
from SubjectScholarshipInformation group by ScholarshipCategory) AS t1  
Group by ScholarshipCategory limit 1;

## Screen Shot:

The screenshot shows the pgAdmin 4 interface. On the left, the Object Browser displays a database structure with tables like 'studentadmissionpaymentdetails' and 'subjectscholarshipinformation'. The 'Query Editor' tab is active, containing the following SQL query:

```

1 Select t1.ScholarshipCategory, MAX(t1.StudentCount) from
2   (Select ScholarshipCategory, count(ScholarshipCategory) as StudentCount
3    from SubjectScholarshipInformation group by ScholarshipCategory) AS t1
4 Group by ScholarshipCategory limit 1;

```

The 'Data Output' tab shows the results of the query:

scholarshipcategory	max
PreMetric	7

A green success message at the bottom right indicates: "Successfully run. Total query runtime: 98 msec. 1 rows affected."

**ANOTHER SIMPLE WAY OF DOING THE ABOVE QUERY IS :-**

**Query:** Select ScholarshipCategory, count(ScholarshipCategory) as StudentCount

from SubjectScholarshipInformation group by ScholarshipCategory limit 1;

## Screen Shot:

This screenshot shows the same pgAdmin 4 interface with a simplified query in the 'Query Editor' tab:

```

1 Select ScholarshipCategory, count(ScholarshipCategory) as StudentCount
2   from SubjectScholarshipInformation group by ScholarshipCategory limit 1;

```

The 'Data Output' tab shows the results:

scholarshipcategory	studentcount
PreMetric	7

A green success message at the bottom right indicates: "Successfully run. Total query runtime: 53 msec. 1 rows affected."

**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.**

**Query:**

```
SELECT t1.StudentRollNo, t2.StudentMarksPercentage, Max(ScholarshipAmount)
FROM
(select * from SubjectScholarshipInformation) as t1
NATURAL JOIN
(select * from StudentSubjectInformation where SubjectObtainedMarks in
(select max(SubjectObtainedMarks) from StudentSubjectInformation)) as t2
GROUP BY t1.studentrollno, t2.StudentMarksPercentage;
```

**Screen Shot:**

The screenshot shows the pgAdmin 4 interface. On the left, the object browser displays several tables: studentadmissionpaymentdetails, studentbasicinformation, studentsubjectinformation, and subjectscholarshipinformation. The subjectscholarshipinformation table is expanded, showing columns: studentrollno, scholarshipname, scholarshipdescription, scholarshipamount, scholarshipcategory, scholarshipdate, and scholarshipgivenby. The main window shows a query editor with the following SQL code:

```
1 Select t1.StudentRollNo, t2.StudentMarksPercentage, Max(ScholarshipAmount) from
2 (select * from SubjectScholarshipInformation) as t1
3 NATURAL JOIN
4 (select * from StudentSubjectInformation where SubjectObtainedMarks in
5 (select max(SubjectObtainedMarks) from StudentSubjectInformation)) as t2
6 GROUP BY t1.studentrollno, t2.StudentMarksPercentage;
```

The data output pane shows the results:

studentrollno	studentmarkspercentage	max
1	98	32000

A green message bar at the bottom right indicates: "Successfully run. Total query runtime: 49 msec. 1 rows affected."

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

Key	Triggers	Stored procedures
Basic	Trigger is a stored procedure that runs automatically when various events happen (eg update, insert, delete).	Stored procedures are pieces of the code written in PL/SQL to do some specific task.
Running Methodology	It can execute automatically based on the events.	It can be invoked explicitly by the user.

Parameter	It can not take input as a parameter.	It can take input as a parameter.
Transaction statements	we can't use transaction statements inside a trigger.	We can use transaction statements like begin transaction, commit transaction, and rollback inside a stored procedure.
Return	Triggers can not return values.	Stored procedures can return values.

**Views** and **User-Defined Functions** almost serve the same purpose. But the major difference is that User-Defined Function can accept parameters, whereas Views cannot. And also the output of the User Defined Function can be directly used in the SELECT clause, whereas you cannot do it with a View.

---

**Thank you. All The Best. Enjoy The Assignment.**