SQL ASSIGNMENT

Assumptions:

- Each Student can only opt for one subject from Physics, Chemistry, Mathematics, English, Computer Science and will be charged with Admission Fees of 10000, 8000, 12000, 6000 and 15000 respectively.
- There are 3 categories for scholarship: High, Medium and Low.
- Students with >95% will get Scholarship of High Category amounted to 8000.
- Students with >90% and <=95% will get Medium Category Scholarship amounted to 6000.
- Students with >80% and <=90% will get Scholarship of Low Category amounted to 3000.

Create database Student;

TABLE 1 : StudentBasicInformation

```
Create table StudentBasicInformation
(
        StudentName varchar(30) NOT NULL,
        StudentSurname varchar(30),
        StudentRollNo int primary key,
        StudentAddress varchar(50),
        StudentGender char(1) not null,
        StudentAge int NOT NULL,
        StudentGuardianName varchar(50),
        Check (StudentAge>0)
)
```

Insert into StudentBasicInformation values('Abhishek','Gupta',1,'Sushant Lok,Gurgaon','M',20,'Alok Gupta');

Insert into StudentBasicInformation values('Tushar','Maheshwari',2,'Jagdish Nagar,Noida','M',21,'Ghanshyam Maheshwari');

```
Insert into StudentBasicInformation values('Architta','Ahuja',3,'Indravihar,Mukherjee
Nagar','F',20,'Navya Ahuja');

Insert into StudentBasicInformation values('Harshit','Saini',4,'Arya
Nagar,Rohini','M',22,'Akshat Saini');

Insert into StudentBasicInformation values('Divya','Gupta',5,'Sector
58,Gurgaon','F',23,'Shivani Gupta');

Insert into StudentBasicInformation values('Sonali','Goyal',6,'Naveen
Colony,Shahdara','F',23,'Pawan Goyal');

Insert into StudentBasicInformation values('Sonali','Raghuvansi',7,'Block
C2,Keshavpuram','F',21,'Sunita Raghuvanshi');

Insert into StudentBasicInformation values('Rishabh','Kanojia',8,'Block A,Govind
Puri','M',20,'Rajat Kanojia');

Insert into StudentBasicInformation values('Mayank','Singh',9,'Sushant
Lok,Gurgaon','M',22,'Abhishek Singh');

Insert into StudentBasicInformation values('Ashish','Sharma',10,'Sector
10,Noida','M',23,'Vipul Sharma');

Insert into StudentBasicInformation values('Shrishti','Tyagi',11,'Sushant
Lok,Gurgaon','F',21,'Pawan Tyagi');

Insert into StudentBasicInformation values('Reeta','Sachdeva',12,'Colony No 3,Lajpat
Nagar','F',22,'Aditi Sachdeva');

Insert into StudentBasicInformation values('Yasmeen','Grover',13,'Bharat
Lok,Noida','F',20,'Roopa Grover');

Insert into StudentBasicInformation values('Satakshi','Aggarwal',14,'chandra
Lok,Gurgaon','f',22,'Aditya Aggarwal');

Insert into StudentBasicInformation values('Shubham','Yadav',15,'Sector
41,Gurgaon','M',23,'Janendra Yadav');


select * from StudentBasicInformation;
```

Data Output

	studentname character varying (30)	studentsurname character varying (30)	studentrollno [PK] integer	studentaddress character varying (50)	studentgender character (1)	studentage integer	studentguardianname character varying (50)
1	Abhishek	Gupta	1	Sushant Lok,Gurgaon	M	20	Alok Gupta
2	Tushar	Maheshwari	2	Jagdish Nagar,Noida	M	21	Ghanshyam Maheshwari
3	Architta	Ahuja	3	Indravihar,Mukherjee Nagar	F	20	Navya Ahuja
4	Harshit	Saini	4	Arya Nagar,Rohini	M	22	Akshat Saini
5	Divya	Gupta	5	Sector 58,Gurgaon	F	23	Shivani Gupta
6	Sonali	Goyal	6	Naveen Colony,Shahdara	F	23	Pawan Goyal
7	Sonali	Raghuvansi	7	Block C2,Keshavpuram	F	21	Sunita Raghuvanshi
8	Rishabh	Kanojia	8	Block A,Govind Puri	M	20	Rajat Kanojia
9	Mayank	Singh	9	Sushant Lok,Gurgaon	M	22	Abhishek Singh
10	Ashish	Sharma	10	Sector 10,Noida	M	23	Vipul Sharma
11	Shrishti	Tyagi	11	Sushant Lok,Gurgaon	F	21	Pawan Tyagi
12	Reeta	Sachdeva	12	Colony No 3,Lajpat Nagar	F	22	Aditi Sachdeva
13	Yasmeen	Grover	13	Bharat Lok,Noida	F	20	Roopa Grover
14	Satakshi	Aggarwal	14	chandra Lok,Gurgaon	f	22	Aditya Aggarwal
15	Shubham	Yadav	15	Sector 41,Gurgaon	M	23	Janendra Yadav

TABLE 2: StudentAdmissionPaymentDetails

```
Create table StudentAdmissionPaymentDetails
(
        StudentRollNo int primary key,
        AmountPaid float ,
        AmountBalance float,
        PaymentMode varchar(20),
        PaymentDate date,
        ReceiptNumber varchar(20),
        Fine float,
        foreign key(StudentRollNo) references StudentBasicInformation(StudentRollNo)
);
```

```
Insert into StudentAdmissionPaymentDetails values(1,8000,2000,'Online','2020-12-20','12356781',30);

Insert into StudentAdmissionPaymentDetails values(2,7000,5000,'Online','2020-12-22','12259787',80);

Insert into StudentAdmissionPaymentDetails values(3,8000,0,'Online','2020-12-10','12222444',0);

Insert into StudentAdmissionPaymentDetails values(4,4000,6000,'Cash','2020-12-25','12353338',150);

Insert into StudentAdmissionPaymentDetails values(5,13000,2000,'Online','2020-12-20','12567854',50);

Insert into StudentAdmissionPaymentDetails values(6,10000,2000,'Cash','2020-12-28','54678932',90);

Insert into StudentAdmissionPaymentDetails values(7,5000,1000,'Online','2020-12-20','34567234',50);

Insert into StudentAdmissionPaymentDetails values(8,1000,5000,'Cash','2020-12-23','23467893',150);

Insert into StudentAdmissionPaymentDetails values(9,15000,0,'Cash','2020-12-20','53456784',0);

Insert into StudentAdmissionPaymentDetails values(10,12000,3000,'Online','2020-12-20','31245690',50);

Insert into StudentAdmissionPaymentDetails values(11,9000,3000,'Online','2020-12-20','36784312',30);

Insert into StudentAdmissionPaymentDetails values(12,7000,1000,'Online','2020-12-27','76589543',70);

Insert into StudentAdmissionPaymentDetails values(13,1000,14000,'Online','2020-12-20','75645678',50);

Insert into StudentAdmissionPaymentDetails values(14,8000,7000,'Cash','2020-12-20','95412306',50);

Insert into StudentAdmissionPaymentDetails values(15,12000,0,'Online','2020-12-20','85471200',0);


select * from StudentAdmissionPaymentDetails;
```

Data Output Explain Messages Notifications

	studentrollno [PK] integer	amountpaid double precision	amountbalance double precision	paymentmode character varying (20)	paymentdate date	receiptnumber character varying (20)	fine double precision
1	1	5000	3000	Online	2020-12-20	12356781	50
2	2	7000	5000	Online	2020-12-22	12259787	80
3	3	8000	0	Online	2020-12-10	12222444	0
4	4	4000	6000	Cash	2020-12-25	12353338	150
5	5	13000	2000	Online	2020-12-20	12567854	50
6	6	10000	2000	Cash	2020-12-28	54678932	90
7	7	5000	1000	Online	2020-12-20	34567234	50
8	8	1000	5000	Cash	2020-12-23	23467893	150
9	9	15000	0	Cash	2020-12-20	53456784	0
10	10	12000	3000	Online	2020-12-20	31245690	50
11	11	9000	3000	Online	2020-12-20	36784312	30
12	12	7000	1000	Online	2020-12-27	76589543	70
13	13	1000	14000	Online	2020-12-20	75645678	50
14	14	8000	7000	Cash	2020-12-20	95412306	50
15	15	12000	0	Online	2020-12-20	85471200	0

TABLE 3: StudentSubjectInformation

```
create table StudentSubjectInformation
(
        SubjectOpted varchar(20),
        StudentRollNo int primary key,
        SubjectTotalMarks int,
        SubjectObtainedMarks float,
        StudentMarksPercentage float,
        ScholarshipOpted char(1),
        foreign key(StudentRollNo) references StudentBasicInformation(StudentRollNo),
        check (SubjectTotalMarks>=SubjectObtainedMarks)
);

Insert into StudentSubjectInformation values('Physics',1,100,90,90,'Y');
Insert into StudentSubjectInformation values('Mathematics',2,100,80,80,'N');
Insert into StudentSubjectInformation values('Chemistry',3,100,50,50,'N');
```

```
Insert into StudentSubjectInformation values('Physics',4,100,85.5,85.5,'Y');
Insert into StudentSubjectInformation values('Computer Science',5,100,60,60,'N');
Insert into StudentSubjectInformation values('Mathematics',6,100,92,92,'Y');
Insert into StudentSubjectInformation values('English',7,100,94,94,'Y');
Insert into StudentSubjectInformation values('English',8,100,88,88,'Y');
Insert into StudentSubjectInformation values('Computer Science',9,100,70,70,'N');
Insert into StudentSubjectInformation values('Computer Science',10,100,75,75,'N');
Insert into StudentSubjectInformation values('Mathematics',11,100,84,84,'N');
Insert into StudentSubjectInformation values('Chemistry',12,100,85,85,'Y');
Insert into StudentSubjectInformation values('Computer Science',13,100,68,68,'N');
Insert into StudentSubjectInformation values('Computer Science',14,100,81,81,'N');
Insert into StudentSubjectInformation values('Mathematics',15,100,98,98,'Y');
```

```
select * from StudentSubjectInformation;
```

Data Output Explain Messages Notifications

	subjectopted character varying (20)	studentrollno [PK] integer	subjecttotalmarks integer	subjectobtainedmarks double precision	studentmarkspercentage double precision	scholarshipopted character (1)
1	Physics	1	100	90	90	Y
2	Mathematics	2	100	80	80	N
3	Chemistry	3	100	50	50	N
4	Physics	4	100	85.5	85.5	Y
5	Computer Science	5	100	60	60	N
6	Mathematics	6	100	92	92	Y
7	English	7	100	94	94	Y
8	English	8	100	88	88	Y
9	Computer Science	9	100	70	70	N
10	Computer Science	10	100	75	75	N
11	Mathematics	11	100	84	84	N
12	Chemistry	12	100	85	85	Y
13	Computer Science	13	100	68	68	N
14	Computer Science	14	100	81	81	N
15	Mathematics	15	100	98	98	Y

TABLE 4: SubjectScholarshipInformation

```
create table SubjectScholarshipInformation
```

```
(
        StudentRollNo int primary key,
        ScholarshipName Varchar(30),
        ScholarshipDescription varchar(20),
        ScholarshipAmount float,
        ScholarshipCategory varchar(20),
        ScholarshipStatus varchar(20),
        AmountProcessed char(1),
        foreign key(StudentRollNo) references StudentBasicInformation(StudentRollNo),
        check (ScholarshipAmount>0)
);
```

Insert into SubjectScholarshipInformation values(1,'PhysicsSchlorship','Excellence',6000,'Medium','pending','N');

Insert into SubjectScholarshipInformation values(4,'PhysicsSchlorship','Very Good',3000,'Low','Approved','Y');

Insert into SubjectScholarshipInformation values(6,'MathematicsSchlorship','Excellence',6000,'Medium','Approved','Y');

Insert into SubjectScholarshipInformation values(7,'EnglishSchlorship','Excellence',6000,'Medium','pending','N');

Insert into SubjectScholarshipInformation values(8,'EnglishSchlorship','Very Good',3000,'Low','Approved','Y');

Insert into SubjectScholarshipInformation values(12,'ChemistrySchlorship','Very Good',3000,'Low','pending','N');

Insert into SubjectScholarshipInformation values(15,'MathematicsSchlorship','Outstanding',8000,'High','Approved','Y');

select * from SubjectScholarshipInformation;

NOTE: There are 7 rows in this table as I have only 7 Students Out of 15 Students who opted for Scholarship from above table records.

Data Output Explain Messages Notifications

	studentrollno [PK] integer	scholarshipname character varying (30)	scholarshipdescription character varying (20)	scholarshipamount double precision	scholarshipcategory character varying (20)	scholarshipstatus character varying (20)	amountp characte
1	1	PhysicsSchlorship	Excellence	6000	Medium	pending	N
2	4	PhysicsSchlorship	Very Good	3000	Low	Approved	Y
3	6	MathematicsSchlorship	Excellence	6000	Medium	Approved	Y
4	7	EnglishSchlorship	Excellence	6000	Medium	pending	N
5	8	EnglishSchlorship	Very Good	3000	Low	Approved	Y
6	12	ChemistrySchlorship	Very Good	3000	Low	pending	N
7	15	MathematicsSchlorship	Outstanding	8000	High	Approved	Y

UPDATING 5 RECORDS :

1. update StudentBasicInformation

set StudentAddress='Sector 20,Rohini'

where StudentRollNo=1;

2. update StudentBasicInformation

set StudentAge=24

where StudentRollNo=10;

TABLE StudentBasicInformation AFTER UPDATION

Student/postgres@PostgreSQL 13

Data Output Explain Messages Notifications

	studentname character varying (30)	studentsurname character varying (30)	studentrollno [PK] integer	studentaddress character varying (50)	studentgender character (1)	studentage integer	studentguardianname character varying (50)
1	Tushar	Maheshwari	2	Jagdish Nagar,Noida	M	21	Ghanshyam Maheshwari
2	Architta	Ahuja	3	Indravihar,Mukherjee Nagar	F	20	Navya Ahuja
3	Harshit	Saini	4	Arya Nagar,Rohini	M	22	Akshat Saini
4	Divya	Gupta	5	Sector 58,Gurgaon	F	23	Shivani Gupta
5	Sonali	Goyal	6	Naveen Colony,Shahdara	F	23	Pawan Goyal
6	Sonali	Raghuvansi	7	Block C2,Keshavpuram	F	21	Sunita Raghuvanshi
7	Rishabh	Kanojia	8	Block A,Govind Puri	M	20	Rajat Kanojia
8	Mayank	Singh	9	Sushant Lok,Gurgaon	M	22	Abhishek Singh
9	Shrishti	Tyagi	11	Sushant Lok,Gurgaon	F	21	Pawan Tyagi
10	Reeta	Sachdeva	12	Colony No 3,Lajpat Nagar	F	22	Aditi Sachdeva
11	Yasmeen	Grover	13	Bharat Lok,Noida	F	20	Roopa Grover
12	Satakshi	Aggarwal	14	chandra Lok,Gurgaon	f	22	Aditya Aggarwal
13	Shubham	Yadav	15	Sector 41,Gurgaon	M	23	Janendra Yadav
14	Abhishek	Gupta	1	Sector 20,Rohini	M	20	Alok Gupta
15	Ashish	Sharma	10	Sector 10,Noida	M	24	Vipul Sharma

Activate Windows

3. update StudentAdmissionPaymentDetails

   set AmountPaid=8000,AmountBalance=0,fine=0

   where StudentRollNo=1;

4. update StudentAdmissionPaymentDetails

   set fine=140

   where StudentRollNo=4;

5. update StudentAdmissionPaymentDetails

   set PaymentDate='2020-12-15'

   where StudentRollNo=3;

TABLE StudentAdmissionPaymentDetails AFTER UPDATION

Student/postgres@PostgreSQL 13

Data Output Explain Messages Notifications

	studentrollno [PK] integer	amountpaid double precision	amountbalance double precision	paymentmode character varying (20)	paymentdate date	receiptnumber character varying (20)	fine double precision
1	2	7000	5000	Online	2020-12-22	12259787	80
2	5	13000	2000	Online	2020-12-20	12567854	50
3	6	10000	2000	Cash	2020-12-28	54678932	90
4	7	5000	1000	Online	2020-12-20	34567234	50
5	8	1000	5000	Cash	2020-12-23	23467893	150
6	9	15000	0	Cash	2020-12-20	53456784	0
7	10	12000	3000	Online	2020-12-20	31245690	50
8	11	9000	3000	Online	2020-12-20	36784312	30
9	12	7000	1000	Online	2020-12-27	76589543	70
10	13	1000	14000	Online	2020-12-20	75645678	50
11	14	8000	7000	Cash	2020-12-20	95412306	50
12	15	12000	0	Online	2020-12-20	85471200	0
13	1	8000	0	Online	2020-12-20	12356781	0
14	4	4000	6000	Cash	2020-12-25	12353338	140
15	3	8000	0	Online	2020-12-15	12222444	0

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

Select StudentName,StudentSurname,StudentRollNo,StudentAddress,StudentGender,StudentAge,StudentGuardianName

from StudentBasicInformation natural join SubjectScholarshipInformation

where ScholarshipAmount>5000;

```
Q-Select the student details records who has received the scholarship more than 5000Rs/-

Select StudentName,StudentSurname,StudentRollNo,StudentAddress,StudentGender,StudentAge,StudentGuardianName
from StudentBasicInformation natural join SubjectScholarshipInformation
where ScholarshipAmount>5000;
```

Data Output | Explain | Messages | Notifications

	studentname character varying (30)	studentsurname character varying (30)	studentrollno [PK] integer	studentaddress character varying (50)	studentgender character (1)	studentage integer	studentguardianname character varying (50)
1	Sonali	Goyal	6	Naveen Colony,Shahdara	F	23	Pawan Goyal
2	Sonali	Raghuvansi	7	Block C2,Keshavpuram	F	21	Sunita Raghuvanshi
3	Shubham	Yadav	15	Sector 41,Gurgaon	M	23	Janendra Yadav
4	Abhishek	Gupta	1	Sector 20,Rohini	M	20	Alok Gupta

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

Select StudentName,StudentSurname

from StudentBasicInformation A join SubjectScholarshipInformation B

on A.StudentRollNo=B.StudentRollNo

where ScholarshipStatus='pending';

Student/postgres@PostgreSQL 13

Query Editor Query History

```
Q-Select the students who opted for scholarship but has not got the scholarship.

Select StudentName,StudentSurname
from StudentBasicInformation A join SubjectScholarshipInformation B
on A.StudentRollNo=B.StudentRollNo
where ScholarshipStatus='pending';
```

Data Output Explain Messages Notifications

	studentname character varying (30)	studentsurname character varying (30)
1	Abhishek	Gupta
2	Sonali	Raghuvansi
3	Reeta	Sachdeva

QUESTION 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 setPercentage()
language plpgsql
as $$
begin
   update StudentSubjectInformation
   set StudentMarksPercentage= (SubjectObtainedMarks/SubjectTotalMarks)*100;
end;$$

call setPercentage();
```

Student/postgres@PostgreSQL 13

Query Editor Query History

```
create or replace procedure setPercentage()
language plpgsql
as $$
begin
    update StudentSubjectInformation
    set StudentMarksPercentage= (SubjectObtainedMarks/SubjectTotalMarks)*100;
end;$$
call setPercentage();
```

Data Output Explain Messages Notifications

	subjectopted character varying (20)	studentrollno [PK] integer	subjecttotalmarks integer	subjectobtainedmarks double precision	studentmarkspercentage double precision	scholarshipopted character (1)
1	Physics	1	100	90	90	Y
2	Mathematics	2	100	80	80	N
3	Chemistry	3	100	50	50	N
4	Physics	4	100	85.5	85.5	Y
5	Computer Science	5	100	60	60	N
6	Mathematics	6	100	92	92	Y
7	English	7	100	94	94	Y
8	English	8	100	88	88	Y
9	Computer Science	9	100	70	70	N
10	Computer Science	10	100	75	75	N

Activate Windows
Go to Settings to activate

QUESTION 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

```
create or replace procedure

setScholarshipCategory( rollno int )

 language plpgsql as $$

declare percentage float;

 begin select StudentMarksPercentage into percentage from StudentSubjectInformation

 where StudentRollNo = rollno;

 if(percentage > 80 and percentage <= 90) then update SubjectScholarshipInformation

 set ScholarshipCategory = 'Low' where StudentRollNo = rollno; end if;

if (percentage > 90 and percentage <= 95) then update SubjectScholarshipInformation

set ScholarshipCategory = 'Medium'  where StudentRollNo = rollno; end if;
```

if (percentage > 95) then update SubjectScholarshipInformation

set ScholarshipCategory = 'High' where StudentRollNo = rollno; end if;

commit;

end;$$

call setScholarshipCategory(8);

Student/postgres@PostgreSQL 13

Query Editor Query History

```
 begin select StudentMarksPercentage into percentage from StudentSubjectInformation
 where StudentRollNo = rollno;
 if(percentage > 80 and percentage <= 90) then update SubjectScholarshipInformation
 set ScholarshipCategory = 'Low' where StudentRollNo = rollno; end if;
if (percentage > 90 and percentage <= 95) then update SubjectScholarshipInformation
set ScholarshipCategory = 'Medium'  where StudentRollNo = rollno; end if;
if (percentage > 95) then update SubjectScholarshipInformation
set ScholarshipCategory = 'High' where StudentRollNo = rollno; end if;
```

Data Output Explain Messages Notifications

	studentrollno [PK] integer	scholarshipname character varying (30)	scholarshipdescription character varying (20)	scholarshipamount double precision	scholarshipcategory character varying (20)	scholarshipstatus character varying (20)	amountprocessed character (1)
1	1	PhysicsSchlorship	Excellence	6000	Medium	pending	N
2	6	MathematicsSchlorship	Excellence	6000	Medium	Approved	Y
3	7	EnglishSchlorship	Excellence	6000	Medium	pending	N
4	4	PhysicsSchlorship	Very Good	3000	Low	Approved	Y
5	12	ChemistrySchlorship	Very Good	3000	Low	pending	N
6	8	EnglishSchlorship	Very Good	3000	Low	Approved	Y
7	15	MathematicsSchlorship	Outstanding	8000	High	Approved	Y

Activate Windows

QUESTION 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 StudentBalanceView as

select StudentName,StudentSurname,A.StudentRollNo,StudentAddress,StudentAge,StudentGender,AmountBalance

from StudentAdmissionPaymentDetails A join StudentBasicInformation B

on A.StudentRollNo=B.StudentRollNo;

select * from StudentBalanceView;

Student/postgres@PostgreSQL 13

Query Editor Query History

```
create or replace view StudentBalanceView as
select StudentName,StudentSurname,A.StudentRollNo,StudentAddress,StudentAge,StudentGender,AmountBalance
from StudentAdmissionPaymentDetails A join StudentBasicInformation B
on A.StudentRollNo=B.StudentRollNo;

select * from StudentBalanceView;
```

Data Output

	studentname character varying (30)	studentsurname character varying (30)	studentrollno integer	studentaddress character varying (50)	studentage integer	studentgender character (1)	amountbalance double precision
1	Tushar	Maheshwari	2	Jagdish Nagar,Noida	21	M	5000
2	Divya	Gupta	5	Sector 58,Gurgaon	23	F	2000
3	Sonali	Goyal	6	Naveen Colony,Shahdara	23	F	2000
4	Sonali	Raghuvansi	7	Block C2,Keshavpuram	21	F	1000
5	Rishabh	Kanojia	8	Block A,Govind Puri	20	M	5000
6	Mayank	Singh	9	Sushant Lok,Gurgaon	22	M	0
7	Ashish	Sharma	10	Sector 10,Noida	24	M	3000
8	Shrishti	Tyagi	11	Sushant Lok,Gurgaon	21	F	3000
9	Reeta	Sachdeva	12	Colony No 3,Lajpat Nagar	22	F	1000
10	Yasmeen	Grover	13	Bharat Lok,Noida	20	F	14000
11	Satakshi	Aggarwal	14	chandra Lok,Gurgaon	22	f	7000
12	Shubham	Yadav	15	Sector 41,Gurgaon	23	M	0
13	Abhishek	Gupta	1	Sector 20,Rohini	20	M	0
14	Harshit	Saini	4	Arya Nagar,Rohini	22	M	6000
15	Architta	Ahuja	3	Indravihar,Mukherjee Nagar	20	F	0

Activate Windows

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

Query Editor Query History

```
Select StudentRollNo,StudentName,StudentSurname,StudentAddress,StudentAge,StudentGender
from StudentBasicInformation
where StudentRollNo Not in
(Select StudentRollNo from SubjectScholarshipInformation);
```

Data Output Explain Messages Notifications

	studentrollno [PK] integer	studentname character varying (30)	studentsurname character varying (30)	studentaddress character varying (50)	studentage integer	studentgender character (1)
1	2	Tushar	Maheshwari	Jagdish Nagar,Noida	21	M
2	3	Architta	Ahuja	Indravihar,Mukherjee Nagar	20	F
3	5	Divya	Gupta	Sector 58,Gurgaon	23	F
4	9	Mayank	Singh	Sushant Lok,Gurgaon	22	M
5	11	Shrishti	Tyagi	Sushant Lok,Gurgaon	21	F
6	13	Yasmeen	Grover	Bharat Lok,Noida	20	F
7	14	Satakshi	Aggarwal	chandra Lok,Gurgaon	22	f
8	10	Ashish	Sharma	Sector 10,Noida	24	M

Activate Windows
Go to Settings to a

QUESTION 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

```
    create or replace function getAmountBalance( rollno int )
returns float
language plpgsql
as $$
declare
  amount integer;
begin
    select AmountBalance into amount from StudentAdmissionPaymentDetails
    where StudentRollNo=rollno;
    return amount;
end;$$


select getAmountBalance( 2 );
```

Student/postgres@PostgreSQL 13

Query Editor Query History

```
create or replace function getAmountBalance( rollno int )
returns float
language plpgsql
as $$
declare
   amount integer;
begin
    select AmountBalance into amount from StudentAdmissionPaymentDetails
    where StudentRollNo=rollno;
    return amount;
end;$$

select getAmountBalance( 2 );
```

Data Output Explain Messages Notifications

	getamountbalance double precision
1	5000

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

Select A.StudentRollNo,StudentName,StudentSurname,StudentAge,StudentGender,StudentMarksPercentage

from StudentBasicInformation A join StudentSubjectInformation B

on A.StudentRollNo=B.StudentRollNo

order by StudentMarksPercentage desc

limit 5;

Query Editor Query History

```
Select A.StudentRollNo,StudentName,StudentSurname,StudentAge,StudentGender,StudentMarksPercentage
from StudentBasicInformation A join StudentSubjectInformation B
on A.StudentRollNo=B.StudentRollNo
order by StudentMarksPercentage desc
limit 5;
```

Data Output Explain Messages Notifications

	studentrollno integer	studentname character varying (30)	studentsurname character varying (30)	studentage integer	studentgender character (1)	studentmarkspercentage double precision
1	15	Shubham	Yadav	23	M	98
2	7	Sonali	Raghuvansi	21	F	94
3	6	Sonali	Goyal	23	F	92
4	1	Abhishek	Gupta	20	M	90
5	8	Rishabh	Kanojia	20	M	88

QUESTION 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)

Select StudentName,StudentSurname,AmountBalance,SubjectOpted,StudentMarksPercentage,ScholarshipCategory

from StudentBasicInformation A inner join StudentAdmissionPaymentDetails B

on A.StudentRollNo=B.StudentRollNo

join StudentSubjectInformation C

on A.StudentRollNo=C.StudentRollNo

left join SubjectScholarshipInformation D

on A.StudentRollNo=D.StudentRollNo;

We used inner join as we wanted to have the Admission payment details information for all the students for whom we had their basic information. Since not all the students opted for scholarship, we used ‘left join’.

Student/postgres@PostgreSQL 13

Query Editor | Query History | Scratch Pad

```
Select StudentName,StudentSurname,AmountBalance,SubjectOpted,StudentMarksPercentage,ScholarshipCategory
from StudentBasicInformation A inner join StudentAdmissionPaymentDetails B
on A.StudentRollNo=B.StudentRollNo
join StudentSubjectInformation C
on A.StudentRollNo=C.StudentRollNo
left join SubjectScholarshipInformation D
on A.StudentRollNo=D.StudentRollNo;
```

Data Output | Explain | Messages | Notifications

	studentname character varying (30)	studentsurname character varying (30)	amountbalance double precision	subjectopted character varying (20)	studentmarkspercentage double precision	scholarshipcategory character varying (20)
1	Abhishek	Gupta	0	Physics	90	Medium
2	Tushar	Maheshwari	5000	Mathematics	80	[null]
3	Architta	Ahuja	0	Chemistry	50	[null]
4	Harshit	Saini	6000	Physics	85.5	Low
5	Divya	Gupta	2000	Computer Science	60	[null]
6	Sonali	Goyal	2000	Mathematics	92	Medium
7	Sonali	Raghuvansi	1000	English	94	Medium
8	Rishabh	Kanojia	5000	English	88	Low
9	Mayank	Singh	0	Computer Science	70	[null]
10	Ashish	Sharma	3000	Computer Science	75	[null]

Activate Windows
Go to Settings to activate Windows.

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

TRUNCATE	DELETE	DROP
▪ TRUNCATE Command is a Data Definition Language(DDL) Command. ▪ It Remove all records from a table, including all spaces allocated for the records are removed. ▪ It deletes all the records from an existing table **but not the table itself.** The structure or schema of the table is preserved. ▪ **SYNTAX:** TRUNCATE TABLE table_name;	▪ The DELETE statement in SQL is a Data Manipulation Language(DML) Command *so it can be rolled back* ▪ Deletes all records from a table, the space for the records remain ▪ We can delete a single record or multiple records depending on the condition specified in the query. ▪ The **DELETE** statement scans every row before deleting it. Thus it is slower as compared to **TRUNCATE** command. If we want to delete all the records of a table, it is preferable to use **TRUNCATE** in place of **DELETE** as the former is faster than the latter. ▪ The DELETE command returns the number of records that were deleted by its execution. ▪ SYNTAX: DELETE FROM table_name [WHERE conditions];	▪ DROP statement is a Data Definition Language(DDL) Command. ▪ It is used to delete existing database objects. It can be used to delete databases, tables, views, triggers, etc ▪ *Objects deleted using DROP are permanently lost and it cannot be rolled back .* ▪ Unlike **TRUNCATE** which only deletes the data of the tables, the DROP command deletes the data of the table as well as removes the entire schema/structure of the table from the database. ▪ **SYNTAX:** DROP objectobject_name object: Keyword representing the type of the database object. object_name: It specifies the name of the object we want to delete.

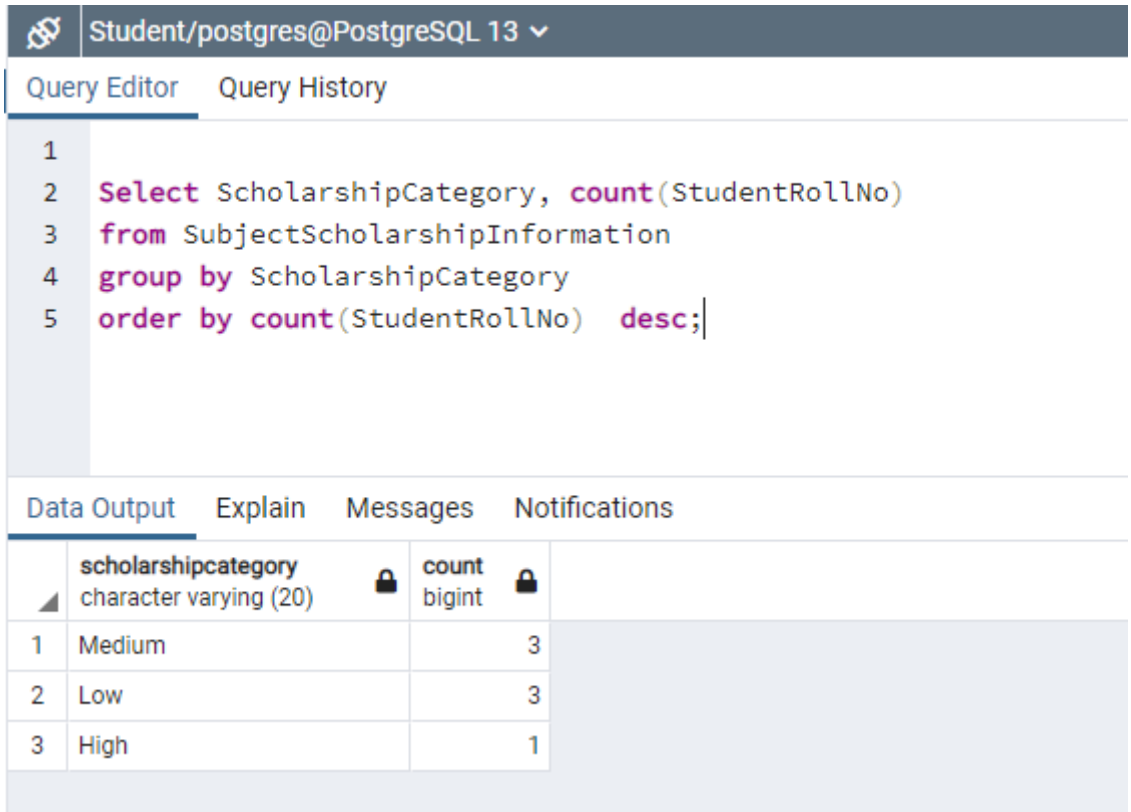
QUESTION 17-Get the count of the Scholarship category which is highly been availed by the students, i.e. get the count of the total number of students corresponding to the each scholarships category

Select ScholarshipCategory, count(StudentRollNo)

from SubjectScholarshipInformation

group by ScholarshipCategory

order by count(StudentRollNo) desc;



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

Select ScholarshipCategory, count(StudentRollNo)

from SubjectScholarshipInformation

group by ScholarshipCategory

having count(StudentRollNo)>=(select count(StudentRollNo)

from SubjectScholarshipInformation group by ScholarshipCategory order by count(StudentRollNo) desc limit 1);

Student/postgres@PostgreSQL 13

Query Editor Query History

```
Select ScholarshipCategory, count(StudentRollNo)
from SubjectScholarshipInformation
group by ScholarshipCategory
having count(StudentRollNo)>=(select count(StudentRollNo)
from SubjectScholarshipInformation group by ScholarshipCategory order by count(StudentRollNo) desc limit 1 )
```

Data Output Explain Messages Notifications

	scholarshipcategory character varying (20)	count bigint
1	Medium	3
2	Low	3

QUESTION 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 StudentName,StudentSurname,A.StudentRollNo,
StudentAddress,StudentAge,StudentMarksPercentage,ScholarshipAmount
from StudentBasicInformation A join StudentSubjectInformation B
on A.StudentRollNo=B.StudentRollNo
left join SubjectScholarshipInformation C
on A.StudentRollNo=C.StudentRollNo
order by StudentMarksPercentage desc,ScholarshipAmount desc
limit 1;
```

Student/postgres@PostgreSQL 13

Query Editor Query History

```
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 StudentName,StudentSurname,A.StudentRollNo,
StudentAddress,StudentAge,StudentMarksPercentage,ScholarshipAmount
from StudentBasicInformation A join StudentSubjectInformation B
on A.StudentRollNo=B.StudentRollNo
left join SubjectScholarshipInformation C
on A.StudentRollNo=C.StudentRollNo
order by StudentMarksPercentage desc,ScholarshipAmount desc
limit 1;
```

Data Output Explain Messages Notifications

	studentname character varying (30)	studentsurname character varying (30)	studentrollno integer	studentaddress character varying (50)	studentage integer	studentmarkspercentage double precision	scholarshipamount double precision
1	Shubham	Yadav	15	Sector 41,Gurgaon	23	98	8000

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

STORED PRODECURES:

- Stored procedures are a pieces of the code written to do some specific task.
- It can be invoked explicitly by the user.
- Stored Procedures are pre-compiled objects which are compiled for the first time and its compiled format is saved.
- It can take input as a parameter.
- We can use transaction statements like begin transaction, commit transaction, and rollback inside a stored procedure.
- Stored procedures can return values.
- An exception can be handled by try-catch block in a Procedure.

TRIGERS :

- Trigger is a stored procedure that runs automatically when various events happen (eg update, insert, delete).
- It can execute automatically based on the events.
- It can not take input as parameter.
- we can't use transaction statements inside a trigger.
- Triggers can not return values .

FUNCTIONS:

- A function is compiled and executed every time whenever it is called. A function must return a value and cannot modify the data received as parameters.
- The function must return a value .

- Functions can have only input parameters for it whereas Procedures can have input or output parameters.
- Functions can be called from Procedure whereas Procedures cannot be called from a Function.
- We can't use Transactions in Function.
- Try-Catch block cannot be used in a Function.

VIEWS :

- A view is a virtual table based on the result-set of an SQL statement.
- A view does NOT accept parameters.
- A View can be used as building block in a larger query
- A View can contain only one single SELECT query
- A View can NOT perform modifications to any table but can (sometimes) be used as the target of an INSERT, UPDATE or DELETE statement.

.