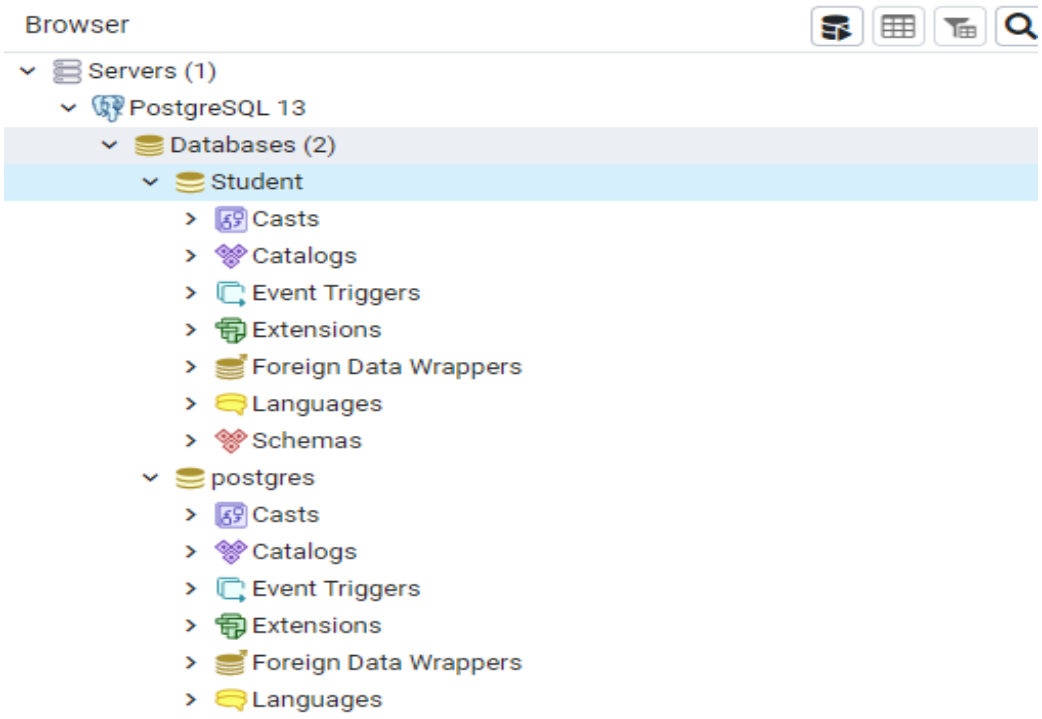
Submitted by - Rohan Kumar Jha
Email - rohan.jha@accolitedigital.com

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**

Query Editor | Query History

```
1  create table StudentBasicInformation(
2      StudentName varchar(20),
3      StudentSurname varchar(20),
4      StudentRollNo int primary key,
5      StudentAddress varchar(30),
6      Email varchar(20),
7      DOB date,
8      Branch varchar(20)
9  );
```

Insert queries -

```
insert into StudentBasicInformation values ('A','a',1,'A/101','a@gmail.com','01-01-2000','CS');
insert into StudentBasicInformation values ('J','j',10,'J/401','j@gmail.com','1999-02-13','ME');
insert into StudentBasicInformation values ('B','b',2,'B/102','b@gmail.com','2000-02-03','CS');
insert into StudentBasicInformation values ('C','c',3,'C/103','c@gmail.com','2001-04-01','ME');
insert into StudentBasicInformation values ('D','d',4,'D/201','d@gmail.com','1997-11-04','IT');
insert into StudentBasicInformation values ('E','e',5,'E/202','e@gmail.com','2000-04-05','IT');
insert into StudentBasicInformation values ('F','f',6,'F/203','f@gmail.com','2002-12-03','CS');
insert into StudentBasicInformation values ('G','g',7,'G/301','g@gmail.com','1998-02-27','CS');
insert into StudentBasicInformation values ('H','h',8,'H/302','h@gmail.com','2000-11-01','EC');
insert into StudentBasicInformation values ('I','i',9,'I/303','i@gmail.com','2001-07-09','EE');
```

Table snap -

	studentname character varying (20)	studentsurname character varying (20)	studentrollno [PK] integer	studentaddress character varying (30)	email character varying (20)	dob date	branch character varying (20)
1	A	a	1	A/101	a@gmail.com	2000-01...	CS
2	J	j	10	J/401	j@gmail.com	1999-02...	ME
3	B	b	2	B/102	b@gmail.com	2000-02...	CS
4	C	c	3	C/103	c@gmail.com	2001-04...	ME
5	D	d	4	D/201	d@gmail.com	1997-11...	IT
6	E	e	5	E/202	e@gmail.com	2000-04...	IT
7	F	f	6	F/203	f@gmail.com	2002-12...	CS
8	G	g	7	G/301	g@gmail.com	1998-02...	CS
9	H	h	8	H/302	h@gmail.com	2000-11...	EC
10	I	i	9	I/303	i@gmail.com	2001-07...	EE

b. StudentAdmissionPaymentDetails

i. Columns

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

Create table -

```
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)
);
```

Insert queries -

```
insert into StudentAdmissionPaymentDetails values (1,80000,20000,'NEFT','id101','SBI','2244');
insert into StudentAdmissionPaymentDetails values (2,70000,30000,'NEFT','id121','HDFC','4422');
insert into StudentAdmissionPaymentDetails values (3,90000,10000,'RTGS','id999','DBS','3131');
insert into StudentAdmissionPaymentDetails values (4,40000,60000,'UPI','id811','Kotak','7124');
insert into StudentAdmissionPaymentDetails values (5,85000,15000,'DD','id891','PNB','5566');
insert into StudentAdmissionPaymentDetails values (6,64000,36000,'Cheque','id141','ICICI','8080');
insert into StudentAdmissionPaymentDetails values (7,95000,5000,'NEFT','id713','Axis','9199');
insert into StudentAdmissionPaymentDetails values (8,10000,90000,'UPI','id603','PNB','3300');
insert into StudentAdmissionPaymentDetails values (9,65000,35000,'DD','id131','SBI','6859');
insert into StudentAdmissionPaymentDetails values (10,99999,1,'NEFT','id151','SBI','7264');
```

Table Snap -

studentrollno [PK] integer	amountpaid integer	amountbalance integer	modeofpayment character varying (20)	transactionid character varying (20)	bankname character varying (20)	bankaccountno character varying (20)
1	80000	20000	NEFT	id101	SBI	2244
2	70000	30000	NEFT	id121	HDFC	4422
3	90000	10000	RTGS	id999	DBS	3131
4	40000	60000	UPI	id811	Kotak	7124
5	85000	15000	DD	id891	PNB	5566
6	64000	36000	Cheque	id141	ICICI	8080
7	95000	5000	NEFT	id713	Axis	9199
8	10000	90000	UPI	id603	PNB	3300
9	65000	35000	DD	id131	SBI	6859
10	99999	1	NEFT	id151	SBI	7264

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 -

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

Insert queries -

```
insert into StudentSubjectInformation values (1,'DS',80,67,83.75,'A2');
insert into StudentSubjectInformation values (2,'DS',80,55,68.75,'B2');
insert into StudentSubjectInformation values (3,'DS',80,74,92.50,'A1');
insert into StudentSubjectInformation values (1,'Algo',70,65,92.86,'A1');
insert into StudentSubjectInformation values (2,'Algo',70,51,72.85,'B1');
insert into StudentSubjectInformation values (3,'Algo',70,67,95.71,'A1');
insert into StudentSubjectInformation values (1,'DBMS',100,87,87.00,'A2');
insert into StudentSubjectInformation values (2,'DBMS',100,96,96.00,'A1');
insert into StudentSubjectInformation values (3,'DBMS',100,83,83.00,'A2');
insert into StudentSubjectInformation values (4,'DS',80,77,96.75,'A1');
```

Table Snap -

studentrollno integer	subjectopted character varying (20)	subjecttotalmarks integer	subjectobtainedmarks integer	studentmarkspercentage numeric (4,2)	grade character varying (2)
1	DS	80	67	83.75	A2
2	DS	80	55	68.75	B2
3	DS	80	74	92.50	A1
1	Algo	70	65	92.86	A1
2	Algo	70	51	72.85	B1
3	Algo	70	67	95.71	A1
1	DBMS	100	87	87.00	A2
2	DBMS	100	96	96.00	A1
3	DBMS	100	83	83.00	A2
4	DS	80	77	96.75	A1

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 -

```
create table SubjectScholarshipInformation(
    StudentRollNo int primary key
            references StudentBasicInformation(StudentRollNo),
    ScholarshipName varchar(20),
    ScholarshipDescription varchar(30),
    ScholarshipAmount int,
    ScholarshipCategory varchar(10)
);
```

Insert Queries -

```
insert into SubjectScholarshipInformation values (1,'AKTU','fee concession',50000,'EWS');
insert into SubjectScholarshipInformation values (2,'CS Olympiad','Reward',30000,'Merit');
insert into SubjectScholarshipInformation values (3,'SIH','project reward',20000,'Project');
insert into SubjectScholarshipInformation values (4,'AKTU','fee concession',50000,'EWS');
insert into SubjectScholarshipInformation values (5,'AKTU','fee concession',50000,'EWS');
insert into SubjectScholarshipInformation values (6,'SIH','project reward',20000,'Project');
insert into SubjectScholarshipInformation values (7,'AKTU','Gold Medalist',100000,'Rank');
```

Table Snap -

studentrollno [PK] integer	scholarshipname character varying (20)	scholarshipdescription character varying (30)	scholarshipamount integer	scholarshipcategory character varying (10)
1	AKTU	fee concession	50000	EWS
2	CS Olympiad	Reward	30000	Merit
3	SIH	project reward	20000	Project
4	AKTU	fee concession	50000	EWS
5	AKTU	fee concession	50000	EWS
6	SIH	project reward	20000	Project
7	AKTU	Gold Medalist	100000	Rank

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**

```
update StudentBasicInformation set StudentAddress='BH/111' where StudentRollNo >=7

update StudentBasicInformation set StudentName='AB' where StudentRollNo=1;

update StudentAdmissionPaymentDetails set AmountBalance=AmountBalance+5000
        where AmountPaid<60000;

update StudentAdmissionPaymentDetails set AmountBalance=0
        where AmountBalance<100;

update StudentSubjectInformation set Grade='A1'
        where StudentMarksPercentage>85.00;
```

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

StudentBasicInformation -

studentname character varying (20)	studentsurname character varying (20)	studentrollno [PK] integer	studentaddress character varying (30)	email character varying (20)	dob date	branch character
B	b	2	B/102	b@gmail.com	2000-02...	CS
C	c	3	C/103	c@gmail.com	2001-04...	ME
D	d	4	D/201	d@gmail.com	1997-11...	IT
E	e	5	E/202	e@gmail.com	2000-04...	IT
F	f	6	F/203	f@gmail.com	2002-12...	CS
J	j	10	BH/111	j@gmail.com	1999-02...	ME
G	g	7	BH/111	g@gmail.com	1998-02...	CS
H	h	8	BH/111	h@gmail.com	2000-11...	EC
I	i	9	BH/111	i@gmail.com	2001-07...	EE
AB	a	1	A/101	a@gmail.com	2000-01...	CS

StudentAdmissionPaymentDetails -

studentrollno [PK] integer	amountpaid integer	amountbalance integer	modeofpayment character varying (20)	transactionid character varying (20)	bankname character varying (20)	bankaccountno character varyin
1	80000	20000	NEFT	id101	SBI	2244
2	70000	30000	NEFT	id121	HDFC	4422
3	90000	10000	RTGS	id999	DBS	3131
5	85000	15000	DD	id891	PNB	5566
6	64000	36000	Cheque	id141	ICICI	8080
7	95000	5000	NEFT	id713	Axis	9199
9	65000	35000	DD	id131	SBI	6859
4	40000	65000	UPI	id811	Kotak	7124
8	10000	95000	UPI	id603	PNB	3300
10	99999	0	NEFT	id151	SBI	7264

StudentSubjectInformation -

studentrollno integer	subjectopted character varying (20)	subjecttotalmarks integer	subjectobtainedmarks integer	studentmarkspercentage numeric (4,2)	grade charact
1	DS	80	67	83.75	A2
2	DS	80	55	68.75	B2
2	Algo	70	51	72.85	B1
3	DBMS	100	83	83.00	A2
3	DS	80	74	92.50	A1
1	Algo	70	65	92.86	A1
3	Algo	70	67	95.71	A1
1	DBMS	100	87	87.00	A1
2	DBMS	100	96	96.00	A1
4	DS	80	77	96.75	A1

7. **Select the student details records who has received the scholarship more than 50000Rs/-**

Query -

```
SELECT * FROM StudentBasicInformation SB
JOIN SubjectScholarshipInformation SS
ON SB.StudentRollNo = SS.StudentRollNo
WHERE SS.ScholarshipAmount>=50000;
```

Result -

studentname character varying (20)	studentsurname character varying (20)	studentrollno integer	studentaddress character varying (30)	email character varying (20)	dob date	branch character
D	d	4	D/201	d@gmail.com	1997-11...	IT
E	e	5	E/202	e@gmail.com	2000-04...	IT
G	g	7	BH/111	g@gmail.com	1998-02...	CS
AB	a	1	A/101	a@gmail.com	2000-01...	CS

studentrollno integer	scholarshipname character varying (20)	scholarshipdescription character varying (30)	scholarshipamount integer	scholarshipcategory character varying (10)
4	AKTU	fee concession	50000	EWS
5	AKTU	fee concession	50000	EWS
7	AKTU	Gold Medalist	100000	Rank
1	AKTU	fee concession	50000	EWS

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 procedure public.setPercentage()
language 'sql'
as $BODY$
update StudentSubjectInformation
set StudentMarksPercentage = (SubjectObtainedMarks/SubjectTotalMarks)*100;
$BODY$;
```

StudentSubjectInformation Table after running Procedure -

studentrollno integer	subjectopted character varying (20)	subjecttotalmarks integer	subjectobtainedmarks integer	studentmarkspercentage numeric (4,2)
1	DS	80	67	83.75
2	DS	80	55	68.75
2	Algo	70	51	72.85
3	DBMS	100	83	83.00
3	DS	80	74	92.50
1	Algo	70	65	92.86
3	Algo	70	67	95.71
1	DBMS	100	87	87.00
2	DBMS	100	96	96.00
4	DS	80	77	96.75

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**

```
call public.setPercentage();
select * from StudentSubjectInformation;

create procedure public.setScholarshipCategory()
language 'sql'
as $BODY$
update SubjectScholarshipInformation
set ScholarshipCategory = 'A-Grade' where StudentRollNo in (
    select StudentRollNo from StudentSubjectInformation
        where StudentMarksPercentage > 90);
$BODY$;

call public.setScholarshipCategory();
```

SubjectScholarshipInformation Table after calling Procedure -

studentrollno [PK] integer	scholarshipname character varying (20)	scholarshipdescription character varying (30)	scholarshipamount integer	scholarshipcategory character varying (10)
5	AKTU	fee concession	50000	EWS
6	SIH	project reward	20000	Project
7	AKTU	Gold Medalist	100000	Rank
1	AKTU	fee concession	50000	A-Grade
2	CS Olympiad	Reward	30000	A-Grade
3	SIH	project reward	20000	A-Grade
4	AKTU	fee concession	50000	A-Grade

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

```
create view balanceAmt as
    select SB.StudentRollNO,SB.StudentName,SP.AmountBalance
        from StudentBasicInformation SB join StudentAdmissionPaymentDetails SP
            on SB.StudentRollNO = SP.StudentRollNo;

select * from balanceAmt;
```

View -

studentrollno integer	studentname character varying (20)	amountbalance integer
1	AB	20000
2	B	30000
3	C	10000
5	E	15000
6	F	36000
7	G	5000
9	I	35000
4	D	65000
8	H	95000
10	J	0

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

14. **Retrieve the top 3 student details as per the StudentMarksPercentage values (use subqueries)**

Query -

```
select * from StudentBasicInformation
    where StudentRollNo in (
        select StudentRollNo from StudentSubjectInformation
            where StudentMarksPercentage in(
                select StudentMarksPercentage from StudentSubjectInformation
            order by StudentMarksPercentage desc limit 3)
    );
```

Result -

studentname character varying (20)	studentsurname character varying (20)	studentrollno [PK] integer	studentaddress character varying (30)	email character varying (20)	dob date	branch characte
D	d	4	D/201	d@gmail.com	1997-11...	IT
B	b	2	B/102	b@gmail.com	2000-02...	CS
C	c	3	C/103	c@gmail.com	2001-04...	ME

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)

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

DELETE:

Basically, it is a Data Manipulation Language Command (DML). It is use to delete the one or more tuples of a table. With the help of “DELETE” command we can either delete all the rows in one go or can delete row one by one. i.e., we can use it as per the requirement or the condition using Where clause.

DROP:

It is a Data Definition Language Command (DDL). It is use to drop the whole table. With the help of “DROP” command we can drop (delete) the whole structure in one go i.e., it removes the

named elements of the schema. By using this command, the existence of the whole table is finished.

TRUNCATE:

It is also a Data Definition Language Command (DDL). It is use to delete all the rows of a relation (table) in one go. With the help of “TRUNCATE” command we can’t delete the single row as here WHERE clause is not used. By using this command, the existence of all the rows of the table is lost. It is comparatively faster than delete command as it deletes all the rows lastly.

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(*) from SubjectScholarshipInformation
    group by ScholarshipCategory;
```

Result -

scholarshipcategory character varying (10)	count bigint
Rank	1
Project	1
A-Grade	4
EWS	1

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

Query -

```
select ScholarshipCategory, count(*) from SubjectScholarshipInformation
    group by ScholarshipCategory order by count(*) desc limit 1;
```

Result -

scholarshipcategory character varying (10)	count bigint
A-Grade	4

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

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

Sr. No.	Key	Triggers	Stored procedures
1	Basic	trigger is a stored procedure that runs automatically when various events happen (eg update, insert, delete)	Stored procedures are a pieces of the code in written in PL/SQL to do some specific task
2	Running Methodology	It can execute automatically based on the events	It can be invoked explicitly by the user
3	Parameter	It can not take input as parameter	It can take input as a parameter
4	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

5	Return	Triggers can not return values	Stored procedures can return values

VIEW

A view is a “virtual” table consisting of a **SELECT** statement, by means of “virtual”
I mean no physical data has been stored by the view -- only the definition of the view is stored inside the database; unless you materialize the view by putting an index on it.

1) By definition you can not pass parameters to the view
2) NO DML operations (e.g. INSERT, UPDATE, and DELETE) are allowed inside the view; ONLY SELECT statements.

Functions -

Functions are subroutines made up of one or more Transact-SQL statements that can be used to encapsulate code for reuse