

SQL Concepts and Fundamentals Assignment

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
 - b. StudentAdmissionPaymentDetails
 - i. Columns
 1. StudentRollNo
 2. AmountPaid
 3. AmountBalance
 4. Add more four basic columns of the name of your own
 - 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
 - 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
3. Insert more than 10 records in each and every table created
4. Snap of the all the tables once the insertion is completed

Query Editor	Query History	Scratch
<pre> 96 97 select * from studentbasicinformation; 98 99 100 </pre>		

Data Output		Explain	Messages	Notifications	
	studentname character varying (10)	studentsurname character varying (10)	studentrollno [PK] integer	studentaddress character varying (30)	studentfathername character varying (10)
1	Raju	Sharma	1	Kanpur	Ramesh
2	Raj	Goel	2	Ranchi	Rajesh
3	Ravi	Jain	3	Delhi	Jay
4	Shiv	Gupta	4	Delhi	Shyam
5	Ajay	Jain	5	Delhi	Ram
6	Ram	Sharma	6	Lucknow	Ramesh
7	Shyam	Mittal	7	Kanpur	Ganesh
8	Kishan	Goel	8	Noida	Keshav
9	Rajesh	Singal	9	Noida	Shiv
10	Prince	Bansal	10	Delhi	Raj

96	
97	select * from studentadmissionpaymentdetails;
98	
99	
100	

Data Output	Explain	Messages	Notifications
<div> <div></div> <div>amountpaid double precision</div> </div> <div> <div></div> <div>amountbalance double precision</div> </div> <div> <div></div> <div>studentrollno [PK] integer</div> </div>			
1	5000	1000	1
2	4000	2000	2
3	6000	0	3
4	5500	500	4
5	6000	0	5
6	6000	0	6
7	6000	0	7
8	5000	1000	8
9	3000	3000	9
10	0	6000	10

Query Editor Query History

```

137
138 Select * from studentsubjectinformation;
139
140
141
142

```

Data Output Explain Messages Notifications

	subjectopted character varying (10)	studentrollno [PK] integer	obtainedmarks integer	totalmarks integer	studentmarkspercentage double precision
1	Science	2	486	500	[null]
2	Law	3	422	500	[null]
3	Mechanics	4	350	500	[null]
4	Computer	5	455	500	[null]
5	English	6	399	500	[null]
6	Science	7	490	500	[null]
7	Law	8	401	500	[null]
8	Sanskrit	9	322	500	[null]
9	Computer	10	300	500	[null]
10	Computer	1	476	500	[null]

Query Editor Query History

Scratch P

```

96
97
98
99 select * from subjectscholarshipinformation;
100
101
102

```

Data Output Explain Messages Notifications

	scholarshipname character varying (10)	studentrollno [PK] integer	scholarshipamount integer	scholarshipcategory character varying (20)	isrequested boolean	isgranted boolean
1	Comp Sch	1	4000	[null]	true	true
2	Law Sch	3	4000	[null]	true	false
3	Sci Sch	2	1000	[null]	true	false
4	Sci Sch	7	4000	[null]	true	true
5	Comp Sch	5	1000	[null]	true	true
6	Comp Sch	6	2000	[null]	true	true
7	Eng Sch	10	1000	[null]	true	false

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
6. Snap of the all the tables post updating.

Query Editor

Query History

Scratch F

```

99 UPDATE studentbasicinformation SET studentfathername = 'Rakesh' WHERE studentrollno = 1 ;
100 UPDATE studentbasicinformation SET studentaddress = 'Gurugram' WHERE studentrollno = 5 ;
101 UPDATE studentbasicinformation SET studentsurname = 'Singala' WHERE studentrollno = 3 ;
102 UPDATE studentbasicinformation SET studentname = 'Vaibhav' WHERE studentrollno = 8 ;
103 UPDATE studentbasicinformation SET studentfathername = 'Ramesh' WHERE studentrollno = 10
104 Select * from studentbasicinformation ;
105

```

Data Output

Explain

Messages

Notifications

	studentname character varying (10)	studentsurname character varying (10)	studentrollno [PK] integer	studentaddress character varying (30)	studentfathername character varying (10)
1	Raj	Goel		2 Ranchi	Rajesh
2	Shiv	Gupta		4 Delhi	Shyam
3	Ram	Sharma		6 Lucknow	Ramesh
4	Shyam	Mittal		7 Kanpur	Ganesh
5	Rajesh	Singal		9 Noida	Shiv
6	Raju	Sharma		1 Kanpur	Rakesh
7	Ajay	Jain		5 Gurugram	Ram
8	Ravi	Singala		3 Delhi	Rakesh
9	Vaibhav	Goel		8 Noida	Keshav
10	Prince	Bansal		10 Delhi	Ramesh

7. Select the student details records who has received the scholarship more than 3000Rs/-

Query Editor

Query History

Scratch Pad

1

create table StudentBasicInformation (

2

StudentName varchar(10) ,

3

StudentSurname varchar(10) ,

4

StudentRollNo int,

5

StudentAddress varchar(30),

6

StudentFatherName varchar(10),

Data Output

Explain

Messages

Notifications

studentname character varying (10)	studentsurname character varying (10)	studentrollno integer	studentaddress character varying (30)	studentfathername character varying (10)	scholarshipname character varying (10)	studentrollno integer	scholarshipamount integer
Raju	Sharma	1	Kanpur	Rakesh	Comp Sch	1	4000
Ravi	Singala	3	Delhi	Rakesh	Law Sch	3	4000
Shyam	Mittal	7	Kanpur	Ganesh	Sci Sch	7	4000

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

Query Editor

Query History

111

112

113

114

115

116

117

```
select * from subjectscholarshipinformation where Isrequested= true and isgranted=false ;
```

Data Output

Explain

Messages

Notifications

	<div><div><div><div><div><div></div><div>scholarshipname</div><div>character varying (10)</div></div></div><div><div></div><div></div><div></div></div></div></div></div>	<div><div><div><div><div><div></div><div>studentrollno</div><div>[PK] integer</div></div></div><div><div></div><div></div><div></div></div></div></div></div>	<div><div><div><div><div><div></div><div>scholarshipamount</div><div>integer</div></div></div><div><div></div><div></div><div></div></div></div></div></div>	<div><div><div><div><div><div></div><div>scholarshipcategory</div><div>character varying (20)</div></div></div><div><div></div><div></div><div></div></div></div></div></div>	<div><div><div><div><div><div></div><div>isrequested</div><div>boolean</div></div></div><div><div></div><div></div><div></div></div></div></div></div>	<div><div><div><div><div><div></div><div>isgranted</div><div>boolean</div></div></div><div><div></div><div></div><div></div></div></div></div></div>
1	Law Sch	3	4000	[null]	true	false
2	Sci Sch	2	1000	[null]	true	false
3	Eng Sch	10	1000	[null]	true	false

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

Query Editor

Query History

115

create or replace procedure percentage()

116

language plpgsql

117

as \$\$

118

begin

119

update studentsubjectinformation

120

set studentmarkspercentage = (obtainedmarks/totalmarks) * 100 ;

121

122

commit;

123

end;\$\$

124

Data Output

Explain

Messages

Notifications

	subjectopted character varying (10)	studentrollno [PK] integer	obtainedmarks double precision	totalmarks double precision	studentmarkspercentage double precision
1	Computer	1	476	500	95.19999999999999
2	Science	2	486	500	97.2
3	Law	3	422	500	84.39999999999999
4	English	6	399	500	79.80000000000001
5	Mechanics	4	350	500	70
6	Sanskrit	9	322	500	64.4
7	Law	8	401	500	80.2
8	Computer	5	455	500	91

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

student/postgres@PostgreSQL 13

Query Editor Query History

```

130 create or replace procedure category()
131 language plpgsql
132 as $$
133 begin
134     update subjectscholarshipinformation
135     set scholarshipcategory = 'BRONZE'
136     where studentrollno in ( select stu.studentrollno from subjectscholarshipinformation as sub
137                             inner join studentsubjectinformation as stu on
138                             stu.studentrollno = sub.studentrollno where stu.studentmarkspercentage >= 70
139     );
140
141     update subjectscholarshipinformation
142     set scholarshipcategory = 'SILVER'
143     where studentrollno in( select stu.studentrollno from subjectscholarshipinformation as sub
144                             inner join studentsubjectinformation as stu on
145                             stu.studentrollno = sub.studentrollno where stu.studentmarkspercentage >= 80

```

Data Output Explain Messages Notifications

	scholarshipname character varying (10)	studentrollno [PK] integer	scholarshipamount integer	scholarshipcategory character varying (20)	isrequested boolean	isgranted boolean
1	Comp Sch	1	4000	GOLD	true	true
2	Law Sch	3	4000	SILVER	true	false
3	Sci Sch	7	4000	GOLD	true	true

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

Query Editor Query History

```

158 CREATE VIEW balanceamount AS
159 SELECT basic.studentname , basic.studentrollno , payment.amountbalance
160 from studentbasicinformation as basic
161 inner join studentadmissionpaymentdetails as payment on
162 basic.studentrollno = payment.studentrollno ;
163
164 Select * from balanceamount ;
165

```

Data Output Explain Messages Notifications

	studentname character varying (10)	studentrollno integer	amountbalance double precision
1	Raju	1	1000
2	Raj	2	2000
3	Ravi	3	0
4	Shiv	4	500
5	Ajay	5	0
6	Ram	6	0
7	Shyam	7	0
8	Vaibhav	8	1000
9	Rajesh	9	3000

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

Query Editor Query History

```

165
166 select basic.studentname , basic.studentrollno , sch.isgranted
167 from studentbasicinformation as basic left outer join
168 subjectscholarshipinformation as sch on
169 sch.studentrollno = basic.studentrollno |
170 where sch.isgranted is null or sch.isgranted = false;
171
172

```

Data Output Explain Messages Notifications

	studentname character varying (10)	studentrollno integer	isgranted boolean
1	Prince	10	false
2	Ravi	3	false
3	Raj	2	false
4	Vaibhav	8	[null]
5	Shiv	4	[null]
6	Rajesh	9	[null]

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

student/postgres@PostgreSQL 13

Query Editor Query History

```

187
188 CREATE OR REPLACE FUNCTION amounttobepaid (rollno int)
189 RETURNS TABLE (
190     studentroll integer,
191     balanceamount float)
192 AS $$
193 BEGIN
194     RETURN QUERY SELECT
195         studentrollno , amountbalance from
196         studentadmissionpaymentdetails where
197         studentrollno = rollno;
198 END; $$
199 LANGUAGE 'plpgsql';
200 select * from amounttobepaid(1) ;
201

```

Data Output Explain Messages Notifications

	studentroll integer	balanceamount double precision
1	1	1000

14. Retrieve the top five student details as per the StudentMarksPercentage values

student/postgres@PostgreSQL 13

Query Editor Query History

```

201
202
203 select * from studentsubjectinformation order by studentmarkspercentage desc limit 5 ;
204
205
206
207
208
209

```

Data Output Explain Messages Notifications

	subjectopted character varying (10)	studentrollno [PK] integer	obtainedmarks double precision	totalmarks double precision	studentmarkspercentage double precision
1	Science	7	490	500	98
2	Science	2	486	500	97.2
3	Computer	1	476	500	95.19999999999999
4	Computer	5	455	500	91
5	Law	3	422	500	84.39999999999999

15. Try to use all the three types of join learned today in a relevant way, and explain the same why you thought of using that particular join for your selected scenarios (try to cover relevant and real time scenarios for all the three studied joins)

Inner join

We can use inner join in the case when want to join two tables in such a way that only those records appear which are present in both the tables. One such case is when we have to find the student names which are granted with the scholarship, in this the record should be present in both the tables that is why we have to use inner join.

student/postgres@PostgreSQL 13

Query Editor Query History

```

219
220
221
222 select stu.studentname , sch.scholarshipamount from studentbasicinformation as stu
223 inner join subjectscholarshipinformation as sch on sch.studentrollno = stu.studentrollno
224 where sch.isgranted = true ;
225

```

Data Output Explain Messages Notifications

	studentname character varying (10)	scholarshipamount integer
1	Ram	2000
2	Shyam	4000
3	Raju	4000
4	Ajay	1000

Left Outer join

We use left outer join where we need all the relevant records from the left table irrespective to the thing whether same record is present in right table or not. One such case is when we have to find the names of students who haven't receive any scholarship, in this two cases there may be student have requested but didn't granted in this case record is present in the scholarship table but if the student haven't requested then its record is not there in scholarship table but we have print their names also. Therefore we have to use left outer join.

Query Editor Query History

```

165
166 select basic.studentname , basic.studentrollno , sch.isgranted
167 from studentbasicinformation as basic left outer join
168 subjectscholarshipinformation as sch on
169 sch.studentrollno = basic.studentrollno |
170 where sch.isgranted is null or sch.isgranted = false;
171
172

```

Data Output Explain Messages Notifications

	studentname character varying (10)	studentrollno integer	isgranted boolean
1	Prince	10	false
2	Ravi	3	false
3	Raj	2	false
4	Vaibhav	8	[null]
5	Shiv	4	[null]
6	Rajesh	9	[null]

Right Outer Join

It is same as left outer join but in it all records present in right table are there irrespective to whether it is present in left table or not. So the above example will fit into it also if we put basicinformation table in right and scholarship table in left.

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

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

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

student/postgres@PostgreSQL 13

Query Editor Query History

```

205
206
207
208 select scholarshipcategory , count(*) from subjectscholarshipinformation
209 group by scholarshipcategory;
210
211
212
213

```

Data Output Explain Messages Notifications

	scholarshipcategory character varying (20)	count bigint
1	[null]	1
2	BRONZE	4
3	GOLD	1
4	SILVER	1

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

student/postgres@PostgreSQL 13

Query Editor Query History

```

205
206
207
208 select scholarshipcategory , count(*) as noofstudents from subjectscholarshipinformation
209 group by scholarshipcategory order by count(*) desc limit 1;
210
211
212
213

```

Data Output Explain Messages Notifications

	scholarshipcategory character varying (20)	noofstudents bigint
1	BRONZE	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

student/postgres@PostgreSQL 13

Query Editor Query History Scratch P. ✕

```

215 select * from subjectscholarshipinformation as sch inner join studentsubjectinformation
216 as sub on sub.studentrollno = sch.studentrollno order by sch.scholarshipamount desc,
217 sub.studentmarkspercentage desc;
218
219
220
221

```

Data Output Explain Messages Notifications

	scholarshipname character varying (10)	studentrollno integer	scholarshipamount integer	scholarshipcategory character varying (20)	isrequested boolean	isgranted boolean	subjectopted character varying (10)	studentrollno integer	obtainedmarks double precision
1	Sci Sch	7	4000	GOLD	true	true	Science	7	490
2	Comp Sch	1	4000	GOLD	true	true	Computer	1	476
3	Law Sch	3	4000	SILVER	true	false	Law	3	422
4	Comp Sch	6	2000	BRONZE	true	true	English	6	399
5	Sci Sch	2	1000	GOLD	true	false	Science	2	486
6	Comp Sch	5	1000	GOLD	true	true	Computer	5	455
7	Eng Sch	10	1000	[null]	true	false	Computer	10	300

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

Triggers:

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

Stored Procedures:

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

Functions:

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

Views:

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

Thank you.