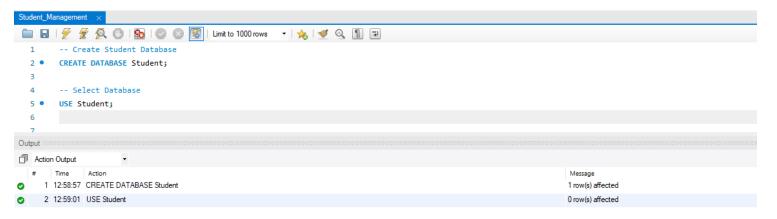
Name: Praveen Kumar Sharma

#### **AU 2021 - SQL Concepts & Fundamentals (ASSIGNMENT)**

#### Q1. Create Student Database



#### Q2. Create the following table under the Student Database:

StudentBasicInformation

StudentAdmissionPaymentDetails

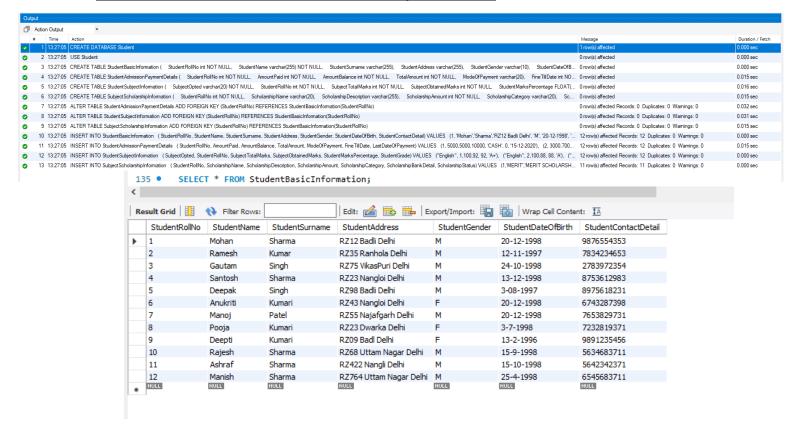
StudentSubjectInformation

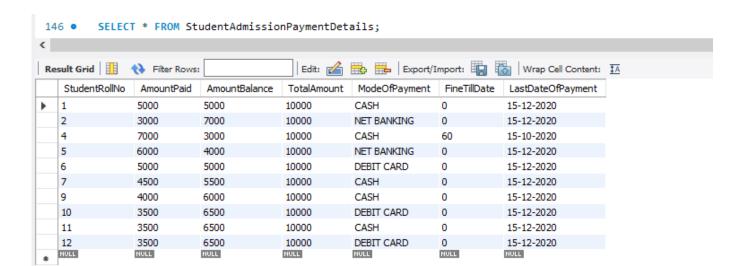
SubjectScholarshipInformation

```
-- Create StudentBasicInformation Table
 8 • ○ CREATE TABLE StudentBasicInformation (
           StudentRollNo int NOT NULL,
           StudentName varchar(255) NOT NULL,
10
11
           StudentSurname varchar(255),
12
           StudentAddress varchar(255),
           StudentGender varchar(10),
           StudentDateOfBirth varchar(20),
14
           StudentContactDetail varchar(22),
15
           PRIMARY KEY (StudentRollNo)
16
17
      - );
18
19
       -- Create StudentAdmissionPaymentDetails Table
20 • ⊝ CREATE TABLE StudentAdmissionPaymentDetails (
           StudentRollNo int NOT NULL,
21
22
           AmountPaid int NOT NULL,
           AmountBalance int NOT NULL,
23
           TotalAmount int NOT NULL,
24
           ModeOfPayment varchar(20),
25
           FineTillDate int NOT NULL,
26
27
           LastDateOfPayment varchar(20),
           PRIMARY KEY (StudentRollNo)
28
      );
29
30
```

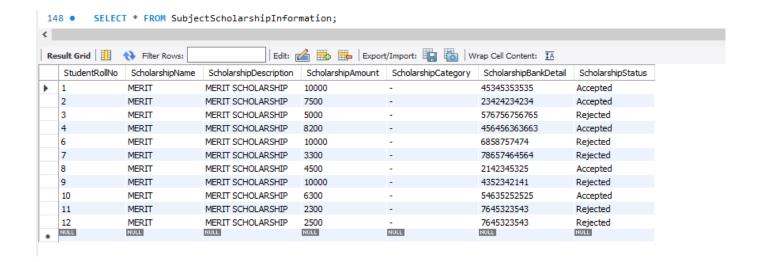
```
-- Create StudentSubjectInformation Table
31
32 • ○ CREATE TABLE StudentSubjectInformation (
            SubjectOpted varchar(20) NOT NULL,
33
34
            StudentRollNo int NOT NULL,
           SubjectTotalMarks int NOT NULL,
35
36
            SubjectObtainedMarks int NOT NULL,
37
            StudentMarksPercentage FLOAT(2) NOT NULL,
            StudentGrade varchar(20),
38
39
            PRIMARY KEY (StudentRollNo)
40
      - );
41
        -- Create SubjectScholarshipInformation Table
42
43 • ○ CREATE TABLE SubjectScholarshipInformation (
           StudentRollNo int NOT NULL,
44
            ScholarshipName varchar(20),
45
            ScholarshipDescription varchar(255),
46
            ScholarshipAmount int NOT NULL,
47
            ScholarshipCategory varchar(20),
48
            ScholarshipBankDetail varchar(20),
49
            ScholarshipStatus varchar(20),
50
            PRIMARY KEY (StudentRollNo)
51
52
      - );
53
```

#### Q3,4. Insert more than 10 records in each and every table created

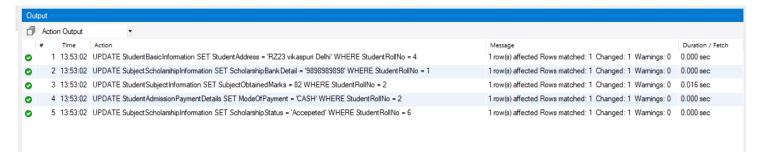


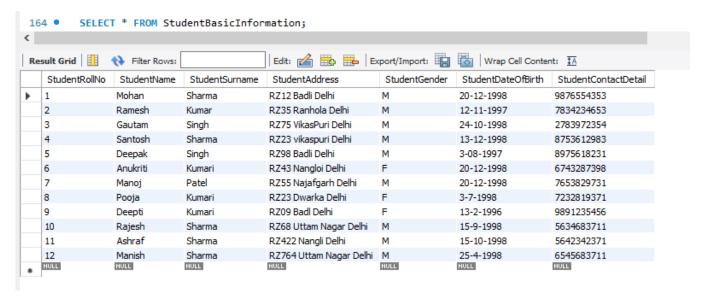


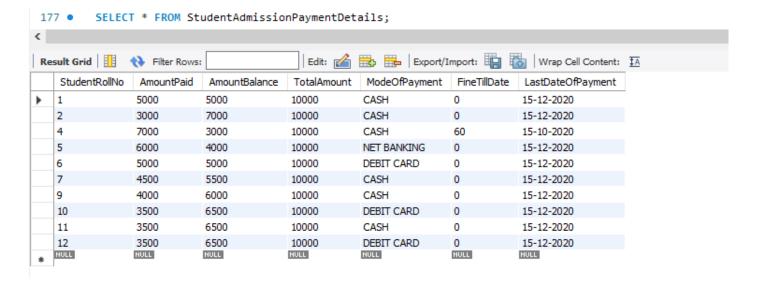
#### SELECT \* FROM StudentSubjectInformation; 147 • Edit: 🝊 🖶 Export/Import: 📳 🐻 | Wrap Cell Content: 🟗 Result Grid Filter Rows: SubjectOpted StudentRollNo SubjectTotalMarks SubjectObtainedMarks StudentMarksPercentage StudentGrade English English Enalish English English English English English English English English NULL NULL NULL NULL NULL NULL



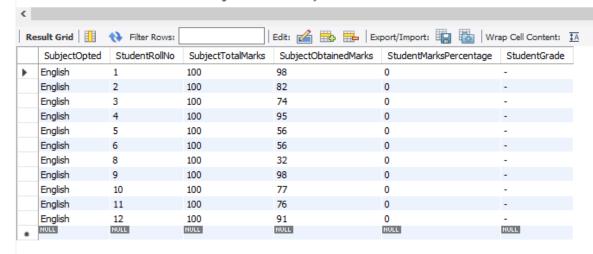
### Q5,6. Update any 5 records of your choice in any table like update the StudentAddress with some other address content and likewise so on with any records of any table of your choice



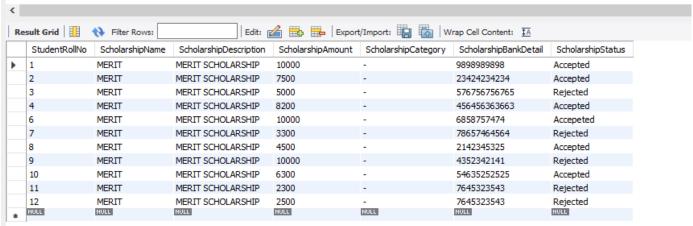




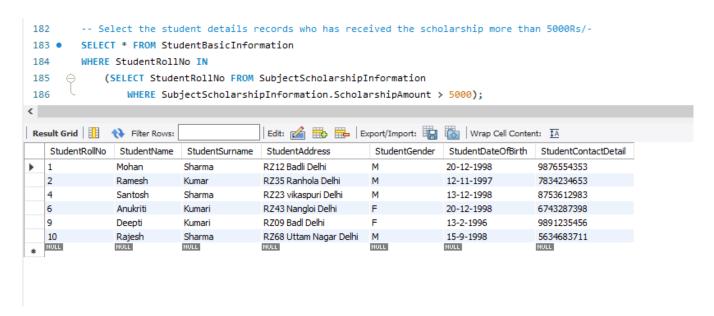
#### 178 • SELECT \* FROM StudentSubjectInformation;



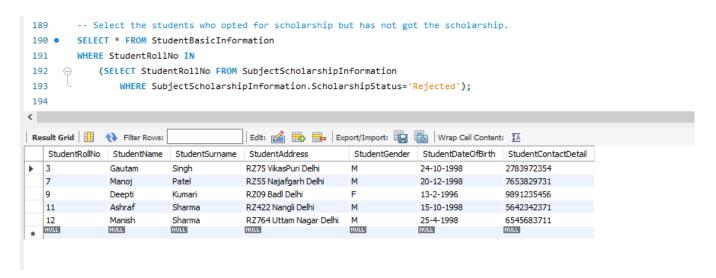
#### 179 • SELECT \* FROM SubjectScholarshipInformation;



#### Q7. Select the student details records who has received the scholarship more than 5000Rs/-



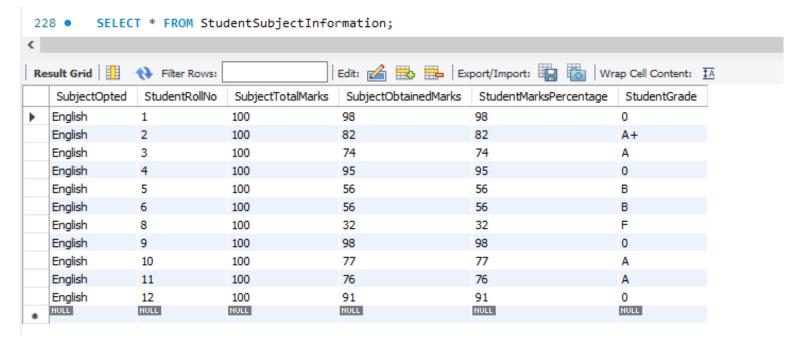
#### Q8. Select the students who opted for scholarship but has not got the scholarship



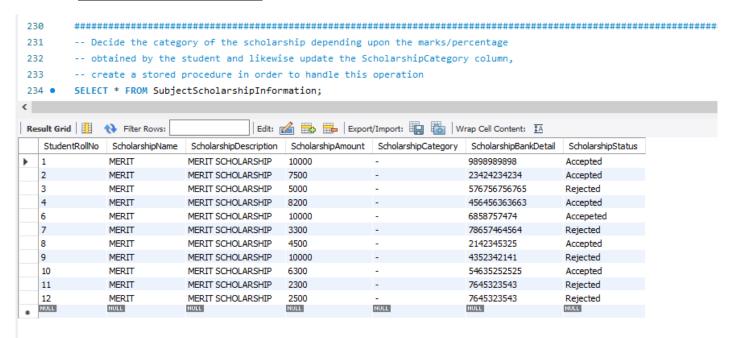
### Q9. Fill in data for the percentage column i.e. StudentMarksPercentage in the table StudentSubjectInformation by creating and using the stored procedure created

19	7 • SELEC	T * FROM Stu	dentSubjectInfo	rmation;		
<						
Re	sult Grid	National Property of the Prope		Edit: 🚄 🖶 🖶 🛭 E	port/Import: 📳 🐻   Wr	ap Cell Content: ‡A
	SubjectOpted	StudentRollNo	SubjectTotalMarks	SubjectObtainedMarks	StudentMarksPercentage	StudentGrade
•	English	1	100	98	0	-
	English	2	100	82	0	-
	English	3	100	74	0	-
	English	4	100	95	0	-
	English	5	100	56	0	-
	English	6	100	56	0	-
	English	8	100	32	0	-
	English	9	100	98	0	-
	English	10	100	77	0	-
	English	11	100	76	0	-
	English	12	100	91	0	-
	NULL	NULL	NULL	NULL	NULL	NULL

```
198
199
      delimiter $$
200 • DROP PROCEDURE IF EXISTS Percentage_Cal;
201
      CREATE procedure Percentage_Cal ()
202
      DETERMINISTIC
203 ⊝ BEGIN
204
          UPDATE StudentSubjectInformation AS sb
205
          SET sb.StudentMarksPercentage=sb.SubjectObtainedMarks/sb.SubjectTotalMarks*100
206
              WHERE sb.StudentRollNo!=0;
207
         UPDATE StudentSubjectInformation AS sb
208
209 SET sb.StudentGrade=CASE
210
              WHEN sb.StudentMarksPercentage>90 THEN '0'
211
               WHEN sb.StudentMarksPercentage>80 THEN 'A+'
              WHEN sb.StudentMarksPercentage>70 THEN 'A'
212
              WHEN sb.StudentMarksPercentage>60 THEN 'B+'
213
214
              WHEN sb.StudentMarksPercentage>50 THEN 'B'
               WHEN sb.StudentMarksPercentage>40 THEN 'C'
215
               WHEN sb.StudentMarksPercentage>33 THEN 'P'
216
               ELSE 'F'
217
218
219
           WHERE sb.StudentRollNo!=0;
220
221
      END $$
222
      delimiter ;
223
224 •
      #-----
225
226
       CALL Percentage_Cal;
```



Q10. 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



```
delimiter $$
237 • DROP PROCEDURE IF EXISTS Category_Cal;
       CREATE procedure Category_Cal ()
238
239
       DETERMINISTIC
240

→ BEGIN

241
          UPDATE SubjectScholarshipInformation AS si
242
           SET si.ScholarshipCategory=CASE
243
               WHEN ((SELECT sp.StudentMarksPercentage FROM StudentSubjectInformation AS sp WHERE sp.StudentRollNo=si.StudentRollNo)>80) THEN 'CAT_1'
               WHEN ((SELECT sp.StudentMarksPercentage FROM StudentSubjectInformation AS sp WHERE sp.StudentRollNo=si.StudentRollNo)>60) THEN 'CAT_2'
244
245
               WHEN ((SELECT sp.StudentMarksPercentage FROM StudentSubjectInformation AS sp WHERE sp.StudentRollNo=si.StudentRollNo)>40) THEN 'CAT_3'
               WHEN ((SELECT sp.StudentMarksPercentage FROM StudentSubjectInformation AS sp WHERE sp.StudentRollNo=si.StudentRollNo)>20) THEN 'CAT_4'
247
               ELSE 'CAT_4'
            END
248
249
            WHERE si.StudentRollNo!=0;
250
      END $$
251
252
       delimiter;
253
254 •
       #-----
255
256
       CALL Category_Cal;
257
```

#### 258 • SELECT \* FROM SubjectScholarshipInformation;

Result Grid   1											
	StudentRollNo	ScholarshipName	ScholarshipDescription	ScholarshipAmount	ScholarshipCategory	ScholarshipBankDetail	ScholarshipStatus				
<b>•</b>	1	MERIT	MERIT SCHOLARSHIP	10000	CAT_1	9898989898	Accepted				
	2	MERIT	MERIT SCHOLARSHIP	7500	CAT_1	23424234234	Accepted				
	3	MERIT	MERIT SCHOLARSHIP	5000	CAT_2	576756756765	Rejected				
	4	MERIT	MERIT SCHOLARSHIP	8200	CAT_1	456456363663	Accepted				
	6	MERIT	MERIT SCHOLARSHIP	10000	CAT_3	6858757474	Accepeted				
	7	MERIT	MERIT SCHOLARSHIP	3300	CAT_4	78657464564	Rejected				
	8	MERIT	MERIT SCHOLARSHIP	4500	CAT_4	2142345325	Accepted				
	9	MERIT	MERIT SCHOLARSHIP	10000	CAT_1	4352342141	Rejected				
	10	MERIT	MERIT SCHOLARSHIP	6300	CAT_2	54635252525	Accepted				
	11	MERIT	MERIT SCHOLARSHIP	2300	CAT_2	7645323543	Rejected				
	12	MERIT	MERIT SCHOLARSHIP	2500	CAT_1	7645323543	Rejected				
	NULL	NULL	NULL	NULL	NULL	NULL	NULL				

### Q11. Create the View which shows balance amount to be paid by the student along with the student detailed information (use join)

```
-- Create the View which shows balance amount to be paid
 261
             -- by the student along with the student detailed information (use join)
 262
            CREATE VIEW BalancePay
 263 •
            AS SELECT a.* , b.AmountBalance
 264
            FROM StudentBasicInformation a INNER JOIN StudentAdmissionPaymentDetails b
 265
            ON a.StudentRollNo=b.StudentRollNo;
 266
268
         SELECT * FROM BalancePay;
Result Grid
              Filter Rows:
                                             Export: Wrap Cell Content: IA
   StudentRollNo
                StudentName
                             StudentSurname
                                             StudentAddress
                                                                   StudentGender
                                                                                 StudentDateOfBirth
                                                                                                   StudentContactDetail
                                                                                                                      AmountBalance
                             Sharma
                                            RZ12 Badli Delhi
                                                                                20-12-1998
                                                                                                  9876554353
   1
  2
                                            RZ35 Ranhola Delhi
                                                                  М
                                                                                12-11-1997
                                                                                                  7834234653
                                                                                                                     7000
                Santosh
                             Sharma
                                            RZ23 vikaspuri Delhi
                                                                                13-12-1998
                                                                                                  8753612983
                                                                                                                     3000
  5
                                            RZ98 Badli Delhi
                                                                                                                     4000
                Deepak
                             Sinah
                                                                  М
                                                                                3-08-1997
                                                                                                  8975618231
  6
                                            RZ43 Nangloi Delhi
                                                                                                                     5000
                Anukriti
                             Kumari
                                                                                20-12-1998
                                                                                                  6743287398
  7
                Manoj
                             Patel
                                            RZ55 Najafgarh Delhi
                                                                                20-12-1998
                                                                                                  7653829731
                                                                                                                     5500
  9
                             Kumari
                                            RZ09 Badl Delhi
                                                                                13-2-1996
                                                                                                  9891235456
                                                                                                                     6000
                Deepti
  10
                Raiesh
                             Sharma
                                            RZ68 Uttam Nagar Delhi
                                                                  М
                                                                                15-9-1998
                                                                                                  5634683711
                                                                                                                     6500
   11
                                            RZ422 Nangli Delhi
                                                                                15-10-1998
                                                                                                  5642342371
                                                                                                                     6500
                Ashraf
                             Sharma
                                                                  М
```

#### Q12. Get the details of the students who haven't got any scholarship (use joins/subqueries)

25-4-1998

6545683711

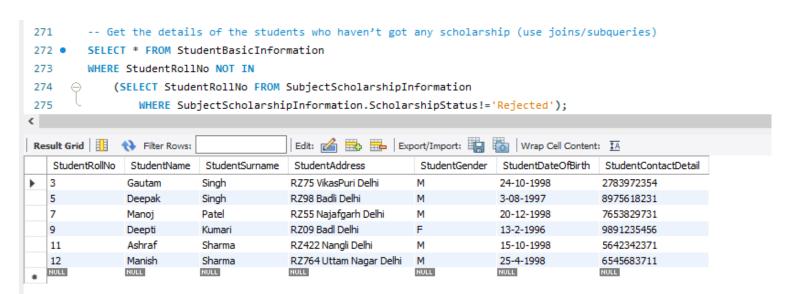
6500

RZ764 Uttam Nagar Delhi

12

Manish

Sharma



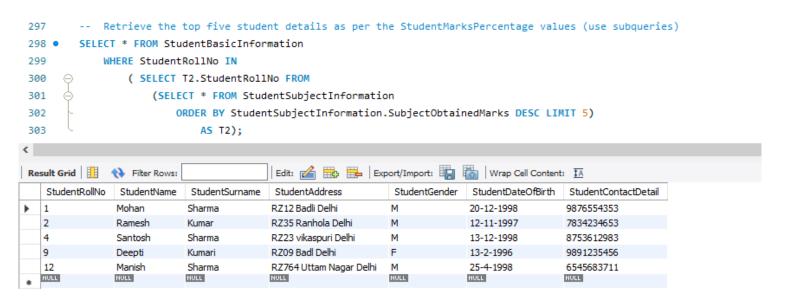
## Q13. Create Stored Procedure which will be return the amount balance to be paid by the student as per the student roll number passed through the stored procedure as the input

```
-- Create Stored Procedure which will be return the amount balance to be paid
278
         -- by the student as per the student roll number passed through the stored procedure as the input
279
280
         delimiter $$
         DROP PROCEDURE IF EXISTS `student`.`Category_Cal`;
281 •
282
         CREATE procedure Category_Cal (IN rollno INT, OUT amount INT)
         DETERMINISTIC
283

→ BEGIN

284
285
             SELECT AmountBalance
286
             INTO amount
287
             FROM StudentAdmissionPaymentDetails
             WHERE StudentRollNo = rollno;
288
         END $$
289
290
         delimiter;
291
292
         #-----
         CALL Category_Cal(2,@balance);
293
294
         SELECT @balance AS Balance_Pay;
295
< □
                                         Export: Wrap Cell Content: 1A
Result Grid
              Filter Rows:
   Balance_Pay
•
  7000
```

### Q14. Retrieve the top five student details as per the StudentMarksPercentage values (use subqueries)

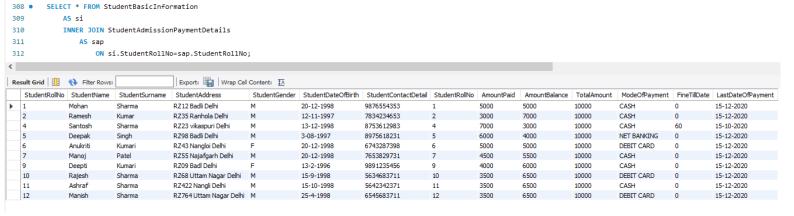


Q15. Try to use all the three types of join learned today in a relevant way, and explain the same why you thought of using that particular join for your selected scenarios (try to cover relevant and real time scenarios for all the three studied joins)

#### 1. INNER JOIN

Selects records that have matching values in both tables.

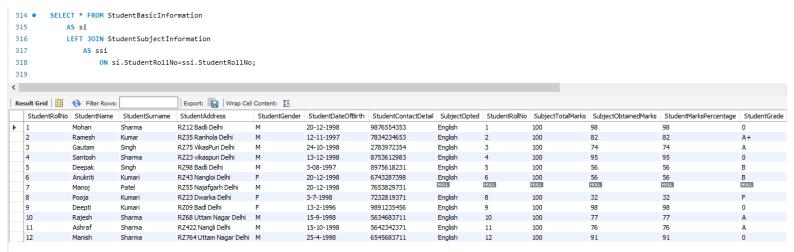
Ex. List the details all students along with admission payment Details:



#### 2. LEFT JOIN

Returns all records from the left table (table1), and the matched records from the right table. The result is NULL from the right side, if there is no match.

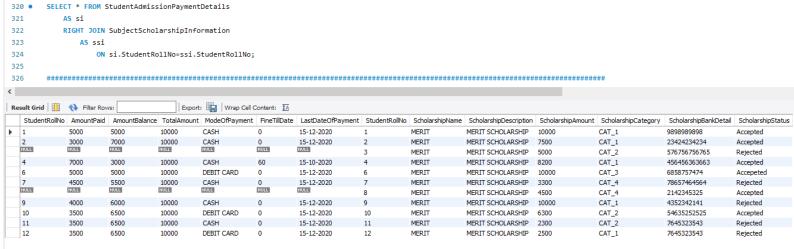
Ex. List the student details along with subject whether he have opted or not if yes, the details of the subject which he have opted.



#### 3. RIGHT JOIN

Returns all records from the right table (table2), and the matched records from the left table (table1). The result is NULL from the left side, when there is no match.

Ex. List the Students who have received the scholarship and status of the Admission payment details whether it is present or not and if present then Complete detail.



#### Q16. Mention the differences between the delete, drop and truncate commands

#### • DELETE:

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.

It is comparatively slower than TRUNCATE cmd.

#### SYNTAX -

If we want to delete all the rows of the table:

#### **DELETE from**;

#### SYNTAX -

If we want to delete the row of the table as per the condition then we use WHERE clause,

#### **DELETE from WHERE;**

#### • 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 or say lost.

#### SYNTAX -

If we want to drop the table:

DROP table;

#### • 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.

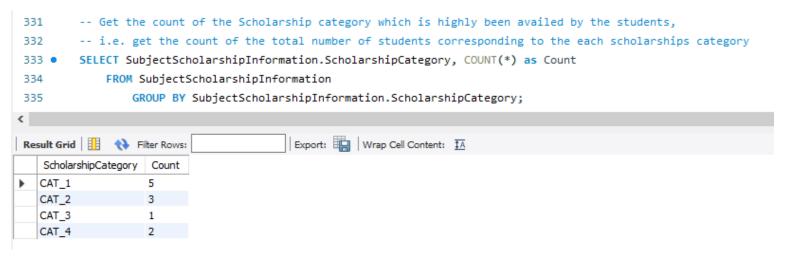
It is comparatively faster than delete command as it deletes all the rows fastly.

#### SYNTAX -

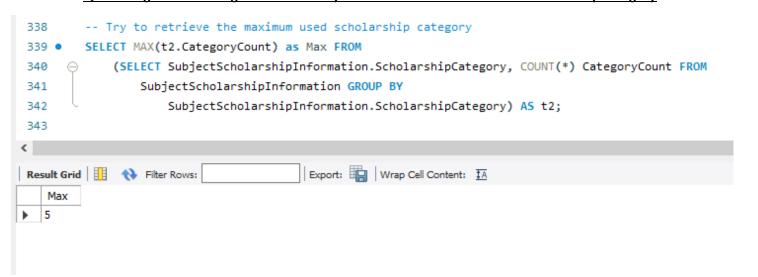
If we want to use truncate:

TRUNCATE;

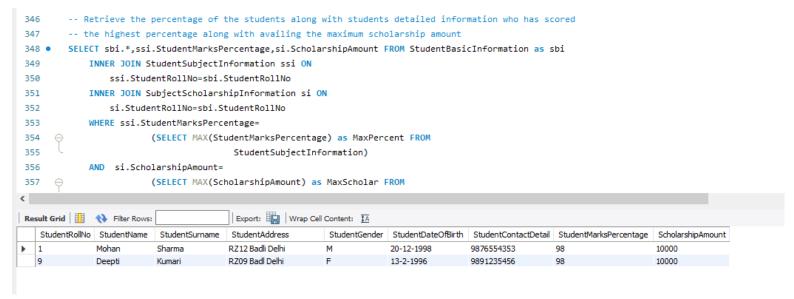
### Q17. Get the count of the Scholarship category which is highly been availed by the students, i.e. get the count of the total number of students corresponding to the each scholarships category



#### Q18. Along with the assignment no. 17 try to retrieve the maximum used scholarship category



# Q 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



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

#### TRIGGERS

- Trigger can be executed automatically on specified action on a table like, update, delete, or update.
- Trigger can't be called from Store Procedure or Function.
- We can't pass a parameter to trigger.
- Trigger never return value on execution.

#### **SYNTAX:**

```
create trigger [trigger_name]
[before | after]
{insert | update | delete}
on [table_name]
[for each row]
[trigger_body]
```

#### • STORED PROCEDURES

- We can execute the stored procedures when required.
- Stored Procedures can't be called from a function because functions can be called from a select statement and Stored Procedures can't be called from. But you can call Store Procedure from Trigger.
- Stored Procedures can accept any type of parameter. Stored Procedures also accept out parameter.
- o Stored Procedures may or may not return any values (Single or table) on execution.

#### SYNTAX:

CREATE PROCEDURE procedure\_name

```
AS sql_statement GO; EXEC procedure_name;
```

#### VIEWS

- o It can be used like a table in most situations, but unlike a table, it can encapsulate very complex calculations and commonly used joins.
- Views are most useful when you always need to join the same set of tables say an Order with an Order
- The view is a virtual table. It does not physically exist. Rather, it is created by query joining one or more tables.
- The fields in a view are fields from one or more real tables in the database.

#### **SYNTAX:**

```
CREATE VIEW view_name AS SELECT column1, column2, ... FROM table_name WHERE condition;
```

#### FUNCTIONS

- We can call a function whenever required. Function can't be executed because a function is not in pre-compiled form.
- Function can be called from Store Procedure or Trigger.
- o Function can accept any type of parameter. But function can't accept out parameter.
- o Function must return any value.

### SYNTAX: