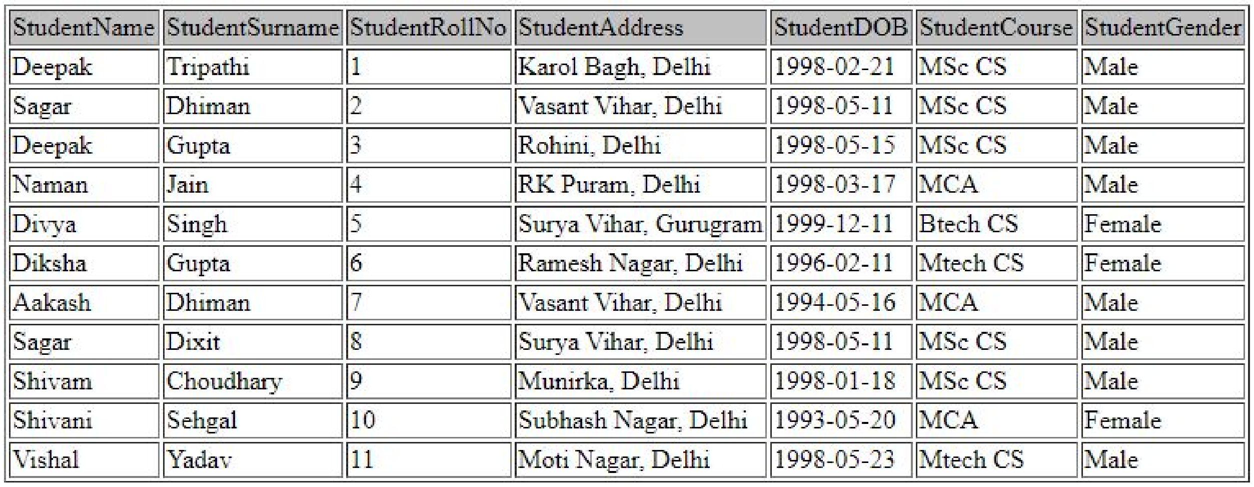
Name: Sahil Prajapati

Assignment of SQL concepts

/* 1 */
CREATE DATABASE Student;

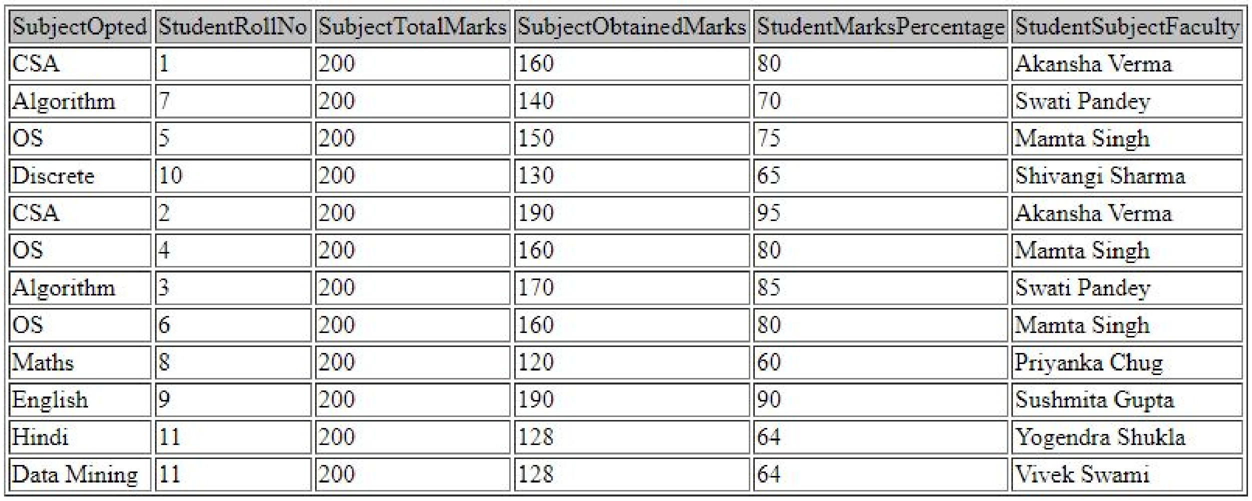
/* 2, 3, 4*/
/* Tables has been created and populated with the data. Kindly see snapshot of populated tables. */

SELECT * FROM studentbasicinformation;



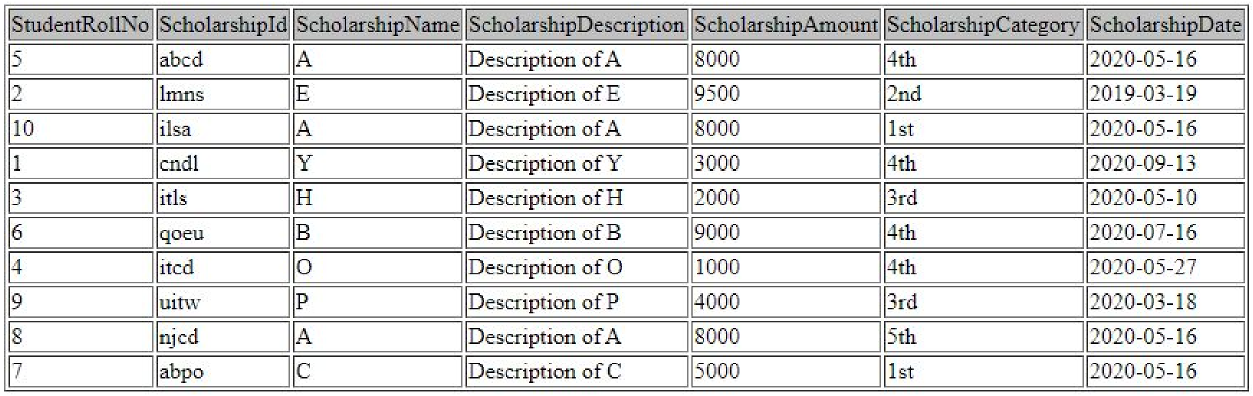
StudentName	StudentSurname	StudentRollNo	StudentAddress	StudentDOB	StudentCourse	StudentGender
Deepak	Tripathi	1	Karol Bagh, Delhi	1998-02-21	MSc CS	Male
Sagar	Dhiman	2	Vasant Vihar, Delhi	1998-05-11	MSc CS	Male
Deepak	Gupta	3	Rohini, Delhi	1998-05-15	MSc CS	Male
Naman	Jain	4	RK Puram, Delhi	1998-03-17	MCA	Male
Divya	Singh	5	Surya Vihar, Gurugram	1999-12-11	Btech CS	Female
Diksha	Gupta	6	Ramesh Nagar, Delhi	1996-02-11	Mtech CS	Female
Aakash	Dhiman	7	Vasant Vihar, Delhi	1994-05-16	MCA	Male
Sagar	Dixit	8	Surya Vihar, Delhi	1998-05-11	MSc CS	Male
Shivam	Choudhary	9	Munirka, Delhi	1998-01-18	MSc CS	Male
Shivani	Sehgal	10	Subhash Nagar, Delhi	1993-05-20	MCA	Female
Vishal	Yadav	11	Moti Nagar, Delhi	1998-05-23	Mtech CS	Male

SELECT * FROM studentsubjectinformation;



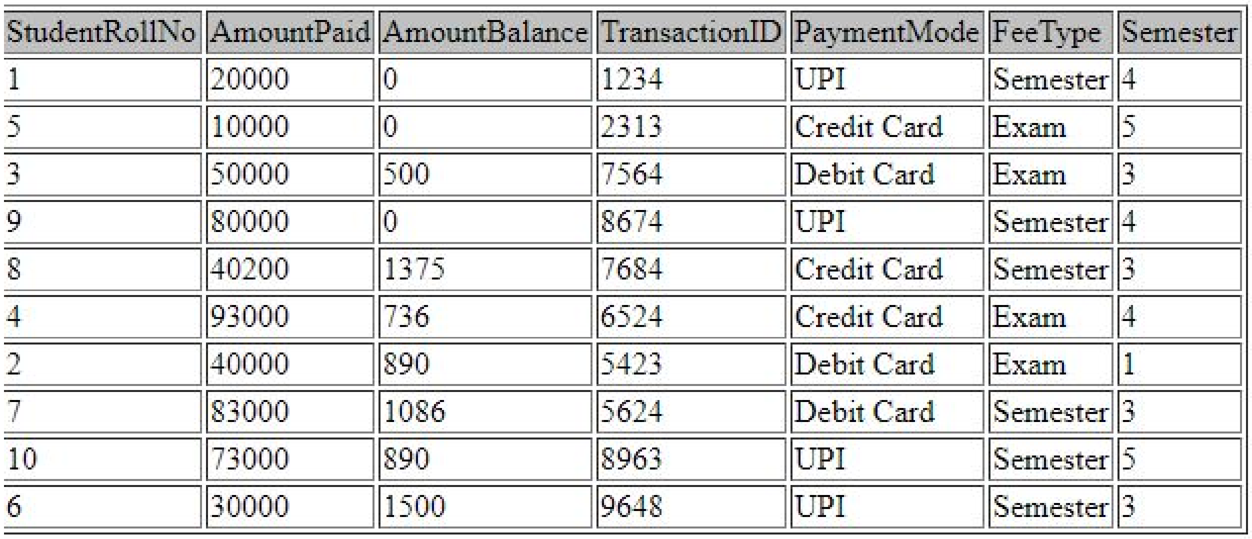
SubjectOpted	StudentRollNo	SubjectTotalMarks	SubjectObtainedMarks	StudentMarksPercentage	StudentSubjectFaculty
CSA	1	200	160	80	Akansha Verma
Algorithm	7	200	140	70	Swati Pandey
OS	5	200	150	75	Mamta Singh
Discrete	10	200	130	65	Shivangi Sharma
CSA	2	200	190	95	Akansha Verma
OS	4	200	160	80	Mamta Singh
Algorithm	3	200	170	85	Swati Pandey
OS	6	200	160	80	Mamta Singh
Maths	8	200	120	60	Priyanka Chug
English	9	200	190	90	Sushmita Gupta
Hindi	11	200	128	64	Yogendra Shukla
Data Mining	11	200	128	64	Vivek Swami

```
SELECT * FROM studentscholarshipinformation;
```



StudentRollNo	ScholarshipId	ScholarshipName	ScholarshipDescription	ScholarshipAmount	ScholarshipCategory	ScholarshipDate
5	abcd	A	Description of A	8000	4th	2020-05-16
2	lmns	E	Description of E	9500	2nd	2019-03-19
10	ilsa	A	Description of A	8000	1st	2020-05-16
1	cndl	Y	Description of Y	3000	4th	2020-09-13
3	itls	H	Description of H	2000	3rd	2020-05-10
6	qoeu	B	Description of B	9000	4th	2020-07-16
4	itcd	O	Description of O	1000	4th	2020-05-27
9	uitw	P	Description of P	4000	3rd	2020-03-18
8	njcd	A	Description of A	8000	5th	2020-05-16
7	abpo	C	Description of C	5000	1st	2020-05-16

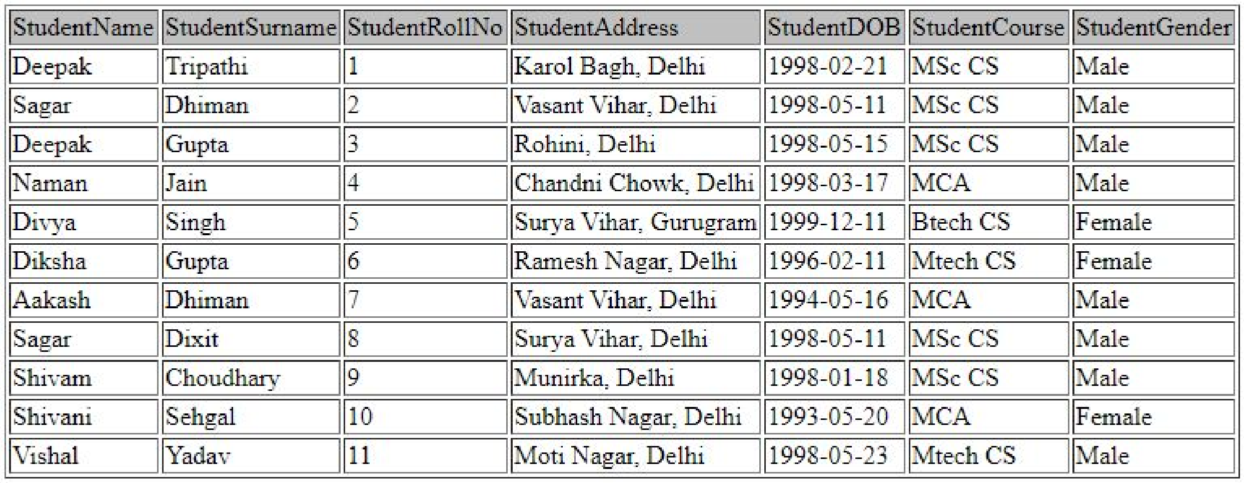
```
SELECT * FROM studentadmissionpaymentdetails;
```



StudentRollNo	AmountPaid	AmountBalance	TransactionID	PaymentMode	FeeType	Semester
1	20000	0	1234	UPI	Semester	4
5	10000	0	2313	Credit Card	Exam	5
3	50000	500	7564	Debit Card	Exam	3
9	80000	0	8674	UPI	Semester	4
8	40200	1375	7684	Credit Card	Semester	3
4	93000	736	6524	Credit Card	Exam	4
2	40000	890	5423	Debit Card	Exam	1
7	83000	1086	5624	Debit Card	Semester	3
10	73000	890	8963	UPI	Semester	5
6	30000	1500	9648	UPI	Semester	3

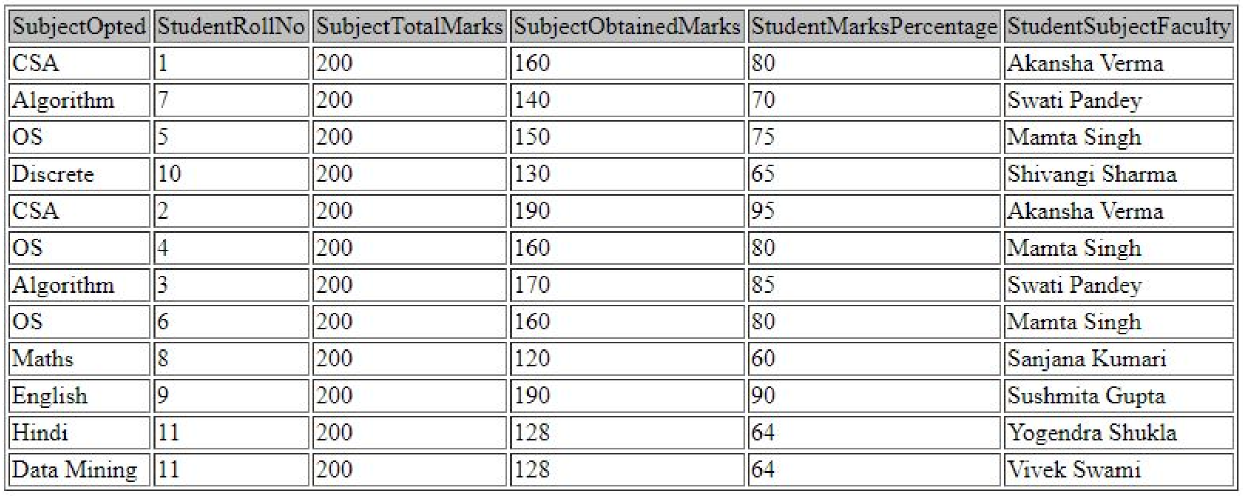
/* 5, 6 */

```
UPDATE studentbasicinformation
SET StudentAddress = "Chandni Chowk, Delhi"
WHERE StudentRollNo = 4;

SELECT * FROM studentbasicinformation;
```



StudentName	StudentSurname	StudentRollNo	StudentAddress	StudentDOB	StudentCourse	StudentGender
Deepak	Tripathi	1	Karol Bagh, Delhi	1998-02-21	MSc CS	Male
Sagar	Dhiman	2	Vasant Vihar, Delhi	1998-05-11	MSc CS	Male
Deepak	Gupta	3	Rohini, Delhi	1998-05-15	MSc CS	Male
Naman	Jain	4	Chandni Chowk, Delhi	1998-03-17	MCA	Male
Divya	Singh	5	Surya Vihar, Gurugram	1999-12-11	Btech CS	Female
Diksha	Gupta	6	Ramesh Nagar, Delhi	1996-02-11	Mtech CS	Female
Aakash	Dhiman	7	Vasant Vihar, Delhi	1994-05-16	MCA	Male
Sagar	Dixit	8	Surya Vihar, Delhi	1998-05-11	MSc CS	Male
Shivam	Choudhary	9	Munirka, Delhi	1998-01-18	MSc CS	Male
Shivani	Sehgal	10	Subhash Nagar, Delhi	1993-05-20	MCA	Female
Vishal	Yadav	11	Moti Nagar, Delhi	1998-05-23	Mtech CS	Male

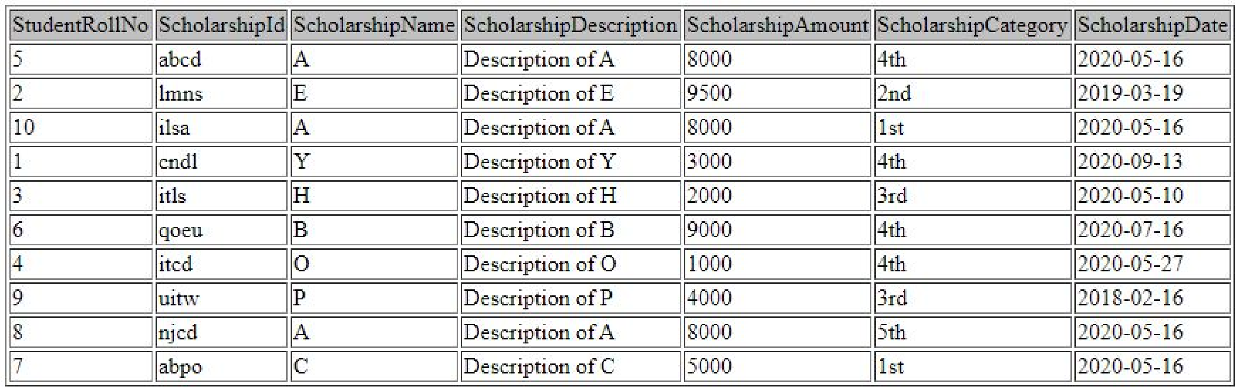
```
UPDATE studentsubjectinformation
SET StudentSubjectFaculty = "Sanjana Kumari"
WHERE StudentRollNo = 8;

SELECT * FROM studentsubjectinformation;
```



SubjectOpted	StudentRollNo	SubjectTotalMarks	SubjectObtainedMarks	StudentMarksPercentage	StudentSubjectFaculty
CSA	1	200	160	80	Akansha Verma
Algorithm	7	200	140	70	Swati Pandey
OS	5	200	150	75	Mamta Singh
Discrete	10	200	130	65	Shivangi Sharma
CSA	2	200	190	95	Akansha Verma
OS	4	200	160	80	Mamta Singh
Algorithm	3	200	170	85	Swati Pandey
OS	6	200	160	80	Mamta Singh
Maths	8	200	120	60	Sanjana Kumari
English	9	200	190	90	Sushmita Gupta
Hindi	11	200	128	64	Yogendra Shukla
Data Mining	11	200	128	64	Vivek Swami

```
UPDATE studentscholarshipinformation
SET ScholarshipDate = 20180216
WHERE StudentRollNo = 9;
```

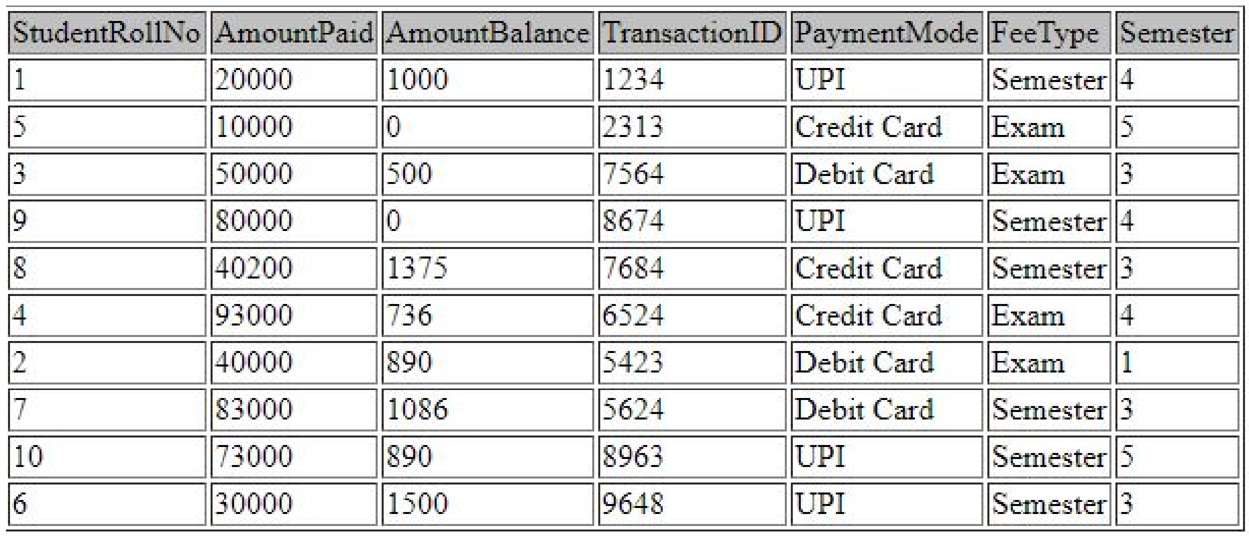
```
SELECT * FROM studentscholarshipinformation;
```



StudentRollNo	ScholarshipId	ScholarshipName	ScholarshipDescription	ScholarshipAmount	ScholarshipCategory	ScholarshipDate
5	abcd	A	Description of A	8000	4th	2020-05-16
2	lmns	E	Description of E	9500	2nd	2019-03-19
10	ilsa	A	Description of A	8000	1st	2020-05-16
1	cndl	Y	Description of Y	3000	4th	2020-09-13
3	itls	H	Description of H	2000	3rd	2020-05-10
6	qoeu	B	Description of B	9000	4th	2020-07-16
4	itcd	O	Description of O	1000	4th	2020-05-27
9	uitw	P	Description of P	4000	3rd	2018-02-16
8	njcd	A	Description of A	8000	5th	2020-05-16
7	abpo	C	Description of C	5000	1st	2020-05-16

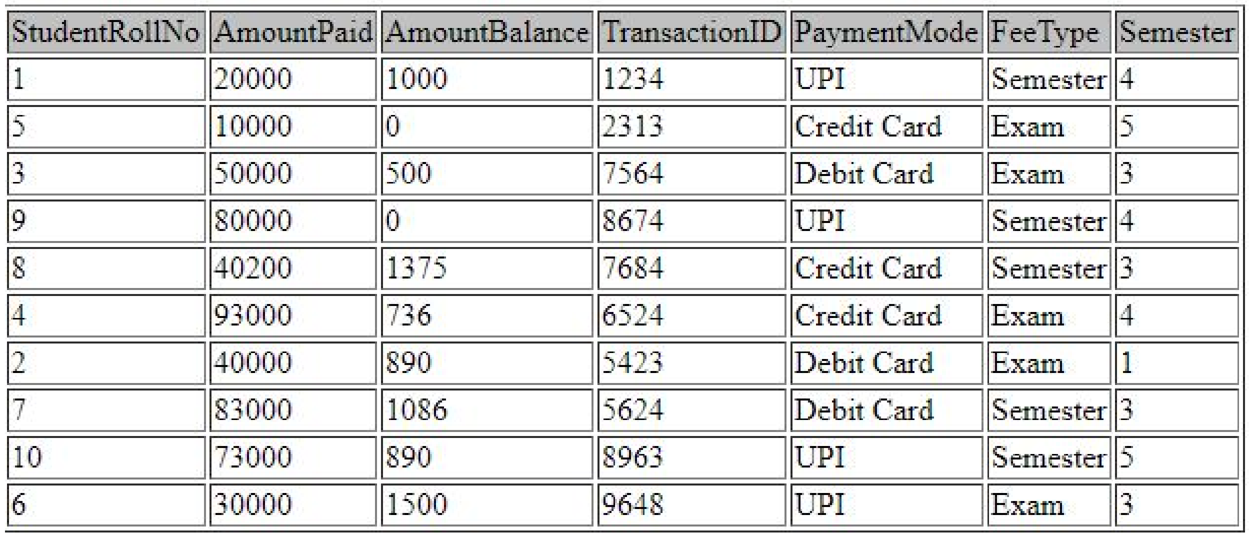
```
UPDATE studentadmissionpaymentdetails
SET AmountBalance = 1000
WHERE StudentRollNo = 1;
```

```
SELECT * FROM studentadmissionpaymentdetails;
```



StudentRollNo	AmountPaid	AmountBalance	TransactionID	PaymentMode	FeeType	Semester
1	20000	1000	1234	UPI	Semester	4
5	10000	0	2313	Credit Card	Exam	5
3	50000	500	7564	Debit Card	Exam	3
9	80000	0	8674	UPI	Semester	4
8	40200	1375	7684	Credit Card	Semester	3
4	93000	736	6524	Credit Card	Exam	4
2	40000	890	5423	Debit Card	Exam	1
7	83000	1086	5624	Debit Card	Semester	3
10	73000	890	8963	UPI	Semester	5
6	30000	1500	9648	UPI	Semester	3

```
UPDATE studentadmissionpaymentdetails
SET FeeType = "Exam"
WHERE StudentRollNo = 6;

SELECT * FROM studentadmissionpaymentdetails;
```



StudentRollNo	AmountPaid	AmountBalance	TransactionID	PaymentMode	FeeType	Semester
1	20000	1000	1234	UPI	Semester	4
5	10000	0	2313	Credit Card	Exam	5
3	50000	500	7564	Debit Card	Exam	3
9	80000	0	8674	UPI	Semester	4
8	40200	1375	7684	Credit Card	Semester	3
4	93000	736	6524	Credit Card	Exam	4
2	40000	890	5423	Debit Card	Exam	1
7	83000	1086	5624	Debit Card	Semester	3
10	73000	890	8963	UPI	Semester	5
6	30000	1500	9648	UPI	Exam	3

```
/* 7 */
SELECT * FROM StudentBasicInformation WHERE StudentRollNo IN
        (SELECT StudentRollNo FROM studentscholarshipinformation WHERE
ScholarshipAmount > 5000);

/* 8 */
SELECT * FROM StudentBasicInformation WHERE StudentRollNo NOT IN
        (SELECT StudentRollNo FROM studentscholarshipinformation);

/* 9 */
Delimiter //
CREATE PROCEDURE enterPercentage(
        IN rollNO INT
   )
   BEGIN
                UPDATE studentsubjectinformation
      SET       StudentMarksPercentage = (SubjectObtainedMarks * 100)/SubjectTotalMarks
      WHERE StudentRollNo = rollNO;
        END//
```

```
delimiter ;
CALL enterPercentage(11);


select * from studentscholarshipinformation;
```

/* 10 */

```
CREATE table vw
(SELECT * FROM studentscholarshipinformation NATURAL JOIN StudentBasicInformation);

Delimiter //
CREATE PROCEDURE updateScholarshipCategory()
        BEGIN
                update studentscholarshipinformation
                set ScholarshipCategory =
                        Case when StudentRollNo IN (Select StudentRollNo from vw where
vw.StudentMarksPercentage between 61 AND 70) THEN "4th"
                                when StudentRollNo IN (Select StudentRollNo from vw where
vw.StudentMarksPercentage between 71 AND 80) THEN "3rd"
                                when StudentRollNo IN (Select StudentRollNo from vw where
vw.StudentMarksPercentage between 81 AND 90) THEN "2nd"
                                when StudentRollNo IN (Select StudentRollNo from vw where
vw.StudentMarksPercentage between 91 AND 100) THEN "1st"
                                else ScholarshipCategory

                        end;
   END//
Delimiter ;

Call updateScholarshipCategory();

create table vw as
 Select * from studentscholarshipinformation Natural JOIN
                                                        studentsubjectinformation;
```

/* 11 */

```
CREATE VIEW BalanceAmountDetails
```

```
AS
SELECT AmountBalance,StudentName, StudentSurname, StudentRollNo, StudentAddress,
StudentDOB, StudentCourse, StudentGender
FROM studentbasicinformation NATURAL JOIN studentadmissionpaymentdetails;

SELECT * FROM BalanceAmountDetails;
```

/* 12 */

```
SELECT * FROM StudentBasicInformation WHERE StudentRollNo NOT IN
        (SELECT StudentRollNo FROM studentscholarshipinformation);
```

/* 13 */

```
Delimiter //
CREATE PROCEDURE AmountBalanceToBePaid(
        IN rollNo INT
   )
   BEGIN
                SELECT AmountBalance
      FROM studentadmissionpaymentdetails
      WHERE StudentRollNo = rollNo;
   END//

Delimiter ;

CALL AmountBalanceToBePaid(5);
CALL AmountBalanceToBePaid(10);
```

/* 14 */

method1

```
SELECT StudentName, StudentSurname
FROM StudentBasicInformation NATURAL JOIN StudentSubjectInformation
ORDER BY StudentMarksPercentage DESC LIMIT 5;
```

method2

```
SELECT StudentRollNo, StudentName
FROM StudentBasicInofrmation
WHERE StudentRollNo IN
```

(SELECT StudentRollNo FROM StudentSubjectInformation ORDER BY
SubjjectObtainedMarks DESC
LIMIT 5);

/* 15 */

- **JOIN**
  SELECT *
  FROM StudentBasicInofrmation as st **JOIN** StudentSubjectInformation sb
  ON st.StudentRollNo = sb.StudentRollNo;

  This JOIN is used so that we can find out the details of all those student and faculty information which are been connected with each other. Meaning that all those student that has been assigned with a faculty, Similarily vice versa.

- **LEFT JOIN**
  SELECT *
  FROM StudentBasicInofrmation as st **LEFT JOIN** StudentScholarshipInformation sb
  ON st.StudentRollNo = sb.StudentRollNo;
  This JOIN is used to find out the information of all the student and whether they have been given their scholarship or not. If for a student the values of scholarship are NULL then it is understood that student is not given the scholarship.

- **RIGHT JOIN**
  SELECT *
  FROM studentadmissionpaymentdetails as sb **RIGHT JOIN** StudentBasicInofrmation as st **ON** st.StudentRollNo = sb.StudentRollNo;

  This JOIN is used to find out the information of all the students that whether they have paid their fees or not. If NULL values are present in payment columns then that student have not paid his/her fee.

/* 16 */

- **DELETE**
  It is used to delte the rows, we can delete row by row or all the rows in one go. The space for the records remain in the database. And we can insert the values again in that table.

- **TRUNCATE**

Remove all records from a table, including all spaces allocated for the records are removed and we can not insert the values again. For that we have to again do the DDL part.

- **DROP**
  It is used to delete objects from the database.

/* 17 */

```
SELECT ScholarshipCategory, Count(*) as CountofStudents
FROM studentscholarshipinformation
GROUP BY ScholarshipCategory;
```

/* 18 */

```
SELECT ScholarshipCategory, Count(*) as CountofStudents
FROM studentscholarshipinformation
GROUP BY ScholarshipCategory
ORDER BY CountofStudents DESC
LIMIT 2;
```

/* 19 */

```
SELECT StudentName, StudentMarksPercentage, ScholarshipAmount,StudentSurname,
StudentRollNo, StudentAddress,
            StudentDOB, StudentCourse, StudentGender
FROM (studentbasicinformation NATURAL JOIN studentscholarshipinformation NATURAL
JOIN studentsubjectinformation)
ORDER BY StudentMarksPercentage DESC
LIMIT 1;
```

/* 20 */

- **TRIGGERS**
  A trigger is a program which is called automatically on occuring of any type of event such as insert, update, or delete. For example, you can define a trigger that is invoked automatically before a new row is inserted into a table.

- **STORED PROCEDURE**
  It is an SQL code that is defined by the user such that it can be used again and again. So if you have an SQL query that you write over and over again, save it as a stored procedure, and then just call it to execute it.

- **VIEW**

Suppose if we have a query which is to be used again and again. And that query is computationally very expensive to be computed. So instead of executing that query again and again, we can save that query as a temporary table. Such that if have to use that query we can simply access it from that temporary table which is called view.

- **FUNCTIONS**
  A function is a stored program that you can pass parameters(if required) into and then return a value. We have many in-built functions also like aggregate funtions , date functions and many more.