SQL Assignment

Submitted by:

Kajal Gupta

Q1. Create Student Database

CREATE DATABASE student;

use student;

Q2. Create the following table under the Student Database:

If I make a column StudentRollNo. in the ScholarshipDetails table then it would create a lot a replicated data as many students would have same kind of scholarship. So I made a new table that contains the StudentRollNo. and ScholarshipName only. Using this table we could match the details of the student and scholarship given.

a. StudentBasicInformation

i. Columns

1. StudentName

2. StudentSurname

3. StudentRollNo

4. StudentAddress

5. Add more three basic columns of the name of your own

```
CREATE TABLE StudentBasicInformation (
   StudentRollNo int NOT NULL,
   StudentName varchar(255) NOT NULL,
   StudentSurname varchar(255),
```

```
    StudentAddress varchar(255),
    Age int,
    ScholarshipOpted varchar(255),
    Gender varchar(255),
    PRIMARY KEY (StudentRollNo)
);
```

b. StudentAdmissionPaymentDetails

i. Columns

1. StudentRollNo

2. AmountPaid

3. AmountBalance

4. Add more four basic columns of the name of your own

```
CREATE TABLE StudentAdmissionPaymentDetails(
    AmountPaid int,
    AmountBalance int,
    StudentRollNo int,
    AdmissionDate date,
    DueDate date,
    FOREIGN KEY (StudentRollNo) REFERENCES StudentBasicInformation(StudentRollNo)
);
```

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 StudentSubjectInformation
(
        StudentRollNo int,
        SubjectOpted varchar(255) NOT NULL,
         SubjectTotalMarks int,
         SubjectObtainedMarks int,
         StudentMarksPercentage int,
        MentorName varchar(255),
        FOREIGN KEY (StudentRollNo) REFERENCES StudentBasicInformation(StudentRollNo)
);
```

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 SubjectScholarshipInformation
(
        ScholarshipName varchar(255),
        ScholarshipDescription varchar(2000),
        ScholarshipAmount int,
        ScholarshipCategory varchar(255),
        PRIMARY KEY (ScholarshipName)
);
```

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

```
INSERT INTO StudentBasicInformation
VALUES (1, 'John', 'Hopkins', '305 - 14th Ave. S. Suite 3B', 19, 'Yes', 'Male');

INSERT INTO StudentBasicInformation
VALUES (2, 'Karl', 'Jablonski', '14th Suite 3B', 20, 'Yes', 'Male');

INSERT INTO StudentBasicInformation
VALUES (3, 'Matti', 'Karttunen', 'Keskuskatu 45    Helsinki', 20, 'Yes', 'Female');

INSERT INTO StudentBasicInformation
VALUES (4, 'Tom', ' B. Erichsen', 'Skagen 21 Stavanger', 20, 'Yes', 'Male');

INSERT INTO StudentBasicInformation
VALUES (5, 'Hekkan', 'Burger', 'Gateveien 15Sandnes', 19, 'No', 'Male');

INSERT INTO StudentBasicInformation
VALUES (6, 'Julia', 'Yasminr', '13 Hibiscus Road', 19, 'Yes', 'Female');

INSERT INTO StudentBasicInformation
VALUES (7, 'Nicholas', 'Wong', '24 Metro Street', 20, 'No', 'Male');

INSERT INTO StudentBasicInformation
VALUES (8, 'Niko', 'Bellic', '53 Hove Beach', 20, 'No', 'Male');

INSERT INTO StudentBasicInformation
VALUES (9, 'David', 'Kim', '4 Lake Garden', 20, 'Yes', 'Male');

INSERT INTO StudentBasicInformation
VALUES (10, 'Carl', 'Johnson', '77 Park Lane', 19, 'Yes', 'Male');

INSERT INTO StudentBasicInformation
```

VALUES (11, 'John', 'Kay', '22 Lin Seng Park', 19, 'No', 'Male');

INSERT INTO StudentAdmissionPaymentDetails
VALUES (1000, 9000, 1, '2008-04-03', '2008-05-01');

INSERT INTO StudentAdmissionPaymentDetails
VALUES (3000, 7000, 2, '2008-04-05', '2008-05-01');

INSERT INTO StudentAdmissionPaymentDetails
VALUES (5000, 5000, 3, '2008-04-08', '2008-05-01');

INSERT INTO StudentAdmissionPaymentDetails
VALUES (7000, 3000, 4, '2008-04-01', '2008-05-01');

INSERT INTO StudentAdmissionPaymentDetails
VALUES (6000, 4000, 5, '2008-04-02', '2008-05-01');

INSERT INTO StudentAdmissionPaymentDetails
VALUES (4000, 6000, 6, '2008-04-12', '2008-05-01');

INSERT INTO StudentAdmissionPaymentDetails
VALUES (1000, 9000, 7, '2008-04-14', '2008-05-01');

INSERT INTO StudentAdmissionPaymentDetails
VALUES (1500, 8500, 8, '2008-04-09', '2008-05-01');

INSERT INTO StudentAdmissionPaymentDetails
VALUES (2500, 7500, 9, '2008-04-08', '2008-05-01');

INSERT INTO StudentAdmissionPaymentDetails

VALUES (9500, 500, 10, '2008-04-03', '2008-05-01');

INSERT INTO StudentAdmissionPaymentDetails

VALUES (10000, 0, 11, '2008-04-07', '2008-05-01');

INSERT INTO StudentSubjectInformation (StudentRollNo, SubjectOpted, SubjectTotalMarks, SubjectObtainedMarks, MentorName)

VALUES (1, 'English', 100, 76, 'Jessy Gill');

INSERT INTO StudentSubjectInformation (StudentRollNo, SubjectOpted, SubjectTotalMarks, SubjectObtainedMarks, MentorName)

VALUES (2, 'Computer Science', 100, 96, 'Maria Garcia');

INSERT INTO StudentSubjectInformation (StudentRollNo, SubjectOpted, SubjectTotalMarks, SubjectObtainedMarks, MentorName)

VALUES (3, 'Political Science', 100, 80, 'Michael Smith');

INSERT INTO StudentSubjectInformation (StudentRollNo, SubjectOpted, SubjectTotalMarks, SubjectObtainedMarks, MentorName)

VALUES (4, 'History', 100, 60, 'Maria Rodriguez');

INSERT INTO StudentSubjectInformation (StudentRollNo, SubjectOpted, SubjectTotalMarks, SubjectObtainedMarks, MentorName)

VALUES (5, 'Statistics', 100, 96, 'Maria Hernandez');

INSERT INTO StudentSubjectInformation (StudentRollNo, SubjectOpted, SubjectTotalMarks, SubjectObtainedMarks, MentorName)

VALUES (6, 'Calculas', 100, 68, 'Joseph Samuel');

INSERT INTO StudentSubjectInformation (StudentRollNo, SubjectOpted, SubjectTotalMarks, SubjectObtainedMarks, MentorName)

```
VALUES (7, 'Geometry', 100, 89, 'Henry');

INSERT INTO StudentSubjectInformation (StudentRollNo, SubjectOpted, SubjectTotalMarks,
SubjectObtainedMarks, MentorName )

VALUES (8, 'English', 100, 46, 'Jessy Gill');

INSERT INTO StudentSubjectInformation (StudentRollNo, SubjectOpted, SubjectTotalMarks,
SubjectObtainedMarks, MentorName )

VALUES (9, 'Geometry', 100, 76, 'Henry');

INSERT INTO StudentSubjectInformation (StudentRollNo, SubjectOpted, SubjectTotalMarks,
SubjectObtainedMarks, MentorName )

VALUES (10, 'Political Science', 100, 46, 'Michael Smith');

INSERT INTO StudentSubjectInformation (StudentRollNo, SubjectOpted, SubjectTotalMarks,
SubjectObtainedMarks, MentorName )

VALUES (11, 'Statistics', 100, 76, 'Maria Hernandez');
```

```
insert into SubjectScholarshipInformation

values('Inspire our Future', 'Scholarship given to students who score more than 90% marks in any
subject', 100, 'Excellent');

insert into SubjectScholarshipInformation

values('Little Heart', 'Scholarship given to students who score more than 80% marks and less than
90% in any subject', 90, 'Very Good');

insert into SubjectScholarshipInformation

values('I matter', 'Scholarship given to students who score more than 70% marks and less than 80%
in any subject', 80, 'Good');
```

Q4. Snap of the all the tables once the insertion is completed

select * from StudentBasicInformation

StudentRollNo	StudentName	StudentSurname	StudentAddress	Age	ScholarshipOpted	Gender
1	John	Hopkins	305 - 14th Ave. S. Suite 3B	19	Yes	Male
2	Karl	Jablonski	14th Suite 3B	20	Yes	Male
3	Matti	Karttunen	Keskuskatu 45Helsinki	20	Yes	Female
4	Tom	B. Erichsen	Skagen 21 Stavanger	20	Yes	Male
5	Hekkan	Burger	Gateveien 15Sandnes	19	No	Male
6	Julia	Yasminr	13 Hibiscus Road	19	Yes	Female
7	Nicholas	Wong	24 Metro Street	20	No	Male
8	Niko	Bellic	53 Hove Beach	20	No	Male
9	David	Kim	4 Lake Garden	20	Yes	Male
10	Carl	Johnson	77 Park Lane	19	Yes	Male
11	John	Kay	22 Lin Seng Park	19	No	Male

select * from StudentAdmissionPaymentDetails;

AmountPaid	AmountBalance	StudentRollNo	AdmissionDate	DueDate
1000	9000	1	2008-04-03	2008-05-01
3000	7000	2	2008-04-05	2008-05-01
5000	5000	3	2008-04-08	2008-05-01
7000	3000	4	2008-04-01	2008-05-01
6000	4000	5	2008-04-02	2008-05-01
4000	6000	6	2008-04-12	2008-05-01
1000	9000	7	2008-04-14	2008-05-01
1500	8500	8	2008-04-09	2008-05-01
2500	7500	9	2008-04-08	2008-05-01
9500	500	10	2008-04-03	2008-05-01
10000	0	11	2008-04-07	2008-05-01

select * from StudentSubjectInformation;

StudentRollNo	SubjectOpted	SubjectTotalMarks	SubjectObtainedMarks	StudentMarksPercentage	MentorName
2	Computer Science	100	96	NULL	Maria Garcia
3	Political Science	100	80	NULL	Michael Smith
4	History	100	60	NULL	Maria Rodriguez
5	Statistics	100	96	NULL	Maria Hernandez
6	Calculas	100	68	NULL	Joseph Samuel
7	Geometry	100	89	NULL	Henry
8	English	100	46	NULL	Jessy Gill
9	Geometry	100	76	NULL	Henry
10	Political Science	100	46	NULL	Michael Smith
11	Statistics	100	76	NULL	Maria Hernandez
1	English	100	76	NULL	Jessy Gill

select * from SubjectScholarshipInformation;

ScholarshipName	ScholarshipDescription	ScholarshipAmount	ScholarshipCategory
Inspire our Future	Scholarship given to students who score more than 90% marks in any subject	100	Excellent
Little Heart	Scholarship given to students who score more than 80% marks and less than 90% in any subject	90	Very Good
I matter	Scholarship given to students who score more than 70% marks and less than 80% in any subject	80	Good

---

I have created this table to match scholarship details with student details

```
create table ScholarshipStudent
(
        StudentRollNo int,
        ScholarshipName varchar(255),
   FOREIGN KEY (StudentRollNo) REFERENCES StudentBasicInformation(StudentRollNo),
   FOREIGN KEY (ScholarshipName) REFERENCES SubjectScholarshipInformation(ScholarshipName)
);
```

StudentRollNo	ScholarshipName

Q5. 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 StudentName = 'Alfred'
WHERE StudentRollNo = 1;

UPDATE StudentBasicInformation
```

```
SET StudentName = 'Alfred'

WHERE StudentRollNo = 1;


UPDATE StudentSubjectInformation

SET SubjectOpted = 'English'

WHERE StudentRollNo = 2;


UPDATE StudentSubjectInformation

SET MentorName = 'Ralph Lauren'

WHERE StudentRollNo = 4;


UPDATE StudentAdmissionPaymentDetails

SET AmountPaid = 5000, AmountBalance=5000

WHERE StudentRollNo = 11;
```

Q6. Snap of the all the tables post updation

StudentRollNo	StudentName	StudentSurname	StudentAddress	Age	ScholarshipOpted	Gender
1	Alfred	Hopkins	305 - 14th Ave. S. Suite 3B	19	Yes	Male
2	Karl	Jablonski	14th Suite 3B	20	Yes	Male
3	Matti	Karttunen	Keskuskatu 45Helsinki	20	Yes	Female
4	Tom	B. Erichsen	Skagen 21 Stavanger	20	Yes	Male
5	Hekkan	Burger	Gateveien 15Sandnes	19	No	Male
6	Julia	Yasminr	13 Hibiscus Road	19	Yes	Female
7	Nicholas	Wong	12 Metro Street	20	No	Male
8	Niko	Bellic	53 Hove Beach	20	No	Male
9	David	Kim	4 Lake Garden	20	Yes	Male
10	Carl	Johnson	77 Park Lane	19	Yes	Male
11	John	Kay	22 Lin Seng Park	19	No	Male
NULL	NULL	NULL	NULL	NULL	NULL	NULL

AmountPaid	AmountBalance	StudentRollNo	AdmissionDate	DueDate
1000	9000	1	2008-04-03	2008-05-01
3000	7000	2	2008-04-05	2008-05-01
5000	5000	3	2008-04-08	2008-05-01
7000	3000	4	2008-04-01	2008-05-01
6000	4000	5	2008-04-02	2008-05-01
4000	6000	6	2008-04-12	2008-05-01
1000	9000	7	2008-04-14	2008-05-01
1500	8500	8	2008-04-09	2008-05-01
2500	7500	9	2008-04-08	2008-05-01
9500	500	10	2008-04-03	2008-05-01
5000	5000	11	2008-04-07	2008-05-01

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

```
select S.StudentName, S.StudentSurname, S.StudentRollNo
from StudentBasicInformation as S inner join ScholarshipStudent on
S.StudentRollNo = ScholarshipStudent.StudentRollNo
inner join SubjectScholarshipInformation on
SubjectScholarshipInformation.ScholarshipName = ScholarshipStudent.ScholarshipName
where SubjectScholarshipInformation.scholarshipAmount > 50;
```

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

```
select S.StudentName, S.StudentSurname, S.StudentRollNo
from StudentBasicInformation as S left outer join ScholarshipStudent on
S.StudentRollNo = ScholarshipStudent.StudentRollNo
where ScholarshipStudent.ScholarshipName = NULL and S.ScholarshipOpted = 'Yes';
```

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

```
DELIMITER //
CREATE PROCEDURE updatePercentage (IN rollno int)
BEGIN
```

```
        Update StudentSubjectInformation
   set StudentMarksPercentage = (SubjectObtainedMarks*100) / SubjectTotalMarks
   where StudentRollNo = rollno;
END//
DELIMITER ;

CALL updatePercentage(1);
```

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.

```
CREATE VIEW subjectbasic AS
SELECT *
FROM StudentBasicInformation natural join StudentSubjectInformation;

Delimiter //
CREATE PROCEDURE updateScholarshipCategory(IN rollno int)
BEGIN
update ScholarshipStudent
set StudentRollNo = rollno and ScholarshipName =
Case when rollno IN (SELECT StudentRollNo from subjectbasic WHERE
subjectbasic.StudentMarksPercentage between 71 and 80) THEN "I matter"

when rollno IN (SELECT StudentRollNo from subjectbasic WHERE
subjectbasic.StudentMarksPercentage between 81 and 90) THEN "Little Heart"

when rollno IN (SELECT StudentRollNo from subjectbasic WHERE
subjectbasic.StudentMarksPercentage between 91 and 100) THEN "Inspire Our Future"

else null
end;
END//
```

```
Delimiter ;
Call updateScholarshipCategory(1);
```

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

```
CREATE VIEW balanceAmount AS
SELECT StudentName, StudentSurname, AmountBalance
FROM StudentBasicInformation join StudentAdmissionPaymentDetails
where StudentBasicInformation.StudentRollNo = StudentAdmissionPaymentDetails.StudentRollNo;


SELECT *
FROM balanceAmount;
```

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

```
select S.StudentRollNo
from StudentBasicInformation as S left outer join ScholarshipStudent on
S.StudentRollNo = ScholarshipStudent.StudentRollNo
where ScholarshipStudent.ScholarshipName is NULL
```

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

```
DELIMITER //
CREATE PROCEDURE procedureBalanceAmount (IN rollno int, OUT balance INT)
BEGIN
        SELECT AmountBalance
   FROM StudentAdmissionPaymentDetails
   WHERE StudentRollNo = rollno;
```

```
END//
DELIMITER ;

CALL procedurebalanceAmount(1, @balance);
```

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

```
SELECT StudentRollNo, StudentName
From StudentBasicInformation
WHERE StudentRollNo IN
(
SELECT StudentRollNo
FROM StudentSubjectInformation
ORDER BY SubjectObtainedMarks DESC
LIMIT 5);
```

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)

LEFT OUTER JOIN – Retrive the details of student who haven't got any scholarship

```
select S.StudentRollNo
from StudentBasicInformation as S left outer join ScholarshipStudent on
S.StudentRollNo = ScholarshipStudent.StudentRollNo
where ScholarshipStudent.ScholarshipName is NULL
```

Here I have used left outer join because this way, I get all the records from the left table and added information from the right table for only those records that have that extra information.

INNER JOIN – Display the scholarship details along with the number of students availing each of the scholarship types.

```
SELECT S1.ScholarshipName, count(S1.ScholarshipName), S2.ScholarshipCategory
 FROM ScholarshipStudent as S1, SubjectScholarshipInformation as S2
 WHERE S1.ScholarshipName = S2.ScholarshipName
 GROUP BY S1.ScholarshipName
 ORDER BY count(S1.ScholarshipName) DESC;
```

Here I have used inner join because I want only the records that are present in both the tables.

RIGHT OUTER JOIN - Retrive the details of student who haven't got any scholarship

```
select S.StudentRollNo
from left outer join ScholarshipStudent, StudentBasicInformation as S on
S.StudentRollNo = ScholarshipStudent.StudentRollNo
where ScholarshipStudent.ScholarshipName is NULL
```

Here I have used right outer join because this way, I get all the records from the right table and added information from the left table for only those records that have that extra information.

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

DELETE	DROP	TRUNCATE
The DELETE statement in SQL is a Data Manipulation Language (DML) Command.	DROP statement is a Data Definition Language (DDL) Command	TRUNCATE command is a Data Definition Language (DDL)operation.
It is use to delete the one or more tuples of a table.	It is use to drop the whole table.	It is use to delete all the rows of a relation (table) in one go but not the table itself.
If used with WHERE clause, selected rows are deleted.	WHERE clause cannot be used.	WHERE clause cannot be used.
The structure or schema of the table is preserved.	The structure or schema of the table is not preserved.	The structure or schema of the table is preserved.
DELETE FROM Employees WHERE Emp_Id = 7;	DROP TABLE Employees;	TRUNCATE TABLE Employees;

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

```
SELECT ScholarshipName, count(*)
FROM ScholarshipStudent
GROUP BY ScholarshipName;
```

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

```
SELECT S1.ScholarshipName, count(S1.ScholarshipName), S2.ScholarshipCategory
 FROM ScholarshipStudent as S1, SubjectScholarshipInformation as S2
 WHERE S1.ScholarshipName = S2.ScholarshipName
 GROUP BY S1.ScholarshipName
 ORDER BY count(S1.ScholarshipName) DESC;
```

Q19. 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 *
FROM StudentBasicInformation left JOIN ScholarshipStudent ON
StudentBasicInformation.StudentRollNo = ScholarshipStudent.StudentRollNo
left JOIN SubjectScholarshipInformation ON ScholarshipStudent.ScholarshipName =
SubjectScholarshipInformation.ScholarshipName
WHERE StudentBasicInformation.StudentRollNo IN(
SELECT StudentRollNo
FROM StudentSubjectInformation
WHERE StudentMarksPercentage =
(SELECT max(StudentMarksPercentage)
FROM StudentSubjectInformation));
```

Q20. Difference between the Triggers, Stored Procedures, Views and Functions

TRIGGERS	STORED PROCEDURE	VIEWS	FUNCTIONS
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	A view is a virtual table based on the result-set of an SQL statement.	Functions are routines that accept parameters, perform an action, such as a complex calculation, and return the result of that action as a value.
Triggers cannot return values	Stored procedures can return values	It does not return a value but a table.	The return value can either be a single scalar value or a result set.