**\*\*Simple SQL Statements:\*\***

1. List the basic information of all students.

```sql

**SELECT \* FROM S;**

```

2. List sno, sname, birthday of all male students.

```sql

**SELECT sno, sname, birthday**

**FROM S**

**WHERE sex = 'M';**

```

3. Add a column named ‘addr’ in table S, then change its length from 20 to 25.

```sql

**ALTER TABLE S**

**ADD COLUMN addr VARCHAR(25);**

```

4. Add a column named ‘registerDate’ in table S, set a default value for column registerDate as the current system date, then drop it.

```sql

**ALTER TABLE S**

**ADD COLUMN registerDate DATE DEFAULT CURRENT\_DATE;**

**ALTER TABLE S**

**DROP COLUMN registerDate;**

```

5. Set a default value 18 for column age in table S.

```sql

**ALTER TABLE S**

**ALTER COLUMN age SET DEFAULT 18;**

```

6. Set column sname as alternate key (not primary key) in table S.

```sql

**ALTER TABLE S**

**ADD CONSTRAINT UQ\_sname UNIQUE (sname);**

```

7. Create a unique index on ascending sno and descending cno in Table SC.

```sql

**CREATE UNIQUE INDEX idx\_unique\_sc ON SC (sno ASC, cno DESC);**

```

8. Add a constraint in table S: male students’ age should be younger than 23 and female students’ age should be younger than 21.

**```sql**

**ALTER TABLE S**

**ADD CONSTRAINT CHK\_age\_gender CHECK (**

**(sex = 'M' AND age < 23) OR**

**(sex = 'F' AND age < 21)**

**);**

```

9. Create a view named View\_80 which stores the rows with scores more than 80 using ‘with check option’, list sno, cno, and grade.

**```sql**

**CREATE VIEW View\_80 AS**

**SELECT sno, cno, grade**

**FROM SC**

**WHERE grade > 80**

**WITH CHECK OPTION;**

```

10. List the tuples with scores more than 90.

**```sql**

**SELECT sno, cno, grade**

**FROM SC**

**WHERE grade > 90;**

```

11. Insert the following rows into View\_80.

**```sql**

**INSERT INTO View\_80 (sno, cno, grade) VALUES ('08300010', '801', 87), ('08300010', '804',** 76);

```

12. In view View\_80, remove the following tuple: sno = 08300010, cno = 801.

**```sql**

**DELETE FROM View\_80 WHERE sno = '08300010' AND cno = '801';**

```

13. In view View\_80, update the following tuples:

Change the grade to 90 of tuple (08300010, 803).

Change the grade to 70 of tuple (08300010, 803).

**```sql**

**UPDATE View\_80 SET grade = 90 WHERE sno = '08300010' AND cno = '803';**

**UPDATE View\_80 SET grade = 70 WHERE sno = '08300010' AND cno = '803';**

```

14. List sno, sname, sex and birthday of female students who were born before 1980-01-01.

**```sql**

**SELECT sno, sname, sex, birthday**

**FROM S**

**WHERE sex = 'F' AND birthday < '1980-01-01';**

```

15. List sno, sname, sex and birthday of male students whose names contain 's1'.

**```sql**

**SELECT sno, sname, sex, birthday**

**FROM S**

**WHERE sex = 'M' AND sname LIKE '%s1%';**

```

16. Search for the teachers who give lectures in English, displaying tno, name and the number of courses that he/she teaches in English.

**```sql**

**SELECT T.tno, T.name, COUNT(Teaching.cno) AS num\_courses**

**FROM Teacher T**

**JOIN Teaching ON T.tno = Teaching.tno**

**JOIN Course C ON Teaching.cno = C.cno**

**WHERE C.language = 'English'**

**GROUP BY T.tno, T.name;**

```

17. List tno, name and title of teachers whose title is not lecturer.

**```sql**

**SELECT tno, name, title**

**FROM Teacher**

**WHERE title <> 'lecturer';**

```

18. List sno of students who took course(s) and did not take the exam(s).

**```sql**

**SELECT DISTINCT sno**

**FROM SC**

**WHERE grade IS NULL;**

```

19. List sno, grade of students who fail the exam, displaying the tuples in descending order.

**```sql**

**SELECT sno, grade**

**FROM SC**

**WHERE grade < 60**

**ORDER BY grade DESC;**

```

20. List tno, tname, birthday of teachers who were born in 1970.

**```sql**

**SELECT tno, tname, birthday**

**FROM Teacher**

**WHERE YEAR(birthday) = 1970;**

**```**

21. List the number of students studying each course.

**```sql**

**SELECT cno, COUNT(sno) AS num\_students**

**FROM SC**

**GROUP BY cno;**

```

22. List the teacher numbers of teachers who teach more than 2 courses, showing the number of courses at the same time.

**```sql**

**SELECT tno, COUNT(cno) AS num\_courses**

**FROM Teaching**

**GROUP BY tno**

**HAVING COUNT(cno) > 2;**

```

23. List the average score, minimum score and maximum score of course 801.

**```sql**

**SELECT AVG(grade) AS avg\_score, MIN(grade) AS min\_score, MAX(grade) AS max\_score**

**FROM SC**

**WHERE cno = '801';**

```

24. List the names, birthday of teachers whose titles are lecturers and who were born after 1960.

`**``sql**

**SELECT name, birthday**

**FROM Teacher**

**WHERE title = 'lecturer' AND YEAR(birthday) > 1960;**

```

25. Calculate the total credits of each student based on Table SC and insert those total credits into Column ‘totalCredits’ of Table S.

**```sql**

**UPDATE S**

**SET totalCredits = (**

**SELECT SUM(credit)**

**FROM SC**

**JOIN Course ON SC.cno = Course.cno**

**WHERE SC.sno = S.sno**

**);**

```

26. Calculate the number of students in each class based on Table S and insert those numbers into Column ‘studentnumber’ of Table class.

**```sql**

**UPDATE class**

**SET studentnumber = (**

**SELECT COUNT(\*)**

**FROM S**

**WHERE S.classno = class.classno**

**);**

**```**

**\*\*3. Complex SQL Statements:\*\***

1. Create a view named new\_View, showing sno, sname, classno, cname and grade.

**```sql**

**CREATE VIEW new\_View AS**

**SELECT S.sno, S.sname, S.classno, C.cname, SC.grade**

**FROM S**

**JOIN SC ON S.sno = SC.sno**

**JOIN Course C ON SC.cno = C.cno;**

**```**

2. In new\_View, list sno, sname, cname and grade of the students in Class named ‘Software0801’.

**```sql**

**SELECT sno, sname, cname, grade**

**FROM new\_View**

**WHERE classno = 'Software0801';**

**```**

3. In new\_View, insert the following value (08300168，s21，Rj0803，Database System，88)

**```sql**

**INSERT INTO new\_View (sno, sname, classno, cname, grade)**

**VALUES ('08300168', 's21', 'Rj0803', 'Database System', 88);**

**```**

4. List each student’s total credits, showing sno, sname and total credits.

**```sql**

**SELECT S.sno, S.sname, SUM(C.credit) AS total\_credits**

**FROM S**

**LEFT JOIN SC ON S.sno = SC.sno**

**LEFT JOIN Course C ON SC.cno = C.cno**

**GROUP BY S.sno, S.sname;**

**```**

5. List each student’s average score and the number of elective courses, showing sno, sname, average score and the number of elective courses.

**```sql**

**SELECT S.sno, S.sname, AVG(SC.grade) AS avg\_score, COUNT(CASE WHEN C.type = 'Elective' THEN 1 END) AS num\_elective\_courses**

**FROM S**

**LEFT JOIN SC ON S.sno = SC.sno**

**LEFT JOIN Course C ON SC.cno = C.cno**

**GROUP BY S.sno, S.sname;**

**```**

6. Search for the students who took course(s) and did not take the exam(s) showing sno, sname, cno and cname.

**```sql**

**SELECT S.sno, S.sname, SC.cno, C.cname**

**FROM S**

**JOIN SC ON S.sno = SC.sno**

**JOIN Course C ON SC.cno = C.cno**

**WHERE SC.grade IS NULL;**

**```**

7. Search for the students who took course(s) and did not pass the exam(s) showing sno, sname, cno , cname and grade.

**```sql**

**SELECT S.sno, S.sname, SC.cno, C.cname, SC.grade**

**FROM S**

**JOIN SC ON S.sno = SC.sno**

**JOIN Course C ON SC.cno = C.cno**

**WHERE SC.grade < 60;**

**```**

8. Search for the students who take the course named “Programming Language”, showing sname and grade.

**```sql**

**SELECT S.sname, SC.grade**

**FROM S**

**JOIN SC ON S.sno = SC.sno**

**JOIN Course C ON SC.cno = C.cno**

**WHERE C.cname = 'Programming Language';**

**```**

9. Search for the students in the class named “Software0801”, showing sno, sname, cno, cname and grade.

**```sql**

**SELECT S.sno, S.sname, SC.cno, C.cname, SC.grade**

**FROM S**

**JOIN SC ON S.sno = SC.sno**

**JOIN Course C ON SC.cno = C.cno**

**WHERE S.classno = 'Software0801';**

**```**

10. List the teaching information of all teachers, showing tname and cname.

**```sql**

**SELECT T.tname, C.cname**

**FROM Teacher T**

**JOIN Teaching ON T.tno = Teaching.tno**

**JOIN Course C ON Teaching.cno = C.cno;**

**```**

11. List the information that the score is less than the average score of the same course, showing sno, sname, cname and by how much his score is below the highest score of the same course.

**```sql**

**SELECT S.sno, S.sname, C.cname, (MAX(SC.grade) - SC.grade) AS score\_difference**

**FROM S**

**JOIN SC ON S.sno = SC.sno**

**JOIN Course C ON SC.cno = C.cno**

**GROUP BY S.sno, S.sname, C.cname**

**HAVING SC.grade < AVG(SC.grade);**

**```**

12. Search for the students who are in the same class with a student named ‘s1’.

**```sql**

**SELECT S.sno, S.sname, S.classno**

**FROM S**

**WHERE classno IN (SELECT classno FROM S WHERE sname = 's1');**

**```**

13. Search for the students who do

not take “Programming Language”, showing the student names.

**```sql**

**SELECT sname**

**FROM S**

**WHERE sno NOT IN (SELECT sno FROM SC WHERE cno = 'Programming Language');**

**```**

14. Search for the teachers who teach both Data Structure and Database System, showing their names.

**```sql**

**SELECT T.tname**

**FROM Teacher T**

**JOIN Teaching ON T.tno = Teaching.tno**

**JOIN Course C ON Teaching.cno = C.cno**

**WHERE C.cname IN ('Data Structure', 'Database System')**

**GROUP BY T.tname**

**HAVING COUNT(DISTINCT C.cname) = 2;**

**```**

15. Search for the teachers who teach all courses, showing their names.

**```sql**

**SELECT T.tname**

**FROM Teacher T**

**JOIN Teaching ON T.tno = Teaching.tno**

**JOIN Course C ON Teaching.cno = C.cno**

**GROUP BY T.tname**

**HAVING COUNT(DISTINCT C.cname) = (SELECT COUNT(\*) FROM Course);**

**```**

16. Find all female students who take both 801 and 802 as elective courses, list their names.

**```sql**

**SELECT sname**

**FROM S**

**WHERE sno IN (**

**SELECT sno FROM SC WHERE cno = '801'**

**INTERSECT**

**SELECT sno FROM SC WHERE cno = '802'**

**) AND sex = 'female';**

**```**

17. Find all students who take neither 801 nor 802 as an elective course, list their student names.

**```sql**

**SELECT sname**

**FROM S**

**WHERE sno NOT IN (**

**SELECT sno FROM SC WHERE cno IN ('801', '802')**

**);**

**```**

18. Search for the female students who got a score of 95.

**```sql**

**SELECT S.sname**

**FROM S**

**JOIN SC ON S.sno = SC.sno**

**WHERE SC.grade = 95 AND S.sex = 'female';**

**```**

19. Search for the female students who take more than 3 courses.

**```sql**

**SELECT S.sname**

**FROM S**

**JOIN SC ON S.sno = SC.sno**

**WHERE S.sex = 'female'**

**GROUP BY S.sno**

**HAVING COUNT(\*) > 3;**

**```**

20. Find the male students whose average grades are more than 80, list their student numbers and student names.

**```sql**

**SELECT S.sno, S.sname**

**FROM S**

**JOIN SC ON S.sno = SC.sno**

**WHERE S.sex = 'male'**

**GROUP BY S.sno, S.sname**

**HAVING AVG(SC.grade) > 80;**

**```**

21. Search for the average score of each course which is taught by a teacher named ‘t1’.

**```sql**

**SELECT C.cno, C.cname, AVG(SC.grade) AS avg\_score**

**FROM Course C**

**JOIN Teaching T ON C.cno = T.cno**

**JOIN SC ON C.cno = SC.cno**

**JOIN Teacher T2 ON T.tno = T2.tno**

**WHERE T2.tname = 't1'**

**GROUP BY C.cno, C.cname;**

**```**

22. Search for the number of male students of each age, list the age with the number of students more than 20, in descending order of the number of students.

**```sql**

**SELECT FLOOR((YEAR(CURRENT\_DATE) - YEAR(birthday)) / 10) \* 10 AS age\_group, COUNT(\*) AS num\_students**

**FROM S**

**WHERE sex = 'male'**

**GROUP BY age\_group**

**HAVING num\_students > 20**

**ORDER BY num\_students DESC;**

**```**

23. Search for the students whose each grade is more than 90, showing their names.

**```sql**

**SELECT sname**

**FROM S**

**WHERE sno IN (**

**SELECT sno**

**FROM SC**

**GROUP BY sno**

**HAVING MIN(grade) > 90**

**);**

**```**

24. Search for the male students who are older than all female students, showing their names.

**```sql**

**SELECT male.sname**

**FROM S AS male**

**WHERE sex = 'male'**

**AND birthday < ALL (**

**SELECT female.birthday**

**FROM S AS female**

**WHERE sex = 'female'**

**);**

**```**

25. Search for the female students who do not take course 802, showing the student names.

**```sql**

**SELECT sname**

**FROM S**

**WHERE sex = 'female'**

**AND sno NOT IN (**

**SELECT sno FROM SC WHERE cno = '802'**

**);**

**```**

26. Search for the students who pass the exams of all courses which he takes, showing their names.

**```sql**

**SELECT sname**

**FROM S**

**WHERE sno IN (**

**SELECT sno**

**FROM SC**

**GROUP BY sno**

**HAVING COUNT(\*) = COUNT(CASE WHEN grade IS NOT NULL THEN 1 END)**

**);**

**```**

27. Find the students who take all the courses and list their names.

**```sql**

**SELECT sname**

**FROM S**

**WHERE sno IN (**

**SELECT sno**

**FROM SC**

**GROUP BY sno**

**HAVING COUNT(\*) = (SELECT COUNT(\*) FROM Course)**

**);**

**```**

28. Search for the students who take all courses that are taken by sno='08300010', showing their names.

**```sql**

**SELECT sname**

**FROM S**

**WHERE sno IN (**

**SELECT sno**

**FROM SC**

**WHERE cno IN (**

**SELECT cno**

**FROM SC**

**WHERE sno = '08300010'**

**)**

**GROUP BY sno**

**HAVING COUNT(DISTINCT cno) = (**

**SELECT COUNT(DISTINCT cno)**

**FROM SC**

**WHERE sno = '08300010'**

**)**

**);**

**```**

29. Search for the student whose average score is the highest.

**```sql**

**SELECT sname**

**FROM S**

**GROUP BY sname**

**HAVING AVG(**

**SELECT AVG(grade)**

**FROM SC**

**WHERE sno = S.sno**

**) = (**

**SELECT MAX(avg\_grade)**

**FROM (**

**SELECT AVG(grade) AS avg\_grade**

**FROM SC**

**GROUP BY sno**

**) AS avg\_grades**

**);**

**```**

30. Search for the students whose average score are higher than the average score of his or her class.

**```sql**

**SELECT sname**

**FROM S**

**WHERE AVG(**

**SELECT AVG(grade)**

**FROM SC**

**WHERE sno = S.sno**

**) > (**

**SELECT AVG(grade)**

**FROM SC**

**JOIN S ON SC.sno = S.sno**

**WHERE S.classno = S.classno**

**)**

**GROUP BY sname;**

**```**

**4. DML**

1. Increase the grade by 5% for the female students who take course(s) taught by a teacher named ‘t1’.

**```sql**

**UPDATE SC**

**SET grade = grade \* 1.05**

**WHERE sno IN (**

**SELECT SC.sno**

**FROM SC**

**JOIN Teaching ON SC.cno = Teaching.cno**

**JOIN Teacher ON Teaching.tno = Teacher.tno**

**WHERE Teacher.tname = 't1'**

**AND SC.sno IN (**

**SELECT sno**

**FROM S**

**WHERE sex = 'female'**

**)**

**);**

**```**

2. Find the students whose every grade is no less than 80. Insert the student numbers, names, and sex into an existing table named STUD(sno, sname, sex).

**```sql**

**INSERT INTO STUD (sno, sname, sex)**

**SELECT sno, sname, sex**

**FROM S**

**WHERE sno NOT IN (**

**SELECT sno**

**FROM SC**

**WHERE grade < 80**

**);**

**```**

3. In table SC, remove the records in which there is no grade.

**```sql**

**DELETE FROM SC**

**WHERE grade IS NULL;**

**```**

4. Remove the score records about a student named ‘s1’.

**```sql**

**DELETE FROM SC**

**WHERE sno = 's1';**

**```**

5. Set the grade to null, which is less than 60 for course ‘Data Structure’.

**```sql**

**UPDATE SC**

**SET grade = NULL**

**WHERE cno = 'Data Structure' AND grade < 60;**

**```**

6. Increase the grade by 5% for the female students whose scores are less than the total average score.

**```sql**

**UPDATE SC**

**SET grade = grade \* 1.05**

**WHERE sno IN (**

**SELECT sno**

**FROM S**

**WHERE sex = 'female'**

**) AND grade < (**

**SELECT AVG(grade)**

**FROM SC**

**);**

**```**

7. For course ‘804’ in table SC, increase the grade by 5% if grade<=75, increase the grade by 4% if grade between 76 and 95, using two update statements.

**```sql**

**UPDATE SC**

**SET grade = CASE**

**WHEN grade <= 75 THEN grade \* 1.05**

**WHEN grade BETWEEN 76 AND 95 THEN grade \* 1.04**

**ELSE grade**

**END**

**WHERE cno = '804';**

**\*\*Exercise 2\*\***

**5. \*\*Stored Procedures\*\***

- \*\*(1) Create a Stored Procedure named ‘insert\_s’, it can insert a record into Table s, including 5 parameters: sno, sname, sex, birthday and classno. And give the SQL statement used to call ‘insert\_s’ with parameter values **```sql**

**DELIMITER //**

**CREATE PROCEDURE insert\_s (IN sno VARCHAR(8), IN sname VARCHAR(20), IN ssex VARCHAR(2), IN sbirthday DATE, IN classno VARCHAR(6))**

**BEGIN**

**INSERT INTO s (sno, sname, ssex, sbirthday, classno) VALUES (sno, sname, ssex, sbirthday, classno);**

**END //**

**DELIMITER ;**

**```**

\*\*Call Statement:\*\*

**```sql**

**CALL insert\_s('08300012', 'John Doe', 'M', '1990-05-15', 'Rj0801');**

**```**

- \*\*(2) Create a Stored Procedure named ‘insert\_course’, it can insert a new course into Table course, including 3 parameters: cno, cname and credit, setting the default value 4 for credit. And give two cases of the SQL statement used to call ‘insert\_course’ with parameter values, one is providing the value of cno and cname, the other is providing cno, cname and credit. Please compare the results of two cases.

**```sql**

**DELIMITER //**

**CREATE PROCEDURE insert\_course (IN cno VARCHAR(6), IN cname VARCHAR(30), IN credit SMALLINT DEFAULT 4)**

**BEGIN**

**INSERT INTO course (cno, cname, ccredit) VALUES (cno, cname, credit);**

**END //**

**DELIMITER ;**

**```**

\*\*Call Statements:\*\*

Providing cno and cname:

**```sql**

**CALL insert\_course('801', 'Mathematics');**

**```**

Providing cno, cname, and credit:

**```sql**

**CALL insert\_course('802', 'English', 3);**

**```**

3 - Create a Stored Procedure named ‘query\_student’, it can search for a student information of sname, sex, birthday and classno according to a given sno. And give the SQL statement used to call ‘query\_student’ using sno “08300012”.

**```sql**

**DELIMITER //**

**CREATE PROCEDURE query\_student (IN sno\_param VARCHAR(8))**

**BEGIN**

**SELECT sname, ssex, sbirthday, classno FROM s WHERE sno = sno\_param;**

**END //**

**DELIMITER ;**

**```**

\*\*Call Statement:\*\*

**```sql**

**CALL query\_student('08300012');**

**```**

4 - Create a Stored Procedure named ‘select\_average’, it can output sname, sex and average score of all students whose average scores are more than 80. Please give the SQL statement used to call ‘select\_average’.

**```sql**

**DELIMITER //**

**CREATE PROCEDURE select\_average ()**

**BEGIN**

**SELECT sname, ssex, AVG(grade) AS average\_score**

**FROM sc**

**GROUP BY sno**

**HAVING AVG(grade) > 80;**

**END //**

**DELIMITER ;**

**```**

\*\*Call Statement:\*\*

**```sql**

**CALL select\_average();**

**```**

5 - Create a Stored Procedure named ‘select\_all’, it can display sno, sname, the number of electives, average score and total credits of the students who take all the courses taken by a given student with parameter: student name. Please give the SQL statement used to call ‘select\_all’ with a given student name.

**```sql**

**DELIMITER //**

**CREATE PROCEDURE select\_all (IN student\_name VARCHAR(20))**

**BEGIN**

**SELECT s.sno, s.sname, COUNT(sc.cno) AS number\_of\_electives, AVG(sc.grade) AS average\_score, s.Totalcredit**

**FROM s**

**LEFT JOIN sc ON s.sno = sc.sno**

**WHERE s.sname = student\_name**

**GROUP BY s.sno;**

**END //**

**DELIMITER ;**

**```**

\*\*Call Statement:\*\*

**```sql**

**CALL select\_all('John Doe');**

**```**

6 - Create a Stored Procedure named ‘select\_less’, it can display sno, sname and average score of the students who average score are less than the average score of a given class with parameter: classno. Please give the SQL statement used to call ‘select\_less’ with a given classno 'Rj0801'.

**```sql**

**DELIMITER //**

**CREATE PROCEDURE select\_less (IN class\_no VARCHAR(6))**

**BEGIN**

**SELECT sno, sname, AVG(grade) AS average\_score**

**FROM sc**

**WHERE sno IN (SELECT sno FROM s WHERE classno = class\_no)**

**GROUP BY sno**

**HAVING AVG(grade) < (SELECT AVG(grade) FROM sc WHERE sno IN (SELECT sno FROM s WHERE classno = class\_no));**

**END //**

**DELIMITER ;**

**```**

\*\*Call Statement:\*\*

**```sql**

**CALL select\_less('Rj0801');**

**```**

**6. \*\*Triggers\*\***

1. - Create a Trigger named ‘display\_trigger’ on Table s, it can display all the students information when inserting a new student into Table s.

**```sql**

**DELIMITER //**

**CREATE TRIGGER display\_trigger**

**AFTER INSERT ON s**

**FOR EACH ROW**

**BEGIN**

**SELECT \* FROM s WHERE sno = NEW.sno;**

**END //**

**DELIMITER ;**

**```**

1. Create a Trigger named ‘increase\_num’ on Table s, it can increase by 1 the number of students in the class which he or she belongs to when inserting a new student into Table s.

**```sql**

**DELIMITER //**

**CREATE TRIGGER increase\_num**

**AFTER INSERT ON s**

**FOR EACH ROW**

**BEGIN**

**UPDATE class SET studentnumber = studentnumber + 1 WHERE classno = NEW.classno;**

**END //**

**DELIMITER ;**

**```**

1. Create a Trigger named ‘decrease\_num’ on Table s, it can decrease by 1 the number of students in the class which he or she belongs to when deleting a new student into Table s.
2. **```sql**

**DELIMITER //**

**CREATE TRIGGER decrease\_num**

**AFTER DELETE ON s**

**FOR EACH ROW**

**BEGIN**

**UPDATE class SET studentnumber = studentnumber - 1 WHERE classno = OLD.classno;**

**END //**

**DELIMITER ;**

**```**

4 - \*\* Create a trigger named ‘transfer\_num’ on Table s. It can automatically update Table class when a student changes from one class to another in Table s, decreasing the total number of students in the original class by 1 and increasing the total number of students in the new class by 1.

**```sql**

**DELIMITER //**

**CREATE TRIGGER transfer\_num**

**AFTER UPDATE ON s**

**FOR EACH ROW**

**BEGIN**

**DECLARE old\_class**

**\_number INT;**

**DECLARE new\_class\_number INT;**

**SELECT studentnumber INTO old\_class\_number FROM class WHERE classno = OLD.classno;**

**SELECT studentnumber INTO new\_class\_number FROM class WHERE classno = NEW.classno;**

**UPDATE class SET studentnumber = old\_class\_number - 1 WHERE classno = OLD.classno;**

**UPDATE class SET studentnumber = new\_class\_number + 1 WHERE classno = NEW.classno;**

**END //**

**DELIMITER ;**

**```**

5 - \*\* Create a trigger named insert\_sc, when inserting a non-existing sno value in Table s into Table sc, it can insert a new row with sno equal to the new value and set null value to other attributes except sno in Table s.

**```sql**

**DELIMITER //**

**CREATE TRIGGER insert\_sc**

**AFTER INSERT ON sc**

**FOR EACH ROW**

**BEGIN**

**IF NOT EXISTS (SELECT 1 FROM s WHERE sno = NEW.sno) THEN**

**INSERT INTO s (sno) VALUES (NEW.sno);**

**END IF;**

**END //**

**DELIMITER ;**

**```**

6 Create a trigger to stop the attempt to lower the grade of a student.

**```sql**

**DELIMITER //**

**CREATE TRIGGER prevent\_lowering\_grade**

**BEFORE UPDATE ON sc**

**FOR EACH ROW**

**BEGIN**

**IF NEW.grade < OLD.grade THEN**

**SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Cannot lower the grade';**

**END IF;**

**END //**

**DELIMITER ;**

**```**

7 - Create a Trigger named ‘tri\_nottoomuch’ on Table sc, it can give warning 'too much courses' , 'The operation is rejected' and reject the operation when the number of electives taken by a student exceeds 3.

**```sql**

**DELIMITER //**

**CREATE TRIGGER tri\_nottoomuch**

**BEFORE INSERT ON sc**

**FOR EACH ROW**

**BEGIN**

**DECLARE elective\_count INT;**

**SELECT COUNT(\*) INTO elective\_count FROM sc WHERE sno = NEW.sno;**

**IF elective\_count >= 3 THEN**

**SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Too much courses';**

**END IF;**

**END //**

**DELIMITER ;**

**```**

8 - \*\* Create twos Triggers named ‘**insert\_sc** ‘ and ‘**delete\_sc**’ on Table sc, ‘**insert\_sc’** can increase the total credits in Table s when a student takes a new course, and ‘**delete\_sc**’ can decrease the total credits in Table s when a student quit from a course.

**```sql**

**DELIMITER //**

**CREATE TRIGGER insert\_sc**

**AFTER INSERT ON sc**

**FOR EACH ROW**

**BEGIN**

**UPDATE s SET Totalcredit = Totalcredit + (SELECT ccredit FROM course WHERE cno = NEW.cno) WHERE sno = NEW.sno;**

**END //**

**DELIMITER ;**

**CREATE TRIGGER delete\_sc**

**AFTER DELETE ON sc**

**FOR EACH ROW**

**BEGIN**

**UPDATE s SET Totalcredit = Totalcredit - (SELECT ccredit FROM course WHERE cno = OLD.cno) WHERE sno = OLD.sno;**

**END //**

**DELIMITER ;**

**```**