DOCUMENTATION

DBS Project
Addition Substitution Window

Team Members
Rishabh Barnwal 2020A7PS1677P
Rachit Agrawal 2020A7PS0033P

System Requirements

Operating System:

1. Microsoft Windows Server 2012 R2 Standard.
2. Oracle Enterprise Linux 6.5 or higher.

Database:

1. Oracle Text.
2. Oracle JVM.
3. Oracle XML DB.

Memory:

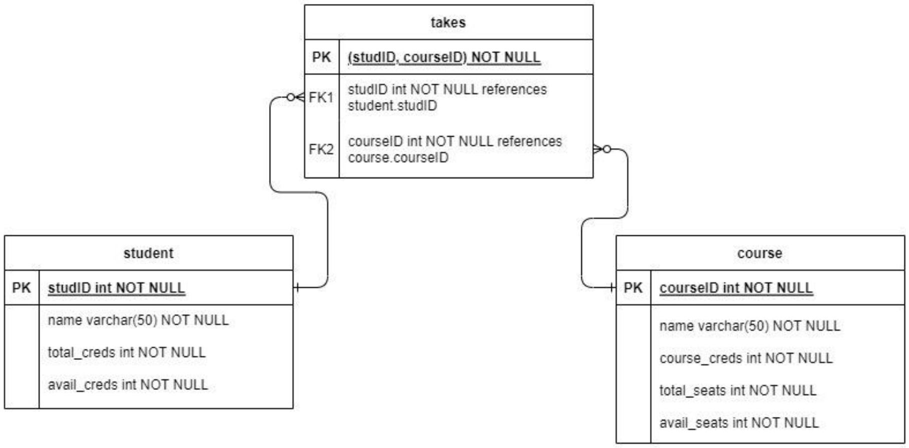
1. 8GB RAM or more.

Disc Space:

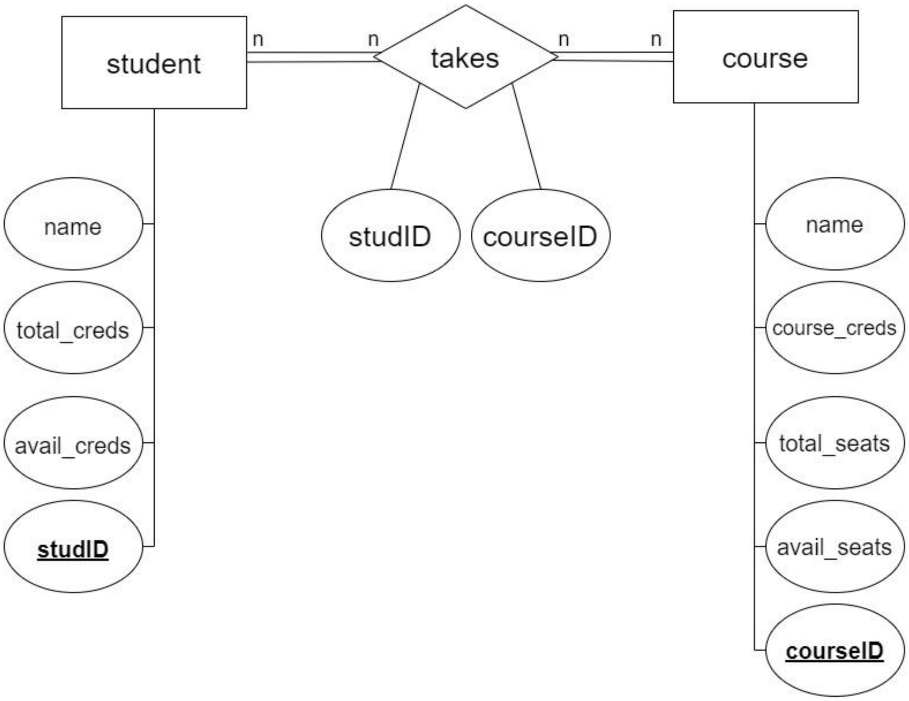
1. 100MB for SQL server and additional 200KB for Database (schema).

ER Diagram

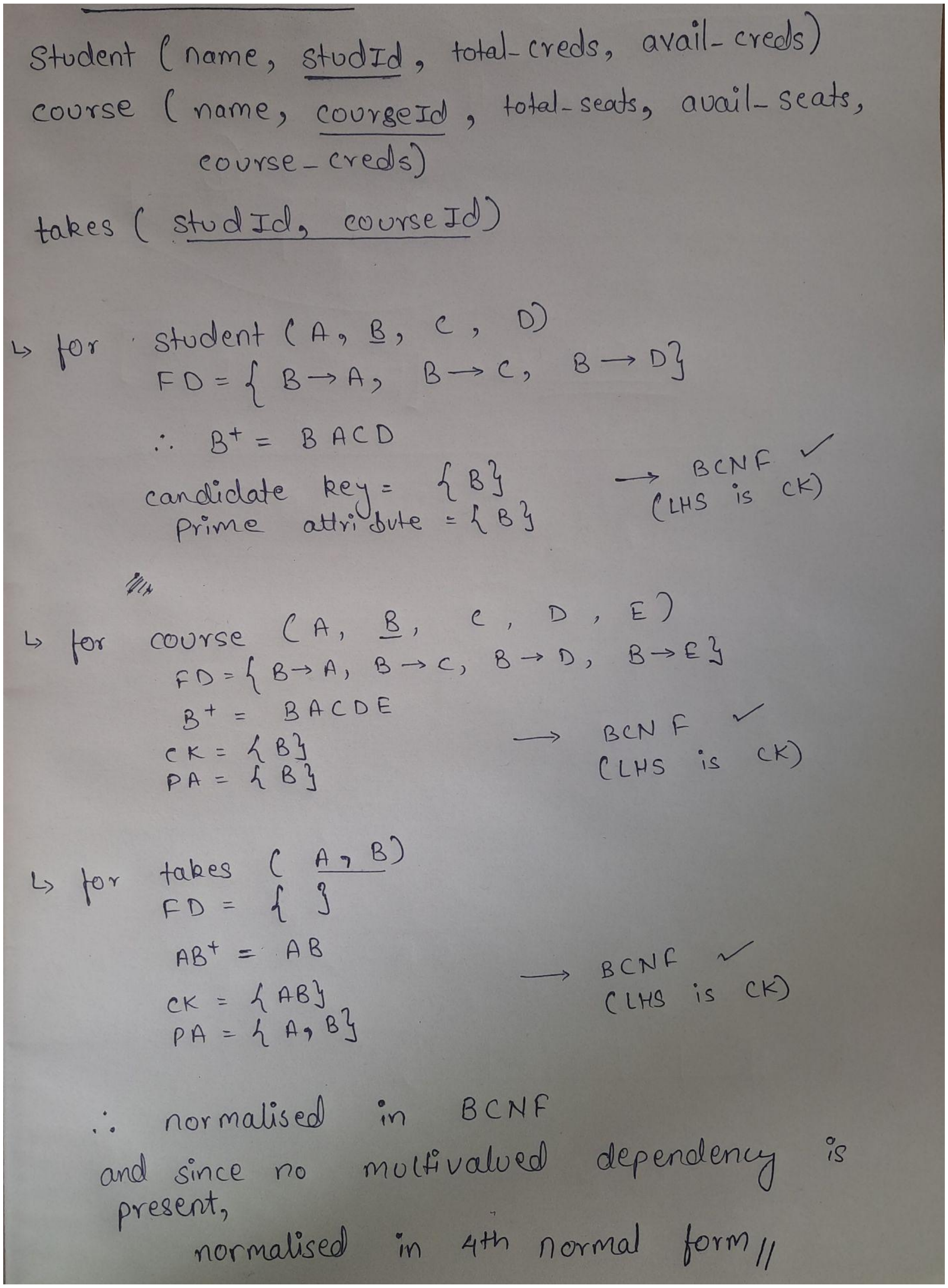
PHYSICAL ER DIAGRAM


takes
PK
(studID, courseID) NOT NULL
FK1
studID int NOT NULL references student.studID
FK2
courseID int NOT NULL references course.courseID
student
PK
studID int NOT NULL
name varchar(50) NOT NULL
total_creds int NOT NULL
avail_creds int NOT NULL
course
PK
courseID int NOT NULL
name varchar(50) NOT NULL
course_creds int NOT NULL
total_seats int NOT NULL
avail_seats int NOT NULL

CONCEPTUAL ER DIAGRAM



Normalization



Student (name, $\underline{\text{studId}}$, total-creds, avail-creds)

course (name, $\underline{\text{courseId}}$, total-seats, avail-seats,
course-creds)

takes ($\underline{\text{studId, courseId}}$)

↳ for student (A, $\underline{B}$, C, D)

FD = { B → A, B → C, B → D}

∴ B^+ = BACD

candidate key = {B} → BCNF ✓

Prime attribute = {B} (LHS is CK)

↳ for course (A, $\underline{B}$, C, D, E)

FD = { B → A, B → C, B → D, B → E}

B^+ = BACDE

CK = {B} → BCNF ✓

PA = {B} (LHS is CK)

↳ for takes ($\underline{A, B}$)

FD = { }

AB^+ = AB

CK = {AB} → BCNF ✓

PA = {A, B} (LHS is CK)

∴ normalised in BCNF

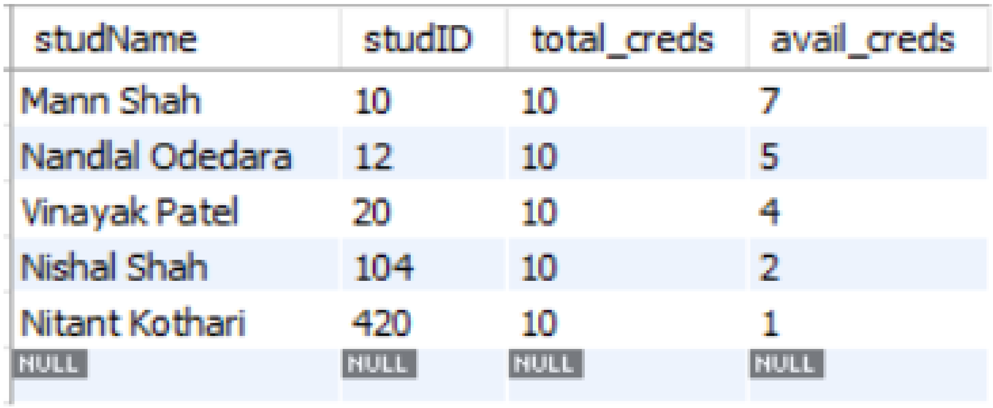
and since no multivalued dependency is present,

normalised in 4th normal form//

List of Tables required

Student table:

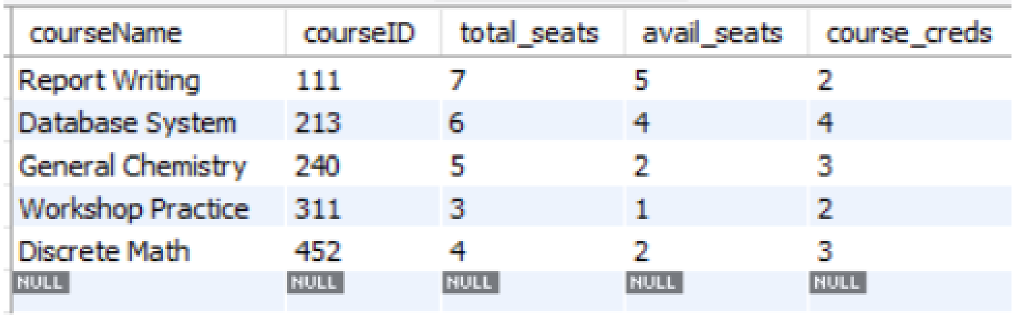
1. studID = int primary key not null
2. studName = varchar(30) not null
3. total_creds = int not null
4. avail_creds = int not null



studName	studID	total_creds	avail_creds
Mann Shah	10	10	7
Nandlal Odedara	12	10	5
Vinayak Patel	20	10	4
Nishal Shah	104	10	2
Nitant Kothari	420	10	1
NULL	NULL	NULL	NULL

Course table:

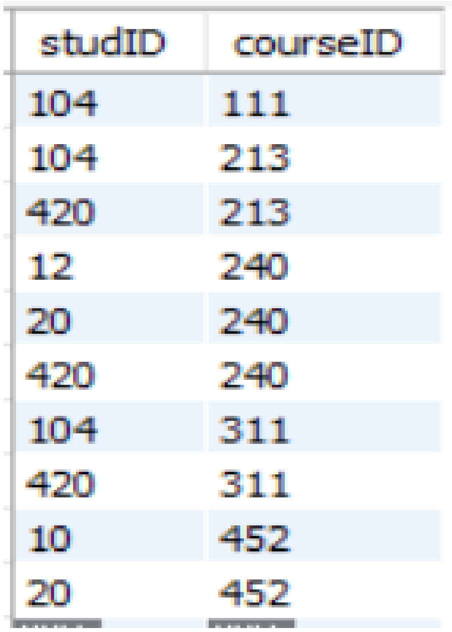
1. courseID = int primary key not null
2. courseName = varchar(30) not null
3. total_seats = int not null
4. avail_seats = int not null
5. course_creds = int not null



courseName	courseID	total_seats	avail_seats	course_creds
Report Writing	111	7	5	2
Database System	213	6	4	4
General Chemistry	240	5	2	3
Workshop Practice	311	3	1	2
Discrete Math	452	4	2	3
NULL	NULL	NULL	NULL	NULL

Takes opted:

1. studID = foreign key not null references student.studID
2. courseID = foreign key not null references course.courseID
3. (studID, courseID) = primary key not null

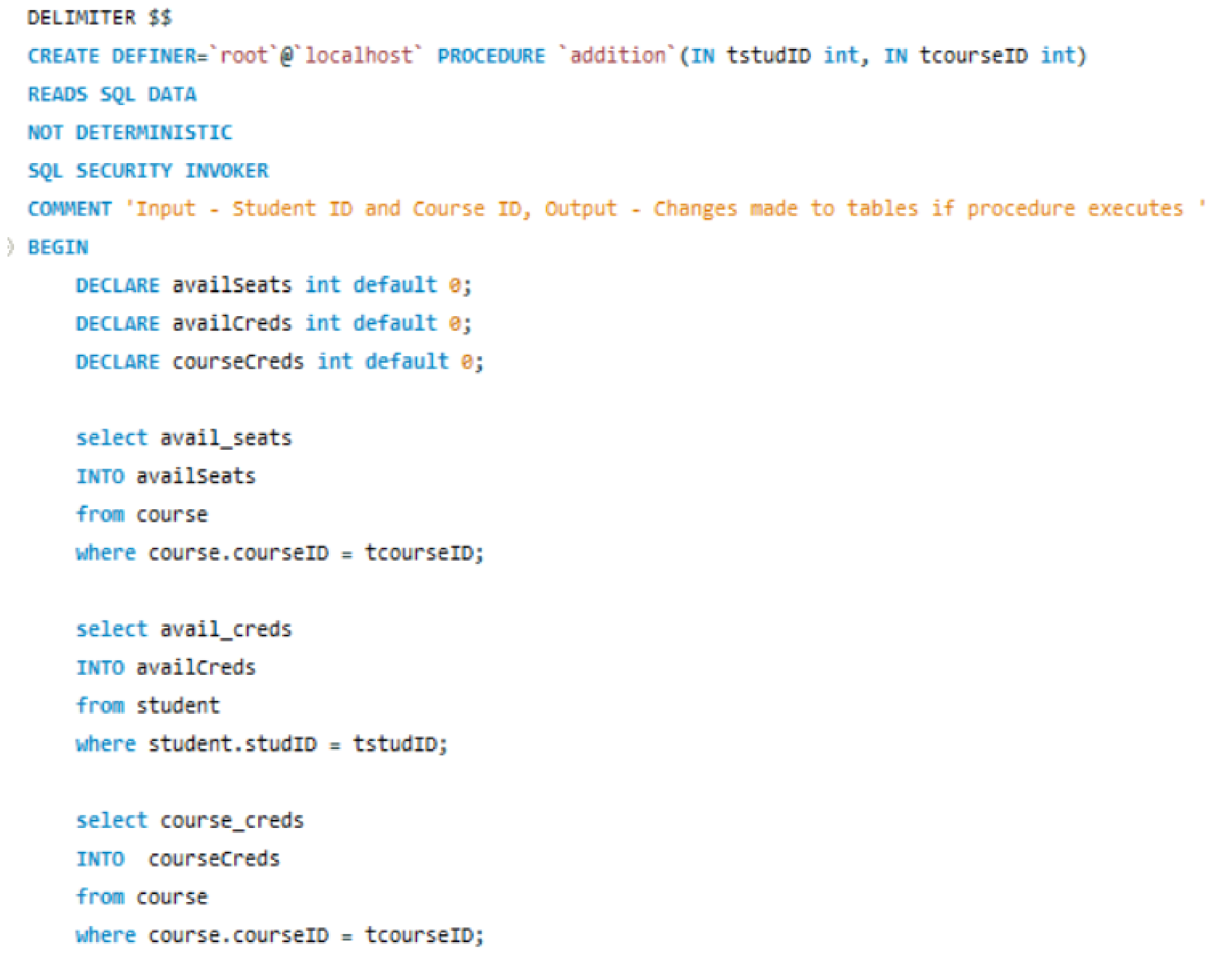


studID	courseID
104	111
104	213
420	213
12	240
20	240
420	240
104	311
420	311
10	452
20	452

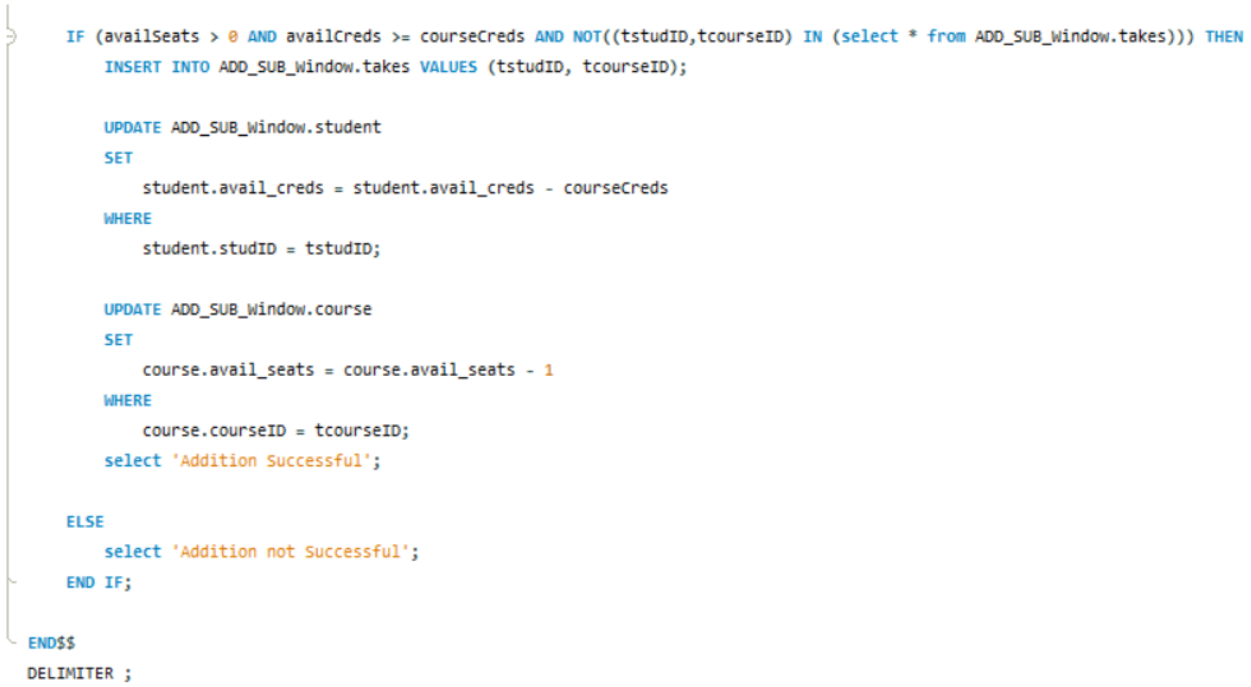
Additional Components

Procedures:

1. Addition:
    a. Inputs = int tstudID, int tcourseID.
    b. Checks = available seats in the course, if the student is already enrolled and available credits with the students.
        i. If fails: Output ERRORMessage.
        ii. Else: commit transaction.
    c. Output = Update takes table, student table, course table.

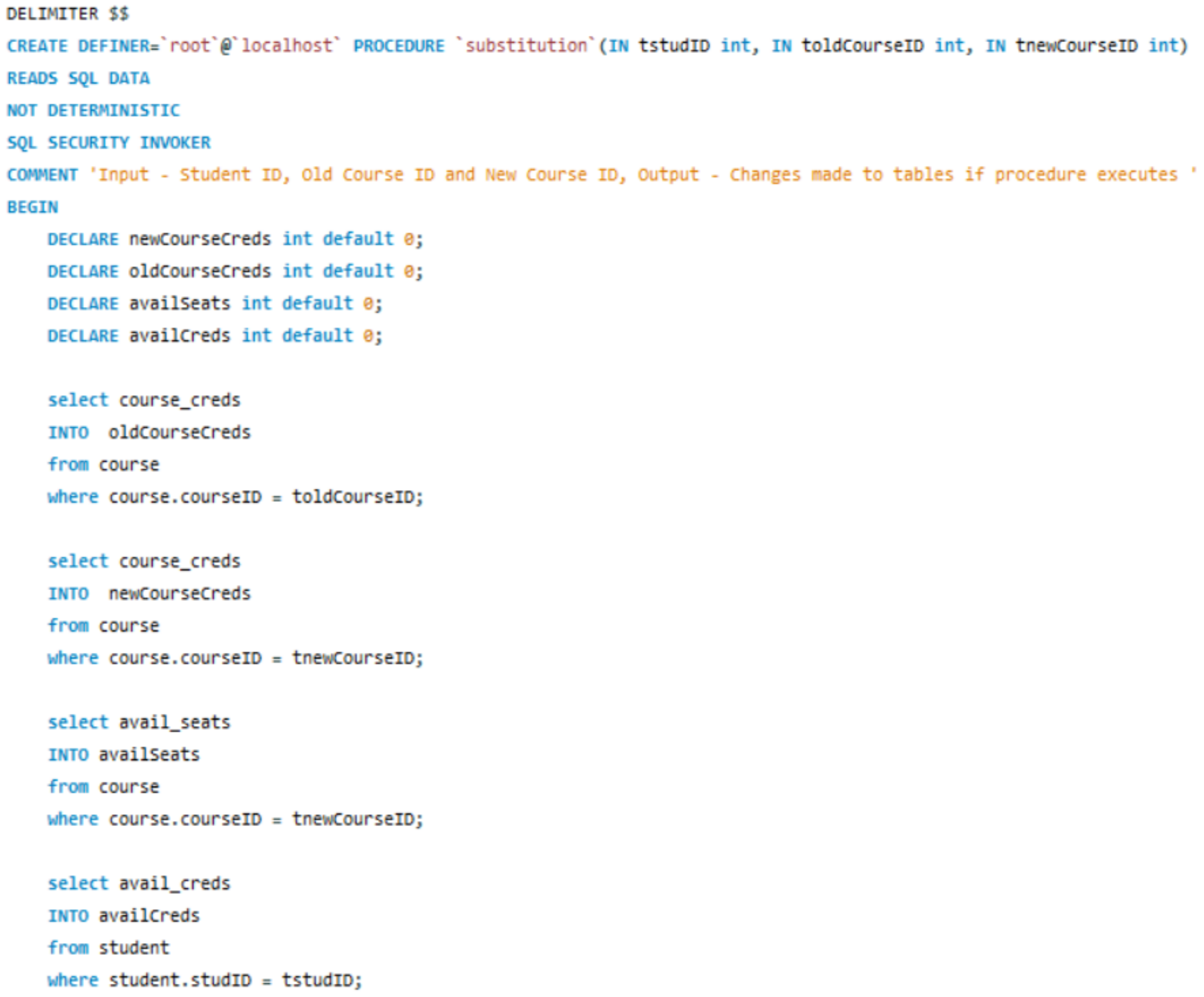


```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `addition`(IN tstudID int, IN tcourseID int)
READS SQL DATA
NOT DETERMINISTIC
SQL SECURITY INVOKER
COMMENT 'Input - Student ID and Course ID, Output - Changes made to tables if procedure executes '
BEGIN
    DECLARE availSeats int default 0;
    DECLARE availCreds int default 0;
    DECLARE courseCreds int default 0;

    select avail_seats
    INTO availSeats
    from course
    where course.courseID = tcourseID;

    select avail_creds
    INTO availCreds
    from student
    where student.studID = tstudID;

    select course_creds
    INTO  courseCreds
    from course
    where course.courseID = tcourseID;
```

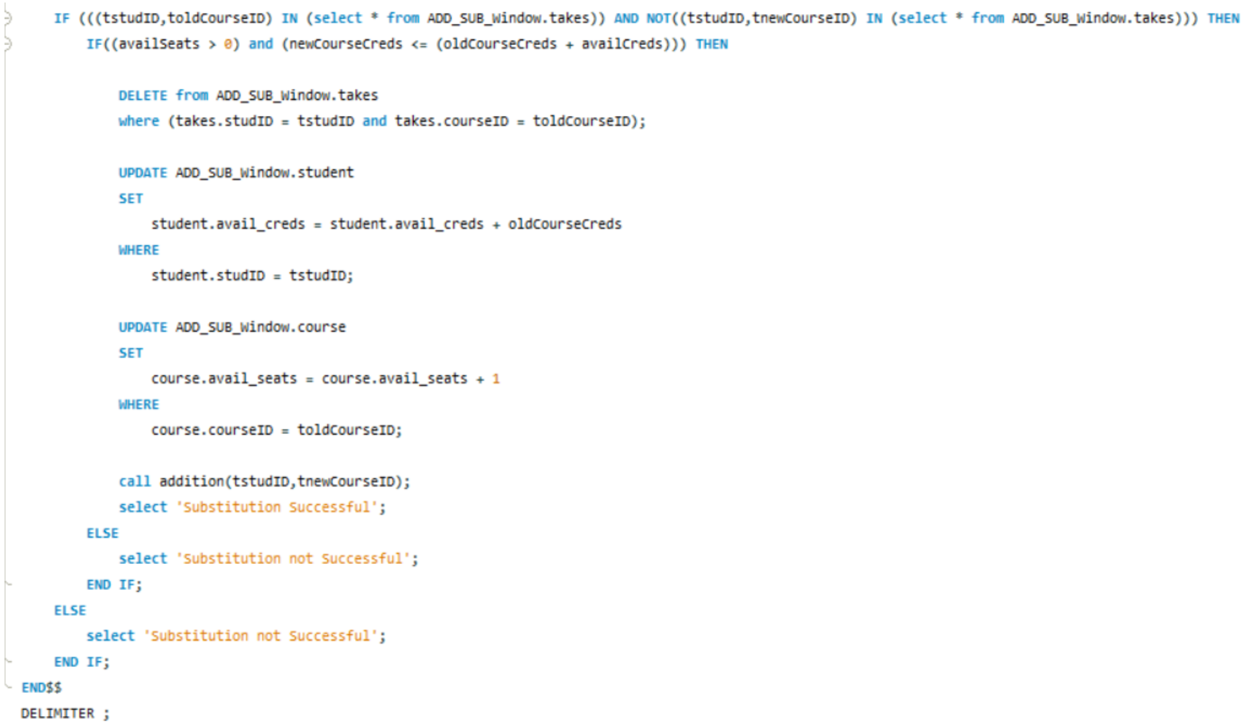


```
    IF (availSeats > 0 AND availCreds >= courseCreds AND NOT((tstudID,tcourseID) IN (select * from ADD_SUB_Window.takes))) THEN
        INSERT INTO ADD_SUB_Window.takes VALUES (tstudID, tcourseID);

        UPDATE ADD_SUB_Window.student
        SET
            student.avail_creds = student.avail_creds - courseCreds
        WHERE
            student.studID = tstudID;

        UPDATE ADD_SUB_Window.course
        SET
            course.avail_seats = course.avail_seats - 1
        WHERE
            course.courseID = tcourseID;
        select 'Addition Successful';

    ELSE
        select 'Addition not Successful';
    END IF;

END$$
DELIMITER ;
```

2. Substitution:
    a. Inputs = int studID, int toldcourseID, int tnewcourseID.
    b. Checks = available seats in the new course, if the student is already enrolled in the new course, available credits can accommodate the difference of credits from both courses.
        i. If fails: Output ERROR Message.
        ii. Else: commit transaction.
    c. Output = Update opted table, student table, course table.

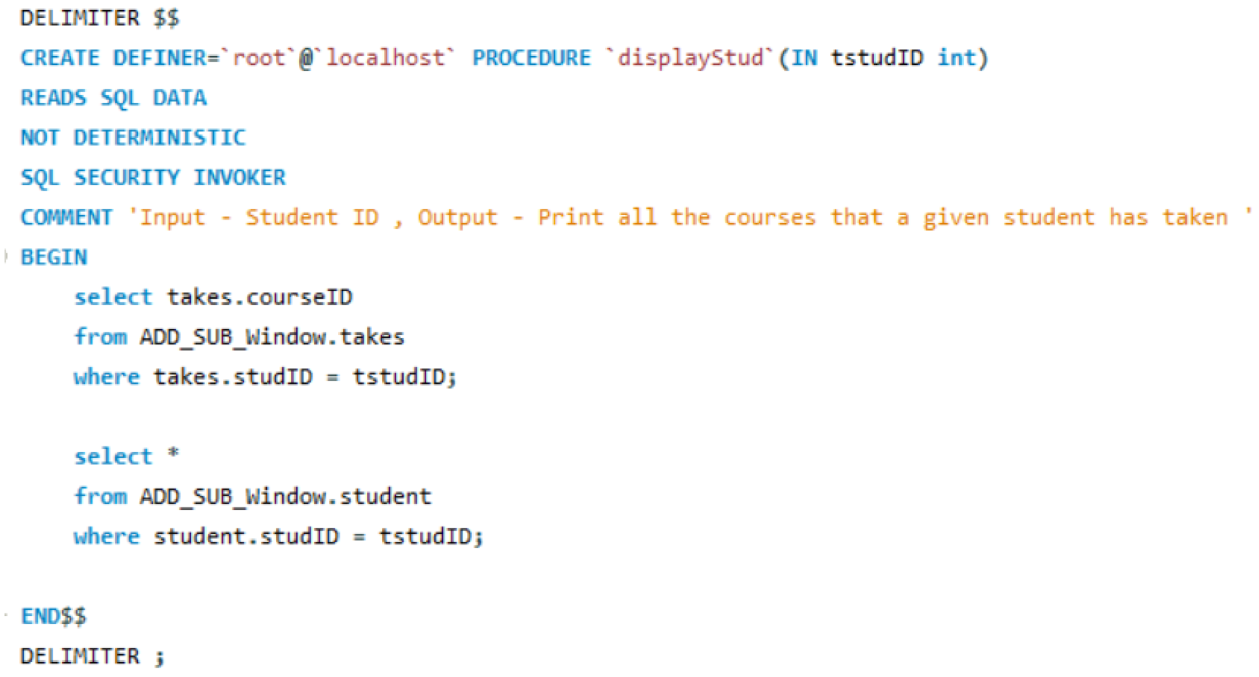


```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `substitution`(IN tstudID int, IN toldCourseID int, IN tnewCourseID int)
READS SQL DATA
NOT DETERMINISTIC
SQL SECURITY INVOKER
COMMENT 'Input - Student ID, Old Course ID and New Course ID, Output - Changes made to tables if procedure executes '
BEGIN
    DECLARE newCourseCreds int default 0;
    DECLARE oldCourseCreds int default 0;
    DECLARE availSeats int default 0;
    DECLARE availCreds int default 0;

    select course_creds
    INTO  oldCourseCreds
    from course
    where course.courseID = toldCourseID;

    select course_creds
    INTO  newCourseCreds
    from course
    where course.courseID = tnewCourseID;

    select avail_seats
    INTO availSeats
    from course
    where course.courseID = tnewCourseID;

    select avail_creds
    INTO availCreds
    from student
    where student.studID = tstudID;
```



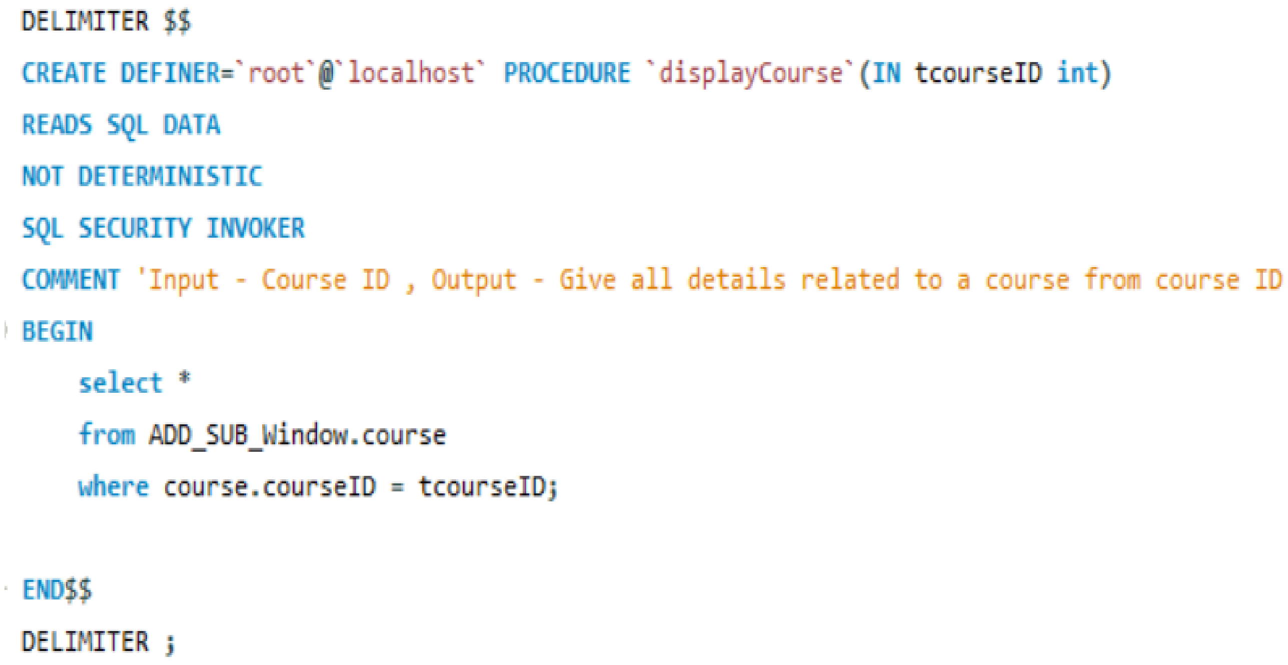
```
    IF (((tstudID,toldCourseID) IN (select * from ADD_SUB_Window.takes)) AND NOT((tstudID,tnewCourseID) IN (select * from ADD_SUB_Window.takes))) THEN
        IF((availSeats > 0) and (newCourseCreds <= (oldCourseCreds + availCreds))) THEN

            DELETE from ADD_SUB_Window.takes
            where (takes.studID = tstudID and takes.courseID = toldCourseID);

            UPDATE ADD_SUB_Window.student
            SET
                student.avail_creds = student.avail_creds + oldCourseCreds
            WHERE
                student.studID = tstudID;

            UPDATE ADD_SUB_Window.course
            SET
                course.avail_seats = course.avail_seats + 1
            WHERE
                course.courseID = toldCourseID;

            call addition(tstudID,tnewCourseID);
            select 'Substitution Successful';
        ELSE
            select 'Substitution not Successful';
        END IF;
    ELSE
        select 'Substitution not Successful';
    END IF;
END$$
DELIMITER ;
```

3. Display Student’s Courses:
    a. Inputs = int studID
    b. Output = all courses the student is enrolled in.



```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `displayStud`(IN tstudID int)
READS SQL DATA
NOT DETERMINISTIC
SQL SECURITY INVOKER
COMMENT 'Input - Student ID , Output - Print all the courses that a given student has taken '
BEGIN
    select takes.courseID
    from ADD_SUB_Window.takes
    where takes.studID = tstudID;

    select *
    from ADD_SUB_Window.student
    where student.studID = tstudID;

END$$
DELIMITER ;
```

4. Display Course details:
    a. Inputs = int courseID
    b. Output = available seats, total seats in course and credits of course.



```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `displayCourse`(IN tcourseID int)
READS SQL DATA
NOT DETERMINISTIC
SQL SECURITY INVOKER
COMMENT 'Input - Course ID , Output - Give all details related to a course from course ID
BEGIN
    select *
    from ADD_SUB_Window.course
    where course.courseID = tcourseID;

END$$
DELIMITER ;
```