CREATE TABLE IF NOT EXISTS course (

code CHAR(3) NOT NULL,

name VARCHAR(30) NOT NULL,

credits TINYINT DEFAULT 50 NOT NULL,

constraint chk\_course CHECK(credits in(50,75,100)),

constraint pri\_course PRIMARY KEY(code),

constraint pri\_course UNIQUE (code));

CREATE TABLE IF NOT EXISTS module (

code CHAR(2) NOT NULL,

name VARCHAR(30) NOT NULL,

cost DECIMAL(8,2) NOT NULL,

credits TINYINT DEFAULT 25 NOT NULL,

course\_code CHAR(3) NOT NULL,

constraint chk\_module CHECK(credits in(25,50)),

constraint pri\_module PRIMARY KEY(code),

constraint pri\_module UNIQUE (code),

constraint for\_module FOREIGN KEY(course\_code) REFERENCES course(code));

CREATE TABLE IF NOT EXISTS delegate (

no INT NOT NULL,

name VARCHAR(30) NOT NULL,

phone VARCHAR(30) NULL,

constraint pri\_delegate PRIMARY KEY(no),

constraint pri\_delegate UNIQUE (no));

CREATE TABLE IF NOT EXISTS take (

no INT NOT NULL,

code CHAR(2) NOT NULL,

grade TINYINT NULL,

constraint pri\_take PRIMARY KEY(no,code),

constraint for\_take\_no FOREIGN KEY(no) REFERENCES delegate(no),

constraint for\_take\_code FOREIGN KEY(code) REFERENCES module(code)

#constraint grade\_pass CHECK(grade >= 40))

);

CREATE TABLE IF NOT EXISTS session (

code CHAR(2) NOT NULL,

date DATE NOT NULL,

room VARCHAR(30) NULL,

constraint for\_session FOREIGN KEY(code) REFERENCES module(code),

constraint pri\_session PRIMARY KEY(date),

constraint pri\_session UNIQUE(date));

INSERT INTO course(code, name, credits)

VALUES ("WSD", "Web Systems Development", 75),

("DDM", "Database Design & Management", 100),

("NSF", "Network Security & Forensics", 75);

INSERT INTO module(code, name, cost, credits, course\_code)

VALUES ("A2", "ASP.NET", 250.00, 25, "WSD"),

("A3", "PHP", 250.00, 25, "WSD"),

("A4", "JavaFX", 350.00, 25, "WSD"),

("B2", "Oracle", 750.00, 50, "DDM"),

("B3", "SQLS", 750.00, 50, "DDM"),

("C2", "Law", 250.00, 25, "NSF"),

("C3", "Forensics", 350.00, 25, "NSF"),

("C4", "Networks", 250.00, 25, "NSF");

INSERT INTO delegate(no, name, phone)

VALUES (2001, "Mike", NULL),

(2002, "Andy", NULL),

(2003, "Sarah", NULL),

(2004, "Karen", NULL),

(2005, "Lucy", NULL),

(2006, "Steve", NULL),

(2007, "Jenny", NULL),

(2008, "Tom", NULL);

INSERT INTO take(no, code, grade)

VALUES (2003, "A2", 68),

(2003, "A3", 72),

(2003, "A4", 53),

(2005, "A2", 48),

(2005, "A3", 52),

(2002, "A2", 20),

(2002, "A3", 30),

(2002, "A4", 50),

(2008, "B2", 90),

(2007, "B2", 73),

(2007, "B3", 63);

INSERT INTO session(code, date, room)

VALUES ("A2", '2019-06-05', "305"),

("A3", '2019-06-06', "307"),

("A4", '2019-06-07', "305"),

("B2", '2019-08-22', "208"),

("B3", '2019-08-23', "208"),

("A2", '2020-05-01', "303"),

("A3", '2020-05-02', "305"),

("A4", '2020-05-03', "303"),

("B2", '2020-07-10', NULL),

("B3", '2020-07-11', NULL);

#View Statement

CREATE VIEW sessions\_in\_future

AS SELECT code, date

FROM session

WHERE date >= CURRENT\_DATE()

WITH CHECK OPTION;

#This should get rejected

INSERT INTO sessions\_in\_future(code, date)

VALUES ("A1", '2019-10-09');

#This should also get rejected

UPDATE sessions\_in\_future

SET date = '2019-10=08'

WHERE code = "A2";

#Procedure Statement

DELIMITER $$

CREATE PROCEDURE new\_schedule(IN code\_course CHAR(3), IN start\_date DATE)

BEGIN

DECLARE finished BOOLEAN DEFAULT FALSE;

DECLARE module\_code CHAR(2);

DECLARE code\_c CURSOR FOR

SELECT `code` FROM module WHERE course\_code = code\_course;

DECLARE CONTINUE HANDLER FOR SQLSTATE '02000'

SET finished = TRUE;

IF (start\_date < DATE\_ADD(CURRENT\_DATE(), INTERVAL 1 MONTH)) THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Date must be at least 1 month in the future';

END IF;

IF WEEKDAY(start\_date) = 5 OR WEEKDAY(start\_date) = 6 THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Date must not be on a weekend';

END IF;

IF code\_course NOT IN ("WSD", "DDM", "NSF") THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Course code does not exist';

END IF;

OPEN code\_c;

WHILE NOT finished DO

FETCH NEXT FROM code\_c INTO module\_code;

IF WEEKDAY(start\_date) = 5 THEN

SET start\_date = DATE\_ADD(start\_date, INTERVAL 2 DAY);

END IF;

IF WEEKDAY(start\_date) = 6 THEN

SET start\_date = DATE\_ADD(start\_date, INTERVAL 1 DAY);

END IF;

INSERT INTO `session`(`code`, `date`, room)

VALUES(module\_code, start\_date, NULL);

SET start\_date = DATE\_ADD(start\_date, INTERVAL 1 DAY);

IF finished = TRUE THEN

CLOSE code\_c;

END IF;

END WHILE;

END$$

CALL new\_schedule("DDM", '2021-05-06');

CREATE TABLE IF NOT EXISTS audit (

audit\_number INT NOT NULL AUTO\_INCREMENT,

username VARCHAR(30) NOT NULL,

system\_date DATE NOT NULL,

old\_grade TINYINT NULL,

new\_grade TINYINT NULL,

module\_code CHAR(2) NOT NULL,

delegate\_no INT NOT NULL,

constraint pri\_audit PRIMARY KEY(audit\_number));

DELIMITER $$

#Trigger Statements

CREATE TRIGGER update\_grade

AFTER UPDATE ON take FOR EACH ROW

BEGIN

IF (NEW.grade <> OLD.grade) THEN

INSERT INTO audit(audit\_number, username, system\_date, old\_grade, new\_grade, module\_code, delegate\_no)

VALUES(audit\_number, CURRENT\_USER(), CURRENT\_DATE(), OLD.grade, NEW.grade, NEW.code, NEW.no);

END IF;

END$$

#Testing the trigger statement

UPDATE take

SET grade = 70

WHERE no = '2003' AND code = 'A2';

#Query Functionality

#1

SELECT code, name, credits

FROM module;

#2

SELECT no, name

FROM delegate

ORDER BY name DESC;

#3

SELECT code, name, credits

FROM course

WHERE name LIKE "%Network%";

#4

SELECT MAX(grade) AS HighestGrade

FROM take;

#5

SELECT no

FROM take WHERE grade = (

SELECT MAX(grade) FROM take);

#6

SELECT no, name

FROM delegate

WHERE no IN (

SELECT no FROM take WHERE grade = (

SELECT MAX(grade) FROM take));

#7

SELECT code, date

FROM session WHERE date

BETWEEN CURRENT\_DATE() AND DATE\_ADD(CURRENT\_DATE(), INTERVAL 1 YEAR)

AND room IS NULL;

#8

SELECT delegate.no, delegate.name AS delegate\_name, module.name AS module\_name, module.code

FROM delegate INNER JOIN take ON delegate.no = take.no

INNER JOIN module ON take.code = module.code

WHERE take.grade < 40;

#9

SELECT delegate.no, delegate.name

FROM delegate INNER JOIN take ON delegate.no = take.no

WHERE take.grade = (

SELECT MAX(grade) FROM take);

#10

SELECT delegate.no, delegate.name, SUM(module.credits) AS TotalCredits, course.code, course.name, course.credits

FROM delegate INNER JOIN take ON delegate.no = take.no

INNER JOIN module ON take.code = module.code

INNER JOIN course ON module.course\_code = course.code

GROUP BY delegate.no, delegate.name;

#11

SELECT delegate.no, delegate.name, SUM(module.credits) AS TotalCredits, course.code, course.name, course.credits

FROM delegate INNER JOIN take ON delegate.no = take.no

INNER JOIN module ON take.code = module.code

INNER JOIN course ON module.course\_code = course.code

GROUP BY delegate.no, delegate.name

HAVING TotalCredits >= course.credits;