Contents

[CREATE TABLE Statements 1](#_Toc149569498)

[CREATE VIEW Statements. 2](#_Toc149569499)

[CREATE PROCEDURE Statements 2](#_Toc149569500)

[CREATE TRIGGER Statements 4](#_Toc149569501)

[DML STATEMENTS IN SQL 5](#_Toc149569502)

DEVELOPING FOR RDBSM

# CREATE TABLE Statements

CREATE TABLE course

(code CHAR(3),

name VARCHAR(30),

CONSTRAINT code PRIMARY KEY (code),

credits TINYINT);

CREATE TABLE module

(code CHAR(2),

name VARCHAR(30) NOT NULL,

cost DECIMAL(8,2)NOT NULL,

credits TINYINT NOT NULL,

course\_code CHAR(3),

CONSTRAINT code PRIMARY KEY (code));

CREATE TABLE session

(code CHAR(2) NOT NULL ,

date Date PRIMARY KEY,

room VARCHAR(30) NULL,

CONSTRAINT code FOREIGN KEY (code) REFERENCES module (code));

CREATE TABLE delegate

(no INT,

name VARCHAR(30) NOT NULL,

phone VARCHAR(30) NULL,

CONSTRAINT no PRIMARY KEY (no));

CREATE TABLE take

(code CHAR(2) ,

no INT NOT NULL,

grade TINYINT NOT NULL,

CONSTRAINT codex PRIMARY KEY (code, no),

CONSTRAINT codey FOREIGN KEY (code) REFERENCES module (code),

CONSTRAINT no1 FOREIGN KEY (no) REFERENCES delegate (no));

# CREATE VIEW Statements.

CREATE VIEW future\_sessions

AS

SELECT code,date

FROM session

WHERE date > CURRENT\_DATE;

WITH CHECK OPTION;

|  |  |
| --- | --- |
| Inserting any date not within the ranges results in the following message…: Error Code: 1369. CHECK OPTION failed 'coursework.future\_sessions' |  |

INSERT INTO future\_sessions

VALUES ('A4','2023.05.21') //

INSERT INTO future\_sessions //

Where as this statement works as it is valid with the check option

VALUES ('A2','2026.05.21')

# CREATE PROCEDURE Statements

DELIMITER $$

CREATE PROCEDURE schedule\_for\_course (IN pk\_code CHAR(3), IN start\_date DATE)

BEGIN

-- declaring variable(s) !

DECLARE complete BOOLEAN DEFAULT FALSE;

-- cursor declaration+

DECLARE cur\_coursecode CURSOR FOR

SELECT 'code' FROM module WHERE course\_code = pk\_code;

DECLARE CONTINUE HANDLER FOR NOT FOUND

SET complete = TRUE;

-- checking course exists !

IF (pk\_code) IS NULL THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = ' COURSE DOES NOT EXIST';

END IF;

-- checking course start date !

IF (start\_date,CURRENT\_DATE() < 31) THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = ' MINIMUM START DATE NEEDS TO BE A MONTH AHEAD';

END IF;

OPEN cur\_coursecode; -- CURSOR;

loopy\_doopy\_do : LOOP -- name loop

FETCH NEXT FROM cur\_coursecode INTO pk\_code; -- put cursor into here

-- when cursor runs out, each module has been processed !

IF complete THEN

LEAVE loopy\_doopy\_do;

END IF;

-- checking for & skipping saturday & sunday !

IF DAYOFWEEK(start\_date) = 1 THEN

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

ELSEIF DAYOFWEEK(start\_date) = 7 THEN

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

END IF;

INSERT INTO `session`

(`code`, `date`, room)

VALUES

(pk\_code, start\_date,NULL);

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

END LOOP;

CLOSE cur\_coursecode;

END $$

# CREATE TRIGGER Statements

DROP TABLE grade\_change;

DROP TRIGGER audit;

CREATE TABLE grade\_change (

updated\_grade TINYINT NOT NULL,

Current\_username VARCHAR(30),

Date\_updated DATETIME

);

DELIMITER $$

CREATE TRIGGER audit

AFTER UPDATE ON take FOR EACH ROW

BEGIN

IF NEW.grade <> OLD.grade THEN

INSERT INTO grade\_change (updated\_grade,Current\_username,Date\_updated)

VALUES(NEW.grade, CURRENT\_USER(), SYSDATE());

ELSE

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = "ERROR";

END IF;

END $$

DELIMITER ;

UPDATE take

SET grade = 100

WHERE code = 'B3' and no = 2007;

# DML STATEMENTS IN SQL

1). SELECT code,name,credits

FROM module

2). SELECT no,name

FROM delegate

GROUP BY no,name

ORDER BY name DESC;

3). SELECT code,name,credits

FROM course

WHERE name LIKE '%NETWORK%';

4). SELECT MAX(grade) AS 'highest'

FROM take;

5). SELECT no AS 'highest'

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\_ADD(CURRENT\_DATE,INTERVAL 365 DAY)

FROM session

WHERE (CURRENT\_DATE) AND (room is NULL)

8) SELECT T.no,T.code,D.name,D.no,M.code,M.name,T.grade

FROM take T

INNER JOIN delegate D ON T.no = D.no

INNER JOIN module M ON T.code = M.code

WHERE grade < 40

9) SELECT D.no,D.name

FROM take T

INNER JOIN delegate D ON T.no = D.no

WHERE T.grade =

(SELECT MAX(grade)

FROM take T );

10). SELECT D.no,D.name,SUM(M.credits),C.code,C.name,C.credits

FROM take T

INNER JOIN delegate D ON T.no = D.no

INNER JOIN module M ON M.code = T.code

INNER JOIN course C on M.course\_code = C.code

GROUP BY D.no,D.name,C.code

11). SELECT D.no,D.name,SUM(M.credits),C.code,C.name,C.credits

FROM take T

INNER JOIN delegate D ON T.no = D.no

INNER JOIN module M ON M.code = T.code

INNER JOIN course C on M.course\_code = C.code

GROUP BY D.no,D.name,C.code

HAVING SUM(M.credits) = C.credits