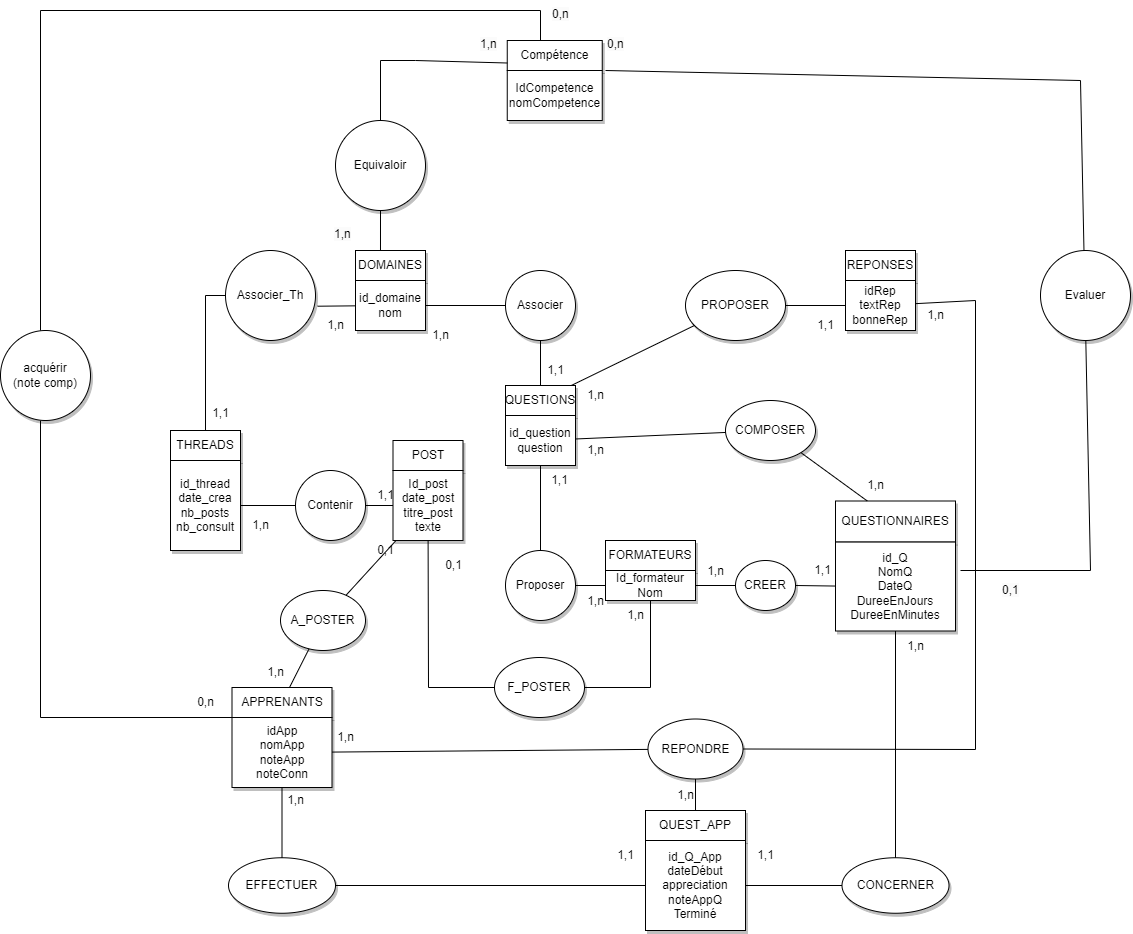
**SAE S2.04 Exploitation d'une base de données : PARTIE 1**

**EXERCICE 1**



**EXERCICE 2**

DROP TABLE IF EXISTS ACQUERIR;

DROP TABLE IF EXISTS EQUIVALOIR;

DROP TABLE IF EXISTS Repondre;

DROP TABLE IF EXISTS Composer;

DROP TABLE IF EXISTS QUEST\_APP;

DROP TABLE IF EXISTS Reponses;

DROP TABLE IF EXISTS Questions;

DROP TABLE IF EXISTS Questionnaires;

DROP TABLE IF EXISTS Competences;

DROP TABLE IF EXISTS POSTS;

DROP TABLE IF EXISTS Threads;

DROP TABLE IF EXISTS FORMATEURS;

DROP TABLE IF EXISTS Apprenants;

DROP TABLE IF EXISTS Domaines;

CREATE TABLE Domaines (

idDomaine INT,

nomDomaine VARCHAR(20) NOT NULL,

CONSTRAINT Pk\_id\_domaine PRIMARY KEY(idDomaine)

);

CREATE TABLE Apprenants (

idApp INT,

nomApp VARCHAR(55) NOT NULL,

noteApp FLOAT DEFAULT NULL,

NoteConn FLOAT DEFAULT NULL,

CONSTRAINT PK\_App PRIMARY KEY (idApp)

);

CREATE TABLE Formateurs (

idFor INT,

nomFor VARCHAR(55) NOT NULL,

CONSTRAINT PK\_App PRIMARY KEY (idFor)

);

CREATE TABLE Threads (

idThread INT,

dateCrea DATE NOT NULL,

nbPosts INT DEFAULT 1,

nbConsult INT DEFAULT 0,

idDomaine INT NOT NULL,

CONSTRAINT Pk\_id\_threads PRIMARY KEY (idThread),

CONSTRAINT Fk\_threads\_domaine FOREIGN KEY (idDomaine)

REFERENCES Domaines(idDomaine) ON UPDATE CASCADE

);

CREATE TABLE POSTS (

idPost INT,

datePost DATE NOT NULL,

titrePost VARCHAR(20) NOT NULL,

textePost TEXT NOT NULL,

idThread INT,

idApp INT,

idFor INT,

CONSTRAINT PK\_ID\_POST PRIMARY KEY (idPost),

CONSTRAINT FK\_posts\_THREADS FOREIGN KEY (idThread)

REFERENCES Threads(idThread) ON UPDATE CASCADE,

CONSTRAINT Fk\_posts\_ID\_APP FOREIGN KEY (idApp)

REFERENCES Apprenants(idApp) ON UPDATE CASCADE,

CONSTRAINT Fk\_posts\_ID\_FOR FOREIGN KEY (idFor)

REFERENCES Formateurs(idFor) ON UPDATE CASCADE

/\*CONSTRAINT CHECK\_app\_for CHECK ((idApp IS NOT NULL AND idFor IS NULL) OR (idApp IS NULL AND idFor IS NOT NULL)) A Faire dans un trigger\*/

);

CREATE TABLE Competences (

id\_comp INT UNIQUE,

nomComp VARCHAR(55) UNIQUE,

CONSTRAINT Pk\_Competences PRIMARY KEY (id\_comp)

);

CREATE TABLE Questionnaires (

idQuest INT,

nomQuest VARCHAR(55) NOT NULL,

dateQuest DATE NOT NULL,

dureeQuest TIME DEFAULT NULL,

idFor INT,

id\_comp INT,

CONSTRAINT PK\_QUESTIONNAIRES PRIMARY KEY (idQuest),

CONSTRAINT Fk\_QUESTIONNAIRES\_For FOREIGN KEY (idFor) REFERENCES Formateurs(idFor) ON UPDATE CASCADE,

CONSTRAINT Fk\_QUESTIONNAIRES\_comp FOREIGN KEY (id\_comp) REFERENCES Competences(id\_comp) ON UPDATE CASCADE

);

CREATE TABLE Questions (

idQuestion INT,

textQuestion TEXT NOT NULL,

idDomaine INT,

idFor INT,

CONSTRAINT PK\_QUESTIONS\_ID\_QUESTION PRIMARY KEY (idQuestion),

CONSTRAINT Fk\_QUESTIONS\_id\_domaine FOREIGN KEY (idDomaine)

REFERENCES Domaines(idDomaine) ON UPDATE CASCADE,

CONSTRAINT Fk\_QUESTIONS\_id\_formateur FOREIGN KEY (idFor)

REFERENCES Formateurs(idFor) ON UPDATE CASCADE

);

CREATE TABLE Reponses (

idRep INT,

textRep TEXT NOT NULL,

bonneRep INT NOT NULL,

idQuestion INT,

CONSTRAINT PK\_idRep PRIMARY KEY (IDREP),

CONSTRAINT Fk\_question FOREIGN KEY (idQuestion)

REFERENCES Questions(idQuestion) ON UPDATE CASCADE

);

CREATE TABLE QUEST\_APP (

idQApp INT,

appreciationQApp TEXT NOT NULL,

noteQApp INT NOT NULL,

dateQApp DATETIME NOT NULL,

Termine BOOLEAN DEFAULT 0 NOT NULL CHECK (Termine IN (0,1)),

idQuest INT,

id\_app INT,

CONSTRAINT PK\_QUEST\_APPS\_id\_Q\_app PRIMARY KEY (idQApp),

CONSTRAINT FK\_QUEST\_APPS\_id\_questionnaire FOREIGN KEY (idQuest)

REFERENCES Questionnaires(idQuest) ON UPDATE CASCADE,

CONSTRAINT FK\_QUEST\_APPS\_ID\_APP42 FOREIGN KEY (ID\_APP)

REFERENCES Apprenants(idApp) ON UPDATE CASCADE

);

CREATE TABLE Composer (

idQuestion INT,

idQuest INT,

CONSTRAINT PK\_Composer\_ID\_QUESTION\_QUESTIONNAIRE PRIMARY KEY (idQuestion, idQuest),

CONSTRAINT Fk\_Composer\_ID\_QUESTION FOREIGN KEY (idQuestion) REFERENCES Questions(idQuestion) ON UPDATE CASCADE,

CONSTRAINT FK\_Composer\_ID\_QUESTIONNAIRE FOREIGN KEY (idQuest) REFERENCES QUESTIONNAIRES(idQuest) ON UPDATE CASCADE

);

CREATE TABLE REPONDRE (

idApp INT,

idQApp INT,

idRep INT,

CONSTRAINT Pk\_REPONDRE\_ID\_REP PRIMARY KEY (idApp,idQApp,idRep),

CONSTRAINT FK\_REPONDRE\_ID\_APP FOREIGN KEY (idApp)

REFERENCES APPRENANTS(idApp) ON UPDATE CASCADE,

CONSTRAINT FK\_REPONDRE\_ID\_QUESTION\_APP FOREIGN KEY (idQApp) REFERENCES QUEST\_APP(idQApp) ON UPDATE CASCADE,

CONSTRAINT FK\_REPONDRE\_ID\_REP FOREIGN KEY (idRep) REFERENCES REPONSES(idRep) ON UPDATE CASCADE

);

Create TABLE EQUIVALOIR (

id\_domaine INT,

id\_comp INT,

CONSTRAINT Pk\_EQUIVALOIR PRIMARY KEY (id\_domaine,id\_comp),

CONSTRAINT FK\_EQUIVALOIR\_domaine FOREIGN KEY (id\_domaine) REFERENCES Domaines(idDomaine) ON UPDATE CASCADE,

CONSTRAINT FK\_EQUIVALOIR\_comp FOREIGN KEY (id\_comp) REFERENCES Competences(id\_comp) ON UPDATE CASCADE

);

CREATE TABLE ACQUERIR (

id\_comp INT,

id\_app INT,

notecomp INT,

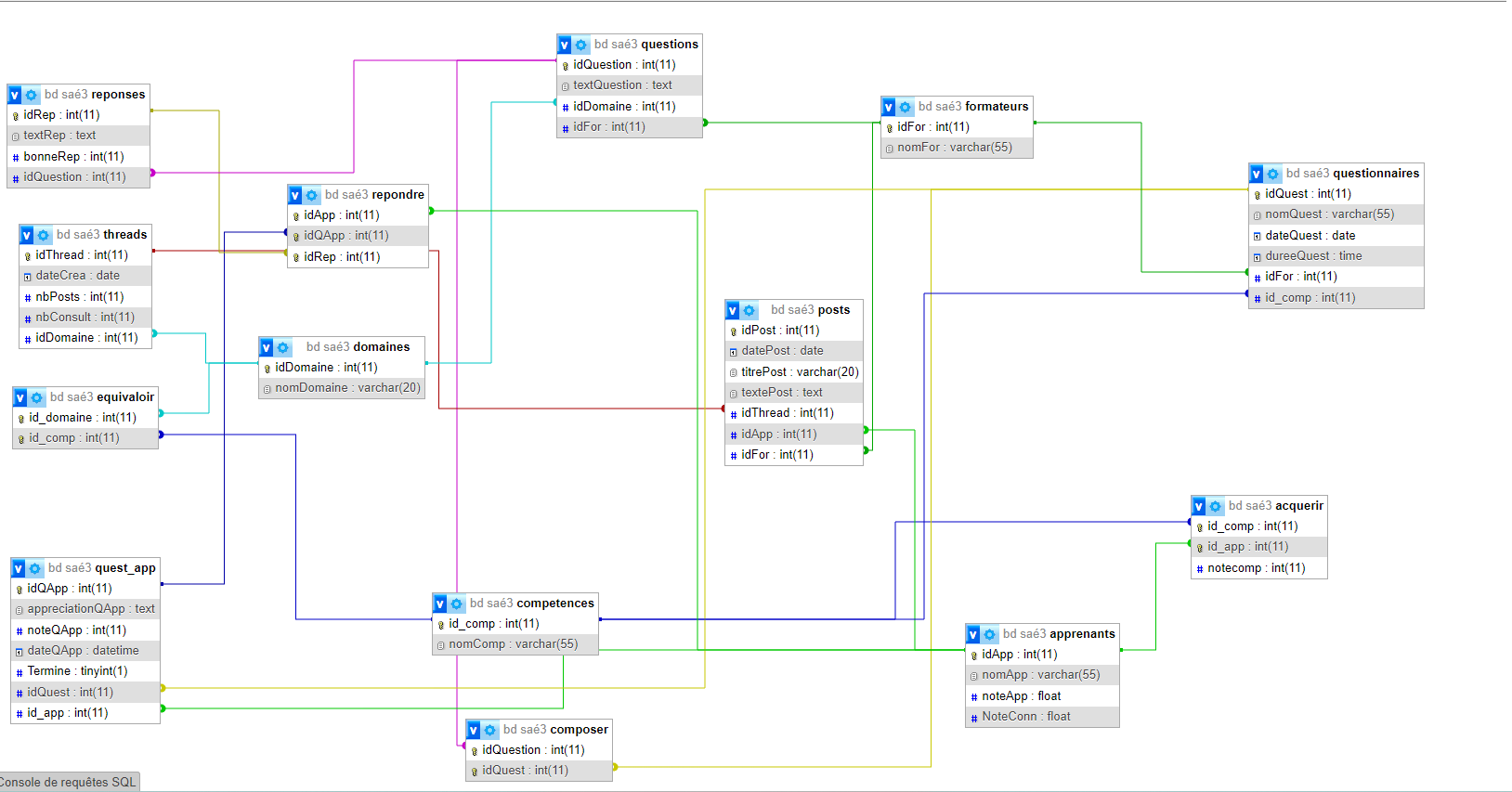
CONSTRAINT PK\_ACQUERIR PRIMARY KEY (id\_comp,id\_app),

CONSTRAINT FK\_ACQUERIR\_app FOREIGN KEY (id\_app) REFERENCES Apprenants(idApp) ON UPDATE CASCADE,

CONSTRAINT FK\_ACQUERIR\_comp FOREIGN KEY (id\_comp) REFERENCES Competences(id\_comp) ON UPDATE CASCADE

);

**EXERCICE 3**



**EXERCICE 4**

INSERT INTO Competences (id\_comp, nomComp)

VALUES (1, 'C1'),

(2, 'C2');

INSERT INTO Domaines (idDomaine, nomDomaine)

VALUES (101, 'D101'),

(102, 'D102'),

(103, 'D103');

INSERT INTO EQUIVALOIR (id\_domaine, id\_comp)

VALUES (101, 1),

(102, 1),

(103, 2);

INSERT INTO formateurs (idFor,nomFor)

VALUES (1,'Sauvage');

INSERT INTO Questionnaires (idQuest, nomQuest, dateQuest, idFor, id\_comp)

VALUES (100, 'E100', CURRENT\_DATE, 1, 1),

(101, 'E101', CURRENT\_DATE, 1, 1),

(110, 'E110', CURRENT\_DATE, 1, 2),

(111, 'E111', CURRENT\_DATE, 1, 2),

(112, 'E112', CURRENT\_DATE, 1, 2),

(113, 'E113', CURRENT\_DATE, 1, 2);

INSERT INTO Questions (idQuestion, textQuestion, idDomaine, idFor)

VALUES (100, 'Question pour D101', 101, 1),

(101, 'Question pour D102', 102, 1),

(102, 'Question pour D101', 101, 1),

(103, 'Question pour D101', 101, 1),

(110, 'Question pour D103', 103, 1),

(111, 'Question pour D103', 103, 1),

(112, 'Question pour D103', 103, 1);

INSERT INTO Composer (idQuestion, idQuest)

VALUES (102, 101),

(103, 101),

(100, 100),

(101, 100),

(110, 110),

(111, 111),

(112, 112),

(110, 113),

(111, 113),

(112, 113);

**EXERCICE 5**

Les deux règles de gestion liées à ces données sont :des noms uniques pour les compétences et le changement des valeurs des compétences lorsqu’un (implémentation avec la contrainte UNIQUE sur la colonne “nomComp” de la table “Compétences”), et le contrôle de changement de compétences. Si des/une valeur est changée, tout ce qui est lié à cette valeur et ce changement va aussi être modifié. (UPDATE CASCADE)

*Vérification des noms de compétences qui doivent être uniques.*

INSERT INTO Competences (id\_comp, nomComp)

VALUES (1, 'C2'),

(2, 'C2');

UNIQUE permet de vérifier que le nom rempli est unique.

*Changement sur les valeurs id\_comp de Questionnaires lors d’une update sur compétences.*

DELETE FROM Competences

WHERE id\_comp = 1

SELECT \* FROM Questionnaires

On UPDATE CASCADE sur la clé étrangère id\_comp

**EXERCICE 6**

Le nom des compétences peut ne pas être renseigné.

INSERT INTO Competences (id\_comp, nomComp)

VALUES (5,'');

On peut faire un trigger pour contrôler le remplissage du nom.

On ne peut pas contrôler que toutes les questions d’un examen de compétence sont toutes du même domaine

INSERT INTO questions (idDomaine, idFor, idQuestion, textQuestion)

VALUES (101, 1, 1, 'question1'),

(102, 1, 2, 'question2');

INSERT INTO composer (idQuestion, idQuest)

VALUES (1,100),

(2,100);

On peut régler cela à la mise à jour des questionnaire avec un trigger

**EXERCICE 7**

DELIMITER //

CREATE OR REPLACE FUNCTION VerifierCompétencesExamen (IdQuest INT, id\_comp INT)

RETURNS BOOLEAN

BEGIN

DECLARE v\_CompetenceCouvert BOOLEAN;

SELECT COUNT(\*) INTO v\_CompetenceCouvert

FROM Questions Q

INNER JOIN COMPOSER C ON Q.idQuestion = C.idQuestion

INNER JOIN questionnaires QU ON C.idQuest = QU.idQuest

INNER JOIN competences COMP ON QU.id\_comp = COMP.id\_comp

INNER JOIN equivaloir EQ ON COMP.id\_comp = EQ.id\_comp

WHERE QU.idQuest = IdQuest AND EQ.id\_domaine IN (SELECT idDomaine FROM Domaines WHERE id\_comp = id\_comp);

RETURN v\_CompetenceCouvert;

END//

DELIMITER ;

**EXERCICE 8**

DELIMITER $$

CREATE OR REPLACE PROCEDURE AffecterCompetenceApprenant (IN idApp INT, IN idComp INT)

BEGIN

DECLARE moy FLOAT;

DECLARE nbnote INT;

DECLARE notemin INT;

SELECT AVG(noteAPPQ) INTO moy

FROM quest\_app

WHERE id\_app = idApp AND idQuest IN (SELECT idQuest FROM questionnaires WHERE id\_comp = idComp);

SELECT COUNT(\*) INTO nbnote

FROM quest\_app

WHERE id\_app = idApp AND idQuest IN (SELECT idQuest FROM questionnaires WHERE id\_comp = idComp);

SELECT MIN(noteAPPQ) INTO notemin

FROM quest\_app

WHERE id\_app = idApp AND idQuest IN (SELECT idQuest FROM questionnaires WHERE id\_comp = idComp);

IF (nbnote = 2 AND moy >= 12) OR

(nbnote = 3 AND moy >= 10 AND notemin >= 8) OR

(nbnote >= 4 AND moy >= 10) THEN

INSERT INTO acquerir (id\_comp, id\_app, notecomp)

VALUES (idComp, idApp, moy);

END IF;

END $$

DELIMITER ;

**EXERCICE 9**

DELIMITER $$

CREATE OR REPLACE PROCEDURE DélivrerCompétenceApprenant ()

BEGIN

DECLARE done INT DEFAULT FALSE;

DECLARE cur\_idApp INT;

DECLARE cur\_idComp INT;

DECLARE cur\_idQuest INT;

DECLARE notes\_count INT;

DECLARE avg\_note FLOAT;

DECLARE cur\_curseur CURSOR FOR

SELECT Q.id\_app, Q.id\_comp, Q.idQuest

FROM QUEST\_APP Q

JOIN POSTS P ON Q.id\_app = P.idApp

JOIN Threads T ON P.idThread = T.idThread

WHERE Q.Termine = 1;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

OPEN cur\_curseur;

lecture: LOOP

FETCH cur\_curseur INTO cur\_idApp, cur\_idComp, cur\_idQuest;

IF done THEN

LEAVE lecture;

END IF;

IF NOT EXISTS (

SELECT \* FROM ACQUERIR

WHERE id\_app = cur\_idApp AND id\_comp = cur\_idComp

) THEN

IF EXISTS (

SELECT 1 FROM Questionnaires

WHERE idQuest = cur\_idQuest AND id\_comp IS NOT NULL

) THEN

SET notes\_count = (

SELECT COUNT(noteQApp)

FROM QUEST\_APP

WHERE id\_app = cur\_idApp AND id\_comp = cur\_idComp

);

SET avg\_note = (

SELECT AVG(noteQApp)

FROM QUEST\_APP

WHERE id\_app = cur\_idApp AND id\_comp = cur\_idComp

);

IF (notes\_count = 2 AND avg\_note >= 12) OR

(notes\_count = 3 AND avg\_note >= 10 AND NOT EXISTS (

SELECT 1

FROM QUEST\_APP

WHERE id\_app = cur\_idApp AND id\_comp = cur\_idComp AND Termine = 1 AND noteQApp < 8

)) OR

(notes\_count >= 4 AND avg\_note >= 10) THEN

CALL AffecterCompétenceApprenant(cur\_idApp, cur\_idComp);

END IF;

END IF;

END IF;

END LOOP;

CLOSE cur\_curseur;

END $$

DELIMITER ;

**EXERCICE 10**

DELIMITER //

CREATE OR REPLACE TRIGGER Verifier\_Creation\_Questionnaire

BEFORE INSERT ON questionnaires

FOR EACH ROW

BEGIN

DECLARE competenceExiste INT;

SELECT COUNT(\*) INTO competenceExiste

FROM competences

WHERE id\_comp = NEW.id\_comp;

IF competenceExiste = 0

THEN SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'La compétence associée au questionnaire n\'existe pas';

END IF;

END //

DELIMITER ;

**EXERCICE 11**

DELIMITER $$

CREATE OR REPLACE TRIGGER Vérifier\_MAJ\_Questionnaire BEFORE UPDATE ON

Questionnaires FOR EACH ROW

BEGIN

DECLARE

compter INT ; IF NEW.id\_comp IS NOT NULL AND NEW.id\_comp != OLD.id\_comp THEN

SELECT COUNT(\*) INTO compter

FROM Questions

WHERE Questions.idQuest = NEW.idQuest AND Questions.idDomaine NOT IN(

SELECT idDomaine

FROM Domaines

JOIN EQUIVALOIR ON Domaines.idDomaine = EQUIVALOIR.id\_domaine

WHERE EQUIVALOIR.id\_comp = NEW.id\_comp

);

IF compter > 0 OR NEW.id\_comp IS NULL THEN SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'maj refusée' ;

END IF ;

END IF ;

END $$

**Exercice 12**

DELIMITER //

CREATE or REPLACE TRIGGER VerifierQuestionEtCompetences

BEFORE UPDATE ON Questions

FOR EACH ROW

BEGIN

DECLARE domaine\_id INT;

DECLARE questionnaire\_id INT;

DECLARE competence\_id INT;

DECLARE compatible BOOLEAN;

SELECT NEW.idDomaine INTO domaine\_id;

SELECT idQuest INTO questionnaire\_id FROM Composer WHERE idQuestion = NEW.idQuestion;

SELECT id\_comp INTO competence\_id FROM Questionnaires WHERE idQuest = questionnaire\_id;

SELECT COUNT(\*) INTO compatible FROM EQUIVALOIR WHERE id\_domaine = domaine\_id AND id\_comp = competence\_id;

IF compatible = 0 THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'La mise à jour est refusée. Le domaine à associer à la question n''est pas compatible avec les compétences des questionnaires associés.';

END IF;

END //

DELIMITER ;

**Exercice 13**

SELECT formateurs.nomFor AS Nomformateur,

competences.nomComp AS competence\_principale,

COUNT(questionnaires.idQuest) AS nb\_questionnaires\_associés

FROM formateurs

JOIN questionnaires ON formateurs.idFor = questionnaires.idFor

JOIN competences ON questionnaires.id\_comp = competences.id\_comp

GROUP BY formateurs.nomFor

ORDER BY nb\_questionnaires\_associés DESC;

**Exercice 14**

SELECT (

SELECT formateurs.nomFor

FROM formateurs

JOIN questions ON questions.idFor = formateurs.idFor

ORDER BY questions.idDomaine

LIMIT 1

) AS MeilleurFormateurDomaine,

(

SELECT formateurs.nomFor

FROM formateurs

JOIN questions ON questions.idFor = formateurs.idFor

JOIN domaines ON domaines.idDomaine = questions.idDomaine

JOIN equivaloir ON equivaloir.id\_domaine = domaines.idDomaine

WHERE equivaloir.id\_comp = C.id\_comp

GROUP BY formateurs.nomFor

ORDER BY COUNT(\*) DESC

LIMIT 1

) AS MeilleurFormateurCompétence

FROM Competences C

LIMIT 1;

**Exercice 15**

SELECT

competences.nomComp,

COUNT(DISTINCT questionnaires.idFor) AS NbFormateurs,

COUNT(DISTINCT quest\_app.id\_app) AS NbApprenants

FROM competences

LEFT JOIN questionnaires ON questionnaires.id\_comp = competences.id\_comp

LEFT JOIN composer ON composer.idQuest = questionnaires.idQuest

LEFT JOIN quest\_app ON quest\_app.idQuest = questionnaires.idQuest

WHERE dureeQuest IS NOT NULL

GROUP BY competences.nomComp;

**Exercice 16**

SELECT Competences.nomComp, COUNT(DISTINCT Questionnaires.idFor) AS nbFormateurs, COUNT(DISTINCT Questionnaires.idQuest) AS nbExamens, COUNT(DISTINCT QUEST\_APP.id\_app) AS nbApp, AVG(CASE WHEN QUEST\_APP.Termine = 1 THEN QUEST\_APP.noteQApp END) AS NoteMoy

FROM Competences

LEFT JOIN Questionnaires ON Competences.id\_comp = Questionnaires.id\_comp

LEFT JOIN QUEST\_APP ON Questionnaires.idQuest = QUEST\_APP.idQuest

GROUP BY Competences.id\_comp, Competences.nomComp

ORDER BY Competences.id\_comp;

**Exercice 17**

SELECT competences.nomComp,

(COUNT(DISTINCT apprenants.idApp) \* 1.0) / NULLIF(COUNT(DISTINCT qa.id\_app), 0) AS Ratio

FROM

Competences

LEFT JOIN

acquerir ON competences.id\_comp = acquerir.id\_comp

LEFT JOIN

Apprenants ON acquerir.id\_app = apprenants.idApp

LEFT JOIN

(SELECT id\_app FROM QUEST\_APP GROUP BY id\_app HAVING COUNT(\*) >= 2) qa ON apprenants.idApp = qa.id\_app

GROUP BY

competences.nomComp;

**Exercice 18**

(SELECT competences.nomComp, AVG(quest\_app.noteQApp) AS moy

FROM competences

JOIN questionnaires ON competences.id\_comp = questionnaires.id\_comp

JOIN quest\_app ON questionnaires.idQuest = quest\_app.idQuest

GROUP BY competences.nomComp

ORDER BY moy DESC LIMIT 1)

UNION

(SELECT competences.nomComp, COUNT(quest\_app.noteQApp) AS nbnote

FROM competences

JOIN questionnaires ON competences.id\_comp = questionnaires.id\_comp

JOIN quest\_app ON questionnaires.idQuest = quest\_app.idQuest

GROUP BY competences.nomComp

ORDER BY nbnote DESC LIMIT 1)

UNION

(SELECT competences.nomComp, AVG(quest\_app.noteQApp) AS moy

FROM competences

JOIN questionnaires ON competences.id\_comp = questionnaires.id\_comp

JOIN quest\_app ON questionnaires.idQuest = quest\_app.idQuest

GROUP BY competences.nomComp

ORDER BY moy ASC LIMIT 1);

**Exercice 19**

SELECT id\_domaine, COUNT(\*) AS nb\_compétences

FROM EQUIVALOIR

GROUP BY id\_domaine

HAVING COUNT(\*) > 1;

**Exercice 20**

SELECT Domaines.idDomaine, Domaines.nomDomaine

FROM Domaines

JOIN EQUIVALOIR ON Domaines.idDomaine = EQUIVALOIR.id\_domaine

GROUP BY Domaines.idDomaine, Domaines.nomDomaine

HAVING COUNT(EQUIVALOIR.id\_comp) = 1;

**Exercice 21**

SELECT domaines.nomDomaine,COUNT(competences.nomComp)

FROM domaines

INNER JOIN equivaloir ON equivaloir.id\_domaine = domaines.idDomaine

INNER JOIN competences on competences.id\_comp = equivaloir.id\_comp

GROUP BY domaines.nomDomaine ;

**Exercice 22**

SELECT

CASE WHEN (SELECT AVG(quest\_app.noteQApp) FROM quest\_app

INNER JOIN questionnaires ON questionnaires.idQuest = quest\_app.idQuest

WHERE questionnaires.dureeQuest IS NOT NULL

AND questionnaires.id\_comp IS NOT NULL)

>

(SELECT AVG(quest\_app.noteQApp) FROM quest\_app

INNER JOIN questionnaires ON questionnaires.idQuest = quest\_app.idQuest

WHERE questionnaires.dureeQuest IS NOT NULL

AND questionnaires.id\_comp IS NULL) THEN 'COMPETENCE'

WHEN (SELECT AVG(quest\_app.noteQApp) FROM quest\_app

INNER JOIN questionnaires ON questionnaires.idQuest = quest\_app.idQuest

WHERE questionnaires.dureeQuest IS NOT NULL

AND questionnaires.id\_comp IS NOT NULL)

<

(SELECT AVG(quest\_app.noteQApp) FROM quest\_app

INNER JOIN questionnaires ON questionnaires.idQuest = quest\_app.idQuest

WHERE questionnaires.dureeQuest IS NOT NULL

AND questionnaires.id\_comp IS NULL) THEN 'GENERALISTE'

ELSE 'Les notes sont aussi élevées'

END AS 'resultat'

FROM quest\_app;

**Exercice 23**

SELECT competences.nomComp AS competence,

formateurs.nomFor AS FormateurPrincipale

FROM formateurs

JOIN questionnaires ON formateurs.idFor = questionnaires.idFor

JOIN competences ON questionnaires.id\_comp = competences.id\_comp

GROUP BY competences.nomComp;

**Exercice 24**

SELECT

CASE WHEN (SELECT COUNT(quest\_app.noteQApp) FROM quest\_app

INNER JOIN questionnaires ON questionnaires.idQuest = quest\_app.idQuest

WHERE questionnaires.dureeQuest IS NOT NULL

AND questionnaires.id\_comp IS NOT NULL)

>

(SELECT COUNT(quest\_app.noteQApp) FROM quest\_app

INNER JOIN questionnaires ON questionnaires.idQuest = quest\_app.idQuest

WHERE questionnaires.dureeQuest IS NOT NULL

AND questionnaires.id\_comp IS NULL) THEN 'COMPETENCE'

WHEN (SELECT COUNT(quest\_app.noteQApp) FROM quest\_app

INNER JOIN questionnaires ON questionnaires.idQuest = quest\_app.idQuest

WHERE questionnaires.dureeQuest IS NOT NULL

AND questionnaires.id\_comp IS NOT NULL)

<

(SELECT COUNT(quest\_app.noteQApp) FROM quest\_app

INNER JOIN questionnaires ON questionnaires.idQuest = quest\_app.idQuest

WHERE questionnaires.dureeQuest IS NOT NULL

AND questionnaires.id\_comp IS NULL)

THEN 'GENERALISTE'

ELSE 'Il y a autant d\'examen fait des 2 côtés'

END AS 'resultat'

FROM quest\_app;

**Exercice 25**

SELECT apprenants.idApp, apprenants.nomApp

FROM apprenants

JOIN quest\_app ON apprenants.idApp = quest\_app.id\_app

JOIN questionnaires ON quest\_app.idQuest = questionnaires.idQuest

LEFT JOIN (

SELECT id\_app, id\_comp, COUNT(\*) AS nbexam

FROM quest\_app

JOIN questionnaires ON quest\_app.idQuest = questionnaires.idQuest

WHERE questionnaires.id\_comp IS NOT NULL

GROUP BY id\_app, id\_comp

) AS nbexamcomp ON quest\_app.id\_app = nbexamcomp.id\_app

WHERE nbexamcomp.nbexam = 1;

**EXERCICE 26**

SELECT apprenants.idApp,

apprenants.nomApp

FROM apprenants

JOIN quest\_app ON apprenants.idApp = quest\_app.id\_app

JOIN questionnaires ON quest\_app.idQuest = questionnaires.idQuest

JOIN competences ON competences.id\_comp=questionnaires.id\_comp

LEFT JOIN (

SELECT id\_app, id\_comp, COUNT(\*) AS nbexam

FROM quest\_app

JOIN questionnaires ON quest\_app.idQuest = questionnaires.idQuest

WHERE questionnaires.id\_comp IS NOT NULL

GROUP BY id\_app, id\_comp

) AS nbexamcomp ON quest\_app.id\_app = nbexamcomp.id\_app

WHERE nbexamcomp.nbexam = 1;

**EXERCICE 27**

SELECT apprenants.idApp, apprenants.nomApp, competences.nomComp

FROM apprenants

JOIN quest\_app ON apprenants.idApp = quest\_app.id\_app

JOIN questionnaires ON quest\_app.idQuest = questionnaires.idQuest

JOIN competences ON competences.id\_comp = questionnaires.id\_comp

LEFT JOIN (

SELECT id\_app, id\_comp, COUNT(\*) AS nbexam

FROM quest\_app

JOIN questionnaires ON quest\_app.idQuest = questionnaires.idQuest

WHERE questionnaires.id\_comp IS NOT NULL

GROUP BY id\_app, id\_comp

) AS nbexamcomp ON quest\_app.id\_app = nbexamcomp.id\_app

LEFT JOIN (

SELECT id\_app, questionnaires.id\_comp, questionnaires.nomQuest

FROM quest\_app

JOIN questionnaires ON quest\_app.idQuest = questionnaires.idQuest

WHERE quest\_app.Termine = 1

) AS new\_exam ON apprenants.idApp = new\_exam.id\_app AND competences.id\_comp = new\_exam.id\_comp

WHERE nbexamcomp.nbexam = 1;

**EXERCICE 28**

SELECT apprenants.idApp, apprenants.nomApp, competences.nomComp,((nbexamcomp.nbexam+1)\*12)-nbexamcomp.note as NoteMin

FROM apprenants

JOIN quest\_app ON apprenants.idApp = quest\_app.id\_app

JOIN questionnaires ON quest\_app.idQuest = questionnaires.idQuest

JOIN competences ON competences.id\_comp = questionnaires.id\_comp

LEFT JOIN (

SELECT id\_app, id\_comp, COUNT(\*) AS nbexam,quest\_app.noteQApp as note

FROM quest\_app

JOIN questionnaires ON quest\_app.idQuest = questionnaires.idQuest

WHERE questionnaires.id\_comp IS NOT NULL

GROUP BY id\_app, id\_comp

) AS nbexamcomp ON quest\_app.id\_app = nbexamcomp.id\_app

LEFT JOIN (

SELECT id\_app, questionnaires.id\_comp, questionnaires.nomQuest

FROM quest\_app

JOIN questionnaires ON quest\_app.idQuest = questionnaires.idQuest

WHERE quest\_app.Termine = 1

) AS new\_exam ON apprenants.idApp = new\_exam.id\_app AND competences.id\_comp = new\_exam.id\_comp

WHERE nbexamcomp.nbexam = 1;