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

**1 Luca F Vert**

**1:**

**CONSTRAINT FK\_posts\_THREADS FOREIGN KEY (idThread)**

**REFERENCES Threads(idThread) ON UPDATE CASCADE,**

La contrainte vérifie que quand on fait un INSERT pour un un post il doit être lié à un thread existant.

**INSERT INTO POSTS (idPost, datePost, titrePost, textePost, idThread, idApp, idFor)**

**VALUES (1, '06-03-2024', 'Un titre', 'plein de mots', 999 (un id qui existe pas), 1, NULL);**

L’INSERT ne sera pas accepté car l’idThread n’existe pas.

**2:**

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

La contrainte met la valeur boolean de Termine a False (0) si elle n’est pas renseigné, elle ne peut pas non plus être nul a cause de de la contrainte NOT NULL.

Et enfin la contrainte CHECK permet de vérifier si la valeur rentré est bien 0 ou 1.

**INSERT INTO QUEST\_APP (idQApp, appreciationQApp, noteQApp, dateQApp, Termine, idQuest, id\_app)**

**VALUES (1, 'Bien', 17, '06-03-2024', 2, 3, 4);**

Cette requête sera refusée car Termine est égale à 2 ou le check vérifie que la valeur soit bien 1 ou 0.

**3:**

**CONSTRAINT Pk\_id\_domaine PRIMARY KEY(idDomaine)**

La contrainte PRIMARY KEY garantit que chaque valeur dans cette colonne est unique et non nulle.

**INSERT INTO Domaines (idDomaine, nomDomaine) VALUES (1, 'Informatique');**

**INSERT INTO Domaines (idDomaine, nomDomaine) VALUES (1, 'Biologie');**

Cette requête sera refusée car idDomaine est égale à 1 dans les deux cas où une clé primaire doit être unique.

**2 Emilien Bleu)**

Dans cette table il y a plusieurs règles de gestion absente qui mettent en péril l’intégrité et la cohérence de la table.

Premièrement, Un poste peut être poster par un apprenant ou un formateur, mais il n’y a pas de règle de gestion qui empêche un poste d’avoir été posté par un formateur et un apprenant en même temps.

Exemple:

INSERT INTO posts VALUES (

1, '2022-06-12', 'titre', 'texte', 1, **1, 1**);

Ensuite, les dates de créations ne sont pas protégé ce qui fait que l'on peut créer des threads ou des posts à des dates futur, de plus rien n'empêche dans l’état actuelle la création d’un poste avant la création d’un thread temporellement et factuellement:

INSERT INTO posts VALUES (1, '2027-12-12', 'titre', 'texte', 1, 1, 0)

INSERT INTO THREADS VALUES (1,’'2027-08-12'’ )

**3 Lucas V BLEU**

DROP FUNCTION IF EXISTS PrctTermine;

DELIMITER $$

CREATE FUNCTION PrctTermine(id\_Q\_app INT)

RETURNS INT

BEGIN

DECLARE Pourc INT;

DECLARE REP INT;

DECLARE TOTAL INT;

SET REP = (SELECT COUNT(DISTINCT ID\_QUESTION\_APP) FROM Repondre

NATURAL JOIN QUEST\_APPS

INNER JOIN REPONSES ON idRep = id\_rep

WHERE ID\_QUESTION\_APP = id\_Q\_app);

SET TOTAL = (SELECT COUNT(DISTINCT id\_question) FROM Composer

INNER JOIN QUESTIONNAIRES ON ID\_Q = ID\_QUESTIONNAIRE

INNER JOIN QUEST\_APPS ON id\_questionnaire = id\_Q

WHERE ID\_QUESTION\_APP = id\_Q\_app ;

SET Pourc = REP / TOTAL;

RETURN Pourc;

END $$

DELIMITER ;

**4 Mohamed El salah aissaoui bleu :**

DELIMITER //

CREATE OR REPLACE PROCEDURE Avancement (

IN p\_IdQuest INT,

OUT p\_NbQuestionsRestantes INT,

OUT p\_TempsRestant TIME

)

BEGIN

DECLARE total\_questions INT;

DECLARE questions\_repondues INT;

DECLARE duree\_quest TIME;

DECLARE temps\_ecoule TIME;

SELECT COUNT(\*) INTO total\_questions

FROM Composer

WHERE idQuest = p\_IdQuest;

SELECT COUNT(\*) INTO questions\_repondues

FROM REPONDRE

WHERE idQApp = p\_IdQuest;

SELECT dureeQuest INTO duree\_quest

FROM Questionnaires

WHERE idQuest = p\_IdQuest;

SET temps\_ecoule = (SELECT SEC\_TO\_TIME(SUM(TIME\_TO\_SEC(duree)))

FROM REPONDRE

WHERE idQApp = p\_IdQuest);

SET p\_TempsRestant = TIMEDIFF(duree\_quest, temps\_ecoule);

SET p\_NbQuestionsRestantes = total\_questions - questions\_repondues;

END //

DELIMITER ;

**5 Lucas V VERT**

DROP FUNCTION IF EXISTS SuccesQuestionnaire

DELIMITER $$

CREATE FUNCTION SuccesQuestionnaire(IdQuest INT)

RETURNS INT;

BEGIN

DECLARE NB INT;

SET NB = (SELECT COUNT(DISTINCT ID\_APP) FROM REPONDRE);

END $$

DELIMITER ;

**6 Lucas V ROUGE**

DROP FUNCTION IF EXISTS NoteQuestionnaire

DELIMITER $$

CREATE FUNCTION NoteQuestionnaire(ID\_Q\_APP\_Select INT)

RETURNS INT

BEGIN

DECLARE Note INT;

SET Note = (SELECT noteAppQ FROM QUEST\_APPS

WHERE ID\_Q\_APP = ID\_Q\_APP\_Select);

END $$

DEMILITER ;

**7 DJAOUEL Jenna BLEU**

DROP PROCEDURE IF EXISTS CalculNotesApp;

DELIMITER //

CREATE PROCEDURE CalculNotesApp(IN idApp INT)

BEGIN

DECLARE noteCon FLOAT;

DECLARE noteApp FLOAT;

SELECT AVG(quest\_app.noteQApp) INTO noteCon

FROM quest\_app

WHERE quest\_app.id\_app = idApp AND quest\_app.idQuest NOT IN

(SELECT questionnaires.idQuest

FROM questionnaires

WHERE questionnaires.dureeQuest IS NOT NULL);

SELECT AVG(quest\_app.noteQApp) INTO noteApp

FROM quest\_app

WHERE quest\_app.id\_app = idApp

AND quest\_app.idQuest IN

(SELECT questionnaires.idQuest

FROM questionnaires

WHERE questionnaires.dureeQuest IS NULL);

UPDATE apprenants

SET apprenants.noteApp = noteApp, apprenants.NoteConn = noteCon

WHERE idApp = idApp;

END //

DELIMITER ;

**8 Mohamed El Salah aissaoui ROUGE**

DELIMITER //

CREATE OR REPLACE PROCEDURE Progression (

IN p\_idApp INT

)

BEGIN

DECLARE done BOOLEAN DEFAULT FALSE;

DECLARE prev\_note INT;

DECLARE current\_note INT;

DECLARE trend VARCHAR(20);

DECLARE cur\_quest CURSOR FOR

SELECT idQuest, noteQApp

FROM QUEST\_APP

WHERE id\_app = p\_idApp

ORDER BY dateQApp;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

DROP TEMPORARY TABLE IF EXISTS temp\_progression;

CREATE TEMPORARY TABLE temp\_progression (

idQuest INT,

noteQApp INT

);

OPEN cur\_quest;

FETCH cur\_quest INTO prev\_note, current\_note; -- Fetch both variables

read\_loop: LOOP

IF done THEN

LEAVE read\_loop;

END IF;

IF prev\_note < current\_note THEN

SET trend = 'Amélioration';

ELSEIF prev\_note > current\_note THEN

SET trend = 'Dégradation';

ELSE

SET trend = 'Stable';

END IF;

INSERT INTO temp\_progression (idQuest, noteQApp)

VALUES (prev\_note, current\_note);

SET prev\_note = current\_note;

END LOOP;

CLOSE cur\_quest;

SELECT \* FROM temp\_progression;

END //

DELIMITER ;

**9 mohamed el salah VERT**

DELIMITER $$

CREATE or REPLACE PROCEDURE CloturerQuestionnaireDate()

BEGIN

UPDATE QUEST\_APP

SET Termine = 1

WHERE date\_add(dateQApp, INTERVAL dureeQuest HOUR) <= NOW()

AND Termine = 0;

END$$

DELIMITER ;

DELIMITER $$

CREATE EVENT IF NOT EXISTS CloturerQuestionnaireEvent

ON SCHEDULE EVERY 1 HOUR

DO

BEGIN

CALL CloturerQuestionnaireDate();

END$$

DELIMITER ;

**10 Emilien Rouge**

DELIMITER $$

CREATE OR REPLACE EVENT CloturerNoteQuestionnaire

ON SCHEDULE EVERY 1 HOUR

STARTS CURRENT\_TIMESTAMP

DO

BEGIN

UPDATE quest\_app

LEFT JOIN repondre ON quest\_app.idQApp = repondre.idQApp

LEFT JOIN reponses ON repondre.idRep = reponses.idRep

SET quest\_app.noteQApp =

CASE

WHEN reponses.idRep = quest\_app.idRep THEN quest\_app.noteQApp + 1

WHEN reponses.idRep IS NULL THEN quest\_app.noteQApp - 1

ELSE quest\_app.noteQApp - 2

END

WHERE quest\_app.Termine = 1;

END $$

DELIMITER ;

**11 Mohamed El Salah Aissaoui BLEU**

DELIMITER

$$

CREATE or REPLACE PROCEDURE CloturerQuestionnaireReponses()

BEGIN

UPDATE

Questionnaires AS q

SET

q.Termine = 1

WHERE NOT

EXISTS(

SELECT

1

FROM

Questions AS qu

LEFT JOIN Composer AS c

ON

qu.idQuestion = c.idQuestion

WHERE

c.idQuest = q.idQuest AND NOT EXISTS(

SELECT

1

FROM

Repondre AS r

WHERE

r.idQuestion = qu.idQuestion

)

) ; END $$

DELIMITER

;

**12 DJAOUEL Jenna VERT**

DROP VIEW IF EXISTS VueStatsApprenant\_1;

CREATE VIEW VueStatsApprenant\_1 AS

SELECT

apprenants.idApp, apprenants.nomApp,

COUNT(quest\_app.idQApp) - COUNT(

CASE WHEN quest\_app.Termine = FALSE

THEN quest\_app.idQApp END)

AS NbExamTermines,

COUNT(CASE WHEN quest\_app.Termine = FALSE

THEN quest\_app.idQApp END)

AS NbExamENcours

FROM Apprenants

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

GROUP BY apprenants.idApp, apprenants.nomApp;

**13 Emilien Bleu**

DROP VIEW IF EXISTS VueStatsApprenant\_2;

CREATE VIEW VueStatsApprenant\_2 AS

SELECT

apprenants.idApp,

COUNT(CASE WHEN quest\_app.Termine = TRUE THEN quest\_app.idQapp END) AS nb\_questionnaires\_realises,

COUNT(CASE WHEN quest\_app.Termine = FALSE THEN quest\_app.idQapp END) AS nb\_questionnaires\_en\_cours

FROM

apprenants

LEFT JOIN

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

LEFT JOIN

repondre ON apprenants.idApp = repondre.IDAPP

LEFT JOIN

reponses ON repondre.IDREP = reponses.idRep

GROUP BY

apprenants.idApp;

**14 Emilien Orange**

DROP VIEW IF EXISTS VueStatsApprenant\_3;

CREATE VIEW VueStatsApprenant\_3 AS

SELECT

apprenants.idApp AS idApprenant,

apprenants.nomApp AS NomApprenant,

AVG(CASE WHEN quest\_app.Termine = 1 AND questionnaires.dureeQuest <> '30 00:00:00' THEN quest\_app.noteQApp END) AS MoyenneExamens,

AVG(CASE WHEN quest\_app.Termine = 1 AND questionnaires.dureeQuest = '30 00:00:00' THEN quest\_app.noteQApp END) AS MoyenneQCM

FROM

apprenants

LEFT JOIN

quest\_app ON Apprenants.idApp = quest\_app.id\_app

LEFT JOIN

questionnaires ON quest\_app.idQuest = questionnaires.idQuest

GROUP BY

apprenants.idApp, apprenants.nomApp;

**15 Lucas V ROUGE**

CREATE VIEW VueStatsApprenant\_4 AS

SELECT

SUM(CASE WHEN quest\_apps.Termine = TRUE THEN

CASE WHEN reponses.bonneRep = TRUE THEN 1 ELSE 0 END

ELSE 0 END) \* 100.0 /

SUM(CASE WHEN quest\_apps.Termine = TRUE THEN 1 ELSE 0 END) AS pourcentage\_global\_bonnes\_reponses

FROM

apprenants

LEFT JOIN

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

LEFT JOIN

repondre ON apprenants.idApp = repondre.ID\_APP

LEFT JOIN

reponses ON repondre.ID\_REP = reponses.idRep

GROUP BY

apprenants.idApp;

**16 Luca F** ????????????????????????????????????????????????

CREATE VIEW VueNotesChrono AS

SELECT id\_app AS idApprenants,

noteQApp AS Note,

dateQApp AS DateNote,

'Questionnaire' AS 'QCM/examen'

FROM QUEST\_APP

UNION ALL

SELECT idFor AS idFormateur,

dateQuest AS DateNote,

NULL AS Note,

'Questionnaire' AS 'QCM/examen'

FROM Questionnaires

ORDER BY DateNote;

**17 Oscar Vert**

CREATE VIEW VueStatsQuestionnaire\_1 AS

SELECT Questionnaires.idQuest AS idQuestionnaire, Questionnaires.nomQuest AS designation,

CASE

WHEN EXISTS (SELECT 1 FROM Composer WHERE Composer.idQuest = Questionnaires.idQuest) THEN 'QCM'

ELSE 'EXAMEN'

END AS typeQuestionnaire,

COUNT(Questions.idQuestion) AS nbQuestions

FROM Questionnaires

LEFT JOIN Composer ON Questionnaires.idQuest = Composer.idQuest

LEFT JOIN Questions ON Composer.idQuestion = Questions.idQuestion

GROUP BY Questionnaires.idQuest, Questionnaires.nomQuest;

**18 DJAOUEL Jenna BLEU**

CREATE VIEW VueStatsQuestionnaire\_2 AS

SELECT

questionnaires.idQuest AS idQuestionnaire,

questionnaires.nomQuest AS designation,

CASE

WHEN questionnaires.dureeQuest IS NOT NULL

THEN 'EXAMEN'

ELSE 'QCM'

END AS type,

SUM(CASE WHEN quest\_app.Termine = 1

THEN 1 ELSE 0 END)

AS AppFini,

SUM(CASE WHEN quest\_app.Termine = 0

THEN 1 ELSE 0 END)

AS AppEnCours

FROM questionnaires

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

GROUP BY questionnaires.idQuest;

**19 Luca F**

CREATE VIEW VueStatsQuestion\_1 AS

SELECT

Q.idQuestion,

Q.textQuestion AS Intitule,

COUNT(R.idRep) AS NombreReponses,

SUM(R.bonneRep) AS NombreBonnesReponses

FROM

Questions Q

LEFT JOIN

Reponses R ON Q.idQuestion = R.idQuestion

GROUP BY

Q.idQuestion, Q.textQuestion;

**20 Oscar Orange**

CREATE VIEW VueStatsQuestion\_2 AS

SELECT Questions.idQuestion, Questions.textQuestion, COUNT(DISTINCT QUEST\_APP.idQApp) AS NombreQuestionnaires, AVG(QUEST\_APP.noteQApp) AS NoteMoyenne

FROM Questions

JOIN QUEST\_APP ON Questions.idQuestion = QUEST\_APP.idQuest

GROUP BY Questions.idQuestion, Questions.textQuestion;

**21 Lucas V VERT**

DELIMITER $$

CREATE TRIGGER VerifierInsertionRéponse

AFTER INSERT ON REPONSES

FOR EACH ROW

BEGIN

DECLARE id\_Q\_app\_check INT;

DECLARE Termine\_check BOOLEAN;

DECLARE Pourc FLOAT;

SET id\_Q\_app\_check = (SELECT id\_questionnaire FROM QUEST\_APPS WHERE id\_Q\_app = NEW.question);

IF id\_Q\_app\_check IS NOT NULL THEN

SELECT Termine INTO Termine\_check FROM QUEST\_APPS WHERE id\_Q\_app = id\_Q\_app\_check;

IF Termine\_check = 0 THEN

SET Pourc = PrctTermine(id\_Q\_app\_check);

IF Pourc = 100 THEN

UPDATE QUEST\_APPS SET Termine = 1 WHERE id\_Q\_app = id\_Q\_app\_check;

IF (SELECT DUREE\_QUEST FROM QUESTIONNAIRES

WHERE id\_Q\_app\_check = id\_questionnaire) <> 30 THEN

CALL NoteQuestionnaire(id\_Q\_app\_check);

END IF;

END IF;

END IF;

END IF;

END$$

DELIMITER ;

**22 Emilien Vert**

DELIMITER $$

CREATE OR REPLACE TRIGGER VerifierMiseAJourReponse

AFTER UPDATE ON reponses

FOR EACH ROW

BEGIN

DECLARE id\_Q\_app\_check INT;

DECLARE Termine\_check BOOLEAN;

DECLARE QCM\_check TIME;

SELECT questionnaires.idQuest INTO id\_Q\_app\_check FROM quest\_app WHERE quest\_app.idQApp = NEW.idQApp;

IF id\_Q\_app\_check IS NOT NULL THEN

SELECT quest\_app.Termine INTO Termine\_check FROM quest\_app WHERE quest\_app.idQApp = NEW.idQApp;

IF Termine\_check = 0 THEN

SELECT questionnaires.dureeQuest INTO QCM\_check FROM questionnaires WHERE questionnaires.idQuest = id\_Q\_app\_check;

IF QCM\_check IS NOT NULL THEN

UPDATE quest\_app SET quest\_app.noteQApp = NULL WHERE quest\_app.idQApp = NEW.idQApp;

END IF;

END IF;

END IF;

END $$

DELIMITER ;

**23 Emilien Orange**

DELIMITER $$

CREATE OR REPLACE TRIGGER VerifierQuestionnaireActivé

AFTER UPDATE ON questionnaires

FOR EACH ROW

BEGIN

IF NEW.dateQuest IS NOT NULL THEN

DECLARE nbQuest INT;

DECLARE nbRep INT;

DECLARE nbBonneRep INT;

SELECT COUNT(\*) INTO nbQuest FROM questions WHERE quest\_app.idQuest = NEW.idQuest;

IF nbQuest < 1 THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Le questionnaire doit avoir au moins une question.';

END IF;

DECLARE questionC CURSOR FOR

SELECT questions.idQuestion FROM questions WHERE quest\_app.idQuest = NEW.idQuest;

OPEN questionC;

question\_loop: LOOP

FETCH questionC INTO iqQuestion;

IF nbQuest = 0 THEN

LEAVE question\_loop;

END IF;

SELECT COUNT(\*) INTO nbRep FROM reponses WHERE idQuestion = idQuestion AND bonneRep = 1;

IF nbBonneRep < 1 THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Chaque question doit avoir au moins une bonne réponse.';

END IF;

END LOOP;

CLOSE questionC;

END IF;

END $$

DELIMITER ;

**24 Oscar Bleu**

DELIMITER $$

CREATE TRIGGER CheckQuestionnaireAnswers

BEFORE INSERT ON QUEST\_APP

FOR EACH ROW

BEGIN

DECLARE ComptageQuestion INT;

SELECT COUNT(\*)

INTO ComptageQuestion

FROM Reponses

WHERE idQuestion = NEW.idQuest;

IF ComptageQuestion < 1 THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Chaque question doit avoir une réponse.';

END IF;

END $$

DELIMITER ;

**25 Oscar Bleu**

SELECT typeQuestionnaire, COUNT(\*) AS nombreQuestionnaires, ROUND((COUNT(\*) \* 100.0) / (SELECT COUNT(\*) FROM Questionnaires), 2) AS pourcentage

FROM (

SELECT

Questionnaires.idQuest,

CASE

WHEN EXISTS (SELECT 1 FROM Composer WHERE Composer.idQuest = Questionnaires.idQuest) THEN 'QCM'

ELSE 'EXAMEN'

END AS typeQuestionnaire

FROM Questionnaires

LEFT JOIN Composer ON Questionnaires.idQuest = Composer.idQuest) AS types

GROUP BY typeQuestionnaire;

**26 Oscar Vert**

SELECT Questionnaires.idQuest, Questionnaires.nomQuest, COUNT(REPONDRE.idApp) AS nbUtilisations

FROM Questionnaires

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

JOIN REPONDRE ON QUEST\_APP.idQApp = REPONDRE.idQApp

GROUP BY Questionnaires.idQuest, Questionnaires.nomQuest

ORDER BY nbUtilisations DESC

LIMIT 3;

**27 mohamed el salah aissaoui Orange**

SELECT d.nomDomaine AS Domaine,

COUNT(DISTINCT q.idQuest) AS Nombre\_Questionnaires,

SUM(CASE WHEN c.idQuestion IS NOT NULL THEN 1 ELSE 0 END) AS Nombre\_Examens\_QCM,

(SELECT COUNT(\*)

FROM QUEST\_APP qa

WHERE qa.idQuest = q.idQuest

GROUP BY qa.idQuest

ORDER BY COUNT(\*) DESC

LIMIT 1) AS Nombre\_Utilisations\_Max

FROM Domaines d

INNER JOIN Threads t ON d.idDomaine = t.idDomaine

INNER JOIN POSTS p ON t.idThread = p.idThread

INNER JOIN QUEST\_APP qa ON p.idPost = qa.idQApp

INNER JOIN Questionnaires q ON qa.idQuest = q.idQuest

LEFT JOIN Composer c ON q.idQuest = c.idQuest

GROUP BY d.idDomaine

ORDER BY COUNT(\*) DESC

LIMIT 3;

**28 DJAOUEL Jenna ROUGE**

SELECT

apprenants.idApp, apprenants.nomApp,

COUNT(DISTINCT questions.idQuestion) AS TotalQuestions,

COUNT(DISTINCT reponses.idRep) AS QuestRepondues,

(COUNT(DISTINCT questions.idQuestion) - COUNT(DISTINCT reponses.idRep)) \* 100 / COUNT(DISTINCT questions.idQuestion)

AS PourcentageSansReponse

FROM apprenants

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

LEFT JOIN composer ON composer.idQuest = quest\_app.idQApp

LEFT JOIN questions ON questions.idQuestion = composer.idQuestion

LEFT JOIN reponses ON reponses.idQuestion = questions.idQuestion

GROUP BY apprenants.nomApp;

**29 DJAOUEL Jenna ORANGE**

SELECT composer.idQuest,

COUNT(DISTINCT questions.idQuestion) AS TotalQuestion,

COUNT(DISTINCT reponses.idRep) AS QuestRepondues,

(COUNT(DISTINCT questions.idQuestion) - COUNT(DISTINCT reponses.idRep)) \* 100 / COUNT(DISTINCT questions.idQuestion) AS PrctgQuestSansRep

FROM composer

LEFT JOIN questions ON questions.idQuestion = composer.idQuestion

LEFT JOIN reponses ON reponses.idQuestion = questions.idQuestion

GROUP BY composer.idQuest;

**30 Oscar Rouge**

WITH ClassementEtudiants AS (

SELECT Apprenants.idApp, Apprenants.nomApp, QUEST\_APP.noteQApp, ROW\_NUMBER() OVER (PARTITION BY Apprenants.idApp ORDER BY QUEST\_APP.noteQApp DESC) AS ClassementNote

FROM Apprenants

JOIN QUEST\_APP ON Apprenants.idApp = QUEST\_APP.id\_app)

SELECT ClassementEtudiants.idApp, ClassementEtudiants.nomApp AS NomEtudiant, ClassementEtudiants.noteQApp AS MeilleureNote

FROM ClassementEtudiants

WHERE ClassementEtudiants.ClassementNote <= 3

ORDER BY ClassementEtudiants.noteQApp DESC;

**31 DJAOUEL Jenna VERT**

SELECT domaines.nomDomaine, COUNT(questionnaires.idQuest) AS

NbQuestionnaireProposes

FROM domaines

JOIN questions ON domaines.idDomaine = questions.idDomaine

JOIN formateurs ON formateurs.idFor = questions.idFor

JOIN questionnaires ON questionnaires.idFor = formateurs.idFor

GROUP BY domaines.nomDomaine

ORDER BY NbQuestionnaireProposes DESC

LIMIT 2;

32 Emilien VERT

SELECT domaines.nomDomaine, COUNT(threads.idThread) AS ActiviteForum

FROM domaines

JOIN threads ON domaines.idDomaine = threads.idDomaine

GROUP BY domaines.nomDomaine

ORDER BY ActiviteForum DESC

LIMIT 2;

**33 DJAOUEL Jenna VERT**

SELECT Questionnaires.nomQuest,

COUNT(DISTINCT domaines.idDomaine) AS NbDomConcernes

FROM questionnaires

INNER JOIN composer ON questionnaires.idQuest = composer.idQuest

INNER JOIN questions ON questions.idQuestion = composer.idQuestion

INNER JOIN domaines ON domaines.idDomaine = questions.idDomaine

GROUP BY questionnaires.nomQuest;

**34 Lucas V VERT**

SELECT domaines.nomDomaine,COUNT(questionnaires.idQuest)

FROM domaines

INNER JOIN questions on questions.idDomaine=domaines.idDomaine

INNER JOIN composer on composer.idQuestion = questions.idQuestion

INNER JOIN questionnaires on questionnaires.idQuest = composer.idQuest

GROUP BY domaines.nomDomaine

**35 Luca F Orange**

SELECT

d.nomDomaine,

COUNT(composer.idQuest) AS nombre\_questionnaires

FROM

Domaines d

JOIN questions ON d.idDomaine = questions.idDomaine

JOIN composer ON questions.idQuestion = composer.idQuestion

GROUP BY d.nomDomaine

**36 Luca F Orange**

**SELECT q.nomQuest AS NomQuestionnaire, COUNT(DISTINCT cq.idQuestion) AS NombreQuestions,**

**GROUP\_CONCAT(DISTINCT d.nomDomaine SEPARATOR ', ') AS DomainesConcernes**

**FROM Questionnaires q**

**JOIN Composer c ON q.idQuest = c.idQuest**

**JOIN Questions cq ON c.idQuestion = cq.idQuestion**

**JOIN Domaines d ON cq.idDomaine = d.idDomaine**

**GROUP BY q.idQuest;**

**37 Luca F Rouge**

SELECT

apprenants.nomApp,

(

(

SELECT

apprenants.noteApp

FROM

apprenants

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

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

GROUP BY

apprenants.idApp

ORDER BY

questionnaires.dateQuest

DESC

LIMIT 1

) +(

SELECT

apprenants.noteApp

FROM

apprenants

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

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

GROUP BY

apprenants.idApp

ORDER BY

questionnaires.dateQuest

DESC

LIMIT 1 OFFSET 1

) +(

SELECT

apprenants.noteApp

FROM

apprenants

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

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

GROUP BY

apprenants.idApp

ORDER BY

questionnaires.dateQuest

DESC

LIMIT 1 OFFSET 2

)

) / 3 AS "Moyennes des 3 premières années",(

(

SELECT

apprenants.noteApp

FROM

apprenants

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

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

GROUP BY

apprenants.idApp

ORDER BY

questionnaires.dateQuest

LIMIT 1

) +(

SELECT

apprenants.noteApp

FROM

apprenants

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

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

GROUP BY

apprenants.idApp

ORDER BY

questionnaires.dateQuest

LIMIT 1 OFFSET 1

) +(

SELECT

apprenants.noteApp

FROM

apprenants

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

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

GROUP BY

apprenants.idApp

ORDER BY

questionnaires.dateQuest

LIMIT 1 OFFSET 2

)

) / 3 AS "Moyennes des 3 dernières années"

FROM

apprenants;

**39 mohamed El Salah Aissaoui Vert**

SELECT q.idQuest, q.nomQuest, COUNT(qu.idQuestion) AS NombreQuestions, AVG(qa.noteQApp) AS MoyenneNotes

FROM Questionnaires q

JOIN QUEST\_APP qa ON q.idQuest = qa.idQuest

JOIN Composer c ON q.idQuest = c.idQuest

JOIN Questions qu ON c.idQuestion = qu.idQuestion

WHERE qu.idDomaine = (SELECT idDomaine FROM Domaines WHERE nomDomaine = 'Informatique')

GROUP BY q.idQuest, q.nomQuest;

OBSERVATION: Le coeff des questionnaires ne sont pas pris en compte. exemple: si un questionnaire est sur 100 et un autre sur 20 le calcul des coeff ne se fera pas.

**40 Mohamed Vert**

Affichez tous les questionnaires créés par un formateur spécifique, avec le nombre total de questions dans chaque questionnaire et la moyenne des notes attribuées par les apprenants à chaque questionnaire:

SELECT q.idQuest, q.nomQuest, COUNT(qu.idQuestion) AS NombreQuestions, AVG(a.noteApp) AS MoyenneNotes

FROM Questionnaires q

JOIN Composer c ON q.idQuest = c.idQuest

JOIN Questions qu ON c.idQuestion = qu.idQuestion

LEFT JOIN QUEST\_APP qa ON q.idQuest = qa.idQuest

LEFT JOIN Apprenants a ON qa.id\_app = a.idApp

WHERE q.idFor = (SELECT idFor FROM Formateurs WHERE nomFor = [VEUILLER ENTER LE NOM DU FORMATEUR ICI] )

GROUP BY q.idQuest, q.nomQuest;

l’exercice demander une requete mais je n’ai pas pu faire en sorte que le nom du formateur soit un parametre donc il faudra l’entrer aprés le =

**41 Lucas V BLEU**

Donner les threads qui sont vieux de plus d'un mois avec un nombre nombre de consultation inférieur à 10 pour pouvoir les supprimer

SELECT threads.idThread FROM threads

WHERE threads.nbConsult<10

AND DATEDIFF(CURRENT\_DATE(),threads.dateCrea) > 30

**42 Lucas V BLEU**

Donner les 3 questions avec le plus haut et le plus bas taux de réussite

SELECT idQuestion, textQuestion, reussites, total\_reponses, pourcentage\_reussite

FROM (

SELECT Q.idQuestion, Q.textQuestion,

SUM(CASE WHEN R.bonneRep = 1 THEN 1 ELSE 0 END) AS reussites,

COUNT(\*) AS total\_reponses,

(SUM(CASE WHEN R.bonneRep = 1 THEN 1 ELSE 0 END) / COUNT(\*)) \* 100 AS pourcentage\_reussite

FROM Questions Q

INNER JOIN Reponses R ON Q.idQuestion = R.idQuestion

GROUP BY Q.idQuestion, Q.textQuestion

) AS stats

UNION ALL

SELECT idQuestion, textQuestion, reussites, total\_reponses, pourcentage\_reussite

FROM (

SELECT Q.idQuestion, Q.textQuestion,

SUM(CASE WHEN R.bonneRep = 1 THEN 1 ELSE 0 END) AS reussites,

COUNT(\*) AS total\_reponses,

(SUM(CASE WHEN R.bonneRep = 1 THEN 1 ELSE 0 END) / COUNT(\*)) \* 100 AS pourcentage\_reussite

FROM Questions Q

INNER JOIN Reponses R ON Q.idQuestion = R.idQuestion

GROUP BY Q.idQuestion, Q.textQuestion

) AS stats

ORDER BY pourcentage\_reussite ASC

LIMIT 3;

**43 Oscar Bleu**

SELECT Formateurs.idFor, Formateurs.nomFor, COUNT(Questionnaires.idQuest) AS nbQuestionnairesProp

FROM Formateurs

LEFT JOIN Questionnaires ON Formateurs.idFor = Questionnaires.idFor

GROUP BY Formateurs.idFor, Formateurs.nomFor

ORDER BY nbQuestionnairesProp DESC;

**44 Lucas V Orange**

SELECT

F.idFor,F.NbQuest,F.Rank,A.NbAComplete

FROM (

SELECT Formateurs.idFor,

COUNT(Questionnaires.idQuest) AS NbQuest,

RANK() OVER (ORDER BY COUNT(Questionnaires.idQuest) DESC)

AS Rank

FROM Formateurs

LEFT JOIN Questionnaires ON Formateurs.idFor = Questionnaires.idFor

GROUP BY idFor

) AS F

JOIN (

SELECT Formateurs.idFor,

COUNT(DISTINCT Apprenants.idApp) AS NbAComplete

FROM Formateurs

LEFT JOIN Questionnaires ON Formateurs.idFor = Questionnaires.idFor

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

LEFT JOIN Apprenants ON QUEST\_APP.id\_app = Apprenants.idApp

GROUP BY Formateurs.idFor

) AS A ON F.idFor = A.idFor

ORDER BY F.Rank;