❷ Périgée : Logiciel de gestion des notes de l'IUT de Saint Denis

On s'intéresse ici au logiciel Périgée qui permet de gérer les notes des étudiants du département Informatique de l'IUT de Saint-Denis de la Réunion. Ce logiciel repose sur une base de données Oracle dont le schéma relationnel vous est communiqué ci-dessous.

PROMOTIONS (idPromotion, nomPromotion, *nbEtudiantsPromotion*)
GROUPES (idGroupe, idPromotion#)
ETUDIANTS (idEtudiant, nomEtudiant, prenomEtudiant, sexeEtudiant, dateNaissanceEtudiant, idGroupe#)
SEMESTRES (idSemestre, dateDebutSemestre, dateFinSemestre, idPromotion#)
ABSENCES (idAbsence, idEtudiant#, dateHeureDebutAbsence, dateHeureFinAbsence)
JUSTIFICATIFSABSENCES (idJustificatifAbsence, idEtudiant#, dateDebutJustificatif, dateFinJustificatif, motifJustificatif)
MODULES (idModule, nomModule, idSemestre#, *coefficientModule*)
MATIERES (idMatiere, nomMatiere, idModule#, coefficientMatiere)
NOTES (idEtudiant#, idMatiere#, note)

La table *Promotions* recense les différentes promotions d'étudiants de l'IUT (en l'occurrence, la promotion de première année et celle de seconde année) et la table *Groupes* inventorie tous les groupes de TD de ces promotions (les groupes Q1, Q2, Q3 et Q4 en première année, et les groupes D1 et D2 en seconde année). Dans la table *Etudiants*, on peut trouver toutes les informations qui concernent les étudiants qui sont affectés à ces groupes de TD.

La table *Semestres* nous indique à quelles dates commencent et se terminent les semestres 1, 2, 3 et 4 et à quelle promotion ils sont rattachés ; par exemple, les semestres 1 et 2 sont rattachés à la première année, et les semestres 3 et 4 à la seconde année. La table *Absences* permet de connaître la date et l'horaire des cours pour lesquels les étudiants ont été contrôlés absents ; avec pour chaque absence, la date et l'heure de début du cours (`dateHeureDebutAbsence`) et de fin du cours (`dateHeureFinAbsence`). Il est à noter que si lors d'une journée, un étudiant a été absent à plusieurs cours, il pourra avoir une absence pour chacun de ces cours (à condition que l'appel ait été fait).
Dans la table *JustificatifsAbsences* sont répertoriés les justificatifs d'absence que les étudiants ont donnés au secrétariat de l'IUT. Un justificatif d'absence possède une date de début (`dateDebutJustificatif`), une date de fin (`dateFinJustificatif`) et un motif d'absence (`motifJustificatif`).

Dans la table *Modules* on peut trouver tous les modules dispensés pendant les différents semestres. Par exemple, le module 111 "Initiation Informatique" est dispensé au premier semestre (de la première année) ; alors que le module 211 "Programmation Web" est dispensé au troisième semestre (de la seconde année).
La table *Matières* permet de connaître les matières qui sont enseignées (et donc évaluées) à l'intérieur d'un module. Par exemple, le module 212 "Culture Générale Informatique" (du semestre 1 de la deuxième année) comprend les matières « UE15 Refactoring Agile », « UE16 Qualité & Tests » et « UE17 eXtreme Programming ». Enfin, la table *Notes* permet de connaître les notes qu'ont obtenues les étudiants dans les différentes matières.

Vous trouverez sur l'ENT le fichier « Périgée.sql » qui vous permettra de générer les tables de la base de données sur votre schéma.

Dans les tables *Promotions* et *Modules* figurent respectivement les attributs `nbEtudiantsPromotion` et `coefficientModule` qui sont indiqués en italique dans le schéma relationnel. Ces attributs sont dits dérivés (au sens UML), c'est-à-dire qu'ils sont calculables à partir de requêtes SQL. Ils engendrent donc de la redondance dans la base de données mais ont tout de même été maintenus car leur contenu est relativement stable. Toutefois, dans le script « Périgée.sql » leur valeur n'a pas encore été initialisée et sur toutes les lignes, ces attributs ont pour le moment la valeur `NULL`.

Première partie – Les Blocs PL/SQL :

1) Bloc PL/SQL simple et paquetage `DBMS_OUTPUT`.
   - Ecrire un bloc PL/SQL qui, grâce au paquetage `DBMS_OUTPUT`, affiche le nombre d'étudiants qui se trouvent dans le groupe qui a pour identifiant 'Q1'.

     On rappelle que pour que le paquetage `DBMS_OUTPUT` fonctionne dans l'interface iSQL*Plus, il faut l'activer avec la commande `SET SERVEROUTPUT ON` avant la première instruction du bloc PL/SQL (cette option sera valable durant toute la session iSQL*Plus).
   - Résultat attendu :

```
Il y a 6 étudiant(s) dans le groupe Q1
```

2) Utilisation de variables.
   - On souhaite maintenant tester le bloc PL/SQL de la question précédente avec d'autres groupes de TD (par exemple les groupes Q2, Q3, Q4, ...). Mais on veut avoir à modifier un minimum de lignes de code lorsque on change le groupe de TD, et ce grâce à l'utilisation d'une variable.

     Si cela n'a pas été fait lors de la question précédente, refactorer le code du bloc PL/SQL afin qu'il ne soit utile de modifier qu'une seule ligne de code lorsque on teste un autre groupe de TD.
   - Résultats attendus :

```
Il y a 6 étudiant(s) dans le groupe Q1

Il y a 5 étudiant(s) dans le groupe Q2

Il y a 0 étudiant(s) dans le groupe Q4
```

3) Les Exceptions : l'exception `NO_DATA_FOUND`.
   - Dans la question précédente il y a un léger problème. En effet, si l'identifiant du groupe n'existe pas (par exemple le Q5), il est indiqué qu'il y a 0 étudiant dans le groupe en question ; on obtient la même réponse si on saisit l'identifiant d'un groupe qui existe bien mais qui n'a aucun étudiant (le Q4).

     En s'inspirant du bloc PL/SQL écrit lors de la question précédente, écrire un nouveau bloc qui indiquerait le cas échéant qu'aucun groupe ne possède l'identifiant affecté à votre variable.

     On utilisera ici l'exception prédéfinie `NO_DATA_FOUND` qui est levée automatiquement lorsqu'une requête `SELECT … INTO` ne retourne rien (il est à noter que si la requête retourne plusieurs lignes, c'est alors l'exception `TOO_MANY_ROWS` qui est levée).
   - Résultats attendus :

```
Il y a 0 étudiant(s) dans le groupe Q4

Il n'y a pas de groupe Q5
```

4) Structures de contrôle : la conditionnelle (`IF … THEN … ELSE … END IF`).
   - Ecrire un nouveau bloc PL/SQL qui fait exactement la même chose que la question précédente mais en utilisant cette fois-ci la structure de contrôle (`IF … THEN … ELSE … END IF`) à la place de la section `EXCEPTION`.
   - Résultats attendus :

```
Il y a 0 étudiant(s) dans le groupe Q4

Il n'y a pas de groupe Q5
```

5) Les variables %ROWTYPE.
   - Ecrire un bloc PL/SQL qui permet d'afficher toutes les informations nominatives de la table *Etudiants* qui concernent l'étudiant E1.

     Pour se simplifier la tâche, et pour faciliter les éventuelles futures maintenances de l'application, on récupèrera les données qui concernent l'étudiant E1 dans une variable %ROWTYPE.

   - Résultat attendu :

```
Identifiant étudiant : E1
Nom étudiant : Alizan
Prénom étudiant : Gaspard
Sexe étudiant : M
Date naissance étudiant : 14/07/05
Groupe étudiant : Q1
```

Deuxième partie – Les Procédures et Fonctions stockées :

Fonctions stockées

6) Fonction stockée nbEtudiantsParGroupe.
   - En vous inspirant de ce que vous avez écrit lors des questions 3 ou 4, écrire une fonction stockée nbEtudiantsParGroupe qui retourne le nombre d'étudiants qui appartiennent au groupe dont l'identifiant p_idGroupe est passé en paramètre. Cette fonction doit avoir la signature suivante :

```
FUNCTION nbEtudiantsParGroupe(p_idGroupe IN Groupes.idGroupe%TYPE)
                                                    RETURN NUMBER
```

     Si on passe en paramètre de cette fonction un identifiant de groupe qui n'existe pas, on souhaite que la fonction retourne NULL (et non pas 0).

     Si lors de la création de votre fonction, vous avez une erreur de compilation (Warning: Function created with compilation errors) vous pouvez demander à Oracle de vous afficher la liste des erreurs grâce à l'instruction SHOW ERRORS.

     On rappelle que sous Oracle, pour tester une fonction stockée, on peut utiliser la pseudo-table DUAL.

   - Résultats attendus :

```
SELECT nbEtudiantsParGroupe('Q1')
FROM DUAL ;

nbEtudiantsParGroupe('Q1')
--------------------------
6

SELECT nbEtudiantsParGroupe('Q5')
FROM DUAL ;

nbEtudiantsParGroupe('Q5')
--------------------------
// quand le résultat est NULL, par défaut, iSQL*Plus n'affiche rien
```

7) Fonction stockée nbEtudiantsParPromotion.
   - Ecrire une fonction stockée nbEtudiantsParPromotion qui retourne le nombre d'étudiants qui appartiennent à une promotion dont l'identifiant p_idPromotion est passé en paramètre. Cette fonction doit avoir la signature suivante :

```
FUNCTION nbEtudiantsParPromotion(
                    p_idPromotion IN Promotions.idPromotion%TYPE)
                                                    RETURN NUMBER
```

     Dans le corps de cette fonction, on pourra appeler la fonction qui a été réalisée à la question précédente (nbEtudiantsParGroupe). Cette fonction pourra être appelée dans le SELECT d'une requête !

On ne vous demande pas de gérer explicitement le cas où l'identifiant de promotion qui est passé en paramètre n'existe pas, ou bien le cas où la promotion existe mais qu'elle n'a pas de groupe (dans ces cas là, votre fonction devrait retourner `NULL` par défaut sans que vous soyez obligé de le programmer).

- Résultat attendu :

```
SELECT nbEtudiantsParPromotion('A1')
FROM DUAL ;

nbEtudiantsParPromotion('A1')
-----------------------------
16
```

- Ecrire une requête SQL qui, en utilisant la fonction écrite ci-dessus, met à jour l'attribut `nbEtudiantsPromotion` de toutes les lignes de la table *Promotions*. Puis, vérifier que dans la table *Promotions* on a bien 16 étudiants en A1 et 11 en A2.

Procédures stockées

8) Procédure stockée affichageInfosEtudiant.
   - En vous inspirant de ce que vous avez écrit lors de la question 5, écrire une procédure stockée `affichageInfosEtudiant` qui affiche toutes les informations nominatives qui concernent l'étudiant dont l'identifiant `p_idEtudiant` est passé en paramètre. Cette procédure doit avoir la signature suivante :

```
PROCEDURE affichageInfosEtudiant(
                              p_idEtudiant IN Etudiants.idEtudiant%TYPE)
```

   On rappelle que sous Oracle, on peut appeler une procédure stockée avec l'instruction `CALL`.

   - Résultat attendu :

```
CALL affichageInfosEtudiant('E1');

Identifiant étudiant : E1
Nom étudiant : Alizan
Prénom étudiant : Gaspard
Sexe étudiant : M
Date naissance étudiant : 14/07/05
Groupe étudiant : Q1
```

9) Procédure stockée miseAJourCoefficientModules.
   - Ecrire une procédure stockée qui met à jour l'attribut `coefficientModule` de toutes les lignes de la table *Modules*. Le coefficient d'un module doit être égal à la somme des coefficients des matières qui sont incluses dans le module. Cette procédure doit avoir la signature suivante :

```
PROCEDURE miseAJourCoefficientModules
```

   - Résultat attendu :

```
CALL miseAJourCoefficientModules() ;


SELECT idModule, coefficientModule
FROM Modules

idModule  coefficientModule
--------- ------------------
M111      6
M112      6
M113      3
M121      6
M122      6
M123      3
M211      6
M212      6
M213      3
M221      6
M222      6
```

Troisième partie – Les Curseurs :

10) Procédure stockée affichageNotesEtudiant.

- Ecrire une procédure stockée `affichageNotesEtudiant` qui affiche toutes les notes obtenues par un étudiant dont l'identifiant `p_idEtudiant` est passé en paramètre. Cette procédure doit avoir la signature suivante :

```
PROCEDURE affichageNotesEtudiant(
                            p_idEtudiant IN Etudiants.idEtudiant%TYPE)
```

- Résultat attendu :

```
CALL affichageNotesEtudiant('E1') ;

Prog ADA : 12
BD : 11
Math : 9
Gestion : 11
Communication : 13
Projet : 14
Initiation Scrum : 14
Prog Objet : 9
BD Avancées : 6
Statistiques : 10
Anglais : 9
Gestion de Projet Agile : 4
```

11) Procédure stockée affichageNotesEtudiantSemestre.

- En vous inspirant de ce que vous avez fait à la question précédente, écrire une procédure stockée `affichageNotesEtudiantSemestre` qui affiche toutes les notes obtenues par un étudiant `p_idEtudiant` au cours du semestre `p_idSemestre` donnés en paramètre. Cette procédure doit avoir la signature suivante :

```
PROCEDURE affichageNotesEtudiantSemestre(
                            p_idEtudiant IN Etudiants.idEtudiant%TYPE,
                            p_idSemestre IN Semestres.idSemestre%TYPE)
```

- Résultat attendu :

```
CALL affichageNotesEtudiantSemestre('E1', 'S2') ;

Prog Objet : 9
BD Avancées : 6
Statistiques : 10
Anglais : 9
Gestion de Projet Agile : 4

CALL affichageNotesEtudiantSemestre('E1', 'S4') ;

L'étudiant E1 n'est pas inscrit au semestre S4
```

12) Procédure stockée affichageToutEtudiantSemestre.

- Ecrire une procédure stockée `affichageToutEtudiantSemestre` qui affiche toutes les informations nominatives qui concernent un étudiant dont l'identifiant `p_idEtudiant` est passé en paramètre, ainsi que les notes qu'il a obtenues au semestre `p_idSemestre`. Cette procédure doit avoir la signature suivante :

```
PROCEDURE affichageToutEtudiantSemestre(
                            p_idEtudiant IN Etudiants.idEtudiant%TYPE,
                            p_idSemestre IN Semestres.idSemestre%TYPE)
```

- Résultat attendu :

```
CALL affichageToutEtudiantSemestre('E10', 'S3') ;

Identifiant étudiant : E10
Nom étudiant : Palleja
Prénom étudiant : Xavier
Sexe étudiant : M
Date naissance étudiant : 09/03/02
Groupe étudiant : D2
Prog PHP : 18
Systèmes et Réseaux : 17
Refactoring Agile : 14
Qualité et Tests : 16
eXtreme Programming : 18
Projet Programmation : 16
```

13) Procédure stockée affichageAbsencesParPromotion.

- Ecrire une procédure stockée `affichageAbsencesParPromotion` qui affiche pour tous les groupes qui appartiennent à une promotion dont l'identifiant `p_idPromotion` est passé en paramètre, la liste des étudiants du groupe avec pour chacun d'entre eux le nombre d'absences. On veut que les groupes affichés soient classés par ordre alphabétique de leur identifiant. Et que dans un groupe les étudiants soient classés par rapport au nombre d'absences qu'ils possèdent (si plusieurs étudiants ont le même nombre d'absences, ils doivent être classés dans l'ordre alphabétique de leur nom). Cette procédure doit avoir la signature suivante :

```
PROCEDURE affichageAbsencesParPromotion(
                        p_idPromotion IN Promotions.idPromotion%TYPE)
```

- Résultat attendu :

```
CALL affichageAbsencesParPromotion('A1');

Groupe : Q1 (6 étudiants)
-----> Alizan Gaspard a été absent 18 fois
-----> Assin Marc a été absent 9 fois
-----> Zeblouse Agathe a été absent 7 fois
-----> Javel Aude a été absent 5 fois
-----> Timettre Vincent a été absent 5 fois
-----> Terrieur Alex a été absent 0 fois

Groupe : Q2 (5 étudiants)
-----> Nanas Judas a été absent 9 fois
-----> Vesselle Aude a été absent 8 fois
-----> Deuf John a été absent 4 fois
-----> Zetofrais Mélanie a été absent 4 fois
-----> Tarembois Guy a été absent 3 fois

Groupe : Q3 (5 étudiants)
-----> Ouzy Jacques a été absent 28 fois
-----> Niohrangina Nicolas a été absent 11 fois
-----> Bono Jean a été absent 5 fois
-----> Kaécouté Xavier a été absent 4 fois
-----> Gator Ali a été absent 1 fois

Groupe : Q4 (0 étudiants)




CALL affichageAbsencesParPromotion('A2');

Groupe : D1 (5 étudiants)
-----> Micoton Mylène a été absent 16 fois
-----> Croque Odile a été absent 11 fois
-----> Némard Jean a été absent 5 fois
-----> Outan Laurent a été absent 3 fois
-----> Triser Jessica a été absent 2 fois

Groupe : D2 (6 étudiants)
-----> Sticé Sophie a été absent 11 fois
-----> Sticko Judas a été absent 9 fois
-----> Bricot Judas a été absent 8 fois
-----> Terrieur Alain a été absent 4 fois
-----> Oraque Anne a été absent 0 fois
-----> Palleja Xavier a été absent 0 fois
```

Quatrième partie – Edition des relevés de notes individuels :

Dans cette partie, on souhaite pouvoir éditer le relevé de notes semestriel d'un étudiant donné. Par exemple, le relevé de notes du semestre S3 de l'éblouissant étudiant E10 doit avoir la forme suivante :

```
Identifiant étudiant : E10
Nom étudiant : Palleja
Prénom étudiant : Xavier
Sexe étudiant : M
Date naissance étudiant : 09/03/02
Groupe étudiant : D2

Prog PHP : 18
Systèmes et Réseaux : 17
Moyenne module Informatique Web : 17,5

Refactoring Agile : 14
Qualité et Tests : 16
eXtreme Programming : 18
Moyenne module Culture Générale Info : 16

Projet Programmation : 16
Moyenne module Projet Final Pratique : 16

Moyenne semestre sans les absences : 16,6
Nombre d'absences non justifiées : 0
Moyenne avec les absences : 16,6

Resultat : O
Classement : 1
```

La moyenne d'un module est calculée avec les notes des différentes matières qui sont incluses dans le module (en prenant en compte les coefficients des matières). La moyenne d'un semestre sans les absences est calculée avec les moyennes des modules (en prenant en compte les coefficients des modules).

La moyenne avec les absences est calculée à partir de la moyenne sans les absences à laquelle on retranche 0,1 point à chaque absence non justifiée. Il est à noter que si l'étudiant n'a qu'une seule absence non justifiée, on ne lui retire rien.

Le résultat indique si l'étudiant valide son semestre (O/N). Pour valider un semestre, il faut avoir au moins 10 de moyenne générale (moyenne générale avec les absences), et il faut également avoir au moins 8 dans chacun des modules. Le classement indique la place de l'étudiant dans sa promotion pour le semestre donné (naturellement, l'étudiant précédent est major de sa promotion).

Ci-dessous, l'exemple du relevé de notes du semestre S1 de l'étudiant E18.

```
Identifiant étudiant : E18
Nom étudiant : Ouzy
Prénom étudiant : Jacques
Sexe étudiant : M
Date naissance étudiant : 30/10/08
Groupe étudiant : Q3

Prog ADA : 8
BD : 11
Moyenne module Initiation Informatique : 9,5

Math : 10
Gestion : 11
Communication : 9
Moyenne module Culture Générale : 10

Projet : 12
Initiation Scrum : 18
Moyenne module Mise en Pratique : 14

Moyenne semestre sans les absences : 10,6
Nombre d'absences non justifiées : 14
Moyenne avec les absences : 9,2

Resultat : N
Classement : 13
```

L'étudiant précédent ne valide pas son semestre car il n'a pas la moyenne générale. Il avait pourtant 10,6 de moyenne sans les absences mais à cause de ses 14 absences non justifiées, sa moyenne a chuté à 9,4 et il ne valide donc pas son semestre. C'est d'ailleurs ce qui risque de vous arriver ce semestre si vous ne venez pas à tous les TD de BD.

Enfin, un dernier exemple de relevé de notes ; celui du semestre S1 de l'étudiant E17.

```
Identifiant étudiant : E17
Nom étudiant : Tarembois
Prénom étudiant : Guy
Sexe étudiant : M
Date naissance étudiant : 08/04/05
Groupe étudiant : Q2

Prog ADA : 12
BD : 13
Moyenne module Initiation Informatique : 12,5

Math : 10
Gestion : 9
Communication : 17
Moyenne module Culture Générale : 12

Projet : 8
Initiation Scrum : 6,5
Moyenne module Mise en Pratique : 7,5

Moyenne semestre sans les absences : 11,3
Nombre d'absences non justifiées : 1
Moyenne avec les absences : 11,3

Resultat : N
Classement : 7
```

L'étudiant précédent ne valide pas son semestre 1, bien qu'il ait la moyenne générale et qu'il soit classé 7ème de promo, car il n'a pas au moins 8 dans tous les modules. En effet il n'a que 7,5 dans le module "Mise en Pratique" (à cause de la matière Scrum où il n'a eu que 6,5 – tant pis pour lui, il n'avait qu'à apprendre à faire un planning poker afin de bien planifier son Backlog de sprint).

Il est à noter que comme l'étudiant n'a qu'une absence, sa moyenne avec les absences n'est pas impactée par les absences.

Pour éditer ces relevés de notes, nous allons commencer par écrire les fonctions qui permettent de calculer les moyennes intermédiaires. Puis, nous écrirons les fonctions qui gèrent les absences. Enfin nous terminerons par coder les fonctions qui calculent le résultat et le classement de l'étudiant.

A. Gestions des moyennes

14) Fonction stockée moyenneEtudiantModule.

- Ecrire une fonction stockée `moyenneEtudiantModule` qui retourne la moyenne qu'a obtenue un étudiant `p_idEtudiant` à un module `p_idModule`.

  Pour calculer la moyenne au module, il faut prendre en compte les coefficients des matières incluses dans le module.

  Par exemple, si dans un module, un étudiant a obtenu 8 à une matière qui a un coefficient 2 et 11 à une matière qui a un coefficient 1, il aura 9 de moyenne au module :
  `(8*2 + 11*1) / (2+1) = 9`
  Autre exemple, si à un module, un étudiant a obtenu 8 à une matière qui a un coefficient 2 et 11 à une matière qui a un coefficient 2, il aura 9,5 de moyenne au module :
  `(8*2 + 11*2) / (2+2) = 9,5`

  La fonction doit avoir la signature suivante :

```
FUNCTION moyenneEtudiantModule(
                          p_idEtudiant IN Etudiants.idEtudiant%TYPE,
                          p_idModule IN Modules.idModule%TYPE)
                                                    RETURN NUMBER
```

- Résultats attendus :

```
SELECT moyenneEtudiantModule('E6', 'M112')
FROM DUAL ;

moyenneEtudiantModule('E6', 'M112')
-----------------------------------
7.5

SELECT moyenneEtudiantModule('E8', 'M113')
FROM DUAL ;

moyenneEtudiantModule('E8', 'M113')
-----------------------------------
13.5
```

15) Fonction stockée valideEtudiantModule.

- Ecrire une fonction stockée `valideEtudiantModule` qui retourne 1 si un étudiant `p_idEtudiant` valide un module `p_idModule` ; 0 sinon. On rappelle qu'un étudiant valide un module si il a au moins 8 de moyenne à ce module. Cette fonction doit avoir la signature suivante :

```
FUNCTION valideEtudiantModule(
                    p_idEtudiant IN Etudiants.idEtudiant%TYPE,
                    p_idModule IN Modules.idModule%TYPE)
                                                    RETURN NUMBER
```

- Résultats attendus :

```
SELECT valideEtudiantModule('E6', 'M112')
FROM DUAL ;

valideEtudiantModule('E6', 'M112')
-----------------------------------
0

SELECT valideEtudiantModule('E19', 'M211')
FROM DUAL ;

valideEtudiantModule('E19', 'M211')
-----------------------------------
1
```

16) Fonction stockée moyenneEtudiantSemestreSansAbs.

- Ecrire une fonction stockée `moyenneEtudiantSemestreSansAbs` qui retourne la moyenne qu'a obtenue un étudiant `p_idEtudiant` au semestre `p_idSemestre`.

  Pour calculer cette moyenne, il faut prendre en compte les coefficients des différents modules qu'a suivis l'étudiant au cours du semestre.

  Cette fonction doit avoir la signature suivante :

```
FUNCTION moyenneEtudiantSemestreSansAbs (
                    p_idEtudiant IN Etudiants.idEtudiant%TYPE,
                    p_idSemestre IN Semestres.idSemestre%TYPE)
                                                    RETURN NUMBER
```

- Résultats attendus :

```
SELECT moyenneEtudiantSemestreSansAbs('E1', 'S1')
FROM DUAL ;

moyenneEtudiantSemestreSansAbs('E1', 'S1')
------------------------------------------
11,8

SELECT moyenneEtudiantSemestreSansAbs('E3', 'S2')
FROM DUAL ;

moyenneEtudiantSemestreSansAbs('E3', 'S2')
------------------------------------------
16,4

SELECT moyenneEtudiantSemestreSansAbs('E3', 'S3')
FROM DUAL ;

moyenneEtudiantSemestreSansAbs('E3', 'S3')
------------------------------------------
```

17) Procédure stockée affichageMoyEtudiantSemestre.

- Ecrire une procédure stockée `affichageMoyEtudiantSemestre` qui affiche toutes les notes et les moyennes obtenues par un étudiant `p_idEtudiant` au cours du semestre `p_idSemestre`. Cette procédure doit avoir la signature suivante :

```
PROCEDURE affichageMoyEtudiantSemestre(
                          p_idEtudiant IN Etudiants.idEtudiant%TYPE,
                          p_idSemestre IN Semestres.idSemestre%TYPE)
```

- Résultat attendu :

```
CALL affichageMoyEtudiantSemestre('E10', 'S3') ;

Identifiant étudiant : E10
Nom étudiant : Palleja
Prénom étudiant : Xavier
Sexe étudiant : M
Date naissance étudiant : 09/03/02
Groupe étudiant : D2

Prog PHP : 18
Systèmes et Réseaux : 17
Moyenne module Informatique Web : 17,5

Refactoring Agile : 14
Qualité et Tests : 16
eXtreme Programming : 18
Moyenne module Culture Générale Info : 16

Projet Programmation : 16
Moyenne module Projet Final Pratique : 16

Moyenne semestre sans les absences : 16,6
```

B. Gestion des absences

18) Fonction stockée typeAbsence

- Pour pouvoir comptabiliser les absences dans la moyenne générale, il faut savoir quelles sont les absences de la table *Absences* qui sont excusées par un justificatif d'absence de la table *JustificatifsAbsences* et celles qui ne le sont pas.

  Pour cela, écrire une fonction stockée `typeAbsence` qui retourne 'E' si l'absence `p_idAbsence` est excusée par un justificatif d'absence, et 'A' dans le cas contraire. Pour savoir si un justificatif peut excuser une absence, il faut s'assurer que l'absence est incluse dans la période du justificatif et que le justificatif et l'absence concernent le même étudiant.

  Cette fonction doit avoir la signature suivante :

```
FUNCTION typeAbsence (p_idAbsence IN Absences.idAbsence%TYPE)
                                                    RETURN VARCHAR
```

- Résultats attendus :

```
SELECT typeAbsence('A92')
FROM DUAL ;

typeAbsence('A92')
------------------
E

SELECT typeAbsence('A96')
FROM DUAL ;

typeAbsence('A96')
------------------
E

SELECT typeAbsence('A100')
FROM DUAL ;

typeAbsence('A100')
-------------------
A
```

19) Fonction stockée nbAbsencesNonJustifiees

- Ecrire une fonction stockée `nbAbsencesNonJustifiees` qui retourne le nombre d'absences non justifiées de l'étudiant `p_idEtudiant` pour le semestre `p_idSemestre`. Cette fonction doit avoir la signature suivante :

```
FUNCTION nbAbsencesNonJustifiees(
                    p_idEtudiant IN Etudiants.idEtudiant%TYPE,
                    p_idSemestre IN Semestres.idSemestre%TYPE)
                                                    RETURN NUMBER
```

- Résultats attendus :

```
SELECT nbAbsencesNonJustifiees('E22', 'S1')
FROM DUAL ;

nbAbsencesNonJustifiees('E22', 'S1')
------------------------------------
3

SELECT nbAbsencesNonJustifiees('E11', 'S2')
FROM DUAL ;

nbAbsencesNonJustifiees('E11', 'S2')
------------------------------------
5
```

20) Fonction stockée moyenneEtudiantSemestreAvecAbs.

- Ecrire une fonction stockée `moyenneEtudiantSemestreAvecAbs` qui retourne la véritable moyenne qu'a obtenue un étudiant `p_idEtudiant` au semestre `p_idSemestre` après avoir retiré de la moyenne générale de l'étudiant les points relatifs aux absences non justifiées.

  Si l'étudiant n'a pas d'absence ou bien s'il n'a qu'une seule absence non justifiée dans le semestre, on ne lui retranche pas de point (sa moyenne avec les absences est identique à la moyenne sans les absences). Sinon, on retire à la moyenne 0,1 point par absence non justifiée. Cette fonction doit avoir la signature suivante :

```
FUNCTION moyenneEtudiantSemestreAvecAbs(
                    p_idEtudiant IN Etudiants.idEtudiant%TYPE,
                    p_idSemestre IN Semestres.idSemestre%TYPE)
                                                    RETURN NUMBER
```

- Résultats attendus :

```
SELECT moyenneEtudiantSemestreAvecAbs('E12', 'S1')
FROM DUAL ;

moyenneEtudiantSemestreAvecAbs('E12', 'S1')
-------------------------------------------
9

SELECT moyenneEtudiantSemestreAvecAbs('E11', 'S2')
FROM DUAL ;

moyenneEtudiantSemestreAvecAbs('E11', 'S2')
-------------------------------------------
4,9
```

21) Procédure stockée affichageAbsEtudiantSemestre.

- Ecrire une procédure stockée `affichageAbsEtudiantSemestre` qui affiche le nombre d'absences non justifiées ainsi que la véritable moyenne générale (absences comprises) de l'étudiant `p_idEtudiant` au cours du semestre `p_idSemestre`. Cette procédure doit avoir la signature suivante :

```
PROCEDURE affichageAbsEtudiantSemestre(
                        p_idEtudiant IN Etudiants.idEtudiant%TYPE,
                        p_idSemestre IN Semestres.idSemestre%TYPE)
```

- Résultats attendus :

```
CALL affichageAbsEtudiantSemestre('E13', 'S2') ;

Nombre d'absences non justifiées : 5
Moyenne avec les absences : 7,7
```

C. Gestion du résultat et du classement

22) Fonction stockée valideSemestre

- Ecrire une fonction stockée `valideSemestre` qui retourne `'O'` si l'étudiant `p_idEtudiant` valide le semestre `p_idSemestre` ; `'N'` sinon.

  Un étudiant valide un semestre s'il a au moins 10 de moyenne générale (absences comprises) ET s'il a également au moins 8 de moyenne dans chacun des modules qu'il a suivi. Cette fonction doit avoir la signature suivante :

```
FUNCTION valideSemestre(p_idEtudiant IN Etudiants.idEtudiant%TYPE,
                        p_idSemestre IN Semestres.idSemestre%TYPE)
                                                    RETURN VARCHAR
```

- Résultats attendus :

```
SELECT valideSemestre('E17', 'S2')
FROM DUAL ;

valideSemestre('E17', 'S2')
---------------------------
N


SELECT valideSemestre('E18', 'S1')
FROM DUAL ;

valideSemestre('E18', 'S1')
---------------------------
N


SELECT valideSemestre('E9', 'S3')
FROM DUAL ;

valideSemestre('E9', 'S3')
--------------------------
O
```

23) Fonction stockée classementEtudiantSemestre

- Ecrire une fonction stockée `classementEtudiantSemestre` qui retourne le classement de l'étudiant `p_idEtudiant` dans sa promotion pour le semestre `p_idSemestre`.

  Le classement est calculé à partir de la moyenne générale des étudiants, absences comprises. L'étudiant qui a la plus forte moyenne de la promotion a le classement 1, le suivant le classement 2 et ainsi de suite. Si deux étudiants ont la même moyenne ils doivent avoir le même classement.

  Cette fonction doit avoir la signature suivante :

```
FUNCTION classementEtudiantSemestre(
                        p_idEtudiant IN Etudiants.idEtudiant%TYPE,
                        p_idSemestre IN Semestres.idSemestre%TYPE)
                                                    RETURN NUMBER
```

→ Il est à noter qu'il est possible d'écrire cette fonction de façon extrêmement simple sans avoir à utiliser un curseur !

- Résultats attendus :

```
SELECT classementEtudiantSemestre('E10', 'S3')
FROM DUAL ;

classementEtudiantSemestre('E10', 'S3')
---------------------------------------
1


SELECT classementEtudiantSemestre('E21', 'S2')
FROM DUAL ;

classementEtudiantSemestre('E21', 'S2')
---------------------------------------
6
```

24) Procédure stockée affichageResEtudiantSemestre.

- Ecrire une procédure stockée `affichageResEtudiantSemestre` qui affiche le résultat et le classement de l'étudiant `p_idEtudiant` au semestre `p_idSemestre`. Cette procédure doit avoir la signature suivante :

```
PROCEDURE affichageResEtudiantSemestre(
                          p_idEtudiant IN Etudiants.idEtudiant%TYPE,
                          p_idSemestre IN Semestres.idSemestre%TYPE)
```

- Résultat attendu :

```
CALL affichageResEtudiantSemestre('E10', 'S3') ;

Resultat : O
Classement : 1
```

25) Procédure stockée affichageReleveNotes.

- Ecrire une procédure stockée `affichageReleveNotes` qui affiche le relevé de notes de l'étudiant `p_idEtudiant` au semestre `p_idSemestre`. Cette procédure doit avoir la signature suivante :

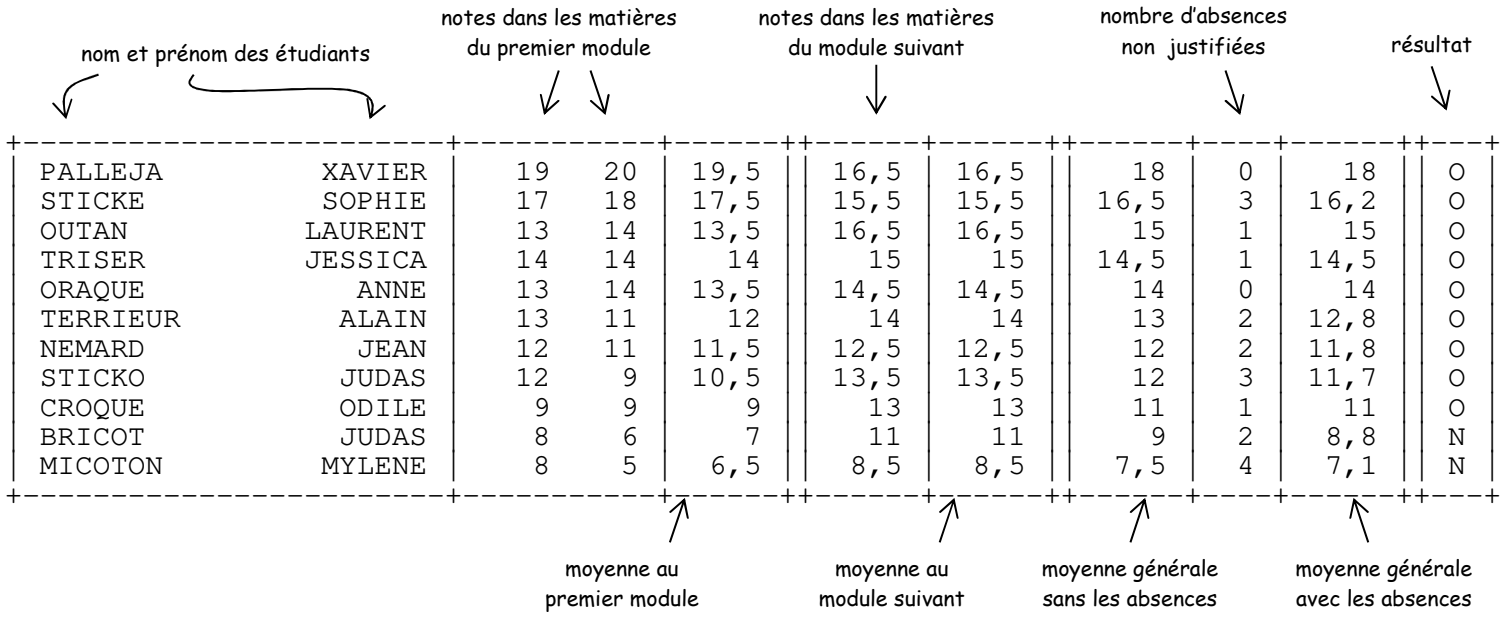
```
PROCEDURE affichageReleveNotes(
                          p_idEtudiant IN Etudiants.idEtudiant%TYPE,
                          p_idSemestre IN Semestres.idSemestre%TYPE)
```

- Résultats attendus :

```
CALL affichageReleveNotes('E10', 'S4') ;

Identifiant étudiant : E10
Nom étudiant : Palleja
Prénom étudiant : Xavier
Sexe étudiant : M
Date naissance étudiant : 09/03/02
Groupe étudiant : D2

BD Objet : 19
UML et Design Patterns : 20
Moyenne module Informatique Compl. : 19,5

Stage : 16,5
Moyenne module Stage Pratique : 16,5

Moyenne semestre sans les absences : 18
Nombre d'absences non justifiées : 0
Moyenne avec les absences : 18

Resultat : O
Classement : 1


CALL affichageReleveNotes('E9', 'S4') ;

Identifiant étudiant : E9
Nom étudiant : Croque
Prénom étudiant : Odile
Sexe étudiant : F
Date naissance étudiant : 14/07/04
Groupe étudiant : D1

BD Objet : 9
UML et Design Patterns : 9
Moyenne module Informatique Compl. : 9

Stage : 13
Moyenne module Stage Pratique : 13

Moyenne semestre sans les absences : 11
Nombre d'absences non justifiées : 1
Moyenne avec les absences : 11

Resultat : O
Classement : 9
```

Cinquième partie – Edition du PV de fin de semestre :

Pour faciliter le déroulement des jurys de fin de semestre, on souhaite pouvoir éditer le Procès-Verbal (PV) d'une promotion pour un semestre donné. Un PV doit afficher dans un même tableau toutes les notes obtenues par les étudiants d'une promotion au cours d'un semestre. Il doit également indiquer toutes les moyennes de ces étudiants (moyennes des modules, moyennes générales – sans et avec les absences), le nombre d'absence qui leur ont été comptabilisées et le résultat.
Par exemple, pour le semestre S4, on souhaite éditer le PV suivant :



PALLEJA	XAVIER	19	20	19,5	16,5	16,5	18	0	18	O
STICKE	SOPHIE	17	18	17,5	15,5	15,5	16,5	3	16,2	O
OUTAN	LAURENT	13	14	13,5	16,5	16,5	15	1	15	O
TRISER	JESSICA	14	14	14	15	15	14,5	1	14,5	O
ORAQUE	ANNE	13	14	13,5	14,5	14,5	14	0	14	O
TERRIEUR	ALAIN	13	11	12	14	14	13	2	12,8	O
NEMARD	JEAN	12	11	11,5	12,5	12,5	12	2	11,8	O
STICKO	JUDAS	12	9	10,5	13,5	13,5	12	3	11,7	O
CROQUE	ODILE	9	9	9	13	13	11	1	11	O
BRICOT	JUDAS	8	6	7	11	11	9	2	8,8	N
MICOTON	MYLENE	8	5	6,5	8,5	8,5	7,5	4	7,1	N

Il est à noter que sur le PV, sont d'abord affichés tous les étudiants qui ont validé le semestre (c'est-à-dire pour qui le résultat est `O`) ; puis en dessous, tous ceux qui n'ont pas validé le semestre (résultat = `N`). Les étudiants qui valident leur semestre sont classés par rapport à leur moyenne générale avec les absences. Même chose pour les étudiants qui ne valident par leur semestre.

On veut que le nom et le prénom des étudiants soient écrits en lettres majuscules. Si un nom ou un prénom fait plus de 10 lettres, il pourra être tronqué.

Pour les notes et les moyennes, on n'affichera pas plus d'une décimale. Si une note ou une moyenne possède plusieurs décimales, elle pourra être tronquée.

26) Procédure stockée affichagePV.

- Ecrire une procédure stockée `affichagePV` qui affiche le Procès Verbal de la promotion au semestre `p_idSemestre`. Cette procédure doit avoir la signature suivante :
  `PROCEDURE affichagePV( p_idSemestre IN Semestres.idSemestre%TYPE)`

  Pour améliorer la lisibilité de votre code et faciliter sa maintenance, il est fortement conseillé de créer d'autres procédures stockées qui seront appelées à l'intérieur du corps de procédure `affichagePV`.

  Afin d'améliorer les performances, vous avez la possibilité, si vous le souhaitez, de rajouter des champs calculés dans votre Schéma Relationnel.

  Pour mettre en forme votre texte, vous pourrez utiliser les fonctions Oracle suivantes :
  - `LPAD()` et `RPAD()` qui permettent d'aligner un texte sur la gauche ou sur la droite et qui permet également de le tronquer si il est trop long.
  - `UPPER()` qui permet de mettre en majuscules toutes les lettres d'un texte.
  - `TRANSLATE()` qui peut être utilisé pour remplacer les lettres accentuées d'un texte par des lettres non accentuées (par exemple, remplacer É par E).

→ Il est à noter qui si on affiche le résultat directement dans le navigateur, on risque de perdre la mise en forme (les espaces consécutifs vont disparaître). Pour cette raison, demandez à iSQL*Plus de ne pas générer le résultat dans le navigateur mais dans un fichier externe que vous pourrez ouvrir avec votre éditeur de texte préféré (par exemple Notepad++) :
Préférences – Emplacement de sortie – Enregistrer dans un fichier HTML.

Comme votre fichier externe sera ouvert avec un éditeur texte, demandez également à iSQL*Plus, lors de la génération du fichier, de ne pas remplacer les sauts de lignes par des balises `<BR>` :
Préférences – Formatage du script – Sortie préformatée – Activer (On).

- Résultats attendus : voir pages suivantes.

```
CALL affichagePV('S1');
```

DEUF	JOHN	15	15	15	19	18	17	18	16,5	15	16	16,4	1	16,4	O
TIMETTRE	VINCENT	16	17	16,5	13	12	14	13	15,5	14	15	14,8	2	14,6	O
KAECOUTE	XAVIER	10	18	14	14	14	17	15	16,5	15	16	14,8	3	14,5	O
NANAS	JUDAS	14	14	14	12	9	12	11	13	14,5	13,5	12,7	3	12,4	O
GATOR	ALI	8	14	11	14	12	13	13	14	12,5	13,5	12,3	0	12,3	O
ZETOFRAIS	MELANIE	9	10	9,5	14	13	12	13	15	15	15	12	2	11,8	O
ALIZAN	GASPARD	12	11	11,5	9	11	13	11	14	14	14	11,8	5	11,3	O
TERRIEUR	ALEX	13	12	12,5	10	9	8	9	13	11,5	12,5	11,1	0	11,1	O
BONO	JEAN	6	13	9,5	9	10	14	11	13	13	13	10,8	1	10,8	O
JAVEL	AUDE	13	12	12,5	10	9	8	9	11	14	12	11	3	10,7	O
TAREMBOIS	GUY	12	13	12,5	10	9	17	12	8	6,5	7,5	11,3	1	11,3	N
ASSIN	MARC	13	14	13,5	9	5	8,5	7,5	12	15	13	11	3	10,7	N
OUZY	JACQUES	8	11	9,5	10	11	9	10	12	18	14	10,6	14	9,2	N
VESSELLE	AUDE	9	11	10	9	7	5	7	11	14	12	9,2	2	9	N
ZEBLOUSE	AGATHE	2	1	1,5	15	13	11	13	15	15	15	8,8	1	8,8	N
NIOHRANGIN	NICOLAS	1	9	5	9	11	7	9	13	8,5	11,5	7,9	5	7,4	N

```
CALL affichagePV('S2');
```

KAECOUTE	XAVIER	16	14	15	17	18	17,5	17	17	16,4	1	16,4	O
DEUF	JOHN	14	14	14	15	15	15	14	14	14,4	2	14,2	O
TIMETTRE	VINCENT	16	18	17	11	12	11,5	14	14	14,2	3	13,9	O
GATOR	ALI	14	14	14	10	11	10,5	13	13	12,4	1	12,4	O
NANAS	JUDAS	13	12	12,5	10	14	12	14	14	12,6	4	12,2	O
TERRIEUR	ALEX	12	12	12	11	13	12	12	12	12	0	12	O
JAVEL	AUDE	12	12	12	11	13	12	12	12	12	1	12	O
ZETOFRAIS	MELANIE	9	9	9	15	14	14,5	12	12	11,8	2	11,6	O
OUZY	JACQUES	13	12	12,5	11	14	12,5	12	12	12,4	9	11,5	O
BONO	JEAN	12	11	11,5	10	13	11,5	12	12	11,6	3	11,3	O
ASSIN	MARC	12	11	11,5	10	9	9,5	15	15	11,4	3	11,1	O
TAREMBOIS	GUY	14	13	13,5	12	12	12	7	7	11,6	2	11,4	N
VESSELLE	AUDE	10	10	10	8	6	7	10	10	8,8	4	8,4	N
NIOHRANGIN	NICOLAS	5	4	4,5	10	11	10,5	11	11	8,2	5	7,7	N
ALIZAN	GASPARD	9	6	7,5	10	9	9,5	4	4	7,6	9	6,7	N
ZEBLOUSE	AGATHE	1	1	1	10	10	10	5	5	5,4	5	4,9	N

```
CALL affichagePV('S3');
```

PALLEJA	XAVIER	18	17	17,5	14	16	18	16	16	16	16,6	0	16,6	O
OUTAN	LAURENT	13	17	15	14	17	11	14	14	14	14,4	1	14,4	O
TRISER	JESSICA	14	16	15	14	13	12	13	15	15	14,2	1	14,2	O
TERRIEUR	ALAIN	13	14	13,5	12	12	10,5	11,5	15	15	13	1	13	O
ORAQUE	ANNE	12	13	12,5	14	13	12	13	14	14	13	0	13	O
NEMARD	JEAN	12	10	11	11	12	8,5	10,5	14	14	11,4	2	11,2	O
CROQUE	ODILE	12	14	13	9	10	8	9	12	12	11,2	8	10,4	O
STICKE	SOPHIE	17	19	18	8	9	4	7	14	14	12,8	6	12,2	N
STICKO	JUDAS	10	5	7,5	14	16	6	12	12	12	10,2	2	10	N
MICOTON	MYLENE	9	8	8,5	11	10	4,5	8,5	10	10	8,8	8	8	N
BRICOT	JUDAS	8	6	7	13	12	,5	8,5	11	11	8,4	4	8	N

```
CALL affichagePV('S4');
```

PALLEJA	XAVIER	19	20	19,5	16,5	16,5	18	0	18	O
STICKE	SOPHIE	17	18	17,5	15,5	15,5	16,5	3	16,2	O
OUTAN	LAURENT	13	14	13,5	16,5	16,5	15	1	15	O
TRISER	JESSICA	14	14	14	15	15	14,5	1	14,5	O
ORAQUE	ANNE	13	14	13,5	14,5	14,5	14	0	14	O
TERRIEUR	ALAIN	13	11	12	14	14	13	2	12,8	O
NEMARD	JEAN	12	11	11,5	12,5	12,5	12	2	11,8	O
STICKO	JUDAS	12	9	10,5	13,5	13,5	12	3	11,7	O
CROQUE	ODILE	9	9	9	13	13	11	1	11	O
BRICOT	JUDAS	8	6	7	11	11	9	2	8,8	N
MICOTON	MYLENE	8	5	6,5	8,5	8,5	7,5	4	7,1	N