Périgée - Eléments de correction :

Première partie – Les blocs PL/SQL

1) Bloc PL/SQL simple et paquetage `DBMS_OUTPUT`.

```
SET SERVEROUTPUT ON
DECLARE
   v_nbEtudiants NUMBER ;
   v_idGroupe Groupes.idGroupe%TYPE;
BEGIN
   v_idGroupe := 'Q1' ;
   SELECT COUNT(*) INTO v_nbEtudiants
   FROM Etudiants
   WHERE idGroupe = v_idGroupe;
   DBMS_OUTPUT.PUT_LINE('Il y a ' || v_nbEtudiants ||
                        ' étudiant(s) dans le groupe ' || v_idGroupe);
END;
```

2) Interactivité grâce aux variables de substitution.

```
SET SERVEROUTPUT ON
   ACCEPT s_idGroupe PROMPT 'Entrer l''identifiant du Groupe : ';
DECLARE
   v_nbEtudiants NUMBER ;
BEGIN
   SELECT COUNT(*) INTO v_nbEtudiants
   FROM Etudiants
   WHERE idGroupe = '&s_idGroupe';
   DBMS_OUTPUT.PUT_LINE('Il y a ' || v_nbEtudiants || '
                        étudiant(s) dans le groupe ' || '&s_idGroupe');
END;
```

3) Structures de contrôle : la conditionnelle (`IF … THEN … ELSE … END IF`).

```
SET SERVEROUTPUT ON
   ACCEPT s_idGroupe PROMPT 'Entrer le code du Groupe : ';
DECLARE
   v_nbEtudiants NUMBER ;
   v_nbGroupes NUMBER;
BEGIN
   SELECT COUNT(*) INTO v_nbGroupes
   FROM Groupes
   WHERE idGroupe = '&s_idGroupe';
   IF v_nbGroupes = 1 THEN
      SELECT COUNT(*) INTO v_nbEtudiants
      FROM Etudiants
      WHERE idGroupe = '&s_idGroupe';
      DBMS_OUTPUT.PUT_LINE('Il y a ' || v_nbEtudiants || '
                           étudiant(s) dans le groupe ' || '&s_idGroupe');
   ELSE
      DBMS_OUTPUT.PUT_LINE('Il n''y a pas de groupe ' || '&s_idGroupe');
   END IF;
END;
```

4) Les Exceptions : l'exception `NO_DATA_FOUND`.

```
SET SERVEROUTPUT ON
   ACCEPT s_idGroupe PROMPT 'Entrer le code du Groupe : ';
DECLARE
   v_nbEtudiants NUMBER ;
   v_idGroupe Groupes.idGroupe%TYPE;
BEGIN
   SELECT idGroupe INTO v_idGroupe
   FROM Groupes
   WHERE idGroupe = '&s_idGroupe';
   SELECT COUNT(*) INTO v_nbEtudiants
   FROM Etudiants
   WHERE idGroupe = '&s_idGroupe';
   DBMS_OUTPUT.PUT_LINE('Il y a ' || v_nbEtudiants || '
                        étudiant(s) dans le groupe ' || '&s_idGroupe');
EXCEPTION
   WHEN NO_DATA_FOUND THEN
      DBMS_OUTPUT.PUT_LINE('Il n''y a pas de groupe ' || '&s_idGroupe');
END;
```


5) Les variables `%ROWTYPE`.

```
SET SERVEROUTPUT ON
DECLARE
   v_idEtudiant Etudiants.idEtudiant%TYPE;
   rty_etudiant Etudiants%ROWTYPE;
BEGIN
   DBMS_OUTPUT.ENABLE ;
   v_idEtudiant := 'E1';
   SELECT * INTO rty_etudiant
   FROM Etudiants
   WHERE idEtudiant = v_idEtudiant;
   DBMS_OUTPUT.PUT_LINE('Identifiant étudiant : ' || rty_etudiant.idEtudiant);
   DBMS_OUTPUT.PUT_LINE('Nom étudiant : ' || rty_etudiant.nomEtudiant);
   DBMS_OUTPUT.PUT_LINE('Prénom étudiant : ' || rty_etudiant.prenomEtudiant);
   DBMS_OUTPUT.PUT_LINE('Sexe étudiant : ' || rty_etudiant.sexeEtudiant);
   DBMS_OUTPUT.PUT_LINE('Date naissance étudiant : ' ||
                                          rty_etudiant.dateNaissanceEtudiant);
   DBMS_OUTPUT.PUT_LINE('Groupe étudiant : ' || rty_etudiant.idGroupe);
EXCEPTION
   WHEN NO_DATA_FOUND THEN
      DBMS_OUTPUT.PUT_LINE('L''etudiant : ' || v_idEtudiant || ' n''existe pas');
END;
```

Deuxième partie – Les Procédures et Fonctions stockées :

6) Fonction stockée nbEtudiantsParGroupe.

```
CREATE OR REPLACE FUNCTION nbEtudiantsParGroupe(
                                        p_idGroupe IN Groupes.idGroupe%TYPE)
                                                        RETURN NUMBER IS
   v_idGroupe Groupes.idGroupe%TYPE;
   v_nbEtudiants NUMBER;
BEGIN
   SELECT idGroupe INTO v_idGroupe
   FROM Groupes
   WHERE idGroupe = p_idGroupe;
   SELECT COUNT(*) INTO v_nbEtudiants
   FROM Etudiants
   WHERE idGroupe = p_idGroupe;
   RETURN v_nbEtudiants;
EXCEPTION
   WHEN NO_DATA_FOUND THEN
      RETURN NULL;
END;

// ici, la section exception est facultative car on souhaite retourner NULL
// lorsqu'une exception est levée (c'est ce qui sera fait par défaut si
// cette section n'est pas présente)
```

7) Fonction stockée nbEtudiantsParPromotion.

```
CREATE OR REPLACE FUNCTION nbEtudiantsParPromotion(
                                  p_idPromotion IN Promotions.idPromotion%TYPE)
                                                        RETURN NUMBER IS
   v_nbEtudiants NUMBER;
BEGIN
   SELECT SUM(nbEtudiantsParGroupe(idGroupe)) INTO v_nbEtudiants
   FROM Groupes
   WHERE idPromotion = p_idPromotion;
   RETURN v_nbEtudiants;
END;


UPDATE Promotions
SET nbEtudiantsPromotion = nbEtudiantsParPromotion(idPromotion);
```

8) Procédure stockée affichageInfosEtudiant.

```
CREATE OR REPLACE PROCEDURE affichageInfosEtudiant(
                              p_idEtudiant IN Etudiants.idEtudiant%TYPE) IS
   rty_etudiant Etudiants%ROWTYPE;
BEGIN
   SELECT * INTO rty_etudiant
   FROM Etudiants
   WHERE idEtudiant = p_idEtudiant;
   DBMS_OUTPUT.PUT_LINE('Identifiant étudiant : ' || rty_etudiant.idEtudiant);
   DBMS_OUTPUT.PUT_LINE('Nom étudiant : ' || rty_etudiant.nomEtudiant);
   DBMS_OUTPUT.PUT_LINE('Prénom étudiant : ' || rty_etudiant.prenomEtudiant);
   DBMS_OUTPUT.PUT_LINE('Sexe étudiant : ' || rty_etudiant.sexeEtudiant);
   DBMS_OUTPUT.PUT_LINE('Date naissance étudiant : ' ||
                                    rty_etudiant.dateNaissanceEtudiant);
   DBMS_OUTPUT.PUT_LINE('Groupe étudiant : ' || rty_etudiant.idGroupe);
EXCEPTION
   WHEN NO_DATA_FOUND THEN
      DBMS_OUTPUT.PUT_LINE('L''etudiant : ' || p_idEtudiant || ' n''existe pas');
END;
```

9) Procédure stockée miseAJourCoefficientModules.

```
CREATE OR REPLACE PROCEDURE miseAJourCoefficientModules IS
BEGIN
   UPDATE Modules mo
   SET coefficientModule = (SELECT SUM(coefficientMatiere)
                            FROM Matieres ma
                            WHERE ma.idModule = mo.idModule);
END;
```

Troisième partie – Les Curseurs :

10) Procédure stockée affichageNotesEtudiant.

```
CREATE OR REPLACE PROCEDURE affichageNotesEtudiant(
                               p_idEtudiant IN Etudiants.idEtudiant%TYPE) IS
   CURSOR curs_notes IS SELECT nomMatiere, note
                        FROM Notes n
                             JOIN Matieres ma ON ma.idMatiere = n.idMatiere
                        WHERE idEtudiant = p_idEtudiant;
   v_ligne curs_notes%ROWTYPE;
BEGIN
   OPEN curs_notes;
   FETCH curs_notes INTO v_ligne;
   WHILE (curs_notes%FOUND) LOOP
      DBMS_OUTPUT.PUT_LINE(v_ligne.nomMatiere || ' : ' || v_ligne.note);
      FETCH curs_notes INTO v_ligne;
   END LOOP;
   CLOSE curs_notes;
END;
```

ou encore

```
CREATE OR REPLACE PROCEDURE affichageNotesEtudiant(
                               p_idEtudiant IN Etudiants.idEtudiant%TYPE) IS
   CURSOR curs_notes IS SELECT nomMatiere, note
                        FROM Notes n
                             JOIN Matieres ma ON ma.idMatiere = n.idMatiere
                        WHERE idEtudiant = p_idEtudiant;
BEGIN
   FOR v_ligne IN curs_notes LOOP
      DBMS_OUTPUT.PUT_LINE(v_ligne.nomMatiere || ' : ' || v_ligne.note);
   END LOOP;
END;
```

ou encore (voir page suivante)


```
CREATE OR REPLACE PROCEDURE affichageNotesEtudiant(
                               p_idEtudiant IN Etudiants.idEtudiant%TYPE) IS
BEGIN
   FOR v_ligne IN (SELECT nomMatiere, note
                   FROM Notes n
                        JOIN Matieres ma ON ma.idMatiere = n.idMatiere
                   WHERE idEtudiant = p_idEtudiant) LOOP
      DBMS_OUTPUT.PUT_LINE(v_ligne.nomMatiere || ' : ' || v_ligne.note);
   END LOOP;
END;
```

11) Procédure stockée affichageNotesEtudiantSemestre.

```
CREATE OR REPLACE PROCEDURE affichageNotesEtudiantSemestre(
                                  p_idEtudiant IN Etudiants.idEtudiant%TYPE,
                                   p_idSemestre IN Semestres.idSemestre%TYPE) IS
   v_inscrit NUMBER;
BEGIN
   SELECT COUNT(*) INTO v_inscrit
   FROM Etudiants e
        JOIN Groupes g ON g.idGroupe = e.idGroupe
        JOIN Semestres s ON s.idPromotion = g.idPromotion
   WHERE idEtudiant = p_idEtudiant
   AND idSemestre = p_idSemestre;
   IF v_inscrit = 1 THEN
      FOR v_ligne IN (SELECT nomMatiere, note
                      FROM Notes n
                           JOIN Matieres ma ON ma.idMatiere = n.idMatiere
                           JOIN Modules mo ON ma.idModule = mo.idModule
                      WHERE idSemestre = p_idSemestre
                      AND idEtudiant = p_idEtudiant) LOOP
         DBMS_OUTPUT.PUT_LINE(v_ligne.nomMatiere || ' : ' || v_ligne.note);
      END LOOP;
   ELSE
      DBMS_OUTPUT.PUT_LINE('L''étudiant ' || p_idEtudiant || ' n''est pas inscrit
                                                        au semestre ' || p_idSemestre);
   END IF;
END;
```

12) Procédure stockée affichageToutEtudiantSemestre.

```
CREATE OR REPLACE PROCEDURE affichageToutEtudiantSemestre(
                                  p_idEtudiant IN Etudiants.idEtudiant%TYPE,
                                   p_idSemestre IN Semestres.idSemestre%TYPE) IS
BEGIN
   affichageInfosEtudiant(p_idEtudiant);
   DBMS_OUTPUT.NEW_LINE();
   affichageNotesEtudiantSemestre(p_idEtudiant, p_idSemestre);
END;
```

13) Procédure stockée affichageAbsencesParPromotion.

```
CREATE OR REPLACE PROCEDURE affichageAbsencesParPromotion (
                             p_idPromotion IN Promotions.idPromotion%TYPE) IS
BEGIN
   FOR v_groupe IN (SELECT idGroupe, nbEtudiantsParGroupe(idGroupe) as nbEtudiants
                    FROM Groupes
                    WHERE idPromotion = p_idPromotion
                    ORDER BY idGroupe ASC) LOOP
      DBMS_OUTPUT.PUT_LINE('Groupe : ' || v_groupe.idGroupe || ' ' ||
                           ' (' || v_groupe.nbEtudiants || ' étudiants)');

      FOR v_etud IN (SELECT nomEtudiant, prenomEtudiant,
                                         COUNT(idAbsence) AS nbAbsences
                     FROM Etudiants e
                          LEFT OUTER JOIN Absences a ON a.idEtudiant = e.idEtudiant
                     WHERE idGroupe = v_groupe.idGroupe
                     GROUP BY e.idEtudiant, nomEtudiant, prenomEtudiant
                     ORDER BY nbAbsences DESC, nomEtudiant ASC) LOOP
         DBMS_OUTPUT.PUT_LINE('-----> ' || v_etud.nomEtudiant || ' ' ||
                                    v_etud.prenomEtudiant || ' a été absent ' ||
                                                 v_etud.nbAbsences || ' fois');
      END LOOP;
   END LOOP;
END;
```