Université Mohamed 1er
Faculté des Sciences
Département d'informatique

Filière SMI Semestre 6

Examens corrigés
Module : Bases de données avancées

Années universitaires 2014/2021

Professeur : M. MOUSSI

Université Mohamed 1er **Année universitaire : 2014/2015**
Faculté des Sciences **Filière SMI, semestre 6.**
Département de Mathématiques
et Informatique

Bases de données avancées
Examen ordinaire
Durée : 1h 30min

Questions de cours :

1. Donner en langage SQL les requêtes permettent (sous Oracle) de :
   a) Créer un utilisateur.
   b) Donner le privilège de connexion à la base de données pour un utilisateur.
   c) Retirer le privilège de création de vue d'un utilisateur.

2. Pour les déclarations des variables suivantes déterminer, avec justification, celles qui sont correctes et celles qui sont incorrectes :

Déclaration	Correcte/ incorrecte	Justification
DECLARE *V1 VARCHAR2(4) ;*		
DECLARE *Var1, var2, var3 NUMBER(2) ;*		
DECLARE *Variable$ BOOLEAN ;*		
DECLARE *2emeVar DATE ;*		
DECLARE *Var_N NUMBER(4,1):=15.43 ;*		
DECLARE *Variable varchar2(4):=23ab ;*		

Exercice 1.
Ecrire en langage PL/SQL une procédure qui prend en paramètre un entier et affiche si cet entier est **glouton** ou non.
On dit qu'un entier est ***"glouton"*** s'il possède strictement plus de diviseurs que chacun des nombres inférieurs à lui. (Exemples : **12** est glouton ; **15** n'est pas glouton).

Exercice 2.
On considère le schéma relationnel suivant :
etudiant (NE, Nom, Adresse, Ville)
matiere (NM, Nom, Coefficient, type)
evaluation (NE, NM, DateEval, Note)

Ecrire un programme PL/SQL qui permet de Saisir un certain numéro de matière et vérifie si toutes les notes obtenues dans cette matière (par tous les étudiants) est inférieur à 10 alors le champ *type* de la table matiere sera mise à ***'not_easy'***, sinon il sera mise à ***'easy'.***

NB. Le programme doit utiliser les curseurs explicites.

Solution Examen ordinaire 2014/2015

Questions de cours :

Voir le cours

Exercice 1.

```
create or replace  procedure glouton(n in number )
is
begin
        declare
          ng number:=0;
          nd number;
          r boolean:=true;
        begin
        --calcul du nombre de diviseur de n
          for i in 1..n  loop
             if mod(n,i)=0 then
                ng:=ng+1;
             end if;
          end loop;
        --calcul du nombre de diviseur de chaque entier
           for p in 1..n loop
              nd:=0;
             for i in 1..p loop
                if mod(p,i)=0 then
                  nd:=nd+1;
                end if;
             if nd>ng then
             r:=false;
             exit;
            end if;
          end loop;
        end loop;

        if r=false then
           dbms_output.put_line('non gloutton');
        else
           dbms_output.put_line('gloutton');
        end if;
        end;
end;
```

Exercice 2:

```
accept numero

declare
        num matiere.nm%type:=&numero;
        note_et evaluation.note%type;
        --nombre des lingnes inf. a 10
        n number:=0 ;
        --nombre total des lignes
        nt number:=0;
        cursor c_note is
         select note from evaluation
         where nm=num;
begin
        for c in c_note loop
          nt:=nt+1;
          if c.note<10 then
             n:=n+1;
          end if;
        end loop;
        dbms_output.put_line('le nombre total est' || nt);
        dbms_output.put_line('le nombre est' || n);
        --toutes les notes sont inferieurs à 10
        if nt=n then
           update matiere
           set type='not_easy'
           where
          nm=&numero;
        else
           update matiere
           set type='easy'
          where
          nm=&numero;
        end if;
end;
```

Université Mohamed 1er
Faculté des Sciences
Département de Mathématiques et Informatique

Année universitaire : 2014/2015
Filière SMI, semestre 6.

Bases de données avancées
Examen rattrapage
Durée : 1h 30min

Questions de cours :

3. Donner la syntaxe générale pour créer une vue.
4. Quelle est l'utilité de la commande *WITH CHECK OPTION?*
5. Quel est l'effet de suppression d'une vue sur les tables référencées.

Exercice 1.

Ecrire en langage PL/SQL une procédure qui prend en paramètre un entier et affiche si cet entier est **square-free** ou non.
On dit qu'un entier positif n est **"square-free"** si pour tous les nombres premiers p le carré de p (p^2) ne divise pas n. (Exemple : 10 est square-free, 18 n'est pas square-free puisqu'il est divisible par 3^2).

Exercice 2.

On considère le schéma relationnel suivant :
etudiant (NE, Nom, Adresse, Ville)
matiere (NM, Nom, Coefficient, état)
evaluation (NE, NM, DateEval, Note)

Ecrire un programme PL/SQL qui lit un certain nom de matière et une date, puis vérifie

- si aucune évaluation n'a eu lieu dans cette matière après cette date alors la matière sera supprimée de la table matière,
- sinon la valeur de l'état de la matière sera mise à « ok ».

NB. Le programme doit utiliser les curseurs et les exceptions utilisateur.

Solution Examen rattrapage 2014/2015

Questions de cours :

Voir le cours

Exercice 1.

```
create or replace  procedure squarefree(n in number )
is
begin
        declare
         ng number:=0;--nombre de diviseur premiers en carré
          np number:=0;
          nd number;
        --  r boolean:=true;
        begin
        --calcul du nombre de diviseur en double des premiers de n
          for i in 2..n  loop
            --je cherche s'il est premier
                  for j in 2..i-1 loop
                     if mod(i,j)=0 then
                    np:=np+1;
                    end if;
                  end loop;
              if np = 0 then
                   --je cherche si ce premier est divisible en carré
                     if mod(n,i*i)=0 then
                    --   r:=false;
                   --exit
                     ng:=ng+1;
                     end if;
                end if;
          end loop;
    if ng > 0 then
           dbms_output.put_line(ng || 'non squarefree');
        else
           dbms_output.put_line(ng || 'squarefree');
    end if;
 end;
 end;
```

Exercice 2.

```
accept nom
accept date_ev

DECLARE
   n number:=0;
   excep1 EXCEPTION;
   excep2 EXCEPTION;
   CURSOR curs_ev is
      select dateEval
```

```
        from evaluation e, matiere m
        where e.NM=m.NM
        and m.Nom='&nom'
        and e.dateEval > &date_ev;

BEGIN
--ou bien utiliser curs_ev%rowcount c'est à dire
      -- open curs_ev;
       --dbms_output.put_line(curs_ev%rowcount);
       --close curs_ev;
   for c in curs_ev loop
    exit when curs_ev%notfound;
    n:=n+1;
   end loop;
  dbms_output.put_line('le nombre est' || n);
if n=0 then raise excep1;
else raise excep2;
end if;
EXCEPTION
       when excep1 then
   delete from  matiere
   where
  nom='&nom';
      when excep2 then
   update matiere
   set etat='ok'
  where
  nom='&nom';
END;
```

Université Mohamed 1er **Année universitaire : 2015/2016**
Faculté des Sciences **Filière SMI, semestre 6.**
Département d'Informatique

Bases de données avancées
Examen ordinaire
Durée : 1h 30min

Questions de cours :

Soit la vue *vue_etud (Nom, Adresse)* sur la table *Etudiant (NE clé primaire de type entier, Nom de type char(10), Adresse de type char(30))*.

1) Donner la syntaxe de création de la vue *vue_etud*.
2) Est-il possible d'insérer des données à travers la cette vue ?
   La réponse doit être Justifiée.
3) Donner la commande permettant d'attribuer le privilège de sélection sur *vue_etud*.

Exercice 1.

1) Ecrire une fonction en langage PL/SQL nommée *test_parfait* qui prend en paramètre un entier *n* et teste si cet entier est parfait ou non.
2) En utilisant la fonction *test_parfait*, écrire un programme PL/SQL qui lit un entier *m* et affiche les nombres parfaits inférieurs à cet entier.

N. B. Un nombre est dit parfait s'il est égal à la somme de ses diviseurs propres.
(**Exemple** : le nombre 6 est parfait, 6=1+2+3)

Exercice 2.

On considère le schéma relationnel suivant :
etudiant (NE, Nom_et, Adresse, Ville)
matiere (NM, Nom_mat, Coefficient, type)
evaluation (NE, NM, DateEval, Note)
meilleurs (NM, Nom_et, note)
Ecrire un programme qui permet de

1) Saisir un entier *n* et un numéro de matière.
2) Récupérer le nom et la note des *n* meilleurs étudiants selon leurs notes dans cette matière.
3) Enregistrer le numéro de la matière, les noms des étudiants et leurs notes dans la table meilleurs.

N.B : Le programme doit utiliser les curseurs.

Solution examen ordinaire 2015/2016

Questions de cours :

Voir le cours

Exercice 1.

1)
```
CREATE OR REPLACE FUNCTION  test_parfait( n in number )
RETURN BOOLEAN
AS
BEGIN
  DECLARE
    S number:= 0;
    R BOOLEAN:= FALSE;
  BEGIN
    FOR  i IN 1..n-1 loop
      IF MOD(n,i)=0 THEN
        S:=S+i;
      END IF;
    END LOOP;
    IF S=n THEN
       R:= TRUE;
    END IF;
    RETURN R;
  END;
END;
/
```

2)
```
ACCEPT  m
BEGIN
FOR  i IN 1..&m loop
  IF test_parfait(i)=TRUE THEN
    DBMS_OUTPUT.PUT_LINE(i);
  END IF;
END LOOP;
END;
/
```

Exercice 2.

```
ACCEPT  n
ACCEPT  num_mat
DECLARE
  lg_meil meilleurs%ROWTYPE ;
  CURSOR  meil_curseur  IS
    SELECT v.NM, e.nom_et, v.note
    FROM etudiant e, evaluation v
    WHERE
    v.NE=e.NE
    and v.NM=&num_mat
```

```
    OEDER BY v.note DESC;
BEGIN
  OPEN meil_curseur;
  FOR  i IN 1..&n  LOOP
      FETCH  meil_curseur INTO  lg_meil;
      INSERT  INTO  meilleurs VALUES (lg_meil.NM, lg_meil.Nom_et, lg_meil.note ) ;
  END LOOP ;
  CLOSE meil_curseur ;
END ;
/
```

Université Mohamed 1er **Année universitaire : 2015/2016**
Faculté des Sciences **Filière SMI, semestre 6.**
Département d'Informatique

Bases de données avancées
Examen rattrapage
Durée : 1h 30min

Questions de cours :

1) Pour les déclarations des variables suivantes en PL/SQL déterminer, avec justification, celles qui sont correctes et celles qui sont incorrectes :

Déclaration	Correcte/ incorrecte	Justification
DECLARE *Var@N NUMBER(4,1):=522.6 ;*		
DECLARE *Var# BOOLEAN ;*		
DECLARE *Var1, var2 NUMBER ;*		
DECLARE *Var VARCHAR2(3) ;*		
DECLARE *V_1ere DATE ;*		
DECLARE *Variable varchar2(4):=a12b ;*		

2) Donner l'instruction permettant d'exécuter les fonctions sous SQL*Plus.

Exercice 1.

Deux nombres entiers *n* et *m* sont dits *amicaux* si la somme des diviseurs de *n* (*n* non compris) est égale à *m* et la somme des diviseurs de *m* (*m* non compris) est égale à *n*. (**Exemple** : 220 et 284 sont amicaux)

1) Ecrire une fonction nommée *fct_ami* qui prend en paramètre deux entiers *n* et *m* et retourne une valeur booléenne selon qu'ils sont amicaux ou non.

2) En utilisant la fonction *fct_ami*, écrire un programme PL/SQL qui lit deux entiers *N* et *M* et affiche un message selon qu'ils sont amicaux ou non.

Exercice 2.

On considère le schéma relationnel suivant :
etudiant (NE, Nom_et, Adresse, Ville)
matiere (NM, Nom_mat, Coefficient, type)
evaluation (#NE,# NM, DateEval, Note)

Ecrire un bloc PL/ SQL qui permet de lire un entier *n* et un certain numéro de matière et qui vérifie :

- Si le nombre des étudiants ayant obtenu la note la plus haute ou bien la note la plus basse est inférieur à *n* alors le programme affiche les noms de ces étudiants.
- Sinon le programme affiche le message « le nombre des étudiants est supérieur à *n* »

N.B : Le bloc doit utiliser les curseurs et les exceptions.

Solution examen rattrapage 2015/2016

Exercice 1.

1)

```
CREATE OR REPLACE FUNCTION  fct_ami( n in number, m in number )
RETURN BOOLEAN
AS
BEGIN
  DECLARE
     S1 number:= 0;
     S2 number:= 0;
     R BOOLEAN:= FALSE;
  BEGIN
     FOR  i IN 1..n-1 loop
        IF MOD(n,i)=0 THEN
           S1:=S1+i;
        END IF;
     END LOOP;
     FOR  i IN 1..m-1 loop
        IF MOD(m,i)=0 THEN
           S2:=S2+i;
        END IF;
     END LOOP;

     IF S1=m and S2=n THEN
          R:= TRUE;
     END IF;
     RETURN R;
  END;
END;
/
```

2)

```
ACCEPT  N
ACCEPT  M

DECLARE
    R BOOLEAN;
BEGIN
    R:=fct_ami(&N,&M);
    IF R=TRUE THEN
       DBMS_OUTPUT.PUT_LINE('les deux entiers sont amicaux');
    ELSE
       DBMS_OUTPUT.PUT_LINE('les deux entiers ne sont pas amicaux');
    END IF;
 END;
/
```

Exercice 2.

```
--saisie des valeurs
ACCEPT  n
ACCEPT  num_mat

DECLARE
        nb_max  number;
        nb_min  number;
        except exception;
        CURSOR liste_etudiant is
        select a.Nom
        from etudiant a join evaluation b on a.NE=b.NE
        where
        b.Note=(select max(Note) from evaluation where b.nm= &num_mat)
        or b.Note=(select min(Note) from evaluation where b.nm= &num_mat)
        and
        nm=&num_mat;

BEGIN
--récupérer le nombre des étudiants ayant obtenu la note max dans la matiere num_mat
        select count(ne) as "nombre ayant max" into nb_max from evaluation
        where
        note=(select max(note) from evaluation where nm=&num_mat)
        and nm=&num_mat;
--récupérer le nombre des étudiants ayant obtenu la note min dans la matiere num_mat
        select count(ne) as "nombre ayant min" into nb_min
        from evaluation
        where
        note=(select min(note) from evaluation where nm=&num_mat)
        and nm=&num_mat;
--si l'un des deux nombres  <=n j'affiche la liste des étudiant
        IF nb_max <= &n or nb_min<=&n THEN
          FOR c IN liste_etudiant
           LOOP
             DBMS_OUTPUT.PUT_LINE(c.nom) ;
           END LOOP ;
-- sinon je déclenche l'exception  qui contient le message
            ELSE  RAISE except;
         END IF;

EXCEPTION
        WHEN except THEN
          DBMS_OUTPUT.PUT_LINE('le nombre est superieur') ;
END ;
/
```

Université Mohamed 1er **Année universitaire : 2016/2017**
Faculté des Sciences **Filière SMI, semestre 6.**
Département d'Informatique

Bases de données avancées
Examen ordinaire
Durée : 1h 30min

Exercice 1. Soit la table ″Personne″ :

Num	**Nom**	**Age**	**Ville**
1	Ahmed	20	Oujda
2	Aicha	23	Casa
3	Mohammed	30	Rabat
4	Zakaria	15	Oujda

1) Donner **avec justification** le résultat d'exécution du programme suivant :

```
DECLARE
  ligne Personne%ROWTYPE ;
  n  Personne.Num%TYPE :=2 ;
BEGIN
  SELECT * INTO ligne FROM  Personne ;
  DBMS_OUTPUT.PUT_LINE ( n ) ;
  DBMS_OUTPUT.PUT_LINE ( ligne.Num) ;
END;
```

2) Quelle est la signification des deux privilèges système suivants :
   a. *CREATE VIEW.*
   b. *CREATE ANY VIEW.*

3) Créer une vue qui ne contient que les personnes qui ont l'âge supérieur ou égale à 21 de sorte que toutes les instructions de modification de données exécutées sur la vue respectent ce critère (âge >=21).

4) Créer un index nommé *nom_idx*, avec unicité des valeurs indexées, sur la colonne Nom de la table Personne.

5) Supprimer l'index *nom_idx* de la colonne Nom.

Exercice 2. On considère le schéma relationnel suivant :
etudiant (NE, Nom_et, Adresse, Ville, resultat)
matiere (NM, Nom_mat, Coefficient, type)
evaluation (NE, NM, DateEval, Note)
Ecrire un programme PL/SQL qui permet de

1) Saisir un certain numéro d'étudiant.
2) Si l'étudiant n'existe pas, le programme affiche le message ″n'existe pas″.
3) Si l'étudiant existe alors :
   a. Si toutes ses notes sont supérieures à 10 alors l'attribut *resulta*t sera mis à jour avec la valeur ″ok″.
   b. Sinon l'attribut *resulta*t sera mis à jour avec la valeur ″ non ok″.

Exercice 3.
Ecrire un programme PL/SQL qui permet de

- Saisir un entier n,
- Calculer et afficher le nombre des diviseurs de n.

Solution Examen ordinaire 2016/2017

Exercice 1. Voir cours

Exercice 2.

```
accept numero
DECLARE
        nombre NUMBER;
        nombre_mat NUMBER ;
        --nom_et etudiant.nom%TYPE;
        --etud etudiant%rowtype;
BEGIN
   --je cherche si l'etudiant existe
  SELECT count(*) into nombre FROM etudiant
  WHERE NE=&numero;
     if nombre>0 then
        --DBMS_OUTPUT.PUT_LINE('existe');
        --je calcule le nombre des matieres avec note <10
         select count(b.NM) into nombre_mat
         from  evaluation
         where Note<10
          and NE=&numero;
            if n=0 then
                    --DBMS_OUTPUT.PUT_LINE('toutes les notes sont >= 10');
                 Update etudiant
                Set resultat='ok'
                Where NE=&numero ;
                Else
                  --DBMS_OUTPUT.PUT_LINE( 'on a des notes <');
                Update etudiant
                Set resultat= 'non ok'
                Where NE=&numero ;
          end if ;
 else
             DBMS_OUTPUT.PUT_LINE('n existe pas');
end if;
end;
```

Exercice 3.

```
accept n

declare
    nd number:=0;--nombre des diviseurs
    m number:=&n;
begin
    --calcul du nombre de diviseurs de n
     for i in 1..&n  loop
            if mod(m,i)=0 then
               nd:=nd+1;
```

```
                end if;
        end loop;
         dbms_output.put_line(nd);
end;
```

Université Mohamed 1er **Année universitaire : 2016/2017**
Faculté des Sciences **Filière SMI, semestre 6.**
Département d'Informatique

Bases de données avancées
Examen rattrapage
Durée : 1h 30min

Exercice 1.

Soit la table ″Personne″ et le programme PL/SQL suivants :

Num	**Nom**	**Age**	**Ville**
1	Ahmed	20	Oujda
2	Aicha	23	Casa
3	Mohammed	30	Rabat
4	Zakaria	15	Oujda

```
CREATE OR REPLACE  FUNCTION  pers_fonct ( a IN Personne.Age%Type )
RETOURN  NUMBER
IS
n , i number ;
BEGIN
  i :=1;
  SELECT COUNT(Num) INTO n FROM  Personne ;
  WHERE Age > a ;
  IF  n=0  THEN
  DBMS_OUTPUT.PUT_LINE('premier cas') ;
  ELSEIF  n=1 THEN
  DBMS_OUTPUT.PUT_LINE(' deuxieme cas') ;
  i:=n+1;
  ELSEIF  n=2 THEN
  DBMS_OUTPUT.PUT_LINE(' troisieme cas') ;
  ELSE  THEN
  DBMS_OUTPUT.PUT_LINE('quatrieme cas') ;
  i:=n+2 ;
  END IF ;
 RETOURN  i
END ;
```

1) Corriger les erreurs de ce programme.
2) Donner l'instruction d'exécution de la fonction *pers_fonct* sous SQL plus.
3) Donner le résultat d'exécution de la fonction *pers_fonct* pour les valeurs a=10, a= 30 et a= 23.

Exercice 2.

On considère le schéma relationnel suivant :

etudiant (NE, Nom_et, Adresse, Ville, resultat)
matiere (NM, Nom_mat, Coefficient, type)
evaluation (NE, NM, DateEval, Note)

Ecrire un programme PL/SQL qui permet de :

1) Saisir un certain numéro d'étudiant.
2) Si cet étudiant a été évalué au moins une fois alors :
   a. Si sa moyenne générale (la moyenne des matières dans lesquelles il a été évalué) est supérieure ou égale à 10 alors toutes ses notes seront augmentées par la valeur 1.
   b. Sinon toutes ses notes seront augmentées par la valeur 2.
3) Si cet étudiant n'a jamais été évalué alors le programme affiche le message *"Etudiant jamais évalué "*

Exercice 3.

Un entier naturel strictement supérieur à 1 est dit *hautement composé* lorsqu'il possède strictement plus de diviseurs que n'importe quel entier qui le précède.

Exemple :

Les nombres hautement composés inférieurs ou égaux à 100 sont : 2, 4, 6, 12, 24, 36, 48 et 60 (par exemple : 6 possède strictement plus de diviseurs que 2, 3, 4 et 5).

Question :

Ecrire un programme PL/SQL qui permet de saisir un entier n et d'afficher les entiers inférieurs à n qui sont hautement composés.

Solution de l'examen rattrapage 2016/2017

Exercice 1.

Soit la table ″Personne″ et le programme PL/SQL suivants :

Num	**Nom**	**Age**	**Ville**
1	Ahmed	20	Oujda
2	Aicha	23	Casa
3	Mohammed	30	Rabat
4	Zakaria	15	Oujda

1) *CREATE OR REPLACE FUNCTION pers_fonct (a IN Personne.Age%Type)*
RETURN *NUMBER*
IS
n number;
i number *;*
BEGIN
i :=1 ;
*SELECT COUNT(Num) INTO n FROM Personne –**pas de point virgule***
WHERE Age > a;

IF n=0 THEN
DBMS_OUTPUT.PUT_LINE('le premier cas');
ELSIF *n=1 THEN*
DBMS_OUTPUT.PUT_LINE('le deuxieme cas');
i:=n+1;
ELSIF *n=2 THEN*
DBMS_OUTPUT.PUT_LINE('le trois cas');
ELSE ***– pas de THEN***
DBMS_OUTPUT.PUT_LINE('le quatr cas');
i:=n+2;
END IF;
RETURN *i* ***;***
END;

2) *select pers_fonct(valeur) from dual;*

3) *a= 10 alors affiche 6 ; a= 30 alors affiche 1 et a= 23 alors affiche 2*

Exercice 2.

On considère le schéma relationnel suivant :

etudiant (NE, Nom_et, Adresse, Ville, resultat)
matiere (NM, Nom_mat, Coefficient, type)
evaluation (NE, NM, DateEval, Note)

```
DECLARE
nombre NUMBER;
moy NUMBER;
nombre_mat NUMBER ;
BEGIN
   --je cherche si l'etudiant a ete evalué
         SELECT count(*) into nombre FROM evaluation
         WHERE NE=&numero;
if nombre>0 then
DBMS_OUTPUT.PUT_LINE('evalué');
--je calcule la moyenne
select
(sum(b.Note*c.Coefficient))/sum(c.Coefficient) into moy
from etudiant a,evaluation b,matiere c
where a.NE=b.NE
and b.NM=c.NM
and b.NE=&numero
group by a.NE;

   if moy>=10 then
         DBMS_OUTPUT.PUT_LINE('la moyenne et sup. ou eg. a 10');
         Update evaluation
         Set Note=Note+1
         Where NE=&numero ;
  Else
         DBMS_OUTPUT.PUT_LINE( 'la moyenne est inf. a 10');
         Update evaluation
         Set Note=Note+2
         Where NE=&numero ;
  end if ;
ELSE
         DBMS_OUTPUT.PUT_LINE('non evalue');
end if;
end;
```

Exercice 3.

Un entier naturel strictement supérieur à 1 est dit hautement composé lorsqu'il possède strictement plus de diviseurs que n'importe quel entier qui le précède.

Exemple :

Les nombres hautement composés inférieurs ou égaux à 100 sont : 2, 4, 6, 12, 24, 36, 48 et 60.

```
accept n

declare
k number:=1;
nb number  ;
n number;
begin
```

```
for i in 2..&n loop
   --je cherche hautement
   -- debut calcul nombre div
     nb:=0;
    for j in 1..i loop
         if mod(i,j)=0 then
         nb:=nb+1;
        end if;
     end loop;
   --fin calcul nombre div
  --comparer le nombre div avec k, k est nombre div ancien
    if(nb>k) then
     DBMS_OUTPUT.PUT_LiNE(i || '  ' || nb || 'est hautement');
     DBMS_OUTPUT.PUT_LiNE(k);
     end if;
          if nb>k  then  k:=nb;
     end if;
end loop;
end;
```

Université Mohamed 1er **Année universitaire : 2017/2018**
Faculté des Sciences **Filière SMI, semestre 6.**
Département d'Informatique

Bases de données avancées
Examen ordinaire
Durée : 1h 30min

Dans toute la suite, on considère le schéma relationnel suivant :
etudiant (NE, Nom, Adresse, Ville)
matiere (NM, Nom, Coefficient, type)
evaluation (#NE, #NM, DateEval, etat)

Exercice 1.
Soit le programme PL/SQL suivant :

```
declare
  n number;
  m number:=1;
begin
  n:='aaa';
  m:=m+2;
  dbms_output.put_line(m);
  dbms_output.put_line(n);
end;
```

1) Donner **avec justification** le résultat d'exécution de ce programme.
2) En utilisant l'instruction *when others* et en supprimant la partie en gras, modifier le programme pour qu'il affiche le message *'vous avez deux affectations… !'*;

Exercice 2.
Donner en langage SQL les requêtes permettant (sous Oracle) de :
1) Créer un utilisateur nommé *user1* en lui affectant le tablespace *system* par défaut et tablespace *temp* comme tablespace temporaire.
2) Donner le privilège de connexion à la base de données pour *user1*.
3) Donner le privilège de création de table dans tous les schémas à *user1*.
4) Donner le privilège de consultation de la table *evaluation* à user1 avec l'option qui lui permet de donner ce privilège à un autre utilisateur.

Exercice 3.
Ecrire un programme PL/SQL qui permet de :
- Saisir un entier n.
- Si n est supérieur ou égal au nombre de lignes de la table matière alors le programme affiche toutes les lignes de la table matière.
- Sinon le programme affiche les n premières lignes de la table matière.

NB. Le programme doit utiliser les curseurs explicites.

Exercice 4.
Un entier naturel (différent de 1) est dit **premier** s'il possède exactement deux diviseurs distincts positifs (1 et lui-même).

1) Ecrire, en langage PL/SQL, une procédure nommé *prcd_premier* qui prend en paramètre un entier *n*,
   - si *n* est premier, la procédure affiche le message '*n est premier'* et affiche aussi son ordre (ordre signifie le 1er, le 2ème, le 3ème …etc).
   - sinon, elle affiche le message '*n n'est pas premier'*.
2) Donner l'instruction qui permet d'exécuter la procédure *prcd_premie*r sous SQL/Plus pour *n=5*.

Exemple : n=2 est le premier entier premier, n=3 est le deuxième entier premier n=5 est le troisième entier premier…

Solution Examen ordinaire 2017/2018

Exercice 1. Voir cours.

Exercice 2. Voir cours.

Exercice 3.

```
accept n
declare
    nb number;
    ma_matiere matiere%rowtype;
    cursor curs is select * from matiere;
begin
     select count(*) into nb from matiere;
    open curs;
      if &n>=nb then
        loop
          fetch curs into ma_matiere;
          dbms_output.put_line(ma_matiere.nm || ' ' || ma_matiere.nom  || ' '
                                  ||ma_matiere.coefficient || ' ' || ma_matiere.type);
          exit when curs%notfound;
       end loop;
    else
      for i in 1..&n loop
       fetch curs into ma_matiere;
       dbms_output.put_line(ma_matiere.nm || ' ' || ma_matiere.nom_mat);
       exit when curs%notfound;
     end loop;
   end if;
 close curs;
end;
```

Exercice 4.

1)

```
create or replace procedure prcd_premier(n in number)
is
begin
  declare
    p number:=1;
 begin
   for i in 2..n-1 loop
      if mod(n,i)=0  then  p:=0;
     end if;
  end loop;
 if p=1 then
   dbms_output.put_line(n || 'est premier');
 else
  dbms_output.put_line(n || 'n est pas premier');
end if;
end;
end;
```

2)

```
Execute  prcd_premier(5) ;
```

Université Mohamed 1er **Année universitaire : 2017/2018**
Faculté des Sciences **Filière SMI, semestre 6.**
Département d'Informatique

Bases de données avancées
Examen de rattrapage
Durée : 1h 30min

Dans tout ce qui suit, on considère le schéma relationnel suivant :
etudiant (NE, Nom_etud, Adresse, Ville)
matiere (NM, Nom_mat, Coefficient)
evaluation (#NE, #NM, DateEval)

Exercice 1.

1) Donner une définition du curseur (en PL/SQL).
2) Quelle est la différence entre un curseur explicite et un curseur implicite.
3) Compléter le tableau suivant par les ordres SQL qu'on peut associer à chaque type de curseur.

Ordre SQL	Curseur explicite	Curseur implicite
Mise à jour	?	?
Interrogation	?	?

Exercice 2.

1) Donner en langage SQL les requêtes permettant (sous Oracle) de Créer une vue nommée *moy_vue* sur les noms et les moyennes des étudiants.
2) Est-il possible d'insérer des données à travers la vue *moy_vue* ? **justifier.**
3) Donner la syntaxe générale de compilation d'une vue.
4) Qu'elle est l'utilité de l'option *FORCE VIEW* ?

Exercice 3.

Deux nombres entiers premiers *n* et *m* sont dits ***jumeaux*** si leur différence est égale à la valeur 2. (**Exemple** : 3 et 5 ; 5 et 7 ; 11 et 13 ; 29 et 31).

3) Ecrire une fonction nommée *fct_jum* qui prend en paramètre deux entiers *n* et *m* et retourne une valeur booléenne selon qu'ils sont jumeaux ou non.

Rappel : Un entier naturel (différent de 1) est dit premier s'il possède exactement deux diviseurs distincts positifs (1 et lui-même).

Exercice 4.

Ecrire un programme PL/SQL qui permet de saisir un certain numéro d'étudiant et vérifie :

- Si *n_haute* et n_basse sont égaux alors le programme affiche *Liste_matiere.*
- Sinon, le programme affiche le massage *'nombres differents'*.

Avec :

n_haute = le nombre des matières dans lesquelles l'étudiant saisit a obtenu la note la plus haute.

n_basse = le nombre des matières dans lesquelles l'étudiant saisit a obtenu la note la plus basse.

Liste_matiere= les noms des matières dans lesquelles l'étudiant saisit a obtenu la note la plus haute ou bien la note la plus basse.

On suppose que *n_haute* et *n_basse* sont strictement supérieur à 0.

NB. Le programme doit utiliser les curseurs explicites.

Solution examen rattrapage 2017/2018

Exercice 1. Voir cours.

Exercice 2. Voir cours.

Exercice 3.

```
create or replace function fct_jum(n in number, m in number)
return boolean
is
begin
  declare
    p number;
    q number;
   R boolean:=false;
   begin
   ---------- je teste si n est premier
     p:=1;
    for j  in 2..n-1 loop
         if mod(n,j)=0  then  p:=0;
        end if;
   end loop;
    ---------- je teste si m est premier
    q:=1;
    for j  in 2..m-1 loop
         if mod(m,j)=0  then  q:=0;
        end if;
   end loop;

  if (p=1and  q=1) then      -----si n et m sont premiers
       if (n-m=2 or m-n=2 ) then
       R := true;
      End if;
  end if;
return R;
end;
end;
```

Exercice 4.

```
--saisie des valeurs
ACCEPT  num_et

DECLARE
        n_haute  number;
        n_basse  number;
        --except exception;
        CURSOR liste_matiere is
        select a.nom_mat
        from matiere a join evaluation b on a.NM=b.NM
        where
        b.Note=(select max(Note) from evaluation where b.NE= &num_et)
        or b.Note=(select min(Note) from evaluation where b.NE= &num_et)
        and
        NE=&num_et;

BEGIN
--récupérer le nombre des matières dans lesquelles l'étudiant a obtenu la note la plus haute
        select count(NM) as "nombre matiere haute" into n_haute from evaluation
        where
        note=(select max(note) from evaluation where NE=&num_et)
        and NE=&num_et;
--récupérer le nombre des matières dans lesquelles l'étudiant a obtenu la note la plus basse
        select count(NM) as "nombre ayant min" into n_basse
        from evaluation
        where
        note=(select min(note) from evaluation where NE=&num_et)
        and NE=&num_et;
--si les deux nombres sont égaux j'affiche la liste des matières
        IF n_haute = n_basse THEN
          FOR c IN liste_matiere
           LOOP
             DBMS_OUTPUT.PUT_LINE(c.nom_mat) ;
           END LOOP ;
-- sinon j'affiche le message
            ELSE
        DBMS_OUTPUT.PUT_LINE ( 'nombre differents' ) ;
         END IF;

END ;
```

Université Mohamed 1er **Année universitaire : 2018/2019**
Faculté des Sciences **Filière SMI, semestre 6.**
Département d'Informatique

Bases de données avancées
Examen ordinaire
Durée : 1h 30min

Dans tout ce qui soit, on considère le schéma relationnel suivant :
etudiant (NE, Nom, Adresse, Ville)
matiere (NM, Nom, Coefficient, type)
evaluation (#NE, #NM, DateEval, etat)

Exercice1.
Soient la suite des opérations, avec questions, suivantes :

1) Se connecter en utilisant ***SCOTT*** et créer un utilisateur ***USER1***,
Question : Donner la syntaxe de création de USER1.

2) Attribuer les privilèges de connexion à la base de données et de création des tables à l'utilisateur *USER1*.
Question : Donner la syntaxe de ces deux privilèges.

3) Se connecter en utilisant ***USER1*** et créer la table ***etudiant*** avec l'extension :

NE	Nom	Adresse	Ville
1	SMI19	Dep. Info., FSO	Oujda

4) Se connecter avec le compte ***SCOTT*** et exécuter le bloc PL/SQL suivant :

```
DECLARE
   ligne etudiant%ROWTYPE ;
  BEGIN
   SELECT * INTO ligne FROM  USER1.etudiant;
   DBMS_OUTPUT.PUT_LINE ( ligne.NE) ;
   DBMS_OUTPUT.PUT_LINE ( ligne.Nom) ;
  EXCEPTION
   WHEN OTHERS THEN
     DBMS_OUTPUT.PUT_LINE ('message1' ) ;
     DBMS_OUTPUT.PUT_LINE ('message2' ) ;
END;
```

Question : Donner **avec justification** le résultat d'exécution de ce bloc.

5) Donner la commande qui permet de retirer les privilèges, attribués dans la question **2)**, de l'utilisateur *USER1*.

Exercice2.
Ecrire une fonction nommée ***moy_abs*** qui prend en paramètres deux entiers **a** et **b** et retourne la moyenne des valeurs absolues des entiers allant de a à b.
N.B. le programme **ne doit pas utiliser la fonction ABS** (la fonction valeur absolue).

Exemple.
Moy_abs(-2, 3)= (|-2|+|-1|+|0|+|1|+|2|+|3|)/6
= (2+1+0+1+2+3)/6
=1,5

Exercice3.

Écrire un bloc PL/SQL qui permet de Saisir un **certain numéro d'étudiant**,

- Si cet étudiant a été évalué au moins une fois, calculer sa moyenne générale, **sans utilisation des fonctions d'agrégats.**
- Sinon, afficher le message ***'aucune évaluation'.***

N.B. le programme **doit utiliser les curseurs et les exceptions utilisateur**.

Indications :

La moyenne générale est donnée par : $\sum note*coefficient / \sum coefficient$

- Créer un curseur contenant les notes et les coefficients.
- A l'aide d'un parcours du curseur, cumuler le coefficient et la note multipliée par le coefficient.
- Exception : si la table ne contient aucune évaluation de l'étudiant, alors le programme sera interrompu.

Solution examen ordinaire 2018/2019

Dans tout ce qui soit, on considère le schéma relationnel suivant :
etudiant (<u>NE</u>, Nom, Adresse, Ville)
matiere (<u>NM</u>, Nom, Coefficient, type)
evaluation (#NE, #NM, DateEval, note, etat)

Exercice1. Voir cours.

Exercice2.

```
create or replace function moy_abs(a in number, b in number)
return number
is
begin

declare
s number:=0;
n number:=0;
m number;
begin
for i in a..b loop
n:=n+1;
if i>0 then
s:=s+i;
else
s:=s+(-i);
end if;
end loop;
m:=s/n;
return m;
end;
end;
```

Exercice3

```
accept num

declare
s number:=0;
s_coeff number:=0;
m number;
n number;
p_note evaluation.note%type;
p_coeff matiere.coefficient%type;
cursor curseur is
     select e.note, m.coefficient
     from evaluation e, matiere m
     where e.nm=m.nm
      and e.ne=&num;
aucune_eval exception;
begin
select count(NE) into n from evaluation
where NE=&num;
if n<1 then raise aucune_eval;
else
open curseur;
loop
fetch curseur into p_note, p_coeff;

exit when curseur%notfound;
s:=s+p_note*p_coeff;
s_coeff:=s_coeff+p_coeff;
dbms_output.put_line(p_note || p_coeff);

end loop;
m:=s/s_coeff;
dbms_output.put_line('somme est ' || s);
dbms_output.put_line('somme coeff est ' || s_coeff);
dbms_output.put_line('moyenne est ' || m);
close curseur;
end if;
exception
when aucune_eval then
dbms_output.put_line('aucune evaluation');

end;
```

Université Mohamed 1er **Année universitaire : 2018/2019**
Faculté des Sciences **Filière SMI, semestre 6.**
Département d'Informatique

Bases de données avancées
Examen rattrapage
Durée : 1h 30min

Dans tout ce qui soit, on considère le schéma relationnel suivant :
etudiant (NE, Nom, Adresse, Ville, age)
matiere (NM, Nom_mat, Coefficient, type)
evaluation (#NE, #NM, DateEval, note)

Exercice1.

On considère **la table Etudiant** avec l'extension

NE	Nom	Adresse	Ville	Age
1	Ahmed	Rue 20, FSO	Oujda	19
2	Rachid	Dep. Info.	Oujda	20

et le programme PL/SQL suivant :

```
declare
 n,m number;
 msg varchar(10);
begin
update etudiant
set ville='any'
where age>20;
m:=1;
n:=sql%rowcount;
IF  n==0  THEN
   msg:='zero' ;
   ELSEIF  n==1 THEN
   msg:='un' ;
   ELSEIF  n==2 THEN
   msg:='deux' ;
   ELSE
   msg:='autre' ;
end if;
dbms_output.put_line(msg);
dbms_output.put_line(m);
end;
```

1) Corriger les erreurs de ce programme.
2) Donner le résultat d'exécution de ce programme.
3) Réécrire le même programme en utilisant l'instruction ***CASE***.

Exercice2

Un entier naturel (différent de 1) est dit premier s'il possède exactement deux diviseurs distincts positifs, 1 et lui-même.

1) Ecrire une fonction nommée *nombre_premier* qui prend en paramètre un entier *n* et retourne le nombre des entiers premiers qui sont inférieurs ou égaux à *n*.
2) Donner l'instruction d'exécution de la fonction *nombre_premier* pour *n=10*.

Exercice3

Ecrire un programme qui permet de saisir un entier *n* et un numéro de matière :

1) Si le nombre des évaluations dans cette matière est inférieur à *n*, le programme affiche le message *'matière non évaluée'*.
2) Sinon, Récupérer le nom et la note des *n* derniers étudiants selon leurs notes dans cette matière et les afficher avec le numéro de la matière.

N.B : Le programme doit utiliser les curseurs explicites et les exceptions utilisateurs.

Solution examen rattrapage 2018/2019

Exercice1. Voir cours.

Exercice2.

1)

```
create or replace function nombre_premier(n in number)
return number
is
begin
 declare
     p number;
     s number:=0;---nombre des nombres premiers
     m  number;
begin
 m:=2;
 loop---repeter de 2 jusqu'a la valeur de m
     p:=1;----boucle qui teste si m est premier ou non
      for i in 2..m-1 loop
          if mod(m,i)=0  then  p:=0;
         end if;
    end loop;
   if p=1 then
      s:=s+1;
   end if;
   m:=m+1;
  exit when m>n;
end loop;
return s;
end;
end;
```

2) *select nombre_premier(10) from dual;*

Exercice3.

```
ACCEPT  n
ACCEPT  num_mat
DECLARE
m number:=&n;
c_nm evaluation.NM%type;
c_nom etudiant.nom%type;
c_note evaluation.note%type;
excep exception;
nb number;--nombre des evaluations
     CURSOR  dern_curseur  IS
     SELECT v.NM, e.nom, v.note
     FROM etudiant e, evaluation v
     WHERE
     v.NE=e.NE
     and v.NM=&num_mat
     ORDER BY v.note Asc;
BEGIN
    select count(*) into nb  ---le nombre des evaluations
    from evaluation
   where NM=&num_mat;
 if nb<m then ---tester si le nombre des evaluations est inferieur à n
       raise excep;
 else
  OPEN dern_curseur;
    FOR  i IN 1..&n  LOOP
       FETCH  dern_curseur INTO  c_nm, c_nom,c_note;
      dbms_output.put_line(c_nm || ' ' || c_nom || '' || c_note);
   END LOOP ;
  CLOSE dern_curseur ;
end if;
EXCEPTION
   when excep then
   dbms_output.put_line('matiere non evaluée');
END ;
                                          /
```

Université Mohamed 1er **Année universitaire : 2020/2021**
Faculté des Sciences **Filière SMI, semestre 6.**
Département d'Informatique

Bases de données avancées
Examen ordinaire
Durée : 1h30min

On considère le schéma relationnel suivant :
etudiant (NE, Nom, Adresse, Ville)
matiere (NM, Nom, Coefficient)
evaluation (#NE, #NM, DateEval, note)

I)
Sous *SQLPLUS*, vous êtes connectés avec le compte *user1*.

1) Donnez la syntaxe de création d'un utilisateur avec les options suivantes : nom utilisateur : *user2*, mot de passe : *smi21*.
2) Quelles sont les privilèges associés à l'utilisateur *user2*.
3) Quelles sont les *tablespaces* affectés à *user2*.
4) Accordez les privilèges connexion à la base de données et création de table sur tous les schémas à l'utilisateur user2 **(donnez la syntaxe).**

Maintenant, vous êtes connectés avec l'utilisateur *user2*.

5) Est-t-il possible de créer une table dans le schéma de l'utilisateur *user3* (un autre utilisateur) ? **justifier.**

II)

1) Donner la syntaxe de création d'une vue, nommée *vue_coefficient*, sur tous les attributs des matières qui ont un coefficient supérieur ou égal à la valeur 4.
2) A travers la vue *vue_coefficient*, est-t-il possible de :
   - **a.** Insérer la ligne *('Mat1', 'informatique',2).*
   - **b.** Insérer la ligne *('Mat2', 'projet',6).*
   - **c.** Modifier la ligne *('Mat1', 'informatique',2).*
   - **d.** Modifier la ligne *('Mat2', 'projet',6).*

3)
   - **a.** Donner la commande SQL permettant d'ajouter une colonne nommée *nobre_heure* de type *NUMBER* à la table matière.
   - **b.** Quelle est l'effet de la commande de la question **a.** sur la vue *vue_coefficient* ?
   - **c.** Comment compiler la vue *vue_coefficient* ?
   - **d.** Après compilation quelle est la nouvelle structure de la vue *vue_coefficient* ?

III) Soit l'extension de la table *matiere* :

NM	**Nom**	**Coefficient**
Mat1	Mathématiques	4
Mat2	Physique	3
Mat3	Langue	2

On considère la saisie d'un entier c et le programme PL/SQL suivants:

```
ACCEPT   c

DECLARE
n , m  number ;
BEGIN
   m :=0;
  SELECT COUNT(coefficient) INTO  n  FROM  matiere;
  WHERE coefficeint > c
  IF  n==1  THEN
  DBMS_OUTPUT.PUT_LINE('un') ;
  ELSEIF  n==2  THEN
  DBMS_OUTPUT.PUT_LINE('deux') ;
  m:=n+1;
  ELSEIF  n==3 THEN
  DBMS_OUTPUT.PUT_LINE(' trois') ;
  ELSE  THEN
  DBMS_OUTPUT.PUT_LINE('autre') ;
  m:=n+2 ;
  END IF ;
 DBMS_OUTPUT.PUT_LINE(m) ;
END ;
```

1) Corriger les erreurs de ce programme.
2) Donner le résultat d'exécution du programme pour les valeurs c=1, c= 3 et c= 4.

Solution Examen ordinaire 2020/2021

On considère le schéma relationnel suivant :
etudiant (NE, Nom, Adresse, Ville)
matiere (NM, Nom, Coefficient)
evaluation (#NE, #NM, DateEval, note)

I)

Voir cours.

II)

Voir cours

III) Voir cours

Université Mohamed 1er **Année universitaire : 2020/2021**
Faculté des Sciences **Filière SMI, semestre 6.**
Département d'Informatique

Bases de données avancées
Examen rattrapage
Durée : 1h30min

On considère le schéma relationnel suivant :
etudiant (NE, Nom, Adresse, Ville)
matiere (NM, Nom, Coefficient)
evaluation (#NE, #NM, DateEval, note)

I)

Sous *SQLPLUS*, vous êtes connectés avec le compte *user1*.

1) Donnez la syntaxe de création d'un utilisateur avec les options suivantes : nom utilisateur : *user2* et mot de passe : *smi21*.
2) Donner la syntaxe de modification du mot de passe, le nouveau mot de passe est *smi21_ratt*.
3) Accordez les privilèges connexion à la base de données et création de table à l'utilisateur user2 en lui donnant la possibilité d'accorder ces privilèges à un autre utilisateur **(donnez la syntaxe).**

Maintenant, vous êtes connectés avec l'utilisateur *user2*.

4) Est-t-il possible de créer une table ? **justifier.**
5) Soit maintenant la table *user2.etudiant* créée par l'utilisateur *user2*. Quelles sont les commandes SQL que l'utilisateur *user1* peut exécuter sur cette table ?

II)

1) Donner la syntaxe de création d'une vue, nommée *vue_coefficient*, sur les noms et les Coefficients des matières qui ont un coefficient supérieur ou égal à la valeur 4.
2) A travers la vue vue_*coefficient*, est-t-il possible de :
   - **a.** Insérer la ligne *('informatique',2).*
   - **b.** Insérer la ligne *('projet',6).*
3) Que signifie l'option *FORCE VIEW* dans la syntaxe de création d'une vue.
4) Donner la commande permettant d'attribuer le privilège de sélection sur la vue *vue_coefficient.*

III)

Ecrire un programme PL/SQL qui permet de

1) Saisir un certain numéro de matière nommé *n*.
2) Si cette matière n'existe pas, le programme affiche le message *"n'existe pas"* et l'insère (avec : NM= *n, Nom_mat= 'info', Coefficient=3).*
3) Si cette matière existe alors :
   a. Si son coefficient est supérieur ou égale à 3, le programme affiche le message : *"existe avec coefficient supérieur à 3".*
   b. Sinon le programme affiche le message : *"existe avec coefficient inférieur à 3".*

Solution examen rattrapage 2020 /2021

On considère le schéma relationnel suivant :
etudiant (NE, Nom, Adresse, Ville)
matiere (NM, Nom, Coefficient)
evaluation (#NE, #NM, DateEval, note)

I) Voir cours.

II) Voir cours

III)

Ecrire un programme PL/SQL qui permet de

1) Saisir un certain numéro de matière nommé *n*.
2) Si cette matière n'existe pas, le programme affiche le message *"n'existe pas"* et l'insère (avec : NM= *n, Nom_mat= 'info', Coefficient=3).*
3) Si cette matière existe alors :
   a. Si son coefficient est supérieur ou égale à 3, le programme affiche le message : *"existe avec coefficient supérieur à 3".*
   b. Sinon le programme affiche le message : *"existe avec coefficient inférieur à 3".*

```
accept numero
DECLARE
        nombre NUMBER;
        nombre_mat NUMBER ;
        --nom_et etudiant.nom%TYPE;
        --etud etudiant%rowtype;
BEGIN
   --je cherche si l'etudiant existe
  SELECT count(*) into nombre FROM etudiant
  WHERE NE=&numero;
     if nombre>0 then
        --DBMS_OUTPUT.PUT_LINE('existe');
        --je calcule le nombre des matieres avec note <10
         select count(b.NM) into nombre_mat
         from  evaluation
         where Note<10
          and NE=&numero;
            if n=0 then
                    --DBMS_OUTPUT.PUT_LINE('toutes les notes sont >= 10');
```

```
                Update etudiant
               Set resultat='ok'
               Where NE=&numero ;
               Else
                 --DBMS_OUTPUT.PUT_LINE( 'on a des notes <');
               Update etudiant
               Set resultat= 'non ok'
               Where NE=&numero ;
          end if ;
 else
             DBMS_OUTPUT.PUT_LINE('n existe pas');
end if;
end;
```

Exercice 2.

On considère le schéma relationnel suivant :
etudiant (NE, Nom, Adresse, Ville)
matiere (NM, Nom, Coefficient, état)
evaluation (NE, NM, DateEval, Note)

Ecrire un programme PL/SQL qui lit un certain nom de matière et une date, puis vérifie

- si aucune évaluation n'a eu lieu dans cette matière après cette date alors la matière sera supprimée de la table matière,
- sinon la valeur de l'état de la matière sera mise à « ok ».

Exercice 2.

```
accept nom
accept date_ev

DECLARE
   n number:=0;
   excep1 EXCEPTION;
   excep2 EXCEPTION;
   CURSOR curs_ev is
      select dateEval
       from evaluation e, matiere m
       where e.NM=m.NM
       and m.Nom='&nom'
       and e.dateEval > &date_ev;

BEGIN
--ou bien utiliser curs_ev%rowcount c'est à dire
     -- open curs_ev;
      --dbms_output.put_line(curs_ev%rowcount);
      --close curs_ev;
  for c in curs_ev loop
   exit when curs_ev%notfound;
```

```
    n:=n+1;
   end loop;
  dbms_output.put_line('le nombre est' || n);
if n=0 then raise excep1;
else raise excep2;
end if;
EXCEPTION
        when excep1 then
   delete from  matiere
   where
  nom='&nom';
       when excep2 then
   update matiere
   set etat='ok'
  where
  nom='&nom';
END;
```