Considérons les tables suivantes:

Employees

Employee_id
First_Name
Last_Name
Email
Phone_Number
Hire_Date
Job_ID
Salary
Commission_PCT
Manager_ID
Department_ID

Departments

Department_ID
Department_name

1. Ecrire une fonction PL/SQL permettant de retourner le salaire moyen de tous les employés d'un département donné qui ne sont pas de managers.

   Corrigé

```
create or replace function Exemple1 (N employees.department_id%type)
 return integer
 IS
 moyen integer;
 begin
  select avg(salary) into moyen
  from employees
  where department_id = N
  and employee_id not in (select manager_id
                          from employees
                           where manager_id is not null);
 return moyen;
 end;
```

2. Ecrire un trigger permettant de lever une exception dans le cas d'une diminution de salaire des employés.

   Corrigé

```
create or replace trigger Exemple2
before update of salary
on employees
for each row
 begin
   if (:new.salary < :old.salary) then
     raise_application_error (-20200, 'interdit de diminuer le salaire');
 end if;
end;
```

3. Pour archiver l'historique des opérations de mise à jour de la table Employees, on a crée une table History (employee_id, operation_date, operation_type). Ecrire un trigger permettant, après l'exécution de chaque opération sur Employees, d'ajouter un tuple concernant cette opération à la table History .

Corrigé

```
create or replace trigger Exemple3
after update or insert or delete
on employees
for each row
declare
type_op varchar2(6);
id employees.employee_id%type;
begin
  if (inserting) then
    type_op := 'insert'; id := :new.employee_id;
  elsif (updating) then
    type_op := 'update';  id := :new.employee_id;
  else
    type_op := 'delete';  id := :old.employee_id;
end if;
insert into history values (id, sysdate, type_op);
end;
```

4. Pour chaque département, afficher le nom du département, le nom (first_name et last_name) et le salaire de l'employé ayant le plus grand salaire de ce département. Le résultat doit être déterminé de plusieurs manières :
   a) Créer une view pour trouver pour chaque département le plus grand salaire puis utiliser cette view dans une requête.
   b) Utiliser une seule requête SQL
   c) Créer une procédure en PL/SQL

Corrigé

a) Utilisation d'une view

```
create view salairemax as
  select department_id, max(salary) maxsalary
  from employees
  group by department_id;

  select d.department_name, (e.first_name || ' ' || e.last_name) as name, e.salary
  from employees e,  departments d, salairemax s
  where e.department_id = d.department_id
  and e.department_id = s.department_id
  and e.salary = s.maxsalary;
```

b) Utiliser une seule requête SQL

Première méthode : utilisation d'une requête imbriquée dans la clause FROM:

```
select d.department_name, (e.first_name || ' ' || e.last_name) as name, e.salary
from employees e,  departments d, (select department_id, max(salary) maxsalary
                                   from employees
                                   group by department_id) s
where e.department_id = d.department_id
and e.department_id = s.department_id
and e.salary = s.maxsalary;
```

Deuxième méthode : utilisation d'une requête imbriquée dans la clause WHERE:

```
select d.department_name, (e.first_name || ' ' || e.last_name) as name, e.salary
from employees e,  departments d
where e.department_id = d.department_id
and (e.department_id, e.salary) in ( select department_id, max(salary) maxsalary
                                     from employees
                                     group by department_id);
```

c) Créer une procédure en PL/SQL

Remarque : On peut déclarer un curseur comme l'une des requêtes de la question (b) et on utilise une boucle pour parcourir tout simplement ce curseur. Mais PL/SQL nous offre la possibilité de simplifier le traitement ; par exemple on peut utiliser deux curseurs, le premier pour trouver pour chaque département le plus grand salaire et le deuxième est un curseur paramétré qui prend comme paramètres l'id d'un département et le plus grand salaire de ce département et cherche les employés de ce département possédant ce salaire.

```
create or replace procedure Exemple4 IS

 cursor cur1 is
   select d.department_id, d.department_name, max(salary) maxsalary
   from employees e , departments d
   where e.department_id = d.department_id
   group by d.department_id, d.department_name;

 cursor cur2(dep employees.department_id%type, sal employees.salary%type) is
   select employee_id , (first_name || ' ' || last_name) as name, salary
   from employees
   where department_id = dep
   and salary = sal;
```

```
begin
dbms_output.put_line ('Department_name     ' || '       Name      ' ||   ' ' Salary');
for x in cur1 loop
  for y in cur2(x.department_id, x.maxsalary) loop
     dbms_output.put_line (x.department_name || '     ' || y.name|| '     '  ||y.salary);
  end loop;
end loop;
end;
```

5. Ecrire une requête SQL permettant d'afficher pour chaque département, l'id du département et les trois premiers salaires les plus élevés. Le résultat doit être trié suivant l'id de département et il doit avoir la forme suivante :

No. Département	**Premier salaire**	**Deuxième salaire**	**Troisième salaire**
30	11000	3100	2900
……	…….	……	……

***Corrigé* :**

```
select e1.department_id "No. Département", max(e1.salary) "Premier Salaire",
       max(e2.salary) "Deuxième salaire", max (e3.salary) "Troisième Salaire"
from employees e1, employees e2, employees e3
where e1.department_id = e2.department_id
and e2.department_id = e3.department_id
and e3.salary < e2.salary
and e2.salary < e1.salary
group by e1.department_id
order by e1.department_id ;
```

6. La requête de l'exercice 5 ne permet pas d'afficher les résultats pour tous les départements ; elle affiche les résultats seulement pour les départements ayant au moins 3 employés avec trois valeurs différentes de salaire. Alors, écrire une procédure PL/SQL permettant d'afficher pour tous les départements les trois premiers salaires les plus élevés, s'il n'existe pas qu'un seul salaire alors afficher pour le deuxième et le troisième , la même valeur que le premier ; et s'il n'existe pas que deux salaires différents alors afficher pour le troisième , la même valeur que le deuxième.

Corrigé

```
create or replace procedure Exemple3 IS
 cursor cur is
   select department_id,  max(salary) max1
   from employees
```

```
  group by department_id
  order by department_id;

 max2 employees.salary%type;
 max3 employees.salary%type;

begin
 dbms_output.put_line ('No. Départ      Premier Salaire      Deuxième salaire
Troisième Salaire');


 for c in cur loop
  select nvl(max(salary), c.max1) into max2
  from employees
  where department_id = c.department_id
  and salary < c.max1;
  select nvl(max(salary), max2) into max3
  from employees
  where department_id = c.department_id
  and salary < max2;
  dbms_output.put_line (c.department_id || '          ' || c.max1 ||'           ' || max2
                                  || '          ' || max3);
 end loop;
end;
```

Remarque: s'il n'existe pas de données la requête SELECT … INTO lève une exception, mais dans le cas d'une fonction (MAX, MIN, AVG, COUNT, SUM) elle ne lève pas une exception, elle retourne NULL.