SERIE 6

```
create table jobs as select * from hr.jobs;
create table employees as select * from hr.employees;
create table departments as select * from hr.departments;

CREATE VIEW sal_par_dept as
select d.department_id, nvl(sum(salary),0) somme from employees e right join departments d on
d.department_id=e.department_id
group by d.department_id;
```

```
CREATE OR REPLACE PACKAGE EMPDEPT is
        PROCEDURE ajout_job (p_id jobs.job_id%type, p_intitule jobs.job_title%type);
        PROCEDURE modif_job (p_id jobs.job_id%type, p_nouveauTitre jobs.job_title%type);
        PROCEDURE listemp;
        PROCEDURE gagneplus(p_nom_emp employees.last_name%type, p_prenom_emp
        employees.first_name%type);
        PROCEDURE jobgagneplus(p_id_job employees.employee_id%type, p_id_sal
        employees.employee_id%type);
        PROCEDURE nsalaires(p_n number);
        PROCEDURE DEPT_SANS_EMP;
        PROCEDURE hierarchie (p_num_emp employees.employee_id%type);
        PROCEDURE dept_somme_sal(p_somme_sal employees.salary%type);
        PROCEDURE rapport_employes;
        FUNCTION  check_sal(p_num_emp employees.employee_id%type)
        RETURN Boolean;
END EMPDEPT;
```

```
CREATE OR REPLACE PACKAGE BODY EMPDEPT is

PROCEDURE ajout_job (p_id jobs.job_id%type, p_intitule jobs.job_title%type)
IS
BEGIN
insert into jobs (job_id,job_title) values (p_id, p_intitule);
commit;
EXCEPTION
WHEN others THEN
raise_application_error(-20000,'Erreur ' || SQLERRM || 'de numéro :' || SQLCODE);
END ajout_job;
--
PROCEDURE modif_job (p_id jobs.job_id%type, p_nouveauTitre jobs.job_title%type)
IS
e_aucune_maj exception;
BEGIN
UPDATE jobs SET job_title=p_nouveauTitre WHERE upper(job_id)=upper(p_id);
IF SQL%ROWCOUNT = 0 THEN
RAISE e_aucune_maj;
END IF;
commit; -- à placer après la conditionnelle
EXCEPTION
WHEN e_aucune_maj THEN
raise_application_error(-20000,'Aucune mise à jour n''a eu lieu');
WHEN others THEN
raise_application_error(-20001,'Erreur ' || SQLERRM || 'de numéro :' || SQLCODE);
END modif_job;
--
```

```
PROCEDURE listemp
IS
BEGIN
DBMS_OUTPUT.PUT_LINE(rpad('Nom employé',30)  || 'Nom manager'|| chr(10) || rpad('-',50,'-'));
FOR nom_enrg IN (select e1.last_name Nom_emp, e1.first_name prenom_emp, e2.first_name
prenom_mgr, e2.last_name Nom_mgr -- alias obligatoire
from employees e1 LEFT JOIN employees e2 ON  e1.manager_id=e2.employee_id)
LOOP
DBMS_OUTPUT.PUT_LINE( rpad(nom_enrg.Nom_emp||' '||nom_enrg.prenom_emp,30)  ||
nom_enrg.prenom_mgr || ' ' ||nom_enrg.Nom_mgr);
END LOOP;
EXCEPTION
WHEN others THEN
raise_application_error(-20000,'Erreur ' || SQLERRM || 'de numéro :' || SQLCODE);
END;
--
PROCEDURE gagneplus(p_nom_emp employees.last_name%type, p_prenom_emp
employees.first_name%type)
IS
v_sal employees.salary%type;
BEGIN
-- test existence de l'employé
select  salary*(1+nvl(COMMISSION_PCT,0)) into v_sal from employees where upper(last_name) =
upper(p_nom_emp) and upper(first_name)=upper(p_prenom_emp);
DBMS_OUTPUT.PUT_LINE('Prénom et Nom Employé' || chr(10)|| lpad('-',20,'-'));
FOR enrg IN (select first_name, last_name
from employees
where  salary*(1+nvl(COMMISSION_PCT,0)) > v_sal)
LOOP
DBMS_OUTPUT.PUT_LINE( enrg.first_name || ' ' ||enrg.last_name);
END LOOP;
EXCEPTION
WHEN NO_DATA_FOUND THEN
raise_application_error(-20000,'Erreur l''employé n''existe pas');
WHEN others THEN
raise_application_error(-20001,'Erreur ' || SQLERRM || 'de numéro :' || SQLCODE);
END;
--
PROCEDURE jobgagneplus(p_id_job employees.employee_id%type, p_id_sal
employees.employee_id%type)
IS
v_sal employees.salary%type;
v_job_id employees.job_id%type;
BEGIN
-- test existence de l'employé
select  salary*(1+nvl(COMMISSION_PCT,0)) into v_sal  from employees where employee_id = p_id_sal;
select job_id into v_job_id from employees where employee_id=p_id_job;
DBMS_OUTPUT.PUT_LINE('Nom Employé gagnant plus que '|| p_id_sal || ' ou ayant le même travail que
'||p_id_job  || chr(10)||  lpad('-',50,'-'));
FOR enrg IN (select first_name, last_name
from employees
where  salary*(1+nvl(COMMISSION_PCT,0)) > v_sal OR job_id= v_job_id AND employee_id != p_id_job)
LOOP
DBMS_OUTPUT.PUT_LINE( enrg.first_name || ' ' || enrg.last_name);
END LOOP;
EXCEPTION
WHEN NO_DATA_FOUND THEN
raise_application_error(-20000,'Erreur l''employé n''existe pas');
WHEN others THEN
raise_application_error(-20001,'Erreur ' || SQLERRM || 'de numéro :' || SQLCODE);
END;
--
```

```
PROCEDURE nsalaires(p_n number) IS
BEGIN
DBMS_OUTPUT.PUT_LINE('Employés percevant les plus gros salaires'|| chr(10) || lpad('-',40,'-'));
FOR enrg IN (SELECT first_name prenom, last_name nom from (SELECT first_name, last_name from
employees order by salary desc) where rownum <= p_n) LOOP
DBMS_OUTPUT.PUT_LINE(enrg.prenom || ' ' || enrg.nom);
END LOOP;
EXCEPTION
WHEN others THEN
raise_application_error(-20000,'Erreur ' || SQLERRM || 'de numéro :' || SQLCODE);
END;
--
PROCEDURE DEPT_SANS_EMP
IS
BEGIN
DBMS_OUTPUT.PUT_LINE('Départements sans employé'|| chr(10) || lpad('-',26,'-'));
FOR enrg IN (select department_name from departments where department_id not in (select
distinct(department_id) from employees where department_id is not null) order by 1) LOOP
DBMS_OUTPUT.PUT_LINE( enrg.department_name);
END LOOP;
EXCEPTION
WHEN others THEN
raise_application_error(-20000,'Erreur ' || SQLERRM || 'de numéro :' || SQLCODE);
END;
--
PROCEDURE hierarchie (p_num_emp employees.employee_id%type)
IS
BEGIN
DBMS_OUTPUT.PUT_LINE('Hierarchie');
FOR enrg IN (select first_name, last_name, LEVEL from employees
connect by prior employee_id=manager_id
start with manager_id=p_num_emp) LOOP
DBMS_OUTPUT.PUT_LINE( rpad(' ', enrg.level+1,'*') ||enrg.LEVEL ||'-'|| enrg.first_name||' '
||enrg.last_name);
END LOOP;
EXCEPTION
WHEN others THEN
raise_application_error(-20000,'Erreur ' || SQLERRM || 'de numéro :' || SQLCODE);
END;
--
PROCEDURE dept_somme_sal(p_somme_sal employees.salary%type)
IS
BEGIN
DBMS_OUTPUT.PUT_LINE('Département où somme des salaires > ' || p_somme_sal || chr(10) || lpad('-
',42,'-') );
FOR enrg IN (select department_id, somme  from sal_par_dept
where somme > p_somme_sal) LOOP
DBMS_OUTPUT.PUT_LINE( rpad(enrg.department_id,10,' ') || CHR(9) || enrg.somme);
END LOOP;
EXCEPTION
WHEN others THEN
raise_application_error(-20000,'Erreur ' || SQLERRM || 'de numéro :' || SQLCODE);
END;
--
PROCEDURE rapport_employes
IS
BEGIN
DBMS_OUTPUT.PUT_LINE('Employés de salaire supérieur au salaire moyen de leur département' ||
chr(10) || lpad('-',50,'-'));
FOR enrg IN (select first_name, last_name FROM EMPLOYEES e where SALARY > (select avg(salary)
from employees where department_id=e.department_id)) LOOP
DBMS_OUTPUT.PUT_LINE( enrg.first_name || ' ' || enrg.last_name);
```

```
END LOOP;
EXCEPTION
WHEN others THEN
raise_application_error(-20000,'Erreur ' || SQLERRM || 'de numéro :' || SQLCODE);
END;
--
FUNCTION  check_sal(p_num_emp employees.employee_id%type)
RETURN boolean IS
v_dept_id employees.department_id%type;
v_sal employees.salary%type;
v_avg_sal employees.salary%type;
BEGIN
SELECT salary, department_id into v_sal, v_dept_id from employees where employee_id=p_num_emp;
SELECT avg(salary) into v_avg_sal from employees where department_id=v_dept_id;
IF v_sal > v_avg_sal THEN
RETURN TRUE;
ELSE
RETURN FALSE;
END IF;
EXCEPTION
WHEN NO_DATA_FOUND THEN
RETURN NULL;
END;
END EMPDEPT;
```

Tests:

```
set verify off
BEGIN
CASE EMPDEPT.check_sal(&p_num)
WHEN TRUE THEN
dbms_output.put_line( 'Salaire > Moyenne des salaires');
WHEN FALSE THEN
dbms_output.put_line( 'Salaire < Moyenne des salaires');
ELSE
dbms_output.put_line('La fonction a renvoyé NULL à cause d''une exception');
END CASE;
END;
```

p_num = 100
p_num = 200
p_num = 1

```
Salaire > Moyenne des salaires

Salaire < Moyenne des salaires

La fonction a renvoyé NULL à cause d'une exception
```