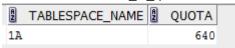
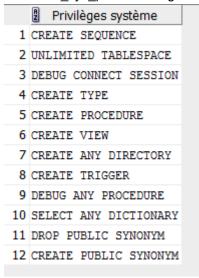
CORRECTIONS BD

SERIE 2

1. select tablespace_name, max_blocks Quota from dba_ts_quotas where username='P1303195';



2. select privilege "Privilèges système" from dba_sys_privs where grantee = 'P1303195';



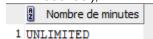
3. select * from dba tab privs where grantee = 'PUBLIC' and OWNER ='SCOTT';



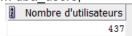
4. select * from dba_role_privs where grantee = 'P1303195';



5. select limit "Nombre de minutes" from dba_profiles where resource_name = 'IDLE_TIME' and profile = (select profile from dba_users where username='P1303195');



6. select count(*) "Nombre d'utilisateurs" from dba_users;



```
SERIE 3
Exercice 1
set serveroutput on
DROP TABLE EMP;
DROP TABLE DEPT;
CREATE TABLE emp AS SELECT * FROM scott.emp;
CREATE TABLE dept AS SELECT * FROM SCOTT.dept;
DECLARE
     v_nb_emp number(3);
BEGIN
     select count(empno) into v nb emp from emp;
     dbms output.put line('Nombre d''employés' | v nb emp);
END;
ACCEPT num emp prompt 'Entrez un numéro d'employé :'
-- suppression commentaires à l'exécution
SET verify OFF
DECLARE
 v moy emp.sal%type;
 v_emp_sal emp.sal%type;
 v emp num emp.empno%type;
BEGIN
      /*génération de VALUE_ERROR (expression). Par exemple saisi de 'a' avec les quotes
       v emp num := to number(&num emp);
       génération de INVALID_NUMBER (ordre SQL). Par exemple saisi de 'a' avec les quotes */
       -- Question a) et b)
      SELECT sal INTO v emp sal FROM emp WHERE empno=&num emp;
       DBMS OUTPUT.put line('Ancien salaire: '||v emp sal);
      SELECT AVG(sal) INTO v mov
      FROM emp
      WHERE job = (SELECT job FROM emp WHERE empno= &num emp);
      DBMS_OUTPUT.put_line('Movenne des salaires' || v_moy);
      IF v_emp_sal < v mov THEN
             v_emp_sal := v_moy;
      ELSE
             v_emp_sal := v_emp_sal*1.1;
      END IF:
      DBMS_OUTPUT.put_line('Nouveau salaire : ' ||v_emp_sal);
      UPDATE emp SET sal = v_emp_sal WHERE empno=&num_emp;
      COMMIT;
EXCEPTION
/*WHEN VALUE ERROR THEN
      RAISE_APPLICATION_ERROR(-20100, 'ERREUR DE TYPE VALUE_ERROR: '|| SQLERRM || ' CODE
      ERREUR' || SQLCODE);*/
WHEN INVALID NUMBER THEN
      RAISE_APPLICATION_ERROR(-20101, 'ERREUR DE TYPE INVALID_NUMBER : '|| SQLERRM || 'CODE
      ERREUR' || SQLCODE);
WHEN NO_DATA_FOUND THEN
      DBMS OUTPUT.put line('EMPLOYE INCONNU' | SQLERRM);
       RAISE:
WHEN OTHERS THEN
      RAISE APPLICATION ERROR(-20111, 'Erreur: '|| SQLERRM || 'Code erreur' || SQLCODE);
END;
Test INVALID_NUMBER Saisie: 'a'
```

Rapport d'erreur :

ORA-20101: ERREUR DE TYPE INVALID NUMBER : ORA-01722: Nombre non valide CODE ERREUR-1722

ORA-06512: à ligne 31

Test VALUE_ERROR Saisie: 'a'

Rapport d'erreur :

ORA-20100: ERREUR DE TYPE : ORA-06502: PL/SQL : erreur numérique ou erreur sur une valeur: erreur de

conversion des caractères en chiffres CODE ERREUR-6502

ORA-06512: à ligne 29

Test NO DATA FOUND Saisie: 5

EMPLOYE INCONNU ORA-01403: aucune donnée trouvée

Test Saisie: 7369

bloc anonyme terminé Ancien salaire : 800

Moyenne des salaires 1016,67 Nouveau salaire : 1016,67

Saisie: 7499

bloc anonyme terminé Ancien salaire : 1600 Moyenne des salaires 1400 Nouveau salaire : 1760

--1-с

UPDATE emp x

set sal = (select decode(least(x.sal, avg(sal)),

avg(sal),x.sal*1.1, avg(sal))

from emp where job=x.job)

where empno=#

commit;

ou

UPDATE emp e SET sal = (CASE WHEN sal >= (SELECT AVG(sal) FROM emp WHERE job=e.job) THEN sal*1.1 ELSE (SELECT AVG(sal) FROM emp WHERE job=e.job) END) WHERE empno=#

€ EMPNO	♦ ENAME	∮ ЈОВ	∯ MGR	♦ HIREDATE	♦ SAL	⊕ сомм	
7369	SMITH	CLERK	7902	17/12/80	800	(null)	20
7499	ALLEN	SALESMAN	7698	20/02/81	1600	300	30
7521	WARD	SALESMAN	7698	22/02/81	1250	500	30
7566	JONES	MANAGER	7839	02/04/81	2975	(null)	20
7654	MARTIN	SALESMAN	7698	28/09/81	1250	1400	30
7698	BLAKE	MANAGER	7839	01/05/81	2850	(null)	30
7782	CLARK	MANAGER	7839	09/06/81	2450	(null)	10
7839	KING	PRESIDENT	(null)	17/11/81	5000	(null)	10
7844	TURNER	SALESMAN	7698	08/09/81	1500	0	30
7900	JAMES	CLERK	7698	03/12/81	950	(null)	30
7902	FORD	ANALYST	7566	03/12/81	3000	(null)	20
7934	MILLER	CLERK	7782	23/01/82	1300	(null)	10

```
3.
set serveroutput on
set verify off
Accept nb prompt 'Entrez un nombre :'
DECLARE
   e negatif exception;
   e to big exception;
   v fact number :=1;
   v nb saisi number(3);
BEGIN
   v nb saisi := to number(&nb);
   IF v nb saisi < 0 THEN
       raise e_negatif;
   ELSIF v_nb_saisi > 83 THEN
       raise e_to_big;
   END IF;
   FOR i in 1.. v_nb_saisi LOOP
       v_fact := v_fact*i;
   END LOOP;
   DBMS OUTPUT.PUT LINE('Valeur de la factorielle : '|| v fact);
EXCEPTION
WHEN VALUE ERROR THEN
       RAISE_APPLICATION_ERROR(-20110, 'Erreur de type: '|| SQLERRM || ' Code erreur ' || SQLCODE);
WHEN e negatif THEN
       RAISE APPLICATION ERROR(-20111, 'Erreur - le nombre saisi ne doit pas être négatif '|| SQLERRM || 'Code
       erreur ' || SQLCODE);
WHEN e_to_big THEN
       RAISE APPLICATION ERROR(-20112, 'Erreur - le nombre saisi doit être inférieur à 84 '|| SQLERRM || ' Code
erreur ' || SQLCODE);
WHEN OTHERS THEN
       RAISE APPLICATION ERROR(-20113, 'Erreur: '|| SQLERRM || 'Code erreur' || SQLCODE);
END;
Tests:
Valeur saisie: -1
Rapport d'erreur -
ORA-20111: Erreur - le nombre saisi ne doit pas être négatif User-Defined Exception Code erreur 1
ORA-06512: à ligne 21
Valeur saisie: 84
Rapport d'erreur -
ORA-20112: Erreur - le nombre saisi doit être inférieur à 84 User-Defined Exception Code erreur 1
ORA-06512: à ligne 23
Valeur saisie: 5
bloc anonyme terminé
```

Valeur de la factorielle : 120

4.

```
CREATE TABLE amorti
(Duree number(2) CONSTRAINT amorti_pk primary key, Somme_due number(10,2), Remb_An number(8,2), Intérêt
number(8,2), Annuité number(8,2));
SET serveroutput on
SET verify off
ACCEPT somme prompt "Entrez la somme empruntée : "
ACCEPT duree prompt "Entrez la durée de l'emprunt : "
ACCEPT taux prompt "Entrez le taux d'intérêt : "
DECLARE
v remb an amorti.somme due%type := &somme/&duree;
v interet amorti.intérêt%type;
v annuite amorti.annuité%type;
v som amorti.somme due%type;
BEGIN
       v som := &somme;
       FOR i IN 1..&duree LOOP
       v_interet := (v_som * &taux)/100;
       v annuite := v remb an + v interet;
       INSERT INTO amorti
       VALUES (i, v_som, v_remb_an, v_interet, v_annuite);
       v som := v som - v remb an;
       END LOOP;
       COMMIT;
END;
select * from amorti;
drop table amorti;
set verify on
bloc anonyme terminé
                                                            INTÉRÉT
DURKE
                    SOMME DUE
                                        REMB AN
                                                                                AMMITTÉ
1
                    12000
                                        4000
                                                            480
                                                                                4480
2
                    8000
                                        4000
                                                            320
                                                                                 4320
3
                    4000
                                        4000
                                                            160
                                                                                 4160
```

5.

```
SET serveroutput on SET verify off
```

ACCEPT num PROMPT 'Table de multiplication de : '

DECLARE

```
TYPE tabch_type IS TABLE OF number(2) INDEX BY BINARY_INTEGER; -- obligatoire tabch_type;
```

BEGIN

END;

```
&num=8
```

```
coll col2
     8
 2 16
    24
 3
 4 32
 5
    40
    48
 6
 7
 8
    64
 9
    72
10
    80
```

6.

```
DROP TABLE AMORTI;
CREATE TABLE AMORTI
(Duree number(2) CONSTRAINT amorti pk primary key, Somme due number(10,2), Remb An number(8,2), Intérêt
number(8,2), Annuité number(8,2));
SET serveroutput on
SET verify off
ACCEPT somme prompt "Entrez la somme empruntée : "
ACCEPT duree prompt "Entrez la durée de l'emprunt : "
ACCEPT taux prompt "Entrez le taux d'intérêt : "
DECLARE
v remb an amorti.somme due%type := &somme/&duree;
v interet amorti.intérêt%type;
v annuite amorti.annuité%type;
v_som amorti.somme_due%type;
```

BEGIN

```
v_som := &somme;
FOR i IN 1..&duree LOOP
v_interet := (v_som * &taux)/100;
v_annuite := v_remb_an + v_interet;
INSERT INTO amorti
VALUES (i, v som, v remb an, v interet, v annuite);
v_som := v_som - v_remb_an;
END LOOP;
COMMIT;
DBMS_OUTPUT.PUT_LINE(rpad('Duree',7)||rpad('Intérêt',10)||'Annuité');
FOR enr amorti IN (SELECT * FROM amorti) LOOP
       DBMS OUTPUT.PUT LINE(rpad(enr amorti.duree,7)||rpad(enr amorti.intérêt,10)||enr amorti.annuité);
END LOOP;
```

END;

bloc anonyme terminé Duree Intérêt Annuité 1 480 4480 2 320 4320 3 160 4160

7.

7 a Solution SQL

```
create table dept as select * from scott.dept;
create table emp as select * from scott.emp;
```

ALTER TABLE dept ADD (budget number(10,2) default 0);

/*default 0 : à l'ajout de la colonne 0 dans la colonne budget pour toutes les lignes. Pour les insertions futures de départements, 0 pour le budget.

```
update dept
set budget = (select nvl(sum(sal),0) from emp
where deptno = dept.deptno);
```

commit;

select * from dept; drop table dept; drop table emp;

```
table DEPT modifié(e).
4 lignes mis à jour.
validé (commit).
                                   BUDGET
   DEPTNO DNAME
                LOC
______ ____
      10 ACCOUNTING NEW YORK
                                      8750
      20 RESEARCH DALLAS
30 SALES CHICAGO
40 OPERATIONS BOSTON
                                      6775
                                      9400
                                        0
table DEPT supprimé(e).
table EMP supprimé(e).
```

7 a Solution PL/SQL

```
set serveroutput on
drop table dept;
drop table emp;
create table dept as select * from scott.dept;
create table emp as select * from scott.emp;
ALTER TABLE dept ADD (budget number(10,2) default 0);
```

DECLARE

CURSOR curseur budget IS select deptno, sum(sal) budget from emp group by deptno;

-- dept 40 non atteint, default 0 nécessaire

BEGIN

```
FOR enr in curseur budget LOOP
       DBMS_OUTPUT.PUT_LINE('Valeur de deptno '||enr.deptno||' Valeur de budget '|| enr.budget);
       update dept
       set budget = enr.budget
       where deptno = enr.deptno;
END LOOP;
COMMIT;
```

END;

```
Valeur de deptno 30 Valeur de budget 9400
Valeur de deptno 20 Valeur de budget 10875
Valeur de deptno 10 Valeur de budget 8750
```

7 b Solution SQL

```
drop table dept;
drop table emp;
create table dept as select * from scott.dept;
create table emp as select * from scott.emp;
```

UPDATE emp SET sal = sal*1.1

WHERE deptno IN (select deptno from dept where upper(loc) IN ('DALLAS','NEW YORK'));

COMMIT;

column ename heading 'Nom|Salarié'

select ename, loc localité, sal salaire from emp JOIN dept ON emp.deptno=dept.deptno where upper(loc) IN ('DALLAS','NEW YORK');

larié	LOC	ALITÉ	SALAIRE
TH	DAL	LAS	880
NES	DAL	LAS	3272,5
ARK	NEW	YORK	2695
DTT	DAL	LAS	3300
1G	NEW	YORK	5500
	larié ITH NES ARK DTT	ITH DAL NES DAL ARK NEW	ITH DALLAS NES DALLAS ARK NEW YORK DTT DALLAS

7 b Solution PL/SQL

set serveroutput on drop table dept; drop table emp; create table dept as select * from scott.dept; create table emp as select * from scott.emp;

DECLARE

CURSOR curseur_sal IS select sal*1.1 nvsal from emp where deptno IN (select deptno from dept where upper(loc) IN ('DALLAS','NEW YORK'))

FOR UPDATE OF SAL; /*verrouillage des lignes de EMP correspondant aux employés travaillant à Dallas ou à New York*/

BEGIN

END;

```
FOR enr in curseur_sal LOOP

update emp

set sal = enr.nvsal

where current of curseur_sal;

END LOOP;

COMMIT;

ename Nom, loc localité, sal salaire
```

select ename Nom, loc localité, sal salaire from emp JOIN dept ON emp.deptno=dept.deptno where upper(loc) IN ('DALLAS','NEW YORK');

⊕ NOM		
SMITH	DALLAS	880
JONES	DALLAS	3272,5
CLARK	NEW YORK	2695
SCOTT	DALLAS	3300
KING	NEW YORK	5500
ADAMS	DALLAS	1210
FORD	DALLAS	3300
MILLER	NEW YORK	1430

7 c Solution SQL

```
drop table dept;
drop table emp;
create table dept as select * from scott.dept;
create table emp as select * from scott.emp;
```

ACCEPT num PROMPT 'Nombre maximum de salaires à afficher : '

select empno, ename, sal from (select empno, ename, sal from emp order by sal desc) where rownum <= #

/* avec num=4 */

		∯ SAL
7839	KING	5000
7788	SCOTT	3000
7902	FORD	3000
7566	JONES	2975

7 c Solution PL/SQL

set verify off

ACCEPT num PROMPT 'Nombre maximum de salaires à afficher : '

DECLARE

```
CURSOR curseur_max_sal IS select empno, ename, sal from emp order by sal DESC; v enrg curseur max sal%ROWTYPE;
```

BEGIN

OPEN curseur max sal:

FETCH curseur max sal INTO v enra:

WHILE curseur_max_sal%ROWCOUNT <= &num AND curseur_max_sal%FOUND LOOP

DBMS_OUTPUT.PUT_LINE('N° EMP ' || v_enrg.empno|| ' NOM '|| v_enrg.ename || ' Salaire '|| v_enrg.sal);

FETCH curseur_max_sal INTO v_enrg;

END LOOP:

IF curseur max sal%ROWCOUNT < &num THEN

DBMS_OUTPUT.PUT_LINE ('Nous ne pouvons afficher que ' ||curseur_max_sal%ROWCOUNT || ' employé(s)'); END IF;

CLOSE curseur max sal;

END;

/* Test avec 16 */

```
bloc anonyme terminé
N° EMP 7839 NOM KING Salaire 5000
N° EMP 7902 NOM FORD Salaire 3000
N° EMP 7566 NOM JONES Salaire 2975
N° EMP 7698 NOM BLAKE Salaire 2850
N° EMP 7782 NOM CLARK Salaire 2450
N° EMP 7499 NOM ALLEN Salaire 1600
N° EMP 7844 NOM TURNER Salaire 1500
N° EMP 7934 NOM MILLER Salaire 1300
N° EMP 7654 NOM MARTIN Salaire 1250
N° EMP 7521 NOM WARD Salaire 1250
N° EMP 7900 NOM JAMES Salaire 950
N° EMP 7369 NOM SMITH Salaire 800
Nous ne pouvons afficher que 12 employé(s)
```