Corrigé des Exercices

Corrigé de l'Exercice 1

1. Initialisez une session SQL*Plus avec votre ID et le mot de passe que votre instructeur vous a remis.
2. Les commandes SQL*Plus accèdent aux bases de données.
   Faux
3. L'ordre SELECT suivant sera convenablement exécuté.
   Vrai

```
SQL> SELECT ename, job, sal Salary
  2  FROM   emp;
```

4. L'ordre SELECT suivant sera convenablement exécuté.
   Vrai

```
SQL> SELECT *
  2  FROM   salgrade;
```

5. Cet ordre comporte trois erreurs de code ; pouvez-vous les trouver ?

```
SQL> SELECT empno, ename
  2         sal x 12 ANNUAL SALARY
  3  FROM   emp;
```

- **Il manque une virgule après ENAME sur la première ligne.**
- **L'opérateur de multiplication est *, et non pas x, comme indiqué à la ligne 2.**
- **L'alias ANNUAL SALARY ne doit pas contenir d'espace. Vous devez écrire ANNUAL_SALARY ou l'inclure entre guillemets.**

6. Affichez la structure de la table DEPT. Sélectionnez toutes les données de la table DEPT.

```
SQL> DESCRIBE dept
SQL> SELECT *
  2  FROM   dept;
```

7. Affichez la structure de la table EMP. Créez une requête pour afficher le nom (name), le poste (job), la date d'embauche (hire date) et le matricule (empno) de chaque employé, en plaçant le matricule en premier. Enregistrez votre ordre SQL dans un fichier nommé *p1q7.sql*.

```
SQL> DESCRIBE emp
```

Corrigé de l'Exercice 1 (suite)

```
SQL> SELECT empno, ename, job, hiredate
  2  FROM   emp;
SQL> SAVE p1q7.sql
Wrote file p1q7.sql
```

8. Exécutez la requête que vous avez placée dans le fichier *p1q7.sql*.

```
SQL> START p1q7.sql
```

9. Créez une requête pour afficher chaque type de poste (job) différent existant dans la table EMP.

```
SQL> SELECT DISTINCT job
  2  FROM   emp;
```

10. Editez *p1q7.sql*. Donnez respectivement les noms suivants aux en-têtes de colonne : Emp #, Employee, Job, et Hire Date. Exécutez à nouveau votre requête.

```
SQL> GET p1q7.sql
  1  SELECT empno, ename, job, hiredate
  2* FROM   emp
SQL> 1 SELECT   empno "Emp #", ename "Employee",
SQL> i
 2i  job "Job", hiredate "Hire Date"
 3i
SQL> SAVE p1q7.sql REPLACE
Wrote file p1q7.sql
SQL> START p1q7.sql
```

11. Affichez le nom concaténé avec le poste en les séparant par une virgule suivie d'un espace, puis nommez la colonne "Employee and Title".

```
SQL> SELECT     ename||', '||job "Employee and Title"
  2  FROM       emp;
```

Corrigé de l'Exercice 1 (suite)

12. Créez une requête pour afficher toutes les données de la table EMP dans une seule colonne d'affichage. Séparez chaque colonne par une virgule. Nommez la colonne d'affichage THE_OUTPUT.

```
SQL> SELECT empno || ',' || ename || ','|| job || ',' ||
  2         mgr || ',' || hiredate || ',' || sal || ',' ||
  3         comm || ',' || deptno THE_OUTPUT
  4  FROM   emp;
```

Corrigé de l'Exercice 2

1. Créez une requête destinée à afficher le nom et le salaire des employés gagnant plus de $2850. Enregistrez l'ordre SQL créé dans un fichier nommé *p2q1.sql*. Exécutez votre requête.

```
SQL> SELECT   ename, sal
  2  FROM     emp
  3  WHERE    sal > 2850;
```

```
SQL> SAVE p2q1.sql
Created file p2q1.sql;
```

2. Créez une requête destinée à afficher le nom et le numéro de département de l'employé dont le matricule est 7566.

```
SQL> SELECT   ename, deptno
  2  FROM     emp
  3  WHERE    empno = 7566;
```

3. Modifiez *p2q1.sql* de manière à afficher le nom et le salaire de tous les employés dont le salaire n'est pas compris entre $1500 et $2850. Enregistrez ce nouvel votre ordre SQL dans un fichier nommé *p2q3.sql*. Exécutez cette requête.

```
SQL> EDIT p2q1.sql

     SELECT    ename, sal
     FROM      emp
     WHERE     sal NOT BETWEEN 1500 AND 2850
     /

SQL> START p2q3.sql
```

4. Affichez le nom, le poste et la date d'entrée (hire date) des employés embauchés entre le 20 février 1981 et le 1 mai 1981. Classez le résultat par date d'embauche croissante.

```
SQL> SELECT   ename, job, hiredate
  2  FROM     emp
  3  WHERE    hiredate BETWEEN '20-Feb-81' AND '01-May-81'
  4  ORDER BY hiredate;
```

Corrigé de l'Exercice 2 (suite)

5. Affichez le nom et le numéro de département de tous les employés des départements 10 et 30, en classant les noms par ordre alphabétique.

```
SQL> SELECT    ename, deptno
  2  FROM      emp
  3  WHERE     deptno IN (10, 30)
  4  ORDER BY  ename;
```

6. Modifiez *p2q3.sql* pour afficher la liste des noms et salaires des employés gagnant plus de $1500 et travaillant dans le département 10 ou 30. Nommez les colonnes Employee et Monthly Salary, respectivement. Enregistrez à nouveau votre ordre SQL dans un fichier nommé *p2q6.sql*. Réexécutez votre requête.

```
SQL> EDIT p2q3.sql
     SELECT    ename "Employee", sal "Monthly Salary"
     FROM      emp
     WHERE     sal  > 1500
     AND       deptno IN (10, 30)
     /
SQL> START p2q6.sql
```

7. Affichez le nom et la date d'embauche de chaque employé entré en 1982.

```
SQL> SELECT   ename, hiredate
  2  FROM     emp
  3  WHERE    hiredate LIKE '%82';
```

8. Affichez le nom et le poste de tous les employés n'ayant pas de manager.

```
SQL> SELECT   ename, job
  2  FROM     emp
  3  WHERE    mgr IS NULL;
```

9. Affichez le nom, le salaire et la commission de tous les employés qui perçoivent des commissions. Triez les données dans l'ordre croissant des salaires et des commissions.

```
SQL> SELECT   ename, sal, comm
  2  FROM     emp
  3  WHERE    comm IS NOT NULL
  4  ORDER BY sal DESC, comm DESC;
```

Corrigé de l'Exercice 2 (suite)

10. Affichez le nom de tous les employés dont la troisième lettre du nom est un *A*.
Remarque : Deux caractères de soulignement (_) sont placés avant le *A* dans la clause WHERE.

```
SQL> SELECT   ename
  2  FROM     emp
  3  WHERE    ename LIKE '__A%';
```

11. Affichez le nom de tous les employés dont le nom contient deux *L* et travaillant dans le département 30 ou dont le manager est 7782.

```
SQL> SELECT   ename
  2  FROM     emp
  3  WHERE    ename LIKE '%L%L%'
  4  AND      deptno = 30
  5  OR       mgr = 7782;
```

12. Affichez le nom, le poste et le salaire de tous les ' CLERK' ou 'ANALYST' dont le salaire est différent de $1000, $3000 ou $5000.

```
SQL> SELECT   ename, job, sal
  2  FROM     emp
  3  WHERE    job IN ('CLERK', 'ANALYST')
  4  AND      sal NOT IN (1000, 3000, 5000);
```

13. Afficher le nom, le salaire et la commission de tous les employés dont le montant de commission est de plus de 10% supérieur au salaire. Enregistrez votre requête sous le nom *p2q13.sql*.

```
SQL> SELECT   ename, sal, comm
     FROM     emp
     WHERE    comm  > sal * 1.1
     /
SQL> START p2q13.sql
```

Corrigé de l'Exercice 3

1. Ecrivez une requête pour afficher la date courante. Nommez la colonne Date.

```
SQL> SELECT sysdate "Date"
  2  FROM   dual;
```

2. Affichez pour chaque employé, le matricule, le nom, le salaire et le salaire augmenté de 15% sous la forme d'un nombre entier. Nommez cette colonne New Salary. Enregistrez votre ordre SQL dans un fichier appelé *p3q2.sql*.

```
SQL> SELECT empno, ename, sal,
  2             ROUND(sal * 1.15, 0) "New Salary"
  3  FROM   emp;

SQL> SAVE p3q2.sql
Created file p3q2.sql;
```

3. Exécutez votre requête à partir du fichier *p3q2.sql*.

```
SQL> START p3q2.sql
```

4. Modifiez votre requête *p3q2.sql* en ajoutant une colonne dans laquelle l'ancien salaire est soustrait du nouveau salaire. Nommez cette colonne Increase. Exécutez à nouveau votre requête.

```
SQL> EDIT p3q2.sql
     SELECT  empno, ename, sal,
             ROUND(sal * 1.15, 0) "New Salary",
             ROUND(sal * 1.15, 0) - sal "Increase"
     FROM    emp
     /
 SQL> START p3q2.sql
```

5. Affichez le nom et la date d'embauche de chaque employé ainsi que la date de révision du salaire qui sera le premier lundi tombant après 6 mois d'activité. Nommez la colonne REVIEW. Les dates devront apparaître dans le format suivant : "Sunday, the Seventh of September, 1981".

```
SQL> SELECT ename, hiredate,
  2             TO_CHAR(NEXT_DAY(ADD_MONTHS(hiredate, 6),
  3             'MONDAY'),
  4             'fmDay, "the" Ddspth "of" MonthYYYY')REVIEW
  5             FROM emp;
```

Corrigé de l'Exercice 3 (suite)

6. Affichez le nom de chaque employé et calculez le nombre de mois travaillés depuis la date d'embauche. Nommez la colonne MONTHS_WORKED. Classez les résultats en fonction du nombre de mois d'ancienneté. Arrondissez le nombre de mois au nombre entier supérieur.

```
SQL> SELECT       ename, ROUND(MONTHS_BETWEEN
  2               (SYSDATE, hiredate)) MONTHS_WORKED
  3  FROM         emp
  4  ORDER BY     MONTHS_BETWEEN(SYSDATE, hiredate);
```

7. Ecrivez une requête affichant les informations suivantes pour chaque employé : <nom de l'employé> gagne <salaire> par mois, mais veut <3 fois son salaire>. Nommez la colonne Salaires de Rêve.

```
SQL> SELECT  ename || ' gagne '
  2          || TO_CHAR(sal, 'fm$99,999.00')
  3          || ' par mois, mais veut '
  4          || TO_CHAR(sal * 3, 'fm$99,999.00')
  5          || '.' "Salaires de Rêve"
  6  FROM    emp;
```

8. Créez une requête pour afficher le nom et le salaire de tous les employés. Le salaire sera formaté de façon à avoir 15 caractères de long, les blancs étant comblés à gauche par des $. Nommez la colonne SALARY.

```
SQL> SELECT  ename,
  2          LPAD(sal, 15, '$') SALARY
  3  FROM    emp;
```

9. Ecrivez une requête pour afficher tous les noms d'employé commençant par les lettres *J*, *A*, ou *M*, ainsi que la longueur du nom. Le nom doit apparaître en minuscules, sauf l'initiale qui sera en majuscules. Donnez à chaque colonne un nom approprié.

```
SQL> SELECT  INITCAP(ename) "Name",
  2          LENGTH(ename) "Length"
  3  FROM    emp
  4  WHERE   ename LIKE 'J%'
  5  OR      ename LIKE 'M%'
  6  OR      ename LIKE 'A%';
```

Corrigé de l'Exercice 3 (suite)

10. Affichez le nom, la date d'embauche ainsi que le jour de la semaine où l'employé à débuté. Nommez la colonne JOUR. Classez les résultats dans l'ordre des jours de la semaine à partir du lundi (monday).

```
SQL> SELECT   ename, hiredate,
  2           TO_CHAR(hiredate, 'DAY') JOUR
  3  FROM     emp
  4  ORDER BY TO_CHAR(hiredate - 1, 'd');
```

11. Créez une requête pour afficher le nom et le montant de la commission de chaque employé. Pour les employés ne touchant aucune commission, affichez "No Commission". Nommez la colonne COMM.

```
SQL> SELECT   ename,
  2           NVL(TO_CHAR(comm), 'No Commission') COMM
  3  FROM     emp;
```

12. Créez une requête pour afficher le nom des employés et leur salaire indiqué par des astérisques. Chaque astérisque représente cent dollars. Triez les données dans l'ordre décroissant des salaires. Nommez la colonne EMPLOYEE_AND_THEIR_SALARIES.

```
SQL> SELECT     rpad(ename, 8) ||rpad(' ', sal/100, '*')
  2                 EMPLOYEE_AND_THEIR_SALARIES
  3  FROM       emp
  4  ORDER BY sal DESC;
```

Corrigé de l'Exercice 4

1. Ecrivez une requête pour afficher le nom, le numéro de département et le département de tous les employés.

```
SQL> SELECT     e.ename, e.deptno, d.dname
  2  FROM       emp e, dept d
  3  WHERE      e.deptno = d.deptno;
```

2. Créez une liste unique de tous les postes du département 30.

```
SQL> SELECT     DISTINCT e.job, d.loc
  2  FROM       emp e, dept d
  3  WHERE      e.deptno = d.deptno
  4  AND        e.deptno = 30;
```

3. Ecrivez une requête pour afficher le nom, le département et la localisation de tous les employés qui touchent une commission.

```
SQL> SELECT     e.ename, d.dname, d.loc
  2  FROM       emp e, dept d
  3  WHERE      e.deptno = d.deptno
  4  AND        e.comm IS NOT NULL;
```

4. Affichez le nom et le nom du département pour tous les employés dont le nom contient la lettre A. Enregistrez votre ordre SQL dans un fichier que vous nommerez *p4q4.sql*.

```
SQL> SELECT     e.ename, d.dname
  2  FROM       emp e, dept d
  3  WHERE      e.deptno = d.deptno
  4  AND        e.ename LIKE '%A%';
```

5. Ecrivez une requête pour afficher le nom, le poste, le numéro de département et le département de tous les employés basés à DALLAS.

```
SQL> SELECT     e.ename, e.job, e.deptno, d.dname
  2  FROM       emp e, dept d
  3  WHERE      e.deptno = d.deptno
  4  AND        d.loc = 'DALLAS';
```

Corrigé de l'Exercice 4 (suite)

6. Affichez le nom et le matricule des employés et de leur manager. Nommez les colonnes Employee, Emp#, Manager, et Mgr#, respectivement. Enregistrez votre ordre SQL dans un fichier nommé *p4q6.sql*.

```
SQL> SELECT  e.ename "Employee", e.empno "Emp#",
  2          m.ename "Manager", m.empno "Mgr#"
  3  FROM    emp e, emp m
  4  WHERE   e.mgr = m.empno;
SQL> SAVE p4q6.sql.
 Created file p4q6.sql;
```

7. Modifiez le fichier *p4q6.sql* pour afficher tous les employés, y compris King, n'ayant pas de manager. Enregistrez à nouveau dans un fichier *p4q7.sql*. Exécutez *p4q7.sql*.

```
SQL> EDIT p4q6.sql
     SELECT e.ename "Employee", e.empno"Emp#",
            m.ename "Manager", m.empno "Mgr#"
     FROM   emp e, emp m
     WHERE  e.mgr = m.empno(+)
     /
SQL> START p4q7.sql
```

8. Créez une requête pour afficher le numéro de département et le nom de tous les employés qui travaillent dans le même département qu'un certain employé. Donnez à chaque colonne un nom approprié.

```
SQL> SELECT e.deptno department, e.ename employee,
  2         c.ename colleague
  3  FROM   emp e, emp c
  4  WHERE  e.deptno = c.deptno
  5  AND    e.empno <> c.empno
  6  ORDER BY    e.deptno, e.ename, c.ename;
```

Corrigé de l'Exercice 4 (suite)

9. Affichez la structure de la table SALGRADE. Créez une requête pour afficher le nom, le poste, le département, le salaire et l'échelon de tous les employés.

```
SQL> DESCRIBE salgrade
SQL> SELECT e.ename, e.job, d.dname, e.sal, s.grade
  2  FROM   emp e, dept d, salgrade s
  3  WHERE  e.deptno = d.deptno
  4  AND    e.sal BETWEEN s.losal AND s.hisal;
```

10. Créez une requête pour afficher le nom et la date d'embauche de tous les employés arrivés avant l'employé Blake. Trier les lignes sur la date d'embauche.

```
SQL> SELECT emp.ename, emp.hiredate
  2  FROM   emp, emp blake
  3  WHERE  blake.ename = 'BLAKE'
  4  AND    blake.hiredate > emp.hiredate
  5  ORDER BY emp.hiredate;
```

11. Affichez les noms et date d'embauche des employés et de leur manager, pour tous les employés ayant été embauchés avant leur manager. Nommez les colonnes Employee, Emp Hiredate, Manager et Mgr Hiredate, respectivement.

```
SQL> SELECT e.ename "Employee", e.hiredate "Emp Hiredate",
  2         m.ename "Manager", m.hiredate "Mgr Hiredate"
  3  FROM   emp e, emp m
  4  WHERE  e.mgr = m.empno
  5  AND    e.hiredate < m.hiredate;
```

Corrigé de l'Exercice 5

Déterminez si les affirmations suivantes sont vraies ou fausses et entourez la réponse correspondante.

1. Les fonctions de groupe agissent sur plusieurs lignes pour produire un seul résultat.

   Vrai

2. Les fonctions de groupe intègrent les valeurs NULL dans leurs calculs.

   Faux. Les fonctions de groupe ignorent les valeurs NULL. Si vous voulez inclure des valeurs
   NULL, utilisez la fonction NVL.

3. La clause WHERE restreint les lignes avant qu'elles soient incluses dans un calcul de groupe.

   Vrai

4. Affichez le salaire maximum, le salaire minimum, la somme des salaires et le salaire moyen de tous les employés. Nommez respectivement les colonnes Maximum, Minimum, Sum et Average.
   Arrondissez les résultats à zéro décimale. Enregistrez votre ordre SQL dans un fichier nommé *p5q4.sql*.

```
SQL> SELECT   ROUND(MAX(sal),0) "Maximum",
  2           ROUND(MIN(sal),0) "Minimum",
  3           ROUND(SUM(sal),0) "Sum",
  4           ROUND(AVG(sal),0) "Average"
  5  FROM     emp;
SQL> SAVE p5q4.sql
Created file p5q4.sql
```

5. Modifiez le fichier *p5q4.sql* pour afficher le salaire minimum, le salaire maximum, la somme des salaires et le salaire moyen pour chaque type de poste. Enregistrez votre fichier sous *p5q5.sql*.
   Exécutez à nouveau votre requête.

```
SQL> EDIT p5q4.sql
     SELECT   job, ROUND(MAX(sal),0) "Maximum",
              ROUND(MIN(sal),0) "Minimum",
              ROUND(SUM(sal),0) "Sum",
              ROUND(AVG(sal),0) "Average"
     FROM     emp
     GROUP BY job
     /
SQL> SAVE p5q5.sql
SQL> START p5q5.sql
```

Corrigé de l'Exercice 5 (suite)

6. Ecrivez une requête pour afficher le nombre de personnes qui occupent le même poste.

```
SQL> SELECT   job, COUNT(*)
  2  FROM     emp
  3  GROUP BY job;
```

7. Déterminez le nombre de personnes ayant des subordonnés, sans en donner la liste. Nommez la colonne "Nombre de Chefs".

```
SQL> SELECT   COUNT(DISTINCT mgr) "Nombre de Chefs"
  2  FROM     emp;
```

8. Ecrivez une requête pour afficher la différence existant entre le salaire maximum et le salaire minimum. Nommez la colonne DIFFERENCE.

```
SQL> SELECT   MAX(sal) - MIN(sal) DIFFERENCE
  2  FROM     emp;
```

9. Affichez le matricule des différents managers et le niveau de salaire le plus bas de leurs employés. Excluez toute ligne où le manager n'est pas identifié. Excluez tout groupe dans lequel le salaire minimum est inférieur à $1000. Triez les résultats par ordre décroissant des salaires.

```
SQL> SELECT   mgr, MIN(sal)
  2  FROM     emp
  3  WHERE    mgr IS NOT NULL
  4  GROUP BY mgr
  5  HAVING   MIN(sal) > 1000
  6  ORDER BY  MIN(sal) DESC;
```

10. Ecrivez une requête pour afficher le nom du département, la localisation, le nombre d'employés et le salaire moyen pour tous les employés de ce département. Nommez les colonnes dname, loc, Nombre d'Employés et Salaire, respectivement.

```
SQL> SELECT   d.dname, d.loc, COUNT(*) "Nombre d''Employés",
  2           ROUND(AVG(e.sal),2) "Salaire"
  3  FROM     emp e, dept d
  4  WHERE    e.deptno = d.deptno
  5  GROUP BY d.dname, d.loc;
```

Corrigé de l'Exercice 5 (suite)

11. Créez une requête pour afficher le nombre total d'employés puis, parmi ces employés, ceux qui ont été embauchés en 1980, 1981, 1982 et 1983. Nommez les colonnes de façon appropriée.

```
SQL> SELECT COUNT(*) total,
  2         SUM(DECODE(TO_CHAR(hiredate, 'YYYY'),
  3                              1980,1,0))"1980",
  4         SUM(DECODE(TO_CHAR(hiredate, 'YYYY'),
  5                              1981,1,0))"1981",
  6         SUM(DECODE(TO_CHAR(hiredate, 'YYYY'),
  7                              1982,1,0))"1982",
  8          SUM(DECODE(TO_CHAR(hiredate, 'YYYY'),
  9                              1983,1,0))"1983"
 10  FROM   emp;
```

12. Créez une requête pour afficher les postes, le salaire de ces postes par numéro de département et le salaire total de ces postes incluant tous les départements. Nommez les colonnes de façon appropriée.

```
SQL> SELECT     job "Job",
  2             SUM(DECODE(deptno, 10, sal)) "Dept 10",
  3             SUM(DECODE(deptno, 20, sal)) "Dept 20",
  4             SUM(DECODE(deptno, 30, sal)) "Dept 30",
  5             SUM(sal) "Total"
  6  FROM       emp
  7  GROUP BY   job;
```

Corrigé de l'Exercice 6

1. Affichez le département qui ne comprend aucun employé.

```
SQL> SELECT deptno,dname
  2  FROM dept
  3  MINUS
  4  SELECT e.deptno,d.dname
  5  FROM emp e,dept d
  6  WHERE e.deptno = d.deptno;
```

2. Retrouvez le poste qui a été attribué dans le deuxième semestre de l'année 1981 **et** de l'année 1982.

```
SQL> SELECT job
 2  FROM emp
 3  WHERE hiredate BETWEEN '01-JUL-81' AND '30-DEC-81'
 4  INTERSECT
 5  SELECT job
 6  FROM emp
 7  WHERE hiredate BETWEEN '01-JUL-82' AND '30-DEC-82';
```

Corrigé de l'Exercice 6 (suite)

3. Ecrivez une requête composée pour afficher la liste des produits (utiliser la table PRICE) en indiquant le pourcentage de remise, l'identifiant des produits (PRODID), ainsi que le nouveau prix (avec remise) et l'ancien prix (STDPRICE):
   - les produits dont le prix est inférieur à $10 sont réduits de 10%
   - ceux dont le prix est compris entre $10 et $30 sont réduits de 15%
   - ceux dont le prix est supérieur à $30 sont réduits de 20%
   - ceux dont le prix est supérieur à $40 ne sont pas réduits.

```
COL discount format a8
SELECT '10% off' discount,prodid, stdprice,stdprice *.9
actprice
FROM price
WHERE stdprice < 10
UNION
SELECT '15% off',prodid, stdprice,stdprice * .85
FROM price
WHERE stdprice between 10 and 30
UNION
SELECT '20% off' ,prodid, stdprice,stdprice * .8
FROM price
WHERE stdprice > 30 and stdprice <= 40
UNION
SELECT 'no disc', prodid, stdprice,stdprice
FROM price
WHERE stdprice > 40
/
```

Corrigé de l'Exercice 6 (suite)

4. Affichez la liste des postes dans les départements 10, 30 et 20, en conservant cet ordre. Affichez le poste et le numéro du département.

```
col dummy noprint

SELECT job,deptno,'x' dummy
FROM emp
WHERE deptno = 10
UNION
SELECT job,deptno,'y'
FROM emp
WHERE deptno = 30
UNION
SELECT job,deptno,'z'
FROM emp
WHERE deptno = 20
ORDER BY 3
/
```

Corrigé de l'Exercice 6 (suite)

5. Affichez le numéro des départements dans lesquels on ne trouve pas de poste ANALYST .

```
SQL>SELECT deptno
  2 FROM dept
  3 MINUS
  4 SELECT deptno
  5 FROM emp
  6 WHERE job = 'ANALYST';
```

6. Affichez tous les postes des départements 10 et 20 qui n'existent que dans l'un ou l'autre de ces départements.

```
SQL>SELECT job
  2 FROM emp
  3 WHERE deptno in (20,10)
  4 MINUS
  5 (SELECT job
  6 FROM emp
  7 WHERE deptno = 10
  8 INTERSECT
  9 SELECT job
 10 FROM emp
 11 WHERE deptno = 20);
```

Corrigé de l'Exercice 7

1. Créez une requête pour afficher le nom et la date d'embauche de tous les employés travaillant dans le même département que Blake, à l'exclusion de Blake.

```
SQL> SELECT   ename, hiredate
  2  FROM     emp
  3  WHERE    deptno IN (SELECT     deptno
  4                      FROM       emp
  5                      WHERE      ename = 'BLAKE')
  6  AND      ename != 'BLAKE';
```

2. Créez une requête pour afficher le matricule et le nom de tous les employés qui gagnent plus que le salaire moyen. Triez les résultats par ordre décroissant des salaires.

```
SQL> SELECT   empno, ename
  2  FROM     emp
  3  WHERE    sal > (SELECT AVG(sal)
  4                   FROM emp)
  5  ORDER BY sal DESC;
```

3. Ecrivez une requête pour afficher le matricule et le nom de tous les employés qui travaillent dans le même département que tout employé dont le nom contient un *T*. Enregistrez votre ordre SQL dans un fichier nommé *p7q3.sql*.

```
SQL> SELECT   empno, ename
  2  FROM     emp
  3  WHERE    deptno IN (SELECT     deptno
  4                      FROM       emp
  5                      WHERE      ename LIKE '%T%');
SQL> SAVE p7q3.sql
Created file p7q3.sql
```

4. Modifiez *p7q3.sql* afin d'afficher le matricule, le nom et le salaire de tous les employés qui gagnent plus que le salaire moyen global et qui travaillent dans un département avec tout employédont le nom contient un *T*. Enregistrez à nouveau votre requête sous le nom *p7q4.sql*, puis réexécutez-la.

```
SQL>  EDIT p7q3.sql
     SELECT empno, ename, sal
         FROM   emp
     WHERE  sal >  (SELECT AVG(sal)
                    FROM   emp)
     AND deptno IN (SELECT deptno
                    FROM               emp
                    WHERE  ename LIKE '%T%')
      /
SQL> SAVE  p7q4.sql
SQL> START p7q4.sql
```

Corrigé de l'Exercice 7 (suite)

5. Affichez le nom, le numéro de département et le poste de tous les employés dont le département est situé à Dallas.

```
SQL> SELECT  ename, deptno, job
  2  FROM  emp
  3  WHERE deptno IN (SELECT deptno
  4              FROM  dept
  5              WHERE loc = 'DALLAS');
```

6. Affichez le nom et le salaire de tous les employés dont le manager est King.

```
SQL> SELECT  ename, sal
  2  FROM    emp
  3  WHERE mgr IN (SELECT    empno
  4           FROM   emp
  5           WHERE    ename = 'KING');
```

7. Affichez le numéro de département, le nom et le poste de tous les employés travaillant dans le département des ventes (Sales).

```
SQL> SELECT  deptno, ename, job
  2  FROM    emp
  3  WHERE deptno IN (SELECT deptno
  4                FROM    dept
  5                WHERE   dname = 'SALES');
```

8. Créez une requête pour afficher les employés qui perçoivent un salaire supérieur à tout employé dont le poste est CLERK. Triez le résultat dans l'ordre décroissant des salaires.

```
SQL> SELECT  ename, job, sal
  2  FROM  emp
  3  WHERE sal > ALL (SELECT sal
  4                   FROM   emp
  5                   WHERE  job = 'CLERK')
  6  ORDER BY  sal DESC;
```

Corrigé de l'Exercice 8

1. Ecrivez une requête pour afficher le nom, le numéro de département et le salaire de tout employé dont le numéro de département et le salaire sont tous les deux équivalents au numéro de département et au salaire de n'importe quel employé touchant une commission.

```
SQL> SELECT   ename, deptno, sal
  2  FROM     emp
  3  WHERE    (sal, deptno) IN
  4                        (SELECT    sal, deptno
  5                         FROM      emp
  6                         WHERE     comm IS NOT NULL);
```

2. Affichez le nom de l'employé, le nom du département et le salaire de tout employé dont le salaire et la commission sont tous les deux équivalents au salaire et à la commission de n'importe quel employé basé à Dallas.

```
SQL> SELECT   ename, dname, sal
  2  FROM     emp e, dept d
  3  WHERE    e.deptno = d.deptno
  4  AND      (sal, NVL(comm,0)) IN
  5                        (SELECT    sal, NVL(comm,0)
  6                         FROM      emp e, dept d
  7                         WHERE     e.deptno = d.deptno
  8                         AND       d.loc = 'DALLAS');
```

3. Créez une requête pour afficher le nom, la date d'embauche et le salaire pour tous les employés touchant le même salaire et la même commission que Scott.

```
SQL> SELECT   ename, hiredate, sal
  2  FROM     emp
  3  WHERE    (sal, NVL(comm,0)) IN
  4                        (SELECT    sal, NVL(comm,0)
  5                         FROM      emp
  6                         WHERE     ename = 'SCOTT')
  7  AND      ename != 'SCOTT';
```

Corrigé de l'Exercice 9

1. Ecrivez une requête pour afficher les trois meilleurs salaires dans la table EMP. Affichez les noms des employés et leur salaire.

```
SQL>SELECT ename, sal
  2  FROM emp e
  3  WHERE 3  >  (SELECT COUNT(*)
  4                FROM emp
  5                WHERE e.sal < sal);
```

2. Recherchez tous les employés qui ne sont pas des responsables.
   a. Utilisez d'abord l'opérateur EXISTS.

```
SQL>SELECT outer.ename
  2  FROM emp outer
  3  WHERE NOT EXISTS (SELECT empno
  4                    FROM emp inner
  5                    WHERE inner.mgr = outer.empno);
```

   b. Pouvez-vous effectuer cette opération à l'aide de l'opérateur IN ? Pourquoi ?

```
SQL>SELECT outer.ename
  2  FROM emp outer
  3  WHERE outer.empno NOT IN (SELECT inner.mgr
                               FROM emp inner);
```

Cette solution est incorrecte. Comme la sous-requête sélectionne une valeur NULL, la requête entière ne ramènera aucune ligne. Toutes les conditions comparant une valeur NULL retournent un résultat NULL. Ainsi, chaque fois que des valeurs NULL risquent d'être intégrées à un ensemble de valeurs, *n'*utilisez *pas* NOT IN à la place de l'opérateur NOT EXISTS.

Pour que cette solution donne le bon résultat il faudrait intégrer une clause de sélection supplémentaire dans la sous-interrogation :

```
SQL>SELECT outer.ename
 2  FROM emp outer
 3  WHERE outer.empno NOT IN (SELECT inner.mgr
                              FROM emp inner
                              WHERE inner.mgr IS NOT NULL);
```

Corrigé de l'Exercice 9 (suite)

3. Ecrivez une requête pour rechercher tous les employés dont le salaire est supérieur au salaire moyen de leur département. Affichez le numéro de chaque employé, son salaire, son numéro de département et le salaire moyen du département.

```
SQL> SELECT e.ename ,e.sal salary,e.deptno ,AVG(a.sal) dept_avg
  2  FROM   emp e, emp a
  3  WHERE  e.deptno = a.deptno
  4  AND    e.sal > (SELECT AVG(sal)
  5                  FROM emp
  6                  WHERE deptno = e.deptno)
  7  GROUP BY e.ename,e.sal,e.deptno
  8  ORDER BY AVG(a.sal);
```

Autre solution :

```
SQL> SELECT e.ename ,e.sal salary,e.deptno ,a.dept_avg
  2  FROM   emp e, (SELECT deptno, AVG(sal) dept_avg
  3                  FROM emp
  4                  GROUP BY DEPTNO) a
  5  WHERE  e.deptno = a.deptno
  6  AND    e.sal > a.dept_avg
  7  ORDER BY a.dept_avg;
```

Corrigé de l'Exercice 9 (suite)

4. Ecrivez une requête pour afficher les employés dont le salaire est inférieur à la moitié du salaire moyen de leur département.

```
SQL>SELECT  ename
  2 FROM emp outer
  3 WHERE outer.sal < (SELECT avg(inner.sal/2)
  4                    FROM emp inner
  5                    WHERE inner.deptno = outer.deptno);
```

5. Ecrivez une requête pour afficher les employés ayant un ou plusieurs collègues de leur département dont les dates d'embauche sont postérieures aux leurs et dont les salaires sont plus élevés que les leurs.

```
SQL>SELECT  ename
  2 FROM emp outer
  3 WHERE EXISTS (SELECT 'Rien'
  4               FROM emp inner
  5               WHERE inner.deptno = outer.deptno
  6               AND inner.hiredate > outer.hiredate
  7               AND inner.sal > outer.sal);
```

Corrigé de l'Exercice 10

Indiquez si les affirmations ci-dessous sont vraies ou fausses :

1. Une variable de substitution à simple "et commercial" (&) n'affiche un message d'invite qu'une seule fois.

   Vrai
   Cependant, si la variable est définie, la variable de substitution à simple perluète n'affichera pas d'invite. Elle prend la valeur de la variable prédéfinie.

2. La commande ACCEPT est une commande SQL.

   Faux
   La commande ACCEPT est une commande SQL*Plus qui est émise à la suite du prompt SQL.

3. Les tâches de la colonne de gauche ne correspondent pas aux commandes de la colonne de droite. Tracez des lignes pour indiquer la commande appropriée à chaque tâche.

Tâche		**Commande**
Modifier les caractéristiques d'affichage d'une colonne		SET LINESIZE
Calculer et imprimer des totaux		TTITLE
Contrôler le nombre de caractères par ligne		COLUMN
Ajouter un titre à l'état		BREAK
Forcer un saut de page à une rupture		SET NEWPAGE
Déterminer le nombre de lignes blanches en haut de page		COMPUTE

Corrigé de l'Exercice 10 (suite)

4. Ecrivez un script pour afficher le nom, le poste et le nom du département des employés travaillant dans un département déterminé. La recherche doit pouvoir s'effectuer indifféremment sur des majuscules ou des minuscules. Enregistrez le fichier script sous *p10q4.sql*.

```
SET ECHO OFF
SET VERIFY OFF
ACCEPT p_location PROMPT 'Please enter the location name: '
COLUMN ename HEADING "EMPLOYEE NAME" FORMAT A15
COLUMN dname HEADING "DEPARTMENT NAME" FORMAT A15
SELECT e.ename, e.job, d.dname
FROM   emp e, dept d
WHERE  e.deptno = d.deptno
AND    LOWER(d.loc) LIKE LOWER('%&p_location%')
/
UNDEFINE p_location
COLUMN ename CLEAR
COLUMN dname CLEAR
SET VERIFY ON
SET ECHO ON
SQL> START p10q4.sql
```

5. Ecrivez un fichier script pour afficher le nom, le poste et la date d'embauche de tous employés ayant commencé entre deux dates spécifiées. Concaténez le nom et le poste en les séparant par une virgule et un espace, et nommez la colonne "Employees". Demandez à
l'utilisateur d'entrer l'intervalle de dates au moyen de la commande ACCEPT. Utilisez le format MM/DD/YY. Enregistrez le fichier sous *p10q5.sql*.

```
SET ECHO OFF
SET VERIFY OFF
ACCEPT low_date DATE FORMAT 'MM/DD/YY' -
PROMPT 'Please enter the low date range (''MM/DD/YY''): '
ACCEPT high_date DATE FORMAT 'MM/DD/YY' -
PROMPT 'Please enter the high date range (''MM/DD/YY''): '
COLUMN EMPLOYEES FORMAT A25
SELECT ename ||', '|| job EMPLOYEES, hiredate
FROM   emp
WHERE  hiredate BETWEEN
           TO_DATE('&low_date', 'MM/DD/YY')
            AND TO_DATE('&high_date', 'MM/DD/YY')
/
UNDEFINE low_date
UNDEFINE high_date
COLUMN EMPLOYEES CLEAR
SET VERIFY ON
SET ECHO ON
SQL> START p10q5.sql
```

Corrigé de l'Exercice 10 (suite)

6. Modifiez *p10q4.sql* pour créer un état contenant le nom du département, le nom, la date d'embauche, le salaire et le salaire annuel de tous les employés d'une même localisation. Demandez à l'utilisateur d'entrer cette localisation. Nommez les colonnes DEPARTMENT NAME, EMPLOYEE NAME, START DATE, SALARY et ANNUAL SALARY, en répartissant les en-têtes sur plusieurs lignes. Enregistrez de nouveau le script sous le nom *p10q6.sql*.

```
    SET ECHO OFF
    SET FEEDBACK OFF
    SET VERIFY OFF
    BREAK ON  dname
    ACCEPT p_location PROMPT 'Please enter the location
name:'
    COLUMN dname HEADING "DEPARTMENT|NAME" FORMAT A15
    COLUMN ename HEADING "EMPLOYEE|NAME" FORMAT A15
    COLUMN hiredate HEADING "START|DATE" FORMAT A15
    COLUMN sal HEADING "SALARY" FORMAT $99,990.00
    COLUMN asal HEADING "ANNUAL|SALARY" FORMAT $99,990.00
    SELECT    d.dname, e.ename, e.hiredate,
              e.sal, e.sal * 12 asal
    FROM      emp e, dept d
    WHERE     e.deptno = d.deptno
    AND       LOWER(d.loc) LIKE LOWER('%&p_location%')
    ORDER BY  dname
    /
    UNDEFINE p_location
    COLUMN dname CLEAR
    COLUMN ename CLEAR
    COLUMN hiredate CLEAR
    COLUMN sal CLEAR
    COLUMN asal CLEAR
    CLEAR BREAK
    SET VERIFY ON
    SET FEEDBACK ON
    SET ECHO  ON
  SQL> START p10q6.sql
```

Corrigé de l'Exercice 10 (suite)

7. Produisez un état semblable à celui ci-après. N'oubliez pas de mettre en forme les données, puis enregistrez votre travail dans un fichier nommé *p10q7.sql*. Le fichier *p10_sel.sql* contient l'ordre SELECT sur lequel vous pouvez vous baser pour créer votre état.

```
set echo off
set feedback off
col today format a12 noprint
col dep format 9999 heading 'Dept'
col job format A9 heading 'Job'
col ename format A7 heading 'Name'
col sal format B99,999.99 heading 'Monthly|Salary'
col comm format 9,990.99 heading 'Annual|Comm'
col totsal format 999,999.99 heading 'Total'



select to_char(sysdate,'mm/dd/yy')today, d.deptno dep, job,
       ename,
sal, comm, sal * 12 + nvl(comm,0) totsal
from emp e, dept d
where e.deptno = d.deptno
order by d.deptno,job
/



clear columns
set feedback on
set echo on
```

Corrigé de l'Exercice 10 (suite)

8. Améliorez la présentation de l'état que vous avez créé précédemment pour obtenir un état semblable à celui ci-après.

```
set echo off
set feedback off
set null ''
col today noprint new_value my_date
col dep format 9999 heading 'Dept'
col job format A9 heading 'Job'
col ename format A7 heading 'Name'
col sal format B99,999.99 heading 'Monthly|Salary'
col commi format 9,990.99 heading 'Annual|Comm'
col totsal format 999,999.99 heading 'Total'

ttitle left my_date center 'Employee Report' -
right 'Page:'  format 99 sql.pno skip 2
btitle center 'Company Confidential'

break on report on dep skip 2 on job
compute sum label total  of sal commi totsal on dep report

select to_char(sysdate,'mm/dd/yy')today, d.deptno dep,
job,              ename,
sal, nvl(comm,0) commi, sal * 12 + nvl(comm,0) totsal
from emp e, dept d
where e.deptno = d.deptno
order by d.deptno,job
/

ttitle off
btitle off
clear breaks
clear computes
clear columns
set feedback on
set echo on
```

Corrigé de l'Exercice 11

1. Examinez les exemples suivants. Sont-ils le résultat d'une requête hiérarchique ? Expliquez pourquoi oui ou pourquoi non.

Exemple 1 : Requête non hiérarchique. L'état est trié en fonction de la colonne SAL.

EMPNO	ENAME	MGR	SAL	DEPTNO
7839	KING		5000	10
7902	FORD	7566	3000	20
7788	SCOTT	7566	3000	20
7566	JONES	7839	2975	20
7698	BLAKE	7839	2850	30
7782	CLARK	7839	2450	10
7499	ALLEN	7698	1600	30

Exemple 2 : Requête non hiérarchique impliquant deux tables.

EMPNO	MGR	DNAME	LOC
7698	7839	SALES	CHICAGO
7782	7839	ACCOUNTING	NEW YORK
7566	7839	RESEARCH	DALLAS

Exemple 3 : Oui, cette requête est hiérarchique, lacolonne RANK représente la valeur de la pseudo-colonne LEVEL.

RANK	ENAME	MGR
1	KING	
2	BLAKE	7839
3	MARTIN	7698
3	ALLEN	7698
3	TURNER	7698
3	JAMES	7698
3	WARD	7698

Corrigé de l'Exercice 11 (suite)

2. Créez un état représentant l'organigramme du département de Jones. Imprimez les noms des employés, leur salaire et leur numéro de département.

```
SQL>SELECT ename,sal deptno
 2 FROM emp
 3 CONNECT BY PRIOR empno = mgr
 4 START WITH ename = 'JONES'
```

3. Créez un état dans lequel figurent les noms de tous les responsables pour lesquels travaille Adams.

```
SQL>SELECT ename
 2 FROM emp
 3 WHERE ename != 'ADAMS'
 4 CONNECT BY PRIOR mgr = empno
 5 START WITH ename = 'ADAMS';
```

4. Créez un état représentant la hiérarchie des dirigeants par une indentation. Affichez les noms des employés, le numéro de leur département ainsi que le numéro de leur responsable. Commencez par l'employé ayant le grade le plus élevé.

```
SQL>COL name FORMAT A20
SQL>SELECT LPAD(' ',3*LEVEL-3) || ename name,mgr,deptno
 2 FROM emp
 3 CONNECT BY PRIOR empno = mgr
 4 START WITH mgr IS NULL;
```

Corrigé de l'Exercice 11 (suite)

S'il vous reste encore du temps, effectuez l'exercice suivant :

5. Créez l'organigramme d'une société représentant la hiérarchie des dirigeants. Commencez par la personne ayant le grade le plus élevé et excluez tous les employés occupant le poste ANALYST, ainsi que le branche de CLARK.

```
SQL>SELECT ename,empno,mgr
  2 FROM emp
  3 WHERE job !='ANALYST'
  4 CONNECT BY PRIOR empno = mgr
  5 AND ename != 'CLARK'
  6 START WITH mgr IS NULL;
```

Corrigé de l'Exercice 12

Insérez des données dans la table MY_EMPLOYEE.

1. Exécutez le script *lab12_1.sql* pour créer la table MY_EMPLOYEE qui va servir pour cette série d'exercices.

```
SQL> START lab12_1.sql
```

2. Affichez la structure de la table MY_EMPLOYEE pour trouver les noms de colonnes.

```
SQL> DESCRIBE my_employee
```

3. Ajoutez la première ligne de données du tableau ci-dessous dans la table MY_EMPLOYEE. N'énumérez pas les colonnes dans la clause INSERT.

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	Patel	Ralph	rpatel	795
2	Dancs	Betty	bdancs	860
3	Biri	Ben	bbiri	1100
4	Newman	Chad	cnewman	750
5	Ropeburn	Audry	aropebur	1550

```
SQL> INSERT INTO my_employee
    2  VALUES (1, 'Patel', 'Ralph', 'rpatel', 795);
```

4. Continuez à remplir la table MY_EMPLOYEE en insérant la seconde ligne des données ci-dessus. Cette fois, mentionnez explicitement les colonnes dans la clause INSERT.

```
SQL> INSERT INTO my_employee (id, last_name, first_name,
    2                         userid, salary)
    3  VALUES (2, 'Dancs', 'Betty', 'bdancs', 860);
```

5. Vérifiez vos ajouts.

```
SQL> SELECT   *
  2  FROM     my_employee;
```

Corrigé de l'Exercice 12 (suite)

6. Créez un script nommé *p12q6.sql* pour charger les lignes dans la table MY_EMPLOYEE en mode interactif. Demandez à l'utilisateur de saisir le prénom (first name), le nom (last-name) et le salaire de chaque employé. Concaténez la première lettre du prénom et les sept premières lettres du nom pour former l'ID utilisateur.

```
SET ECHO OFF
SET VERIFY OFF
ACCEPT p_first_name
PROMPT 'Please enter the employee's first name: '
ACCEPT p_last_name
PROMPT 'Please enter the employee's last name: '
ACCEPT p_id
PROMPT 'Please enter the employee number: '
ACCEPT p_salary PROMPT 'Please enter the employee's
salary: '
INSERT INTO   my_employee
VALUES        (&p_id, '&p_last_name', '&p_first_name',
              substr('&p_first_name', 1, 1) ||
              substr('&p_last_name', 1, 7), &p_salary)
/
SET VERIFY ON
SET ECHO ON;
```

7. Insérez les deux lignes de données d'exemples suivantes dans la table en exécutant le script que vous avez créé.

```
SQL> START p12q6.sql
SQL> START p12q6.sql
```

8. Vérifiez vos ajouts dans la table.

```
SQL> SELECT   *
  2  FROM     my_employee;
```

9. Validez vos ajouts.

```
SQL> COMMIT;
```

Corrigé de l'Exercice 12 (suite)

Mettez à jour et supprimez des données de la table MY_EMPLOYEE.

10. Remplacez le nom de l'employé 3 par Drexler.

```
SQL> UPDATE my_employee
  2  SET    last_name = 'Drexler'
  3  WHERE  id = 3;
```

11. Saisissez un salaire de 1000 pour tous les employés ayant un salaire inférieur à 900.

```
SQL> UPDATE my_employee
  2  SET    salary = 1000
  3  WHERE  salary < 900;
```

12. Vérifiez vos modifications.

```
SQL> SELECT last_name, salary
  2  FROM   my_employee;
```

13. Supprimez Betty Dancs de la table MY_EMPLOYEE.

```
SQL> DELETE
  2  FROM   my_employee
  3  WHERE  last_name = 'Dancs'
  4  AND    first_name = 'Betty';
```

14. Vérifiez vos modifications.

```
SQL> SELECT *
  2  FROM   my_employee;
```

15. Validez toutes les modifications en instance.

    Contrôlez la transaction de données effectuée dans les tables MY_EMPLOYEE.

```
SQL> COMMIT;
```

16. Insérez la dernière ligne des donnés d'exemple dans la table en exécutant le script que vous avez créé à l'étape 6.

```
SQL> START p12q6.sql
```

Corrigé de l'Exercice 12 (suite)

17. Vérifiez l'ajout.

```
SQL> SELECT *
  2  FROM   my_employee;
```

18. Définissez une étiquette intermédiaire dans le traitement de la transaction.

```
SQL> SAVEPOINT a;
```

19. Videz entièrement la table.

```
SQL> DELETE
  2  FROM   my_employee;
```

20. Vérifiez que la table est vide.

```
SQL> SELECT *
  2  FROM   my_employee;
```

21. Rejetez la dernière opération DELETE sans annuler l'opération INSERT précédente.

```
SQL> ROLLBACK TO SAVEPOINT a;
```

22. Vérifiez que la dernière ligne est restée intacte.

```
SQL> SELECT *
  2  FROM   my_employee;
```

23. Rendez les ajouts définitifs.

```
SQL> COMMIT;
```

Corrigé de l'Exercice 13

1. Créez la table DEPARTMENT d'après le tableau suivant. Saisissez la syntaxe dans un script que vous nommerez *p13q1.sql*, puis exécutez ce script pour créer la table. Vérifiez la création de la table.

Nom de la colonne	Id	Name
Type de clé		
Nulls/Unique		
Table FK		
Colonne FK		
Type de données	Number	Varchar2
Longueur	7	25

```
SQL> EDIT p13q1.sql
     CREATE TABLE department
              (id NUMBER(7),
              name VARCHAR2(25))
     /
SQL> START p13q1.sql
SQL> DESCRIBE department;
```

2. Remplissez la table DEPARTMENT avec les données de la table DEPT. N'utilisez que les colonnes dont vous avez besoin.

```
SQL> INSERT INTO department
  2  SELECT     deptno, dname
  3  FROM       dept;
```

3. Créez la table EMPLOYEE d'après le tableau suivant. Saisissez la syntaxe dans un script que vous nommerez *p13q3.sql*, puis exécutez ce script pour créer la table. Vérifiez la création de la table.

Nom de la colonne	ID	LAST_NAME	FIRST_NAME	DEPT_ID
Type de clé				
Nulls/Unique				
Table FK				
Colonne FK				
Type de données	Number	Varchar2	Varchar2	Number
Longueur	7	25	25	7

Corrigé de l'Exercice 13 (suite)

```
CREATE TABLE  employee
(id           NUMBER(7),
last_name     VARCHAR2(25),
first_name    VARCHAR2(25),
dept_id       NUMBER(7))
/
SQL> START p13q3.sql
SQL> DESCRIBE employee;
```

4. Modifiez la table EMPLOYEE pour pouvoir allonger les noms de famille des employés. Vérifiez votre modification.

```
SQL> ALTER TABLE employee
  2  MODIFY (last_name   VARCHAR2(50));
SQL> DESCRIBE employee
```

5. Vérifiez que les tables DEPARTMENT et EMPLOYEE sont bien enregistrées dans le dictionnaire de données (*partie* : USER_TABLES).

```
SQL> SELECT       table_name
  2  FROM         user_tables
  3  WHERE        table_name IN ('DEPARTMENT', 'EMPLOYEE');
```

6. Créez la table EMPLOYEE2 sur la base de la structure et des données de la table EMP mais n'incluez que les colonnes EMPNO, ENAME et DEPTNO. Nommez les colonnes de votre nouvelle table respectivement ID,LAST_NAME et DEPT_ID.

```
SQL> CREATE TABLE  employee2 AS
  2  SELECT       empno id, ename last_name, deptno dept_id
  3  FROM         emp;
```

7. Supprimez la table EMPLOYEE.

```
SQL> DROP TABLE   employee;
```

8. Renommez la table EMPLOYEE2 en EMPLOYEE.

```
SQL> RENAME employee2 TO employee;
```

Corrigé de l'Exercice 13 (suite)

9. Ajoutez un commentaire aux définitions des tables DEPARTMENT et EMPLOYEE pour décrire chaque table. Vérifiez vos ajouts dans le dictionnaire de données.

```
SQL> COMMENT ON TABLE employee IS 'Employee Information';
SQL> COMMENT ON TABLE department IS 'Department Information';
SQL> COLUMN table_name FORMAT A15
SQL> COLUMN table_type FORMAT A10
SQL> COLUMN comments FORMAT A40
SQL> SELECT *
  2  FROM   user_tab_comments
  3  WHERE  table_name = 'DEPARTMENT'
  4  OR     table_name = 'EMPLOYEE';
```

Corrigé de l'Exercice 14

1. Ajoutez une contrainte PRIMARY KEY de niveau table dans la table EMPLOYEE en utilisant la colonne ID. La contrainte doit être activée dès sa création.

```
SQL> ALTER TABLE         employee
  2  ADD CONSTRAINT      employee_id_pk PRIMARY KEY (id);
```

2. Créez une contrainte PRIMARY KEY dans la table DEPARTMENT en utilisant la colonne ID. La contrainte doit être activée dès sa création.

```
SQL> ALTER TABLE         department
  2  ADD CONSTRAINT      department_id_pk PRIMARY KEY(id);
```

3. Ajoutez une clé étrangère dans la table EMPLOYEE qui permettra de contrôler que l'employé n'est pas associé à un département inexistant.

```
SQL> ALTER TABLE employee
  2  ADD CONSTRAINT employee_dept_id_fk FOREIGN KEY
                    (dept_id)
  3      REFERENCES department(id);
```

4. Vérifiez que les contraintes ont été ajoutées en les recherchant dans USER_CONSTRAINTS.
   Prenez note des types et des noms des contraintes. Enregistrez votre ordre dans un fichier appelé *p14q4.sql*.

```
SQL> SELECT   constraint_name, constraint_type
  2  FROM     user_constraints
  3  WHERE    table_name IN ('EMPLOYEE', 'DEPARTMENT');
SQL> SAVE p14q4.sql;
```

5. Recherchez les noms et types d'objets dans la vue USER_OBJECTS du dictionnaire de données correspondant aux tables EMPLOYEE et DEPARTMENT. Vous pouvez mettre les colonnes en forme pour qu'elles soient plus lisibles. Remarquez que pour chaque nouvelle table un nouvel index a été créé.

```
SQL> COLUMN   object_name  FORMAT A30
SQL> COLUMN   object_type  FORMAT A30
SQL> SELECT   object_name, object_type
  2  FROM     user_objects
  3  WHERE    object_name LIKE 'EMPLOYEE%'
  4  OR       object_name LIKE 'DEPARTMENT%';
```

Si vous avez le temps, faites l'exercice suivant :

6. Modifiez la table EMPLOYEE. Ajoutez une colonne SALARY dont le type de donnée est NUMBER , avec une précision 7.

```
SQL> ALTER TABLE employee
  2  ADD (salary NUMBER(7));
```

Corrigé de l'Exercice 15

1. Créez la vue EMP_VU à partir de la table EMP contenant des numéros et des noms d'employés, avec leur numéro de département. Modifiez l'en-tête de la colonne des noms d'employés en la nommant EMPLOYEE.

```
SQL> CREATE VIEW emp_vu AS
  2  SELECT      empno, ename employee, deptno
  3  FROM        emp;
```

2. Affichez le contenu de la vue EMP_VU.

```
SQL> SELECT      *
  2  FROM        emp_vu;
```

3. Sélectionnez le nom de la vue (view_name) et le texte correspondant dans la table USER_VIEWS du dictionnaire de données.

```
SQL> COLUMN view_name FORMAT A30
SQL> COLUMN text FORMAT A50
SQL> SELECT      view_name, text
  2  FROM        user_views;
```

4. A partir de votre vue EMP_VU, faites une requête pour afficher tous les noms des employés et le numéro de leur département.

```
SQL> SELECT      employee, deptno
  2  FROM        emp_vu;
```

5. Créez la vue DEPT20 avec les numéros et les noms de tous les employés du département 20. Nommez les colonnes de la vue EMPLOYEE_ID, EMPLOYEE et DEPARTMENT_ID, respectivement. Cette vue ne doit pas autoriser l'affectation d'un employé à un autre département.

```
SQL> CREATE VIEW dept20 AS
  2  SELECT      empno employee_id, ename employee,
  3                     deptno department_id
  4  FROM        emp
  5  WHERE       deptno = 20
  6  WITH CHECK OPTION CONSTRAINT emp_dept_20;
```

Corrigé de l'Exercice 15 (suite)

6. Affichez la structure et le contenu de la vue DEPT20.

```
SQL> DESCRIBE dept20
SQL> SELECT *
  2  FROM   dept20;
```

7. Essayez d'affecter l'employé Smith au département 30.

```
SQL> UPDATE dept20
  2  SET    department_id = 30
  3  WHERE  employee = 'SMITH';
```

S'il vous reste du temps, effectuez les exercices suivants :

8. Créez la vue SALARY_VU de façon à afficher le nom de tous les employés, le nom de leur département, leur salaire et leur barème de salaire. Nommez les colonnes respectivement Employee,Department, Salary et Grade.

```
SQL> CREATE VIEW salary_vu AS
  2  SELECT ename employee, dname department,
  3         sal salary, grade
  4  FROM   emp e, dept d, salgrade s
  5  WHERE  e.deptno = d.deptno
  6  AND    e.sal between s.losal and s.hisal;
```

Corrigé de l'Exercice 16

1. Créez une séquence pour l'utiliser avec la clé primaire de la table DEPARTMENT. Cette séquence doit commencer avec le numéro 60 et avoir une valeur maximale de 200.

Attribuez la valeur 10 à l'incrément et nommez la séquence DEPT_ID_SEQ.

```
SQL> CREATE SEQUENCE dept_id_seq
  2  START WITH 60
  3  INCREMENT BY 10
  4  MAXVALUE 200;
```

2. Ecrivez un script pour afficher les informations suivantes relatives à vos séquences : nom de la séquence, valeur maximale, taille de l'incrément et dernier numéro de séquence. Nommez le script *p16q2.sql* puis exécutez-le.

```
SQL> EDIT p16q2.sql
     SELECT    sequence_name, max_value,
               increment_by, last_number
     FROM      user_sequences
     /
SQL> START p16q2.sql
```

3. Ecrivez un script interactif pour insérer une ligne dans la table DEPARTMENT. Nommez votre script *p16q3.sql*. Veillez à utiliser la séquence que vous avez créée pour la colonne ID.
Créez une invite personnalisée pour la saisie du nom du département. Exécutez votre script. Ajoutez les deux départements Education et Administration, puis validez -les.

```
SQL> EDIT p16q3.sql
     SET ECHO OFF
     SET VERIFY OFF
     ACCEPT name PROMPT 'Please enter the department name: '
     INSERT INTO         department (id, name)
     VALUES    (dept_id_seq.NEXTVAL, '&name')
     /
     SET VERIFY ON
     SET ECHO ON
SQL> START p16q3.sql
SQL> SELECT   *
  2  FROM     department;
```

4. Créez un index non unique dans la colonne FOREIGN KEY de la table EMPLOYEE.

```
SQL> CREATE INDEX employee_dept_id_idx ON employee
                  (dept_id);
```

Corrigé de l'Exercice 16 (suite)

5. Affichez les index et l'unicité existant dans le dictionnaire de données pour la table EMPLOYEE. Enregistrez l'ordre dans le script *p16q5.sql*.

```
SQL> SELECT index_name, table_name, uniqueness
  2  FROM   user_indexes
  3  WHERE  table_name = 'EMPLOYEE';
SQL> SAVE p16q5.sql;
```

Corrigé de l'Exercice 17

1. Quel privilège doit avoir un utilisateur pour se connecter à Oracle8 Server ? S'agit-il d'un privilège système ou objet ?

   Le privilège système CREATE SESSION.

2. Quel privilège doit avoir un utilisateur pour créer des tables ?

   Le privilège CREATE TABLE.

3. Si vous créez une table, qui peut transmettre à d'autres utilisateurs les privilèges liés à votre table ?

   Vous-même, ou tout autre utilisateur, pouvez transmettre ces privilèges à l'aide de l'option WITH GRANT OPTION.

4. Vous êtes administrateur de base de données et vous devez créer un grand nombre d'utilisateurs qui exigent les mêmes privilèges système. Comment pouvez-vous simplifier cette tâche ?

   Créez un rôle contenant les privilèges système, puis accordez ce rôle aux utilisateurs.

5. Quel ordre pouvez-vous utiliser pour modifier votre mot de passe ?

   L'ordre ALTER USER.

6. Autorisez un autre utilisateur à lire votre table DEPT. Demandez à un utilisateur de vous accorder un droit de lecture sur sa table DEPT.

```
Team 2 executes the GRANT statement.
SQL> GRANT    select
  2  ON       dept
  3  TO       <user1>;
Team 1 executes the GRANT statement.
SQL> GRANT    select
  2  ON       dept
  3  TO       <user2>;
     WHERE    user1 is the name of team 1 and user2 is the
              name of team 2.
```

7. Interrogez toutes les lignes de votre table DEPT.

```
SQL> SELECT   *
  2  FROM     dept;
```

Corrigé de l'Exercice 17 (suite)

8. Ajoutez une nouvelle ligne à votre table DEPT. L'équipe 1 doit ajouter le département Education portant le numéro 50, et l'équipe 2 le département Administration portant le numéro 50. Enregistrez vos modifications.

```
Team 1 executes this INSERT statement.
SQL> INSERT INTO dept(deptno, dname)
  2  VALUES (50, 'Education');
SQL> COMMIT;
Team 2 executes this INSERT statement.
SQL> INSERT INTO dept(deptno, dname)
  2  VALUES (50, 'Administration');
SQL> COMMIT;
```

9. Créez un synonyme pour la table DEPT de l'autre équipe.

```
Team 1 creates a synonym named team2.
SQL> CREATE SYNONYM  team2
  2  FOR <user2>.DEPT;
Team 2 creates a synonym named team1.
SQL> CREATE SYNONYM  team1
  2  FOR <user1>.DEPT;
```

10. Interrogez toutes les lignes de la table DEPT de l'autre équipe en utilisant votre synonyme.

```
Team 1 executes this SELECT statement.
SQL> SELECT *
  2  FROM   team2;
Team 2 executes this SELECT statement.
SQL> SELECT *
  2  FROM   team1;
```

Corrigé de l'Exercice 17 (suite)

11. Interrogez le dictionnaire de données USER_TABLES pour afficher les informations relatives aux tables qui vous appartiennent.

```
SQL> SELECT table_name
  2  FROM   user_tables;
```

12. Interrogez la vue ALL_TABLES du dictionnaire de données pour afficher les informations relatives à toutes les tables auxquelles vous pouvez accéder. Excluez les tables qui vous appartiennent.

```
SQL> SELECT table_name, owner
  2  FROM   all_tables
  3  WHERE  owner != <your account>;
```

13. Retirez le privilège SELECT à l'autre équipe.

```
Team 1 revokes the privilege.
SQL> REVOKE select
  2  ON     dept
  3  FROM   user2;
Team 1 revokes the privilege.
SQL> REVOKE select
  2  ON     dept
  3  FROM   user1;
```

Corrigé de l'Exercice 18

1. Ecrivez un script pour décrire et sélectionner les données de vos tables. Utilisez CHR(10) pour générer un saut de ligne dans votre état.

```
SET TERMOUT OFF ECHO OFF FEEDBACK OFF
SET PAGESIZE 0

SPOOL my_file1.sql

SELECT 'DESC ' || TABLE_NAME || CHR(10) ||
'SELECT * FROM ' || TABLE_NAME || ';'
FROM USER_TABLES
/

SPOOL OFF

SET TERMOUT ON ECHO ON FEEDBACK ON
SET PAGESIZE 24
```

2. A l'aide de SQL, générez un script SQL pour retirer tous les privilèges à un utilisateur. Utilisez les vues USER_TAB_PRIVS_MADE et USER_COL_PRIVS_MADE du dictionnaire de données.
   a. Exécutez le script *privs.sql* pour accorder des privilèges à l'utilisateur SYSTEM.
   b. Interrogez les vues du dictionnaire de données pour vérifier ces privilèges.
   c. Créez un script pour les retirer.

```
SELECT   *
FROM USER_TAB_PRIVS_MADE
WHERE GRANTEE = 'SYSTEM';

SELECT   *
FROM USER_COL_PRIVS_MADE
WHERE GRANTEE = 'SYSTEM';
```

Corrigé de l'Exercice 18 (suite)

```
SET VERIFY OFF
SET TERMOUT OFF ECHO OFF FEEDBACK OFF
SET PAGESIZE 0

SPOOL my_file2.sql

SELECT 'REVOKE '|| privilege || ' ON ' ||
table_name || '  FROM SYSTEM;'
FROM USER_TAB_PRIVS_MADE
WHERE GRANTEE = 'SYSTEM'
/
SELECT DISTINCT 'REVOKE '|| privilege || ' ON ' ||
table_name || '  FROM SYSTEM;'
FROM USER_COL_PRIVS_MADE
WHERE GRANTEE = 'SYSTEM'
/

SPOOL OFF

SET VERIFY ON
SET TERMOUT ON ECHO ON FEEDBACK ON
SET PAGESIZE 24
```

Corrigé de l'Exercice 19

1. Créez les tables en vous basant sur les tableaux ci-dessous. Choisissez les types de données appropriés et n'oubliez pas d'ajouter des contraintes d'intégrité.

   a. Nom de la table : MEMBER

Colonne	MEMBER_ID	LAST_NAME	FIRST_NAME	ADDRESS	CITY	PHONE	JOIN_DATE
Type de clé	PK						
Null/ Unique	NN,U	NN					NN
Valeur par défaut							Date Système
Type de données	Number	Varchar2	Varchar2	Varchar2	Varchar2	Varchar2	Date
Longueur	10	25	25	100	30	15	

```
CREATE TABLE member
(member_id     NUMBER(10)
                  CONSTRAINT member_member_id_pk PRIMARY
KEY,
last_name      VARCHAR2(25)
                  CONSTRAINT member_last_name_nn NOT NULL,
first_name     VARCHAR2(25),
address        VARCHAR2(100),
city           VARCHAR2(30),
phone          VARCHAR2(15),
join_date      DATE DEFAULT SYSDATE
               CONSTRAINT member_join_date_nn NOT NULL);
```

Corrigé de l'Exercice 19 (suite)

b. Nom de la table : TITLE

Colonne	TITLE_ID	TITLE	DESCRIPTION	RATING	CATEGORY	RELEASE_DATE
Type de clé	PK					
Null/ Unique	NN,U	NN	NN			
Check				G, PG, R, NC17, NR	DRAMA, COMEDY, ACTION, CHILD, SCIFI, DOCUMENTARY	
Type de données	Number	Varchar2	Varchar2	Varchar2	Varchar2	Date
Longueur	10	60	400	4	20	

```
CREATE TABLE title
(title_id    NUMBER(10)
   CONSTRAINT title_title_id_pk PRIMARY KEY,
 title       VARCHAR2(60)
   CONSTRAINT title_title_nn NOT NULL,
 description VARCHAR2(400)
   CONSTRAINT title_description_nn NOT NULL,
 rating      VARCHAR2(4)
   CONSTRAINT title_rating_ck CHECK
       (rating IN ('G', 'PG', 'R', 'NC17', 'NR')),
 category    VARCHAR2(20),
   CONSTRAINT title_category_ck CHECK
       (category IN ('DRAMA', 'COMEDY', 'ACTION',
        'CHILD', 'SCIFI', 'DOCUMENTARY')),
release_date DATE);
```

Corrigé de l'Exercice 19 (suite)

c. Nom de la table : TITLE_COPY

Colonne	COPY_ID	TITLE_ID	STATUS
Type de clé	PK	PK,FK	
Null/ Unique	NN,U	NN,U	NN
Table réf. FK		TITLE	
Col. Réf. FK		TITLE_ID	
Check			AVAILABLE, DESTROYED, RENTED, RESERVED
Type de données	Number	Number	Varchar2
Longueur	10	10	15

```
      CREATE TABLE title_copy
      (copy_id      NUMBER(10),
       title_id     NUMBER(10)
      CONSTRAINT title_copy_title_if_fk REFERENCES
title(title_id),
       status       VARCHAR2(15)
                    CONSTRAINT title_copy_status_nn NOT NULL
                    CONSTRAINT title_copy_status_ck CHECK
                    (status IN ('AVAILABLE', 'DESTROYED',
                    'RENTED', 'RESERVED')),
      CONSTRAINT title_copy_copy_id_title_id_pk
                    PRIMARY KEY (copy_id, title_id));
```

Corrigé de l'Exercice 19 (suite)

d. Nom de la table : RENTAL

Colonne	BOOK_DATE	COPY_ID	TITLE_ID	MEMBER_ID	ACT_RET_DATE	EXP_RET_DATE
Type de clé	PK	PK,FK1	PK,FK1	PK,FK2		
Valeur par défaut	Date Système					Date Système + 2 jours
Table réf. FK		title_copy	title_copy	member		
Col. réf. FK		copy_id	title_id	member_id		
Type de données	Date	Number	Number	Number	Date	Date
Longueur		10	10	10		

```
CREATE TABLE rental
(book_date  DATE DEFAULT SYSDATE,
copy_id          NUMBER(10),
title_id    NUMBER(10),
member_id   NUMBER(10)
  CONSTRAINT rental_member_id_fk
  REFERENCES member(member_id),
act_ret_date     DATE,
exp_ret_date     DATE DEFAULT SYSDATE + 2,
CONSTRAINT rental_book_date_copy_title_pk PRIMARY KEY
       (book_date, member_id, copy_id, title_id),
CONSTRAINT rental_copy_id_title_id_fk FOREIGN KEY
       (copy_id, title_id)
       REFERENCES title_copy(copy_id, title_id));
```

Corrigé de l'Exercice 19 (suite)

e. Nom de la table : RESERVATION

Colonne	RES_DATE	MEMBER_ID	TITLE_ID
Type de clé	PK	PK,FK1	PK,FK2
Null/ Unique	NN,U	NN,U	NN
Table réf. FK		MEMBER	TITLE
Colonne réf. FK		member_id	title_id
Type de données	Date	Number	Number
Longueur		10	10

```
CREATE TABLE reservation
(res_date   DATE,
 member_id  NUMBER(10)
       CONSTRAINT reservation_member_id
       REFERENCES member(member_id),
 title_id   NUMBER(10)
       CONSTRAINT reservation_title_id
       REFERENCES title(title_id),
 CONSTRAINT reservation_resdate_mem_tit_pk PRIMARY KEY
       (res_date, member_id, title_id));
```

Corrigé de l'Exercice 19 (suite)

2. Vérifiez que les tables et les contraintes ont été créées correctement dans le dictionnaire de données.

```
SQL> SELECT   table_name
  2  FROM     user_tables
  3  WHERE    table_name IN ('MEMBER', 'TITLE', 'TITLE_COPY',
  4                         'RENTAL', 'RESERVATION');
```

```
SQL> COLUMN constraint_name FORMAT A20
SQL> COLUMN table_name FORMAT  A15
SQL> SELECT   constraint_name, constraint_type,
  2           table_name
  3   FROM    user_constraints
  4   WHERE   table_name IN ('MEMBER', 'TITLE', 'TITLE_COPY',
  5                       'RENTAL', 'RESERVATION');
```

3. Créez des séquences pour identifier de façon unique chaque ligne des tables MEMBER et TITLE.

   a. Pour la table MEMBER, le premier numéro de membre doit être 101. N'autorisez pas la mise en mémoire cache des valeurs et nommez la séquence MEMBER_ID_SEQ.

```
SQL> CREATE SEQUENCE member_id_seq
  2  START WITH 101
  3  NOCACHE;
```

   b. Pour la table TITLE, le premier numéro de titre doit être 92. N'autorisez pas la mise en mémoire cache des valeurs et nommez la séquence TITLE_ID_SEQ.

```
SQL> CREATE SEQUENCE title_id_seq
  2  START WITH 92
  3  NOCACHE;
```

   c. Vérifiez que les séquences existent dans le dictionnaire de données.

```
SQL> SELECT   sequence_name, increment_by, last_number
  2  FROM     user_sequences
  3  WHERE    sequence_name IN ('MEMBER_ID_SEQ',
                               'TITLE_ID_SEQ');
```

4. Ajoutez des données dans les tables. Créez un script pour chaque ensemble de données à ajouter.

   a. Ajoutez des titres de films dans la table TITLE. Ecrivez un script pour saisir les informations relatives aux films et enregistrez le script sous le nom *p19q4a.sql.* Utilisez les séquences pour identifier chaque titre de façon unique. N'oubliez pas que les simples quotes dans un champ de caractères sont soumis à un traitement spécial. Vérifiez vos ajouts.

```
SQL> EDIT p19q4a.sql
SET ECHO OFF
INSERT INTO title(title_id, title, description, rating,
                  category, release_date)
VALUES (title_id_seq.NEXTVAL, 'Willie and Christmas Too',
       'All of Willie''s friends made a Christmas list for
        Santa, but Willie has yet to add his own wish list.',
       'G', 'CHILD', '05-OCT-95')
/
INSERT INTO title(title_id , title, description, rating,
                  category, release_date)
VALUES (title_id_seq.NEXTVAL, 'Alien Again', 'Yet another
       installment of science fiction history.  Can the
       heroine save the planet from the alien life form?',
       'R', 'SCIFI', '19-MAY-95')
/
INSERT INTO title(title_id, title, description, rating,
                  category, release_date)
VALUES (title_id_seq.NEXTVAL, 'The Glob', 'A meteor crashes
       near a small American town and unleashed carnivorous
       goo in this classic.', 'NR', 'SCIFI', '12-AUG-95')
/
INSERT INTO title(title_id, title, description, rating,
                  category, release_date)
VALUES (title_id_seq.NEXTVAL, 'My Day Off', 'With a little
       luck and a lot ingenuity, a teenager skips school for
       a day in New York.', 'PG', 'COMEDY', '12-JUL-95')
/
...
COMMIT
/
SET ECHO ON

SQL> SELECT   title
  2  FROM     title;
```

Corrigé de l'Exercice 19 (suite)

Title	Description	Rating	Category	Release_date
Willie and Christmas Too	Tous les amis de Willie ont fait leur commande de cadeaux au Père Noël, mais Willie n'a pas encore rédigé sa liste.	G	CHILD	05-OCT-95
Alien Again	Une nouvelle histoire de science-fiction. L'héroïne pourra-t-elle sauver la planète des extra-terrestres ?	R	SCIFI	19-MAY-95
The Glob	Un météore s'écrase près d'une petite ville américaine, ce qui déchaîne une entité cannibale dans ce classique kitsch.	NR	SCIFI	12-AUG-95
My Day Off	Avec un peu de chance et beaucoup d'ingénuité, un adolescent new-yorkais décide de faire une journée d'école buissonnière.	PG	COMEDY	12-JUL-95
Miracles on Ice	Une fillette de six ans doute de la réalité du Père Noël, mais elle découvre que les miracles existent encore.	PG	DRAMA	12-SEP-95
Soda Gang	Un jeune couple doit affronter un gang de féroces trafiquants de drogue dont il a découvert le repaire.	NR	ACTION	01-JUN-95

Corrigé de l'Exercice 19 (suite)

b. Ajoutez des données dans la table MEMBER. Ecrivez le script *p19q4b.sql* pour permettre aux utilisateurs de saisir des informations, puis exécutez-le. Vérifiez que vous utilisez bien la séquence destinée à ajouter des numéros de membres.

First Name	Last Name	Address	State	Phone	Join Date
Carmen	Velasquez	283 King Street	Seattle	206-899-6666	08-MAR-90
LaDoris	Ngao	5 Modrany	Bratislava	586-355-8882	08-MAR-90
Midori	Nagayama	68 Via Centrale	Sao Paolo	254-852-5764	17-JUN-91
Mark	Lewis	6921 King Way	Lagos	63-559-7777	07-APR-90
Audry	Ropeburn	86 Chu Street	Hong Kong	41-559-87	18-JAN-91
Molly	Urguhart	3035 Laurier	Quebec	418-542-9988	18-JAN-91

```
SQL> EDIT p19q4b.sql
SET ECHO OFF
SET VERIFY OFF
INSERT INTO member(member_id, first_name, last_name, address,
                   city, phone, join_date)
VALUES (member_id_seq.NEXTVAL, '&first_name', '&last_name',
                   '&address', '&city', '&phone', '&join_date')
/
COMMIT
/
SET VERIFY ON
SET ECHO ON
SQL> START p19q4b.sql
```

Corrigé de l'Exercice 19 (suite)

c. Ajoutez les copies de films suivantes dans la table TITLE_COPY :

Title	Copy Number	Status
Willie and Christmas Too	1	Available
Alien	1	Available
	2	Rented
The Glob	1	Available
My Day Off	1	Available
	2	Available
	3	Rented
Miracles on Ice	1	Available
Soda Gang	1	Available

```
SQL> INSERT INTO title_copy(copy_id, title_id, status)
  2  VALUES (1, 92, 'AVAILABLE');
SQL> INSERT INTO title_copy(copy_id, title_id, status)
  2  VALUES (1, 93, 'AVAILABLE');
SQL> INSERT INTO title_copy(copy_id, title_id, status)
  2  VALUES (2, 93, 'RENTED');
SQL> INSERT INTO title_copy(copy_id, title_id, status)
  2  VALUES (1, 94, 'AVAILABLE');
SQL> INSERT INTO title_copy(copy_id, title_id, status)
  2  VALUES (1, 95, 'AVAILABLE');
SQL> INSERT INTO title_copy(copy_id, title_id,status)
  2 VALUES (2, 95, 'AVAILABLE');
 SQL> INSERT INTO title_copy(copy_id, title_id,status)
  2  VALUES (3, 95, 'RENTED');
 SQL> INSERT INTO title_copy(copy_id, title_id,status)
   2  VALUES (1, 96, 'AVAILABLE');
 SQL> INSERT INTO title_copy(copy_id, title_id,status)
   2  VALUES (1, 97, 'AVAILABLE');
```

Corrigé de l'Exercice 19 (suite)

d. Ajoutez les vidéos louées suivantes dans la table RENTAL :

Remarque : Le numéro du titre peut être différent, en fonction du numéro de séquence.

Title	Copy_number	Customer	Date_ Rented	Date_return_expected	Date_returned
92	1	101	3 days ago	1 day ago	2 days ago
93	2	101	1 day ago	1 day from now	
95	3	102	2 days ago	Today	
97	1	106	4 days ago	2 days ago	2 days ago

```
SQL> INSERT INTO rental(title_id, copy_id, member_id,
  2             book_date, exp_ret_date, act_ret_date)
  3  VALUES (92, 1, 101, sysdate-3, sysdate-1, sysdate-2);
SQL> INSERT INTO rental(title_id, copy_id, member_id,
  2             book_date, exp_ret_date, act_ret_date)
  3 VALUES  (93, 2, 101, sysdate-1, sysdate-1, NULL);
SQL> INSERT INTO rental(title_id, copy_id, member_id,
  2             book_date, exp_ret_date, act_ret_date)
  3 VALUES  (95, 3, 102, sysdate-2, sysdate, NULL);
SQL> INSERT INTO rental(title_id, copy_id, member_id,
  2             book_date, exp_ret_date,act_ret_date)
  3  VALUES (97, 1, 106, sysdate-4, sysdate-2, sysdate-2);
SQL> COMMIT;
```

Corrigé de l'Exercice 19 (suite)

5. Créez la vue TITLE_AVAIL pour afficher les titres de films et la disponibilité de chaque copie, ainsi que la date de retour prévue en cas de location. Interrogez toutes les lignes de cette vue.

```
SQL> CREATE VIEW title_avail AS
  2    SELECT   t.title, c.copy_id, c.status, r.exp_ret_date
  3    FROM     title t, title_copy c, rental r
  4    WHERE    t.title_id = c.title_id
  5    AND      c.copy_id = r.copy_id(+)
  6    AND      c.title_id = r.title_id(+);

SQL> SELECT    *
  2  FROM      title_avail;
  3  ORDER BY title, copy_id;
```

6. Modifiez les données des tables.

   a. Ajoutez un nouveau titre, "Interstellar Wars", film de science-fiction classé PG. Sa date de sortie est le 07-JUL-77 et sa description est la suivante : "Film d'action interstellaire. Les rebelles pourront-ils sauver les humains de l'Empire du Mal ?" Vérifiez que vous ajoutez bien un enregistrement du titre pour chacune des deux copies.

```
SQL> INSERT INTO title(title_id, title, description, rating,
  2            category, release_date)
  3  VALUES (title_id_seq.NEXTVAL, 'Interstellar Wars',
  4         'Futuristic interstellar action movie.  Can the
  5             rebels save the humans from the evil Empire?',
  6             'PG', 'SCIFI', '07-JUL-77');
```

   b. Entrez deux réservations : une pour Carmen Velasquez, qui souhaite louer "Interstellar Wars", et l'autre pour Mark Lewis, qui souhaite louer "Soda Gang".

```
SQL> INSERT INTO reservation (res_date, member_id, title_id)
  2  VALUES (SYSDATE, 101, 98);
SQL> INSERT INTO reservation (res_date, member_id, title_id)
  2  VALUES (SYSDATE, 101, 99);
```

Corrigé de l'Exercice 19 (suite)

c. La cliente Carmen Velasquez loue la copie 1 du film "Interstellar Wars". Supprimez sa réservation pour ce film. Enregistrez les informations relatives à la location. Autorisez la valeur par défaut pour la date de retour prévue. A l'aide de la vue que vous avez créée, vérifiez que la location a été enregistrée.

```
SQL> INSERT INTO rental(title_id, copy_id, member_id)
  2  VALUES (98, 1,101);
SQL> UPDATE    title_copy
  2  SET       status= 'RENTED'
  3  WHERE     title_id = 98
  4  AND       copy_id = 1;
SQL> DELETE
  2  FROM      reservation
  3  WHERE     member_id = 101;
SQL> SELECT    *
  2  FROM      title_avail
  3  ORDER BY  title, copy_id;
```

7. Modifiez une des tables.

a. Ajoutez une colonne PRICE à la table TITLE pour enregistrer le prix d'achat de la cassette vidéo. La largeur de cette colonne doit permettre la saisie de huit chiffres et de deux décimales. Vérifiez vos modifications.

```
SQL> ALTER TABLE title
  2  ADD (price          NUMBER(8,2));
SQL> DESCRIBE title
```

Corrigé de l'Exercice 19 (suite)

b. Créez le script *p19q7b.sql* pour mettre à jour le prix de chaque cassette vidéo selon la liste suivante. Pour cet exercice, utilisez les numéros des titres disponibles.

Title	Price
Willie and Christmas Too	25
Alien Again	35
The Glob	35
My Day Off	35
Miracles on Ice	98
Soda Gang	35
Interstellar Wars	29

```
SET ECHO OFF
SET VERIFY OFF
UPDATE        title
SET           price = &price
WHERE         title_id = &title_id
/
SET VERIFY OFF
SET ECHO OFF
SQL> START p19q7b.sql
```

c. Assurez-vous qu'à l'avenir, le prix de chaque titre sera indiqué. Vérifiez la contrainte.

```
SQL> ALTER TABLE title
  2  MODIFY (price CONSTRAINT title_price_nn NOT NULL);
SQL> SELECT   constraint_name, constraint_type,
  2           search_condition
  3  FROM     user_constraints
  4  WHERE    table_name = 'TITLE';
```

Corrigé de l'Exercice 19 (suite)

8. Créez l'état "Customer History" qui devra contenir l'historique des locations de cassettes vidéode chaque client. Indiquez le nom du client, les films loués, les dates et la durée des locations.
   Additionnez le nombre de locations de tous les clients pour la période concernée, puis enregistrez le script sous le nom de fichier *p19q8.sql*.

```
SQL> EDIT p19q8.sql
SET ECHO OFF
SET VERIFY OFF
SET PAGESIZE 30
COLUMN member    FORMAT A17
COLUMN title FORMAT A15
COLUMN book_date FORMAT A9
COLUMN duration  FORMAT 9999999
BREAK ON member  SKIP 1 ON REPORT
SELECT m.first_name||' '||m.last_name MEMBER, t.title,
       r.book_date, r.act_ret_date - r.book_date DURATION
FROM   member m, title t, rental r
WHERE  r.member_id = m.member_id
AND         r.title_id = t.title_id
ORDER BY    member
/
CLEAR BREAK
COLUMN member CLEAR
COLUMN title CLEAR
COLUMN book_date CLEAR
COLUMN duration CLEAR
SET VERIFY ON
SET PAGESIZE 24
SET ECHO ON;
```