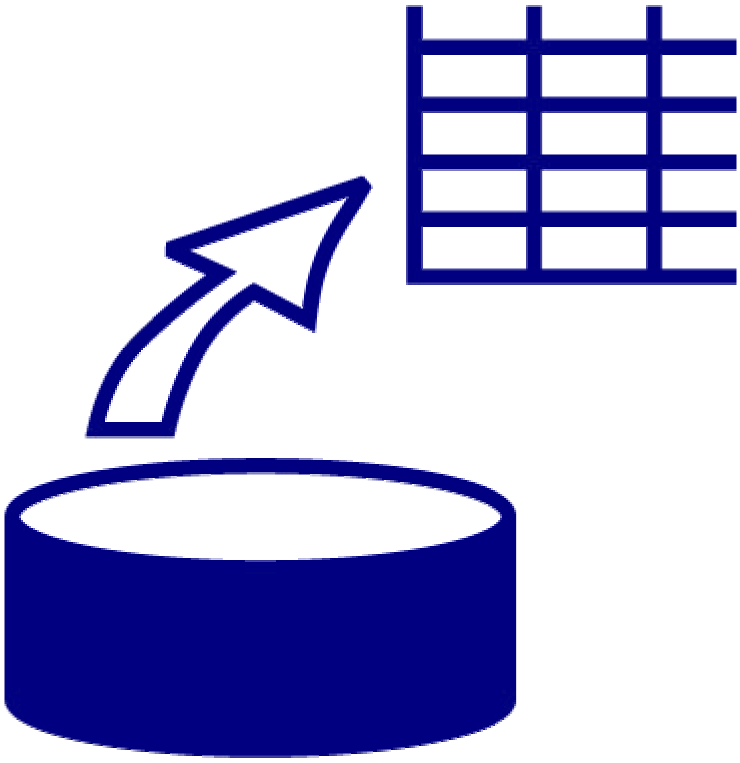
Bases de Datos

Resolución de Ejercicios Prácticos



Antonio Pérez Ruth
José Manuel Molina Camacho
Grupo S14
I.T. Informática de Sistemas
Curso 2000 – 2001
Córdoba a cinco de junio de dos mil uno

ÍNDICE

NOTA: En los ejercicios relacionados con fechas, las fechas tendrán un formato distinto al de clase (*formato inglés*), que será el *formato español*, debido a que en casa la version de Oracle que tengo es la 8.1.7.0.0 en castellano. No obstante, otros ejercicios tendrán el formato inglés.

Además, los resultados de las distintas consultas puede que sean distintos debido a que las tablas que estamos manejando ya han sido varias veces modificadas por ejercicios de temas posteriores. A lo que se añade la presencia de otro empleado más, DOE, que se crea al cargar la tabla.

TEMA 1. MANDATO SELECT BÁSICO.

1.1. Initiate a SQL*Plus session using the user ID and password provided by the instructor.

1.2. SQL*Plus commands access the database. **TRUE** / False

1.3. Will the SELECT statement execute successfully?

```
SELECT ename, job, sal Salary
FROM emp;
```

```
ENAME       JOB           SALARY
----------  ---------     ----------
KING        PRESIDENT     5000
BLAKE       MANAGER       2850
CLARK       MANAGER       2450
JONES       MANAGER       2975
MARTIN      SALESMAN      1250
ALLEN       SALESMAN      1600
TURNER      SALESMAN      1500
JAMES       CLERK         950
WARD        SALESMAN      1250
FORD        ANALYST       3000
SMITH       CLERK         800
SCOTT       ANALYST       3000
ADAMS       CLERK         1100
MILLER      CLERK         1300
DOE         CLERK

15 filas seleccionadas.
```

1.4. Will the SELECT statement execute successfully?

```
SELECT *
FROM salgrade;

     GRADE       LOSAL      HISAL
     ----------  ---------- ----------
     1           700        1200
     2           1201       1400
     3           1401       2000
     4           2001       3000
     5           3001       9999
```

1.5. There are three coding errors in this statement. Can you identify them?
Una corrección podría ser,

```
SELECT empno, ename, sal*12 ANNUAL_SALARY
FROM emp;

   EMPNO    ENAME        ANNUAL_SALARY
   ---------- ----------   -------------
   7839     KING         60000
   7698     BLAKE        34200
   7782     CLARK        29400
   7566     JONES        35700
   7654     MARTIN       15000
   7499     ALLEN        19200
   7844     TURNER       18000
   7900     JAMES        11400
   7521     WARD         15000
   7902     FORD         36000
   7369     SMITH        9600
   7788     SCOTT        36000
   7876     ADAMS        13200
   7934     MILLER       15600
   8000     DOE

15 filas seleccionadas.
```

1.6. Show the structure of the DEPT table. SELECT all data from the DEPT table.

```
DESCRIBE dept;
```

```
Nombre              ¿Nulo?       Tipo
 -------------------  --------     ----------------------------
 DEPTNO             NOT NULL     NUMBER(2)
 DNAME                           VARCHAR2(14)
 LOC                             VARCHAR2(13)
```

```
SELECT *
FROM dept;
```

```
   DEPTNO  DNAME           LOC
   ----------  --------------    -------------
      10     ACCOUNTING    NEW YORK
      20     RESEARCH      DALLAS
      30     SALES         CHICAGO
      40     OPERATIONS    BOSTON
```

1.7. Show the structure of the EMP table. Create a query to display the name, job, hire date and employee number for each employee, with employee number appearing first. Save your SQL statement to a file named *p1q7.sql.*

```
DESCRIBE emp;
```

```
Nombre               ¿Nulo?            Tipo
 ---------------------  --------          ----------------------------
 EMPNO               NOT NULL          NUMBER(4)
 ENAME                                 VARCHAR2(10)
 JOB                                   VARCHAR2(9)
 MGR                                   NUMBER(4)
 HIREDATE                              DATE
 SAL                                   NUMBER(7,2)
 COMM                                  NUMBER(7,2)
 DEPTNO              NOT NULL          NUMBER(2)
 ASTERISK                              VARCHAR2(30)
```

```
SELECT empno, ename, job, hiredate
FROM emp;
```

```
save p1q7
```

1.8. Run your query in the file *p1q7.sql*

```
run p1q7
```

```
    EMPNO    ENAME        JOB              HIREDATE
    ----------  ----------   ---------        --------
     7839      KING         PRESIDENT        17/11/81
     7698      BLAKE        MANAGER          01/05/81
     7782      CLARK        MANAGER          09/06/81
     7566      JONES        MANAGER          02/04/81
     7654      MARTIN       SALESMAN         28/09/81
     7499      ALLEN        SALESMAN         20/02/81
     7844      TURNER       SALESMAN         08/09/81
     7900      JAMES        CLERK            03/12/81
     7521      WARD         SALESMAN         22/02/81
     7902      FORD         ANALYST          03/12/81
     7369      SMITH        CLERK            17/12/80
     7788      SCOTT        ANALYST          09/12/82
     7876      ADAMS        CLERK            12/01/83
     7934      MILLER       CLERK            23/01/82
     8000      DOE          CLERK            08/05/01

15 filas seleccionadas.
```

1.9. Create a query to display unique jobs from the EMP table.

```
SELECT DISTINCT job
FROM emp;
```

```
JOB
---------
ANALYST
CLERK
MANAGER
PRESIDENT
SALESMAN
```

1.10. Load *p1q7.sql* into the SQL buffer. Name the column headings Emp #, Employee, Job, and Hire Date, respectively. Rerun your query.

```
SELECT empno "Emp  #", ename "Employee", job "JobHire", hiredate "Date"
FROM emp;
```

```
    Emp  #     Employee      JobHire           Date
    -------    ----------    ---------         --------
    7839       KING          PRESIDENT         17/11/81
    7698       BLAKE         MANAGER           01/05/81
    7782       CLARK         MANAGER           09/06/81
    7566       JONES         MANAGER           02/04/81
    7654       MARTIN        SALESMAN          28/09/81
    7499       ALLEN         SALESMAN          20/02/81
    7844       TURNER        SALESMAN          08/09/81
    7900       JAMES         CLERK             03/12/81
    7521       WARD          SALESMAN          22/02/81
    7902       FORD          ANALYST           03/12/81
    7369       SMITH         CLERK             17/12/80
    7788       SCOTT         ANALYST           09/12/82
    7876       ADAMS         CLERK             12/01/83
    7934       MILLER        CLERK             23/01/82
    8000       DOE           CLERK             08/05/01

15 filas seleccionadas.
```

1.11. Display the name concatenated with the job, separated by a comma and space, and name the column Employee and Title.

```
SELECT ename||', '||job AS "Employee and Title"
FROM emp;
```

```
Employee and Title
---------------------
KING, PRESIDENT
BLAKE, MANAGER
CLARK, MANAGER
JONES, MANAGER
MARTIN, SALESMAN
ALLEN, SALESMAN
TURNER, SALESMAN
JAMES, CLERK
WARD, SALESMAN
FORD, ANALYST
SMITH, CLERK
SCOTT, ANALYST
ADAMS, CLERK
MILLER, CLERK
DOE, CLERK

15 filas seleccionadas.
```

1.12. Create a query to display all the data from the EMP table. Separate each column by a comma. Name the column THE_OUTPUT.

```
SELECT empno||','||ename||','||job||','||hiredate||','||sal||','||comm||','||
          deptno AS THE_OUTPUT
FROM emp;
```

```
THE_OUTPUT
--------------------------------------------------------------------------------
7839,KING,PRESIDENT,17/11/81,5000,,10
7698,BLAKE,MANAGER,01/05/81,2850,,30
7782,CLARK,MANAGER,09/06/81,2450,,10
7566,JONES,MANAGER,02/04/81,2975,,20
7654,MARTIN,SALESMAN,28/09/81,1250,1400,30
7499,ALLEN,SALESMAN,20/02/81,1600,320,30
7844,TURNER,SALESMAN,08/09/81,1500,0,30
7900,JAMES,CLERK,03/12/81,950,,30
7521,WARD,SALESMAN,22/02/81,1250,500,30
7902,FORD,ANALYST,03/12/81,3000,,20
7369,SMITH,CLERK,17/12/80,800,80,20
7788,SCOTT,ANALYST,09/12/82,3000,,20
7876,ADAMS,CLERK,12/01/83,1100,,20
7934,MILLER,CLERK,23/01/82,1300,195,10
8000,DOE,CLERK,08/05/01,,0,10

15 filas seleccionadas.
```

TEMA 2. RESTRICCIÓN Y ORDENACIÓN DE LOS DATOS RECUPERADOS

2.1. Create a query to display the name and salary of employees earning more than $2850. Save your SQL statement to a file named *p2q1.sql*. Run your query.

```
SELECT ename, sal
FROM emp
WHERE sal > 2850;
```

```
ENAME       SAL
----------  ----------
KING        5000
JONES       2975
FORD        3000
SCOTT       3000
```

2.2. Create a query to display the employee name and department number for employee number 7566.

```
SELECT ename, deptno
FROM emp
WHERE empno = 7566;
```

```
ENAME       DEPTNO
----------  ----------
JONES       20
```

2.3. Modify *p2q1.sql* to display the name and salary for all employees whose salary is not in the range of $1500 and $2850. Resave your SQL statement to a file named *p2q3.sql*. Rerun your query

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

```
ENAME       SAL
----------  ----------
KING        5000
JONES       2975
MARTIN      1250
JAMES       950
WARD        1250
FORD        3000
SMITH       800
SCOTT       3000
ADAMS       1100
MILLER      1300

10 filas seleccionadas.
```

2.4. Display the employee name, job and start date of employees hired between February 20, 1981, and May 1, 1981. Order the query in ascending order of start date.

```
SELECT ename, job, hiredate
FROM emp
WHERE hiredate BETWEEN '20/02/2081' AND '01/05/2081'
ORDER BY hiredate ASC;
```

```
ENAME       JOB             HIREDATE
----------  ---------       --------
ALLEN       SALESMAN        20/02/81
WARD        SALESMAN        22/02/81
JONES       MANAGER         02/04/81
BLAKE       MANAGER         01/05/81
```

2.5. Display the employee name and department number of all employees in departments 10 and 30 in alphabetical order by name

```
SELECT ename, deptno
FROM emp
WHERE deptno IN (10, 30)
ORDER BY ename ASC;
```

```
ENAME       DEPTNO
----------  ----------
ALLEN       30
BLAKE       30
CLARK       10
DOE         10
JAMES       30
KING        10
MARTIN      30
MILLER      10
TURNER      30
WARD        30

10 filas seleccionadas.
```

2.6. Modify *p2q3.sql* to list the name and salary of employees who earn more than $1500 and are in department 10 or 30. Label the columns Employee and Monthly Salary, respectively. Resave your SQL statement to a file named *p2q6.sql*. Rerun your query.

```
SELECT ename "Employee", sal "Monthly Salary"
FROM emp
WHERE sal > 1500 AND (deptno = 10 OR deptno = 30);
```

```
Employee    Salary
----------  ----------
KING        5000
BLAKE       2850
CLARK       2450
ALLEN       1600
```

2.7. Display the name and hire date of every employee who was hired in 1982

```
SELECT ename, hiredate
FROM emp
WHERE hiredate LIKE '%82';
```

```
ENAME      HIREDATE
----------  --------
SCOTT      09/12/82
MILLER     23/01/82
```

2.8. Display the name and title of all employees who do not have a manager

```
SELECT ename, job
FROM emp
WHERE mgr IS NULL;
```

```
ENAME        JOB
----------   ---------
KING         PRESIDENT
```

2.9. Display the name, salary, and commission for all employees who earn commissions. Sort data in descending order of salary and commissions.

```
SELECT ename, sal, comm
FROM emp
WHERE comm IS NOT NULL
ORDER BY sal DESC, comm DESC;
```

```
ENAME        SAL          COMM
----------   ----------   ----------
DOE                       0
ALLEN        1600         320
TURNER       1500         0
MILLER       1300         195
MARTIN       1250         1400
WARD         1250         500
SMITH        800          80

7 filas seleccionadas.
```

2.10. Display the names of all employees where the third letter of their name is an *A*.

```
SELECT ename
FROM emp
WHERE ename LIKE '__A%';
```

```
ENAME
----------
BLAKE
CLARK
ADAMS
```

2.11. Display the name of all employees that have two *L*s in their name and are in department 30 or their manager is 7782.

```
SELECT ename
FROM emp
WHERE ename LIKE '%LL%' AND (deptno = 30 OR mgr = 7782);
```

```
ENAME
----------
ALLEN
MILLER
```

2.12. Display the name, job, and salary for all employees whose job is Clerk or Analyst and their salary is not equal to $1000, $3000, or $5000

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

```
ENAME       JOB         SAL
----------  ---------   ----------
JAMES       CLERK       950
SMITH       CLERK       800
ADAMS       CLERK       1100
MILLER      CLERK       1300
```

2.13. Modify *p2q6.sql* to display the name, salary, and commission for all employees whose commission amount is greater than their salary increased by 10%. Rerun your query. Resave your query as *p2q13.sql*

```
SELECT ename, sal, comm
FROM emp
WHERE comm > 0.1*sal+sal AND (deptno = 10 OR deptno = 30);
```

```
ENAME          SAL        COMM
----------     ----------  ----------
MARTIN         1250       1400
```

TEMA 3. FUNIONES DE SQL QUE ACTÚAN SOBRE UNA SOLA FILA

3.1. Write a query to display the current date. Label the column Date.

```
SELECT sysdate "Date"
FROM dual;
```

```
Date
--------
25/05/01
```

3.2. Display the employee number, name, salary, and salary increase by 15% expressed as a whole number. Label the column New Salary. Save your SQL statement to a file named *p3q2.sql*.

```
SELECT empno, ename, sal, ROUND (sal + sal*15/100) “New Salary”
FROM emp;
```

```
save p3q2
```

3.3. Run your query in the file *p3q2.sql*.

```
start p3q2

    EMPNO      ENAME       SAL          New Salary
    ---------- ----------  ----------   ------------
    7839       KING        5000         5750
    7698       BLAKE       2850         3278
    7782       CLARK       2450         2818
    7566       JONES       2975         3421
    7654       MARTIN      1250         1438
    7499       ALLEN       1600         1840
    7844       TURNER      1500         1725
    7900       JAMES       950          1093
    7521       WARD        1250         1438
    7902       FORD        3000         3450
    7369       SMITH       800          920
    7788       SCOTT       3000         3450
    7876       ADAMS       1100         1265
    7934       MILLER      1300         1495
    8000       DOE

15 filas seleccionadas.
```

3.4. Modify your query *p3q2.sql* to add an additional column that will subtract the old salary from the new salary. Label the column Increase. Rerun your query.

```
SELECT empno, ename, sal, ROUND (sal+sal*15/100) "New Salary",
        ROUND (sal+sal*15/100)-sal "Increase"
FROM emp;
```

```
    EMPNO      ENAME      SAL        New Salary     Increase
    ---------- ---------- ------     -------------  ----------
    7839       KING       5000       5750           750
    7698       BLAKE      2850       3278           428
    7782       CLARK      2450       2818           368
    7566       JONES      2975       3421           446
    7654       MARTIN     1250       1438           188
    7499       ALLEN      1600       1840           240
    7844       TURNER     1500       1725           225
    7900       JAMES      950        1093           143
    7521       WARD       1250       1438           188
    7902       FORD       3000       3450           450
    7369       SMITH      800        920            120
    7788       SCOTT      3000       3450           450
    7876       ADAMS      1100       1265           165
    7934       MILLER     1300       1495           195
    8000       DOE

15 filas seleccionadas.
```

3.5. Display the employee's name, hire date, and salary review date, which is the first Monday after six months of service. Label the column REVIEW. Format the dates to appear in the format similar to "Sunday, the Seventh of September, 1981."

```
SELECT ename, hiredate, TO_CHAR (NEXT_DAY (ADD_MONTHS
            (hiredate,6), 'LUNES'), 'fmDay", the" fmDd
FROM emp;
```

```
        ENAME       HIREDATE  REVIEW
        ----------  ---------- ------------------------------------------------
        KING        17/11/81 Lunes, the Eighteenth of Mayo, 2082
        BLAKE       01/05/81 Lunes, the Third of Noviembre, 2081
        CLARK       09/06/81 Lunes, the Fifteenth of Diciembre, 2081
        JONES       02/04/81 Lunes, the Sixth of Octubre, 2081
        MARTIN      28/09/81 Lunes, the Thirtieth of Marzo, 2082
        ALLEN       20/02/81 Lunes, the Twenty-Fifth of Agosto, 2081
        TURNER      08/09/81 Lunes, the Ninth of Marzo, 2082
        JAMES       03/12/81 Lunes, the Eighth of Junio, 2082
        WARD        22/02/81 Lunes, the Twenty-Fifth of Agosto, 2081
        FORD        03/12/81 Lunes, the Eighth of Junio, 2082
        SMITH       17/12/80 Lunes, the Twenty-Third of Junio, 2081
        SCOTT       09/12/82 Lunes, the Fourteenth of Junio, 2083
        ADAMS       12/01/83 Lunes, the Nineteenth of Julio, 2083
        MILLER      23/01/82 Lunes, the Twenty-Seventh of Julio, 2082
        DOE         08/05/01 Lunes, the Twelfth of Noviembre, 2001

15 filas seleccionadas.
```

3.6. For each employee display the employee name and calculate the number of months between today and the date the employee was hired. Label the column MONTHS_WORKED. Order your results by the number of months employed. Round the number of months up to the closest whole number.

NOTA: Debido a que en las tablas sólo aparecen los años con los dos últimos digitos, y que se referían a fechas del siglo pasado, salen esas cantidades tan raras. Habría que modificar el script que crea la tabla y reescribir los años con el formato de 4 dígitos.

```
SELECT ename, ROUND (MONTHS_BETWEEN (sysdate, hiredate))
            WORKED
FROM emp;
```

```
ENAME       WORKED
----------  ----------
KING        -966
BLAKE       -959
CLARK       -960
JONES       -958
MARTIN      -964
ALLEN       -957
TURNER      -963
JAMES       -966
WARD        -957
FORD        -966
SMITH       -955
SCOTT       -978
ADAMS       -980
MILLER      -968
DOE           1

15 filas seleccionadas.
```

3.7. Write a query that produces the following for each employee: <employee name> earns <salary> monthly but wants <3 times salary>. Label the column Dream Salaries

```
SELECT ename|| ' earns ' || TO_CHAR(sal, 'fm$99,999.00') || ' monthly but
            wants ' || TO_CHAR(sal*3, 'fm$99,999.00') "DreamSalaries"
FROM emp;
```

```
Dream Salaries
--------------------------------------------------------
KING earns $5,000.00 monthly but wants $15,000.00
BLAKE earns $2,850.00 monthly but wants $8,550.00
CLARK earns $2,450.00 monthly but wants $7,350.00
JONES earns $2,975.00 monthly but wants $8,925.00
MARTIN earns $1,250.00 monthly but wants $3,750.00
ALLEN earns $1,600.00 monthly but wants $4,800.00
TURNER earns $1,500.00 monthly but wants $4,500.00
JAMES earns $950.00 monthly but wants $2,850.00
WARD earns $1,250.00 monthly but wants $3,750.00
FORD earns $3,000.00 monthly but wants $9,000.00
SMITH earns $800.00 monthly but wants $2,400.00
SCOTT earns $3,000.00 monthly but wants $9,000.00
ADAMS earns $1,100.00 monthly but wants $3,300.00
MILLER earns $1,300.00 monthly but wants $3,900.00
DOE earns  monthly but wants

15 filas seleccionadas.
```

3.8. Create a query to display name and salary for all employees. Format the salary to be 15 characters long, left-padded with *$*. Label the column SALARY

```
SELECT ename, LPAD(sal,15,'$') SALARY
FROM emp;
```

```
ENAME       SALARY
----------  ---------------
KING        $$$$$$$$$$$5000
BLAKE       $$$$$$$$$$$2850
CLARK       $$$$$$$$$$$2450
JONES       $$$$$$$$$$$2975
MARTIN      $$$$$$$$$$$1250
ALLEN       $$$$$$$$$$$1600
TURNER      $$$$$$$$$$$1500
JAMES       $$$$$$$$$$$$950
WARD        $$$$$$$$$$$1250
FORD        $$$$$$$$$$$3000
SMITH       $$$$$$$$$$$$800
SCOTT       $$$$$$$$$$$3000
ADAMS       $$$$$$$$$$$1100
MILLER      $$$$$$$$$$$1300
DOE

15 filas seleccionadas.
```

3.9. Write a query that will display the employee's name with the first letter capitalized and all other letters lowercase and the length of their name, for all employees whose name starts with *J*, *A*, or *M*. Give each column an appropriate label

```
SELECT INITCAP(ename) “Name”, LENGTH(ename) “Length”
FROM emp
WHERE ename LIKE 'J%' OR ename LIKE 'A%' OR ename LIKE 'M%';
```

```
Name        Length
----------  -------------
Jones       5
Martin      6
Allen       5
James       5
Adams       5
Miller      6

6 filas seleccionadas.
```

3.10. Display the name, hire date, and day of the week on which the employee started. Label the column DAY. Order the results by the day of the week starting with Monday.

```
SELECT ename, hiredate, TO_CHAR(hiredate, 'DAY') DAY
FROM emp
ORDER BY TO_CHAR(hiredate, 'D') ASC;
```

```
ENAME       HIREDATE    DAY
----------  ----------  ---------
KING        17/11/81    LUNES
CLARK       09/06/81    LUNES
TURNER      08/09/81    LUNES
SMITH       17/12/80    MARTES
DOE         08/05/01    MARTES
ADAMS       12/01/83    MARTES
JONES       02/04/81    MIÉRCOLES
JAMES       03/12/81    MIÉRCOLES
SCOTT       09/12/82    MIÉRCOLES
FORD        03/12/81    MIÉRCOLES
BLAKE       01/05/81    JUEVES
ALLEN       20/02/81    JUEVES
MILLER      23/01/82    VIERNES
WARD        22/02/81    SÁBADO
MARTIN      28/09/81    DOMINGO

15 filas seleccionadas.
```

3.11. Create a query that will display the employee name and commission amount. If the employee does not earn commission, put "No Commission." Label the column COMM.

```
SELECT ename, NVL(TO_CHAR(comm), 'No commision') comm
FROM emp;
```

```
ENAME       COMM
----------  ---------------------------------------
KING        No commision
BLAKE       No commision
CLARK       No commision
JONES       No commision
MARTIN      1400
ALLEN       320
TURNER      0
JAMES       No commision
WARD        500
FORD        No commision
SMITH       80
SCOTT       No commision
ADAMS       No commision
MILLER      195
DOE         0

15 filas seleccionadas.
```

TEMA 4. OBTENIENDO DATOS DESDE MULTIPLES TABLAS

4.1. Write a query to display the name, department number, and department name for all employees.

```
SELECT emp.ename, emp.deptno, dept.dname
FROM emp, dept
WHERE emp.deptno = dept.deptno
ORDER BY emp.deptno;
```

```
ENAME       DEPTNO      DNAME
----------  ----------  -----------
KING        10          ACCOUNTING
CLARK       10          ACCOUNTING
DOE         10          ACCOUNTING
MILLER      10          ACCOUNTING
JONES       20          RESEARCH
SCOTT       20          RESEARCH
ADAMS       20          RESEARCH
SMITH       20          RESEARCH
FORD        20          RESEARCH
BLAKE       30          SALES
MARTIN      30          SALES
ALLEN       30          SALES
TURNER      30          SALES
JAMES       30          SALES
WARD        30          SALES

15 filas seleccionadas.
```

4.2. Create a unique listing of all jobs that are in department 30

```
SELECT DISTINCT emp.job, dept.loc
FROM emp, dept
WHERE emp.deptno = dept.deptno AND emp.deptno = 30;
```

```
JOB            LOC
---------      -------------
CLERK          CHICAGO
MANAGER        CHICAGO
SALESMAN       CHICAGO
```

4.3. Write a query to display the employee name, department name, and location of all employees who earn a commission.

```
SELECT DISTINCT emp.ename, dept.dname, dept.loc
FROM emp, dept
WHERE emp.deptno = dept.deptno AND emp.comm IS NOT NULL;
```

```
ALLEN       SALES        CHICAGO
DOE         ACCOUNTING        NEW YORK
MARTIN      SALES             CHICAGO
MILLER      ACCOUNTING        NEW YORK
SMITH       RESEARCH          DALLAS
TURNER      SALES             CHICAGO
WARD        SALES             CHICAGO
```

4.4. Display the employee name and department name for all employees who have an *A* in their name. Save your SQL statement in a file called *p4q4.sql*.

```
SELECT emp.ename, dept.dname
FROM emp, dept
WHERE emp.deptno = dept.deptno AND emp.ename LIKE '%A%'
ORDER BY dept.dname ASC;
```

```
ENAME       DNAME
----------  --------------
CLARK       ACCOUNTING
ADAMS       RESEARCH
BLAKE       SALES
MARTIN      SALES
ALLEN       SALES
JAMES       SALES
WARD        SALES

7 filas seleccionadas.
```

4.5. Write a query to display the name, job, department number, and department name for all employees who work in DALLAS.

```
SELECT emp.ename, emp.job, emp.deptno, dept.dname
FROM emp, dept
WHERE emp.deptno = dept.deptno AND dept.loc='DALLAS';
```

```
ENAME       JOB         DEPTNO      DNAME
----------  --------    ----------  --------------
JONES       MANAGER     20          RESEARCH
FORD        ANALYST     20          RESEARCH
SMITH       CLERK       20          RESEARCH
SCOTT       ANALYST     20          RESEARCH
ADAMS       CLERK       20          RESEARCH
```

4.6. Display the employee name and employee number along with their manager's name and manager number. Label the columns Employee, Emp#, Manager, and Mgr#, respectively. Save your SQL statement to a file called *p4q6.sql*.

```
SELECT worker.ename "Employee", worker.empno "Emp#",
        manager.ename "Manager", worker.mgr "Mgr#"
FROM emp worker, emp manager
WHERE worker.mgr = manager.empno;
```

```
Employee    Emp#   Manager     Mgr#
----------  ------ ----------  ----------
BLAKE       7698   KING        7839
CLARK       7782   KING        7839
JONES       7566   KING        7839
MARTIN      7654   BLAKE       7698
ALLEN       7499   BLAKE       7698
TURNER      7844   BLAKE       7698
JAMES       7900   BLAKE       7698
WARD        7521   BLAKE       7698
FORD        7902   JONES       7566
SMITH       7369   FORD        7902
SCOTT       7788   JONES       7566
ADAMS       7876   SCOTT       7788
MILLER      7934   CLARK       7782
DOE         8000   BLAKE       7698

14 filas seleccionadas.
```

4.7. Modify *p4q6.sql* to display all employees including King, who has no manager. Resave as *p4q7.sql*. Run *p4q7.sql*.

```
SELECT worker.ename "Employee", worker.empno "Emp#",
            manager.ename "Manager", worker.mgr "Mgr#"
FROM emp worker, emp manager
WHERE worker.mgr = manager.empno(+)
ORDER BY "Mgr#";
```

```
Employee      Emp#   Manager      Mgr#
------------  ------ ----------   ----------
FORD          7902   JONES        7566
SCOTT         7788   JONES        7566
MARTIN        7654   BLAKE        7698
ALLEN         7499   BLAKE        7698
DOE           8000   BLAKE        7698
JAMES         7900   BLAKE        7698
WARD          7521   BLAKE        7698
TURNER        7844   BLAKE        7698
MILLER        7934   CLARK        7782
ADAMS         7876   SCOTT        7788
BLAKE         7698   KING         7839
CLARK         7782   KING         7839
JONES         7566   KING         7839
SMITH         7369   FORD         7902
KING          7839

15 filas seleccionadas.
```

4.8. Create a query that will display the employee name, department number, and all the employees that work in the same department as a given employee. Give each column an appropriate label.

```
SELECT DISTINCT emplo.deptno departament, emplo.ename employee,
                                colle.ename colleague
FROM emp emplo, emp colle
WHERE emplo.deptno = colle.deptno
            AND emplo.ename NOT IN (colle.ename);
```

```
  DEPT  EMP         COLLEAGUE
---------- ----------  ----------
        10 CLARK       DOE
        10 CLARK       KING
        10 CLARK       MILLER
        10 DOE         CLARK
        10 DOE         KING
        10 DOE         MILLER
        10 KING        CLARK
        10 KING        DOE
        10 KING        MILLER
        10 MILLER      CLARK
        10 MILLER      DOE
        10 MILLER      KING
        20 ADAMS       FORD
        20 ADAMS       JONES
        20 ADAMS       SCOTT
        20 ADAMS       SMITH
        20 FORD        ADAMS
        20 FORD        JONES
        20 FORD        SCOTT
        20 FORD        SMITH
        20 JONES       ADAMS
        20 JONES       FORD
        20 JONES       SCOTT
        20 JONES       SMITH
        20 SCOTT       ADAMS
        20 SCOTT       FORD
        20 SCOTT       JONES
        20 SCOTT       SMITH
        20 SMITH       ADAMS
        20 SMITH       FORD
        20 SMITH       JONES
        20 SMITH       SCOTT
        30 ALLEN       BLAKE
        30 ALLEN       JAMES
        30 ALLEN       MARTIN
        30 ALLEN       TURNER
        30 ALLEN       WARD
        30 BLAKE       ALLEN
        30 BLAKE       JAMES
        30 BLAKE       MARTIN
        30 BLAKE       TURNER
        30 BLAKE       WARD
        30 JAMES       ALLEN
        30 JAMES       BLAKE
        30 JAMES       MARTIN
        30 JAMES       TURNER
        30 JAMES       WARD
        30 MARTIN      ALLEN
        30 MARTIN      BLAKE
        30 MARTIN      JAMES
        30 MARTIN      TURNER
        30 MARTIN      WARD
        30 TURNER      ALLEN
        30 TURNER      BLAKE
        30 TURNER      JAMES
        30 TURNER      MARTIN
        30 TURNER      WARD
        30 WARD        ALLEN
        30 WARD        BLAKE
        30 WARD        JAMES
        30 WARD        MARTIN
        30 WARD        TURNER

62 filas seleccionadas.
```

4.9. Show the structure of the SALGRADE table. Create a query that will display the name, job, department name, salary, and grade for all employees.

```
DESCRIBE salgrade
```

```
Nombre                  ¿Nulo?        Tipo
-----------------------------------   ---------------------------
GRADE                                 NUMBER
LOSAL                                 NUMBER
HISAL                                 NUMBER
```

```
SELECT e.ename ENAME, e.job JOB, d.dname DNAME,
          e.sal SAL, s.grade GRADE
FROM emp e, salgrade s, dept d
WHERE e.deptno = d.deptno AND e.sal BETWEEN s.losal AND s.hisal;
```

```
ENAME       JOB            DNAME         SAL       GRADE
----------  ---------      -------------- ---------- ----------
MILLER      CLERK          ACCOUNTING     1300     2
CLARK       MANAGER        ACCOUNTING     2450     4
KING        PRESIDENT      ACCOUNTING     5000     5
SMITH       CLERK          RESEARCH        800     1
FORD        ANALYST        RESEARCH       3000     4
SCOTT       ANALYST        RESEARCH       3000     4
JONES       MANAGER        RESEARCH       2975     4
ADAMS       CLERK          RESEARCH       1100     1
JAMES       CLERK          SALES           950     1
TURNER      SALESMAN       SALES          1500     3
BLAKE       MANAGER        SALES          2850     4
ALLEN       SALESMAN       SALES          1600     3
MARTIN      SALESMAN       SALES          1250     2
WARD        SALESMAN       SALES          1250     2

14 filas seleccionadas.
```

4.10. Create a query to display the name and hire date of any employee hired after employee Blake.

```
SELECT e.ename, e.hiredate
FROM emp e, emp b
WHERE e.hiredate < b.hiredate AND b.ename = 'BLAKE';
```

```
ENAME       HIREDATE
----------  --------
JONES       02/04/81
ALLEN       20/02/81
WARD        22/02/81
SMITH       17/12/80
DOE         08/05/01
```

4.11. Display all employees' names and hire dates along with their manager's name and hire date for all employees who were hired before their managers. Label the columns Employee, Emp Hiredate, Manager, and Mgr Hiredate, respectively.

```
SELECT worker.ename "Employee", worker.hiredate "Emp Hiredate",
        manager.ename "Manager", manager.hiredate "Mgr Hiredate"
FROM emp worker, emp manager
WHERE worker.mgr = manager.empno
       AND worker.hiredate < manager.hiredate;
```

```
Employee    EmpHi      Manager     Mgr Hire
----------  ---------- ----------  --------
BLAKE       01/05/81 KING          17/11/81
CLARK       09/06/81 KING          17/11/81
JONES       02/04/81 KING          17/11/81
ALLEN       20/02/81 BLAKE         01/05/81
WARD        22/02/81 BLAKE         01/05/81
SMITH       17/12/80 FORD          03/12/81
DOE         08/05/01 BLAKE         01/05/81
7 filas seleccionadas.
```

4.12. Create a query that displays the employees name and the amount of the salaries of the employees are indicated through asterisks. Each asterisk signifies a hundred dollars. Sort the data in descending order of salary. Label the column EMPLOYEE_AND_THEIR_SALARIES.

```
SELECT ename|| ' ' || RPAD('*', TRUNC(sal/100), '*')   AS
            EMPLOYEE_AND_THEIR_SALARIES
FROM emp
WHERE sal IS NOT NULL
ORDER BY sal DESC;
```

```
EMPLOYEE_AND_THEIR_SALARIES
--------------------------------------------------------------------------------
KING **************************************************
FORD ******************************
SCOTT ******************************
JONES *****************************
BLAKE ****************************
CLARK ************************
ALLEN ****************
TURNER ***************
MILLER *************
MARTIN ************
WARD ************
ADAMS ***********
JAMES *********
SMITH ********
14 filas seleccionadas.
```

TEMA 5. FUNCIONES DE AGREGACIÓN

5.1. Group functions work across many rows to produce one result. **TRUE** / False

5.2. Group functions include nulls in calculations: True / **FALSE**

5.3. The WHERE clause restricts rows prior to inclusion in a group calculation:
TRUE / False

5.4. Display the highest, lowest, sum, and average salary of all employees. Label the columns Maximum, Minimum, Sum, and Average, respectively. Round your results to the decimal position. Save your SQL statement in a file called *p5q4.sql*.

```
SELECT MAX(sal) "Maximum", MIN(sal) "Minimum", SUM(sal) "Sum",
            ROUND(AVG(sal)) "Average"
FROM emp;
```

```
  Maximum    Minimum      Sum       Average
   ----------   ----------    ----------   ----------
    5000        800           29025      2073
```

5.5. Modify *p5q4.sql* to display the minimum, maximum, sum, and average salary for each job type. Resave to a file called *p5q5.sql*. Rerun your query.

```
SELECT job, MAX(sal) "Maximum", MIN(sal) "Minimum",
           SUM(sal) "Sum", ROUND(AVG(sal)) "Average"
FROM emp
GROUP BY job;
```

```
JOB              Maximum    Minimum       Sum         Average
---------        ----------    ------------    ----------    ----------
ANALYST       3000         3000            6000      3000
CLERK            1300         800              4150      1038
MANAGER       2975         2450            8275      2758
PRESIDENT     5000         5000            5000      5000
SALESMAN      1600         1250            5600      1400
```

5.6. Write a query to display the number of people with the same job.

```
SELECT job, COUNT(*)
FROM emp
GROUP BY job;
```

```
JOB               COUNT(*)
---------          -------------
ANALYST               2
CLERK                 5
MANAGER               3
PRESIDENT             1
SALESMAN              4
```

5.7. Determine the number of managers without listing them. Label the column Number of Managers.

```
SELECT COUNT(DISTINCT (mgr)) "Number of Managers"
FROM emp
```

```
Number of Managers
-------------------------
            6
```

5.8. Write a query that will display the difference between the highest and lowest salaries. Label the column DIFFERENCE.

```
SELECT (MAX(sal) - MIN(sal)) diference
FROM emp;
```

```
DIFERENCE
----------
    4200
```

5.9. Display the manager number and the salary of the lowest paid employee for that manager. Exclude anyone where the manager id is not known. Exclude any groups where the minimum salary is less than $1000. Sort the output in descending order of salary.

```
SELECT mgr, MIN(sal)
FROM emp
GROUP BY mgr
HAVING MIN(sal) > 1000 AND mgr IS NOT NULL
ORDER BY MIN(sal) DESC;
```

```
      MGR   MIN(SAL)
---------- ----------
     7566       3000
     7839       2450
     7782       1300
     7788       1100
```

5.10. Write a query to display the department name, location name, number of employees, and the average salary for all employees in that department. Label the columns' dname, loc, Number of People, and Salary, respectively.

```
SELECT dept.dname, dept.loc, COUNT(dept.dname) "Number of People",
            AVG(emp.sal) "Salary"
FROM emp, dept
WHERE emp.deptno = dept.deptno
GROUP BY dept.dname, dept.loc;
```

```
DNAME            LOC            Nº of People   Salary
--------------   -------------  -------------- ---------
ACCOUNTING       NEW YORK       4              2916,66667
RESEARCH         DALLAS         5              2175
SALES            CHICAGO        6              1566,66667
```

5.11. Create a query that will display the total number of employees and of that total the number who were hired in 1980, 1981, 1982, and 1983. Give appropriate column headings.

```
SELECT COUNT(*) "TOTAL",
        SUM(DECODE(TO_CHAR(hiredate,'yy'),'80',1,0)) "1980",
        SUM(DECODE(TO_CHAR(hiredate,'yy'),'81',1,0)) "1981",
        SUM(DECODE(TO_CHAR(hiredate,'yy'),'82',1,0)) "1982",
        SUM(DECODE(TO_CHAR(hiredate,'yy'),'83',1,0)) "1983"
FROM emp;
```

```
   TOTAL      1980       1981       1982       1983
---------- ---------- ---------- ---------- ----------
    15          1         10          2          1
```

5.12. Create a matrix query to display the job, the salary for that job based upon department number and the total salary for that job for all departments, giving each column an appropriate heading.

```
SELECT job "Job",
        SUM(DECODE(deptno,10,sal)) "Dept 10",
        SUM(DECODE(deptno,20,sal)) "Dept 20",
        SUM(DECODE(deptno,30,sal)) "Dept 30",
        SUM(DECODE(deptno,40,sal)) "Dept 40",
        SUM(sal) "Total"
FROM emp
GROUP BY job;
```

```
Job          Dept 10    Dept 20    Dept 30    Dept 40      Total
---------  ---------- ---------- ---------- ---------- ----------
ANALYST                   6000                              6000
CLERK          1300       1900        950                   4150
MANAGER        2450       2975       2850                   8275
PRESIDENT      5000                                         5000
SALESMAN                             5600                   5600
```

TEMA 6. SUBCONSULTAS

6.1. Write a query to display the employee name and hire date for all employees in the same department as Blake. Exclude Blake

```
SELECT ename, hiredate
FROM emp
WHERE deptno = (SELECT deptno
                FROM emp
                WHERE empno = 7698)
      AND empno <> 7698
```

```
ENAME      HIREDATE
---------- --------
MARTIN     28/09/81
ALLEN      20/02/81
TURNER     08/09/81
JAMES      03/12/81
WARD       22/02/81
```

6.2. Create a query to display the employee number and name for all employees who earn more than the average salary. Sort the results in descending order of salary.

```
SELECT empno, ename
FROM emp
WHERE sal > (SELECT AVG(sal)
             FROM emp)
ORDER BY sal DESC;
```

```
EMPNO ENAME
---------- ----------
      7839 KING
      7902 FORD
      7788 SCOTT
      7566 JONES
      7698 BLAKE
      7782 CLARK

6 filas seleccionadas.
```

6.3. Write a query that will display the employee number and name for all employees who work in a department with any employee whose name contains a *T*. Save your SQL statement in a file called *p6q3.sql*.

```
SELECT empno, ename
FROM emp
WHERE deptno IN  (SELECT deptno
                  FROM emp
                  WHERE ename LIKE '%T%');
```

```
EMPNO ENAME
---------- ----------
    7566 JONES
    7788 SCOTT
    7876 ADAMS
    7369 SMITH
    7902 FORD
    7698 BLAKE
    7654 MARTIN
    7499 ALLEN
    7844 TURNER
    7900 JAMES
    7521 WARD
11 filas seleccionadas.
```

6.4. .Display the employee name, department number, and job title for all employees whose department location is Dallas.

```
SELECT ename, deptno, job
FROM emp
WHERE deptno IN  (SELECT deptno
                  FROM dept
                  WHERE loc= 'DALLAS');
```

```
ENAME   DEPTNO   JOB
----------  ---------------------
JONES        20      MANAGER
FORD         20      ANALYST
SMITH        20      CLERK
SCOTT        20      ANALYST
ADAMS        20      CLERK
```

6.5. Display the employee name and salary of all employees who report to King.

```
SELECT ename, sal
FROM emp
WHERE mgr IN  (SELECT empno
               FROM emp
               WHERE ename = 'KING');
```

```
ENAME          SAL
----------     ----------
BLAKE          2850
CLARK          2450
JONES          2975
```

6.6. Display the department number, name, and job for all employees in the Sales department.

```
SELECT deptno, ename, job
FROM emp
WHERE deptno IN  (SELECT deptno
                  FROM dept
                  WHERE dname = 'SALES');
```

```
DEPTNO        ENAME         JOB
------------  ----------    ---------
      30      BLAKE         MANAGER
      30      MARTIN        SALESMAN
      30      ALLEN         SALESMAN
      30      TURNER        SALESMAN
      30      JAMES         CLERK
      30      WARD          SALESMAN
6 filas seleccionadas.
```

6.7. Modify *p6q3.sql* to display the employee number, name, and salary for all employees who earn more than the average salary and who work in a department with any employee with a *T* in their name. Resave as *p6q7.sql*. Rerun your query.

```
SELECT empno, ename, sal
FROM emp
WHERE sal > (SELECT AVG(sal)
                FROM emp)
          AND deptno IN (SELECT deptno
                            FROM emp
                            WHERE ename LIKE '%T%');
```

```
EMPNO ENAME      SAL
---------- ----------  --------
     7566 JONES       2975
     7788 SCOTT       3000
     7902 FORD        3000
     7698 BLAKE       2850
```

TEMA 7. SUBCONSULTAS CON MÚLTIPLES COLUMNAS

7.1. Write a query to display the name, department number, and salary of any employee whose department number and salary matches both the department number and salary of any employee who earns a commission.

```
SELECT empno, ename, sal
FROM emp
WHERE (deptno, sal) IN  (SELECT deptno, sal
                         FROM emp
                         WHERE comm IS NOT NULL);
```

```
EMPNO ENAME      SAL
---------- ----------  ----------
     7934 MILLER     1300
     7369 SMITH      800
     7654 MARTIN     1250
     7521 WARD       1250
     7844 TURNER     1500
     7499 ALLEN      1600

6 filas seleccionadas.
```

7.2. Display the name, department name, and salary of any employee whose salary and commission matches both the salary and commission of any employee located in Dallas.

```
SELECT ename, dept.dname, sal
FROM emp, dept
WHERE (sal, NVL(comm, -1)) IN
                (SELECT sal, NVL(comm, -1)
                FROM emp
                WHERE deptno =  (SELECT deptno
                                 FROM dept
                                 WHERE loc = 'DALLAS'))
          AND emp.deptno = dept.deptno;
```

```
ENAME        DNAME        SAL
----------   -----------  ----------
SMITH        RESEARCH     800
ADAMS        RESEARCH     1100
JONES        RESEARCH     2975
FORD         RESEARCH     3000
SCOTT        RESEARCH     3000
```

7.3. Create a query to display the name, hire date, and salary for all employees who have both the same salary and commission as Scott.

```
SELECT ename, hiredate, sal
FROM emp
WHERE (sal, NVL(comm,-1)) IN (SELECT sal,NVL(comm,-1)
                              FROM emp
                              WHERE ename = 'SCOTT')
           AND ename <> 'SCOTT'
```

```
ENAME        HIREDATE     SAL
----------   -----------  ----------
FORD         03/12/81     3000
```

7.4. Create a query to display the employees that earn a salary that is higher than the salary of any of the CLERKS. Sort the results on salary from highest to lowest.

```
SELECT ename, job, sal
FROM emp
WHERE sal>(SELECT max(sal)
           FROM emp
           WHERE job='CLERK')
ORDER BY sal DESC;
```

```
ENAME      JOB        SAL
---------- ---------  ----------
KING       PRESIDENT  5000
FORD       ANALYST    3000
SCOTT      ANALYST    3000
JONES      MANAGER    2975
BLAKE      MANAGER    2850
CLARK      MANAGER    2450
ALLEN      SALESMAN   1600
TURNER     SALESMAN   1500

8 filas seleccionadas.
```

TEMA 9. MANIPULACIÓN DE DATOS.

9.1. Run the /home/db/InformaciónGeneral/practicasdb2000/practica9/***lab9_1.sql*** script to build the MY_EMPLOYEE table that will be used for the lab.

```
START lab9_1.sql
```

9.2. Describe the structure of the MY_EMPLOYEE table to identify the column names.

```
DESCRIBE my_employee
```

```
Nombre                                    ¿Nulo?   Tipo
----------------------------------------- -------- ----------------------------
ID                                        NOT NULL NUMBER(4)
LAST_NAME                                          VARCHAR2(25)
FIRST_NAME                                         VARCHAR2(25)
USERID                                             VARCHAR2(8)
SALARY                                             NUMBER(9,2)
```

9.3. Add the first row of data to the MY_EMPLOYEE table from the sample data below. Do not list the columns in the INSERT clause.

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	Patel	**Ralph**	rpatel	795
2	Dancs	Betty	bdancs	860
3	Biri	Ben	bbiri	110
4	Newman	Chad	cnewman	750
5	Ropeburn	Audry	aropebur	1550

```
INSERT INTO my_employee
VALUES (1 ,'Patel' ,'Ralph' ,'tpatel' ,795);
```

```
1 fila creada.
```

9.4. Populate the MY_EMPLOYEE table with the second row of sample data from the list above. This time, list the columns explicitly in the INSERT clause.

```
INSERT INTO my_employee
VALUES (2 ,'Dancs' ,'Betty' ,'bdancs' , 860);
```

```
1 fila creada.
```

9.5. Confirm your addition to the table.

```
SELECT *
FROM my_employee;
```

```
    ID LAST_NAME  FIRST_NAME     USERID   SALARY
------ ------------------- --------------------    --------      ----------
     1 Patel               Ralph               tpatel       1000
     2 Dancs               Betty               bdancs       860
```

9.6. Create a script named *loademp.sql* to load rows into the MY_EMPLOYEE table interactively. Prompt the user for the employee's first name, last name, and salary. Concatenate the first letter of the first name and the first seven characters of the last name to produce the userid.

```
ACCEPT      my_emplo_id PROMPT 'Introduzca el ID del trabajador: '
ACCEPT      my_emplo_first_name PROMPT 'Introduzca el NOMBRE del
             trabajador: '
ACCEPT      my_emplo_last_name PROMPT 'Introduzca el APELLIDO del
            trabajador: '
ACCEPT      my_emplo_salary PROMPT 'Introduzca el SALARIO del
             trabajador: '
INSERT      INTO my_employee
VALUES     (&my_emplo_id,
            '&my_emplo_last_name',
            '&my_emplo_first_name',
            LOWER(CONCAT( SUBSTR('&my_emplo_first_name',1,1),
            SUBSTR('&my_emplo_last_name',1,7))),
            &my_emplo_salary);
```

9.7. Populate the table with the next two rows of sample data by running the script you created.

```
start p9q6
start p9q6
```

9.8. Confirm your additions to the table.

```
SELECT *
FROM my_employee;
```

```
   ID LAST_NAME     FIRST_NAME      USERID    SALARY
------ ----------------     -----------------     --------      ----------
     1 Patel              Ralph               tpatel      1000
     2 Dancs              Betty               bdancs       860
     3 Biri               BEn                 bbiri       1100
     4 Newman             Chad                cnewman    750
```

9.9. Make the data additions permanent.

```
COMMIT
```

```
Validación terminada.
```

9.10. Change the last name of employee 3 to Drexler.

```
UPDATE my_employee
SET last_name = 'Drexler'
WHERE ID = 3;
```

9.11. Change the salary to 1000 for all employees with a salary less than 900.

```
UPDATE my_employee
SET salary = 1000
WHERE salary < 900;
```

9.12. Verify your changes to the table.

```
SELECT last_name, salary
FROM my_employee;
```

```
LAST_NAME                SALARY
------------------------ ----------
Patel             1000
Dancs             1000
Biri          1100
Newman              1000
```

9.13. Delete Betty Dancs from the MY_EMPLOYEE table.

```
DELETE my_employee
WHERE first_name = 'Betty' AND last_name = 'Dancs';
```

9.14. Confirm your changes to the table.

```
SELECT id, last_name, first_name, userid, salary
FROM my_employee;
```

```
     ID LAST_NAME    FIRST_NAME    USERID    SALARY
------- ------------------- ------------------- -------- ----------
      1 Patel              Ralph              tpatel      1000
      3 Biri               BEn                bbiri       1100
      4 Newman             Chad               cnewman     1000
```

9.15. Commit all pending changes.

```
COMMIT
```

```
Validación terminada.
```

9.16. Populate the table with the last row of sample data by running the script you created in step 6.

```
START p6q2
```

9.17. Confirm your addition to the table.

```
SELECT id, last_name, first_name, userid, salary
FROM my_employee;
```

```
   ID LAST_NAME      FIRST_NAME     USERID     SALARY
----- -------------------    --------------------    --------       ----------
    5 Ropeburn           Audry                aropebur    1550
    1 Patel               Ralph                tpatel       1000
    3 Drexler            Ben                  bbiri        1100
    4 Newman             Chad                 cnewman    1000
```

9.18. Mark an intermediate point in the processing of the transaction.

```
SAVEPOINT p9q18
```

```
Punto de seguridad creado.
```

9.19. Empty the entire table.

```
  1* DELETE my_Employee
```

```
4 filas borradas.
```

9.20. Confirm that the table is empty.

```
SELECT *
FROM my_employee;
```

```
ninguna fila seleccionada
```

9.21. Discard the most recent DELETE operation without discarding the earlier INSERT operation.

```
ROLLBACK TO SAVEPOINT p9q18
```

```
Rollback terminado.
```

9.22. Confirm that the new row is still intact.

```
SELECT *
FROM my_Employee;
```

```
   ID LAST_NAME      FIRST_NAME        USERID     SALARY
------ --------------------  ---------------------  --------  ----------
     5 Ropeburn           Audry                aropebur   1550
     1 Patel              Ralph                tpatel     1000
     3 Drexler            Ben                  bbiri      1100
     4 Newman             Chad                 cnewman    1000
```

9.23. Make the data addition permanent.

```
COMMIT
```

```
Validación terminada.
```

TEMA 10. CREANDO Y MANEJANDO TABLAS.

10.1. Create the DEPARTMENT table based on the table instance chart given below. Enter the syntax in a script called *p10q1.sql*, then execute the script to create the table. Confirm that the table is created.

Column Name	Id	Name
Key Type		
Nulls / Unique		
FK Table		
FK Column		
Datatype	Number	Varchar2
Length	7	25

```
CREATE TABLE DEPARTAMENT
 (Id NUMBER(7), Name VARCHAR2(25));
```

```
Tabla creada.
```

```
DESCRIBE departament
```

```
Nombre                          ¿Nulo?   Tipo
 ------------------------------- -------- ---------------------------
 ID                                       NUMBER(7)
 NAME                                     VARCHAR2(25)
```

10.2. Populate the DEPARTMENT table with data from the DEPT table. Include only columns that you need.

```
INSERT INTO DEPARTAMENT (id, name)
SELECT deptno, dname
FROM DEPT
```

```
4 filas creadas.
```

10.3. Create the EMPLOYEE table based on the table instance chart given below. Enter the syntax in a script called *p10q3.sql*, and then execute the script to create the table. Confirm that the table is created.

```
CREATE TABLE EMPLOYEE
      (ID NUMBER(7),
       LAST_NAME VARCHAR2(25),
       FIRST_NAME VARCHAR2(25),
       DEPT_ID NUMBER(7));
```

```
Tabla creada.
```

```
DESCRIBE employee
```

```
Nombre                            ¿Nulo?   Tipo
---------------------------------------- ------  ----------------------------
ID                                          NUMBER(4)
LAST_NAME                                   VARCHAR2(10)
DEPT_ID                                     NUMBER(2)
SALARY                                      NUMBER(7)
```

10.4. Modify the EMPLOYEE table to allow for longer employee last names. Confirm your modification.

```
ALTER TABLE EMPLOYEE
MODIFY (LAST_NAME VARCHAR2(50));
```

```
Tabla modificada.
```

10.5. Confirm that both the DEPARTMENT and EMPLOYEE tables are stored in the data dictionary. (*Hint*: USER_TABLES).

```
SELECT DISTINCT TABLE_NAME
FROM user_tables
WHERE table_name IN ('DEPARTAMENT','EMPLOYEE');
```

```
TABLE_NAME
------------------------------
DEPARTAMENT
EMPLOYEE
```

10.6. Create the EMPLOYEE2 table based on the structure of the EMP table, include only the EMPNO, ENAME and DEPTNO columns. Name the columns in your new table ID, LAST_NAME and DEPT_ID, respectively.

```
CREATE TABLE employee2
AS
SELECT empno ID, ename LAST_NAME, deptno DEPT_ID
FROM emp
```

```
Tabla creada.
```

10.7. Drop the EMPLOYEE table.

```
DROP TABLE employee
```

```
Tabla borrada.
```

10.8. Rename the EMPLOYEE2 table to EMPLOYEE.

```
RENAME employee2 TO employee
```

```
Tabla cambiada de nombre.
```

10.9. Add a comment to the DEPARTMENT and EMPLOYEE table definitions describing the tables. Confirm your additions in the data dictionary.

```
COMMENT ON TABLE departament
IS 'Tabla que almacena el ID y el NAME de los departamentos'

COMMENT ON TABLE employee
IS 'Tabla que almacena el ID, el LAST_NAME y del DEPT_ID de cada
    empleado'

SELECT TABLE_NAME, COMMENTS
FROM USER_TAB_COMMENTS
WHERE table_name IN ('DEPARTAMENT','EMPLOYEE');
```

```
TABLE_NAME      COMMENTS
--------------------- ----------------------------------------------------------
DEPARTAMENT    Tabla que almacena el ID y el NAME de los
               departamentos
EMPLOYEE       Tabla que almacena el ID, el LAST_NAME y del
               DEPT_ID de cada empleado
```

TEMA 11. INCLUYENDO CONSTRAINTS.

11.1. Add a table level PRIMARY KEY constraint to the EMPLOYEE table using the ID column. The constraint should be enabled at creation.

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

```
Tabla modificada.
```

11.2. Create a PRIMARY KEY constraint on the DEPARTMENT table using the ID column. The constraint should be enabled at creation.

```
ALTER TABLE departament
ADD CONSTRAINT departament_id_pk
PRIMARY KEY (id);
```

```
Tabla modificada.
```

11.3. Add a foreign key reference on the EMPLOYEE table that will ensure that the employee is not assigned to a nonexistent department.

```
ALTER TABLE employee
ADD CONSTRAINT employee_dept_id_fk
FOREIGN KEY (dept_id) REFERENCES departament(id);
```

```
Tabla modificada.
```

11.4. Confirm that the constraints were added by querying USER_CONSTRAINTS. Note the types and names of the constraints. Save your statement text in a file called *p11q4.sql*.

```
SELECT constraint_name, constraint_type C, search_condition
FROM user_constraints
WHERE table_name IN ('EMPLOYEE', 'DEPARTAMENT');
```

```
CONSTRAINT_NAME              C  SEARCH_CONDITION
------------------------------ - --------------------------------------------------------
--------------------
DEPARTAMENT_ID_PK              P
SYS_C001720                 C "ID" IS NOT NULL
SYS_C001721                 C "DEPT_ID" IS NOT NULL
EMPLOYEE_ID_PK               P
EMPLOYEE_DEPT_ID_FK            R
```

11.5. Display the object names and types from the USER_OBJECTS data dictionary view EMPLOYEE and DEPARTMENT tables. You may want to format the columns for readability. Notice that the new tables and a new index were created.

```
SELECT DISTINCT object_name, object_type
FROM user_objects
WHERE object_name LIKE 'DEPARTAMENT%'
        OR object_name LIKE 'EMPLOYEE%';
```

```
OBJECT_NAME                    OBJECT_TYPE
---------------------------------------- ------------------
DEPARTAMENT                    TABLE
DEPARTAMENT_ID_PK              INDEX
EMPLOYEE                       TABLE
EMPLOYEE_DEPT_ID_IDX           INDEX
EMPLOYEE_ID_PK                 INDEX
```

11.6. Modify the EMPLOYEE table. Add a SALARY column of NUMBER data type, precision 7.

```
ALTER TABLE employee
ADD (salary NUMBER(7));
```

```
Tabla modificada.
```

TEMA 12. CREANDO VISTAS.

12.1. Create a view called EMP_VU based on the employee number, employee name, and department number from the EMP table. Change the heading for the employee name to EMPLOYEE.

```
CREATE VIEW emp_vu
AS SELECT empno, ename EMPLOYEE, deptno
FROM emp;
```

```
Vista creada.
```

12.2. Display the content's of the EMP_VU view.

```
SELECT *
FROM emp_vu;
```

```
EMPNO EMPLOYEE          DEPTNO
---------- ----------          ----------
     7839 KING              10
     7698 BLAKE             30
     7782 CLARK             10
     7566 JONES             20
     7654 MARTIN            30
     7499 ALLEN             30
     7844 TURNER            30
     7900 JAMES             30
     7521 WARD              30
     7902 FORD              20
     7369 SMITH             20
     7788 SCOTT             20
     7876 ADAMS             20
     7934 MILLER            10
     8000 DOE               10

15 filas seleccionadas.
```

12.3. SELECT the view_name and text from the data dictionary USER_VIEWS.

```
SELECT view_name, text
FROM user_views
WHERE view_name = 'EMP_VU';
```

```
VIEW_NAME       TEXT
------------------   ------------------------------------------------------------------
EMP_VU          SELECT empno, ename EMPLOYEE, deptno
                FROM emp
```

12.4. Using your view EMP_VU, enter a query to display all employee names and department numbers.

```
SELECT employee, deptno
FROM emp_vu;
```

```
EMPLOYEE  DEPTNO
----------  ----------
KING        10
BLAKE       30
CLARK       10
JONES       20
MARTIN      30
ALLEN       30
TURNER      30
JAMES       30
WARD        30
FORD        20
SMITH       20
SCOTT       20
ADAMS       20
MILLER      10
DOE         10

15 filas seleccionadas.
```

12.5. Create a view named DEPT20 that contains the employee number, employee name, and department number for all employees in department 20. Label the view column EMPLOYEE_ID, EMPLOYEE, and DEPARTMENT_ID. Do not allow an employee to be reassigned to another department through the view.

```
CREATE VIEW dept20
   AS SELECT ename EMPLOYEE_ID, ename EMPLOYEE,
                  Deptno DEPARTAMENT_ID
       FROM emp
       WHERE deptno = 20
       WITH READ ONLY;
```

```
Vista creada.
```

12.6. Display the structure and contents of the DEPT20 view.

```
DESCRIBE dept20
```

```
Nombre                         ¿Nulo?   Tipo
--------------------------------  --------   ---------------------------
 EMPLOYEE_ID                              VARCHAR2(10)
 EMPLOYEE                                 VARCHAR2(10)
 DEPARTAMENT_ID                           NOT NULL NUMBER(2)
```

```
SELECT *
FROM dept20;
```

```
EMPLOYEE_I   EMPLOYEE    DEPARTAMENT_ID
----------   ----------   -------------------------
JONES        JONES          20
FORD         FORD           20
SMITH        SMITH          20
SCOTT        SCOTT          20
ADAMS        ADAMS          20
```

12.7. Attempt to reassign Smith to department 30.

```
UPDATE dept20
SET DEPARTAMENT_ID = 30
WHERE EMPLOYEE = 'Smith';
```

```
ERROR en línea 2:
ORA-01733: columna virtual no permitida aquí
```

12.8. Create a view called SALARY_VU based on the employee name, department name, salary and salary grade for all employees. Label the columns Employee, Department, Salary and Grade, respectively.

```
CREATE VIEW salary_vu
    AS SELECT emp.ename "Employee", dept.dname "Departament",
                 emp.sal "Salary", salgrade.grade "Grade"
        FROM emp, dept, salgrade
        WHERE  emp.deptno = dept.deptno
                 AND emp.sal BETWEEN salgrade.losal
                 AND salgrade.hisal;
```

```
Vista creada.
```

TEMA 13. OTROS OBJETOS DE BASE DE DATOS.

13.1. Create a sequence to be used with the DEPARTMENT table's primary key column. The sequence should start at 60 and have a maximum value of 200. Have your sequence increment by ten numbers. Name the sequence DEPT_ID_SEQ.

```
CREATE SEQUENCE dept_id_seq
INCREMENT BY 10
START WITH 60
MAXVALUE 200;
```

```
Secuencia creada.
```

13.2. Write a script to display the following information about your sequences: sequence name, maximum value, increment size, and last number. Name the script *p13q2.sql*. Execute your script.

```
SELECT sequence_name, max_value, increment_by, last_number
FROM user_sequences;
```

```
SEQUENCE_NAME   MAX_VALUE INCREMENT_BY LAST_NUMBER
------------------------- ------------------ ---------------------- ----------------
CUSTID             1,0000E+27          1          109
DEPT_ID_SEQ               200          10         210
…                         …            …          …
ORDID              1,0000E+27          1          622
PRODID             1,0000E+27          1          200381
```

13.3. Write an interactive script to insert a row into the DEPARTMENT table. Name your script *p13q3.sql*. Be sure to use the sequence that you created for the ID column. Create a customized prompt to enter the department name. Execute your script. Add two departments named Education and Administration. Confirm your additions.

```
ACCEPT departament_name PROMPT 'Inserte nombre del Departamento: '
INSERT INTO departament (id, name)
VALUES(dept_id_seq.NEXTVAL, '&departament_name');
```

```
1 fila creada
```

13.4. Create a non-unique index on the FOREIGN KEY column in the EMPLOYEE table.

```
CREATE INDEX employee_dept_id_idx
ON employee(dept_id)
```

```
Indice creado.
```

13.5. Display the indexes and uniqueness that exist in the data dictionary for the EMPLOYEE table.   Save the statement into a script named *p13q5.sql*.

```
SELECT index_name, table_name, uniqueness
FROM user_indexes
WHERE table_name = 'EMPLOYEE';
```

```
INDEX_NAME                     TABLE_NAME           UNIQUENES
------------------------------ -------------------- ---------
EMPLOYEE_DEPT_ID_IDX           EMPLOYEE             NONUNIQUE
EMPLOYEE_ID_PK                 EMPLOYEE             UNIQUE
```

TEMA 16. DECLARACION DE VARIABLES.

16.1. Evaluate each of the following declarations. Determine which of them are *not* legal and explain why.

```
DECLARE
        v_id NUMBER(4);
BEGIN
        v_id := 4;
END;
```

Procedimiento PL/SQL terminado correctamente.

```
DECLARE
        v_x, v_y, v_z VARCHAR2(10);
BEGIN
        v_x := a;
END;
```

No se admiten las declaraciones múltiples de una variable.

```
DECLARE
        v_birthdate DATE NOT NULL;
BEGIN
        v_birthdate := SYSDATE;
END;
```

U omitimos NOT NULL, o le añadimos := SYSDATE

```
DECLARE
        v_in_stock BOOLEAN := 1;
BEGIN
        v_in_stock := 0;
END;
```

El tipo de la expresión no es correcto, debe ser TRUE o FALSE.

```
DECLARE
        TYPE name_table_type IS TABLE OF VARCHAR(20)
        INDEX BY BINARY_INTEGER;
        dept_name_table name_table_type;
        v_id NUMBER(4);
BEGIN
        v_id := 25;
END;
```

Procedimiento PL/SQL terminado correctamente.

16.2. In each of the following assignments, determine the data type of the resulting expression.

```
v_days_to_go := v_due_date – SYSDATE;
```

Si lo que hay a la derecha de ':=' es de tipo SYSDATE, el resultado también lo será.

```
v_sender := USER || ':' || TO_CHAR(v_dept_no);
```

Si 'USER' es de tipo CHAR, el resultado es CHAR.

```
v_sum := $100,000 + $250,000;
```

Si el símbolo '$' forma parte de los números (es sólo formato), entonces es NUMBER.

```
v_flag := TRUE;
```

Tipo BOOLEAN.

```
v_n1 := v_n1 > (2 * v_n3);
```

Tipo BOOLEAN.

```
v_value := NULL;
```

Cualquier tipo.

16.3. Create an anonymous block to output the phrase "My PL/SQL Block Works" to the screen.

```
VAR temp VARCHAR2(50);
BEGIN
        :temp := 'Mi Bloque PL/SQL funciona';
END;

PRINT temp;
```

```
TEMP
--------------------------------------------------------------------------------
Mi Bloque PL/SQL funciona
```

16.4. Create a block that declares two variables. Assign the value of these PL/SQL variables to SQL*Plus host variables and print the results of the PL/SQL variables to the screen. Execute your PL/SQL block. Save your PL/SQL block to a file named *p16q4.sql*.

```
VAR G_CHAR CHAR(50);
VAR G_NUMERO NUMBER;

DECLARE
        V_CHAR CHAR(50);
        V_NUM NUMBER;
BEGIN
        V_CHAR := '42 is the answer';
        V_NUM := TO_NUMBER(SUBSTR (V_CHAR, 1, 2) );
        :G_CHAR := V_CHAR;
        :G_NUMERO := V_NUM;
END;

PRINT G_CHAR;
PRINT G_NUMERO;
```

```
Procedimiento PL/SQL terminado correctamente.
G_CHAR
--------------------------------------------------------------------------------
42 is the answer

G_NUMERO
----------------
     42
```

TEMA 17. ESCRIBIENDO MANDATOS EJECUTABLES.

17.1. Evaluate the PL/SQL block on the previous page and determine each of the following values according to the rules of scoping

PL/SQL Block

```
DECLARE
        v_weight        NUMBER(3) := 600;
        v_message       VARCHAR2(255) := 'Product 10012';
BEGIN
                                SUB-BLOCK
          DECLARE
                  v_weight        NUMBER(3) := 1;
                  v_message       VARCHAR2(255) := 'Product 11001';
                  v_new_locn      VARCHAR2(50) := 'Europe';
          BEGIN
                  v_weight := v_weight + 1;
                  v_new_locn := 'Western ' || v_new_locn;
          END;

        v_weight := v_weight + 1;
        v_message := v_message || ' is in stock';
        v_new_locn      := 'Western ' || v_new_locn;

END;
```

The value of V_WEIGHT in the subbloc is
2
The value of V_NEW_LOCN in the subblock is
Western Europe
The value of V_WEIGHTin the main block is
601
The value of V_MESSAGE in the main block is
Product 10012 is in stock

The value of V_NEW_LOCNin the main block is
NOT DEFINED

17.2. Suppose you embed a subblock within a block, as shown on the previous page. You declare two variables, V_CUSTOMER and V_CREDIT_RATING, in the main block. You also declare two variables, V_CUSTOMER and V_NAME, in the subblock. Determine the values for each of the following cases.

Scope Example

```
DECLARE
  v_customer         VARCHAR2(50) := 'Womansport';
  v_credit_rating    VARCHAR2(50) := 'EXCELLENT';
BEGIN

      DECLARE
        v_customer   NUMBER(7) := 201;
        v_name       VARCHAR2(25) := 'Unisports';
      BEGIN
        v_customer          v_name         v_credit_rating
      END;

        v_customer          v_name         v_credit_rating

END;
```

The value of V_COSTUMER in the subblock is
201
The value of V_NAME in the subblock is
Unisports
The value of V_CREDIT_RATING in the subblock is
EXCELLENT
The value of V_COSTUMER in the main block is
Woman Sport

The value of V_NAME in the main block is
NOT DEFINED
The value of V_CREDIT_RATING in the main block is
EXCELLENT

17.3. Create and execute a PL/SQL block that accepts two numbers through SQL*Plus substitution variables. The first number should be divided by the second number and have the second number added to the result. The result should be written to a PL/SQL variable and printed to the screen.

```
ACCEPT v_primera PROMPT 'PRIMERA VARIABLE: '
ACCEPT v_segunda PROMPT 'SEGUNDA VARIABLE: '

DECLARE
        v_result NUMBER(3);
BEGIN
        v_result := ( TO_NUMBER(&v_primera) /
                        TO_NUMBER(&v_segunda) ) +
                        TO_NUMBER(&v_segunda);
        DBMS_OUTPUT.PUT_LINE(v_result);
END;
```

```
start p17q3a
PRIMERA VARIABLE: 2
SEGUNDA VARIABLE: 4
antiguo  4: v_result := (TO_NUMBER(&v_primera) /
                        TO_NUMBER(&v_segunda)) +
                        TO_NUMBER(&v_segunda);
nuevo  4:  v_result := ( TO_NUMBER(2) / TO_NUMBER(4) ) +
                        TO_NUMBER(4);

5

Procedimiento PL/SQL terminado correctamente.
```

17.4. Build a PL/SQL block that computes the total compensation for one year. The annual salary and the annual bonus percentage are passed to the PL/SQL block through SQL*Plus substitution variables and the bonus needs to be converted from a whole number to a decimal (for example, 15 to .15). If the salary is null, set it to zero before computing the total compensation. Execute the PL/SQL block. Reminder: Use the NVL function to handle null values.
Note: To test the NVL function type NULL at the prompt; pressing [Return] results in a missing expression error.

```
ACCEPT v_amount PROMPT 'CANTIDAD ANUAL: '
ACCEPT v_percentage PROMPT 'PORCENTAJE: '

DECLARE
        v_result NUMBER(6);
        v_decimalpercentage NUMBER(6);
BEGIN
        v_decimalpercentage := TO_NUMBER(&v_percentage) / 100.0;
        DBMS_OUTPUT.PUT_LINE(v_decimalpercentage);
        v_result := (TO_NUMBER(&v_amount) * v_decimalpercentage) +
                        TO_NUMBER(&v_amount);
        DBMS_OUTPUT.PUT_LINE(v_result);
END;
```

```
start p17q4
CANTIDAD ANUAL: 50000
PORCENTAJE: 10
antiguo  5:    v_decimalpercentage := TO_NUMBER(&v_percentage) / 100.0;
nuevo  5: v_decimalpercentage := TO_NUMBER(10) / 100.0;
antiguo 7: v_result := ( TO_NUMBER(&v_amount) * v_decimalpercentage )
                              + TO_NUMBER(&v_amount);
nuevo 7: v_result := ( TO_NUMBER(50000) * v_decimalpercentage ) +
                              TO_NUMBER(50000);
0
50000

Procedimiento PL/SQL terminado correctamente.
```

TEMA 18. INTERACTUANDO CON ORACLE SERVER.

18.1. Create a PL/SQL block that SELECTs the maximum department number in the DEPT table and store it in a SQL*Plus variable. Print the results to the screen. Save your PL/SQL block to a file named *p18q1.sql*.

```
VARIABLE g_maximo NUMBER

DECLARE
        v_maximo dept.deptno%TYPE;
BEGIN
        SELECT MAX(deptno)
        INTO v_maximo
        FROM dept;
        :g_maximo := v_maximo;
END;
/
PRINT g_maximo
```

```
SQL> start p18q1
Procedimiento PL/SQL terminado correctamente.
G_MAXIMO
----------------
     40
```

18.2. Create a PL/SQL block that inserts a new department into the DEPT table. Save your PL/SQL block to a file named *p18q2.sql*

a. Use the department number retrieved from exercise 1 and add 10 to that number as the input department number for the new department
b. Use a parameter for the department name
c. Leave the location null for now
d. Execute the PL/SQL block
e. Display the new department that you created

```
 ACCEPT dept_nombre PROMPT 'Por favor itroduzca el NOMBRE del
                    DEPARTAMENTO: '

DECLARE
          v_maximo dept.deptno%TYPE;
          v_localizacion dept.loc%TYPE;
          v_nombre dept.dname%TYPE;
BEGIN
          SELECT MAX(deptno)
          INTO v_maximo
          FROM dept;
          v_maximo := v_maximo + 10;
          INSERT INTO dept( deptno, dname, loc)
                    VALUES (v_maximo,'&dept_nombre', NULL );
          SELECT deptno, dname, loc
          INTO v_maximo, v_nombre, v_localizacion
          FROM dept
          WHERE deptno = v_maximo;
          dbms_output.put_line ('DEPTNO   DNAME   LOC');
          dbms_output.put_line (TO_CHAR(v_maximo) ||'  '|| v_nombre ||'   '||
                                        v_localizacion);
END;
/
SELECT *
FROM dept
WHERE dname = 'EDUCACION';
```

```
SQL> start p18q2
Por favor itroduzca el NOMBRE del DEPARTAMENTO: EDUCACION
antiguo  13:             VALUES (v_maximo,'&dept_nombre', NULL );
nuevo  13:               VALUES (v_maximo,'EDUCACION', NULL );
Procedimiento PL/SQL terminado correctamente.

DEPTNO DNAME          LOC
 ---------- ----------------- -------------
         50 EDUCACION
```

18.3. Create a PL/SQL block that updates the location for an existing department. Save your PL/SQL block to a file named *p18q3.sql*

a. Use a parameter for the department number
b. Use a parameter for the department location
c. Test the PL/SQL block
d. Display the department number, department name, and location for the updated department
e. Display the department that you updated

```
ACCEPT dept_numero PROMPT 'Por favor itroduzca el NUMERO del
            DEPARTAMENTO a MODIFICAR: '
ACCEPT dept_loc PROMPT 'Por favor itroduzca la NUEVA
            LOCALIZACION del DEPARTAMENTO: '

DECLARE
        v_numero dept.deptno%TYPE;
        v_localizacion dept.loc%TYPE;
        v_nombre dept.dname%TYPE;
BEGIN
        v_numero := &dept_numero;
        UPDATE  dept
        SET loc = '&dept_loc'
        WHERE deptno = v_numero;
        SELECT deptno, dname, loc
        INTO v_numero, v_nombre, v_localizacion
        FROM dept
        WHERE deptno = v_numero;
        dbms_output.put_line ('DEPTNO   DNAME   LOC');
        dbms_output.put_line (TO_CHAR(v_numero) ||'  '|| v_nombre ||'   '||
            v_localizacion);
END;
```

```
SQL> start p18q3
Por favor itroduzca el NUMERO del DEPARTAMENTO a MODIFICAR: 50
Por favor itroduzca la NUEVA LOCALIZACION del DEPARTAMENTO:
HOUSTON
antiguo   7:    v_numero := &dept_numero;
nuevo   7:      v_numero := 50;
antiguo  10:    SET loc = '&dept_loc'
nuevo  10:      SET loc = 'HOUSTON'
DEPTNO   DNAME   LOC
50  EDUCACION   HOUSTON
Procedimiento PL/SQL terminado correctamente.
```

18.4. Create a PL/SQL block that deletes the department created in exercise 2. Save your PL/SQL block to a file named *p18q4.sql*

a. Use a parameter for the department number.

b. Print to the screen the number of rows affected.

c. Test the PL/SQL block.

d. What happens if you enter a department number that does not exist?

e. Confirm that the department has been deleted.

```
ACCEPT dept_numero PROMPT 'Por favor itroduzca el NUMERO del
        DEPARTAMENTO a BORRAR: '

DECLARE
      v_numero dept.deptno%TYPE;
BEGIN
      v_numero := &dept_numero;
      DELETE  dept
      WHERE deptno = v_numero;
      dbms_output.put_line (' G_RESULT ');
      dbms_output.put_line ( TO_CHAR(SQL%ROWCOUNT) ||
           ‘ fila(s) borrada(s)’ );
END;
```

```
SQL> start p18q4
Por favor itroduzca el NUMERO del DEPARTAMENTO a BORRAR: 50
antiguo   5:    v_numero := &dept_numero;
nuevo   5:      v_numero := 50;
G_RESULT
1 fila(s) borrada(s)
Procedimiento PL/SQL terminado correctamente.
```

TEMA 19. ESTRUCTURAS DE CONTROL

19.1. Run the script /home/db/InformaciónGeneral/practicasdb2000/tema19/*lab19_1.sql* to create the MESSAGES table. Write a PL/SQL block to insert numbers into the MESSAGES table.

   a. Insert the numbers 1 to 10 excluding 6 and 8.
   b. Commit before the end of the block.
   c. SELECT from the MESSAGES table to verify that your PL/SQL block worked.

```
CREATE TABLE messages
 (results   VARCHAR2(60));
```

```
Tabla creada.
```

```
BEGIN
        FOR i IN 1..10 LOOP
                IF i <> 6 THEN
                        IF i <> 8 THEN
                                INSERT INTO messages
                                VALUES (i);
                        END IF;
                END IF;
        END LOOP;
        COMMI1T;
END;
/
SELECT *
FROM messages;
```

```
SQL> start p19q1;
Procedimiento PL/SQL terminado correctamente.
RESULTS
------------------------------------------------------------
1
2
3
4
5
7
9
10

8 filas seleccionadas.
```

19.2. Create a PL/SQL block that computes the commission amount for a given employee based on the employee's salary

a. Run the script /home/db/InformaciónGeneral/practicasdb2000/tema19/*lab19_2.sql* to insert a new employee into the EMP table. **Note:** The employee will have a NULL salary.
b. Accept the employee number as user input with a SQL*Plus substitution variable.
c. If the employee's salary is less than $1,000, set the commission amount for the employee to 10% of the salary.
d. If the employee's salary is between $1,000 and $1,500, set the commission amount for the employee to 15% of the salary.
e. If the employee's salary exceeds $1,500, set the commission amount for the employee to 20% of the salary.
f. If the employe's salary is NULL,set the commission amount for the employe to 0.
g. Commit.
h. Test the PL/SQL block for each case using the following test cases, and check each updated commission.

Employee Number	**Salary**	**Resulting Commission**
7369	800	80
7934	1300	195
7499	1600	320
8000	NULL	NULL

```
INSERT INTO emp
VALUES (8000, 'DOE', 'CLERK', 7698, SYSDATE, NULL, NULL, 10);
```

```
1 fila creada.
```

```
ACCEPT num_empleado PROMPT 'INSERTE el NUMERO
          de EMPLEADO: '
DECLARE
        v_num NUMBER(6);
        v_sal NUMBER(6);
BEGIN
        v_num := &num_empleado;
        SELECT NVL(sal, 0)
        INTO v_sal
        FROM emp
        WHERE empno = v_num;
        IF v_sal < 1000 THEN
                UPDATE emp
                SET comm = v_sal*0.1
                WHERE empno = v_num;
        ELSIF v_sal >= 1000 AND v_sal <= 1500 THEN
                UPDATE emp
                SET comm = v_sal*0.15
                WHERE empno = v_num;
                ELSIF v_sal > 1500 THEN
                UPDATE emp
                SET comm = v_sal*0.2
                WHERE empno = v_num;
        ELSIF v_sal = 0 THEN
                UPDATE emp
                SET comm = v_sal*0
                WHERE empno = v_num;
        END IF;
END;
/
COMMIT
```

```
 SQL> start p19q2 … (* 4)
 1  SELECT empno, ename, sal, comm
 2  FROM emp
 3* WHERE empno IN (8000,7499,7934,7369)

EMPNO ENAME            SAL      COMM
---------- ----------        ----------     ----------
     7369 SMITH             800        80
     7934 MILLER           1300       195
     7499 ALLEN            1600       320
     8000 DOE                 0
```

19.3. Modify *p16q4.sql* to insert the text “Number is odd” or “Number is even,” depending on whether the value is odd or even, into the MESSAGES table. Query the MESSAGES table to determine if your PL/SQL block worked.

```
VARIABLE resultado VARCHAR2(60)

DECLARE
        v_number NUMBER(3);
        v_valor_fila VARCHAR2(60);
BEGIN
        SELECT COUNT(*)
        INTO v_number
        FROM messages;
        dbms_output.put_line (v_number);
        IF MOD(v_number,2)=0 THEN
                :resultado := 'Numero par';
        ELSE
                :resultado := 'Numero impar';
        END IF;
END;
/
PRINT resultado
```

```
8
Procedimiento PL/SQL terminado correctamente.
RESULTADO
--------------------------------------------------------------------------------
Numero par
```

19.4. Add a new column to the EMP table for storing asterisk (*).

```
ALTER TABLE emp
ADD (asterisk VARCHAR2(30));
```

```
Tabla modificada.
```

19.5. Create a PL/SQL block that rewards an employee by appending an asterisk in the STARS column for every $100 of the employee's salary. Round the employee's salary to the nearest whole number. Save your PL/SQL block to a file called *p19q5.sql*

a. Accept the employee ID as user input with a SQL*Plus substitution variable.
b. Initialize a variable to contain a string of asterisks.
c. Append an asterisk to the string for every $100 of the salary amount. For example, if the employee has a salary amount of $800, the string of asterisks should contain eight asterisks.
d. Update the STARS column for the employee with the string of asterisks.
e. Commit.
f. Test the block for employees who have no salary and for an employee who has a salary.

```
ACCEPT emp_id PROMPT 'Introduzca el ID el EMPLEADO: '
DECLARE
        v_asterisk VARCHAR2(30);
        v_num NUMBER(6);
        v_sal NUMBER(6);
BEGIN
        SELECT NVL(sal,0)
        INTO v_sal
        FROM emp
        WHERE empno = &emp_id;
        v_num := TRUNC(v_sal / 100);
        IF v_num <> 0 THEN
                FOR i IN 1..v_num LOOP
                        v_asterisk := v_asterisk || '*';
                END LOOP;
        END IF;
        UPDATE emp
        SET asterisk = v_asterisk
        WHERE empno = &emp_id;
END;
/
COMMIT
```

```
SQL> start p19q3 … (* 2)
1  SELECT empno, sal, asterisk
2  FROM emp
3* WHERE empno IN (8000,7934)

  EMPNO       SAL       ASTERISK
 ----------  ----------  -----------------------------
   7934       1300      *************
   8000
```

TEMA 20. TRABAJANDO CON TIPOS COMPUESTOS.

20.1. Run the script /home/db/InformaciónGeneral/practicasdb2000/tema20/*lab20_1.sql* to create a new table for storing employees and their salaries.

```
CREATE TABLE top_dogs
 (name VARCHAR2(25), salary NUMBER(11,2))
```

```
Tabla creada.
```

20.2. Write a PL/SQL block to retrieve the name and salary of a given employee from the EMP table based on the employee's number, incorporate PL/SQL tables

a. Declare two PL/SQL tables, ENAME_TABLE and SAL_TABLE, to temporarily store the names and salaries.
b. As each name and salary is retrieved within the loop, store them in the PL/SQL tables.
c. Outside the loop, transfer the names and salaries from the PL/SQL tables into the TOP_DOGS table.
d. Empty the TOP_DOGS table and test the practice.

```
DECLARE
        TYPE ename_table_type IS TABLE OF emp.ename%TYPE
        INDEX BY BINARY_INTEGER;
        ename_table ename_table_type;
        TYPE sal_table_type IS TABLE OF emp.sal%TYPE
        INDEX BY BINARY_INTEGER;
        sal_table sal_table_type;
        temp_empno emp.empno%TYPE;
BEGIN
        temp_empno := &temporal;
        SELECT ename
        INTO ename_table(1)
        FROM emp
        WHERE empno = temp_empno;
        SELECT sal
        INTO sal_table(1)
        FROM emp
        WHERE empno = temp_empno;
        DBMS_OUTPUT.PUT_LINE('NAME      SALARY');
        DBMS_OUTPUT.PUT_LINE(ename_table(1) || '      ' ||
              TO_CHAR(sal_table(1)) );
END;
```

```
SQL> start p20q2
Introduzca un valor para temporal: 7934
antiguo  14:    temp_empno := &temporal;
nuevo  14:      temp_empno := 7934;
NAME    SALARY
MILLER  1300
Procedimiento PL/SQL terminado correctamente.
```

TEMA 21. CURSORES.

21.1. Create a PL/SQL block that determines the top employees with respect to salaries.

a. Accept a number *n* as user input with a SQL*Plus substitution parameter.
b. In a loop, get the last names and salaries of the top *n* people with respect to salary in the EMP table.
c. Store the names and salaries in the TOP_DOGS table.
d. Assume that no two employees have the same salary.
e. Test a variety of special cases, such a *n* = 0, where *n* is greater than the number of employees in the EMP table. Empty the TOP_DOGS table after each test.

```
DELETE FROM top_dogs;
ACCEPT numero_n PROMPT 'Inserte un valor: '

DECLARE
        CURSOR c1 IS
                SELECT empleados1.ename, empleados1.sal
                FROM emp empleados1
                WHERE  &numero_n >      (SELECT COUNT (*)
                                FROM emp empleados2
                                WHERE NVL(empleados1.sal,0) <
                                                empleados2.sal)
                ORDER BY empleados1.sal DESC;
        top_record c1%ROWTYPE;
BEGIN
        IF NOT c1%ISOPEN THEN
                OPEN c1;
        END IF;
        LOOP
                FETCH c1 INTO top_record;
                EXIT WHEN c1%NOTFOUND;
                INSERT INTO top_dogs
                VALUES (top_record.ename, top_record.sal);
                -- DBMS_OUTPUT.PUT_LINE(top_record.ename || '      ' ||
                        TO_CHAR(top_record.sal) );
        END LOOP;
        CLOSE c1;
END;
/

SELECT *
FROM top_dogs;
```

```
SQL> start p21q1
3 filas borradas.
Inserte un valor: 5
antiguo   5:    WHERE  &numero_n >      (SELECT COUNT (*)
nuevo   5:      WHERE  5 >      (SELECT COUNT (*)
Procedimiento PL/SQL terminado correctamente.
NAME                      SALARY
------------------------- ----------
KING                       5000
FORD                       3000
SCOTT                      3000
JONES                      2975
BLAKE                      2850
```

21.2. Consider the case where several employees have the same salary. If one person is listed, then all people who have the same salary should also be listed.

a. For example, if the user enters a value of 2 for *n*, then King, Ford and Scott should be displayed. (These employees are tied for second highest salary.)
b. If the user enters a value of 3, then King, Ford, Scott, and Jones should be displayed.
c. Delete all rows from TOP_DOGS and test the practice.

```
DELETE FROM top_dogs;
ACCEPT numero_n PROMPT 'Inserte un valor: '

DECLARE
        CURSOR c1 IS
        SELECT DISTINCT empleados1.sal
        FROM emp empleados1
        WHERE  &numero_n > (SELECT COUNT
                         (DISTINCT(empleados2.sal))
                        FROM emp empleados2
                        WHERE NVL(empleados1.sal,0) < empleados2.sal)
        ORDER BY empleados1.sal DESC;
        CURSOR c2 IS
                SELECT empleados3.ename, empleados3.sal
                FROM emp empleados3
                ORDER BY empleados3.sal DESC;
        top_record c1%ROWTYPE;
        top_record2 c2%ROWTYPE;
```

```
BEGIN
        IF NOT c2%ISOPEN THEN
        OPEN c2;
END IF;
        -- recorremos la lista mayor con el cursor2, y cada vez que leamos
            un elemento de esta lista
        -- veremos si tal elemento está también la lista con las 'x' distintos
            sueldos mayores
        LOOP
               FETCH c2 INTO top_record2;
               EXIT WHEN c2%NOTFOUND;
               IF NOT c1%ISOPEN THEN
                      OPEN c1;
               END IF;
               LOOP
                      FETCH c1 INTO top_record;
                      EXIT WHEN c1%NOTFOUND;
                      IF top_record2.sal = top_record.sal THEN
                             INSERT INTO top_dogs
                             VALUES (top_record2.ename, top_record2.sal);
                      END IF;
               END LOOP;
               CLOSE c1;
        END LOOP;
        CLOSE c2;
END;
/
SELECT * FROM top_dogs
```

```
SQL> start p21q2
3 filas borradas.
Inserte un valor: 3
antiguo 5: WHERE  &numero_n > (SELECT COUNT
                                              (DISTINCT(empleados2.sal))
nuevo 5:  WHERE  3 > (SELECT COUNT (DISTINCT(empleados2.sal))
Procedimiento PL/SQL terminado correctamente.
NAME                    SALARY
------------------------ ----------
KING                     5000
FORD                     3000
SCOTT                    3000
JONES                    2975
```

TEMA 22. CURSORES EXPLÍCITOS AVANZADOS.

22.1. Write a query to retrieve all the departments and the employees in each department. Insert the results in the MESSAGES table. Use a cursor to retrieve the department number and pass the department number to a cursor to retrieve the employees in that department.

```
DELETE FROM messages;

DECLARE
        CURSOR c1 IS
                SELECT DISTINCT deptno
                FROM dept;
        CURSOR c2 (v_deptno dept.deptno%TYPE) IS
                SELECT ename
                FROM emp
                WHERE deptno = v_deptno;
        temp_deptno dept.deptno%TYPE;
        temp_ename emp.ename%TYPE;
BEGIN
        IF NOT c1%ISOPEN THEN
                OPEN c1;
        END IF;
        LOOP
                FETCH c1 INTO temp_deptno;
                EXIT WHEN c1%NOTFOUND;
                IF NOT c2%ISOPEN THEN
                        OPEN c2 (temp_deptno);
                END IF;
                LOOP
                        FETCH c2 INTO temp_ename;
                        EXIT WHEN c2%NOTFOUND;
                        INSERT INTO messages (results)
                        VALUES (CONCAT (temp_ename, CONCAT
                                (' - Department ', TO_CHAR(temp_deptno))));
                END LOOP;
                CLOSE c2;
        END LOOP;
        CLOSE c1;
END;
/

SELECT *
FROM messages
```

```
SQL> start p22q1
8 filas borradas.
Procedimiento PL/SQL terminado correctamente.
RESULTS
------------------------------------------------------------
KING - Department 10
CLARK - Department 10
MILLER - Department 10
DOE - Department 10
JONES - Department 20
FORD - Department 20
SMITH - Department 20
SCOTT - Department 20
ADAMS - Department 20
BLAKE - Department 30
MARTIN - Department 30
ALLEN - Department 30
TURNER - Department 30
JAMES - Department 30
WARD - Department 30

15 filas seleccionadas.
```

22.2. Modify *p19q5.sql* to incorporate the FOR UPDATE and WHERE CURRENT OF functionality in cursor processing.

```
ACCEPT p_empno PROMPT 'Introduzca el numero de empleado: '

DECLARE
        v_empno emp.empno%TYPE := &p_empno;
        v_asterisk emp.asterisk%TYPE := NULL;
        CURSOR emp_cursor IS
                SELECT empno, NVL(ROUND(sal/100), 0) sal
                FROM emp
                WHERE empno = v_empno
                FOR UPDATE;
BEGIN
        FOR emp_record IN emp_cursor LOOP
        BEGIN
                FOR i IN 1..emp_record.sal LOOP
                        v_asterisk := v_asterisk || '*';
                END LOOP;
                UPDATE emp
                SET asterisk = v_asterisk
                WHERE CURRENT OF emp_cursor;
                v_asterisk := NULL;
        END;
        END LOOP;
        COMMIT;
END;
/
SELECT empno, sal, asterisk
FROM emp
WHERE empno = &p_empno;
```

```
SQL> start p22q2
Introduzca el numero de empleado: 7900
antiguo   2:    v_empno emp.empno%TYPE := &p_empno;
nuevo   2:      v_empno emp.empno%TYPE := 7900;
Procedimiento PL/SQL terminado correctamente.
antiguo   3: WHERE empno = &p_empno
nuevo   3: WHERE empno = 7900
     EMPNO        SAL ASTERISK
---------- ---------- ------------------------------
      7900        950 **********
```

TEMA 23. MANEJO DE EXCEPCIONES.

23.1. Write a PL/SQL block to SELECT the name of the employee with a given salary value.

a. If the salary entered returns more than one row, handle the exception with an appropriate exception handler and insert into the MESSAGES table, the message “More than one employee with a salary of <*salary*>.”
b. If the salary entered does not return any rows, handle the exception with an appropriate exception handler and insert into the MESSAGES table, the message “No employee with a salary of <*salary*>.”
c. If the salary entered returns only one row, insert into the MESSAGES table the employee’s name and the salary amount.
d. Handle any other exception with an appropriate exception handler and insert into the MESSAGES table, the message “Some other error occurred.”
e. Test the block for a variety of test cases.

```
DELETE FROM messages;

ACCEPT emp_sal PROMPT 'Introduzca un salario: '
DECLARE
        v_ename emp.ename%TYPE := NULL;
        data_found EXCEPTION;
BEGIN
        SELECT ename
        INTO v_ename
        FROM emp
        WHERE sal = &emp_sal;
        RAISE data_found;
EXCEPTION
        WHEN TOO_MANY_ROWS THEN
                INSERT INTO messages (results)
                VALUES ('Mas de un empleado con el salario &emp_sal');
        WHEN NO_DATA_FOUND THEN
                INSERT INTO messages (results)
                VALUES ('No hay empleados con el salario &emp_sal');
        WHEN data_found THEN
                INSERT INTO messages (results)
                VALUES (v_ename || ' - ' || &emp_sal);
        WHEN OTHERS THEN
                INSERT INTO messages (results)
        VALUES ('Algun otro erroe ha ocurrido');
END;
/
SELECT *
FROM messages;
```

```
SQL> start p23q1
0 filas borradas.
Procedimiento PL/SQL terminado correctamente.
RESULTS
-----------------------------------------------------------
SMITH - 800
```

23.2. Modify *p18q3.sql* to add an exception handler

a. Write an exception handler for the error to pass a message to the user that the specified department does not exist. Execute the PL/SQL block by entering a department that does not exist.

```
ACCEPT dept_numero PROMPT 'Por favor itroduzca el NUMERO del
                DEPARTAMENTO a MODIFICAR: '
ACCEPT dept_loc PROMPT 'Por favor itroduzca la NUEVA
                LOCALIZACION del DEPARTAMENTO: '
DECLARE
        v_numero dept.deptno%TYPE;
        v_localizacion dept.loc%TYPE;
        v_nombre dept.dname%TYPE;
BEGIN
        v_numero := &dept_numero;
        UPDATE  dept
        SET loc = '&dept_loc' WHERE deptno = v_numero;
        SELECT deptno, dname, loc
        INTO v_numero, v_nombre, v_localizacion
        FROM dept
        WHERE deptno = v_numero;
        dbms_output.put_line ('DEPTNO   DNAME   LOC');
        dbms_output.put_line (TO_CHAR(v_numero) ||'  '|| v_nombre ||'   '||
                        v_localizacion);
EXCEPTION
                WHEN NO_DATA_FOUND THEN
                        dbms_output.put_line ('No existe tal departamento');
END;
```

```
SQL> start p23q2
Por favor itroduzca el NUMERO del DEPARTAMENTO a MODIFICAR: 50
Por favor itroduzca la NUEVA LOCALIZACION del DEPARTAMENTO:
HOUSTON
antiguo   7:    v_numero := &dept_numero;
nuevo   7:      v_numero := 50;
antiguo  10:     SET loc = '&dept_loc'
nuevo  10:       SET loc = 'HOUSTON'
Procedimiento PL/SQL terminado correctamente.
```

23.3. Write a PL/SQL block that prints the names of the employees who make plus or minus $100 of the salary value entered

a. If there is no employee within that salary range, print a message to the user indicating that is the case. Use an exception for this case.
b. If there are one or more employees within that range, the message should indicate how many employees are in that salary range.
c. Handle any other exception with an appropriate exception handler, the message should indicate that some other error occurred.

```
ACCEPT salary PROMPT 'Introduzca salario: '
DECLARE
        CURSOR c1 IS
                SELECT sal
                FROM emp
                WHERE sal BETWEEN &salary-100 AND &salary+100;
        nada_seleccionado EXCEPTION;
        algo_seleccionado EXCEPTION;
        v_sal emp.sal%TYPE;
        v_numero_empleados NUMBER(6) := 0;
BEGIN
        OPEN c1;
        LOOP
                FETCH c1 INTO v_sal;
                EXIT WHEN c1%NOTFOUND;
                v_numero_empleados := v_numero_empleados + 1;
        END LOOP;
        CLOSE c1;
        IF v_numero_empleados = 0 THEN
                RAISE nada_seleccionado;
        ELSE
                RAISE algo_seleccionado;
        END IF;
EXCEPTION
        WHEN algo_seleccionado THEN
                dbms_output.put_line ('Hay ' || v_numero_empleados  ||
                        ' empleado(s) con un salario entre ' || (&salary-100) ||
                        ' y ' || (&salary+100) );
        WHEN nada_seleccionado THEN
                dbms_output.put_line('No hay empleado(s) con un salario entre '
                        || (&salary-100) || ' y ' || (&salary+100) );
        WHEN OTHERS THEN
                dbms_output.put_line ('Algun otro error a ocurrido');
END;
```

```
SQL> start p23q3
Introduzca salario: 3000
Hay 3 empleado(s) con un salario entre 2900 y 3100
Procedimiento PL/SQL terminado correctamente.
```

TEMA 24. OPERADORES DE CONJUNTOS.

24.1. Display the department that has no employees.

```
SELECT deptno, dname
FROM dept
MINUS
SELECT emp.deptno, dept.dname
FROM emp, dept
WHERE emp.deptno = dept.deptno;
```

```
DEPTNO DNAME
 ---------- --------------
        40 OPERATIONS
```

24.2. Find the job that was filled in the last half of 1981 and the same job that was filled during the same period in 1982.

```
SELECT job
FROM emp
WHERE hiredate >= TO_DATE('July 1, 81', 'Month dd, YY')
        AND hiredate <= TO_DATE('December 31, 81', 'Month dd, YY')
INTERSECT
SELECT job
FROM emp
WHERE hiredate >= TO_DATE('July 1, 82', 'Month dd, YY')
        AND hiredate <= TO_DATE('December 31, 82', 'Month dd, YY');
```

```
JOB
---------
ANALYST
```

24.3. Write a compound query to produce a list of products showing discount percentages, product id, and old and new actual price. Products under $10 are reduced by 10%, products between $10 and $30 are reduced by 15%, products over $30 are reduced by 20%, and products over $40 are not reduced at all.

```
SELECT DISTINCT '10%off' DISCOUNT, prodid, actualprice,
            (actualprice - 0.1*actualprice) STDPRICE
FROM item WHERE actualprice < 10
UNION
SELECT DISTINCT '15%off' DISCOUNT, prodid, actualprice,
            (actualprice - 0.15*actualprice) STDPRICE
FROM item WHERE actualprice >= 10 AND actualprice <= 30
UNION
SELECT DISTINCT '20%off' DISCOUNT, prodid, actualprice,
            (actualprice - 0.2*actualprice) STDPRICE
FROM item WHERE actualprice > 30 AND actualprice <= 40
UNION
SELECT DISTINCT 'no disc' DISCOUNT, prodid, actualprice,
            actualprice STDPRICE
FROM item WHERE actualprice > 40;
```

```
DISCOUN    PRODID ACTUALPRICE   STDPRICE
------- ---------- ----------- ----------
10%off      100871           5       4,5
10%off      101863           9       8,1
10%off      200380           4       3,6
15%off      100860          30      25,5
15%off      100870          25     21,25
15%off      100870          28      23,8
15%off      101860          24      20,4
15%off      101863          10       8,5
15%off      200376          22      18,7
15%off      200376          24      20,4
20%off      100860          35        28
20%off      100861          35        28
20%off      102130          34      27,2
no disc     100860          44        44
no disc     100860          56        56
no disc     100861          42        42
no disc     100861          45        45
no disc     100861         405       405
no disc     100861        4511      4511
no disc     100870          45        45
no disc     100871          55        55
no disc     100871          56        56
no disc     100890          50        50
no disc     100890          58        58
no disc     101863         125       125
25 filas seleccionadas.
```

24.4. Produce a list of jobs for departments 10, 30, and 20 in that order. Display job and department number.

```
SELECT DISTINCT job, deptno
FROM emp WHERE deptno = 10
UNION ALL
SELECT DISTINCT job, deptno
FROM emp WHERE deptno = 30
UNION ALL
SELECT DISTINCT job, deptno
FROM emp WHERE deptno = 20;
```

```
JOB           DEPTNO
--------- ----------
CLERK             10
MANAGER           10
PRESIDENT         10
CLERK             30
MANAGER           30
SALESMAN          30
ANALYST           20
CLERK             20
MANAGER           20

9 filas seleccionadas.
```

24.5. List the department number for departments without the job title ANALYST.

```
SELECT deptno FROM dept
MINUS
SELECT deptno FROM emp WHERE job = 'ANALYST'
```

```
    DEPTNO
----------
        10
        30
        40
```

24.6. List all job titles in department 10 and 20 that do not occur in both departments.

```
SELECT job FROM emp WHERE deptno = 10 OR deptno =20
MINUS
  (SELECT job FROM emp WHERE deptno = 10
   INTERSECT
   SELECT job FROM empWHERE deptno = 20)
```

```
JOB
---------
ANALYST
PRESIDENT
```

TEMA 25. SUBCONSULTAS CORRELACIONADAS

25.1. Write a query to display the top three earners in the EMP table. Display their names and salaries.

```
SELECT empleados1.ename "Nombre", empleados1.sal  "Salary"
FROM emp empleados1
WHERE  3 > (SELECT COUNT (*)
                FROM emp empleados2
                WHERE NVL(empleados1.sal,0) < empleados2.sal)
```

```
Nombre      Salary
----------  ----------
KING          5000
FORD          3000
SCOTT         3000
```

25.2. Find all employees who are not a supervisor

a. Do this using the EXISTS operator first.
b. Can this be done using the IN operator? Why, or why not?

```
SELECT empleados1.ename "Nombre"
FROM emp empleados1
WHERE NOT EXISTS (SELECT empno
                        FROM emp empleados2
                        WHERE empleados1.empno = empleados2.mgr)
```

```
Nombre
----------
MARTIN
ALLEN
TURNER
JAMES
WARD
SMITH
ADAMS
MILLER
DOE

9 filas seleccionadas.
```

```
SELECT ename "Nombre"
FROM emp
WHERE empno NOT IN (SELECT NVL(mgr,0)
                                        FROM emp)
```

```
Nombre
----------
MARTIN
ALLEN
TURNER
JAMES
WARD
SMITH
ADAMS
MILLER
DOE

9 filas seleccionadas.
```

25.3. Write a query to find all employees who make more than the average salary in their department. Display employee number, salary, department number, and the average salary for the department. Sort by average salary.

```
SELECT empleados1.ename "NOMBRE", empleados1.sal "SALARY",
          empleados1.deptno "DEPTNO", AVG(empleados3.sal) "DEPT_AVG"
FROM emp empleados1, emp empleados3
WHERE empleados1.sal > (SELECT AVG(empleados2.sal)
                                         FROM emp empleados2
                                         WHERE empleados1.deptno = empleados2.deptno)
             AND empleados1.deptno = empleados3.deptno
GROUP BY empleados1.ename, empleados1.sal, empleados1.deptno
ORDER BY AVG(empleados3.sal)
```

```
NOMBRE         SALARY     DEPTNO   DEPT_AVG
---------- ---------- ---------- ----------
ALLEN            1600         30 1566,66667
BLAKE            2850         30 1566,66667
JONES            2975         20       2175
FORD             3000         20       2175
SCOTT            3000         20       2175
KING             5000         10 2916,66667

6 filas seleccionadas.
```

25.4. Write a query to display employees who earn less than half the average salary in their department.

```
SELECT empleados1.ename "Nombre"
FROM emp empleados1
WHERE sal < (SELECT AVG(sal)/2
             FROM emp empleados2
             WHERE empleados1.deptno = empleados2.deptno)
```

```
Nombre
----------
SMITH
MILLER
```

25.5. Write a query to display employees who have one or more co-workers in their department with later hiredates but higher salaries.

```
SELECT empleados1.ename "NOMBRE"
FROM emp empleados1
WHERE 1 <= (SELECT COUNT (*)
             FROM emp empleados2
             WHERE empleados1.deptno = empleados2.deptno
                   AND empleados1.hiredate < empleados2.hiredate
                   AND empleados1.sal < empleados2.sal)
```

```
NOMBRE
----------
CLARK
JONES
ALLEN
WARD
SMITH
```

ANEXO I. MATERIAL ELECTRÓNICO.