



## **Práctica Bases de Datos Avanzadas**

### **Consultas Avanzadas**

Breidy Núñez | 2017-5633

Felipe Ramírez | 2018-6104

Luis Franco | 2018-6342

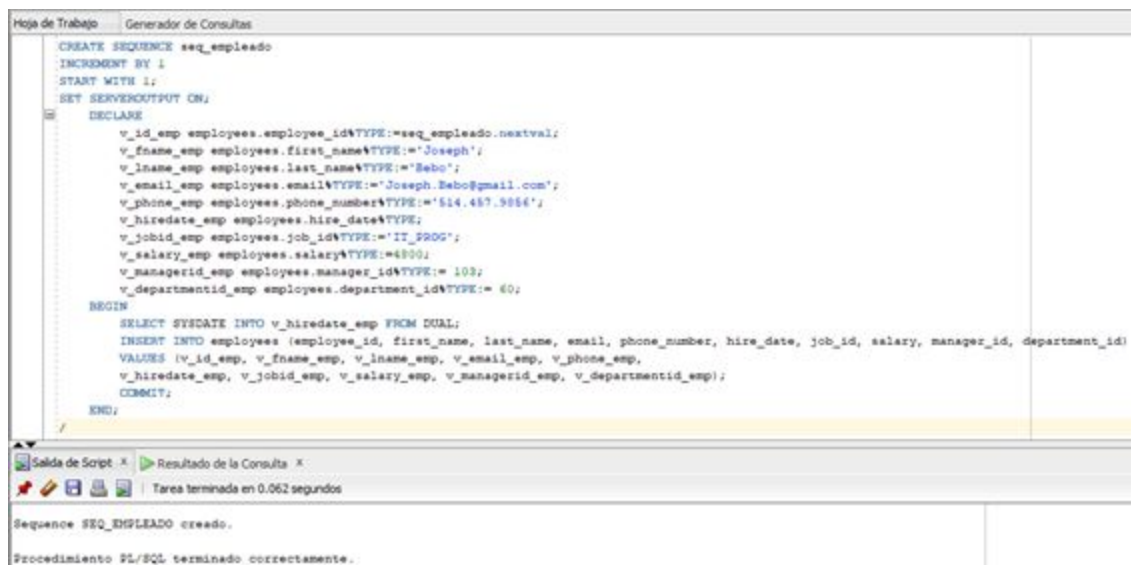
**Prof. Leandro Fondeur**

18/7/2019

I. Realice los siguientes ejercicios luego de estudiar el documento D64254GC11\_les05.ppt:

5.1 Construya un código de PL/SQL donde se inserte un nuevo registro en la tabla de empleados. El ID del empleado debe llenarse usando la secuencia creada para los fines. Genere una secuencia y asigne el valor directamente a una variable, tal como lo permite Oracle 11g. Recuerde asentar los cambios a la BD dentro del bloque de código.

ID del empleado: Secuencia // Nombre: Joseph // Apellido: Bebo  
Email: Joseph.Bebo@gmail.com // Teléfono: 514-457-98565-  
Fecha Ingreso: Fecha Actual (use la función) // Puesto: IT\_PROG  
Salario: 4800 // Comisión: Ignorar // ID del Supervisor: 103  
ID del Departamento: 60



```
Hoja de Trabajo  Generador de Consultas

CREATE SEQUENCE seq_empleado
INCREMENT BY 1
START WITH 1;
SET SERVEROUTPUT ON;

DECLARE
v_id_emp employees.employee_id%TYPE:=seq_empleado.nextval;
v_fname_emp employees.first_name%TYPE:='Joseph';
v_lname_emp employees.last_name%TYPE:='Bebo';
v_email_emp employees.email%TYPE:='Joseph.Bebo@gmail.com';
v_phone_emp employees.phone_number%TYPE:='514.457.9856';
v_hiredate_emp employees.hire_date%TYPE;
v_jobid_emp employees.job_id%TYPE:='IT_PROG';
v_salary_emp employees.salary%TYPE:=4800;
v_managerid_emp employees.manager_id%TYPE:= 103;
v_departmentid_emp employees.department_id%TYPE:= 60;

BEGIN
SELECT SYSDATE INTO v_hiredate_emp FROM DUAL;
INSERT INTO employees (employee_id, first_name, last_name, email, phone_number, hire_date, job_id, salary, manager_id, department_id)
VALUES (v_id_emp, v_fname_emp, v_lname_emp, v_email_emp, v_phone_emp,
v_hiredate_emp, v_jobid_emp, v_salary_emp, v_managerid_emp, v_departmentid_emp);
COMMIT;

END;
```

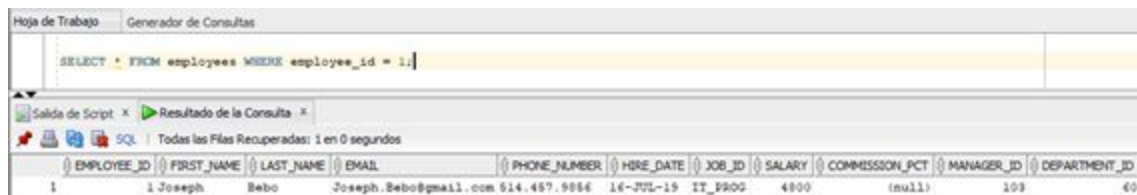
Salida de Script x Resultado de la Consulta x

Tarea terminada en 0.062 segundos

Sequence SEQ\_EMPLEADO creado.

Procedimiento PL/SQL terminado correctamente.

Fuera del bloque realice una consulta para mostrar el registro insertado.



```
Hoja de Trabajo  Generador de Consultas

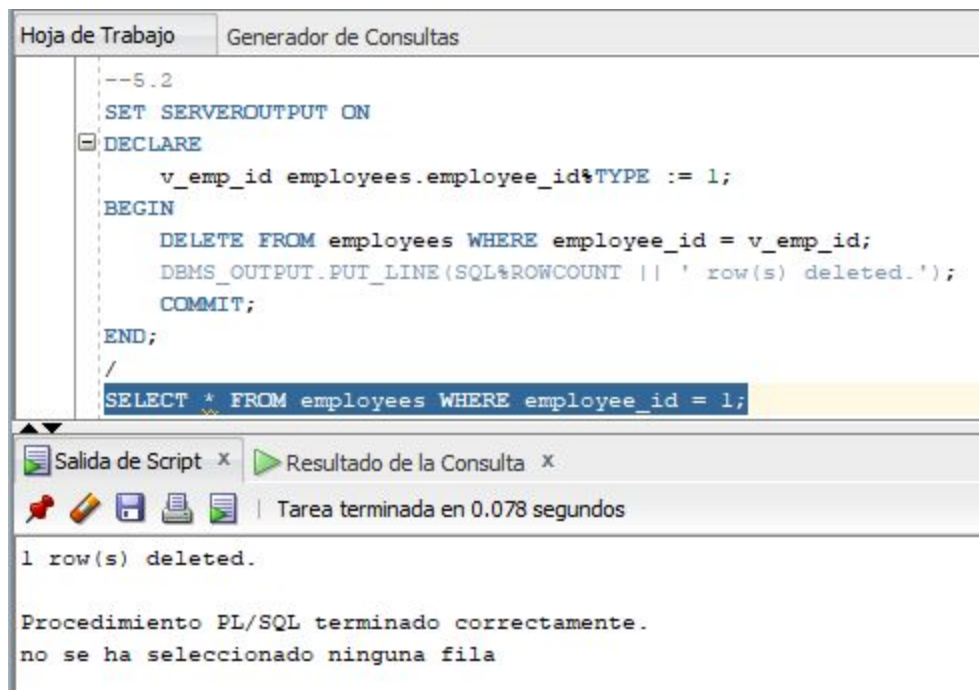
SELECT * FROM employees WHERE employee_id = 1;
```

Salida de Script x Resultado de la Consulta x

Todas las Filas Recuperadas: 1 en 0 segundos

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID
1	Joseph	Bebo	Joseph.Bebo@gmail.com	514.457.9856	16-JUL-19	IT_PROG	4800	(null)	103	60

5.2 Construya un código PL/SQL que le pase un valor a una variable en la sección declarativa y que en el bloque del programa use dicho valor para eliminar un registro basado en el ID del empleado. Puede usar como valor el ID del empleado creado en el ejercicio 5.1. Muestre por pantalla la cantidad de registros borrados usando el atributo de cursor SQL correspondiente. Recuerde asentar los cambios. Fuera del bloque PL/SQL haga una consulta para validar que el empleado se haya borrado exitosamente.



The screenshot shows the Oracle SQL Developer interface. The top pane, titled 'Hoja de Trabajo' and 'Generador de Consultas', contains the following PL/SQL code:

```
--5.2
SET SERVEROUTPUT ON
DECLARE
  v_emp_id employees.employee_id%TYPE := 1;
BEGIN
  DELETE FROM employees WHERE employee_id = v_emp_id;
  DBMS_OUTPUT.PUT_LINE(SQL%ROWCOUNT || ' row(s) deleted. ');
  COMMIT;
END;
/
SELECT * FROM employees WHERE employee_id = 1;
```

The bottom pane shows the execution results. It includes a toolbar with icons for saving, printing, and other actions. The status bar indicates 'Tarea terminada en 0.078 segundos'. The output area displays the following messages:

```
1 row(s) deleted.






Procedimiento PL/SQL terminado correctamente.
no se ha seleccionado ninguna fila
```

5.3 Construya un código PL/SQL que asigne un monto a una variable y actualice los salarios sumando a los mismos el valor introducido en la variable, de todos los empleados que trabajan en la posición IT\_PROG. Asigne la cantidad de registros actualizados a una variable y muestre por pantalla el valor de dicha variable.

```
set SERVEROUTPUT on
declare
v_monto employees.salary%TYPE :=500;
v_rows_updated varchar(30);
begin
update employees
set salary= salary + v_monto
where job_id='IT_PROG';
v_rows_updated :=(sql%rowcount|| ' Registros actualizados' );
dbms_output.put_line(v_rows_updated);
end;
```

Salida de Script x

Resultado de la Consulta x

     | Tarea terminada en 0.069 segundos

5 Registros actualizados

Procedimiento PL/SQL terminado correctamente.

5.4 Construya un código PL/SQL que asigne un valor a una variable y realice una consulta por ID del empleado usando dicha variable. Declare la variable con el mismo nombre de la columna de ID del empleado. Diga cuál es el resultado de la corrida del programa y explique lo que ocurra

```
declare
employee_id employees.employee_id%TYPE := 100;
first_name varchar(25);
begin
    select first_name into first_name
    from employees
    where employee_id = employee_id;

end;
```

Salida de Script x Resultado de la Consulta x

Tarea terminada en 0.052 segundos

Informe de error -  
ORA-01422: exact fetch returns more than requested number of rows  
ORA-06512: at line 5  
01422. 00000 - "exact fetch returns more than requested number of rows"  
\*Cause: The number specified in exact fetch is less than the rows returned.  
\*Action: Rewrite the query or change number of rows requested

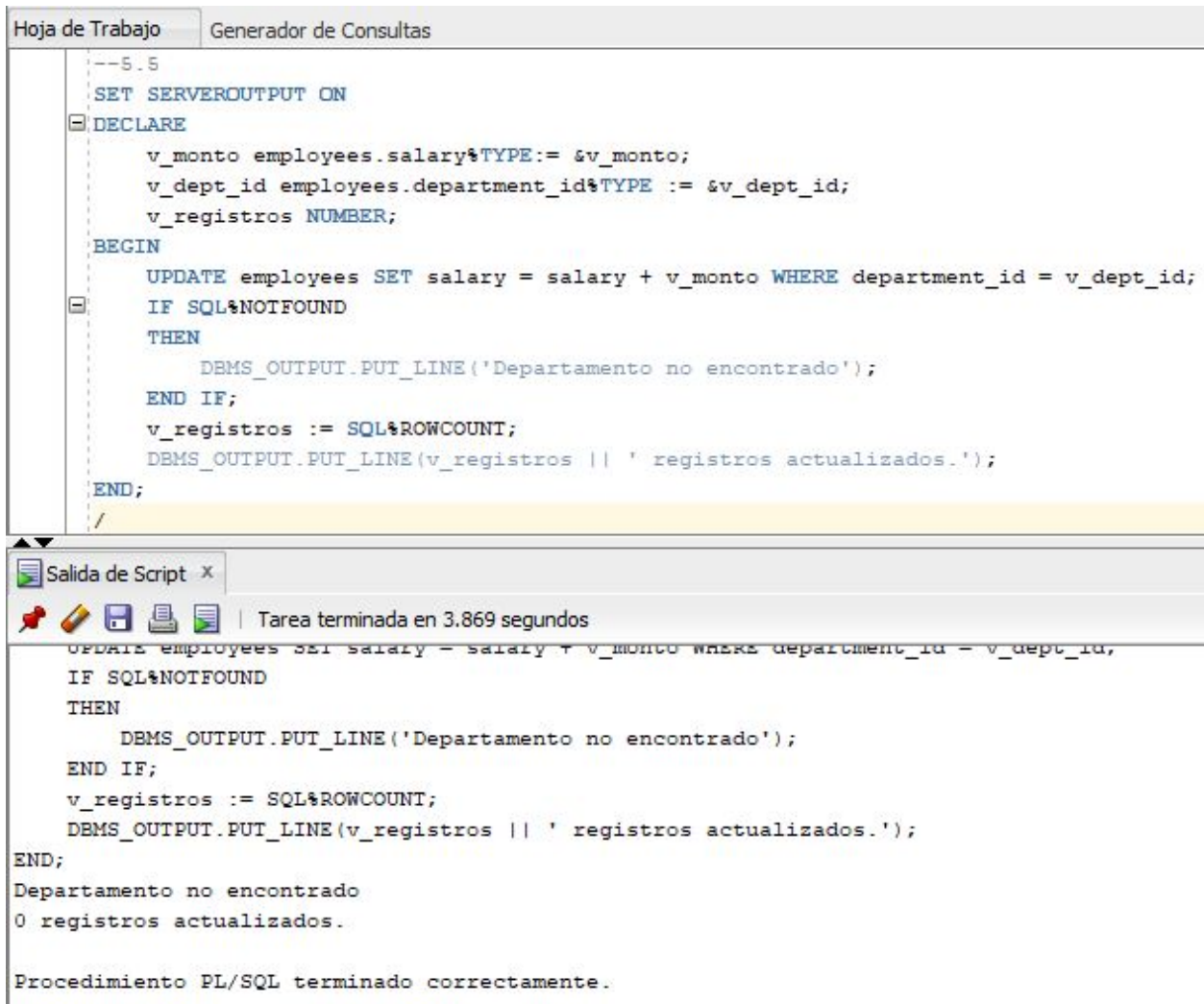
*El problema aquí está en la ambigüedad nombrando las variables. PL/SQL tiene ventajas y desventajas ya que después del SELECT sabe que le hablamos de una columna; después del INTO le hablamos de una variable, pero el punto está en el WHERE que primero analiza las columnas y después las variables. Por eso en (employees\_id = employees\_id) estás comparando lo mismo y la condición se cumplirá en todos los casos y devolverá más registros de los necesarios.*

5.5 Construya un código PL/SQL que pida un monto y un ID de departamento, y actualice los salarios sumando a los mismos el valor introducido, de todos los empleados que trabajan en el ID de departamento dado. Asigne la cantidad de registros actualizados a una variable. Use SQL%NOTFOUND para validar si haciendo la actualización no se llegó a actualizar ningún registro. Si no actualizó ningún registro, muestre el siguiente mensaje:

Departamento no encontrado

En cualquier escenario muestre el siguiente mensaje por pantalla:

<<cantidad de registros>> registros actualizados



The screenshot displays a software interface with two main panels. The top panel, titled 'Hoja de Trabajo' and 'Generador de Consultas', contains a PL/SQL script. The script starts with a comment '--5.5', followed by 'SET SERVEROUTPUT ON'. It then declares three variables: 'v\_monto' of type 'employees.salary%TYPE' assigned to '&v\_monto;', 'v\_dept\_id' of type 'employees.department\_id%TYPE' assigned to '&v\_dept\_id;', and 'v\_registros' of type 'NUMBER;'. The 'BEGIN' block contains an 'UPDATE employees SET salary = salary + v\_monto WHERE department\_id = v\_dept\_id;' statement, followed by an 'IF SQL%NOTFOUND THEN' condition. Inside the 'THEN' block, 'DBMS\_OUTPUT.PUT\_LINE('Departamento no encontrado');' is called. After 'END IF;', 'v\_registros := SQL%ROWCOUNT;' is assigned, and 'DBMS\_OUTPUT.PUT\_LINE(v\_registros || ' registros actualizados.');" is called. The script ends with 'END;' and a forward slash '/'.

The bottom panel, titled 'Salida de Script x', shows the execution results. It includes a status bar indicating 'Tarea terminada en 3.869 segundos'. The output text is as follows:

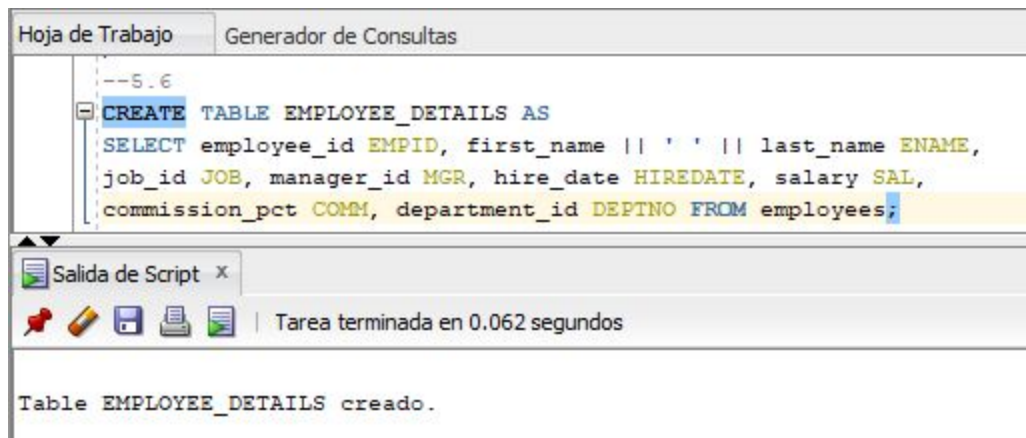
```
UPDATE employees SET salary = salary + v_monto WHERE department_id = v_dept_id,
IF SQL%NOTFOUND
THEN
    DBMS_OUTPUT.PUT_LINE('Departamento no encontrado');
END IF;
v_registros := SQL%ROWCOUNT;
DBMS_OUTPUT.PUT_LINE(v_registros || ' registros actualizados.');
```

Departamento no encontrado  
0 registros actualizados.

Procedimiento PL/SQL terminado correctamente.

5.6 Construya un código PL/SQL que lea información de una tabla y si los registros existen en otra tabla que actualice los valores en ésta, de lo contrario que inserte el registro. Para lograr el objetivo, realice las siguientes actividades:

1) Crear la tabla EMPLOYEE\_DETAILS usando un sub-query de la tabla EMPLOYEES, seleccionando el ID del empleado (EMPID), el nombre concatenado al apellido con un espacio de por medio (ENAME), el ID de la posición (JOB), el ID del supervisor (MGR), la fecha de contratación (HIREDATE), el salario (SAL), la comisión (COMM) y el ID del departamento (DEPTNO).



The screenshot shows the SQL Developer interface with the 'Hoja de Trabajo' (Worksheet) tab active. The SQL editor contains the following code:

```
--5.6
CREATE TABLE EMPLOYEE_DETAILS AS
SELECT employee_id EMPID, first_name || ' ' || last_name ENAME,
       job_id JOB, manager_id MGR, hire_date HIREDATE, salary SAL,
       commission_pct COMM, department_id DEPTNO FROM employees;
```

Below the editor, the 'Salida de Script' (Script Output) window shows the message: 'Tarea terminada en 0.062 segundos' (Task completed in 0.062 seconds). At the bottom, a status bar indicates: 'Table EMPLOYEE\_DETAILS creado.' (Table EMPLOYEE\_DETAILS created).

2) Insertar un nuevo registro en la tabla de empleados con los siguientes valores:

ID del empleado [usar la secuencia], Nombre [Luis], Apellido [Ross],  
Correo [lross@gmail.com], Teléfono [809-222-5425], Fecha [fecha del sistema],  
Posición [IT\_PROG], Salario [4800], Comisión [NULL], ID del supervisor [103],  
ID del departamento [60]

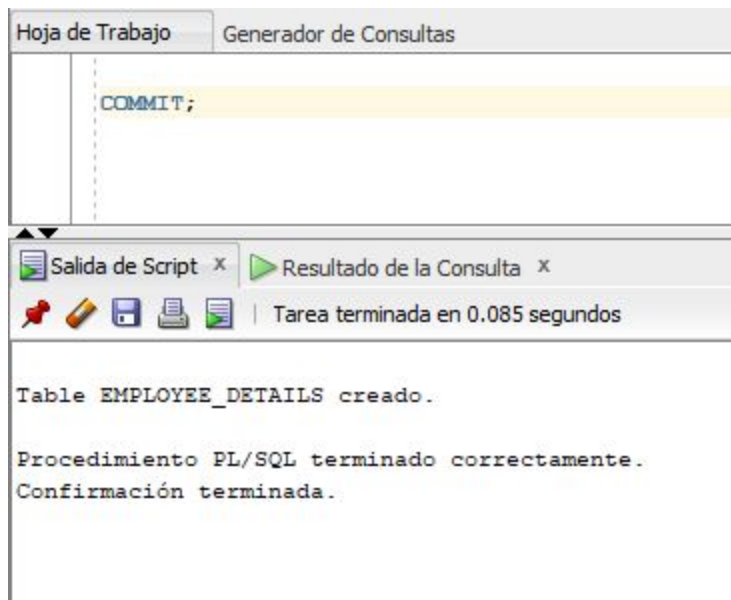


The screenshot shows the SQL Developer interface with the 'Hoja de Trabajo' (Worksheet) tab active. The SQL editor contains the following PL/SQL code:

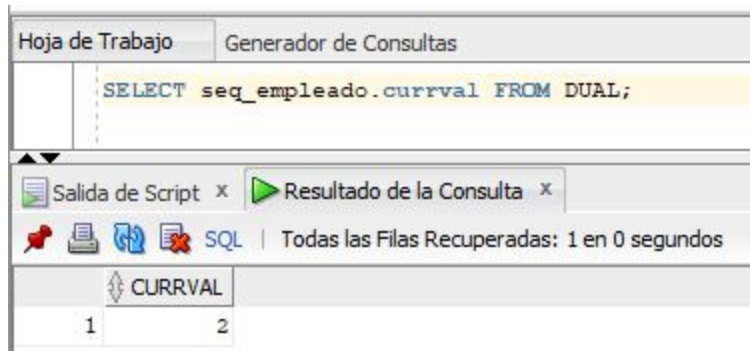
```
DECLARE
v_sequence NUMBER := seq_empleado.nextval;
v_date DATE;
BEGIN
SELECT SYSDATE INTO v_date FROM DUAL;
INSERT INTO employees
VALUES (v_sequence, 'Luis', 'Ross', 'lross@gmail.com', '809.222.5425', v_date, 'IT_PROG', 4800, null, 103, 60);
END;
```

Below the editor, the 'Salida de Script' (Script Output) window shows the message: 'Tarea terminada en 0.069 segundos' (Task completed in 0.069 seconds). At the bottom, a status bar indicates: 'Table EMPLOYEE\_DETAILS creado.' (Table EMPLOYEE\_DETAILS created) and 'Procedimiento PL/SQL terminado correctamente.' (PL/SQL procedure terminated successfully).

3) Asentar los cambios en la BD.



4) Revisar la secuencia actual que se usó para insertar el valor en el ID del empleado.





5) Revisar si existe un registro en la tabla EMPLOYEE\_DETAILS con el ID del empleado igual al número de secuencia retornado en el paso 4.

Hoja de Trabajo

Generador de Consultas

6) Construir un código PL/SQL que permita actualizar la tabla EMPLOYEE\_DETAILS con los nuevos cambios realizados en la tabla EMPLOYEES, mediante un MERGE.

Hoja de Trabajo		Generador de Consultas
<pre> BEGIN MERGE INTO employee_details ed USING employees e ON (e.employee_id = ed.EMPID) WHEN MATCHED THEN UPDATE SET ed.ENAME = e.first_name    ' '    e.last_name, ed.JOB = e.job_id, ed.MGR = e.manager_id, ed.HIREDATE = e.hire_date, ed.SAL = e.salary, ed.COMM = e.commission_pct, ed.DEPTNO = e.department_id WHEN NOT MATCHED THEN INSERT VALUES (e.employee_id, e.first_name    ' '    e.last_name, e.job_id, e.manager_id, e.hire_date, e.salary, e.commission_pct, e.department_id); END; </pre>		
Salida de Script x Resultado de la Consulta x		
Tarea terminada en 0.046 segundos		
Procedimiento PL/SQL terminado correctamente.		

7) Revisar nuevamente si existe un registro en la tabla EMPLOYEE\_DETAILS con el ID del empleado igual al número de secuencia retornado en el paso 4.

Hoja de Trabajo

Generador de Consultas

SELECT \* FROM employee\_details WHERE EMPID = 2;

Salida de Script x

Resultado de la Consulta x

SQL | Todas las Filas Recuperadas: 1 en 0 segundos

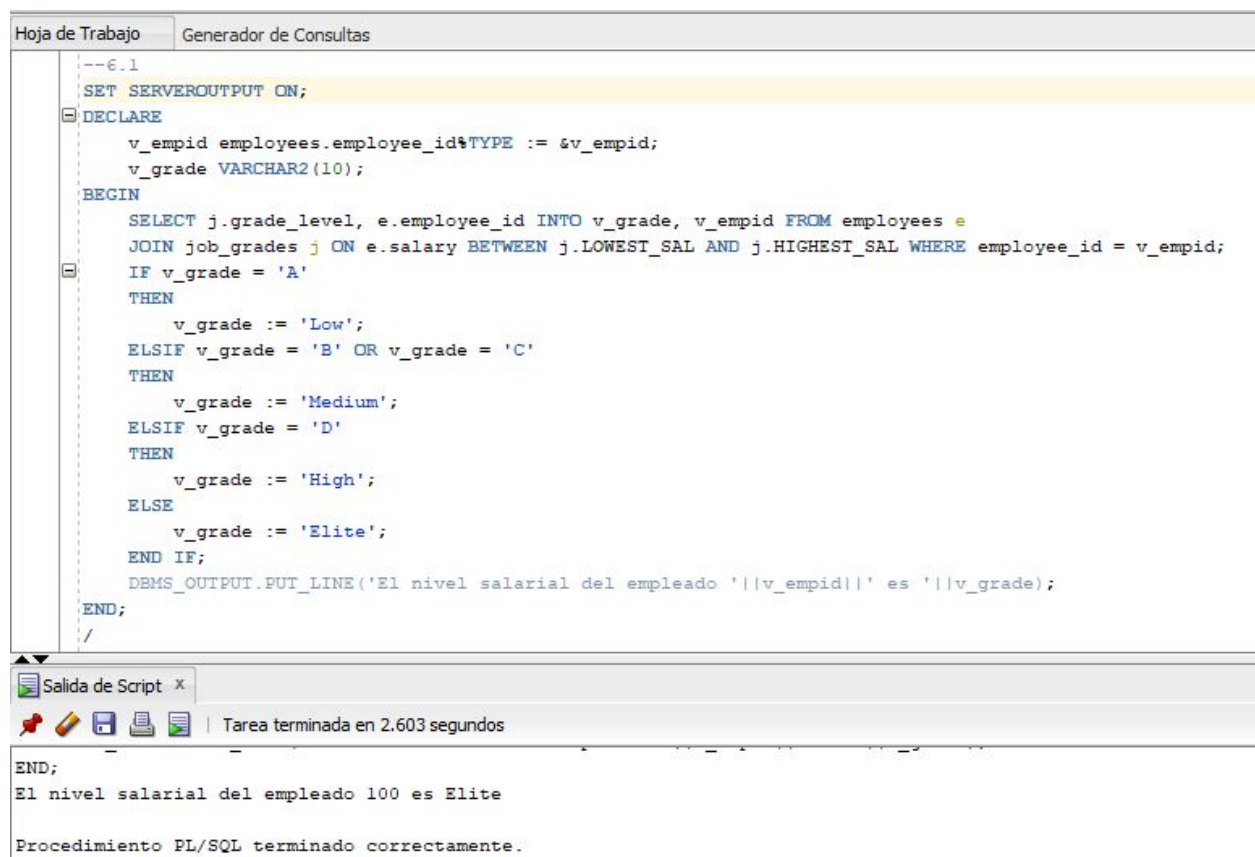
EMPID	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
1	2 Luis Ross	IT_PROG	103	16-JUL-19	4800	(null)	60

II. Realice los siguientes ejercicios luego de estudiar el documento D64254GC11\_les06.ppt:

6.1 Construya un código PL/SQL que lea los datos (ID del empleado y nivel salarial) de un empleado que el usuario introduzca por pantalla. Relacione sus datos con la tabla JOB\_GRADES para conocer su nivel salarial. Si el nivel del empleado es 'A' se desea desplegar "Low", si es 'B' o 'C' que despliegue 'Medium', si es 'D' que despliegue 'High' y si no es ninguno de estos niveles, que despliegue 'Elite'. Muestre por pantalla

El nivel salarial del empleado <<ID del empleado>> es <<nivel salarial>>

Use la estructura IF.



The screenshot shows a PL/SQL script in a 'Generador de Consultas' window. The script defines a variable `v_empid` of type `employees.employee_id%TYPE` and a variable `v_grade` of type `VARCHAR2(10)`. It then performs a `SELECT` query joining `employees` and `job_grades` tables. An `IF` statement follows, mapping grade levels 'A', 'B', 'C', 'D' to 'Low', 'Medium', 'High', and 'Elite' respectively. The script uses `DBMS_OUTPUT.PUT_LINE` to display the result. The output window at the bottom shows the execution result for employee ID 100, which is 'Elite', and a confirmation message that the PL/SQL procedure completed successfully.

```
--6.1
SET SERVEROUTPUT ON;
DECLARE
    v_empid employees.employee_id%TYPE := &v_empid;
    v_grade VARCHAR2(10);
BEGIN
    SELECT j.grade_level, e.employee_id INTO v_grade, v_empid FROM employees e
    JOIN job_grades j ON e.salary BETWEEN j.LOWEST_SAL AND j.HIGHEST_SAL WHERE employee_id = v_empid;
    IF v_grade = 'A'
    THEN
        v_grade := 'Low';
    ELSIF v_grade = 'B' OR v_grade = 'C'
    THEN
        v_grade := 'Medium';
    ELSIF v_grade = 'D'
    THEN
        v_grade := 'High';
    ELSE
        v_grade := 'Elite';
    END IF;
    DBMS_OUTPUT.PUT_LINE('El nivel salarial del empleado '||v_empid||' es '||v_grade);
END;
/
```

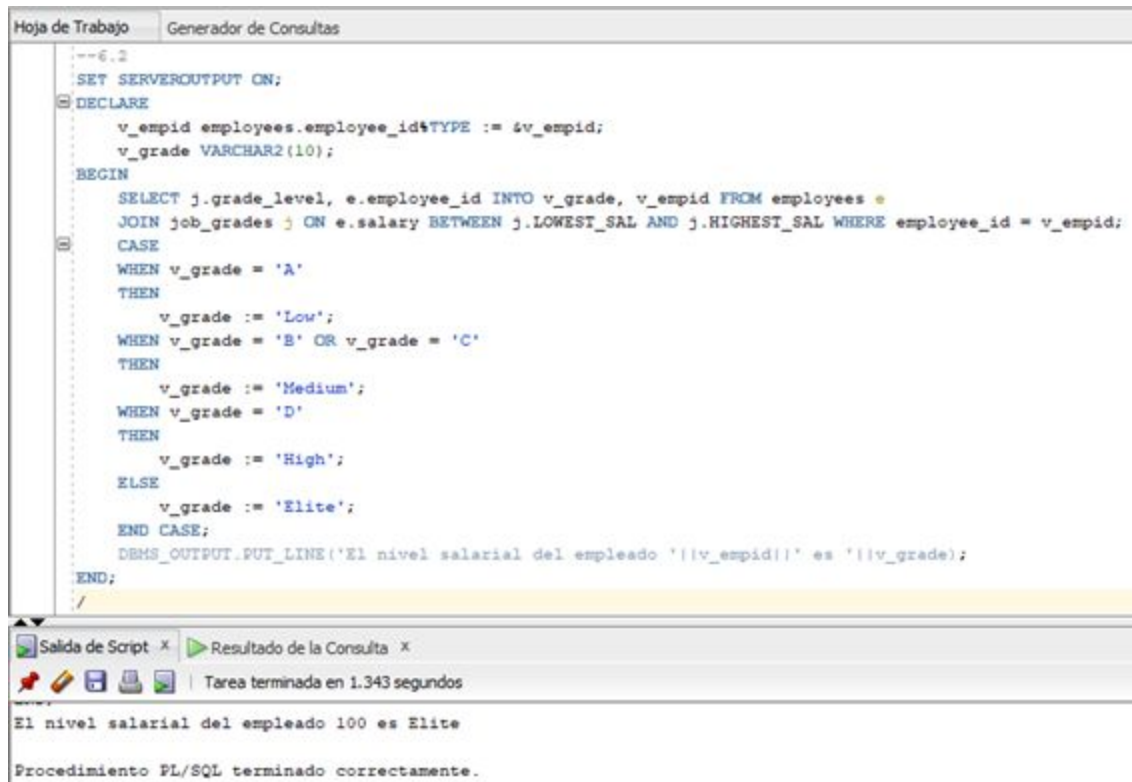
Salida de Script x

Tarea terminada en 2.603 segundos

END;  
El nivel salarial del empleado 100 es Elite

Procedimiento PL/SQL terminado correctamente.

## 6.2 Reconstruya el código del ejercicio 6.1 y use la estructura CASE.



The screenshot shows the Oracle SQL Developer interface. The top pane, titled 'Hoja de Trabajo' and 'Generador de Consultas', contains a PL/SQL script. The script starts with a comment '--6.2', followed by 'SET SERVEROUTPUT ON;'. It then declares two variables: 'v\_empid' of type 'employees.employee\_id%TYPE' and 'v\_grade' of type 'VARCHAR2(10)'. The 'BEGIN' block contains a 'SELECT' statement that joins 'employees' and 'job\_grades' tables, filtering by 'employee\_id = v\_empid'. This is followed by a 'CASE' statement that assigns values to 'v\_grade' based on the 'grade\_level' from the join: 'A' becomes 'Low', 'B' or 'C' becomes 'Medium', 'D' becomes 'High', and any other value becomes 'Elite'. After the 'END CASE;', there is a 'DBMS\_OUTPUT.PUT\_LINE' statement that prints the salary level for the employee with ID 'v\_empid'. The script ends with 'END;' and a forward slash '/' to execute it.

```
--6.2
SET SERVEROUTPUT ON;
DECLARE
    v_empid employees.employee_id%TYPE := &v_empid;
    v_grade VARCHAR2(10);
BEGIN
    SELECT j.grade_level, e.employee_id INTO v_grade, v_empid FROM employees e
    JOIN job_grades j ON e.salary BETWEEN j.LOWEST_SAL AND j.HIGHEST_SAL WHERE employee_id = v_empid;
    CASE
        WHEN v_grade = 'A'
        THEN
            v_grade := 'Low';
        WHEN v_grade = 'B' OR v_grade = 'C'
        THEN
            v_grade := 'Medium';
        WHEN v_grade = 'D'
        THEN
            v_grade := 'High';
        ELSE
            v_grade := 'Elite';
        END CASE;
    DBMS_OUTPUT.PUT_LINE('El nivel salarial del empleado '||v_empid||' es '||v_grade);
END;
/
```

The bottom pane shows the execution results. It has two tabs: 'Salida de Script' and 'Resultado de la Consulta'. The 'Resultado de la Consulta' tab is active, showing the output of the script: 'El nivel salarial del empleado 100 es Elite'. Below this, it states 'Procedimiento PL/SQL terminado correctamente.'

Salida de Script x Resultado de la Consulta x

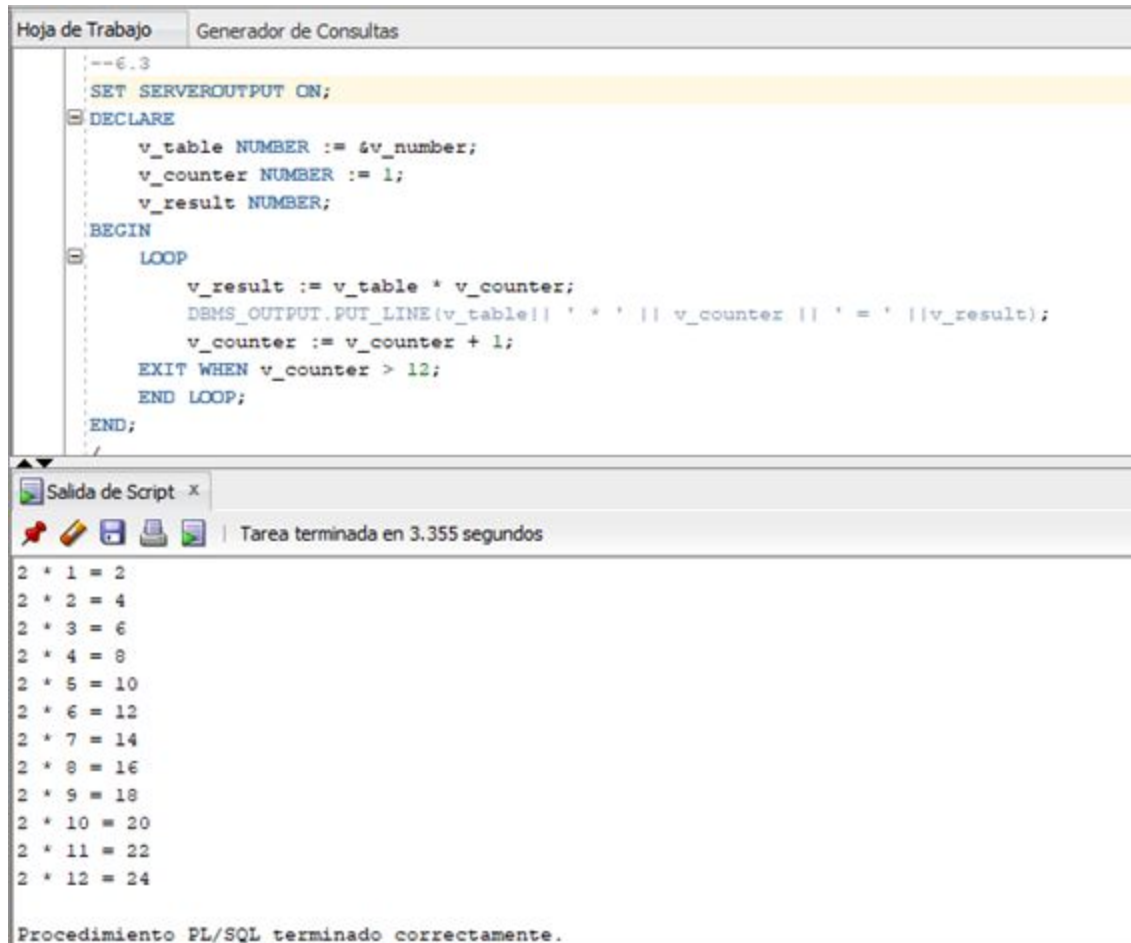
Tarea terminada en 1.343 segundos

El nivel salarial del empleado 100 es Elite

Procedimiento PL/SQL terminado correctamente.

6.3 Construya un código PL/SQL para mostrar la tabla de multiplicar seleccionada por el usuario. Cuando el contador llegue a 12 salir del ciclo. Use un ciclo simple. Muestre en pantalla la información como:

tabla \* contador = resultado



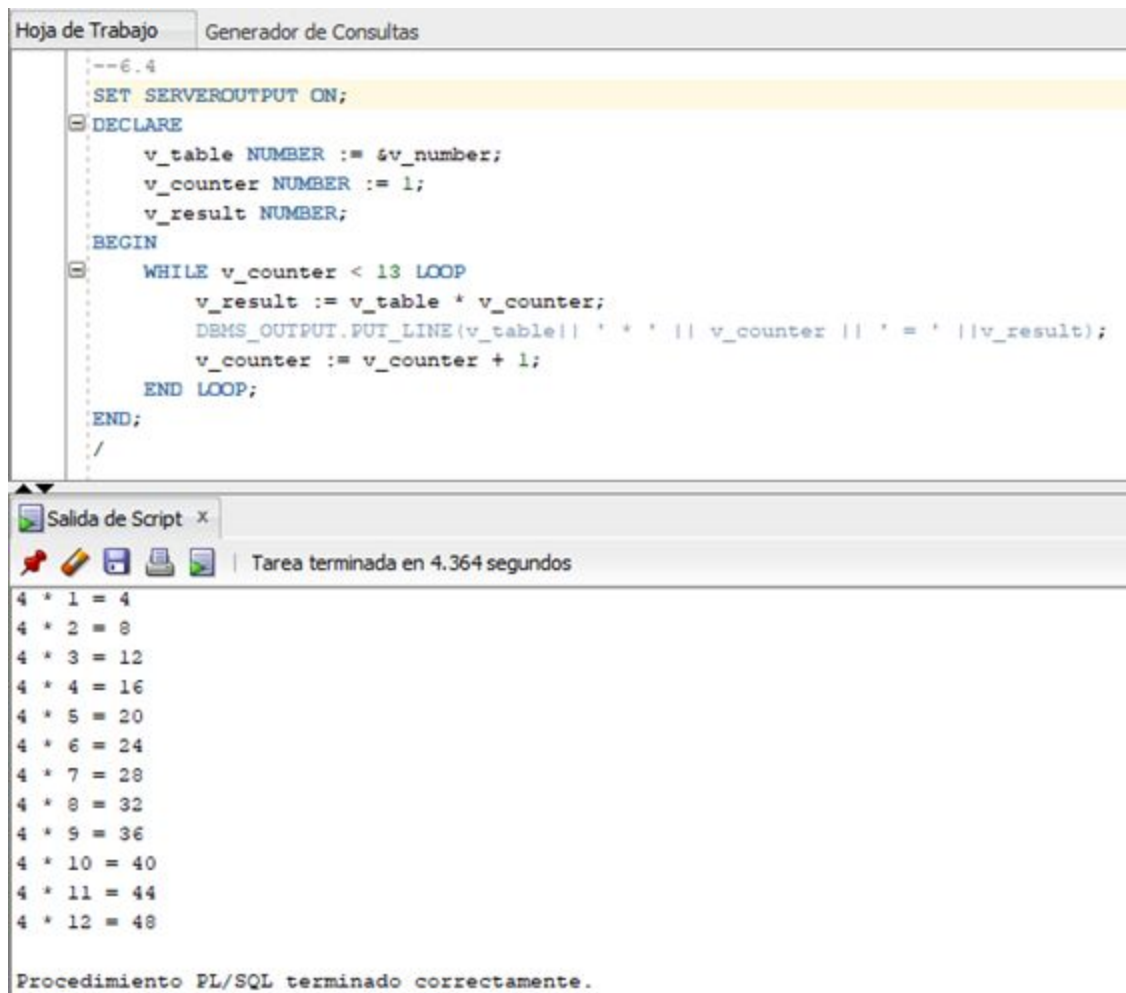
The screenshot displays the Oracle SQL Developer environment. The top pane, titled 'Generador de Consultas', contains a PL/SQL script. The script starts with a comment '--6.3', followed by 'SET SERVEROUTPUT ON;'. It then declares three variables: 'v\_table' of type NUMBER, 'v\_counter' of type NUMBER, and 'v\_result' of type NUMBER. The script enters a 'BEGIN' block containing a 'LOOP'. Inside the loop, it calculates 'v\_result := v\_table \* v\_counter;', outputs the result using 'DBMS\_OUTPUT.PUT\_LINE(v\_table || ' \* ' || v\_counter || ' = ' || v\_result);', increments the counter with 'v\_counter := v\_counter + 1;', and exits the loop when 'v\_counter > 12;'. The script ends with 'END LOOP;' and 'END;'. The bottom pane, titled 'Salida de Script', shows the output of the script, which is a multiplication table for the number 2, ranging from 2 \* 1 = 2 to 2 \* 12 = 24. Below the output, a status message reads 'Procedimiento PL/SQL terminado correctamente.'

```
--6.3
SET SERVEROUTPUT ON;
DECLARE
    v_table NUMBER := &v_number;
    v_counter NUMBER := 1;
    v_result NUMBER;
BEGIN
    LOOP
        v_result := v_table * v_counter;
        DBMS_OUTPUT.PUT_LINE(v_table || ' * ' || v_counter || ' = ' || v_result);
        v_counter := v_counter + 1;
        EXIT WHEN v_counter > 12;
    END LOOP;
END;
```

2 \* 1 = 2  
2 \* 2 = 4  
2 \* 3 = 6  
2 \* 4 = 8  
2 \* 5 = 10  
2 \* 6 = 12  
2 \* 7 = 14  
2 \* 8 = 16  
2 \* 9 = 18  
2 \* 10 = 20  
2 \* 11 = 22  
2 \* 12 = 24

Procedimiento PL/SQL terminado correctamente.

6.4 Reconstruya el código del ejercicio 6.3 usando el ciclo WHILE.



The screenshot displays the SQL Developer interface. The top pane, titled 'Hoja de Trabajo' and 'Generador de Consultas', contains a PL/SQL script. The script starts with a comment '--6.4', followed by 'SET SERVEROUTPUT ON;'. It then declares three variables: 'v\_table' of type NUMBER with a value of 4, 'v\_counter' of type NUMBER with a value of 1, and 'v\_result' of type NUMBER. The script enters a 'BEGIN' block containing a 'WHILE' loop that runs as long as 'v\_counter' is less than or equal to 13. Inside the loop, 'v\_result' is calculated as 'v\_table' multiplied by 'v\_counter', the result is printed using 'DBMS\_OUTPUT.PUT\_LINE', and 'v\_counter' is incremented by 1. The loop ends with 'END LOOP;', and the entire block concludes with 'END;'. The bottom pane, titled 'Salida de Script', shows the execution results. It indicates that the task was completed in 4.364 seconds. The output consists of 12 lines, each showing the multiplication of 4 by a counter from 1 to 12. At the bottom, a message states 'Procedimiento PL/SQL terminado correctamente.'

```
--6.4
SET SERVEROUTPUT ON;

DECLARE
    v_table NUMBER := 4;
    v_counter NUMBER := 1;
    v_result NUMBER;
BEGIN
    WHILE v_counter <= 13 LOOP
        v_result := v_table * v_counter;
        DBMS_OUTPUT.PUT_LINE(v_table || ' * ' || v_counter || ' = ' || v_result);
        v_counter := v_counter + 1;
    END LOOP;
END;
/
```

Salida de Script x

Tarea terminada en 4.364 segundos

```
4 * 1 = 4
4 * 2 = 8
4 * 3 = 12
4 * 4 = 16
4 * 5 = 20
4 * 6 = 24
4 * 7 = 28
4 * 8 = 32
4 * 9 = 36
4 * 10 = 40
4 * 11 = 44
4 * 12 = 48
```

Procedimiento PL/SQL terminado correctamente.

6.5 Reconstruya el código del ejercicio 6.3 usando el ciclo FOR.

```
SET SERVEROUTPUT ON
DECLARE
v_table NUMBER:= &v_num;
v_result NUMBER;
BEGIN

    FOR i IN 1..12 loop
        v_result := v_table * i;
        DBMS_OUTPUT.PUT_LINE(v_table||' * '||i||' = '||v_result);
    END LOOP;
END;
/
```

Salida de Script x Resultado de la Consulta x

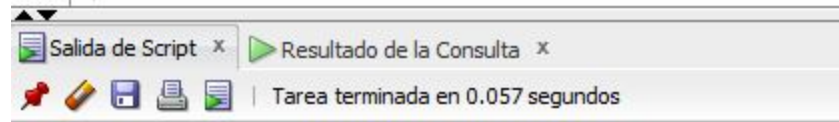
Tarea terminada en 4.553 segundos

```
4 * 1 = 4
4 * 2 = 8
4 * 3 = 12
4 * 4 = 16
4 * 5 = 20
4 * 6 = 24
4 * 7 = 28
4 * 8 = 32
4 * 9 = 36
4 * 10 = 40
4 * 11 = 44
4 * 12 = 48
```

Procedimiento PL/SQL terminado correctamente.

6.6 Construya un código PL/SQL que muestre los números pares del 2 al 10 en reversa. Use el ciclo FOR.

```
SET SERVEROUTPUT ON
BEGIN
  FOR i IN REVERSE 1..10 loop
    IF MOD(i,2)=0 THEN
      DBMS_OUTPUT.PUT_LINE(i);
    END IF;
  END LOOP;
END;
/
```

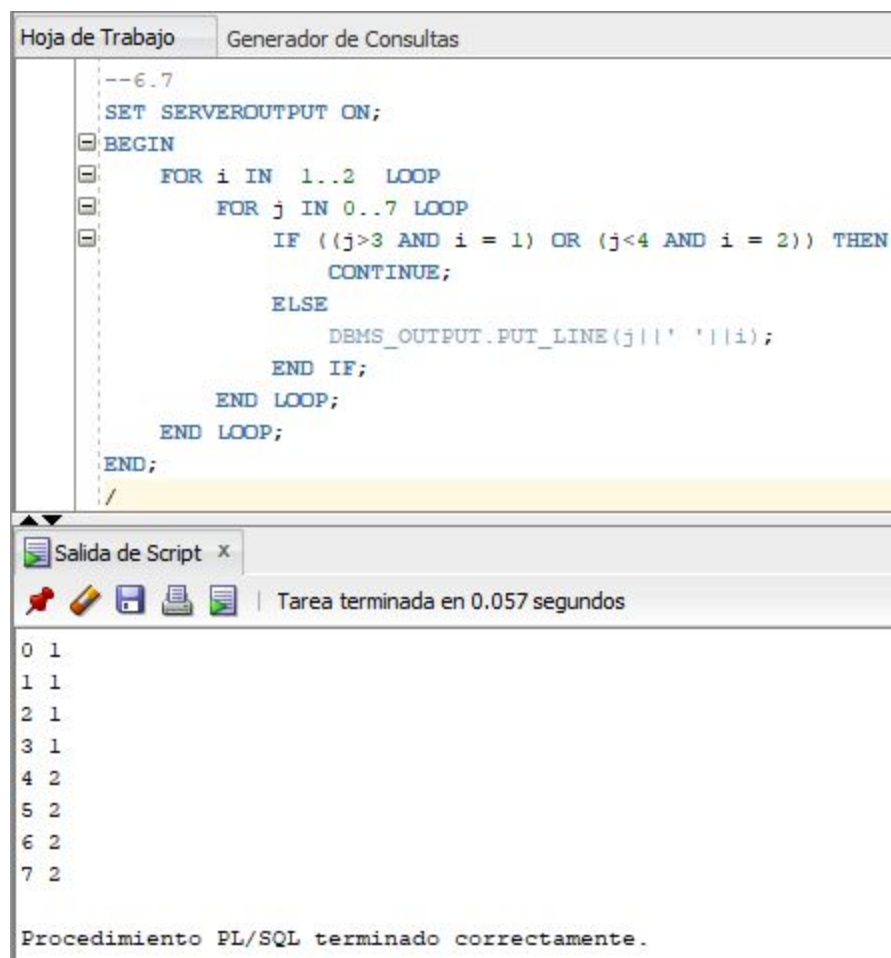


10  
8  
6  
4  
2

Procedimiento PL/SQL terminado correctamente.

6.7 Construya un código PL/SQL que utilizando ciclos FOR anidados genere las siguientes parejas de números:

0 1  
1 1  
2 1  
3 1  
4 2  
5 2  
6 2  
7 2



The screenshot shows a PL/SQL script editor window titled "Hoja de Trabajo" and "Generador de Consultas". The script is as follows:

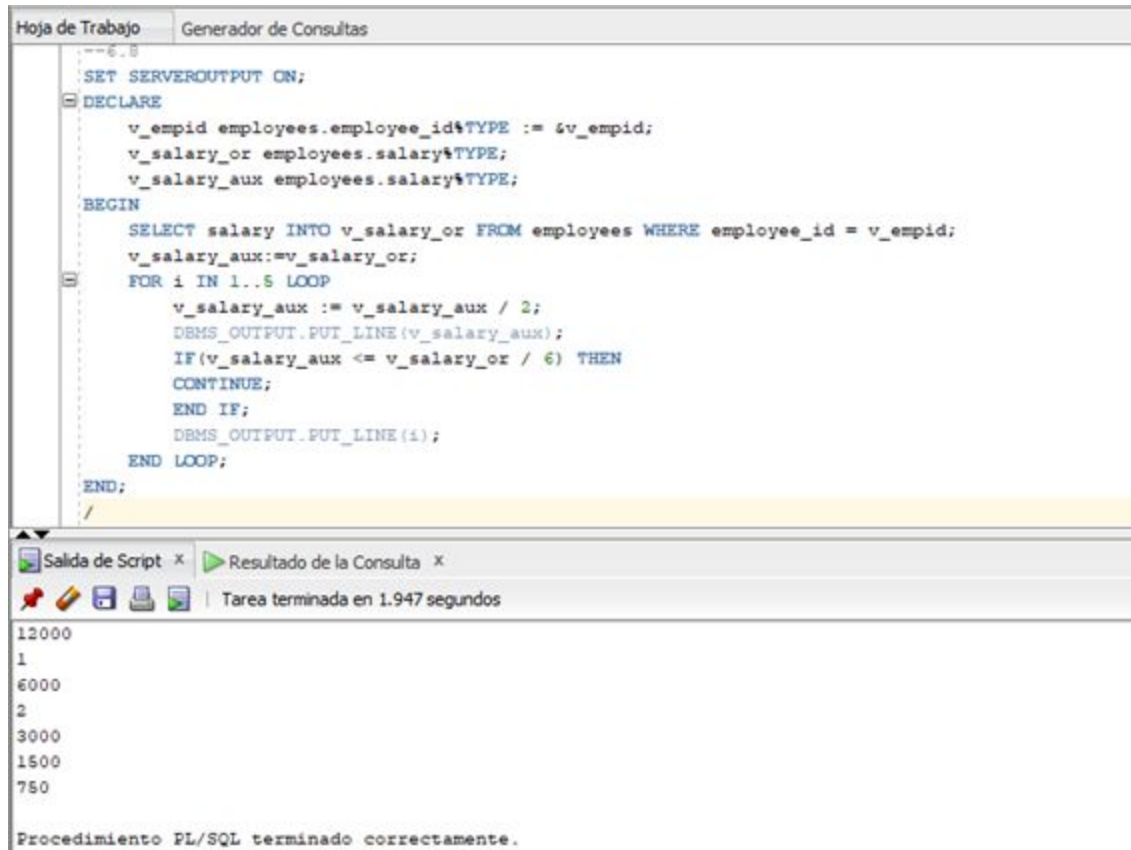
```
--6.7  
SET SERVEROUTPUT ON;  
BEGIN  
  FOR i IN 1..2 LOOP  
    FOR j IN 0..7 LOOP  
      IF ((j>3 AND i = 1) OR (j<4 AND i = 2)) THEN  
        CONTINUE;  
      ELSE  
        DBMS_OUTPUT.PUT_LINE(j||' '||i);  
      END IF;  
    END LOOP;  
  END LOOP;  
END;  
/
```

Below the script editor is a status bar that says "Salida de Script x" and "Tarea terminada en 0.057 segundos". Below that is a window showing the output of the script:

```
0 1  
1 1  
2 1  
3 1  
4 2  
5 2  
6 2  
7 2  
  
Procedimiento PL/SQL terminado correctamente.
```



6.8 Construya un código PL/SQL que busque en la tabla de empleados el salario de un empleado introducido por el usuario. Crear un ciclo que dé cinco (5) iteraciones. Durante cada iteración se quiere que el salario se divida entre dos (2) y se muestre en pantalla el resultado. En otra sección del ciclo se desea mostrar en pantalla el número de iteración, pero sólo mientras el resultado de dividir el salario entre dos (2) sea  $\geq$  al salario entre seis (6). Para este último requerimiento use CONTINUE.



The screenshot shows the Oracle SQL Developer interface. The top pane, titled 'Hoja de Trabajo' and 'Generador de Consultas', contains a PL/SQL script. The script sets server output on, declares variables for employee ID, salary, and an auxiliary salary variable, then begins a loop that divides the salary by 2 five times, printing the result and the iteration number. The bottom pane shows the output of the script, which displays the salary (12000), the iteration number (1), and the result of the division (6000). The task is completed in 1.947 seconds.

```
--6.8
SET SERVEROUTPUT ON;
DECLARE
    v_empid employees.employee_id%TYPE := &v_empid;
    v_salary employees.salary%TYPE;
    v_salary_aux employees.salary%TYPE;
BEGIN
    SELECT salary INTO v_salary_or FROM employees WHERE employee_id = v_empid;
    v_salary_aux:=v_salary_or;
    FOR i IN 1..5 LOOP
        v_salary_aux := v_salary_aux / 2;
        DBMS_OUTPUT.PUT_LINE(v_salary_aux);
        IF(v_salary_aux <= v_salary_or / 6) THEN
            CONTINUE;
        END IF;
        DBMS_OUTPUT.PUT_LINE(i);
    END LOOP;
END;
```

Salida de Script x Resultado de la Consulta x

Tarea terminada en 1.947 segundos

12000  
1  
6000  
2  
3000  
1500  
750

Procedimiento PL/SQL terminado correctamente.