**Desarrollo de aplicaciones multiplataforma**

**Acceso a datos**

**Ejercicios obligatorios-Tema 1**

1. Realizar un programa que solicite un número de empleado determinado, el programa debe comprobar si el salario es mayor que 1000, y si no es así actualizar el salario con una subida del 20% y, en una tabla temporal, insertar el nombre del empleado y el mensaje "Sueldo actualizado". Si ya superaba los 1000, se inserta en la tabla temporal el nombre del empleado y el mensaje "No necesita actualización".

**RESPUESTA:**

**SET SERVEROUTPUT ON;**

**drop table tabla\_temporal;**

**create table tabla\_temporal(id\_empleado number(4) , mensage varchar2(50));**

**declare**

**sueldo employee.salary%type;**

**codigo employee.employee\_id%type := '&id\_empleado';**

**begin**

**select salary into sueldo from employee where employee\_id = codigo;**

**if(sueldo > 1000) then**

**insert into tabla\_temporal values(codigo,'No necesita actualización');**

**DBMS\_OUTPUT.put\_line('Id:'||codigo||' Sueldo:'||sueldo);**

**ELSE**

**UPDATE employee**

**SET salary = salary\*1.20**

**WHERE employee\_id = codigo;**

**insert into tabla\_temporal values(codigo,'Sueldo actualizado');**

**end if;**

**end;**

1. Hacer un programa que solicite por teclado un código de cliente e inserte en la tabla temporal el número total de pedidos de ese cliente (tot\_ped), el precio total de esos pedidos (precio\_tot), el código del cliente (cod\_cli) y el nombre del cliente (nombre\_cli).

**RESPUESTA:**

SET SERVEROUTPUT ON;

DROP TABLE tabla\_temporal;

CREATE TABLE tabla\_temporal (

cod\_cli NUMBER(4),

nombre\_cli VARCHAR2(50),

precio\_tot NUMBER(8),

tot\_ped NUMBER(8)

);

DECLARE

codigo\_cliente NUMBER := &id\_cliente; -- Declaración y asignación de la variable código\_cliente.

nombre customer.name%TYPE;

pedido\_total sales\_order.total%TYPE; -- Debe ser sales\_order.total, ya que se obtiene de la tabla sales\_order.

precio\_total item.total%TYPE;

BEGIN

FOR cliente IN (SELECT

c.customer\_id,

c.name,

so.total AS pedido\_total,

i.total AS precio\_total

FROM

customer c

JOIN sales\_order so ON c.customer\_id = so.customer\_id

JOIN item i ON so.order\_id = i.order\_id

WHERE

c.customer\_id = codigo\_cliente)

LOOP

nombre := cliente.name;

pedido\_total := cliente.pedido\_total;

precio\_total := cliente.precio\_total;

insert into tabla\_temporal values(codigo\_cliente,nombre,precio\_total,pedido\_total);

END LOOP;

END;

/

1. Crearse un registro t\_reg\_emple con la siguiente estructura: código del empleado, nombre y el job. Insertar en una tabla temporal el nombre del empleado y job para el empleado 7782.

**RESPUESTA:**

**SET SERVEROUTPUT ON;**

**DROP TABLE tabla\_temporal;**

**CREATE TABLE tabla\_temporal (**

**codigo\_empleado NUMBER(4),**

**nombre VARCHAR2(50),**

**trabajo VARCHAR2(50)**

**);**

**DECLARE**

**TYPE t\_reg\_emple IS RECORD (**

**codigo\_empleado NUMBER(4),**

**nombre employee.first\_name%TYPE,**

**trabajo VARCHAR2(50)**

**);**

**empleado t\_reg\_emple;**

**BEGIN**

**SELECT**

**employee\_id,**

**last\_name,**

**function**

**INTO empleado**

**FROM**

**employee**

**JOIN job ON job.job\_id = employee.job\_id**

**WHERE**

**employee\_id = 7782;**

**INSERT INTO tabla\_temporal VALUES (**

**empleado.codigo\_empleado,**

**empleado.nombre,**

**empleado.trabajo**

**);**

**dbms\_output.put\_line('Fila insertada en tabla\_temporal.');**

**-- Mostrar los valores insertados.**

**dbms\_output.put\_line('Código: '**

**|| empleado.codigo\_empleado**

**|| ', Nombre: '**

**|| empleado.nombre**

**|| ', Trabajo: '**

**|| empleado.trabajo);**

**END;**

**/**

1. Insertar 50 filas en una tabla. Recorre todas las filas de la tabla y las inserta en una tabla temporal. Crear la tabla students e insertar algunos datos.

**RESPUESTA:**

SET SERVEROUTPUT ON;

DROP TABLE students;

CREATE TABLE students (

codigo\_alumno NUMBER(4),

nombre VARCHAR2(50),

apellido VARCHAR2(50)

);

BEGIN

--crear 50 estudiantes llamado loli pop

FOR i IN 1..50 LOOP

INSERT INTO students (codigo\_alumno, nombre, apellido)

VALUES (i, 'Loli', 'Pop');

END LOOP;

END;

/

drop TABLE tabla\_temporal;

CREATE TABLE tabla\_temporal (

codigo\_alumno NUMBER(4),

nombre VARCHAR2(50),

apellido VARCHAR2(50)

);

BEGIN

--inserta en una tabla temporal

INSERT INTO tabla\_temporal (codigo\_alumno, nombre, apellido)

SELECT codigo\_alumno, nombre, apellido FROM students;

END;

/

1. Llenar una tabla PL con los nombres de los clientes y cargarla en una tabla temporal de sql que contenga el índice de la tabla PL y el nombre del cliente, y visualizar ésta.

**RESPUESTA:**

-- Eliminar la tabla, la secuencia y el trigger si existen

DROP TABLE tabla\_temporal;

DROP TABLE tabla\_temporal1;

DROP SEQUENCE tabla\_id\_seq;

DROP TRIGGER tri\_autoincremento\_id;

-- Crear la tabla para el índice PL

CREATE TABLE tabla\_temporal (

id\_temporal NUMBER PRIMARY KEY,

nombre\_cliente VARCHAR2(50)

);

-- Crear la tabla para guardar solo el nombre de los clientes

CREATE TABLE tabla\_temporal1 (

nombre\_cliente VARCHAR2(50)

);

-- Crear una secuencia para el autoincremento de id +1

CREATE SEQUENCE tabla\_id\_seq START WITH 1 INCREMENT BY 1 NOCACHE NOCYCLE;

-- Crear un trigger que se activa antes de insertar en la tabla\_temporal1

CREATE OR REPLACE TRIGGER tri\_autoincremento\_id

BEFORE INSERT ON tabla\_temporal1

FOR EACH ROW

BEGIN

INSERT INTO tabla\_temporal (id\_temporal, nombre\_cliente)

VALUES (tabla\_id\_seq.NEXTVAL, :new.nombre\_cliente);

END;

/

-- Insertar datos en la tabla\_temporal1 usando un bucle FOR

BEGIN

FOR j IN 1..10 LOOP

INSERT INTO tabla\_temporal1 VALUES ('Pepe' || j);

END LOOP;

END;

/

1. Llenar una tabla temporal con los códigos de clientes, nombre del cliente y código de empleado de tres formas posibles
   1. usando bucle LOOP,

DROP TABLE tabla\_temporal;

-- Crear la tabla para el índice PL

CREATE TABLE tabla\_temporal (

id\_cliente NUMBER,

nombre\_cliente VARCHAR2(50),

id\_empleado NUMBER(4)

);

DECLARE

CURSOR c1 IS

SELECT

customer\_id,

name,

salesperson\_id

FROM

customer;

id\_cus customer.customer\_id%TYPE;

nombre customer.name%TYPE;

id\_emp customer.salesperson\_id%TYPE;

contador number(4);

BEGIN

select count(\*) into contador from customer;

OPEN c1;

LOOP

FETCH c1 INTO

id\_cus,

nombre,

id\_emp;

INSERT INTO tabla\_temporal (

id\_cliente,

nombre\_cliente,

id\_empleado

) VALUES (

id\_cus,

nombre,

id\_emp

);

contador := contador-1;

exit when contador = 0;

END LOOP;

CLOSE c1;

END;

/

* 1. usando bucle WHILE y

-- Elimina la tabla temporal si existe

DROP TABLE tabla\_temporal;

-- Crea la tabla para el índice PL

CREATE TABLE tabla\_temporal (

id\_cliente NUMBER,

nombre\_cliente VARCHAR2(50),

id\_empleado NUMBER(4)

);

DECLARE

CURSOR c1 IS

SELECT

customer\_id,

name,

salesperson\_id

FROM

customer;

id\_cus customer.customer\_id%TYPE;

nombre customer.name%TYPE;

id\_emp customer.salesperson\_id%TYPE;

BEGIN

OPEN c1;

FETCH c1 INTO

id\_cus,

nombre,

id\_emp;

WHILE c1%found LOOP

FETCH c1 INTO

id\_cus,

nombre,

id\_emp;

INSERT INTO tabla\_temporal (

id\_cliente,

nombre\_cliente,

id\_empleado

) VALUES (

id\_cus,

nombre,

id\_emp

);

END LOOP;

CLOSE c1;

END;

/

* 1. usando bucle FOR.

-- Elimina la tabla temporal si existe

DROP TABLE tabla\_temporal;

-- Crea la tabla para el índice PL

CREATE TABLE tabla\_temporal (

id\_cliente NUMBER PRIMARY KEY,

nombre\_cliente VARCHAR2(50),

id\_empleado NUMBER(4)

);

BEGIN

-- Cursor para recorrer los registros de la tabla customer

FOR c IN (SELECT customer\_id, name, salesperson\_id FROM customer) LOOP

-- Inserta los valores en la tabla\_temporal

INSERT INTO tabla\_temporal (id\_cliente, nombre\_cliente, id\_empleado)

VALUES (c.customer\_id, c.name, c.salesperson\_id);

END LOOP;

END;

/

**RESPUESTA:**

1. A los pedidos del cliente 2, añadir 10 unidades más a la cantidad e insertar en una tabla temporal el código de artículo y un mensaje que diga "Diez unidades más vendidas".

**RESPUESTA:**

DROP TABLE tabla\_temporal;

-- Crear la tabla

CREATE TABLE tabla\_temporal (

id\_articulo NUMBER,

mensaje VARCHAR2(50)

);

DECLARE

id\_product NUMBER;

BEGIN

UPDATE item

SET

quantity = quantity + 10,

total = (

SELECT

(quantity + 10) \* actual\_price

FROM

item i,

sales\_order so,

customer c

WHERE

i.order\_id = so.order\_id

AND so.customer\_id = c.customer\_id

AND so.order\_id = 501

AND c.customer\_id = 204

)

WHERE

order\_id IN (

SELECT

order\_id

FROM

sales\_order so,

customer c

WHERE

so.customer\_id = c.customer\_id

AND so.order\_id = 501

AND c.customer\_id = 204

);

SELECT

i.product\_id

INTO id\_product

FROM

item i,

sales\_order so,

customer c

WHERE

i.order\_id = so.order\_id

AND so.customer\_id = c.customer\_id

AND so.order\_id = 501

AND c.customer\_id = 204;

-- Cambié las comillas dobles por comillas simples

INSERT INTO tabla\_temporal (

id\_articulo,

mensaje

) VALUES (

id\_product,

'Diez unidades mas vendidas'

);

-- Para verificar si la inserción fue exitosa

DBMS\_OUTPUT.PUT\_LINE('Registro insertado en tabla\_temporal.');

END;

/

1. Crear un procedimiento que añada nuevos pedidos a un cliente determinado. El procedimiento recibe el idped, idart, cantidad y codcli.

**RESPUESTA:**

SET SERVEROUTPUT ON;

DROP TABLE tabla\_temporal;

DROP PROCEDURE aniadir\_pedido;

--tabla temporal para aniadir producto

CREATE TABLE tabla\_temporal (

order\_id REAL,

product\_id REAL,

quantity REAL,

customer\_id REAL

);

--procedimiento aniadir producto

CREATE OR REPLACE PROCEDURE aniadir\_pedido (

idped IN NUMBER,

idart IN NUMBER,

cantidad IN NUMBER,

codcli IN NUMBER

) IS

BEGIN

-- Insertar el nuevo pedido en la tabla de pedidos

INSERT INTO tabla\_temporal (

order\_id,

product\_id,

quantity,

customer\_id

) VALUES (

idped,

idart,

cantidad,

codcli

);

-- Mensaje de éxito

dbms\_output.put\_line('Nuevo pedido añadido para el cliente ' || codcli);

EXCEPTION

WHEN OTHERS THEN

-- Manejar cualquier error que pueda ocurrir durante la inserción

dbms\_output.put\_line('Error: '

|| sqlcode

|| ' - '

|| sqlerrm);

END;

/

--ejecutar procedimiento

execute aniadir\_pedido(1,2,3,4);

1. Procedimiento que borra los pedidos del cliente especificado. Recibe el codcli.

**RESPUESTA:**

SET SERVEROUTPUT ON;

DROP TABLE tabla\_temporal;

DROP PROCEDURE borrar\_pedido;

--tabla temporal para aniadir producto

CREATE TABLE tabla\_temporal (

order\_id REAL,

product\_id REAL,

quantity REAL,

customer\_id REAL

);

--procedimiento aniadir producto

CREATE OR REPLACE PROCEDURE borrar\_pedido (

codcli IN NUMBER

) IS

BEGIN

-- borrar todos los pedidos en la tabla de pedidos

delete from tabla\_temporal where customer\_id = codcli;

-- Mensaje de éxito

dbms\_output.put\_line('Exito en borrar los pedidos para el cliente ' || codcli);

EXCEPTION

WHEN OTHERS THEN

-- Manejar cualquier error que pueda ocurrir durante la inserción

dbms\_output.put\_line('Error: '

|| sqlcode

|| ' - '

|| sqlerrm);

END;

/

--ejecutar procedimiento

execute borrar\_pedido(1);

1. Procedimiento para cambiar el oficio de un empleado. Se pasa al codemp y el nuevo oficio. Insertará en TEMP oficio\_ant, nuevo, codemp.

**RESPUESTA:**

DROP PROCEDURE cambio\_oficio;

DROP TABLE temp;

CREATE TABLE temp (

oficio\_antiguo NUMBER,

oficio\_nuevo NUMBER,

cod\_emp NUMBER PRIMARY KEY

);

CREATE OR REPLACE PROCEDURE cambio\_oficio (

oficio\_ant IN NUMBER,

nuevo IN NUMBER,

codemp IN NUMBER

) IS

BEGIN

INSERT INTO temp (

oficio\_antiguo,

oficio\_nuevo,

cod\_emp

) VALUES (

oficio\_ant,

nuevo,

codemp

);

END;

/

--ejecutar procedimiento

execute cambio\_oficio(1,2,7788);

1. Crear una función "Anual" para devolver el salario anual cuando se pasa el salario mensual y la comisión de un empleado. Asegurarse que controla nulos. Utilizar una variable de acoplamiento para ver lo que devuelve.

**RESPUESTA:**

set SERVEROUTPUT ON;

DROP PROCEDURE anual;

CREATE OR REPLACE PROCEDURE anual (

salario\_mensual IN NUMBER,

comision IN NUMBER,

codemp IN NUMBER

) IS

BEGIN

if salario\_mensual = null then

dbms\_output.put\_line('Salario nulo');

ELSIF comision = null then

dbms\_output.put\_line('Comision nulo');

else

dbms\_output.put\_line('Codigo empleado '||codemp||' Salario anual : '||to\_number(salario\_mensual\*14-nvl(comision,0)\*12));

end if;

END;

/

--ejecutar procedimiento

execute anual(2000,150,7788);

1. Crear un paquete "Actualiza" que tiene tres procedimientos y una tabla PL:
   1. Un procedimiento de alta de un pedido.
   2. Un procedimiento de baja de los pedidos y detalles de un cliente concreto.
   3. Una tabla PL que almacenará los códigos de los pedidos.
   4. Un procedimiento "Listar" que tiene como parámetro 'in' el código de cliente y como parámetro de salida una tabla PL con los códigos de los pedidos del cliente especificado, además se grabará en una tabla temporal el código del pedido y el código del artículo.

**RESPUESTA:**

1. Crear un trigger para ver como suceden los eventos de activación. Vamos a utilizar la tabla plantilla y vamos a cambiar a las "Enfermeras" por "Interinos".

**RESPUESTA:**

set SERVEROUTPUT ON;

DROP PACKAGE Actualiza;

drop table tabla\_temporal;

create table tabla\_temporal(

id\_pedido number,

id\_cliente number

);

--insertar datos a la tabla

insert into tabla\_temporal values(1,1001);

insert into tabla\_temporal values(2,1001);

insert into tabla\_temporal values(3,1002);

CREATE OR REPLACE PACKAGE Actualiza is

PROCEDURE alta\_pedido (id\_order NUMBER , id\_clien NUMBER);

PROCEDURE baja\_pedido (id\_cli NUMBER);

END;

/

CREATE OR REPLACE PACKAGE BODY Actualiza AS

PROCEDURE alta\_pedido(id\_order NUMBER, id\_clien NUMBER) AS

BEGIN

INSERT INTO tabla\_temporal VALUES (id\_order , id\_clien);

END;

PROCEDURE baja\_pedido(id\_cli NUMBER) AS

BEGIN

DELETE FROM tabla\_temporal WHERE id\_cliente = id\_cli ;

END;

END;

/

--ejecutar procedimiento

BEGIN

Actualiza.alta\_pedido(4, 1003);

Actualiza.baja\_pedido(1001);

END;

/

1. Crear un disparador a nivel de fila tal que después de insertar, modificar o borrar un detalle de la tabla detalles introduzca en la tabla temporal el usuario , la fecha del sistema el código de pedido que se ha modificado, insertado o borrado, así como un mensaje que diga "Detalle dado de alta", "Detalle borrado", "Detalle modificado" según corresponda.

**RESPUESTA:**

DROP TRIGGER detalle;

DROP TABLE tabla\_temporal;

DROP TABLE detalles;

CREATE TABLE tabla\_temporal (

usuario VARCHAR2(50),

fecha VARCHAR2(99),

mensaje VARCHAR2(99)

);

CREATE TABLE detalles (

usuario VARCHAR2(50),

fecha VARCHAR2(99),

mensaje VARCHAR2(99)

);

CREATE OR REPLACE TRIGGER detalle AFTER

UPDATE OR INSERT OR DELETE ON detalles

FOR EACH ROW

BEGIN

IF updating THEN

INSERT INTO tabla\_temporal (

usuario,

fecha,

mensaje

) VALUES (

user,

to\_char(sysdate, 'hh24:mi DD/MM/YY'),

'Detalle modificado'

);

END IF;

IF deleting THEN

INSERT INTO tabla\_temporal (

usuario,

fecha,

mensaje

) VALUES (

user,

to\_char(sysdate, 'hh24:mi DD/MM/YY'),

'Detalle borrado'

);

END IF;

IF inserting THEN

INSERT INTO tabla\_temporal (

usuario,

fecha,

mensaje

) VALUES (

user,

to\_char(sysdate, 'hh24:mi DD/MM/YY'),

'Detalle dado de alta'

);

END IF;

END;

/

--Comprobaciones

INSERT INTO detalles (

usuario,

fecha,

mensaje

) VALUES (

user,

to\_char(sysdate, 'hh24:mi DD/MM/YY'),

'SIIIIIII'

);

UPDATE detalles

SET

mensaje = 'holaa';

DELETE detalles

WHERE

mensaje = 'holaa';

1. Disparador que inserta una fila en la tabla Temp con el texto ‘subida de salario empleado’ y el número del empleado al que se le ha subido el salario. El disparador se activará después de actualizar el salario de la tabla empleador.

**RESPUESTA:**

DROP TABLE temporal;

CREATE TABLE temporal(

informacion VARCHAR2(50)

);

CREATE OR REPLACE TRIGGER subirsalario

AFTER UPDATE ON employee

FOR EACH ROW

BEGIN

IF :NEW.salary > :OLD.salary THEN

INSERT INTO temporal VALUES('Subida de salario empleado: ' || :NEW.employee\_id);

END IF;

END;

/

UPDATE employee SET salary = 5500 WHERE employee\_id = 7788;

SELECT \* FROM temporal;

1. Trigger que se dispara cada vez que se borra un empleado, guardando el número empleado, apellido y departamento en la tabla TEMP.

**RESPUESTA:**

DROP TABLE temporal;

CREATE TABLE temporal(

empleado\_id NUMBER(4,0),

apellido\_empleado VARCHAR2(20),

depart\_id NUMBER(2,0)

);

CREATE OR REPLACE TRIGGER BorraEmpleado

AFTER DELETE ON employee

FOR EACH ROW

BEGIN

INSERT INTO temporal VALUES(:OLD.employee\_id, :OLD.last\_name, :OLD.department\_id);

END;

/

1. Limitar a 5 (0 al 4) el número de detalles de cada pedido.

**RESPUESTA:**

CREATE OR REPLACE TRIGGER LimitarDetalles

BEFORE INSERT ON sales\_order

FOR EACH ROW

DECLARE

CountDetalles NUMBER;

BEGIN

-- Contar los detalles actuales para el pedido

SELECT COUNT(\*) INTO CountDetalles FROM sales\_order WHERE order\_id = :NEW.order\_id;

IF CountDetalles >= 5 THEN

RAISE\_APPLICATION\_ERROR(-20001, 'No se pueden agregar más de 5 detalles a este pedido.');

END IF;

END;

/

1. Solicitar un código de cliente por teclado; si existe se inserta su código en una tabla temporal y el mensaje 'EXISTE', y si no existe se dispara la excepción NO\_DATA\_FOUND y se inserta en la tabla temporal el mensaje 'NO EXISTE'.

**RESPUESTA:**

DROP TABLE temporal;

CREATE TABLE temporal(

clid NUMBER(6,0),

INFO VARCHAR2(50 BYTE)

);

DECLARE

cliente\_id NUMBER;

cliente\_id2 NUMBER;

BEGIN

cliente\_id := &id;

SELECT customer\_id INTO cliente\_id2 FROM customer WHERE customer\_id = cliente\_id;

INSERT INTO temporal VALUES (cliente\_id2,'EXISTE');

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

INSERT INTO temporal VALUES (cliente\_id,'NO EXISTE');

END;

/

SELECT \* FROM temporal;

1. Se solicita un código de artículo por teclado y se inserta en una tabla temporal el precio y la descripción de este artículo si está pedido, es decir, si su código está en la tabla detalle. Si el artículo no está pedido, se genera una excepción de usuario con el mensaje "El artículo (código) no lo ha pedido ningún cliente"

**RESPUESTA:**

DECLARE

codigo\_articulo NUMBER;

precio NUMBER;

descripcion VARCHAR2(999);

BEGIN

-- Solicitar el codigo del articulo

codigo\_articulo := &codigo\_articulo;

-- Verificar si el artículo esta

SELECT

precio,

descripcion

INTO

precio,

descripcion

FROM

detalles

WHERE

codigo\_articulo = codigo\_articulo;

-- insertar en la tabla temporal

INSERT INTO tabla\_temporal (

codigo\_articulo,

precio,

descripcion

) VALUES (

codigo\_articulo,

precio,

descripcion

);

-- Si no se encuentra , lanzar una excepción

EXCEPTION

WHEN no\_data\_found THEN

raise\_application\_error(-20001, 'El artículo ('

|| codigo\_articulo

|| ') no lo ha pedido ningún cliente');

END;

/

1. Modificar el ejercicio 2 añadiendo excepciones, de tal forma que en el gestor de excepciones controlemos que select ha fallado, insertando en una tabla temporal cliente no existe, o bien articulo no existe, según haya fallado uno u otro.

**RESPUESTA:**

1. Crear un trigger que se ejecutará automáticamente cuando se elimine algún empleado en la tabla correspondiente visualizando el número y el nombre de los empleados borrados.

**RESPUESTA:**

DROP TABLE temporal;

CREATE TABLE temporal(

empleado\_id NUMBER(4,0),

apellido\_empleado VARCHAR2(20),

);

CREATE OR REPLACE TRIGGER BorraEmpleado

AFTER DELETE ON employee

FOR EACH ROW

BEGIN

INSERT INTO temporal VALUES(:OLD.employee\_id, :OLD.last\_name);

END;

/

1. Crear un procedimiento que reciba como parámetro un código de empleado, y modificar el salario de un empleado en función del número de empleados que tiene a su cargo:
   1. si no tiene ningún empleado a su cargo subirle 50 euros.
   2. si tiene 1 empleado a su cargo subirle 80 euros.
   3. si tiene 2 empleados a su cargo subirle 100 euros.
   4. si tiene más de tres empleados a su cargo subirle 110 euros.
   5. si es el PRESIDENTE su salario se incrementa en 30 euros.

**RESPUESTA:**

1. Hacer dos bloques PL/SQL uno de los cuales escribe ‘Hola mundo’ al derecho y otro del revés.

**RESPUESTA:**

1. Hacer un bloque o proceso que muestre por pantalla el nombre y el jefe de cada empleado, con WHILE.

**RESPUESTA:**

1. Visualizar los apellidos de los empleados pertenecientes al departamento pasado por parámetro, numerándolos secuencialmente.

**RESPUESTA:**

1. Escribir un procedimiento que reciba todos los datos de un nuevo empleado y procese la transacción de alta, gestionando los siguientes errores:
   1. no\_existe\_departamento
   2. no\_existe\_director
   3. numero\_empleado\_duplicado
   4. salario nulo con RAISE\_APPLICATION\_ERROR
   5. Otros posibles errores de Oracle visualizando código y mensaje de error.

**RESPUESTA:**

1. Escribe un procedimiento (recibe como parámetro el oficio) el procedimiento subirá el sueldo de todos los empleados que ganen menos del salario medio de su oficio. La subida será del 50% de la diferencia entre el salario de los empleados y la media de su oficio. Se deberá gestionar los posibles errores.

**RESPUESTA:**

1. Escribir un programa que incremente el salario de los empleados de un determinado departamento que se pasará como primer parámetro. El incremento será una cantidad en euros que se pasará como segundo parámetro en la llamada. El programa deberá informar del número de filas afectadas por la actualización. Los salarios se han de actualizar individualmente.

**RESPUESTA:**

1. Crear un programa que reciba un número de empleado y una cantidad que se incrementará al salario del empleado correspondiente. Utilizar una excepción definida por el usuario denominada salario\_nulo y la predefinida when no\_data\_found

**RESPUESTA:**