Programación de bases de datos

Ejercicios en Oracle:

```
/*PL/SQL utiliza cursores para gestionar las instrucciones SELECT. Un cursor es un conjunto de registros devuelto por
una instrucción SQL. Técnicamente los cursores son fragmentos de memoria reservados para procesar los resultados
de una consulta SELECT
Hay una forma clásica-extendida de utilizar los cursores (pg 241 del libro):

Metemos el cursor dentro de un procedure
*/
SET SERVEROUTPUT ON
create or replace procedure proccurhoteles1 IS

 CURSOR CurHoteles1 IS -- CurHoteles1 es un cursor explícito
       SELECT * FROM Hotel; -- Almacena varios registros
    registro CurHoteles1%ROWTYPE; -- Es necesario declarar esta variable
BEGIN
OPEN CurHoteles1; -- Abrimos cursor
LOOP
    FETCH CurHoteles1 INTO registro; -- Recuperamos un registro
    EXIT WHEN CurHoteles1%NOTFOUND; -- Salimos si no hay más registros
    DBMS_OUTPUT.PUT_LINE ('Cód. Hotel : ' || registro.ID);
    DBMS_OUTPUT.PUT_LINE ('Habitaciones: ' || registro.NHABS);
    DBMS_OUTPUT.PUT_LINE (' ---- ');
END LOOP;
CLOSE CurHoteles1; -- Cerramos cursor
END;
/


/*hacemos la llamada con call proccurhoteles1()*/
```

```
/*
Excepciones: identificador SQL que surge durante la ejecución del código, causado por:
1- un error detectado por Oracle (por ejemplo si un SELECT no devuelve datos ocurre el error ORA-01403
llamado NO_DATA_FOUND).
2-Que el propio programador las lance (comando RAISE).
La captura se realiza utilizando el bloque EXCEPTION. Revisar listado en página 270


*/

SET SERVEROUTPUT ON

create or replace procedure procExcep IS
    x1 NUMBER := 0;
    y1 NUMBER := 3;
    res1 NUMBER;
BEGIN
    res1:=y1/x1;
    DBMS_OUTPUT.PUT_LINE(res1);
EXCEPTION
    WHEN ZERO_DIVIDE THEN
        DBMS_OUTPUT.PUT_LINE('No se puede dividir por cero') ;
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error inesperado') ;
END;
/

--call procExcep()
```

```
/* TRIGGER
Cuándo se ejecuta: BEFORE-AFTER
Evento que da lugar a la ejecución del trigger: INSERT, UPDATE, DELETE
 Para determinar la operación que se ha disparado: INSERTING, UPDATING UPDATING(COL), DELETING
*/
/*Crear un trigger cada vez que se modifique la tabla empleados*/
/*tabla de auditoria*/
CREATE TABLE audi_empleados (
    oper varchar2(50),
    usuario varchar2(50),
    fecha date
);

CREATE OR REPLACE TRIGGER controlEmpleados
    AFTER INSERT OR DELETE OR UPDATE ON empleados FOR EACH ROW
BEGIN
    IF INSERTING THEN
        insert into audi_empleados values ('insert', USER, sysdate);
    ELSIF UPDATING THEN
        insert into audi_empleados values ('update', USER, sysdate);
    ELSIF DELETING THEN
        insert into audi_empleados values ('delete', USER, sysdate);
    END IF;

END;
/
```

Ejercicios en MySQL:

```
DELIMITER $$
/* Procedimiento que usa un cursor para contar el total de jardineria.clientes ***CON WHILE*/
DROP PROCEDURE if EXISTS test.proccursor2;
CREATE PROCEDURE test.proccursor2()
BEGIN
    DECLARE tmp VARCHAR(200);
    DECLARE total INT;
    DECLARE l_last_row_fetched BOOL;

    /* Declaramos el cursor para almacenar lo registros*/
    DECLARE CURSOR2 CURSOR FOR
            SELECT CodigoCliente FROM jardineria.clientes;
    /* Declaramos el manejador*/
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET l_last_row_fetched=1;
    /*Inicializamos las variables*/
    SET total=0, l_last_row_fetched=0; /* 0 false, 1 true*/

    /*  Abrimos el cursor y lo recorremos */
    OPEN cursor2;
        loopcursor2: WHILE (l_last_row_fetched=0) DO  /* mientras last_row es falso*/
            FETCH cursor2 INTO tmp;
                SET total=total + 1;
                IF l_last_row_fetched = 1 THEN
                    LEAVE loopcursor2;
                END IF;
        END WHILE loopcursor2;
    CLOSE CURSOR2; /* Cerramos el cursor*/
    SELECT CONCAT ("Total registros tabla clientes: ",total) AS TOTAL;

END; $$
```

```
DELIMITER $$
/* Procedimiento que usa un cursor para los pagos de jardineria.clientes ***CON WHILE*/
DROP PROCEDURE if EXISTS test.proccursoract9;
CREATE PROCEDURE test.proccursoract9(importe INT)
BEGIN
    DECLARE vCliente VARCHAR(50);
    DECLARE vTotal VARCHAR(50);
    DECLARE cad VARCHAR(2000);

    DECLARE l_last_row_fetched BOOL;

    /* Declaramos el cursor para almacenar lo registros*/
    DECLARE cursoract9 CURSOR FOR
        SELECT C.NombreCliente, SUM(P.Cantidad) as totalPagos
            FROM jardineria.clientes C natural join jardineria.pagos P
            GROUP BY C.CodigoCliente
            HAVING SUM(P.Cantidad) > importe
            ORDER BY 2 DESC ;

    /* Declaramos el manejador*/
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET l_last_row_fetched=1;
    /*Inicializamos las variables*/
    SET l_last_row_fetched=0; /* 0 false, 1 true*/
    SET cad="";
    /*  Abrimos el cursor y lo recorremos */
    OPEN cursoract9;
        loopcursoract9: WHILE (l_last_row_fetched=0) DO  /* mientras last_row es falso*/
            FETCH cursoract9 INTO vCliente,vTotal;
                SET cad=CONCAT(cad,"; ", vCliente, " ", vTotal," ");
                IF l_last_row_fetched = 1 THEN
                    LEAVE loopcursoract9;
                END IF;
        END WHILE loopcursoract9;
    CLOSE cursoract9; /* Cerramos el cursor*/
    SELECT CONCAT ("Total clientes con importe mayor de: ",importe,"***** ",cad) AS TOTAL;

END; $$
```

```
DELETE FROM sakila.actor WHERE actor_id IN (444,445,446);

insert into sakila.actor
values (444, 'Juan', 'Garcia',CURRENT_TIME),
    (445, 'Juan', 'Garcia',CURRENT_TIME),
    (446, 'Juan', 'Garcia',CURRENT_TIME);

DROP TABLE if EXISTS test.auditaGarcia;
create table test.auditaGarcia (
    col1 VARCHAR(200)
);

DELIMITER $$
DROP TRIGGER if EXISTS sakila.ANTES_BORRAR_ACTOR;

CREATE TRIGGER sakila.ANTES_BORRAR_ACTOR BEFORE DELETE ON sakila.ACTOR
FOR EACH ROW
BEGIN
    INSERT INTO test.auditaGarcia SELECT CONCAT("El actor ", OLD.actor_id, " de apellido ",
        old.last_name, " fue borrado el día " ,NOW());

END;
```