EXCEPCIONES:

1- Crear una SELECT (no un cursor explícito) que devuelva el nombre de un empleado pasándole el EMPLOYEE_ID en el WHERE,

• Comprobar en primer lugar que funciona pasando un empleado existente

• Pasar un empleado inexistente y comprobar que genera un error

• Crear una zona de EXCEPTION controlando el NO_DATA_FOUND para que pinte un mensaje como “Empleado inexistente”

```
DECLARE
dataEmpleado employees%rowtype;
BEGIN
     SELECT * INTO dataEmpleado from employees WHERE EMPLOYEE_ID=100;
    dbms_output.put_line(dataEmpleado.first_name);
EXCEPTION
    WHEN NO_DATA_FOUND THEN
    dbms_output.put_line('Empleado inexistente');
END;
```

```
Steven

Procedimiento PL/SQL terminado correctamente.
```

```
DECLARE
dataEmpleado employees%rowtype;
BEGIN
     SELECT * INTO dataEmpleado from employees WHERE EMPLOYEE_ID=1;
    dbms_output.put_line(dataEmpleado.first_name);
EXCEPTION
    WHEN NO_DATA_FOUND THEN
    dbms_output.put_line('Empleado inexistente');
END;
```

```
Empleado inexistente

Procedimiento PL/SQL terminado correctamente.
```

2- Modificar la SELECT para que devuelva más de un empleado, por ejemplo poniendo EMPLOYEE_ID> 100. Debe generar un error. Controlar la excepción para que genere un mensaje como “Demasiados empleados en la consulta”

```
DECLARE
dataEmpleado employees%rowtype;
BEGIN
     SELECT * INTO dataEmpleado from employees WHERE EMPLOYEE_ID>100;
    dbms_output.put_line(dataEmpleado.first_name);
EXCEPTION
    WHEN NO_DATA_FOUND THEN
    dbms_output.put_line('Empleado inexistente');
    WHEN TOO_MANY_ROWS THEN
    dbms_output.put_line('Demasiados empleados en la consulta ');
END;
```

```
Demasiados empleados en la consulta

Procedimiento PL/SQL terminado correctamente.
```

3- Modificar la consulta para que devuelva un error de división por CERO, por ejemplo, vamos a devolver el salario en vez del nombre y lo dividimos por 0. En este caso, en vez de controlar la excepción directamente, creamos una sección WHEN OTHERS y pintamos el error con SQLCODE y SQLERR

```
DECLARE
salaryEntreCero employees.salary%type;
BEGIN
     SELECT salary/0 INTO salaryEntreCero from employees WHERE EMPLOYEE_ID=100;
    dbms_output.put_line(salaryEntreCero);
EXCEPTION
    WHEN NO_DATA_FOUND THEN
    dbms_output.put_line('Empleado inexistente');
    WHEN TOO_MANY_ROWS THEN
    dbms_output.put_line('Demasiados empleados en la consulta ');
    when others then
    dbms_output.put_line(SQLCODE);
    dbms_output.put_line(SQLERRM);
END;
```

```
-1476
ORA-01476: el divisor es igual a cero

Procedimiento PL/SQL terminado correctamente.
```

4- El error -00001 es clave primaria duplicada.

a) Aunque ya existe una predefinida (DUP_VAL_ON_INDEX) vamos a crear una excepción no -predefinida que haga lo mismo.
b) Vamos a usar la tabla REGIONS para hacerlo más fácil
c) Usamos PRAGMA EXCEPTION_INIT y creamos una excepción denominada “duplicado”.
d) Cuando se genere ese error debemos pintar “Clave duplicada, intente otra”.

```
DECLARE
    DUPLICADO EXCEPTION;
    PRAGMA EXCEPTION_INIT(DUPLICADO,-00001);
BEGIN
     INSERT INTO REGIONS VALUES ('1', 'kazajistan');
EXCEPTION
    WHEN DUPLICADO THEN
     dbms_output.put_line('Clave duplicada, intente otra');
END;
```

```
Clave duplicada, intente otra

Procedimiento PL/SQL terminado correctamente.
```

Práctica con EXCEPCIONES DE USUARIO

1- Crear una Excepción personalizada denominada

CONTROL_REGIONES.

• Debe dispararse cuando al insertar o modificar una región

queramos poner una clave superior a 200. Por ejemplo usando una

variable con ese valor.

• En ese caso debe generar un texto indicando algo así como

“Codigo no permitido. Debe ser inferior a 200”

• Recordemos que las excepciones personalizadas deben

dispararse de forma manual con el RAISE.

```
DECLARE
    CONTROL_REGIONES EXCEPTION;
    datoDummyRegionID regions.region_id%TYPE:=201;
BEGIN
    if datoDummyRegionID>200 then
    raise CONTROL_REGIONES;
    end if;
     INSERT INTO REGIONS VALUES (datoDummyRegionID, 'region numero:'||datoDummyRegionID);
EXCEPTION
    WHEN CONTROL_REGIONES THEN
     dbms_output.put_line('Codigo no permitido. Debe ser inferior a 200');
END;
```

```
Codigo no permitido. Debe ser inferior a 200

Procedimiento PL/SQL terminado correctamente.
```

Práctica con RAISE_APPLICATION_ERROR

1. Modificar la practica anterior para disparar un error con RAISE_APPLICATION

en vez de con PUT_LINE.

a. Esto permite que la aplicación pueda capturar y gestionar el error que

devuelve el PL/SQL

```
DECLARE
    CONTROL_REGIONES EXCEPTION;
    datoDummyRegionID regions.region_id%TYPE:=201;
BEGIN
    if datoDummyRegionID>200 then
    raise CONTROL_REGIONES;
    end if;
     INSERT INTO REGIONS VALUES (datoDummyRegionID, 'region numero:'||datoDummyRegionID);
EXCEPTION
    WHEN CONTROL_REGIONES THEN
     RAISE_APPLICATION_ERROR(-20420,'El codigo debe serinferior a 200');
END;
```

```
Informe de error -
ORA-20420: El codigo debe serinferior a 200
ORA-06512: en línea 11
```