EFA
MORATALAZ

Tema 5: PL/SQL.

Requisitos de entrega

1. Se pide entregar los dos ficheros que se indican a lo largo de la presente práctica. Estos dos ficheros se deberán entregar en uno solo el cual deberá tener el siguiente nombre **BBDD_UT5_PRAC_NOMBRE_APELLIDO1_APELLIDO2.sql (Se deben respetar los caracteres en mayúsculas)**.
2. Esta entrega se tendrá que realizar a través de EducamosCLM con su usuario de centro.

Penalizaciones

1. Si algunos de los ficheros que se indican no tiene el nombre exacto que se **solicita se restarán 1,5 puntos a la nota final.**
2. El código debe estar correctamente sangrado, si este no lo está **se restará 1 punto a la nota final.**
3. Cada fichero debe tener un comentario en el que se indique a que ejercicio corresponda la query que se muestra, sino aparece **se restará 1 punto a la nota final.**

Cree un nuevo usuario llamado **SCOTT_PL_SQL** con passwd **SCOTT_PL_SQL**, posteriormente cree una nueva conexión y ejecute el script **BBDD_SCOTT.sql** que se proporciona y a continuación realice los siguientes ejercicios:

NOTA: Las instrucciones para crear el usuario deben estar incluidas en la resolución de este ejercicio.

NOTA 2: Para todos los ejercicios debe tenerse un bloque anónimo en el que se valide el ejercicio que se esta realizando, permitiendo al usuario insertar la información que se desee.

EJERCICIOS DE BLOQUES ANONIMOS

1. Crear una tabla VENDEDORES, compuesta por tres columnas: nº empleado, salario y comisión. Hacer un programa PL/SQL1que para todos los empleados de la tabla EMP que tengan comisión no nula los inserte en la tabla creada anteriormente.

```
CREATE TABLE vendedores (
    Empno NUMBER (4),
    Sal NUMBER (7,2),
    Comm NUMBER (7,2));
```

2. Actualizar los vendedores con una comisión mayor que 350$ con un incremento del 15% de su salario. Si la operación afecta a más de tres empleados, deshacer la transacción, en cualquier otro caso validar la transacción. Introducir en la tabla TEMP la operación que se ha realizado. **NOTA: Para este ejercicio use la función SQL%ROWCOUNT, que determina cuantos registros se han modificado/eliminado/intertado tras realizar una operación CRUD.**

```
CREATE TABLE temp (
    Código NUMBER (7),
    Mensaje VARCHAR2 (80));
```

3. Actualizar el trabajo a DIRECTOR a todos aquellos empleados cuyo salario sea mayor que 2000$. Almacenar el número de empleados actualizados por la operación en la tabla TEMP. Si los afectados son más de cinco personas, borrar los empleados cuyo salario sea mayor que 3000$, insertar en la tabla TEMP el número de empleados borrados y validar la transacción. **NOTA: Para este ejercicio use la función SQL%ROWCOUNT, que determina cuantos registros se han modificado/eliminado/intertado tras realizar una operación CRUD.**

4. Insertar en la tabla TEMP 100 filas. En la primera columna se insertará un índice secuencial (1, 2, 3...) y en la segunda columna un comentario indicando si el número generado es par o impar.

5. Crear una tabla MAS_PAGADOS, compuesta de dos columnas, nombre y salario. Insertar en esta tabla el nombre y el salario de los cinco empleados mejor pagados de la tabla EMP. **Lo desarrollare en clase**

```
CREATE TABLE mas_pagados (
    Nombre VARCHAR2 (10),
    Salario NUMBER (7,2));
```

6. Realizar un bloque PL/SQL en el que se determine el total de ganancias de los empleados (salario y comisión) para el departamento 20, cuantos empleados de este departamento tienen su salario por encima de 2000$ y cuantos tienen la comisión mayor que su salario.

7. Calcular por medio de un bloque PL/SQL anónimo el total de ganancias de los empleados (salario y comisión) y cuántos de éstos tienen un salario superior a 2000$ para todos los departamentos de la empresa.
   Almacenar el resultado en la tabla TEMP2. **Lo desarrollare en clase**

```
CREATE TABLE temp2 (
    Código: NUMBER (5),
    Importe: NUMBER (7,2),
    Mensaje: VARCHAR2 (50));
```

EJERCICIOS FUNCIONES Y PROCEDIMIENTOS

8. Crear el procedimiento QUIEN_EJECUTA, que inserte en la tabla TABLA_LOG el nombre del usuario y la fecha del sistema.

```
CREATE TABLE tabla_log (

Id_usuario VARCHAR2(30),

Fecha DATE);
```

9. Crear un procedimiento que actualice el salario de un empleado pasado por parámetro, aumentándolo en un porcentaje que también se pasará como argumento. Registrar en la TABLA_LOG el usuario y la fecha en que se ejecuta el procedimiento.

10. Crear una función que muestre la suma de los salarios de un departamento dado por un parámetro.

11. Hacer un procedimiento ALTA_SAL_DEPT al que se le pase un número de departamento desde el entorno de llamada, e inserte en la tabla SALARIO_DEPT el número de departamento y la suma de salarios de los empleados de ese departamento. Si el departamento no existe, mostrar un mensaje por pantalla.

```
CREATE TABLE salario_dept (

    Deptno NUMBER (2),

    Sal_tot NUMBER (7,2));
```

12. Crear una función que devuelva el salario medio de los empleados en un departamento determinado y con un trabajo determinado. Estos valores se pasarán como argumentos de entrada. Controlar excepciones.

13. Crear una función que devuelva el mayor y menor de 3 números.

EJERCICIOS CURSORES

14. Ejercicio sobre la Biblioteca. Con motivo del 10º aniversario de la Biblioteca, queremos regalar unos libros a los socios más fieles. Para ello tendremos en cuenta la antigüedad, el uso de la biblioteca, así como el sexo.
    A los socios/as con más de 3 años de antigüedad, le vamos a regalar 1 ó 2 libros dependiendo del número de préstamos solicitados;
    - Hombres:
        - Se le regalará el ejemplar: 'Las edades del Hombre' si han solicitado 2 o menos libros en los 2 últimos años.
        - Si el número de préstamos es superior a 2, también se le regalará el ejemplar: 'Como ser un buen DBA Oracle'.
    - Mujeres:
        - Se le regalará el ejemplar: 'D. Quijote de la Mancha'. si han solicitado 2 o menos libros en los 2 últimos años.
        - Si el número de préstamos es superior a 2, también se le regalará el ejemplar: 'La vida es sueño'.

    Realizar código que extrae el nombre del socio con los volúmenes a regalar. Mostrar también la información de los socios que no tienen regalo, con el nombre del mismo y el texto 'Sin regalo'.

15. Incrementar la comisión, en función del salario, de los empleados de Boston y Nueva York según su antigüedad y cargo, según la siguiente tabla:
    - Boston:
        - Antigüedad. Incremento
            - < 10 años → 10%
            - >=10 y <= 20 → 15%
            - >20 → 20%
    - New York:
        - Cargo Incremento
            - Manager → 0%
            - Analyst →12%
            - Otros →17%

EJERCICIOS TRIGGERS

16. Crear un trigger que por cada borrado en la tabla DEPT guarde los valores en una tabla de históricos.

```
CREATE TABLE historicos_dept (
    DEPTNO NOT NULL NUMBER (2),
    DNAME VARCHAR2 (14),
    LOC VARCHAR2 (13));
```

17. Crear un disparador de base de datos sobre la tabla DEPT que compruebe que el número de departamento es múltiplo de 10 por cada inserción o actualización en esta tabla. Si no lo es, cambiar por el múltiplo de 10 más próximo.

18. Crear un disparador en el cual cada vez que se haga una operación de modificado sobre la tabla EMP, realice una actualización de la columna salario_total de la tabla SALARIO_DEPARTAMENTOS (tabla que contiene la suma de salarios por departamento), es decir, que cuando se inserte una nueva fila, se borre o se actualice el salario controlar el salario total por departamentos.

19. Crear un trigger de fila sobre la tabla EMP que se dispare cuando se actualice o inserte sobre esta tabla. Comprobar que el salario se encuentra entre el máximo y el mínimo de ese trabajo para ese departamento (Tabla Salgrade, pasar el campo Grade a Departamento multiplicándolo * 10). Si supera el máximo o inferior al mínimo, poner dicho máximo o mínimo en la Inserción o Actualización que ha realizado el usuario. Si el departamento, si no existe, crearlo.

EJERCICIOS PAQUETES

20. Crear un paquete GESTION_DE_DEPART, en el cual se van a declarar varios procedimientos y una función:
    - ALTA_DEPT con tres argumentos: número, nombre y localidad del departamento. Este procedimiento debe comprobar que los números de departamento son múltiplos de 10, si no es así daría un error.
    - BAJA_DEPT con un argumento. Dado un número de departamento, borrarlo.

    - MOD_DEPT (función) con dos argumentos. Dado un número de departamento, actualizar la localidad al destino indicado en el parámetro. Informar con un booleano si se ha realizado correctamente la operación.

21. Crear un paquete GESTION_DE_EMPLEADOS, en el que se declaren tres procedimientos:

    - ALTA_EMP con tres argumentos: nombre del empleado, trabajo y jefe. Este procedimiento debe insertar un empleado en la tabla EMP. El resto de los datos se calcularán de la siguiente forma: el número de empleado será el siguiente al último insertado, el salario se obtendrá invocando a la función SALARIO_MEDIO, la fecha de alta, la del sistema, y el departamento será el mismo que el de su jefe. La comisión, dependiendo del trabajo, si es vendedor =0, si no lo es, comisión nula.

    - BAJA_EMP con un argumento. Dado un número de empleado, borrarlo. Si el empleado es jefe de un grupo, asignar como nuevo jefe del grupo al empleado de mayor salario, a este último asignarle el jefe de su antiguo jefe y a cada uno de los empleados del grupo, actualizarles el jefe.

    - MOD_EMP con dos argumentos. Dado un número de empleado, actualizar su departamento. Si el departamento no existe, darlo de alta utilizando algún procedimiento almacenado.