Ejercicios BD06 - programas mysql

Base de datos **BD06_1819:**

1. Crea un procedimiento (**ejemplo1_sp.sql**) simple que muestre por pantalla el texto "esto es un ejemplo de procedimiento" (donde la cabecera de la columna sea “mensaje”).
2. Crea un procedimiento (**ejemplo2_sp.sql**) donde se declaren tres variables internas, una de tipo entero con valor de 1, otra de tipo varchar(10) con valor nulo por defecto y otra de tipo decimal con 4 dígitos y dos decimales.
   Dentro del procedimiento, modificar el valor de la variable entera, la variable de tipo texto, realizar alguna operación matemáticas con las variables y mostrar los resultados en una select.
3. Crea un procedimiento (**ejemplo3_sp.sql**) donde se declaren dos variables internas, una de tipo char y otra de tipo enum.
   Dentro del procedimiento, modificar el valor de la variables y mostrar el resultado.
   Provocar diferentes errores al asignar valores no permitidos a las variables
4. Crea un procedimiento (**ejemplo4_sp.sql**) donde se declaren varias variables internas, unas de tipo entera y otra de tipo fecha. Poner por defecto la fecha actual.
   Dentro del procedimiento, modificar el valor de la variables y mostrar el resultado.
   Provocar diferentes errores al asignar valores no permitidos a las variables.
5. Asignar cualquier valor a una variable definida por el usuario de distintas formas (mediante una select y mediante la clausula SET)
   Crear un procedimiento (**ejemplo5_sp.sql**) que modifique dicha variable, muestre el valor de dicha variable tanto dentro del procedimiento, como fuera de él.
6. Crear una tabla llamada usuarios, que almecene varios usuarios y sus passwords.
   Crear un procedimiento (**ejemplo6_sp.sql**) que reciba dos parámetros de entrada (un usuario y un password) y que inserte un nuevo usuario con esos datos. Comprobar que si modificamos una de las variables de entrada dentro del procedimiento, fuera de él la variable no cambia (los parámetros de entrada son una copia de los parámetros originales).
7. Crear un procedimiento (**ejemplo7_sp.sql**) con una variable de de salida (paso de parámetros por variable) y comprueba que por defecto su valor es Nulo. Modifica su valor y muestralo tanto dentro del procedimiento como fuera de él.
8. Crea un procedimiento (**ejemplo8_sp.sql**) con una variable de entrada-salida. El procedimiento debe cambiar el valor de la vriable. Realizar una prueba mostrando el valor de la variable antes y despúes de llamar al procedimiento.
9. Crea un procedimiento (**ejemplo9_sp.sql**) que compruebe si un usuario existe o no en la base de datos. Mostrar un mensaje en forma de select.
10. Crea un procedimiento (**ejemplo10_sp.sql**) con dos parámetros de entrada (usuario y contraseña) y uno de salida (mensaje) Hacer uso de IF, ELSE para comprobar si un usuario pasado por parámetros existe en la base de datos con esa contraseña. Devolvemos en la variable de salida estos tres posibles valores: usuario no existe, password incorrecto o usuario correcto
11. Crea una nueva tabla **datos_personales** con los campos: usuario, edad, ultimo_login, num_accesos, (donde usuario es clave foránea de la tabla creada anteriormente). Insertar algunos datos de ejemplos.
    Crea un procedimiento (**ejemplo11_sp.sql**) Hacer uso de **case...when..** para comprobar

si un usuario pasado por parámetros es mayor de edad o no.

12. Crea un procedimiento (**ejemplo12_sp.sql**) que borre si existe la tabla artículos. Si no existe debe crearla e insertar 10 artículos haciendo uso de un buce WHILE...DO
13. Hacer un procedimiento (**ejemplo13_sp.sql**) de forma que cuando un usuario se autentifique (comprobar que el usuario existe en la BD), actualice la tabla datos_personales (aumente el número de acceso y almacene la fecha actual). Si por casualidad no existen datos en la tabla datos_personales para ese usuario, hay que añadirlos.
    Mostrar los datos de la tabla datos personales para ese usuario.
14. Crea un procedimiento (**ejemplo14_sp.sql***) que pasandole un número muestre una select para cada número comprendido entre el número y 1. Si el número es 0 o negativo no mostrará nada. Hacer uso de LOOP para implementar dicho procedimiento.
15. Crear un procedimiento (**ejemplo15_sp.sql**) que pasándole un número, inserte en la tabla de artículos el artículo con clave el número p, controlar los errores de clave duplicada y error al insertar null como clave.
16. Crear una función (**ejemplo16_fun.sql**) de manera que reciba una cantidad de euros y devuelva el valor expresado en pesetas. Llamar a dicha función en el interior de una select.
17. Crea una función (**ejemplo17_fun.sql**) de forma que pasandole un usuario nos devuelva su edad actual (debe comprobar su fecha de nacimiento en el caso de existir y calcular su edad actual).
18. Crea una tabla **sistema** que almacene el número total de usuarios actuales y la fecha de la última actualización (último usuario, modificado o borrado).
    Crear un disparador (**ejemplo18_trig.sql**) sobre la tabla usuarios de forma que cuando se cree un usuario nuevo, actualice dicha tabla.

Base de datos **torneo_futbol:**

1. Crear un procedimiento llamado **crear_tabla_aux_partido** de forma que copie la tabla partido con el nombre de aux_partido, elimine la columna resultado y añada dos columnas nuevas, goles_local y goles_visitante.
2. Crear un procedimiento **cargar_datos_aux_partido**, de forma que compruebe si existe la tabla aux_partido. En el caso de no existir debe llamar al procedimiento **crear_tabla_aux_partido**. SI la tabla existe, debe cargar los datos de la tabla partido en la nueva tabla **aux_partido**, teniendo en cuenta que resultados es un string y debe almacena su contenido en los campos goles_local y goles_visitante.
3. Crear una función llamada **obtener_puntuacion** que reciba el nombre de un equipo de futbol y devuelva los puntos obtenidos. Si se produce algún tipo de excepción o bien el equipo no existe, devolver -1.
4. Haciendo uso de la función anterior, crear un procedimiento llamado **actualizar_calificacion** que cree una nueva tabla si no existe llamada clasificación, de forma que actualice la clasificación de todos los equipos de la base de datos, indicando el orden de calificación.
5. Crear un procedimiento llamado **crear_tabla_logs_equipos** que cree una tabla temporal nueva si no existe llamada **logs_equipos**, con los campos, fecha actual de tipo timestamp, orden (que puede ser insertar, eliminar o actualizar), nombre del equipo y total de jugadores.
6. Crear un trigger de forma que cuando alguien inserte un jugador, actualice en una tabla temporal llamada **logs_equipos** (la cree si no existe), la fecha actual, la ejecución (“insertar”) el nombre del equipo, y la cantidad total de jugadores.

7. Realizar la misma acción cuando se actualice un jugador existente.
8. Realizar la misma acción cuando se borre un jugador.
9. Crear dos **triggers** de forma que cuando se actualice el resultado de un partido o se inserte el resultado de un partido nuevo se actualice la clasificación del torneo.