EVALUACION	Parcial 2	**GRUPO**	**Todos**	**FECHA**	01/06/2024
MATERIA	Base de Datos 2				
CARRERA	Analista Programador/Analista en Tecnologías de la Información				
CONDICIONES	Puntos 30 Sin material Duración 2 horas				

Dado el siguiente modelo relacional:

Usuarios (**Username**, Nombre, Apellido, Mail, Contraseña, FechaRegistro)

Juegos (**GameID**, Nombre, Desarrollador, Género, Precio, FechaLanzamiento)

Comentarios:(**CommentID**, UserID, GameID, Comentario, Puntuación, FechaComentario)

FK: UserID -> Usuarios.Username
FK: GameID-> Juegos.GameID

Transacciones (TransactionID, UserID, GameID, FechaTransacción, MontoTotal)

FK: UserID -> Usuarios.Username
FK: GameID-> Juegos.GameID

Se pide:

1- **VISTA (6 ptos)**

Escribir una vista que permita mostrar información de los usuarios (usuario, nombre y apellido), la cantidad de juegos que han comentado y el total gastado en transacciones para cada usuario.

```
CREATE VIEW VistaUsuariosComentariosTransacciones AS
     (
      SELECT  U.Username, U.Nombre, U.Apellido, COUNT(C.CommentID) AS
              CantidadComentarios, SUM(T.MontoTotal) AS TotalGastado
      FROM Usuarios U JOIN Comentarios C ON U.Username = C.Username
                      JOIN Transacciones T ON U.Username = T.UserID
      GROUP BY U.Username, U.Nombre, U.Apellido
      );
```

2- **PROCEDIMIENTO ALMACENADO (6 ptos)**

Escribir un procedimiento que reciba un usuario y permita obtener la fecha de su comentario más antiguo, la fecha de su comentario más reciente, y un mensaje si el usuario no tiene comentarios.

DELIMITER //

```
CREATE PROCEDURE ObtenerInfoComentariosUsuario( @p_Username
VARCHAR(50), @FechaComentarioAntiguo DATE OUT,
@FechaComentarioReciente DATE OUT, @Mensaje VARCHAR(255) OUT)

BEGIN

    DECLARE totalComentarios INT;


    SELECT COUNT(CommentID) INTO totalComentarios

    FROM Comentarios WHERE Username = p_Username;


    SET CantidadComentarios = totalComentarios;


    IF totalComentarios = 0 THEN

        SET Mensaje = 'El usuario no tiene comentarios';

        SET FechaComentarioAntiguo = NULL;

        SET FechaComentarioReciente = NULL;

    ELSE

        SELECT MIN(FechaComentario) INTO FechaComentarioAntiguo

        FROM Comentarios WHERE Username = p_Username;


        SELECT MAX(FechaComentario) INTO FechaComentarioReciente

        FROM Comentarios WHERE Username = p_Username;


        SET Mensaje = 'El usuario tiene comentarios';

    END IF;

END
```

3- **FUNCION (6 ptos)**

Crear una función que dado un rango de fechas retorne el nombre del mejor juego, siendo este el que más ha recaudado por concepto de transacciones en el rango de fechas indicado.

En caso de existir más de un juego que cumpla ser el mejor devolver uno cualquiera

```
CREATE FUNCTION FN_MejorJuego( @f1 Date, @f1 date) returns varchar(100)
AS
BEGIN
    Declare @nomJuego varchar(100);
    Declare @gameID int;

    Select @gameID = T.GameID
    From Transacciones T
    Where T. FechaTransacción between @f1 and @f2
    Group by T.GameID
    Order by Sum(T.MontoTotal) asc;

    Select @nomJuego = J.nombre
    From Juegos J
    Where J.GameId = @gameID;

    Return @nomJuego;
END
```

4- **TRIGGER (6 ptos)**

Implementar un trigger que permita solo permita agregar juegos o cambiar los datos de los juegos que aún no han sido lanzados

Considerar múltiples registros

Implementar una solución que afecte a todos los que cumplen la condición y que descarte a los que no.

```
CREATE TRIGGER Trg_ModificacionDatosjuegos
ON Juegos
Instead of insert, update
AS
BEGIN
   If Exists(select * from Inserted) and not Exists(select * from deleted)
      Begin
      Insert into Juegos
            Select I.*
            From Inserted I
            Where I.FechaLanzamiento < getdate() ;
      End
   Else
      Begin
      Update Juegos
      Set Nombre = I.nombre, Desarrollador = I.Desarrollador,
          Genero = I.Genero , Precio = I.Precio,
          FechaLanzamiento = I.FechaLanzamiento
      From Juegos J, Inserted I
      Where J.GameID = I.GameID and I.FechaLanzamiento < getdate()
END
```

5- **TRIGGER (6 ptos)**

Suponer que la tabla juegos tiene un nuevo campo llamado ‘PuntajeAcumulado’
Juegos (**GameID**, Nombre, Desarrollador, Género, Precio, FechaLanzamiento, PuntajeAcumulado)

Implementar un trigger que cada vez que se agregue, eliminen comentarios se actualice el ‘puntajeacumulado’

```
CREATE TRIGGER Trg_CtrlDePuntajes
ON Comentarios
After Insert, Delete
AS
BEGIN
   If Exists(select * from Inserted) and not Exists(select * from deleted)
     begin – operacion disaparadora insert
     Update juegos
     Set puntajeacumulado = puntajeacumulado + I.Puntaje
     From Juegos J, Inserted I
     Where J.GameID = I.GameID
     end
  Else
    begin – operacion disaparadora delete
     Update juegos
     Set puntajeacumulado = puntajeacumulado - D.Puntaje
     From Juegos J, Deleted D
     Where J.GameID = D.GameID
     End;
END
```