

IISSI-1 Examen de laboratorio (Evaluación por curso Diciembre-2023)	
Apellidos, nombre:	Grupo:

## Instrucciones

- Utilice HeidiSQL para abrir la conexión del root con contraseña iissi\$root y:
  - Cree un nuevo usuario cuyo nombre sea UVUS\_USER.
  - Cree una conexión para el nuevo usuario y ejecute el script SQL proporcionado.
- Implemente la solución a los ejercicios planteados. Estos ejercicios están basados en unos requisitos adicionales (Catálogo de requisitos) que se suministran a continuación.
- En cada ejercicio, incluya el código desarrollado (con formato de texto) y las capturas de pantalla que muestren el resultado de ejecutar dicho ejercicio.
- Debe subir a la actividad de EV este documento con el código y las capturas de pantallas. Antes de subir la actividad, avise a un profesor para que valide la misma.

## Catálogo de requisitos

RI-1-01 Requisito de información
<p>Como: Profesor de la asignatura</p> <p>Quiero: Tener disponible información sobre las valoraciones que cada usuario haya podido realizar para cada juego, teniendo en cuenta que un usuario puede valorar muchos juegos, y un juego puede ser valorado por muchos usuarios. Es necesario guardar información de la fecha de la valoración, la puntuación que ha otorgado cada usuario a cada juego, su opinión en forma de texto, el número de “likes” que ha recibido esa valoración por parte de otros usuarios y el veredicto final sobre el juego. Todos los atributos son obligatorios salvo el número de “likes”, que por defecto será 0.</p> <p>Para: Evaluar al estudiante</p>

RN-1-01 Regla de negocio de la puntuación
<p>Como: Profesor de la asignatura</p> <p>Quiero: El usuario podrá otorgar a cada juego una puntuación entre 0 y 5, ambas inclusive.</p> <p>Para: Evaluar al estudiante</p>

RN-1-02 Regla de negocio del veredicto
<p>Como: Profesor de la asignatura</p> <p>Quiero: El usuario asignará un veredicto final a su valoración sobre cada juego, este veredicto podrá ser ‘Imprescindible’, ‘Recomendado’, ‘Comprar en rebajas’, ‘No merece la pena’.</p> <p>Para: Evaluar al estudiante</p>

RN-1-03 Regla de negocio de valoraciones	
Como:	Profesor de la asignatura
Quiero:	Un usuario no podrá hacer más de una valoración sobre un juego determinado.
Para:	Evaluar al estudiante

Prueba1 Inserción de valoración de juegos	
Como:	Profesor de la asignatura
Quiero:	<p>✓ Insertar el progreso del usuario 1 en el juego 2, puntuación = 5, un comentario de texto cualquiera, y veredicto 'Imprescindible', fecha de hoy.</p> <p>✓ Insertar el progreso del usuario 2 en el juego 4, puntuación = 3, un comentario de texto cualquiera, y estado 'Comprar en rebajas', fecha de hoy.</p> <p>✓ Insertar el progreso del usuario 3 en el juego 3, puntuación = 4, un comentario de texto cualquiera, y estado 'Recomendado', fecha de hoy.</p> <p>✓ Insertar el progreso del usuario 4 en el juego 5, puntuación = 1, un comentario de texto cualquiera, y estado 'No merece la pena', fecha de hoy.</p> <p>✓ Insertar el progreso del usuario 2 en el juego 3, puntuación = 4.5, un comentario de texto cualquiera, y estado 'Imprescindible', fecha de hoy.</p> <p>✗ Insertar el progreso del usuario 1 en el juego 6, puntuación = 10, un comentario de texto cualquiera, y estado 'Imprescindible'. (RN-1-01)</p> <p>✗ Insertar el progreso del usuario 3 en el juego 1, puntuación = 3, un comentario de texto cualquiera, y estado 'Ni fu ni fa'. (RN-1-02)</p> <p>✗ Insertar el progreso del usuario 3 en el juego 3, puntuación = 2, comentario 'No era para tanto', y estado 'No merece la pena'. (RN-1-03)</p> <p>✗ Insertar el progreso del usuario 6 en el juego 8, puntuación = 3, un comentario de texto cualquiera, y estado 'Comprar en rebajas'. (referencia no existente)</p>
Para:	Evaluar al estudiante.

### Ejercicio-1 (2 puntos)

Implemente los requisitos proporcionados (RI/RN).

(La tabla puntúa 1 y cada restricción 0,5)

Incluya aquí el código de creación de la nueva tabla y una captura visualizando la tabla creada

```
#Ejercicio 1
CREATE TABLE Valoraciones(
    idValoracion INT NOT NULL AUTO_INCREMENT,
    jugadoresId INT not null,
    videojuegoId INT not null,
    fechaValoracion DATE DEFAULT(CURDATE()),
    puntuacion DECIMAL(2,1) NOT NULL CHECK (puntuacion >=0 AND puntuacion <=5),
    opinion TEXT NOT NULL,
    numLikes INT DEFAULT (0),
    veredictoFinal ENUM ('Imprescindible', 'Recomendado', 'Comprar en rebajas', 'No merece la pena') NOT NULL,
    PRIMARY KEY(idValoracion),
    FOREIGN KEY(jugadoresId) REFERENCES jugadores(jugadorId)
    ON UPDATE RESTRICT
    ON DELETE RESTRICT,
    FOREIGN KEY(videojuegoId) REFERENCES videojuegos(videojuegoId)
    ON UPDATE RESTRICT
    ON DELETE RESTRICT,
    UNIQUE (jugadoresId, videojuegoId)
);
```

Calificación: \_\_\_\_\_

### Ejercicio-2 (2 puntos)

Codifique un procedimiento almacenado **que inserte una nueva valoración de un usuario concreto para un juego dado** en la tabla creada en el ejercicio anterior, que será llamado tantas veces como progresos se deseen añadir.

Para comprobar las RN y las restricciones de Integridad, se llamará al procedimiento con los parámetros que aparecen en **Prueba1-Inserción de valoración de juegos**. Las valoraciones en verde se insertarán y las marcadas en rojo serán rechazadas.

(Si la inserción se realizar directamente con varios comandos INSERT de SQL, entonces se puntúa **0,5**)

Incluya aquí el código del procedimiento y las llamadas correspondientes, o los comandos SQL de inserción, así como una captura del contenido de la tabla y de los mensajes de error de las filas rechazadas.

```
#Ejercicio 2
DELIMITER //
CREATE OR REPLACE PROCEDURE insertarValoracion(
    IN p_jugadorId INT,
    IN p_videojuegoId INT,
    IN p_fechaValoracion DATE,
    IN p_puntuacion DECIMAL(2,1),
    IN p_opinion TEXT,
    IN p_veredicto VARCHAR(64)
)
BEGIN
    INSERT INTO valoraciones(jugadoresId, videojuegoId, fechaValoracion, puntuacion, opinion, veredictoFinal)
    VALUES (p_jugadorId, p_videojuegoId, p_fechaValoracion, p_puntuacion, p_opinion, p_veredicto);

END//
DELIMITER ;

CALL InsertarValoracion(1, 2, CURDATE(), 5., '¡Excelente!', 'Imprescindible');
CALL InsertarValoracion(2, 4, '2024-12-03', 3.0, 'Bueno, pero mejor en oferta.', 'Comprar en rebajas');
CALL insertarValoracion(3, 3, '2024-12-03', 4.0, 'Bueno, pero mejor en oferta.', 'Recomendado');
CALL insertarValoracion(4, 5, '2024-12-03', 1.0, 'Bueno, pero mejor en oferta.', 'No merece la pena');
CALL insertarValoracion(2, 3, '2024-12-03', 4.5, 'Bueno, pero mejor en oferta.', 'Imprescindible');
```

Calificación:

### Ejercicio-3 (1 puntos)

Cree una consulta que devuelva todos los usuarios, sus juegos y las valoraciones respectivas, ordenados por videojuegosId.

Un ejemplo de resultado de esta consulta es el siguiente:

	nickn	nombreJuego	Progres	UsuarioId	VideojuegoId	fecha	puntuacion	comentario	likes	veredicto
1	Currito92	The Legend of Zelda: Tears	1	1	2	2023-06-12	5	Si has jugado a Breath of the wild y te gustó, este te gustará	0	Imprescindible
2	ManTrini67	Maniac Mansion	5	2	3	2000-09-01	5	Mejor que Monkey Island. De los mejores juegos de si	0	Imprescindible
3	ISSI_USER	Maniac Mansion	3	3	3	1990-07-13	4	Historia de los videojuegos. Hay que jugarlo.	0	Recomendado
4	ManTrini67	Horizon: Zero Dawn	2	2	4	2022-05-24	3	Buen juego si te gustan los mapas con muchos iconito	0	Comprar en rebajas
5	Samus	Pokemon Beige Clarito	4	4	9	2023-12-16	1	El pokemon beige oscuro me han dicho que está mejk	0	No merece la pena

Incluya la captura que muestre la consulta y su resultado

```
#Ejercicio 3
SELECT jugadores.nickname, videojuegos.nombre AS nombreJuego, valoraciones.jugadoresId AS UsuarioId,
valoraciones.videojuegoId AS VideojuegosId, videojuegos.fechaLanzamiento AS Fecha, valoraciones.puntuacion AS
Puntuacion, valoraciones.opinion AS Comentario, valoraciones.numLikes AS Likes, valoraciones.veredictoFinal AS veredicto
FROM valoraciones
JOIN jugadores ON (valoraciones.jugadoresId = jugadores.jugadorId)
JOIN videojuegos ON (valoraciones.videojuegoId = videojuegos.videojuegoId)
ORDER BY videojuegos.videojuegoId;
```

Calificación:

#### **Ejercicio-4 (1 puntos)**

Codifique un trigger para impedir que la fecha de una valoración sea anterior a la fecha de lanzamiento del juego, y posterior a la fecha actual. Añada una instrucción que haga saltar el trigger.

**Comentado [JG1]:** Me falta el caso de prueba para los triggers. Yo he metido un par de ellos en las soluciones.

Incluya aquí el código de la función y la llamada correspondiente, así como capturas del contenido de las tablas

```
#Ejercicio 4
DELIMITER //
CREATE TRIGGER ValorarFecha
BEFORE INSERT ON Valoraciones FOR EACH ROW
BEGIN
    DECLARE lanzamiento DATE;
    SELECT fechaLanzamiento INTO lanzamiento
    FROM Videojuegos
    WHERE videojuegoId = NEW.videojuegoId;

    IF NEW.fechaValoracion < lanzamiento OR NEW.fechaValoracion > CURDATE() THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Fecha de valoración no válida';
    END IF;
END;

CALL insertarValoracion(1, 4, '2025-12-01', 4., 'Me ha gustado mucho', 'Imprescindible');
```

Calificación: \_\_\_\_\_

#### **Ejercicio-5 (1 puntos)**

Codifique una función que devuelva el número de valoraciones de un usuario dado.

Realice una prueba de la función con Usuarioid=2.

Incluya aquí el código de la función y la llamada correspondiente, así como capturas del contenido de las tablas

```
#Ejercicio 5
DELIMITER //
CREATE OR REPLACE FUNCTION
fNumValoraciones(usuario INT) RETURNS INT
BEGIN
    RETURN (
        SELECT COUNT(*)
        FROM valoraciones
        WHERE (valoraciones.jugadoresId = usuario)
    );
END//
DELIMITER ;

SELECT fNumValoraciones(2) AS ValoracionesDeUsuario;
```

Calificación: \_\_\_\_\_

#### **Ejercicio-6 (1 puntos)**

Cree una consulta que devuelva los juegos y la media de las valoraciones recibidas, ordenados de mayor a menor. En el listado deben aparecer todos los juegos, tengan o no valoración.

Un ejemplo de resultado de esta consulta es el siguiente:

ASC nombreJuego	mediaValoracion
The Legend of Zelda: Tears of the Kingdom	5
Maniac Mansion	4.5
Horizon: Zero Dawn	3
Pokemon Beige Clarito	1
The Legend of Zelda: Breath of the Wild	[NULL]
Pokemon Amarillo	[NULL]
Super Metroid	[NULL]
Final Fantasy IX	[NULL]
Pokemon Rojo	[NULL]

Incluya la captura que muestre la consulta y su resultado

#Ejercicio 6

```
SELECT videojuegos.nombre, AVG(valoraciones.puntuacion) AS mediaValoracion
FROM videojuegos
LEFT JOIN valoraciones ON (videojuegos.videojuegoId = valoraciones.videojuegoId)
GROUP BY videojuegos.nombre
ORDER BY mediaValoracion DESC;
```

Calificación: \_\_\_\_\_

### Ejercicio-7 (1 puntos)

Cree un trigger que impida valorar un juego que esté en fase 'Beta'. (1 punto)

Incluya aquí el código del procedimiento, la llamada correspondiente, así como capturas de pantalla de las tablas

#Ejercicio 7

```
DELIMITER //
CREATE TRIGGER ImpedirBeta
BEFORE INSERT ON Valoraciones
FOR EACH ROW
BEGIN
    DECLARE estado VARCHAR(20);
    SELECT estado INTO estado
    FROM Videojuegos
    WHERE videojuegoId = NEW.videojuegoId;

    IF estado = 'Beta' THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'No se permite valorar juegos en fase Beta';
    END IF;
END//
DELIMITER ;

CALL insertarValoracion(1, 9, CURDATE(), 4.0, 'muy buen juego', 'Imprescindible');
```

Calificación: \_\_\_\_\_

### Ejercicio-8 (1 puntos)

Cree un procedimiento **pAddUsuarioValoracion** que, dentro de una transacción, inserte un usuario y una valoración de dicho usuario a un videojuego dado. Incluya los parámetros que considere oportunos.

Realice dos llamadas: una que inserte ambos (el usuario y la valoración) correctamente, y una en la que el segundo rompa alguna restricción y aborte la transacción. Incluya capturas de pantallas.

Incluya aquí el código del procedimiento, la llamada correspondiente, así como capturas de pantalla de las tablas

#Ejercicio 8

```
DELIMITER //
CREATE PROCEDURE pAddUsuarioValoracion1(
```

```

IN p_jugadorId INT,
IN p_nickname VARCHAR(255),
IN p_videojuegoId INT,
IN p_fechaValoracion DATE,
IN p_puntuacion DECIMAL(2,1),
IN p_opinion TEXT,
IN p_veredicto VARCHAR(64)
)
BEGIN
-- Iniciar la transacción
START TRANSACTION;

-- Verificar si el jugador ya existe
IF (SELECT COUNT(*) FROM jugadores WHERE jugadorId = p_jugadorId) > 0 THEN
ROLLBACK;
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'El jugador ya existe';
ELSE
-- Insertar nuevo jugador
INSERT INTO jugadores(jugadorId, nickname)
VALUES (p_jugadorId, p_nickname);
END IF;

-- Verificar si el videojuego existe
IF (SELECT COUNT(*) FROM videojuegos WHERE videojuegoId = p_videojuegoId) = 0 THEN
-- Si el videojuego no existe, abortar la transacción
ROLLBACK;
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'El videojuego no existe';
ELSE
-- Insertar la valoración
INSERT INTO valoraciones (
jugadorId, videojuegoId, fechaValoracion, puntuacion, opinion, veredicto
)
VALUES (
p_jugadorId, p_videojuegoId, p_fechaValoracion, p_puntuacion, p_opinion, p_veredicto
);
END IF;

-- Confirmar la transacción
COMMIT;
END //
DELIMITER ;

```

Calificación: \_\_\_\_\_

**Nota importante:** una vez que haya terminado el examen y haya pasado a este documento todas las capturas de pantalla, por favor, elimine su usuario de HeidiSQL y el esquema de la base de datos.

Este mismo documento también debe ser eliminado del ordenador, una vez se haya subido a EV y se haya comprobado que la entrega se ha realizado correctamente.

Esta es mi base de datos:

DROP DATABASE IF EXISTS Videojuegos;

CREATE DATABASE Videojuegos ;

USE Videojuegos;

```
DROP TABLE if EXISTS Progresos;
```

```
DROP TABLE if EXISTS Jugadores;
```

```
DROP TABLE if EXISTS Videojuegos;
```

```
CREATE TABLE Videojuegos(  
    videojuegold INT AUTO_INCREMENT NOT NULL PRIMARY KEY,  
    nombre VARCHAR(80) NOT NULL,  
    fechaLanzamiento DATE,  
    logros INT,  
    estado ENUM('Lanzado', 'Beta', 'Acceso anticipado'),  
    precioLanzamiento DOUBLE  
);
```

```
CREATE TABLE Jugadores(  
    jugadorid INT NOT NULL AUTO_INCREMENT PRIMARY KEY,  
    nickname VARCHAR(60) NOT NULL  
);
```

```
INSERT INTO Videojuegos(nombre, fechaLanzamiento,logros, estado,precioLanzamiento) VALUES ('The  
Legend of Zelda: Breath of the Wild', '2017-03-03', 76, 'Lanzado', 69.99);
```

```
INSERT INTO Videojuegos(nombre, fechaLanzamiento,logros, estado,precioLanzamiento) VALUES ('The  
Legend of Zelda: Tears of the Kingdom', '2023-05-12', 139, 'Lanzado', 79.99);
```

```
INSERT INTO Videojuegos(nombre, fechaLanzamiento,logros, estado,precioLanzamiento) VALUES  
('Maniac Mansion', '1987-01-01', 1, 'Lanzado', 49.98);
```

```
INSERT INTO Videojuegos(nombre, fechaLanzamiento,logros, estado,precioLanzamiento) VALUES  
('Horizon: Zero Dawn', '2017-02-28', 31, 'Lanzado', 79.99);
```

```
INSERT INTO Videojuegos(nombre, fechaLanzamiento,logros, estado,precioLanzamiento) VALUES  
('Super Metroid', '1994-04-28', 1, 'Lanzado', 69.99);
```

```
INSERT INTO Videojuegos(nombre, fechaLanzamiento,logros, estado,precioLanzamiento) VALUES ('Final  
Fantasy IX', '2001-02-16', 9, 'Lanzado', 69.99);
```

```
INSERT INTO Videojuegos(nombre, fechaLanzamiento,logros, estado,precioLanzamiento) VALUES  
('Pokemon Rojo', '1999-11-01', 151, 'Lanzado', 49.98);
```

```
INSERT INTO Videojuegos(nombre, fechaLanzamiento,logros, estado,precioLanzamiento) VALUES  
('Pokemon Amarillo', '2000-06-16', 155, 'Lanzado', 49.98);
```

```
INSERT INTO Videojuegos(nombre, fechaLanzamiento,logros, estado,precioLanzamiento) VALUES  
('Pokemon Beige Clarito', '2023-12-15', 3, 'Beta', 2000000);
```

```
INSERT INTO Jugadores(nickname) VALUES ('Currito92');  
INSERT INTO Jugadores(nickname) VALUES ('MariTrini67');  
INSERT INTO Jugadores(nickname) VALUES ('IISSI_USER');  
INSERT INTO Jugadores(nickname) VALUES ('Samus');  
INSERT INTO Jugadores(nickname) VALUES ('Aran');
```