UF2177: Ejercicio Repaso

Deberás entregar un PDF que contenga el diseño de la base de datos (con una captura del diagrama ER), los scripts de creación e inserción de datos, las respuestas a las consultas, el trigger y el procedimiento almacenado.

El **script de creación** se puede generar directamente usando la opción **Forward Engineer** de MySQL, y los datos de inserción pueden crearse utilizando herramientas de IA.

No se debe crear nada adicional a lo especificado en el enunciado.

Los scripts pueden dividirse hasta un máximo de tres archivos (no más): uno para la creación de tablas, otro para la inserción de datos, y uno más que incluya las consultas, el trigger y el procedimiento.

Además, incluye capturas de pantalla en el PDF mostrando los resultados de las consultas y evidencia de que el trigger y el procedimiento funcionan correctamente.

Recuerda poner el nombre y apellidos en el PDF de entrega y en cada script creado.

Enunciado:

Una organización que realiza competencias de **drones** necesita un sistema para gestionar los **pilotos** que participan, los **drones** que utilizan, las **carreras** realizadas en cada evento, y los **resultados** de cada carrera. Además, quieren mantener un registro del rendimiento de cada piloto y drone en la competencia.

Requerimientos Funcionales:

Drones: Cada drone tiene un número de identificación, un modelo, velocidad máxima, y una capacidad de batería (medida en minutos de vuelo).

Pilotos: Los pilotos que manejan los drones tienen un nombre, edad, y nivel de habilidad (por ejemplo, principiante, intermedio, avanzado).

Carreras: Durante las competencias, se realizan varias carreras. Cada carrera tiene una fecha, una pista de competición y una ciudad de competición.

Resultados: Para cada carrera, se registran los resultados de los drones, incluyendo su posición final, tiempo en la carrera, y si tuvieron algún problema técnico.

Base de Datos:

Crea las siguientes tablas en tu base de datos:

Una tabla para los drones, que incluya el número de identificación, modelo, velocidad máxima y capacidad de batería.

Una tabla para los pilotos, con el nombre, edad, nivel de habilidad.

Una tabla para las carreras, con la fecha, pista de competición y ciudad de la competición.

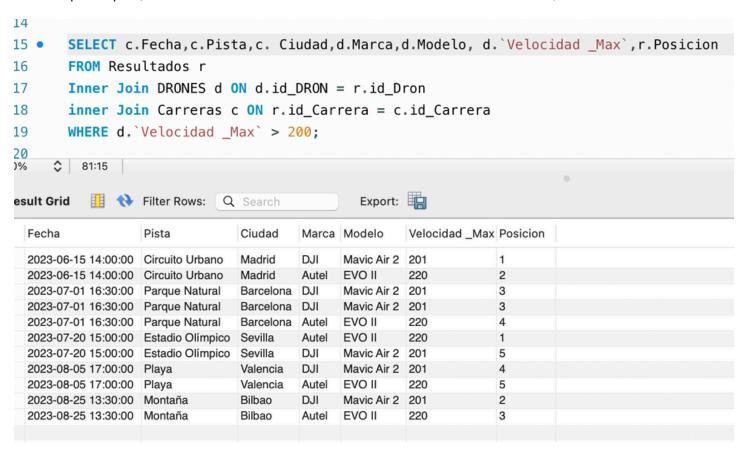
Una tabla para los resultados, que registre para cada carrera el drone, piloto, posición final en la carrera, tiempo de carrera y problemas técnicos (si los hubo).

Consultas:

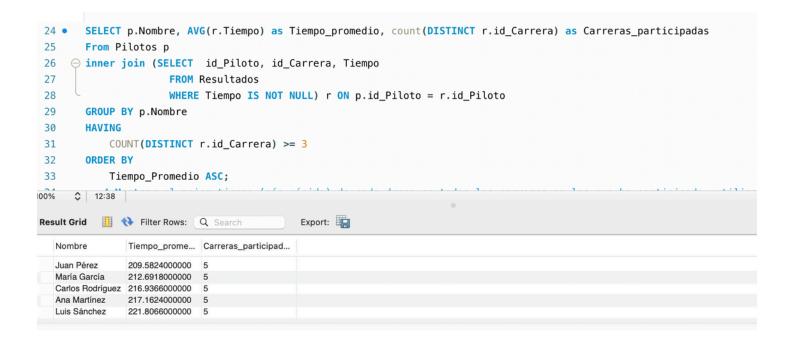
Mostrar todas las carreras en las que participó un piloto específico, incluyendo el drone que utilizó y su posición final. (Usar JOIN entre pilotos, carreras y resultados).



Mostrar todas las carreras en las que un drone con velocidad máxima mayor a 200 km/h participó. (Usar WHERE con una condición sobre la velocidad del drone).



Calcular el tiempo promedio de carrera por cada piloto, utilizando una subconsulta o WITH. Mostrar solo los pilotos que hayan participado en al menos 3 carreras.



Mostrar el mejor tiempo (más rápido) de cada drone en todas las carreras en las que ha participado, utilizando PARTITION BY para agrupar por drone.

```
agrapar por aronce

    WITH RankedTimes AS (

         SELECT d.id_DRON, d.Marca, d.Modelo, r.id_Carrera, r.tiempo,
             ROW_NUMBER() OVER (PARTITION BY d.id_DRON ORDER BY r.tiempo ASC) AS tiempo_rank
        FROM DRONES d
        INNER JOIN Resultados r ON d.id_DRON = r.id_Dron
        WHERE r.tiempo IS NOT NULL
     )
     SELECT
        id_DRON,
        Marca,
        Modelo,
         id_Carrera AS Carrera_Mejor_Tiempo,
        tiempo AS Mejor_Tiempo
     FROM
        RankedTimes -- el with
    WHERE
        tiempo_rank = 1
     ORDER BY
        tiempo ASC;
```

Mostrar el nombre de los pilotos y el número de carreras que han ganado (posición 1). Si no han ganado ninguna carrera, mostrar "Ninguna". Utilizar IFNULL para manejar los casos sin victorias y LOWER para mostrar los nombres de los pilotos en minúsculas.

```
56 •
       SELECT
            LOWER(p.Nombre) AS Nombre_Piloto,
57
            IFNULL(victorias.Carreras_Ganadas, 'Ninguna') AS Carreras_Ganadas
58
59
       FROM
60
            Pilotos p

─ LEFT JOIN (
61
62
            SELECT
63
                id_Piloto,
64
                COUNT(*) AS Carreras_Ganadas
65
            FROM
                Resultados
66
            WHERE
67
68
                Posicion = 1
            GROUP BY
69
70
                id_Piloto
71
      ) victorias ON p.id_Piloto = victorias.id_Piloto
       ORDER BY
72
73
            CASE
74
                WHEN victorias.Carreras_Ganadas IS NULL THEN 0
75
                ELSE victorias.Carreras_Ganadas
76
            END DESC,
77
            p.Nombre;
0%
     $ 13:66
          Filter Rows: Q Search
                                               Export:
Result Grid
  Nombre_Piloto Carreras_Ganadas
  juan pérez
  ana martínez
  carlos rodríguez Ninguna
  luis sánchez
              Ninguna
  maría garcía
              Ninguna
```

Trigger:

Crea un **trigger** que se active cuando se inserte un nuevo resultado de una carrera, actualizando automáticamente el rendimiento del drone (velocidad promedio y tiempo total en carrera) en la tabla de drones.

```
ALTER TABLE DRONES
        ADD COLUMN velocidad_promedio DECIMAL(10,2) DEFAULT 0,
        ADD COLUMN tiempo_total_carrera TIME DEFAULT '00:00:00';
        DELIMITER //
85
 86
 87 •
        CREATE TRIGGER actualizar_rendimiento_dron
        AFTER INSERT ON Resultados
        FOR EACH ROW
        BEGIN
            DECLARE total_tiempo TIME;
            DECLARE total_carreras INT;
 93
            DECLARE velocidad_avg DECIMAL(10,2);
 94
 95
            -- Calcular el tiempo total en carreras para el dron
 96
            SELECT SEC_TO_TIME(SUM(TIME_TO_SEC(tiempo))) INTO total_tiempo
            FROM Resultados
 97
            WHERE id_Dron = NEW.id_Dron AND tiempo IS NOT NULL;
 98
 99
100
            -- Contar el número total de carreras para el dron
101
            SELECT COUNT(*) INTO total_carreras
102
            FROM Resultados
103
            WHERE id_Dron = NEW.id_Dron AND tiempo IS NOT NULL;
104
            -- Calcular la velocidad promedio
105
            -- Asumimos que todas las carreras tienen la misma distancia, por ejemplo, 1000 metros
106
            SET velocidad_avg = (1000 * total_carreras) / (TIME_TO_SEC(total_tiempo) / 3600);
107
108
            -- Actualizar la tabla DRONES
109
            UPDATE DRONES
110
            SET
111
                velocidad_promedio = velocidad_avg,
112
113
                tiempo_total_carrera = total_tiempo
            WHERE id_DRON = NEW.id_Dron;
114
        END //
115
116
117
        DELIMITER;
```

Procedimiento (Formulario):

Crea un **procedimiento almacenado** que permita registrar una **nueva carrera** y el **resultado de un drone** en esa carrera. El procedimiento debe:

Registrar la carrera: Incluir la fecha y la pista de la competencia.

Insertar el resultado del drone: Incluir el drone, piloto, posición final, tiempo de carrera, y cualquier problema técnico.

Verificar la existencia: Comprobar si el drone y el piloto están registrados en el sistema antes de insertar el resultado.

Actualizar estadísticas: Si el drone queda en primer lugar, actualizar el número de victorias del piloto y el rendimiento del drone.

*Se debe crear un campo en la tabla de piloto para almacenar el número de victorias.

```
ALTER TABLE Pilotos
  ADD COLUMN numero_victorias INT DEFAULT 0;
  ALTER TABLE Carreras
  MODIFY COLUMN Fecha DATETIME NOT NULL,
  MODIFY COLUMN Pista VARCHAR(100) NOT NULL;
  DELIMITER //
  CREATE PROCEDURE registrar_carrera_y_resultado(
      IN p_fecha DATETIME,
      IN p_pista VARCHAR(100),
      IN p_ciudad VARCHAR(100),
      IN p_id_dron INT,
      IN p_id_piloto INT,
      IN p_posicion INT,
      IN p_tiempo TIME,
      IN p_problema_tecnico VARCHAR(200)
  )

→ BEGIN

      DECLARE v_id_carrera INT;
      DECLARE v_drone_existe INT;
      DECLARE v_piloto_existe INT;
      -- Iniciar transacción
      START TRANSACTION;
      -- Verificar existencia del drone y piloto
      SELECT COUNT(*) INTO v_drone_existe FROM DRONES WHERE id_DRON = p_id_dron;
      SELECT COUNT(*) INTO v_piloto_existe FROM Pilotos WHERE id_Piloto = p_id_piloto;
      IF v_drone_existe = 0 OR v_piloto_existe = 0 THEN
          SIGNAL SQLSTATE '45000'
          SET MESSAGE_TEXT = 'El drone o el piloto no existen en el sistema.';
          ROLLBACK;
      ELSE
```

```
55
               -- Registrar la carrera
56
               INSERT INTO Carreras (Fecha, Pista, Ciudad)
57
               VALUES (p_fecha, p_pista, p_ciudad);
58
59
               SET v_id_carrera = LAST_INSERT_ID();
70
               -- Insertar el resultado
71
               INSERT INTO Resultados (id_Dron, id_Carrera, id_Piloto, Posicion, tiempo, Problema_tecnico)
72
73
               VALUES (p_id_dron, v_id_carrera, p_id_piloto, p_posicion, p_tiempo, p_problema_tecnico);
74
75
               -- Si el drone queda en primer lugar
76
               IF p_posicion = 1 THEN
77
                   -- Actualizar número de victorias del piloto
78
                   UPDATE Pilotos
79
                   SET numero_victorias = numero_victorias + 1
30
                   WHERE id_Piloto = p_id_piloto;
31
32
                   -- Actualizar rendimiento del drone
33
                   UPDATE DRONES
34
                   SET
35
                       velocidad_promedio = (
                            SELECT AVG(1000 / TIME_TO_SEC(tiempo) * 3600)
36
37
                            FROM Resultados
38
                            WHERE id_Dron = p_id_dron AND Posicion = 1
39
                        ),
90
                        tiempo_total_carrera = (
91
                            SELECT SEC_TO_TIME(SUM(TIME_TO_SEC(tiempo)))
32
                            FROM Resultados
93
                            WHERE id_Dron = p_id_dron
94
                   WHERE id_DRON = p_id_dron;
€
               END IF;
96
37
98
               COMMIT;
99
               SELECT 'Carrera y resultado registrados con éxito.' AS Mensaje;
10
16
           END IF;
)2
       END //
```