

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).

```

5 • Select p.Nombre, c.Pista, c.Ciudad, c.Fecha, r.Posicion, d.Marca, d.Modelo
6 FROM Resultados r
7 inner join Pilotos p ON r.id_Piloto = p.id_Piloto
8 inner join Carreras c ON r.id_Carrera = c.id_Carrera
9 inner Join DRONES d ON r.id_Dron = d.id_DRON
10 WHERE p.Nombre like 'Luis%';
11

```

100% 1:10

Result Grid Filter Rows: Search Export:

	Nombre	Pista	Ciudad	Fecha	Posicion	Marca	Modelo
	Luis Sánchez	Parque Natural	Barcelona	2023-07-01 16:30:00	2	PowerVision	PowerEgg X
	Luis Sánchez	Circuito Urbano	Madrid	2023-06-15 14:00:00	5	PowerVision	PowerEgg X
	Luis Sánchez	Estadio Olímpico	Sevilla	2023-07-20 15:00:00	5	DJI	Mavic Air 2
	Luis Sánchez	Playa	Valencia	2023-08-05 17:00:00	5	Autel	EVO II
	Luis Sánchez	Montaña	Bilbao	2023-08-25 13:30:00	5	Skydio	2

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).

```

14
15 • SELECT c.Fecha,c.Pista,c. Ciudad,d.Marca,d.Modelo, d.`Velocidad _Max`,r.Posicion
16 FROM Resultados r
17 Inner Join DRONES d ON d.id_DRON = r.id_Dron
18 inner Join Carreras c ON r.id_Carrera = c.id_Carrera
19 WHERE d.`Velocidad _Max` > 200;
20

```

0% 81:15

Result Grid Filter Rows: Search Export:

Fecha	Pista	Ciudad	Marca	Modelo	Velocidad _Max	Posicion
2023-06-15 14:00:00	Circuito Urbano	Madrid	DJI	Mavic Air 2	201	1
2023-06-15 14:00:00	Circuito Urbano	Madrid	Autel	EVO II	220	2
2023-07-01 16:30:00	Parque Natural	Barcelona	DJI	Mavic Air 2	201	3
2023-07-01 16:30:00	Parque Natural	Barcelona	DJI	Mavic Air 2	201	3
2023-07-01 16:30:00	Parque Natural	Barcelona	Autel	EVO II	220	4
2023-07-20 15:00:00	Estadio Olímpico	Sevilla	Autel	EVO II	220	1
2023-07-20 15:00:00	Estadio Olímpico	Sevilla	DJI	Mavic Air 2	201	5
2023-08-05 17:00:00	Playa	Valencia	DJI	Mavic Air 2	201	4
2023-08-05 17:00:00	Playa	Valencia	Autel	EVO II	220	5
2023-08-25 13:30:00	Montaña	Bilbao	DJI	Mavic Air 2	201	2
2023-08-25 13:30:00	Montaña	Bilbao	Autel	EVO II	220	3

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.

```

24 • SELECT p.Nombre, AVG(r.Tiempo) as Tiempo_promedio, count(DISTINCT r.id_Carrera) as Carreras_participadas
25 From Pilotos p
26 inner join (SELECT id_Piloto, id_Carrera, Tiempo
27             FROM Resultados
28             WHERE Tiempo IS NOT NULL) r ON p.id_Piloto = r.id_Piloto
29 GROUP BY p.Nombre
30 HAVING
31     COUNT(DISTINCT r.id_Carrera) >= 3
32 ORDER BY
33     Tiempo_Promedio ASC;

```

100% 12:38

Result Grid Filter Rows: Search Export:

Nombre	Tiempo_prome...	Carreras_participad...
Juan Pérez	209.5824000000	5
María García	212.6918000000	5
Carlos Rodríguez	216.9366000000	5
Ana Martínez	217.1624000000	5
Luis Sánchez	221.8066000000	5

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

```

5 • WITH RankedTimes AS (
6     SELECT d.id_DRON, d.Marca, d.Modelo, r.id_Carrera, r.tiempo,
7           ROW_NUMBER() OVER (PARTITION BY d.id_DRON ORDER BY r.tiempo ASC) AS tiempo_rank
8     FROM DRONES d
9     INNER JOIN Resultados r ON d.id_DRON = r.id_Dron
10    WHERE r.tiempo IS NOT NULL
11 )
12 SELECT
13     id_DRON,
14     Marca,
15     Modelo,
16     id_Carrera AS Carrera_Mejor_Tiempo,
17     tiempo AS Mejor_Tiempo
18 FROM
19     RankedTimes -- el with
20 WHERE
21     tiempo_rank = 1
22 ORDER BY
23     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
57     LOWER(p.Nombre) AS Nombre_Piloto,
58     IFNULL(victorias.Carreras_Ganadas, 'Ninguna') AS Carreras_Ganadas
59 FROM
60     Pilotos p
61 LEFT JOIN (
62     SELECT
63         id_Piloto,
64         COUNT(*) AS Carreras_Ganadas
65     FROM
66         Resultados
67     WHERE
68         Posicion = 1
69     GROUP BY
70         id_Piloto
71 ) victorias ON p.id_Piloto = victorias.id_Piloto
72 ORDER BY
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

Result Grid



Filter Rows:

Search

Export:



Nombre_Piloto	Carreras_Ganadas	
juan p��rez	4	
ana mart��nez	1	
carlos rodr��guez	Ninguna	
luis s��nchez	Ninguna	
mar��a garc��a	Ninguna	

Trigger:

Crea un **trigger** que se active cuando se inserte un nuevo resultado de una carrera, actualizando autom  ticamente 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';

85 DELIMITER //
86
87 • CREATE TRIGGER actualizar_rendimiento_dron
--
AFTER INSERT ON Resultados
FOR EACH ROW
--
BEGIN
    DECLARE total_tiempo TIME;
    DECLARE total_carreras INT;
    DECLARE velocidad_avg DECIMAL(10,2);

93
94
95 -- Calcular el tiempo total en carreras para el dron
96 SELECT SEC_TO_TIME(SUM(TIME_TO_SEC(tiempo))) INTO total_tiempo
97 FROM Resultados
98 WHERE id_Dron = NEW.id_Dron AND tiempo IS NOT NULL;
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
105 -- Calcular la velocidad promedio
106 -- Asumimos que todas las carreras tienen la misma distancia, por ejemplo, 1000 metros
107 SET velocidad_avg = (1000 * total_carreras) / (TIME_TO_SEC(total_tiempo) / 3600);
108
109 -- Actualizar la tabla DRONES
110 UPDATE DRONES
111 SET
112     velocidad_promedio = velocidad_avg,
113     tiempo_total_carrera = total_tiempo
114 WHERE id_DRON = NEW.id_Dron;
115 END //
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
```

```

35 -- Registrar la carrera
36 INSERT INTO Carreras (Fecha, Pista, Ciudad)
37 VALUES (p_fecha, p_pista, p_ciudad);
38
39 SET v_id_carrera = LAST_INSERT_ID();
40
41 -- Insertar el resultado
42 INSERT INTO Resultados (id_Dron, id_Carrera, id_Piloto, Posicion, tiempo, Problema_tecnico)
43 VALUES (p_id_dron, v_id_carrera, p_id_piloto, p_posicion, p_tiempo, p_problema_tecnico);
44
45 -- Si el drone queda en primer lugar
46 IF p_posicion = 1 THEN
47     -- Actualizar número de victorias del piloto
48     UPDATE Pilotos
49     SET numero_victorias = numero_victorias + 1
50     WHERE id_Piloto = p_id_piloto;
51
52     -- Actualizar rendimiento del drone
53     UPDATE DRONES
54     SET
55         velocidad_promedio = (
56             SELECT AVG(1000 / TIME_TO_SEC(tiempo) * 3600)
57             FROM Resultados
58             WHERE id_Dron = p_id_dron AND Posicion = 1
59         ),
60         tiempo_total_carrera = (
61             SELECT SEC_TO_TIME(SUM(TIME_TO_SEC(tiempo)))
62             FROM Resultados
63             WHERE id_Dron = p_id_dron
64         )
65     WHERE id_DRON = p_id_dron;
66 END IF;
67
68 COMMIT;
69
70 SELECT 'Carrera y resultado registrados con éxito.' AS Mensaje;
71 END IF;
72 END //
73

```