Evaluación: TERCERA EVALUACIÓN	Fecha:16-5-2024	Calificación	IES SEGUNDO DE CHOMÓN
Módulo/Materia: Bases de Datos	Curso:1º DAM		
Departamento: INFORMÁTICA			
Nombre y apellidos:			

INSTRUCCIONES DE LA PRUEBA:

- Puedes utilizar la hoja resumen durante el examen. NO se puede utilizar Internet
- Entrega: crea un documento pdf y adjunta capturas de pantalla en las que se vean las sentencias SQL completas y las tablas con los resultados de las consultas. Si la entrega no se realiza de la forma indicada no se corregirá.
- Utiliza la base de datos de "ciclismo" borra la que tengas para evitar fallos y vuelve a importar el archivo que os dejo en Moodle.

1. **Ganador_etapa_max_altura_media -** Crea una función que muestre cual es el ganador de la etapa (etapa, dorsal y nombre) que tiene la mayor altura media de sus puertos. **(1 punto).** Si hubiera varias etapas con la misma altura media, elige la que tenga más kilómetros **(0.25 puntos)**

   Haz que devuelva la siguiente frase:

   "El ciclista …… con dorsal…. ha ganado la etapa …, que tiene una altura media de … metros".

2. **Ganancias_por_equipo (2.25 puntos).**
   - Cada equipo se queda el 40% de las ganancias de sus ciclistas, sabiendo esto, crea una función a la que pasando por consola un dorsal de un ciclista nos devuelva cuánto dinero ha ganado su equipo (teniendo en cuenta a todos los miembros del equipo).
   - Si de un equipo no ha ganado dinero nadie, debe aparecer que la cantidad es 0 no puede aparecer null.
   - Prúebalo con los ciclistas 1,3,4 y 27.

```
mysql> drop function ganancias_equipo;
Query OK, 0 rows affected (0.00 sec)

mysql> DELIMITER //
mysql> CREATE FUNCTION ganancias_equipo(a_dorsal INT) RETURNS VARCHAR(255)
    -> DETERMINISTIC
    -> BEGIN
    ->
    -> SET @dorsal = (SELECT nomeq FROM ciclista WHERE dorsal = a_dorsal);
    ->
    ->
    -> set @consulta =  (select(SUM(premio)*0.4)as suma FROM llevar ll JOIN maillot m O
N m.codigo = ll.codigo JOIN ciclista c ON ll.dorsal = c.dorsal where nomeq = @dorsal GR
OUP BY nomeq);
    ->
    ->
    -> IF @consulta IS NULL THEN
    -> RETURN CONCAT("0");
    -> else
    -> RETURN CONCAT(@consulta);
    -> end if;
    ->
    ->
    -> END//
Query OK, 0 rows affected (0.00 sec)

mysql> DELIMITER ;
mysql>
mysql> SELECT ganancias_equipo(2);
+---------------------+
| ganancias_equipo(2) |
+---------------------+
| 1200000             |
+---------------------+
1 row in set (0.00 sec)

mysql>
```

3. **Rellena_dificultad() <u>(1 ptos)</u>**

   - Antes de seguir añade una columna a la tabla puerto llamada “dificultad”.

```
mysql> alter table puerto add column dificulad VARCHAR(255);
Query OK, 0 rows affected (0.01 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

   - Una vez hecho esto crea el procedimiento que rellenará la dificultad de los registros existentes en función de la pendiente si la pendiente es de menos de 20 dificultad=fácil, entre 20 y 30 dificultad=media y mayor de 30 dificultad=máxima.

```
mysql> call rellenar;
Query OK, 2 rows affected (0.00 sec)

mysql> select * from puerto;
+---------------+--------+-----------+-----------+--------+--------+-----------+
| nompuerto     | altura | categoria | pendiente | netapa | dorsal | dificulad |
+---------------+--------+-----------+-----------+--------+--------+-----------+
| p1            |   2489 | 1         |        34 |      2 | 3      | maxima    |
| p2            |   2789 | 1         |        44 |      4 | 3      | maxima    |
| Puerto F      |   2500 | E         |        17 |      4 | 2      | facil     |
| Puerto fff    |   2500 | E         |        17 |      4 | 2      | facil     |
| Puerto nuevo1 |   2500 | a         |        17 |      4 | 1      | facil     |
| Puerto otro   |   2500 | E         |        17 |      4 | 1      | facil     |
| Puerto1       |   2500 | E         |        23 |      1 | 2      | media     |
+---------------+--------+-----------+-----------+--------+--------+-----------+
7 rows in set (0.00 sec)
```

```sql
drop procedure rellenar;
DELIMITER //
CREATE PROCEDURE rellenar()
BEGIN




UPDATE puerto SET dificulad = "facil" WHERE pendiente <
20;

UPDATE puerto SET dificulad = "media" where pendiente
>=20 and pendiente <=30;

UPDATE puerto SET dificulad = "maxima" where pendiente
>30;


END//

DELIMITER ;

call rellenar;
```

4. **Alta_puerto() (1.5 puntos)**

Crea un procedimiento para dar de alta un puerto dados su nombre, altura, categoría y pendiente. El campo de la dificultad deberá calcularse dentro del procedimiento, es decir no se pasará por parámetro.

```
drop procedure alta_puerto;

DELIMITER //

CREATE PROCEDURE alta_puerto(
p_nompuerto VARCHAR(45),
p_altura FLOAT,
p_categoria VARCHAR(45),
p_pendiente FLOAT
)
BEGIN

insert into puerto (nompuerto, altura, categoria, pendiente) VALUES ("p_nompuerto", p_altura, "p_categoria", p_pendiente
);

call rellenar;

END//


delimiter ;

call alta_puerto("puertoexamen", 3000, "E", 40);
```

5. **Ganancias_ciclistas (2.5 ptos) (Utiliza cursores para resolver el *ejercicio. En caso contrario no se corregirá)***

Como hemos dicho antes cada equipo se queda el 40% de las ganancias de sus ciclistas, por tanto, el ciclista se quedará el 60%, sabiendo esto, crea un procedimiento que cree una tabla (la clave primaria debe ser el dorsal del ciclista y deberá tener foráneas) para guardar las ganancias de todos los ciclistas. Dicha tabla deberá tener las siguientes columnas: *dorsal, nombre_ciclista, equipo, gananciasciclista*.

```
mysql> select * from ganancias_ciclistas;
+--------+------------------+---------------+--------+
| dorsal | nombre           | nombre_equipo | premio |
+--------+------------------+---------------+--------+
|      1 | Miguel Indurain  | Banesto       | 400000 |
|      2 | Pedro Delgado    | Banesto       | 400000 |
|      3 | Alex Zulle       | Navigare      | 400000 |
|      4 | Alessio Di Basco | TVM           | 200000 |
+--------+------------------+---------------+--------+
4 rows in set (0.00 sec)
```

```
drop procedure ganancias;
DELIMITER //

CREATE PROCEDURE ganancias()
BEGIN
DECLARE p_dorsal INT;
DECLARE p_nombre_ciclista VARCHAR(25);
DECLARE p_nombre_equipo VARCHAR(45);
DECLARE p_premio FLOAT;
DECLARE var_final BOOLEAN DEFAULT 0;
DECLARE cursor1 CURSOR FOR  select c.dorsal,c.nombre, c.nomeq, (SUM(premio)*0.6)as suma FROM llevar ll JOIN maillot m ON
m.codigo = ll.codigo JOIN ciclista c ON ll.dorsal = c.dorsal GROUP BY dorsal;

 DECLARE CONTINUE HANDLER FOR NOT FOUND SET var_final =1;

 drop table Ganancias_ciclistas;
CREATE TABLE Ganancias_ciclistas(
dorsal INT AUTO_INCREMENT PRIMARY KEY,
nombre VARCHAR(25),
nombre_equipo VARCHAR(45),
premio FLOAT,
FOREIGN KEY (dorsal) REFERENCES ciclista (dorsal),
FOREIGN KEY (nombre_equipo) REFERENCES ciclista (nomeq)
);

OPEN cursor1;

bucle:LOOP

FETCH cursor1 INTO p_dorsal, p_nombre_ciclista, p_nombre_equipo, p_premio ;

if var_final = 1 THEN
    leave bucle;
```

```
 drop table Ganancias_ciclistas;
CREATE TABLE Ganancias_ciclistas(
dorsal INT AUTO_INCREMENT PRIMARY KEY,
nombre VARCHAR(25),
nombre_equipo VARCHAR(45),
premio FLOAT,
FOREIGN KEY (dorsal) REFERENCES ciclista (dorsal),
FOREIGN KEY (nombre_equipo) REFERENCES ciclista (nomeq)
);

OPEN cursor1;

bucle:LOOP

FETCH cursor1 INTO p_dorsal, p_nombre_ciclista, p_nombre_equipo, p_premio ;

if var_final = 1 THEN
    leave bucle;
end if;

INSERT INTO Ganancias_ciclistas VALUES (p_dorsal, p_nombre_ciclista, p_nombre_equipo, p_premio);


end loop bucle;
CLOSE cursor1;

END//

delimiter ;

call ganancias;
```

Evaluación: TERCERA EVALUACIÓN	Fecha:16-5-2024	Calificación	IES SEGUNDO DE CHOMÓN
Módulo/Materia: Bases de Datos	Curso:1º DAM		
Departamento: INFORMÁTICA			
Nombre y apellidos:			

6. Trigger (1.5 ptos)

- Crea una tabla llamada “mensajes” con una columna “id” int auto_increment y otra llamada “mensaje” de tipo varchar.
- Una vez hecho esto crea un trigger que al insertar un nuevo ciclista vaya rellenando la tabla mensajes según lo siguiente:

  - Si en el equipo hay 10 ciclistas rellene el mensaje que diga:
    “En el equipoX no se pueden inscribir más ciclistas”
  - Si hay menos de 10 ciclistas el mensaje dirá:
    “La inscripción sigue abierta en el equipoX”
  - Si hay más de 10 ciclistas el mensaje dirá:
    “Error, revisar las inscripciones del equipoX”.

- Una vez hecho el trigger para comprobar si funciona inserta 4 ciclistas al equipo ‘Amore Vita’ y muestra la tabla mensajes para ver cómo va cambiando.
- Una vez hecho el trigger para comprobar si funciona inserta 4 ciclistas al equipo ‘Amore Vita’ y muestra la tabla mensajes para ver cómo va cambiando.

```sql
SELECT ganancias_equipo(2);


CREATE TABLE mensajes(
id INT AUTO_INCREMENT PRIMARY KEY,
mensaje VARCHAR(255)
);


DELIMTITER //
CREATE TRIGGER rellenar
before INSERT on ciclista FOR EACH ROW
BEGIN
set @conteo = (select count(dorsal) FROM ciclista GROUP BY nomeq);

if @conteo = 10 THEN
INSERT INTO mensajes(mensaje) VALUES ("En el equipo", new.nomeq ,"no se pueden inscribir más");
END IF;
if @conteo <10 THEN
INSERT INTO mensajes (mensaje) VALUES ("En el equipo", new.nomeq ,"esta abierto para mas");
END iF;
if @conteo > 10 THEN
INSERT INTO mensajes (mensaje) VALUES ("En el equipo", new.nomeq ,"error revisar ");
END IF;



END //
DELIMITER ;
```

EJERCICIO 1	EJERCICIO 2	EJERCICIO 3	EJERCICIO 4	EJERCICIO 5	EJERCICIO 6	NOTA FINAL
1.25	2.25	1	1.5	2.5	1.5	