

RED DE LABORATORIOS

BASE DE DATOS

2021/2022



AUTORA: CLAUDIA YAZMIN LÓPEZ LAFITA

INDICE

ENUNCIADO DEL SUPUESTO.	2
MODELO ENTIDAD-RELACIÓN	3
MODELO RELACIONAL	4
CONSULTAS	5
VISTAS	7
PROGRAMAS	8
FUNCIONES	9
PROCEDIMIENTOS	11
TRIGGER	15
CONCLUSIONES	19

1. ENUNCIADO DEL SUPUESTO.

En España existe una red de laboratorios analíticos que se encarga de analizar muestras para realizar un informe nacional de biodiversidad y caracterización de la misma con los factores ambientales. De los laboratorios queremos conocer su código, su CIF, el nombre, la dirección así como su localidad, provincia y número de teléfono.

Estas muestras se recolectan en un área; en un área se puede recolectar una o varias muestras, mientras que las muestras pueden localizarse en varias áreas simultáneamente. Se quiere saber la cantidad de muestras recolectadas en cada área y la fecha cuando se hizo. De las muestras queremos conocer el código numérico de cada muestra, la familia, género y especie a que pertenece, así como el tipo (flora/fauna) y el peso. De las áreas queremos conocer su código, el tipo de muestreo (cuadrante, transecto) que se ha realizado y las coordenadas geográficas (longitud y latitud).

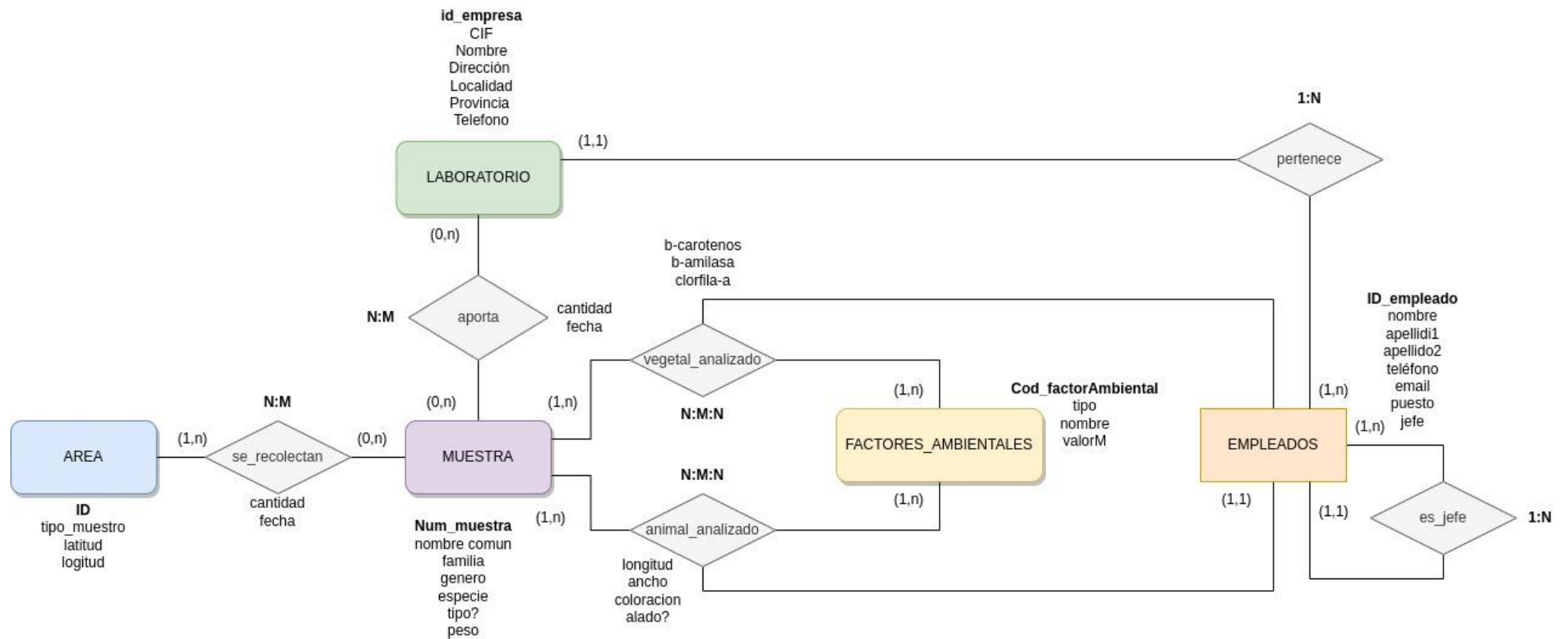
Al tratarse de una red de laboratorios estos aportan muestras y por lo tanto queremos almacenar la fecha y cantidad de muestras que aporta cada laboratorio. En los laboratorios trabajan empleados que se encargan de analizar las muestras; estos empleados pertenecen a una y únicamente una empresa, pero en una empresa pueden trabajar más de un empleado. De los empleados se quiere guardar el identificador del empleado, el nombre, los apellidos, el teléfono, el correo electrónico y el puesto que ocupa (practicante, becario, titular), además queremos conocer quién es jefe de quién.

Los análisis de una muestra son realizados por un único empleado, pero este puede analizar más de una muestra. En los análisis se quiere conocer qué factores ambientales se han interactuado con las muestras; además en los análisis de muestras tipo flora se quiere conocer si la cantidad de betacarotenos, clorofila-a son bajas, medias o altas y si hay o no presencia de amilasa. De la misma manera de los análisis de las muestras tipo fauna queremos guardar la longitud, el ancho del espécimen, el color y si son alados o no.

De los factores ambientales queremos conocer el tipo (químico, físico), nombre (subtipo), el valor medio y su código numérico.

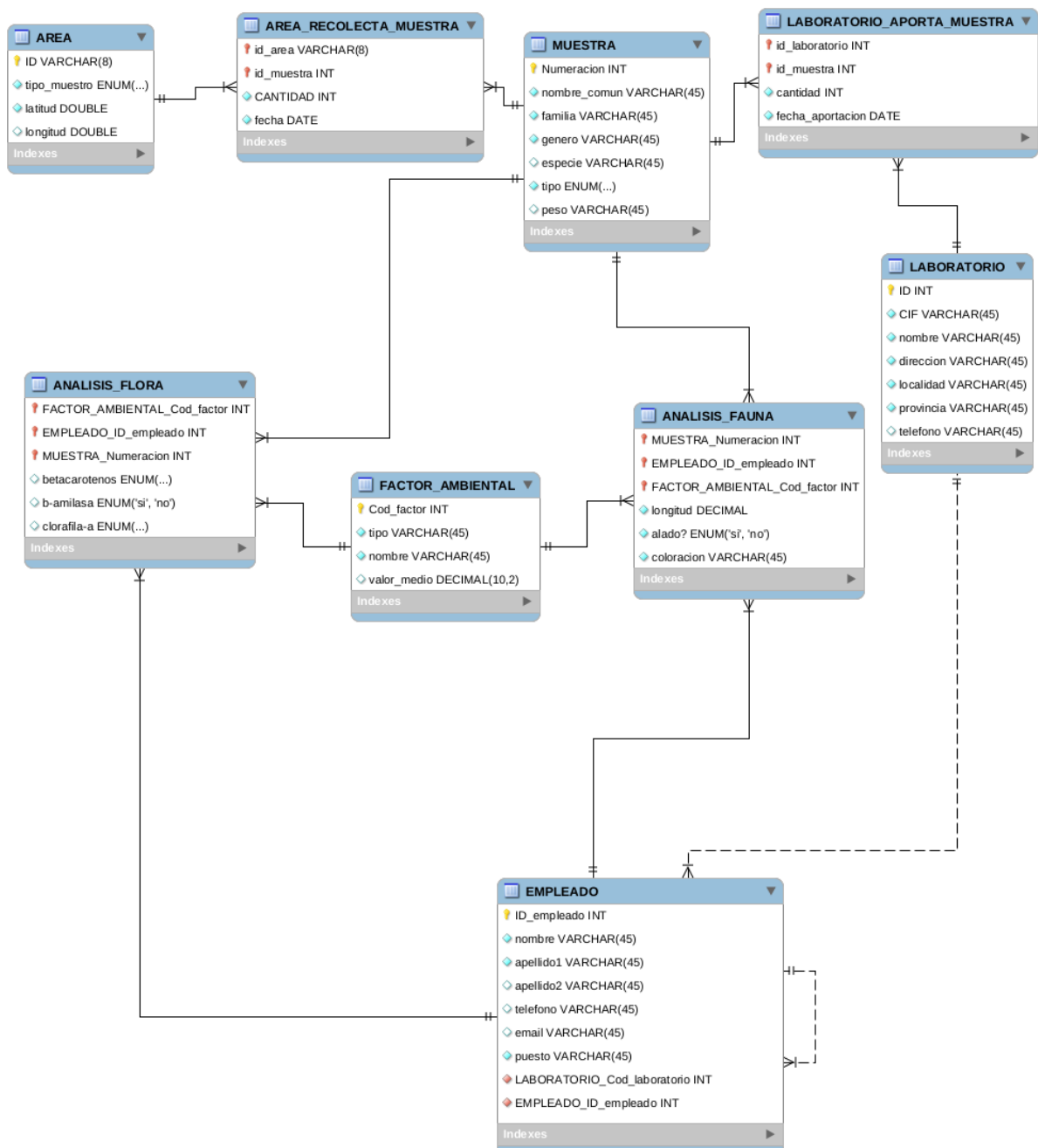
2. MODELO ENTIDAD-RELACIÓN

Al trasladar el enunciado al modelo racional hemos obtenido un total de cinco entidades. Tendremos un total de 4 relaciones N:M, dos de las cuales son debido a una relación a tres, que darán lugar a 4 tablas una vez lo pasemos al modelo racional; así como dos relaciones 1:N, obteniendo un total de 6 relaciones.



3. MODELO RELACIONAL

Al trasladar el modelo entidad-relación al modelo relacional, podemos observar que se obtiene un total de 9 tablas 4 de las cuales son el resultado de relaciones N:M entre las entidades y las cuales disponen de atributos propios.

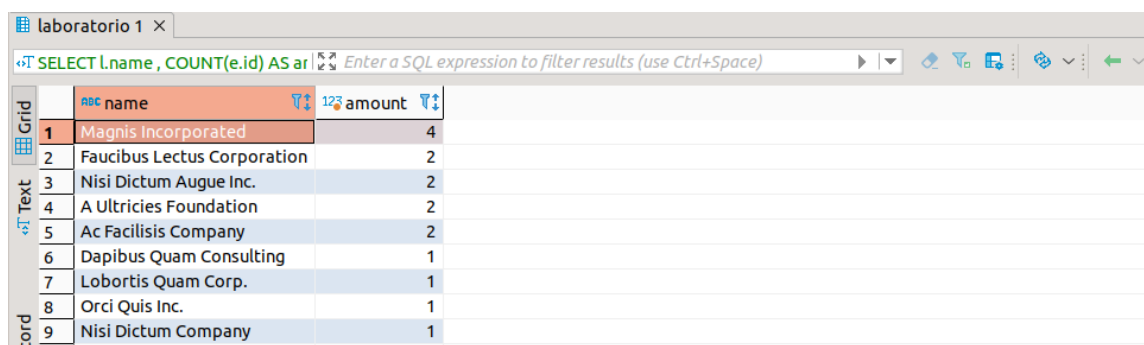


4. CONSULTAS

En este apartado se anunciará una serie de consultas para nuestro proyecto, además se incluirá el código para obtener los datos esperados así como una captura de pantalla del resultado. Las consultas serán multitablas y con subconsulta.

- Se quiere conocer la cantidad de empleados de cada laboratorio y ordenarlos de mayor a menor.

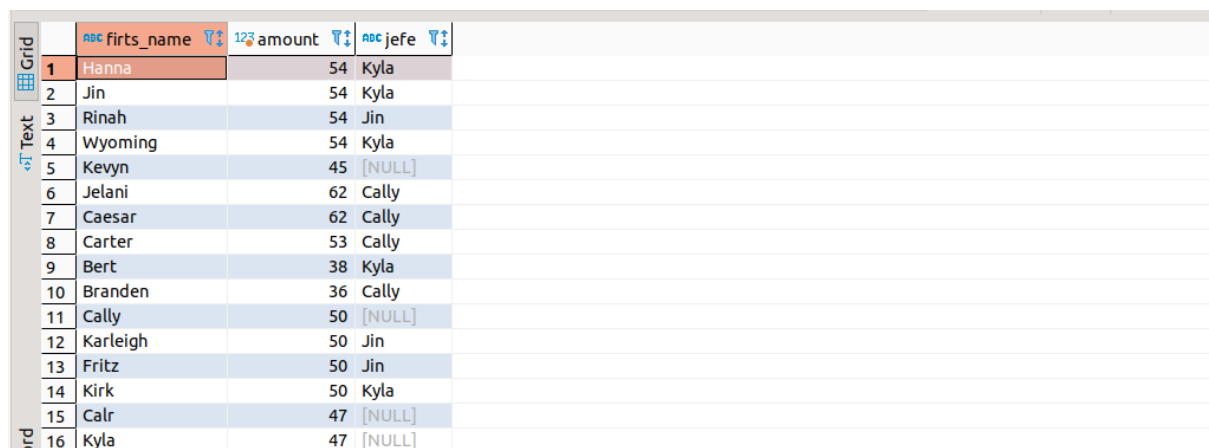
```
SELECT l.name , COUNT(e.id) AS amount FROM laboratorio l
INNER JOIN empleado e ON l.IdLab = e.Id_laboratorio
GROUP BY l.IdLab
ORDER BY COUNT(e.id) DESC;
```



Grid	name	amount
1	Magnis Incorporated	4
2	Faucibus Lectus Corporation	2
3	Nisi Dictum Augue Inc.	2
4	A Ultricies Foundation	2
5	Ac Facilisis Company	2
6	Dapibus Quam Consulting	1
7	Lobortis Quam Corp.	1
8	Orci Quis Inc.	1
9	Nisi Dictum Company	1

- Cuántas muestras ha analizado, de tipo vegetal, cada empleado (independientemente del laboratorio donde trabaja) y quien el nombre de su jefe.

```
SELECT e.firts_name, COUNT(m.id) AS amount , e2.firts_name AS jefe FROM muestra m
INNER JOIN lab_aporta_muestra lam ON m.id = lam.id_muestra
INNER JOIN laboratorio l ON lam.id_laboratorio = l.IdLab
INNER JOIN empleado e ON l.IdLab = e.Id_laboratorio
LEFT JOIN empleado e2 ON e2.id = e.Id_jefe
WHERE m.classification = 'vegetable'
GROUP BY e.id, e2.id;
```



Grid	firts_name	amount	jefe
1	Hanna	54	Kyla
2	Jin	54	Kyla
3	Rinah	54	Jin
4	Wyoming	54	Kyla
5	Kevyn	45	[NULL]
6	Jelani	62	Cally
7	Caesar	62	Cally
8	Carter	53	Cally
9	Bert	38	Kyla
10	Branden	36	Cally
11	Cally	50	[NULL]
12	Karleigh	50	Jin
13	Fritz	50	Jin
14	Kirk	50	Kyla
15	Calr	47	[NULL]
16	Kyla	47	[NULL]

- Que área es la que ha recolectado más muestras de tipo animal.

```
SELECT a.ID, COUNT(m.id) AS amount FROM area a
INNER JOIN area_recolecta_muestra arm ON a.ID = arm.id_area
INNER JOIN muestra m ON arm.id_muestra = m.id
WHERE m.classification = 'animal'
GROUP BY a.ID
ORDER BY COUNT(m.id) DESC LIMIT 1;
```

	ABC ID	123 amount
1	S63434S	3

- La cantidad de muestras que aporta cada laboratorio de cada tipo. Queremos que se vea el nombre del laboratorio y la cantidad de cada tipo especificado. Se ordenará alfabéticamente por el nombre del laboratorio.

```
SELECT l.name, m.classification, COUNT(m.classification) AS Amount FROM muestra m
INNER JOIN lab_aporta_muestra lam
ON m.id = lam.id_muestra
INNER JOIN laboratorio l ON lam.id_laboratorio = l.IdLab
GROUP BY l.ID, m.classification
ORDER BY l.name ASC;
```

	ABC name	ABC classification	123 Amount
1	A Ultricies Foundation	animal	14
2	A Ultricies Foundation	vegetable	50
3	Ac Facilis Company	animal	10
4	Ac Facilis Company	vegetable	47
5	Ac Mattis Foundation	vegetable	55
6	Ac Mattis Foundation	animal	12
7	Dapibus Quam Consulting	vegetable	53
8	Dapibus Quam Consulting	animal	12
9	Faucibus Lectus Corporation	animal	5

- Cuales son las especies con una longitud mayor a la media en el tipo animal. Queremos conocer su nombre común (name) y su nombre científico (name_scientific) y la longitud.

```
SELECT m.name, m.name_scientific, af.legth FROM analisis_fauna af
INNER JOIN muestra m ON af.id_muestra = m.id
WHERE af.legth = (
SELECT MAX(af2.legth) FROM analisis_fauna af2);
```

muestra(+) 1 x			
« SELECT m.name, m.name_scientific Enter a SQL expression to filter results (use Ctrl+Space)			
Grid	ABC name	ABC name_scientific	123 legth
1	Wallaroo	Macropus robustus	197.06

5. VISTAS

En este apartado se realizará vistas (view) de las consultas anteriormente realizadas, nos centraremos simplemente en dos consultas. Para cada una de las vistas se mostrará el código sql para la formación de las vistas

- Cuántas muestras ha analizado, de tipo vegetal, cada empleado (independientemente del laboratorio donde trabaja) y quien el nombre de su jefe.

```
CREATE VIEW animal AS (SELECT m.name, m.name_scientific, af.legth
FROM analisis_fauna af
INNER JOIN muestra m ON af.id_muestra = m.id
WHERE af.legth = (
SELECT MAX(af2.legth) FROM analisis_fauna af2));
```

Grid	name	name_scientific	legth
1	Wallaroo	Macropus robustus	197.06

- Cuales son las especies con una longitud mayor a la media en el tipo animal. Queremos conocer su nombre común (name) y su nombre científico (name_scientific) y la longitud.

```
CREATE VIEW control_samples_employee AS (
SELECT e.firts_name, COUNT(m.id), e2.firts_name AS jefe FROM muestra m
INNER JOIN lab_aporta_muestra lam ON m.id = lam.id_muestra
INNER JOIN laboratorio l ON lam.id_laboratorio = l.IdLab
INNER JOIN empleado e ON l.IdLab = e.Id_laboratorio
LEFT JOIN empleado e2 ON e2.id = e.Id_jefe
WHERE m.classification = 'vegetable'
GROUP BY e.id, e2.id);
```

Grid	firts_name	amount	jefe
1	Hanna	54	Kyla
2	Jin	54	Kyla
3	Rinah	54	Jin
4	Wyoming	54	Kyla
5	Kevyn	45	[NULL]
6	Jelani	62	Cally
7	Caesar	62	Cally
8	Carter	53	Cally
9	Bert	38	Kyla
10	Branden	36	Cally
11	Cally	50	[NULL]
12	Karleigh	50	Jin
13	Fritz	50	Jin
14	Kirk	50	Kyla
15	Calr	47	[NULL]
16	Kyla	47	[NULL]

6. PROGRAMAS

El modelo entidad-relación se ha desarrollado con el programa [Diagrams](#), es una aplicación web gratuita y de código abierto que te permite crear una gran variedad de diagramas desde cualquier navegador web. Con esta herramienta puedes crear y editar una gran variedad de diagramas como: diagramas de flujo, diagramas entidad-relación, diagramas UML, organigramas, diagramas de procesos, mapas mentales, modelos de procesos de negocios, entre otros.

Para el modelo relacional se ha usado el programa [MySQL Workbench](#), el cual es una herramienta visual de diseño de bases de datos que integra desarrollo de software, administración de bases de datos, diseño de bases de datos, gestión y mantenimiento para el sistema de base de datos MySQL.

Para la creación de las tablas junto con su referencias así como sus consultas y la creación de las de las vistas se ha usado el programa [DBeaver](#) 21.3.3. es una aplicación de software cliente de SQL y una herramienta de administración de bases de datos. Para las bases de datos relacionales, utiliza la interfaz de programación de aplicaciones JDBC para interactuar con las bases de datos a través de un controlador JDBC. En nuestro caso el lenguaje usado es mysql, el cual hemos instalado previamente mediante comandos.

Para la obtención de los datos se han utilizado dos páginas web. La primera [Mockaroo](#), se trata de una herramienta web desarrollada por Mark Brocato, desde la que vamos a poder generar hasta 100.000 líneas de datos que podremos exportar en formato CSV, Tab-Delimited, JSON, SQL, Excel y DBUnit XML. La importancia de la herramienta radica en la posibilidad de probar con miles de datos que simulan ser reales, ahorrando tiempo y cobertura sobre los múltiples casos de pruebas que se nos puedan ocurrir.

La segunda página [generatedata](#), el sitio web proporciona una funcionalidad adicional sobre el script gratuito para que las empresas administren sus propias cuentas de usuarios y les permita registrar y administrar fácilmente sus propios conjuntos de datos. Una vez indicados los datos que desea que tenga la tabla puede obtener las tablas en formato CSV o los comandos SQL para la DDL.

La incorporación de los datos a las tablas se realizó por carga masiva. Esta se realizó siguiendo los pasos indicados en <https://dbeaver.com/docs/wiki/Data-transfer/>.

7. FUNCIONES

Una función en MySQL es una rutina creada para tomar unos parámetros, procesarlos y retornar en una salida.

Se diferencian de los procedimientos en las siguientes características:

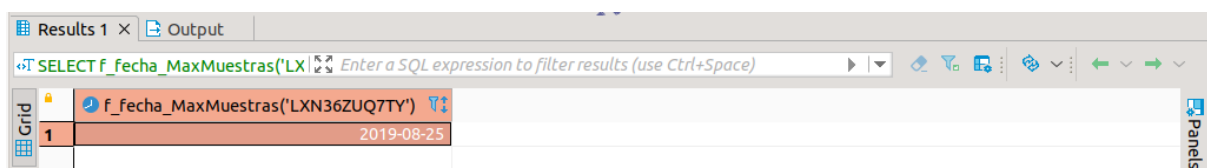
- Solamente pueden tener parámetros de entrada IN y no parámetros de salida OUT o INOUT
 - Deben retornar en un valor con algún tipo de dato definido
 - Pueden usarse en el contexto de una sentencia SQL
 - Solo retornan un valor individual, no un conjunto de registros.
- **Crear una función "f_num_empleado_laboratorio" que reciba como parámetro el CIF de un laboratorio y devuelva la fecha donde más ejemplares de una muestra ha aportado. Se debe comprobar si el laboratorio pertenece o no a la red de laboratorio.**

```
DELIMITER $$
CREATE FUNCTION f_fecha_MaxEjemplares (cif VARCHAR(20))
RETURNS DATE DETERMINISTIC
COMMENT 'Devuelve la fecha en la que un laboratorio ha aportado más
muestras'
BEGIN
    DECLARE v_estaLab INT DEFAULT 0;
    DECLARE v_salida DATE ;

    SELECT COUNT(*) INTO v_estaLab
    FROM laboratorio l
    WHERE l.CIF = cif;

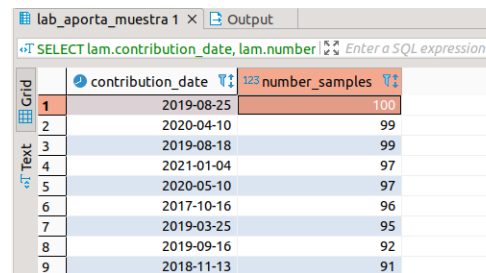
    IF v_estaLab > 0 THEN
        SELECT lam.contribution_date
        INTO v_salida
        FROM laboratorio l INNER JOIN lab_aporta_muestra lam
        ON l.IdLab = lam.id_laboratorio
        WHERE l.CIF = 'LXN36ZUQ7TY' AND lam.number_samples = (
            SELECT MAX(lam2.number_samples)
            FROM lab_aporta_muestra lam2);
    END IF;
    RETURN v_salida;
END$$
DELIMITER ;

SELECT f_fecha_MaxMuestras('LXN36ZUQ7TY');
```



Para comprobar que se incluye una query con el listado de las fechas del laboratorio pasado por parámetro junto con la cantidad de ejemplares de una muestras aportadas ordenadas de mayor a menor.

```
SELECT lam.contribution_date,
lam.number_samples
FROM lab_aporta_muestra lam INNER JOIN
laboratorio l
ON lam.id_laboratorio = l.IdLab
WHERE l.CIF = 'LXN36ZUQ7TY'
ORDER BY lam.number_samples DESC;
```



	contribution_date	number_samples
1	2019-08-25	100
2	2020-04-10	99
3	2019-08-18	99
4	2021-01-04	97
5	2020-05-10	97
6	2017-10-16	96
7	2019-03-25	95
8	2019-09-16	92
9	2018-11-13	91

- Crear una función “f_informacionArea” que pasándole como parámetro la latitud y longitud, así como el tipo de muestras (animal, vegetal) nos saque por consola la cantidad de muestras que se han recolectado con el siguiente formato:

"En la área xxxx se han recolectado yyy muestras del tipo zzzz"

xxxx --> ID del área

yyyy --> cantidad de muestras

zzzz --> classification de la muestras

DELIMITER \$\$

```
CREATE FUNCTION f_informacionArea (latitud DOUBLE, longitud DOUBLE,
tipo VARCHAR(9))
```

```
RETURNS VARCHAR (300) DETERMINISTIC
```

```
COMMENT 'la cantidad de muestras de un tipo determinado en dos
coordenadas dadas'
```

```
BEGIN
```

```
DECLARE v_salida VARCHAR (300) DEFAULT '';
```

```
DECLARE v_area VARCHAR (7) DEFAULT '';
```

```
DECLARE v_cantidad INT DEFAULT 0;
```

```
SELECT COUNT(m.id) INTO v_cantidad
```

```
FROM area a INNER JOIN area_recolecta_muestra arm
```

```
ON a.ID = arm.id_area
```

```
INNER JOIN muestra m
```

```
ON arm.id_muestra = m.id
```

```
WHERE a.latitud = latitud AND a.longitude = longitud
```

```
AND m.classification = tipo;
```

```
SELECT a.ID INTO v_area
```

```
FROM area a
```

```
WHERE a.latitud = latitud AND a.longitude = longitud;
```

```
SET v_salida = CONCAT('En el área ', v_area, ' se han
recolectado ', v_cantidad, ' muestras de tipo ', tipo);
```

```
RETURN v_salida;
```

```
END$$
```

```
DELIMITER ;
```

```
SELECT f_informacionArea(55.7597032, 37.6108925, 'animal')
```

Grid	Results 1
1	En el área H9552 se han recolectado 3 muestras de tipo animal

8. PROCEDIMIENTOS

- Crea un procedimiento "p_infoMuestraEmpleado" que se le pase como parámetro el identificador de un empleado y nos saque por consola los datos y los análisis de las muestras que ha analizado..

En este enunciado, se obtendrá dos ventanas que corresponden a dos tablas, una para cada tipo de muestras (animal, vegetal). Se podría hacer un UNION dentro del procedimiento, pero los análisis de las muestras varía con cada tipo no se obtendría un resultado de correcta visualización.

```
DELIMITER $$
```

```
CREATE PROCEDURE p_infoMuestraEmpleado (idEmpleado INT)
```

```
COMMENT 'Devuelve los datos de todas las especies y análisis de las  
muestras que ha analizado'
```

```
BEGIN
```

```
SELECT m.*, af.id_factoAmbiental, af.legth, af.wins, af.color  
FROM empleado e INNER JOIN analisis_fauna af  
ON e.id = af.id_empleado  
INNER JOIN muestra m  
ON af.id_muestra = m.id  
WHERE e.id = idEmpleado;
```

```
SELECT m.*, afl.id_factoAmbiental, afl.carotenos, afl.amilasa,  
afl.clorofila  
FROM empleado e INNER JOIN analisis_flora afl  
ON e.id = afl.id_empleado  
INNER JOIN muestra m  
ON afl.id_muestra = m.id  
WHERE e.id = idEmpleado;
```

```
END$$
```

```
DELIMITER ;
```

```
CALL p_infoMuestraEmpleado(5);
```

En la primera ventana nos sale por consola las muestras de tipo animal que ha analizado el empleado con id = 5.

CALL p_infoMuestraEmpleado(5) 2 Data filter is not supported											
Grid		id	name	family	genus	name_scientific	classification	weight	id_factoAmbiental	leg	
1	518	Western pygmy possum	Araucariaceae	Cercatetus	Cercatetus concinnus	animal		21	1	16	
2	541	Red hartebeest	Verrucariaceae	Alcelaphus	Alcelaphus buselaphus caama	animal		65.62	3	12	
3	570	Pale-throated three-toed sloth	Malvaceae	Bradypus	Bradypus tridactylus	animal		63.13	3	8	
4	578	Gemsbok	Cactaceae	Oryx	Oryx gazella	animal		68.59	1	4	
5	582	Deer	Verbenaceae	Odocoilenus	Odocoilenus virginianus	animal		47.99	1	4	
6	621	Hoffman's sloth	Viscaceae	Choloepus	Choloepus hoffmani	animal		93.59	2	4	
7	1,001	prueba	prueba	prueba	prueba	animal		0	2	4	

En la primera ventana nos sale por consola las muestras de tipo vegetal que ha analizado el empleado con id = 5.

muestra(+) 1		muestra(+) 1 (2) x									
CALL p_informMuestraEmpleado(5) Data filter is not supported											
Grid		id	asc name	asc family	asc genus	asc name_scientific	asc classification	123 weight	id_factoAmbiental	asc carotenos	asc amilasa
Text	1	22	Grouse	Celastraceae	Centrocerus	Centrocerus urophasianus	vegetable	13.63	1	media	no
	2	45	Turkey	Melastomataceae	Alectura	Alectura lathamii	vegetable	30.69	3	media	no
	3	74	Starling	Ranunculaceae	Lamprotornis	Lamprotornis chalybaeus	vegetable	64.91	3	alta	si
	4	82	Sheep	Poaceae	Ovis	Ovis dalli stonei	vegetable	15.33	1	alta	no
	5	86	Lizard	Fabaceae	Varanus	Varanus sp.	vegetable	59.53	1	media	no
	6	125	Otter	Polemoniaceae	Aonyx	Aonyx capensis	vegetable	65.37	2	alta	no
	7	204	Miner's cat	Lamiaceae	Bassariscus	Bassariscus astutus	vegetable	40.23	2	baja	no
	8	227	Baboon	Pteridaceae	Theropithecus	Theropithecus gelada	vegetable	63.36	3	alta	no
	9	256	White rhinoc	Candelariaceae	Ceratotherium	Ceratotherium simum	vegetable	70.95	2	media	no
	10	264	Bobcat	Scrophulariaceae	Felis	Felis rufus	vegetable	71.91	3	media	no
	11	268	Turkey vultur	Fabaceae	Cathartes	Cathartes aura	vegetable	69.99	2	media	si
	12	307	Blue-tongued	Sphagnaceae	Tiliqua	Tiliqua scincoides	vegetable	93.2	3	media	si
	13	313	Nelson grouse	Malvaceae	Ammospermophilus	Ammospermophilus nelsoni	vegetable	10.52	2	baja	si
	14	336	Southern whi	Gesneriaceae	Eurocephalus	Eurocephalus anguitimens	vegetable	41.29	3	alta	si

- Crea un procedimiento “p_informe_muestras_recolectadas” que al pasarle por parámetro dos fechas nos dé como resultado la cantidad de muestras de cada tipo se han recolectado por estaciones de cada año.

En este procedimientos al querer que nos cumpla una serie de select diferentes será necesario la declaración de cursores. Única y exclusivamente los elementos que cumplan con dichas condiciones podrán ser alterados. Con los cursores podremos trabajar con cada uno de los elementos (filas) de nuestra consulta sin tener que obtener nuevos conjuntos.

Las consultas en las que se basarán nuestros cursores son las siguientes:

- En la primera obtendremos la cantidad de ejemplares (amount) que se han recolectados por años.

```
SELECT YEAR(arm.date), SUM(arm.amount)
FROM area_recolecta_muestra arm
GROUP BY YEAR(arm.date);
```

- En la segunda queremos saber la cantidad de ejemplares (amount) cada trimestre de cada año. Por lo tanto, cada año tendremos 4 filas en representación de los trimestres.

```
SELECT YEAR(arm.date), QUARTER(arm.date), SUM(arm.amount)
FROM area_recolecta_muestra arm
GROUP BY YEAR(arm.date), QUARTER(arm.date)
ORDER BY YEAR(arm.date);
```

Una vez determinados nuestros selects procedemos al desarrollo de nuestro procedimiento con las declaraciones de los cursores.

```
DELIMITER $$
CREATE PROCEDURE p_informeRecoleccionEjemplares()
BEGIN
    DECLARE v_result VARCHAR(3000) DEFAULT '===== INFORME RECOLECCION
    DE EJEMPLARES =====\n-----\n\nEjemplares Totales: ';
    DECLARE v_total INT DEFAULT 0;
    DECLARE done BOOL DEFAULT FALSE;
    DECLARE v_ano INT DEFAULT 2000;
    DECLARE v_cuarto INT DEFAULT 1;
    DECLARE n integer DEFAULT 1;
```

```

-- declaramos el cursor de años
    DECLARE c1 CURSOR FOR
        SELECT YEAR(arm.date), SUM(arm.amount)
        FROM area_recolecta_muestra arm
        GROUP BY YEAR(arm.date);
-- declaramos cursor para los cuartos por años
    DECLARE c2 CURSOR FOR
        SELECT YEAR(arm.date), QUARTER(arm.date), SUM(arm.amount)
        FROM area_recolecta_muestra arm
        GROUP BY YEAR(arm.date), QUARTER(arm.date)
        ORDER BY YEAR(arm.date);
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

    SELECT SUM(arm.amount) INTO v_total
    FROM area_recolecta_muestra arm;

    SET v_result = CONCAT(v_result,v_total,'\n','\n-----
    Ejemplares por años-----\n','\n');
-- abrimos el cursor c1
    OPEN c1;
    WHILE (NOT done) do
        FETCH c1 INTO v_ano, v_total;
        IF (NOT done) THEN
            SET v_result = CONCAT(v_result,'En ',v_ano,':
            ',v_total,' ejemplares\n');
        END IF;
    END WHILE;
    CLOSE c1;
    SET v_result = CONCAT(v_result,'\n ----- Ejemplares por
    cuartos de cada año -----\n','\n');
-- abrimos el cursor c2
    OPEN c2;
    SET done = FALSE;
    WHILE (NOT done) do
        FETCH c2 INTO v_ano,v_cuarto,v_total;
        IF (NOT done) THEN
            SET v_result = CONCAT(v_result,n,'. En ',v_ano,'
            en el Q',v_cuarto,': ',v_total,' ejemplares\n');
            SET n=n+1;
        END IF;
    END WHILE;
    CLOSE c2;
    SET n=1;
    SELECT v_result;
END$$
DELIMITER ;
CALL p_informeRecoleccionEjemplares();

```

Para comprobar que el procedimiento ha salido correctamente copiamos el resultado de la llamada y lo pegamos en un Editor de Texto obteniendo lo siguiente.

```

===== INFORME RECOLECCION
DE EJEMPLARES =====
-----

Ejemplares Totales:43899

-----
Ejemplares por años-----

En 2018:10236 ejemplares
En 2020:9281 ejemplares
En 2021:8650 ejemplares
En 2019:9900 ejemplares
En 2017:5832 ejemplares

----- Ejemplares por cuartos de cada año -----

1. En 2017en el Q2: 601 ejemplares
2. En 2017en el Q3: 2109 ejemplares
3. En 2017en el Q4: 3122 ejemplares
4. En 2018en el Q1: 2498 ejemplares
5. En 2018en el Q2: 2386 ejemplares
6. En 2018en el Q3: 2600 ejemplares
7. En 2018en el Q4: 2752 ejemplares
8. En 2019en el Q1: 2572 ejemplares
9. En 2019en el Q2: 2408 ejemplares
10. En 2019en el Q3: 2280 ejemplares
11. En 2019en el Q4: 2640 ejemplares
12. En 2020en el Q1: 2440 ejemplares
13. En 2020en el Q2: 2599 ejemplares
14. En 2020en el Q3: 2311 ejemplares
15. En 2020en el Q4: 1931 ejemplares
16. En 2021en el Q1: 1798 ejemplares
17. En 2021en el Q2: 2586 ejemplares
18. En 2021en el Q3: 1914 ejemplares
19. En 2021en el Q4: 2352 ejemplares

```

- Crea un procedimiento “p_areaRecoleccion_Laboratorio” que se le pase por parámetro el CIF de un laboratorio y nos devuelva la información del área donde se han recolectado los ejemplares cuya fecha de aportación es aquella obtenida por la función “f_fecha_MaxEjemplares”. Recordemos que esta función nos devuelve la fecha en la que un laboratorio ha aportado más ejemplares de muestras.

```

DELIMITER $$
CREATE PROCEDURE p_areaRecoleccion_Laboratorio (cif VARCHAR(20))
COMMENT 'Devuelve la información de la muestra y área cuya fecha de
aportación es aquella que se obtiene en la función f_fecha_MaxEjemplares'
BEGIN
    DECLARE v_fecha DATE;
    DECLARE v_idMuestras INT DEFAULT 0;

    SELECT f_fecha_MaxEjemplares(cif) INTO v_fecha;

    SELECT lam.id_muestra INTO v_idMuestras
    FROM lab_aporta_muestra lam INNER JOIN laboratorio l
    ON lam.id_laboratorio = l.ID
    WHERE lam.contribution_date = v_fecha;

    SELECT a.*, m.*
    FROM area a INNER JOIN area_recolecta_muestra arm
    ON a.ID = arm.id_area
    INNER JOIN muestra m ON m.id = arm.id_muestra
    WHERE arm.id_muestra =v_idMuestras;
END$$
DELIMITER ;

```

```
CALL p_areaRecoleccion_Laboratorio('LXN36ZUQ7TY');
```

Grid	asc ID	asc sampling	latitud	longitude	id	asc name	asc family	asc genus	asc name_scientific	asc classification	weight
1	8653	cuadrante	38.942	117.059	546	Kangaroo	Cyperaceae	Macropus	Macropus agilis	animal	93.99
2	1788	cuadrante	26.338	115.363	546	Kangaroo	Cyperaceae	Macropus	Macropus agilis	animal	93.99

9. TRIGGER

En el siguiente apartado definiremos y solucionaremos dos trigger que se ejecutarán, al insertar, actualizar o eliminar elementos en alguna de las nuevas tablas.

- Crea un trigger que cuando se añade un análisis de tipo flora, este le asignado al empleado que menor análisis lleve realizados. En caso de que el empleado sea el de mayor análisis realizado este debe de mandar error indicando la cantidad de análisis hechos.

En este apartado deberemos de localizar primero de todo al empleado que menor análisis realizado y el factor ambiental cuya participación ha tenido menor ocurrencia en los análisis, estos datos le hemos obtenido con las siguientes consultas

```
SELECT af.id_empleado , COUNT(af.id_muestra)
FROM analisis_fauna af
GROUP BY af.id_empleado
ORDER BY COUNT(af.id_muestra) ASC
LIMIT 1;
```

Grid	id_empleado	COUNT(af.id_muestra)
1	5	7

```
SELECT af.id_factoAmbiental, COUNT(af.id_muestra)
FROM analisis_fauna af
GROUP BY af.id_factoAmbiental
ORDER BY COUNT(af.id_muestra) ASC
LIMIT 1;
```

Grid	id_factoAmbiental	COUNT(af.id_muestra)
1	2	60

Teniendo en localizados estos valores procedemos a la creación y comprobación del trigger

```
DELIMITER $$
CREATE TRIGGER t_analisisEmpleado
AFTER INSERT ON muestra FOR EACH ROW
BEGIN
    DECLARE v_idEmpleadoMenor INT DEFAULT 0;
    DECLARE v_idFactoAmbientalMenor INT DEFAULT 0;
```



```

DECLARE v_temp INT DEFAULT 0;
DECLARE v_temp2 INT DEFAULT 0;

SELECT af.id_empleado , COUNT(af.id_muestra)
INTO v_idEmpleadoMenor, v_temp
FROM analisis_fauna af
GROUP BY af.id_empleado
ORDER BY COUNT(af.id_muestra) ASC
LIMIT 1;

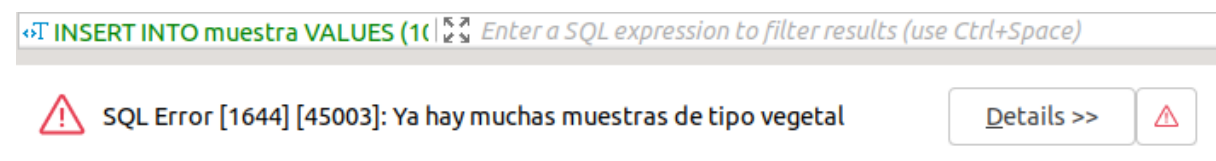
SELECT af.id_factoAmbiental, COUNT(af.id_muestra)
INTO v_idFactoAmbietantalMenor, v_temp2
FROM analisis_fauna af
GROUP BY af.id_factoAmbiental
ORDER BY COUNT(af.id_muestra) ASC
LIMIT 1;

IF (NEW.classification = 'animal') THEN
    INSERT INTO analisis_fauna (id_muestra,id_factoAmbiental,
    id_empleado,legth, wins, color) VALUES (
    NEW.id , v_idFactoAmbietantalMenor, v_idEmpleadoMenor, 0.0,
    'no', 'none');
ELSE
    SIGNAL SQLSTATE '45003'
    SET message_text='Ya hay muchas muestras de tipo
    vegetal';
END IF;
END$$
DELIMITER ;

INSERT INTO muestra VALUES (1001,
'prueba','prueba','prueba','prueba','vegerable', 0.0);

```

En este primer caso podemos ver que la clasificación no coincide con animal, por lo tanto cabe esperar que nos saque por consola indicada.



```

INSERT INTO muestra VALUES (1001,
'prueba','prueba','prueba','prueba','animal', 0.0);

```

En esta caso al tener el tipo de clasificación de tipo animal, cabe esperar que tengamos en la tabla muestra un nuevo elemento. Además de tener en la tabla análisis fauna un nuevo elemento con el id_empleado = 5 e id_factoAmbiental = 2.

Tabla muestra:

muestra							
Enter a SQL expression to filter results (use Ctrl+Space)							
Grid	id	name	family	genus	name_scientific	classification	weight
1	1,001	prueba	prueba	prueba	prueba	animal	0
2	1,000	Red-billed tropic bird	Asteraceae	Phaethon	Phaethon aethereus	vegetable	43.47
3	999	Striated heron	Lecanoraceae	Butorides	Butorides striatus	vegetable	22.77
4	998	Australian brush turkey	Rhizocarpaceae	Alectura	Alectura lathamii	vegetable	64.62
5	997	Oriental white-backed vulture	Fabaceae	Cyns	Cyns bengalensis	vegetable	28.89

Tabla análisis_fauna:

analisis_fauna							
Grid	id_muestra	id_factoAmbiental	id_empleado	length	wins	color	
1	1,001	2	5	0	no	none	
2	696	1	13	189.5	no	mostaza	
3	695	1	12	33.13	si	mostaza	
4	694	1	4	182.73	no	mostaza	
5	693	1	15	158.63	no	mostaza	

- Crea la siguiente tabla, revisión Nombre Científico. En esta tabla guardaremos los nombres antiguos y nuevos de las especies que han sido renombradas (actualizadas) tras análisis filogenéticos. La tabla está compuesta por:
 - idMuestra
 - generoAntiguo
 - EspecieAntigua
 - generoNuevo
 - EspecieNueva

Para la comprobación del correcto funcionamiento del trigger modificaremos el género y nombre científico de la muestra con id=5 (aquella que está marcada en la imagen).

Grid	id	name	family	genus	name_scientific	classification	weight
1	1	Possum	Agavaceae	Pseudocheirus	Pseudocheirus peregrinus	animal	2.25
2	2	Galapagos sea lion	Chenopodiaceae	Zalophus	Zalophus californicus	vegetable	37.56
3	3	Toddy cat	Cuscutaceae	Paradoxurus	Paradoxurus hermaphroditus	vegetable	84.17
4	4	Helmeted guinea fowl	Euphorbiaceae	Numida	Numida meleagris	animal	51.76
5	5	Arctic lemming	Brassicaceae	Dicrostonyx	Dicrostonyx groenlandicus	vegetable	85.83
6	6	Common turkey	Boraginaceae	Meleagris	Meleagris gallopavo	vegetable	84.51

```
CREATE TABLE revision_nom_cientifico(
    idMuestra INT,
    generoAntiguo VARCHAR (16),
    especieAntigua VARCHAR (30),
    generoNuevo VARCHAR (16),
    especieNueva VARCHAR (30)
);
```

```
DELIMITER $$
```

```
CREATE TRIGGER t_revisionNombres
```

```
BEFORE UPDATE ON muestra FOR EACH ROW
```

```
BEGIN
```

```
    INSERT INTO revision_nom_cientifico (idMuestra, generoAntiguo,
    especieAntigua, generoNuevo, especieNueva) VALUES (
    OLD.id, OLD.genus, OLD.name_scientific, NEW.genus,
    NEW.name_scientific
    );
```

```
END$$
```

```
DELIMITER ;
```

```
UPDATE muestra m SET m.genus = 'Gatinyo', m.name_scientific = 'Gatinyo
Familiaris' WHERE m.id = 5;
```

Comprobamos que se ha actualizado correctamente los campos indicados. Además de comprobar que se han insertado en la nueva tabla los datos indicados.

	123 id	abc name	abc family	abc genus	abc name_scientific	abc classification	123 weight
1	1	Possum	Agavaceae	Pseudocheirus	Pseudocheirus peregrinus	animal	2.25
2	2	Galapagos sea lion	Chenopodiaceae	Zalophus	Zalophus californicus	vegetable	37.56
3	3	Toddy cat	Cuscutaceae	Paradoxurus	Paradoxurus hermaphroditus	vegetable	84.17
4	4	Helmeted guinea fowl	Euphorbiaceae	Numida	Numida meleagris	animal	51.76
5	5	Arctic lemming	Brassicaceae	Gatinyo	Gatinyo Familiaris	vegetable	85.83
6	6	Common turkey	Boraginaceae	Meleagris	Meleagris gallopavo	vegetable	84.51

En la tabla podemos comprobar que el generoAntiguo y especieAntigua corresponden con el genus y name_scientific previa a la actualización. Mientras que en el generoNuevo y la especieNueva correspondiente tras la actualización.

	123 idMuestra	abc generoAntiguo	abc especieAntigua	abc generoNuevo	abc especieNueva
1	5	Dicrostonyx	Dicrostonyx groenlandicus	Gatinyo	Gatinyo Familiaris

10. CONCLUSIONES

Para terminar, podemos decir que el desarrollo de este proyecto no solo nos ha permitido conocer de primera mano cómo hacer consultas, vistas, funciones, procedimientos y trigger. Sino que nos ha permitido conocer cómo nace una base de datos.

El enunciado en sí es lo más rápido de escribir, dado que es plasmar tus ideas con palabras que permitan posteriormente desarrollar el modelo entidad relación. Pero llegados ya al segundo punto nuestra paciencia se pone a juego, dado que el modelo relacional debe ser más normalizado y coherente para poder pasarlo a tablas. En este punto lo más complicado es determinar las relaciones y los tributos de las relaciones N:M.

Una vez superado el paso a tablas es un poco más fácil dado que tienes que ir comprobando que tu tabla cumple con los atributos (campos) del modelo Entidad-Relación. Pero es en la carga masiva de datos lo que conlleva una gran paciencia, buena vista y memoria, dado que las tablas creadas en CSV deben de cumplir con las mismas características que tus tablas creadas en DBeaver, sino la subida de datos será imposible. Los principales problemas en este paso son las repeticiones de filas, las Primary Key que no cumplen los requisitos, los datos que no son de la misma unidad que nuestra tabla... entre otras. Pero con paciencia, determinación y con buen ojos estos problemas tienen solución.

Por su parte los apartados de creación de consultas, vistas, procedimientos, funciones y trigger son más de imaginación. Esto en sí es una piedra más en el camino dado que no sabes si es muy simple, demasiado complejo o sin sentido. Los procedimientos en sí para crear estos apartados son simplemente el entendimiento de los necesarios para que funcionen estos apartados.

En resumen este proyecto no solo me ha permitido conocer mano a mano la creación de una pseudo base de datos, dado que no se compara con las reales, en todos sus aspectos. Considero que aunque el género que he elegido es muy complejo dado que se ha simplificado el desarrollo de investigación de análisis de biodiversidad es un pequeño paso para afianzar los conocimientos de la asignatura.