

DOCUMENTACION PROYECTO FINAL MYSQL-MELISSA MENESES ACEVEDO

Ejercicio Asignado: El parque zoo Santafé “parque de la conservación”



Para comenzar a realizar este proyecto final, enfocado en el consumo de alimentos por los animales en el Parque de la Conservación, empecé por construir correctamente el modelo entidad relación que modelaba este sistema:

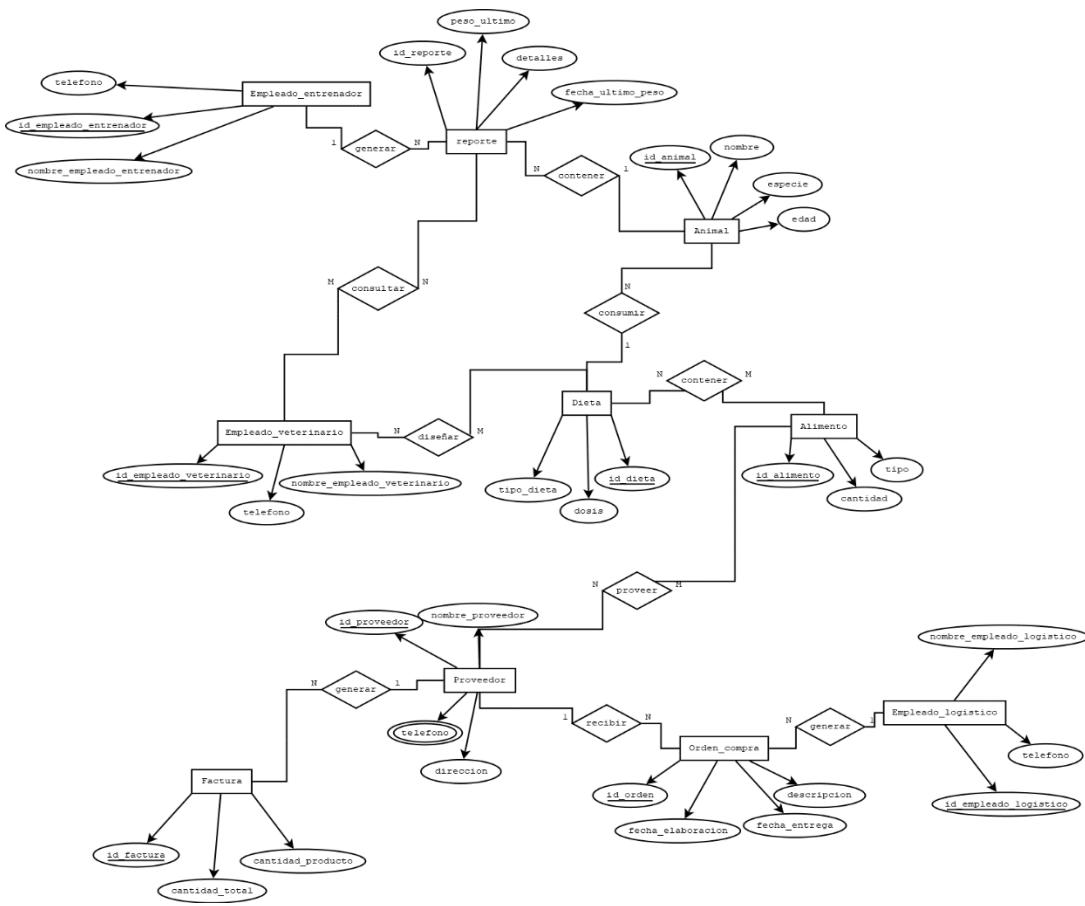
- ★ Lo primero que hice, fue determinar que entidades eran importantes para esta construcción, es decir cuáles eran las principales y aquellas que son importantes para que este sistema de alimento funcione: empleado_logistico , factura, proveedor, alimento, reporte, animal, empleado_entrenador, dieta, empleado_veterinario y orden de compra.
- ★ Tras esto, lo que hago es que, para cada una de las entidades, identifico sus atributos, simples o compuestos y defino para cada una sus llaves primarias.
- ★ Luego, tras tener los pasos anteriores claros y bien definidos, procedo a establecer e identificar las posibles relaciones entre aquellas entidades, determinando las siguientes:
 - ⊕ Empleado_entrenador-reporte: (1-N)un entrenador puede generar uno o varios reportes, y a su vez, un reporte puede ser generado por un entrenador.
 - ⊕ Reporte-animal: (N:1)un reporte puede contener un animal, y, un animal puede estar contenido uno o varios reportes.
 - ⊕ Reporte-empleado_veterinario: (N:M) un veterinario puede consultar uno o varios reportes y de igual forma, un reporte puede ser consultado por uno varios veterinarios.
 - ⊕ Empleado_veterinario-dieta: (N:M)un veterinario puede diseñar una o varias dietas y , una dieta puede ser diseñada por uno varios veterinarios.

- Animal-dieta: (N:1) un animal puede consumir una dieta, y una dieta puede ser consumida por uno o varios animales.
- Dieta-alimento: (N:M) una dieta puede contener uno o muchos alimentos y un alimento puede estar contenido en una o varias dietas.
- Alimento-proveedor: (N:M) un alimento puede ser proveído por uno o muchos proveedores y, un proveedor puede proveer uno o varios alimentos.
- Proveedor-factura: (1:N) un proveedor puede generar una o varias facturas, pero una factura puede ser generada por un proveedor.
- Empleado_logistico-orden_compra: (1:N) un logístico puede generar una o varias órdenes de compra, pero una orden de compra debe ser generada por un logístico.
- Orden_compra-proveedor: (N:1) una orden de compra la recibe un proveedor, pero un proveedor puede recibir una o varias órdenes de compra.

Aclaración: la entidad "proveedor" está indirectamente relacionada con la entidad "orden de compra" a través del empleado logístico, ya que es el empleado logístico quien emite la orden de compra al proveedor para obtener el alimento necesario para las especies.

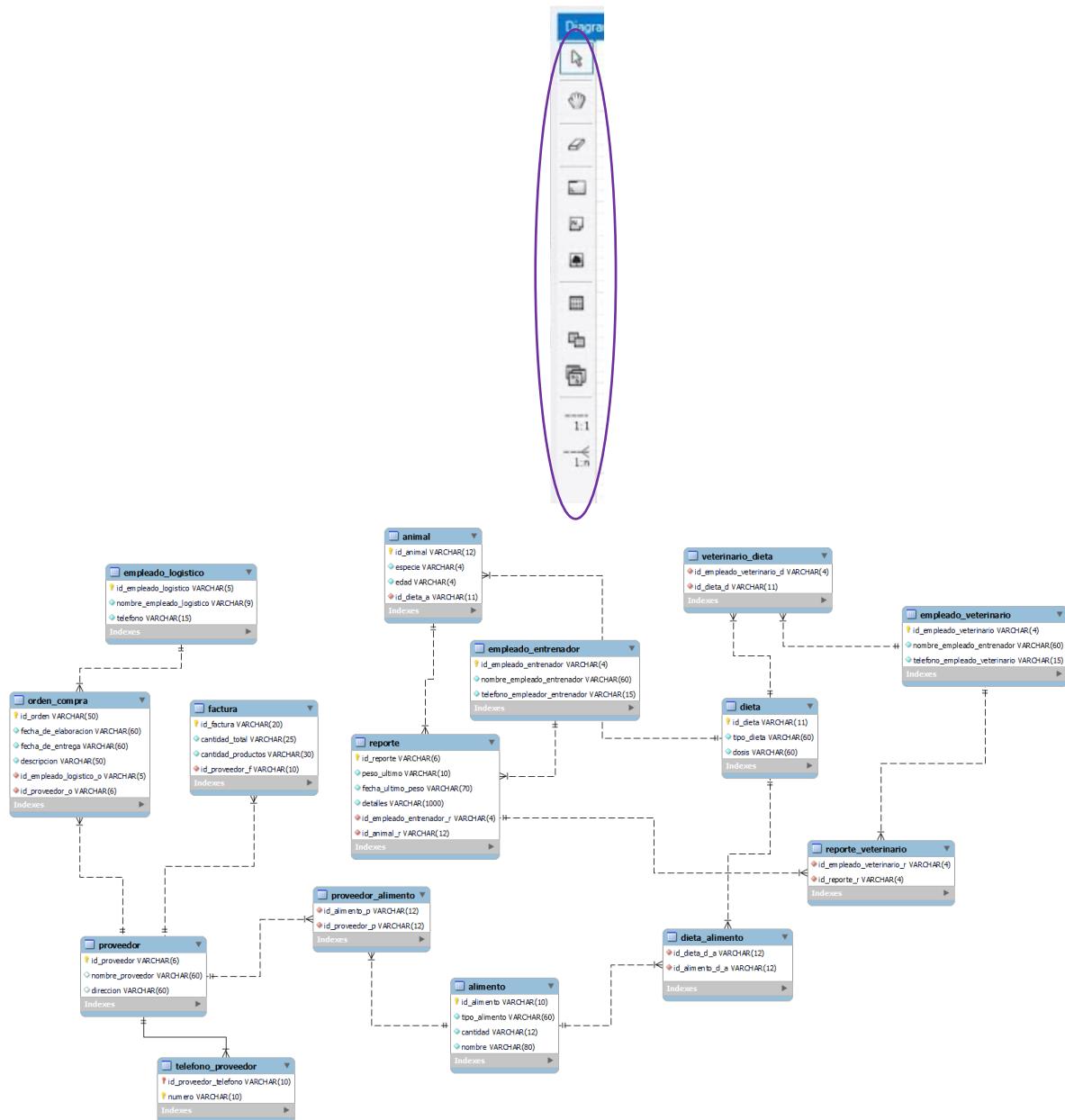
- ★ Luego de haber hecho todo esto, procedí a normalizar para asegurar que los próximos datos a ingresar se almacenen de manera eficiente y libre de redundancias en la base de datos relacional. Entonces esto fue lo que validé con mi modelo:
 - La primera forma normal establece que cada atributo de una tabla debe contener un único valor y no una lista de valores. Esto significa que cada celda en una tabla debe contener un solo valor atómico y no debe contener múltiples valores o una lista separada por comas. **CHECK**
 - La segunda forma normal establece que cada atributo no clave de una tabla debe depender completamente de la clave primaria. Esto significa que cada columna en una tabla debe estar relacionada directamente con la clave primaria de la tabla, y no con una clave candidata secundaria. **CHEK**
 - La tercera forma normal establece que cada atributo de una tabla debe ser independiente de otros atributos que no sean clave. Esto significa que no debe haber dependencias transitivas entre los atributos de la tabla, es decir, un atributo no debe depender de otro atributo que no sea clave, sino que debe depender directamente de la clave primaria. **CHECK**

Finalmente, tras haber validado que mi modelo cumpliera las reglas y estuviese creado eficientemente y libre de redundancias , así fue como me quedó :



Posterior a todo lo anterior, transformé mi modelo entidad relación a un modelo relacional por medio del programa Wokbench de la siguiente manera:

1. Transformando mis entidades principales en tablas
2. Agregando los atributos correspondientes y así mismo las llaves a mis tablas
3. Luego, empiezo a crear las relaciones entre mis entidades teniendo claro si son 1:N en donde la entidad correspondiente al muchos recibe la clave primaria de la 1 y queda como una foránea o si son N:M en donde debo crear un tabla intermedia que me recibe las dos claves primarias de las entidades relacionadas pero que me quedan finalmente como dos foráneas.
4. Y validando estas relaciones, y asegurándome de conectar las tablas correctamente usando los símbolos de la imagen de abajo, así quedaría el modelo relacional listo:



Luego, tras esto creé también mis sentencias SQL:

CREATE TABLE – para crear mi nueva tabla

ATRIBUTO VARCHAR (CANTIDAD DE CARACTERES) NOT NULL + PRIMARY KEY (si es el caso)

FOREIGN KEY – para las llaves foráneas + REFERENCES + TABLA+ (LLAVE PRIMARIA DE LA TABLA)

PRIMARY KEY – para las llaves primarias

```
1 • CREATE DATABASE ZOO_santaFe;
2 USE ZOO_santaFe;
3
4
5 • CREATE TABLE empleado_entrenador(
6   id_empleado_entrenador VARCHAR(4) NOT NULL PRIMARY KEY,
7   nombre_empleado_entrenador VARCHAR(60) NOT NULL,
8   telefono_empleador_entrenador VARCHAR(15) NOT NULL
9 );
10
11
12 • CREATE TABLE empleado_veterinario(
13   id_empleado_veterinario VARCHAR(4) NOT NULL PRIMARY KEY,
14   nombre_empleado_entrenador VARCHAR(60) NOT NULL,
15   telefono_empleado_veterinario VARCHAR(15) NOT NULL
16 );
17
18 • CREATE TABLE reporte_veterinario(
19   id_empleado_veterinario_r VARCHAR(4) NOT NULL,
20   id_reporte_r VARCHAR(4) NOT NULL,
21   FOREIGN KEY (id_empleado_veterinario_r) REFERENCES empleado_veterinario(id_empleado_veterinario),
22   FOREIGN KEY (id_reporte_r) REFERENCES reporte(id_reporte)
23
24
25 • CREATE TABLE animal (
26   id_animal VARCHAR(12) NOT NULL PRIMARY KEY,
27   especie VARCHAR(4) NOT NULL,
28   edad VARCHAR(4) NOT NULL,
29   id_dieta_a VARCHAR(11) NOT NULL,
30   FOREIGN KEY (id_dieta_a) REFERENCES dieta(id_dieta)
31 );
32
33 • CREATE TABLE dieta (
34   id_dieta VARCHAR(11) NOT NULL PRIMARY KEY,
35   tipo_dieta VARCHAR(60) NOT NULL,
36   dosis VARCHAR(6) NOT NULL
37 );
38
39
40 • CREATE TABLE alimento(
41   id_alimento VARCHAR(10) NOT NULL PRIMARY KEY,
42   tipo_alimento VARCHAR(60) NOT NULL,
43   cantidad VARCHAR(12) NOT NULL
44 );
45
46 • CREATE TABLE proveedor(
47   id_proveedor VARCHAR(5) NOT NULL PRIMARY KEY,
48   nombre_proveedor VARCHAR(60),
49   direccion VARCHAR(60)
50 );
51
52 • CREATE TABLE telefono_proveedor(
53   id_proveedor_telefono VARCHAR(10) NOT NULL,
54   numero VARCHAR(10) NOT NULL,
55   PRIMARY KEY(id_proveedor_telefono,numero),
56   FOREIGN KEY (id_proveedor_telefono) REFERENCES proveedor(id_proveedor)
57 );
58
59 • CREATE TABLE factura(
60   id_factura VARCHAR(20) NOT NULL PRIMARY KEY,
61   cantidad_total VARCHAR(25) NOT NULL,
62   cantidad_productos VARCHAR(9) NOT NULL,
63   id_proveedor_f VARCHAR(10) NOT NULL,
64   FOREIGN KEY (id_proveedor_f) REFERENCES proveedor(id_proveedor)
65 );
66
```

```
64 FOREIGN KEY(id_proveedor_f) REFERENCES proveedor(id_proveedor)
65 );
66
67 • CREATE TABLE orden_compra(
68 id_orden VARCHAR(50) NOT NULL PRIMARY KEY,
69 fecha_de_elaboracion VARCHAR(60) NOT NULL,
70 fecha_de_entrega VARCHAR(60) NOT NULL,
71 descripcion VARCHAR(50) NOT NULL,
72 id_empleado_logistico_o VARCHAR(5) NOT NULL,
73 id_proveedor_o VARCHAR(6) NOT NULL,
74 FOREIGN KEY (id_empleado_logistico_o) REFERENCES empleado_logistico(id_empleado_logistico),
75 FOREIGN KEY (id_proveedor_o) REFERENCES proveedor(id_proveedor)
76 );
77
78 • CREATE TABLE empleado_logistico(
79 id_empleado_logistico VARCHAR(5) NOT NULL PRIMARY KEY,
80 nombre_empleado_logistico VARCHAR(9) NOT NULL,
81 telefono VARCHAR(15) NOT NULL
82 );
83
84
85
86 • CREATE TABLE veterinario_dieta(
87 id_empleado_veterinario_d VARCHAR(4) NOT NULL,
88 id_dieta_d VARCHAR(11) NOT NULL,
89 FOREIGN KEY (id_empleado_veterinario_d) REFERENCES empleado_veterinario(id_empleado_veterinario),
90 FOREIGN KEY (id_dieta_d) REFERENCES dieta(id_dieta)
91 );
92
93
94
95
96 • CREATE TABLE dieta_alimento(
97 id_dieta_d_a VARCHAR(12) NOT NULL,
98 id_alimento_d_a VARCHAR(12) NOT NULL,
99 FOREIGN KEY (id_dieta_d_a) REFERENCES dieta(id_dieta),
100 FOREIGN KEY (id_alimento_d_a) REFERENCES alimento(id_alimento)
101 );
102
103
104
105
106
107
108
109
110 • CREATE TABLE proveedor_alimento(
111 id_alimento_p VARCHAR(12) NOT NULL,
112 id_proveedor_p VARCHAR(12) NOT NULL,
113 FOREIGN KEY (id_alimento_p) REFERENCES alimento(id_alimento),
114 FOREIGN KEY (id_proveedor_p) REFERENCES proveedor(id_proveedor)
115 );
116
117
118 • CREATE TABLE reporte(
119 id_reporte VARCHAR(5) NOT NULL PRIMARY KEY,
120 peso_ultimo VARCHAR(10) NOT NULL,
121 fecha_ultimo_peso VARCHAR(70) NOT NULL,
122 detalles VARCHAR(1000) NOT NULL,
123 id_empleado_entrenador_r VARCHAR(4) NOT NULL,
124 id_animal_r VARCHAR(12) NOT NULL,
125 FOREIGN KEY (id_empleado_entrenador_r) REFERENCES empleado_entrenador(id_empleado_entrenador),
126 FOREIGN KEY (id_animal_r) REFERENCES animal(id_animal)
```

Luego de crear mi Script, continué insertando registros a mi base de datos de la siguiente manera:

```
INSERT INTO + NOMBRE TABLA+ VALUES (INGRESO LOS VALORES);
```

Nota: en la sentencia de arriba, me di cuenta de que no era necesarios escribir todos los atributos de mi tabla como parámetros, debido a que si voy a ingresar datos para cada columna, no es necesario hacerlo.

Luego, así fue como ingresé mis datos, pero antes... ¿Con que intención inserté estos datos?

Con la intención de poder tener algunos pocos registros en mi base de datos, en algunas tablas, para así poder validar el correcto funcionamiento de mis consultas, para ver si me funcionaban:

```
121 -- inserto algunos regisrtos para validar las consultas
122 • INSERT INTO dieta VALUES ('01A','carnivora','9 kilos diarios');
123 • INSERT INTO dieta VALUES('01B','marina','5 kilos pescados y 3 kilos mariscos');
124
125 • INSERT INTO animal VALUES ('1001','leon','10 años','01A');
126 • INSERT INTO animal VALUES ('10015','foca','15 años','01B');
127 • INSERT INTO animal VALUES ('1006','lobo','5años','01B');
128
129 • INSERT INTO proveedor VALUES('M-01','Albeiro Sepulveda','Fontidueño 8A-13');
130
131 • INSERT INTO factura VALUES('F-001','5.126.350','390 toneladas','M-01');
132
133 • INSERT INTO alimento VALUES('L-001','humedo','10 kilos','carne fria + jamon');
134
135 • INSERT INTO empleado_veterinario VALUES('v001','Fermin Aguirre','3112225499');
136
137 • INSERT INTO empleado_logistico VALUES ('1001','Sara Roma','3112758950');
138
139 • INSERT INTO empleado_entrenador VALUES ('e001','Camilo Perez','32000025021');
140
141 • INSERT INTO telefono_proveedor VALUES('M-01','6045896523');
142 • INSERT INTO telefono_proveedor VALUES('M-01','3112887544');
```

Y, tras haber dejado esos registros listos, procedí a crear mis consultas :

```

149    -- consulta para ver las dietas de todos los animales
150 •  SELECT animal.id_animal, dieta.tipo_dieta, dieta.dosis
151   FROM animal
152   JOIN dieta ON animal.id_dieta_a = dieta.id_dieta;
153
154    -- consulta para obtener el promedio de edad de todos los animales
155 •  SELECT AVG(edad) AS promedio_edad
156   FROM animal;
157
158    -- Consulta para obtener todos los detalles de la dieta del animal con ID '1001':
159 •  SELECT *
160   FROM dieta d
161   JOIN animal a
162   ON d.id_dieta = a.id_dieta_a
163   WHERE a.id_animal = '1001';
164

165    -- consulta para ver los detalles de todas las facturas, incluyendo el nombre del proveedor,
166    -- la cantidad total de productos comprados y la fecha de entrega de cada orden de compra:
167
168 •  SELECT f.id_factura, p.nombre_proveedor, f.cantidad_total, f.cantidad_productos, o.fecha_de_entrega
169   FROM factura f
170   JOIN proveedor p ON f.id_proveedor_f = p.id_proveedor
171   JOIN orden_compra o ON f.id_proveedor_f = o.id_proveedor_o;
172
173    -- Consulta para obtener todos los proveedores de alimentos y sus números de teléfono:
174 •  SELECT p.nombre_proveedor, tp.numero
175   FROM proveedor p
176   JOIN telefono_proveedor tp
177   ON p.id_proveedor = tp.id_proveedor_telefono;
178
179    -- Consulta para obtener todos los detalles de la factura con ID 'F-001':
180 •  SELECT *
181   FROM factura
182   WHERE id_factura = 'F-001';
183
184

185    -- Consulta para obtener el nombre y teléfono del empleado logístico responsable de la orden de compra con ID '0-111':
186 •  SELECT e.nombre_empleado_logistico, e.telefono
187   FROM empleado_logistico e
188   JOIN orden_compra o ON e.id_empleado_logistico = o.id_empleado_logistico_o
189   WHERE o.id_orden = '0-111';
190
191    -- Consulta para obtener el número de teléfono de todos los empleados entrenadores:
192 •  SELECT telefono_empleador_entrenador
193   FROM empleado_entrenador;
194
195    -- Consulta para obtener los detalles de la orden de compra más reciente:
196 •  SELECT *
197   FROM orden_compra
198   ORDER BY fecha_de_elaboracion DESC LIMIT 1;
199
200    -- Consulta para obtener el número de teléfono y dirección del proveedor con ID 'M-01':
201 •  SELECT p.nombre_proveedor, tp.numero, p.direccion
202   FROM proveedor p
203   JOIN telefono_proveedor tp ON p.id_proveedor = tp.id_proveedor_telefono
204   WHERE p.id_proveedor = 'M-01';

```

Las consultas que creé fueron las siguientes:

1. "consulta para ver las dietas de todos los animales": En esta consulta uní las tablas de "animal" y "dieta" y selecciona el id del animal, el tipo de dieta y la dosis de la dieta para cada animal.

-
2. "consulta para obtener el promedio de edad de todos los animales": En esta consulta seleccioné la edad de todos los animales y calcula el promedio de edad.
 3. "Consulta para obtener todos los detalles de la dieta del animal con ID '1001)": EN esta consulta uní las tablas de "animal" y "dieta" y selecciona todos los detalles de la dieta para el animal con el id 1001.
 4. "consulta para ver los detalles de todas las facturas": En esta consulta une las tablas de "factura", "proveedor" y "orden_compra" y selecciona el id de la factura, el nombre del proveedor, la cantidad total de productos comprados y la fecha de entrega de cada orden de compra.
 5. "Consulta para obtener todos los proveedores de alimentos y sus números de teléfono": En esta consulta uní las tablas de "proveedor" y "telefono_proveedor" y selecciona el nombre del proveedor y su número de teléfono.
 6. "Consulta para obtener todos los detalles de la factura con ID 'F-001)": En esta consulta seleccioné todos los detalles de la factura con el id "F-001".
 7. "Consulta para obtener el nombre y teléfono del empleado logístico responsable de la orden de compra con ID 'O-111)": En esta consulta uní las tablas de "empleado_logistico" y "orden_compra" y selecciona el nombre y número de teléfono del empleado logístico responsable de la orden de compra con el id "O-111".
 8. "Consulta para obtener el número de teléfono de todos los empleados entrenadores": En esta consulta seleccioné el número de teléfono de todos los empleados entrenadores.
 9. "Consulta para obtener los detalles de la orden de compra más reciente": En esta consulta seleccioné todos los detalles de la orden de compra más reciente.

10. "Consulta para obtener el número de teléfono y dirección del proveedor con ID 'M-01": En esta consulta uní las tablas de "proveedor" y "telefono_proveedor" y seleccioné el nombre del proveedor, su número de teléfono y su dirección para el proveedor con el id "M-01".

11. "Consulta para obtener todos los reportes veterinarios de los animales": Finalmente, en esta consulta uní las tablas de "animal", "reporte" y "empleado_entrenador" y seleccioné el id del animal, el peso más reciente del animal, la fecha del peso más reciente, los detalles del reporte, el nombre del entrenador y su número de teléfono.

Luego de esto, continué identificando y creando las vistas:

La primera vista, llamada `vista_entrenadores`, la utilicé para llevar un registro de los entrenadores y los animales que les han sido asignados. La vista la creé a partir de una consulta que combina datos de tres tablas diferentes: la tabla `empleado_entrenador`, que contiene información sobre los entrenadores; la tabla `reporte`, que contiene información sobre el peso y la salud de los animales; y la tabla `animal`, que contiene información sobre los animales en sí mismos. En la consulta utilicé JOINs para combinar los datos de estas tablas y selecciono los campos `nombre_empleado_entrenador` y `especie`.

Esta vista, la consideré importante ya que, partiendo de un contexto real , en un zoológico , pienso que es relevante llevar un registro en una base de datos sobre los entrenadores y sus animales asignados , pues facilita y así mismo lleva un orden del sistema y puede resultar muy eficiente en muchas situaciones.

La segunda vista que creé, llamada `compras_por_proveedor`, la utilicé para mostrar información sobre las compras que se han realizado a cada proveedor. La vista la creé a partir de una consulta que combina datos de dos tablas diferentes: la tabla `proveedor`, que contiene información sobre los proveedores; y la tabla `factura`, que contiene información sobre las compras realizadas. En la consulta utilicé un LEFT JOIN para incluir todos los proveedores, incluso aquellos que no han tenido ninguna compra, y luego agrupé los resultados por `id_proveedor`, calculando la cantidad total de compras, el total de compras y el total de productos comprados.

La segunda vista la creé a causa de que, aplicando el ejercicio a la práctica más real , al contexto del zoológico , llevar un buen registro en la base de datos sobre las compras que se le hacen a un proveedor resulta indispensable para llevar un orden y cuidar el sistema financiero del zoológico, qué entra , cada cuanto entra , cuanto entra , y cuanto sale por ejemplo.

La tercera vista, llamada dieta_prescrita, la utilicé para mostrar información sobre las dietas prescritas por los veterinarios. La vista la creé a partir de una consulta que combina datos de tres tablas diferentes: la tabla dieta, que contiene información sobre las dietas; la tabla veterinario_dieta, que relaciona las dietas con los veterinarios que las prescriben; y la tabla empleado_veterinario, que contiene información sobre los veterinarios. En la consulta utilicé JOINs para combinar los datos de estas tablas y selecciona los campos id_dieta, tipo_dieta, dosis, id_empleado_veterinario, y nombre_empleado_entrenador. La verdad, es que esta vista me gustó mucho porque considero que llevar un registro de dietas prescritas para los animales , es un indicio de orden y efectividad en un sistema , y esto me asegura que en este sistema todas y cada una de las personas adecuadas pueden saber en cualquier momento a que animal le corresponde qué dieta , además como nuestro ejercicio estaba enfocado al consumo de alimentos por los animales , llevar este control es indispensable para un buen funcionamiento.

La cuarta vista, llamada informe_salud_animal, la utilicé para mostrar información sobre el estado de salud de los animales. La vista la creé a partir de una consulta que combina datos de dos tablas diferentes: la tabla animal, que contiene información sobre los animales; y la tabla reporte, que contiene información sobre su salud. En la consulta utilicé un INNER JOIN para combinar los datos de estas tablas y seleccioné los campos id_animal, especie, edad, fecha_ultimo_peso, peso_ultimo y detalles. En esta vista seleccioné el informe más reciente para cada animal, utilizando una subconsulta para seleccionar la fila con la fecha más reciente.

Y, finalmente, construí esta vista porque me parece súper importante también, debido a que según el estado de salud de los animales, se les receta una dieta , que contendrá ciertos alimentos , así que , tener un control y llevar un registro de estos estados de salud es super importante , si mismo como estar modificándolos , insertando o eliminándolos según los casos.

Así se ven las vistas que creé:

```
Query 1 script_zoologico_santafe.x
213 -- vista para llevar registro de entrenadores y sus animales asignados
214 • CREATE VIEW vista_entrenadores AS
215   SELECT e.nombre_empleado_entrenador, a.especie
216   FROM empleado_entrenador e
217   JOIN reporte r ON e.id_empleado_entrenador = r.id_empleado_entrenador_r
218   JOIN animal a ON r.id_animal_r = a.id_animal;
219
220 -- Vista de las compras realizadas a cada proveedor junto con la cantidad total y la cantidad de productos adquiridos
221 • CREATE VIEW compras_por_proveedor AS
222   SELECT p.id_proveedor, p.nombre_proveedor, COUNT(f.id_factura) AS cantidad_compras, SUM(f.cantidad_total) AS total
223   FROM proveedor p
224   LEFT JOIN factura f ON f.id_proveedor_f = p.id_proveedor
225   GROUP BY p.id_proveedor;
226
227 -- Vista de la dieta prescrita por el veterinario
228 • SELECT d.id_dieta, d.tipo_dieta, d.dosis, e.id_empleado_veterinario, e.nombre_empleado_entrenador
229   FROM dieta d
230   INNER JOIN veterinario_dieta vd ON vd.id_dieta_d = d.id_dieta
231   INNER JOIN empleado_veterinario e ON e.id_empleado_veterinario = vd.id_empleado_veterinario_d;
232
233 -- Vista de informe de salud del animal:
234 • CREATE VIEW informe_salud_animal AS
235   SELECT a.id_animal, a.especie, a.edad, r.fecha_ultimo_peso, r.peso_ultimo, r.detalles
236   FROM animal a
```

Tras esto, seguí también creando los procedimientos, pero antes me aseguré nuevamente de entender la teoría, de donde pude entender y reforzar lo siguiente: Los procedimientos se utilizan a menudo para simplificar y estandarizar tareas comunes o complejas que deben realizarse en una base de datos, como la inserción o actualización de registros, la generación de informes, la validación de datos, la creación de tablas, entre otros.

La principal ventaja de los procedimientos es que pueden reducir la cantidad de código repetitivo y mejorar la eficiencia del rendimiento, ya que el código del procedimiento se compila y se ejecuta una vez en lugar de enviar múltiples solicitudes a la base de datos. Además, los procedimientos pueden proteger la base de datos contra el acceso no autorizado y la manipulación de datos, ya que el acceso a los procedimientos puede ser controlado por permisos y roles de usuario.

Y entonces, uno de los procedimientos que creé fueron estos:

- Procedimiento para insertar un nuevo empleado entrenador: Este procedimiento inserta un nuevo empleado entrenador en la tabla de "empleado_entrenador" con los valores proporcionados en los parámetros.
- Procedimiento para mostrar los animales que consumen una dieta específica: Este procedimiento muestra los detalles de todos los animales en la tabla de "animal" que tienen una dieta especificada en el parámetro "dieta_a".
- Procedimiento para insertar un nuevo reporte veterinario: Este procedimiento inserta un nuevo reporte veterinario en la tabla "reporte_veterinario" con los valores proporcionados en los parámetros.
- Procedimiento para insertar un nuevo animal: Este procedimiento inserta un nuevo animal en la tabla de "animal" con los valores proporcionados en los parámetros.

- Procedimiento para agregar una nueva dieta: Este procedimiento agrega una nueva dieta en la tabla de "dieta" con los valores proporcionados en los parámetros.
- Procedimiento para actualizar la edad de un animal: Este procedimiento actualiza la edad de un animal en la tabla de "animal" con el valor proporcionado en el parámetro.
- Procedimiento para mostrar la dosis de todos los animales en el zoo: Este procedimiento muestra los detalles de la dosis de cada animal en la tabla de "animal" que están relacionados con la tabla "dieta" a través de la columna "id_dieta_a".

Aquí la imagen de los procedimientos creados:

La forma que utilicé para hacer esto fue:

```
CREATE PROCEDURE nombre_del_procedimiento (lista_de_parámetros)
```

```
BEGIN
```

```
-- Código del procedimiento aquí
```

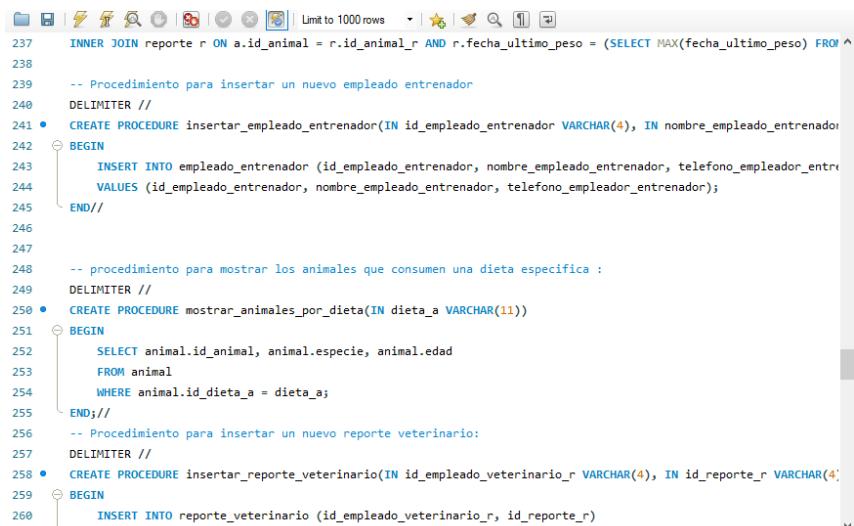
```
END;
```

Donde:

nombre_del_procedimiento: es el nombre que se le asigna al procedimiento almacenado.

lista_de_parámetros: es una lista de parámetros opcionales que se pueden utilizar para recibir valores de entrada y salida en el procedimiento.

BEGIN y END: son las palabras clave que indican el inicio y fin del cuerpo del procedimiento almacenado.



```

237 INNER JOIN reporte r ON a.id_animal = r.id_animal_r AND r.fecha_ultimo_peso = (SELECT MAX(fecha_ultimo_peso) FROM
238
239 -- Procedimiento para insertar un nuevo empleado entrenador
240 DELIMITER //
241 • CREATE PROCEDURE insertar_empleado_entrenador(IN id_empleado_entrenador VARCHAR(4), IN nombre_empleado_entrenador
242 BEGIN
243     INSERT INTO empleado_entrenador (id_empleado_entrenador, nombre_empleado_entrenador, telefono_empleador_entrenador)
244     VALUES (id_empleado_entrenador, nombre_empleado_entrenador, telefono_empleador_entrenador);
245 END//
246
247
248 -- procedimiento para mostrar los animales que consumen una dieta específica :
249 DELIMITER //
250 • CREATE PROCEDURE mostrar_animales_por_dieta(IN dieta_a VARCHAR(11))
251 BEGIN
252     SELECT animal.id_animal, animal.especie, animal.edad
253     FROM animal
254     WHERE animal.id_dieta_a = dieta_a;
255 END//
256 -- Procedimiento para insertar un nuevo reporte veterinario:
257 DELIMITER //
258 • CREATE PROCEDURE insertar_reporte_veterinario(IN id_empleado_veterinario_r VARCHAR(4), IN id_reporte_r VARCHAR(4))
259 BEGIN
260     INSERT INTO reporte_veterinario (id_empleado_veterinario_r, id_reporte_r)
261     VALUES (id_empleado_veterinario_r, id_reporte_r);
262 END//

```

```

-- ...
256 -- Procedimiento para insertar un nuevo reporte veterinario:
257 DELIMITER //
258 • CREATE PROCEDURE insertar_reporte_veterinario(IN id_empleado_veterinario_r VARCHAR(4), IN id_reporte_r VARCHAR(4),
259 BEGIN
260     INSERT INTO reporte_veterinario (id_empleado_veterinario_r, id_reporte_r)
261     VALUES (id_empleado_veterinario_r, id_reporte_r);
262 END;//
263
264 -- Procedimiento para insertar un nuevo animal:
265 DELIMITER //
266 • CREATE PROCEDURE insertar_animal(IN id_animal VARCHAR(12), IN especie VARCHAR(4), IN edad VARCHAR(4), IN id_dieta_a
267 BEGIN
268     INSERT INTO animal (id_animal, especie, edad, id_dieta_a)
269     VALUES (id_animal, especie, edad, id_dieta_a);
270 END;//
271
272 -- procedimiento para agregar una nueva dieta
273 DELIMITER //
274 • CREATE PROCEDURE agregar_dieta(IN id_dieta VARCHAR(11), IN tipo_dieta VARCHAR(60), IN dosis VARCHAR(6))
275 BEGIN
276     INSERT INTO dieta (id_dieta, tipo_dieta, dosis) VALUES (id_dieta, tipo_dieta, dosis);
277 END;//
278
279 -- procedimiento para actualizar la edad de un animal
280 DELIMITER //
281 • CREATE PROCEDURE actualizar_edad_animal(IN id_animal VARCHAR(12), IN edad VARCHAR(4))
282 BEGIN
283     UPDATE animal SET edad = edad WHERE id_animal = id_animal;
284 END;//
285
286 -- procedimiento para mostrar la dosis de todos los animales en el zoo
287 DELIMITER //
288
289 • CREATE PROCEDURE mostrar_dosis_animales()
290 BEGIN
291     SELECT a.id_animal, d.tipo_dieta, d.dosis
292     FROM animal a
293     JOIN dieta d ON a.id_dieta_a = d.id_dieta;
294 END //;
295
296 DELIMITER ;
297

```

Y, finalmente, construí los triggers , que son objetos que se utilizan para automatizar ciertas acciones o procesos en respuesta a eventos específicos que ocurren en la base de datos; es decir, un trigger es como un conjunto de acciones que se activan automáticamente en respuesta a un cambio en la base de datos, como una inserción, actualización o eliminación de datos en una tabla específica.

Estos fue lo que hice:

En primer lugar, definí una tabla llamada "control_de_cambios_zoologico" que se utilizará para registrar los cambios en la base de datos. La tabla tiene las columnas "id_cambio" (clave primaria autoincremental), "usuario" (que indica quién realizó el cambio), "accion" (que indica qué tipo de cambio se realizó), "fecha" (la fecha y hora en que se realizó el cambio) y "lugar" (que indica en qué tabla se realizó el cambio).

Luego, definí varios triggers que se ejecutan en respuesta a ciertos eventos en mi base de datos:

- ★ "insertar_cambio": se ejecuta después de cada inserción en la tabla "animal" y registra el usuario, la acción ("INSERTAR"), la fecha y el lugar ("animal") en la tabla "control_de_cambios_zoologico".
- ★ "trigger_eliminar_animal": se ejecuta después de cada eliminación en la tabla "animal" y registra el usuario, la acción ("Eliminar"), la fecha y el lugar ("animal") en la tabla "control_de_cambios_zoologico".
- ★ "insertar_cambio_dieta": se ejecuta después de cada inserción en la tabla "dieta" y registra el usuario, la acción ("INSERTAR"), la fecha y el lugar ("dieta") en la tabla "control_de_cambios_zoologico".
- ★ "trigger_eliminar_proveedor_alimento": se ejecuta después de cada eliminación en la tabla "proveedor_alimento" y registra el usuario, la acción ("Eliminar"), la fecha y el lugar ("proveedor_alimento") en la tabla "control_de_cambios_zoologico".
- ★ "trigger_modificar_reporte": se ejecuta después de cada actualización en la tabla "reporte" y registra el usuario, la acción ("Modificar"), la fecha y el lugar ("reporte") en la tabla "control_de_cambios_zoologico".

Finalmente, se realizan algunas acciones en la base de datos, como eliminar registros de la tabla "animal". Cada vez que se realiza una acción que dispara un trigger, se registra un nuevo registro en la tabla "control_de_cambios_zoologico". De esta manera, es posible hacer un seguimiento de los cambios realizados en la base de datos y saber quién los realizó y cuándo.

Adjunto imágenes:

```
317    -- trigger que se ejecuta cuando se borra un animal
318    DELIMITER //
319  •  CREATE TRIGGER trigger_eliminar_animal
320    AFTER DELETE ON animal
321    FOR EACH ROW
322    BEGIN
323      INSERT INTO control_de_cambios_zoologico (usuario, accion, fecha, lugar)
324        VALUES (USER(), 'Eliminar', NOW(), 'animal');
325    END//'
326
327    DELETE FROM animal
328    WHERE id_animal= '1006';
329
330    DELETE FROM animal
331    WHERE id_animal= '1003';
332
333    -- trigger que se ejecuta cuando se inserta una nueva dieta
334    DELIMITER //
335  •  CREATE TRIGGER insertar_cambio_dieta
336    AFTER INSERT ON dieta
337    FOR EACH ROW
338      INSERT INTO control_de_cambios_zoologico (usuario, accion, fecha, lugar)
339        VALUES (USER(), 'INSERTAR', NOW(), 'dieta');
340    END//'
```

```
>>>
333    -- trigger que se ejecuta cuando se inserta una nueva dieta
334    DELIMITER //
335 •  CREATE TRIGGER insertar_cambio_dieta
336     AFTER INSERT ON dieta
337     FOR EACH ROW
338     INSERT INTO control_de_cambios_zoologico (usuario, accion, fecha, lugar)
339     VALUES (USER(), 'INSERTAR', NOW(), 'dieta');
340     END//
341
342    -- trigger que se ejecuta cuando se elimina un proveedor
343    DELIMITER //
344 •  CREATE TRIGGER trigger_eliminar_proveedor_alimento
345     AFTER DELETE ON proveedor_alimento
346     FOR EACH ROW
347     BEGIN
348         INSERT INTO control_de_cambios_zoologico (usuario, accion, fecha, lugar)
349         VALUES (USER(), 'Eliminar', NOW(), 'proveedor_alimento');
350     END//
351
352    -- trigger que se ejecuta cuando se modifica un reporte |
353    DELIMITER //
354 •  CREATE TRIGGER trigger_modificar_reporte
355     AFTER UPDATE ON reporte
356     FOR EACH ROW
357     BEGIN
358         INSERT INTO control_de_cambios_zoologico (usuario, accion, fecha, lugar)
359         VALUES (USER(), 'Modificar', NOW(), 'reporte');
360     END//
361
362
363
```

¿Está conforme con el resultado obtenido según el contexto o cree que hubiera obtenido un mejor resultado con una base de datos no relacional?

Creo que la elección entre una base de datos relacional y una no relacional depende de las necesidades específicas del sistema y los datos que se quieran almacenar. Ambos tipos de bases de datos tienen sus propias ventajas y desventajas.

En el caso de un zoológico, es probable y ya evidenciamos con este trabajo que se debe almacenar una gran cantidad de datos, como información sobre animales, alimentación, cuidados, registros médicos, entre otros. Además, ya evidenciamos que hay relaciones entre los datos, como la relación entre un animal y su dieta.

En general, considero que las bases de datos relacionales son más adecuadas para almacenar datos estructurados con relaciones definidas, mientras que las bases de datos no relacionales son más adecuadas para almacenar datos no estructurados o semiestructurados, como datos de sensores o registros de redes sociales.

Dicho lo anterior , una base de datos relacional podría ser la mejor opción para un zoológico en este caso , ya que evidenciamos finalmente que hay muchas relaciones entre los diferentes tipos de datos que se quieren almacenar. Además, las bases de datos relacionales suelen ser más escalables y ofrecen un mayor control de integridad de los datos. Así que si quedé conforme y aprendí mucho.

INYECTAR DATOS DESDE JAVA:

Para iniciar a poblar mi base de datos en MySql desde java, empecé por crear , el proyecto en IntelliJ idea , y allí crear los paquetes que fuese a necesitar:

Asigné, luego un paquete llamado mysql en donde cree dos clases: MySqlOperation y MySqlConstants.

La clase "MySqlOperation" tiene los siguientes atributos:

- Connection: representa una conexión a la base de datos MySQL.
- Statement: representa una declaración SQL que se ejecutará en la base de datos.
- ResultSet: representa el resultado de una consulta a la base de datos.
- sqlStatement: la consulta SQL que se ejecutará en la base de datos.
- server: el nombre del servidor donde se encuentra la base de datos.
- DataBaseName: el nombre de la base de datos a la que se conectará.
- user: el nombre de usuario utilizado para conectarse a la base de datos.
- password: la contraseña utilizada para conectarse a la base de datos.

Esta clase además implementa los métodos de la interfaz "DataBase":

- ★ configureDataBaseConnection(): este método se encarga de configurar la conexión a la base de datos utilizando los atributos server, DataBaseName, user y password. También crea un objeto Statement que se utilizará para ejecutar consultas SQL.
- ★ executeSqlStatement(): este método ejecuta la consulta SQL definida en el atributo sqlStatement y almacena el resultado en el atributo resultSet.
- ★ getResultSet(): este método devuelve el objeto ResultSet que contiene los resultados de una consulta SQL.
- ★ close(): este método se encarga de cerrar los objetos ResultSet, Statement y Connection cuando se termina de utilizarlos.
- ★ printResultSet(): este método imprime el resultado de una consulta SQL en la consola.
- ★ executeSqlStatementVoid(): este método ejecuta una consulta SQL que no devuelve ningún resultado.

Esta clase además utiliza constantes definidas en la clase "MySqlConstants" para establecer la URL de conexión a la base de datos y el nombre del controlador JDBC de MySQL.

Luego de tener todo esto configurado, creé un nuevo paquete llamado `models` , que en Java es una colección de clases que representan los objetos y conceptos del mundo real de una aplicación. Estas clases suelen contener atributos y métodos que representan las características y comportamientos de los objetos que se modelan. As fui que cree 10 clases que así mismo corresponden a mis tablas creadas en la base de datos del Zooologico SnataFe : alimento , animal , dieta , empleado_entrenador, empleado_logistico, empleado_veterianrio,factura, orden_de_compra, proveedor y reporte; luego , en cada una de las clases lo que hice fue crear los atrubutos que así mismo correspodne a las columnas que anteriormente abia creado en mis tablas , por ejemplo , para mi tabla alimento en mi base de datos Zoo_santafe, había creado estas columnas : id_alimento, tipo_aliemnto y cantidad , y entonces para mi clase en java también cree estos atributos:

Table: `alimento`

Columns:
`id_alimento` varchar(10) PK
`tipo_alimento` varchar(60)
`cantidad` varchar(12)

```
2 usages
public class Alimento {
    3 usages
    private String id_alimento;
    3 usages
    private String tipo_alimento ;
    3 usages
    private String cantidad ;
    3 usages
    private String nombre ;

    no usages
    public Alimento(String id_alimento, String tipo_alimento, String cantidad, String nombre) {
        this.id_alimento = id_alimento;
        this.tipo_alimento = tipo_alimento;
        this.cantidad = cantidad;
        this.nombre = nombre;
    }
    1 usage
    public Alimento (){}}

    1 usage
    public String getId_alimento() {
        return id_alimento;
    }
}
```

Así fue como creé todas las clases correspondientes a mi base de datos en java, y luego tras haberlas creado todas ordenadamente con todos sus atributos , constructores y métodos getters and setters , procedí en la clase `Main` , en mi clase principal, a crear las funciones para poder poblar las tablas correspondientes.

El orden que utilicé fue el siguiente:

Creé, por ejemplo, un método llamado `insertarVeterinario()` que inserta 50 objetos `Empleado_veterinario` en una tabla llamada `empleado_veterinario` en una base de datos MySQL.

En este método utiliza un bucle for para iterar 50 veces, creando un nuevo objeto Empleado_veterinario en cada iteración. Cada objeto lo configuro con un ID único, un nombre generado aleatoriamente utilizando una biblioteca de generación de datos falsos llamada "faker", y un número de teléfono generado aleatoriamente.

Luego, este método configura una sentencia SQL de inserción usando los valores de las propiedades del objeto Empleado_veterinario y la asigna al atributo sqlStatement de una instancia de una clase MySqlOperation que implementa la interfaz DataBase y maneja la conexión y operaciones de base de datos. La sentencia SQL es ejecutada llamando al método executeSqlStatementVoid() de la misma instancia mySqlOperation.

En resumen, este método que creé es una forma de insertar múltiples registros en una tabla de base de datos utilizando objetos Java y sentencias SQL dinámicas.

De esta misma forma fuí creando los demás métodos que iba necesitando: insertarEntrnador, insertarAlimento,insertarDieta,insertar_dieta_alimento,insertar_logistico, insertarProveedor, insertarFactura, insertarOrdenCompra, insertarProveedor_alimento,insertarTelefono_proveedor, insertarVeterinarioDieta,insertarReporte, insertarReporte_veterinario e insertarAnimal.

Y tras crear el método, lo llamaba, se ejecutaba y me dirigía a mi base de datos, a una tabla en especial, para validar el ingreso de los 50 registros.

Tabla insertarVeterinario, su función creada y los registros:

The screenshot shows the Oracle Database SQL Developer interface. At the top, the code for the `insertarVeterinario` function is displayed:

```
no usages
private static void insertarVeterinario(){

    for (int i = 0; i < 50; i++) {
        Empleado_veterinario vet = new Empleado_veterinario();
        vet.setId_empleado_veterinario("V"+i);
        vet.setNombre_empleado_veterinario(faker.name().firstName());
        vet.setTelefono_empleado_veterinario(faker.phoneNumber().cellPhone().toString());

        mySqlOperation.setSqlStatement("insert into empleado_veterinario values('"+vet.getId_empleado_veterinario()+
            "','" +vet.getNombre_empleado_veterinario()+"','"+vet.getTelefono_empleado_veterinario()+"');");
        mySqlOperation.executeSqlStatementVoid();
    }
}
```

Below the code, a result grid displays 50 rows of data, each representing a newly inserted record:

id_empleado_veterinario	nombre_empleado_entrenador	telefono_empleado_veterinario
V0	Emerald	597-393-3020
V1	Bennie	(956) 738-6696
V10	Quinton	717-165-0921
V11	Celena	837-059-0549
V12	Ramonita	362-424-2310
V13	Truman	447-080-6920
V14	Herb	062-216-4631
V15	Malcolm	486-359-1481
V16	Clora	138-776-66 [138-776-6618]
V17	Tony	668-767-8783
V18	Lynwood	1-730-323-5107
V19	Tommie	(233) 883-3440
V2	Catharine	885-076-5873
V20	Dionne	1-702-555-7513
V21	Porsha	144-024-1491
V22	Stewart	(123) 499-2386
V23	Cortez	877-324-7562
V24	Demetra	677-176-6621
V25	Grant	1-725-723-4374
V26	Karen	174-876-7100

Result Grid | Filter Rows: [] Edit: [] Export/Import: [] Wrap Cell Content: []

	id_empleado_veterinario	nombre_empleado_entrenador	telefono_empleado_veterinario
V28	Sammie	1-784-839-1077	
V29	Long	880-482-5834	
V3	Keisha	(217) 760-9888	
V30	Iris	1-045-145-9971	
V31	Misha	081-660-0216	
V32	Thuy	1-001-452-6752	
V33	Sharen	426-475-4089	
V34	Claretha	1-055-738-8739	
V35	Ozell	519-930-5337	
V36	Mack	1-156-206-3995	
V37	Joey	(014) 559-3299	
V38	Willie	616-367-8405	
V39	Bernie	(073) 498-5600	
V4	Irving	223-973-7731	
V40	Betsey	503-218-5961	
V41	Emilia	(070) 004-0636	
V42	Wonda	156-489-0803	
V43	Jermaine	1-743-717-5529	
V44	Edmundo	(101) 600-6697	
V45	Boris	1-590-251-6091	
V46	Denis	600-080-4832	
V47	Mari, Denis	1-388-716-5545	
V48	Dewey	1-688-941-6648	
V49	Audry	070-628-5081	
V5	Melvina	1-180-527-4385	
V6	Dominga	487-832-3937	
V7	Bradford	(142) 749-7486	
V8	Colleen	855-266-4534	
V9	Gertude	178-659-7526	

Result Grid | Filter Rows: [] Edit: [] Export/Import: [] Wrap Cell Content: []

	id_empleado_veterinario	nombre_empleado_entrenador	telefono_empleado_veterinario
V38	Willie	616-367-8405	
V39	Bernie	(073) 498-5600	
V4	Irving	223-973-7731	
V40	Betsey	503-218-5961	
V41	Emilia	(070) 004-0636	
V42	Wonda	156-489-0803	
V43	Jermaine	1-743-717-5529	
V44	Edmundo	(101) 600-6697	
V45	Boris	1-590-251-6091	
V46	Denis	600-080-4832	
V47	Mari, Denis	1-388-716-5545	
V48	Dewey	1-688-941-6648	
V49	Audry	070-628-5081	
V5	Melvina	1-180-527-4385	
V6	Dominga	487-832-3937	
V7	Bradford	(142) 749-7486	
V8	Colleen	855-266-4534	
V9	Gertude	178-659-7526	

Tabla insertarEntrenador, su función creada y los registros:

Este método que creé es un método que inserta 50 registros en una tabla llamada "empleado_entrenador" en una base de datos MySQL. Cada registro representa un empleado entrenador y tiene tres columnas: "id_empleado_entrenador", "nombre_empleado_entrenador" y "telefono_empleado_entrenador". El método utiliza la librería Faker para generar un nombre aleatorio y un número de teléfono falso para cada empleado entrenador.

El método utiliza un bucle "for" para iterar 50 veces y en cada iteración crea un objeto "Empleo_entrenador" y le asigna un ID único que es una cadena compuesta por "E" y un número de serie del 0 al 50. A continuación, se establecen el nombre y el teléfono del empleado entrenador con los valores generados por la librería Faker.

A continuación, el método utiliza un objeto "mySqlOperation" para insertar el registro en la tabla "empleado_entrenador" en la base de datos. Utiliza el método "setSqlStatement" para establecer una cadena SQL que inserta los valores de las tres columnas en la tabla. Luego, utiliza el método "executeSqlStatementVoid" para ejecutar la consulta SQL sin esperar ningún valor de retorno.

```

no usages
private static void insertarEntrenador(){
    for (int i=0 ; i<50 ; i++){
        Empleado_entrenador e= new Empleado_entrenador();
        e.setId_empleado_entrenador(""+i);
        e.setNombre_empleado_entrenador(faker.name().firstName());
        e.setTelefono_empleado_entrenador(faker.phoneNumber().cellPhone().toString());

        mySqlOperation.setSqlStatement("insert into empleado_entrenador values ('"+e.getId_empleado_entrenador()+"','"+e.getNombre_empleado_entrenador()+"')");
        mySqlOperation.executeSqlStatementVoid();
    }
}

```

_entrenador

	id_empleado_entrenador	nombre_empleado_entrenador	telefono_empleado_entrenador
>	E0	Arlen	(827) 361-4804
	E1	Lynne	1-789-068-8576
	E10	Nick	(577) 062-8605
	E11	Reda	(846) 168-7668
	E12	Elliott	001.988.8865
	E13	Mervin	(432) 422-9967
	E14	Laci	(580) 393-4460
	E15	Tracey	(379) 147-8367
	E16	Amada	(992) 640-3968
	E17	Bud	273.229.7911
	E18	Alphonso	759.169.1518
	E19	Burt	1-943-194-0633
	E2	Suk	1-681-370-1074
	E20	Kiana	925.797.6384
	E21	Donovan	604.562.4545
	E22	Willan	(239) 367-1511
	E23	Carita	597-447-6047
	E24	Grazyna	108.803.2749
	E25	Dalton	(180) 526-1353
	E26	Elvie	411-837-8194

_entrenador_1

	id_empleado_entrenador	nombre_empleado_entrenador	telefono_empleado_entrenador
>	E28	Raymond	1-574-422-5724
	E29	Dirk	1-732-280-5474
	E3	Myrtice	1-309-602-2082
	E30	Darrel	454-399-4905
	E31	Flavia	557.556.8954
	E32	Heath	(120) 965-9456
	E33	Chong	624-851-6709
	E34	Lorraine	184-354-7836
	E35	Ahmad	(835) 823-3676
	E36	Faustino	845.355.0320
	E37	Kris	064.756.0714
	E38	Wm	522.538.8018
	E39	Myrtie	845.600.1761
	E4	Chadwick	823.615.1008
	E40	Julian	1-780-881-931
	E41	Jason	568.221.4617
	E42	Estta	(516) 000-1802
	E43	Jarrett	(457) 796-4610
	E44	Guy	1-102-166-3908
	E45	Krysta	355-983-9392

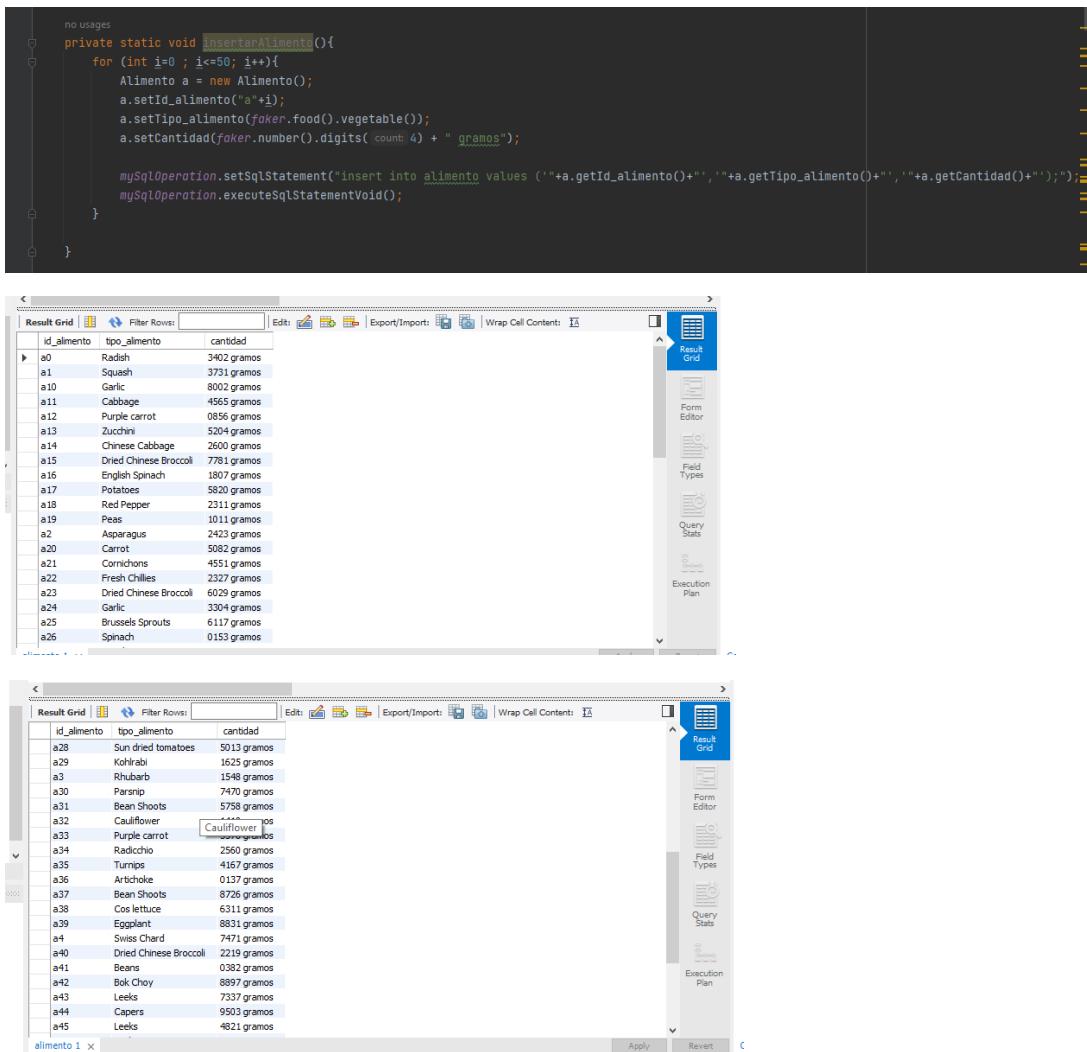
_entrenador_1

	id_empleado_entrenador	nombre_empleado_entrenador	telefono_empleado_entrenador
>	E38	Wm	522.538.8018
	E39	Myrtie	845.600.1761
	E4	Chadwick	823.615.1008
	E40	Julian	1-780-881-931
	E41	Jason	568.221.4617
	E42	Estta	(516) 000-1802
	E43	Jarrett	(457) 796-4610
	E44	Guy	1-102-166-3908
	E45	Krysta	355-983-9392
	E46	Launa	(438) 996-7776
	E47	Abe	1-066-966-3681
	E48	Mike	210.144.5809
	E49	Racquel	1-618-835-9413
	E5	Long	(203) 489-2204
	E50	Deetta	1-397-993-8724
	E6	Sam	(157) 100-2720
	E7	Duane	627.648-4520
	E8	Valentine	(657) 382-1876
	E9	Abraham	999.699.4455
	E10	Willis	999.699.4455

Tabla insertarAlimento, su función creada y los registros:

Este método que creé , es un método que inserta 50 registros en una tabla llamada "alimento" en una base de datos MySQL. Cada registro representa un alimento y tiene tres columnas: "id_alimento", "tipo_alimento" y "cantidad". El método utiliza la librería Faker para generar un tipo de alimento aleatorio y una cantidad aleatoria en gramos para cada alimento.

El método utiliza un bucle "for" para iterar 50 veces y en cada iteración crea un objeto "Alimento" y le asigna un ID único que es una cadena compuesta por "a" y un número de serie del 0 al 50. A continuación, se establecen el tipo de alimento y la cantidad del alimento con los valores generados por la librería Faker.



The screenshot shows three separate Result Grid windows from MySQL Workbench. Each window displays a table with three columns: id_alimento, tipo_alimento, and cantidad. The data is as follows:

id_alimento	tipo_alimento	cantidad
a0	Radish	3402 gramos
a1	Squash	3731 gramos
a10	Garlic	8002 gramos
a11	Cabbage	4565 gramos
a12	Purple carrot	0856 gramos
a13	Zucchini	5204 gramos
a14	Chinese Cabbage	2600 gramos
a15	Dried Chinese Broccoli	7781 gramos
a16	English Spinach	1807 gramos
a17	Potatoed	5820 gramos
a18	Red Pepper	2311 gramos
a19	Pear	1011 gramos
a2	Asparagus	2423 gramos
a20	Carrot	5082 gramos
a21	Cornichons	4551 gramos
a22	Fresh Chilles	2327 gramos
a23	Dried Chinese Broccoli	6029 gramos
a24	Garlic	3304 gramos
a25	Brussels Sprouts	6117 gramos
a26	Spinach	0153 gramos

id_alimento	tipo_alimento	cantidad
a28	Sun dried tomatoes	5013 gramos
a29	Kohlrabi	1625 gramos
a3	Rhubarb	1548 gramos
a30	Parsnip	7470 gramos
a31	Bean Shoots	5758 gramos
a32	Cauliflower	2446 gramos
a33	Purple carrot	0556 gramos
a34	Radicchio	2560 gramos
a35	Turnip	4167 gramos
a36	Artichoke	0137 gramos
a37	Bean Shoots	8726 gramos
a38	Cos lettuce	6311 gramos
a39	Eggplant	8831 gramos
a4	Swiss Chard	7471 gramos
a40	Dried Chinese Broccoli	2219 gramos
a41	Beans	0392 gramos
a42	Bok Choy	8897 gramos
a43	Leeks	7337 gramos
a44	Capers	9503 gramos
a45	Leeks	4821 gramos

id_alimento	tipo_alimento	cantidad
a46	Broccoli	1000 gramos
a47	Onion	1000 gramos
a48	Carrot	1000 gramos
a49	Potato	1000 gramos
a50	Cabbage	1000 gramos

A screenshot of a MySQL Workbench interface showing a result grid. The grid has three columns: 'id_alimento', 'tipo_alimento', and 'cantidad'. The data includes various food items like Cos lettuce, Eggplant, Swiss Chard, etc., with their respective IDs and quantities.

<code>id_alimento</code>	<code>tipo_alimento</code>	<code>cantidad</code>
a38	Cos lettuce	6311 gramos
a39	Eggplant	8831 gramos
a4	Swiss Chard	7471 gramos
a40	Dried Chinese Broccoli	2219 gramos
a41	Beans	0382 gramos
a42	Bok Choy	8897 gramos
a43	Leeks	7337 gramos
a44	Capers	9503 gramos
a45	Leeks	4821 gramos
a46	Garlic	6758 gramos
a47	Kale	4504 gramos
a48	Peppers	2715 gramos
a49	Artichoke	2516 gramos
a5	Cornichons	3617 gramos
a50	Broccoli	0883 gramos
a6	Okra	4250 gramos
a7	Capers	2675 gramos
a8	Cornichons	2920 gramos
a9	Fresh Chillies	4044 gramos
• nulls	nulls	nulls

Tabla insertarDieta, su función creada y los registros:

Este método que creé , es un método que inserta 50 registros en una tabla llamada "dieta" en una base de datos MySQL. Cada registro representa una dieta y tiene tres columnas: "id_dieta", "tipo_dieta" y "dosis". El método utiliza la librería Faker para generar un tipo de dieta aleatorio y una dosis aleatoria en gramos para cada dieta.

El método utiliza un bucle "for" para iterar 50 veces y en cada iteración crea un objeto "Dieta" y le asigna un ID único que es una cadena compuesta por "D" y un número de serie del 0 al 50. A continuación, se establecen el tipo de dieta y la dosis de la dieta con los valores generados por la librería Faker.

```

no usages
private static void insertarDieta(){
    for (int i=0; i<=50 ; i++){
        Dieta d= new Dieta();
        d.setId_dieta("D"+i);
        d.setTipo_dieta(faker.food().dish());
        d.setDosis(faker.number().digits( count: 3 ) + " gramos");

        mySqlOperation.setSqlStatement("insert into dieta values ('"+d.getId_dieta()+"','"+d.getTipo_dieta()+"','"+d.getDosis()+"');");
        mySqlOperation.executeSqlStatementVoid();
    }
}

```

<code>id_dieta</code>	<code>tipo_dieta</code>	<code>dosis</code>
D0	Som Tam	736 gramos
D1	Tacos	244 gramos
D10	Pettuccine Alfredo	481 gramos
D11	Pettuccine Alfredo	492 gramos
D12	Souvlaki	768 gramos
D13	Chicken Parm	337 gramos
D14	Pork Sausage Roll	830 gramos
D15	Meatballs with Sauce	567 gramos
D16	Tiramisú	579 gramos
D17	Chicken Wings	050 gramos
D18	Vegetable Soup	313 gramos
D19	Massaman Curry	336 gramos
D2	Souvlaki	093 gramos
D20	Pho	370 gramos
D21	Massaman Curry	700 gramos
D22	Massaman Curry	419 gramos
D23	Katsu Curry	071 gramos
D24	Tacos	137 gramos
D25	Pierogi	262 gramos
D26	Tiramisú	721 gramos

The screenshot shows two separate result grids from MySQL Workbench, both titled 'dieta 1'. Each grid displays the same 50 rows of data from the 'dieta' table. The columns are 'id_dieta', 'tipo_dieta', and 'dosis'. The data includes various meal types like Pizza, Vegetable Soup, and Fish and Chips, along with their corresponding doses in grams.

id_dieta	tipo_dieta	dosis
D28	Pizza	085 gramos
D29	Vegetable Soup	737 gramos
D3	Fettuccine Alfredo	788 gramos
D30	Pasta Carbonara	550 gramos
D31	Fish and Chips	193 gramos
D32	Lasagne	776 gramos
D33	Fish and Chips	422 gramos
D34	Pork Belly Buns	845 gramos
D35	Barbecue Ribs	036 gramos
D36	Ricotta Stuffed Ravioli	535 gramos
D37	Souvlaki	375 gramos
D38	Chicken Fajitas	856 gramos
D39	Fish and Chips	743 gramos
D4	Risotto with Seafood	217 gramos
D40	Ricotta Stuffed Ravioli	107 gramos
D41	Lasagne	068 gramos
D42	Chilli con Carne	687 gramos
D43	Caesar Salad	116 gramos
D44	Teriyaki Chicken Do...	591 gramos
D45	Massaman Curry	110 gramos
D46	Lasagne	133 gramos
D47	Poutine	718 gramos
D48	California Maki	669 gramos
D49	Lasagne	526 gramos
D5	Ebiten maki	704 gramos
D50	Ricotta Stuffed Ravioli	837 gramos
D6	Chilli con Carne	207 gramos
D7	Pasta Carbonara	336 gramos
D8	Pasta Carbonara	699 gramos
D9	Fish and Chips	247 gramos

Tabla insertar_dieta_alimento, su función creada y los registros:

Este método que creé , es un método que inserta registros en una tabla de relación "dieta_alimento" en una base de datos MySQL. Esta tabla de relación se utiliza para relacionar las tablas "dieta" y "alimento" utilizando sus identificadores únicos (id_dieta e id_alimento respectivamente).

El método utiliza un bucle "for" para iterar 50 veces y en cada iteración crea dos variables de cadena llamadas "id_dieta_d_a" e "id_alimento_d_a". El valor de estas variables se asigna concatenando las cadenas "D" o "a" respectivamente con el número de serie del 0 al 50.

```
no usages
private static void insertar_dieta_alimento() {

    for (int i=0; i<=50 ;i++) {
        String id_dieta_d_a= ("D"+i);
        String id_alimento_d_a= ("a"+i);
        mySqlOperation.setSqlStatement("insert into dieta_alimento values('"+id_dieta_d_a+"','"+id_alimento_d_a+"');");
        mySqlOperation.executeSqlStatementVoid();
    }
}
```

The image displays three separate MySQL Workbench result grids side-by-side. Each grid has a header row with columns 'id_dieta_d_a' and 'id_alimento_d_a'. The first grid, labeled 'eta_alimento 1', contains 19 rows from D0 to D19. The second grid, also labeled 'eta_alimento 1', contains 40 rows from D21 to D60. The third grid, labeled 'eta_alimento 2', contains 50 rows from D31 to D80. All grids are set to 'Read Only' mode.

id_dieta_d_a	id_alimento_d_a
D0	a0
D1	a1
D2	a2
D3	a3
D4	a4
D5	a5
D6	a6
D7	a7
D8	a8
D9	a9
D10	a10
D11	a11
D12	a12
D13	a13
D14	a14
D15	a15
D16	a16
D17	a17
D18	a18
D19	a19

id_dieta_d_a	id_alimento_d_a
D21	a21
D22	a22
D23	a23
D24	a24
D25	a25
D26	a26
D27	a27
D28	a28
D29	a29
D30	a30
D31	a31
D32	a32
D33	a33
D34	a34
D35	a35
D36	a36
D37	a37
D38	a38
D39	a39
D40	a40

id_dieta_d_a	id_alimento_d_a
D31	a31
D32	a32
D33	a33
D34	a34
D35	a35
D36	a36
D37	a37
D38	a38
D39	a39
D40	a40
D41	a41
D42	a42
D43	a43
D44	a44
D45	a45
D46	a46
D47	a47
D48	a48
D49	a49
D50	a50

Tabla insertar_logistico, su función creada y los registros:

Este método que creé, es un método que inserta 50 registros en una tabla llamada "empleado_logistico" en una base de datos MySQL. Cada registro representa un empleado logístico y tiene tres columnas: "id_empleado_logistico", "nombre_empleado_logistico" y "telefono".

El método utiliza un bucle "for" para iterar 50 veces y en cada iteración crea un objeto "Empleado_logistico" y le asigna un ID único que es una cadena compuesta por "L" y un número de serie del 0 al 50. A continuación, se establecen el nombre y el número de teléfono del empleado logístico con los valores generados por la librería Faker.

```
sages
vate static void insertar_logistica(){
for (int i=0; i<50; i++){
    Empleado_logistico l= new Empleado_logistico();
    l.setId_empleado_logistico("+"+i);
    l.setNombre_empleado_logistico(faker.name().name());
    l.setTelefono(faker.phoneNumber().phoneNumber());
    mySqlOperation.setSqlStatement("insert into empleado_logistico values (""+l.getId_empleado_logistico()+
        "", "+l.getNombre_empleado_logistico()+"', '"+l.getTelefono()+"');");
    mySqlOperation.executeSqlStatementVoid();
}
}
```

Result Grid | Filter Rows: Edit: Export/Import: Wrap Cell Content:

id_empleado_logistco	nombre_empleado_logistco	telefono
L0	Jerald Jaskolski	(847) 589-4996
L1	Jong Renner DVM	823-368-9051
L10	Mitch Mayer	(507) 367-1982 x3466
L11	Al Bogisch	035.734.1046
L12	Brunilda Reigner	085-909-5459 x95889
L13	Giselle Dickens	(650) 614-0099
L14	Ms. Nativedad Nader	281.638-1952
L15	Lachelle Schnier	656.590.6763 x303
L16	Mard Toy	727.259.3061
L17	Oscar Corwin	1-963-739-7676
L18	Inez Daniel	377-800-6976 x2294
L19	Miss Jeannett Bradtke	587.950.2893 x33329
L2	Dion Koepf	1-996-640-9484
L20	Mrs. Jarrett Rohan	344-284-6919 x182
L21	Ms. Lonna Schuster	1-986-348-7049
L22	Socorro Didkson	1-517-121-5849 x6374
L23	Mamrie Mohr	260-802-2307
L24	Lavonne Luelwitz III	673.769.7582 x286
L25	Antoinette Legros	776-776-2557
L26	Kyle Wilkinson DDS	1-247-835-2056 x49786

sdo_logistco 1

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

id_empleado_logistico	nombre_empleado_logistico	telefono
L28	Anita Rath	527-3553 x63029
L29	Scott Ryan	1-688-064-0815
L3	Dorsey Runolfsson	1-073-696-3641 x957
L30	Gabriel Beer	309-610-8993
L31	Elnore Leonann	1-395-567-9371
L32	Carlee VonRueden	087-776-6527
L33	Marge Little	190.385.2336
L34	Jerrie Leffler	(031) 506-3917 x974
L36	Major Jerde	499.598.1112
L37	Sanda Tremblay DDS	040.188.7668
L38	Donald Walter	361-162-6721 x388
L39	Cletus Crist	374-508-3335
L4	Ray Tremblay	686-162-0533 x85838
L40	Vennie Brakus	1-490-140-8951
L41	Ignacio Schumm DVM	1-689-350-2076
L42	Mr. Kristopher Kuvalis	1-208-734-3263
L43	Reed Jast	055-514-3552
L44	Betty Satterfield	(094) 574-4915 x32112
L45	Abel Bailey	874-650-8280 x24285
L46	Adan Lakin V	168-646-6803 x921

undo logistico 1 x | Apply | Revert |

Result Grid | Form Editor | Field Types | Query Stats | Execution Plan

Result Grid | Filter Rows: Edit: Export/Import: Wrap Cell Content:

id_empleado_logistico	nombre_empleado_logistico	telefono
L38	Donald Walter	361-162-6721 x388
L39	Cletus Crist	374-508-3335
L4	Ray Tremblay	686-162-0533 x85838
L40	Vennie Braku	1-490-140-8951
L41	Ignacio Schumm DVM	1-689-350-2076
L42	Mr. Kristopher Kuvalis	1-208-734-3263
L43	Reed Jast	055-514-3552
L44	Betty Satterfield	(094) 574-4915 x32112
L45	Abel Baley	874-650-8280 x24285
L46	Adan Lakin V	168-446-6803 x921
L47	Harland Kemmer	(420) 926-5099
L48	Audry Cruickshank	704-182-3980
L49	Marvin Batz MD	(775) 586-5707 x8445
L5	Chi Champlin DDS	(945) 654-4997 x80430
L50	Keenan Hoeger	1-938-514-7139 x52502
L6	Joel Ward	(001) 788-5882
L7	Elie Friesen	1-137-563-8333
L8	Tasha Declow	1-057-480-2015
L9	Tonie Sauer	539.060.4863
L10	Wanda	0000000000

Result Grid | Form Editor | Field Types | Query Stats | Execution Plan

Tabla insertarProveedor, su función creada y los registros:

Este método que creé , es un método que inserta registros en una tabla llamada "proveedor" en una base de datos MySQL. Cada registro representa un proveedor y tiene tres columnas: "id_proveedor", "nombre_proveedor" y "direccion".

El método utiliza un bucle "for" para iterar 50 veces y en cada iteración crea un objeto "Proveedor" y le asigna un ID único que es una cadena compuesta por "P" y un número de serie del 0 al 50. A continuación, se establecen el nombre y la dirección del proveedor con los valores generados por la librería Faker.

```
no usages
private static void insertarProveedor(){
    for (int i=0; i<=50 ;i++){
        Proveedor p = new Proveedor();
        p.setId_proveedor("P"+i);
        p.setNombre_proveedor(faker.name().name());
        p.setDireccion(faker.address().firstName());
        mySqlOperation.setSqlStatement("insert into proveedor_values ('"+p.getId_proveedor()+
        "','" +p.getNombre_proveedor()+"','"+p.getDireccion()+"');");
        mySqlOperation.executeSqlStatementVoid();
    }
}
```

Result Grid		
id_proveedor	nombre_proveedor	direccion
P0	Lejana Williamson III	Vernice
P1	Dr. Whitney Leach	Renda
P10	Rich Swanson III	Amparo
P11	Palma Bins	Lauren
P12	Fausina Kautzer	Antoine
P13	Miss Constance Kozey	Libby
P14	Mitchel Pfeifferst	Damick
P15	Arnold Jacobi	Loretta
P16	Nick Reichel	Caroline
P18	Julieta Grady	Shona
P19	Tony Greenfelder	Carrol
P2	Lona Paucke	Chere
P20	Desirae Ryan	Gabriel
P21	Brittney McLaughlin	Mitch
P22	Miss Dorsey Armstrong	Renee
P23	Grover Jacobs	Oleta
P24	Betsy Carter I	Dusty
P25	Toya Hauck	Norene
P26	Kathryne Smith DDS	Harley
P28	Sherna Upton	Clyde

Result Grid		
id_proveedor	nombre_proveedor	direccion
P32	Angel Prohaska DVM	Elmer
P33	Antoine Kling	Jeffry
P35	Janice Towne	Victor
P36	Niki Ziemann	Jerie
P37	Dr. Margene Stark	Bernardo
P38	Laraine Barrows	Charley
P39	Miss Gibson	Gaylene
P4	Unwood Schinner	Dorotha
P40	Lil Labadie Sr.	Tara
P41	Lee Watsica	Vernon
P42	Tobie Denesik	Neely
P43	Dr. Alessandra Funk	Luciano
P44	Terry Stokes	Burl
P45	Sari Dickens	Lazaro
P46	Mathew Hilpert	Tula
P47	Evan Bradtke	Shauna
P48	Guillemina Lynch	Thomasina
P49	Laura Moore	Amiee
P5	Millard Radke	Frederick
P50	Ernestine Larkin	Jed

Tabla insertarFactura, su función creada y los registros:

Este método que creé, es un método llamado "insertarFactura()" que genera y luego inserta 50 objetos de tipo "Factura" en una base de datos MySQL. Cada objeto de tipo Factura tiene un ID de factura, una cantidad de productos, una cantidad total y un ID de proveedor.

Así fue como construí las líneas del código:

En la primera línea del método, se utiliza un bucle "for" para crear 50 objetos "Factura" y luego insertarlos en la base de datos.

En el segundo línea del bucle, se crea un nuevo objeto "Factura" y se almacena en una variable llamada "f".

En la tercera línea del bucle, se establece el ID de factura del objeto "Factura" como "f" seguido de un número de iteración, es decir, "f0", "f1", "f2", etc.

En la cuarta línea del bucle, se establece la cantidad de productos de la factura como un número de 3 dígitos generado por el objeto "faker".

En la quinta línea del bucle, se establece la cantidad total de la factura como un número aleatorio generado por el objeto "faker" dentro de un rango específico.

En la sexta línea del bucle, se establece el ID de proveedor de la factura como "P" seguido de un número de iteración, es decir, "P0", "P1", "P2", etc.

En la séptima línea del bucle, se construye una sentencia SQL para insertar la factura en la base de datos utilizando los valores de los atributos de la factura.

En la octava línea del bucle, se ejecuta la sentencia SQL utilizando un objeto "mySqlOperation" que se encarga de conectarse a la base de datos y ejecutar la sentencia. La sentencia no devuelve resultados, por lo que se utiliza el método "executeSqlStatementVoid()".

```

no usages
private static void insertarFactura(){

    for (int i=0; i<=50 ;i++){
        Factura f= new Factura();
        f.setId_factura("f"+i);
        f.setCantidad_productos(faker.number().digits( count: 3));
        f.setCantidad_total(String.valueOf(faker.number().numberBetween(1000000, 2500000)));
        f.setId_proveedor_f("P"+i);
        mySqlOperation.setSqlStatement("insert into factura values ('"+f.getId_factura()+
            "','" +f.getCantidad_productos()+"','"+f.getCantidad_total()+"','"+f.getId_proveedor_f()+"');");
        mySqlOperation.executeSqlStatementVoid();
    }
}

```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

	id_factura	cantidad_total	cantidad_productos	id_proveedor_f
f0	405	7642219	P0	
f1	486	10955940		7642219
f10	827	4566567	P10	
f11	675	12375047	P11	
f12	213	13476278	P12	
f13	675	17693147	P13	
f14	357	13186161	P14	
f15	933	19695036	P15	
f16	883	21759840	P16	
f18	382	22468477	P18	
f19	391	4908976	P19	
f2	550	14367488	P2	
f20	212	8341437	P20	
f21	563	11977170	P21	
f22	151	8479825	P22	
f23	933	11497874	P23	
f24	980	17273255	P24	
f25	767	19414792	P25	
f26	141	18565503	P26	
f28	679	12291545	P28	

factura 1 | Apply | Revert | Cc

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

	id_factura	cantidad_total	cantidad_productos	id_proveedor_f
f3	152	4130256		P3
f30	310	18271123		P30
f31	438	7643317		P31
f32	695	16578775		P32
f33	958	11074169		P33
f35	113	6227964		P35
f36	015	3335511		P36
f37	928	14326875		P37
f38	653	20901900		P38
f39	508	9545688		P39
f4	772	12857905		P4
f40	085	12162185		P40
f41	751	13652849		P41
f42	388	23105456		P42
f43	845	6687341		P43
f44	802	9142179		P44
f45	007	16227731		P45
f46	744	2179680		P46
f47	346	8975008		P47
f48	256	3760037		P48

factura 1 | Apply | Revert | Cc

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

	id_factura	cantidad_total	cantidad_productos	id_proveedor_f
f38	653	20901900		P38
f39	508	9545688		P39
f4	772	12857905		P4
f40	085	12162185		P40
f41	751	13652849		P41
f42	388	23105456		P42
f43	845	6687341		P43
f44	802	9142179		P44
f45	007	16227731		P45
f46	744	2179680		P46
f47	346	8975008		P47
f48	256	3760037		P48
f49	411	5905118		P49
f5	897	15487505		P5
f50	653	14111455		P50
f6	787	13481758		P6
f7	166	21026612		P7
f8	861	9817303		P8
f9	555	17422067		P9
*	NULL	NULL	NULL	NULL

factura 1 | Apply | Revert | Cc

Tabla insertarOrdenCompra, su función creada y los registros:

Este método que creé, es un método llamado "insertarOrdenCompra()" que genera y luego inserta 50 objetos de tipo "OrdenCompra" en una base de datos MySQL. Cada objeto de tipo OrdenCompra tiene un ID de orden, una descripción, una fecha de elaboración, una fecha de entrega, un ID de empleado logístico y un ID de proveedor.

Aquí hay una explicación de las líneas del código:

En la primera línea del método, se utiliza un bucle "for" para crear 50 objetos "OrdenCompra" y luego insertarlos en la base de datos.

En la segunda línea del bucle, se crea un nuevo objeto "OrdenCompra" y se almacena en una variable llamada "o".

En la tercera línea del bucle, se establece el ID de orden del objeto "OrdenCompra" como "O-" seguido de un número de iteración, es decir, "O-0", "O-1", "O-2", etc.

En la cuarta línea del bucle, se establece la descripción de la orden como un tipo de tarjeta de crédito generado por el objeto "faker".

En la quinta línea del bucle, se establece la fecha de elaboración de la orden como una fecha aleatoria generada por el objeto "faker" dentro de un rango específico.

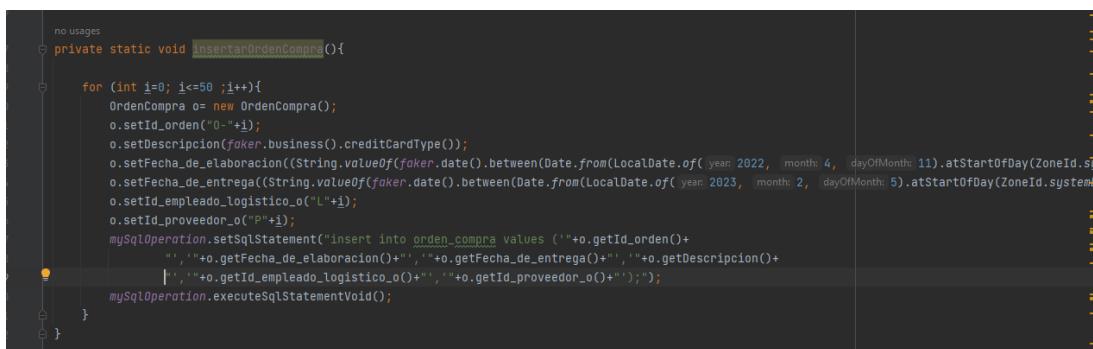
En la sexta línea del bucle, se establece la fecha de entrega de la orden como una fecha aleatoria generada por el objeto "faker" dentro de un rango específico.

En la séptima línea del bucle, se establece el ID de empleado logístico de la orden como "L" seguido de un número de iteración, es decir, "L0", "L1", "L2", etc.

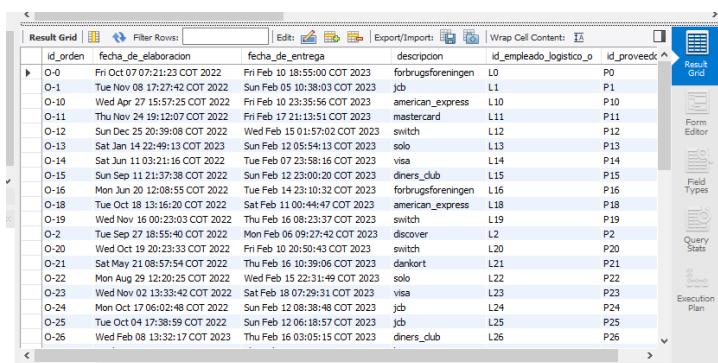
En la octava línea del bucle, se establece el ID de proveedor de la orden como "P" seguido de un número de iteración, es decir, "P0", "P1", "P2", etc.

En la novena línea del bucle, se construye una sentencia SQL para insertar la orden en la base de datos utilizando los valores de los atributos de la orden.

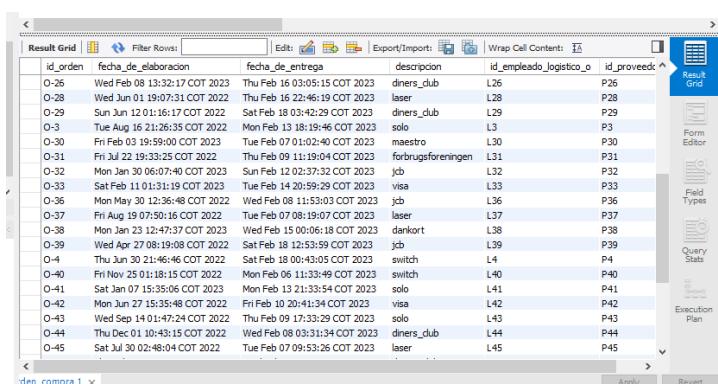
En la décima línea del bucle, se ejecuta la sentencia SQL utilizando un objeto "mySqlOperation" que se encarga de conectarse a la base de datos y ejecutar la sentencia. La sentencia no devuelve resultados, por lo que se utiliza el método "executeSqlStatementVoid()".



```
private static void insertarOrdenCompra(){  
    for (int i=0; i<=50 ;i++){  
        OrdenCompra o = new OrdenCompra();  
        o.setId_orden("O-"+i);  
        o.setDescripcion(faker.business().creditCardType());  
        o.setFecha_de_elaboracion((String.valueOf(faker.date().between(Date.from(LocalDate.of( year: 2022, month: 4, dayOfMonth: 11).atStartOfDay(ZoneId.systemDefault()), LocalDate.of( year: 2023, month: 2, dayOfMonth: 5).atStartOfDay(ZoneId.systemDefault()))).format(DateTimeFormatter.ofPattern("yyyy-MM-dd"))));  
        o.setId_empleado_logistico_o("L"+i);  
        o.setId_proveedor_o("P"+i);  
        mySqlOperation.setSqlStatement("insert into orden_compra values ('"+o.getId_orden()+"  
       ','"+o.getFecha_de_elaboracion()+"','"+o.getFecha_de_entrega()+"','"+o.getDescripcion()+"  
        |','"+o.getId_empleado_logistico_o()+"','"+o.getId_proveedor_o()+"');");  
        mySqlOperation.executeSqlStatementVoid();  
    }  
}
```



id_orden	fecha_de_elaboracion	fecha_de_entrega	descripcion	id_empleado_logistico_o	id_proveedor_o
O-0	Fri Oct 07 07:21:23 COT 2022	Fri Feb 10 18:55:00 COT 2023	forbrugsforsningen	L0	P0
O-1	Tue Nov 08 17:27:42 COT 2022	Sun Feb 05 10:38:03 COT 2023	jcb	L1	P1
O-10	Wed Apr 27 15:57:25 COT 2022	Fri Feb 10 23:35:56 COT 2023	american_express	L10	P10
O-11	Thu Nov 24 19:12:07 COT 2022	Fri Feb 17 21:13:51 COT 2023	mastercard	L11	P11
O-12	Sun Dec 25 20:39:08 COT 2022	Wed Feb 15 01:57:02 COT 2023	switch	L12	P12
O-13	Sat Jan 14 22:49:13 COT 2023	Sun Feb 12 05:54:13 COT 2023	solo	L13	P13
O-14	Sat Jun 11 03:21:16 COT 2022	Tue Feb 07 23:56:16 COT 2023	visa	L14	P14
O-15	Sun Sep 11 21:37:38 COT 2022	Sun Feb 12 23:06:20 COT 2023	diners_club	L15	P15
O-16	Mon Jun 20 12:08:55 COT 2022	Tue Feb 14 23:10:32 COT 2023	forbrugsforsningen	L16	P16
O-18	Tue Oct 18 13:16:23 COT 2022	Sat Feb 11 00:44:47 COT 2023	american_express	L18	P18
O-19	Wed Nov 16 00:23:03 COT 2022	Thu Feb 16 08:23:37 COT 2023	switch	L19	P19
O-2	Tue Sep 27 18:55:00 COT 2022	Mon Feb 06 09:27:42 COT 2023	discover	L2	P2
O-20	Wed Oct 19 20:23:33 COT 2022	Fri Feb 10 20:50:43 COT 2023	switch	L20	P20
O-21	Sat May 21 08:57:54 COT 2022	Thu Feb 16 10:39:06 COT 2023	dankort	L21	P21
O-22	Mon Aug 29 12:20:25 COT 2022	Wed Feb 15 22:31:49 COT 2023	solo	L22	P22
O-23	Wed Nov 02 13:33:42 COT 2022	Sat Feb 18 07:29:31 COT 2023	visa	L23	P23
O-24	Mon Oct 17 06:02:48 COT 2022	Sun Feb 12 08:38:48 COT 2023	jcb	L24	P24
O-25	Tue Oct 04 17:38:59 COT 2022	Sun Feb 12 06:18:57 COT 2023	jcb	L25	P25
O-26	Wed Feb 08 13:32:17 COT 2023	Thu Feb 16 03:05:15 COT 2023	diners_club	L26	P26



id_orden	fecha_de_elaboracion	fecha_de_entrega	descripcion	id_empleado_logistico_o	id_proveedor_o
O-26	Wed Feb 08 13:32:17 COT 2023	Thu Feb 16 03:05:15 COT 2023	diners_club	L26	P26
O-28	Wed Jun 01 19:07:31 COT 2022	Tue Feb 16 22:46:19 COT 2023	laser	L28	P28
O-29	Sun Jun 12 01:16:17 COT 2022	Sat Feb 18 03:42:29 COT 2023	diners_club	L29	P29
O-3	Tue Aug 16 21:26:35 COT 2022	Mon Feb 13 18:19:46 COT 2023	solo	L3	P3
O-30	Fri Feb 03 19:59:00 COT 2022	Fri Feb 07 01:02:40 COT 2023	maestro	L30	P30
O-31	Fri Jul 22 19:33:25 COT 2022	Thu Feb 09 11:19:04 COT 2023	forbrugsforsningen	L31	P31
O-32	Mon Jan 30 06:07:49 COT 2023	Sun Feb 12 02:37:32 COT 2023	jcb	L32	P32
O-33	Sat Feb 11 01:31:19 COT 2023	Thu Feb 14 20:59:29 COT 2023	visa	L33	P33
O-36	Mon May 30 12:36:48 COT 2022	Wed Feb 08 11:53:03 COT 2023	jcb	L36	P36
O-37	Fri Aug 19 07:50:16 COT 2022	Tue Feb 07 08:19:07 COT 2023	laser	L37	P37
O-38	Mon Jan 23 12:47:37 COT 2023	Wed Feb 15 00:06:18 COT 2023	dankort	L38	P38
O-39	Wed Apr 27 08:19:08 COT 2022	Sat Feb 18 12:53:59 COT 2023	jcb	L39	P39
O-4	Thu Jun 30 21:46:46 COT 2022	Sat Feb 18 00:43:05 COT 2023	switch	L4	P4
O-40	Fri Nov 25 01:18:15 COT 2022	Mon Feb 05 11:30:49 COT 2023	switch	L40	P40
O-41	Sat Jan 07 15:35:05 COT 2022	Mon Feb 13 21:30:54 COT 2023	solo	L41	P41
O-42	Mon Jun 27 15:35:48 COT 2022	Fri Feb 10 20:41:34 COT 2023	visa	L42	P42
O-43	Wed Sep 14:01:47:24 COT 2022	Thu Feb 09 17:33:29 COT 2023	solo	L43	P43
O-44	Thu Dec 01 10:43:15 COT 2022	Wed Feb 08 03:31:34 COT 2023	diners_club	L44	P44
O-45	Sat Jul 30 02:48:04 COT 2022	Tue Feb 07 09:53:26 COT 2023	laser	L45	P45

A screenshot of a database management system interface showing a result grid. The grid has columns: id_order, fecha_de_elaboracion, fecha_de_entrega, descripción, id_empleado_logístico, and id_proveedor. The data consists of 50 rows of order information, with the last row being a summary or footer.

id_order	fecha_de_elaboracion	fecha_de_entrega	descripción	id_empleado_logístico	id_proveedor
O-39	Wed Apr 27 08:19:08 COT 2022	Sat Feb 18 12:53:59 COT 2023	jcb	L39	P39
O-4	Thu Jun 30 21:46:46 COT 2022	Sat Feb 18 00:43:05 COT 2023	switch	L4	P4
O-40	Fri Nov 25 01:18:15 COT 2022	Mon Feb 06 11:33:49 COT 2023	switch	L40	P40
O-41	Sat Jan 07 15:35:06 COT 2023	Mon Feb 13 21:33:54 COT 2023	solo	L41	P41
O-42	Mon Jun 27 15:35:48 COT 2022	Fri Feb 10 20:41:34 COT 2023	visa	L42	P42
O-43	Wed Sep 14 01:47:24 COT 2022	Thu Feb 09 17:33:29 COT 2023	solo	L43	P43
O-44	Thu Dec 01 10:43:15 COT 2022	Wed Feb 08 03:31:34 COT 2023	diners_club	L44	P44
O-45	Sat Jul 30 02:48:04 COT 2022	Tue Feb 07 09:53:26 COT 2023	laser	L45	P45
O-46	Thu Feb 09 08:23:02 COT 2023	Wed Feb 15 16:50:43 COT 2023	diners_club	L46	P46
O-47	Thu May 26 19:53:19 COT 2023	Fri Feb 17 09:17:33 COT 2023	mastercard	L47	P47
O-48	Tue Aug 30 13:50:12 COT 2022	Mon Feb 13 05:31:22 COT 2023	maestro	L48	P48
O-49	Tue Dec 06 06:47:30 COT 2022	Tue Feb 07 05:29:47 COT 2023	dankort	L49	P49
O-5	Tue Oct 18 20:47:30 COT 2022	Sat Feb 18 14:11:52 COT 2023	switch	L5	P5
O-50	Sun May 01 15:10:51 COT 2022	Sun Feb 05 23:41:29 COT 2023	american_express	L50	P50
O-6	Wed Nov 23 08:30:00 COT 2022	Tue Feb 07 14:37:53 COT 2023	jcb	L6	P6
O-7	Mon Nov 14 17:11:24 COT 2022	Wed Feb 08 05:37:03 COT 2023	american_express	L7	P7
O-8	Thu May 12 21:59:42 COT 2022	Wed Feb 15 13:17:43 COT 2023	jcb	L8	P8
O-9	Wed Nov 30 18:33:41 COT 2022	Mon Feb 06 16:58:22 COT 2023	solo	L9	P9

Tabla insertarProveedor_alimento, su función creada y los registros:

Este es un método llamado "insertarProveedor_alimento()" que inserta 50 filas en una tabla de la base de datos llamada "proveedor_alimento". La tabla tiene dos columnas: "id_alimento" e "id_proveedor", que se utilizan para registrar la relación entre los proveedores de alimentos y los alimentos que proveen.

Aquí hay una explicación de las líneas del código que construí:

En la primera línea del método, se utiliza un bucle "for" para crear 50 parejas de IDs de alimento y proveedor y luego insertarlas en la tabla.

En la segunda línea del bucle, se crea un ID de alimento como una cadena de texto que comienza con "a" seguido de un número de iteración, es decir, "a0", "a1", "a2", etc.

En la tercera línea del bucle, se crea un ID de proveedor como una cadena de texto que comienza con "P" seguido de un número de iteración, es decir, "P0", "P1", "P2", etc.

En la cuarta línea del bucle, se construye una sentencia SQL para insertar la pareja de IDs en la tabla "proveedor_alimento" utilizando los valores de los IDs de alimento y proveedor.

En la quinta línea del bucle, se ejecuta la sentencia SQL utilizando un objeto "mySqlOperation" que se encarga de conectarse a la base de datos y ejecutar la sentencia. La sentencia no devuelve resultados, por lo que se utiliza el método "executeSqlStatementVoid()".

```
3 no usages
4 private static void insertarProveedor_alimento(){
5
6     for (int i=0; i<=50 ;i++){
7         String id_alimento_p="a"+i;
8         String id_proveedor_p="P"+i;
9
10        mySqlOperation.setSqlStatement("insert into proveedor_alimento values('"+id_alimento_p+"','"+id_proveedor_p+"');");
11        mySqlOperation.executeSqlStatementVoid();
12    }
13 }
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

a0	P0
a1	P1
a2	P2
a3	P3
a4	P4
a5	P5
a6	P6
a7	P7
a8	P8
a9	P9
a10	P10
a11	P11
a12	P12
a13	P13
a14	P14
a15	P15
a16	P16
a18	P18
a19	P19
a20	P20

dot_allimento1 x Read Only

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

a22	P22
a23	P23
a24	P24
a25	P25
a26	P26
a28	P28
a29	P29
a30	P30
a31	P31
a32	P32
a33	P33
a35	P35
a36	P36
a37	P37
a38	P38
a39	P39
a40	P40
a41	P41
a42	P42
a43	P43

dot_allimento1 x Read Only

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

a30	P30
a31	P31
a32	P32
a33	P33
a35	P35
a36	P36
a37	P37
a38	P38
a39	P39
a40	P40
a41	P41
a42	P42
a43	P43
a44	P44
a45	P45
a46	P46
a47	P47
a48	P48
a49	P49
a50	P50

dot_allimento1 x Read Only

Tabla insertarTelefono_proveedor, su función creada y los registros:

Este es un método llamado "insertarTelefono_proveedor()" que inserta 50 filas en una tabla de la base de datos llamada "telefono_proveedor". La tabla tiene dos columnas: "id_proveedor" y "numero", que se utilizan para registrar los números de teléfono de los proveedores.

Aquí hay una explicación de las líneas del código:

En la primera línea del método, se utiliza un bucle "for" para crear 50 números de teléfono y luego insertarlos en la tabla.

En la segunda línea del bucle, se crea un ID de proveedor como una cadena de texto que comienza con "P" seguido de un número de iteración, es decir, "P0", "P1", "P2", etc. Este ID se utiliza para asociar el número de teléfono con el proveedor.

En la tercera línea del bucle, se genera un número de teléfono aleatorio utilizando el objeto "faker" que se utiliza para generar datos falsos. El método "phoneNumber().cellPhone()" genera un número de teléfono móvil aleatorio.

En la cuarta línea del bucle, se construye una sentencia SQL para insertar el ID de proveedor y el número de teléfono en la tabla "telefono_proveedor" utilizando los valores generados anteriormente.

En la quinta línea del bucle, se ejecuta la sentencia SQL utilizando un objeto "mySqlOperation" que se encarga de conectarse a la base de datos y ejecutar la sentencia. La sentencia no devuelve resultados, por lo que se utiliza el método "executeSqlStatementVoid()".

```
no usages
184     private static void insertarTelefono_proveedor(){
185         for (int i=0; i<=50 ;i++){
186             String id_proveedor_telefono="P"+i;
187             String numero=faker.phoneNumber().cellPhone().toString();
188             mySqlOperation.setSqlStatement("insert into telefono_proveedor values('"+id_proveedor_telefono+"','"+numero+"');");
189             mySqlOperation.executeSqlStatementVoid();
190         }
191     }
192 }
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

o_proveedor 1

	id_proveedor	telefono	numero
P0		454.109.0880	
P1		(253) 292-3637	
P10		1-500-306-5807	
P11		1-895-398-2985	
P12		228.696.8973	
P13		437-389-5440	
P14		021-779-5013	
P15		026-332-5622	
P16		279.188.1578	
P18		1-366-721-7876	
P19		160-053-9527	
P2		1-096-037-3276	
P20		(665) 926-3026	
P21		802.002.4444	
P22		1-350-490-2634	
P23		664.203.3993	
P24		723.631.0275	
P25		1-307-119-4951	
P26		1-766-301-0411	
P28		1-653-151-2546	

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

o_proveedor 1

	id_proveedor	telefono	numero
P3		341.581.9652	
P30		941.438.9038	
P31		(620) 334-5597	
P32		1-431-703-8976	
P33		(357) 243-6786	
P35		(081) 164-5602	
P36		(246) 430-8195	
P37		1-187-798-5835	
P38		087-519-2754	
P39		(704) 289-8183	
P4		280-170-3006	
P40		447-940-2280	
P41		1-884-554-1993	
P42		727-132-0262	
P43		(569) 577-1476	
P44		882.726.4074	
P45		244.166.6950	
P46		640-335-6589	
P47		097.798.7123	
P48		012.336.6540	
P49		150-255-2896	
P5		515.047.6251	
P50		(379) 710-9089	
P6		(275) 409-1605	
P7		849-211-5690	
P8		579.639.6935	
P9		208-953-2107	

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

o_proveedor 1

	id_proveedor	telefono	numero
P38		087-519-2754	
P39		(704) 289-8183	
P4		280-170-3006	
P40		447-940-2280	
P41		1-884-554-1993	
P42		727-132-0262	
P43		(569) 577-1476	
P44		882.726.4074	
P45		244.166.6950	
P46		640-335-6589	
P47		097.798.7123	
P48		012.336.6540	
P49		150-255-2896	
P5		515.047.6251	
P50		(379) 710-9089	
P6		(275) 409-1605	
P7		849-211-5690	
P8		579.639.6935	
P9		208-953-2107	

Tabla insertarVeterinarioDieta, su función creada y los registros:

Este es un método llamado "insertarVeterinarioDieta()" que inserta 50 filas en una tabla de la base de datos llamada "veterinario_dieta". La tabla tiene dos columnas: "id_empleado_veterinario" e "id_dieta", que se utilizan para registrar los veterinarios y las dietas que se asignan a los animales.

Aquí hay una explicación de las líneas del código:

En la primera línea del método, se utiliza un bucle "for" para crear 50 registros y luego insertarlos en la tabla.

En la segunda línea del bucle, se crea un ID de empleado veterinario como una cadena de texto que comienza con "V" seguido de un número de iteración, es decir, "V0", "V1", "V2", etc. Este ID se utiliza para asociar el veterinario con la dieta asignada.

En la tercera línea del bucle, se crea un ID de dieta como una cadena de texto que comienza con "D" seguido de un número de iteración, es decir, "D0", "D1", "D2", etc. Este ID se utiliza para asociar la dieta con el veterinario asignado.

En la cuarta línea del bucle, se construye una sentencia SQL para insertar el ID de veterinario y el ID de dieta en la tabla "veterinario_dieta" utilizando los valores generados anteriormente.

```
no usages
private static void insertarVeterinarioDieta(){
    for (int i=0; i<=50 ;i++){
        String id_empleado_veterinario_d="V"+i;
        String id_dieta_d="D"+i;
        mySqlOperation.setSqlStatement("insert into veterinario_dieta values('"+id_empleado_veterinario_d+"','"+id_dieta_d+"');");
        mySqlOperation.executeUpdateSqlStatementVoid();
    }
}
```

	Result Grid	Filter Rows:	Export:	Wrap Cell Content:	
1	id_empleado_veterinario_d	id_dieta_d			
2	V0	D0			
3	V1	D1			
4	V2	D2			
5	V3	D3			
6	V4	D4			
7	V5	D5			
8	V6	D6			
9	V7	D7			
10	V8	D8			
11	V9	D9			
12	V10	D10			
13	V11	D11			
14	V12	D12			
15	V13	D13			
16	V14	D14			
17	V15	D15			
18	V16	D16			
19	V17	D17			
20	V18	D18			
21	V19	D19			

V21	D21
V22	D22
V23	D23
V24	D24
V25	D25
V26	D26
V27	D27
V28	D28
V29	D29
V30	D30
V31	D31
V32	D32
V33	D33
V34	D34
V35	D35
V36	D36
V37	D37
V38	D38
V39	D39
V40	D40

V30	D30
V31	D31
V32	D32
V33	D33
V34	D34
V35	D35
V36	D36
V37	D37
V38	D38
V39	D39
V40	D40
V41	D41
V42	D42
V43	D43
V44	D44
V45	D45
V46	D46
V47	D47
V48	D48
V49	D49

Tabla insertarReporte, su función creada y los registros:

Este método que creé se encarga de insertar datos en la tabla "reporte" de una base de datos. La función utiliza un ciclo for para generar y agregar un total de 50 informes de manera aleatoria.

Para generar los datos de cada informe, se utiliza un objeto de la clase "Reporte", que tiene los siguientes atributos: id_reporte, detalles, fecha_ultimo_peso, id_animal_r, id_empleado_entrenador_r y peso_ultimo.

Se utiliza la biblioteca Java Faker para generar datos aleatorios para algunos de estos atributos, como los detalles (síntomas médicos) y la fecha del último peso (una fecha aleatoria en los últimos dos años).

El id_reporte se genera concatenando la letra "R" con el número del ciclo. El id_animal_r y el id_empleado_entrenador_r se generan concatenando la letra "Animal" o la letra "E" respectivamente, con el número del ciclo.

Luego de crear el objeto Reporte y asignarle sus atributos aleatorios, se utiliza la función "setSqlStatement" para crear una sentencia SQL de inserción con los valores del objeto, y luego se utiliza "executeSqlStatementVoid" para ejecutar la sentencia y agregar el informe a la tabla "reporte" de la base de datos.

```

no usages
private static void insertarReporte(){

    for (int i=0; i<=50 ; i++) {
        Reporte r = new Reporte();
        r.setId_reporte("R-" + i);
        r.setDetalles(faker.medical().symptoms());
        r.setFecha_ultimo_peso(String.valueOf(faker.date().between(Date.from(LocalDate.of( year: 2023, month: 2, dayOfMonth: 2).atStartOfDay(ZoneId.systemDefault()))).format(DateTimeFormatter.ofPattern("dd/MM/yyyy"))));
        r.setId_animal_r("Animal-" + i);
        r.setId_empleado_entrenador_r("E" + i);
        r.setPeso_ultimo(String.valueOf(faker.number().randomDouble( maxNumberOfDecimals: 1, min: 1, max: 650)));


        mySqlOperation.setSqlStatement("insert into reporte values('" + r.getId_reporte() + "','" + r.getFecha_ultimo_peso() + "','" + r.getDetalles() + "','" + r.getId_empleado_entrenador_r() + "','" + r.getId_animal_r() + "')");
        mySqlOperation.executeSqlStatementVoid();
    }
}

```

id_reporte	peso_ultimo	fecha_ultimo_peso	detalles	id_empleado_entrenador_r	id_animal_r
R-0	411.6	Thu Feb 16 05:59:35 COT 2023	Vomiting coffee ground material	E0	Animal-0
R-1	88.9	Tue Feb 07 02:28:35 COT 2023	Fainting	E1	Animal-1
R-10	327.7	Thu Feb 09 04:05:46 COT 2023	Obesity	E10	Animal-10
R-11	338.1	Wed Feb 08 14:01:45 COT 2023	Confusion	E11	Animal-11
R-12	188.2	Mon Feb 06 20:03:52 COT 2023	Depressed	E12	Animal-12
R-13	69.6	Sat Feb 04 11:16:28 COT 2023	Unsteady gait (Trouble walking)	E13	Animal-13
R-14	269.9	Sat Feb 11 00:36:05 COT 2023	Confusion	E14	Animal-14
R-15	148.8	Fri Feb 10 11:24:26 COT 2023	Throat pain	E15	Animal-15
R-16	620.5	Sat Feb 18 16:21:02 COT 2023	Swallowing problem (Dysphagia)	E16	Animal-16
R-17	607.7	Mon Feb 06 15:41:29 COT 2023	Mouth ulcers	E17	Animal-17
R-18	164.4	Fri Feb 17 15:52:43 COT 2023	Sinus pain and pressure	E18	Animal-18
R-19	81.8	Sun Feb 19 04:36:19 COT 2023	Vertigo (Room spinning)	E19	Animal-19
R-2	411.8	Thu Feb 02 15:46:53 COT 2023	Ringing in ears (Tinnitus)	E2	Animal-2
R-20	590.1	Mon Feb 13 14:16:06 COT 2023	Face numbness (paresthesias)	E20	Animal-20
R-21	39.7	Fri Feb 17 17:15:43 COT 2023	Tremors	E21	Animal-21
R-22	349.0	Thu Feb 09 16:02:51 COT 2023	Choking	E22	Animal-22
R-23	212.8	Thu Feb 02 13:19:27 COT 2023	Lethargy (Sluggishness)	E23	Animal-23
R-24	626.0	Sat Feb 04 03:34:13 COT 2023	Hand redness	E24	Animal-24
R-25	92.6	Fri Feb 17 23:47:56 COT 2023	Mouth pain	E25	Animal-25
R-26	428.4	Mon Feb 13 20:45:05 COT 2023	Nasal congestion	E26	Animal-26

id_reporte	peso_ultimo	fecha_ultimo_peso	detalles	id_empleado_entrenador_r	id_animal_r
R-28	242.7	Sat Feb 18 09:34:02 COT 2023	Incontinence (leaking urine)	E28	Animal-28
R-29	580.5	Fri Feb 17 12:17:45 COT 2023	Hand numbness (paresthesias)	E29	Animal-29
R-3	564.0	Thu Feb 16 04:16:18 COT 2023	Hemoptysis (Coughing blood)	E3	Animal-3
R-30	504.0	Sat Feb 18 09:45:55 COT 2023	Facial pain	E30	Animal-30
R-31	550.6	Wed Feb 15 13:35:29 COT 2023	Amenorrhea (No menstruation)	E31	Animal-31
R-32	433.0	Thu Feb 16 23:25:48 COT 2023	Inconsolable baby	E32	Animal-32
R-33	408.6	Sat Feb 11 08:23:37 COT 2023	Impotence	E33	Animal-33
R-34	103.0	Thu Feb 16 05:26:39 COT 2023	Overdose	E34	Animal-34
R-35	495.6	Wed Feb 15 17:56:22 COT 2023	Frenquenterization (Frequency)	E35	Animal-35
R-36	56.3	Sat Feb 11 13:15 (Wed Feb 15 17:56:22 COT 2023)		E36	Animal-36
R-37	15.8	Tue Feb 07 19:42:06 COT 2023	Ringing in ears (Tinnitus)	E37	Animal-37
R-38	546.9	Sat Feb 04 00:53:15 COT 2023	Overdose	E38	Animal-38
R-39	45.0	Thu Feb 02 16:48:43 COT 2023	Heart pulsations and palpitations	E39	Animal-39
R-4	625.6	Sun Feb 19 00:06:17 COT 2023	Face numbness (paresthesias)	E4	Animal-4
R-40	58.7	Wed Feb 15 01:47:01 COT 2023	Low heart rate	E40	Animal-40
R-41	333.5	Wed Feb 08 18:51 COT 2023	Hemoptysis (Coughing blood)	E41	Animal-41
R-42	368.0	Wed Feb 08 16:52:33 COT 2023	Sore throat	E42	Animal-42
R-43	274.0	Fri Feb 03 02:00:47 COT 2023	Skin Itching	E43	Animal-43
R-44	236.6	Sun Feb 05 07:03:26 COT 2023	Hand redness	E44	Animal-44
R-45	71.7	Sun Feb 12 19:01:21 COT 2023	Unsteady gait (Trouble walking)	E45	Animal-45

A screenshot of a database management system interface showing a result grid. The grid has columns: id_reporte, peso_ultimo, fecha_ultimo_peso, detalles, id_empleado_entrenador_r, and id_animal_r. The data consists of 50 rows, each representing a report entry with various symptoms and associated animal IDs.

id_reporte	peso_ultimo	fecha_ultimo_peso	detalles	id_empleado_entrenador_r	id_animal_r
R-38	546.9	Sat Feb 04 00:53:15 COT 2023	Overdose	E38	Animal-38
R-39	45.0	Thu Feb 02 16:48:43 COT 2023	Heart pulsations and palpitations	E39	Animal-39
R-4	625.6	Sun Feb 19 00:06:17 COT 2023	Face numbness (paresthesias)	E4	Animal-4
R-40	58.7	Wed Feb 15 01:47:01 COT 2023	Low heart rate	E40	Animal-40
R-41	333.5	Wed Feb 08 18:51:01 COT 2023	Hemoptysis (Coughing blood)	E41	Animal-41
R-42	368.0	Wed Feb 08 16:52:33 COT 2023	Sore throat	E42	Animal-42
R-43	274.0	Fri Feb 03 02:00:47 COT 2023	Skin itching	E43	Animal-43
R-44	236.6	Sun Feb 05 07:03:26 COT 2023	Hand redness	E44	Animal-44
R-45	71.7	Sun Feb 12 19:01:21 COT 2023	Unsteady gait (Trouble walking)	E45	Animal-45
R-46	443.7	Sat Feb 04 22:42:23 COT 2023	Leg ache or pain	E46	Animal-46
R-47	136.3	Tue Feb 14 18:25:12 COT 2023	Anxiety (Nervousness)	E47	Animal-47
R-48	406.8	Wed Feb 15 05:25:16 COT 2023	Fever in the returning traveler	E48	Animal-48
R-49	202.6	Wed Feb 15 16:00:22 COT 2023	Chest pressure	E49	Animal-49
R-5	503.6	Sun Feb 05 20:07:43 COT 2023	Facial drop (weakness)	E5	Animal-5
R-50	71.4	Sun Feb 05 16:15:02 COT 2023	Cough	E50	Animal-50
R-6	431.1	Sat Feb 18 15:47:07 COT 2023	Statorrhea (Excess fat in stool)	E6	Animal-6
R-7	166.0	Mon Feb 06 07:06:05 COT 2023	Mouth pain	E7	Animal-7
R-8	606.6	Fri Feb 17 14:14:44 COT 2023	Insomniac baby	E8	Animal-8
R-9	259.8	Thu Feb 09 09:43:39 COT 2023	Fever in the returning traveler	E9	Animal-9
*	NULL	NULL	NULL	NULL	NULL

Tabla insertarReporte_veterinario, su función creada y los registros:

Este método inserta datos en la tabla reporte_veterinario de una base de datos mediante sentencias SQL. La función itera 50 veces en un bucle for y en cada iteración asigna valores a dos variables id_empleado_veterinario_r e id_reporte_r para luego utilizarlas en la sentencia SQL. La sentencia SQL se construye utilizando la función setSqlStatement del objeto mySqlOperation, y luego se ejecuta mediante la función executeSqlStatementVoid

```

no usages
private static void insertarReporte_veterinario(){
    for (int i=0; i<=50 ;i++){
        String id_empleado_veterinario_r=(V"+i);
        String id_reporte_r=(R- " + i);
        mySqlOperation.setSqlStatement("insert into reporte_veterinario values("+id_empleado_veterinario_r+", "+id_reporte_r+");");
        mySqlOperation.executeSqlStatementVoid();
    }
}

```

A screenshot of a database management system interface showing a result grid. The grid has columns: id_empleado_veterinario_r and id_reporte_r. The data consists of 50 rows, each representing a report entry with corresponding employee and report IDs.

id_empleado_veterinario_r	id_reporte_r
V0	R-0
V3	R-3
V7	R-7
V15	R-15
V17	R-17
V20	R-20
V23	R-23
V24	R-24
V30	R-30
V34	R-34
V36	R-36
V44	R-44
V46	R-46
V0	R-0
V3	R-3
V7	R-7
V15	R-15
V17	R-17
V20	R-20
V23	R-23

Result Grid | Filter Rows: Export: Wrap Cell Content:

e_veterinario 1

V15	R-15
V17	R-17
V20	R-20
V23	R-23
V24	R-24
V30	R-30
V34	R-34
V36	R-36
V44	R-44
V46	R-46
V0	R-0
V3	R-3
V7	R-7
V15	R-15
V17	R-17
V20	R-20
V23	R-23
V24	R-24
V30	R-30
V34	R-34

Read Only

Result Grid | Filter Rows: Export: Wrap Cell Content:

veterinario 1

V23	R-23
V24	R-24
V30	R-30
V34	R-34
V36	R-36
V44	R-44
V46	R-46
V0	R-0
V3	R-3
V7	R-7
V15	R-15
V17	R-17
V20	R-20
V23	R-23
V24	R-24
V30	R-30
V34	R-34
V36	R-36
V44	R-44
V46	R-46

Tabla insertarAnimal, su función creada y los registros:

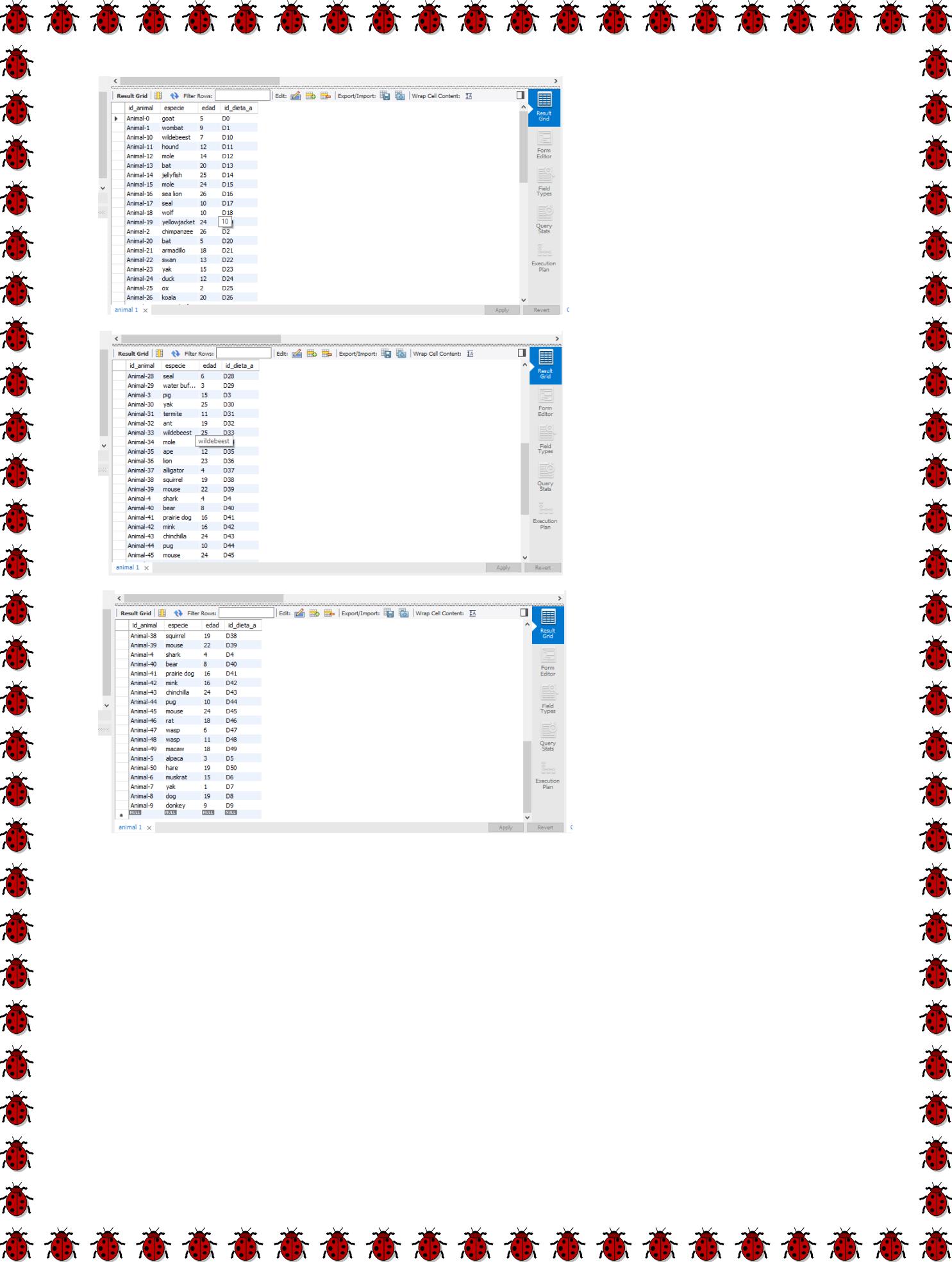
Con este método que creé , realizo una inserción de datos en una tabla llamada "animal". El ciclo for se ejecuta 51 veces ($i <= 50$) y en cada iteración se crea un nuevo objeto de la clase Animal. A este objeto se le asigna un valor para su identificador (id_animal), que toma el valor de "Animal-" seguido del número de iteración, su especie (obtenida con faker.animal().name()), su edad (un número aleatorio entre 1 y 28), y su dieta (un identificador de dieta, que toma el valor de "D" seguido del número de iteración).

Luego se ejecuta una sentencia SQL para insertar estos datos en la tabla "animal" utilizando los valores del objeto Animal creado.

```

no usages
private static void insertarAnimal(){
    for (int i=0; i<=50 ;i++){
        Animal animal= new Animal();
        animal.setId_animal("Animal-"+i);
        animal.setEspecie(faker.animal().name());
        animal.setEdad(String.valueOf(faker.number().numberBetween(1, 28)));
        animal.setId_dieta_a("D"+i);
        mySqlOperation.setSqlStatement("insert into animal values('"+ animal.getId_animal() + "', '"+"
            + animal.getEspecie() + "', '" + animal.getEdad() + "', '" +
            + animal.getId_dieta_a() + "')");
        mySqlOperation.executeSqlStatementVoid();
    }
}

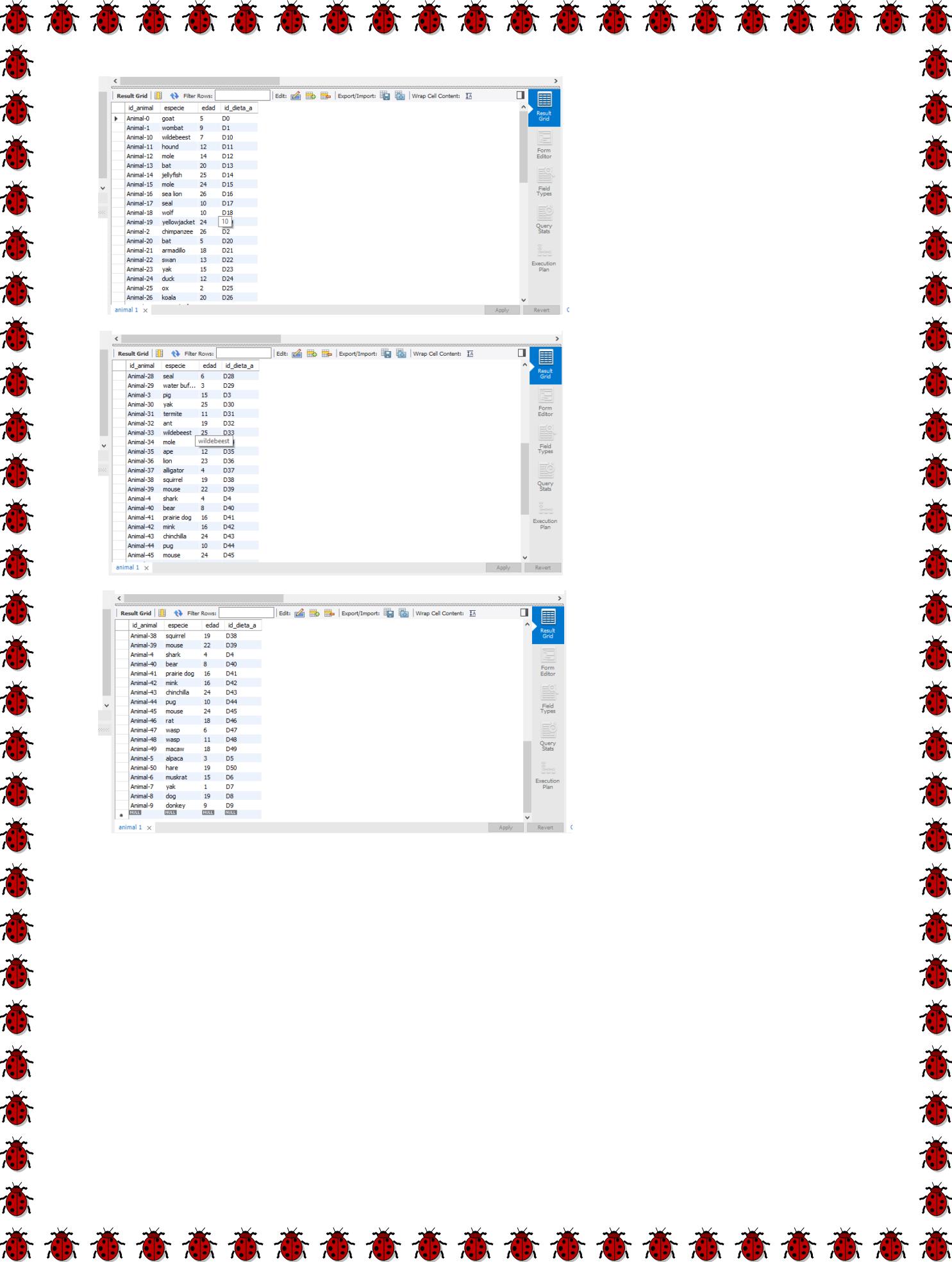
```



Result Grid | Filter Rows: | Edits: | Export/Import: | Wrap Cell Content: |

	id_animal	especie	edad	id_diet_a
▶	Animal-0	goat	5	D0
	Animal-1	wombat	9	D1
	Animal-10	wildebeest	7	D10
	Animal-11	hound	12	D11
	Animal-12	moar	14	D12
	Animal-13	batt	20	D13
	Animal-14	jellyfish	25	D14
	Animal-15	mole	24	D15
	Animal-16	sea lion	26	D16
	Animal-17	seal	10	D17
	Animal-18	wolf	10	D18
	Animal-19	yellowjacket	10	D19
	Animal-2	chimpanzee	26	D2
	Animal-20	bat	5	D20
	Animal-21	armadillo	18	D21
	Animal-22	swan	13	D22
	Animal-23	yak	15	D23
	Animal-24	duck	12	D24
	Animal-25	ox	2	D25
	Animal-26	koala	20	D26

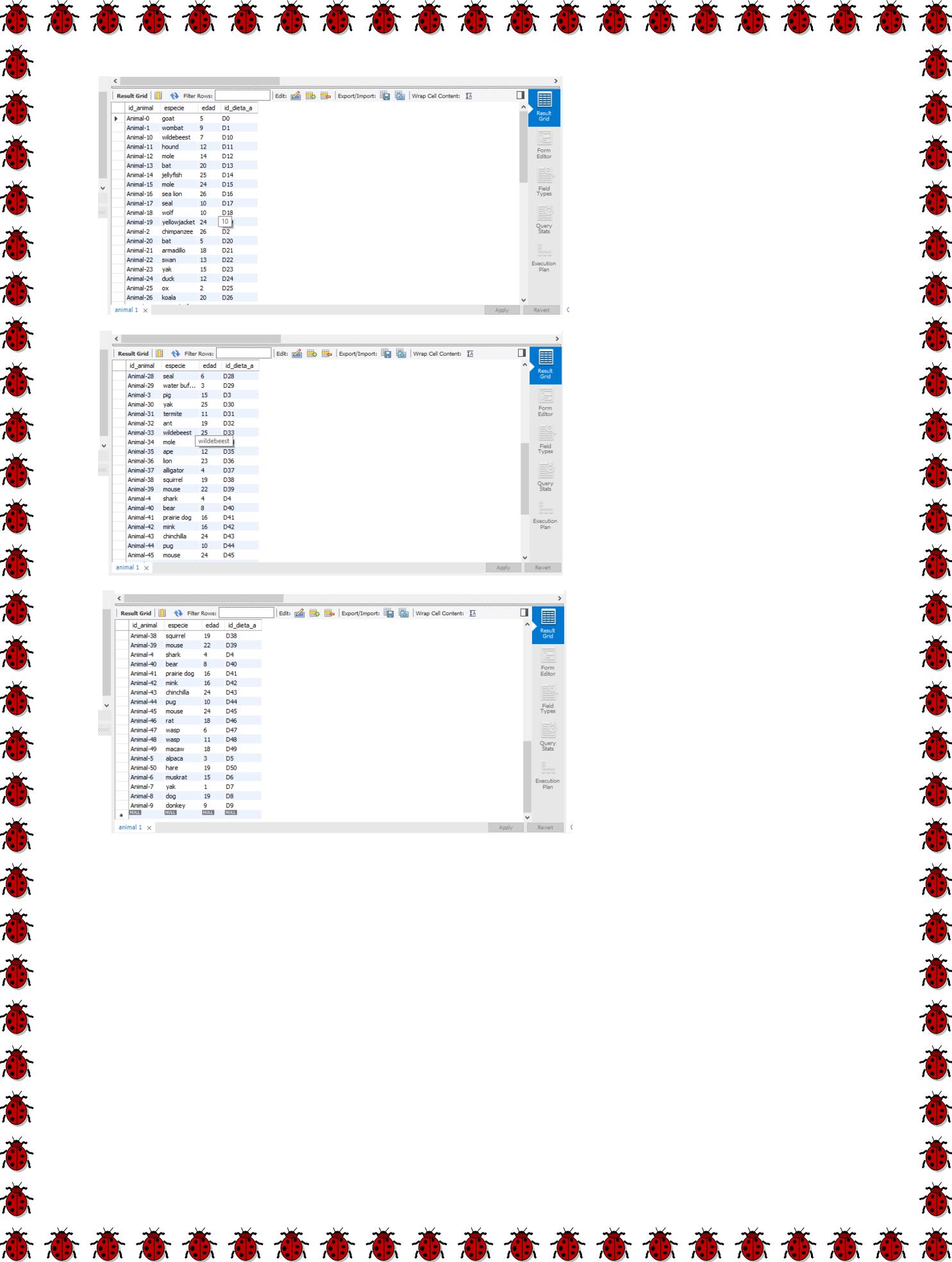
animal 1 x Apply Revert



Result Grid | Filter Rows: | Edits: | Export/Import: | Wrap Cell Content: |

	id_animal	especie	edad	id_diet_a
	Animal-28	seal	6	D28
	Animal-29	water buf...	3	D29
	Animal-30	pig	15	D3
	Animal-30	yak	25	D30
	Animal-31	termite	11	D31
	Animal-32	ant	19	D32
	Animal-33	wildebeest	25	D33
	Animal-34	mole	wildebeest	D34
	Animal-35	ape	12	D35
	Animal-36	lion	23	D36
	Animal-37	alligator	4	D37
	Animal-38	squirrel	19	D38
	Animal-39	mouse	22	D39
	Animal-4	shark	4	D4
	Animal-40	bear	8	D40
	Animal-41	prairie dog	16	D41
	Animal-42	mink	16	D42
	Animal-43	chinchilla	24	D43
	Animal-44	pug	10	D44
	Animal-45	mouse	24	D45
	Animal-46	rat	18	D46
	Animal-47	wasp	6	D47
	Animal-48	wasp	11	D48
	Animal-49	macaw	18	D49
	Animal-5	alpaca	3	D5
	Animal-50	hare	19	D50
	Animal-6	muskrat	15	D6
	Animal-7	yak	1	D7
	Animal-8	dog	19	D8
	Animal-9	donkey	9	D9

animal 1 x Apply Revert



Result Grid | Filter Rows: | Edits: | Export/Import: | Wrap Cell Content: |

	id_animal	especie	edad	id_diet_a
	Animal-38	squirrel	19	D38
	Animal-39	mouse	22	D39
	Animal-4	shark	4	D4
	Animal-40	bear	8	D40
	Animal-41	prairie dog	16	D41
	Animal-42	mink	16	D42
	Animal-43	chinchilla	24	D43
	Animal-44	pug	10	D44
	Animal-45	mouse	24	D45
	Animal-46	rat	18	D46
	Animal-47	wasp	6	D47
	Animal-48	wasp	11	D48
	Animal-49	macaw	18	D49
	Animal-5	alpaca	3	D5
	Animal-50	hare	19	D50
	Animal-6	muskrat	15	D6
	Animal-7	yak	1	D7
	Animal-8	dog	19	D8
	Animal-9	donkey	9	D9
	Animal-10	zebra	10	D10
	Animal-11	zebra	10	D11
	Animal-12	zebra	10	D12
	Animal-13	zebra	10	D13
	Animal-14	zebra	10	D14
	Animal-15	zebra	10	D15
	Animal-16	zebra	10	D16
	Animal-17	zebra	10	D17
	Animal-18	zebra	10	D18
	Animal-19	zebra	10	D19
	Animal-20	zebra	10	D20
	Animal-21	zebra	10	D21
	Animal-22	zebra	10	D22
	Animal-23	zebra	10	D23
	Animal-24	zebra	10	D24
	Animal-25	zebra	10	D25
	Animal-26	zebra	10	D26

animal 1 x Apply Revert

