DOCUMENTACIÓN

James Arley Muñoz Borja
CC: 1152202244

Reto base de datos (Sofka U)

ZOO (Ejercicio B)

El parque zoo santafe “parque de la conservación” quiere registrar en una base de datos el consumo de alimentos por los animales que tiene en su sede.

Usted acaba de hablar con el administrador y el le comenta que tienen una clasificación para los animales (mamiferos, aves, anfibios, peces y reptíles) de los cuales usted debe seleccionar 3 para el MVP.

- Tierragro empresa de alimentos para diferentes especies es uno de los 5 proveedores del parque, pero se esperan que almenos lleguen 10 nuevos proveedores.
- Dentro del parque hay varios roles para las personas, empleados cuidadores, empleados logisticos, empleados veterinarios, empleados entrenadores, visitantes.
- El veterinario esta encargado de realizar consultas a sus especies y de diseñar la dieta de cada especie.
- El alimento de cada especie es diferente y tiene una dosis y un tipo (humeda, seca, etc).
- Uno de los roles de empleado del Zoo debe contactarse con el proveedor para solicitar alimentos y debe asear cada una de las hábitas de las especies.
- El proveedor recibe una orden de compra revisa que tenga todo el alimento que le piden y con el genera una factura, a final de mes el gerente del Zoo consulta las facturas que debe a sus distintos proveedores y genera su pago correspondiente.
- Los empleados entrenadores son los encargados de llevar el peso de cada especie e informar a un veterinario en que condición están.
- El alimento es una entidad fuerte y debe contener sus características.

Se pide:

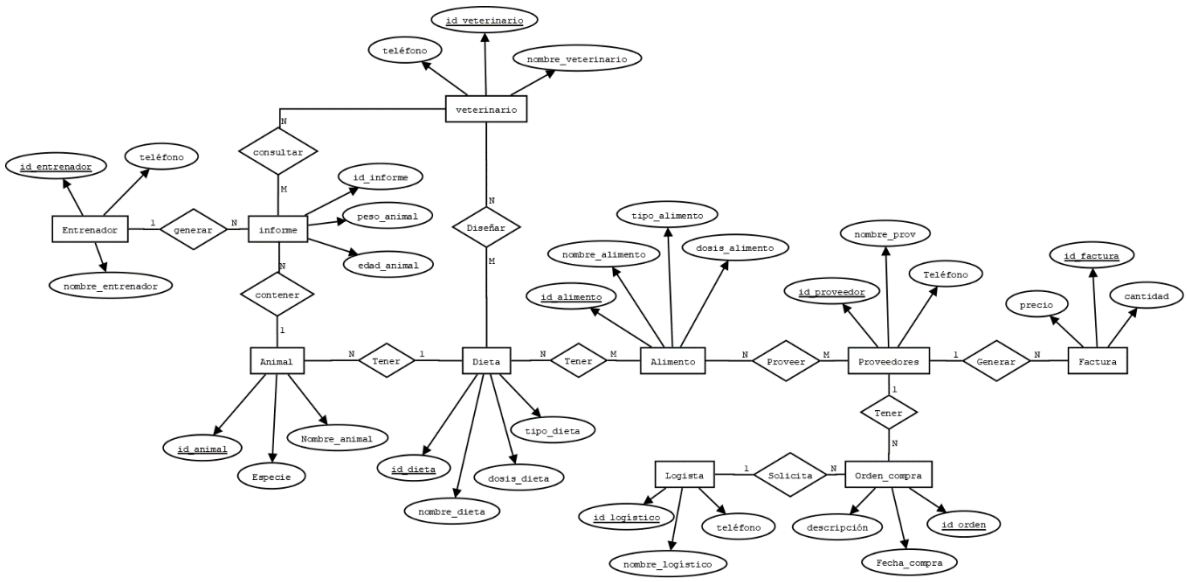
- **Hay que indicar que ejercicio fue asignado**

R/: se me asignó el ejercicio B del zoológico.

- **Realizar el modelo E-R**

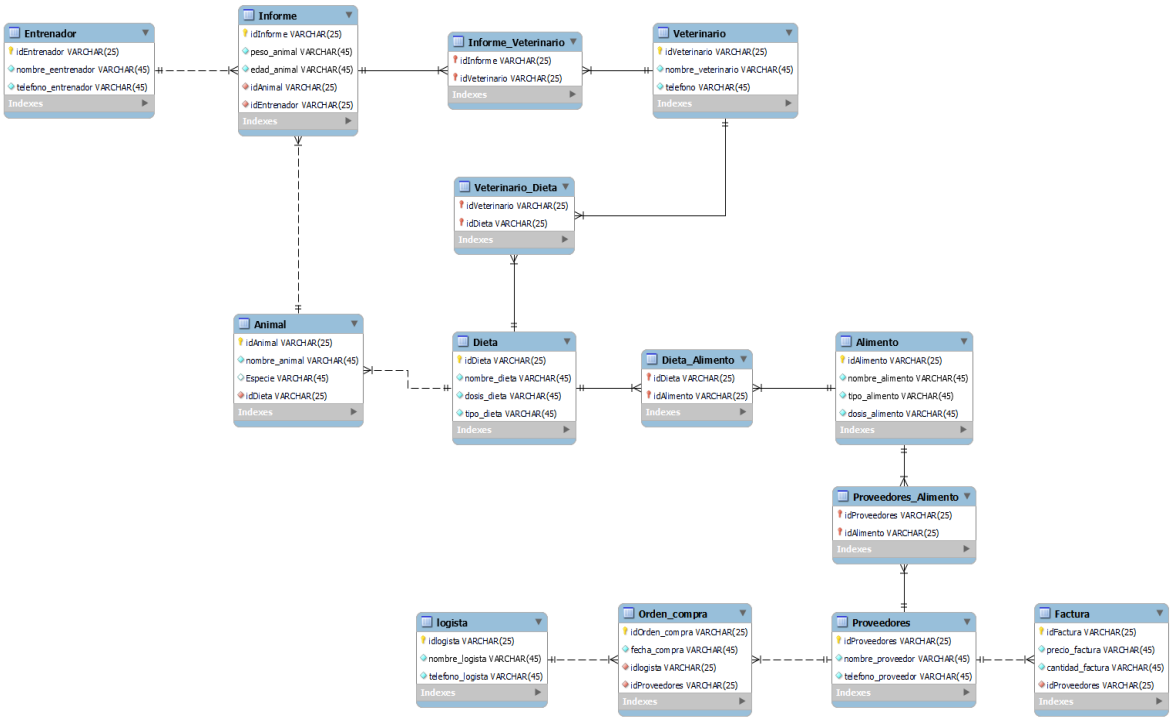
R/:
Para el modelo entidad relación, se crearon las entidades: Animal, informe, entrenador, veterinario, logístico, dieta, alimento, proveedores, orden_compra, factura con sus siguientes atributos.
las entidades se relacionan de la siguiente manera:
Entrenador-informe: un entrenador genera uno o muchos informes y un reporte es generado por un entrenador.
Informe-veterinario: un veterinario puede consultar uno o muchos informes y un informe puede ser consultado por uno o muchos veterinarios.
Veterinario-dieta: un veterinario puede diseñar una o muchas dietas y una dieta puede ser diseñada por uno o muchos veterinarios.
Informe-animal: Un informe puede contener un animal y un animal puede contener muchos informes
Animal-dieta: un animal puede tener una dieta y una dieta puede ser consumida por uno o muchos animales.
Dieta-alimento: una dieta puede tener uno o muchos alimentos y un alimento puede tener una o muchas dietas.
Alimento-proveedor: un alimento se puede proveer por uno o muchos proveedores y un proveedor puede proveer uno o muchos alimentos.
Proveedor-factura: un proveedor puede generar una o muchas facturas, pero una factura puede ser generada por un proveedor.
Orden_compra-proveedor: un proveedor puede tener una o muchas órdenes de compra, pero una orden de compra tiene un solo proveedor.
Logístico-orden_compra: un logístico puede generar una o muchas órdenes de compra, pero una orden de compra debe ser generada por un logístico.

se adjunta el modelo entidad relación creado y se adjunta al repositorio:



- **Realizar el modelo relacional**

R/: Creado el MER se procede a crear el modelo relacional quedando de la siguiente forma:



- **Normalizar correctamente**

R/: Normalización

1N (Primera Normalización)

El modelo relacional cumple con la primera normalización porque los atributos de las entidades son atómicos, no hay atributos multivaluados y se garantiza que no hay registros repetidos.

2N (Segunda normalización)

El modelo relacional cumple con la segunda normalización porque está en la primera forma y además todos los valores de las columnas dependen únicamente de una llave primaria de la tabla

3N (Tercera Normalización)

El modelo relacional cumple con la tercera normalización porque está en la segunda forma y no tiene dependencias transitivas, como se puede observar las columnas en cada tabla dependen directamente de la clave primaria de esa misma tabla

- **Escribir con sentencias SQL toda la definición de la base de datos.**

R/: adjunto imágenes de las sentencias SQL, de igual manera estarán adjuntas en el script.

```sql
CREATE SCHEMA Zoo_Santafe;
USE ZOO_Santafe;

-- Tabla Dieta
CREATE TABLE Dieta (
  idDieta VARCHAR(25) NOT NULL,
  nombre_dieta VARCHAR(45) NOT NULL,
  dosis_dieta VARCHAR(45) NOT NULL,
  tipo_dieta VARCHAR(45) NOT NULL,
  PRIMARY KEY (idDieta));

-- Tabla Animal
CREATE TABLE Animal (
  idAnimal VARCHAR(25) NOT NULL,
  nombre_animal VARCHAR(45) NOT NULL,
  Especie VARCHAR(45) NULL,
  idDieta VARCHAR(25) NOT NULL,
  PRIMARY KEY (idAnimal),
  FOREIGN KEY (idDieta)
  REFERENCES Dieta (idDieta));

-- Tabla logista
CREATE TABLE logista (
  idlogista VARCHAR(25) NOT NULL,
  nombre_logista VARCHAR(45) NOT NULL,
  telefono_logista VARCHAR(45) NOT NULL,
  PRIMARY KEY (idlogista));

-- Tabla Entrenador
CREATE TABLE Entrenador (
  idEntrenador VARCHAR(25) NOT NULL,
  nombre_eentrenador VARCHAR(45) NOT NULL,
  telefono_entrenador VARCHAR(45) NOT NULL,
  PRIMARY KEY (idEntrenador));

-- Tabla Veterinario
CREATE TABLE Veterinario (
  idVeterinario VARCHAR(25) NOT NULL,
```

```
39      nombre_veterinario VARCHAR(45) NOT NULL,
40      telefono VARCHAR(45) NOT NULL,
41      PRIMARY KEY (idVeterinario));
42
43    -- Tabla Alimento
44    CREATE TABLE Alimento (
45      idAlimento VARCHAR(25) NOT NULL,
46      nombre_alimento VARCHAR(45) NOT NULL,
47      tipo_alimento VARCHAR(45) NOT NULL,
48      dosis_alimento VARCHAR(45) NOT NULL,
49      PRIMARY KEY (idAlimento));
50
51    -- Tabla Proveedores
52    CREATE TABLE Proveedores (
53      idProveedores VARCHAR(25) NOT NULL,
54      nombre_proveedor VARCHAR(45) NOT NULL,
55      telefono_proveedor VARCHAR(45) NOT NULL,
56      PRIMARY KEY (idProveedores));
57
58    -- Tabla Orden_compra
59    CREATE TABLE Orden_compra (
60      idOrden_compra VARCHAR(25) NOT NULL,
61      fecha_compra VARCHAR(45) NOT NULL,
62      idlogista VARCHAR(25) NOT NULL,
63      idProveedores VARCHAR(25) NOT NULL,
64      PRIMARY KEY (idOrden_compra),
65      FOREIGN KEY (idlogista)
66      REFERENCES logista (idlogista),
67      FOREIGN KEY (idProveedores)
68      REFERENCES Proveedores (idProveedores));
69
70    -- Tabla Animal_Entrenador
71    CREATE TABLE Animal_Entrenador (
72      idAnimal VARCHAR(25) NOT NULL,
73      idEntrenador VARCHAR(25) NOT NULL,
74      PRIMARY KEY (idAnimal, idEntrenador),
75      FOREIGN KEY (idAnimal)
76      REFERENCES Animal (idAnimal),
```

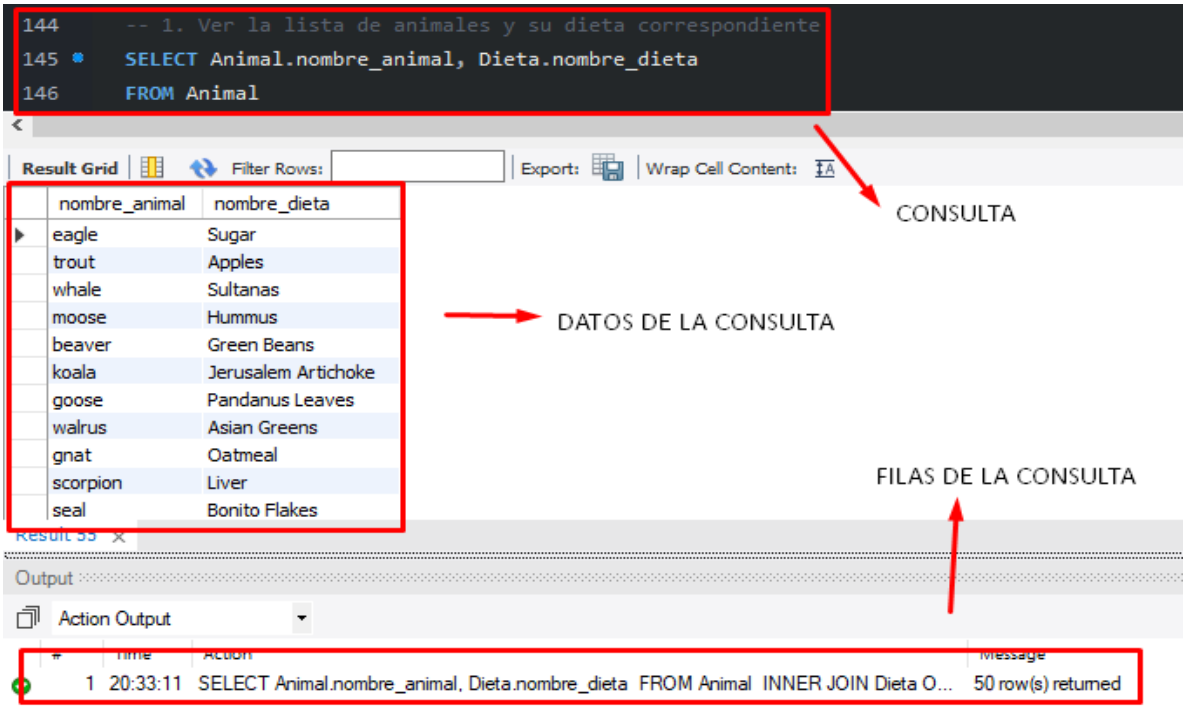
```
    FOREIGN KEY (idEntrenador)
    REFERENCES Entrenador (idEntrenador));

-- Tabla Entrenador_Veterinario
CREATE TABLE Entrenador_Veterinario (
    idEntrenador VARCHAR(25) NOT NULL,
    idVeterinario VARCHAR(25) NOT NULL,
    PRIMARY KEY (idEntrenador, idVeterinario),
    FOREIGN KEY (idEntrenador)
    REFERENCES Entrenador (idEntrenador),
    FOREIGN KEY (idVeterinario)
    REFERENCES Veterinario (idVeterinario));

-- Tabla Dieta_Alimento
CREATE TABLE Dieta_Alimento (
    idDieta VARCHAR(25) NOT NULL,
    idAlimento VARCHAR(25) NOT NULL,
    PRIMARY KEY (idDieta, idAlimento),
    FOREIGN KEY (idDieta)
    REFERENCES Dieta (idDieta),
    FOREIGN KEY (idAlimento)
    REFERENCES Alimento (idAlimento));

-- Tabla Veterinario_Dieta
CREATE TABLE Veterinario_Dieta (
    idVeterinario VARCHAR(25) NOT NULL,
    idDieta VARCHAR(25) NOT NULL,
    PRIMARY KEY (idVeterinario, idDieta),
    FOREIGN KEY (idVeterinario)
    REFERENCES Veterinario (idVeterinario),
    FOREIGN KEY (idDieta)
    REFERENCES Dieta (idDieta));

-- Tabla Proveedores_Alimento
CREATE TABLE Proveedores_Alimento (
    idProveedores VARCHAR(25) NOT NULL,
    idAlimento VARCHAR(25) NOT NULL,
    PRIMARY KEY (idProveedores, idAlimento),
```

```
115     FOREIGN KEY (idProveedores)
116     REFERENCES Proveedores (idProveedores),
117     FOREIGN KEY (idAlimento)
118     REFERENCES Alimento (idAlimento));
119
120   -- Tabla Factura
121   CREATE TABLE Factura (
122     idFactura VARCHAR(25) NOT NULL,
123     precio_factura VARCHAR(45) NOT NULL,
124     cantidad_factura VARCHAR(45) NOT NULL,
125     idProveedores VARCHAR(25) NOT NULL,
126     PRIMARY KEY (idFactura),
127     FOREIGN KEY (idProveedores)
128     REFERENCES Proveedores (idProveedores));
129
130     CREATE TABLE Informe (
131     idInforme VARCHAR(25) NOT NULL,
132     peso_animal VARCHAR(45) NULL,
133     edad_animal VARCHAR(45) NULL,
134     idAnimal VARCHAR(25) NOT NULL,
135     idEntrenador VARCHAR(25) NOT NULL,
136     PRIMARY KEY (idInforme),
137     FOREIGN KEY (idAnimal)
138     REFERENCES Animal (idAnimal),
139     FOREIGN KEY (idEntrenador)
140     REFERENCES Entrenador (idEntrenador));
```

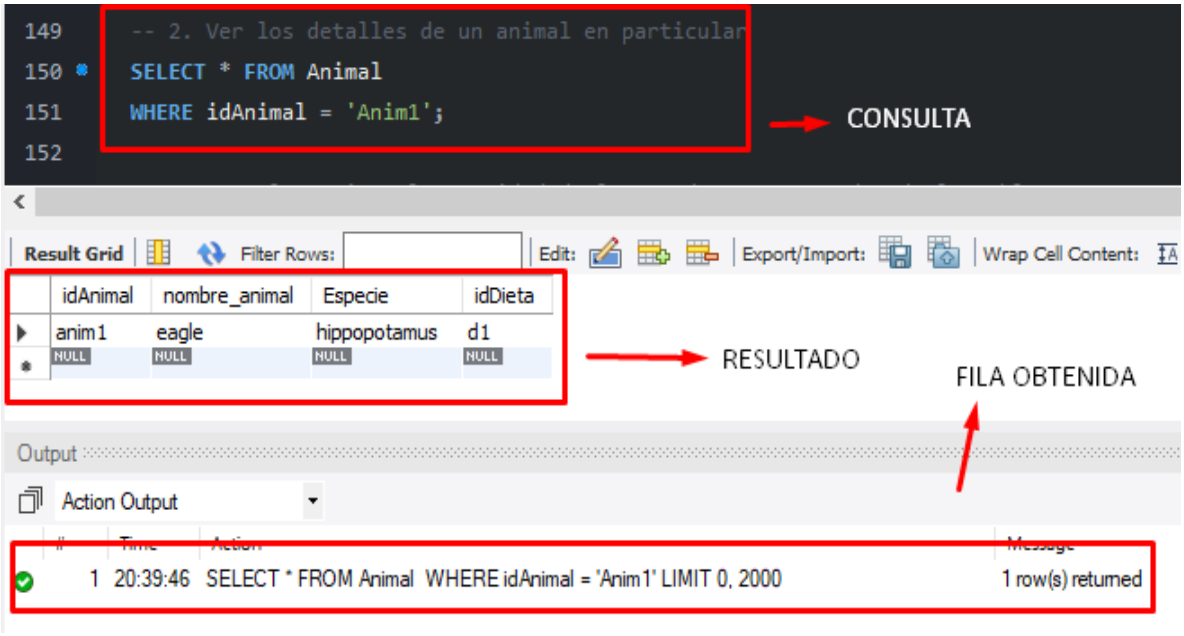
- **Escribir consultas que me permitan ver la información de cada tabla o de varias tablas (10).**

R/: Se realizaron 10 consultas que hacen lo siguiente:

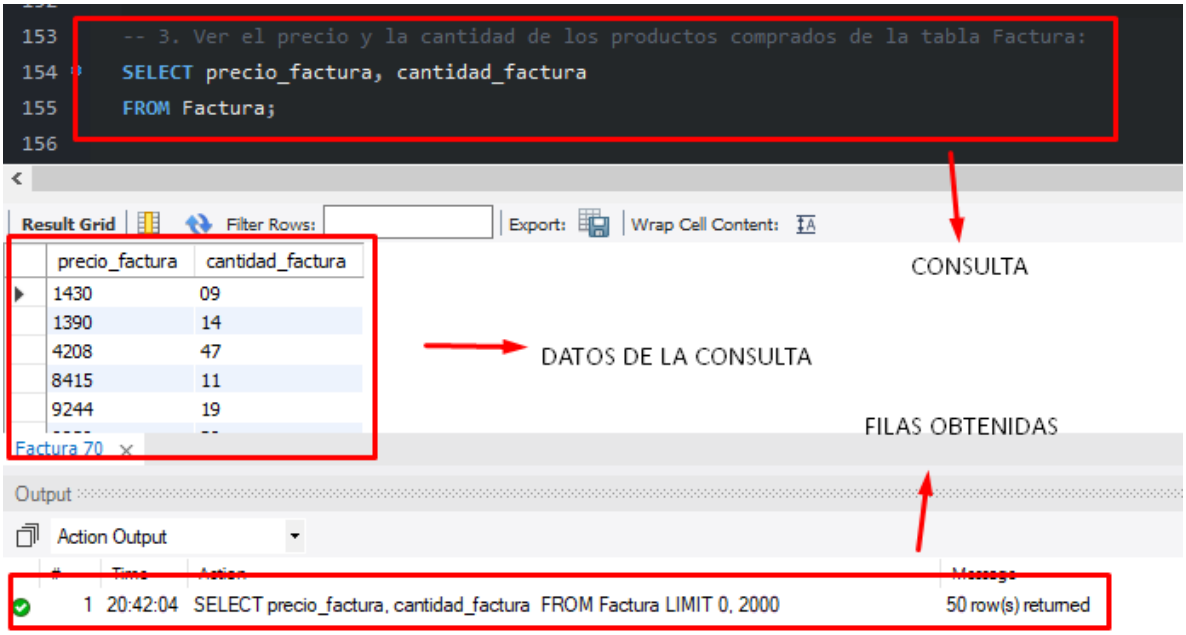
Consulta 1: Ver la lista de animales y su dieta correspondiente:



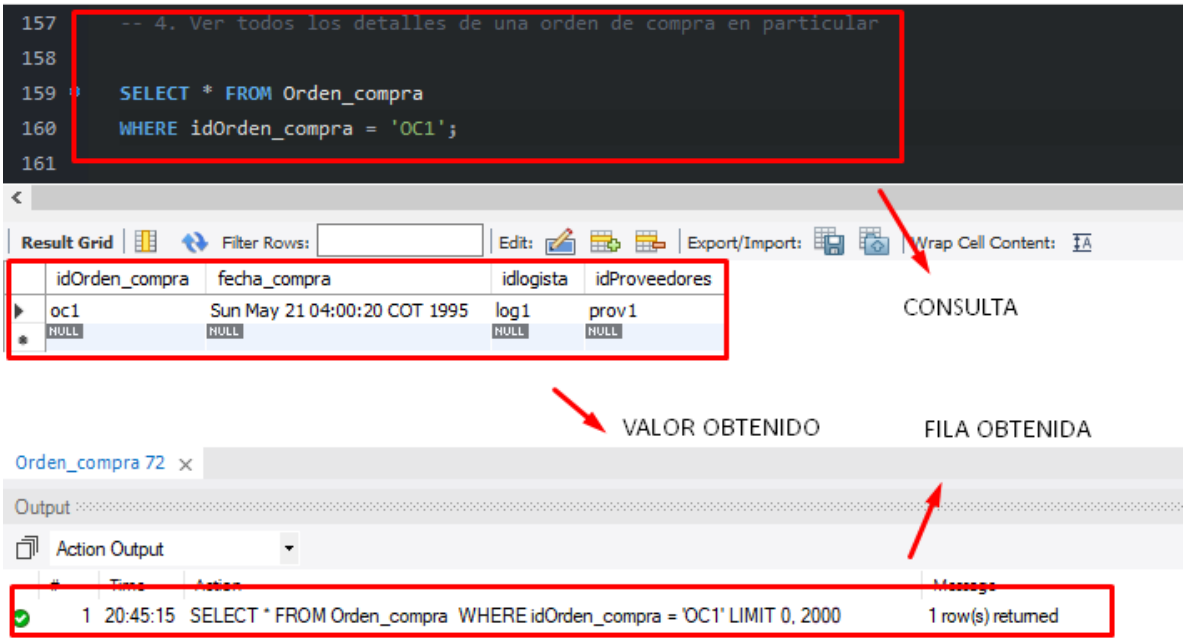
Consulta 2: Ver los detalles de un animal en particular:



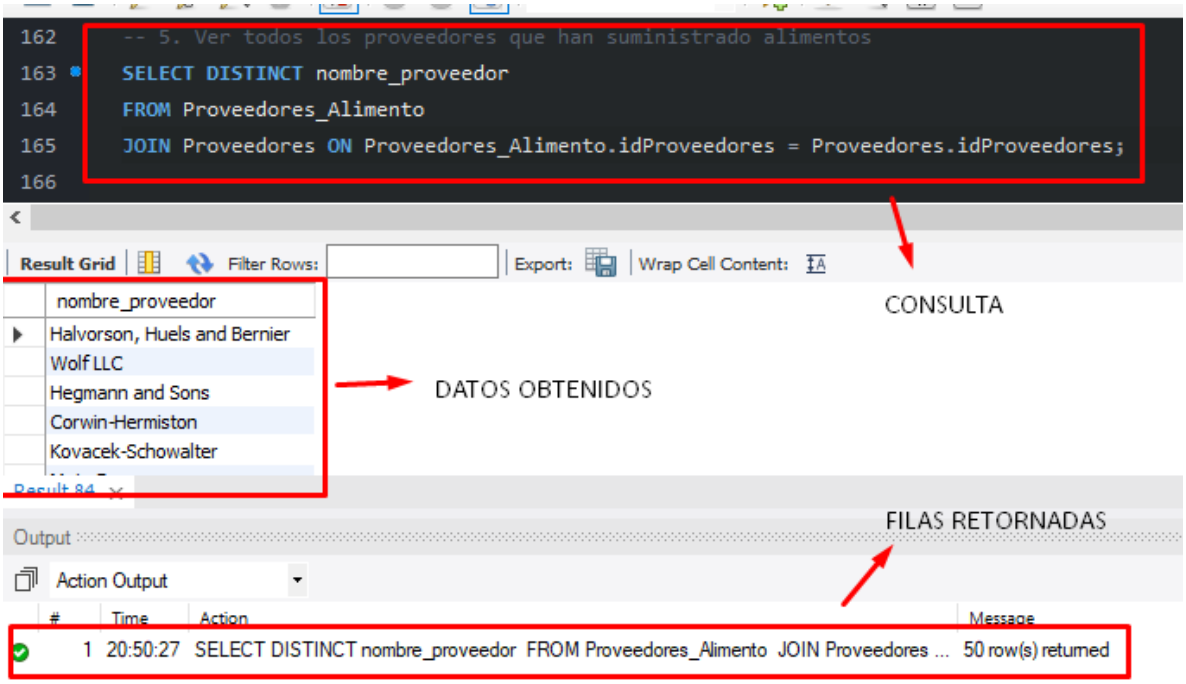
Consulta 3: Ver el precio y la cantidad de los productos comprados de la tabla Factura:



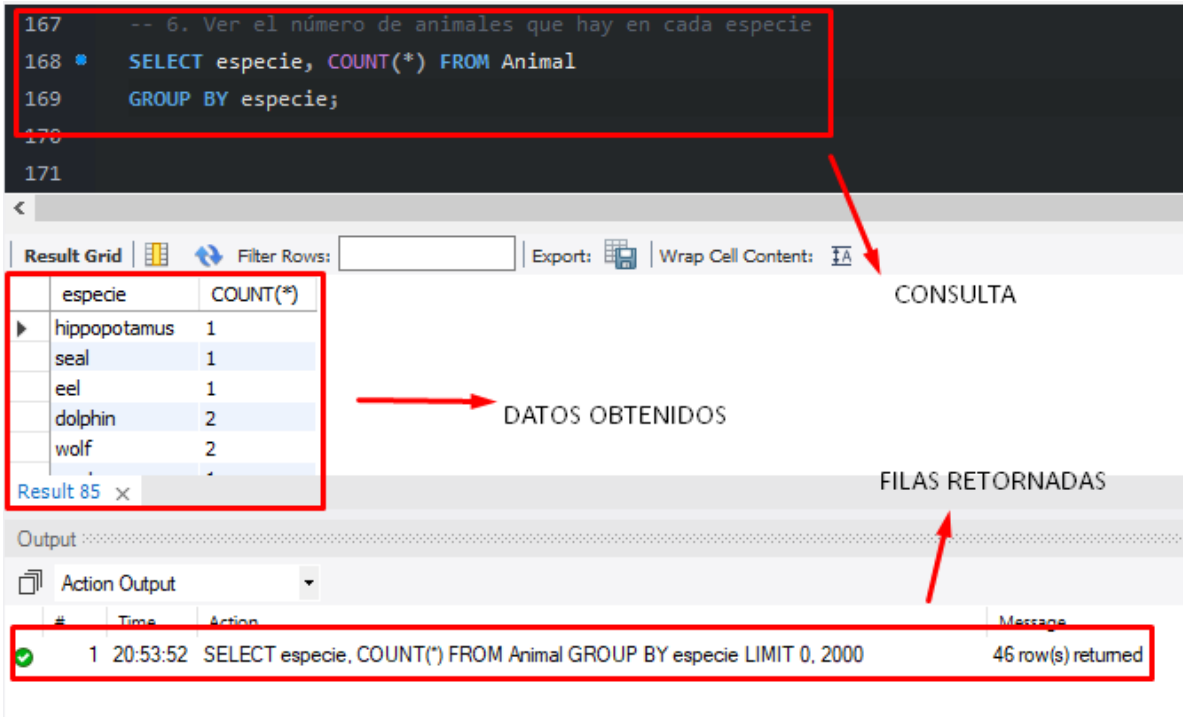
Consulta 4: Ver todos los detalles de una orden de compra en particular:



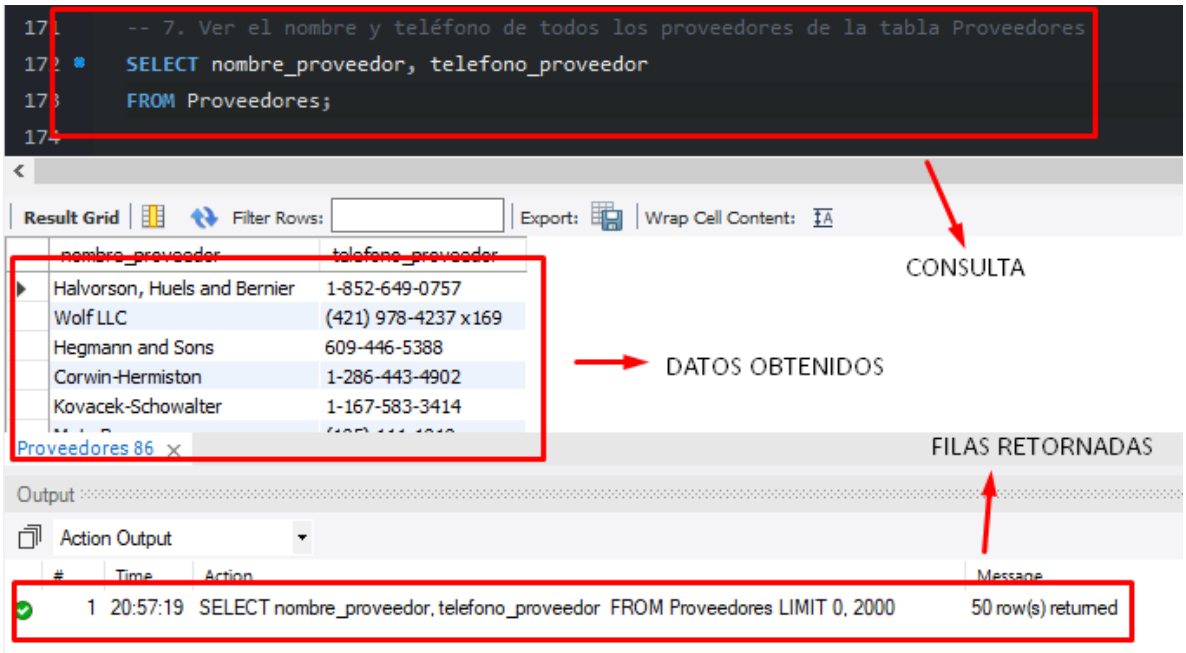
Consulta 5: Ver todos los proveedores que han suministrado alimentos:



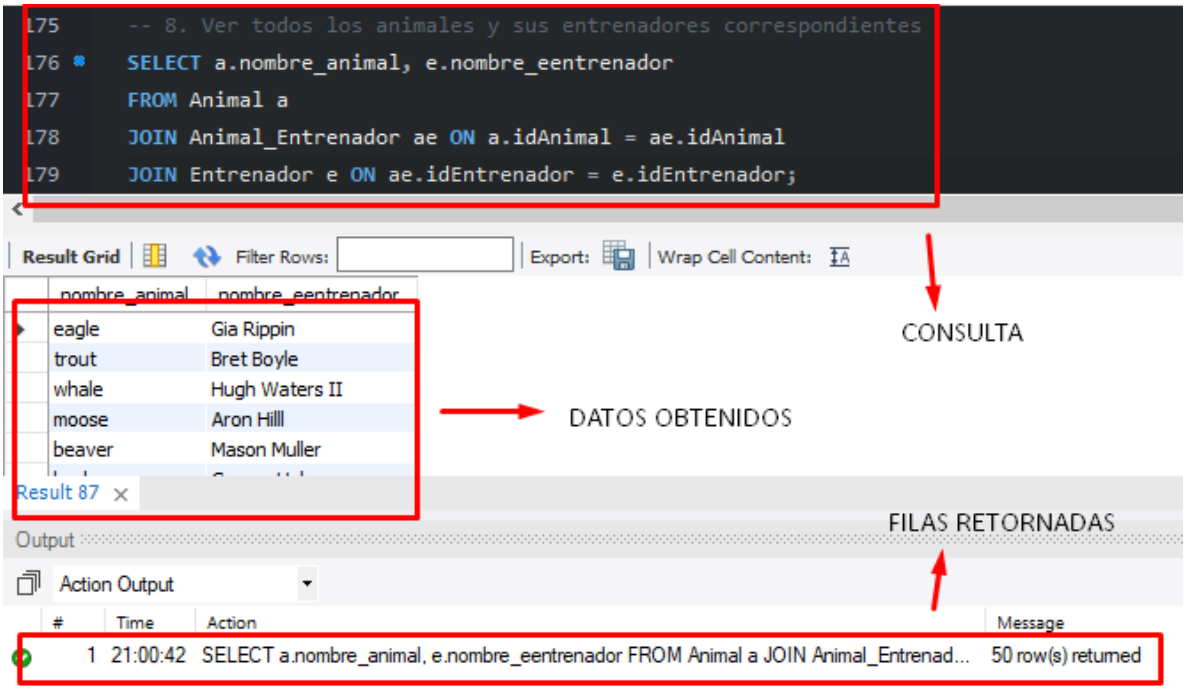
Consulta 6: Ver el número de animales que hay en cada especie:



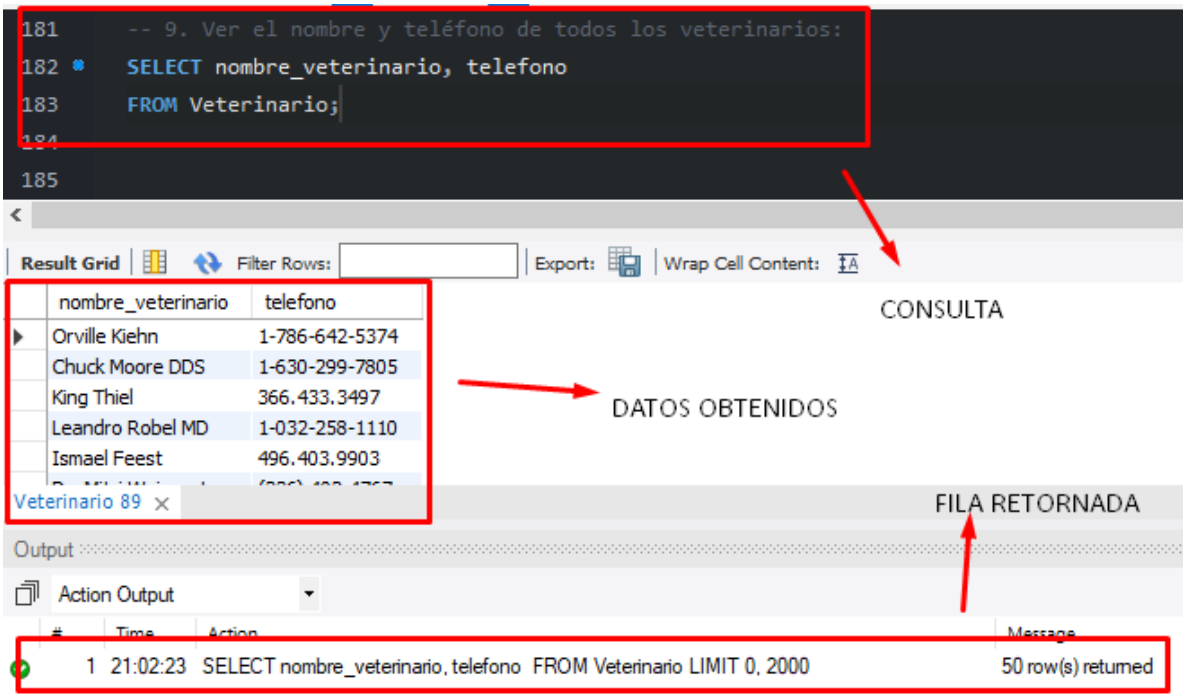
Consulta 7: Ver el nombre y teléfono de todos los proveedores de la tabla Proveedores:



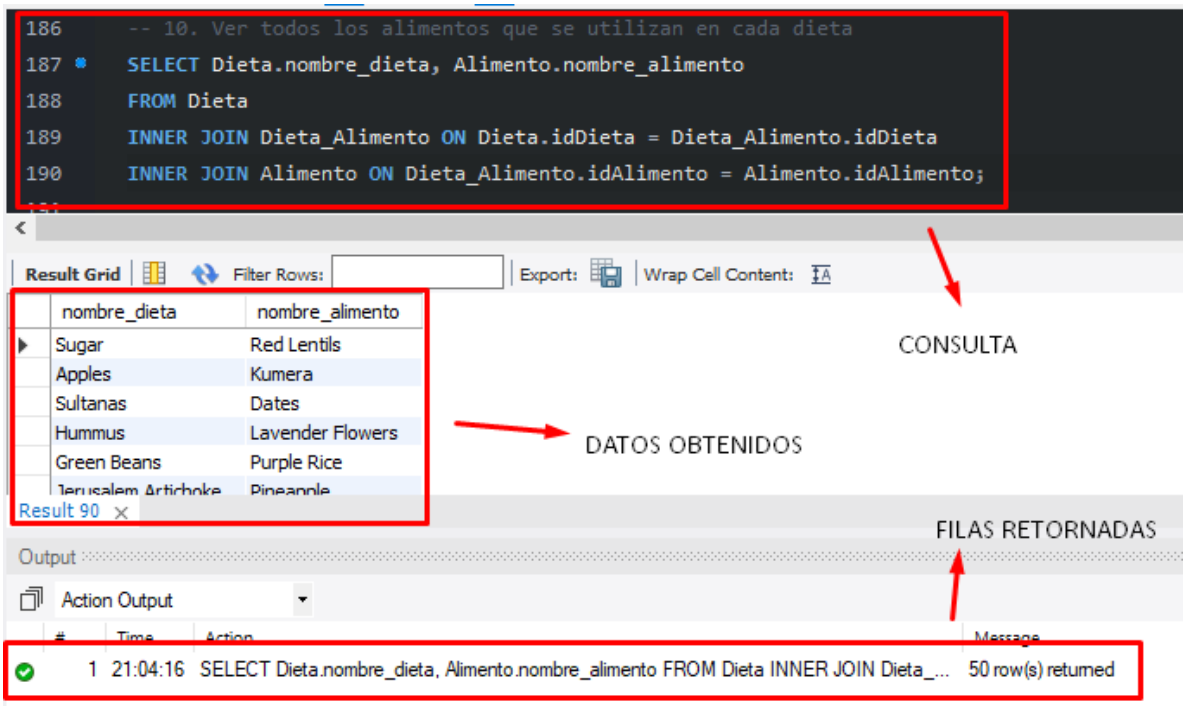
Consulta 8: Ver todos los animales y sus entrenadores correspondientes:



Consulta 9: Ver el nombre y teléfono de todos los veterinarios:



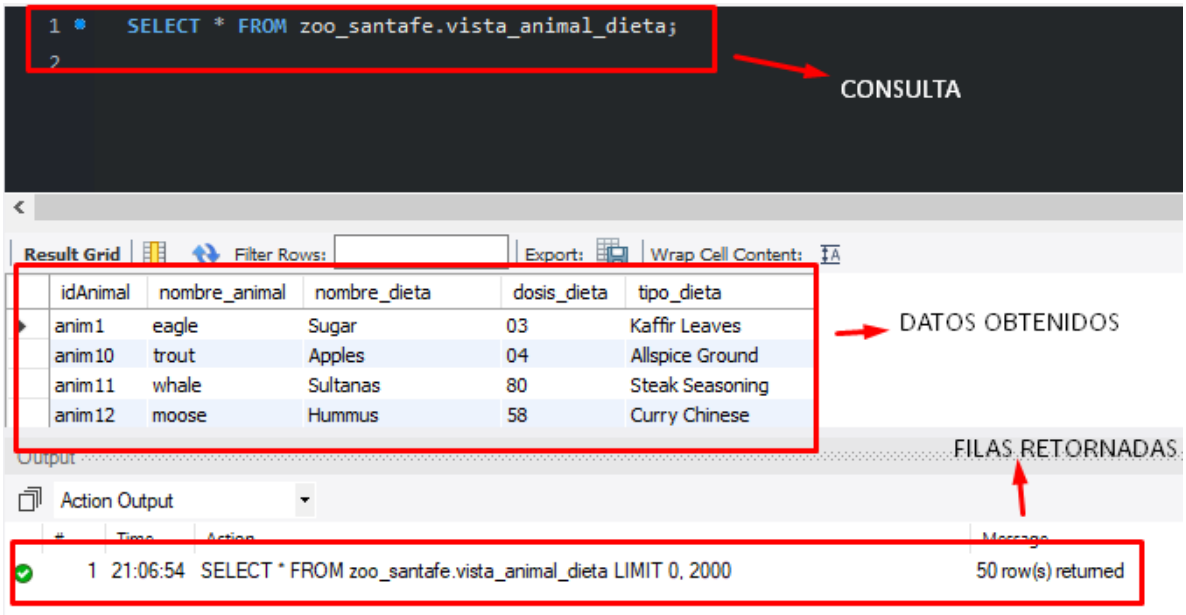
Consulta 10: Ver todos los alimentos que se utilizan en cada dieta



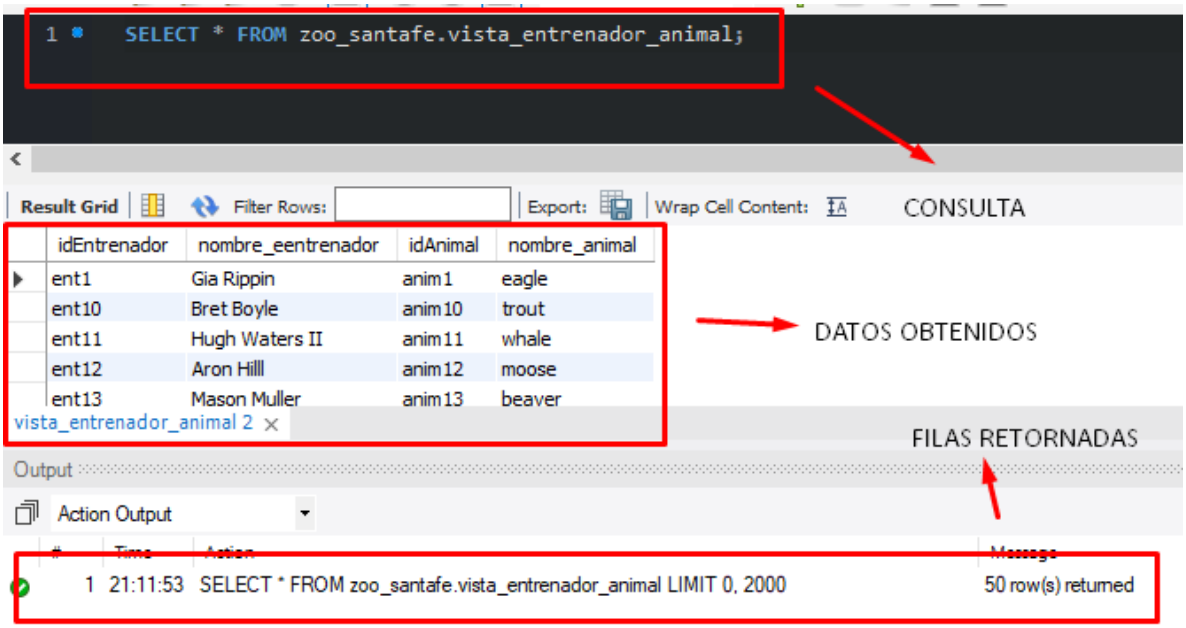
- **Generar de 4 a 6 vistas donde se evidencie lo más importante de cada ejercicio (haga una selección muy responsable de la información realmente importante según el contexto).**

R/: Se generan 4 vistas que hacen lo siguiente:

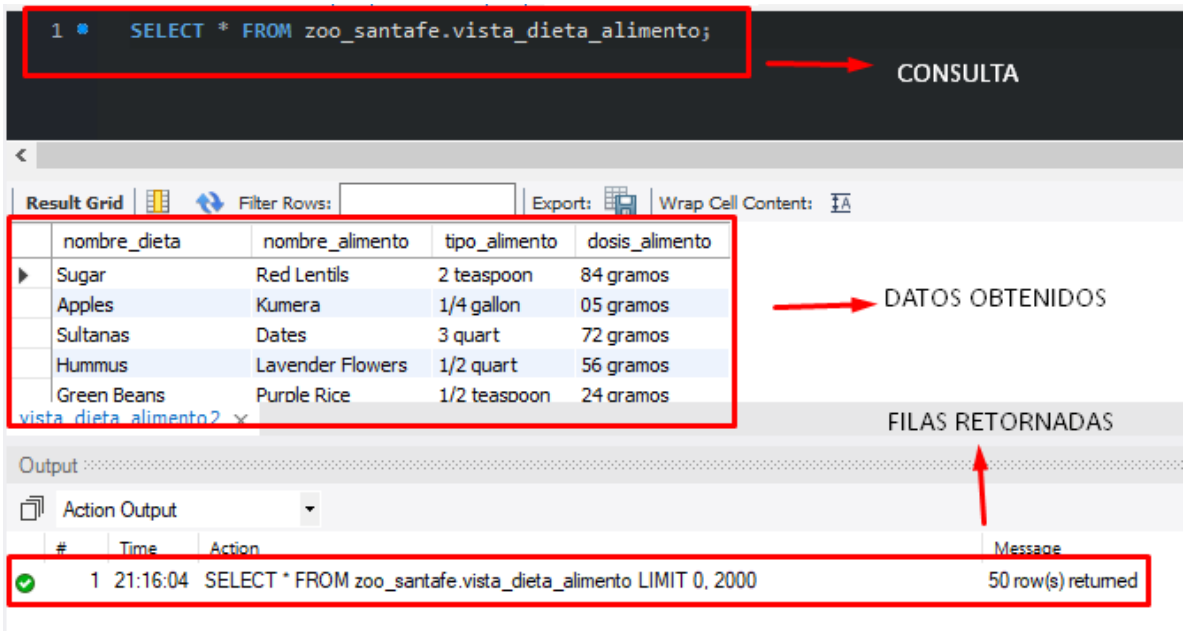
Vista 1: muestra los animales y sus dietas correspondientes lo que puede ser útil para monitorear y asegurarse de que los animales reciban la alimentación adecuada y en las cantidades correctas.



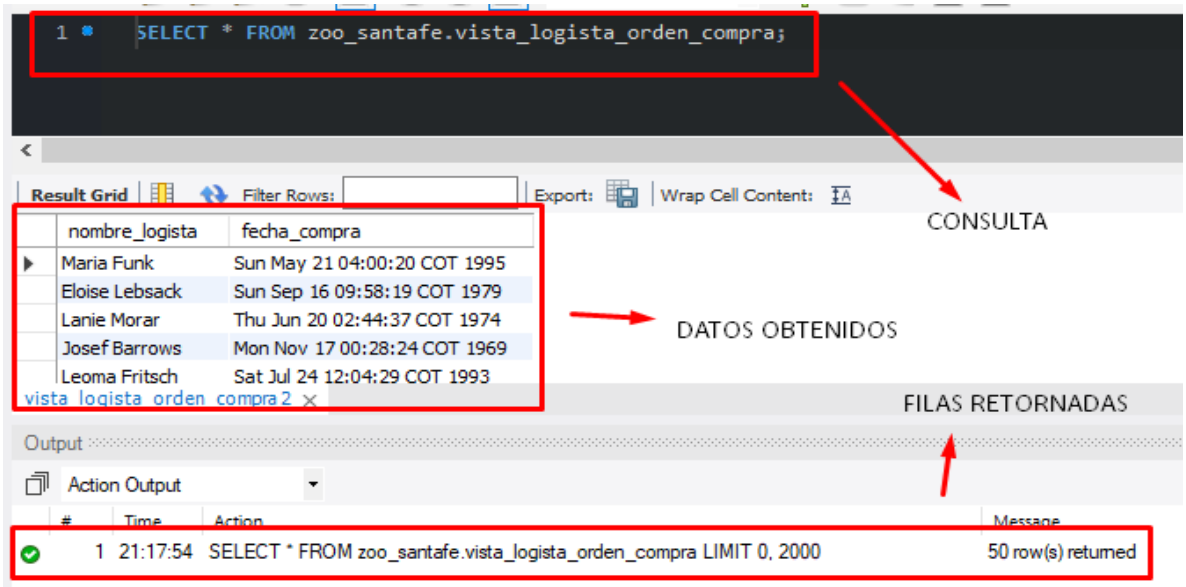
Vista 2: muestra los entrenadores y sus animales asignados lo que puede ser útil para asegurarse de que cada animal esté siendo “atendido” adecuadamente.



vista 3: muestra las dietas y sus alimentos correspondientes lo que puede ser útil para crear planes de alimentación específicos para cada animal y monitorear el uso de los alimentos en el zoológico.



vista 4: muestra los logistas y las órdenes de compra que han realizado lo que puede ser útil para rastrear el historial de compras y el suministro de alimentos para el zoológico.



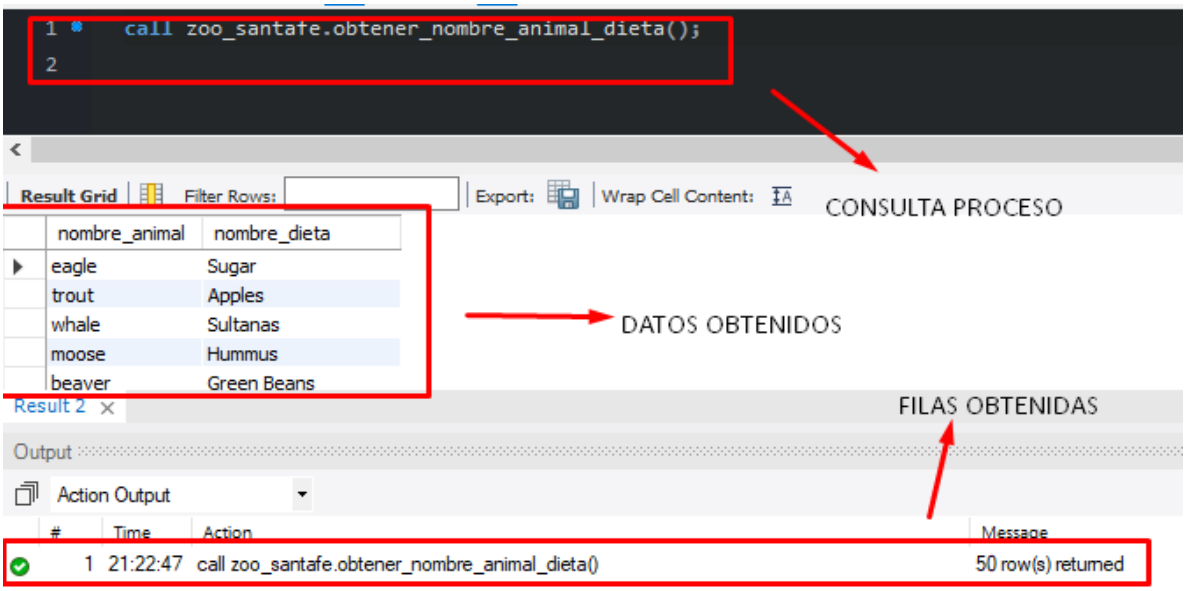
Adjunto imagen de las vistas creadas:

```
# VISTAS GENERADAS

-- 1. Vista de animales y sus dietas correspondientes
CREATE VIEW Vista_Animal_Dieta AS
SELECT Animal.idAnimal, Animal.nombre_animal, Dieta.nombre_dieta, Dieta.dosis_dieta, Dieta.tipo_dieta
FROM Animal
INNER JOIN Dieta ON Animal.idDieta = Dieta.idDieta;


-- 2. Vista de entrenadores y sus animales asignados
CREATE VIEW Vista_Entrenador_Animal AS
SELECT Entrenador.idEntrenador, Entrenador.nombre_eentrenador, Animal.idAnimal, Animal.nombre_animal
FROM Entrenador
INNER JOIN Animal_Entrenador ON Entrenador.idEntrenador = Animal_Entrenador.idEntrenador
INNER JOIN Animal ON Animal_Entrenador.idAnimal = Animal.idAnimal;


-- 3. Vista de dietas y sus alimentos correspondientes
CREATE VIEW Vista_Dieta_Alimento AS
SELECT d.nombre_dieta, a.nombre_alimento, a.tipo_alimento, a.dosis_alimento
FROM Dieta d
INNER JOIN Dieta_Alimento da ON d.idDieta = da.idDieta
INNER JOIN Alimento a ON da.idAlimento = a.idAlimento;

-- 4. Vista de los logistas y las órdenes de compra que han realizado
CREATE VIEW vista_Logista_Orden_Compra AS
SELECT l.nombre_logista, o.fecha_compra
FROM logista l
INNER JOIN Orden_compra o ON l.idlogista = o.idlogista;
```

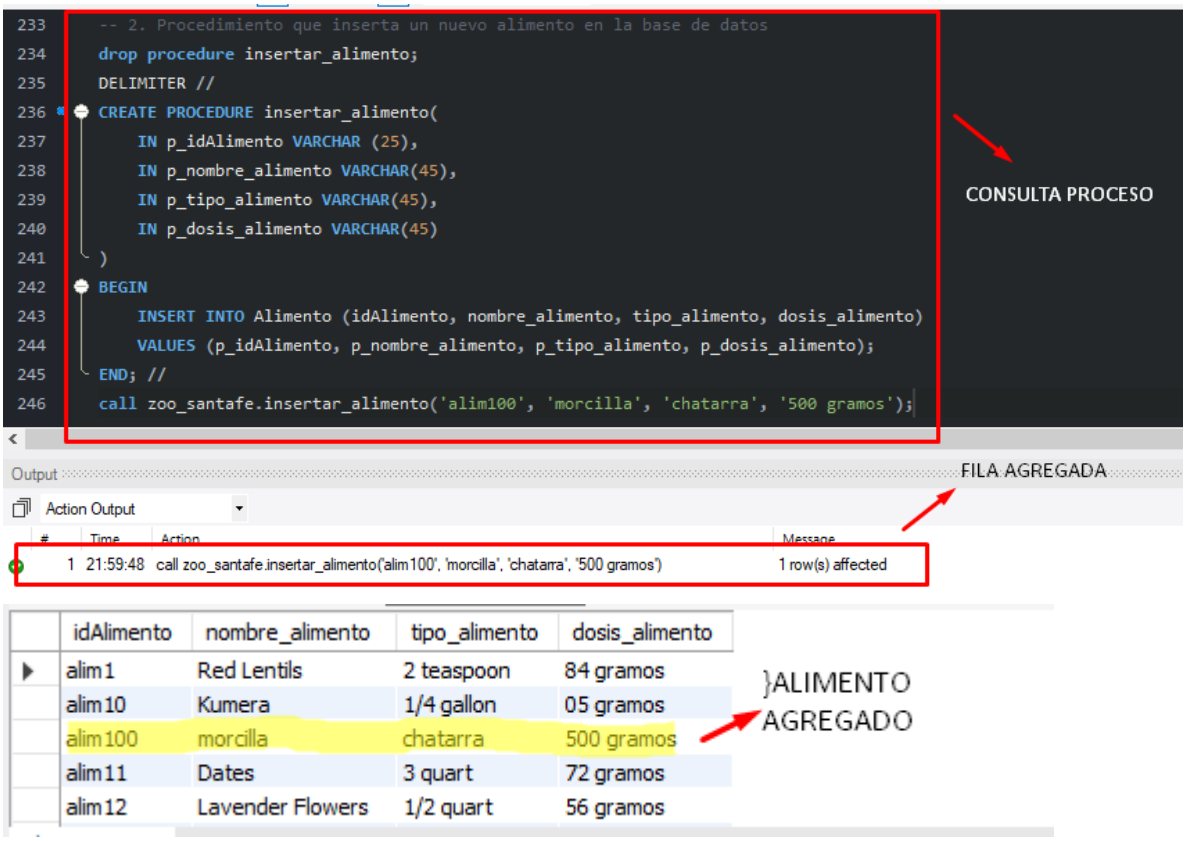
- **Generar al menos 4 procedimientos almacenados.**

R/: Se genera 4 procedimientos que hacen lo siguiente:

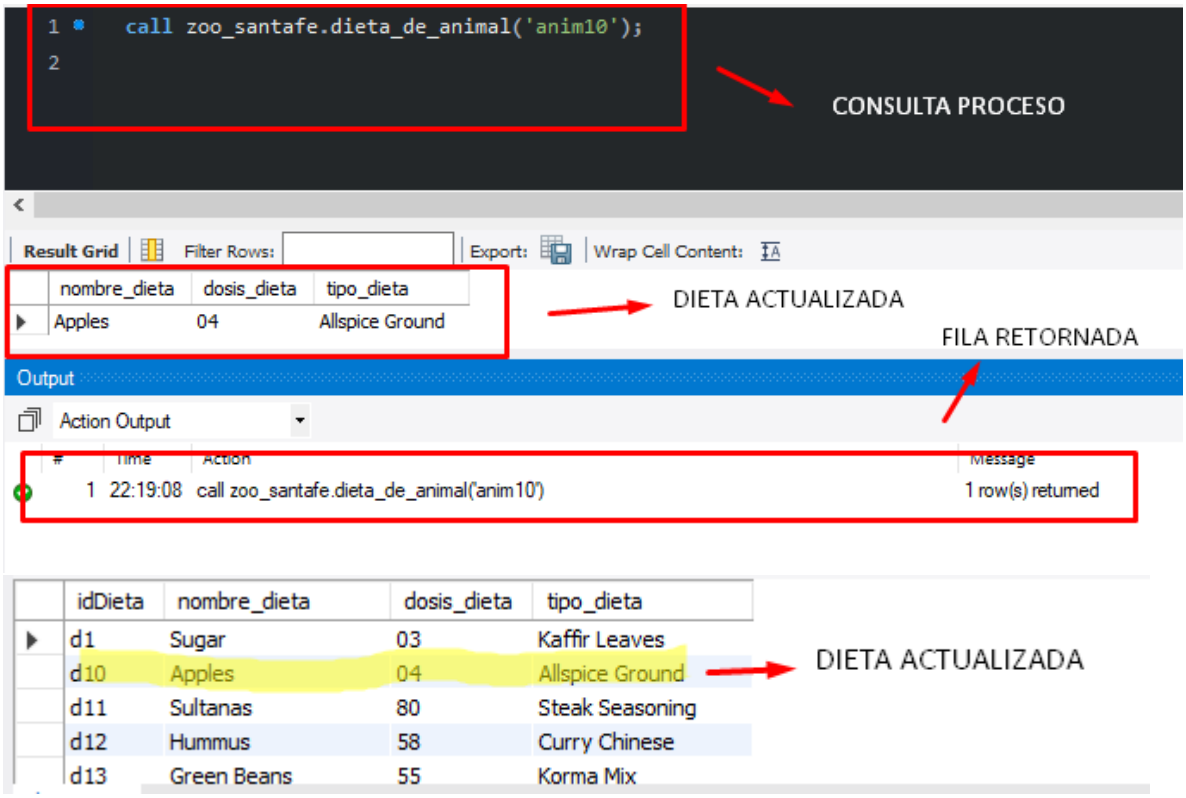
procedimiento 1: obtiene el nombre del animal y su dieta.



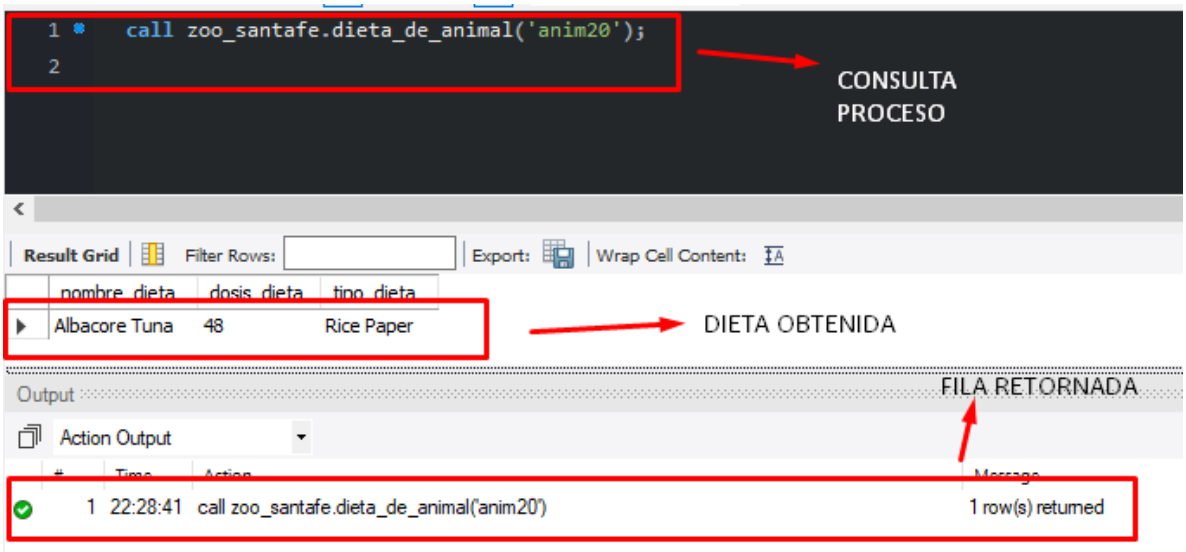
procedimiento 2: inserta un nuevo alimento en la base de datos.



procedimiento 3: actualiza la dieta asociada a un animal.



procedimiento 4: obtiene la dieta de un animal dado su ID.



Adjunto imagen de los procedimientos creados:

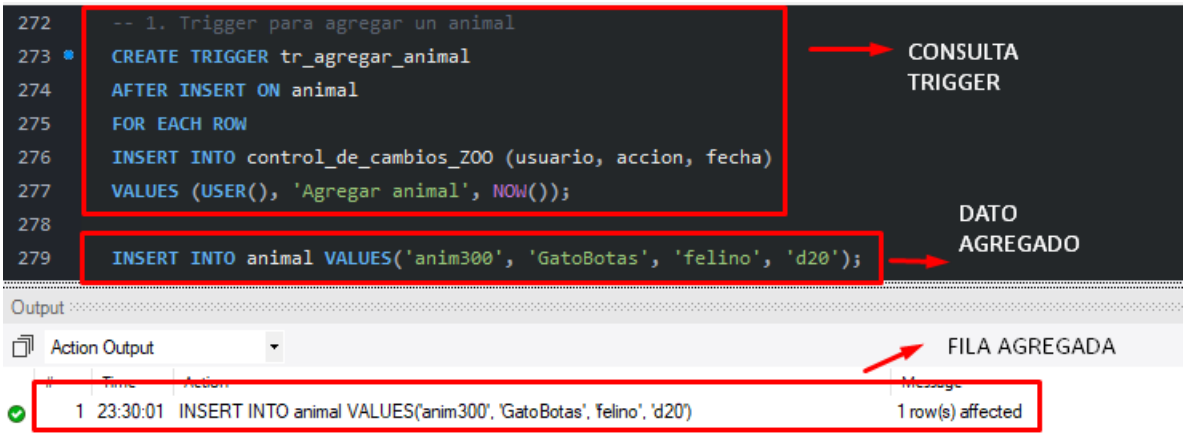
```
221  # PROCEDIMIENTOS GENERADOS
222
223  -- 1. Procedimiento para obtener el nombre del animal y su dieta
224  DELIMITER //
225  CREATE PROCEDURE obtener_nombre_animal_dieta()
226  BEGIN
227      SELECT Animal.nombre_animal, Dieta.nombre_dieta
228      FROM Animal
229      INNER JOIN Dieta ON Animal.idDieta = Dieta.idDieta;
230  END; //
231
232  -- 2. Procedimiento que inserta un nuevo alimento en la base de datos
233  DELIMITER //
234  CREATE PROCEDURE insertar_alimento(
235      IN p_nombre_alimento VARCHAR(45),
236      IN p_tipo_alimento VARCHAR(45),
237      IN p_dosis_alimento VARCHAR(45)
238  )
239  BEGIN
240      INSERT INTO Alimento (nombre_alimento, tipo_alimento, dosis_alimento)
241      VALUES (p_nombre_alimento, p_tipo_alimento, p_dosis_alimento);
242  END; //
243
244  -- 3. Procedimiento que actualiza la dieta asociada a un animal
245  DELIMITER //
246  CREATE PROCEDURE actualizar_dieta_animal(
247      IN p_idAnimal VARCHAR(25),
248      IN p_idDieta VARCHAR(25)
249  )
250  BEGIN
251      UPDATE Animal
252      SET idDieta = p_idDieta
253      WHERE idAnimal = p_idAnimal;
254  END; //
```

```
256  -- 4. Procedimiento para obtener la dieta de un animal dado su ID:
257  DELIMITER //
258  CREATE PROCEDURE dieta_de_animal(IN animal_id VARCHAR(25))
259  BEGIN
260      SELECT d.nombre_dieta, d.dosis_dieta, d.tipo_dieta
261      FROM Dieta d
262      INNER JOIN Animal a ON d.idDieta = a.idDieta
263      WHERE a.idAnimal = animal_id;
264  END; //
```

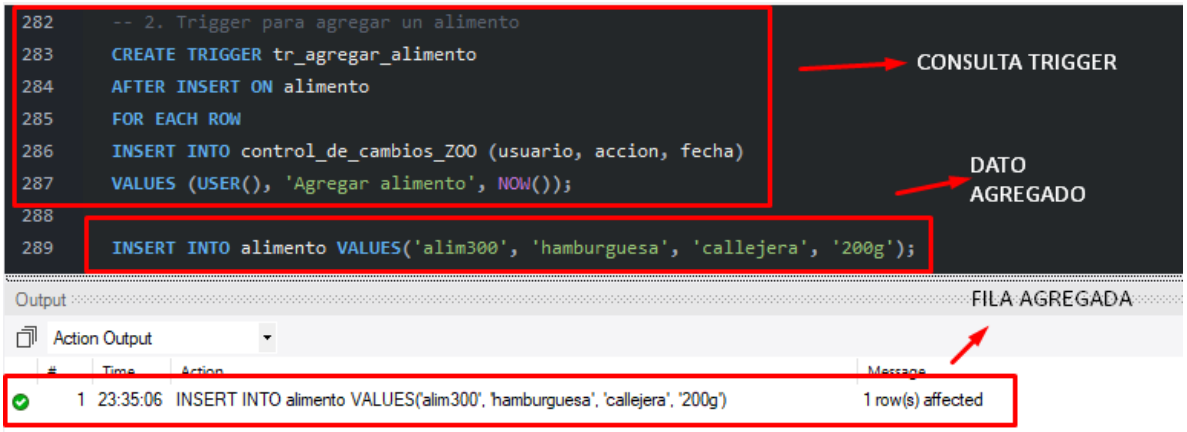
- **Generar al menos 4 triggers**

R/: Se genera 4 triggers que hacen lo siguiente:

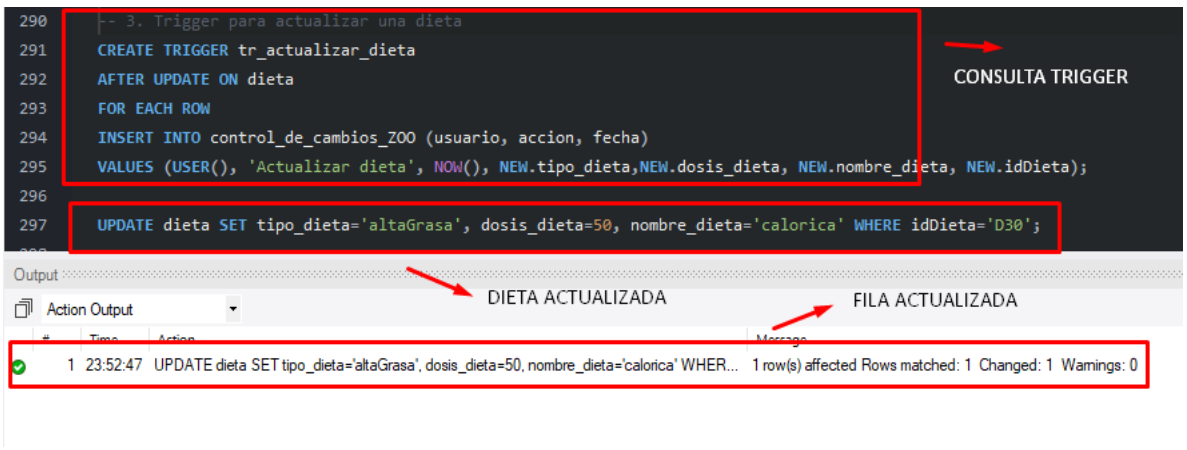
trigger 1: agrega un animal a la base de datos.



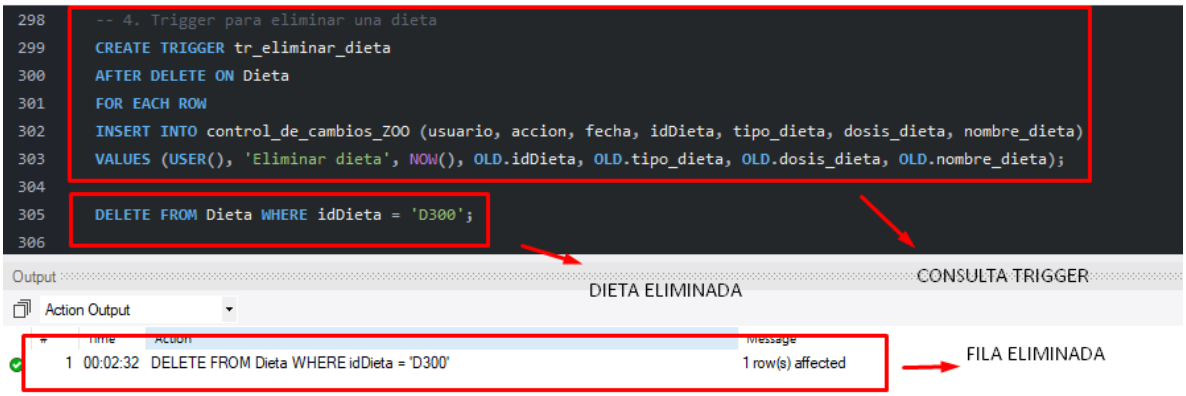
trigger 2: agrega un alimento a la base de datos.



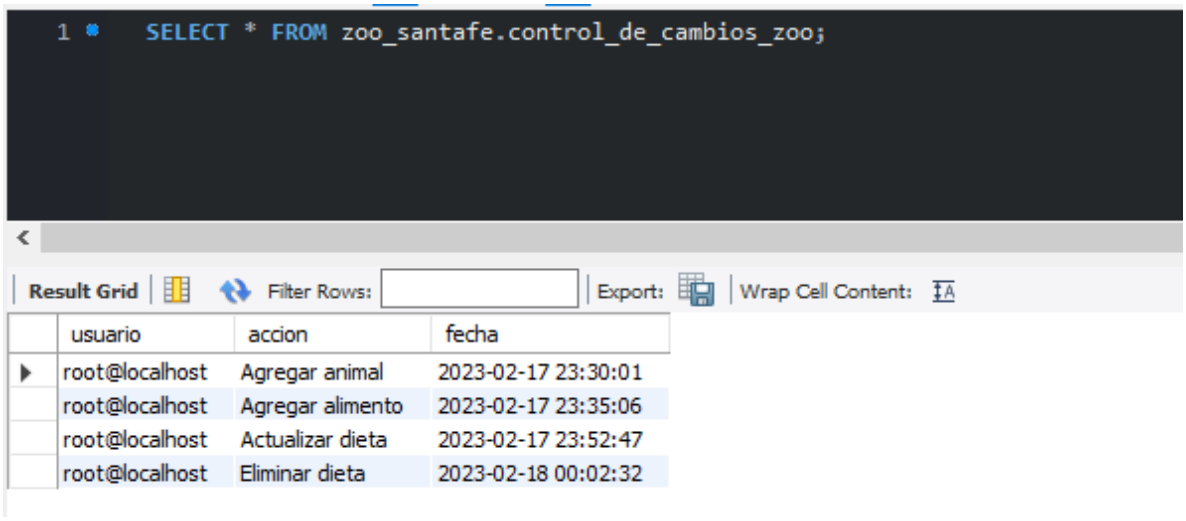
trigger 3: actualiza una dieta de la base de datos.



trigger 4: elimina una dieta de la base de datos.



Adjunto registro de los movimientos con las acciones de los 4 triggers:



usuario	accion	fecha
root@localhost	Agregar animal	2023-02-17 23:30:01
root@localhost	Agregar alimento	2023-02-17 23:35:06
root@localhost	Actualizar dieta	2023-02-17 23:52:47
root@localhost	Eliminar dieta	2023-02-18 00:02:32

- Poblar la base de datos (50 registros por tabla) utilizando una conexión desde Java.

R/: se procede a mostrar la base de datos poblada a través de JAVA, adjuntaré el método que se ejecuta y la evidencia en workbench.

Método para la tabla alimentos:

```
private static void insertarAlimento() {  Complexity is 5 Everything is cool!
    for (int i = 1; i <= 50; i++) {
        Alimento alimento = new Alimento();
        alimento.setIdAlimento("alim" + i);
        alimento.setNombre_alimento(faker.food().ingredient());
        alimento.setTipo_alimento(faker.food().measurement());
        alimento.setDosis_alimento(faker.number().digits( count: 2) + " gramos");
        mySqlOperation.setSqlStatement("insert into alimento values('"
                + alimento.getIdAlimento() + "','" + alimento.getNombre_alimento()
                + "','" + alimento.getTipo_alimento() + "','" + alimento.getDosis_alimento()
                + "');");
        mySqlOperation.executeSqlStatementVoid();
    }
}
```

```
1 •   SELECT * FROM zoo_santafe.alimento;
```

CONSULTA

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

idAlimento	nombre_alimento	tipo_alimento	dosis_alimento
alim1	Red Lentils	2 teaspoon	84 gramos
alim10	Kumera	1/4 gallon	05 gramos
alim11	Dates	3 quart	72 gramos
alim12	Lavender Flowers	1/2 quart	56 gramos
alim13	Purple Rice	1/2 teaspoon	24 gramos
alim14	Pineapple	1/2 teaspoon	78 gramos
alim15	Vinegar	1/2 quart	31 gramos
alim16	Melon	3 tablespoon	41 gramos
alim17	Spelt	1/2 quart	71 gramos
alim18	Chestnut	1/2 gallon	06 gramos
alim19	Coconut Oil	1/4 quart	24 gramos

DATOS POBLADOS

FILAS AGREGADAS

alimento 1

Output

Action Output

#	Time	Action	Message
1	18:32:47	SELECT * FROM zoo_santafe.dieta LIMIT 0, 2000	50 row(s) returned

Método para poblar la tabla dieta:

```
private static void insertarDieta() {  Complexity is 4 Everything is cool!
    for (int i = 1; i <= 50; i++) {
        Dieta dieta = new Dieta();
        dieta.setIdDieta("d" + i);
        dieta.setNombre_dieta(faker.food().ingredient());
        dieta.setDosis_dieta(faker.number().digits( count: 2));
        dieta.setTipo_dieta(faker.food().spice());
        mySqlOperation.setSqlStatement("insert into dieta values('" + dieta.getIdDieta()
                + "', '" + dieta.getNombre_dieta() + "', '" + dieta.getDosis_dieta() + "', '"
                + dieta.getTipo_dieta() + "');");
        mySqlOperation.executeSqlStatementVoid();
    }
}
```

```
1 •   SELECT * FROM zoo_santafe.dieta;
```

CONSULTA

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

idDieta	nombre_dieta	dosis_dieta	tipo_dieta
d1	Sugar	03	Kaffir Leaves
d10	Apples	04	Allspice Ground
d11	Sultanas	80	Steak Seasoning
d12	Hummus	58	Curry Chinese
d13	Green Beans	55	Korma Mix
d14	Jerusalem Artichoke	63	Chilli Ground
d15	Pandanus Leaves	19	Mustard Seed Black
d16	Asian Greens	66	Allspice Whole
d17	Oatmeal	49	Curry Mild
d18	Liver	38	Vanilla Bean

DATOS POBLADOS

FILAS AGREGADAS

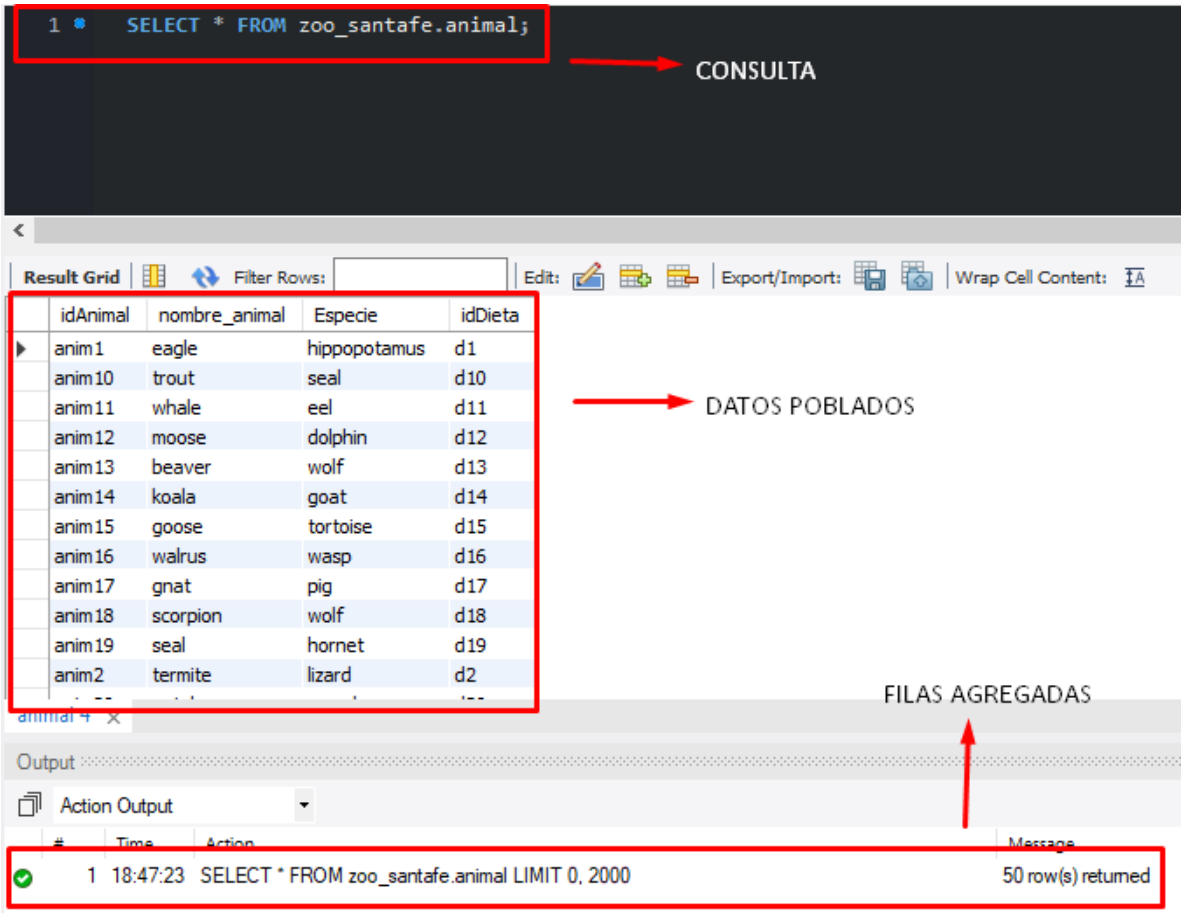
Output

Action Output

#	Time	Action	Message
1	18:32:47	SELECT * FROM zoo_santafe.dieta LIMIT 0, 2000	50 row(s) returned

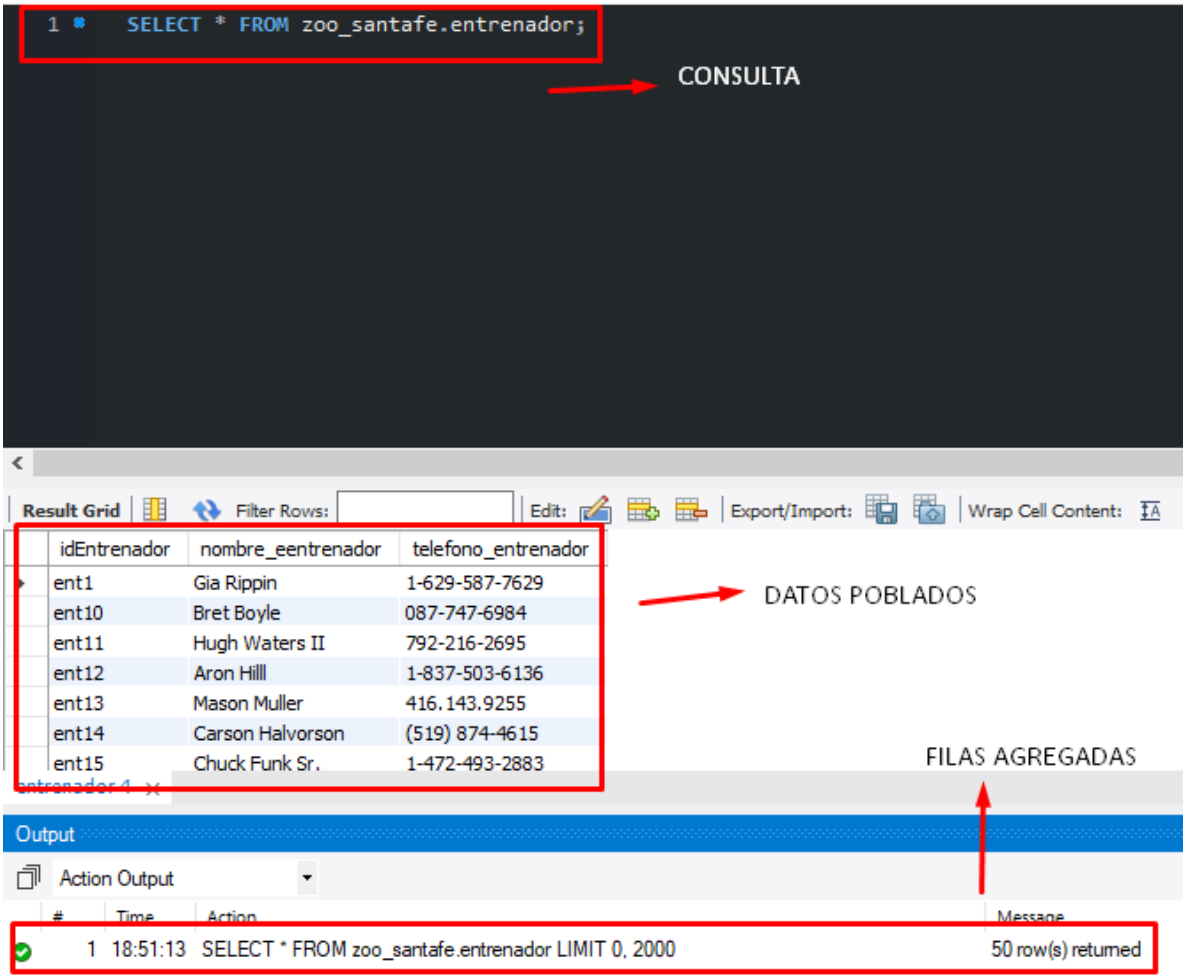
Método para poblar la tabla animales:

```java
private static void insertarAnimales() {  Complexity is 5 Everything is cool!
    for (int i = 1; i <= 50; i++) {
        Animal animal = new Animal();
        animal.setIdAnimal("anim" + i);
        animal.setNombre_animal(faker.animal().name());
        animal.setEspecie(faker.animal().name());
        animal.setIdDieta("d" + i);
        mySqlOperation.setSqlStatement("insert into animal values('" + animal.getIdAnimal()
                + "','" + animal.getNombre_animal()
                + "','" + animal.getEspecie() + "','" + animal.getIdDieta() + "');");
        mySqlOperation.executeSqlStatementVoid();
    }
}
```



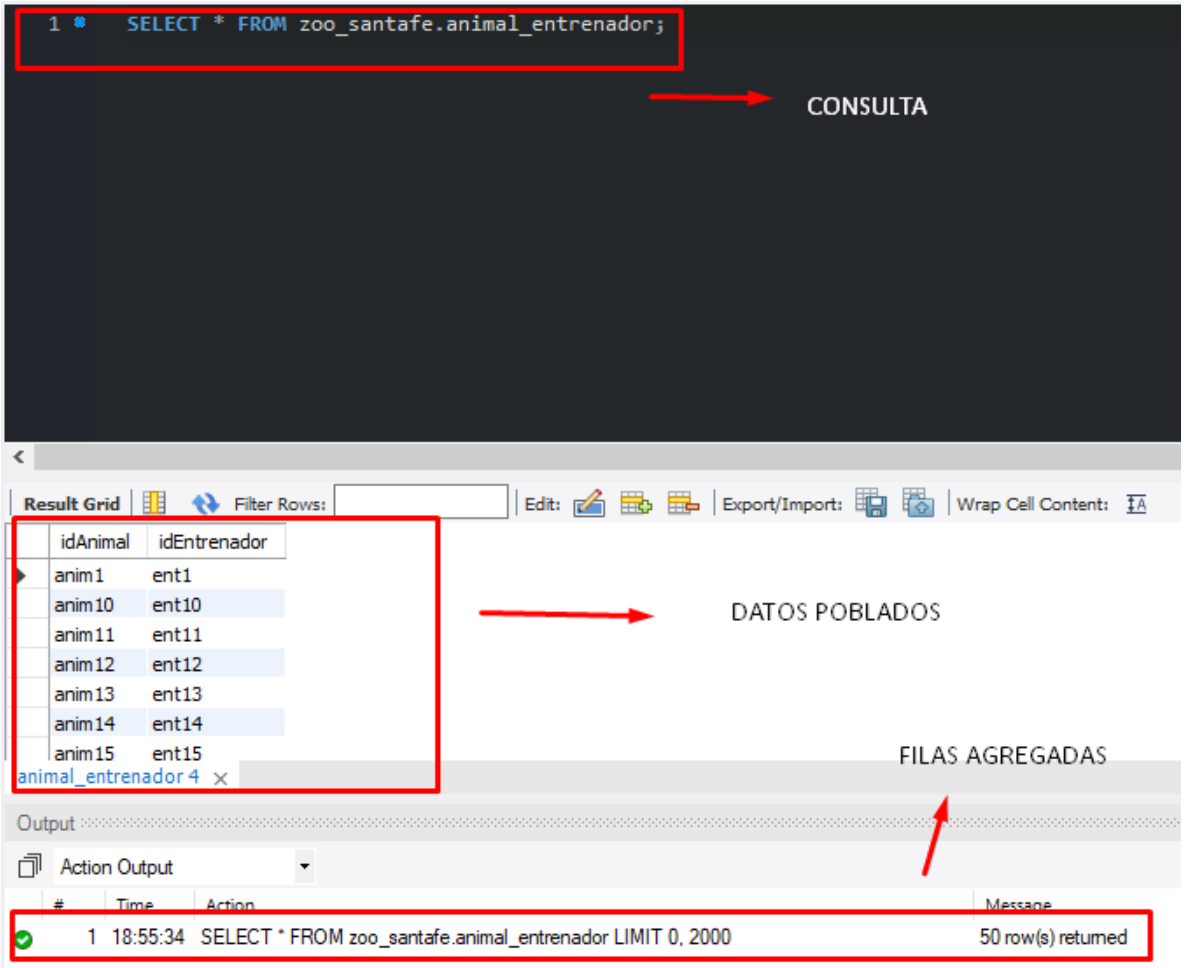
Método para poblar la tabla entrenador:

```java
private static void insertarEntrenador() {  Complexity is 4 Everything is cool!
    for (int i = 1; i <= 50; i++) {
        Entrenador entrenador = new Entrenador();
        entrenador.setIdEntrenador("ent" + i);
        entrenador.setNombre_eentrenador(faker.name().fullName());
        entrenador.setTelefono_entrenador(faker.phoneNumber().cellPhone());
        mySqlOperation.setSqlStatement("insert into entrenador values('" +
                entrenador.getIdEntrenador() + "','" + entrenador.getNombre_eentrenador()
                + "','" + entrenador.getTelefono_entrenador() + "');");
        mySqlOperation.executeSqlStatementVoid();
    }
}
```



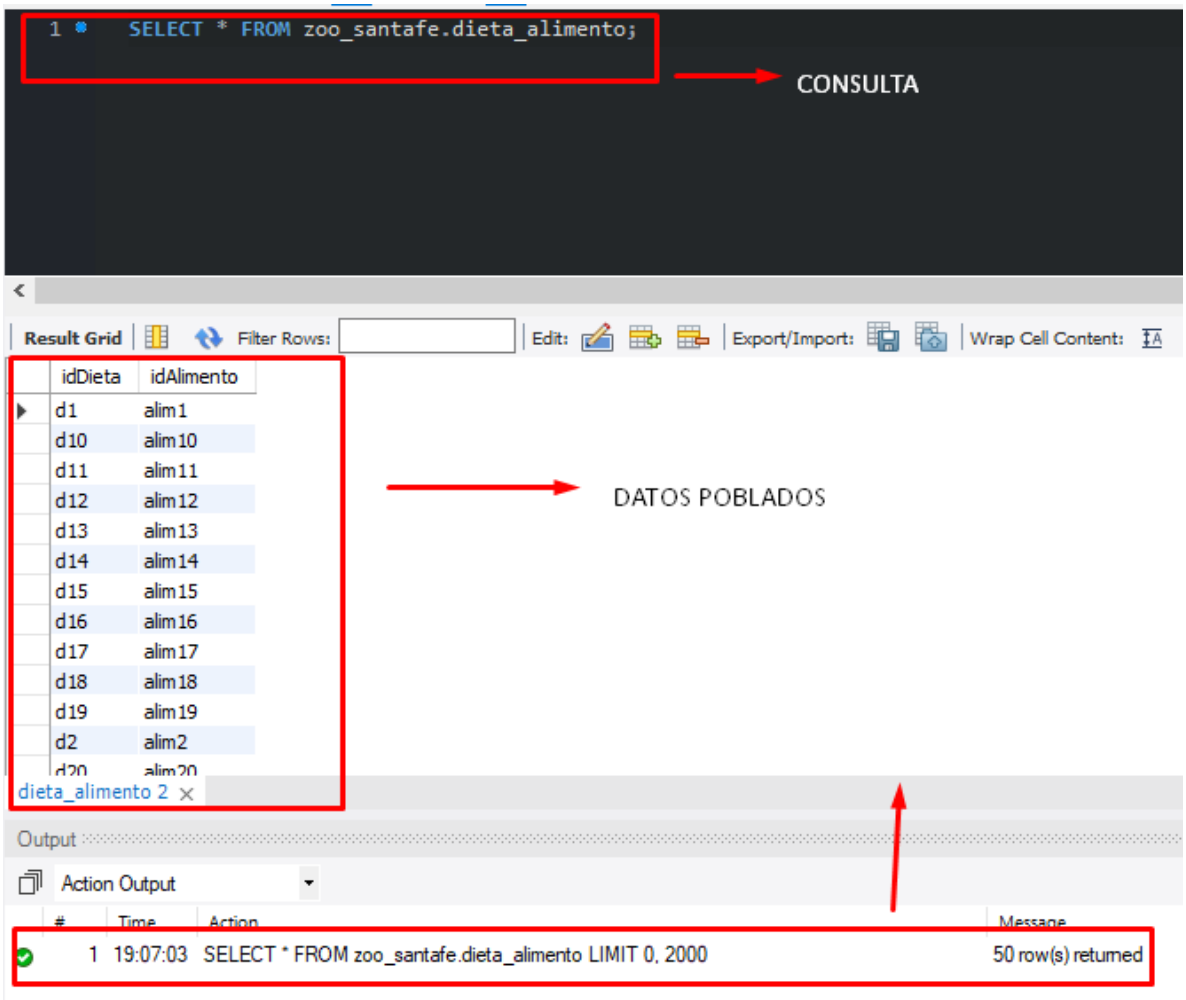
Método para poblar la tabla animal_entrenador:

```
1 usage  James Muñoz *
private static void insertarAnimalEntrenador() {  Complexity is 5 Everything is cool!
    for (int i = 1; i <= 50; i++) {
        Animal_entrenador animal_entrenador = new Animal_entrenador();
        animal_entrenador.setIdAnimal("anim" + i);
        animal_entrenador.setIdEntrenador("ent" + i);
        mySqlOperation.setSqlStatement("insert into animal_entrenador values('"
                + animal_entrenador.getIdAnimal() + "','" + animal_entrenador.getIdEntrenador() + "');");
        mySqlOperation.executeSqlStatementVoid();
    }
}
```



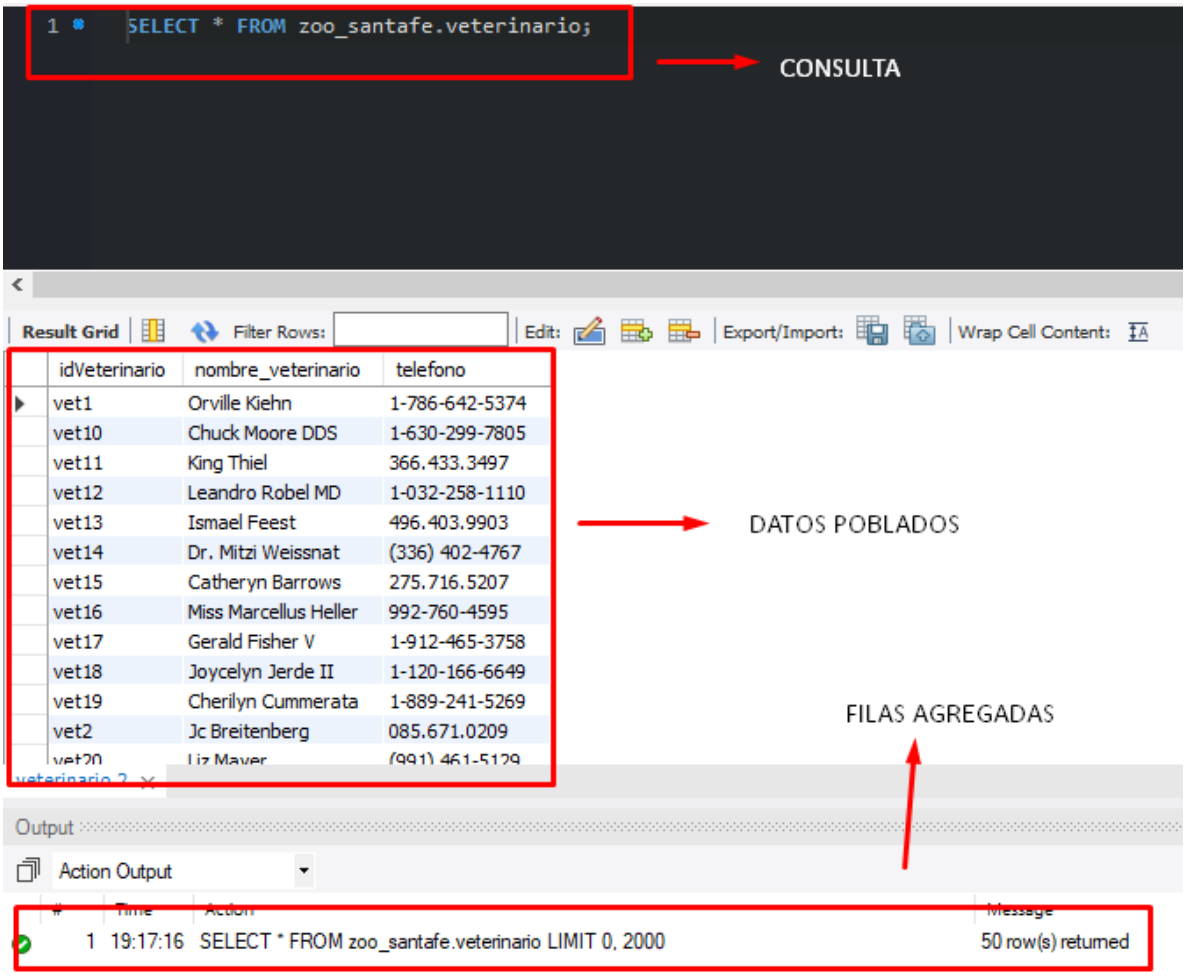
Método para poblar la tabla Dieta_alimento:

```
private static void insertarDietaAlimento() {  Complexity is 5 Everything is cool!
    for (int i = 1; i <= 50; i++) {
        Dieta_alimento dieta_alimento = new Dieta_alimento();
        dieta_alimento.setIdDieta("d" + i);
        dieta_alimento.setIdAlimento("alim" + i);
        mySqlOperation.setSqlStatement("insert into dieta_alimento values('"
                + dieta_alimento.getIdDieta() + "','" + dieta_alimento.getIdAlimento() + "');");
        mySqlOperation.executeSqlStatementVoid();
    }
}
```



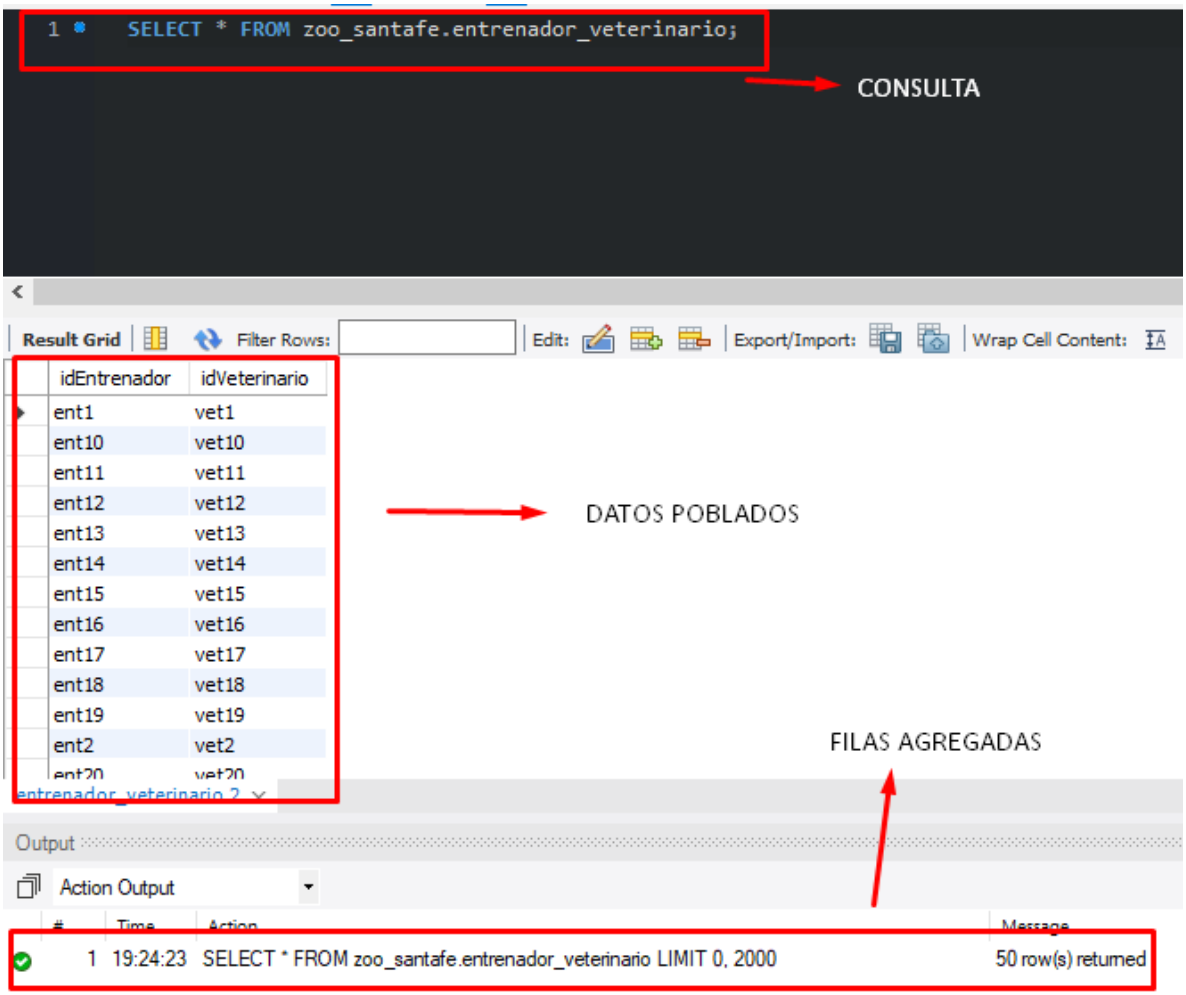
Método para poblar la tabla veterinario:

```java
private static void insertarVeterinario() {  Complexity is 4 Everything is cool!
    for (int i = 1; i <= 50; i++) {
        Veterinario veterinario = new Veterinario();
        veterinario.setIdVeterinario("vet" + i);
        veterinario.setNombre_veterinario(faker.name().fullName());
        veterinario.setTelefono(faker.phoneNumber().cellPhone());
        mySqlOperation.setSqlStatement("insert into veterinario values('" +
                veterinario.getIdVeterinario() + "','" + veterinario.getNombre_veterinario()
                + "','" + veterinario.getTelefono() + "');");
        mySqlOperation.executeSqlStatementVoid();
    }
}
```



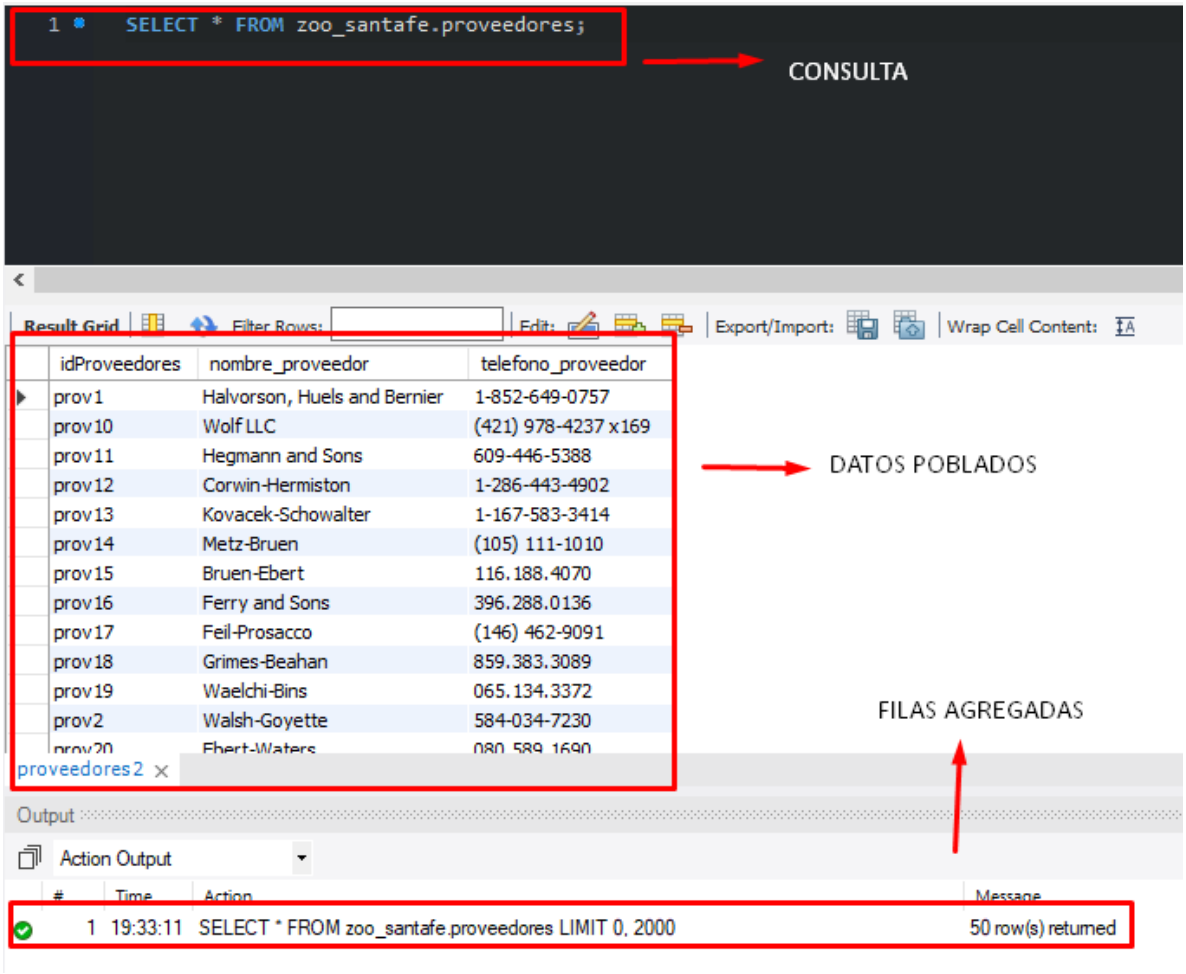
Método para poblar la tabla Entrenador_veterinario:

```java
private static void insertarEntrenadorVeterinario() {  Complexity is 5 Everything is cool!
    for (int i = 1; i <= 50; i++) {
        Entrenador_veterinario entrenador_veterinario = new Entrenador_veterinario();
        entrenador_veterinario.setIdEntrenador("ent" + i);
        entrenador_veterinario.setIdVeterinario("vet" + i);
        mySqlOperation.setSqlStatement("insert into entrenador_veterinario values('"
                + entrenador_veterinario.getIdEntrenador() + "','" +
                entrenador_veterinario.getIdVeterinario() + "');");
        mySqlOperation.executeSqlStatementVoid();
    }
}
```



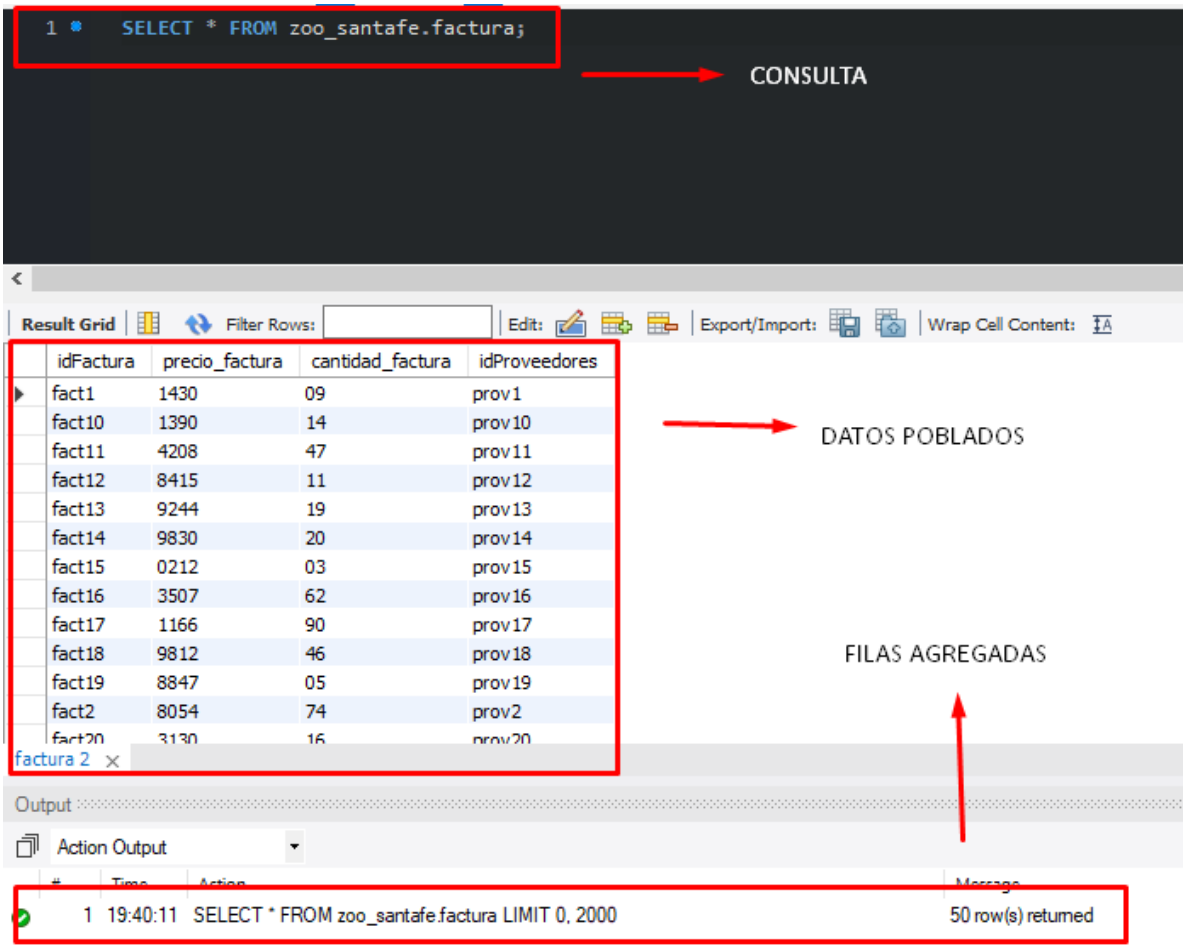
Método para poblar la tabla proveedores:

```java
private static void insertarProveedor() {  Complexity is 4 Everything is cool!
    for (int i = 1; i <= 50; i++) {
        Proveedores proveedor = new Proveedores();
        proveedor.setIdProveedores("prov" + i);
        proveedor.setNombre_proveedor(faker.company().name());
        proveedor.setTelefono_proveedor(faker.phoneNumber().phoneNumber());
        mySqlOperation.setSqlStatement("INSERT INTO proveedores VALUES('" +
                proveedor.getIdProveedores() + "','" + proveedor.getNombre_proveedor()
                + "','" + proveedor.getTelefono_proveedor() + "');");
        mySqlOperation.executeSqlStatementVoid();
    }
}
```



Método para poblar la tabla Factura:

```java
private static void insertarFactura() {  Complexity is 5 Everything is cool!
    for (int i = 1; i <= 50; i++) {
        Factura factura = new Factura();
        factura.setIdFactura("fact" + i);
        factura.setPrecio_factura(faker.number().digits( count: 4));
        factura.setCantidad_factura(faker.number().digits( count: 2));
        factura.setIdProveedores("prov" + i);
        mySqlOperation.setSqlStatement("insert into factura values('" + factura.getIdFactura()
                + "','" + factura.getPrecio_factura() + "','" + factura.getCantidad_factura()
                + "','" + factura.getIdProveedores() + "');");
        mySqlOperation.executeSqlStatementVoid();
    }
}
```



Método para poblar la tabla Informe:

```
private static void insertarInforme() {  Complexity is 6 It's time to do something...
    for (int i = 1; i <= 50; i++) {
        Informe informe = new Informe();
        informe.setIdInforme("inf" + i);
        informe.setPeso_animal(String.valueOf(faker.number().randomDouble( maxNumberOfDecimals: 1,  min: 1,  max: 500)));
        informe.setEdad_animal(String.valueOf(faker.number().numberBetween(1, 20)));
        informe.setIdAnimal("anim" + i);
        informe.setIdEntrenador("ent" + i);
        mySqlOperation.setSqlStatement("insert into informe values('" +
                informe.getIdInforme() + "','" + informe.getPeso_animal()
                + "','" + informe.getEdad_animal() + "','" + informe.getIdAnimal() + "','"
                + informe.getIdEntrenador() + "');");
        mySqlOperation.executeSqlStatementVoid();
    }
}
```

```
1 •   SELECT * FROM zoo_santafe.informe;
```

CONSULTA

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

idInforme	peso_animal	edad_animal	idAnimal	idEntrenador
inf1	267.6	11	anim1	ent1
inf10	436.5	12	anim10	ent10
inf11	9.1	18	anim11	ent11
inf12	196.4	15	anim12	ent12
inf13	368.0	1	anim13	ent13
inf14	450.9	9	anim14	ent14
inf15	260.4	12	anim15	ent15
inf16	62.3	16	anim16	ent16
inf17	62.1	9	anim17	ent17
inf18	145.8	13	anim18	ent18
inf19	173.4	2	anim19	ent19
inf2	54.4	11	anim2	ent2
inf20	470.4	13	anim20	ent20

DATOS POBLADOS

informe 2

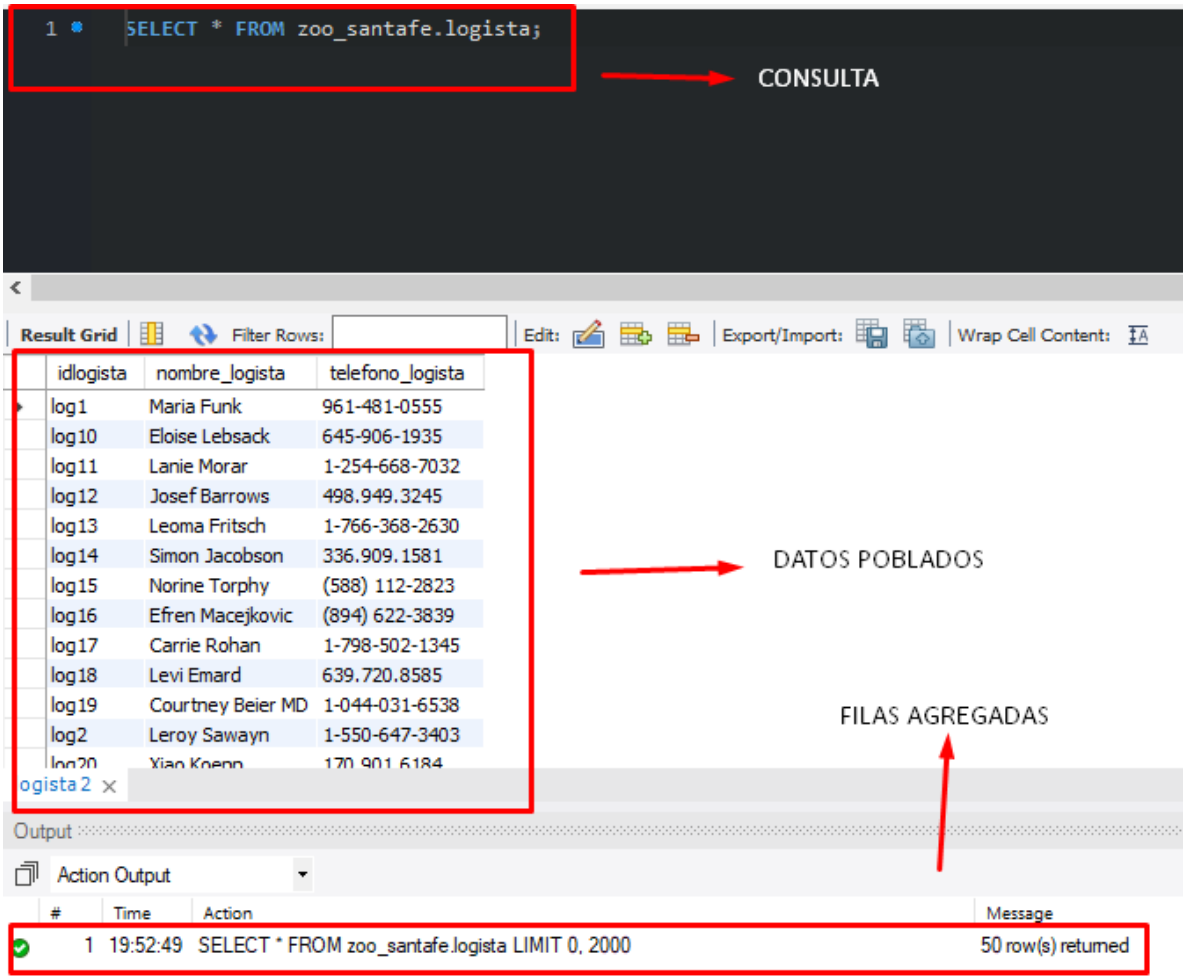
Action Output

#	Time	Action	Message
1	19:48:26	SELECT * FROM zoo_santafe.informe LIMIT 0, 2000	50 row(s) returned

FILAS AGREGADAS

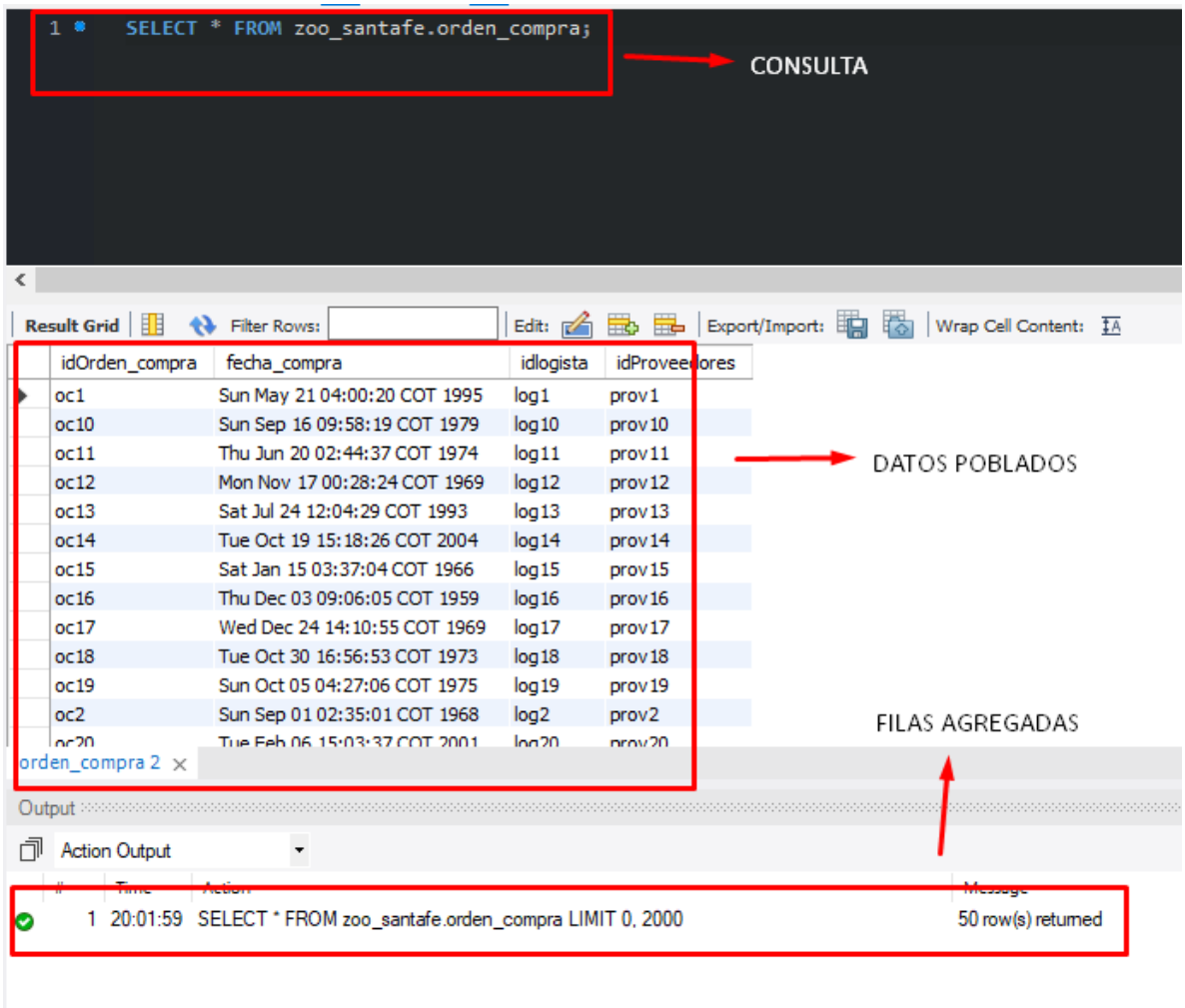
Método para poblar la tabla Logista:

```java
private static void insertarLogista() {  Complexity is 4 Everything is cool!
    for(int i = 1; i <= 50; i++) {
        Logista logista = new Logista();
        logista.setIdlogista("log" + i);
        logista.setNombre_logista(faker.name().fullName());
        logista.setTelefono_logista(faker.phoneNumber().cellPhone());
        mySqlOperation.setSqlStatement("insert into logista values('" +
                logista.getIdlogista() + "','" + logista.getNombre_logista()
                + "','" + logista.getTelefono_logista() + "');");
        mySqlOperation.executeSqlStatementVoid();
    }
}
```



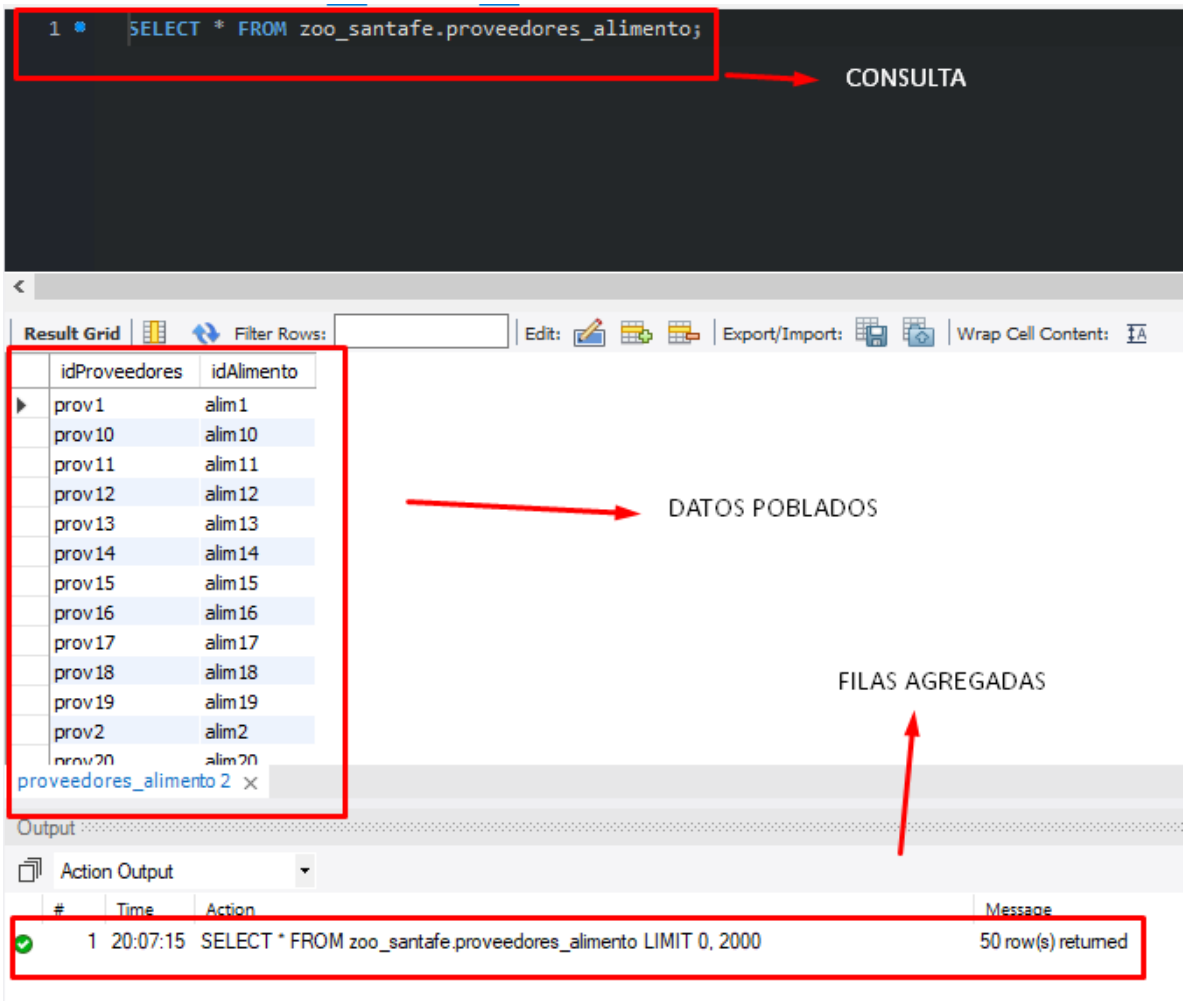
Método para poblar la tabla Orden_compra:

```java
private static void insertarOrdenCompra() {  Complexity is 6 It's time to do something...
    for (int i = 1; i <= 50; i++) {
        Orden_compra orden_compra = new Orden_compra();
        orden_compra.setIdOrden_compra("oc" + i);
        orden_compra.setFecha_compra(String.valueOf(faker.date().birthday()));
        orden_compra.setIdlogista("log" + i);
        orden_compra.setIdProveedores("prov" + i);
        mySqlOperation.setSqlStatement("insert into orden_compra values('"
                + orden_compra.getIdOrden_compra()
                + "','" + orden_compra.getFecha_compra() + "','" +
                orden_compra.getIdlogista() + "','"
                + orden_compra.getIdProveedores() + "');");
        mySqlOperation.executeSqlStatementVoid();
    }
}
```



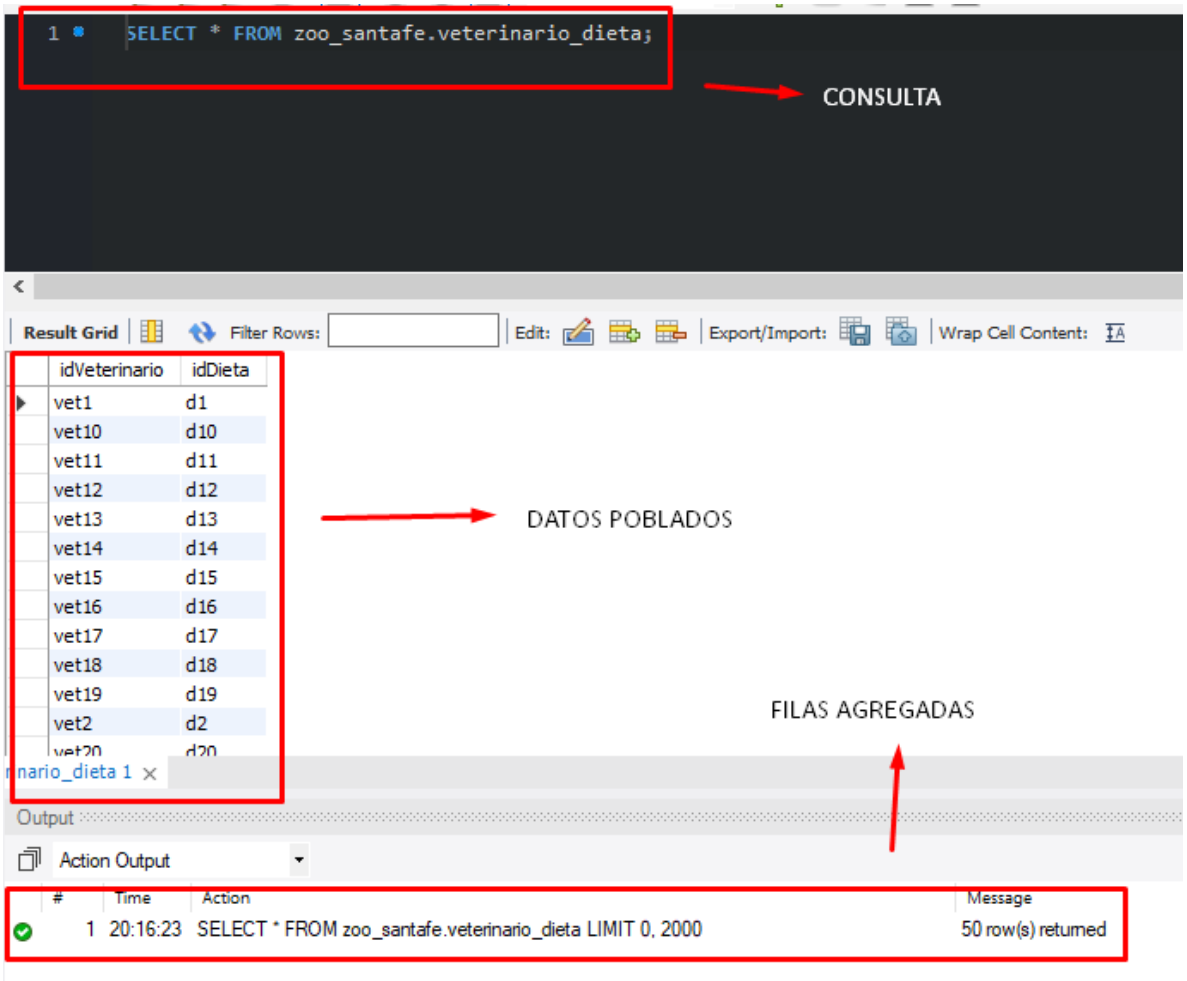
Método para poblar la tabla Proveedores_alimento:

```
1 usage   James Muñoz *
private static void insertarProveedoresAlimento() {   Complexity is 5 Everything is cool!
    for (int i = 1; i <= 50; i++) {
        Proveedores_alimento proveedoresAlimento = new Proveedores_alimento();
        proveedoresAlimento.setIdProveedores("prov" + i);
        proveedoresAlimento.setIdAlimento("alim" + i);
        mySqlOperation.setSqlStatement("insert into proveedores_alimento values('"
                + proveedoresAlimento.getIdProveedores() + "','" +
                proveedoresAlimento.getIdAlimento() + "');");
        mySqlOperation.executeSqlStatementVoid();
    }
}
```



Método para poblar la tabla Veterinario_dieta:

```
1 usage  new *
private static void insertarVeterinarioDieta() {  Complexity is 5 Everything is cool!
    for(int i = 1; i <= 50; i++) {
        Veterinario_dieta veterinarioDieta = new Veterinario_dieta();
        veterinarioDieta.setIdVeterinario("vet" + i);
        veterinarioDieta.setIdDieta("d" + i);
        mySqlOperation.setSqlStatement("insert into veterinario_dieta values('"
                + veterinarioDieta.getIdVeterinario()
                + "','" + veterinarioDieta.getIdDieta() + "');");
        mySqlOperation.executeSqlStatementVoid();
    }
}
```



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

R/: Si estoy conforme, ya que con la base relacional que se usó fue posible alcanzar el objetivo que se requería en el ejercicio sin embargo algunas cosas como el manejo de los datos de las dietas de los animales hubieran podido ser más óptimos en una base de datos no relacional pero aun así se pudo trabajar de manera adecuada con esta base que se utilizó.