

## Almacén Don pepe (Ejercicio C)

En el contexto de la formación inicial de SofkaU se propone realiza el presente reto. A continuación se presenta el paso a paso de la solución obtenida:

### ● Indicar que ejercicio fue asignado

El ejercicio asignado se titula Tienda Virtual Don pepe (Ejercicio C)

### ● Realizar el modelo E-R

Para realizar el modelo entidad relación se realizó una abstracción del enunciado y se identificaron las siguientes entidades, atributos, comportamientos y requerimientos:

**Requerimientos:** Se requiere una base de datos para manejar el almacén de Don Pepe, en la que el cliente pueda realizar pedidos desde su casa.

Las entidades de la base de datos son:

#### Responsable

**-Responsable:** Esta entidad representa a un responsable en el almacén. Esta persona es la encargada de recibir la información de los productos solicitados por el cliente y empacarlos en una cesta. Luego de empacar los productos el responsable en el almacén entrega la cesta al domiciliario para posterior entrega. Los atributos asociados a la entidad responsables son: cedula\_res, nombre\_res y telefono\_res.

**Cardinalidad:** Un responsable en almacén empaqueta muchos pedidos. Un pedido es empacado por solo un responsable.

**Participación:** El pedido puede existir sin el responsable y el responsable existe sin el pedido.

#### Domiciliario

**-Domiciliario:** Esta entidad representa un domiciliario que recibe una cesta de parte del responsable en el almacén con los productos solicitados y posteriormente la lleva a la casa del cliente. La entidad domiciliario tiene los siguientes atributos: cedula\_dom, nombre\_dom y telefono\_dom.

**Cardinalidad:** Un domiciliario puede entregar muchos pedidos. Un pedido puede ser entregado por solo un domiciliario.

Un domiciliario puede usar muchas furgonetas. Una furgoneta puede ser usada solo por un domiciliario.

Un domiciliario puede trabajar en una sola zona, en una zona pueden trabajar muchos domiciliarios.

**Participación:** El domiciliario no puede existir sin la furgoneta y la furgoneta no puede existir sin el domiciliario.

El domiciliario no puede existir sin una zona. Una zona si puede existir sin un domiciliario.

### Cliente

**-Cliente:** Esta entidad representa al cliente que realiza un pedido, el cual tiene asociado una canasta con unos productos específicos. La entidad cliente tiene los siguientes atributos: ID\_cli, cedula\_cli, nombre\_cli, telefono\_cli, direccion\_cli, email\_cli, contraseña\_cli y zona\_cli.

**Cardinalidad:** Un cliente puede realizar muchos pedidos, un pedido puede ser realizado solo por un cliente.

En una zona pueden vivir muchos clientes, un cliente solo puede vivir en una zona.

**Participación:** Un pedido no existe sin un cliente. Un cliente existe sin un pedido.

Una zona existe sin un cliente. Un cliente existe sin una zona.

### Pedido

**-Pedido:** Esta entidad representa un pedido realizado por un cliente. La entidad pedido tiene los siguientes atributos: codigo\_pedido, fecha\_pedido, total\_importe, datos\_pago (número de tarjeta y fecha de caducidad), estado.

**Cardinalidad:** Un pedido puede generar solo una factura y una factura puede ser generada por solo un pedido.

Un pedido solo puede tener una cesta asociada. Una cesta solo puede tener un pedido asociado.

**Participación:** Una factura no existe sin un pedido. Un pedido si existe sin una factura.

Un pedido puede existir sin una cesta. Una cesta no puede existir sin un pedido.

### Cesta

**-Cesta:** Esta entidad representa la cesta de productos que solicita el cliente. En esta entidad se relacionan todos los productos que solicita un cliente en un pedido específico. La entidad cesta tiene los siguientes atributos: ID\_cesta, ID\_producto y cantidad.

**Cardinalidad:** Una cesta puede tener varios productos. Un producto puede ser tenido en muchas cestas.

**Participación:** Una cesta no existe sin un producto. Un producto si existe sin una cesta.

### Producto

**-Producto:** Esta entidad representa los productos que se tienen en la tienda. La entidad producto tiene los siguientes atributos: ID\_producto, nombre\_prod, marca, origen, fotografia, unidades\_disponibles, dimensiones (volumen y peso), valor\_compra y valor\_venta.

**Cardinalidad:** Un producto pertenece a una sola categoría, a una categoría pueden pertenecer muchos productos.

Un producto puede ser proveído por muchos proveedores, un proveedor puede proveer muchos productos.

**Participación:** Un producto no existe sin una categoría, una categoría no existe sin un producto.

Un proveedor existe sin un producto. Un producto no existe sin un proveedor.

### **Furgoneta**

**-Furgoneta:** Esta entidad representa una furgoneta en la cual el domiciliario entrega el pedido. La entidad furgoneta tiene los siguientes atributos: matricula\_fur y estado\_fur.

### **Zona**

**-Zona:** Esta entidad representa una zona de la ciudad. La entidad zona tiene los siguientes atributos: ID\_zona, codigo\_postal y localidad\_zona.

**-Categoría:** Esta entidad representa una categoría a la cual pertenece un producto. Pueden haber diferentes categorías como: carnes, lácteos, granos, cereales, verduras, frutas, tubérculos, entre otros. Por otra parte según la categoría se pueden identificar condiciones de almacenamiento que pueden ser frío, congelado o seco. La entidad categoría tiene los siguientes atributos: ID\_categoria, nombre\_cat, almacenamiento y observaciones.

### **Proveedor**

**-Proveedor:** Esta entidad representa un proveedor que provee a la tienda con los productos. La entidad proveedor tiene los siguientes atributos: ID\_proveedor, nombre\_prov y telefono\_prov.

### **Factura**

**-Factura:** Esta entidad representa la factura que se genera cuando un pedido ha finalizado. La entidad factura tiene los siguientes atributos: ID\_factura y total.

### **Categoría**

**-Categoría:** Esta entidad representa una categoría a la que pertenece cada alimento. La tabla Categoría tiene los siguientes atributos: ID\_categoria, nombre\_cat, almacenamiento y observaciones. Pueden haber distintas categorías:

- A. Carnes.
- B. Granos.
- C. Lácteos.
- D. Verduras.
- E. Frutas.
- F. No perecederos.
- G. Bebidas.

El atributo estado es un atributo derivado que se asignaría por la lógica de la aplicación. Puede tener los siguientes valores:

- A. Creado por el cliente.
- B. En verificación de existencias.
- C. En verificación de responsable de almacén.
- D. En verificación de domiciliario en zona de cliente.
- E. Cancelado por existencias.
- F. En espera a verificación de cliente aceptando nuevas condiciones.
- G. Cancelado por no existencia de domiciliario en la zona.

- H. Empacado por responsable.
- I. Entregado por responsable a domiciliario.
- J. En ruta.
- K. Entregado a cliente.
- L. Pagado por el cliente.
- M. Finalizado.

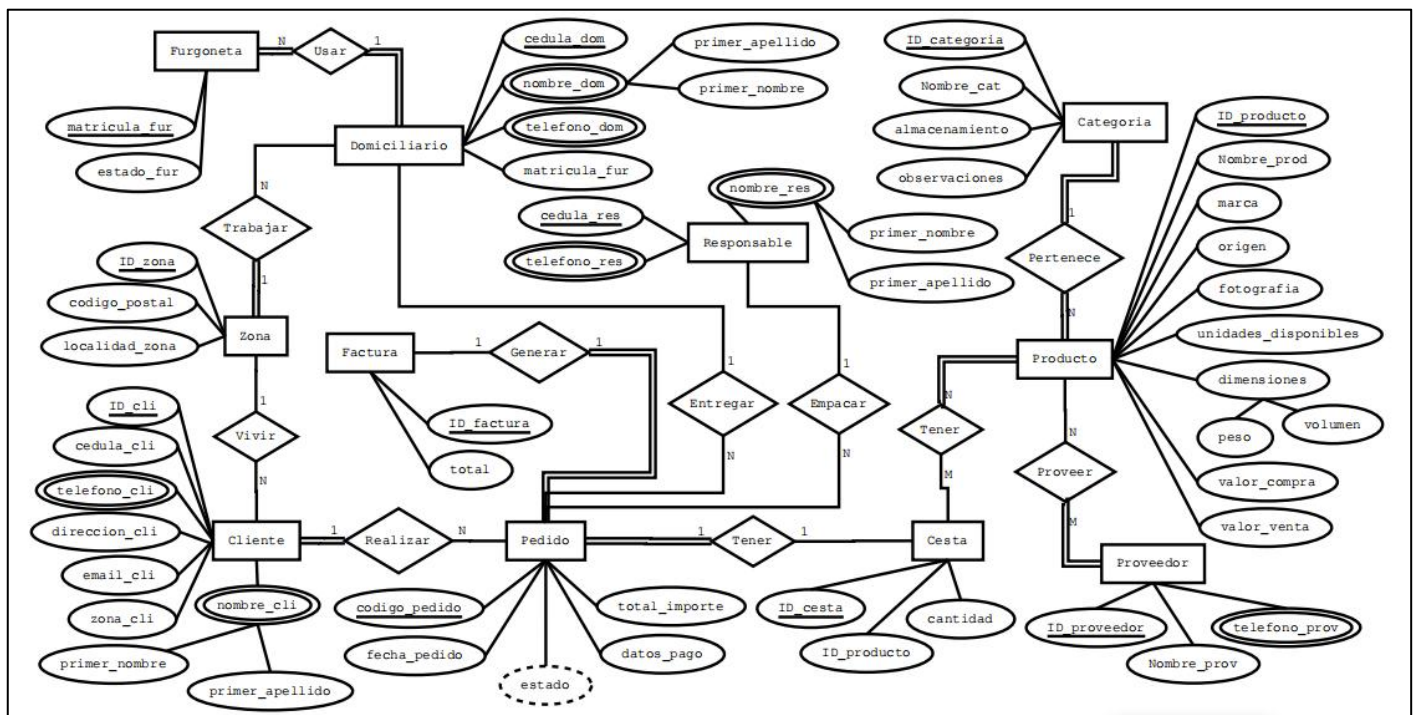
#### Otras consideraciones:

-Los domiciliarios son conocidos de Don Pepe ellos se encargan de llevar los domicilios a los clientes.

-Las furgonetas son propiedad de cada domiciliario, cada domiciliario tiene el deber de reportarle a Don Pepe cualquier novedad con el estado de su furgoneta.

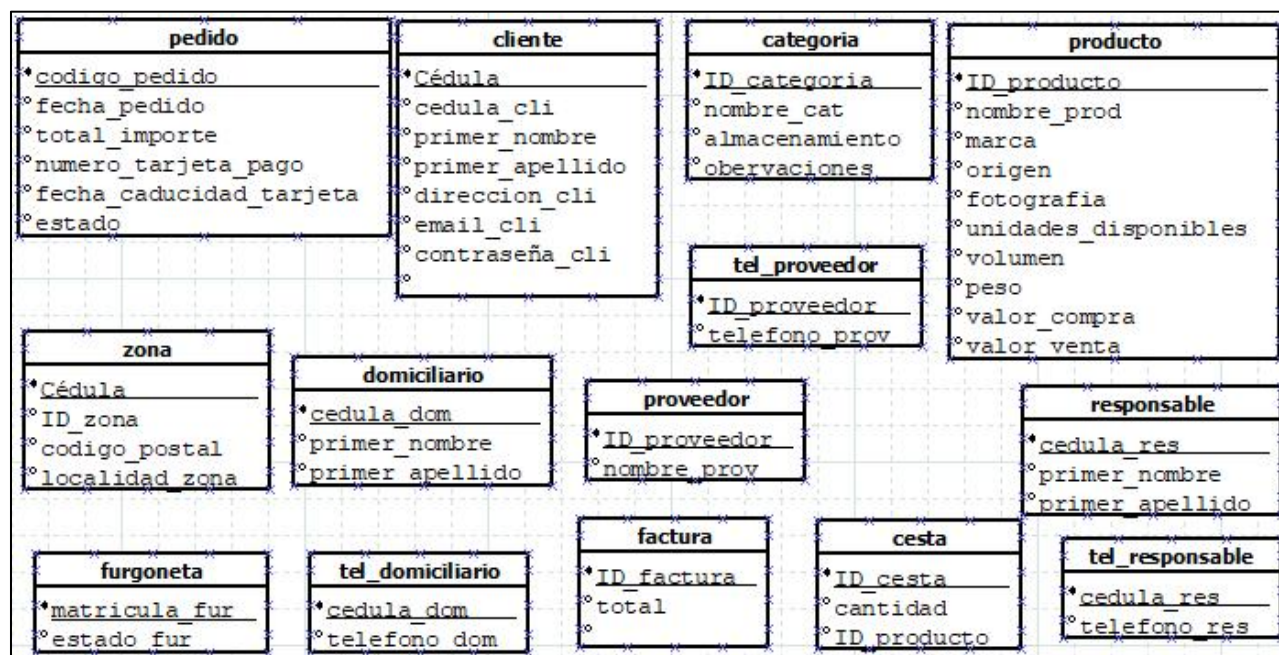
-Un domiciliario no puede realizar domicilios si no tiene furgoneta.

En función de lo anterior se realizó el siguiente diagrama E/R:



## ● Realizar el modelo relacional

En primer lugar se crean las tablas de cada entidad:



Una vez se tienen las tablas se identifican las relaciones según la cardinalidad. Para realizar el modelo relacional se tuvo en cuenta lo siguiente:

-Los atributos asociados a la tabla Responsable son: cedula\_res, primer\_nombre y primer\_apellido. cedula\_res es clave primaria.

-Los atributos asociados a la tabla Tel\_responsable son: cedula\_res y telefono\_res.

-La tabla domiciliario tiene los siguientes atributos: cedula\_dom, primer\_nombre, primer\_apellido y ID\_zona. cedula\_dom es clave primaria, ID\_zona es una llave foránea asociada a la tabla Zona.

-La tabla Telefono\_dom tiene los siguientes atributos: cedula\_dom y telefono\_dom.

-La tabla furgoneta tiene los siguientes atributos: matricula\_fur, estado\_fur y cedula\_dom. matricula\_fur es llave primaria y cedula\_dom es llave foránea relacionada con la tabla Domiciliario.

-La tabla Cliente tiene los siguientes atributos: ID\_cli, cedula\_cli, primer\_nombre, primer\_apellido, direccion\_cli, email\_cli, contraseña\_cli y ID\_zona. ID\_cli es una llave primaria, ID\_zona es llave foránea asociada a la tabla Zona.

-La tabla Cliente\_Pedido tiene los siguientes atributos: ID\_cli\_codigo\_pedido, ID\_cli y codigo\_pedido. ID\_cli\_codigo\_pedido es llave primaria, ID\_cli es llave foránea

relacionada con la tabla Cliente, codigo\_pedido es llave foranea relacionada con la tabla Pedido.

-La tabla Zona tiene los siguientes atributos: ID\_zona, codigo\_postal y localidad\_zona.

-La tabla Pedido tiene los siguientes atributos: codigo\_pedido, fecha\_pedido, total\_importe, número\_tarjeta\_pago, fecha\_caducidad\_tarjeta, estado, cedula\_res, cedula\_dom, ID\_cesta. El atributo codigo\_pedido es llave primaria, el atributo cedula\_dom es llave foranea relacionada con la tabla Domiciliario, el atributo cedula\_res es llave foranea relacionada con la tabla Responsable, el atributo ID\_cesta es llave foranea relacionada con la tabla Cesta.

-La tabla Cesta tiene los siguientes atributos: ID\_cesta, ID\_producto, cantidad, codigo\_pedido. El atributo ID\_cesta es llave primaria, el atributo ID\_producto es llave foranea relacionada con la tabla Producto, el atributo codigo\_pedido es llave foranea a la tabla Pedido.

-La tabla Producto tiene los siguientes atributos: ID\_producto, nombre\_prod, marca, origen, fotografia, unidades\_disponibles, volumen, peso, valor\_compra, valor\_venta, ID\_categoria. ID\_producto es la llave primaria, ID\_categoria es llave foranea asociada a la tabla Categoria.

-La tabla Producto\_Proveedor tiene los siguientes atributos: ID\_producto\_ID\_proveedor, ID\_producto, ID\_proveedor. ID\_producto\_ID\_proveedor es llave primaria, ID\_producto es llave foranea relacionada con la tabla Producto, ID\_proveedor es llave foranea relacionada con la tabla Proveedor.

-La tabla Categoria tiene los siguientes atributos: ID\_categoria, nombre\_cat, almacenamiento y observaciones.

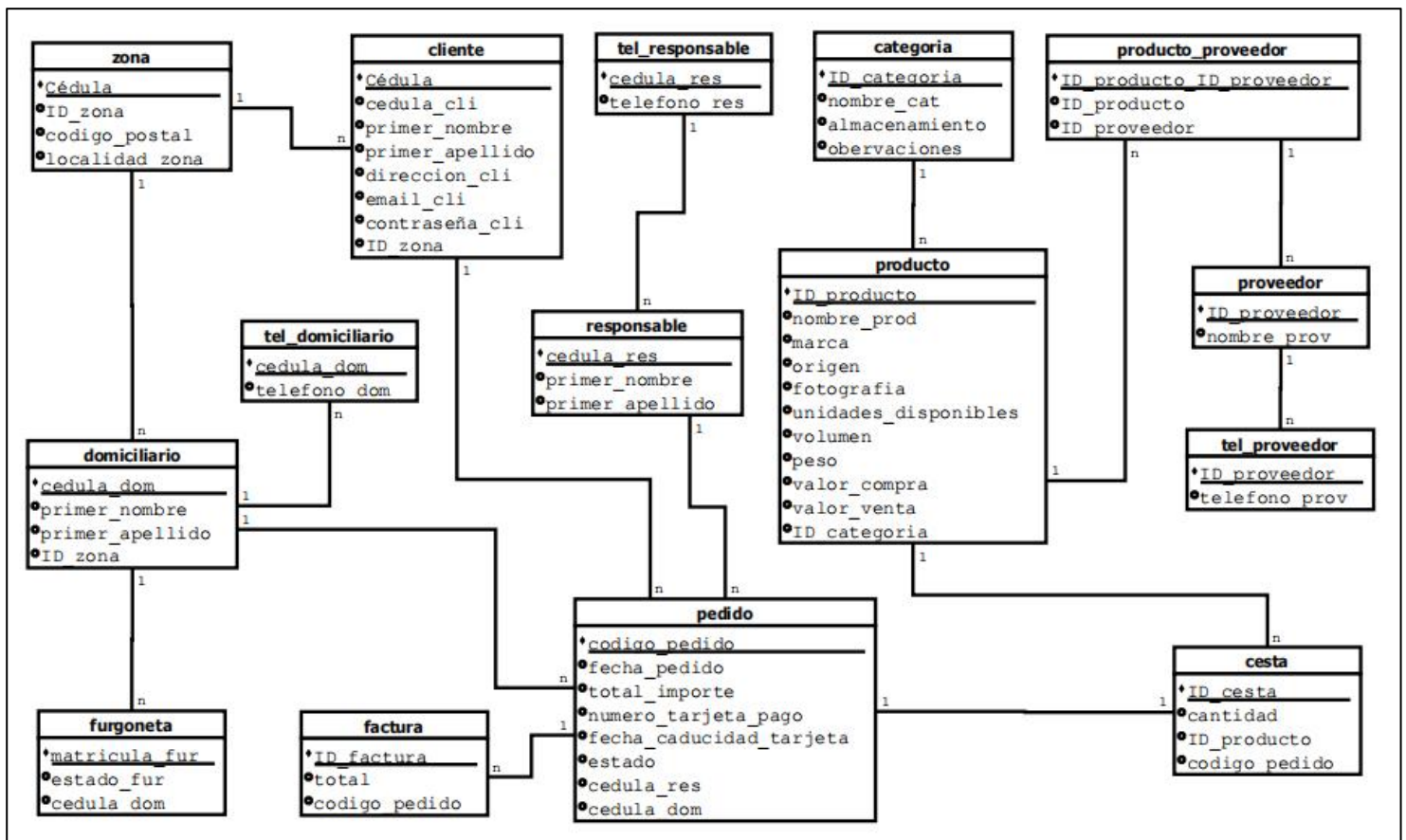
-La tabla Proveedor tiene los siguientes atributos: ID\_proveedor y nombre\_prov.

-La tabla Telefono\_prov tiene los siguientes atributos: ID\_proveedor y telefono\_prov. ID\_proveedor es llave primaria.

-La tabla factura tiene los siguientes atributos: ID\_factura, total y codigo\_pedido. ID\_factura es la llave primaria, codigo\_pedido es llave foranea relacionada con la tabla Pedido.

Para las relaciones N:M se crean las nuevas tablas respectivas, se tienen en cuenta las relaciones según las llaves foraneas, en consecuencia se obtiene el siguiente modelo relacional:





## ● Normalizar correctamente

Se identifica que el modelo relacional no tiene atributos repetidos, cada tabla tiene una llave primaria, no hay atributos multivaluados, todos los atributos tienen valores atómicos. En consecuencia se encuentra en 1ra forma normal.

Por otra parte todos los valores de las columnas dependen únicamente de la llave primaria de cada tabla y las tablas tienen una única llave primaria que identifica a la tabla y sus atributos dependen de ella. En consecuencia se encuentra en 2da forma normal.

El modelo relacional esta en 2da forma normal, en 1ra forma normal y cada atributo que no está incluido en la clave primaria no depende transitivamente de la clave primaria. En consecuencia esta en 3ra forma normal. Por lo anterior se ha normalizado el modelo.

## ● Escribir con sentencias SQL toda la definición de la base de datos.

Se crea la tabla Responsable:

The screenshot shows a SQL IDE with a schema named 'TiendaPepe'. The 'Tables' folder is expanded, showing a list of tables including 'Responsable'. The SQL editor displays the following code:

```
1 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
2 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
3 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_...
4
5 CREATE SCHEMA IF NOT EXISTS `TiendaPepe` DEFAULT CHARACTER SET utf8 ;
6 USE `TiendaPepe` ;
7
8 CREATE TABLE IF NOT EXISTS `TiendaPepe`.`Responsable` (
9   `cedula_res` VARCHAR(50) NOT NULL,
10  `primer_nombre` VARCHAR(50) NOT NULL,
11  `primer_apellido` VARCHAR(50) NOT NULL,
12  PRIMARY KEY (`cedula_res`)
13 ) ENGINE = InnoDB;
14
15 CREATE TABLE IF NOT EXISTS `TiendaPepe`.`Tel_responsable` (
16   `cedula_res` VARCHAR(50) NOT NULL,
17   `telefono_res` VARCHAR(50) NOT NULL,
18   PRIMARY KEY (`cedula_res`),
19   CONSTRAINT `Tel_responsable_Responsable`
20     FOREIGN KEY (`cedula_res`)
21     REFERENCES `TiendaPepe`.`Responsable` (`cedula_res`)
```

Annotations in the image:

- Red boxes highlight the schema creation and the 'Responsable' table creation.
- Arrows point from text to the schema and table creation statements.
- Text: "Sentencias para ingresar las tablas que están relacionadas con otras y requieren de su creación previa para ser creadas. Con estas sentencias se omite el chequeo y se permite la creación de las tablas." points to the SET statements at the top.
- Text: "Crea la base de datos TiendaPepe" points to the 'CREATE SCHEMA' statement.
- Text: "Se crea la tabla Responsable" points to the 'CREATE TABLE' statement for 'Responsable'.

Se crea la tabla Tel\_responsable:

The screenshot shows the SQL code for creating the 'Tel\_responsable' table:

```
1 CREATE TABLE IF NOT EXISTS `TiendaPepe`.`Tel_responsable` (
2   `cedula_res` VARCHAR(50) NOT NULL,
3   `telefono_res` VARCHAR(50) NOT NULL,
4   PRIMARY KEY (`cedula_res`),
5   CONSTRAINT `Tel_responsable_Responsable`
6     FOREIGN KEY (`cedula_res`)
7     REFERENCES `TiendaPepe`.`Responsable` (`cedula_res`)
8     ON DELETE CASCADE
9     ON UPDATE CASCADE
10 ) ENGINE = InnoDB;
```

Annotations in the image:

- A red box highlights the 'ON DELETE CASCADE' and 'ON UPDATE CASCADE' clauses.
- An arrow points from the text "Para cuando se borre un registro en esta tabla se borre también en las demás tablas relacionadas a esta." to the highlighted clauses.

Se crea la tabla domiciliario:

The screenshot shows the SQL code for creating the 'Domiciliario' table:

```
26 CREATE TABLE IF NOT EXISTS `TiendaPepe`.`Domiciliario` (
27   `cedula_dom` VARCHAR(50) NOT NULL,
28   `primer_nombre` VARCHAR(50) NOT NULL,
29   `primer_apellido` VARCHAR(50) NOT NULL,
30   `ID_zona` VARCHAR(50) NOT NULL,
31   PRIMARY KEY (`cedula_dom`),
32   CONSTRAINT `Domiciliario_Zona`
33     FOREIGN KEY (`ID_zona`)
34     REFERENCES `TiendaPepe`.`Zona` (`ID_zona`)
35     ON DELETE CASCADE
36     ON UPDATE CASCADE
37 ) ENGINE = InnoDB;
```

Annotations in the image:

- A red box highlights the entire table creation statement.
- An arrow points from the text "Se crea la tabla domiciliario" to the table creation statement.

Se crea la tabla Tel\_domiciliario:



```
38
39 • CREATE TABLE IF NOT EXISTS `TiendaPepe`.`Tel_domiciliario` (
40     `cedula_dom` VARCHAR(50) NOT NULL,
41     `telefono_dom` VARCHAR(50) NOT NULL,
42     PRIMARY KEY (`cedula_dom`),
43     CONSTRAINT `Telefono_dom_Domiciliario`
44         FOREIGN KEY (`cedula_dom`)
45         REFERENCES `TiendaPepe`.`Domiciliario` (`cedula_dom`)
46         ON DELETE CASCADE
47         ON UPDATE CASCADE
48 ) ENGINE = InnoDB;
```

Se crea la tabla Furgoneta:

```
50 • CREATE TABLE IF NOT EXISTS `TiendaPepe`.`Furgoneta` (
51     `matricula_fur` VARCHAR(50) NOT NULL,
52     `estado_fur` VARCHAR(50) NOT NULL,
53     `cedula_dom` VARCHAR(50) NOT NULL,
54     PRIMARY KEY (`matricula_fur`),
55     CONSTRAINT `Furgoneta_Domiciliario`
56         FOREIGN KEY (`cedula_dom`)
57         REFERENCES `TiendaPepe`.`Domiciliario` (`cedula_dom`)
58         ON DELETE CASCADE
59         ON UPDATE CASCADE
60 ) ENGINE = InnoDB;
61
```

Se crea la tabla Cliente:

```
62 • CREATE TABLE IF NOT EXISTS `TiendaPepe`.`Cliente` (
63     `ID_cli` VARCHAR(50) NOT NULL,
64     `cedula_cli` VARCHAR(50) NOT NULL,
65     `primer_nombre` VARCHAR(50) NOT NULL,
66     `primer_apellido` VARCHAR(50) NOT NULL,
67     `direccion_cli` VARCHAR(50) NOT NULL,
68     `email_cli` VARCHAR(50) NOT NULL,
69     `contraseña_cli` VARCHAR(50) NOT NULL,
70     `ID_zona` VARCHAR(50) NOT NULL,
71     PRIMARY KEY (`ID_cli`),
72     CONSTRAINT `Cliente_Zona`
73         FOREIGN KEY (`ID_zona`)
74         REFERENCES `TiendaPepe`.`Zona` (`ID_zona`)
75         ON DELETE CASCADE
76         ON UPDATE CASCADE
77 ) ENGINE = InnoDB;
78
```

Se crea la tabla Zona:

```
79 • CREATE TABLE IF NOT EXISTS `TiendaPepe`.`Zona` (
80     `ID_zona` VARCHAR(50) NOT NULL,
81     `codigo_postal` VARCHAR(50) NOT NULL,
82     `localidad_zona` VARCHAR(50) NOT NULL,
83     PRIMARY KEY (`ID_zona`)
84 ) ENGINE = InnoDB;
```

Se crea la tabla Factura:

```
85
86 ● CREATE TABLE IF NOT EXISTS `TiendaPepe`.`Factura` (
87     `ID_factura` VARCHAR(45) NOT NULL,
88     `total` VARCHAR(45) NOT NULL,
89     `codigo_pedido` VARCHAR(45) NOT NULL,
90     PRIMARY KEY (`ID_factura`),
91     FOREIGN KEY (`codigo_pedido`)
92     REFERENCES `TiendaPepe`.`Pedido` (`codigo_pedido`)
93 ) ENGINE = InnoDB;
```

Se crea la tabla Pedido:

```
95 ● CREATE TABLE IF NOT EXISTS `TiendaPepe`.`Pedido` (
96     `codigo_pedido` VARCHAR(45) NOT NULL,
97     `fecha_pedido` VARCHAR(45) NOT NULL,
98     `total_importe` VARCHAR(45) NOT NULL,
99     `numero_tarjeta_pago` VARCHAR(16) NOT NULL,
100     `fecha_caducidad_tarjeta` VARCHAR(16) NOT NULL,
101     `estado` VARCHAR(45) NOT NULL,
102     `cedula_res` VARCHAR(45) NOT NULL,
103     `cedula_dom` VARCHAR(45) NOT NULL,
104     `ID_cli` VARCHAR(45) NOT NULL,
105     PRIMARY KEY (`codigo_pedido`),
106     CONSTRAINT `Pedido_Domiciliario1`
107     FOREIGN KEY (`cedula_dom`)
108     REFERENCES `TiendaPepe`.`Domiciliario` (`cedula_dom`)
109     ON DELETE NO ACTION
110     ON UPDATE NO ACTION,
111     CONSTRAINT `Pedido_Responsable1`
112     FOREIGN KEY (`cedula_res`)
113     REFERENCES `TiendaPepe`.`Responsable` (`cedula_res`)
114     ON DELETE NO ACTION
115     ON UPDATE NO ACTION,
```

Se crea la tabla Cesta:

```
122
123 ● CREATE TABLE IF NOT EXISTS `TiendaPepe`.`Cesta` (
124     `ID_cesta` VARCHAR(45) NOT NULL,
125     `ID_producto` VARCHAR(45) NOT NULL,
126     `cantidad` VARCHAR(45) NOT NULL,
127     `codigo_pedido` VARCHAR(45) NOT NULL,
128     PRIMARY KEY (`ID_cesta`),
129     CONSTRAINT `Cesta_Producto1`
130     FOREIGN KEY (`ID_producto`)
131     REFERENCES `TiendaPepe`.`Producto` (`ID_producto`),
132     CONSTRAINT `Cesta_Pedido1`
133     FOREIGN KEY (`codigo_pedido`)
134     REFERENCES `TiendaPepe`.`Pedido` (`codigo_pedido`)
135 ) ENGINE = InnoDB;
136
```

Se crea la tabla Categoria:

```
136
137 ● CREATE TABLE IF NOT EXISTS `TiendaPepe`.`Categoria` (
138     `ID_categoria` VARCHAR(45) NOT NULL,
139     `nombre_cat` VARCHAR(45) NOT NULL,
140     `almacenamiento` VARCHAR(45),
141     `observaciones` VARCHAR(50),
142     PRIMARY KEY (`ID_categoria`)
143 ) ENGINE = InnoDB;
144
```

Se crea la tabla Producto:

```
144
145 • CREATE TABLE IF NOT EXISTS `TiendaPepe`.`Producto` (
146     `ID_producto` VARCHAR(45) NOT NULL,
147     `nombre_prod` VARCHAR(45) NOT NULL,
148     `marca` VARCHAR(45) NOT NULL,
149     `origen` VARCHAR(45) NOT NULL,
150     `fotografia` VARCHAR(45),
151     `unidades_disponibles` VARCHAR(45) NOT NULL,
152     `volumen` VARCHAR(45),
153     `peso` VARCHAR(45),
154     `valor_compra` VARCHAR(45) NOT NULL,
155     `valor_venta` VARCHAR(45) NOT NULL,
156     `ID_categoria` VARCHAR(45) NOT NULL,
157     PRIMARY KEY (`ID_producto`),
158     FOREIGN KEY (`ID_categoria`)
159     REFERENCES `TiendaPepe`.`Categoria` (`ID_categoria`)
160 ) ENGINE = InnoDB;
```

Se crea la tabla Producto\_Proveedor:

```
162 • CREATE TABLE IF NOT EXISTS `TiendaPepe`.`Producto_Proveedor` (
163     `ID_producto_ID_proveedor` VARCHAR(45) NOT NULL,
164     `ID_producto` VARCHAR(45) NOT NULL,
165     `ID_proveedor` VARCHAR(45) NOT NULL,
166     PRIMARY KEY (`ID_producto_ID_proveedor`,`ID_producto`,`ID_proveedor`),
167     FOREIGN KEY (`ID_producto`)
168     REFERENCES `TiendaPepe`.`Producto` (`ID_producto`),
169     FOREIGN KEY (`ID_proveedor`)
170     REFERENCES `TiendaPepe`.`Proveedor` (`ID_proveedor`)
171 ) ENGINE = InnoDB;
172
```

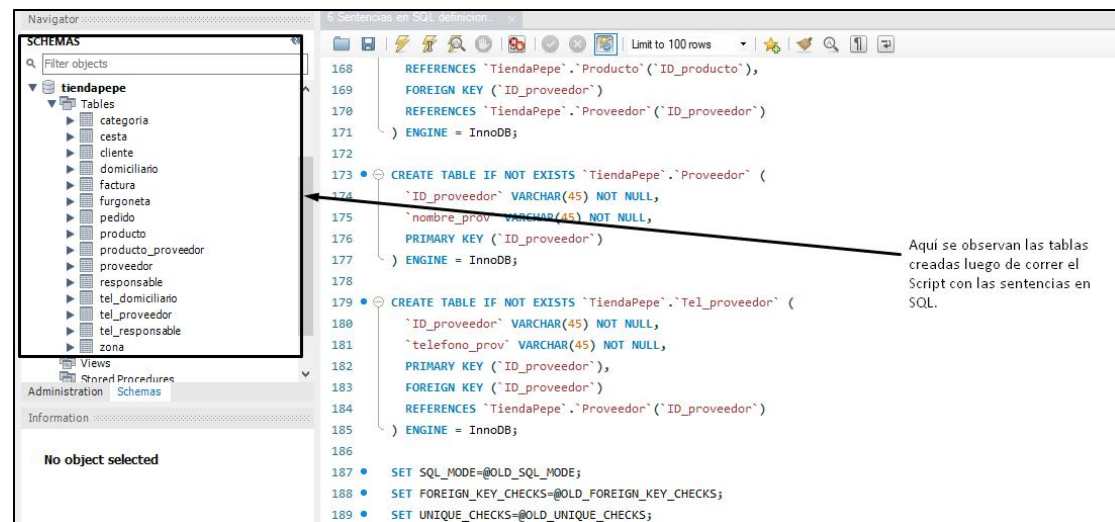
Se crea la tabla Proveedor:

```
172
173 • CREATE TABLE IF NOT EXISTS `TiendaPepe`.`Proveedor` (
174     `ID_proveedor` VARCHAR(45) NOT NULL,
175     `nombre_prov` VARCHAR(45) NOT NULL,
176     PRIMARY KEY (`ID_proveedor`)
177 ) ENGINE = InnoDB;
178
```

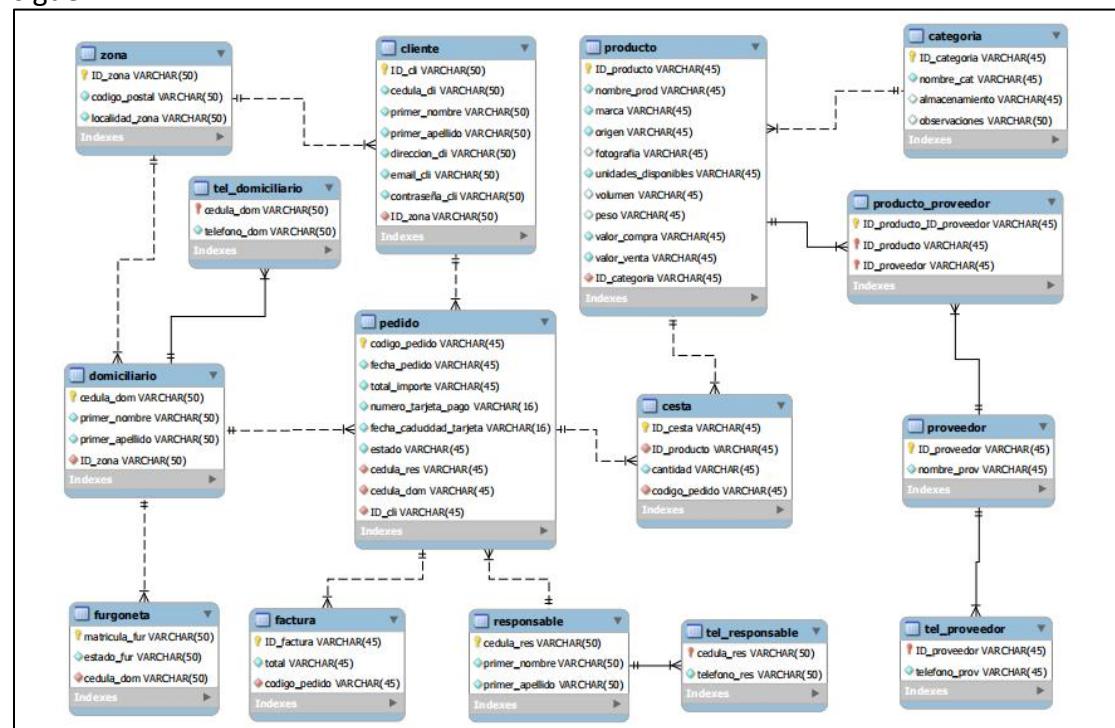
Se crea la tabla Tel\_proveedor:

```
179 • CREATE TABLE IF NOT EXISTS `TiendaPepe`.`Tel_proveedor` (
180     `ID_proveedor` VARCHAR(45) NOT NULL,
181     `telefono_prov` VARCHAR(45) NOT NULL,
182     PRIMARY KEY (`ID_proveedor`),
183     FOREIGN KEY (`ID_proveedor`)
184     REFERENCES `TiendaPepe`.`Proveedor` (`ID_proveedor`)
185 ) ENGINE = InnoDB;
```

A continuación se observa la creación de las tablas en MySQL:



Realizando ingeniería inversa se obtiene el modelo relacional en Workbench como sigue:



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



Para mostrar ejemplos las consultas previamente se insertan algunos registros en cada una de las tablas como sigue:

Ingreso de registros en la tabla proveedor:

```
7 • SELECT * FROM tiendapepe.proveedor;
8 • INSERT INTO `TiendaPepe`.`Proveedor`(`ID_proveedor`,`nombre_prov`)
9   VALUES
10  ('PROV01','Soluciones logisticas de transporte'),
11  ('PROV02','Cargolargo'),
12  ('PROV03','Transportes la garza');
```

Ingreso de registros en la tabla tel\_proveedor:

```
14 • SELECT * FROM tiendapepe.tel_proveedor;
15 • INSERT INTO `TiendaPepe`.`Tel_proveedor`(`ID_proveedor`,`telefono_prov`)
16   VALUES
17  ('PROV01','3193202102'),
18  ('PROV02','3196966956'),
19  ('PROV03','3195858499');
```

Ingreso de registros en la tabla categoria:

```
21 • SELECT * FROM tiendapepe.categoria;
22 • INSERT INTO `TiendaPepe`.`Categoria`(`ID_categoria`,`nombre_cat`,`almacenamiento`,`observaciones`)
23   VALUES
24  ('CA01','carnes','Almacenamiento en congelador','Llevar tiempos de viaje y almacenamiento'),
25  ('CA02','granos','Almacenamiento en lugar fresco','Acomodar periodicamente'),
26  ('CA03','vegetales','Almacenamiento en lugar fresco','Revisar el estado y hacer cambio'),
27  ('CA04','frutas','Almacenamiento en lugar fresco','Revisar el estado y hacer cambio'),
28  ('CA05','lacteos','Almacenamiento en congelador','Llevar tiempos de viaje y almacenamiento'),
29  ('CA06','Aseo y hogar','Almacenamiento al aire libre','Limpiar con trapo húmedo cada 3 días');
```

Ingreso de registros en la tabla producto:

```
31 • SELECT * FROM tiendapepe.producto;
32 • INSERT INTO `TiendaPepe`.`producto`(`ID_producto`,`nombre_prod`,`marca`,`origen`,`fotografia`,`unidades`)
33   VALUES
34  ('PR001','Jabón líquido','Axion','Colombiano','fotografía1','59','42','50','13000','14500','CA06');
35 • INSERT INTO `TiendaPepe`.`producto`(`ID_producto`,`nombre_prod`,`marca`,`origen`,`fotografia`,`unidades`)
36   VALUES
37  ('PR002','Carne de res','Frigor','Argentina','fotografía2','23','10','500','12000','15000','CA01');
38 • INSERT INTO `TiendaPepe`.`producto`(`ID_producto`,`nombre_prod`,`marca`,`origen`,`fotografia`,`unidades`)
39   VALUES
40  ('PR003','Frijol rojo','Luker','Colombia','fotografía3','87','5','100','4000','6000','CA02');
41 • INSERT INTO `TiendaPepe`.`producto`(`ID_producto`,`nombre_prod`,`marca`,`origen`,`fotografia`,`unidades`)
42   VALUES
43  ('PR004','Leche deslactosada','Alpina','Colombia','fotografía4','48','5','1000','4000','6000','CA05');
44 • INSERT INTO `TiendaPepe`.`producto`(`ID_producto`,`nombre_prod`,`marca`,`origen`,`fotografia`,`unidades`)
45   VALUES
46  ('PR005','Tomate','Del huerto','Colombia','fotografía5','72','2','50','1500','2000','CA03');
```

Ingreso de registros en la tabla cesta:

```
61 • SELECT * FROM tiendapepe.cesta;
62 • INSERT INTO `TiendaPepe`.`Cesta`(`ID_cesta`,`ID_producto`,`cantidad`,`codigo_pedido`)
63 VALUES
64 ('C01', 'PRO01','12','P01'),
65 ('C02', 'PRO04','5','P01'),
66 ('C03', 'PRO09','16','P01'),
67 ('C04', 'PRO03','12','P01'),
68 ('C05', 'PRO08','4','P02'),
69 ('C06', 'PRO05','11','P03'),
70 ('C07', 'PRO01','12','P03'),
71 ('C08', 'PRO04','9','P03'),
72 ('C09', 'PRO06','1','P03');
```

Ingreso de registros en la tabla domiciliario:

```
74 • SELECT * FROM tiendapepe.domiciliario;
75 • INSERT INTO `TiendaPepe`.`Domiciliario`(`cedula_dom`,`primer_nombre`,`primer_apellido`,`ID_zona`)
76 VALUES
77 ('896698', 'Duran','Deles','Zona1'),
78 ('852258', 'Dario','Durango','Zona2'),
79 ('874478', 'Demian','Dolores','Zona3');
```

Ingreso de registros en la tabla tel\_domiciliario:

```
81 • SELECT * FROM tiendapepe.tel_domiciliario;
82 • INSERT INTO `TiendaPepe`.`Tel_domiciliario`(`cedula_dom`,`telefono_dom`)
83 VALUES
84 ('896698', '320896698'),
85 ('852258', '320852258'),
86 ('874478', '320874478');
```

Ingreso de registros en la tabla cliente:

```
88 • SELECT * FROM tiendapepe.cliente;
89 • INSERT INTO `TiendaPepe`.`Cliente`(`ID_cli`,`cedula_cli`,`primer_nombre`,`primer_apellido`,`direccion_cli`)
90 VALUES
91 ('Cli1','456654','Carlos','Cubides','Dirección1','carlos@gmail.com','carlos1234','Zona1'),
92 ('Cli2','478874','Carla','Claron','Dirección2','carla@gmail.com','carla1234','Zona2'),
93 ('Cli3','423324','Cintia','Camero','Dirección3','cintia@gmail.com','cintia1234','Zona3');
```

Ingreso de registros en la tabla furgoneta:

```
95 • SELECT * FROM tiendapepe.furgoneta;
96 • INSERT INTO `TiendaPepe`.`Furgoneta`(`matricula_fur`,`estado_fur`,`cedula_dom`)
97 VALUES
98 ('MCK-896','operativa','896698'),
99 ('BAS-852','en mantenimiento','852258'),
100 ('LOD-874','operativa','874478');
```

Ingreso de registros en la tabla pedido:



```
102 • SELECT * FROM tiendapepe.pedido;
103 • INSERT INTO `TiendaPepe`.`Pedido`(`codigo_pedido`,`fecha_pedido`,`total_importe`,`numero_tarjeta_pago`,`f
104 VALUES
105 ('P01','17/02/2023','658000','123321','17/05/2024','En ruta','987789','896698','Cli1'),
106 ('P02','17/02/2023','120000','145541','17/06/2025','En ruta','963369','852258','Cli2'),
107 ('P03','17/02/2023','340000','159951','28/04/2025','Entregado a cliente','951159','874478','Cli3');
```

Ingreso de registros en la tabla factura:

```
109 • SELECT * FROM tiendapepe.factura;
110 • INSERT INTO `TiendaPepe`.`Factura`(`ID_factura`,`total`,`codigo_pedido`)
111 VALUES
112 ('F01','658000','P01'),
113 ('F02','120000','P02'),
114 ('F03','340000','P03');
```

Ingreso de registros en la tabla responsable:

```
116 • SELECT * FROM tiendapepe.responsable;
117 • INSERT INTO `TiendaPepe`.`Responsable`(`cedula_res`,`primer_nombre`,`primer_apellido`)
118 VALUES
119 ('987789','Rita','Ruales'),
120 ('963369','Román','Rincón'),
121 ('951159','Rina','Romelo');
```

Ingreso de registros en la tabla tel\_responsable:

```
123 • SELECT * FROM tiendapepe.tel_responsable;
124 • INSERT INTO `TiendaPepe`.`Tel_responsable`(`cedula_res`,`telefono_res`)
125 VALUES
126 ('987789','311987789'),
127 ('963369','311963369'),
128 ('951159','311951159');
```

Ingreso de registros en la tabla producto\_proveedores:

```
130 • SELECT * FROM tiendapepe.producto_proveedor;
131 • INSERT INTO `TiendaPepe`.`Producto_Proveedor`(`ID_producto_ID_proveedor`,`ID_producto`,`ID_proveedor`)
132 VALUES
133 ('PP01','PRO01','PROV02'),
134 ('PP02','PRO02','PROV02'),
135 ('PP03','PRO03','PROV02');
```

Ingreso de registros en la tabla zona:

```
137 • SELECT * FROM tiendapepe.zona;
138 • INSERT INTO `TiendaPepe`.`Zona`(`ID_zona`,`codigo_postal`,`localidad_zona`)
139 VALUES
140 ('Zona1','111111','Localidad1'),
141 ('Zona2','222222','Localidad2'),
142 ('Zona3','333333','Localidad3');
```

Para ver la información de cada tabla se escriben las siguientes consultas:

### Consulta tabla proveedor:

The screenshot shows a database query tool with a list of 10 SQL queries. The first query is highlighted with a red box and labeled 'Consulta' with an arrow. The query is: `SELECT * FROM tiendapepe.proveedor;`

Below the queries, the 'Result Grid' shows the results of the query, labeled 'Resultado' with an arrow. The results are as follows:

ID_proveedor	nombre_prov
PROV01	Soluciones logísticas de transporte
PROV02	Cargolargo
PROV03	Transportes la garza
NULL	NULL

The 'Result Grid' tab is selected, and the 'proveedor5' tab is visible in the bottom tab bar.

### Consulta tabla tel\_proveedor:

The screenshot shows the same database query tool with the second query highlighted by a red box. The query is: `SELECT * FROM tiendapepe.tel_proveedor;`

The 'Result Grid' shows the results of this query. The results are as follows:

ID_proveedor	telefono_prov
PROV01	3193202102
PROV02	3196966956
PROV03	3195858499
NULL	NULL

The 'Result Grid' tab is selected, and the 'tel\_proveedor6' tab is visible in the bottom tab bar.

### Consulta tabla categoria:

8 Sentencias en SQL para crear... 6 Sentencias en SQL definicion... 9 Consultas para ver informacio... x cesta - Table cesta

Limit to 100 rows

1 • SELECT \* FROM tiendapepe.proveedor;  
2 • SELECT \* FROM tiendapepe.tel\_proveedor;  
3 • SELECT \* FROM tiendapepe.categoria;  
4 • SELECT \* FROM tiendapepe.producto;  
5 • SELECT \* FROM tiendapepe.cesta;  
6 • SELECT \* FROM tiendapepe.domiciliario;  
7 • SELECT \* FROM tiendapepe.tel\_domiciliario;  
8 • SELECT \* FROM tiendapepe.cliente;  
9 • SELECT \* FROM tiendapepe.furgoneta;  
10 • SELECT \* FROM tiendapepe.pedido;

Result Grid

ID_categoria	nombre_cat	almacenamiento	observaciones
CA01	carnes	Almacenamiento en congelador	Llevar tiempos de viaje y almacenamiento
CA02	granos	Almacenamiento en lugar fresco	Acomodar periodicamente
CA03	vegetales	Almacenamiento en lugar fresco	Revisar el estado y hacer cambio
CA04	frutas	Almacenamiento en lugar fresco	Revisar el estado y hacer cambio
CA05	lacteos	Almacenamiento en congelador	Llevar tiempos de viaje y almacenamiento
CA06	Aseo y hogar	Almacenamiento al aire libre	Limpiar con trapo húmedo cada 3 días
NULL	NULL	NULL	NULL

proveedor5 tel\_proveedor6 categoria 7 x producto 8 cesta 9 domiciliario 10 tel\_domiciliario 11 clien Apply Revert

Consulta tabla producto:

8 Sentencias en SQL para crear... 6 Sentencias en SQL definicion... 9 Consultas para ver informacio... x cesta - Table cesta

Limit to 100 rows

1 • SELECT \* FROM tiendapepe.proveedor;  
2 • SELECT \* FROM tiendapepe.tel\_proveedor;  
3 • SELECT \* FROM tiendapepe.categoria;  
4 • SELECT \* FROM tiendapepe.producto;  
5 • SELECT \* FROM tiendapepe.cesta;  
6 • SELECT \* FROM tiendapepe.domiciliario;  
7 • SELECT \* FROM tiendapepe.tel\_domiciliario;  
8 • SELECT \* FROM tiendapepe.cliente;  
9 • SELECT \* FROM tiendapepe.furgoneta;  
10 • SELECT \* FROM tiendapepe.pedido;

Result Grid

ID_producto	nombre_prod	marca	origen	fotografia	unidades_disponibles	volumen	peso	valor_compra	valor_venta
PRO01	Jabón líquido	Axion	Colombiano	fotografia1	59	42	50	13000	14500
PRO02	Carne de res	Frigor	Argentina	fotografia2	23	10	500	12000	15000
PRO03	Frijol rojo	Luker	Colombia	fotografia3	87	5	100	4000	6000
PRO04	Leche deslactosada	Alpina	Colombia	fotografia4	48	5	1000	4000	6000
PRO05	Tomate	Del huerto	Colombia	fotografia5	72	2	50	1500	2000
PRO06	Manzana	La Montaña	Colombia	fotografia6	65	2	60	2500	3000
PRO07	Arroz	Diana	Colombia	fotografia7	91	10	500	5000	7000
PRO08	Yogurt	Alquería	Colombia	fotografia8	36	5	800	2500	3500
PRO09	Atún en lata	Van Camps	México	fotografia9	64	5	100	4000	5000
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

proveedor5 tel\_proveedor6 categoria 7 producto 8 x cesta 9 domiciliario 10 tel\_domiciliario 11 clien Apply Revert

Consulta tabla cesta:

8 Sentencias en SQL para crear... 6 Sentencias en SQL definicion... 9 Consultas para ver informacio... x cesta - Table cesta

Limit to 100 rows

1 • SELECT \* FROM tiendapepe.proveedor;  
2 • SELECT \* FROM tiendapepe.tel\_proveedor;  
3 • SELECT \* FROM tiendapepe.categoria;  
4 • SELECT \* FROM tiendapepe.producto;  
5 • SELECT \* FROM tiendapepe.cesta;  
6 • SELECT \* FROM tiendapepe.domiciliario;  
7 • SELECT \* FROM tiendapepe.tel\_domiciliario;  
8 • SELECT \* FROM tiendapepe.cliente;  
9 • SELECT \* FROM tiendapepe.furgoneta;  
10 • SELECT \* FROM tiendapepe.pedido;

Result Grid

ID_cesta	ID_producto	cantidad	codigo_pedido
C01	PRO01	12	P01
C02	PRO04	5	P01
C03	PRO09	16	P01
C04	PRO03	12	P01
C05	PRO08	4	P02
C06	PRO05	11	P03
C07	PRO01	12	P03
C08	PRO04	9	P03
C09	PRO06	1	P03
NULL	NULL	NULL	NULL

proveedor 5 tel\_proveedor 6 categoria 7 producto 8 cesta 9 x domiciliario 10 tel\_domiciliario 11 clien Apply Revert

Consulta tabla domiciliario:

8 Sentencias en SQL para crear... 6 Sentencias en SQL definicion... 9 Consultas para ver informacio... x cesta - Table cesta

Limit to 100 rows

4 • SELECT \* FROM tiendapepe.producto;  
5 • SELECT \* FROM tiendapepe.cesta;  
6 • SELECT \* FROM tiendapepe.domiciliario;  
7 • SELECT \* FROM tiendapepe.tel\_domiciliario;  
8 • SELECT \* FROM tiendapepe.cliente;  
9 • SELECT \* FROM tiendapepe.furgoneta;  
10 • SELECT \* FROM tiendapepe.pedido;  
11 • SELECT \* FROM tiendapepe.factura;  
12 • SELECT \* FROM tiendapepe.responsable;  
13 • SELECT \* FROM tiendapepe.tel\_responsable;

Result Grid

cedula_dom	primer_nombre	primer_apellido	ID_zona
852258	Dario	Durango	Zona2
874478	Demian	Dolores	Zona3
896698	Duran	Deles	Zona1
NULL	NULL	NULL	NULL

proveedor 5 tel\_proveedor 6 categoria 7 producto 8 cesta 9 domiciliario 10 x tel\_domiciliario 11 clien Apply Revert

Consulta tabla tel\_domiciliario:



The screenshot shows a database application interface with a SQL editor and a result grid. The SQL editor contains a list of queries, with the 7th query, `SELECT * FROM tiendapepe.tel_domiciliario;`, highlighted with a red box. The result grid below shows the data for this query:

cedula_dom	telefono_dom
852258	320852258
874478	320874478
896698	320896698
NULL	NULL

The bottom of the interface shows a tab bar with the following tabs: domiciliario 10, **tel\_domiciliario 11**, cliente 12, furgoneta 13, pedido 14, factura 15, responsable16, tel\_r, Apply, and Revert.

Consulta tabla cliente:

The screenshot shows the same database application interface, but with the 8th query, `SELECT * FROM tiendapepe.cliente;`, highlighted with a red box. The result grid below shows the data for this query:

ID_cli	cedula_cli	primer_nombre	primer_apellido	direccion_cli	email_cli	contraseña_cli	ID_zona
Cli1	456654	Carlos	Cubides	Dirección1	carlos@gmail.com	carlos1234	Zona1
Cli2	478874	Carla	Claron	Dirección2	carla@gmail.com	carla1234	Zona2
Cli3	423324	Cintia	Camero	Dirección3	cintia@gmail.com	cintia1234	Zona3
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

The bottom of the interface shows a tab bar with the following tabs: domiciliario 10, tel\_domiciliario 11, **cliente 12**, furgoneta 13, pedido 14, factura 15, responsable16, tel\_r, Apply, and Revert.

Consulta tabla furgoneta:

The screenshot shows a database management interface with a list of SQL queries. Query 9, `SELECT * FROM tiendapepe.furgoneta;`, is highlighted with a red box. Below the queries, the 'Result Grid' displays the data for the 'furgoneta' table. The grid has three columns: 'matricula\_fur', 'estado\_fur', and 'cedula\_dom'. The data is as follows:

matricula_fur	estado_fur	cedula_dom
BAS-852	en mantenimiento	852258
LOD-874	operativa	874478
MCK-896	operativa	896698
NULL	NULL	NULL

The bottom of the interface shows a tab labeled 'furgoneta 13' selected.

Consulta tabla pedido:

The screenshot shows the same database management interface, but now query 10, `SELECT * FROM tiendapepe.pedido;`, is highlighted with a red box. The 'Result Grid' displays the data for the 'pedido' table. The grid has eight columns: 'codigo\_pedido', 'fecha\_pedido', 'total\_importe', 'numero\_tarjeta\_pago', 'fecha\_caducidad\_tarjeta', 'estado', 'cedula\_res', and 'cedula\_do'. The data is as follows:

codigo_pedido	fecha_pedido	total_importe	numero_tarjeta_pago	fecha_caducidad_tarjeta	estado	cedula_res	cedula_do
P01	17/02/2023	658000	123321	17/05/2024	En ruta	987789	896698
P02	17/02/2023	120000	145541	17/06/2025	En ruta	963369	852258
P03	17/02/2023	340000	159951	28/04/2025	Entregado a cliente	951159	874478
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

The bottom of the interface shows a tab labeled 'pedido 14' selected.

Consulta tabla factura:



The screenshot shows a database management tool interface. The top bar indicates '9 Consultas para ver informacio...'. The main area displays a list of SQL queries. Query 11, 'SELECT \* FROM tiendapepe.factura;', is highlighted with a red box. Below the queries, the 'Result Grid' shows the results of query 11. The grid has three columns: 'ID\_factura', 'total', and 'codigo\_pedido'. The data rows are:

ID_factura	total	codigo_pedido
F01	658000	P01
F02	120000	P02
F03	340000	P03
NULL	NULL	NULL

The bottom status bar shows the current query is 'actura 15'.

Consulta tabla responsable:

The screenshot shows the same database management tool interface. The top bar indicates '9 Consultas para ver informacio...'. The main area displays a list of SQL queries. Query 12, 'SELECT \* FROM tiendapepe.responsable;', is highlighted with a red box. Below the queries, the 'Result Grid' shows the results of query 12. The grid has three columns: 'cedula\_res', 'primer\_nombre', and 'primer\_apellido'. The data rows are:

cedula_res	primer_nombre	primer_apellido
951159	Rina	Romelo
963369	Román	Rincón
987789	Rita	Ruales
NULL	NULL	NULL

The bottom status bar shows the current query is 'responsable 16'.

Consulta tabla tel\_responsable:

The screenshot shows a database management interface with a list of 15 SQL queries. Query 13, `SELECT * FROM tiendapepe.tel_responsable;`, is highlighted with a red box. Below the queries, the 'Result Grid' displays the data for this query:

cedula_res	telefono_res
951159	311951159
963369	311963369
987789	311987789
NULL	NULL

The bottom of the interface shows a breadcrumb trail with 'tel\_responsable17' highlighted in a red box.

Consulta tabla producto\_proveedor:

The screenshot shows the same database management interface, but with query 14, `SELECT * FROM tiendapepe.producto_proveedor;`, highlighted with a red box. The 'Result Grid' displays the data for this query:

ID_producto_ID_proveedor	ID_producto	ID_proveedor
PP01	PRO01	PROV02
PP02	PRO02	PROV02
PP03	PRO03	PROV02
NULL	NULL	NULL

The bottom of the interface shows a breadcrumb trail with 'producto proveedor33' highlighted in a red box.

## Consulta tabla zona:

8 Sentencias en SQL para crear... 6 Sentencias en SQL definicion... 9 Consultas para ver informacio... cesta - Table cesta

Limit to 100 rows

6 • SELECT \* FROM tiendapepe.domiciliario;  
7 • SELECT \* FROM tiendapepe.tel\_domiciliario;  
8 • SELECT \* FROM tiendapepe.cliente;  
9 • SELECT \* FROM tiendapepe.furgoneta;  
10 • SELECT \* FROM tiendapepe.pedido;  
11 • SELECT \* FROM tiendapepe.factura;  
12 • SELECT \* FROM tiendapepe.responsable;  
13 • SELECT \* FROM tiendapepe.tel\_responsable;  
14 • SELECT \* FROM tiendapepe.producto\_proveedor;  
15 • SELECT \* FROM tiendapepe.zona;

Result Grid

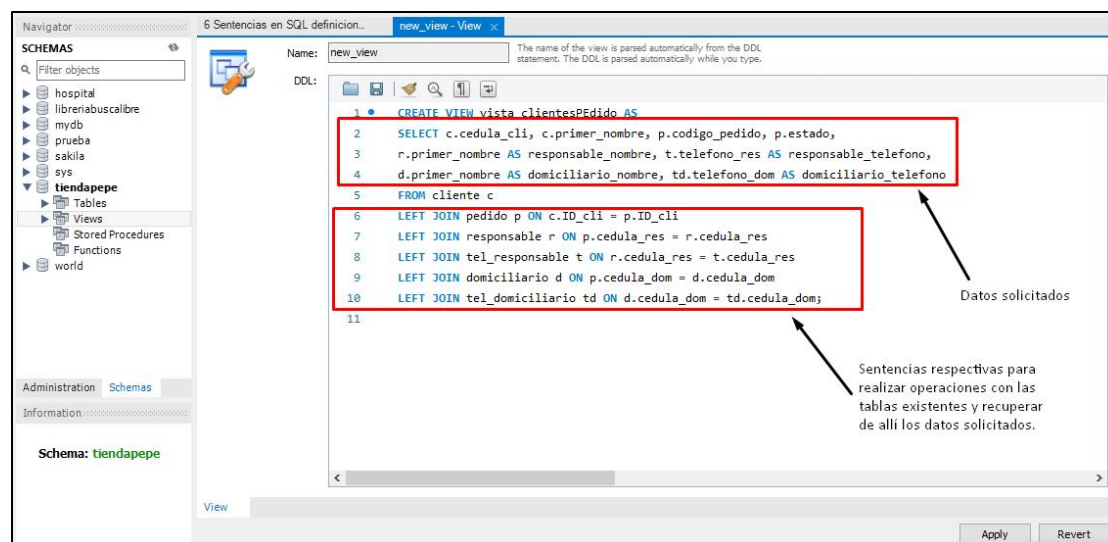
ID_zona	codigo_postal	localidad_zona
Zona1	111111	Localidad1
Zona2	222222	Localidad2
Zona3	333333	Localidad3
NULL	NULL	NULL

pedido 29 factura 30 responsable31 tel\_responsable32 producto\_proveedor33 zona 34 producto Apply Revert

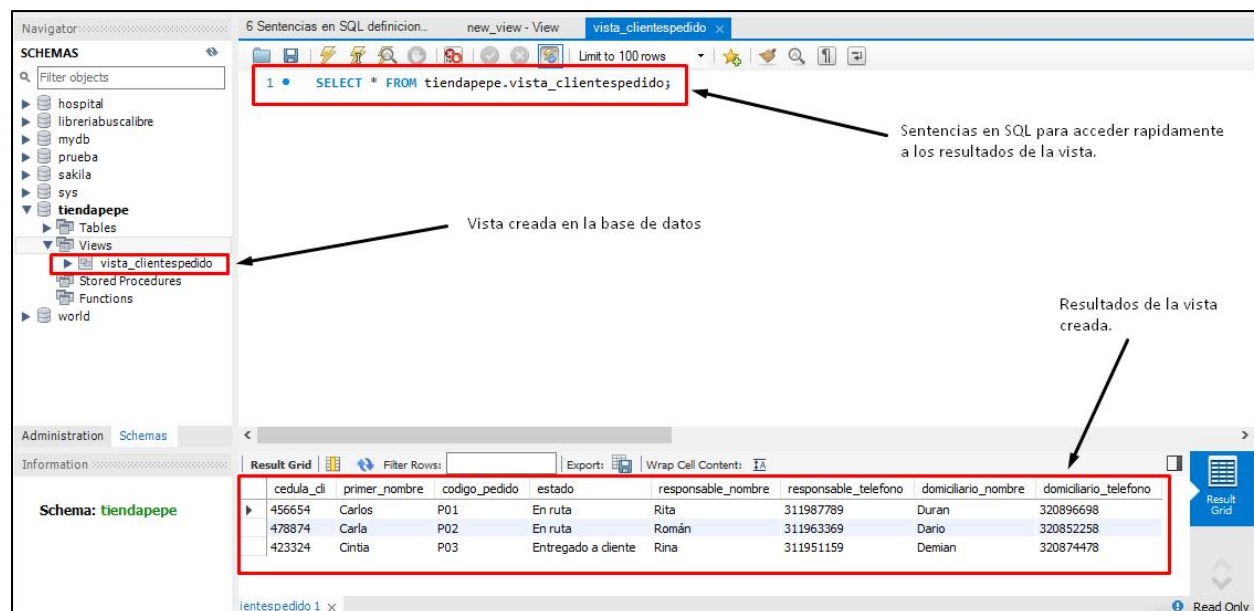
- Generar de 4 a 6 vistas donde se evidencie lo más importante de cada ejercicio (haga una selección muy responsable de la información realmente importante según el contexto).

## Vista Número 1: vista\_clientesPEdido

La vista número 1 esta asociada a la vista en la que don Pepe puede ver en tiempo real el estado de sus pedidos, es decir, en que estado esta cada pedido, quien es el encargado de empacar y quien es el encargado de entregar el domicilio. De esta forma si hay alguna novedad con el pedido don Pepe puede llamar inmediatamente a la persona respectiva para solicitar información. Por ejemplo si un cliente llama enojado diciendo que el servicio esta demorado y el estado del pedido es "En ruta" don Pepe puede saber inmediatamente a cual domiciliario llamar para solicitar más información y prestar soporte en caso de que sea necesario. En la siguiente imagen se muestran las sentencias usadas para realizar la vista deseada:



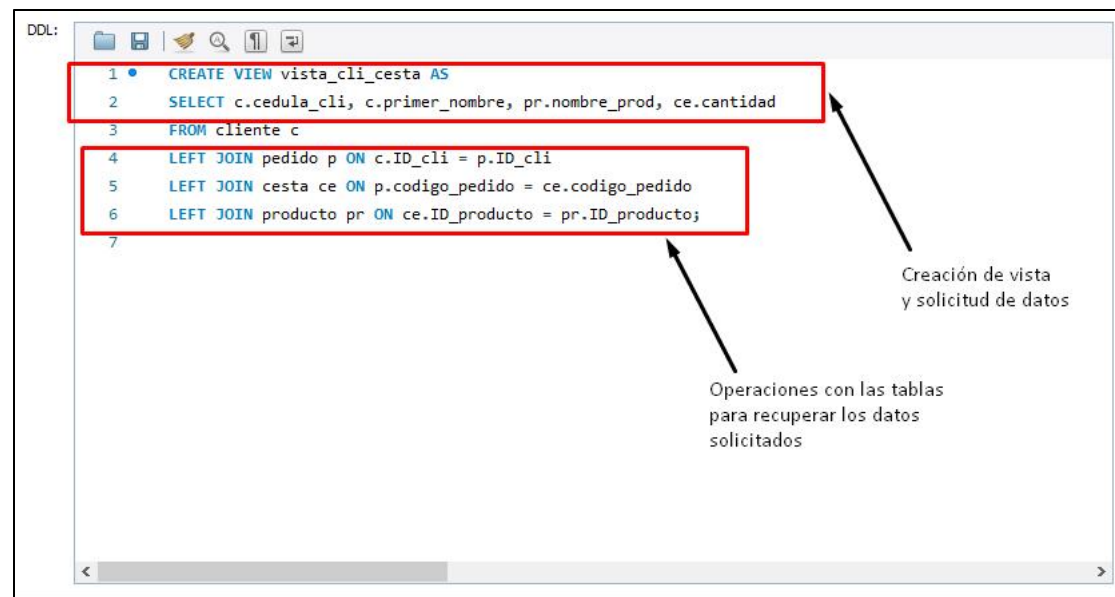
A continuación se encuentra la respuesta a la vista generada según los datos almacenados:



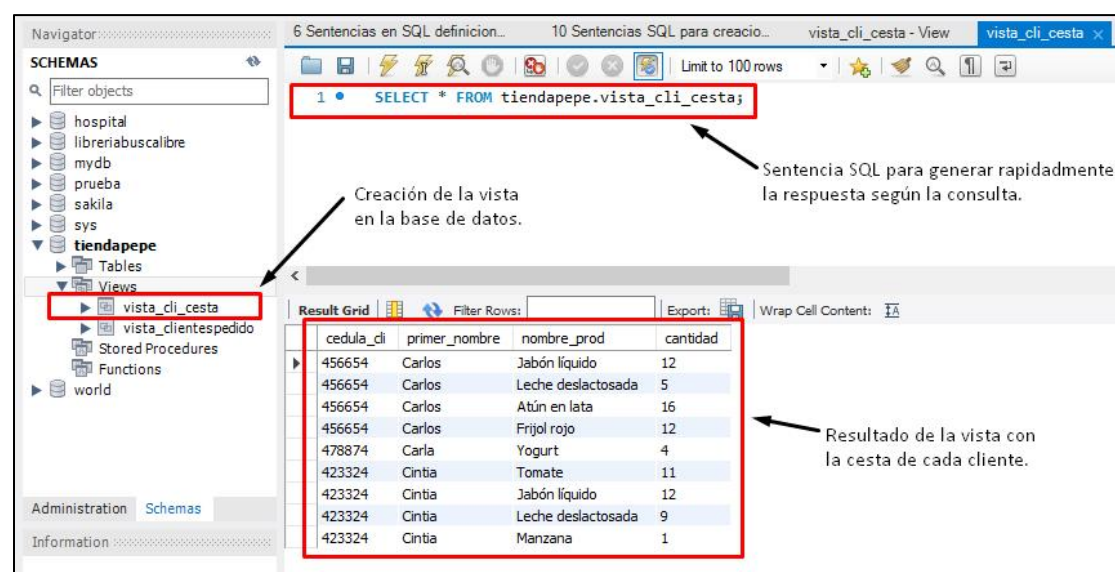
Para este caso se puede identificar que para el pedido de Carla el encargado del domicilio es Dario. Si Carla reportase alguna novedad en ese momento el número al que habría que llamar sería 320852258. De esa forma se podría prestar un pronto apoyo a Dario en caso de alguna novedad en la entrega.

## Vista Número 2: vista\_cli\_cesta

El responsable de empacar el pedido en el almacén debe tener a la mano la información necesaria para identificar la cesta de cada pedido. Es decir debe tener un listado con los nombres y la cantidad de productos que debe empacar para determinado pedido. En consecuencia se genera la siguiente vista:



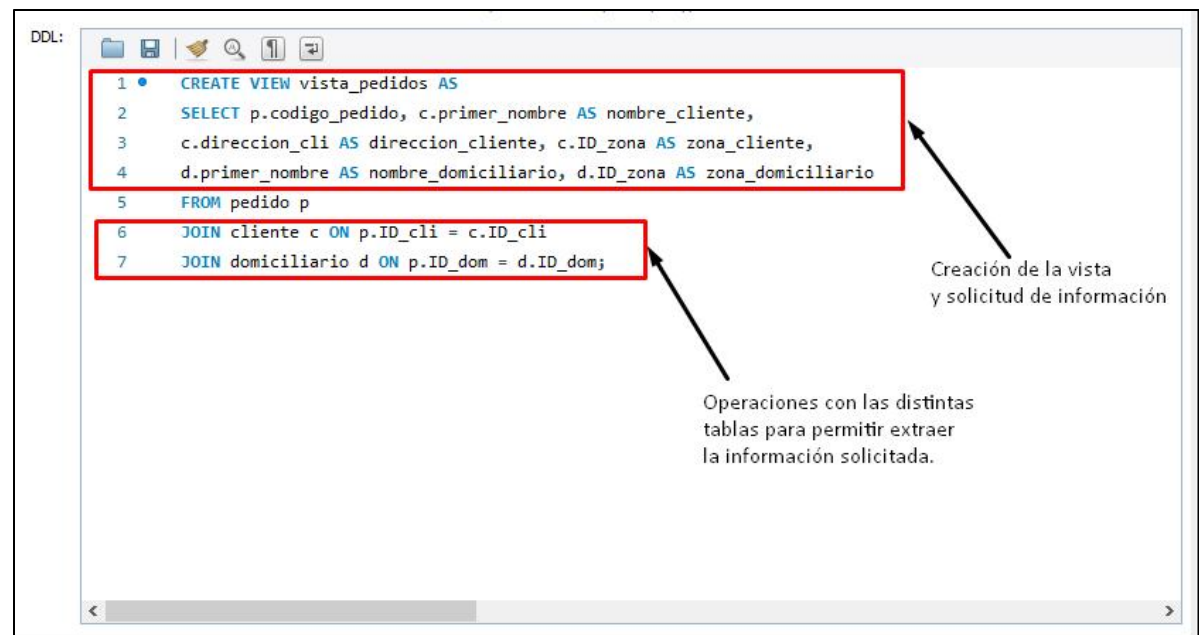
A continuación se encuentra la respuesta a la vista generada según los datos almacenados:



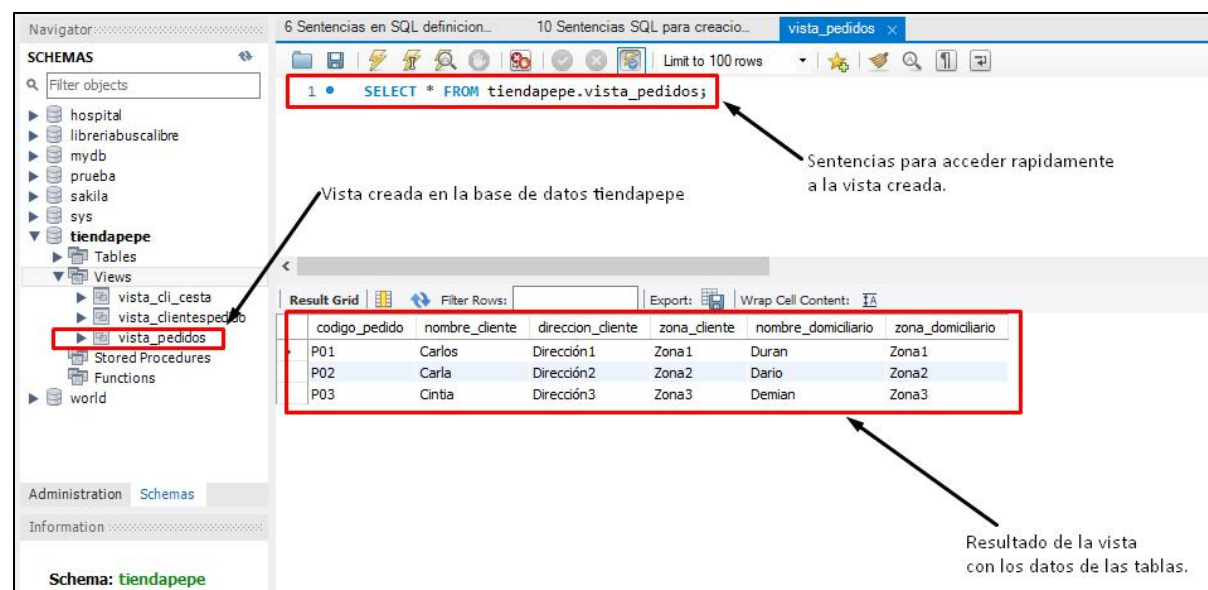
### Vista Número 3: vista\_pedidos

Don Pepe (o en la lógica del programa) debe revisar que el domiciliario escogido para un envío reparta en la misma zona que el cliente, ya que si el domiciliario escogido no está en la zona donde vive el cliente el pedido tendría que cancelarse. Debe haber una forma de identificar que la zona donde reparte el domiciliario coincida con la zona donde vive el cliente. En consecuencia se genera la siguiente vista:





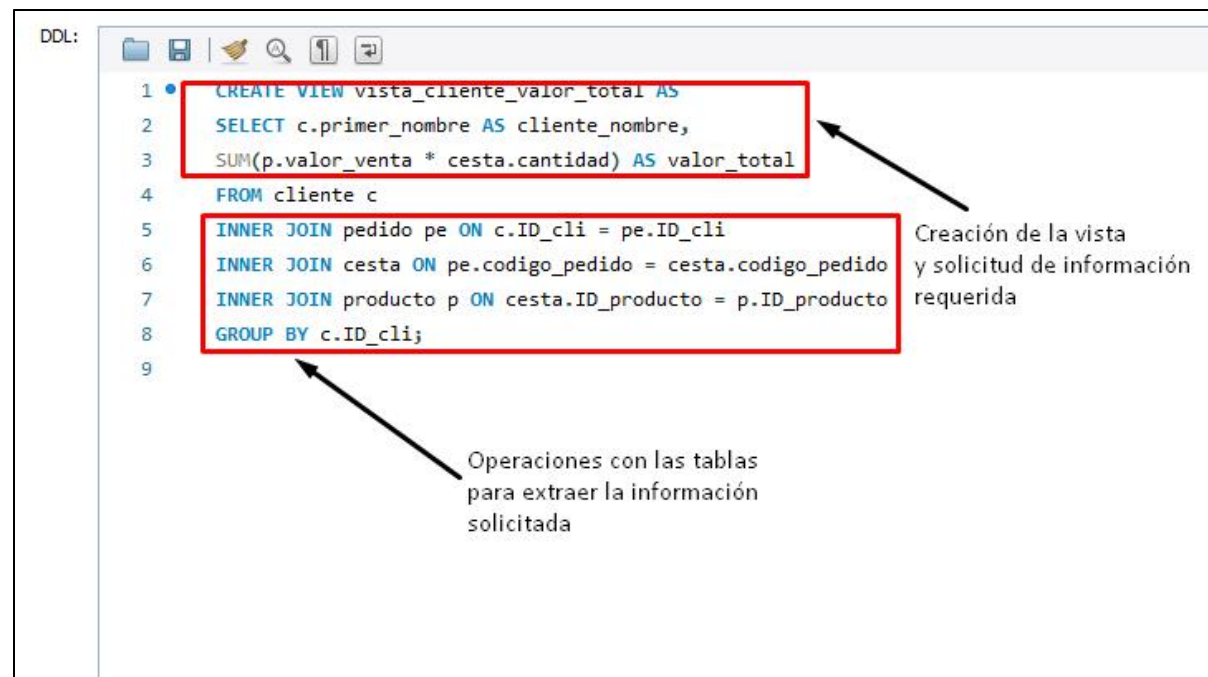
A continuación se encuentra la respuesta a la vista generada según los datos almacenados:



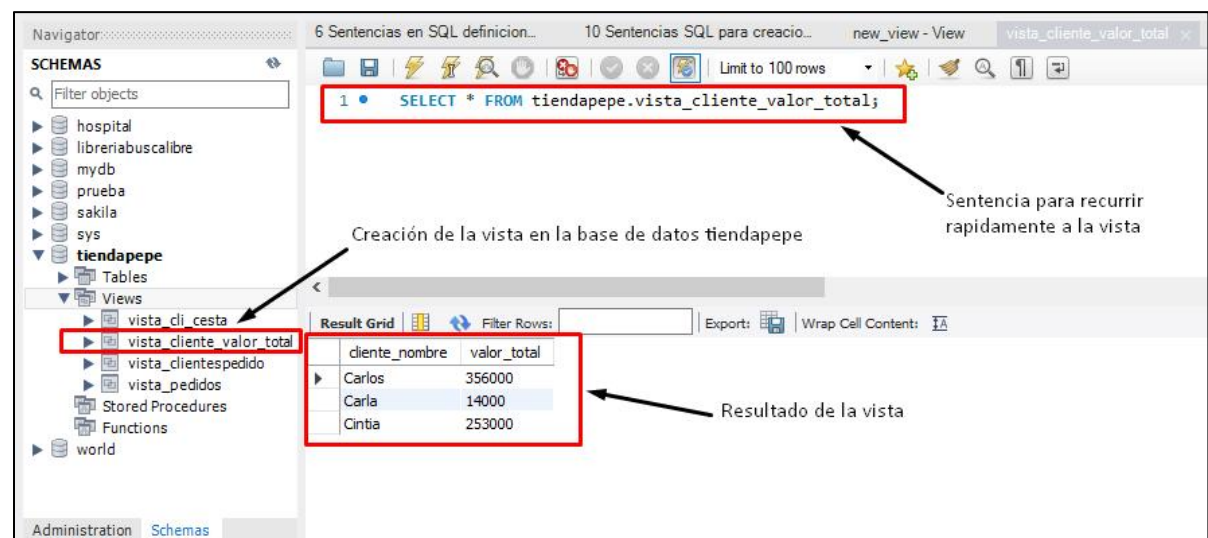
## Vista Número 4: vista\_cliente\_valor\_total

Para generar la factura al cliente se debe conocer el número total de productos vendidos y multiplicarlo por el costo de venta de cada producto para obtener el total del pedido. La información resultante es un insumo para ser informado al domiciliario, al cliente y para colocarlo en la factura que luego se le envía al cliente. En consecuencia se genera la siguiente vista:





A continuación se encuentra la respuesta a la vista generada según los datos almacenados:



## ● Generar al menos 4 procedimientos almacenados.

### Procedimiento 1: mostrar\_coincidencia\_zonas

Es requerido comparar las zonas en las cuales el cliente vive y el domiciliario realiza domicilios para este caso se crea un procedimiento almacenado para comparar las dos zonas y devolver un resultado indicando si es un “pedido aceptado” o “no es posible aceptar pedido” y devuelve una tabla como sigue:

The screenshot shows the SQL Developer interface with the 'mostrar\_coincidencia\_zonas' stored procedure being created. The procedure code is as follows:

```
CREATE DEFINER='root'@'localhost' PROCEDURE `mostrar_coincidencia_zonas`()
BEGIN
    CREATE TABLE temp_vista_pedidos (
        codigo_pedido VARCHAR(50),
        nombre_cliente VARCHAR(50),
        direccion_cliente VARCHAR(50),
        zona_cliente VARCHAR(50),
        nombre_domiciliario VARCHAR(50),
        zona_domiciliario VARCHAR(50),
        estado_pedido VARCHAR(50)
    );
    INSERT INTO temp_vista_pedidos
    SELECT p.codigo_pedido,
           c.primer_nombre AS nombre_cliente,
           c.direccion_cli AS direccion_cliente,
           c.ID_zona AS zona_cliente,
           d.primer_nombre AS nombre_domiciliario,
           d.ID_zona AS zona_domiciliario,
           CASE WHEN c.ID_zona = d.ID_zona THEN 'pedido aceptado' ELSE 'no es posible aceptar pedido' END AS estado_pedido
    FROM pedido p
    INNER JOIN cliente c ON p.ID_cli = c.ID_cli
    INNER JOIN domiciliario d ON p.cedula_dom = d.cedula_dom;
```

Annotations in the image:

- Creation of the stored procedure `mostrar_coincidencia_zonas`.
- Creation of the table `temp_vista_pedidos` to observe the results of the procedure.
- Comparison between the client's living zone and the delivery zone of the domiciliary.

En consecuencia se generan los siguientes resultados producto de la ejecución del procedimiento almacenado:

The screenshot shows the execution of the stored procedure `mostrar_coincidencia_zonas` and the resulting data grid. The SQL statement executed is:

```
SELECT * FROM tiendapepe.temp_vista_pedidos;
```

The resulting data grid is as follows:

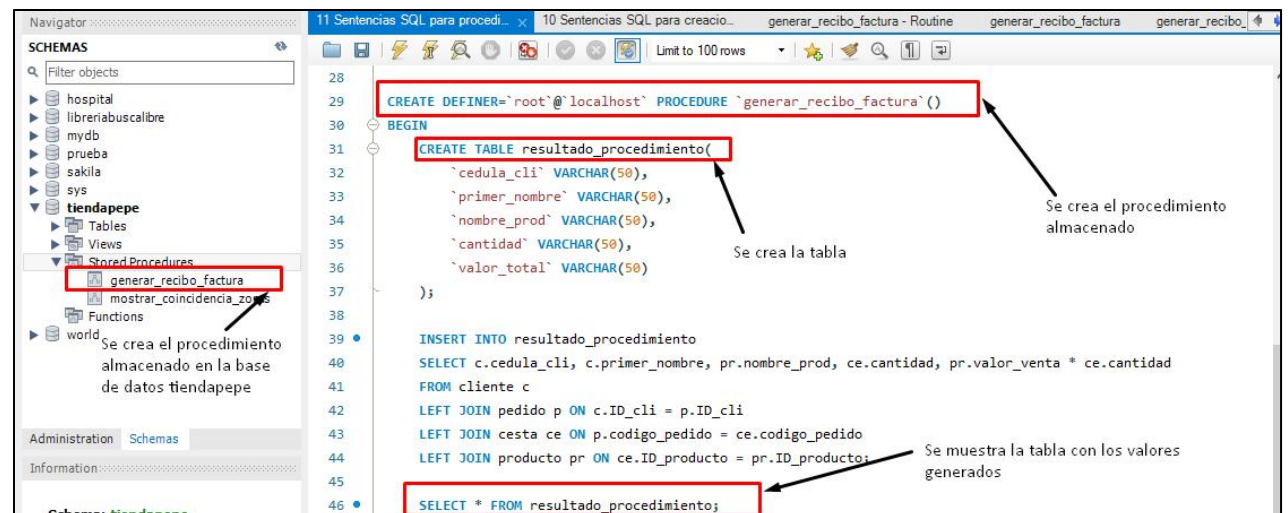
codigo_pedido	nombre_cliente	direccion_cliente	zona_cliente	nombre_domiciliario	zona_domiciliario	estado_pedido
P01	Carlos	Dirección1	Zona1	Duran	Zona1	pedido aceptado
P02	Carla	Dirección2	Zona2	Dario	Zona2	pedido aceptado
P03	Cintia	Dirección3	Zona3	Demian	Zona3	pedido aceptado

Annotations in the image:

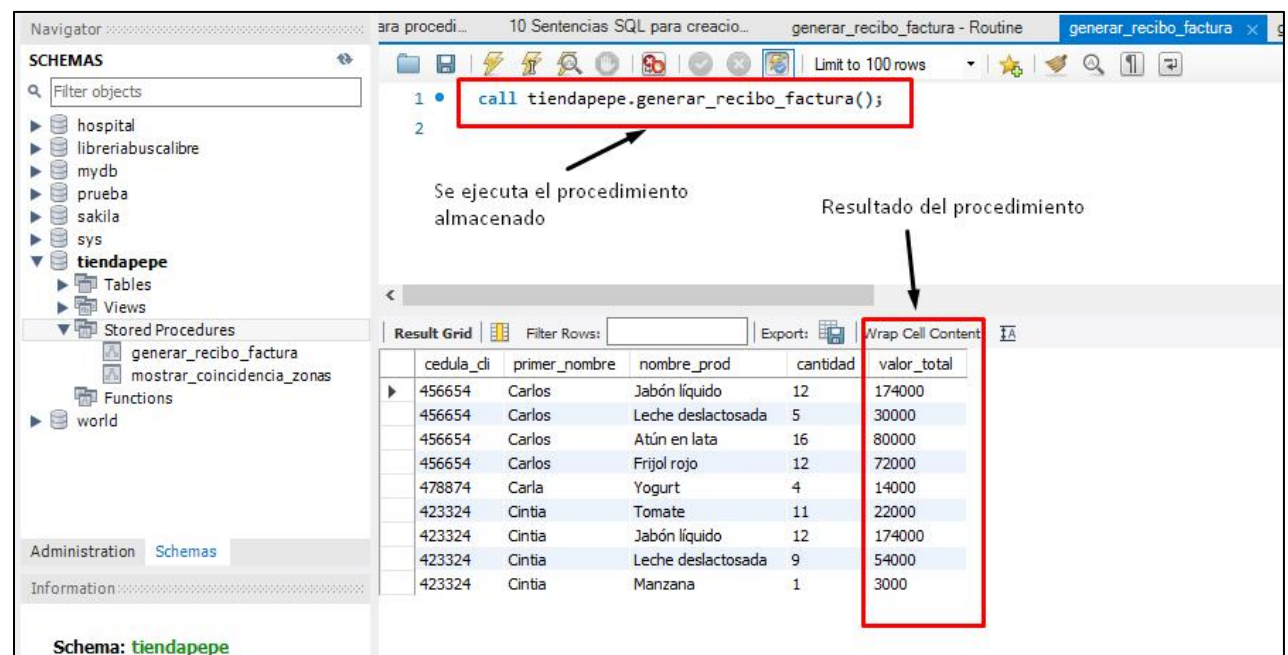
- Table created in the procedure `mostrar_coincidencia_zonas`.
- Procedure created in the database `tiendapepe`.
- Results where it is identified that the three orders are accepted because the living zones of the client and the delivery zone of the domiciliary coincide.

## Procedimiento 2: generar\_recibo\_factura

Con el propósito de generar una factura completa al cliente es conveniente dar el detallado de la cantidad de productos de su pedido y el valor de la multiplicación individual del número de productos por el valor de venta de cada producto. En consecuencia se generó el siguiente procedimiento almacenado:



Se generan los siguientes resultados producto de la ejecución del procedimiento almacenado:



### Procedimiento 3: generar\_tabla\_ganancias

Por otra parte es de interes de don Pepe el conocer la ganancia de cada grupo de productos vendido de cada pedido. De esta manera se puede observar cuales son los productos que generan más venta y más ganancia para la tienda. En consecuencia se generó el siguiente procedimiento almacenado para conocer la ganancia de cada producto:



The screenshot shows a SQL IDE with the following components:

- Navigator:** A tree view on the left showing the database schema. The 'Stored Procedures' folder is expanded, and 'generar\_tabla\_ganancias' is highlighted. A note below the schema says: "Se crea el procedimiento almacenado en la base de datos tiendapepe".
- SQL Editor:** The main window displays the SQL code for creating the procedure:
 

```

      48 CREATE DEFINER='root'@'localhost' PROCEDURE `generar_tabla_ganancias`()
      49 BEGIN
      50
      51 CREATE TABLE procedimiento_ganancias (
      52     `codigo_pedido` VARCHAR(50),
      53     `cedula_cli` VARCHAR(50),
      54     `nombre_prod` VARCHAR(50),
      55     `cantidad` VARCHAR(50),
      56     `precio_compra_total` VARCHAR(50),
      57     `valor_total` VARCHAR(50),
      58     `ganancia` VARCHAR(50)
      59 );
      60
      61 INSERT INTO procedimiento_ganancias
      62 SELECT p.codigo_pedido, c.cedula_cli, pr.nombre_prod, ce.cantidad,
      63        ce.cantidad * pr.valor_compra AS precio_compra_total,
      64        ce.cantidad * pr.valor_venta AS valor_total,
      65        (ce.cantidad * pr.valor_venta) - (ce.cantidad * pr.valor_compra) AS ganancia
      66 FROM cliente c
      67 LEFT JOIN pedido p ON c.ID_cli = p.ID_cli
      68 LEFT JOIN cesta ce ON p.codigo_pedido = ce.codigo_pedido
      69 LEFT JOIN producto pr ON ce.ID_producto = pr.ID_producto;
      70
      
```
- Annotations:**
  - An arrow points to the procedure definition line (48) with the text: "Se crea el procedimiento almacenado".
  - An arrow points to the INSERT statement (61) with the text: "Se crean las sentencias que realizan la suma producto de las columnas y realizan la resta final para dar a conocer la ganancia por número de productos vendido".

Se generan los siguientes resultados producto de la ejecución del procedimiento almacenado:

The screenshot shows the SQL IDE after executing the procedure. The 'Result Grid' displays the following data:

codigo_pedido	cedula_cli	nombre_prod	cantidad	precio_compra_total	valor_total	ganancia
P01	456654	Jabón líquido	12	156000	174000	18000
P01	456654	Leche deslactosada	5	20000	30000	10000
P01	456654	Atún en lata	16	64000	80000	16000
P01	456654	Frijol rojo	12	48000	72000	24000
P02	478874	Yogurt	4	10000	14000	4000
P03	423324	Tomate	11	16500	22000	5500
P03	423324	Jabón líquido	12	156000	174000	18000
P03	423324	Leche deslactosada	9	36000	54000	18000
P03	423324	Manzana	1	2500	3000	500

**Annotations:**

- An arrow points to the SQL statement `SELECT * FROM tiendapepe.procedimiento_ganancias;` with the text: "Se muestra el resultado del procedimiento".
- An arrow points to the 'procedimiento\_ganancias' table in the Navigator with the text: "Tabla creada en el procedimiento".
- Four arrows point to specific columns in the Result Grid:
  - Under 'codigo\_pedido': "Se identifica el código del pedido"
  - Under 'cantidad': "Se identifica la cantidad"
  - Under 'precio\_compra\_total': "Se identifica el precio total de compra de los productos"
  - Under 'ganancia': "Se da a conocer la ganancia por los productos vendidos"

## Procedimiento 4: calcular\_ganancia\_total

Por otra parte es de interés de don Pepe el conocer la ganancia total de todos los pedidos en curso. De esta manera se puede observar la ganancia total por la venta de todos los productos en la tienda. En consecuencia se generó el siguiente procedimiento almacenado para conocer la ganancia total por los pedidos:

```
CREATE DEFINER='root'@'localhost' PROCEDURE `calcular_ganancia_total`()
BEGIN
    DECLARE ganancia_total VARCHAR(50);
    SELECT SUM(ganancia) INTO ganancia_total FROM procedimiento_ganancias;
    CREATE TABLE Ganancias_total (ganancia_total VARCHAR(50));
    INSERT INTO Ganancias_total VALUES (ganancia_total);
    SELECT ganancia_total;
END
```

Se crea el procedimiento

Se suman los valores de la última columna de la tabla creada en el anterior procedimiento

Se da a conocer el valor total obtenido

Se genera el siguiente resultado producto de la ejecución del procedimiento almacenado:

The screenshot shows a database IDE with a 'Navigator' pane on the left and a main editor pane. The 'Navigator' pane shows a tree of database objects, including 'Stored Procedures'. The main editor pane shows the SQL code for the 'calcular\_ganancia\_total' procedure. The code is as follows:

```
1 • call tiendapepe.calcular_ganancia_total();
```

The code is highlighted with a red box. An arrow points to it with the text 'Ejecución del procedimiento almacenado'.

Below the code, the 'Result Grid' is visible. It contains one row with the following data:

ganancia_total
114000

The result grid is also highlighted with a red box. An arrow points to it with the text 'Resultado de la ejecución. Para este caso según los pedidos la ganancia total es de 114.000'.

- Generar al menos 4 triggers

### Trigger 1: Actualizar\_stock\_producto

Con este trigger se actualiza la cantidad de productos disponibles en la tabla producto en función de la cantidad solicitada por el cliente en la tabla cesta.

```
CREATE TRIGGER actualizar_stock_producto
AFTER INSERT ON cesta
FOR EACH ROW
BEGIN
    UPDATE producto
    SET unidades_disponibles = unidades_disponibles - NEW.cantidad WHERE ID_producto = NEW.ID_producto
END;
```

### Trigger 2: Actualizar\_stock\_producto

Con este trigger se impide eliminar un cliente de la base de datos tienda pepe si este tiene un pedido con estado pendiente.

```
CREATE TRIGGER impide Eliminacion_cliente
BEFORE DELETE ON cliente
FOR EACH ROW
BEGIN
    DECLARE num_pedidos INT
    SELECT COUNT(*) INTO num_pedidos FROM pedido WHERE ID_cli = OLD.ID_cli AND estado = 'Pendiente';
    IF num_pedidos > 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'No se puede eliminar el cliente debido a que tiene pedidos pendientes';
    END IF;
END;
```

### Trigger 3: verificar\_precio\_venta\_producto

Con este trigger se da un aviso a la persona que este ingresando nueva información sobre un producto que el precio de venta del producto al cliente es menor al precio de compra al proveedor.

```
CREATE TRIGGER verificar_precio_venta_producto
BEFORE INSERT OR UPDATE ON producto
FOR EACH ROW
BEGIN
    IF NEW.valor_venta < NEW.valor_compra THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'El precio de venta debe ser mayor o igual al precio de compra';
    END IF;
END;
```



## Trigger 4: borrar\_registros\_relacionados

Con este trigger cuando se elimina un pedido se borran los datos asociados en la tabla canasta, de esta forma se le indica al responsable en el almacén que ya no se deben empacar los productos asociados a esa cesta.

```
CREATE TRIGGER borrar_registros_relacionados
AFTER DELETE ON pedido
FOR EACH ROW
BEGIN
    DELETE FROM cesta WHERE codigo_pedido = OLD.codigo_pedido;
END;
```

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

Si estoy conforme con el resultado, el uso base de datos relacionadas permite generar interacciones entre las tablas, ahorrar tiempo en las consultas, generar información nueva a partir de los datos originales, evitar la repetición de registros, entre otras que con una base de datos no relacional no sería posible.