

RETO BD

Tienda Virtual Don pepe (Ejercicio C)

Don pepe quiere que sus clientes puedan realizar compras desde sus casas. El junto a su esposa tienen una cantidad domiciliarios conocidos que se encargan de llevar los pedidos a los clientes.

A continuación, se muestra la conversación que se tuvo con don pepe:

- ¡Veee mijo! yo quiero que más gente me compre los producticos, cuando llega un vecino nuevo a la cuadra yo lo apunto en un cuadernito. ¿Entiendo don pepe, y no le gustaría que le comprarán por internet?
- Eh hh mijo pues no es mala idea y que hago con mi clientela?
- Pues don pepe hacemos un video tutorial para usar la aplicación, y le pedimos una información a sus clientes indicando sus datos personales (ID, cedula, Nombre, Dirección, Teléfono, email y password) a través de un formulario de registro. Una vez registrado podrá acceder a la realización de pedidos con su email y su password.
- ¡eeeeee yo no te creo! ¿Así de fácil? ¿Como motilando calvos?
- Don pepe ojalá fuera así de sencillo déjeme le cuento mejor, Los productos que oferta el supermercado deben estar divididos en diversas categorías. Los datos necesarios para cada categoría son: nombre de la categoría, condiciones de almacenamiento (frío, congelado, seco) y observaciones. También debemos detallar la información de los productos (nombre, marca, origen, dimensiones (volumen y peso), una fotografía, la categoría y unidades disponibles). ¡no mijo eso me va salir muy caro con tanto detalle!
- don pepe todo lo contrario va aumentar mucho sus ganancias espéreme le cuento algo más, la aplicación permitirá visualizar un listado de productos ordenado por categoría, permitiendo seleccionar los productos que desee comprar mediante una caja de texto donde se indicará el número de unidades seleccionadas. La aplicación llevará la cuenta (cesta de la compra) de los productos que el cliente ha ido seleccionando. La aplicación permitirá también efectuar un pedido con todos los productos que lleve almacenados en su cesta de la compra. Los datos del pedido son: código del pedido, fecha del pedido, cliente, dirección de entrega, productos pedidos, importe total del pedido y datos de pago (número de tarjeta y fecha de caducidad).

Para poder generar un pedido se deberán dar dos situaciones:

- El cliente deberá pertenecer a una zona (Código Postal) donde existan domiciliarios. Un domiciliario se identifica mediante un nombre, número de matrícula de la furgoneta y zona donde reparte.
- Debe haber unidades suficientes por cada producto para satisfacer las demandas de cada pedido.

Una vez generado el pedido se mostrará al usuario una página con los datos de su pedido, se restarán del stock las unidades pedidas y se emitirá una nota de entrega a los responsables de almacén para que sirvan ese pedido.

Se pide:

- Indicar que ejercicio fue asignado
- Realizar el modelo E-R
- Realizar el modelo relacional
- Normalizar correctamente
- Escribir con sentencias SQL toda la definición de la base de datos.
- Escribir consultas que me permitan ver la información de cada tabla o de varias tablas (10).
- 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).
- Generar al menos 4 procedimientos almacenados.
- Generar al menos 4 triggers
- 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?
- documente muy bien su proceso (paso a paso) en un archivo PDF escriba todas las aclaraciones o especificaciones necesarias para realizar el ejercicio.

Solución:

Para comenzar se realiza una exhaustiva lectura del ejercicio y se logran identificar distintas entidades que son necesarias para la creación del diagrama entidad relación de la base de datos que permite el funcionamiento de la tienda de don pepe.

Entidades:

- Cliente
- Pedido
- Cesta de compras
- Producto
- Categoría
- Domiciliario
- Zona

Relaciones:

Un cliente puede crear uno o muchas cestas de compras y una cesta de compras puede ser creada por un cliente (1 N)

Un cliente puede estar en una zona, y una zona puede tener uno o muchos clientes (1 N).

Un cliente puede realizar uno o muchos pedidos, un pedido puede ser realizado por un cliente (1N)

Un pedido puede tener 1 cesta de compras, y una cesta de compras puede tener un pedido (1 1)

Una cesta de compras puede contener varios productos, y un producto puede estar presente en varias cestas de compras (N M)

```

    erDiagram
        Cliente ||--o{ Zona : Esta
        Cliente ||--o{ Pedido : Realiza
        Cliente ||--o{ CestaDeCompras : Crea
        Pedido ||--o{ CestaDeCompras : Tiene
        Pedido ||--o{ Domiciliario : Entrega
        CestaDeCompras ||--o{ Producto : Contiene
        Categoria ||--o{ Producto : Tiene
  
```

Diagrama de bases de datos para un sistema de comercio electrónico. El diagrama muestra las relaciones entre Clientes, Zonas, Pedidos, Productos, Cestas de compras, Categorías y Domiciliarios. Los atributos de cada entidad están listados en óvalos, y las relaciones se representan con los símbolos de los diagramas de entidad-relación.

Entidades y sus atributos:

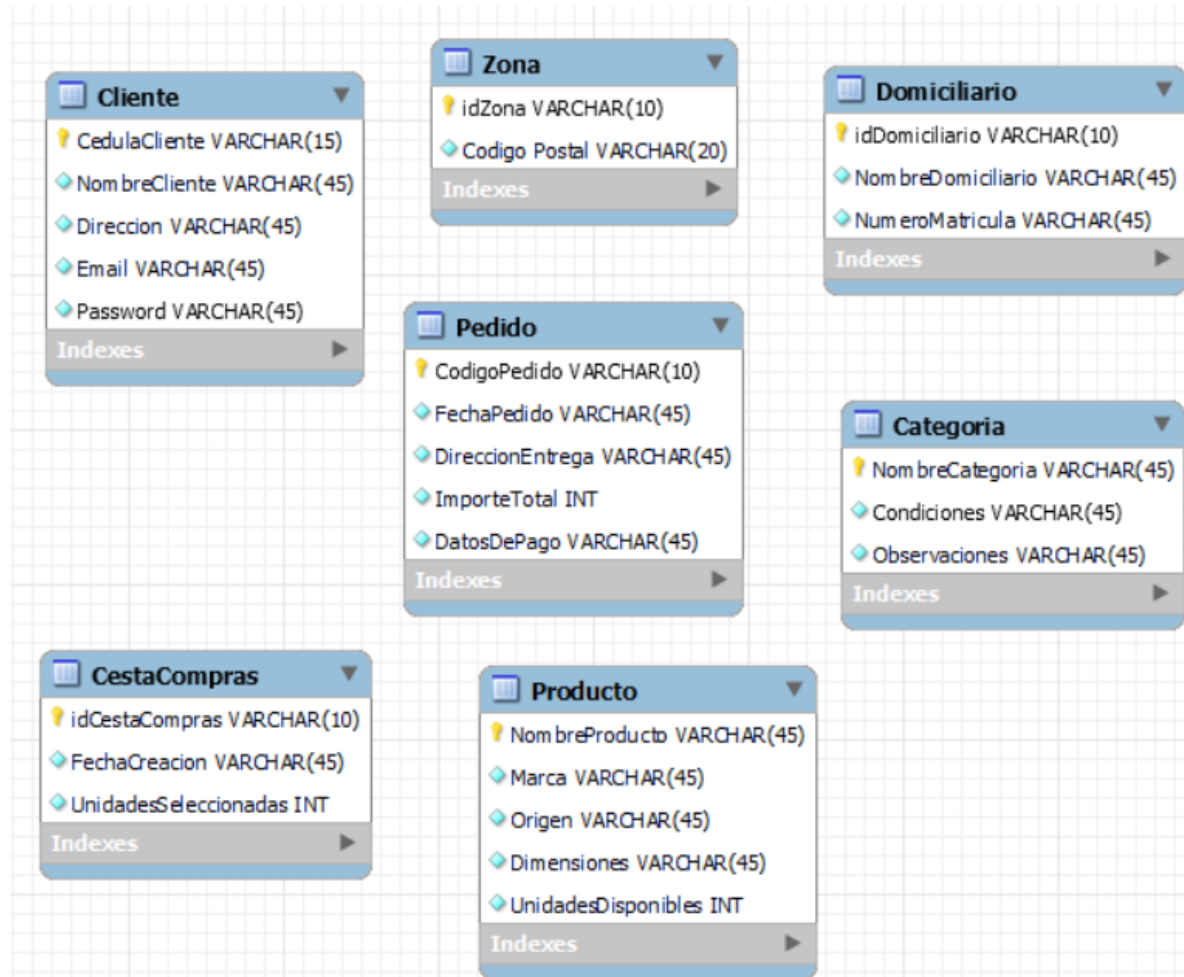
- Cliente:** Nombre, Cedula, Telefono, Direccion, Email, Password.
- Zona:** IDZona (clave primaria),Codigo postal.
- Pedido:** FechaPedido, Importe total, DatosdePago, Direccion entrega, CodigoPedido (clave primaria).
- Producto:** NombreProducto (clave primaria), Origen, Marca, Dimensiones, Fotografia, UnidadesDisponibles.
- Categoria:** NombreCategoria, Condiciones, Observaciones.
- Domiciliario:** IDDocimiliario (clave primaria), NombreDomiciliario, Numero de matricula.
- Cesta de compras:** IDCesta (clave primaria), FechaCracion.

Relaciones:

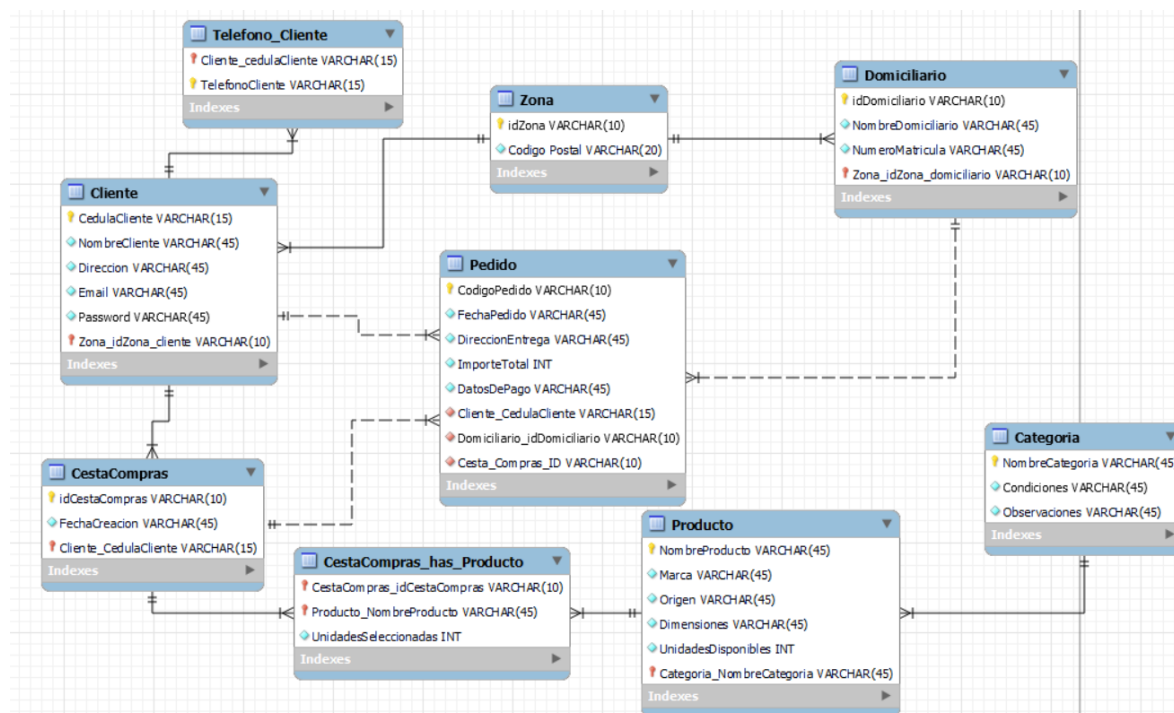
- Cliente y Zona:** Relación "Esta" (1:N).
- Cliente y Pedido:** Relación "Realiza" (1:N).
- Cliente y Cesta de compras:** Relación "Crea" (1:N).
- Pedido y Zona:** Relación "Esta" (1:N).
- Pedido y Domiciliario:** Relación "Entrega" (1:N).
- Pedido y Cesta de compras:** Relación "Tiene" (1:1).
- Cesta de compras y Producto:** Relación "Contiene" (N:M).
- Producto y Categoria:** Relación "Tiene" (1:N).

Una vez pensado y creado el Diagrama Entidad-Relación, se procederá a crear el modelo relacional de la base de datos, en esta ocasión se decidió utilizar MySQL Workbench por su practicidad.

Se crean las distintas tablas de la base de datos:



Una vez representadas las tablas con sus atributos, el siguiente paso es representar las relaciones y colocar las llaves foráneas en su lugar para materializarlas, así mismo se crean tablas adicionales para las relaciones de muchos a muchos en este caso productos y cesta de compras esta tabla tomara el atributo unidades seleccionadas de tal manera que cuando se seleccione un producto se especifique cuantas unidades se tomaron de las disponibles, y por ultimo se crea una tabla adicional para los teléfonos del cliente, resultando de esta manera:



Una vez hecho este proceso, procedo a pasar el chequeo de normalización con cada una de las formas normales, evaluando el modelo relacional, en este caso se cumplen los requisitos.

Normalización:

N1:

Normas	Estado
Todos los atributos tienen valores atómicos	Cumple
No hay atributos multivaluados	Cumple
No existen registros duplicados	Cumple
Se eliminaron todas las columnas repetidas	Cumple
Definir clave principal	Cumple

N2:

Normas	Estado
Cumple con la norma 1	Cumple
Los valores de las columnas dependen solo de la llave primaria	Cumple
Las tablas tienen una única llave primaria que las identifique	Cumple

N3:

Normas	Estado
Cumple la norma 2	Cumple
Los atributos no incluidos en la clave primaria no dependen transitivamente la clave primaria	Cumple

Siguiendo las indicaciones del ejercicio, Procedo a escribir las sentencias SQL, que me permiten definir mi base de datos conforme el modelo relacional planteado.

```
1 • CREATE SCHEMA IF NOT EXISTS `TiendaDonPepe` ;
2 • USE `TiendaDonPepe` ;
3
4   -- Crear tabla Zona
5 • CREATE TABLE IF NOT EXISTS Zona (
6     IdZona VARCHAR(10) PRIMARY KEY,
7     codigoPostal VARCHAR(20)
8   );
9   -- Crear tabla cliente
10 • CREATE TABLE IF NOT EXISTS Cliente (
11     CedulaCliente VARCHAR(15) PRIMARY KEY,
12     NombreCliente VARCHAR(45),
13     Direccion VARCHAR(45),
14     Email VARCHAR(45),
15     Password VARCHAR(45),
16     IdZona VARCHAR(10),
17     FOREIGN KEY (IdZona) REFERENCES Zona (IdZona)
18   );
```

```

19  -- Crear tabla Domiciliario
20  • CREATE TABLE IF NOT EXISTS Domiciliario (
21      idDomiciliario VARCHAR(10) PRIMARY KEY,
22      NombreDomiciliario VARCHAR(45),
23      NumeroMatricula VARCHAR(45),
24      idZona VARCHAR(10),
25      FOREIGN KEY (idZona) REFERENCES Zona(IdZona)
26  );
27  -- Crear tabla CestaCompra
28  • CREATE TABLE IF NOT EXISTS CestaCompras (
29      idCestaCompras VARCHAR(10) PRIMARY KEY,
30      FechaCreacion VARCHAR(45),
31      CedulaCliente VARCHAR(15),
32      FOREIGN KEY (CedulaCliente) REFERENCES Cliente(CedulaCliente)
33  );

  -- Crear tabla pedido
  • CREATE TABLE IF NOT EXISTS Pedido(
      CodigoPedido VARCHAR(10) PRIMARY KEY,
      FechaPedido VARCHAR(45),
      DireccionEntrega VARCHAR(45),
      ImporteTotal INT,
      DatosDePago VARCHAR(45),
      CedulaCliente VARCHAR(15),
      FOREIGN KEY (CedulaCliente) REFERENCES Cliente(CedulaCliente),
      idDomiciliario VARCHAR(10),
      FOREIGN KEY (idDomiciliario) REFERENCES Domiciliario(idDomiciliario),
      idCestaCompras VARCHAR(10),
      FOREIGN KEY (idCestaCompras) REFERENCES CestaCompras(idCestaCompras)
  );

  -- Crear tabla Categoria
  • CREATE TABLE IF NOT EXISTS Categoria (
      NombreCategoria VARCHAR(45) PRIMARY KEY,
      Condiciones VARCHAR(45),
      Observaciones VARCHAR(45)
  );

```

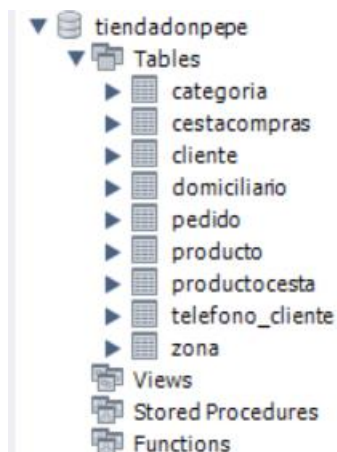


```

54      -- Crear tabla producto
55      CREATE TABLE IF NOT EXISTS Producto(
56      NombreProducto VARCHAR(45) PRIMARY KEY,
57      Marca VARCHAR(45),
58      Origen VARCHAR(45),
59      Dimensiones VARCHAR(45),
60      UnidadesDisponibles INT,
61      NombreCategoria VARCHAR(45),
62      FOREIGN KEY (NombreCategoria) REFERENCES Categoria(NombreCategoria)
63      );
64      -- Crear tabla ProductosCesta
65      CREATE TABLE IF NOT EXISTS ProductoCesta(
66      NombreProducto VARCHAR(45),
67      idCestaCompras VARCHAR(10),
68      UnidadesSeleccionadas INT,
69      PRIMARY KEY (NombreProducto, idCestaCompras),
70      FOREIGN KEY (NombreProducto) REFERENCES Producto(NombreProducto),
71      FOREIGN KEY (idCestaCompras) REFERENCES CestaCompras(idCestaCompras)
72      );
73      -- Crear tabla TelefonoCliente
74      CREATE TABLE IF NOT EXISTS Telefono_Cliente (
75      CedulaCliente VARCHAR(15),
76      TelefonoCliente VARCHAR(15),
77      PRIMARY KEY (CedulaCliente, TelefonoCliente),
78      FOREIGN KEY (CedulaCliente) REFERENCES Cliente(CedulaCliente)
79      );

```

De esta manera es posible crear la base de datos del negocio de don pepe:



Ahora que se ha logrado crear la base de datos, creare distintas consultas que permiten obtener información de la tabla.

- **Obtener cedula y nombre del cliente acompañado del id de la zona y el código postal**

```
1 -- Obtener cedula y nombre del cliente acompañado de el id de la zona y el codigo postal
2 • SELECT Cliente.CedulaCliente, Cliente.NombreCliente, Zona.IdZona, Zona.codigoPostal
3 FROM Cliente
4 INNER JOIN Zona ON Cliente.IdZona = Zona.IdZona;
```

Resultado:

	CedulaCliente	NombreCliente	IdZona	codigoPostal
▶	79246	jOSE	001	11515

- **Obtener datos del cliente junto con su número de teléfono o sus números de teléfono**

```
-- obtener datos del cliente junto con su numero de telefono o sus numeros de telefono
SELECT Cliente.CedulaCliente, Cliente.NombreCliente,
GROUP_CONCAT(Telefono_Cliente.TelefonoCliente SEPARATOR ', ') as Telefonos
FROM Cliente
INNER JOIN Telefono_Cliente ON Cliente.CedulaCliente = Telefono_Cliente.CedulaCliente
group by Cliente.CedulaCliente;
```

Resultado:

	CedulaCliente	NombreCliente	Telefonos
▶	79246	jOSE	3105465265, 3224691646

- **obtener el número de cédula de un cliente, su nombre, el nombre de los productos que compró y las unidades compradas**

```
• SELECT Cliente.CedulaCliente, Cliente.NombreCliente, ProductoCesta.NombreProducto, ProductoCesta.UnidadesSeleccionadas
FROM Cliente
INNER JOIN CestaCompras ON Cliente.CedulaCliente = CestaCompras.CedulaCliente
INNER JOIN ProductoCesta ON CestaCompras.idCestaCompras = ProductoCesta.idCestaCompras;
```

Resultado:

	CedulaCliente	NombreCliente	NombreProducto	UnidadesSeleccionadas
▶	79246	jOSE	Fabuloso	12

- Obtener la cedula del cliente, el nombre del cliente, el código del pedido y el nombre del domiciliario asignado

```
-- Obtener la cedula del cliente, el nombre del cliente, el código del pedido y el nombre del domiciliario asignado
SELECT Cliente.CedulaCliente, Cliente.NombreCliente, Pedido.CodigoPedido, Domiciliario.NombreDomiciliario
FROM Cliente
JOIN Pedido ON Cliente.CedulaCliente = Pedido.CedulaCliente
JOIN Domiciliario ON Pedido.idDomiciliario = Domiciliario.idDomiciliario;
```

Resultado:

	CedulaCliente	NombreCliente	CodigoPedido	NombreDomiciliario
▶	79246	jOSE	PED01	Fabian puerta

- Obtener una tabla con los siguientes atributos: CodigoPedido, FechaPedido, DireccionEntrega, ImporteTotal, y DatosDePago de pedido.

```
2 • SELECT CodigoPedido, FechaPedido, DireccionEntrega, ImporteTotal, DatosDePago
3 FROM Pedido;
```

Resultados:

	CodigoPedido	FechaPedido	DireccionEntrega	ImporteTotal	DatosDePago
▶	PED01	16/02/2023	CRA 140	120000	Tarjeta
*	NULL	NULL	NULL	NULL	NULL

- Obtener todas las categorías y las condiciones de almacenamiento.

```
24 -- Obtener todas las categorías y las condiciones de almacenamiento
25 • select NombreCategoria, Condiciones from categoria;
```

Resultados:

	NombreCategoria	Condiciones
▶	ASEO	Lugar Fresco
*	NULL	NULL

- Obtener ID de zona, código postal, y numero de domiciliarios en las zonas

```

26 -- Obtener ID de zona, codigo postal, y numero de domiciliarios en la zona
27 • SELECT Zona.IdZona, Zona.codigoPostal, COUNT(Domiciliario.idDomiciliario) as NumeroDomiciliarios
28 FROM Zona
29 LEFT JOIN Domiciliario ON Zona.IdZona = Domiciliario.idZona
30 GROUP BY Zona.IdZona, Zona.codigoPostal;

```

Resultados:

Result Grid			
Filter Rows:			
	IdZona	codigoPostal	NumeroDomiciliarios
▶	001	11515	1

- Obtener ID de zona, código postal, y número de clientes en las zonas

```

31 -- Obtener ID de zona, codigo postal, y numero de clientes en las zonas
32 • SELECT Zona.IdZona, Zona.codigoPostal, COUNT(cliente.CedulaCliente) as NumeroClientes
33 FROM Zona
34 LEFT JOIN cliente ON Zona.IdZona = cliente.idZona
35 GROUP BY Zona.IdZona, Zona.codigoPostal;

```

Resultado:

Result Grid			
Filter Rows:			
	IdZona	codigoPostal	NumeroDomiciliarios
▶	001	11515	1

- Obtener número de pedidos realizados por cada cliente

```

36 -- Obtener numero de pedidos realizados por cada cliente
37 • SELECT Cliente.CedulaCliente, Cliente.NombreCliente, COUNT(Pedido.CodigoPedido) as NumeroPedidos
38 FROM Cliente
39 LEFT JOIN Pedido ON Cliente.CedulaCliente = Pedido.CedulaCliente
40 GROUP BY Cliente.CedulaCliente, Cliente.NombreCliente;

```

Resultado:

Result Grid			
Filter Rows:			
	CedulaCliente	NombreCliente	NumeroPedidos
▶	79246	jOSE	1

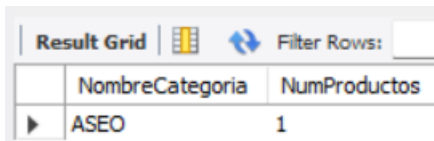
- Obtener número de productos en una categoría

```

41 -- Obtener numero de productos en una categoria
42 • SELECT NombreCategoria, COUNT(*) as NumProductos
43 FROM Producto
44 GROUP BY NombreCategoria;

```

Resultados:



	NombreCategoria	NumProductos
▶	ASEO	1

VISTAS

Una vez generadas distintas consultas que nos permitieron ver información de las distintas tablas, crearemos 5 vistas que permitan obtener la información mas importante de la base de datos de la Tienda Don Pepe.

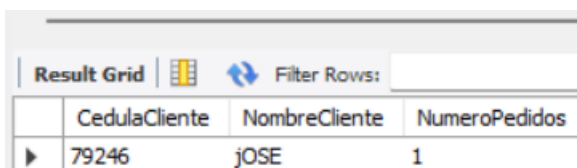
Numero de pedidos hechos por un cliente:

Como administrador de la tienda es importante saber que cliente realiza mas pedidos, ya que con esta información es posible ofrecerle mas ofertas a los clientes fieles maximizando las ganancias y haciendo crecer el negocio.

```
-- Vista Numero de pedidos por cada cliente
• create view NumeroDePedidosCliente as
  SELECT Cliente.CedulaCliente, Cliente.NombreCliente, COUNT(Pedido.CodigoPedido) as NumeroPedidos
  FROM Cliente
  LEFT JOIN Pedido ON Cliente.CedulaCliente = Pedido.CedulaCliente
  GROUP BY Cliente.CedulaCliente, Cliente.NombreCliente;

• select * from NumeroDePedidosCliente;
```

Resultado:



	CedulaCliente	NombreCliente	NumeroPedidos
▶	79246	jOSE	1

Detalles pedidos del cliente:

Con esta vista se responden las mayores preguntas en el mundo de las entregas, ¿Quién lo pidió?, ¿Cuál es el código de ese pedido?, y ¿Quién lo entrega?, es muy importante tener esta información presente para que la logística pueda funcionar de la manera mas fluida posible.

```
9 -- Obtener la informacion del cliente, su pedido y el domiciliario que se le asigno la entrega
.0 • create view detallesPedido as
.1 SELECT Cliente.CedulaCliente, Cliente.NombreCliente, Pedido.CodigoPedido, Domiciliario.NombreDomiciliario
.2 FROM Cliente
.3 JOIN Pedido ON Cliente.CedulaCliente = Pedido.CedulaCliente
.4 JOIN Domiciliario ON Pedido.idDomiciliario = Domiciliario.idDomiciliario;
.5 • select * from detallesPedido;
```

Resultados:

	CedulaCliente	NombreCliente	CodigoPedido	NombreDomiciliario
▶	79246	jOSE	PED01	Fabian puerta

Domiciliarios en zona:

Es importante saber cuantos domiciliarios hay disponibles en cada zona, y si en una zona donde un cliente pidió algo hay algún domiciliario, esto es importante debido a que uno de los requisitos para realizar un pedido es que en la zona de entrega haya domiciliarios.

```
-- Domiciliarios en la zona
• create view DomiciliariosenZona as
SELECT Zona.IdZona, Zona.codigoPostal, COUNT(Domiciliario.idDomiciliario) as NumeroDomiciliarios
FROM Zona
LEFT JOIN Domiciliario ON Zona.IdZona = Domiciliario.idZona
GROUP BY Zona.IdZona, Zona.codigoPostal;
• select * from DomiciliariosenZona;
```

Resultados:

	IdZona	codigoPostal	NumeroDomiciliarios
▶	001	11515	1

Clientes en la zona:

Así como es importante saber el numero de domiciliarios en las zonas, es importante saber el numero de clientes, ya que de esta manera es posible sectorizar la clientela, y de ser necesario contratar mas domiciliarios para las zonas con mayor flujo de compras.

```
5 -- Clientes en la zona
6 • create view ClientesenZona as
7 SELECT Zona.IdZona, Zona.codigoPostal, COUNT(cliente.CedulaCliente) as NumeroClientes
8 FROM Zona
9 LEFT JOIN cliente ON Zona.IdZona = cliente.idZona
10 GROUP BY Zona.IdZona, Zona.codigoPostal;
11 • select * from ClientesenZona;
```

Resultados:

	IdZona	codigoPostal	NumeroClientes
▶	001	11515	1

PROCEDIMIENTO

Siguiendo con los requerimientos del ejercicio se realizan 4 procedimientos que permiten insertar datos en las distintas tablas.

Insertar datos en cesta de compras:

```
1  -- Procedimientos
2  -- Crear cesta de compras
3  DELIMITER //
4  ● CREATE PROCEDURE insertarCestaCompras(IN p_idCestaCompras VARCHAR(10),
5    IN p_fechaCreacion VARCHAR(45), IN p_cedulaCliente VARCHAR(15))
6  ○ BEGIN
7    INSERT INTO CestaCompras(idCestaCompras, FechaCreacion, CedulaCliente)
8    VALUES (p_idCestaCompras, p_fechaCreacion, p_cedulaCliente);
9  ○ END //
10 DELIMITER ;
11 ● CALL insertarCestaCompras('CC001', '2023-02-16', '79246');
```

Result Grid

Filter Rows:

	idCestaCompras	FechaCreacion	CedulaCliente
▶	CC001	2023-02-16	79246
	CC11	16/02/2023	79246
●	NULL	NULL	NULL

Llenar cesta de compras:

```
    -- Llenar cesta de compras
    DELIMITER //
    ● CREATE PROCEDURE insertarProductoCesta (IN p_NombreProducto VARCHAR(45),
      IN p_IdCestaCompras VARCHAR(10), IN p_UnidadesSeleccionadas INT
    )
    ○ BEGIN
      INSERT INTO ProductoCesta (NombreProducto, idCestaCompras, UnidadesSeleccionadas)
      VALUES (p_NombreProducto, p_IdCestaCompras, p_UnidadesSeleccionadas);
    ○ END //
    DELIMITER ;
    ● call insertarProductoCesta("Fabuloso", "CC001", 10);
```

Result Grid

Filter Rows:

Edit:

	NombreProducto	idCestaCompras	UnidadesSeleccionadas
▶	Fabuloso	CC001	10
	Fabuloso	CC11	12
*	NULL	NULL	NULL

Insertar pedido:

```
23  -- realizar pedido
24  DELIMITER //
25  CREATE PROCEDURE insertarPedido (IN p_CodigoPedido VARCHAR(10), IN p_FechaPedido VARCHAR(45), IN p_DireccionEntrega VARCHAR(45),
26                                  IN p_ImporteTotal INT, IN p_DatosDePago VARCHAR(45), IN p_CedulaCliente VARCHAR(15),
27                                  IN p_idDomiciliario VARCHAR(10), IN p_idCestaCompras VARCHAR(10))
28  BEGIN
29      INSERT INTO Pedido (CodigoPedido, FechaPedido, DireccionEntrega, ImporteTotal, DatosDePago, CedulaCliente,
30                          idDomiciliario, idCestaCompras)
31      VALUES (p_CodigoPedido, p_FechaPedido, p_DireccionEntrega, p_ImporteTotal, p_DatosDePago, p_CedulaCliente,
32              p_idDomiciliario, p_idCestaCompras);
33  END //
34  DELIMITER ;
35  CALL insertarPedido('P001', '2022-02-16', 'Calle 123', 100, 'Efectivo', '79246', 'dom01', 'CC001')
```

Result Grid		Filter Rows:		Edit:	Export/Import:		Wrap Cell Content:	
	CodigoPedido	FechaPedido	DireccionEntrega	ImporteTotal	DatosDePago	CedulaCliente	idDomiciliario	idCestaCompras
▶	P001	2022-02-16	Calle 123	100	Efectivo	79246	dom01	CC001
	PED01	16/02/2023	CRA 140	120000	Tarjeta	79246	dom01	CC11
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Insertar cliente:

```
36  -- Insertar cliente
37  DELIMITER //
38  CREATE PROCEDURE insertarCliente(
39      IN pCedulaCliente VARCHAR(15),
40      IN pNombreCliente VARCHAR(45),
41      IN pDireccion VARCHAR(45),
42      IN pEmail VARCHAR(45),
43      IN pPassword VARCHAR(45),
44      IN pIdZona VARCHAR(10)
45  )
46  BEGIN
47      INSERT INTO Cliente (CedulaCliente, NombreCliente, Direccion, Email, Password, IdZona)
48      VALUES (pCedulaCliente, pNombreCliente, pDireccion, pEmail, pPassword, pIdZona);
49  END //
50  DELIMITER ;
51  CALL insertarCliente('123456789', 'Juan Perez', 'Calle 123', 'juanperez@gmail.com', 'password', '001');
```

Result Grid

Filter Rows:

Edit:

Export/Import:

	CedulaCliente	NombreCliente	Direccion	Email	Password	IdZona
▶	123456789	Juan Perez	Calle 123	juanperez@gmail.com	password	001
	79246	jOSE	140 CRA	AFSFSF	1231	001
+	NULL	NULL	NULL	NULL	NULL	NULL

TRIGGERS

A continuación, se crean distintos triggers que permitirán un monitoreo de la base de datos

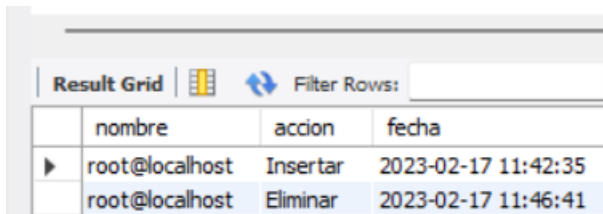
Como primera medida se crean dos tablas para monitorear la creación y eliminación de pedidos, clientes:

```
1 • create table control_de_cambios_pedidos(  
2     nombre varchar(50),  
3     accion varchar(30),  
4     fecha datetime default current_timestamp  
5 );  
6  
7 • create table control_de_cambios_clientes(  
8     nombre varchar(50),  
9     accion varchar(30),  
10    fecha datetime default current_timestamp  
11 );
```

Triggers para la tabla pedido:

```
13  -- Trigger insercion de datos pedidos  
14  DELIMITER //  
15 • create trigger trigger_ins_pedidos after insert on pedido  
16     for each row  
17  begin  
18  insert into control_de_cambios_pedidos values (  
19      user(), 'Insertar', now()  
20  );  
21  end;  
22  //  
23  DELIMITER ;  
24  -- Trigger eliminacion de datos pedidos  
25  DELIMITER //  
26 • create trigger trigger_DEL_pedidos after delete on pedido  
27     for each row  
28  begin  
29  insert into control_de_cambios_pedidos values (  
30      user(), 'Eliminar', now()  
31  );  
32  end;  
33  //  
34  DELIMITER ;
```

Resultados:



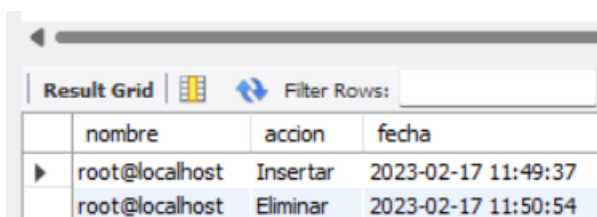
	nombre	accion	fecha
▶	root@localhost	Insertar	2023-02-17 11:42:35
	root@localhost	Eliminar	2023-02-17 11:46:41

Triggers para la tabla clientes:

```
--
-- Trigger insercion de datos clientes
DELIMITER //
• create trigger trigger_ins_clientes after insert on cliente
  for each row
  begin
    insert into control_de_cambios_clientes values (
      user(), 'Insertar', now()
    );
  end;
//
DELIMITER ;

-- Trigger eliminacion de datos clientes
DELIMITER //
• create trigger trigger_DEL_clientes after delete on cliente
  for each row
  begin
    insert into control_de_cambios_clientes values (
      user(), 'Eliminar', now()
    );
  end;
//
DELIMITER ;
```

Resultados:



	nombre	accion	fecha
▶	root@localhost	Insertar	2023-02-17 11:49:37
	root@localhost	Eliminar	2023-02-17 11:50:54

Llenado de tablas por medio de JAVA

Como parte del ejercicio, se procede a generar 50 registros para cada tabla de la base de datos, logrando utilizar herramientas como java Faker, la cual ayuda a generar palabras aleatorias de la categoría que se le especifique.

Para comenzar establecemos la conexión entre java y MySQL, para utilizar Java para generar consultas.

Este es un proceso sencillo, se realiza si:

- Se crea un proyecto en java, en este caso usaremos gradle para manejar las dependencias que se usaran, en este caso se usaran dos dependencias: un conector de MySQL y JAVA.FAKER.

```
dependencies {  
    testImplementation 'org.junit.jupiter:junit-jupiter-api:5.8.1'  
    testRuntimeOnly 'org.junit.jupiter:junit-jupiter-engine:5.8.1'  
    // https://mvnrepository.com/artifact/com.mysql/mysql-connector-j  
    implementation 'com.mysql:mysql-connector-j:8.0.32'  
    // https://mvnrepository.com/artifact/com.github.javafaker/javafaker  
    implementation 'com.github.javafaker:javafaker:1.0.2'  
}
```

- Se crea una interfaz donde se definen todos los métodos que se usaran para interactuar con la base de datos:

```
2 usages 1 implementation  
public interface DataBase {  
    2 usages 1 implementation  
    public void configureDataBaseConnection();  
    11 usages 1 implementation  
    public void executeSqlStatement();  
    10 usages 1 implementation  
    public ResultSet getResultSet();  
    2 usages 1 implementation  
    public void close();  
    1 usage 1 implementation  
    public void printResultSet() throws SQLException;  
    10 usages 1 implementation  
    public void executeSqlStatementvoid();  
}
```

- Se definen las constantes que serán usadas para hacer posible la conexión, en este caso el driver del conector, y la URL de la base de datos:

```
public class MySqlConstants {
    2 usages
    public static final String MY_SQL_JDBC_DRIVER= "com.mysql.cj.jdbc.Driver";
    2 usages
    public static final String CONNECTION_STRING= "jdbc:mysql://%s/%s?user=%s&password=%s";
}
```

- Se crea una clase en la cual se implementa la interfaz previamente creada:

```
3 usages
public class MySqlOperation implements DataBase {
```

- Se crean variables que serán utilizadas en cada uno de los métodos:

```
private Connection connection= null;
5 usages
private Statement statement= null;
7 usages
private ResultSet resultSet= null;
4 usages
private String sqlStatement;
3 usages
private String server;
3 usages
private String dataBaseName;
3 usages
private String user;
3 usages
private String password;
```

- Aparte de las variables que por defecto se dejaron en NULL, a las demás se les crean GETTERS Y SETTERS.

- Se implementan y se define el código para cada método que permite la conexión con MySQL.

Método para configurar la conexión.

```
@Override
public void configureDataBaseConnection() {
    try {
        Class.forName(MY_SQL_JDBC_DRIVER);
        connection= DriverManager.getConnection(
            String.format(CONNECTION_STRING,
                this.server,
                this.dataBaseName,
                this.user,
                this.password)
        );
        statement=connection.createStatement();
    }catch (Exception e){
        close();
        System.out.println(e.getMessage());
    }
}
```

Método para ejecutar las sentencias:

```
@Override
public void executeSqlStatement() {
    try {
        configureDataBaseConnection();
        resultSet = statement.executeQuery(sqlStatement);
    }catch (Exception e){
        System.out.println(e.getMessage());
    }
}
```

Método para retornar el resultado de las sentencias:

```
@Override
public ResultSet getResultSet() { return resultSet; }
```

Método para cerrar la conexión

```
@Override
public void close() { Complexity is 8 It's time to do s
    try{
        if(resultSet !=null){
            resultSet.close();
        }
        if(statement !=null){
            statement.close();
        }
        if(connection !=null){
            connection.close();
        }
    }catch (Exception e){
        System.out.println(e.getMessage());
    }
}
```

Método para imprimir los resultados de las sentencias:

```
@Override
public void printResultSet() throws SQLException { Complexity is 6 It's time to do something...
    ResultSetMetaData resultSetMetaData = resultSet.getMetaData();
    int totalColumnNumber = resultSetMetaData.getColumnCount();
    while(resultSet.next()) {
        for (int columnNumber =1; columnNumber <= totalColumnNumber; columnNumber++){
            if(columnNumber > 1){
                System.out.print(",\t");
            }
            String columnValue = resultSet.getString(columnNumber);
            System.out.print(resultSetMetaData.getColumnName(columnNumber)+ ": " + columnValue)
        }
        System.out.println("");
    }
}
```

Un segundo método para ejecutar las sentencias, que no retornan nada:

```
@Override
public void executeSqlStatementvoid() {
    try{
        configureDataBaseConnection();
        statement.execute(sqlStatement);
    }catch (Exception e){
        System.out.println(e.getMessage());
    }
}
```

- Ahora vamos a el main de nuestro programa y definimos unas variables que son mas especificas y permiten la conexión:

```
public class Main { Complexity is 9 It's time to do something...
    1 usage
    private static final String SERVER= "localhost";
    2 usages
    private static final String DATA_BASE_NAME="tiendadonpepe";
    1 usage
    private static final String USER="root";
    1 usage
    private static final String PASSWORD="Colombia.2022";
    1 usage
```

- Se crea una instancia de la clase en la cual se definieron los métodos anteriormente definidos.

```
58 usages
private static final MySqlOperation mySqlOperation = new MySqlOperation();
```

- Se crean dos métodos mas en el main.java uno para abrir la conexión especificando las variables antes creadas, y uno para cerrar la conexión:

```
1 usage
public static void openConnection(){
    mySqlOperation.setServer(SERVER);
    mySqlOperation.setDataBaseName(DATA_BASE_NAME);
    mySqlOperation.setUser(USER);
    mySqlOperation.setPassword(PASSWORD);
}
```



```

1 usage
public static void closeConnection(){
    mySqlOperation.close();
}

```

- Al invocar estos dos métodos en el main ejecutable se realiza la conexión.

```

▶ public static void main(String[] args) throws SQLException{
    openConnection();
    closeConnection();
}

```

De esta manera se finaliza la conexión, pero aun queda el paso mas importante, crear los métodos que me permitan generar 50 registros aleatorios para cada tabla.

Para crear cada método decidí no complicarme, por ello modele una clase para cada tabla, en esta clase cree un constructor y los respectivos GETTERS Y SETTERS.

Método insertar 50 zonas:

- Clase zona

```

public class Zona { Complexity is 3 Everything is cool!
    3 usages
    private String idZona;
    3 usages
    private String codigoPostal;

    1 usage
    public Zona(String idZona, String codigoPostal) {
        this.idZona = idZona;
        this.codigoPostal = codigoPostal;
    }
}

```

(Y sus GETTERS Y SETTERS)

Metodo:

```
public static void create50ZonasBD() { Complexity is 5 Everything is cool.
    Faker faker = new Faker();
    List<Zona> zonas = new ArrayList<>();
    for (int i = 0; i < 50; i++) {
        String idZona = faker.regexify("[A-Z]{2}\\d{3}");
        String codigoPostal = faker.address().zipCode();
        Zona zona = new Zona(idZona, codigoPostal);
        zonas.add(zona);
    }

    for (Zona zona : zonas) {
        mySqlOperation.setSqlStatement(String.format("INSERT INTO `tiendadonpepe`.`zona` (`IdZona`, `codigoPostal`) " +
            " VALUES ('%s', '%s');", zona.getIdZona(), zona.getCodigoPostal()));
        mySqlOperation.executeSqlStatementvoid();
    }
}
```

Resultado:

Result Grid		
	IdZona	codigoPostal
▶	001	11515
	AN959	07072-1151
	AV670	43936-5918
	BA321	49165
	BY412	87712-9200
	BY903	95592-4241
	DL005	66014
	DW766	78950-5969
	DZ953	65315
	FD289	56994-1864
	FI113	23070
	HH315	49950-3545
	HU716	75418
	IC668	79068-7474
	ID647	37906-3762
	IU096	83762

Método clientes:

Nuevamente se creo un modelo para la clase cliente:

```
public class Cliente { Complexity is 3 Everything is cool!
    3 usages
    private String cedulaCliente;
    3 usages
    private String nombreCliente;
    3 usages
    private String direccion;
    3 usages
    private String email;
    3 usages
    private String password;
    3 usages
    private String idZona;

    1 usage
    public Cliente(String cedulaCliente, String nombreCliente, String direccion,
        String email, String password, String idZona) {
        this.cedulaCliente = cedulaCliente;
        this.nombreCliente = nombreCliente;
        this.direccion = direccion;
        this.email = email;
        this.password = password;
        this.idZona = idZona;
    }
}
```

(Respectivos GETTERS Y SETTERS)

En esta ocasión debido a que clientes contaba con una llave foránea el método para insertar clientes aleatorios es un poco distinto, primero se crea una consulta dentro del método que retorna un array con las ids de todas las zonas creadas aleatoriamente previamente, luego se usa es array junto a un ciclo que permite crear distintos clientes sin tener problemas con las llaves foráneas.

Método:

- Consulta de las ids de las zonas:

```
public static void insertarClientes() throws SQLException{ Complexity is 5 Everything is cool!
    Faker faker = new Faker();
    List<String> zonaIds = new ArrayList<>();
    mySqlOperation.setSqlStatement("SELECT IdZona FROM tiendadonpepe.zona;");
    mySqlOperation.executeSqlStatement();
    ResultSet resultSet = mySqlOperation.getResultSet();
    while (resultSet.next()) {
        zonaIds.add(resultSet.getString(columnLabel: "IdZona"));
    }
    String[] zonaIdsArray = zonaIds.toArray(new String[0]);
}
```

- Creación de 50 usuarios aleatorios:

```
String[] zonaIdsArray = zonaIds.toArray(new String[0]);

List<Cliente> clientes = new ArrayList<>();
for (int i = 0; i < 52; i++){
    String cedula = faker.regexify("\\d{9}");
    String nombre = faker.name().fullName();
    String direccion = faker.address().fullAddress();
    String email = faker.internet().emailAddress();
    String password = faker.internet().password();
    String idZona = zonaIdsArray[i];
    Cliente cliente = new Cliente(cedula, nombre, direccion, email, password, idZona);
    clientes.add(cliente);
}




for(Cliente cliente: clientes){
    mySqlOperation.setSqlStatement(String.format("INSERT INTO `tiendadonpepe`.`cliente` " +
        " (`CedulaCliente`, `NombreCliente`, `Direccion`, `Email`, `Password`, `IdZona`) " +
        " VALUES ('%s', '%s', '%s', '%s', '%s', '%s')", cliente.getCedulaCliente(),
        cliente.getNombreCliente(), cliente.getDireccion(), cliente.getEmail(),
        cliente.getPassword(), cliente.getIdZona()));
    mySqlOperation.executeSqlStatementvoid();
}
}
```

Resultados:

Result Grid						
Filter Rows:						
Edit: Export/Import: Wrap Cell Content:						
	CedulaCliente	NombreCliente	Direccion	Email	Password	IdZona
▶	014290636	Lamar Gislason	0288 Buckridge Shoals, Aubreytown, MS 15256...	brendan.windler@hotmail.com	cxdgbfv8tx	BY412
	031087779	Dick Bernier	30818 Laurine Lock, West Floyd, TX 34065	thurman.wiegand@yahoo.com	rri7jnn8ldm62	OS891
	045760083	Nenita Huels	002 Moore Ways, North Roscoeland, TN 75254	ted.howell@yahoo.com	j91k8gttgp	BA321
	052819781	Kiersten Kirlin	Suite 496 08449 Annita Hills, DuBuqueberg, TX ...	seymour.brekke@yahoo.com	43bbavhbgpu	IC668
	076629228	Gaylene Littel	Suite 475 412 Walker Court, Faeborg, OR 40510	jamey.jacobi@yahoo.com	k2pwyewgo5g9	XW036
	094567779	Lady Beer	775 Erdman Trail, Crooksmouth, OK 37713	clarinda.okon@yahoo.com	jmmn643b	JM117
	123	Andres	calle 10	andres@gmail.com	12345	001
	123456789	Juan Perez	Calle 123	juanperez@gmail.com	password	001
	134010137	Willard Bergnaum	Suite 814 7145 Blake Causeway, Kirlinport, PA ...	eddie.metz@gmail.com	qwum42fry	LH570
	143927656	Arnulfo Kshlerin	3132 Labadie Mills, Galenstad, NV 40875	alison.oconnell@hotmail.com	qwqwwnya	VB701
	145929149	Salvador Hoppe	3295 Monahan Plaza, West Ula, OR 46040	jayson.stiedemann@yahoo.com	ce6k77b6owl4	OS105
	180260389	Daphne Hand	27153 Lashaun Road, East Diana, NY 01554-3856	kendall.casper@gmail.com	jk4eb2dowdyj	LH570
	182923893	Lenora Schulist V	1987 Antone Terrace, Sherylhaven, AK 06700	valentine.kihn@yahoo.com	krsi4gif	TQ009
	199423392	Rex Emard	0635 Edmond Locks, New Neelyview, UT 60052	roland.walter@gmail.com	pw1330w9e6b...	JQ263
	207668469	Mina Pfannerstill	Suite 360 322 Grant Inlet, Lannyton, VA 91375	lenora.witting@gmail.com	9vubyzfc4pj0mr4	XQ683
	211089007	Miss Rosaline Na...	1159 Dannie Road, Hoegershire, NV 43177-6210	florencio.cremin@hotmail.com	zk4is54303cp	TO100
	213455813	Verline Wiegand	Suite 498 5326 Russel Wall, South Heide, OR 4...	maybelle.purdy@hotmail.com	bsskslw2yt	XW036
	217028545	Cliff Farrell	1956 Jacobson Meadow, South Meeport, NY 63...	roscoe.schoen@gmail.com	a6uqt9uvb6mn...	YF596
	251887930	Mrs. Roseanna ...	Suite 560 25454 Rosenda Street, Lindsaytown, ...	gracie.flatley@yahoo.com	lab4mx1cm	RH613
	261050192	Hubert Hahn III	267 Heather Streets, South Edison, NE 36381	althea.lang@yahoo.com	4glfasgqm9wi	001
	266214842	Lelia Schultz	Suite 331 94243 Towne Village, West Jonahbur...	garland.oconner@hotmail.com	haf20wg2yxie0c	LE272
	277146967	Barton Willms	Apt. 058 35495 Toby Coves, Ezequielshire, MA ...	henriette.collier@gmail.com	dgvuw376jda5...	BY903
	285065678	Beula Johnston	075 Poulos Walks, West Sherley, LA 34520-5566	dakota.kohler@yahoo.com	iqbsf0sphkdb	NJ124
	294771078	Amara Lehner I	2618 Brekke Brooks, Port Rocco, TX 16612	isaiah.wisokv@hotmail.com	36dfalt5nu9nr36	RH613

Los métodos para las demás tablas, cuentan con la misma lógica, por ello omitiré la parte en la que muestro el código de cada uno de ellos en esta documentación y mostrare los resultados para las tablas restantes, por supuesto si desea ver ese código, los métodos de inserción se encuentran en el main.java, y los modelos de cada tabla se encuentran en la carpeta modelos.

Resultados domiciliarios:

Result Grid   Filter Rows: <input type="text"/> Edit:  				
	idDomiciliario	NombreDomiciliario	NumeroMatricula	idZona
▶	AB797	Benito Carroll	509838	QD024
	AK399	Ezequiel Gusikowski	321440	VB701
	AK945	Mrs. Marybelle Ledner	134760	LH570
	AN887	Mr. Rona Hagenes	631067	ID647
	AS505	Valentin Zieme	221602	TO100
	AX175	Tawanda Bednar PhD	250090	XW036
	BY084	Marilou Heller DVM	809292	JW993
	BZ928	Maryalice Green IV	385124	XW036
	CK274	Wenona Koelpin	400024	DW766
	CP010	Mrs. Un Johns	820757	BY412
	CU922	Doyle Stoltenberg	293978	RA108
	CW385	Byron Bode	244456	JU104
	CZ524	Gertude Harvey	994263	UX328
	DG767	Kareem Schuster	488274	OS105
	DJ784	Cherly Fay	421414	ID647
	DK439	Ms. Season Murray	071692	TQ009
	dom01	Fabian puerta	asd123	001
	DP780	Lamar Brown	629893	QD024
	DR998	Chauncey Considine	826607	NZ452
	FA667	Leena Schoen Jr.	649188	JW993
	FN604	Diane Stoltenberg	871409	WQ195
	FP097	Miss Nicola Kunde	233991	PX700
	GC876	Theo Johns	968539	VB575
	GH389	Shameka Huels II	201346	RP149
	GI059	Dr. Shonna Jakubowski	964856	TH754

Resultados Cesta Compras:

	idCestaCompras	FechaCreacion	CedulaCliente
▶	AF524	1965-01-06	926036282
	AM210	1979-04-01	744316027
	AS569	1983-01-17	682976830
	BG508	1965-10-23	052819781
	BV222	1960-08-16	725431234
	CH997	1996-10-18	942534021
	CK220	1997-02-27	266214842
	EL382	1991-02-27	655716294
	EM161	1964-07-10	374599004
	EV373	1985-02-20	145929149
	FS388	1982-02-16	868689254
	GD191	1973-08-09	261050192
	GI489	1983-02-08	840548794
	GT490	1962-04-08	422320312
	IB818	1960-08-09	301279790
	IC199	2003-08-13	199423392
	IK727	1985-12-26	844160396
	IV087	1978-12-12	996549477
	IV552	1987-07-07	442361259
	KM703	1982-06-05	79246
	LB571	1963-10-13	425358352
	LF324	1958-03-29	847589099
	LO446	1980-05-25	610119126

Resultados pedidos:

	CodigoPedido	FechaPedido	DireccionEntrega	ImporteTotal	DatosDePago	CedulaCliente	idDomiciliario	idCestaCompras
▶	AE031	2023-03-05	8998 Jame Loop, Rutherfordton, NE 93709	40	Tarjeta de crÃ©dito	868689254	DJ784	RT258
	BC006	2023-03-03	19309 Cornell Mission, Ashlifurt, SC 96829-7270	47	Tarjeta de crÃ©dito	285065678	IE742	MM788
	CX277	2023-03-03	Apt. 735 61777 Krajcik Harbors, Rueckercheste...	89	Tarjeta de crÃ©dito	79246	KA417	PZ306
	DD772	2023-03-10	526 Donetta Plaza, South Martinberg, WV 8213...	80	Efectivo	045760083	CP010	EV373
	DX121	2023-03-07	Apt. 477 33001 Roseanna Ports, Trinidadchest...	57	Tarjeta de crÃ©dito	704432572	QW094	IB818
	DZ704	2023-02-21	Apt. 877 860 Lesley River, North Hymanhaven,...	47	Efectivo	840548794	HE795	NM444
	EL840	2023-03-03	Suite 165 4558 Napoleon Park, Lake Vashti, NH ...	49	Efectivo	997405086	ND258	SI016
	EV516	2023-02-24	Apt. 336 5634 Lisha Street, East Jerome, VA 84...	48	Efectivo	996549477	WO810	EM161
	FG240	2023-03-17	126 Schaefer Square, Malisaland, LA 23288-7114	51	Efectivo	907589930	RG453	MB336
	FJ485	2023-03-01	78768 Prohaska Knoll, South Teofilaburgh, KS 6...	32	Tarjeta de crÃ©dito	779678477	WG936	TT205
	FZ735	2023-03-10	Apt. 314 218 Streich Inlet, Port Keven, CO 64136	56	Tarjeta de crÃ©dito	134010137	BY084	FS388
	GF153	2023-03-15	Apt. 897 3748 Schaden Mission, Swiftstad, NV ...	78	Tarjeta de crÃ©dito	926036282	NZ378	QN775
	HG135	2023-02-27	Apt. 728 003 Zieme Pike, Altenwerthborough, R...	80	Tarjeta de crÃ©dito	374599004	XO206	OD532
	HM587	2023-02-28	Apt. 356 6642 Ashli Cliffs, Nolandtown, ME 28008	95	Efectivo	432275773	RN287	EL382
	HP666	2023-03-12	518 Harris Isle, West Michelbury, KY 01042	36	Efectivo	610119126	LV214	TU895
	IG119	2023-03-15	Suite 498 37209 McDermott Point, Mdkinleypor...	40	Tarjeta de crÃ©dito	973046023	UO037	IK727
	IU869	2023-03-04	Apt. 361 32613 Omar Viaduct, Port Allenmouth,...	80	Tarjeta de crÃ©dito	655716294	ZR326	QK765
	JE653	2023-03-15	Suite 818 1933 Kirby Lake, Kerlukebury, WY 97...	61	Tarjeta de crÃ©dito	123456789	IN953	LT589
	KA929	2023-02-20	32825 Schoen Shore, East Dia, PA 73104-5681	10	Efectivo	052819781	XG738	MC355
	KJ414	2023-02-27	9434 Luigi Roads, Schultzburgh, CT 88479-4179	27	Tarjeta de crÃ©dito	725431234	OV907	OE821
	KQ939	2023-03-13	44916 Purdy Hill, Jenkinsstad, KY 78335	73	Efectivo	014290636	IX702	IC199
	KS216	2023-03-18	Suite 678 756 Crist Plain, East Ruthebury, KS 0...	79	Efectivo	425358352	ID104	UQ799
	LI967	2023-03-20	Apt. 186 03923 Crona Camp, Alvaroville, SC 46...	39	Efectivo	942534021	YL632	GT490
	ML435	2023-03-05	Apt. 808 3487 Graham Harbor, West Portia, MN...	21	Tarjeta de crÃ©dito	844160396	YW571	UL029
	MO241	2023-03-16	7589 Jacobi Plains, Efrainchester, GA 73893-1696	36	Efectivo	846621565	AK945	UD276

Resultados categoría:

	NombreCategoría	Condiciones	Observaciones
▶	ASEO	Lugar Fresco	Almacen fresco
	Automotive	Frio	Unde sequi recusandae.
	Automotive & Electronics	Seco	Possimus porro sunt ut ad qui accusamus corrupti.
	Automotive, Books & Electronics	Seco	Est facilis porro consequatur at at.
	Automotive, Kids & Music	Seco	Voluptatem voluptas qui assumenda.
	Automotive, Movies & Shoes	Seco	Sunt cum iure harum alias eligendi aut.
	Automotive, Movies & Toys	Seco	Ut nesciunt et ratione.
	Baby	Seco	Temporibus qui iure saepe inventore ut rem.
	Baby & Clothing	Congelado	Qui consequatur ut quis tempore.
	Baby & Garden	Seco	Nemo harum quos rerum ullam et et occaecati.
	Baby & Toys	Congelado	Qui ipsum tenetur.
	Baby, Computers & Games	Frio	Voluptatem enim reiciendis esse unde.
	Baby, Movies & Sports	Frio	Maxime ducimus cum delectus aut culpa maxime...
	Beauty	Frio	Ab ut ea facilis quo molestias.
	Beauty & Shoes	Seco	Ipsam autem corporis rerum distinctio aut est.
	Beauty, Home & Tools	Frio	Quia sit nihil iure est neque aut iusto.
	Books	Frio	Error qui velit consequatur in.
	Books & Games	Frio	Repellat dolor nihil nostrum.
	Books & Grocery	Seco	Qui quo minima unde vitae.
	Books, Movies & Sports	Seco	Quo magni aperiam dicta.
	Books, Movies & Toys	Seco	Quam eaque qui enim.
	Clothing	Seco	Nulla qui doloreque consectetur.
	Clothing, Health & Outdoors	Frio	Doloremque vel cupiditate vel incidunt vitae und...
	Computers	Seco	Autem labore hic quis.


Resultados productos:

Resultados						
Filter Rows: [] Col: [] Export/Import: [] Wrap Cell Contents: []						
	NombreProducto	Marca	Origen	Dimensiones	UnidadesDisponibles	NombreCategoría
▶	Aerodynamic Bronze Coat	Ernsner, Waters and Howe	nacional	90x20x50	50	Automotive, Movies & Toys
	Aerodynamic Bronze Pants	Wunsch, Olson and Dietrich	nacional	90x10x80	50	Industrial, Outdoors & Tools
	Aerodynamic Iron Lamp	Murphy-Schinner	nacional	80x30x70	50	Baby, Computers & Games
	Aerodynamic Leather Lamp	Bartoletti-Glover	nacional	20x20x90	50	Health & Shoes
	Aerodynamic Plastic Bench	Cassin-Hartmann	nacional	70x10x20	50	Grocery & Tools
	Aerodynamic Rubber Bag	Beier-Crona	internacional	10x00x70	50	Garden & Industrial
	Aerodynamic Silk Coat	Yundt, McCullough and Herzog	internacional	20x80x30	50	Electronics, Outdoors & Shoes
	Aerodynamic Wooden Pants	Auer-Glover	nacional	20x90x80	50	Clothing
	Awesome Copper Gloves	Bahringer and Sons	internacional	00x80x10	50	Books, Movies & Sports
	Awesome Granite Pants	Hammes, Trantow and Lubo...	internacional	80x70x80	50	Electronics & Music
	Awesome Paper Chair	Howe-Metz	internacional	80x20x60	50	Automotive
	Awesome Rubber Knife	Dietrich-Morar	internacional	80x10x20	50	Games
	Awesome Steel Table	Leuschke Inc	internacional	30x30x50	50	Computers
	Awesome Wooden Hat	Stracke-Upton	internacional	40x60x10	50	Electronics
	Durable Aluminum Table	Feeney, Graham and Schulist	internacional	00x40x40	50	Electronics
	Durable Bronze Clock	Barrows and Sons	nacional	20x70x30	50	Computers & Toys
	Durable Cotton Lamp	Ullrich, Sporer and Hagenes	internacional	10x90x80	50	Books, Movies & Sports
	Durable Linen Shirt	Lang-Boyer	nacional	90x30x90	50	Beauty, Home & Tools
	Durable Plastic Gloves	Lindgren Group	internacional	10x10x40	50	Computers & Games
	Durable Rubber Keyboard	Dietrich, Auer and Klein	nacional	60x00x20	50	Jewelry
	Durable Rubber Lamp	Langworth-Lemke	internacional	50x40x20	50	Industrial, Jewelry & Tools
	Enormous Copper Computer	Turcotte, Kozey and Beier	nacional	30x20x50	50	Games
	Enormous Iron Plate	Towne, Hoeger and Jacobi	internacional	30x40x90	50	Clothing, Health & Outdoors

Resultados producto cestas:

	NombreProducto	idCestaCompras	UnidadesSeleccionadas
▶	Aerodynamic Bronze Coat	CK220	21
	Aerodynamic Iron Lamp	QN775	46
	Aerodynamic Wooden Pants	FS388	5
	Awesome Copper Gloves	IK727	1
	Awesome Paper Chair	BG508	40
	Awesome Steel Table	UD276	26
	Durable Bronze Clock	IV087	31
	Durable Cotton Lamp	XM777	8
	Durable Linen Shirt	RT258	35
	Durable Plastic Gloves	NM444	21
	Enormous Iron Plate	AF524	9
	Enormous Marble Shirt	MM788	41
	Ergonomic Concrete Coat	CH997	15
	Ergonomic Iron Shirt	UK131	18
	Ergonomic Leather Table	UL029	43
	Ergonomic Paper Bottle	SY990	10
	Ergonomic Silk Bench	EV373	19
	Ergonomic Steel Plate	IV552	22
	Fabuloso	LT589	38
	Fantastic Marble Lamp	GT490	4
	Gorgeous Bronze Hat	OD532	26
	Gorgeous Concrete Shirt	XR120	4
	Gorgeous Concrete Table	TS118	20

Resultados teléfonos clientes:

Result Grid   Filter Rows: <input type="text"/>		
	CedulaCliente	TelefonoCliente
▶	014290636	551.225.1964
	031087779	(247) 208-5548
	045760083	1-333-726-5380
	052819781	155-131-6672
	094567779	017-694-4851
	123	561-850-3377
	123456789	574.988.5802
	134010137	1-691-152-1044
	145929149	(540) 610-9483
	180260389	234.603.1085
	199423392	431.785.9413
	261050192	1-224-123-5144
	266214842	331-803-1461
	277146967	1-723-504-4302
	285065678	(061) 627-2654
	301279790	(114) 531-7247
	302935460	954.191.3695
	374599004	(985) 044-7587
	422320312	316.826.8001
	425358352	380.745.8350
	432275773	225.794.6158

De esta manera llegamos al fin de este trabajo contestando la pregunta final.

¿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: La verdad estoy bastante conforme con este trabajo, pude realizar bastantes cosas que me permitieron afianzar mis conocimientos sobre bases de datos, y logre pensar desde cero una lógica para crear los registros aleatorios de la que me siento un poco orgulloso, seguramente con bases de datos no relacionales hubiese sido un poco mas sencillo crear los registros aleatorios, pero este desafío algo complejo me deja una sensación de satisfacción cuando me di cuenta que funcionaba.