



Indumentaria Femenina

ENVÍOS A TODO EL PAÍS % HASTA 30% DE DESCUENTO 12 CUOTAS SIN INTERÉS

GENESElore

HOME NOSOTROS CONTACTO PRODUCTOS LOGIN

REBAJAS
HASTA 40% OFF
3 CUOTAS SIN INTERÉS

Promos!!

Polera Beige
Talle unico
[Ver Producto](#)

Saco Camel
Talle unico
[Ver Producto](#)

Engomado negro
Talle unico (36 a 40)
[Ver Producto](#)

blanca y pintitas
Talle unico (1, 2 y 3)
[Ver Producto](#)

Preguntas Frecuentes

¿Cómo Compro?
En nuestra web o en nuestra tienda.

¿Hacen envíos?

¿Qué horarios atienden?

Curso: SQL
Comisión: 50055
Profesor: VILLENA REINES,
NANCY ELIZABETH.

Tutor: LO PRESTI, LEONEL
Alumno: VIVAS, LORENA SUSANA
Inicio: 17/10/2023 Final: 18/01/2024



GENESElore

Índice

Introducción.....	3
Diagrama de Entidad – Relación (DER)	4
Definición de Tablas	5
Vistas (View)	6
Funciones (Functions).....	7
Procedimientos (Stored Procedures).....	8
Disparadores (Triggers).....	9
Usuarios (Users)	10
Transacciones (TCL)	11
Restauración (Back Up).....	12
Reportes	12, 13 y 14
Herramientas y metodologías usadas	14

Introducción

Es un ecommerce de venta de ropa femenina. Todo el mundo podrá ver los productos, pero sólo podrán hacer la compra y solicitar el envío los clientes logueados.

Los envíos se realizarán a todas las provincias Argentinas.

Objetivo

Recabar información sobre que producto compro el cliente, detalles del producto nombre del producto, categoría, talle, cantidad, generar un detalle del pedido para posterior envío donde se acentuará a donde se envía quien lo transporta, quien es el cliente su dirección. Así como también control de las ventas realizadas importe fecha, destino.

Situación problemática

La necesidad de llevar un registro de los pedidos que nos hacen y las ventas realizadas para que funcione el negocio.

Modelo de negocio

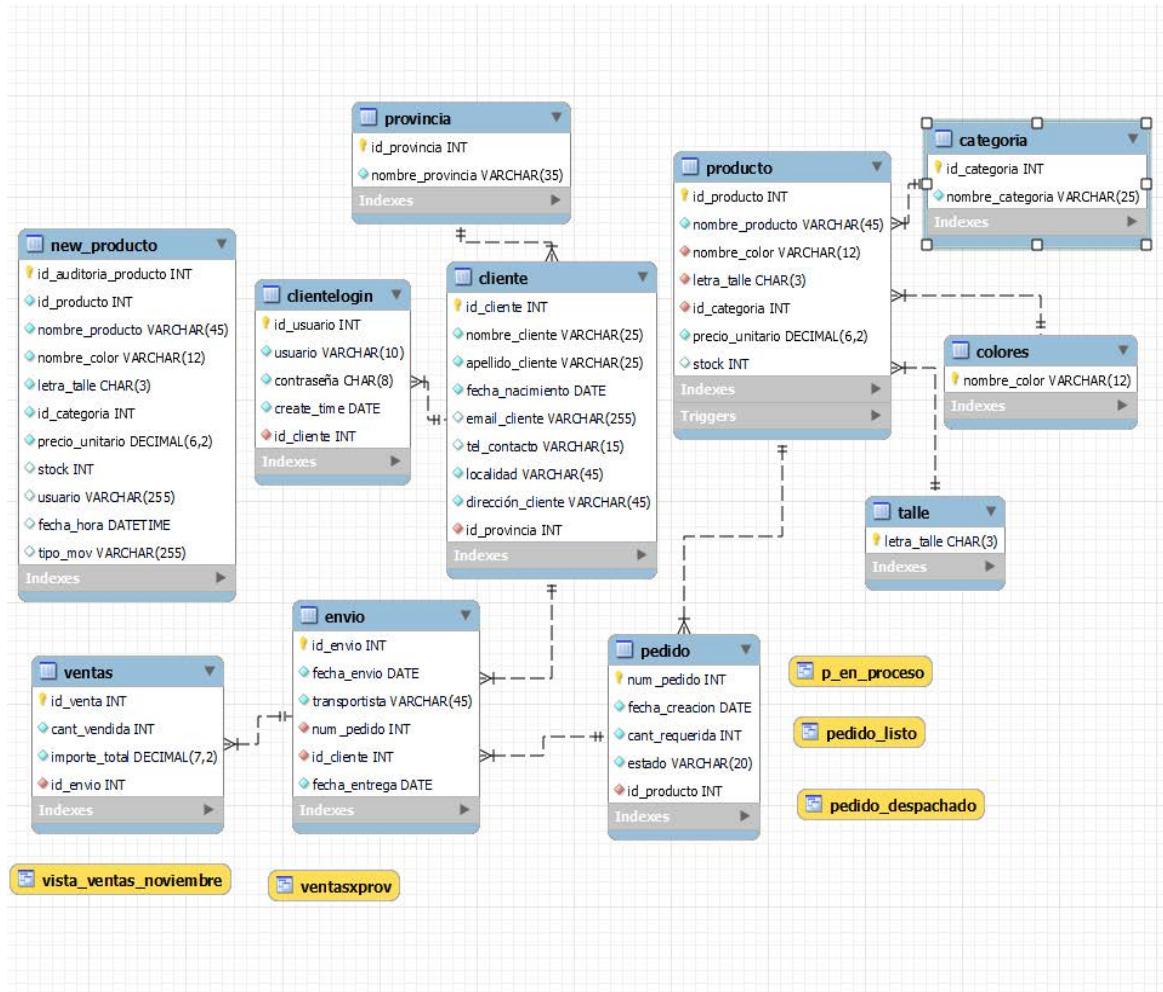
Crear una base de datos la que se utilizará en dicha página web lo que hará que esta sea una aplicación de venta de indumentaria femenina.

La base de datos abarcará los siguientes puntos:

- Detalle de los datos del cliente.
- Detalle de los productos.
- Estado de los pedidos.
- Destino de los envíos.
- Información de las ventas realizadas.



Diagrama de Entidad – Relación (DER)





Definición de Tablas

Se crearon las siguientes tablas. En las mismas podrá ver la descripción de cada campo con su tipo de dato y si el mismo es una llave primaria (PK) o una llave foránea (FK).

A	B	C	D	E	F	G	H	I
TABLA	CAMPO	CLAVE	TIPO DE DATOS	TABLA	CAMPO	CLAVE	TIPO DE DATOS	
categoria	id-categoría	PK	INT	pedido	num_pedido	PK	INT	
	nombre_categoria		VARCHAR(25)		fecha_creacion		DATE	
colores	nombre_color	PK	VARCHAR(12)		cant_requerida		INT	
					estado		VARCHAR(20)	
talle	letra_talle	PK	CHAR(3)		id_producto	FK	INT	
provincia	id-provincia	PK	INT	envio	id_envio	PK	INT	
	nombre_provincia		VARCHAR(35)		fecha_envio		DATE	
producto	id_producto	PK	INT		transportista		VARCHAR(45)	
	nombre_producto		VARCHAR(45)		num_pedido	FK	INT	
	nombre_color	FK	VARCHAR(12)		id_cliente	FK	INT	
	letra_talle	FK	CHAR(3)		fecha_entrega		DATE	
	id_categoria	FK	INT					
	precio_unitario		DECIMAL(6,2)					
	stock		INT					
cliente	id_cliente	PK	INT					
	nombre_cliente		VARCHAR(25)					
	apellido_cliente		VARCHAR(25)					
	email_cliente		VARCHAR(255)					
	fecha_nacimiento		DATE					
	tel_contacto		VARCHAR(15)					
	localidad		VARCHAR(45)					
	direccion_cliente		VARCHAR(45)					
	id_provincia	FK	INT					
clienteLogin	id_usuario	PK	INT					
	usuario		VARCHAR(10)					
	contraseña		CHAR(8)					
	create_time		DATE					
	id_cliente	FK	INT					



Creación de las Vistas

Se crearon las siguientes vistas para facilitar la visualización de datos para el armado de reportes y para a través de esta información tomar ciertas acciones.

Algunas de las vistas creadas son:

```
1  -- creación vistas--
2  use geneseindumentaria;
3
4  -- vista de la tabla pedidos que muestra aquellos pedidos que se encuentran en proceso--
5  CREATE OR REPLACE VIEW p_en_proceso AS
6  (SELECT * FROM pedido
7   WHERE estado like upper('en proceso'));
8
9  SELECT * FROM p_en_proceso;
10
11 -- vista de la tabla pedidos que muestra aquellos pedidos que se encuentran listos--
12 CREATE OR REPLACE VIEW pedido_listo AS
13 (SELECT * FROM pedido
14   WHERE estado like upper('listo'));
15
16 SELECT * FROM pedido_listo;
17
18 -- vista de la tabla pedidos que muestra aquellos pedidos que se encuentran despachados--
19 CREATE OR REPLACE VIEW pedido_despachado AS
20 (SELECT * FROM pedido
21   WHERE estado like upper('despachado'));
22
23 SELECT * FROM pedido_despachado;
24
25 -- vista que muestra los productos despachados(vendidos) en el mes de noviembre --
26 CREATE VIEW vista_ventas_noviembre AS
27 SELECT
28   *
29  FROM
30   envio
31 WHERE
32   MONTH(fecha_entrega) = 11 AND YEAR(fecha_entrega) = 2023;
33
34 SELECT * FROM vista_ventas_noviembre;
```

```
36  -- vista que muestra las ventas por provincia con nombre del cliente--
37 CREATE OR REPLACE VIEW ventasxprov AS
38 (SELECT
39   c.id_provincia,
40   c.nombre_cliente,
41   c.apellido_cliente,
42   c.id_cliente,
43   pr.nombre_provincia
44  FROM
45   cliente c
46  INNER JOIN
47   provincia pr ON c.id_provincia = pr.id_provincia
48 WHERE
49   c.id_cliente IN (SELECT id_cliente FROM envio);
50
51 SELECT * FROM ventasxprov;
52
```

Vistas que muestran el estado de los pedidos, aquellos que estan en proceso, listos o despachados.

Vistas que muestran pedidos despachados en cierto mes o fecha,

Vista que muestra las ventas por provincia.

```
75
76  -- VISTA pedidos despachados con nombre de producto--
77 CREATE OR REPLACE VIEW ped_despachado_nombreprod AS
78 SELECT
79   p.id_producto,
80   pd.nombre_producto,
81   pd.precio_unitario,
82   p.cant_requerida,
83   p.estado,
84   p.num_pedido
85  FROM
86   pedido p
87  INNER JOIN
88   producto pd ON p.id_producto = pd.id_producto
89  WHERE estado like upper('despachado')
90  AND fecha_creacion = '2024-01-20'
91 ;
92 SELECT * FROM ped_despachado_nombreprod;
93
94
```

```
53  -- vista que muestra detalle de las ventas(cliente, cuanto abono, cuando se entrego el producto, quien lo entrego,cantidad de producto--
54 CREATE VIEW vista_ventas_envio_pedido AS
55 SELECT DISTINCT
56   v.id_venta,
57   v.importe_total,
58   e.fecha_entrega,
59   e.id_envio,
60   e.transportista,
61   cl.direccion_cliente,
62   p.num_pedido,
63   p.cant_requerida
64  FROM
65   ventas v
66  JOIN
67   envio e ON v.id_envio = e.id_envio
68  JOIN
69   pedido p ON e.num_pedido = p.num_pedido
70  JOIN
71   cliente cl ON e.id_cliente = cl.id_cliente;
72
73  -- consultando la vista--
74 SELECT * FROM vista_ventas_envio_pedido;
```



Vista que muestra el pedido despachado buscando por nombre del producto y vista que muestra en panorama más amplio el detalle de las ventas. (Quien es el cliente, que producto compro, cuento abono, quien lo envió, cantidad de producto que se vendió.

Funciones (Functions)

Se desarrollaron las siguientes funciones para facilitar algunos cálculos según sea necesario.

- Función que dando un precio unitario y cantidad de productos te devuelve el precio total.
- Función que dando el id del cliente trae su Nombre y Apellido.

```
-- FUNCIONES--  
-- Función que dando un precio unitario y cantidad de productos me devuelve el precio total--  
DELIMITER //  
CREATE FUNCTION f_importeTotal(precio_unitario DECIMAL(6,2), cant_requerida INT)  
RETURNS DECIMAL(11,2)  
NO SQL  
BEGIN  
    DECLARE total_products DECIMAL(11,2);  
    SET total_products = precio_unitario * cant_requerida;  
    RETURN total_products;  
END//  
DELIMITER ;  
  
-- Función que dando el id del cliente me trae su Nombre y Apellido--  
DELIMITER //  
CREATE FUNCTION get_clienteName (param_id_cliente INT )  
RETURNS VARCHAR(50)  
READS SQL DATA  
BEGIN  
    DECLARE clienteName VARCHAR (50);  
    SET clienteName =(SELECT concat(nombre_cliente, ' ', apellido_cliente) from cliente where id_cliente = param_id_cliente) ;  
    RETURN clienteName;  
END  
//  
DELIMITER ;  
  
-- probando las funciones--  
SELECT f_importeTotal(3800.00, 2) AS total_compra;  
  
SELECT get_clienteName (2) AS nombre_clientebyid;
```



Procedimientos (Stored Procedures)

Se desarrollaron los siguientes procedimientos para la organización de tablas y registros.

3 Procedimientos:

- 1- Actualiza el precio en la tabla producto.
- 2-Actualiza el stock en la tabla producto.
- 3- Procedimiento que informa a logística para despacho del producto.

```
-- procedimiento que actualiza el precio en la tabla producto --
DELIMITER //
• CREATE PROCEDURE sp_actualizar_precio (
    IN p_id_producto INT,
    IN p_nuevo_precio DECIMAL(6,2))
BEGIN
    UPDATE producto
    SET precio_unitario = p_nuevo_precio
    WHERE id_producto = p_id_producto;
END //
DELIMITER ;
• -- probando procedimiento--
call sp_actualizar_precio (15,2700.00);

-- procedimiento que actualiza el stock en la tabla producto --
DELIMITER //
• CREATE PROCEDURE sp_actualizar_stock (
    IN p_id_producto INT,
    IN p_nuevo_stock INT)
BEGIN
    UPDATE producto
    SET stock = p_nuevo_stock
    WHERE id_producto = p_id_producto;
END //
DELIMITER ;
• -- probando procedimiento--
call sp_actualizar_stock (9,10);

-- procedimiento que informa a logística para despacho del producto --
DELIMITER //
• CREATE PROCEDURE sp_pase_transportista()
BEGIN
    SELECT * FROM pedido
    WHERE estado LIKE 'listo';
END //
DELIMITER ;
• -- probando procedimiento--
call sp_pase_transportista();
```



Disparadores (Triggers)

Además de modificar una tabla se crea una tabla nueva donde se registran dichos movimientos para poder ser consultada ante una duda.

Se desarrollaron los siguientes disparadores para tomar control de algunas tablas.

```
-- creación triggers--  
  
▶ CREATE TABLE new_producto(  
    id_auditoria_producto INT PRIMARY KEY AUTO_INCREMENT,  
    id_producto INT NOT NULL,  
    nombre_producto VARCHAR(45) NOT NULL,  
    nombre_color VARCHAR(12) NOT NULL,  
    letra_talle CHAR(3) NOT NULL,  
    id_categoria INT NOT NULL,  
    precio_unitario DECIMAL(6,2) NOT NULL,  
    stock INT,  
    usuario VARCHAR (255),  
    fecha_hora DATETIME,  
    tipo_mov VARCHAR(255)  
);  
  
▶ DROP TRIGGER IF EXISTS tr_add_new_producto;  
  
▶ CREATE TRIGGER tr_add_new_producto  
AFTER INSERT ON producto  
FOR EACH ROW  
INSERT INTO new_producto VALUES  
    ( DEFAULT, new.id_producto, new.nombre_producto, new.nombre_color, new.letra_talle, new.id_categoria, new.precio_unitario, new.stock,  
    USER(), NOW(),'se inserto un nuevo producto');  
  
    #Insercion en la tabla producto de un nuevo registro  
▶ INSERT INTO geneseindumentaria.producto VALUES( DEFAULT, 'buzo', 'blanco', 'M', '4', 3800.00, 3);  
  
▶ SELECT * FROM geneseindumentaria.new_producto;
```



Usuarios (Users)

Hemos generados 2 usuarios para hacer soporte de lectura y actualización de toda la base de datos. Solo fueron creados a modo de ejemplo ya que podremos crear N cantidad de usuario y con los permisos que así se requiera.

Los usuarios son los siguientes:

- '**usuario'@'localhost**: este usuario solo tendrá permiso de lectura de la base de datos.
- '**administrador'@'localhost**: este usuario podrá leer, insertar y actualizar información de la base de datos.

```
DROP USER 'administrador'@'localhost';
DROP USER 'usuario'@'localhost';
-- creación de usuarios --
#Creacion de usuario + contraseña
CREATE USER 'administrador'@'localhost' IDENTIFIED BY 'm3i4la!$';
CREATE USER 'usuario'@'localhost' IDENTIFIED BY 'delfin123';

-- permiso solo lectura sobre todas las tablas--
GRANT SELECT ON geneseindumentaria.* TO 'usuario'@'localhost';

-- permisos lectura, inserción y modificación ---
GRANT ALL ON geneseindumentaria.* TO 'administrador'@'localhost';

-- ninguno podra eliminar registros de las tablas--
REVOKE DELETE ON *.* FROM 'administrador'@'localhost';
REVOKE DELETE ON *.* FROM 'usuario'@'localhost' ;
```



Transacciones (TCL)

Se ejecutaron dos transacciones. Para ellas en primer lugar se debe deshabilitar la función autocommit y luego indicamos el comienzo de una transacción nueva.

```
-- TRANSACCION que de acuerdo a la cantidad requerida de producto descuenta el stock--  
-- Inicio la transacción  
START TRANSACTION;  
    UPDATE pedido  
        SET cant_requerida = 2  
    WHERE id_producto = 11;  
        SELECT * FROM pedido;  
/* ROLLBACK;*/  
        SELECT * FROM producto;  
-- Hacer el registro en la tabla producto  
UPDATE producto  
SET stock = 6  
WHERE id_producto = 11;  
-- Confirmar la transacción  
COMMIT;  
    SELECT * FROM producto;  
  
-- TRANSACCION que registra como ventas los pedidos despachados--  
-- Inicio la transacción  
• START TRANSACTION;  
• INSERT INTO ventas (id_venta, cant_vendida, importe_total, id_envio)  
VALUES (null,1,'8500.00', 5),  
        (null,1,'8500.00', 6),  
        (null,1,'4000.00', 7),  
        (null,1,'8500.00', 8)  
        ;  
• SAVEPOINT primeros4;  
• INSERT INTO ventas (id_venta, cant_vendida, importe_total, id_envio)  
VALUES (null,1,'1200.00', 9),  
        (null,1,'8500.00', 10),  
        (null,1,'3800.00', 11),  
        (null,1,'8500.00', 12)  
        ;  
• ROLLBACK TO SAVEPOINT primeros4;  
  
-- release SAVEPOINT primeros4;
```

Restauración (Back Up)

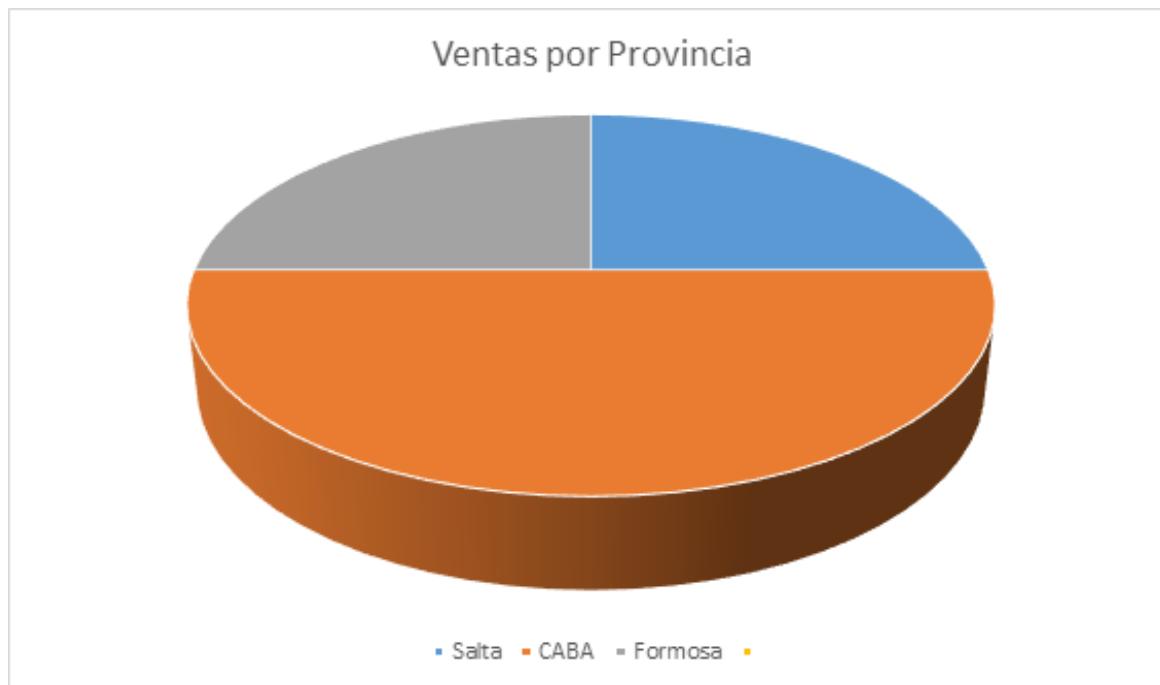
Se realizo el respaldo de toda la base datos antes mencionada con todas las tablas y su contenido, vistas, funciones, disparadores, usuario y transacciones, etc.

Reportes/Informes

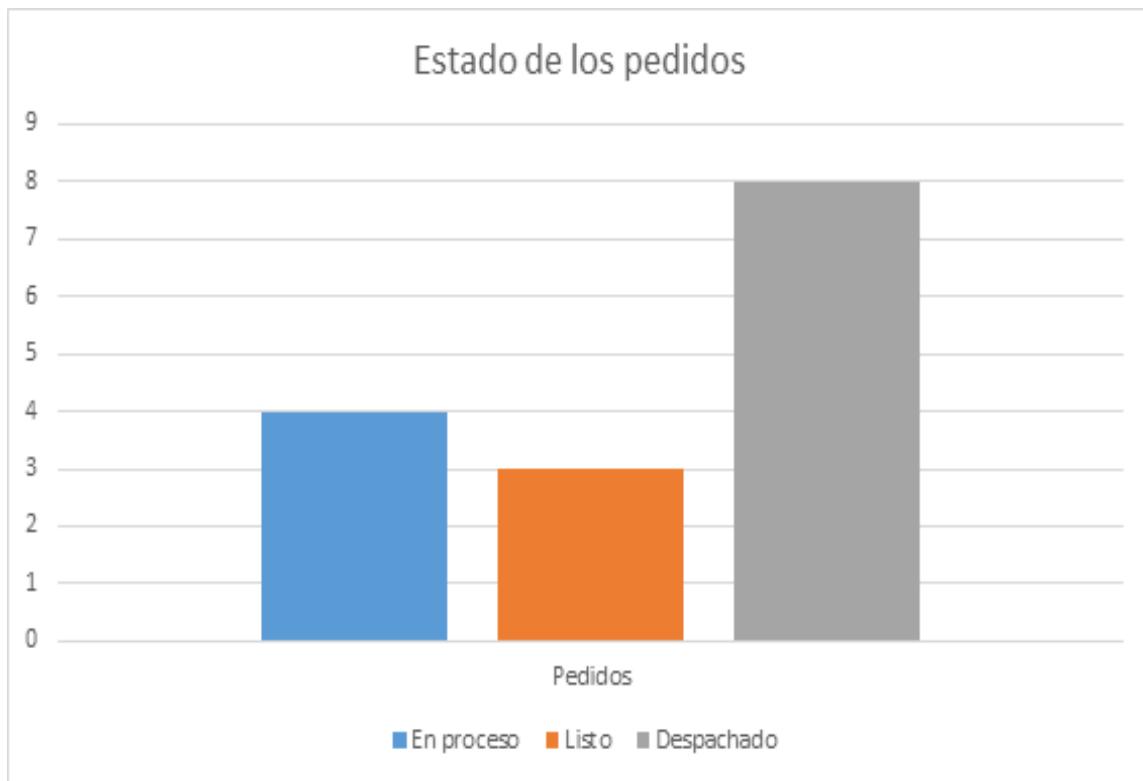
Se generaron algunos gráficos con la información recabada con la presente base de datos.

Algunos ejemplos :

El siguiente gráfico muestra las ventas realizadas por provincia.

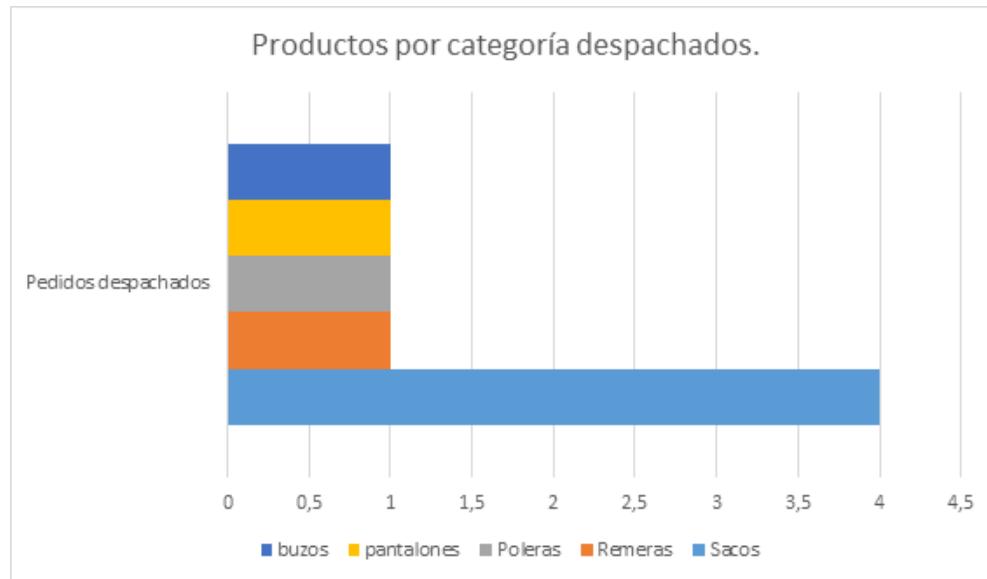


En el siguiente gráfico se muestran el estado de los pedidos.
Cuales se encuentran en proceso, cuales listos y cuales ya fueron despachados.





GENESElore



Herramientas y metodologías usadas

- **MySQL Workbench:** en este programa he creado los scripts y DER.
- **Microsoft Excel:** Lo utilicé para la descripción de las tablas.
- **Adobe Acrobat:** el mismo es utilizado para visualizar este informe.

Curso: SQL

Comisión: 50055

Alumna: Vivas, Lorena Susana