**Entrega Proyecto Final SQL CODERHORUSE**

**Introducción:**

Mi formación profesional al momento de iniciar el curso de SQL de CODERHOUSE me permitía desarrollar un sitio web completo, pero todavía me faltaba los conocimientos necesarios para poder desarrollar una interacción completa y eficiente con una base de datos SQL.

La operativa e-commerce es la parte que más me interesa del desarrollo web por tal razón, como proyecto final en el curso de SQL de CODERHOUSE, me propuse realizar una base de datos que se pudiera vincular a la operativa de un sitio web de e-commerce.

**Modelo de Negocio:**

El modelo de negocio al que apunto el desarrollo de mi proyecto final es la comercialización de productos vía web en la modalidad de e-commerce, que permita realizar la transacción completa a través de la web, con esto me refiero al contacto con el cliente, el registro de este, la operación comercial (venta del producto o servicio), envío del producto, realizar la gestión de cobro y la gestión de reposición de productos.

**Objetivo:**

Como objetivos del Proyecto Final realizado, me propuse dos:

1. Comprender y ejercitar todos los elementos necesarios para poder gestionar correctamente una base de datos SQL.
2. Poder desarrollar una base de datos inicial (elemental) que me permita incorporar los elementos que intervienen en la gestión de ventas en un sitio de e-commerce. Como todo proceso de aprendizaje arrancamos por algo simple y con practica la podré ir desarrollando hasta llegar a alcanzar una funcionalidad plena compatible con un sitio de e-commerce.

**Problemáticas para trabajar:**

En un sitio de e-commerce a mi entender encuentro las siguientes problemáticas en las cuales hacer foco para obtener un desarrollo optimo:

1. Gestión de cobranzas.
2. Gestión de envíos.
3. Gestión de manejo de stock y contacto con proveedores.

Problemáticas abordadas en el Proyecto Final de una manera elemental que iré profundizando en próximas actualización del proyecto, ya que estos trabajos evolucionan constantemente a media maduran los conocimientos adquiridos en el curso realizado.

**Ventas E-commerce**

**Esquema ER – Tablas DDL**

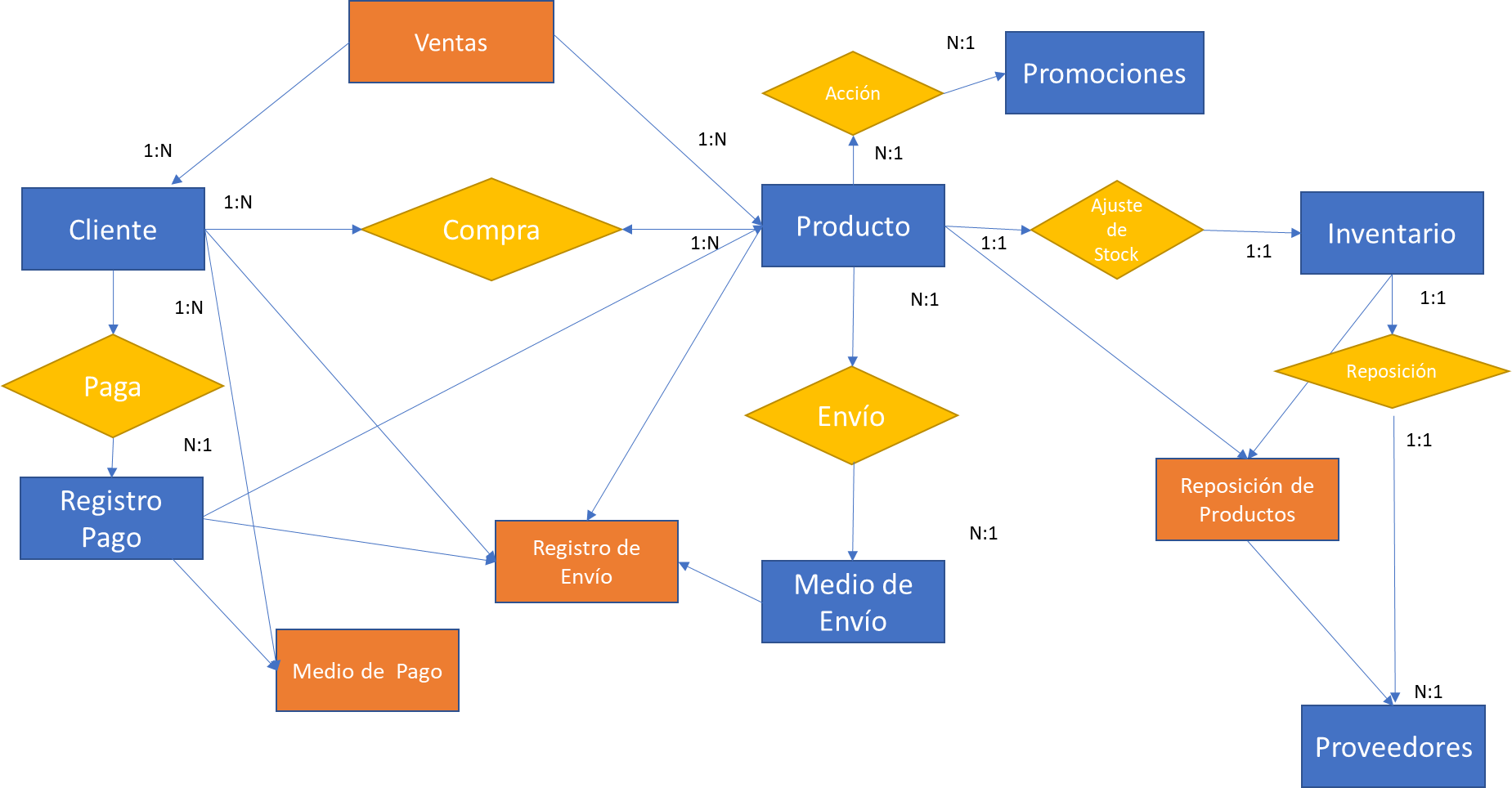
**Alcance del Proyecto:**

Desarrollamos una base de datos en SQL, para la administración de un sitio de comercio electrónico, que recibe pedidos de productos, procesa las ventas, los pagos, los envíos y la reposición de productos, ajustando el stock de estos en el inventario, solicitando su reposición a los distintos proveedores.

**Información que se adjunta:**

1. Esquema de Entidad Relación
2. Definición de Tablas
3. Tablas en lenguaje SQL DDL
4. Adjuntamos archivo para MySQL Workbench

**Esquema de Entidad Relación:**



**Definición de Tablas**





**Tablas en lenguaje SQL DDL**

CREATE TABLE CLIENTE (

CLIENTE\_ID INT NOT NULL AUTO\_INCREMENT,

NOMBRE VARCHAR (50) NOT NULL,

APELLIDO VARCHAR (50) NOT NULL,

DOCUMENTO\_NUMERO INT NOT NULL,

DOCUMENTO\_TIPO INT NOT NULL,

DIRECCION VARCHAR (50) NOT NULL,

CP VARCHAR (50) NOT NULL,

PROVINCIA VARCHAR (50) NOT NULL,

E\_MAIL VARCHAR (50) NOT NULL,

TELEFONO\_NUMERO INT NOT NULL,

PRIMARY KEY(CLIENTE\_ID)

);

CREATE TABLE PRODUCTO (

PRODUCTO\_ID INT NOT NULL AUTO\_INCREMENT,

PRECIO INT NOT NULL,

CLIENTE\_ID INT,

NOMBRE VARCHAR (50) NOT NULL,

CARACTERISTICAS VARCHAR (250) NOT NULL,

PRIMARY KEY(PRODUCTO\_ID),

FOREIGN KEY(CLIENTE\_ID)

REFERENCES CLIENTE(CLIENTE\_ID)

);

CREATE TABLE INVENTARIO (

INVENTARIO\_ID INT NOT NULL AUTO\_INCREMENT,

PRODUCTO\_ID INT NOT NULL,

CANTIDAD INT NOT NULL,

PRIMARY KEY(INVENTARIO\_ID),

FOREIGN KEY(PRODUCTO\_ID)

REFERENCES PRODUCTO(PRODUCTO\_ID)

);

CREATE TABLE PROMOCIONES (

PROMOCIONES\_ID INT NOT NULL AUTO\_INCREMENT,

DESCRIPTION VARCHAR (200),

PRODUCTO\_ID INT,

PRIMARY KEY(PROMOCIONES\_ID),

FOREIGN KEY(PRODUCTO\_ID)

REFERENCES PRODUCTO(PRODUCTO\_ID)

);

CREATE TABLE MEDIO\_PAGO (

MEDIO\_PAGO\_ID INT NOT NULL AUTO\_INCREMENT,

CLIENTE\_ID INT,

TARJETA\_CREDITO BINARY,

EFECTIVO BINARY,

TRANSFERENCIA\_BANCARIA BINARY,

PRIMARY KEY(MEDIO\_PAGO\_ID),

FOREIGN KEY(CLIENTE\_ID)

REFERENCES CLIENTE(CLIENTE\_ID)

);

CREATE TABLE REGISTRO\_PAGO (

REGISTRO\_PAGO\_ID INT NOT NULL AUTO\_INCREMENT,

FECHA\_PAGO DATE,

FECHA\_RECHAZO DATE,

MEDIO\_PAGO\_ID INT,

CLIENTE\_ID INT,

PRIMARY KEY(REGISTRO\_PAGO\_ID),

FOREIGN KEY(CLIENTE\_ID)

REFERENCES CLIENTE(CLIENTE\_ID),

FOREIGN KEY(MEDIO\_PAGO\_ID)

REFERENCES MEDIO\_PAGO(MEDIO\_PAGO\_ID)

);

CREATE TABLE MEDIO\_ENVIO (

MEDIO\_ENVIO\_ID INT NOT NULL AUTO\_INCREMENT,

PRODUCTO\_ID INT,

DESCRIPCION VARCHAR (200),

TRANSPROTE VARCHAR (50) NOT NULL,

FECHA\_ENVIO DATE NOT NULL,

PRIMARY KEY(MEDIO\_ENVIO\_ID),

FOREIGN KEY(PRODUCTO\_ID)

REFERENCES PRODUCTO(PRODUCTO\_ID)

);

CREATE TABLE REGISTRO\_ENVIO (

REGISTRO\_ENVIO\_ID INT NOT NULL AUTO\_INCREMENT,

REGISTRO\_PAGO\_ID INT,

CLIENTE\_ID INT,

PRODUCTO\_ID INT,

MEDIO\_ENVIO\_ID INT,

PRIMARY KEY(REGISTRO\_ENVIO\_ID),

FOREIGN KEY(PRODUCTO\_ID)

REFERENCES PRODUCTO(PRODUCTO\_ID),

FOREIGN KEY(CLIENTE\_ID)

REFERENCES CLIENTE(CLIENTE\_ID),

FOREIGN KEY(MEDIO\_ENVIO\_ID)

REFERENCES MEDIO\_ENVIO(MEDIO\_ENVIO\_ID)

);

CREATE TABLE PROVEEDORES (

PROVEEDORES\_ID INT NOT NULL AUTO\_INCREMENT,

RAZON\_SOCIAL VARCHAR (50) NOT NULL,

DIRECCION VARCHAR (50) NOT NULL,

TELEFONO INT NOT NULL,

E\_MAIL VARCHAR (50) NOT NULL,

PRODUCTO\_ID INT,

PRIMARY KEY(PROVEEDORES\_ID),

FOREIGN KEY(PRODUCTO\_ID)

REFERENCES PRODUCTO(PRODUCTO\_ID)

);

CREATE TABLE REPOSICION\_PRODUCTO (

REPOSICION\_PRODUCTO\_ID INT NOT NULL AUTO\_INCREMENT,

PRODUCTO\_ID INT,

PROVEEDORES\_ID INT,

INVENTARIO\_ID INT,

PRIMARY KEY(REPOSICION\_PRODUCTO\_ID),

FOREIGN KEY(PRODUCTO\_ID)

REFERENCES PRODUCTO(PRODUCTO\_ID),

FOREIGN KEY(PROVEEDORES\_ID)

REFERENCES PROVEEDORES(PROVEEDORES\_ID),

FOREIGN KEY(INVENTARIO\_ID)

REFERENCES INVENTARIO(INVENTARIO\_ID)

);

CREATE TABLE `ventas` (

`VENTAS\_ID` int NOT NULL AUTO\_INCREMENT,

`CLIENTE\_ID` int DEFAULT NULL,

`DESCRIPTION` varchar(200) DEFAULT NULL,

`CANTIDAD` int NOT NULL,

`PRODUCTO\_ID` int DEFAULT NULL,

`MEDIO\_PAGO\_ID` int DEFAULT NULL,

PRIMARY KEY (`VENTAS\_ID`),

KEY `CLIENTE\_ID` (`CLIENTE\_ID`),

KEY `PRODUCTO\_ID` (`PRODUCTO\_ID`),

KEY `MEDIO\_PAGO\_ID` (`MEDIO\_PAGO\_ID`),

CONSTRAINT `ventas\_ibfk\_1` FOREIGN KEY (`CLIENTE\_ID`) REFERENCES `cliente` (`CLIENTE\_ID`),

CONSTRAINT `ventas\_ibfk\_2` FOREIGN KEY (`PRODUCTO\_ID`) REFERENCES `producto` (`PRODUCTO\_ID`),

CONSTRAINT `ventas\_ibfk\_3` FOREIGN KEY (`MEDIO\_PAGO\_ID`) REFERENCES `medio\_pago` (`MEDIO\_PAGO\_ID`)

);

**Base de Datos: ventas\_ecommerce**

**DESARROLLO DE VISTAS – FUNCIONES – STORED PROCEDURES - TRIGGERS**

**Listado de Vistas:**

1. inversión\_total\_productos
2. producto\_precio>$10000
3. precio\_promedio\_ventas
4. productoxproveedor
5. promocionesxproducto
6. ventasxcliente

**Detalle de cada Vista: tablas intervinientes y objetivos:**

**1.inversion\_total\_productos**

**Script:**

CREATE

ALGORITHM = UNDEFINED

DEFINER = `root`@`localhost`

SQL SECURITY DEFINER

VIEW `inversion\_total\_productos` AS

SELECT

`producto`.`PRODUCTO\_ID` AS `PRODUCTO\_ID`,

`producto`.`NOMBRE` AS `NOMBRE`,

(`producto`.`PRECIO` \* `producto`.`CANTIDAD`) AS `INVERSION\_TOTAL`

FROM

`producto`

**Tablas intervinientes:**

Tabla interviniente PRODUCTO.

Columnas de la tabla PRODUCTO en la VISTA:

1. CANTIDAD
2. PRECIO

**Objetivos:**

Esta vista nos muestra la inversión total que tiene la empresa a recuperar en forma total, sin descontar los costos ni calcular las ganancias.

**2. producto\_precio>$10000**

**Script:**

CREATE

ALGORITHM = UNDEFINED

DEFINER = `root`@`localhost`

SQL SECURITY DEFINER

VIEW `producto\_precio>$10000` AS

SELECT

`producto`.`PRODUCTO\_ID` AS `PRODUCTO\_ID`,

`producto`.`PRODUCTO\_PAGO` AS `PRODUCTO\_PAGO`,

`producto`.`PRECIO` AS `PRECIO`,

`producto`.`NOMBRE` AS `NOMBRE`,

`producto`.`CARACTERISTICAS` AS `CARACTERISTICAS`,

`producto`.`CANTIDAD` AS `CANTIDAD`

FROM

`producto`

WHERE

(`producto`.`PRECIO` > 10000)

**Tablas intervinientes:**

Tabla interviniente PRODUCTO.

Columnas de la tabla PRODUCTO en la VISTA:

1. PRODUCTO\_ID
2. PRODUCTO\_PAGO
3. NOMBRE
4. CARACTERISTICAS
5. CANTIDAD
6. PRECIO

**Objetivos:**

Esta vista nos muestra los productos cuyo precio sea superior a los $ 10.000. Nos permite seleccionar los productos de un rango de valor predeterminado.

Nos muestra, toda la información referida a los productos del rango de valor seleccionado en este caso los de valor superior a los $ 10.000.

**3. precio\_promedio\_ventas**

**Script:**

CREATE

ALGORITHM = UNDEFINED

DEFINER = `root`@`localhost`

SQL SECURITY DEFINER

VIEW `precio\_promedio\_ventas` AS

SELECT

AVG(`producto`.`PRECIO`) AS `PRECIO\_PROMEDIO`

FROM

`producto`

**Tablas intervinientes:**

Tabla interviniente PRODUCTO.

Columnas de la tabla PRODUCTO en la VISTA:

PRECIO

**Objetivos:**

Utilizando una función predefinida de SQL AVG( ), nos permite calcular el precio promedio de los productos comercializados.

**4. productoxproveedor**

**Script:**

CREATE

ALGORITHM = UNDEFINED

DEFINER = `root`@`localhost`

SQL SECURITY DEFINER

VIEW `productoxproveedor` AS

SELECT

`producto`.`NOMBRE` AS `PRODUCTO`,

`proveedores`.`RAZON\_SOCIAL` AS `PROVEEDOR`

FROM

(`producto`

JOIN `proveedores` ON ((`producto`.`PRODUCTO\_ID` = `proveedores`.`PROVEEDORES\_ID`)))

**Tablas intervinientes:**

Tablas intervinientes PRODUCTO y PROVEEDORES.

Columnas de la tabla PRODUCTO en la VISTA:

NOMBRE

Columnas de la tabla PROVEEDORES en la VISTA:

RAZON SOCIAL

**Objetivos:**

Esta vista nos permite vincular el nombre del producto con el nombre de la Razón Social del proveedor de este.

**5. promocionesxproducto**

**Script:**

CREATE

ALGORITHM = UNDEFINED

DEFINER = `root`@`localhost`

SQL SECURITY DEFINER

VIEW `promocionesxproducto` AS

SELECT

`promociones`.`DESCRIPTION` AS `PROMOCION`,

`producto`.`NOMBRE` AS `PRODUCTO`,

`producto`.`CANTIDAD` AS `CANTIDAD`,

`promociones`.`DESCUENTO` AS `DESCUENTO`

FROM

(`promociones`

LEFT JOIN `producto` ON ((`promociones`.`PROMOCIONES\_ID` = `producto`.`PRODUCTO\_ID`)))

**Tablas intervinientes:**

Tablas intervinientes PRODUCTO y PROMOCIONES.

Columnas de la tabla PRODUCTO en la VISTA:

NOMBRE

CANTIDAD

Columnas de la tabla PROMOCIONES en la VISTA:

DESCRIPTION

DESCUENTOS

**Objetivos:**

Muestra las promociones que aplican sobre cada producto en particular.

**6. ventasxcliente**

**Script:**

CREATE

ALGORITHM = UNDEFINED

DEFINER = `root`@`localhost`

SQL SECURITY DEFINER

VIEW `ventasxcliente` AS

SELECT

`ventas`.`CLIENTE\_ID` AS `CLIENTE`,

COUNT(`ventas`.`VENTAS\_ID`) AS `cantidadFacturas`

FROM

`ventas`

GROUP BY `CLIENTE`

**Tablas intervinientes:**

Tablas intervinientes CLIENTE y VENTAS.

Columnas de la tabla CLIENTE en la VISTA:

CLIENTE\_ID

Columnas de la tabla VENTAS en la VISTA:

VENTAS\_ID

**Objetivos:**

Nos permite conocer la cantidad de ventas por cliente.

**Listado de Funciones:**

1. precio\_con\_iva
2. precio\_promedio
3. productos\_stock\_porID

**Detalle de cada Función: tablas intervinientes y objetivos:**

1. **precio\_con\_iva**

**Script:**

CREATE DEFINER=`root`@`localhost` FUNCTION `precio\_con\_iva`(productoID INT) RETURNS varchar(40) CHARSET utf8mb4

READS SQL DATA

BEGIN

DECLARE iva DECIMAL(10.2);

DECLARE resultado VARCHAR(40);

SET iva = 1.21;

SET resultado = (SELECT precio\*iva FROM venta\_ecommerce.producto WHERE producto\_id = productoID);

RETURN resultado;

END

**Tablas intervinientes:**

Tablas intervinientes PRODUCTO.

Columnas de la tabla PRODUCTO en la VISTA:

PRECIO

**Objetivos:**

Nos permite calcular el precio de ventas más el IVA de cualquier producto de la tabla productos.

1. **precio\_promedio**

**Script:**

CREATE DEFINER=`root`@`localhost` FUNCTION `precio\_promedio`() RETURNS varchar(50) CHARSET utf8mb4

READS SQL DATA

BEGIN

DECLARE resultado VARCHAR(50);

SET resultado = (SELECT avg(precio) FROM producto);

RETURN resultado;

END

**Tablas intervinientes:**

Tablas intervinientes PRODUCTO.

Columnas de la tabla PRODUCTO en la VISTA:

PRECIO

**Objetivos:**

Nos permite calcular el precio promedio de los productos de la tabla productos.

1. **productos\_stock\_porID**

**Script:**

CREATE DEFINER=`root`@`localhost` FUNCTION `productos\_stock\_porID`(productoID INT) RETURNS varchar(30) CHARSET utf8mb4

DETERMINISTIC

BEGIN

DECLARE resultado INT;

SET resultado = (SELECT cantidad FROM producto WHERE producto\_id = productoID);

RETURN resultado;

END

**Tablas intervinientes:**

Tablas intervinientes PRODUCTO.

Columnas de la tabla PRODUCTO en la VISTA:

PRODUCTO\_ID

CANTIDAD

**Objetivos:**

Nos permite conocer el stock de productos, seleccionando un producto por su ID.

**Listado de Sotred Procedures:**

1. sp\_consulta\_productos
2. sp\_ultimo\_cliente
3. sp\_crear\_cliente

**Detalle de cada Stored Procedures: tablas intervinientes y objetivos:**

1. **sp\_consulta\_productos**

**Script:**

CREATE DEFINER=`root`@`localhost` PROCEDURE `sp\_consulta\_productos`(IN campo char(20))

BEGIN

IF campo <> '' THEN

SET @ordenar = CONCAT('ORDER BY ', campo);

ELSE

SET @ordenar = '';

END IF;

SET @clausula = CONCAT('SELECT \* FROM venta\_ecommerce.producto ', @ordenar, ';');

PREPARE ejecutarSQL from @clausula;

EXECUTE ejecutarSQL;

DEALLOCATE PREPARE ejecutarSQL;

END

**Tablas intervinientes:**

Tablas intervinientes PRODUCTO.

Columnas de la tabla PRODUCTO en la VISTA:

Todas las columnas de la tabla PRODUCTO ya que puede elegir cualquiera.

**Objetivos:**

Nos permite seleccionar cualquier columna de la tabla PRODCTO y ordenar dicha tabla a partir de dicha columna de manera decreciente.

**2. sp\_ultimo\_cliente**

**Script:**

CREATE DEFINER=`root`@`localhost` PROCEDURE `sp\_ulitmo\_cliente`(OUT cliente Int)

BEGIN

SELECT MAX(CLIENTE\_ID) INTO CLIENTE

FROM CLIENTE;

END

**Tablas intervinientes:**

Tablas intervinientes CLIENTE.

Columnas de la tabla PRODUCTO en la VISTA:

CLIENTE\_ID

**Objetivos:**

Nos permite obtener el valor del último ID utilizado para dar de alta a un cliente.

1. **sp\_crear\_cliente**

**Script:**

CREATE DEFINER=`root`@`localhost` PROCEDURE `sp\_crear\_cliente`(IN nombre VARCHAR(50), IN apellido VARCHAR(10), IN documento\_tipo VARCHAR(10), IN documento\_numero INT, IN direccion VARCHAR(50), IN cp VARCHAR(50), IN provincia VARCHAR(50), IN e\_mail VARCHAR(50), IN telefono\_numero INT)

BEGIN

DECLARE nuevo\_cliente INT;

CALL sp\_ulitmo\_cliente(@nuevo\_cliente);

SET @nuevo\_cliente = @nuevo\_cliente + 1;

INSERT INTO CLIENTE VALUES(@nuevo\_cliente, nombre, apellido, documento\_tipo, documento\_numero, direccion, cp, provincia, e\_mail, telefono\_numero);

END

**Tablas intervinientes:**

Tablas intervinientes CLIENTE.

Columnas de la tabla PRODUCTO en la VISTA:

CLIENTE\_ID

NOMBRE

APELLIDO

DOCUMENTO\_TIPO

DOCUMETO\_NUMERO

DIRECCION

CP

PROVINCIA

TELEFONO

E-MAIL

**Objetivos:**

Nos permite crear un nuevo cliente, incorporando todos los datos de la TABLA CLIENTES, y mantener el orden de los CLIENTES\_ID, ya que con el SP sp\_ultimo\_cliente, podemos obtener el último valor del CLIENTE\_ID antes de la creación de este.

**TRIGGERS**

Triggers vinculados a la Tabla Productos:

1. ins\_producto
2. actualización\_precio

Triggers vinculados a la tabla Proveedores:

1. Ins\_alta\_proveedor
2. Ins\_delete\_proveedor

**Paso a paso en la Inserción de Datos para las tablas de venta\_ecommerce:**

1. Generamos un archivo que respete el formato de la Tabla sobre la que vamos a realizar la inserción de datos, esto quiere decir que tenemos que respetar la cantidad de columnas y los tipos de datos que van dentro de cada columna de acuerdo con lo definido originalmente al momento de la creación de la tabla en MySQL Workbench.
2. Una vez que tenemos el archivo en Excel generado siguiendo los preceptos del punto 1, realizamos una transformación de Excel a CVS utilizando la opción de Exportar que brinda Excel, transformado el archivo lo guardamos y nos preparamos para realizar la exportación correspondiente.
3. En MySQL Workbench seleccionamos la tabla de destino de la importación de datos (en donde realizaremos la inserción) Export/Import, y procedemos a insertar los datos correspondientes.

**Listado de Herramientas utilizadas:**

1. MySQL Workbench.: Herramienta utilizada para la creación de la Base de Datos, la inserción de los datos, manejo de datos y creación de funciones, vistas y stored porcedures.
2. Excel: Herramienta utilizada para la definición de las tablas intervinientes en la base de datos e importación de datos.
3. Word: Herramienta utilizada para la elaboración del informe del proyecto.
4. Google Chrome: Navegador web utilizado para la consulta en la web.