



# Proyecto base de datos MySQL

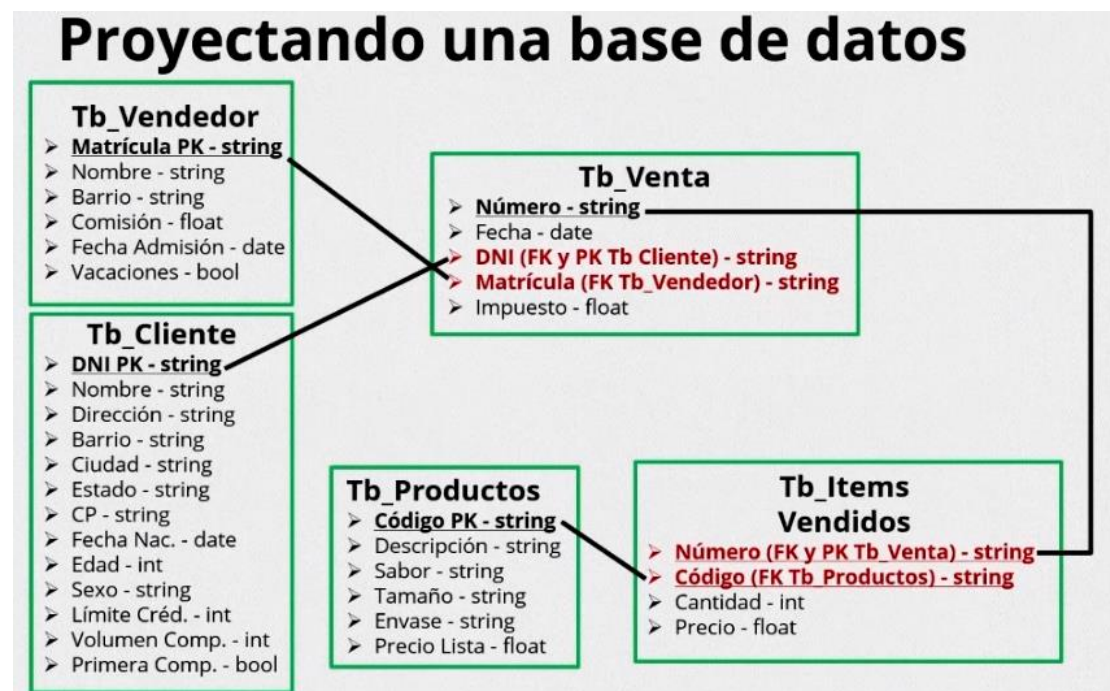
PROYECTO FINAL

Torres Leandro Joel |Proyecto Final| Febrero 2022|Alura

## Introducción

Esta documentación es elaborada para concluir un curso realizado en la web Alura.com en bases de datos con MySQL. El mismo es de carácter personal creando así una documentación propia para cualquier interesado en la temática, y realizando un proyecto propio en código y algunas aclaraciones matemáticas.

El proyecto se basa en crear una base de datos ,tabla de la figura, realizar las relaciones, insertar datos y consultas. Para ello utilizaremos el Workbench de MySQL. La idea es presentar un proyecto básico de la metodología de trabajo de base de datos relacionales en MySQL.



## Creación de la base de datos , tablas y atributos

Debemos que tener cuidado con las primary key, porque no deben estar vacías a la hora de declararlas

Creando la tabla cliente con DNI como llave primaria, por lo que no acepta duplicados

```
CREATE TABLE tb_clientes (  
  DNI VARCHAR(100) NOT NULL,  
  NOMBRE VARCHAR(100) NULL,  
  DIRECCION VARCHAR(150) ,  
  BARRIO VARCHAR(50) ,  
  CIUDAD VARCHAR(50) ,  
  ESTADO VARCHAR(10) ,  
  CP VARCHAR(10) ,  
  FECHA_NACIMIENTO DATE ,  
  EDAD SMALLINT ,  
  SEXO VARCHAR(1) ,
```

```
LIMITE_CREDITO FLOAT,
VOLUMEN_COMPRA FLOAT,
PRIMERA_COMPRA BIT,
PRIMARY KEY (DNI) );
```

Tabla vendedor con llave primaria matricula

```
CREATE TABLE tb_vendedor (
MATRICULA VARCHAR(50) NOT NULL,
NOMBRE VARCHAR(100) ,
BARRIO VARCHAR(100) ,
COMISION FLOAT,
FECHA_ADMISION DATE,
VACACIONES BIT(1) ,
PRIMARY KEY (MATRICULA) );
```

Tabla productos con llave primaria codigo

```
CREATE TABLE tb_productos (
CODIGO VARCHAR(10) NOT NULL,
DESCRIPCION VARCHAR(100) ,
SABOR VARCHAR(50) ,
TAMANO VARCHAR(50) ,
ENVASE VARCHAR(50) ,
PRECIO FLOAT,
PRIMARY KEY (CODIGO) );
```

Tabla ventas, que la renombramos como facturas con llave primaria número y llaves foráneas DNI y MATRICULA para vincular con la tabla de clientes y vendedor

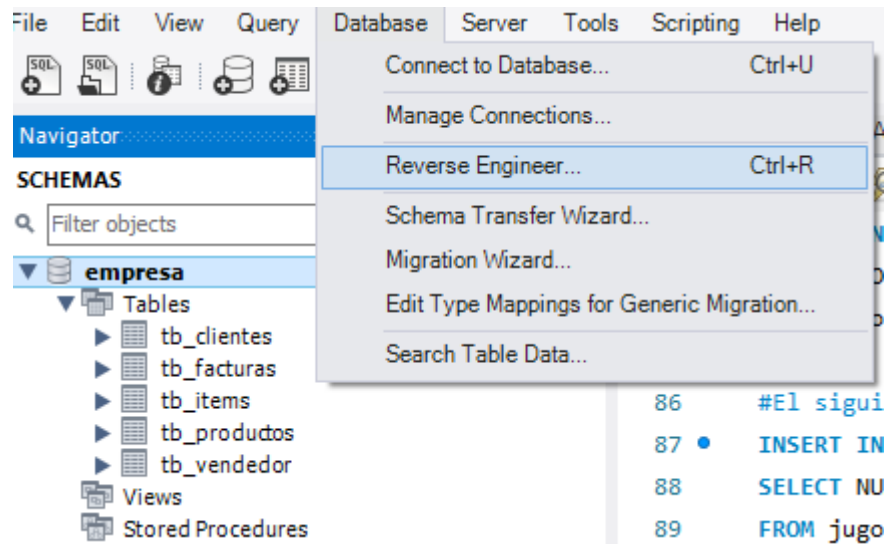
```
CREATE TABLE tb_facturas (
NUMERO INT NOT NULL,
FECHA DATE,
DNI VARCHAR(11) NOT NULL,
MATRICULA VARCHAR(5) NOT NULL,
IMPUESTO FLOAT,
PRIMARY KEY (NUMERO) ,
FOREIGN KEY (DNI) REFERENCES tb_clientes (DNI) ,
FOREIGN KEY (MATRICULA) REFERENCES tb_vendedor (MATRICULA) );
```

Tabla ítems con dos llaves primarias “NUMERO” y “CODIGO”, que también son llaves foráneas.

```
CREATE TABLE tb_items (
NUMERO INT NOT NULL,
CODIGO VARCHAR(10) NOT NULL,
CANTIDAD INT,
PRECIO FLOAT,
PRIMARY KEY (NUMERO, CODIGO) ,
```

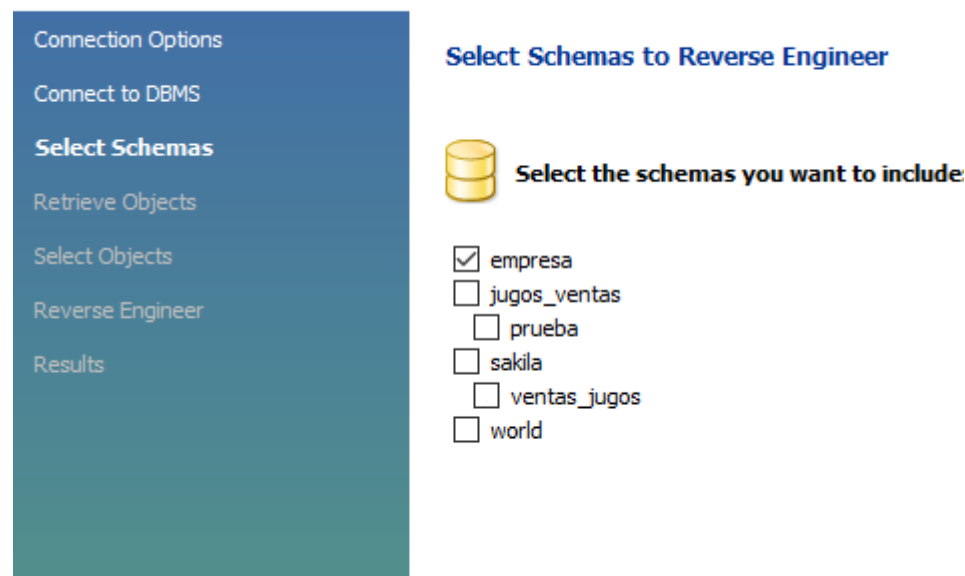
```
FOREIGN KEY (NUMERO) REFERENCES tb_facturas (NUMERO) ,
FOREIGN KEY (CODIGO) REFERENCES tb_productos (CODIGO) );
```

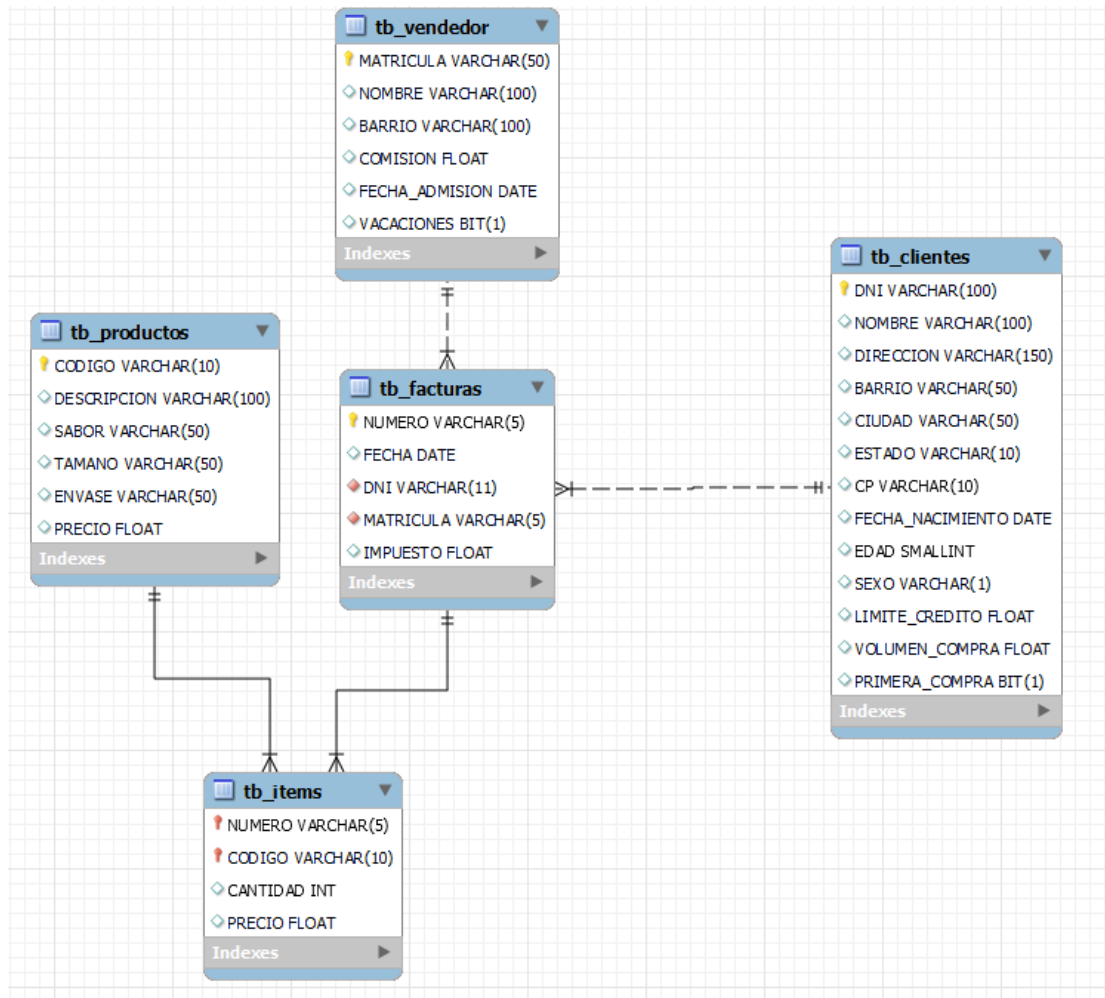
Por último podemos ver nuestro modelo relacional ,debemos ir a Database y luego a Reverse Engineer



Le damos a todo next, dejando la configuración determinada, solo queda seleccionar nuestra base de datos en las opciones

#### Reverse Engineer Database





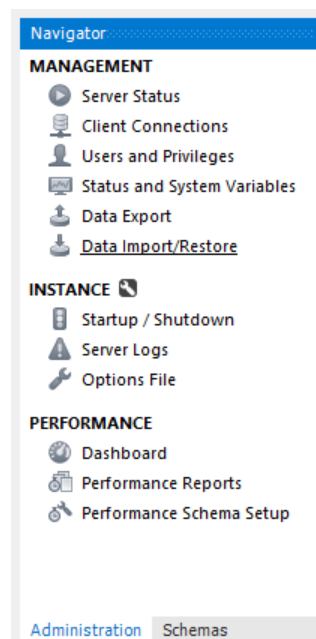
## Dataset

Una vez creado las tablas cargaremos nuestro dataset para importar los registros a nuestras tablas, el dataset se encuentra en la carpeta “Dataset\_DumpJugosVentas”

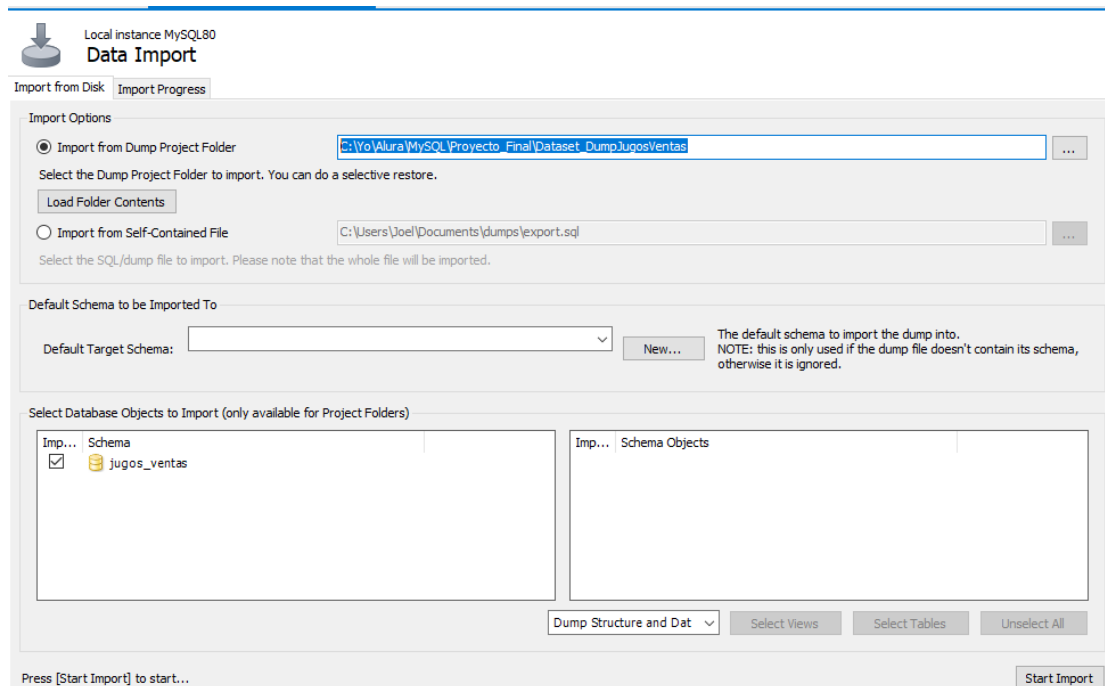
Para ello debemos crear otro schema con el mismo nombre del dataset que queremos importar, en nuestro caso se llama “jugos\_ventas”.

```
CREATE DATABASE jugos_ventas;  
USE jugos_ventas;
```

Con ayuda del workbench de MySQL vamos a administración, luego a “Data import/Restore”



Por último buscamos la ruta de nuestro dataset y damos a “Start Import”



Realizado esto, ya tendremos nuestro dataset importado

## Importando registros

```
#Importamos registros
USE empresa;
```

```
INSERT INTO tb_clientes
SELECT DNI, NOMBRE, DIRECCION_1 AS DIRECCION,
BARRIO, CIUDAD, ESTADO, CP, FECHA_DE_NACIMIENTO AS FECHA_NACIMIENTO,
EDAD, SEXO, LIMITE_DE_CREDITO AS LIMITE_CREDITO, VOLUMEN_DE_COMPRA AS
VOLUMEN_COMPRA, PRIMERA_COMPRA
FROM jugos_ventas.tabla_de_clientes;
```

```
INSERT INTO tb_vendedor
SELECT MATRICULA, NOMBRE, BARRIO, PORCENTAJE_COMISION AS
COMISION, FECHA_ADMISION, VACACIONES
FROM jugos_ventas.tabla_de_vendedores;
```

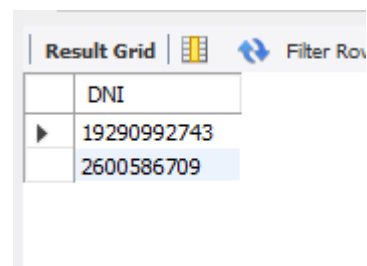
```
INSERT INTO tb_productos
SELECT CODIGO_DEL_PRODUCTO AS CODIGO, NOMBRE_DEL_PRODUCTO AS
DESCRIPCION, SABOR, TAMANO, ENVASE, PRECIO_DE_LISTA AS PRECIO
FROM jugos_ventas.tabla_de_productos;
```

Notamos que justo cuando realizamos la siguiente consulta

```
INSERT INTO tb_facturas
SELECT NUMERO, FECHA_VENTA AS FECHA, DNI, MATRICULA, IMPUESTO
FROM jugos_ventas.facturas;
```

Nos tira un error, esto se debe a que la tabla facturas tiene llave foráneas , una de ellas “DNI” posee datos que no se encuentran en la tabla “clientes” de la cual se relaciona que es primaria, esto lo podemos notar realizando la siguiente consulta

```
SELECT DISTINCT DNI
FROM jugos_ventas.facturas
WHERE DNI NOT IN (SELECT DNI from tb_clientes);
```



The screenshot shows a 'Result Grid' window with a table containing DNI values. The table has two rows: the first row contains '19290992743' and the second row contains '2600586709'. The second row is highlighted in blue.

DNI
19290992743
2600586709

Y los datos DNI que nos faltan son '19290992743' y '2600586709'

Nos muestra los registros no repetidos (DISTINC) que se encuentran en el dataset “jugos\_ventas” en la tabla “facturas” pero no se encuentran en nuestra tabla “clientes”, por lo antes dicho, esto no debe pasar, entonces para solucionarlo vamos a agregar los DNI faltantes en la tabla clientes , con otros valores aleatorios.

```
INSERT INTO tb_clientes (DNI, NOMBRE, DIRECCION, BARRIO, CIUDAD,
ESTADO, CP, FECHA_NACIMIENTO, EDAD, SEXO, LIMITE_CREDITO,
VOLUMEN_COMPRA, PRIMERA_COMPRA)
VALUES ('19290992743', 'Rodrigo Villa', 'Libertadores 65', 'Héroes',
'Ciudad de México', 'EM', '21002020', '1998-05-30', 22, 'M', 120000,
220000, 0),
('2600586709', 'Raúl Meneses', 'Estudiantes 89', 'Centro', 'Ciudad
de México', 'EM', '22002012', '1999-08-13', 21, 'M', 120000, 210000,
1);
```

Continuamos importando los datos del dataset

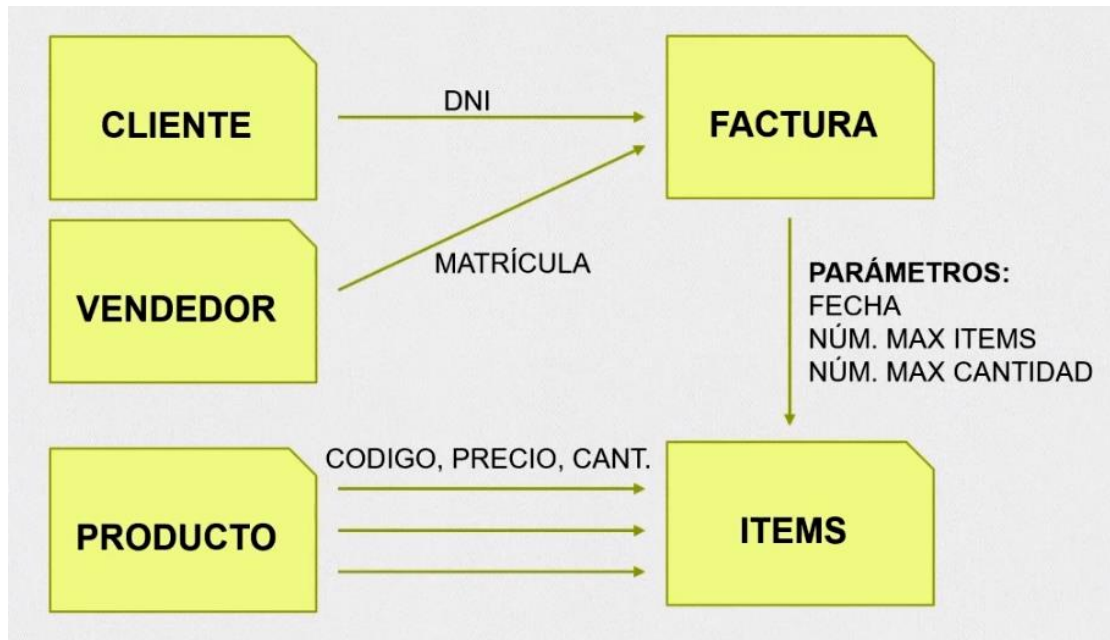
```
INSERT INTO tb_facturas
SELECT NUMERO, FECHA_VENTA AS FECHA, DNI, MATRICULA, IMPUESTO
FROM jugos_ventas.facturas;
```

```
INSERT INTO tb_items
SELECT NUMERO, CODIGO_DEL_PRODUCTO AS CODIGO, CANTIDAD, PRECIO
FROM jugos_ventas.items_facturas;
```

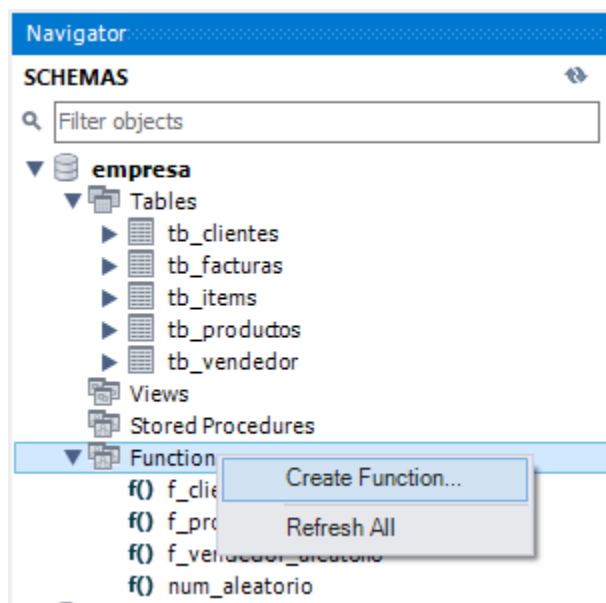


## Generando ventas ficticias

En este apartado vamos a generar ventas (facturas) las cuales tomarán clientes y vendedores de manera aleatoria. Luego los parámetros FECHA, NUM. MAX ITEMS y NUM MAX CANTIDAD se utilizarán en la tabla ITEMS y también



Primero vamos a crear una función para obtener un número aleatorio para los clientes y vendedores, para ello:



En el caso de que sucediera un error lo mas probable es que no tengamos habilitado el permiso para crear funciones, por lo que solo tendríamos que ejecutar la siguiente consulta y se soluciona el problema

```
SET GLOBAL log_bin_trust_function_creators = 1;
```

Antes de presentar el código ,una breve explicación de la lógica matemática del algoritmo

*Usaremos la función RAND que nos devuelve un valor "X" entre 0 y 1*

$$0 \leq RAND \leq 1$$

*Podemos modificar el rango de esto realizando la búsqueda de la función*

*$X = X(RAND)$ , o sea una función que contenga a RAND*

*que cumpla que:*

$$m \leq X \leq M \quad \text{con } M: \text{valor máximo} \quad m: \text{valor mínimo}$$

*entonces*

$$0 \leq X - m \leq M - m$$

*operando*

$$0 \leq \frac{X - m}{M - m} \leq 1$$

*Entonces como RAND también se encuentra entre 0 y 1*

$$RAND = \frac{X - m}{M - m} \rightarrow RAND * (M - m) = X - m \rightarrow RAND * (M - m) + m = X$$

*Por lo tanto usaremos la funcion*

$$X = RAND * (M - m) + m$$

Entonces el código de la función “num\_aleatorio” es la siguiente

```
CREATE DEFINER='root'@'localhost' FUNCTION `num_aleatorio`(vmax
int,vmin int) RETURNS int
BEGIN
DECLARE vresultado INT;
SELECT FLOOR((RAND() * (vmax-vmin))+vmin) INTO vresultado;
RETURN vresultado;
END
```

Notemos que utilizamos la funcion FLOOR(X), está función nos permite truncar en numeros enteros para no tener decimales.

La función retorna un valor aleatorio entre el valor maximo “vmax” y el mínimo “vmin” que queramos, ya que queremos tomar distintos registros y no podemos superarnos del número de cantidad máxima de registros que disponemos en la tabla. Por ejemplo si tenemos 300 registros en una tabla, no podemos tomar el registro 301.

Por lo antes aclarado tomaremos “vmax” usando la función COUNT(\*), la cual nos devuelve la cantidad de registros que tiene una tabla. Por consiguiente realizaremos 3 funciones más para obtener claves primarias de clientes, vendedores y productos aleatoriamente.

```
CREATE DEFINER=`root`@`localhost` FUNCTION `f_cliente_aleatorio`()
RETURNS varchar(11) CHARSET utf8mb4
BEGIN
#Hay que tener en cuenta que el tipo de variable y cantidad de
caracteres
#de la tabla a la que se hace referencia, en este caso vresultado
toma un DNI de tb_clientes
DECLARE vresultado VARCHAR(11);
DECLARE valeatorio INT;
DECLARE vmax INT;
SELECT COUNT(*) INTO vmax FROM tb_clientes;
#como la funcion num_aleatorio crea un valor entre 1 y el vmax, si le
resto -1
# entonces ahora varia entre 0 y vmax-1, para con el limit poder
tomar
SET valeatorio=num_aleatorio(0,vmax);
SELECT DNI INTO vresultado FROM tb_clientes LIMIT valeatorio,1;
RETURN vresultado;
END
```

```
CREATE DEFINER=`root`@`localhost` FUNCTION `f_vendedor_aleatorio`()
RETURNS varchar(50) CHARSET utf8mb4
BEGIN
#Hay que tener en cuenta que el tipo de variable y cantidad de
caracteres
#de la tabla a la que se hace referencia, en este caso vresultado
toma un DNI de tb_vendedor
DECLARE vresultado VARCHAR(50);
DECLARE valeatorio INT;
DECLARE vmax INT;
SELECT COUNT(*) INTO vmax FROM tb_vendedor;
#como la funcion num_aleatorio crea un valor entre 1 y el vmax, si le
resto -1
# entonces ahora varia entre 0 y vmax-1, para con el limit poder
tomar
SET valeatorio=num_aleatorio(0,vmax);
SELECT MATRICULA INTO vresultado FROM tb_vendedor LIMIT valeatorio,1;
RETURN vresultado;
END
```

```

CREATE DEFINER='root'@'localhost' FUNCTION `f_producto_aleatorio`()
RETURNS varchar(10) CHARSET utf8mb4
BEGIN
#Hay que tener en cuenta que el tipo de variable y cantidad de
caracteres
#de la tabla a la que se hace referencia, en este caso vresultado
toma un DNI de tb_productos
DECLARE vresultado VARCHAR(10);
DECLARE valeatorio INT;
DECLARE vmax INT;
SELECT COUNT(*) INTO vmax FROM tb_productos;
#como la funcion num_aleatorio crea un valor entre 1 y el vmax, si le
resto -1
# entonces ahora varia entre 0 y vmax-1, para con el limit poder
tomar
SET valeatorio=num_aleatorio(0,vmax);
SELECT CODIGO INTO vresultado FROM tb_productos LIMIT valeatorio,1;
RETURN vresultado;
END

```

Por último agregaremos un “Stored Procuders” para simular una venta ficticia de la fecha que escribamos y de la cantidad de productos que queremos comprar

```

CREATE DEFINER='root'@'localhost' PROCEDURE `venta_ficticia`(fecha
DATE,maxitem INT)
BEGIN
DECLARE vcliente VARCHAR(11);
DECLARE vvendedor VARCHAR(5);
DECLARE numfactura INT;
DECLARE codproducto VARCHAR(10);
DECLARE prec FLOAT;
DECLARE contador INT DEFAULT 1;

SELECT MAX(NUMERO) + 1 INTO numfactura FROM tb_facturas;
SET vcliente = f_cliente_aleatorio();
SET vvendedor = f_vendedor_aleatorio();
INSERT INTO tb_facturas (NUMERO, FECHA, DNI, MATRICULA, IMPUESTO)
VALUES (numfactura, fecha, vcliente, vvendedor, 0.16);
#CALL for_items(numfactura,maxitem);

WHILE contador <= maxitem DO
SET codproducto=f_producto_aleatorio();
SELECT PRECIO INTO prec FROM tb_productos WHERE CODIGO=codproducto;
INSERT INTO tb_items (NUMERO,CODIGO,CANTIDAD,PRECIO) VALUES
(numfactura,codproducto,1,prec);
SET contador=contador+1;
END WHILE;

END

```

## Conclusiones

Finalmente podemos concluir que a la hora de armar una base de datos debemos tener mucho cuidado con la dependencia entre registros de tablas debido a si son primarias en una tabla y foráneas en otra. También la utilización de funciones y Stored Procedures para creación de cálculos y consultas de invocación rápida

Este proyecto solo es una pequeña introducción del lenguaje, por lo que se podría complejizar más combinando muchas variables, por ejemplo elegir cantidad específicas de productos y agregar el precio total en los ítems, o actualizar precios de los productos modificando el precio en ítems, entre muchas cosas.