

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO  
FACULTAD DE CIENCIAS, 2025-II  
FUNDAMENTOS DE BASES DE DATOS



---

PROYECTO FINAL:  
*Bazar “La Gatita Emprendedora”*

---

NOMBRE DEL EQUIPO:  
NAMEisNULL

INTEGRANTES:  
Flores Mata Ricardo - 422127808  
Martínez Leal José María - 317243970  
Matute Cantón Sara Lorena - 319331622  
Sánchez Cruz Norma Selene - 320198508  
Suárez Ortiz Joshua Daniel - 320151260  
Villegas Martínez Vania Victoria - 418003114

# Índice

<b>1 Modelo Entidad - Relación</b>	<b>4</b>
1.1 ENTIDADES . . . . .	4
1.1.1 <b>Bazar</b> . . . . .	4
1.1.2 <b>Negocio</b> . . . . .	4
1.1.3 <b>Estand:</b> . . . . .	5
1.1.4 <b>Emprendedor:</b> . . . . .	6
1.1.5 <b>PersonalOrganizador</b> . . . . .	7
1.1.6 <b>Ticket</b> . . . . .	7
1.1.7 <b>Producto</b> . . . . .	8
1.1.8 <b>Servicio</b> . . . . .	9
1.1.9 <b>Cliente</b> . . . . .	9
1.1.10 <b>Tarjeta</b> . . . . .	10
1.2 RELACIONES . . . . .	11
1.2.1 <b>Agendar</b> . . . . .	11
1.2.2 <b>Escoger</b> . . . . .	11
1.2.3 <b>Cobrar</b> . . . . .	12
1.2.4 <b>Comprar</b> . . . . .	12
1.2.5 <b>Domiciliar</b> . . . . .	13
1.2.6 <b>Emprender</b> . . . . .	13
1.2.7 <b>Pagar</b> . . . . .	14
1.2.8 <b>Vender</b> . . . . .	15
1.2.9 <b>Ofrecer</b> . . . . .	16
1.2.10 <b>Registrar</b> . . . . .	16
1.2.11 <b>Registrar</b> . . . . .	17
1.2.12 <b>Trabajar</b> . . . . .	17
<b>2 Modelo Relacional</b>	<b>19</b>
2.1 Traducción del Modelo E/R al Modelo Relacional . . . . .	19
2.1.1 Entidades fuertes . . . . .	19
2.1.2 Entidades débiles . . . . .	24
2.1.3 Relaciones que generan tabla . . . . .	25
2.1.4 Relaciones que modifican tablas existentes . . . . .	26
2.2 Modelo Relacional (Tablas) . . . . .	28
<b>3 Normalización</b>	<b>33</b>
3.1 Dependencias funcionales, redundancia, anomalías y normalización . . . . .	34
3.2 Modelo Relacional Normalizado . . . . .	42

---

<b>4 Lenguaje para Definición de Datos</b>	<b>47</b>
4.1 Políticas de mantenimiento para las llaves foráneas	47
4.2 Creación	49
<b>5 Disparadores Funciones y SPs</b>	<b>52</b>
5.1 Instalación de Disparadores / Funciones SPs	52
<b>6 Lenguaje de Manipulación de Datos</b>	<b>54</b>
6.1 Poblamiento	54
6.2 Nota importante:	54

# 1 Modelo Entidad - Relación

Para empezar, tenemos que el modelo de datos entidad-relación (E-R) está basado en la percepción del mundo real que consta de una colección de objetos básicos, llamados **entidades**, y de **relaciones** entre estos objetos. Además, deberemos tener en mente que los **componentes** de este modelo son: una **entidad** junto con los **atributos** (simples, compuesto, multivaluados, derivados) que la describen así como el atributo para *identificar* únicamente, una **relación**. Así como las **restricciones**, es decir, la **cardinalidad** y la **participación**; y con las **extensiones**: **especialización** y **generalización**.

A continuación se verán las correcciones realizadas, las mejoras implementadas y las justificaciones técnicas, con base en las observaciones obtenidas durante la *práctica 03 parte 1* y la *práctica 03 parte 2*, así como en revisiones posteriores.

## ENTIDADES

Primero se mostrará la **regla de negocio** en la que nos basamos para la **entidad** de nuestro **Modelo Entidad - Relación**.

### **Regla de negocio:**

El **bazar** es itinerante, es decir, cuando realiza el evento el bazar no siempre será en el mismo lugar, para ello es necesario guardar la ubicación del lugar en donde se realizará (**calle**, **número interior**, **número exterior**, **colonia**, **estado**), además es necesario saber las **amenidades** con las que cuenta (estacionamiento, carpas, toma de corrientes, wifi, baño, etc...) las cuales dependerán de si el bazar se realizará **al aire libre** o **al cubierto**.

El bazar puede durar entre un día a varios días. Para ello es necesario guardar ademas la **fecha de inicio** y de **termino** del bazar.

### 1.1.1.Bazar

- Entidad fuerte.
- Tiene como **identificador** a **IdBazar**, lo consideramos como int.
- **Amenidad** es un **atributo multivaluado** para poder guardar entre todas las opciones: estacionamiento, carpas, toma de corrientes, wifi, baño, etc. En el **atributo simple Modalidad** guardamos si el bazar es al aire libre o en un lugar cerrado.
- Además, de contar con los **atributos simples**: **FechaInicio** y **FechaTermino**; y el **atributo compuesto** **Ubicacion** (Modalidad, Calle, NumeroInterior, NumeroExterior, Colonia y Estado).

### **Regla de negocio:**

Se deberá tener registro sobre los **negocios** para ello se necesitara guardar: **Nombre del negocio**, **teléfono(s)**, **correo(s)**, enlaces a las **redes sociales** del negocio, **rango de precios** y una breve **descripción** de lo que se venderá.

### 1.1.2.Negocio

- Entidad fuerte.
- Tiene como **identificador** **IdNegocio** y lo consideramos como int.
- Consideramos los atributos **Telefono**, **Correo** y **RedSocial** como **atributos multivaluados** para poder guardar la cantidad de datos que se requieran según el negocio.

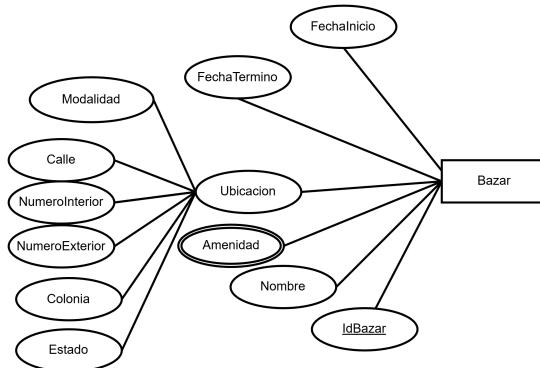


Figure 1.1: Entidad Bazar en el Modelo Entidad - Relación

- Para poder asegurar que el rango de precios se guarde correctamente, decidimos hacerlo como **atributo compuesto** RangoPrecio (PrecioMinimo y PrecioMaximo).

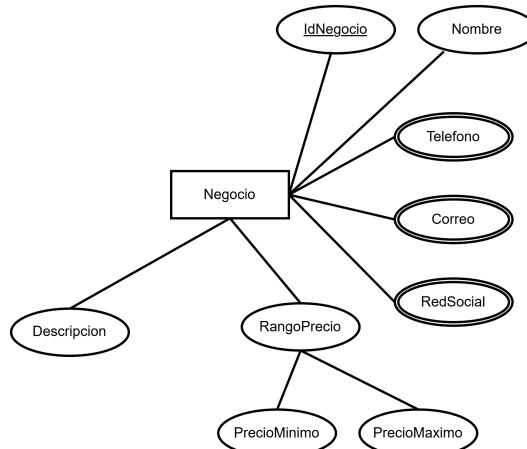


Figure 1.2: Entidad Negocio en el Modelo Entidad - Relación

### Regla de negocio:

Se tendrá un **estand**, donde estos podrán presentar sus productos, para ello es necesario guardar: El **numero de stand** y el **precio**. Por defecto cada stand tiene el **paquete** básico sin precio extra. El cual cuenta con las **amenidades** de 1 mesa y 2 sillas. El paquete premium que cuenta con 2 mesas y 4 sillas, cuyo precio es un 10% más del precio base del stand. Y el paquete emprendedor que cuenta con 3 mesas, 6 sillas, una pantalla táctil y toma de corriente para los productos, cuyo precio es un 30% más del precio base del stand.

#### 1.1.3.Estand:

- Entidad fuerte.
- Tiene como identificador **Número** y lo consideramos como int.
- Además de los **atributos simples** **Precio** y **Paquete**, el **atributo multivaluado** **Amenidad** pues varia dependiendo del paquete pudiendo tener cierta cantidad de mesas y sillas, una pantalla táctil y toma corriente y el **atributo derivado** **PrecioTotal** pues este se calculará despendiendo del paquete.

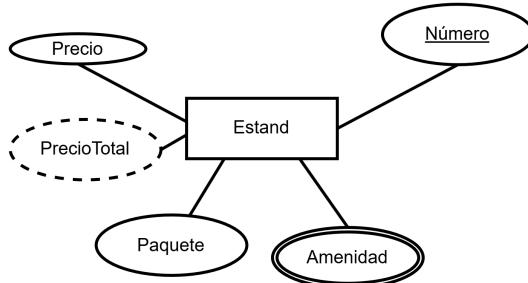


Figure 1.3: Entidad Estand en el Modelo Entidad - Relación

**Regla de negocio:**

Por cada **empreendedor** sera necesario guardar: **RFC**, **nombre completo** (**nombre**, **apellido paterno**, **apellido materno**), **domicilio** (**calle**, **numero interior**, **numero exterior**, **colonia**, **estado**), **telefono(s)**, **correo(s)**, **fecha de nacimiento** y **genero**.

**1.1.4. Emprendedor:**

- Entidad fuerte.
- Tiene como identificador **RFC**, considerado como una cadena de 13 caracteres: 4 letras, 6 dígitos, 2 letras y 1 dígito.
- Consideramos los atributos **Telefono** y **Correo** como **atributos multivaluados** pues podría tener más de uno.
- **Atributos compuestos** como **NombreCompleto** (**Nombre**, **ApellidoPaterno**, **ApellidoMaterno**) y **Domicilio** (**Calle**, **Colonia**, **Estado**, **NumeroExterior**, **NumeroInterior**); así como los **atributos simples** **Genero** y **FechaNacimiento**.

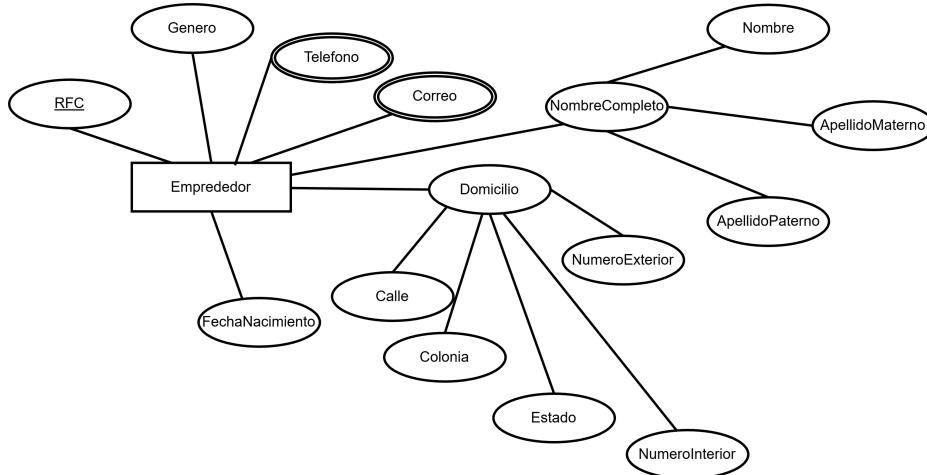


Figure 1.4: Entidad Emprendedor en el Modelo Entidad - Relación

**Regla de negocio:**

Es necesario tener el registro del **personal organizador**, los cuales se tendrán a **personal de limpieza**, **personal de seguridad** y **personal medico**. Estos se estarán rotando durante el evento, para ello es necesario guardar: **RFC**, **nombre completo** (**nombre**, **apellido paterno**, **apellido materno**), **domicilio** (**calle**, **numero interior**, **numero exterior**, **colonia**, **estado**), **telefono(s)**, **correo(s)**, **horario** (**hora inicio** y **hora final**) y **salario**. Hay que tener en cuenta que durante el evento se tendrá 3 horarios de 8 a.m. a 12 p.m., 12 p.m. a 4 p.m. y de 4 p.m. a 8 p.m.

Recordemos que la **generalización** observa que conjuntos de entidades, *comparten características comunes* (se describen mediante los mismos atributos y participan en los mismos conjunto de relaciones). Basada en sus similitudes, la generalización sintetiza estos conjuntos de entidades en uno solo, es decir, de nivel más alto.

### 1.1.5. PersonalOrganizador

Es una entidad fuerte así como una **superentidad** y por otro lado el personal de **limpieza**, personal de **seguridad** y personal **medico** son entidades fuertes y son las **subentidades**, de esta forma nos permite economizar la representación para no repetir atributos compartidos.

- Tiene como **identificador** **RFC**, considerado como una cadena de 13 caracteres: 4 letras, 6 dígitos, 2 letras y 1 dígito.
- Consideramos los atributos **Horario**, **Telefono** y **Correo** como **atributos multivaluados** para poder guardar toda la información de cada trabajador.
- Los **atributos compuestos** de **NombreCompleto** (Nombre, ApellidoPaterno, ApellidoMaterno) y **Domicilio** (Calle, Colonia, Estado, NumeroExterior, NumeroInterior), además de un **atributo simple** **Salario**.
- Consideramos una **especialización total con disyunción** pues realmente para este caso primero no nos interesa cualquier otro tipo de personal y esta planteado para que tenga que no permita ser más de un tipo de personal.

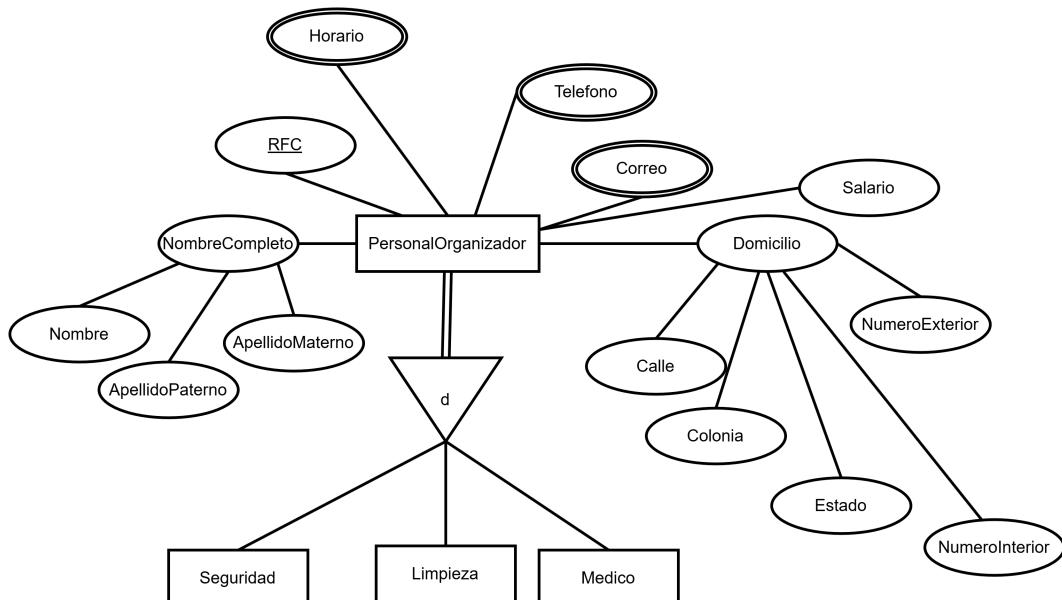


Figure 1.5: Superentidad Personal organizador y sus subentidades en el Modelo Entidad - Relación

### Regla de negocio:

Se necesitara generar un **ticket** con el **precio total** a pagar y el precio final con el impuesto del 20%.

### 1.1.6. Ticket

- Es una entidad fuerte.

- Su **identificador** es **IdTicket** y lo consideramos como int.
- El atributo Precio Total lo consideramos como **atributo derivado** pues así podremos calcular tanto el precio total a pagar como el precio final con el impuesto del 20%.

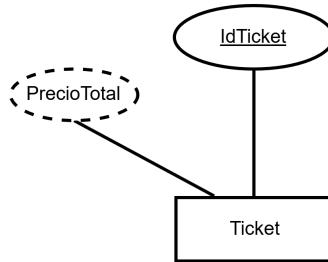


Figure 1.6: Entidad Ticket en el Modelo Entidad - Relación

### Regla de negocio:

Para los **productos** será necesario guardar: **Nombre del producto**, **tipo de producto**, **precio**, **presentación** (bolsa, lata, botella, etc), **descripción del producto**, en caso de ser un alimento o un **producto perecedero** (como cosméticos o medicamentos) se necesita guardar también **fecha de preparación** y **fecha de caducidad**.

#### 1.1.7. Producto

- Lo manejaremos como una **entidad débil** pues consideramos que su existencia (al menos en la base de datos) depende de que un negocio lo ofrezca y así mismo puede que más de un negocio venda un mismo producto por lo que esto nos ayudará a identificar de qué negocio proviene.
- También por 1.1.4 podemos ponerla como una **superentidad** y así **Producto Perecedero** será una **subentidad** debido a que, en este caso, también necesitamos guardar **FechaCaducidad** y **FechaPreparacion** (mismos que son **atributos simples**) pues permite economizar la representación para no repetir atributos compartidos.
- Consideraremos una **especialización parcial con disyunción** pues el caso en el que más encaja es “Algunas entidades de nivel más alto pueden no pertenecer a algún conjunto de entidades de nivel más bajo”.
- Su **identificador** es **IdProducto** y lo consideramos como int.
- Y los demás **atributos simples** de Producto son **Nombre**, **Tipo**, **Precio**, **Presentacion** y **Descripcion**.

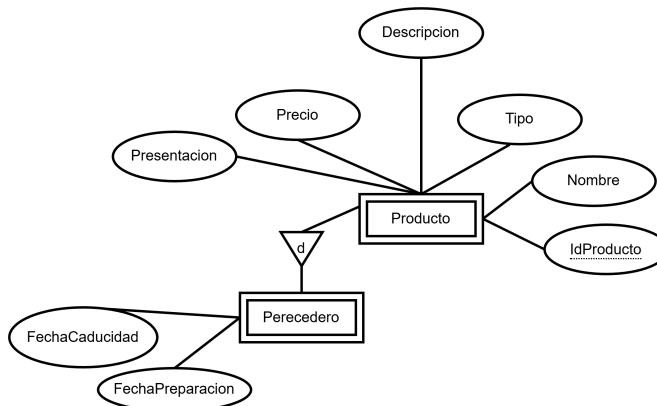


Figure 1.7: Superentidad débil Producto y su subentidad débil Perecedero en el Modelo Entidad - Relación

### Regla de negocio:

En el caso de los **servicios** es necesario guardar: **Nombre del servicio**, **tipo del servicio**, **duración**, **precio** y una **descripción del servicio** que se esta proporcionando.

#### 1.1.8.Servicio

- Lo manejaremos como una **entidad débil** pues consideramos que su existencia (al menos en la base de datos) depende de que un negocio lo ofrezca y así mismo puede que más de un negocio realice un mismo servicio por lo que esto nos ayudará a identificar de que negocio proviene.
- Su **identificador** es **IdServicio** y lo consideramos como int.
- Y los demás **atributos simples** de Servicio son **Nombre**, **Tipo**, **Precio**, **Duracion** y **Descripcion**.

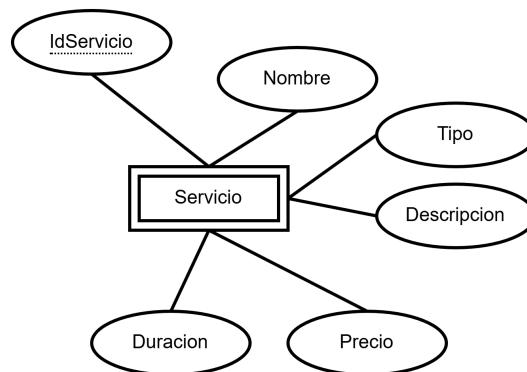


Figure 1.8: Entidad débil Servicio en el Modelo Entidad - Relación

Nota: Separamos Producto y Servicio en dos entidades débiles separadas debido a que pensamos que un negocio puede ofrecer servicios y vender productos al mismo tiempo sin que una afecte a la otra.

### Regla de negocio:

Respecto a los **clientes** que van al bazar, para ello es necesario guardar: Su **nombre completo** (**nombre**, **apellido paterno**, **apellido materno**), y **método de pago** (efectivo, tarjeta). En caso de que el cliente desee realizar sus compras en linea. Sera necesario guardar su **domicilio de entrega** (**calle**, **número interior**, **número exterior**, **colonia** y **estado**)

#### 1.1.9.Cliente

Para este punto lo mejor es utilizar [1.1.4](#) dejando así que **Cliente** es una entidad fuerte así como una **superentidad** y por otro lado **Fisico** y **Virtual** sean entidades fuertes y las **subentidades**, de esta forma nos permite economizar la representación para no repetir atributos compartidos.

- Tanto el **identificador** **IdCliente** como el **atributo compuesto** de **NombreCompleto** (**Nombre**, **ApellidoPaterno**, **ApellidoMaterno**) lo compartirán.
- Consideramos una **especialización total con traslape** primero porque en caso de que no fuera total tendríamos el problema de que puedan considerarse clientes que ni están en el bazar (sería muy ambiguo) y además sabemos que existe la posibilidad que un mismo cliente desee hacer una compra tanto en forma fisica como en virtual y el **traslape** lo permite.

- Consideramos el atributo **MetodoPago** como **atributos multivaluados** pues así nos aseguramos que un mismo cliente tenga la posibilidad de pagar tanto en efectivo como con tarjeta.
- Y por el lado de **Virtual** tenemos un atributo compuesto **Domicilio** (Calle, Colonia, Estado, NumeroExterior, NumeroInterior).

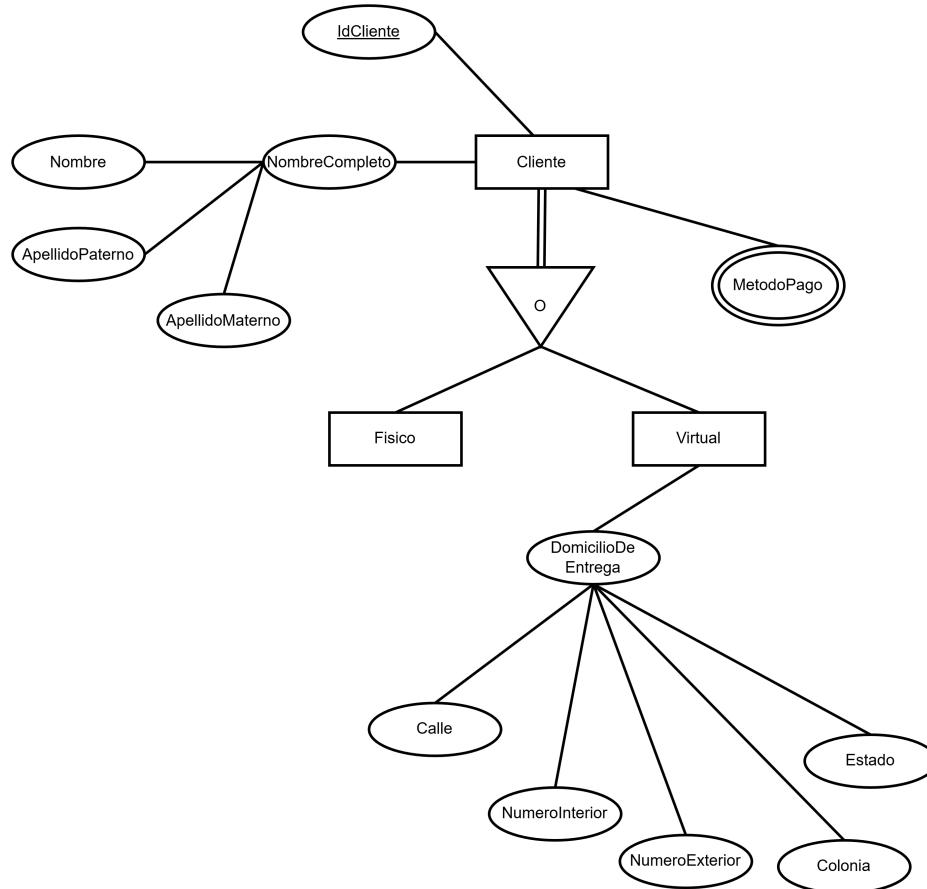


Figure 1.9: Superentidad Cliente y sus subentidades en el Modelo Entidad - Relación

### Regla de negocio:

Y la información de su **tarjeta** (número de tarjeta, vencimiento y **CVV**).

#### 1.1.10.Tarjeta

- Entidad fuerte.
- Tiene el **identificador** **NumeroTarjeta**.
- Atributos simples** **vencimiento** y **CVV**.

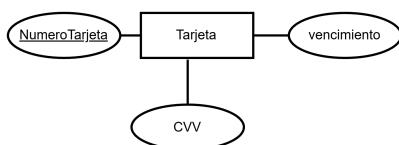


Figure 1.10: Entidad Tarjeta en el Modelo Entidad - Relación

## RELACIONES

Ahora, para esta parte igual mostraremos la **regla de negocio** en la que nos estamos basando para esa **relación** de nuestro **Modelo Entidad - Relación**. Por lo que, únicamente se mostrará la relación y no los atributos de las entidades pero estas siguen siendo parte de nuestro modelo.

### **Regla de negocio:**

Se tendrá una agenda con los **negocios** que asistirán al **bazar** para vender sus mercancías esto con la finalidad de que todos los emprendedores tengan la oportunidad de vender, donde se deben considerar: El negocio y **fecha que asistirá**. Un negocio **puede asistir varias veces** al bazar por mientras este dure, así que sera importante tener un control sobre los negocios que se presentarán, ya que por día se estarán presentando un total de 20 negocios.

#### 1.2.1. Agendar

- Decidimos poner la **cardinalidad varios a varios** para ambos lados y la razón es que nos indican que un negocio puede asistir varias veces al bazar, es decir, un negocio puede ser agendado varias veces al bazar, también sabemos que un bazar puede agendar hasta 20 negocios por día y además con esto no tendremos problemas con que un mismo negocio se pueda agendar en los siguientes bazares.
- En cuanto a la **participación** esta será **total** para **negocio** pues tenemos que si un negocio asistirá al bazar esta obligada a estar en la relación **agendar** y en por otro lado la **participación** para **bazar** es **parcial** pues sabemos que el bazar existe aún sin que haya agendado negocios.
- Como nos piden considerar la **FechaAsistencia** del negocio al bazar, este será un **atributo simple** pero de la relación.

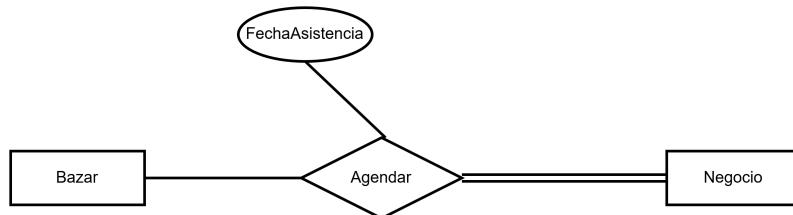


Figure 1.11: Relación Agendar de las entidades Bazar y Negocio en el Modelo Entidad - Relación

### **Regla de negocio:**

De la **regla de negocio 1.2** omitimos<sup>1</sup> la información que se presentará a continuación: El **negocio**, **estand** donde se presentará, el paquete del estand que se **escogió**.

Para **cada uno de los negocios se tendrá un estand**, donde estos podrán presentar sus productos.

#### 1.2.2. Escoger

- La **cardinalidad** es **varios a uno** la razón es que un negocio solo **escoge** un estand pero un estand, sin embargo, puede ser **escogido** por más de un negocio (aunque claro considerando que no será en la misma fecha).
- En cuanto a la **participación** esta será **total** para **negocio** pues un negocio debe tener un estand. Pero en cambio tenemos que un estand puede no estar asignado, esto puede ocurrir cuando ningún negocio lo ha

<sup>1</sup>La razón por la que lo omitimos es porque no se tenía que abordar en la relación anterior.

escogido, dejando así que la **participación** es **parcial** para **estand**.



Figure 1.12: Relación Escoger de las entidades Estand y Negocio en el Modelo Entidad - Relación

### **Regla de negocio:**

Para ello se necesitara generar un **ticket** con el **empreendedor** que **cobro** el pago.

#### 1.2.3. Cobrar

- Para empezar la **cardinalidad** es de **varios a uno**, pues tenemos que varios tickets pueden tener al mismo emprendedor que hizo el **cobro**, sin embargo, un emprendedor al **cobrar** una compra solo genera el ticket de respectiva venta.
- En cuanto a la **participación** esta será **total** para **ticket** pues un ticket depende que que el emprendedor haya realizado la venta. Pero en cambio tenemos que un emprendedor existira aún si no realiza ninguna venta y no necesariamente estará en todos los tickets, dejando así que la **participación** es **parcial** para **empreendedor**.

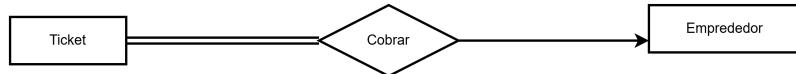


Figure 1.13: Relación Cobrar de las entidades Ticket y Emprendedor en el Modelo Entidad - Relación

### **Regla de negocio:**

Para ello se necesitara generar un **ticket** con el nombre del **cliente**

#### 1.2.4. Comprar

- Para empezar la **cardinalidad** es de **uno a varios**, pues tenemos que varios tickets pueden tener al mismo cliente que hizo la compra, sin embargo, un cliente al **comprar** solo genera el ticket de respectiva venta.
- En cuanto a la **participación** esta será **total** para **ticket** pues un ticket depende que que el cliente haya realizado la compra. Pero en cambio tenemos que un cliente no necesariamente estará en todos los tickets, dejando así que la **participación** es **parcial** para **cliente**.

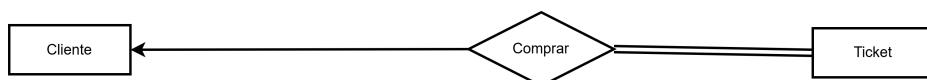


Figure 1.14: Relación Comprar de las entidades Cliente y Ticket en el Modelo Entidad - Relación

### Regla de negocio:

Retomaremos cierta información de la [regla de negocio 1.1.8](#) y de la [regla de negocio 1.1.9](#).

En caso de que el [cliente](#) desee realizar sus compras en linea. Sera necesario [guardar](#) la información de su [tarjeta](#).

#### 1.2.5.Domiciliar

- Para empezar la [cardinalidad](#) es de [uno a varios](#), un cliente puede tener varias tarjetas y varias tarjetas pueden estar asociadas a un mismo cliente pero cada Tarjeta está asociada solo con una entidad Domiciliar.
- En cuanto a la [participación](#) esta será [total](#) de ambos lados porque para que un cliente compre en línea, es necesario que registre una tarjeta y a la vez, cada tarjeta registrada es de un cliente en particular.

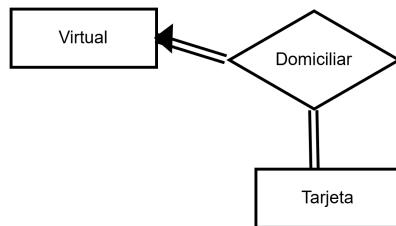


Figure 1.15: Relación Domiciliar de las entidades Virtual y Tarjeta en el Modelo Entidad - Relación

### Regla de negocio:

Cada [negocio a lo más](#) podrá tener 2 [empreendedores](#) a cargo, para que los días en que ocurra el bazar, no se tenga problemas con el cupo.

#### 1.2.6.Emprender

- Con el [atributo derivado](#) [NumeroDeEmprededores](#) de la relación podremos calcular cuantos emprendedores están a cargo de ese negocio (donde deberemos cuidar que sean a lo más 2).
- Para empezar la [cardinalidad](#) es de [varios a uno](#), debido a que un emprendedor se asocia con a lo sumo un negocio. Un negocio, sin embargo, se puede asociar con cualquier número de emprendedores (ninguno o varios) pero en este caso debe de ser a lo más 2.
- En cuanto a la [participación](#) esta será [total](#) de ambos lados la razón es porque el caso de uso menciona que por cada negocio se tiene que tener 2 emprendedores, entonces no puede pasar que un negocio no tenga emprendedores encargados de este.

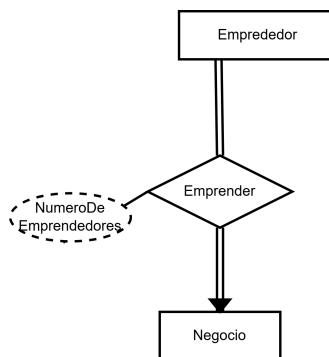


Figure 1.16: Relación Emprender de las entidades Emprededor y Negocio en el Modelo Entidad - Relación

### Regla de negocio:

Cada vez que algún cliente efectuó una compra en presencial o en linea el bazar se quedara con la ganancia del 20% sobre el precio total a pagar.

#### 1.2.7.Pagar

- Con el **atributo simple** Comision de la relación guardar la ganancia, ya que justamente siempre se estara cobrando un 20%.
- Para empezar la **cardinalidad** es de **varios a uno**, debido a que un ticket se asocia con a lo sumo un bazar. Un bazar, sin embargo, se puede asociar con cualquier número de tickets (ninguno o varios).
- En cuanto a la **participación** esta será **total** para **ticket** pues obligadamente cada ticket debe pertenecer a un bazar. Sin embargo, en el peor escenario, puede pasar que un bazar no tenga tickets, dejando así que la **participación** es **parcial** para **bazar**.

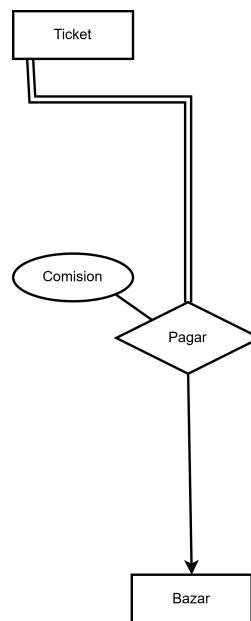


Figure 1.17: Relación Pagar de las entidades Ticket y Bazar en el Modelo Entidad - Relación

### Regla de negocio:

Para ello se necesitara generar un ticket con el negocio en donde se realizo la compra

#### Imprimir:

- Para empezar la **cardinalidad** es de **varios a uno**, pues tenemos que varios tickets pueden tener al mismo negocio en donde se realizo la compra, sin embargo, un negocio al **cobrar** una venta solo genera el ticket de respectiva venta.
- En cuanto a la **participación** esta será **total** para **ticket** pues un ticket depende que que el negocio haya realizado la venta. Pero en cambio tenemos que un negocio existira aún si no realiza ninguna venta y no necesariamente estará en todos los tickets, dejando así que la **participación** es **parcial** para **empreendedor**.

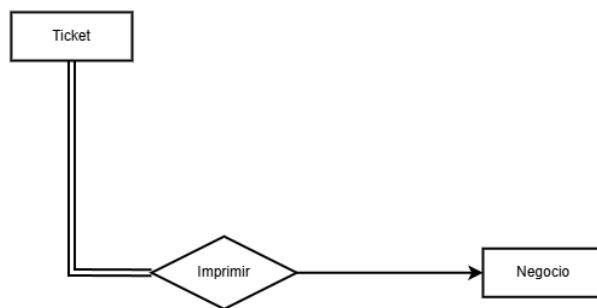


Figure 1.18: Relación Imprimir de las entidades Ticket y Negocio en el Modelo Entidad - Relación

### Regla de negocio:

Cada negocio debe de tener un registro de la cantidad de productos que se tiene en stock. Esto con la finalidad de que aquellos negocios que prefieran vender su mercancía en linea, puedan realizarlo desde la página del bazar.

#### 1.2.8.Vender

- Es una **relación identificador** de la **entidad débil Producto**. Esto es necesario pues existe el caso donde dos o más negocios vendan el mismo producto ej. leche de fresas por lo cual de esta manera sabremos que leche de fresa pertenece a qué negocio.
- Con el **atributo simple Stock** podremos guardar la cantidad de productos en stock pero es necesario también contar con un **atributo derivado StockDisponible** para cumplir con el objetivo pero pues esto se calculará respecto a si se han vendido y cuestiones así por lo que están como atributos de la relación.
- Para empezar la **cardinalidad** es de **varios a uno**, pues tenemos que varios productos pueden ser vendidos en el mismo negocio, sin embargo, un negocio solo puede vender su respectivo producto y de la misma forma un producto específico solo se vende en el negocio al que pertenece.
- En cuanto a la **participación** esta será **total** para **producto** pues un producto depende que el negocio lo venda. Pero en cambio tenemos que en un negocio no necesariamente tiene que vender todos los productos, dejando así que la **participación** es **parcial** para **negocio**.

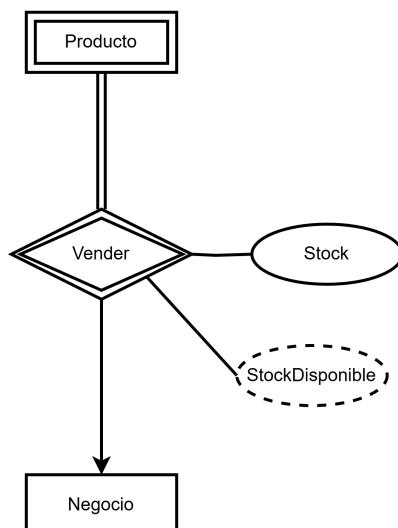


Figure 1.19: Relación Vender de las entidades Producto y Negocio en el Modelo Entidad - Relación

### Regla de negocio:

Por cada uno de los **negocios** se necesitara tener un control de los **servicios** que están proporcionando.

#### 1.2.9.Ofrecer

- Es una **relación identificador** de la **entidad débil Servicio**. Esto es necesario pues existe el caso donde dos o más negocios vendan el mismo servicio por lo cual de esta manera sabremos que servicio pertenece a que negocio.
- Para empezar la **cardinalidad** es de **varios a uno**, pues tenemos que varios servicios pueden ser **ofrecidos** en el mismo negocio, sin embargo, un negocio solo puede **ofrecer** su respectivo servicio y de la misma forma un servicio en específico solo se ofrece en el negocio al que pertenece.
- En cuanto a la **participación** esta será **total** para **servicio** pues un servicio depende que el negocio lo ofrezca. Pero en cambio tenemos que en un negocio no necesariamente tiene que ofrecer todos los servicios, dejando así que la **participación** es **parcial** para **negocio**.

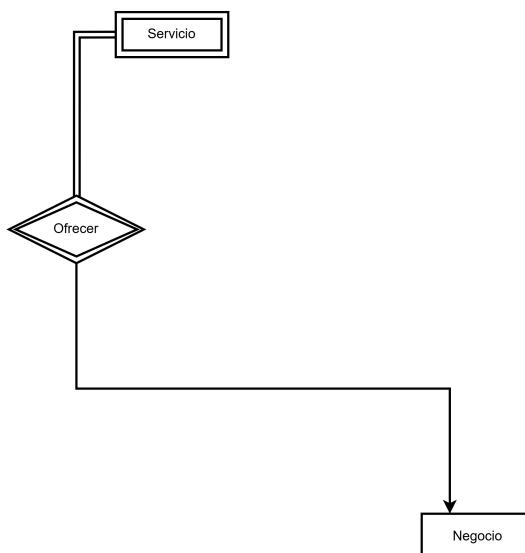


Figure 1.20: Relación Ofrecer de las entidades Servicio y Negocio en el Modelo Entidad - Relación

### Regla de negocio:

Retomamos la regla de negocio 1.1.6. Y para ello se necesitara **generar** un **ticket** con los **productos** que se compraron.

#### 1.2.10.Registrar

- Contara con un **atributo simple Cantidad** en la relación.
- Para empezar la **cardinalidad** es de **varios a varios**, pues tenemos que varios productos pueden estar **registrados** en el mismo ticket, varios ticket pueden tener el mismo producto **registrado**.
- En cuanto a la **participación** esta será **parcial** para ambos lados pues podría pasar que el negocio no ofrezca servicios, y el servicio no tenga ningún producto.

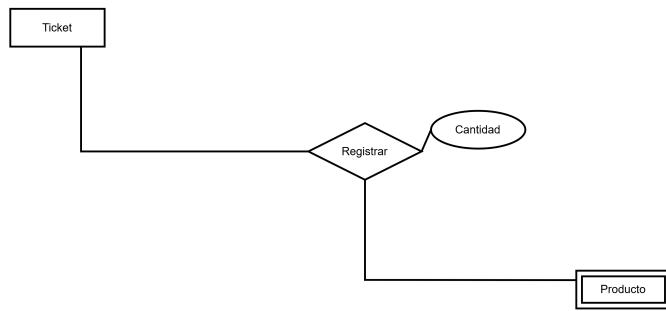


Figure 1.21: Relación Registrar de las entidades Ticket y Producto en el Modelo Entidad - Relación

### **Regla de negocio:**

Retomamos la regla de negocio 1.1.7. Y para ello se necesitara generar un **ticket** con los **servicios** que se compraron.

#### 1.2.11. Registrar

- Será una relación fuerte<sup>2</sup>. Contará con un **atributo simple** **Duracion** en la relación.
- Para empezar la **cardinalidad** es de **varios a varios**, pues tenemos que varios servicios pueden estar registrados en el mismo ticket, varios ticket pueden tener el mismo servicio registrado.
- En cuanto a la **participación** esta será **parcial** para ambos lados pues podría pasar que el negocio nada más da productos, y no ofrecen ningun servicio.

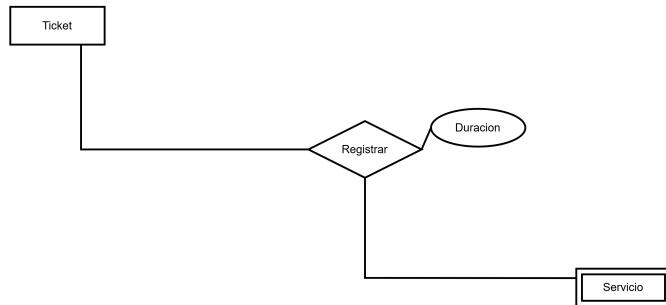


Figure 1.22: Relación Registrar de las entidades Ticket y Servicio en el Modelo Entidad - Relación

### **Regla de negocio:**

Retomamos la regla de negocio 1.1.4.

#### 1.2.12. Trabajar

- Con un atributo simple **FechaAsistencia** en la relación.
- Para empezar la **cardinalidad** es de **varios a varios**, pues tenemos que varias personas pueden **trabajar** en el mismo bazar, también una persona puede **trabajar** en varios bazares.

<sup>2</sup>Esta relación utiliza el mismo verbo de la anterior y en el **Modelo Entidad - Relación** no hay ningún problema pero tenemos en mente que para el **Modelo Relacional** se tendrá que nombrar respectivamente pero por el momento lo dejaremos así.

- En cuanto a la **participación** esta será **total** para ambos lados pues **PersonalOrganizador** porque se necesita asignarles un bazar donde trabajar. Y por otro lado, la idea es que si ocurre un **bazar** siempre se tenga a gente que esté cuidando, limpiando y curando a la gente que visita el bazar.

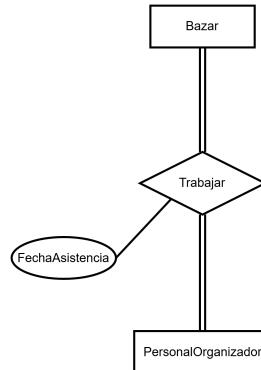


Figure 1.23: Relación Trabajar de las entidades Bazar y PersonalOrganizador en el Modelo Entidad - Relación

De este modo, hemos realizado nuestro **Modelo Entidad-Relación**, el cual se encuentra en la carpeta **Diagramas**.

## 2 Modelo Relacional

Modelo Relacional (conceptos)  $\Rightarrow$  Modelo Lógico de la BD (esquema BD)

Diseño lógico de la BD: Crear estructuras de BD estables, correctamente expresadas en un lenguaje técnico  
 $\Rightarrow$  SQL

### Traducción del Modelo E/R al Modelo Relacional

#### Estrategia a seguir:

1. Entidades fuertes
  - Atributos multivaluados
  - Herencia
2. Entidades débiles
  - Atributos multivaluados
  - Herencia
3. Relaciones que generan tabla
  - N:M
  - 1:1 (Parcial ambos)
  - 1:1 (Total ambos)
4. Relaciones que modifican tablas existentes
  - 1:N
  - 1:1 (Total 1 lado)

#### Regla general:

- \* Para cada tipo de entidad y cada tipo relación **existe una tabla (relación) única** con el **nombre de tal tipo**.
- \* Cada **relación** (tabla) tiene una **cantidad de atributos** (columnas) **fija**, cada uno con **nombre único**.

#### 2.1.1. Entidades fuertes

Para esta primera parte nos enfocaremos en lo siguiente:

Un tipo de entidad fuerte se convierte en una **relación** con los mismos atributos que la describen. La **llave primaria** de la relación es el **identificador del tipo entidad**.

Como **entidad fuerte** **Ticket** (modelo Entidad - Relación) nos daría

**Ticket** (**IdTicket**<sup>PK</sup>)

- Su identificador es **IdTicket** y lo consideramos como un entero positivo consecutivo.

- El atributo **PrecioTotal** no se traduce porque es calculado (y depende del 20% de comisión para el bazar).

**Atributos multivaluados:** Estos atributos se convierten en **tablas** y no en columnas. Si  $M$  es un **atributo multivaluado** se crea una **tabla  $T$**  con una columna que corresponde a la **llave primaria** del **tipo de entidad** del que  $M$  es atributo y **otra para el atributo**.

Como **entidad fuerte Bazar** (modelo Entidad - Relación) nos daría

**Bazar**(IdBazar<sup>PK</sup>, NombreBazar, Calle, NúmeroInterior, NúmeroExterior, Colonia, Estado, Modalidad, FechaInicio, FechaFin)

- Tiene como **llave primaria** IdBazar, el cual se considera como un **entero positivo consecutivo**.
- Para los atributos NombreBazar, Calle, NúmeroInterior, NúmeroExterior, Colonia y Estado; su dominio es **varchar** y su longitud se considera en el modelo relacional.
- En el atributo Modalidad guardamos si el bazar es al aire libre o en un lugar cerrado.
- FechaInicio y FechaFin son de tipo **date** en un formato DD/MM/YYYY.

Pero recordemos que cuenta con un **atributo multivaluado Amenidad**,

**AmenidadBazar** (IdBazar, AmenidadBazar)<sup>PK</sup>

- Tabla del atributo Multivaluado Amenidades.
- **AmenidadBazar** es de tipo **varchar**, para poder registrar todas las amenidades de un Bazar.
- **llave primaria compuesta** por IdBazar y AmenidadBazar
- **llave foránea** IdBazar la cual proviene como **llave primaria** de **Bazar**.

Como **entidad fuerte Negocio** en el **modelo Entidad - Relación** nos daría

**Negocio** (IdNegocio, NombreNegocio, Descripción, PrecioMínimo, PrecioMáximo)

- Tiene como **llave primaria** IdNegocio y lo consideramos como un entero positivo consecutivo.
- Los atributos NombreNegocio y Descripción son de dominio **varchar** y su longitud se especifica en el modelo relacional.
- PrecioMínimo y PrecioMáximo se consideran como **dominio numerico**, el cual tiene una precisión de 2 decimales.

pero recordemos que cuenta con el **atributo multivaluado Telefono**,

**TelefonoNegocio** (IdNegocio, Teléfono)<sup>PK</sup>

- Tabla del atributo Multivaluado **Telefono** de **Negocio**.
- Teléfono es de tipo **char** de longitud 10.
- **llave primaria compuesta** por IdNegocio y Teléfono.

- o llave foránea IdNegocio la cual proviene como llave primaria de Negocio.

también que cuenta con el **atributo multivaluado** Correo,

**CorreoNegocio** (IdNegocio, Correo)  
PK

- o Tabla del atributo Multivaluado Correo de Negocio.
- o Correo es de tipo **varchar** de longitud 50.
- o **llave primaria compuesta** por IdNegocio y Correo.
- o **llave foránea** IdNegocio la cual proviene como llave primaria de Negocio.

y por último, recordemos que cuenta con el **atributo multivaluado** RedSocial,

**RedSocialNegocio** (IdNegocio, RedSocial)  
PK

- o Tabla del atributo Multivaluado RedSocial de Negocio.
- o Correo es de tipo **varchar** de longitud 50.
- o **llave primaria compuesta** por IdNegocio y RedSocial.
- o **llave foránea** IdNegocio la cual proviene como llave primaria de Negocio.

Como **entidad fuerte** Estand (modelo Entidad - Relación) nos daría

**Estand** (Número, Paquete, Precio)

- o Tiene como **llave primaria** Número y lo consideramos como int entre 1 y 20.
- o El atributo Paquete lo consideramos para guardar el Paquete que eligieran los emprendedores al momento de reservar el Estand para ocupar en el Bazar, por lo que su dominio es char(1), únicamente para elegir el número del paquete.
- o El atributo Precio es para guardar el precio del paquete que se haya elegido, por lo que su **dominio es money**.

recordemos que cuenta con el **atributo multivaluado** Amenidad,

**AmenidadEstand** (Número, AmenidadEstand)  
PK

- o Tabla del atributo Multivaluado Amenidad de Estand.
- o AmenidadEstand es de tipo **varchar** de longitud 100.
- o **llave primaria compuesta** por Número y AmenidadEstand.
- o **llave foránea** Número la cual proviene como llave primaria de Estand.

Siguiendo nuestra estrategia, ahora nos podemos enfocar en la **herencia**.

Como **entidad fuerte PersonalOrganizador** (modelo Entidad - Relación) en la traducción del modelo entidad-relación al modelo relacional, siguiendo las reglas estándar para un caso de **especialización total con disyunción**, se representaría como una superentidad de las entidades especializadas: **Seguridad, Limpieza y Médico**, lo que resultaría en la creación de 3 tablas separadas en la base de datos:

- Seguridad
- Limpieza
- Médico

En este diseño o traducción en particular no estamos de acuerdo, puesto que lo único que diferencia de Médico, Limpieza y Seguridad, únicamente es el título, pues estas entidades especializadas no tienen un atributo de más. Y adjunto a esto queremos cambiar el diseño por dos razones principales:

- **Mejor organización del personal organizador:** Mantener la información de los trabajadores en una única tabla facilita la administración y consulta de datos.
- **Garantizar unicidad en la categoría del trabajador:** Cada trabajador debe pertenecer únicamente a una categoría. Es decir, no puede ser simultáneamente un Médico y un Empleado de Limpieza, por lo que cada RFC debe aparecer solo una vez en la base de datos.

Para lograr esto, en lugar de dividir la información en múltiples tablas, optaremos por un diseño que mantenga una única tabla **PersonalOrganizador** con un atributo que indique a qué categoría pertenece (Seguridad, Limpieza o Médico). Así, cada trabajador tendrá un solo registro en la base de datos, asegurando que cada RFC esté asociado exclusivamente a un tipo de trabajador.

Por lo que, siguiendo con la versión reducida del Modelo Relacional, tenemos que:

**PersonalOrganizador** (RFC<sup>PK</sup>, NombrePersonalOrganizador, APaternoPersonalOrganizador, AMaternoPersonalOrganizador, Calle, NúmeroExterior, NúmeroInterior, Colonia, Estado, Salario, esSeguridad, esLimpieza, esMédico)

- Tiene como identificador RFC, considerado como un tipo de dato **char**: 4 letras, 6 dígitos, 2 letras y 1 dígito.
- Los atributos NombreTrabajador, APaternoTrabajador, AMaterno, Calle, NúmeroExterior, NúmeroInterior, Colonia y Estado; son de tipo **varchar** y su longitud se define en el diagrama del modelo relacional.
- FechaNacimiento es de tipo **date** en un formato DD/MM/YYYY.
- Salario es de tipo **money**
- esSeguridad, esLimpieza, esMédico; son de tipo **boolean** y solamente uno puede tener el valor **true**. Esto nos permite cumplir con la restricción de los casos de uso<sup>1</sup>.

recordemos que cuenta con el **atributo multivaluado Horario**,

RFC HorarioPersonalOrganizador

**HorarioPersonalOrganizador** (RFC, HorarioPersonalOrganizador)

- Tabla del atributo Multivaluado Horario de **PersonalOrganizador**.
- Horario es de tipo **varchar** de longitud 30.
- Llave primaria compuesta por RFC y HorarioPersonalOrganizador.

<sup>1</sup>La **totalización** de la especialización se resolverá más adelante con un disparador.

- Llave Foránea RFC la cual proviene como llave primaria de **PersonalOrganizador**.

recordemos que cuenta con el **atributo multivaluado** **Telefono**,

**TelefonoPersonalOrganizador** (RFC, TeléfonoPersonalOrganizador)  
PK

- Tabla del atributo Multivaluado Teléfono de **PersonalOrganizador**.
- TeléfonoPersonalOrganizador es de tipo **char** de longitud 10.
- Llave primaria compuesta por RFC y TeléfonoPersonalOrganizador.
- Llave Foránea RFC la cual proviene como llave primaria de **PersonalOrganizador**.

recordemos que cuenta con el **atributo multivaluado** **Correo**,

**CorreoPersonalOrganizador** (RFC, CorreoPersonalOrganizador)  
PK

- Tabla del atributo Multivaluado Correo de **PersonalOrganizador**.
- CorreoPersonalOrganizador es de tipo **varchar** de longitud 50.
- Llave primaria compuesta por RFC y CorreoPersonalOrganizador.
- Llave Foránea RFC la cual proviene como llave primaria de **PersonalOrganizador**.

Como **entidad fuerte** **Emprendedor** (modelo Entidad - Relación) nos daría

**Emprendedor** (RFC<sup>PK</sup>, NombreEmprendedor, APaternoEmprendedor, AMaternoEmprendedor, Calle, NúmeroExterior, NúmeroInterior, Colonia, Estado, Género, FechaNacimiento)

- Tiene como identificador RFC, considerado como un tipo de dato *char*: 4 letras, 6 dígitos, 2 letras y 1 dígito.
- Los atributos NombreEmprendedor, APaternoEmprendedor, AMaternoEmprendedor, Calle, NúmeroExterior, NúmeroInterior, Colonia y Estado; son de tipo **varchar** y su longitud se define en el diagrama del modelo relacional.
- Género es de tipo **char** de longitud 1.

recordemos que cuenta con el **atributo multivaluado** **Telefono**,

**TelefonoEmprendedor** (RFC, TelefonoEmprendedor)  
PK

- Tabla del atributo Multivaluado Teléfono de **Emprendedor**.
- TeléfonoEmprendedor es de tipo **char** de longitud 10.
- Llave primaria compuesta por RFC y TelefonoEmprendedor.
- Llave Foránea RFC la cual proviene como llave primaria de **Emprendedor**.

recordemos que cuenta con el **atributo multivaluado** **Correo**,

**CorreoEmprendedor** (RFC, CorreoEmprendedor)  
PK

- Tabla del atributo Multivaluado Correo de **Emprendedor**.
- CorreoEmprendedor es de tipo **varchar** de longitud 50.
- Llave primaria compuesta por RFC y CorreoEmprendedor.
- Llave Foránea RFC la cual proviene como llave primaria de **Emprendedor**.

Como **entidad fuerte** **Cliente** (modelo Entidad - Relación) en la traducción del modelo entidad-relación al modelo relacional, siguiendo las reglas estándar para un caso de **especialización total con traslape**, obtenemos

**Cliente** (IdCliente<sup>PK</sup>, NombreCliente, APaternoCliente, AMaternoCliente, Calle, NumeroExterior, NumeroInterior, Colonia, Estado, esFisico, esVirtual)

- Es superentidad de **Físico y Virtual**, consideramos una especialización total con traslape. Por lo que se genera una única tabla Cliente con todos los atributos
- Su identificador es IdCliente y lo consideramos como entero positivo consecutivo.
- NombreCliente, APaternoCliente, AMaternoCliente, Calle, NúmeroExterior, NúmeroInterior, Colonia, Estado son atributos de tipo **varchar** y su longitud se especifica en el modelo relacional.
- esFisico y esVirtual son atributos de tipo **boolean** y no hay problema si ambos están en modo **true**<sup>2</sup>.

Pero recordemos que la entidad Fisico cuenta con el **atributo multivaluado** **MetodoPago**, entonces

**MetodoPago** (IdCliente, MetodoPago)<sup>PK</sup>

- MetodoPago es de tipo **varchar** para elegir entre efectivo o tarjeta.
- Llave primaria compuesta por IdCliente y MetodoPago.
- Llave foránea IdCliente que proviene de **Cliente** que es llave primaria.

Como **entidad fuerte** **Tarjeta** (modelo Entidad - Relación) y considerando a la entidad **Cliente** obtenemos,

**Tarjeta** (NumeroTarjeta<sup>PK</sup>, Vencimiento, CVV)

- NumeroTarjeta es la llave primaria y lo consideramos como tipo de dato **char** y su longitud es de 16 caracteres.
- Vencimiento es de tipo **date** en un formato DD/MM/YYYY.
- CVV es de tipo **varchar** y tiene una longitud de 4 caracteres (porque existen tarjetas con un cvv de longitud 3 y otras de 4).

### 2.1.2. Entidades débiles

Ahora, pasaremos a las entidades débiles teniendo en cuenta que,

Un tipo entidad débil se convierte en una **relación** con sus **mismos atributos** y su **llave es compuesta** (**llave entidad fuerte + llave entidad débil**)

<sup>2</sup>La **totalización** de la especialización se resolverá más adelante con un disparador.

Como entidad débil **Producto** (modelo Entidad - Relación) nos daría,

**Producto** (<sup>PK</sup>IdNegocio, IdProducto, NombreProducto, Descripción, Tipo, Precio, Presentación)

- IdNegocio es heredado de Negocio porque **Producto** es una entidad débil.
- IdProducto es de tipo **int** como entero positivo consecutivo.
- NombreProducto, Descripción, Tipo, Presentación son de tipo **varchar** y su longitud se especifica en el modelo relacional.
- PrecioProducto es de tipo **money**.

Como entidad débil **Perecedero** (modelo Entidad - Relación) nos daría,

**Perecedero** (<sup>PK</sup>IdNegocio, IdProducto, FechaPreparación, FechaCaducidad)

- Hereda IdNegocio y IdProducto de la tabla **Producto**.
- FechaPreparación y FechaCaducidad son de tipo **date** en un formato DD/MM/YYYY.

Como entidad débil **Servicio** (modelo Entidad - Relación) nos daría,

**Servicio** (<sup>PK</sup>IdNegocio, IdServicio, NombreServicio, Descripción, Tipo, PrecioServicio, Duración)

- IdNegocio es heredado de Negocio porque **Producto** es una entidad débil.
- IdServicio es de tipo **int** como entero positivo consecutivo.
- NombreServicio, Descripción, Tipo son de tipo **varchar** y su longitud se especifica en el modelo relacional.
- PrecioServicio es de tipo **money**.
- Duración es de tipo **time** (dado que solamente necesitamos las horas y/o minutos).

### 2.1.3.Relaciones que generan tabla

Para trabajar esta parte tenemos que:

Las relaciones *N:M* se convierten en una **relación**, los atributos que la conforman se forman con el **identificador** de cada una de las **entidades que relaciona** junto con los **atributos** de la **relación** (si existen).

La relación **Agendar** (modelo Entidad - Relación) nos daría

**Agendar** (idBazar<sup>FK</sup>, IdNegocio<sup>FK</sup>, FechaAsistencia)

- idBazar, IdNegocio son **llaves foráneas** por la cardinalidad n:m de la relación Agendar entre Bazar y Negocio.
- FechaAsistencia se considera de tipo date.

La relación **Trabajar** (modelo Entidad - Relación) nos daría

**Trabajar** (idBazar<sup>FK</sup>, RFC<sup>FK</sup>, FechaAsistencia)

- idBazar, RFC son **llaves foráneas** por la cardinalidad n:m de la relación Trabajar entre Bazar y Personal Organizador.
- FechaAsistencia se considera de tipo date.

La **relación Registrar** (modelo Entidad - Relación) nos daría

**RegistrarProducto** (IdTicket<sup>FK</sup>, IdNegocio<sup>FK</sup>, IdProducto<sup>FK</sup>, Cantidad)

- IdTicket y (IdNegocio, IdProducto) son **llaves foráneas** por la cardinalidad n:m de la relación Registrar entre Ticket y Producto.
- Cantidad es un dato de tipo **int**

La **relación Registrar** (modelo Entidad - Relación) nos daría

**RegistrarServicio** (IdTicket<sup>FK</sup>, IdNegocio<sup>FK</sup>, IdServicio<sup>FK</sup>, Duración)

- IdTicket y (IdNegocio, IdServicio) son **llaves foráneas** por la cardinalidad n:m de la relación Registrar entre Ticket y Servicio.
- Duración es un atributo de tipo **time** (dado que solamente necesitamos las horas y/o minutos que dura el servicio).

No contamos con relaciones **1:1** por lo que hemos terminado con las relaciones que generan tabla.

#### 2.1.4. Relaciones que modifican tablas existentes

Para la última parte veremos que

Relaciones 1 : N. En la relación **B** se incluye la **llave** de la relación **A** más los atributos de **R**.

La **relación Escoger** (modelo Entidad - Relación) tendríamos que modificar a la **entidad fuerte Emprendedor**, dejando así

**Negocio** (IdNegocio<sup>PK</sup>, Numero<sup>FK</sup>, NombreNegocio, Descripción, PrecioMínimo, PrecioMáximo)

- Relación entre **Estand** y **Negocio**.
- Numero es **llave foránea**, por la cardinalidad 1:n modifica la tabla **Negocio**

La **relación Comprar** (modelo Entidad - Relación) tendríamos que modificar a la **entidad fuerte Ticket**, dejando así

**Ticket** (IdTicket<sup>PK</sup>, idCliente<sup>FK</sup>)

- Relación entre **Cliente** y **Ticket**.
- Como tiene cardinalidad **1:N** modifica la tabla **Ticket** mandando la llave primaria de **Cliente**, es decir, idCliente se agrega a la tabla **Ticket** como **llave foránea**.

La **relación Pagar** (modelo Entidad - Relación) tendríamos que modificar a la **entidad fuerte Ticket**, dejando así

**Ticket** (IdTicket<sup>PK</sup>, idCliente<sup>FK</sup>, idBazar<sup>FK</sup>, ComisionBazar)

- Relación entre **Bazar** y **Ticket**.
- Como tiene cardinalidad **1:N** modifica la tabla **Ticket** mandando la llave primaria de **Bazar**, es decir, idBazar se agrega a la tabla **Ticket** como **llave foránea**.
- El atributo de la relación pagar, **ComisionBazar**, también se agrega a la tabla **Ticket**.

La **relación Imprimir** (modelo Entidad - Relación) tendríamos que modificar a la **entidad fuerte Ticket**, dejando así

**Ticket** (IdTicket<sup>PK</sup>, idCliente<sup>FK</sup>, idBazar<sup>FK</sup>, idNegocio<sup>FK</sup>, ComisionBazar)

- Relación entre **Negocio** y **Ticket**.
- Como tiene cardinalidad **1:N** modifica la tabla **Ticket** mandando la llave primaria de **Negocio**, es decir, idNegocio se agrega a la tabla **Ticket** como **llave foránea**.

La **relación Cobrar** (modelo Entidad - Relación) tendríamos que modificar a la **entidad fuerte Ticket**, dejando así

**Ticket** (IdTicket<sup>PK</sup>, idCliente<sup>FK</sup>, idBazar<sup>FK</sup>, idNegocio<sup>FK</sup>, RFC<sup>FK</sup>, ComisionBazar)

- Relación entre **Emprendedor** y **Ticket**.
- Como tiene cardinalidad **1:N** modifica la tabla **Ticket** mandando la llave primaria de **Emprendedor**, es decir, RFC se agrega a la tabla **Ticket** como **llave foránea**.

La **relación Emprender** (modelo Entidad - Relación) tendríamos que modificar a la **entidad fuerte Emprendedor**, dejando así

**Emprendedor** (RFC<sup>PK</sup>, IdNegocio<sup>FK</sup>, NombreEmprendedor, APaternoEmprendedor, AMaternoEmprendedor, Calle, NúmeroExterior, NúmeroInterior, Colonia, Estado, Género, FechaNacimiento)

- Relación entre **Negocio** y **Emprendedor**.
- Como tiene cardinalidad **1:N** modifica la tabla **Emprendedor** mandando la llave primaria de **Negocio**, es decir, IdNegocio se agrega a la tabla **Emprendedor** como **llave foránea**.
- Como su único atributo es calculado, no se agrega al modelo relacional.

La **relación Domiciliar** (modelo Entidad - Relación) tendríamos que modificar a la **entidad fuerte Tarjeta**, dejando así

**Tarjeta** (NumeroTarjeta<sup>PK</sup>, IdCliente<sup>FK</sup>, Vencimiento, CVV)

- Relación entre **ClienteVirtual** (que por la traducción, la tabla se llama Cliente) y **Tarjeta**.
- Como tiene cardinalidad **1:N** modifica la tabla **Tarjeta** mandando la llave primaria de **Cliente**, es decir, IdCliente se agrega a la tabla **Tarjeta** como **llave foránea**.

**Excepción:** Sabemos que cuando se trata de una **relación débil** la **relación** sería redundante porque ya está considerada en la **entidad débil** pero cuando la relación tiene atributos, en ese caso, se aplica alguna de las reglas anteriores (dependerá del tipo de cardinalidad). Y justo esto pasa con la **relación Vender** (modelo Entidad - Relación) tendríamos que modificar a la **entidad débil Producto**, dejando así

**Producto** (PK IdNegocio, IdProducto, NombreProducto, Descripción, Tipo, Precio, Presentación, Stock)

- Relación débil entre **Negocio** y **Producto**.
- Por su debilidad, no se genera tabla (porque ya está considerada en la tabla **Producto**). Sin embargo, al tener atributos, se agregan a la tabla **Producto**, en este caso, solamente se agrega Stock.

Por último, esta la relación **Ofrecer** que es la relación débil entre **Negocio** y **Servicio** pero no se traduce porque ya está considerada en la tabla **Servicio** y como no tiene atributos, tampoco modifica ninguna tabla.

## Modelo Relacional (Tablas)

A continuación, se mostrarán las relaciones resultantes después de nuestra conversión del **Modelo E-R** a **Modelo Relacional** (Pautas de traducción),

**Producto** (PK IdNegocio, IdProducto, NombreProducto, Descripción, Tipo, Precio, Presentación, Stock)

**Tarjeta** (PK NumeroTarjeta, IdCliente, Vencimiento, CVV)

Producto	
PK1,FK	<u>IdNegocio: int</u>
PK2	<u>idProducto: serial4</u>
	NombreProducto:varchar(100)
	Descripción: text
	Tipo: varchar(50)
	Precio: money
	Presentación: varchar(100)
	Stock: int

Tarjeta	
PK	<u>NúmeroTarjeta: char(16)</u>
FK1	<u>IdCliente: int</u>
	Vencimiento: date
	CVV: varchar(4)

Figure 2.2: Relación Tarjeta en Modelo Relacional

Figure 2.1: Relación Producto en Modelo Relacional

**RedSocialNegocio** (PK IdNegocio, RedSocial)

RedSocialNegocio	
PK1, FK	<u>IdNegocio: int</u>
PK2	<u>RedSocial: varchar(50)</u>

Figure 2.3: Relación RedSocialNegocio en Modelo Relacional

**CorreoNegocio** (PK IdNegocio, Correo)

CorreoNegocio	
PK1, FK	<u>IdNegocio: int</u>
PK2	<u>Correo: varchar(50)</u>

Figure 2.4: Relación CorreoNegocio en Modelo Relacional

**Ticket** (IdTicket<sup>PK</sup>, idCliente<sup>FK</sup>, idBazar<sup>FK</sup>, idNegocio<sup>FK</sup>, RFC<sup>FK</sup>, ComisionBazar)

**Negocio** (IdNegocio<sup>PK</sup>, Numero<sup>FK</sup>, NombreNegocio, Descripción, PrecioMínimo, PrecioMáximo)

Ticket	
PK	<u>IdTicket: serial4</u>
FK1	<u>IdCliente: int</u>
FK2	<u>idBazar: int</u>
FK3	<u>idNegocio: int</u>
FK4	<u>RFC: char(13)</u>
	ComisiónBazar: numeric(10,2)

Figure 2.5: Relación Ticket en Modelo Relacional

Negocio	
PK	<u>IdNegocio: serial4</u>
FK	<u>Numero: int</u>
	NombreNegocio: varchar(100)
	Descripción: text
	PrecioMínimo: numeric(10,2)
	PrecioMáximo: numeric(10,2)

Figure 2.6: Relación Negocio en Modelo Relacional

**TelefonoNegocio** (IdNegocio, Teléfono)

TelefonoNegocio	
PK1, FK	<u>IdNegocio: int</u>
PK2	<u>Teléfono: char(10)</u>

Figure 2.7: Relación TelefonoNegocio en Modelo Relacional

**RegistrarServicio** (IdTicket<sup>FK</sup>, IdNegocio<sup>FK</sup>, IdServicio<sup>FK</sup>, Duración)

RegistrarServicio	
FK1	<u>IdTicket: int</u>
FK2	<u>IdNegocio: int</u>
FK2	<u>IdServicio: int</u>
	Duración: time

Figure 2.8: Relación RegistrarServicio en Modelo Relacional

**RegistrarProducto** (IdTicket<sup>FK</sup>, IdNegocio<sup>FK</sup>, IdProducto<sup>FK</sup>, Cantidad)

**Trabajar** (idBazar<sup>FK</sup>, RFC<sup>FK</sup>, FechaAsistencia)

RegistrarProducto	
FK1	<u>IdTicket: int</u>
FK2	<u>IdNegocio: int</u>
FK3	<u>IdProducto: int</u>
	Cantidad: int

Figure 2.9: Relación RegistrarProducto en Modelo Relacional

Trabajar	
FK1	<u>idBazar: int</u>
FK2	<u>RFC: char(13)</u>
	FechaAsistenciaTrabajar: date

Figure 2.10: Relación Trabajar en Modelo Relacional

**Agendar** ([idBazar](#)<sup>FK</sup>, [IdNegocio](#)<sup>FK</sup>, FechaAsistencia)

Agendar	
FK1	<u><a href="#">idBazar: int</a></u>
FK2	<u><a href="#">IdNegocio: int</a></u>
FechaAsistencia: date	

Figure 2.11: Relación Agendar en Modelo Relacional

**MetodoPago** ([IdCliente](#), [MetodoPago](#))  
PK

MetodoPago	
PK1,FK	<u><a href="#">IdCliente: int</a></u>
PK2	<u><a href="#">MetodoPago: varchar(20)</a></u>

Figure 2.12: Relación MetodoPago en Modelo Relacional

**Cliente** ([IdCliente](#)<sup>PK</sup>, NombreCliente, APaternoCliente, AMaternoCliente, **Calle**, NumeroExterior, NumeroInterior, Colonia, Estado, esFisico, esVirtual)

**Estand**<sup>PK</sup> ([Número](#), Paquete, Precio)

Cliente	
PK	<u><a href="#">IdCliente: serial4</a></u>
	NombreCliente: varchar(100)
	APaternoCliente: varchar(100)
	AMaternoCliente: varchar(100)
	Calle: varchar(100)
	NúmeroExterior: varchar(50)
	NúmeroInterior: varchar(50)
	Colonia: varchar(100)
	Estado: varchar(100)
	EsFísico: bool
	EsVirtual: bool

Figure 2.13: Relación Cliente en Modelo Relacional

Estand	
PK	<u><a href="#">NúmeroEstand: int</a></u>
	Paquete: char(1)
	Precio: money

Figure 2.14: Relación Estand en Modelo Relacional

**AmenidadEstand** ([Número](#), [AmenidadEstand](#))  
PK

AmenidadEstand	
PK1,FK	<u><a href="#">NúmeroEstand: int</a></u>
PK2	<u><a href="#">AmenidadEstand: varchar(100)</a></u>

Figure 2.15: Relación AmenidadEstand en Modelo Relacional

**CorreoEmprededor** ([RFC](#), [CorreoEmprededor](#))  
PK

CorreoEmprededor	
PK1,FK	<u><a href="#">RFC: char(13)</a></u>
PK2	<u><a href="#">CorreoEmprededor: varchar(50)</a></u>

Figure 2.16: Relación CorreoEmprededor en Modelo Relacional

**Emprendedor** (RFC<sup>PK</sup>, IdNegocio<sup>FK</sup>, NombreEmprendedor, APaternoEmprendedor, AMaternoEmprendedor, Calle, NúmeroExterior, NúmeroInterior, Colonia, Estado, Género, FechaNacimiento)

**TelefonoEmprendedor** (RFC, TelefonoEmprendedor)

Emprendedor	
PK	<u>RFC: char(13)</u>
FK	<u>idNegocio: serial4</u>
	NombreEmprendedor: varchar(100) APaternoEmprendedor: varchar(100) AMaternoEmprendedor: varchar(100) Calle: varchar(100) NúmeroExterior: varchar(50) NúmeroInterior: varchar(50) Colonia: varchar(100) Estado: varchar(100) Género: char(1) FechaNacimiento: date

TeléfonoEmprendedor	
PK1, FK	<u>RFC: char(13)</u>
PK2	<u>TeléfonoEmprendedor: char(10)</u>

Figure 2.17: Relación Emprendedor en Modelo Relacional

Figure 2.18: Relación TelefonoEmprendedor en Modelo Relacional

**PersonalOrganizador** (RFC<sup>PK</sup>, NombrePersonalOrganizador, APaternoPersonalOrganizador, AMaternoPersonalOrganizador, Calle, NúmeroExterior, NúmeroInterior, Colonia, Estado, Salario, esSeguridad, esLimpieza, esMédico)

**CorreoPersonalOrganizador** (RFC, CorreoPersonalOrganizador)

PersonalOrganizador	
PK	<u>RFC: char(13)</u>
	NombrePersonalOrganizador: varchar(100) APaternoPersonalOrganizador: varchar(100) AMaternoPersonalOrganizador: varchar(100) Calle: varchar(100) NúmeroExterior: varchar(50) NúmeroInterior: varchar(50) Colonia: varchar(100) Estado: varchar(100) Salario: money EsSeguridad: bool EsLimpieza: bool EsMédico: bool

CorreoPersonalOrganizador	
PK1, FK	<u>RFC: char(13)</u>
PK2	<u>CorreoPersonalOrganizador: varchar(50)</u>

Figure 2.19: Relación PersonalOrganizador en Modelo Relacional

Figure 2.20: Relación CorreoPersonalOrganizador en Modelo Relacional

**TelefonoPersonalOrganizador** (RFC, TeléfonoPersonalOrganizador)  
PK

**HorarioPersonalOrganizador** (RFC, HorarioPersonalOrganizador)  
PK

TelefonoPersonalOrganizador	
PK1, FK	<u>RFC: char(13)</u>
PK2	<u>TeléfonoPersonalOrganizador: char(10)</u>

Figure 2.21: Relación TelefonoPersonalOrganizador en Modelo Relacional

HorarioPersonalOrganizador	
PK1, FK	<u>RFC: char(13)</u>
PK2	<u>Horario: varchar(30)</u>

Figure 2.22: Relación HorarioPersonalOrganizador en Modelo Relacional

**Bazar**(IdBazar<sup>PK</sup>, NombreBazar, Calle, NumeroInterior, NumeroExterior, Colonia, Estado, Modalidad, FechaInicio, FechaFin)

**AmenidadBazar** (IdBazar, AmenidadBazar)  
PK

Bazar	
PK	<u>IdBazar: serial4</u>
	NombreBazar: varchar(100)
	Calle: varchar(100)
	NumeroInterior: varchar(50)
	NumeroExterior: varchar(50)
	Colonia: varchar(100)
	Estado: : varchar(100)
	Modalidad: varchar(80)
	FechaInicio: date
	FechaFin: date

AmenidadBazar	
PK1,FK	<u>idBazar: int</u>
PK2	<u>AmenidadBazar: varchar(100)</u>

Figure 2.24: Relación AmenidadBazar en Modelo Relacional

Figure 2.23: Relación Bazar en Modelo Relacional

**Perecedero** (IdNegocio, IdProducto, FechaPreparacion, FechaCaducidad)  
PK

**Servicio** (IdNegocio, IdServicio, NombreServicio, Descripcion, Tipo, PrecioServicio, Duración)

Perecedero	
PK1,FK1	<u>IdNegocio: int</u>
PK2,FK2	<u>IdProducto: int</u>
	FechaPreparacion: date
	FechaCaducidad: date

Figure 2.25: Relación Perecedero en Modelo Relacional

Servicio	
PK1,FK	<u>IdNegocio: int</u>
PK2	<u>idServicio: serial4</u>
	NombreServicio:varchar(100)
	Descripcion: text
	Tipo: varchar(50)
	Precio: money
	Duración: time

Figure 2.26: Relación Servicio en Modelo Relacional

# 3 Normalización

Uno de los principales problemas que se presentan cuando se convierte directamente diseños de bases de datos del modelo E/R al modelo relacional es la **REDUNDANCIA**.

**Redundancia.** Consiste en que un hecho se repita en más de una tupla de **forma innecesaria**. Una de las causas más comunes es debido al hecho de tratar de incluir en una relación atributos **univaluados** y **multivaluados**.

Otros conceptos que tendremos en mente son:

- Una **dependencia funcional**, denotada por  $X \rightarrow Y$ , ocurre entre **dos conjuntos de atributos  $X$  e  $Y$**  que son **subconjuntos de  $R$**  especifica **una restricción** sobre las **tuplas posibles** que pueden formar un **estado de la relación  $R$** .
- Una **dependencia funcional transitiva** se presenta cuando  $A$ ,  $B$  y  $C$  son **atributos de una relación** tales que si  $A \rightarrow B$  y  $B \rightarrow C$ , entonces  $C$  **depende transitivamente** de  $A$  a través de  $B$  (siempre que  $A$  no dependa funcionalmente de  $B$  o  $C$ ).

## Objetivos de la normalización

- Minimizar las redundacia de datos, evitando anomalías y conservando espacio de almacenamiento.
- Simplificar el cumplimiento de las **restricciones de integridad referencial**.
- Hacer más fácil el **mantenimiento** de datos (**insert**, **delete**, **update**)
- Proporcionar un **mejor diseño** (representación mejorada del mundo real) y una **base sólida** para el crecimiento futuro.
- Conservar las dependencias funcionales.

## Formas Normales

**Primera forma normal.** Cualquier atributo multivaluado (incluso grupos repetidos) han sido removidos. Solo se permiten valores atómicos y posiblemente nulos.

**Segunda forma normal.** Cualquier dependencia funcional parcial se ha removido (los atributos que no son llave se identifican por toda la llave primaria)

**Tercera forma normal.** Cualquier dependencia transitiva parcial se ha removido (los atributos que no son llave se identifican por toda la llave primaria)

Para este punto, tenemos la **primera forma normal** y **segunda forma normal**, debido a que realizamos un modelo E/R y después su traducción al **modelo relacional**. Por eso nos enfocaremos en la **Tercera forma normal**.

## Tercera Forma Normal

Una relación  $R$  está en **Tercera Forma Normal (3NF)** con respecto a  $F$ , si para toda dependencia no trivial  $A_1, A_2, A_3, \dots, A_n \rightarrow B$ , se tiene que:

1. El lado izquierdo  $\{ A_1, A_2, A_3, \dots, A_n \}$  es una **superllave** o bien,
2. El lado derecho  $B$ , es miembro de alguna llave candidata de  $R$ .

## Dependencias funcionales, redundancia, anomalías y normalización

En el **Modelo Relacional** teníamos la **relación**

**Cliente** (IdCliente<sup>PK</sup>, NombreCliente, APaternoCliente, AMaternoCliente, Calle, NumeroExterior, NumeroInterior, Colonia, Estado, esFisico, esVirtual)

El esquema ya cumple con la **Primera Forma Normal**, ya que **todos los atributos multivaluados han sido removidos**, permitiendo únicamente valores atómicos y posiblemente nulos. Pues recordemos que **Cliente** en el **Modelo Entidad - Relación** este cuenta con un **atributo multivaluado MetodoPago**

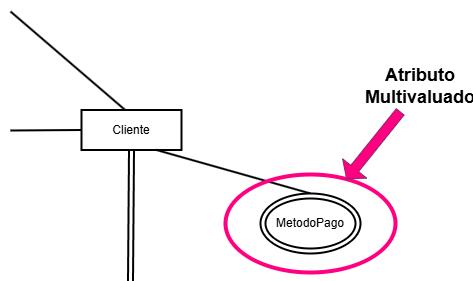


Figure 3.1: Enfocando el atributo multivaluado **MetodoPago** de la Entidad **Cliente** en el **Modelo E-R**

y al traducirlo a **Modelo Relacional** se **convierte en tabla** con una columna que corresponde a la **llave primaria** de Cliente

Cliente	
PK	<u>IdCliente: serial4</u>
	NombreCliente: varchar(100)
	APaternoCliente: varchar(100)
	AMaternoCliente: varchar(100)
	Calle: varchar(100)
	NúmeroExterior: varchar(50)
	NúmeroInterior: varchar(50)
	Colonia: varchar(100)
	Estado: varchar(100)
	EsFisico: bool
	EsVirtual: bool

MetodoPago	
PK1,FK	<u>IdCliente: int</u>
PK2	<u>MétodoPago: varchar(20)</u>

Figure 3.3: Relación MetodoPago en Modelo Relacional

Figure 3.2: Relación Cliente en Modelo Relacional

se esperaría que al buscar al **cliente N** debería solo regresar uno, es decir, al que corresponde el **cliente N**. También esperaríamos que a partir del IdCliente pudieramos obtener el NombreCliente, APaternoCliente, AMaternoCliente, Calle, NumeroExterior, NumeroInterior, Colonia, Estado, esFisico, esVirtual. Por lo tanto, todos sus atributos **dependen funcionalmente** de este identificador y esto nos diría que se cumple la **Segunda Forma Normal**.

IdCliente → NombreCliente, APaternoCliente, AMaternoCliente, Calle, NumeroExterior, NumeroInterior, Colonia, Estado, esFisico, esVirtual

Pero aquí existe un problema de **redundancia**, pues sabemos que la información del domicilio solo se guarda **en caso de que el cliente desee realizar sus compras en linea**. Sin embargo, aquí se está guardando la información del domicilio aún si el cliente compra de forma presencial y además puede que no desee comprar en linea, esto se puede visualizar de una mejor manera con la siguiente tabla

#### Cliente

<u>IdCliente</u> <sup>PK</sup>	NombreCliente	APaternoCliente	AMaternoCliente	Calle	NumeroExterior	NumeroInterior	Colonia	Estado	esFisico	esVirtual
1	Ana	Pérez	López	Av. Central	123	A	Roma	CDMX	true	true
2	Luis	Gómez	Ruiz	-	-	-	-	-	true	false
3	María	Torres	Jiménez	Calle 5	567	B	Centro	CDMX	false	true

Si se descompone la relación en

**Cliente** (IdCliente<sup>PK</sup>, NombreCliente, APaternoCliente, AMaternoCliente, esFisico, esVirtual)

**ClienteDomicilio** (IdCliente<sup>FK</sup>, Calle, NumeroExterior, NumeroInterior, Colonia, Estado)

Podemos ver que esto en base a la tabla mostrada anteriormente se reflejaría como,

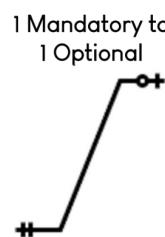
#### Cliente

<u>IdCliente</u> <sup>PK</sup>	NombreCliente	APaternoCliente	AMaternoCliente	esFisico	esVirtual
1	Ana	Pérez	López	true	true
2	Luis	Gómez	Ruiz	true	false
3	María	Torres	Jiménez	false	true

#### ClienteDomicilio

<u>IdCliente</u> <sup>FK</sup>	Calle	NumeroExterior	NumeroInterior	Colonia	Estado
1	Av. Central	123	A	Roma	CDMX
3	Calle 5	567	B	Centro	CDMX

Ahora, para los cambios en el **Modelo Relacional** tenemos que cada uno tendrá su tabla, y en cuanto a su **cardinalidad en el modelo relacional** será **1 Mandatory to 1 Optional**, del lado de **Cliente** es **1 Mandatory** pues cada cliente debe existir y este puede o no tener domicilio (esto depende de si desea comprar en linea) y del lado **ClienteDomicilio** es **1 Optional** pues un domicilio puede o no estar asociado a un cliente.



En el **Modelo Relacional** teníamos la **relación**

**PersonalOrganizador** (RFC<sup>PK</sup>, NombrePersonalOrganizador, APaternoPersonalOrganizador, AMaternoPersonalOrganizador, Calle, NúmeroExterior, NúmeroInterior, Colonia, Estado, Salario, esSeguridad, esLimpieza, esMédico)

El esquema ya cumple con la **Primera Forma Normal**, ya que **todos los atributos multivaluados han sido removidos**, permitiendo únicamente valores atómicos y posiblemente nulos. Pues recordemos que **PersonalOrganizador** en el **Modelo Entidad - Relación** este cuenta con los **atributos multivaluados Horario, Telefono y Correo**

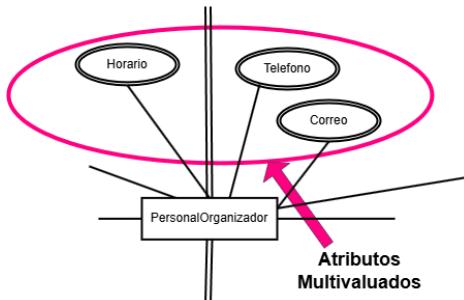


Figure 3.4: Enfocando los atributos multivaluados de la Entidad PersonalOrganizador en el **Modelo E-R**

y al traducirlo a **Modelo Relacional** cada uno **se convierte en tabla** con una columna que corresponde a la **Llave primaria** de PersonalOrganizador

PersonalOrganizador	
PK	RFC: char(13)
NombrePersonalOrganizador:	varchar(100)
APaternoPersonalOrganizador:	varchar(100)
AMaternoPersonalOrganizador:	varchar(100)
Calle:	varchar(100)
NúmeroExterior:	varchar(50)
NúmeroInterior:	varchar(50)
Colonia:	varchar(100)
Estado:	varchar(100)
Salario:	money
esSeguridad:	bool
esLimpieza:	bool
esMédico:	bool

Figure 3.5: Relación PersonalOrganizador en Modelo Relacional

CorreoPersonalOrganizador	
PK1, FK	RFC: char(13)
PK2	CorreoPersonalOrganizador: varchar(50)

Figure 3.6: Relación CorreoPersonalOrganizador en Modelo Relacional

HorarioPersonalOrganizador	
PK1, FK	RFC: char(13)
PK2	Horario: varchar(30)

Figure 3.7: Relación HorarioPersonalOrganizador en Modelo Relacional

TelefonoPersonalOrganizador	
PK1, FK	RFC: char(13)
PK2	TeléfonoPersonalOrganizador: char(10)

Figure 3.8: Relación TelefonoPersonalOrganizador en Modelo Relacional

se esperaría que al buscar al **personal organizador N** debería solo regresar uno, es decir, al que corresponde el **personal organizador N**. También esperaríamos que a partir del **RFC** pudieramos obtener el NombrePersonalOrganizador, APaternoPersonalOrganizador, AMaternoPersonalOrganizador, Calle, NúmeroExterior, NúmeroInterior, Colonia, Estado, Salario, esSeguridad, esLimpieza, esMédico. Por lo tanto, todos sus atributos **dependen funcionalmente** de este identificador y esto nos diría que se cumple la **Segunda Forma Normal**

<u>RFC</u>	→ NombrePersonalOrganizador, APaternoPersonalOrganizador, AMaternoPersonalOrganizador, Calle, NúmeroExterior, NúmeroInterior, Colonia, Estado, Salario, esSeguridad, esLimpieza, esMédico
------------	---

Pero nuestros atributos booleanos **esSeguridad, esLimpieza, esMédico** pueden provocar **redundancia y/o anomalías** si llegarán a ocurrir casos como que se quieran modificar los roles o si quisieran agregar otro tipo

de personal Organizador. Esto se puede visualizar de una mejor manera con la siguiente tabla

### PersonalOrganizador

RFC	NombrePersonalOrganizador	APaternoPersonalOrganizador	AMaternoPersonalOrganizador	Calle	NúmeroExterior	NúmeroInterior	Colonia	Estado	Salario	esSeguridad	esLimpieza	esMédico
MAHS991127643	Sofia	Martinez	Hernandez	Av. Central	123	A	Roma	CDMX	5000	false	true	false
PALH800229XYZ	Hugo	Pacoli	Luisan	Calle 5	567	B	Centro	CDMX	6000	true	false	false
MELM8305281H0	Monica	Mendez	Luna	Av. Central	123	C	Roma	CDMX	8000	false	false	true

Entonces eliminaremos la dependencia transitiva, dejando así,

**PersonalOrganizador** (RFC<sup>PK</sup>, NombrePersonalOrganizador, APaternoPersonalOrganizador, AMaternoPersonalOrganizador, Calle, NúmeroExterior, NúmeroInterior, Colonia, Estado, Salario)

**RolPersonalOrganizador** (RFC<sup>FK</sup>, Rol)

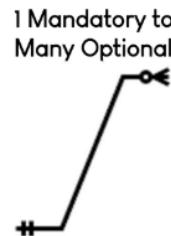
### PersonalOrganizador

RFC	NombrePersonalOrganizador	APaternoPersonalOrganizador	AMaternoPersonalOrganizador	Calle	NúmeroExterior	NúmeroInterior	Colonia	Estado	Salario
MAHS991127643	Sofia	Martinez	Hernandez	Av. Central	123	A	Roma	CDMX	5000
PALH800229XYZ	Hugo	Pacoli	Luisan	Calle 5	567	B	Centro	CDMX	6000
MELM8305281H0	Monica	Mendez	Luna	Av. Central	123	C	Roma	CDMX	8000

### RolPersonalOrganizador

RFC	Rol
MAHS991127643	limpieza
PALH800229XYZ	seguridad
MELM8305281H0	medico

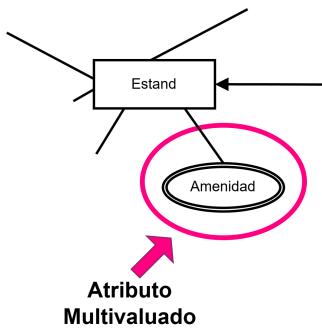
Cada uno tendrá su tabla, y en cuanto a su **cardinalidad en el modelo relacional** será **1 Mandatory to many Optional**, del lado de **PersonalOrganizador** es **1 Mandatory** pues debe tener al menos un Rol obligatoriamente y del lado **RolPersonalOrganizador** es **many Optional** puede estar asignado a uno o más **PersonalOrganizador**, pero no está obligado.



En el **Modelo Relacional** teníamos la **relación**

**Estand**<sup>PK</sup> (Numero, Paquete, Precio)

El esquema ya cumple con la **Primera Forma Normal**, ya que **todos los atributos multivaluados han sido removidos**, permitiendo únicamente valores atómicos y posiblemente nulos. Pues recordemos que **Estand** en el **Modelo Entidad - Relación** este cuenta con un **atributo multivaluado Amenidad**

Figure 3.9: Enfocando el atributo multivalorado **Amenidad** de la Entidad **Estand** en el **Modelo E-R**

y al traducirlo a **Modelo Relacional** se convierte en tabla con una columna que corresponde a la **llave primaria** de **Estand**

Estand	
PK	NúmeroEstand: int
	Paquete: char(1)
	Precio: money

AmenidadEstand	
PK1,FK	NúmeroEstand: int
PK2	AmenidadEstand: varchar(100)

Figure 3.10: Relación Estand en Modelo Relacional

Figure 3.11: Relación AmenidadEstand en Modelo Relacional

se esperaría que al buscar al **estand N** debería solo regresar uno, es decir, al que corresponde el **estand N**. También esperaríamos que a partir del **NumeroEstand** pudieramos obtener el Paquete, Precio. Esto, en principio, nos estaría indicando que se cumple la **Segunda Forma Normal**

$$\text{NúmeroEstand} \rightarrow \text{Paquete, Precio}$$

Pero en realidad el precio depende del paquete escogido y del precio base del Estand, lo que genera una dependencia transitiva,

$$\begin{array}{l} \text{NúmeroEstand} \rightarrow \text{Paquete} \\ \text{Paquete} \rightarrow \text{Precio} \end{array}$$

y esto además de crear redundancia nos daría problemas si queremos actualizar. Y no es el único problema pues estamos considerando que las amenidades dependen del Estand y en realidad dependen del Paquete

**AmenidadEstand** (Número, AmenidadEstand)  
PK

Si se descompone la relación en

**Paquete** (IdPaquete<sup>PK</sup>, Paquete)

Con IdPaquete como llave primaria y Paquete como **varchar(20)**

**Estand** (NumeroEstand<sup>PK</sup>, PrecioBaseEstand, IdPaquete<sup>FK</sup>)

Manteniendo a IdPaquete como llave primaria y PrecioBaseEstand como **money** pero se le agrega una llave foranea IdPaquete y que es **int**.

<u>PaqueteAmenidad</u> ( <u>IdPaquete</u> , <u>AmenidadPaquete</u> )	
	PK

Ahora su llave primaria compuesta es IdPaquete, AmenidadPaquete pero también se considera como una llave foranea a IdPaquete.

Y para que se vea el objetivo de la normalización, será a través de tablas

#### Paquete

<u>IdPaquete</u> <sup>PK</sup>	Paquete
1	Basico
2	Premium
3	Emprendedor

Estos serán todos nuestros registros en esta tabla pues hasta el momento solo son los que nos solicitan.

#### PaqueteAmenidades

<u>IdPaquete</u> <sup>PK</sup>	<u>AmenidadPaquete</u> <sup>PK</sup>
1	1 mesa
1	2 sillas
2	2 mesas
2	4 sillas
3	3 mesas
3	6 sillas
3	Pantalla tactil
3	Toma de corriente

Estos serán todos nuestros registros en esta tabla pues hasta el momento solo son los que nos solicitan.

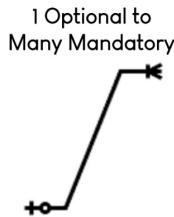
Y de esta forma logramos quitar la redundancia, pues así estand puede tener la información del paquete sin la necesidad de tener la misma información de paquetes en diferentes tuplas de la tabla (con esto también podrá calcular el precio final) como se verá a continuación

#### Estand

<u>NumeroEstand</u> <sup>PK</sup>	<u>IdPaquete</u> <sup>FK</sup>	PrecioBaseEstand
1	2	350
2	3	400
3	2	390
4	1	300
5	3	500
6	1	200

Cada uno tendrá su tabla, donde se verán reflejados los cambios descritos anteriormente, y en cuanto a su **cardinalidad en el modelo relacional** será **1 Optional to many mandatory**, del lado de Paquete es **1 Optional** pues

cada paquete puede ser asignado a varios estands y **Estand** es **many Mandatory** pues cada Estand debe tener un único paquete asignado. Y es lo mismo para Paquete y PaqueteAmenidad, la **cardinalidad en el modelo relacional** será **1 Optional to many mandatory**, del lado de **Paquete** es **1 Optional** pues cada paquete cuenta con varias amenidades.



En el **Modelo Relacional** teníamos la **relación**

**Negocio** (IdNegocio<sup>PK</sup>, Numero<sup>FK</sup>, NombreNegocio, Descripción, PrecioMínimo, PrecioMáximo)

El esquema ya cumple con la **Primera Forma Normal**, ya que **todos los atributos multivaluados han sido removidos**, permitiendo únicamente valores atómicos y posiblemente nulos. Pues recordemos que **Negocio** en el **Modelo Entidad - Relación** este cuenta con los **atributos multivaluados RedSocial, Telefono y Correo**

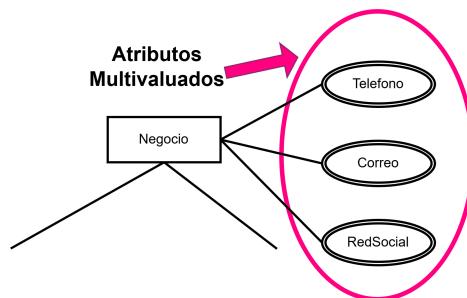


Figure 3.12: Enfocando los atributos multivaluados de la Entidad Negocio en el **Modelo E-R**

y al traducirlo a **Modelo Relacional** cada uno se convierte en **tabla** con una columna que corresponde a la **Llave primaria** de Negocio

Negocio	
PK	<u>IdNegocio</u> : serial4
FK	<u>Numero</u> : int
	NombreNegocio: varchar(100)
	Descripción: text
	PrecioMínimo: numeric(10,2)
	PrecioMáximo: numeric(10,2)

Figure 3.13: Relación Negocio en Modelo Relacional

CorreoNegocio	
PK1, FK	<u>IdNegocio</u> : int
PK2	<u>Correo</u> : varchar(50)

Figure 3.14: Relación CorreoNegocio en Modelo Relacional

RedSocialNegocio	
PK1, FK	<u>IdNegocio</u> : int
PK2	<u>RedSocial</u> : varchar(50)

Figure 3.15: Relación RedSocialNegocio en Modelo Relacional

TelefonoNegocio	
PK1, FK	<u>IdNegocio</u> : int
PK2	<u>Teléfono</u> : char(10)

Figure 3.16: Relación TelefonoNegocio en Modelo Relacional

se esperaría que al buscar el **negocio**  $N$  debería solo regresar uno, es decir, al que corresponde el **negocio**  $N$ . También esperaríamos que a partir del **IdNegocio** pudieramos obtener el **Numero**, NombreNegocio, Descripción, PrecioMínimo, PrecioMáximo. Esto, en principio, nos estaría indicando que se cumple la **Segunda Forma Normal**

<u><b>IdNegocio</b></u>	→	<u><b>Numero</b></u> , NombreNegocio, Descripción, PrecioMínimo, PrecioMáximo
-------------------------	---	---

Pero tenemos que tanto **PrecioMínimo** como **PrecioMáximo** lo estamos manejando como que dependen del **negocio** pero en realidad dependen de los productos y servicios que se ofrecen. Entonces para eliminar la dependencia transitiva, seguimos la regla de **descomposición**, obteniendo así,

**Negocio** (**IdNegocio**<sup>PK</sup>, **Numero**<sup>FK</sup>, NombreNegocio, Descripción)

**RangoPrecioNegocio** (**IdNegocio**<sup>PK</sup>, PrecioMinimo, PrecioMaximo)

Cada uno tendrá su tabla, y en cuanto a su **cardinalidad en el modelo relacional** será **1 Mandatory to 1 Optional** del lado **RangoPrecioNegocio** tiene que ser **1 Mandatory** pues cada RangoPrecioNegocio debe pertenecer a un único negocio, lo que lo hace mandatorio en ese lado y del lado de **Negocio** es **1 Mandatory** pues cada negocio puede tener un rango de precios pero no es obligatorio.

Ahora, veremos dos casos pues prácticamente es la misma situación y la misma solución. En el **Modelo Relacional** teníamos la **relación**

**RegistrarServicio** (**IdTicket**<sup>FK</sup>, **IdNegocio**<sup>FK</sup>, **IdServicio**<sup>FK</sup>, Duración)

**RegistrarProducto** (**IdTicket**<sup>FK</sup>, **IdNegocio**<sup>FK</sup>, **IdProducto**<sup>FK</sup>, Cantidad)

Podemos ver que los registros dependen de las llaves **IdTicket**, **IdNegocio**, **IdProducto**, en el caso de **RegistrarProducto**, y **IdTicket**, **IdNegocio**, **IdServicio**, en el caso de **RegistrarServicio** lo cual puede generar errores si alguno falta, pero sabemos que de entonces al eliminar dependencias transitivas tenemos que,

**RegistrarServicio** (**IdTicket**<sup>FK</sup>, **IdServicio**<sup>FK</sup>, Duración)

**RegistrarProducto** (**IdTicket**<sup>FK</sup>, **IdProducto**<sup>FK</sup>, Cantidad)

Por último, en el **Modelo Relacional** teníamos la **relación**

**Tarjeta** (**NumeroTarjeta**<sup>PK</sup>, **IdCliente**<sup>FK</sup>, Vencimiento, CVV)

<u><b>NumeroTarjeta</b></u>	→	<u><b>IdCliente</b></u> , Vencimiento, CVV
-----------------------------	---	--

se esperaría que al buscar la **tarjeta**  $N$  debería solo regresar uno, es decir, al que corresponde a la **tarjeta**  $N$ . También esperaríamos que a partir del **NumeroTarjeta** pudieramos obtener **IdCliente**, Vencimiento, CVV. Pero esto puede generar problemas de redundancia en casos donde un cliente tiene varias tarjetas, por lo que

**Tarjeta** (**NumeroTarjeta**<sup>PK</sup>, **IdCliente**<sup>FK</sup>)

**TarjetaInformacion** (**NumeroTarjeta**<sup>PK</sup>, Vencimiento, CVV)

Cada una tendrá su tabla y en cuanto a la **cardinalidad en el modelo relacional** tenemos que cada **NumeroTarjeta** en Tarjeta debe tener exactamente un registro en **TarjetaInformacion** y que cada entrada en **TarjetaInformacion**

debe corresponder a una tarjeta registrada en Tarjeta. Además esta normalización ayuda a la integridad de datos.

## Modelo Relacional Normalizado

A continuación, se mostrará las relaciones del Modelo Relacional con las actualizaciones que hubieron al realizar la normalización.

Producto	
PK1,FK	<u>IdNegocio</u> : int
PK2	<u>idProducto</u> : serial4
	NombreProducto:varchar(100)
	Descripción: text
	Tipo: varchar(50)
	Precio: money
	Presentación: varchar(100)
	Stock: int

Ticket	
PK	<u>IdTicket</u> : serial4
FK1	<u>IdCliente</u> : int
FK2	<u>idBazar</u> : int
FK3	<u>IdNegocio</u> : int
FK4	<u>RFC</u> : char(13)
	ComisiónBazar: numeric(10,2)

Figure 3.17: Relación Producto en Modelo Relacional

Figure 3.18: Relación Ticket en Modelo Relacional

RedSocialNegocio	
PK1, FK	<u>IdNegocio</u> , <u>RedSocial</u>
PK2	<u>RedSocial</u> : varchar(50)

Figure 3.19: Relación RedSocialNegocio en Modelo Relacional

CorreoNegocio	
PK1, FK	<u>IdNegocio</u> : int
PK2	<u>Correo</u> : varchar(50)

Figure 3.20: Relación CorreoNegocio en Modelo Relacional

TelefonoNegocio	
PK1, FK	<u>IdNegocio</u> , <u>Teléfono</u>
PK2	<u>Teléfono</u> : char(10)

Figure 3.21: Relación TelefonoNegocio en Modelo Relacional

Trabajar	
FK1	<u>idBazar</u> : int
FK2	<u>RFC</u> : char(13)
	FechaAsistenciaTrabajar: date

Figure 3.22: Relación Trabajar en Modelo Relacional

**Agendar** ([idBazar](#)<sup>FK</sup>, [IdNegocio](#)<sup>FK</sup>, FechaAsistencia)

Agendar	
<b>FK1</b>	<u><a href="#">idBazar</a></u> : int
<b>FK2</b>	<u><a href="#">IdNegocio</a></u> : int
	FechaAsistencia: date

Figure 3.23: Relación Agendar en Modelo Relacional

**MetodoPago** ([IdCliente](#), [MetodoPago](#))  
PK

MetodoPago	
<b>PK1,FK</b>	<u><a href="#">IdCliente</a></u> : int
<b>PK2</b>	<u><a href="#">MetodoPago</a></u> : varchar(20)

Figure 3.24: Relación MetodoPago en Modelo Relacional

**Emprendedor** ([RFC](#)<sup>PK</sup>, [IdNegocio](#)<sup>FK</sup>, NombreEmprendedor, APaternoEmprendedor, AMaternoEmprendedor, Calle, NúmeroExterior, NúmeroInterior, Colonia, Estado, Género, FechaNacimiento)

**Bazar**([IdBazar](#)<sup>PK</sup>, NombreBazar, Calle, NumeroInterior, NumeroExterior, Colonia, Estado, Modalidad, FechaInicio, FechaFin)

Emprendedor	
<b>PK</b>	<u><a href="#">RFC</a></u> : char(13)
<b>FK</b>	<u><a href="#">idNegocio</a></u> : serial4
	NombreEmprendedor: varchar(100)
	APaternoEmprendedor: varchar(100)
	AMaternoEmprendedor: varchar(100)
	Calle: varchar(100)
	NúmeroExterior: varchar(50)
	NúmeroInterior: varchar(50)
	Colonia: varchar(100)
	Estado: varchar(100)
	Género: char(1)
	FechaNacimiento: date

Figure 3.25: Relación Emprendedor en Modelo Relacional

Bazar	
<b>PK</b>	<u><a href="#">IdBazar</a></u> : serial4
	NombreBazar: varchar(100)
	Calle: varchar(100)
	NúmeroInterior: varchar(50)
	NúmeroExterior: varchar(50)
	Colonia: varchar(100)
	Estado: varchar(100)
	Modalidad: varchar(80)
	FechaInicio: date
	FechaFin: date

Figure 3.26: Relación Bazar en Modelo Relacional

**TelefonoEmprendedor** ([RFC](#), [TelefonoEmprendedor](#))  
PK

TeléfonoEmprendedor	
<b>PK1, FK</b>	<u><a href="#">RFC</a></u> : char(13)
<b>PK2</b>	<u><a href="#">TelefonoEmprendedor</a></u> : char(10)

Figure 3.27: Relación TelefonoEmprendedor en Modelo Relacional

**CorreoEmprendedor** ([RFC](#), [CorreoEmprendedor](#))  
PK

CorreoEmprendedor	
<b>PK1,FK</b>	<u><a href="#">RFC</a></u> : char(13)
<b>PK2</b>	<u><a href="#">CorreoEmprendedor</a></u> : varchar(50)

Figure 3.28: Relación CorreoEmprendedor en Modelo Relacional

**TelefonoPersonalOrganizador** (RFC, TeléfonoPersonalOrganizador)  
PK

**HorarioPersonalOrganizador** (RFC, HorarioPersonalOrganizador)  
PK

TelefonoPersonalOrganizador	
PK1, FK	<u>RFC: char(13)</u>
PK2	<u>TeléfonoPersonalOrganizador: char(10)</u>

Figure 3.29: Relación TelefonoPersonalOrganizador en Modelo Relacional

HorarioPersonalOrganizador	
PK1, FK	<u>RFC: char(13)</u>
PK2	<u>Horario: varchar(30)</u>

Figure 3.30: Relación HorarioPersonalOrganizador en Modelo Relacional

**CorreoPersonalOrganizador** (RFC, CorreoPersonalOrganizador)  
PK

**AmenidadBazar** (IdBazar, AmenidadBazar)  
PK

CorreoPersonalOrganizador	
PK1, FK	<u>RFC: char(13)</u>
PK2	<u>CorreoPersonalOrganizador: varchar(50)</u>

Figure 3.31: Relación CorreoPersonalOrganizador en Modelo Relacional

AmenidadBazar	
PK1, FK	<u>idBazar: int</u>
PK2	<u>AmenidadBazar: varchar(100)</u>

Figure 3.32: Relación AmenidadBazar en Modelo Relacional

**Perecedero** (IdNegocio, IdProducto, FechaPreparacion, FechaCaducidad)  
PK

**Servicio** (IdNegocio, IdServicio, NombreServicio, Descripcion, Tipo, PrecioServicio, Duración)  
PK

Perecedero	
PK1,FK1	<u>IdNegocio: int</u>
PK2,FK2	<u>IdProducto: int</u>
	FechaPreparacion: date
	FechaCaducidad: date

Figure 3.33: Relación Perecedero en Modelo Relacional

Servicio	
PK1,FK	<u>IdNegocio: int</u>
PK2	<u>idServicio: serial4</u>
	NombreServicio:varchar(100)
	Descripcion: text
	Tipo: varchar(50)
	Precio: money
	Duración: time

Figure 3.34: Relación Servicio en Modelo Relacional

## Actualización

**Negocio** (IdNegocio<sup>PK</sup>, Numero<sup>FK</sup>, NombreNegocio, Descripción)

**RangoPrecioNegocio** (IdNegocio<sup>PK</sup>, PrecioMínimo, PrecioMáximo)

Negocio		RangoPrecioNegocio	
PK	<u>IdNegocio: serial4</u>	PK	<u>IdNegocio: serial4</u>
FK	<u>Numero: int</u>		PrecioMinimo: numeric(10,2)
	NombreNegocio: varchar(100)		PrecioMaximo: numeric(10,2)
	Descripción: text		

Figure 3.35: Relación Negocio en Modelo Relacional

Figure 3.36: Relación RangoPrecioNegocio en Modelo Relacional

**PersonalOrganizador** (RFC<sup>PK</sup>, NombrePersonalOrganizador, APaternoPersonalOrganizador, AMaternoPersonalOrganizador, Calle, NúmeroExterior, NúmeroInterior, Colonia, Estado, Salario)

**RolPersonalOrganizador** (RFC<sup>FK</sup>, Rol)

PersonalOrganizador		RolPersonalOrganizador	
PK	<u>RFC: char(13)</u>	FK	<u>RFC: char(13)</u>
	NombrePersonalOrganizador: varchar(100) APaternoPersonalOrganizador: varchar(100) AMaternoPersonalOrganizador: varchar(100) Calle: varchar(100) NúmeroExterior: varchar(50) NúmeroInterior: varchar(50) Colonia: varchar(100) Estado: varchar(100) Salario: money		Rol: varchar(10)

Figure 3.37: Relación PersonalOrganizador en Modelo Relacional

Figure 3.38: Relación RolPersonalOrganizador en Modelo Relacional

**Paquete** (IdPaquete<sup>PK</sup>, Paquete)

**Estand** (NumeroEstand<sup>PK</sup>, PrecioBaseEstand, IdPaquete<sup>FK</sup>)

**PaqueteAmenidad** (IdPaquete, AmenidadPaquete)

Paquete	
PK	<u>IdPaquete: int</u>
	Paquete: char(20)

Figure 3.39: Relación Paquete en Modelo Relacional

PaqueteAmenidad	
PK1, FK	<u>IdPaquete: int</u>
PK2	<u>AmenidadPaquete varchar(100)</u>

Figure 3.40: Relación PaqueteAmenidad en Modelo Relacional

Estand	
PK	<u>NumeroEstand: int</u>
FK	<u>IdPaquete: int</u>
	PrecioBaseEstand: money

Figure 3.41: Relación Estand en Modelo Relacional

**Cliente** (IdCliente<sup>PK</sup>, NombreCliente, APaternoCliente, AMaternoCliente, esFisico, esVirtual)

**ClienteDomicilio** (IdCliente<sup>FK</sup>, Calle, NumeroExterior, NumeroInterior, Colonia, Estado)

Cliente	
PK	<u>IdCliente: serial4</u>
	NombreCliente: varchar(100)
	APaternoCliente: varchar(100)
	AMaternoCliente: varchar(100)
	EsFisico: bool
	EsVirtual: bool

Figure 3.42: Relación Cliente en Modelo Relacional

DomicilioCliente	
FK	<u>IdCliente: serial4</u>
	Calle: varchar(100)
	NúmeroExterior: varchar(50)
	NúmeroInterior: varchar(50)
	Colonia: varchar(100)
	Estado: varchar(100)

Figure 3.43: Relación ClienteDomicilio en Modelo Relacional

**Tarjeta** (NumeroTarjeta<sup>PK</sup>, IdCliente<sup>FK</sup>)

**InformacionTarjeta** (NumeroTarjeta<sup>PK</sup>, Vencimiento, CVV)

Tarjeta	
PK	<u>NumeroTarjeta: char(16)</u>
FK1	<u>IdCliente: int</u>

Figure 3.44: Relación Tarjeta en Modelo Relacional

InformacionTarjeta	
PK	<u>NumeroTarjeta: char(16)</u>
	Vencimiento: date
	CVV: varchar(4)

Figure 3.45: Relación Tarjeta en Modelo Relacional

**RegistrarServicio** (IdTicket<sup>FK</sup>, IdServicio<sup>FK</sup>, Duración)

**RegistrarProducto** (IdTicket<sup>FK</sup>, IdProducto<sup>FK</sup>, Cantidad)

RegistrarServicio	
FK1	<u>IdTicket: int</u>
FK2	<u>IdServicio: int</u>
	Duración: time

Figure 3.46: Relación RegistrarServicio en Modelo Relacional

RegistrarProducto	
FK1	<u>IdTicket: int</u>
FK2	<u>IdProducto: int</u>
	Cantidad: int

Figure 3.47: Relación RegistrarProducto en Modelo Relacional

# 4 Lenguaje para Definición de Datos

Un esquema de base de datos se especifica mediante un conjunto de definiciones expresadas mediante un lenguaje especial llamado **lenguaje de definición de datos (DDL)**.

Se debe especificar el almacenamiento y los métodos de acceso usados por el sistema de bases de datos por un conjunto de instrucciones en un tipo especial de DDL denominado **lenguaje de almacenamiento y definición de datos**. Estas instrucciones definen los detalles de implementación de los esquemas de base de datos, que se ocultan usualmente a los usuarios.

## Políticas de mantenimiento para las llaves foráneas

A continuación, presentaremos nuestra investigación sobre cuáles son las políticas de mantenimiento para las llaves foráneas. Misma que aplicaremos en nuestro *script DDL.sql*

### 1. ¿Qué es una política de mantenimiento de llaves foráneas?

Una política de mantenimiento de llaves foráneas define el comportamiento de la base de datos cuando se elimina o actualiza un registro de una tabla que está siendo referenciado por otra. Estas políticas permiten mantener la integridad referencial entre tablas, evitando inconsistencias y errores en los datos [1].

### 2. ¿Cómo se indica en SQL cada política?

En PostgreSQL las políticas de mantenimiento se especifican en la definición de las llaves foráneas usando las sentencias:

```
FOREIGN KEY (columna_hija)
REFERENCES tabla_padre (columna_padre)
ON DELETE accion
ON UPDATE accion;
```

Listing 4.1: Definición de llave foránea en PostgreSQL

Las acciones posibles son:

- CASCADE
- SET NULL
- SET DEFAULT
- RESTRICT
- NO ACTION

### 3. ¿Cuál es el objeto y funcionamiento de cada política?

- **CASCADE:** Cuando se elimina o actualiza un registro padre, automáticamente se eliminan o actualizan los registros hijos [3].
- **SET NULL:** Al eliminar o actualizar un registro padre, los campos de la llave foránea en los hijos se establecen en NULL.
- **SET DEFAULT:** Similar a SET NULL, pero asigna el valor por defecto de la columna.

- **RESTRICT**: Impide la eliminación o actualización del registro padre si existen registros hijos relacionados.
- **NO ACTION**: Igual que RESTRICT, pero la comprobación ocurre al final de la transacción.

#### 4. Ventajas y desventajas de cada política

##### ○ **CASCADE**

- Ventaja: Automatiza mantenimiento y evita registros huérfanos.
- Desventaja: Riesgo de eliminar muchos registros sin darse cuenta.

##### ○ **SET NULL**

- Ventaja: Permite conservar registros hijos.
- Desventaja: Puede dejar registros incompletos.

##### ○ **SET DEFAULT**

- Ventaja: Útil si existe un valor por defecto representativo.
- Desventaja: Requiere que la columna tenga un valor default definido.

##### ○ **RESTRICT**

- Ventaja: Protege los datos de ser eliminados por accidente.
- Desventaja: Requiere eliminar o actualizar primero las filas hijas.

##### ○ **NO ACTION**

- Ventaja: Permite trabajar con transacciones complejas.
- Desventaja: Es más complicado de entender o predecir su efecto.

#### 5. Política seleccionada y justificación

Para el esquema que estamos trabajando, se utilizarán dos políticas de mantenimiento de llaves foráneas: **ON DELETE CASCADE** y **ON UPDATE CASCADE**.

La política **ON DELETE CASCADE** se utilizará porque resulta conveniente que, al eliminar un registro de la tabla padre (por ejemplo, un **Bazar**), también se eliminen automáticamente todas las filas relacionadas en las tablas hijas (como **AmenidadBazar**). Esto evita que existan registros huérfanos y simplifica el mantenimiento de la base de datos, asegurando que los datos mantengan coherencia e integridad.

Por otro lado, también se empleará la política **ON UPDATE CASCADE**, ya que de esta manera, si por alguna razón se modifica el valor del identificador principal (**IdBazar**) en la tabla padre, este cambio se propagará automáticamente a las tablas hijas que dependan de dicho identificador. Aunque no es común actualizar los valores de las llaves primarias, esta política permite mantener la integridad referencial de forma automática en caso de que esta situación ocurra.

En resumen, ambas políticas en conjunto permiten:

- Facilitar el mantenimiento de los datos.
- Evitar registros huérfanos.
- Asegurar que los cambios en las llaves primarias no generen inconsistencias.

Por estas razones, consideramos que `ON DELETE CASCADE` y `ON UPDATE CASCADE` son las políticas más adecuadas para nuestro esquema en PostgreSQL.

## Creación

Parte de esto se trabaja en la *Práctica 05* misma que se centra en la definición del esquema utilizando DDL (Lenguaje para Definición de Datos), asegurando que cada tabla contenga los atributos correctos con restricciones adecuadas para mantener la integridad de los datos. Y por otra parte en la *Práctica 06* la cual aborda el mantenimiento de llaves foráneas, especificando las políticas de actualización y eliminación (CASCADE, SET NULL, etc.), además del uso del comando COMMENT para mejorar la documentación del esquema.

A continuación, presentaremos algunos fragmentos de código de nuestro script **DDL.sql** con su justificación, que a nuestra consideración sirve para ejemplificar nuestro proceso.

### Crear Tablas

**schema\_name:** Es el nombre del esquema al que pertenece la nueva tabla. Si no se especifica un esquema, los objetos creados se agregan a el esquema public.

Con el siguiente código se asegura de que no haya conflictos con un esquema anterior y creamos el esquema en PostgreSQL para agrupar todas las tablas de la base de datos.

```
DROP schema      if exists GatitaEmprendedora cascade;
CREATE SCHEMA GatitaEmprendedora;
```

**table\_name:** Es el nombre de la nueva tabla. Los nombre de tabla deben seguir las convenciones de nombrado para identificadores.

Con el siguiente código se ve como creamos la tabla Bazar

```
CREATE TABLE GatitaEmprendedora.Bazar (
    IdBazar SERIAL,
    NombreBazar VARCHAR(100),
    Calle VARCHAR(100),
    NumeroInterior VARCHAR(50),
    NumeroExterior VARCHAR(50),
    Colonia VARCHAR(100),
    Estado VARCHAR(100),
    Modalidad VARCHAR(80),
    FechaInicio DATE,
    FechaFin DATE
);
```

### Alter table

El comando **ALTER TABLE**, nos permite realizar modificaciones a nuestras tablas, tales como: añadir nuevas columnas, renombrar la estructura o cambiar el tipo de dato. Lo utilizaremos para manejar la integridad de datos. Esto es con la finalidad de mantener una estructura de nuestras tablas con los atributos y el tipo de atributo en la definición de cada tabla, y las restricciones aparte utilizando el comando.

## Restricciones de Dominio:

**NOT NULL:** La restricción NOT NULL especifica que la columna no acepta valores NULL. Un valor NULL no es lo mismo que cero (0), en blanco o que una cadena de caracteres de longitud cero. NULL, significa que no hay ninguna entrada. La presencia de un valor NULL suele implicar que el valor es desconocido o no está definido. Se recomienda evitar la aceptación de valores NULL, dado que pueden implicar una mayor complejidad en las consultas y actualizaciones.

Por otra parte hay opciones para las columnas como las restricciones PRIMARY KEY, que no pueden utilizarse con columnas que aceptan valores NULL.

```
-- Hacer columnas NOT NULL
ALTER TABLE GatitaEmprendedora.Cliente
ALTER COLUMN NombreCliente SET NOT NULL;

ALTER TABLE GatitaEmprendedora.Cliente
ALTER COLUMN APaternoCliente SET NOT NULL;

ALTER TABLE GatitaEmprendedora.Cliente
ALTER COLUMN AMaternoCliente SET NOT NULL;
```

**CHECK:** La restricción CHECK exige la integridad del dominio mediante la limitación de los valores que se pueden asignar a una columna. Una restricción CHECK especifica una condición de búsqueda booleana (se evalúa como TRUE o FALSE) que se aplica a todos los valores que se indican en la columna. Se rechazan todo los valores que se evalúan como FALSE. En una misma columna se pueden especificar varias restricciones CHECK. Las condiciones de búsqueda booleanas son las siguientes: =, <>, >, <, <=, >=, between, in

```
-- Restriccion para verificar que FechaFin > FechaInicio
ALTER TABLE GatitaEmprendedora.Bazar
ADD CONSTRAINT chk_fechas_bazar CHECK (FechaFin >= FechaInicio);

-- Restriccion para que al menos uno de EsFisico o EsVirtual sea TRUE
ALTER TABLE GatitaEmprendedora.Cliente
ADD CONSTRAINT chk_tipo_cliente CHECK (
    (EsFisico IS TRUE) OR (EsVirtual IS TRUE)
);
```

## Restricciones de integridad de entidad

**PRIMARY KEY:** La restricción PRIMARY KEY identifica la columna o el conjunto de columnas cuyos valores identifican de forma exclusiva a cada una de las filas de una tabla. Dos filas de la tabla no pueden tener el mismo valor de llave primaria. No se pueden asignar valores NULL a ninguna de las columnas de una llave primaria. Todas las tablas tienen que tener una llave primaria.

```
ALTER TABLE GatitaEmprendedora.Bazar
ADD CONSTRAINT pk_Bazar PRIMARY KEY (IdBazar);
```

## Restricciones de integridad referencial

**FOREIGN KEY:** La restricción FOREIGN KEY identifica y exige las relaciones entre las tablas. Una llave foránea de una tabla apunta a una llave primaria de otra tabla. No se puede insertar una fila que tenga un valor de llave foránea (excepto NULL) si no hay una llave primaria con dicho valor.

En el código se puede observar como definimos una llave foránea con mantenimiento (pues aplicamos las políticas que investigamos, es decir, una política de eliminación y actualización en cascada para mantener consistencia cuando se borra/modifica un bazar)

```
ALTER TABLE GatitaEmprendedora.AmenidadBazar
ADD CONSTRAINT pk_AmenidadBazar PRIMARY KEY (IdBazar, AmenidadBazar);
ALTER TABLE GatitaEmprendedora.AmenidadBazar
ADD CONSTRAINT fk_AmenidadBazar FOREIGN KEY (IdBazar)
    REFERENCES GatitaEmprendedora.Bazar(IdBazar)
    ON DELETE CASCADE ON UPDATE CASCADE;
```

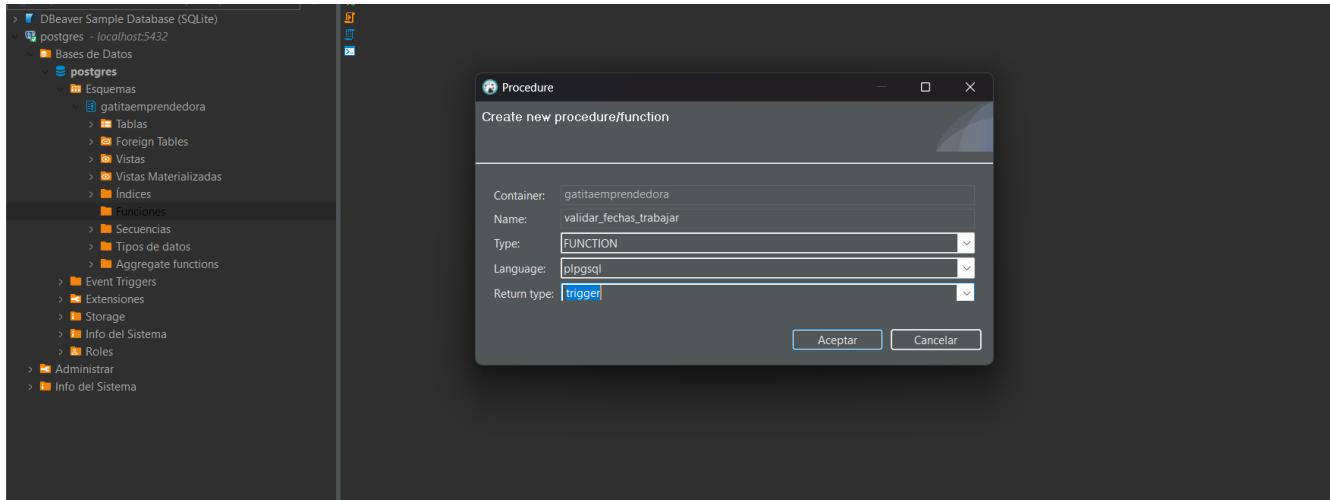
Se puede ver como se realizó la documentación para facilitar la comprensión del esquema y su propósito dentro del sistema.

```
COMMENT ON TABLE GatitaEmprendedora.Cliente IS 'Tabla que almacena la información de los
clientes que participan en el bazar, ya sea de manera física o virtual.';
COMMENT ON COLUMN GatitaEmprendedora.Cliente.IdCliente IS 'Identificador único y
consecutivo de cada cliente dentro del sistema。';
COMMENT ON CONSTRAINT pk_Cliente ON GatitaEmprendedora.Cliente IS 'Llave primaria de la
tabla Cliente que identifica de manera única a cada cliente registrado。';
```

# 5 Disparadores Funciones y SPs

## Instalación de Disparadores / Funciones SPs

- Crear una nueva función en el esquema GatitaEmprendedora, con las siguientes configuraciones:



- Copiar y pegar el código fuente del disparador / función / sp:

```

9 -- Uso:
10 -- Se usa como parte de un trigger BEFORE INSERT OR UPDATE en la tabla
11 -- GatitaEmprendedora.Trabajar.
12 --
13 -- Notas:
14 -- - Si la fecha de asistencia está fuera del rango, lanza una excepción
15 -- y bloquea la operación.
16 ====
17 CREATE OR REPLACE FUNCTION GatitaEmprendedora.validar_fechas_trabajar()
18 RETURNS TRIGGER AS $$
19 DECLARE
20     v_fechaInicio DATE;
21     v_fechaFin DATE;
22 BEGIN
23     -- Obtener las fechas de inicio y fin del bazar asociado
24     SELECT fechaInicio, fechaFin
25     INTO v_fechaInicio, v_fechaFin
26     FROM GatitaEmprendedora.Bazar
27     WHERE idBazar = NEW.idBazar;
28
29     -- Validar que la fecha de asistencia esté dentro del rango permitido
30     IF NEW.fechaAsistencia NOT BETWEEN v_fechaInicio AND v_fechaFin THEN
31         RAISE EXCEPTION 'La fecha de asistencia (%) está fuera del rango permitido: % - %',
32                         NEW.fechaAsistencia, v_fechaInicio, v_fechaFin;
33     END IF;
34
35     RETURN NEW;
36 END;
37 $$ LANGUAGE plpgsql;

```

### 3. Guardar y ejecutar disparador / función / sp:

```

1 -- =====
2 -- Función: GatitaEmprendedora.validar_fechas_trabajar
3 -- Autor: NAMEisNULL
4 -- Propósito:
5 -- Esta función valida que la fecha de asistencia esté dentro del rango de fechas del bazar correspondiente, si no, lanza una excepción
6 --
7 -- Uso:
8 -- Se usa como parte de un trigger en la tabla GatitaEmprendedora.Trabajar.
9 --
10 -- Notas:
11 -- - Si la fecha de asistencia es fuera del rango permitido, lanza una excepción y bloquea la operación.
12 --
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27 WHERE idBazar = NEW.idBazar;
28
29 -- Validar que la fecha de asistencia esté dentro del rango permitido
30 IF NEW.fechaAsistencia NOT BETWEEN v_fechaInicio AND v_fechaFin THEN
31   RAISE EXCEPTION 'La fecha de asistencia (%) está fuera del rango permitido: % - %',
32                     NEW.fechaAsistencia, v_fechaInicio, v_fechaFin;
33 END IF;
34
35 RETURN NEW;
36 END;
37 $$ LANGUAGE plpgsql;

```

### 4. Agregar como triger (si es el caso en que sea un triger) a la definición de la tabla.

```

DROP TRIGGER IF EXISTS trg_validar_fechas_trabajar ON GatitaEmprendedora.Trabajar;
CREATE TRIGGER trg_validar_fechas_trabajar
BEFORE INSERT OR UPDATE ON GatitaEmprendedora.Trabajar
FOR EACH ROW
EXECUTE FUNCTION GatitaEmprendedora.validar_fechas_trabajar();

```

# 6 Lenguaje de Manipulación de Datos

El Lenguaje de Manipulación de Datos (Data Manipulation Language, DML), es un idioma proporcionado por los Sistemas Manejadores de Bases de Datos (SMBD) que **permite a los usuarios de la misma llevar a cabo las tareas de consulta o modificación de los datos contenidos en la Bases de Datos**. El lenguaje de manipulación de datos más popular hoy en dia es SQL, usado para recuperar y manipular datos en una base de datos relacional.

## Poblamiento

Ocupamos la herramienta Mockaroo.

Para poblar la base de datos es necesario empezar por las entidades que únicamente tienen Primary Keys para conservar la consistencia de los datos.

### Nota importante:

Es probable que al ejecutar el Script *DML.sql*, se obtenga un error en la tabla **Agendar**. Esto es debido a que el atributo *fechaAsistencia* es calculado aleatoriamente entre la *fechaInicio* y *fechaFin* de cada bazar que esté en la tabla **Agendar**. Dado que no tenemos control sobre la aleatoriedad de los datos es probable que el script intente insertar dos tuplas iguales pero esto es para garantizar que los negocios asistan a diferentes bazares en fechas distintas.

*Para corregirlo únicamente hay que ejecutar nuevamente el script para que los valores aleatorios no coincidan.*

### Tabla Cliente

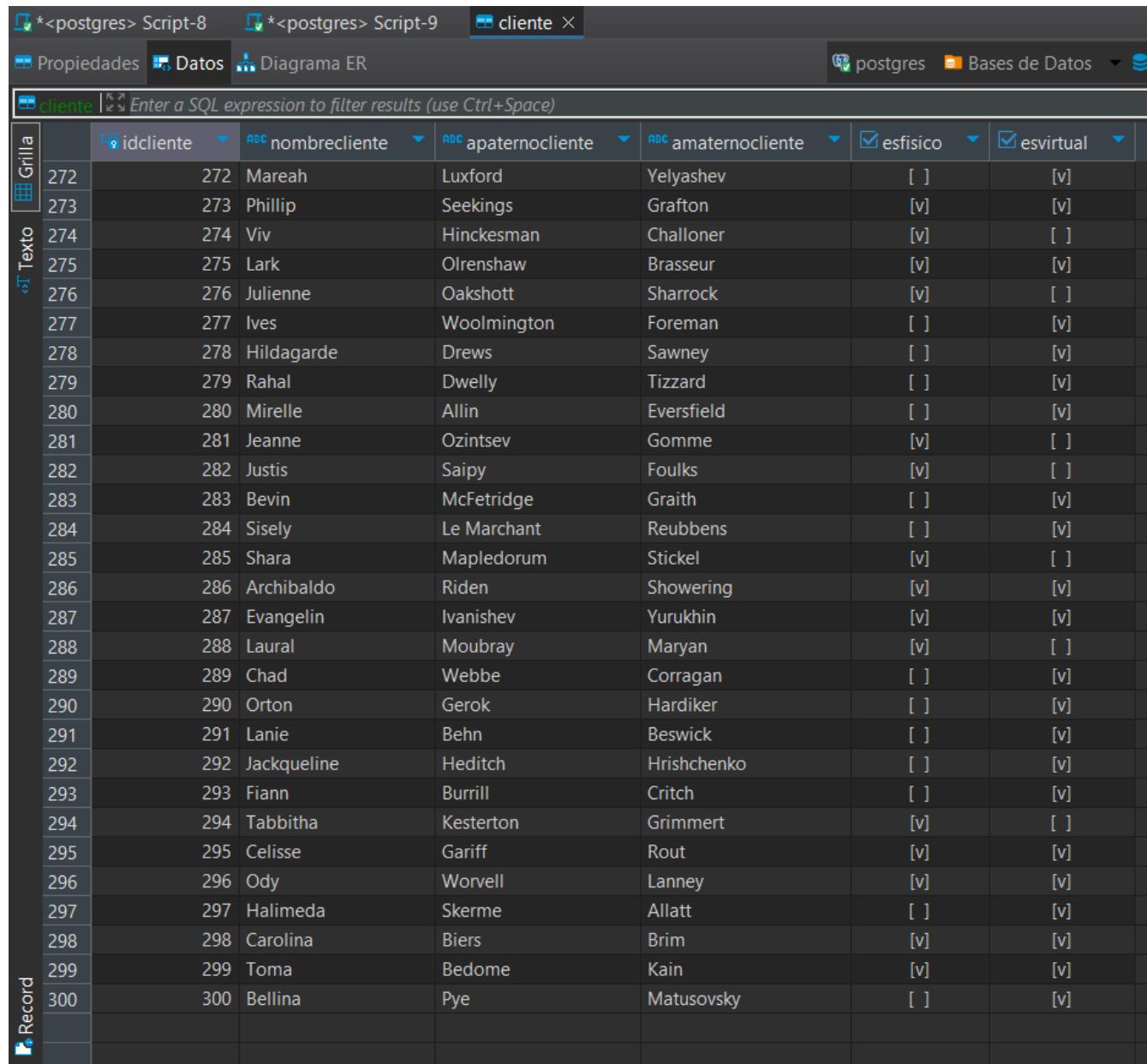
Captura de Mockaroo:

Field Name	Type	Options
IdCliente	Row Number	blank: 0 % $\Sigma$ X
NombreCliente	First Name	blank: 0 % $\Sigma$ X
APaternoCliente	Last Name	blank: 0 % $\Sigma$ X
AMaternoCliente	Last Name	blank: 0 % $\Sigma$ X
EsFisico	Boolean	blank: 0 % $\Sigma$ X
EsVirtual	Boolean	blank: 0 % $\Sigma$ X

Como se puede ver, tenemos una restricción para el atributo **EsVirtual** para garantizar que al menos uno sea True, de la siguiente manera:

```
if this == false and field('EsFisico') == false then
    not this
else
    this
end
```

Y así se ve la tabla poblada:



The screenshot shows the 'cliente' table in DBeaver. The table has columns: idcliente, nombrecliente, apaternocliente, amaternocliente, esfisico, and esvirtual. The 'esfisico' column contains checked checkboxes, and the 'esvirtual' column contains checked checkboxes. The data consists of 300 rows, each with a unique ID from 272 to 300, and various names for the first and last names.

	idcliente	nombrecliente	apaternocliente	amaternocliente	esfisico	esvirtual
Grilla	272	Mareah	Luxford	Yelyashev	[ ]	[v]
Texto	273	Phillip	Seekings	Grafton	[v]	[v]
	274	Viv	Hinckesman	Challoner	[v]	[ ]
	275	Lark	Olrershaw	Brasseur	[v]	[v]
	276	Julienne	Oakshott	Sharrock	[v]	[ ]
	277	Ives	Woolmington	Foreman	[ ]	[v]
	278	Hildagarde	Drews	Sawney	[ ]	[v]
	279	Rahal	Dwelly	Tizzard	[ ]	[v]
	280	Mirelle	Allin	Eversfield	[ ]	[v]
	281	Jeanne	Ozintsev	Gomme	[v]	[ ]
	282	Justis	Saipy	Foulks	[v]	[ ]
	283	Bevin	McFetridge	Graith	[ ]	[v]
	284	Sisely	Le Marchant	Reubbens	[ ]	[v]
	285	Shara	Mapledorum	Stickel	[v]	[ ]
	286	Archibaldo	Riden	Showering	[v]	[v]
	287	Evangelin	Ivanishev	Yurukhin	[v]	[v]
	288	Laural	Moubray	Maryan	[v]	[ ]
	289	Chad	Webbe	Corragan	[ ]	[v]
	290	Orton	Gerok	Hardiker	[ ]	[v]
	291	Lanie	Behn	Beswick	[ ]	[v]
	292	Jackqueline	Heditch	Hrishchenko	[ ]	[v]
	293	Fian	Burrill	Critch	[ ]	[v]
	294	Tabitha	Kesterton	Grimmert	[v]	[ ]
	295	Celisse	Gariff	Rout	[v]	[v]
	296	Ody	Worvell	Lanney	[v]	[v]
	297	Halimeda	Skerme	Allatt	[ ]	[v]
	298	Carolina	Biers	Brim	[v]	[v]
	299	Toma	Bedome	Kain	[v]	[v]
Record	300	Bellina	Pye	Matusovsky	[ ]	[v]

Ahora, podemos poblar las tablas que dependen de la tabla **Cliente**:

## Tabla Domicilio Cliente

## Captura de Mockaroo:

Y así se ve la tabla poblada:

	idcliente	calle	numeroexterior	numerointerior	colonia	estado
272	272	Corben	96742	Room 1223	West End	New Jersey
273	273	Helena	84976	PO Box 91506	West End	Pennsylvania
274	274	Rieder	17700	Apt 114	Pinehurst	Michigan
275	275	Algoma	447	Suite 81	Highland Park	Texas
276	276	Stoughton	3	Suite 58	Greenbriar	Pennsylvania
277	277	Bashford	8037	Apt 385	Meadowbrook	Washington
278	278	Nevada	14688	PO Box 4834	Hillcrest	Massachusetts
279	279	Buena Vista	7	14th Floor	Downtown	Pennsylvania
280	280	Onsgard	49604	20th Floor	Highland Park	Michigan
281	281	Mcguire	1	PO Box 88788	Maplewood	Virginia
282	282	Sloan	5	Suite 78	Cedar Ridge	Michigan
283	283	Melody	0197	Room 195	Greenbriar	Texas
284	284	Oak	1366	18th Floor	East Side	Texas
285	285	Miller	63790	Room 1560	Riverside	Arizona
286	286	Lakewood	4199	9th Floor	Riverside	Florida
287	287	Eastlawn	636	Suite 14	Riverside	Washington
288	288	Homewood	8	Room 844	East Side	Ohio
289	289	Toban	44	Room 1044	Pinehurst	Washington
290	290	Esker	169	Room 123	Sunset Heights	North Carolina
291	291	Rigney	5	10th Floor	Oakwood	Virginia
292	292	Everett	3	Room 1908	Oakwood	California
293	293	Kedzie	5	20th Floor	West End	Florida
294	294	Shelley	54	PO Box 71667	Riverside	Georgia
295	295	Loeprich	467	Suite 63	West End	California
296	296	Atwood	82	15th Floor	Willow Creek	New Jersey
297	297	Hintze	1	Suite 6	Greenbriar	Ohio
298	298	Esch	3	PO Box 72506	Cedar Ridge	Illinois
299	299	Debra	1	PO Box 3960	Highland Park	California
300	300	Elmside	8	11th Floor	West End	Washington

## Tabla Método Pago

Captura de Mockaroo:

Field Name	Type	Options
IdCliente	Row Number	blank: 0 % $\Sigma$ X
MetodoPago	Custom List	Efectivo, Tarjeta $\Theta$ cartesian blank: 0 % $\Sigma$ X

Como se puede ver, tenemos una restricción para el atributo **IdCliente** para garantizar que el atributo realmente sea multivalorado, de la siguiente manera:

this/2

Y así se ve la tabla poblada:

The screenshot shows the Mockaroo interface with the 'metodopago' table selected. The table has two columns: 'idcliente' and 'metodopago'. The 'idcliente' column contains values from 572 to 600, and the 'metodopago' column contains values 'Efectivo' or 'Tarjeta'. The 'Grilla' (grid) view is selected.

idcliente	metodopago
572	Efectivo
573	Tarjeta
574	Efectivo
575	Tarjeta
576	Efectivo
577	Tarjeta
578	Efectivo
579	Tarjeta
580	Efectivo
581	Tarjeta
582	Efectivo
583	Tarjeta
584	Efectivo
585	Tarjeta
586	Efectivo
587	Tarjeta
588	Efectivo
589	Tarjeta
590	Efectivo
591	Tarjeta
592	Efectivo
593	Tarjeta
594	Efectivo
595	Tarjeta
596	Efectivo
597	Tarjeta
598	Efectivo
599	Tarjeta
600	Efectivo

## Tabla Tarjeta

Captura de Mockaroo:

Field Name	Type	Options
Numerotarjeta	Credit Card #	All Card Types
Idcliente	Row Number	blank: 0 % <input type="button" value="Σ"/> <input type="button" value="X"/>

Y así se ve la tabla poblada:

The screenshot shows a PostgreSQL database interface with the 'tarjeta' table selected. The table has two columns: 'numerotarjeta' (Credit Card Number) and 'idcliente' (Client ID). The data is as follows:

numerotarjeta	idcliente
5108757374886856	272
5048370340653021	273
5048372584059980	274
5048371551444159	275
5108752122167485	276
5108759485899711	277
5108750262140239	278
5108753883466660	279
5048371400779110	280
5048378269419647	281
5048378515443631	282
5048371325162095	283
5048378285807510	284
5048378372789449	285
5048372893355236	286
5048377478036549	287
5048376786495975	288
5108750255697583	289
5108753439773767	290
5108756135008305	291
5048379115220908	292
5048378147741881	293
5108753623319591	294
5048376462164556	295
5108756384464142	296
5108753067146302	297
5108758949737889	298
5108757058855003	299
5048372491977589	300

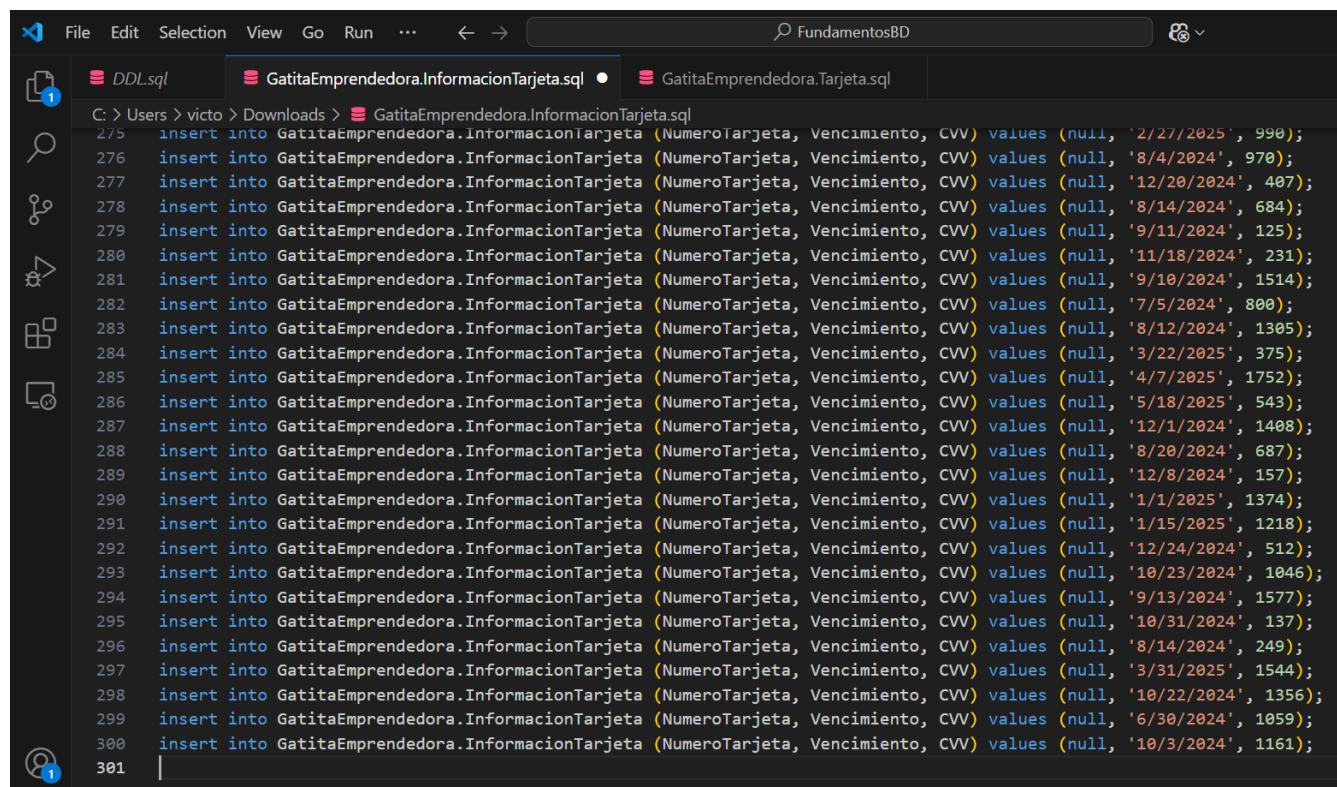
## Tabla Información Tarjeta

Captura de Mockaroo:

Field Name	Type	Options
NumeroTarjeta	Blank	blank: 0 % $\Sigma$ X
Vencimiento	Datetime	05/31/2024 <input type="button" value="Calendar"/> to 05/31/2025 <input type="button" value="Calendar"/> format: m/d/yyyy blank: 0 % $\Sigma$ X
cvv	Number	min: 100 max: 2000 decimals: 0 blank: 0 % $\Sigma$ X

Dejamos el tipo de dato de Vencimiento como "DATE" porque era el que más se acercaba a lo que queremos pero la idea es tener como prioridad MES y AÑO.

Como se puede ver, generamos la columna de número de tarjeta con valores null y esto es porque Mockaroo nos genera números aleatorios cada vez, por lo que necesitamos garantizar que haya números de tarjetas existentes en la BD. Con ayuda de un editor de código (Visual Studio Code) copiamos y pegamos de manera eficiente todos los números de tarjetas ya existentes de la tabla Tarjeta.

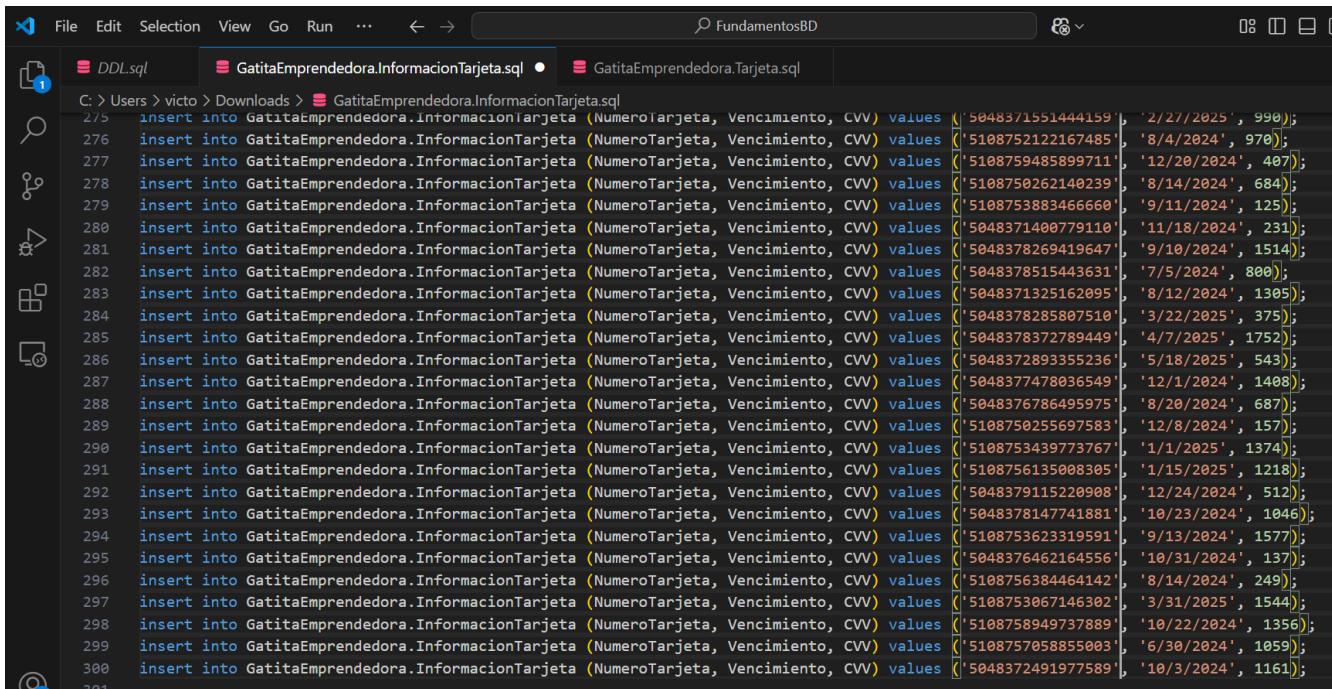


```

File Edit Selection View Go Run ... ← → ⌂ FundamentosBD
DDL.sql GatitaEmprendedora.InformacionTarjeta.sql ● GatitaEmprendedora.Tarjeta.sql

C: > Users > victo > Downloads > GatitaEmprendedora.InformacionTarjeta.sql
275 insert into GatitaEmprendedora.InformacionTarjeta (NumeroTarjeta, Vencimiento, CVV) values (null, '2/27/2025', 990);
276 insert into GatitaEmprendedora.InformacionTarjeta (NumeroTarjeta, Vencimiento, CVV) values (null, '8/4/2024', 970);
277 insert into GatitaEmprendedora.InformacionTarjeta (NumeroTarjeta, Vencimiento, CVV) values (null, '12/20/2024', 407);
278 insert into GatitaEmprendedora.InformacionTarjeta (NumeroTarjeta, Vencimiento, CVV) values (null, '8/14/2024', 684);
279 insert into GatitaEmprendedora.InformacionTarjeta (NumeroTarjeta, Vencimiento, CVV) values (null, '9/11/2024', 125);
280 insert into GatitaEmprendedora.InformacionTarjeta (NumeroTarjeta, Vencimiento, CVV) values (null, '11/18/2024', 231);
281 insert into GatitaEmprendedora.InformacionTarjeta (NumeroTarjeta, Vencimiento, CVV) values (null, '9/10/2024', 1514);
282 insert into GatitaEmprendedora.InformacionTarjeta (NumeroTarjeta, Vencimiento, CVV) values (null, '7/5/2024', 800);
283 insert into GatitaEmprendedora.InformacionTarjeta (NumeroTarjeta, Vencimiento, CVV) values (null, '8/12/2024', 1305);
284 insert into GatitaEmprendedora.InformacionTarjeta (NumeroTarjeta, Vencimiento, CVV) values (null, '3/22/2025', 375);
285 insert into GatitaEmprendedora.InformacionTarjeta (NumeroTarjeta, Vencimiento, CVV) values (null, '4/7/2025', 1752);
286 insert into GatitaEmprendedora.InformacionTarjeta (NumeroTarjeta, Vencimiento, CVV) values (null, '5/18/2025', 543);
287 insert into GatitaEmprendedora.InformacionTarjeta (NumeroTarjeta, Vencimiento, CVV) values (null, '12/1/2024', 1408);
288 insert into GatitaEmprendedora.InformacionTarjeta (NumeroTarjeta, Vencimiento, CVV) values (null, '8/20/2024', 687);
289 insert into GatitaEmprendedora.InformacionTarjeta (NumeroTarjeta, Vencimiento, CVV) values (null, '12/8/2024', 157);
290 insert into GatitaEmprendedora.InformacionTarjeta (NumeroTarjeta, Vencimiento, CVV) values (null, '1/1/2025', 1374);
291 insert into GatitaEmprendedora.InformacionTarjeta (NumeroTarjeta, Vencimiento, CVV) values (null, '1/15/2025', 1218);
292 insert into GatitaEmprendedora.InformacionTarjeta (NumeroTarjeta, Vencimiento, CVV) values (null, '12/24/2024', 512);
293 insert into GatitaEmprendedora.InformacionTarjeta (NumeroTarjeta, Vencimiento, CVV) values (null, '10/23/2024', 1846);
294 insert into GatitaEmprendedora.InformacionTarjeta (NumeroTarjeta, Vencimiento, CVV) values (null, '9/13/2024', 1577);
295 insert into GatitaEmprendedora.InformacionTarjeta (NumeroTarjeta, Vencimiento, CVV) values (null, '10/31/2024', 137);
296 insert into GatitaEmprendedora.InformacionTarjeta (NumeroTarjeta, Vencimiento, CVV) values (null, '8/14/2024', 249);
297 insert into GatitaEmprendedora.InformacionTarjeta (NumeroTarjeta, Vencimiento, CVV) values (null, '3/31/2025', 1544);
298 insert into GatitaEmprendedora.InformacionTarjeta (NumeroTarjeta, Vencimiento, CVV) values (null, '10/22/2024', 1356);
299 insert into GatitaEmprendedora.InformacionTarjeta (NumeroTarjeta, Vencimiento, CVV) values (null, '6/30/2024', 1059);
300 insert into GatitaEmprendedora.InformacionTarjeta (NumeroTarjeta, Vencimiento, CVV) values (null, '10/3/2024', 1161);
301 |

```

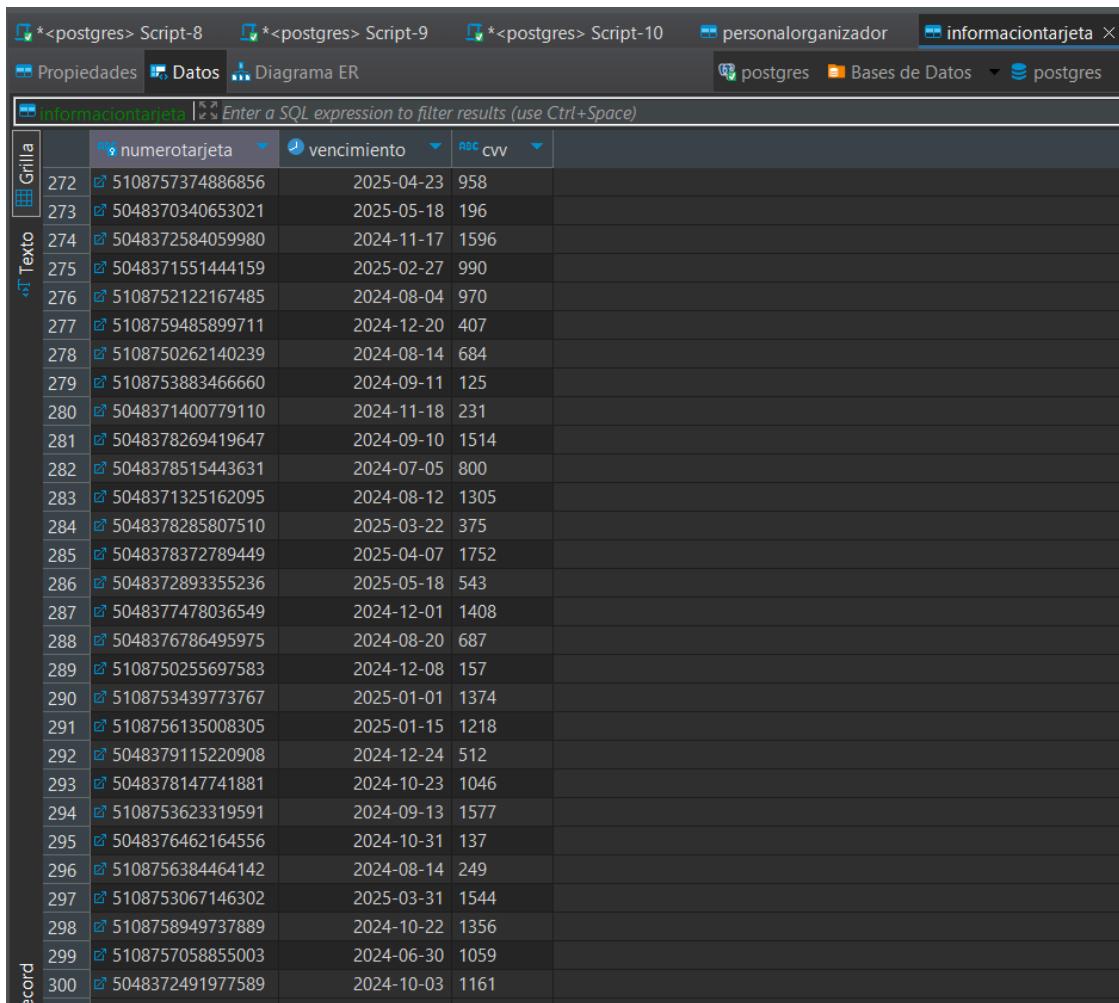


```

C: > Users > victo > Downloads > GatitaEmprendedora.InformacionTarjeta.sql
275 insert into GatitaEmprendedora.InformacionTarjeta (NumeroTarjeta, Vencimiento, CVV) values ('5048371551444159', '2/27/2025', 990);
276 insert into GatitaEmprendedora.InformacionTarjeta (NumeroTarjeta, Vencimiento, CVV) values ('5108752122167485', '8/4/2024', 970);
277 insert into GatitaEmprendedora.InformacionTarjeta (NumeroTarjeta, Vencimiento, CVV) values ('5108759485899711', '12/20/2024', 407);
278 insert into GatitaEmprendedora.InformacionTarjeta (NumeroTarjeta, Vencimiento, CVV) values ('5108750262140239', '8/14/2024', 684);
279 insert into GatitaEmprendedora.InformacionTarjeta (NumeroTarjeta, Vencimiento, CVV) values ('5108753883466660', '9/11/2024', 125);
280 insert into GatitaEmprendedora.InformacionTarjeta (NumeroTarjeta, Vencimiento, CVV) values ('5048371400779110', '11/18/2024', 231);
281 insert into GatitaEmprendedora.InformacionTarjeta (NumeroTarjeta, Vencimiento, CVV) values ('5048378269419647', '9/10/2024', 1514);
282 insert into GatitaEmprendedora.InformacionTarjeta (NumeroTarjeta, Vencimiento, CVV) values ('504837815443631', '7/5/2024', 800);
283 insert into GatitaEmprendedora.InformacionTarjeta (NumeroTarjeta, Vencimiento, CVV) values ('5048371325162095', '8/12/2024', 1305);
284 insert into GatitaEmprendedora.InformacionTarjeta (NumeroTarjeta, Vencimiento, CVV) values ('5048378285807510', '3/22/2025', 375);
285 insert into GatitaEmprendedora.InformacionTarjeta (NumeroTarjeta, Vencimiento, CVV) values ('5048378372789449', '4/7/2025', 1752);
286 insert into GatitaEmprendedora.InformacionTarjeta (NumeroTarjeta, Vencimiento, CVV) values ('5048372893355236', '5/18/2025', 543);
287 insert into GatitaEmprendedora.InformacionTarjeta (NumeroTarjeta, Vencimiento, CVV) values ('5048377478036549', '12/1/2024', 1408);
288 insert into GatitaEmprendedora.InformacionTarjeta (NumeroTarjeta, Vencimiento, CVV) values ('5048376786495975', '8/20/2024', 687);
289 insert into GatitaEmprendedora.InformacionTarjeta (NumeroTarjeta, Vencimiento, CVV) values ('5108750255697583', '12/8/2024', 157);
290 insert into GatitaEmprendedora.InformacionTarjeta (NumeroTarjeta, Vencimiento, CVV) values ('5108753439773767', '1/1/2025', 1374);
291 insert into GatitaEmprendedora.InformacionTarjeta (NumeroTarjeta, Vencimiento, CVV) values ('5108756135008305', '1/15/2025', 1218);
292 insert into GatitaEmprendedora.InformacionTarjeta (NumeroTarjeta, Vencimiento, CVV) values ('5048379115220908', '12/24/2024', 512);
293 insert into GatitaEmprendedora.InformacionTarjeta (NumeroTarjeta, Vencimiento, CVV) values ('5048378147741881', '10/23/2024', 1046);
294 insert into GatitaEmprendedora.InformacionTarjeta (NumeroTarjeta, Vencimiento, CVV) values ('5108753623319591', '9/13/2024', 1577);
295 insert into GatitaEmprendedora.InformacionTarjeta (NumeroTarjeta, Vencimiento, CVV) values ('5048376462164556', '10/31/2024', 137);
296 insert into GatitaEmprendedora.InformacionTarjeta (NumeroTarjeta, Vencimiento, CVV) values ('5108756384464142', '8/14/2024', 249);
297 insert into GatitaEmprendedora.InformacionTarjeta (NumeroTarjeta, Vencimiento, CVV) values ('5108753067146302', '3/31/2025', 1544);
298 insert into GatitaEmprendedora.InformacionTarjeta (NumeroTarjeta, Vencimiento, CVV) values ('5108758949737889', '10/22/2024', 1356);
299 insert into GatitaEmprendedora.InformacionTarjeta (NumeroTarjeta, Vencimiento, CVV) values ('5108757058855003', '6/30/2024', 1059);
300 insert into GatitaEmprendedora.InformacionTarjeta (NumeroTarjeta, Vencimiento, CVV) values ('5048372491977589', '10/3/2024', 1161);

```

Y así se ve la tabla poblada:



record	numerotarjeta	vencimiento	cvv
272	5108757374886856	2025-04-23	958
273	5048370340653021	2025-05-18	196
274	5048372584059980	2024-11-17	1596
275	5048371551444159	2025-02-27	990
276	5108752122167485	2024-08-04	970
277	5108759485899711	2024-12-20	407
278	5108750262140239	2024-08-14	684
279	5108753883466660	2024-09-11	125
280	5048371400779110	2024-11-18	231
281	5048378269419647	2024-09-10	1514
282	5048378515443631	2024-07-05	800
283	5048371325162095	2024-08-12	1305
284	5048378285807510	2025-03-22	375
285	5048378372789449	2025-04-07	1752
286	5048372893355236	2025-05-18	543
287	5048377478036549	2024-12-01	1408
288	5048376786495975	2024-08-20	687
289	5108750255697583	2024-12-08	157
290	5108753439773767	2025-01-01	1374
291	5108756135008305	2025-01-15	1218
292	5048379115220908	2024-12-24	512
293	5048378147741881	2024-10-23	1046
294	5108753623319591	2024-09-13	1577
295	5048376462164556	2024-10-31	137
296	5108756384464142	2024-08-14	249
297	5108753067146302	2025-03-31	1544
298	5108758949737889	2024-10-22	1356
299	5108757058855003	2024-06-30	1059
300	5048372491977589	2024-10-03	1161

Ahora, ya acabamos con las tablas que dependen directa y únicamente de la tabla **Cliente**, por lo que podemos seguir con otra tabla que tenga PKs. **Tabla Bazar**

Captura de Mockaroo:

Field Name	Type	Options
IdBazar	Row Number	blank: 0% $\Sigma$ X
NombreBazar	Custom List	Central Market, Farmers Market, City Market, Harbor Market, Sunset Market, Riverfront Market, Mou  random  blank: 0% $\Sigma$ X
Calle	Street Name	blank: 0% $\Sigma$ X
NumeroExterior	Street Number	blank: 0% $\Sigma$ X
NumeroInterior	Address Line 2	blank: 0% $\Sigma$ X
Colonia	Custom List	Downtown, Midtown, Uptown, West End, East Side, South Park, North Hills, Riverdale, Sunset Height  random  blank: 0% $\Sigma$ X
Estado	Custom List	California, Texas, Florida, New York, Pennsylvania, Illinois, Ohio, Georgia, North Carolina, Michigan, N  random  blank: 0% $\Sigma$ X
Modalidad	Custom List	presencial, virtual  random  blank: 0% $\Sigma$ X
FechaInicio	Datetime	01/01/2023  to 12/31/2023  format: m/d/yyyy  blank: 0% $\Sigma$ X
FechaFin	Datetime	01/01/2024  to 12/31/2024  format: m/d/yyyy  blank: 0% $\Sigma$ X

Y así se ve la tabla poblada:

	idbazar	nombrebazar	calle	numerointerior	numeroexterior	colonia	estado	modalidad	fechainicio	fechafin
272	272	Pine Market	Oak Valley	Room 1956	2	Downtown	Washington	presencial	2023-10-03	2024-11-09
273	273	Ocean Market	Mosinee	Apt 1203	88	Maplewood	Arizona	presencial	2023-05-04	2024-05-12
274	274	Hill Market	Maple	Apt 474	82783	Oakwood	Ohio	presencial	2023-09-26	2024-11-23
275	275	Riverfront Market	Gulseth	Apt 582	7	Hillcrest	Michigan	presencial	2023-06-18	2024-04-29
276	276	City Market	Becker	Room 614	4	Uptown	Michigan	virtual	2023-01-06	2024-11-09
277	277	Harbor Market	Redwing	PO Box 55674	594	Riverdale	Washington	presencial	2023-03-01	2024-11-18
278	278	Riverfront Market	Union	Apt 210	806	Hillcrest	North Carolina	virtual	2023-08-24	2024-11-26
279	279	Hill Market	Schlimgen	Apt 191	91222	Midtown	Georgia	virtual	2023-02-01	2024-03-15
280	280	Meadow Market	Old Gate	Apt 377	184	South Park	Florida	presencial	2023-03-11	2024-03-12
281	281	Farmers Market	Oakridge	Suite 55	94	North Hills	Georgia	virtual	2023-10-07	2024-01-23
282	282	Grove Market	Scott	PO Box 68238	0	Oakwood	North Carolina	virtual	2023-07-15	2024-03-10
283	283	Mountain Market	Bluestem	PO Box 36670	295	West End	California	virtual	2023-04-16	2024-09-27
284	284	Grove Market	Ludington	1st Floor	67291	Greenfield	Arizona	presencial	2023-07-30	2024-01-20
285	285	Farmers Market	Menomonie	Suite 90	26893	Downtown	Michigan	virtual	2023-11-23	2024-05-01
286	286	Ocean Market	Bashford	Suite 1	0	Oakwood	Texas	virtual	2023-03-31	2024-11-26
287	287	City Market	Messerschmidt	8th Floor	98	Hillcrest	Virginia	presencial	2023-06-02	2024-11-13
288	288	Mountain Market	Northview	Suite 58	979	South Park	Michigan	presencial	2023-01-12	2024-05-15
289	289	Grove Market	Swallow	PO Box 20581	970	Uptown	Washington	virtual	2023-12-18	2024-05-29
290	290	Pine Market	Shasta	10th Floor	9520	Downtown	Massachusetts	virtual	2023-03-07	2024-09-20
291	291	Farmers Market	Comanche	Room 895	59647	East Side	California	virtual	2023-07-30	2024-08-14
292	292	Sunset Market	Westridge	1st Floor	2463	North Hills	California	presencial	2023-06-16	2024-08-25
293	293	Harbor Market	Dayton	11th Floor	3	Willowbrook	Texas	virtual	2023-07-23	2024-11-04
294	294	Ocean Market	Bartillon	Suite 25	7	Pinecrest	Virginia	presencial	2023-07-21	2024-05-17
295	295	Hill Market	Fair Oaks	15th Floor	5	North Hills	Washington	virtual	2023-10-08	2024-11-22
296	296	Grove Market	Myrtle	Room 534	382	Downtown	Michigan	presencial	2023-03-03	2024-07-01
297	297	Lake Market	Saint Paul	18th Floor	4	Sunset Heights	Florida	virtual	2023-06-04	2024-12-10
298	298	Harbor Market	West	4th Floor	59438	Pinecrest	Arizona	virtual	2023-11-22	2024-10-10
299	299	Farmers Market	Rowland	PO Box 88259	3	Hillcrest	Florida	presencial	2023-10-20	2024-01-16
300	300	Meadow Market	Morrow	Room 1618	37	Sunset Heights	Michigan	presencial	2023-07-26	2024-11-25

Ahora, podemos poblar las tablas que dependen de la tabla **Bazar**:

### Tabla Amenidad Bazar

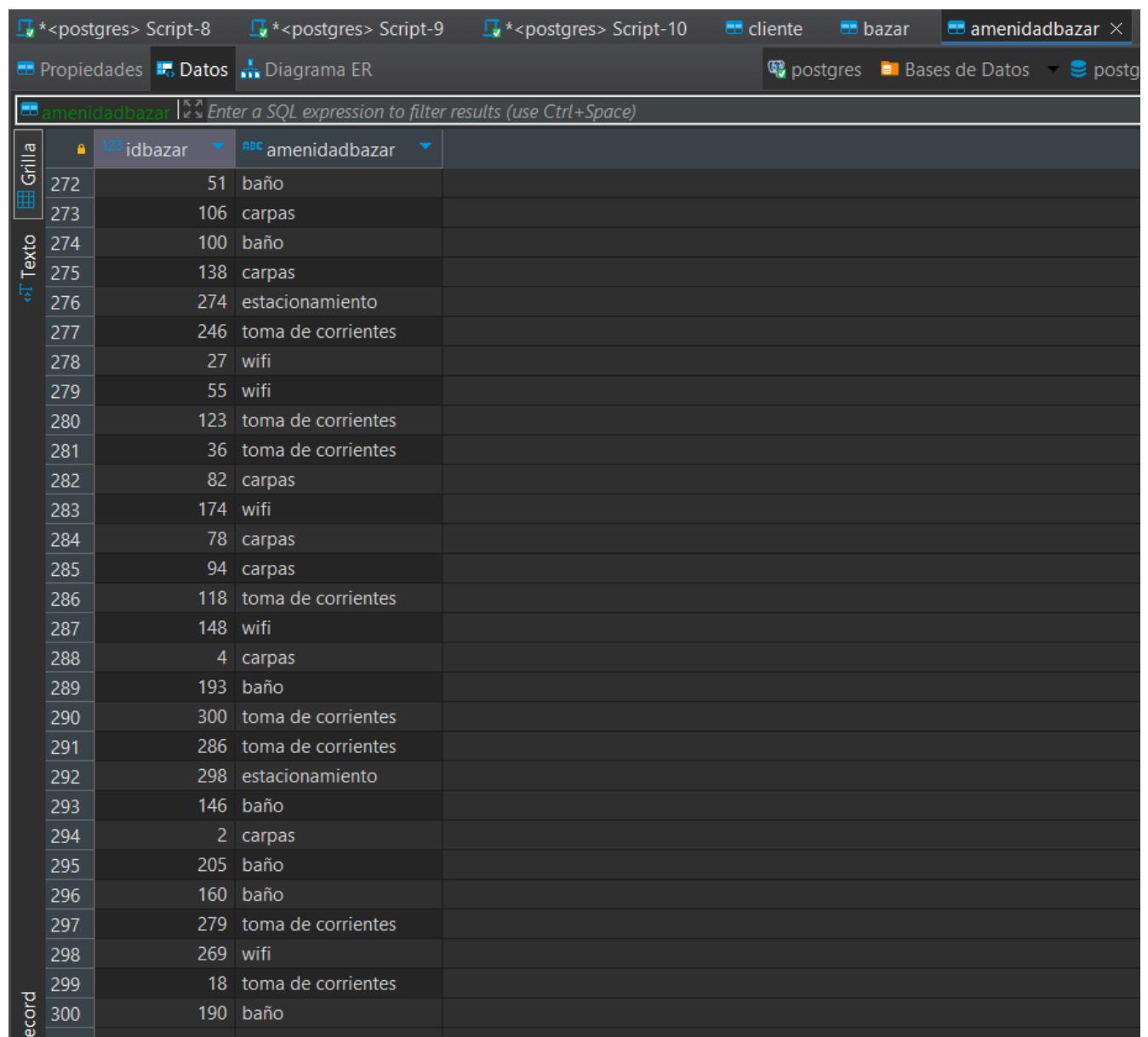
Captura de Mockaroo:

Field Name	Type	Options
IdBazar	Row Number	blank: 0 % $\Sigma$ X
AmenidadBazar	Custom List	estacionamiento, carpas, toma de corrientes, wifi, baño random ▾ blank: 0 % $\Sigma$ X

Como se puede ver, tenemos una restricción, esto es para que haya variedad en las amenidades que tenga cada bazar, de la siguiente manera:

random(1, 300)

Y así se ve la tabla poblada:



The screenshot shows the pgAdmin interface with the 'amenidadbazar' table selected. The table has two columns: 'idbazar' (integer) and 'amenidadbazar' (text). The data consists of 300 rows, each containing a unique ID from 272 to 300 and a random amenity name from a predefined list: baño, carpas, estacionamiento, toma de corrientes, or wifi.

idbazar	amenidadbazar
272	baño
273	carpas
274	baño
275	carpas
276	estacionamiento
277	toma de corrientes
278	wifi
279	wifi
280	toma de corrientes
281	toma de corrientes
282	carpas
283	wifi
284	carpas
285	carpas
286	toma de corrientes
287	wifi
288	carpas
289	baño
290	toma de corrientes
291	toma de corrientes
292	estacionamiento
293	baño
294	carpas
295	baño
296	baño
297	toma de corrientes
298	wifi
299	toma de corrientes
300	baño

Ahora, ya acabamos con las tablas que dependen directa y únicamente de la tabla **Bazar**, por lo que podemos seguir con otra tabla que tenga PKs.

### Tabla Personal Organizador

Captura de Mockaroo:

Field Name	Type	Options
RFC	Character Sequence	^^^^#####^# blank: 0% Σ X
NombrePersonalOrganizad	First Name	blank: 0% Σ X
APaternoPersonalOrganizad	Last Name	blank: 0% Σ X
AMaternoPersonalOrganizad	Last Name	blank: 0% Σ X
Calle	Street Name	blank: 0% Σ X
NumeroExterior	Street Number	blank: 0% Σ X
NumeroInterior	Address Line 2	blank: 0% Σ X
Colonia	City	blank: 0% Σ X
Estado	State	restrict states... Only US blank: 0% Σ X
Salario	Money	between 1 and 1000 in \$ blank: 0% Σ X

Y así se ve la tabla poblada:

Record	rfc	nombre	apaterno	amaterno	calle	numeroexterior	numerointerior	colonia	estado	salario
272	AHPS874691D13	Xerxes	Dillimore	Canto	Buena Vista	1666	3rd Floor	Dallas	Texas	\$848.69
273	DUXO428507CC9	Meria	McGinn	Bitterton	Homewood	975	Apt 709	Houston	Texas	\$871.13
274	ZLMN039850SP7	Augy	Sleightholme	Kovnot	Buena Vista	77	Suite 55	Birmingham	Alabama	\$75.86
275	EMHO546301ZZ8	Anica	Sapsseed	Setterfield	Larry	1	19th Floor	Scranton	Pennsylvania	\$103.69
276	NAMG434585QD8	Hilton	McGir	McTague	Anderson	284	20th Floor	Tempe	Arizona	\$836.90
277	IQVJ254680AK9	Kerstin	Puddifer	Mercey	Schurz	7075	Room 193	Washington	District of Columbia	\$490.70
278	IAHDB39909NE2	Dolph	Shipcott	Wenderoth	Gale	3931	PO Box 95139	Charlotte	North Carolina	\$506.65
279	RUPK118824IJ5	Darleen	Muneely	Wheelwright	Northwestern	61	PO Box 5877	North Port	Florida	\$231.40
280	WCKW099795IW3	Chantalle	Quiddihy	Pound	4th	42	Suite 77	Los Angeles	California	\$413.44
281	OKDB686714RN7	Leoine	de Tocqueville	McAneny	Hansons	27	Suite 43	Harrisburg	Pennsylvania	\$381.35
282	LNJD639898ZF2	Mirma	Cottom	Rainford	Homewood	8	Room 399	Zephyrhills	Florida	\$112.11
283	PLEL185289CS6	Mercy	Arnot	Tribell	Montana	44	Room 1790	Chicago	Illinois	\$820.80
284	SSPB754027UQ8	Tybie	Poynzer	Aramchik	Sachs	9	Apt 776	Erie	Pennsylvania	\$52.39
285	PFVB347658JC9	Lucius	Paling	Sinney	Gerald	6	Room 1589	Chicago	Illinois	\$674.15
286	BZEY489816PL8	Jeremias	Riddlesden	Vedishchev	Lawn	7520	10th Floor	Hagerstown	Maryland	\$706.45
287	NSGO788580JF0	Ronnicka	Calvert	Lyddyard	Almo	9081	Suite 55	Jacksonville	Florida	\$349.85
288	VGUT580953RH5	Phelia	Brimicombe	Mollatt	Corscot	938	7th Floor	Melbourne	Florida	\$85.45
289	EWHB671134ZR7	Melva	Meo	Curtis	Rockefeller	742	Apt 233	New Orleans	Louisiana	\$315.01
290	ENOI952607ZE9	Nyssa	Mosdall	Sutehall	Oakridge	19	PO Box 70958	Littleton	Colorado	\$28.88
291	TGDO797068WP0	Cherida	Bake	Harmes	7th	235	18th Floor	Littleton	Colorado	\$380.36
292	AXFY004529PJ1	Ezri	Matchett	Rookes	Dovetail	80026	Room 975	Sacramento	California	\$92.25
293	WABB829145BV7	Arvin	Moxom	Santacrole	Sutherland	427	Room 1365	Des Moines	Iowa	\$764.37
294	BFR125699GT9	Osbourne	Trim	Poley	Oak	673	Apt 459	Long Beach	California	\$886.14
295	SLBB556080GV3	Fan	Reford	Fosken	Shoshone	566	Suite 32	Garden Grove	California	\$266.82
296	VUKT158637QNO	Daune	Plewright	Randalston	Vidon	990	PO Box 34074	Grand Forks	North Dakota	\$291.64
297	TZMJ008217DX9	Lucian	Andrewwartha	Taffee	Bobwhite	1411	Room 1722	Indianapolis	Indiana	\$219.86
298	MUND585751OL0	Leon	Gristwhaita	Dymock	Butternut	4	Apt 1130	Roanoke	Virginia	\$335.73
299	KMDA254191VR1	Winni	Curtayne	Dawltrey	Sauthoff	4	Room 11	New York City	New York	\$82.93
300	EPRW584492NQ7	Lauraine	Comforth	Forryan	Schlimgen	5	13th Floor	Lake Worth	Florida	\$202.60

Ahora, podemos poblar las tablas que dependen de la tabla **Personal Organizador**:

### Tabla Horario Personal Organizador

Captura de Mockaroo:

Cabe aclarar que elegimos la leyenda "matutino" para el horario 8-12, "vespertino1" para 12-4 y "vespertino2" para 4-8.

Field Name	Type	Options
RFC	Blank	blank: 0% Σ X
HorarioPersonalOrganizad	Custom List	matutino, vespertino1, vespertino2 random ▾ blank: 0% Σ X

Como se puede ver, generamos la columna de RFC con valores null y esto es porque Mockaroo nos genera números aleatorios cada vez, por lo que necesitamos garantizar que haya RFCs en la BD. Con ayuda de un editor de código (Visual Studio Code) copiamos y pegamos de manera eficiente todos los RFCs ya existentes de la tabla Personal Organizador.

```

C: > Users > victo > Downloads > GatitaEmprendedora.HorarioPersonalOrganizador.sql
265 insert into GatitaEmprendedora.HorarioPersonalOrganizador (RFC, HorarioPersonalOrganizador) values (null, 'vespertino2');
266 insert into GatitaEmprendedora.HorarioPersonalOrganizador (RFC, HorarioPersonalOrganizador) values (null, 'vespertino1');
267 insert into GatitaEmprendedora.HorarioPersonalOrganizador (RFC, HorarioPersonalOrganizador) values (null, 'matutino');
268 insert into GatitaEmprendedora.HorarioPersonalOrganizador (RFC, HorarioPersonalOrganizador) values (null, 'matutino');
269 insert into GatitaEmprendedora.HorarioPersonalOrganizador (RFC, HorarioPersonalOrganizador) values (null, 'matutino');
270 insert into GatitaEmprendedora.HorarioPersonalOrganizador (RFC, HorarioPersonalOrganizador) values (null, 'vespertino2');
271 insert into GatitaEmprendedora.HorarioPersonalOrganizador (RFC, HorarioPersonalOrganizador) values (null, 'matutino');
272 insert into GatitaEmprendedora.HorarioPersonalOrganizador (RFC, HorarioPersonalOrganizador) values (null, 'vespertino1');
273 insert into GatitaEmprendedora.HorarioPersonalOrganizador (RFC, HorarioPersonalOrganizador) values (null, 'vespertino1');
274 insert into GatitaEmprendedora.HorarioPersonalOrganizador (RFC, HorarioPersonalOrganizador) values (null, 'matutino');
275 insert into GatitaEmprendedora.HorarioPersonalOrganizador (RFC, HorarioPersonalOrganizador) values (null, 'vespertino2');
276 insert into GatitaEmprendedora.HorarioPersonalOrganizador (RFC, HorarioPersonalOrganizador) values (null, 'matutino');
277 insert into GatitaEmprendedora.HorarioPersonalOrganizador (RFC, HorarioPersonalOrganizador) values (null, 'vespertino2');
278 insert into GatitaEmprendedora.HorarioPersonalOrganizador (RFC, HorarioPersonalOrganizador) values (null, 'vespertino2');
279 insert into GatitaEmprendedora.HorarioPersonalOrganizador (RFC, HorarioPersonalOrganizador) values (null, 'vespertino2');
280 insert into GatitaEmprendedora.HorarioPersonalOrganizador (RFC, HorarioPersonalOrganizador) values (null, 'vespertino1');
281 insert into GatitaEmprendedora.HorarioPersonalOrganizador (RFC, HorarioPersonalOrganizador) values (null, 'matutino');
282 insert into GatitaEmprendedora.HorarioPersonalOrganizador (RFC, HorarioPersonalOrganizador) values (null, 'vespertino1');
283 insert into GatitaEmprendedora.HorarioPersonalOrganizador (RFC, HorarioPersonalOrganizador) values (null, 'vespertino2');
284 insert into GatitaEmprendedora.HorarioPersonalOrganizador (RFC, HorarioPersonalOrganizador) values (null, 'vespertino2');
285 insert into GatitaEmprendedora.HorarioPersonalOrganizador (RFC, HorarioPersonalOrganizador) values (null, 'vespertino1');
286 insert into GatitaEmprendedora.HorarioPersonalOrganizador (RFC, HorarioPersonalOrganizador) values (null, 'matutino');
287 insert into GatitaEmprendedora.HorarioPersonalOrganizador (RFC, HorarioPersonalOrganizador) values (null, 'matutino');
288 insert into GatitaEmprendedora.HorarioPersonalOrganizador (RFC, HorarioPersonalOrganizador) values (null, 'vespertino2');
289 insert into GatitaEmprendedora.HorarioPersonalOrganizador (RFC, HorarioPersonalOrganizador) values (null, 'vespertino1');
290 insert into GatitaEmprendedora.HorarioPersonalOrganizador (RFC, HorarioPersonalOrganizador) values (null, 'matutino');
291 insert into GatitaEmprendedora.HorarioPersonalOrganizador (RFC, HorarioPersonalOrganizador) values (null, 'matutino');
292 insert into GatitaEmprendedora.HorarioPersonalOrganizador (RFC, HorarioPersonalOrganizador) values (null, 'vespertino2');
293 insert into GatitaEmprendedora.HorarioPersonalOrganizador (RFC, HorarioPersonalOrganizador) values (null, 'vespertino1');
294 insert into GatitaEmprendedora.HorarioPersonalOrganizador (RFC, HorarioPersonalOrganizador) values (null, 'vespertino1');
295 insert into GatitaEmprendedora.HorarioPersonalOrganizador (RFC, HorarioPersonalOrganizador) values (null, 'vespertino2');
296 insert into GatitaEmprendedora.HorarioPersonalOrganizador (RFC, HorarioPersonalOrganizador) values (null, 'vespertino2');
297 insert into GatitaEmprendedora.HorarioPersonalOrganizador (RFC, HorarioPersonalOrganizador) values (null, 'matutino');
298 insert into GatitaEmprendedora.HorarioPersonalOrganizador (RFC, HorarioPersonalOrganizador) values (null, 'vespertino2');
299 insert into GatitaEmprendedora.HorarioPersonalOrganizador (RFC, HorarioPersonalOrganizador) values (null, 'matutino');
300 insert into GatitaEmprendedora.HorarioPersonalOrganizador (RFC, HorarioPersonalOrganizador) values (null, 'vespertino1');

```

```

C:\> Users > victo > Downloads > GatitaEmprendedora.HorarioPersonalOrganizador.sql
200 insert into GatitaEmprendedora.HorarioPersonalOrganizador (RFC, HorarioPersonalOrganizador) values ('VTC0288920H01', 'vespertino2');
266 insert into GatitaEmprendedora.HorarioPersonalOrganizador (RFC, HorarioPersonalOrganizador) values ('CQNU885981A08', 'vespertino1');
267 insert into GatitaEmprendedora.HorarioPersonalOrganizador (RFC, HorarioPersonalOrganizador) values ('SYHL381947FF8', 'matutino');
268 insert into GatitaEmprendedora.HorarioPersonalOrganizador (RFC, HorarioPersonalOrganizador) values ('CMP161394UJ7', 'matutino');
269 insert into GatitaEmprendedora.HorarioPersonalOrganizador (RFC, HorarioPersonalOrganizador) values ('CYXJ7641600Q1', 'vespertino2');
270 insert into GatitaEmprendedora.HorarioPersonalOrganizador (RFC, HorarioPersonalOrganizador) values ('RKKD49665HF2', 'matutino');
271 insert into GatitaEmprendedora.HorarioPersonalOrganizador (RFC, HorarioPersonalOrganizador) values ('AHP5874691D13', 'vespertino1');
272 insert into GatitaEmprendedora.HorarioPersonalOrganizador (RFC, HorarioPersonalOrganizador) values ('DUXO428507CC9', 'vespertino1');
273 insert into GatitaEmprendedora.HorarioPersonalOrganizador (RFC, HorarioPersonalOrganizador) values ('ZLMN039850SP7', 'matutino');
274 insert into GatitaEmprendedora.HorarioPersonalOrganizador (RFC, HorarioPersonalOrganizador) values ('EMHO546301ZZ8', 'vespertino2');
275 insert into GatitaEmprendedora.HorarioPersonalOrganizador (RFC, HorarioPersonalOrganizador) values ('NAMG434585QD8', 'matutino');
276 insert into GatitaEmprendedora.HorarioPersonalOrganizador (RFC, HorarioPersonalOrganizador) values ('IQVJ254680AK9', 'vespertino2');
277 insert into GatitaEmprendedora.HorarioPersonalOrganizador (RFC, HorarioPersonalOrganizador) values ('IAHD639909NE2', 'vespertino2');
278 insert into GatitaEmprendedora.HorarioPersonalOrganizador (RFC, HorarioPersonalOrganizador) values ('RUPK118824IUS', 'vespertino2');
279 insert into GatitaEmprendedora.HorarioPersonalOrganizador (RFC, HorarioPersonalOrganizador) values ('WCKW099795IW3', 'vespertino1');
280 insert into GatitaEmprendedora.HorarioPersonalOrganizador (RFC, HorarioPersonalOrganizador) values ('OKDB686714RN7', 'matutino');
281 insert into GatitaEmprendedora.HorarioPersonalOrganizador (RFC, HorarioPersonalOrganizador) values ('LNJD639898ZF2', 'vespertino1');
282 insert into GatitaEmprendedora.HorarioPersonalOrganizador (RFC, HorarioPersonalOrganizador) values ('PLEL185289CS6', 'vespertino2');
283 insert into GatitaEmprendedora.HorarioPersonalOrganizador (RFC, HorarioPersonalOrganizador) values ('SSPB754027UQ8', 'vespertino2');
284 insert into GatitaEmprendedora.HorarioPersonalOrganizador (RFC, HorarioPersonalOrganizador) values ('PFVB347658JC9', 'vespertino1');
285 insert into GatitaEmprendedora.HorarioPersonalOrganizador (RFC, HorarioPersonalOrganizador) values ('BZEY489816PL8', 'matutino');
286 insert into GatitaEmprendedora.HorarioPersonalOrganizador (RFC, HorarioPersonalOrganizador) values ('NSGO788580JF0', 'matutino');
287 insert into GatitaEmprendedora.HorarioPersonalOrganizador (RFC, HorarioPersonalOrganizador) values ('VGUT580953RH5', 'vespertino1');
288 insert into GatitaEmprendedora.HorarioPersonalOrganizador (RFC, HorarioPersonalOrganizador) values ('EWHB671134ZR7', 'vespertino1');
289 insert into GatitaEmprendedora.HorarioPersonalOrganizador (RFC, HorarioPersonalOrganizador) values ('ENOI952607ZE9', 'matutino');
290 insert into GatitaEmprendedora.HorarioPersonalOrganizador (RFC, HorarioPersonalOrganizador) values ('TGDO797068WP0', 'matutino');
291 insert into GatitaEmprendedora.HorarioPersonalOrganizador (RFC, HorarioPersonalOrganizador) values ('AXFY004529PJ1', 'vespertino2');
292 insert into GatitaEmprendedora.HorarioPersonalOrganizador (RFC, HorarioPersonalOrganizador) values ('WABB829145BY7', 'vespertino1');
293 insert into GatitaEmprendedora.HorarioPersonalOrganizador (RFC, HorarioPersonalOrganizador) values ('BFBR125699GT9', 'vespertino1');
294 insert into GatitaEmprendedora.HorarioPersonalOrganizador (RFC, HorarioPersonalOrganizador) values ('SLBB556080GV3', 'vespertino2');
295 insert into GatitaEmprendedora.HorarioPersonalOrganizador (RFC, HorarioPersonalOrganizador) values ('VUKT158637QNO', 'vespertino2');
296 insert into GatitaEmprendedora.HorarioPersonalOrganizador (RFC, HorarioPersonalOrganizador) values ('TZMJ008217DX9', 'matutino');
297 insert into GatitaEmprendedora.HorarioPersonalOrganizador (RFC, HorarioPersonalOrganizador) values ('MUND585751LO0', 'vespertino2');
298 insert into GatitaEmprendedora.HorarioPersonalOrganizador (RFC, HorarioPersonalOrganizador) values ('KMDA254191VR1', 'matutino');
299 insert into GatitaEmprendedora.HorarioPersonalOrganizador (RFC, HorarioPersonalOrganizador) values ('EPRW584492NQ7', 'vespertino1');
300 insert into GatitaEmprendedora.HorarioPersonalOrganizador (RFC, HorarioPersonalOrganizador) values ('EPRW584492NQ7', 'vespertino1');

```

Y así se ve la tabla poblada:

	rfc	horariopersonalorganizador
272	AHPS874691D13	vespertino1
273	DUXO428507CC9	vespertino1
274	ZLMN039850SP7	matutino
275	EMHO546301ZZ8	vespertino2
276	NAMG434585QD8	matutino
277	IQVJ254680AK9	vespertino2
278	IAHD639909NE2	vespertino2
279	RUPK118824IUS	vespertino2
280	WCKW099795IW3	vespertino1
281	OKDB686714RN7	matutino
282	LNJD639898ZF2	vespertino1
283	PLEL185289CS6	vespertino2
284	SSPB754027UQ8	vespertino2
285	PFVB347658JC9	vespertino1
286	BZEY489816PL8	matutino
287	NSGO788580JF0	matutino
288	VGUT580953RH5	vespertino2
289	EWHB671134ZR7	vespertino1
290	ENOI952607ZE9	matutino
291	TGDO797068WP0	matutino
292	AXFY004529PJ1	vespertino2
293	WABB829145BY7	vespertino1
294	BFBR125699GT9	vespertino1
295	SLBB556080GV3	vespertino2
296	VUKT158637QNO	vespertino2
297	TZMJ008217DX9	matutino
298	MUND585751LO0	vespertino2
299	KMDA254191VR1	matutino
300	EPRW584492NQ7	vespertino1

## Tabla Teléfono Personal Organizador

Captura de Mockaroo:

Field Name	Type	Options
RFC	Blank	blank: 0% Σ X
TelefonoPersonalOrg	Digit Sequence	##### blank: 0% Σ X

Como se puede ver, generamos la columna de **RFC** con valores null y esto es porque Mockaroo nos genera números aleatorios cada vez, por lo que necesitamos garantizar que haya **RFCs** en la BD. Con ayuda de un editor de código (Visual Studio Code) copiamos y pegamos de manera eficiente todos datos ya existentes de la tabla **Personal Organizador**.

```

File Edit Selection View Go Run Terminal Help ← → FundamentosBD
EXPLORER ... DDLsql GatitaEmprendedora.TelefonoPersonalOrganizador.sql GatitaEmprendedora.PersonalOrganizador.sql DML.sql
FUNDAMENTOSBD
> Prácticas
> ProyectoFinal_NAMEis...
> Diagramas
> Docs
> SQL
DDL.sql
disparadores_funcio...
DML.sql
Correcciones.md
Reparto.md
Tareas
Lineamientos_FBD.pdf
Links.md
README_NAMEisNULL...
 README.md
266 insert into GatitaEmprendedora.TelefonoPersonalOrganizador (RFC, TelefonoPersonalOrganizador) values (null, '3148121933');
267 insert into GatitaEmprendedora.TelefonoPersonalOrganizador (RFC, TelefonoPersonalOrganizador) values (null, '8934734525');
268 insert into GatitaEmprendedora.TelefonoPersonalOrganizador (RFC, TelefonoPersonalOrganizador) values (null, '3455223834');
269 insert into GatitaEmprendedora.TelefonoPersonalOrganizador (RFC, TelefonoPersonalOrganizador) values (null, '4986054226');
270 insert into GatitaEmprendedora.TelefonoPersonalOrganizador (RFC, TelefonoPersonalOrganizador) values (null, '8718768990');
271 insert into GatitaEmprendedora.TelefonoPersonalOrganizador (RFC, TelefonoPersonalOrganizador) values (null, '3403370274');
272 insert into GatitaEmprendedora.TelefonoPersonalOrganizador (RFC, TelefonoPersonalOrganizador) values (null, '9662301699');
273 insert into GatitaEmprendedora.TelefonoPersonalOrganizador (RFC, TelefonoPersonalOrganizador) values (null, '7316403023');
274 insert into GatitaEmprendedora.TelefonoPersonalOrganizador (RFC, TelefonoPersonalOrganizador) values (null, '1339463034');
275 insert into GatitaEmprendedora.TelefonoPersonalOrganizador (RFC, TelefonoPersonalOrganizador) values (null, '2175075486');
276 insert into GatitaEmprendedora.TelefonoPersonalOrganizador (RFC, TelefonoPersonalOrganizador) values (null, '7467213306');
277 insert into GatitaEmprendedora.TelefonoPersonalOrganizador (RFC, TelefonoPersonalOrganizador) values (null, '3264801556');
278 insert into GatitaEmprendedora.TelefonoPersonalOrganizador (RFC, TelefonoPersonalOrganizador) values (null, '4962382342');
279 insert into GatitaEmprendedora.TelefonoPersonalOrganizador (RFC, TelefonoPersonalOrganizador) values (null, '5986463527');
280 insert into GatitaEmprendedora.TelefonoPersonalOrganizador (RFC, TelefonoPersonalOrganizador) values (null, '6487862025');
281 insert into GatitaEmprendedora.TelefonoPersonalOrganizador (RFC, TelefonoPersonalOrganizador) values (null, '9608381953');
282 insert into GatitaEmprendedora.TelefonoPersonalOrganizador (RFC, TelefonoPersonalOrganizador) values (null, '6818391862');
283 insert into GatitaEmprendedora.TelefonoPersonalOrganizador (RFC, TelefonoPersonalOrganizador) values (null, '3151349650');
284 insert into GatitaEmprendedora.TelefonoPersonalOrganizador (RFC, TelefonoPersonalOrganizador) values (null, '8265156923');
285 insert into GatitaEmprendedora.TelefonoPersonalOrganizador (RFC, TelefonoPersonalOrganizador) values (null, '2312653681');
286 insert into GatitaEmprendedora.TelefonoPersonalOrganizador (RFC, TelefonoPersonalOrganizador) values (null, '7401670866');
287 insert into GatitaEmprendedora.TelefonoPersonalOrganizador (RFC, TelefonoPersonalOrganizador) values (null, '5730425595');
288 insert into GatitaEmprendedora.TelefonoPersonalOrganizador (RFC, TelefonoPersonalOrganizador) values (null, '0841054438');
289 insert into GatitaEmprendedora.TelefonoPersonalOrganizador (RFC, TelefonoPersonalOrganizador) values (null, '2253155430');
290 insert into GatitaEmprendedora.TelefonoPersonalOrganizador (RFC, TelefonoPersonalOrganizador) values (null, '5821627347');
291 insert into GatitaEmprendedora.TelefonoPersonalOrganizador (RFC, TelefonoPersonalOrganizador) values (null, '4942715271');
292 insert into GatitaEmprendedora.TelefonoPersonalOrganizador (RFC, TelefonoPersonalOrganizador) values (null, '0471675195');
293 insert into GatitaEmprendedora.TelefonoPersonalOrganizador (RFC, TelefonoPersonalOrganizador) values (null, '1497382674');
294 insert into GatitaEmprendedora.TelefonoPersonalOrganizador (RFC, TelefonoPersonalOrganizador) values (null, '7333950053');
295 insert into GatitaEmprendedora.TelefonoPersonalOrganizador (RFC, TelefonoPersonalOrganizador) values (null, '5687823761');
296 insert into GatitaEmprendedora.TelefonoPersonalOrganizador (RFC, TelefonoPersonalOrganizador) values (null, '7342287536');
297 insert into GatitaEmprendedora.TelefonoPersonalOrganizador (RFC, TelefonoPersonalOrganizador) values (null, '3969757478');
298 insert into GatitaEmprendedora.TelefonoPersonalOrganizador (RFC, TelefonoPersonalOrganizador) values (null, '6913802985');
299 insert into GatitaEmprendedora.TelefonoPersonalOrganizador (RFC, TelefonoPersonalOrganizador) values (null, '8797873608');
300 insert into GatitaEmprendedora.TelefonoPersonalOrganizador (RFC, TelefonoPersonalOrganizador) values (null, '2738970014');
301

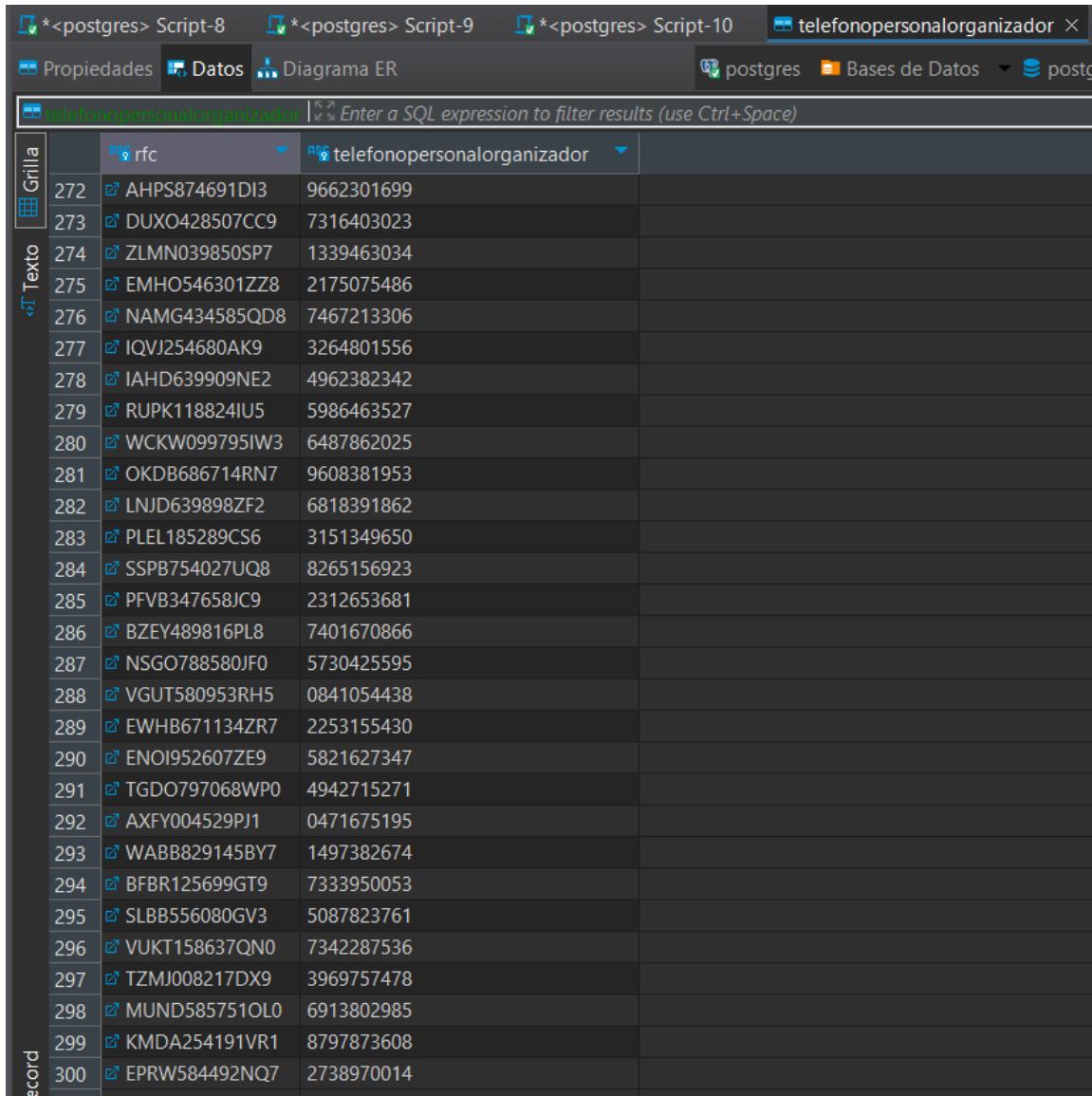
```

```

File Edit Selection View Go Run Terminal Help ← → FundamentosBD
EXPLORER ... DDLsql GatitaEmprendedora.TelefonoPersonalOrganizador.sql GatitaEmprendedora.PersonalOrganizador.sql DML.sql
FUNDAMENTOSBD
> Prácticas
> ProyectoFinal_NAMEis...
> Diagramas
> Docs
> SQL
DDL.sql
disparadores_funcio...
DML.sql
Correcciones.md
Reparto.md
Tareas
Lineamientos_FBD.pdf
Links.md
README_NAMEisNULL...
 README.md
266 insert into GatitaEmprendedora.TelefonoPersonalOrganizador (RFC, TelefonoPersonalOrganizador) values ('VTC0288920901', '3148121933');
267 insert into GatitaEmprendedora.TelefonoPersonalOrganizador (RFC, TelefonoPersonalOrganizador) values ('CQWU885981AD08', '8934734525');
268 insert into GatitaEmprendedora.TelefonoPersonalOrganizador (RFC, TelefonoPersonalOrganizador) values ('SVHI381947FF8', '3455223834');
269 insert into GatitaEmprendedora.TelefonoPersonalOrganizador (RFC, TelefonoPersonalOrganizador) values ('OMPP161394UG7', '4986054226');
270 insert into GatitaEmprendedora.TelefonoPersonalOrganizador (RFC, TelefonoPersonalOrganizador) values ('CYXJ764160001', '8718768990');
271 insert into GatitaEmprendedora.TelefonoPersonalOrganizador (RFC, TelefonoPersonalOrganizador) values ('RKKD046655HF2', '3403370274');
272 insert into GatitaEmprendedora.TelefonoPersonalOrganizador (RFC, TelefonoPersonalOrganizador) values ('AHP5874691D13', '9662301699');
273 insert into GatitaEmprendedora.TelefonoPersonalOrganizador (RFC, TelefonoPersonalOrganizador) values ('DUX0428597CC9', '7316403023');
274 insert into GatitaEmprendedora.TelefonoPersonalOrganizador (RFC, TelefonoPersonalOrganizador) values ('ZLNM099859SP7', '1339463034');
275 insert into GatitaEmprendedora.TelefonoPersonalOrganizador (RFC, TelefonoPersonalOrganizador) values ('EMH0546301728', '2175075486');
276 insert into GatitaEmprendedora.TelefonoPersonalOrganizador (RFC, TelefonoPersonalOrganizador) values ('NAMG434585Q08', '7467213306');
277 insert into GatitaEmprendedora.TelefonoPersonalOrganizador (RFC, TelefonoPersonalOrganizador) values ('IQVJ254680AK9', '3264801556');
278 insert into GatitaEmprendedora.TelefonoPersonalOrganizador (RFC, TelefonoPersonalOrganizador) values ('IAHD639909NE2', '4962382342');
279 insert into GatitaEmprendedora.TelefonoPersonalOrganizador (RFC, TelefonoPersonalOrganizador) values ('RUPK1188241U5', '5986463527');
280 insert into GatitaEmprendedora.TelefonoPersonalOrganizador (RFC, TelefonoPersonalOrganizador) values ('WCKM099751W3', '6487862025');
281 insert into GatitaEmprendedora.TelefonoPersonalOrganizador (RFC, TelefonoPersonalOrganizador) values ('OKDB686714RN7', '9608381953');
282 insert into GatitaEmprendedora.TelefonoPersonalOrganizador (RFC, TelefonoPersonalOrganizador) values ('LNJD03989272', '6818391862');
283 insert into GatitaEmprendedora.TelefonoPersonalOrganizador (RFC, TelefonoPersonalOrganizador) values ('PLEL185289CS6', '3151349650');
284 insert into GatitaEmprendedora.TelefonoPersonalOrganizador (RFC, TelefonoPersonalOrganizador) values ('SSPB754027U08', '8265156923');
285 insert into GatitaEmprendedora.TelefonoPersonalOrganizador (RFC, TelefonoPersonalOrganizador) values ('PFVB3476583C9', '2312653681');
286 insert into GatitaEmprendedora.TelefonoPersonalOrganizador (RFC, TelefonoPersonalOrganizador) values ('BZEY489816PL8', '7401670866');
287 insert into GatitaEmprendedora.TelefonoPersonalOrganizador (RFC, TelefonoPersonalOrganizador) values ('NSG0788580JF0', '5730425595');
288 insert into GatitaEmprendedora.TelefonoPersonalOrganizador (RFC, TelefonoPersonalOrganizador) values ('VGUT580953R15', '0841054438');
289 insert into GatitaEmprendedora.TelefonoPersonalOrganizador (RFC, TelefonoPersonalOrganizador) values ('EWHB671134ZK7', '2253155430');
290 insert into GatitaEmprendedora.TelefonoPersonalOrganizador (RFC, TelefonoPersonalOrganizador) values ('ENOT952607Z9', '5821627347');
291 insert into GatitaEmprendedora.TelefonoPersonalOrganizador (RFC, TelefonoPersonalOrganizador) values ('TGD0797068WP0', '4942715271');
292 insert into GatitaEmprendedora.TelefonoPersonalOrganizador (RFC, TelefonoPersonalOrganizador) values ('AFXY004529P11', '0471675195');
293 insert into GatitaEmprendedora.TelefonoPersonalOrganizador (RFC, TelefonoPersonalOrganizador) values ('WAB8829145BV7', '1497382674');
294 insert into GatitaEmprendedora.TelefonoPersonalOrganizador (RFC, TelefonoPersonalOrganizador) values ('BFBR125699GT9', '7333950053');
295 insert into GatitaEmprendedora.TelefonoPersonalOrganizador (RFC, TelefonoPersonalOrganizador) values ('SLBB556080GV3', '5687823761');
296 insert into GatitaEmprendedora.TelefonoPersonalOrganizador (RFC, TelefonoPersonalOrganizador) values ('VUKT158637Q08', '7342287536');
297 insert into GatitaEmprendedora.TelefonoPersonalOrganizador (RFC, TelefonoPersonalOrganizador) values ('TMJ008217DX9', '3969757478');
298 insert into GatitaEmprendedora.TelefonoPersonalOrganizador (RFC, TelefonoPersonalOrganizador) values ('MUND05857510L0', '6913802985');
299 insert into GatitaEmprendedora.TelefonoPersonalOrganizador (RFC, TelefonoPersonalOrganizador) values ('KMDA254191VR1', '8797873608');
300 insert into GatitaEmprendedora.TelefonoPersonalOrganizador (RFC, TelefonoPersonalOrganizador) values ('EPRW584492NQ7', '2738970014');
301

```

Y así se ve la tabla poblada:

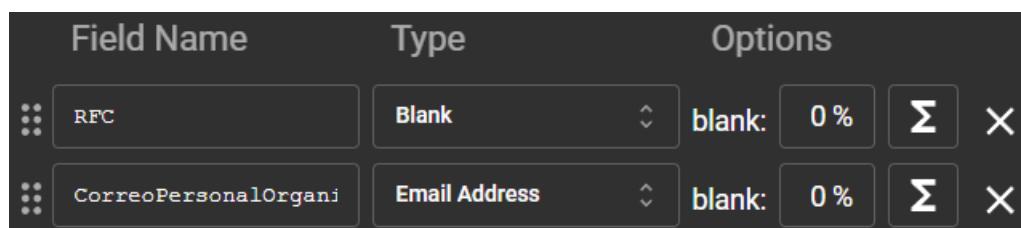


The screenshot shows the pgAdmin interface with the 'telefonopersonalorganizador' table selected. The table has two columns: 'rfc' and 'telefonopersonalorganizador'. The data consists of 300 rows, each containing a unique identifier (number from 272 to 300) and a pair of values separated by a space.

	rfc	telefonopersonalorganizador
272	AHPS874691DI3	9662301699
273	DUXO428507CC9	7316403023
274	ZLMN039850SP7	1339463034
275	EMHO546301ZZ8	2175075486
276	NAMG434585QD8	7467213306
277	IQVJ254680AK9	3264801556
278	IAHD639909NE2	4962382342
279	RUPK118824IU5	5986463527
280	WCKW099795IW3	6487862025
281	OKDB686714RN7	9608381953
282	LNJD639898ZF2	6818391862
283	PLEL185289CS6	3151349650
284	SSPB754027UQ8	8265156923
285	PFVB347658JC9	2312653681
286	BZEY489816PL8	7401670866
287	NSGO788580JF0	5730425595
288	VGUT580953RH5	0841054438
289	EWHB671134ZR7	2253155430
290	ENOI952607ZE9	5821627347
291	TGDO797068WP0	4942715271
292	AXFY004529PJ1	0471675195
293	WABB829145BY7	1497382674
294	BFBR125699GT9	7333950053
295	SLBB556080GV3	5087823761
296	VUKT158637QN0	7342287536
297	TZMJ008217DX9	3969757478
298	MUND585751OL0	6913802985
299	KMDA254191VR1	8797873608
300	EPRW584492NQ7	2738970014

### Tabla Correo Personal Organizador

Captura de Mockaroo:



The table defines the structure of the 'CorreoPersonalOrganizador' field. It has three columns: 'Field Name', 'Type', and 'Options'. The 'Field Name' column contains 'RFC' and 'CorreoPersonalOrganizador'. The 'Type' column contains 'Blank' and 'Email Address'. The 'Options' column contains 'blank: 0 %' and 'Σ X' for both rows.

Field Name	Type	Options
RFC	Blank	blank: 0 % Σ X
CorreoPersonalOrganizador	Email Address	blank: 0 % Σ X

Como se puede ver, generamos la columna de **RFC** con valores null y esto es porque Mockaroo nos genera números aleatorios cada vez, por lo que necesitamos garantizar que haya **RFCs** en la BD. Con ayuda de un editor de código (Visual Studio Code) copiamos y pegamos de manera eficiente todos datos ya existentes de la tabla **Personal Organizador**.

```

File Edit Selection View Go Run Terminal Help ⏪ ⏴ FundamentosBD
EXPLORER DDL.sql GatitaEmprendedora.Cre...sql PersonalOrganizador.sql DML.sql
C:\> Users > victo > Downloads > GatitaEmprendedora.Cre...
265 insert into GatitaEmprendedora.Cre...
266 insert into GatitaEmprendedora.Cre...
267 insert into GatitaEmprendedora.Cre...
268 insert into GatitaEmprendedora.Cre...
269 insert into GatitaEmprendedora.Cre...
270 insert into GatitaEmprendedora.Cre...
271 insert into GatitaEmprendedora.Cre...
272 insert into GatitaEmprendedora.Cre...
273 insert into GatitaEmprendedora.Cre...
274 insert into GatitaEmprendedora.Cre...
275 insert into GatitaEmprendedora.Cre...
276 insert into GatitaEmprendedora.Cre...
277 insert into GatitaEmprendedora.Cre...
278 insert into GatitaEmprendedora.Cre...
279 insert into GatitaEmprendedora.Cre...
280 insert into GatitaEmprendedora.Cre...
281 insert into GatitaEmprendedora.Cre...
282 insert into GatitaEmprendedora.Cre...
283 insert into GatitaEmprendedora.Cre...
284 insert into GatitaEmprendedora.Cre...
285 insert into GatitaEmprendedora.Cre...
286 insert into GatitaEmprendedora.Cre...
287 insert into GatitaEmprendedora.Cre...
288 insert into GatitaEmprendedora.Cre...
289 insert into GatitaEmprendedora.Cre...
290 insert into GatitaEmprendedora.Cre...
291 insert into GatitaEmprendedora.Cre...
292 insert into GatitaEmprendedora.Cre...
293 insert into GatitaEmprendedora.Cre...
294 insert into GatitaEmprendedora.Cre...
295 insert into GatitaEmprendedora.Cre...
296 insert into GatitaEmprendedora.Cre...
297 insert into GatitaEmprendedora.Cre...
298 insert into GatitaEmprendedora.Cre...
299 insert into GatitaEmprendedora.Cre...
300 insert into GatitaEmprendedora.Cre...

```

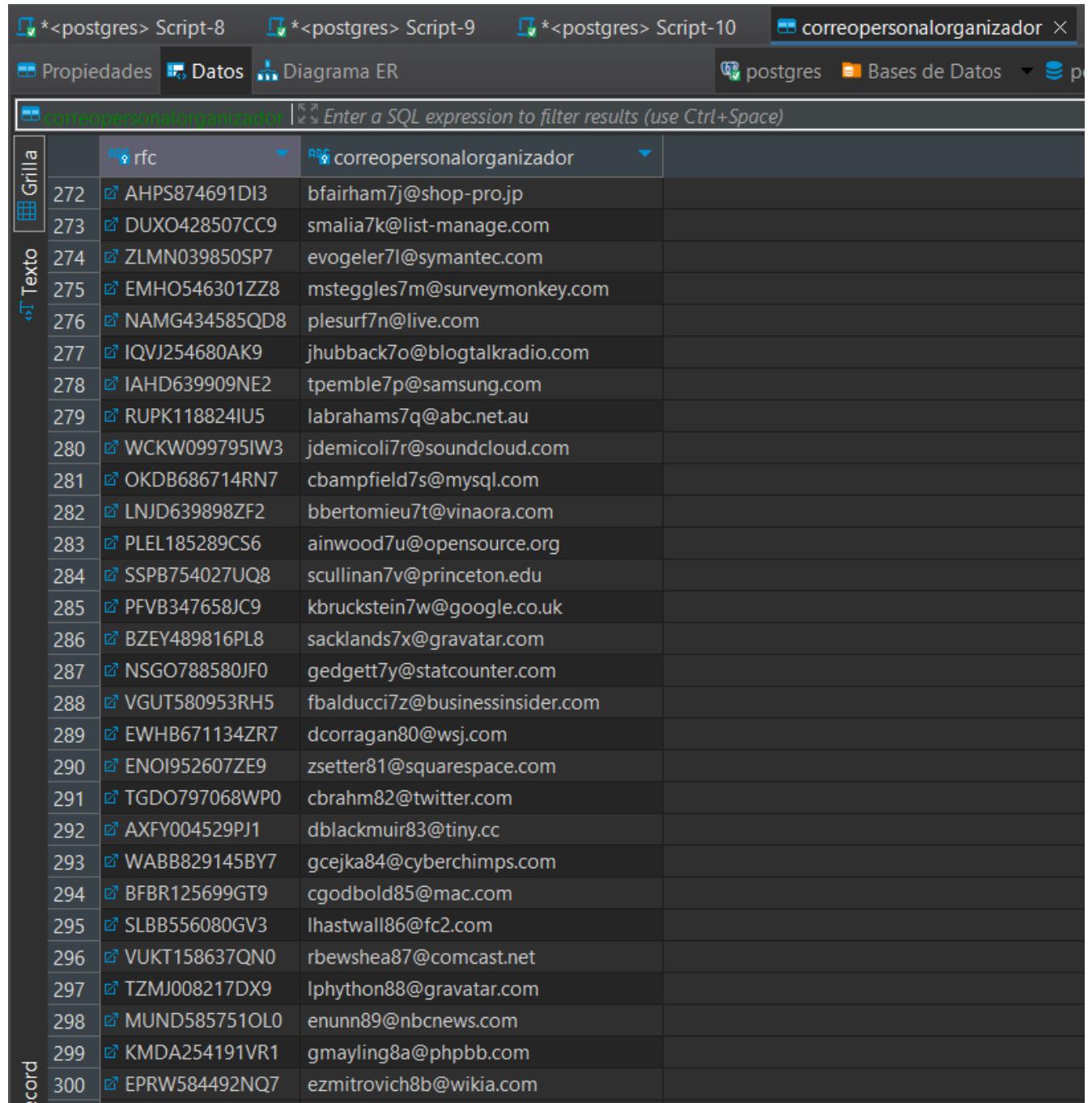
  

```

File Edit Selection View Go Run Terminal Help ⏪ ⏴ FundamentosBD
EXPLORER DDL.sql GatitaEmprendedora.Cre...sql PersonalOrganizador.sql DML.sql
C:\> Users > victo > Downloads > GatitaEmprendedora.Cre...
265 insert into GatitaEmprendedora.Cre...
266 insert into GatitaEmprendedora.Cre...
267 insert into GatitaEmprendedora.Cre...
268 insert into GatitaEmprendedora.Cre...
269 insert into GatitaEmprendedora.Cre...
270 insert into GatitaEmprendedora.Cre...
271 insert into GatitaEmprendedora.Cre...
272 insert into GatitaEmprendedora.Cre...
273 insert into GatitaEmprendedora.Cre...
274 insert into GatitaEmprendedora.Cre...
275 insert into GatitaEmprendedora.Cre...
276 insert into GatitaEmprendedora.Cre...
277 insert into GatitaEmprendedora.Cre...
278 insert into GatitaEmprendedora.Cre...
279 insert into GatitaEmprendedora.Cre...
280 insert into GatitaEmprendedora.Cre...
281 insert into GatitaEmprendedora.Cre...
282 insert into GatitaEmprendedora.Cre...
283 insert into GatitaEmprendedora.Cre...
284 insert into GatitaEmprendedora.Cre...
285 insert into GatitaEmprendedora.Cre...
286 insert into GatitaEmprendedora.Cre...
287 insert into GatitaEmprendedora.Cre...
288 insert into GatitaEmprendedora.Cre...
289 insert into GatitaEmprendedora.Cre...
290 insert into GatitaEmprendedora.Cre...
291 insert into GatitaEmprendedora.Cre...
292 insert into GatitaEmprendedora.Cre...
293 insert into GatitaEmprendedora.Cre...
294 insert into GatitaEmprendedora.Cre...
295 insert into GatitaEmprendedora.Cre...
296 insert into GatitaEmprendedora.Cre...
297 insert into GatitaEmprendedora.Cre...
298 insert into GatitaEmprendedora.Cre...
299 insert into GatitaEmprendedora.Cre...
300 insert into GatitaEmprendedora.Cre...

```

Y así se ve la tabla poblada:

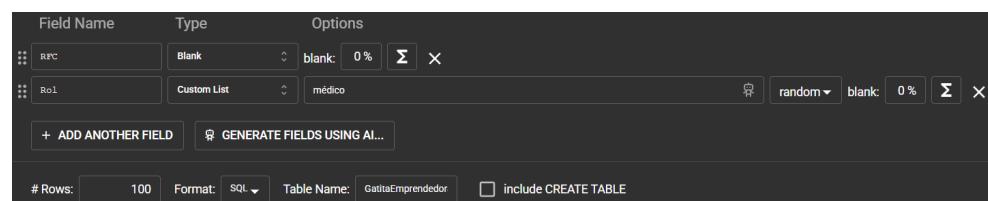


The screenshot shows the pgAdmin interface with the database 'correopersonalorganizador' selected. The table 'correopersonalorganizador' is displayed with 300 rows. The columns are 'rfc' and 'correo'. The 'rfc' column contains values like AHPS874691DI3, DUXO428507CC9, ZLMN039850SP7, etc., and the 'correo' column contains email addresses such as bfairham7j@shop-pro.jp, smalia7k@list-manage.com, evogeler7l@symantec.com, etc.

Grilla	rfc	correo
272	AHPS874691DI3	bfairham7j@shop-pro.jp
273	DUXO428507CC9	smalia7k@list-manage.com
274	ZLMN039850SP7	evogeler7l@symantec.com
275	EMHO546301ZZ8	msteggles7m@surveymonkey.com
276	NAMG434585QD8	plesurf7n@live.com
277	IQVJ254680AK9	jhubback7o@blogtalkradio.com
278	IAHD639909NE2	tpemble7p@samsung.com
279	RUPK118824IU5	labrahams7q@abc.net.au
280	WCKW099795IW3	jdemicoli7r@soundcloud.com
281	OKDB686714RN7	cbampfield7s@mysql.com
282	LNJD639898ZF2	bbertomieu7t@vinaora.com
283	PLEL185289CS6	ainwood7u@opensource.org
284	SSPB754027UQ8	scullinan7v@princeton.edu
285	PFVB347658JC9	kbrickstein7w@google.co.uk
286	BZEY489816PL8	sacklands7x@gravatar.com
287	NSGO788580JF0	gedgett7y@statcounter.com
288	VGUT580953RH5	fbalducci7z@businessinsider.com
289	EWHB671134ZR7	dcorragan80@wsj.com
290	ENOI952607ZE9	zsetter81@squarespace.com
291	TGDO797068WP0	cbrahm82@twitter.com
292	AXFY004529PJ1	dblackmuir83@tiny.cc
293	WABB829145BY7	gcejka84@cyberchimps.com
294	BFBR125699GT9	cgodbold85@mac.com
295	SLBB556080GV3	lhastwall86@fc2.com
296	VUKT158637QN0	rbewshea87@comcast.net
297	TZMJ008217DX9	lpython88@gravatar.com
298	MUND585751OL0	enunn89@nbcnews.com
299	KMDA254191VR1	gmayling8a@phpbb.com
300	EPRW584492NQ7	ezmitrovich8b@wikia.com

### Tabla Rol Personal Organizador

Captura de Mockaroo:



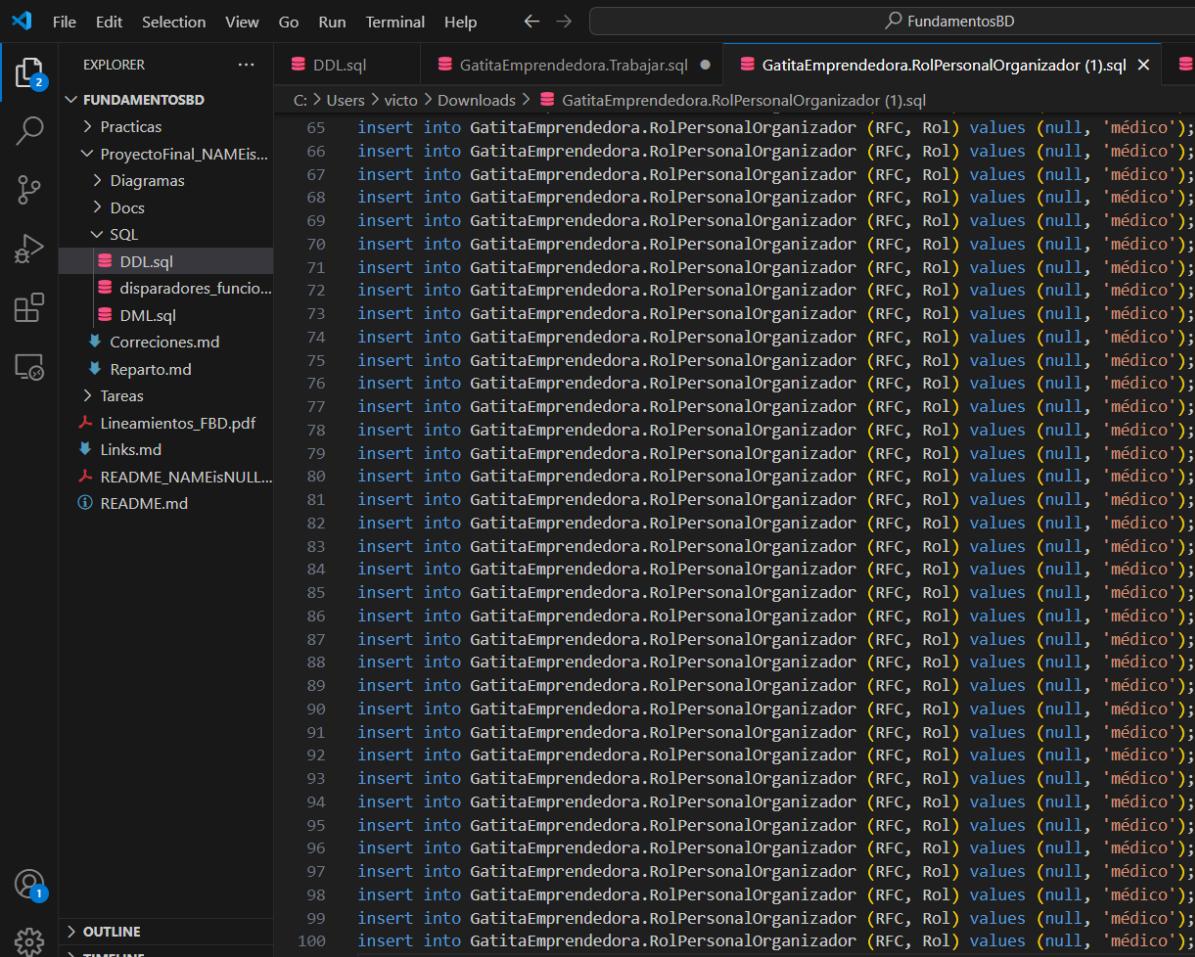
The screenshot shows the Mockaroo interface for generating data. It displays two fields: 'rfc' (Type: Blank, Options: blank: 0 %) and 'Rol' (Type: Custom List, Options: médico). Below the fields are buttons for '+ ADD ANOTHER FIELD' and 'GENERATE FIELDS USING AI...'. At the bottom, there are settings for '# Rows: 100', 'Format: SQL', 'Table Name: GatitaEmprendedor', and 'include CREATE TABLE'.

Field Name	Type	Options
RFC	Blank	blank: 0% $\Sigma$ X
Rol	Custom List	seguridad random $\downarrow$ blank: 0% $\Sigma$ X
<a href="#">+ ADD ANOTHER FIELD</a> <a href="#">GENERATE FIELDS USING AI...</a>		
# Rows:	100	Format: SQL <input type="button" value="SQL"/>
Table Name: GatitaEmprendedor <input type="checkbox"/> include CREATE TABLE		

Field Name	Type	Options
RFC	Blank	blank: 0% $\Sigma$ X
Rol	Custom List	limpieza random $\downarrow$ blank: 0% $\Sigma$ X
<a href="#">+ ADD ANOTHER FIELD</a> <a href="#">GENERATE FIELDS USING AI...</a>		
# Rows:	100	Format: SQL <input type="button" value="SQL"/>
Table Name: GatitaEmprendedor <input type="checkbox"/> include CREATE TABLE		

Como se puede ver, para cada rol generamos la columna de **RFC** con valores null y esto es porque Mockaroo nos genera números aleatorios cada vez, por lo que necesitamos garantizar que haya **RFCs** en la BD. Con ayuda de un editor de código (Visual Studio Code) copiamos y pegamos de manera eficiente todos datos ya existentes de la tabla **Personal Organizador** para cada tabla de rol que hicimos.



The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer (Left):** Shows a folder named "FUNDAMENTOSBD" containing subfolders "Prácticas", "ProyectoFinal\_NAMEisNULL...", "Diagramas", "Docs", and "SQL". Inside the "SQL" folder, there are files "DDL.sql", "disparadores\_funcio...", "DMLsql", "Correcciones.md", "Reparto.md", "Tareas", "Lineamientos\_FBD.pdf", "Links.md", and "README\_NAMEisNULL...".
- Code Editor (Right):** Displays the content of the "GatitaEmprendedora.RolPersonalOrganizador (1).sql" file. The code consists of 100 lines of SQL insert statements, each inserting a new row into the "RolPersonalOrganizador" table with columns "RFC" and "Rol". The "RFC" column is set to null and the "Rol" column is set to 'médico'. The file path shown in the status bar is "C:\Users>victo\Downloads\GatitaEmprendedora.RolPersonalOrganizador (1).sql".

```

File Edit Selection View Go Run Terminal Help ⏪ ⏩ FundamentosBD
EXPLORER ... DDL.sql GatitaEmprendedora.Trabajar.sql ● GatitaEmprendedora.RolPersonalOrganizador (1).sql
C: > Users > victo > Downloads > GatitaEmprendedora.RolPersonalOrganizador (1).sql
65 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values (null, 'médico');
66 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values (null, 'médico');
67 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values (null, 'médico');
68 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values (null, 'médico');
69 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values (null, 'médico');
70 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values (null, 'médico');
71 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values (null, 'médico');
72 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values (null, 'médico');
73 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values (null, 'médico');
74 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values (null, 'médico');
75 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values (null, 'médico');
76 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values (null, 'médico');
77 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values (null, 'médico');
78 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values (null, 'médico');
79 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values (null, 'médico');
80 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values (null, 'médico');
81 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values (null, 'médico');
82 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values (null, 'médico');
83 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values (null, 'médico');
84 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values (null, 'médico');
85 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values (null, 'médico');
86 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values (null, 'médico');
87 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values (null, 'médico');
88 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values (null, 'médico');
89 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values (null, 'médico');
90 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values (null, 'médico');
91 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values (null, 'médico');
92 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values (null, 'médico');
93 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values (null, 'médico');
94 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values (null, 'médico');
95 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values (null, 'médico');
96 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values (null, 'médico');
97 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values (null, 'médico');
98 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values (null, 'médico');
99 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values (null, 'médico');
100 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values (null, 'médico');

```

The screenshot shows a Visual Studio Code (VS Code) interface with several tabs open in the background, indicating multiple database files are being worked on simultaneously. The main editor area displays a large SQL script for populating a table named 'RolPersonalOrganizador' with 100 rows, each containing values for 'RFC' and 'Rol' and the string 'seguridad' in the 'descripción' column.

```
C: > Users > victo > Downloads > GatitaEmprendedora.RolPersonalOrganizador (1).sql • GatitaEmprendedora.RolPersonalOrganizador (2).sql • GatitaEmprendedora.Personal
64 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values (null, 'seguridad')
65 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values (null, 'seguridad')
66 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values (null, 'seguridad')
67 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values (null, 'seguridad')
68 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values (null, 'seguridad')
69 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values (null, 'seguridad')
70 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values (null, 'seguridad')
71 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values (null, 'seguridad')
72 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values (null, 'seguridad')
73 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values (null, 'seguridad')
74 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values (null, 'seguridad')
75 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values (null, 'seguridad')
76 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values (null, 'seguridad')
77 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values (null, 'seguridad')
78 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values (null, 'seguridad')
79 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values (null, 'seguridad')
80 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values (null, 'seguridad')
81 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values (null, 'seguridad')
82 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values (null, 'seguridad')
83 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values (null, 'seguridad')
84 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values (null, 'seguridad')
85 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values (null, 'seguridad')
86 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values (null, 'seguridad')
87 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values (null, 'seguridad')
88 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values (null, 'seguridad')
89 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values (null, 'seguridad')
90 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values (null, 'seguridad')
91 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values (null, 'seguridad')
92 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values (null, 'seguridad')
93 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values (null, 'seguridad')
94 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values (null, 'seguridad')
95 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values (null, 'seguridad')
96 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values (null, 'seguridad')
97 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values (null, 'seguridad')
98 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values (null, 'seguridad')
99 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values (null, 'seguridad')
100 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values (null, 'seguridad')
```

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** On the left, under the 'FUNDAMENTOSBD' folder, there are files and folders: DDL.sql, disparadores\_funcio..., DMLsql, Correcciones.md, Reparto.md, Tareas, Lineamientos\_FBD.pdf, Links.md, README\_NAMEisNULL..., and README.md.
- Code Editor:** The main area displays a SQL script titled 'rolPersonalOrganizador (1).sql'. The script contains 100 insert statements into the 'RolPersonalOrganizador' table, all with the value 'limpieza' for the 'rol' column.
- Status Bar:** At the bottom, it shows '100' lines of code.

```

File Edit Selection View Go Run Terminal Help ← → ⚡ FundamentosBD
EXPLORER ... DDLsql GatitaEmprendedora.Trabajar.sql ● GatitaEmprendedora.RolPersonalOrganizador (1).sql ● GatitaEmprendedora.RolPersonalOrganizador (1).sql
C:\Users>victo>Downloads>GatitaEmprendedora.RolPersonalOrganizador (1).sql
64 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, RoI) values ('1QBF651221GY0', 'médico');
65 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, RoI) values ('XJCZ046954VC3', 'médico');
66 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, RoI) values ('DLWR541477ZK6', 'médico');
67 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, RoI) values ('RAOA796368LAS', 'médico');
68 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, RoI) values ('UZJC838077UU2', 'médico');
69 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, RoI) values ('YSNW750087JX3', 'médico');
70 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, RoI) values ('YCS0999434ZH7', 'médico');
71 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, RoI) values ('PCXI666990WT8', 'médico');
72 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, RoI) values ('QLD115450VF1', 'médico');
73 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, RoI) values ('UCYA953025QV4', 'médico');
74 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, RoI) values ('GGTL14314Q09', 'médico');
75 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, RoI) values ('RDIZ298459DK7', 'médico');
76 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, RoI) values ('RCKM561377IM3', 'médico');
77 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, RoI) values ('V2IE521448JK8', 'médico');
78 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, RoI) values ('GIXH891121BZ3', 'médico');
79 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, RoI) values ('XYMG712247ML3', 'médico');
80 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, RoI) values ('SLTW816188SG1', 'médico');
81 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, RoI) values ('FCXM053582ZNO', 'médico');
82 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, RoI) values ('KVD1452281IP6', 'médico');
83 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, RoI) values ('ORTE817295NU1', 'médico');
84 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, RoI) values ('JNEK111541IV1', 'médico');
85 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, RoI) values ('VPWY650191WB2', 'médico');
86 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, RoI) values ('LDLJ735973YE3', 'médico');
87 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, RoI) values ('QNWZ968158ANS', 'médico');
88 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, RoI) values ('TLGK199932OTA4', 'médico');
89 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, RoI) values ('HGSW540280NI3', 'médico');
90 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, RoI) values ('JSEI180123DNA4', 'médico');
91 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, RoI) values ('HJMD516635CP3', 'médico');
92 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, RoI) values ('PYJB248808FJ2', 'médico');
93 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, RoI) values ('WULH743775EXT', 'médico');
94 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, RoI) values ('AXPF599717DD4', 'médico');
95 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, RoI) values ('TRVW647557DG4', 'médico');
96 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, RoI) values ('DMKT169487Q82', 'médico');
97 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, RoI) values ('WZPV545358ZC2', 'médico');
98 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, RoI) values ('TLOM599841AM0', 'médico');
99 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, RoI) values ('ZIDL742597TR3', 'médico');
100 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, RoI) values ('PWLZ970086EB8', 'médico');

```

```

File Edit Selection View Go Run Terminal Help ← → ⚡ FundamentosBD
EXPLORER ... GatitaEmprendedora.Trabajar.sql ● GatitaEmprendedora.RolPersonalOrganizador (1).sql ● GatitaEmprendedora.PersonalOrganizador.sql
C:\Users>victo>Downloads>GatitaEmprendedora.RolPersonalOrganizador (2).sql
64 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, RoI) values ('LYUZ2614/80U1', 'seguridad');
65 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, RoI) values ('XVH638264YH6', 'seguridad');
66 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, RoI) values ('PGET565252ZL2', 'seguridad');
67 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, RoI) values ('HXFY598350VB5', 'seguridad');
68 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, RoI) values ('ZWLR527567XN3', 'seguridad');
69 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, RoI) values ('IZGP048510LH6', 'seguridad');
70 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, RoI) values ('LHAD628371BC5', 'seguridad');
71 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, RoI) values ('IZKH574324OK8', 'seguridad');
72 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, RoI) values ('YMF611475YH1', 'seguridad');
73 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, RoI) values ('GGBP661239MQ3', 'seguridad');
74 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, RoI) values ('HYRG815810NV7', 'seguridad');
75 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, RoI) values ('TXAF879022MK8', 'seguridad');
76 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, RoI) values ('CMGH486966WL9', 'seguridad');
77 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, RoI) values ('DMYT407202XG2', 'seguridad');
78 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, RoI) values ('MVEN52343GM5', 'seguridad');
79 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, RoI) values ('AEAU357481KS1', 'seguridad');
80 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, RoI) values ('OIDH001634D19', 'seguridad');
81 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, RoI) values ('OBPE486992RU5', 'seguridad');
82 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, RoI) values ('CZAW898539AC0', 'seguridad');
83 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, RoI) values ('OKUE919882XG0', 'seguridad');
84 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, RoI) values ('DSB664785RT5', 'seguridad');
85 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, RoI) values ('BRXL257577L04', 'seguridad');
86 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, RoI) values ('BVHD153545NC7', 'seguridad');
87 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, RoI) values ('ZAER843991ZH8', 'seguridad');
88 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, RoI) values ('HNRZ775191RB5', 'seguridad');
89 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, RoI) values ('MGYB713451LK7', 'seguridad');
90 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, RoI) values ('OCEN422514XX7', 'seguridad');
91 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, RoI) values ('VLI1446911M1', 'seguridad');
92 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, RoI) values ('BYMS281268NY1', 'seguridad');
93 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, RoI) values ('ZUWM994746RX5', 'seguridad');
94 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, RoI) values ('PJLZ266065Y3', 'seguridad');
95 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, RoI) values ('HYQE162388Y55', 'seguridad');
96 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, RoI) values ('HNLG678904RN1', 'seguridad');
97 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, RoI) values ('WCP0498742R09', 'seguridad');
98 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, RoI) values ('XLKO042021NP3', 'seguridad');
99 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, RoI) values ('QZVB293388CW0', 'seguridad');
100 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, RoI) values ('BGNZ438243J33', 'seguridad');

```

```

File Edit Selection View Go Run Terminal Help ← → ○ FundamentosBD
EXPLORER C: > Users > victo > Downloads > GatitaEmprendedora.RolPersonalOrganizador(1).sql ● GatitaEmprendedora.RolPersonalOrganizador(2).sql
FUNDAMENTOSBD
> Practicas
> Diagramas
> Docs
> SQL
DDLsql
disparadores_funcionales.sql
DMLsql
Correcciones.md
Reparto.md
Tareas
Lineamientos_FBD.pdf
Links.md
README_NAMEisNULL...
READMD.md

```

The screenshot shows a code editor window with several tabs open. The main tab contains a large block of SQL code used to insert data into a table named 'rolpersonalorganizador'. The code consists of 100 INSERT statements, each specifying an 'RFC' and a 'rol' value ('médico' or 'seguridad') and a corresponding 'limpieza' value.

```

65 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values ('QAMP864570M4', 'limpieza');
66 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values ('VTC0288920H01', 'limpieza');
67 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values ('CQMU885981AD8', 'limpieza');
68 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values ('SYHI381947F8', 'limpieza');
69 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values ('CMP161394U7', 'limpieza');
70 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values ('CYXJ7641680Q1', 'limpieza');
71 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values ('RKKD049665HF2', 'limpieza');
72 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values ('AHP5874691D13', 'limpieza');
73 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values ('DUX0428507CC9', 'limpieza');
74 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values ('ZLMN039850SP7', 'limpieza');
75 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values ('EMH0546301ZB', 'limpieza');
76 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values ('NAMG434585QD8', 'limpieza');
77 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values ('IQVJ254680AK9', 'limpieza');
78 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values ('IAHD63999NE2', 'limpieza');
79 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values ('RUPK118824IU5', 'limpieza');
80 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values ('WCKW099795IW3', 'limpieza');
81 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values ('OKDB686714RN7', 'limpieza');
82 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values ('LNJD639898ZF2', 'limpieza');
83 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values ('PLEL185289CS6', 'limpieza');
84 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values ('SSPB754027UQ8', 'limpieza');
85 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values ('PFBV347658JC9', 'limpieza');
86 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values ('BZEY489816PL8', 'limpieza');
87 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values ('NSG0788580F0', 'limpieza');
88 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values ('VGUT580953RH5', 'limpieza');
89 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values ('EWHB671134ZR7', 'limpieza');
90 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values ('ENO1952607Z9', 'limpieza');
91 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values ('TG0D797068P0', 'limpieza');
92 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values ('AXFY004529PJ1', 'limpieza');
93 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values ('WABB829145BY7', 'limpieza');
94 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values ('BFBR125699GT9', 'limpieza');
95 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values ('SLBB556080GV3', 'limpieza');
96 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values ('VUKT158637NO', 'limpieza');
97 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values ('IZM008217DX9', 'limpieza');
98 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values ('MUND585751OL0', 'limpieza');
99 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values ('KMDA254191VR1', 'limpieza');
100 insert into GatitaEmprendedora.RolPersonalOrganizador (RFC, Rol) values ('EPRW584492NQ7', 'limpieza');

```

Y así se ve la tabla poblada ya completa:

The screenshot shows a PostgreSQL database interface with multiple tabs open. The 'Datos' tab is active, displaying a table named 'rolpersonalorganizador' with two columns: 'rfc' and 'rol'. The table contains 102 rows of data, where 'rfc' values are listed vertically and 'rol' values are either 'médico' or 'seguridad'.

	rfc	rol
73	UCYA953025QV4	médico
74	GGLT714314OQ9	médico
75	RDUZ298459DK7	médico
76	RCKM361377IM3	médico
77	VZJE521448JK8	médico
78	GIXH891121BZ3	médico
79	XYMG712247ML3	médico
80	SLTW816188SG1	médico
81	FCXM053582ZN0	médico
82	KVDJ452281IP6	médico
83	ORTE817295NU1	médico
84	JNEK111541IV1	médico
85	VPWY650191WB2	médico
86	LDLJ735973YE3	médico
87	QNNZ968158AN5	médico
88	TLGK199932OT4	médico
89	HGSW540280NI3	médico
90	JSEI180123DN4	médico
91	HJWD516635CP3	médico
92	PYJB248080FJ2	médico
93	WULH743775EX7	médico
94	AXPF599717DD4	médico
95	TRVV647557DG4	médico
96	DMKT169487QZB	médico
97	WZPV545358ZC2	médico
98	TLOM599841AN0	médico
99	ZIDL742597TR3	médico
100	PWLZ97008EB8	médico
101	LNYM346606AC0	seguridad
102	MUEC770752QE1	seguridad

The screenshot shows two identical database tables named 'rolpersonalorganizador' in a PostgreSQL environment. Both tables have two columns: 'rfc' and 'rol'. The data in both tables is identical, showing various employee IDs (RFCs) and their assigned roles ('rol').

	rfc	rol
172	YWFD611475YH1	seguridad
173	GGBP661230MQ3	seguridad
174	HYRG815810NV7	seguridad
175	TXAF879022MK8	seguridad
176	CWGH486966WL9	seguridad
177	DMYT407202XG2	seguridad
178	WVEN523436ML5	seguridad
179	AEAU357481KS1	seguridad
180	OIDH001634DI9	seguridad
181	OBPE486922RU5	seguridad
182	CZAW898539AC0	seguridad
183	DKUE919882XG0	seguridad
184	DSSB604785RT5	seguridad
185	BRLX257577LO4	seguridad
186	BVHD153545NC7	seguridad
187	ZAER843991ZH8	seguridad
188	HNRZ775191RB5	seguridad
189	MGYB713451IK7	seguridad
190	OCEN422514XX7	seguridad
191	VJL446911MJ1	seguridad
192	BYMS281268NY1	seguridad
193	ZUVM994746RX5	seguridad
194	PJLZ266065YY3	seguridad
195	HYQE162388YG5	seguridad
196	HNLA678904RN1	seguridad
197	WCP0498742RD9	seguridad
198	XLK0042021NP3	seguridad
199	QZVB293388CW0	seguridad
200	BGNZ438243JB3	seguridad
201	LCCX964438BK8	limpieza
272	AHPS874691D13	limpieza
273	DUXO428507C9	limpieza
274	ZLMN039850SP7	limpieza
275	EMHO546301ZZ8	limpieza
276	NAMG434585QD8	limpieza
277	IQVJ254680AK9	limpieza
278	IAHD639909NE2	limpieza
279	RUPK118824IU5	limpieza
280	WCKW099795IW3	limpieza
281	OKDB686714RN7	limpieza
282	LNJD639898ZF2	limpieza
283	PLEL185289CS6	limpieza
284	SSPB754027UQ8	limpieza
285	PFVB347658JC9	limpieza
286	BZEY489816PL8	limpieza
287	NSGO788580JF0	limpieza
288	VGUT580953RH5	limpieza
289	EWHB671134ZR7	limpieza
290	ENOI952607ZE9	limpieza
291	TGDO797068WP0	limpieza
292	AXFY004529PJ1	limpieza
293	WABB829145BY7	limpieza
294	BFBR125699GT9	limpieza
295	SLBB556080GV3	limpieza
296	VUKT158637QN0	limpieza
297	TZMJ008217DX9	limpieza
298	MUND585751OL0	limpieza
299	KMDA254191VR1	limpieza
300	EPRW584492NQ7	limpieza

Entonces, con las tablas **Cliente** y **Bazar** pobladas, ya podemos poblar **Trabajar**.

### Tabla Trabajar

Captura de Mockaroo:

Field Name	Type	Options
IdTrabajar	Row Number	blank: 0 % $\Sigma$ X
IdBazar	Row Number	blank: 0 % $\Sigma$ X
RFC	Blank	blank: 0 % $\Sigma$ X
FechaAsistencia	Datetime	05/31/2024 <input type="button" value="Calendar"/> to 05/31/2024 <input type="button" value="Calendar"/> format: m/d/yyyy <input type="checkbox"/> blank: 0 % $\Sigma$ X
<a href="#">+ ADD ANOTHER FIELD</a>		<a href="#">GENERATE FIELDS USING AI...</a>
# Rows: 950		Format: SQL <input type="checkbox"/> Table Name: GatitaEmprendedor <input type="checkbox"/> include CREATE TABLE

Como se puede ver, tenemos una restricción, de la siguiente manera, esto es para garantizar tener cada bazar 3 veces (para que cada uno tenga al menos un personal de cada rol):

```
if this % 300 == 0 then
    300
else
    this % 300
end
```

Ahora bien, notemos cómo nos genera el código Mockaroo, null en la columna de RFC y la misma fecha en la que estarían trabajando todos los empleados:

```

File Edit Selection View Go Run Terminal Help ⏎ ↻ 🔍 FundamentosBD
C:\Users>victo>Downloads>GatitaEmprendedora.Trabajador (1).sql
915 insert into GatitaEmprendedora.Trabajador (IdTrabajador, idBazar, RFC, FechaAsistencia) values (915, 15, null, '5/31/2024');
916 insert into GatitaEmprendedora.Trabajador (IdTrabajador, idBazar, RFC, FechaAsistencia) values (916, 16, null, '5/31/2024');
917 insert into GatitaEmprendedora.Trabajador (IdTrabajador, idBazar, RFC, FechaAsistencia) values (917, 17, null, '5/31/2024');
918 insert into GatitaEmprendedora.Trabajador (IdTrabajador, idBazar, RFC, FechaAsistencia) values (918, 18, null, '5/31/2024');
919 insert into GatitaEmprendedora.Trabajador (IdTrabajador, idBazar, RFC, FechaAsistencia) values (919, 19, null, '5/31/2024');
920 insert into GatitaEmprendedora.Trabajador (IdTrabajador, idBazar, RFC, FechaAsistencia) values (920, 20, null, '5/31/2024');
921 insert into GatitaEmprendedora.Trabajador (IdTrabajador, idBazar, RFC, FechaAsistencia) values (921, 21, null, '5/31/2024');
922 insert into GatitaEmprendedora.Trabajador (IdTrabajador, idBazar, RFC, FechaAsistencia) values (922, 22, null, '5/31/2024');
923 insert into GatitaEmprendedora.Trabajador (IdTrabajador, idBazar, RFC, FechaAsistencia) values (923, 23, null, '5/31/2024');
924 insert into GatitaEmprendedora.Trabajador (IdTrabajador, idBazar, RFC, FechaAsistencia) values (924, 24, null, '5/31/2024');
925 insert into GatitaEmprendedora.Trabajador (IdTrabajador, idBazar, RFC, FechaAsistencia) values (925, 25, null, '5/31/2024');
926 insert into GatitaEmprendedora.Trabajador (IdTrabajador, idBazar, RFC, FechaAsistencia) values (926, 26, null, '5/31/2024');
927 insert into GatitaEmprendedora.Trabajador (IdTrabajador, idBazar, RFC, FechaAsistencia) values (927, 27, null, '5/31/2024');
928 insert into GatitaEmprendedora.Trabajador (IdTrabajador, idBazar, RFC, FechaAsistencia) values (928, 28, null, '5/31/2024');
929 insert into GatitaEmprendedora.Trabajador (IdTrabajador, idBazar, RFC, FechaAsistencia) values (929, 29, null, '5/31/2024');
930 insert into GatitaEmprendedora.Trabajador (IdTrabajador, idBazar, RFC, FechaAsistencia) values (930, 30, null, '5/31/2024');
931 insert into GatitaEmprendedora.Trabajador (IdTrabajador, idBazar, RFC, FechaAsistencia) values (931, 31, null, '5/31/2024');
932 insert into GatitaEmprendedora.Trabajador (IdTrabajador, idBazar, RFC, FechaAsistencia) values (932, 32, null, '5/31/2024');
933 insert into GatitaEmprendedora.Trabajador (IdTrabajador, idBazar, RFC, FechaAsistencia) values (933, 33, null, '5/31/2024');
934 insert into GatitaEmprendedora.Trabajador (IdTrabajador, idBazar, RFC, FechaAsistencia) values (934, 34, null, '5/31/2024');
935 insert into GatitaEmprendedora.Trabajador (IdTrabajador, idBazar, RFC, FechaAsistencia) values (935, 35, null, '5/31/2024');
936 insert into GatitaEmprendedora.Trabajador (IdTrabajador, idBazar, RFC, FechaAsistencia) values (936, 36, null, '5/31/2024');
937 insert into GatitaEmprendedora.Trabajador (IdTrabajador, idBazar, RFC, FechaAsistencia) values (937, 37, null, '5/31/2024');
938 insert into GatitaEmprendedora.Trabajador (IdTrabajador, idBazar, RFC, FechaAsistencia) values (938, 38, null, '5/31/2024');
939 insert into GatitaEmprendedora.Trabajador (IdTrabajador, idBazar, RFC, FechaAsistencia) values (939, 39, null, '5/31/2024');
940 insert into GatitaEmprendedora.Trabajador (IdTrabajador, idBazar, RFC, FechaAsistencia) values (940, 40, null, '5/31/2024');
941 insert into GatitaEmprendedora.Trabajador (IdTrabajador, idBazar, RFC, FechaAsistencia) values (941, 41, null, '5/31/2024');
942 insert into GatitaEmprendedora.Trabajador (IdTrabajador, idBazar, RFC, FechaAsistencia) values (942, 42, null, '5/31/2024');
943 insert into GatitaEmprendedora.Trabajador (IdTrabajador, idBazar, RFC, FechaAsistencia) values (943, 43, null, '5/31/2024');
944 insert into GatitaEmprendedora.Trabajador (IdTrabajador, idBazar, RFC, FechaAsistencia) values (944, 44, null, '5/31/2024');
945 insert into GatitaEmprendedora.Trabajador (IdTrabajador, idBazar, RFC, FechaAsistencia) values (945, 45, null, '5/31/2024');
946 insert into GatitaEmprendedora.Trabajador (IdTrabajador, idBazar, RFC, FechaAsistencia) values (946, 46, null, '5/31/2024');
947 insert into GatitaEmprendedora.Trabajador (IdTrabajador, idBazar, RFC, FechaAsistencia) values (947, 47, null, '5/31/2024');
948 insert into GatitaEmprendedora.Trabajador (IdTrabajador, idBazar, RFC, FechaAsistencia) values (948, 48, null, '5/31/2024');
949 insert into GatitaEmprendedora.Trabajador (IdTrabajador, idBazar, RFC, FechaAsistencia) values (949, 49, null, '5/31/2024');
950 insert into GatitaEmprendedora.Trabajador (IdTrabajador, idBazar, RFC, FechaAsistencia) values (950, 50, null, '5/31/2024');

```

Por lo que, con ayuda de un editor de código (Visual Studio Code) copiamos y pegamos de manera eficiente todos datos ya existentes de la tabla **Personal Organizador** y también para darle variedad a esta tabla.

```

C: > Users > victo > Downloads > GatitaEmprendedora.Trabajar (1).sql
867 insert into GatitaEmprendedora.Trabajar (IdTrabajador, IdBazar, RFC, FechaAsistencia) values (867, 267, 'CQI08033981J06', '5/31/2024')
868 insert into GatitaEmprendedora.Trabajar (IdTrabajador, IdBazar, RFC, FechaAsistencia) values (868, 268, 'SYHI381947FF8', '5/31/2024')
869 insert into GatitaEmprendedora.Trabajar (IdTrabajador, IdBazar, RFC, FechaAsistencia) values (869, 269, 'CMPP161394UG7', '5/31/2024')
870 insert into GatitaEmprendedora.Trabajar (IdTrabajador, IdBazar, RFC, FechaAsistencia) values (870, 270, 'CYXJ7641600Q1', '5/31/2024')
871 insert into GatitaEmprendedora.Trabajar (IdTrabajador, IdBazar, RFC, FechaAsistencia) values (871, 271, 'RKKD496655HF2', '5/31/2024')
872 insert into GatitaEmprendedora.Trabajar (IdTrabajador, IdBazar, RFC, FechaAsistencia) values (872, 272, 'AHP5874691D13', '5/31/2024')
873 insert into GatitaEmprendedora.Trabajar (IdTrabajador, IdBazar, RFC, FechaAsistencia) values (873, 273, 'DUX042850TC9', '5/31/2024')
874 insert into GatitaEmprendedora.Trabajar (IdTrabajador, IdBazar, RFC, FechaAsistencia) values (874, 274, 'ZLMN039850SP7', '5/31/2024')
875 insert into GatitaEmprendedora.Trabajar (IdTrabajador, IdBazar, RFC, FechaAsistencia) values (875, 275, 'ENH0546301Z28', '5/31/2024')
876 insert into GatitaEmprendedora.Trabajar (IdTrabajador, IdBazar, RFC, FechaAsistencia) values (876, 276, 'NAMG434585Q08', '5/31/2024')
877 insert into GatitaEmprendedora.Trabajar (IdTrabajador, IdBazar, RFC, FechaAsistencia) values (877, 277, 'IQVJ254680AK9', '5/31/2024')
878 insert into GatitaEmprendedora.Trabajar (IdTrabajador, IdBazar, RFC, FechaAsistencia) values (878, 278, 'IAHD639909NE2', '5/31/2024')
879 insert into GatitaEmprendedora.Trabajar (IdTrabajador, IdBazar, RFC, FechaAsistencia) values (879, 279, 'RUPK118824IUS', '5/31/2024')
880 insert into GatitaEmprendedora.Trabajar (IdTrabajador, IdBazar, RFC, FechaAsistencia) values (880, 280, 'WCKW099795IW3', '5/31/2024')
881 insert into GatitaEmprendedora.Trabajar (IdTrabajador, IdBazar, RFC, FechaAsistencia) values (881, 281, 'OKDB686714RN7', '5/31/2024')
882 insert into GatitaEmprendedora.Trabajar (IdTrabajador, IdBazar, RFC, FechaAsistencia) values (882, 282, 'LNJD639898ZF2', '5/31/2024')
883 insert into GatitaEmprendedora.Trabajar (IdTrabajador, IdBazar, RFC, FechaAsistencia) values (883, 283, 'PLEL185289CS6', '5/31/2024')
884 insert into GatitaEmprendedora.Trabajar (IdTrabajador, IdBazar, RFC, FechaAsistencia) values (884, 284, 'SSPB754027UQ8', '5/31/2024')
885 insert into GatitaEmprendedora.Trabajar (IdTrabajador, IdBazar, RFC, FechaAsistencia) values (885, 285, 'PFVB347658JC9', '5/31/2024')
886 insert into GatitaEmprendedora.Trabajar (IdTrabajador, IdBazar, RFC, FechaAsistencia) values (886, 286, 'BZEY489816PL8', '5/31/2024')
887 insert into GatitaEmprendedora.Trabajar (IdTrabajador, IdBazar, RFC, FechaAsistencia) values (887, 287, 'NSG07885801F09', '5/31/2024')
888 insert into GatitaEmprendedora.Trabajar (IdTrabajador, IdBazar, RFC, FechaAsistencia) values (888, 288, 'VGUT580953RH5', '5/31/2024')
889 insert into GatitaEmprendedora.Trabajar (IdTrabajador, IdBazar, RFC, FechaAsistencia) values (889, 289, 'EWHB671134ZRT', '5/31/2024')
890 insert into GatitaEmprendedora.Trabajar (IdTrabajador, IdBazar, RFC, FechaAsistencia) values (890, 290, 'ENOI952607ZE9', '5/31/2024')
891 insert into GatitaEmprendedora.Trabajar (IdTrabajador, IdBazar, RFC, FechaAsistencia) values (891, 291, 'TGDD0797068uP0', '5/31/2024')
892 insert into GatitaEmprendedora.Trabajar (IdTrabajador, IdBazar, RFC, FechaAsistencia) values (892, 292, 'AXFY004529P11', '5/31/2024')
893 insert into GatitaEmprendedora.Trabajar (IdTrabajador, IdBazar, RFC, FechaAsistencia) values (893, 293, 'WABB829145BY7', '5/31/2024')
894 insert into GatitaEmprendedora.Trabajar (IdTrabajador, IdBazar, RFC, FechaAsistencia) values (894, 294, 'BFBR125699GT9', '5/31/2024')
895 insert into GatitaEmprendedora.Trabajar (IdTrabajador, IdBazar, RFC, FechaAsistencia) values (895, 295, 'SLBB556080GV3', '5/31/2024')
896 insert into GatitaEmprendedora.Trabajar (IdTrabajador, IdBazar, RFC, FechaAsistencia) values (896, 296, 'VUKT11586370N6', '5/31/2024')
897 insert into GatitaEmprendedora.Trabajar (IdTrabajador, IdBazar, RFC, FechaAsistencia) values (897, 297, 'TZM008217DX9', '5/31/2024')
898 insert into GatitaEmprendedora.Trabajar (IdTrabajador, IdBazar, RFC, FechaAsistencia) values (898, 298, 'MUND585751OL0', '5/31/2024')
899 insert into GatitaEmprendedora.Trabajar (IdTrabajador, IdBazar, RFC, FechaAsistencia) values (899, 299, 'KNDAA254191VRL', '5/31/2024')
900 insert into GatitaEmprendedora.Trabajar (IdTrabajador, IdBazar, RFC, FechaAsistencia) values (900, 300, 'EPRW584492HQ7', '5/31/2024')

```

Finalmente, le pedimos a mockaroo que nos generara un script para garantizar que las fechas en las que una persona parte de la organización, trabaje cuando cada bazar esté pasando de la siguiente manera:

Field Name	Type	Options
id	Row Number	blank: 0 % <span style="border: 1px solid #ccc; padding: 2px;">Σ</span> <span style="border: 1px solid #ccc; padding: 2px;">×</span>
f	SQL Expression	example: DEFAULT blank: 0 % <span style="border: 1px solid #ccc; padding: 2px;">Σ</span> <span style="border: 1px solid #ccc; padding: 2px;">×</span>

Con las restricciones de la siguiente manera y en ese orden:

```

if this % 300 == 0 then
    300
else
    this % 300
end

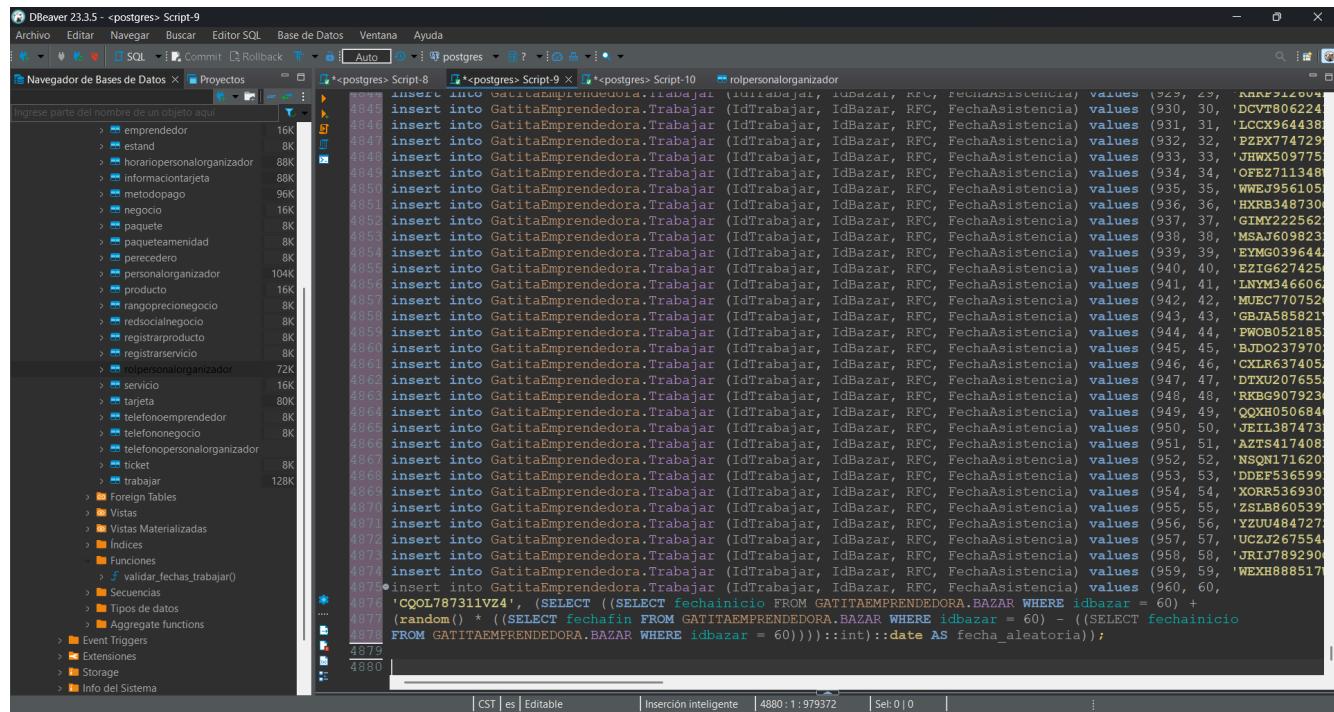
```

```

concat('
        (SELECT fechainicio FROM GATITAEMPRENDEDORA.BAZAR WHERE idbazar = ',
        field('id'),
        ') + (random() * ((SELECT fechafin FROM GATITAEMPRENDEDORA.BAZAR WHERE idbazar = ',
        field('id'), ') - ((SELECT fechainicio FROM GATITAEMPRENDEDORA.BAZAR WHERE idbazar = ',
        field('id'), '))))::int)::date AS fecha_aleatoria)'

```

Por lo que, el script final para insertar estos datos es:



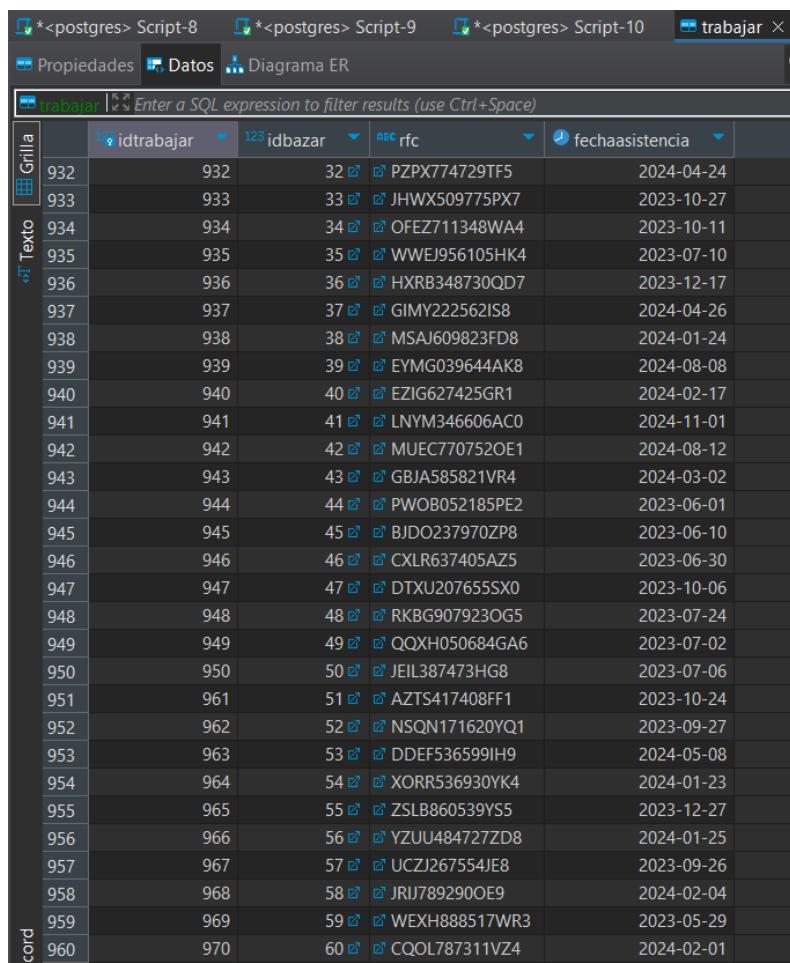
```

DBeaver 23.3.5 - <postgres> Script-9
Archivo Editor Navegar Buscar Editor SQL Base de Datos Ventana Ayuda
Navegador de Bases de Datos Proyectos
Ingrésese parte del nombre de un objeto aquí
    > emprendedor 16K
    > estand 8K
    > horariopersonalorganizador 88K
    > informaciontarjeta 88K
    > metodopago 96K
    > negocio 16K
    > paquete 8K
    > paqueteamenidad 8K
    > percedero 8K
    > personalorganizador 104K
    > producto 16K
    > rangoprecionegocio 8K
    > redsocialnegocio 8K
    > registrarproducto 8K
    > registrarservicio 8K
    > tipopersonalorganizador 72K
    > servicio 16K
    > tarjeta 80K
    > telefonoemprendedor 8K
    > telefononegocio 8K
    > telefonopersonalorganizador 8K
    > ticket 8K
    > trabajar 128K
        > Foreign Tables
        > Vista
        > Vistas Materializadas
        > Índices
        > Funciones
            > validar_fechas_trabajar()
        > Secuencias
        > Tipos de datos
        > Aggregate functions
    > Event Triggers
    > Extensiones
    > Storage
    > Info del Sistema

*<postgres> Script-8 *<postgres> Script-9 *<postgres> Script-10 rolpersonalorganizador
4845 insert into GatitaEmprendedora.Trabajar (Idtrabajador, Idbazar, RFC, FechaAsistencia) values (929, 29, 'RNH9512694')
4846 insert into GatitaEmprendedora.Trabajar (Idtrabajador, Idbazar, RFC, FechaAsistencia) values (930, 30, 'DCVTB06224')
4847 insert into GatitaEmprendedora.Trabajar (Idtrabajador, Idbazar, RFC, FechaAsistencia) values (931, 31, 'LCCX964438')
4848 insert into GatitaEmprendedora.Trabajar (Idtrabajador, Idbazar, RFC, FechaAsistencia) values (932, 32, 'PZPX774729')
4849 insert into GatitaEmprendedora.Trabajar (Idtrabajador, Idbazar, RFC, FechaAsistencia) values (933, 33, 'JHWX509775')
4850 insert into GatitaEmprendedora.Trabajar (Idtrabajador, Idbazar, RFC, FechaAsistencia) values (935, 35, 'WWEJ956105')
4851 insert into GatitaEmprendedora.Trabajar (Idtrabajador, Idbazar, RFC, FechaAsistencia) values (936, 36, 'GBJA585821')
4852 insert into GatitaEmprendedora.Trabajar (Idtrabajador, Idbazar, RFC, FechaAsistencia) values (937, 37, 'GIMY222562')
4853 insert into GatitaEmprendedora.Trabajar (Idtrabajador, Idbazar, RFC, FechaAsistencia) values (938, 38, 'MSAJ609823')
4854 insert into GatitaEmprendedora.Trabajar (Idtrabajador, Idbazar, RFC, FechaAsistencia) values (939, 39, 'EYMG039644')
4855 insert into GatitaEmprendedora.Trabajar (Idtrabajador, Idbazar, RFC, FechaAsistencia) values (940, 40, 'EZIG627425')
4856 insert into GatitaEmprendedora.Trabajar (Idtrabajador, Idbazar, RFC, FechaAsistencia) values (941, 41, 'LNYM346606')
4857 insert into GatitaEmprendedora.Trabajar (Idtrabajador, Idbazar, RFC, FechaAsistencia) values (942, 42, 'MUEC770752')
4858 insert into GatitaEmprendedora.Trabajar (Idtrabajador, Idbazar, RFC, FechaAsistencia) values (943, 43, 'RKBG907923')
4859 insert into GatitaEmprendedora.Trabajar (Idtrabajador, Idbazar, RFC, FechaAsistencia) values (944, 44, 'PWOB052185')
4860 insert into GatitaEmprendedora.Trabajar (Idtrabajador, Idbazar, RFC, FechaAsistencia) values (945, 45, 'BJDO237970')
4861 insert into GatitaEmprendedora.Trabajar (Idtrabajador, Idbazar, RFC, FechaAsistencia) values (946, 46, 'CXLR637405')
4862 insert into GatitaEmprendedora.Trabajar (Idtrabajador, Idbazar, RFC, FechaAsistencia) values (947, 47, 'DTXU207655')
4863 insert into GatitaEmprendedora.Trabajar (Idtrabajador, Idbazar, RFC, FechaAsistencia) values (948, 48, 'QQXH050684')
4864 insert into GatitaEmprendedora.Trabajar (Idtrabajador, Idbazar, RFC, FechaAsistencia) values (949, 49, 'YZUU484727')
4865 insert into GatitaEmprendedora.Trabajar (Idtrabajador, Idbazar, RFC, FechaAsistencia) values (950, 50, 'JEIL387473')
4866 insert into GatitaEmprendedora.Trabajar (Idtrabajador, Idbazar, RFC, FechaAsistencia) values (951, 51, 'AZTS417408')
4867 insert into GatitaEmprendedora.Trabajar (Idtrabajador, Idbazar, RFC, FechaAsistencia) values (952, 52, 'NSQN171620')
4868 insert into GatitaEmprendedora.Trabajar (Idtrabajador, Idbazar, RFC, FechaAsistencia) values (953, 53, 'DDEF536599')
4869 insert into GatitaEmprendedora.Trabajar (Idtrabajador, Idbazar, RFC, FechaAsistencia) values (954, 54, 'XORR536930')
4870 insert into GatitaEmprendedora.Trabajar (Idtrabajador, Idbazar, RFC, FechaAsistencia) values (955, 55, 'ZSLB860539')
4871 insert into GatitaEmprendedora.Trabajar (Idtrabajador, Idbazar, RFC, FechaAsistencia) values (956, 56, 'UCZQ267554')
4872 insert into GatitaEmprendedora.Trabajar (Idtrabajador, Idbazar, RFC, FechaAsistencia) values (957, 57, 'RJIJ789290')
4873 insert into GatitaEmprendedora.Trabajar (Idtrabajador, Idbazar, RFC, FechaAsistencia) values (958, 58, 'WEXH888517')
4874 insert into GatitaEmprendedora.Trabajar (Idtrabajador, Idbazar, RFC, FechaAsistencia) values (959, 59, 'CQOL787311VZ4')
4875 •insert into GatitaEmprendedora.Trabajar (Idtrabajador, Idbazar, RFC, FechaAsistencia) values (960, 60, 'CQOL787311VZ4')
4876 'CQOL787311VZ4', (SELECT fechainicio FROM GATITAEMPRENDEDORA.BAZAR WHERE idbazar = 60) +
4877 (random() * ((SELECT fechainicio FROM GATITAEMPRENDEDORA.BAZAR WHERE idbazar = 60))::int)::date AS fecha_aleatoria));
4878
4879
4880

```

Y así quedó la tabla poblada:



	idtrabajador	idbazar	rfc	fechaasistencia
Grilla	932	32	PZPX774729TF5	2024-04-24
Texto	933	33	JHWX509775PX7	2023-10-27
record	934	34	OFEZ711348WA4	2023-10-11
	935	35	WWEJ956105HK4	2023-07-10
	936	36	HXR8348730QD7	2023-12-17
	937	37	GIMY222562IS8	2024-04-26
	938	38	MSAJ609823FD8	2024-01-24
	939	39	EYMG039644AK8	2024-08-08
	940	40	EZIG627425GR1	2024-02-17
	941	41	LNYM346606AC0	2024-11-01
	942	42	MUEC770752OE1	2024-08-12
	943	43	GBJA585821VR4	2024-03-02
	944	44	PWOB052185PE2	2023-06-01
	945	45	BJDO237970ZP8	2023-06-10
	946	46	CXLR637405AZ5	2023-06-30
	947	47	DTXU207655SX0	2023-10-06
	948	48	RKBG907923OG5	2023-07-24
	949	49	QQXH050684GA6	2023-07-02
	950	50	JEIL387473HG8	2023-07-06
	951	51	AZTS417408FF1	2023-10-24
	952	52	NSQN171620YQ1	2023-09-27
	953	53	DDEF536599IH9	2024-05-08
	954	54	XORR536930YK4	2024-01-23
	955	55	ZSLB860539YS5	2023-12-27
	956	56	YZUU484727ZD8	2024-01-25
	957	57	UCZJ267554JE8	2023-09-26
	958	58	RJIJ789290OE9	2024-02-04
	959	59	WEXH888517WR3	2023-05-29
	960	60	CQOL787311VZ4	2024-02-01

Entonces ya podemos pasar con otra tabla que tenga únicamente PKs:

### Tabla Estand

Captura de Mockaroo:

Field Name	Type	Options
NumeroEstand	Row Number	blank: 0 % $\Sigma$ X
IdPaquete	Number	min: 1 max: 3 decimals: 0 blank: 0 % $\Sigma$ X
PrecioBase	Money	between 50 and 300 in \$ blank: 0 % $\Sigma$ X

Y así se ve la tabla poblada (usted disculpe el cambio de colores en el dbeaver):

	123 ↗ numeroestand	123 preciobase	123 ↗ idpaquete
1	1	\$104.63	2
2	2	\$201.43	1
3	3	\$246.99	1
4	4	\$170.62	2
5	5	\$274.08	3
6	6	\$50.04	2
7	7	\$104.12	3
8	8	\$122.22	2
9	9	\$220.67	2
10	10	\$139.34	2
11	11	\$70.61	2
12	12	\$144.17	3
13	13	\$110.92	2
14	14	\$106.94	1
15	15	\$244.53	3
16	16	\$62.95	2
17	17	\$281.37	1
18	18	\$66.03	1
19	19	\$217.40	1
20	20	\$147.79	2

Teniendo la tabla **Estand** poblada, podemos pasar a las tablas que dependen de ella.

## Tabla Paquete

Captura de Mockaroo:

Field Name	Type	Options
IdPaquete	Row Number	blank: 0 % $\Sigma$ X
Paquete	Custom List	Básico,Premium,Emprendedor sequential blank: 0 % $\Sigma$ X

Y así se ve la tabla poblada:

idpaquete	A-Z paquete
1	Básico
2	Premium
3	Emprendedor

Y, a su vez, teniendo **Paquete** poblada, podemos poblar **Amenidad paquete**.

## Tabla Amenidad Paquete

Captura de Mockaroo:

Field Name	Type	Options
IdPaquete	Number	min: 1 max: 3 decimals: 0 blank: 0 % $\Sigma$ X
Amenidad	Blank	blank: 0 % $\Sigma$ X

Aquí podemos notar que el campo de Amenidad es null y esto es porque llenarla en Visual Studio Code era más práctico para esta tabla, de la siguiente manera:

```
insert into GatitaEmprendedora.PaqueteAmenidad (IdPaquete, Amenidad) values (1, '1 mesa');
insert into GatitaEmprendedora.PaqueteAmenidad (IdPaquete, Amenidad) values (1, '2 sillas');
insert into GatitaEmprendedora.PaqueteAmenidad (IdPaquete, Amenidad) values (2, '2 mesas');
insert into GatitaEmprendedora.PaqueteAmenidad (IdPaquete, Amenidad) values (2, '4 sillas');
insert into GatitaEmprendedora.PaqueteAmenidad (IdPaquete, Amenidad) values (3, '3 mesas');
insert into GatitaEmprendedora.PaqueteAmenidad (IdPaquete, Amenidad) values (3, '6 sillas');
insert into GatitaEmprendedora.PaqueteAmenidad (IdPaquete, Amenidad) values (3, 'Pantalla táctil');
insert into GatitaEmprendedora.PaqueteAmenidad (IdPaquete, Amenidad) values (3, 'Toma de corriente');
```

Así, la tabla ya poblada queda como:

idpaquete	amenidad
1	1 mesa
2	2 sillas
3	2 mesas
4	4 sillas
5	3 mesas
6	6 sillas
7	Pantalla táctil
8	Toma de corriente

Ya podemos pasar a poblar **Negocio**.

## Tabla Negocio

Captura de Mockaroo:

Dejamos un 10% de null en la columna de Estand porque no siempre un negocio tendrá asignado uno, esto depende de su asistencia a algún bazar.

Field Name	Type	Options
IdNegocio	Row Number	blank: 0 % $\Sigma$ X
NumeroEstand	Number	min: 1 max: 20 decimals: 0 blank: 10% $\Sigma$ X
NombreNegocio	Fake Company Name	blank: 0 % $\Sigma$ X
Descripcion	Paragraphs	at least 1 but no more than 3 blank: 0 % $\Sigma$ X

Y así se ve la tabla poblada:

idnegocio	numeroestand	nombrenegocio	descripcion
272	272	18 Cruickshank Inc	Morbi non lectus. Aliquam sit amet diam in magna bibendum imperdiet. Nullam orci pede, venenatis non, sodales sed, Fusce consequat. Nulla nisl. Nunc nisl. Duis bibendum, felis sed interdum venenatis, turpis enim blandit mi, in porttitor
273	273	13 Langosh, Sauer and Howe	Duis bibendum, felis sed interdum venenatis, turpis enim blandit mi, in porttitor pede justo eu massa. Donec dapibus. C
274	274	11 Koch, Dooley and Hand	In hac habitasse platea dictumst. Morbi vestibulum, velit id pretium iaculis, diam erat fermentum justo, nec condimentu
275	275	14 Koss-Padberg	Phasellus sit amet erat. Nulla tempus. Vivamus in felis eu sapien cursus vestibulum. Proin eu mi. Nulla ac enim. In temp
276	276	4 Schiller-Kovacek	Nullam sit amet turpis elementum ligula vehicula consequat. Morbi a ipsum. Integer a nibh. In quis justo. Maecenas rh
277	277	1 Rodriguez-Parker	Phasellus in felis. Donec semper sapien a libero. Nam dui. Proin leo odio, porttitor id, consequat in, consequat ut, nulla
278	278	2 Kuhlman, Oberbrunner and Yundt	Proin leo odio, porttitor id, consequat in, consequat ut, nulla. Sed accumsan felis. Ut at dolor quis odio consequat varius
279	279	[NULL] Buckridge, Reilly and Reilly	Doyle Group
280	280	14 Doyle Group	Morbi non lectus. Aliquam sit amet diam in magna bibendum imperdiet. Nullam orci pede, venenatis non, sodales sed,
281	281	19 Mohr Group	Cras non velit nec nisi vulputate nonummy. Maecenas tincidunt lacus at velit. Vivamus vel nulla eget eros elementum pe
282	282	13 Rutherford-Ankunding	In sagittis dui vel nisl. Duis ac nibh. Fusce lacus purus, aliquet at, feugiat non, pretium quis, lectus. Suspendisse potenti.
283	283	6 Koch Group	Morbi non lectus. Aliquam sit amet diam in magna bibendum imperdiet. Nullam orci pede, venenatis non, sodales sed,
284	284	12 Mayer-Kilback	Aenean fermentum. Donec ut mauris eget massa tempor convallis. Nulla neque libero, convallis eget, eleifend luctus, ul
285	285	[NULL] Kshlerin and Sons	In hac habitasse platea dictumst. Morbi vestibulum, velit id pretium iaculis, diam erat fermentum justo, nec condimentu
286	286	7 Anderson-Williamson	Sed ante. Vivamus tortor. Duis mattis egestas metus. Aenean fermentum. Donec ut mauris eget massa tempor convalli
287	287	12 Mayert Group	Maecenas leo odio, condimentum id, luctus nec, molestie sed, justo. Pellentesque viverra pede ac diam. Cras pellentesq
288	288	10 Langosh LLC	Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Vivamus vestibulum sagittis sapi
289	289	7 Ondricka-Tillman	Integer ac leo. Pellentesque ultrices mattis odio. Donec vitae nisi. Nam ultrices, libero non mattis pulvinar, nulla pede
290	290	2 Mante, Stehr and Howe	Proin eu mi. Nulla ac enim. In tempor, turpis nec euismod scelerisque, quam turpis adipiscing lorem, vitae mattis nibh li
291	291	3 West, Stanton and Jacobson	Etiam vel augue. Vestibulum rutrum rutrum neque. Aenean auctor gravida sem. Praesent id massa id nisl venenatis laci
292	292	10 Hane, Reichel and Schultz	Mauris enim leo, rhoncus sed, vestibulum sit amet, cursus id, turpis. Integer aliquet, massa id lobortis convallis, tortor ris
293	293	18 Mitchell-Murray	Nam ultrices, libero non mattis pulvinar, nulla pede ullamcorper augue, a suscipit nulla elit ac nulla. Sed vel enim sit am
294	294	19 Nikolaus Inc	Praesent blandit. Nam nulla. Integer pede justo, lacinia eget, tincidunt eget, tempus vel, pede.
295	295	[NULL] Ortiz Inc	Fusce posuere felis sed lacus. Morbi sem mauris, laoreet ut, rhoncus aliquet, pulvinar sed, nisl. Nunc rhoncus dui vel sem
296	296	12 Spencer, Kub and Franecki	Cras mi pede, malesuada in, imperdiet et, commodo vulputate, justo. In blandit ultrices enim. Lorem ipsum dolor sit am
297	297	14 Rempel-Medhurst	Morbi non lectus. Aliquam sit amet diam in magna bibendum imperdiet. Nullam orci pede, venenatis non, sodales sed,
298	298	9 Sawaya-Abshire	Pellentesque at nulla. Suspendisse potenti. Cras in purus eu magna vulputate luctus. Cum sociis natoque penatibus et r
299	299	14 Schamberger-Gulgowski	Fusce posuere felis sed lacus. Morbi sem mauris, laoreet ut, rhoncus aliquet, pulvinar sed, nisl. Nunc rhoncus dui vel sem
300	300	8 Wuckert-Larson	Duis aliquam convallis nunc. Proin at turpis a pede posuere nonummy. Integer non velit. Donec diam neque, vestibulu

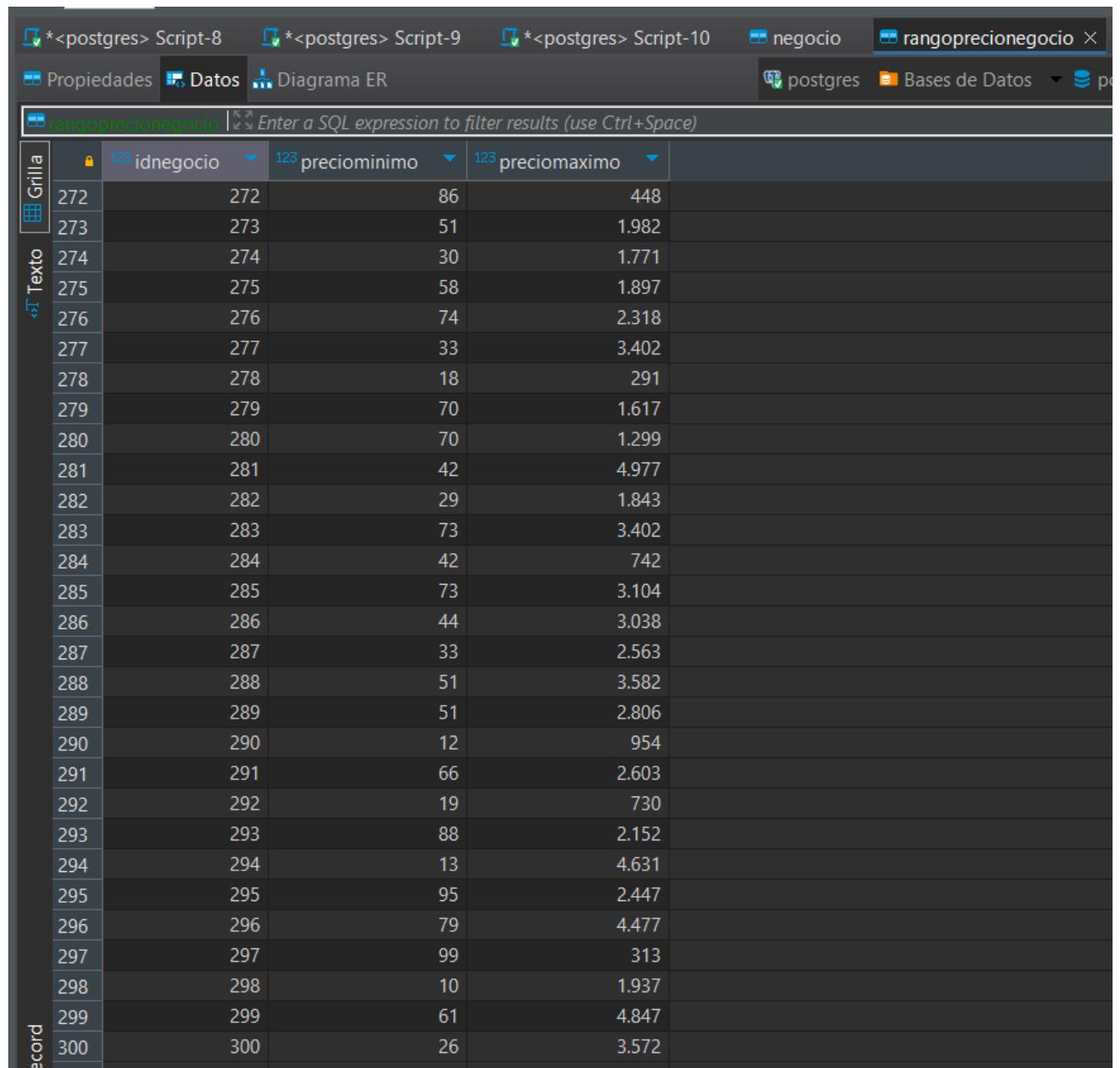
Entonces, ahora las tablas que dependen directamente de Negocio.

## Tabla Rango Precio Negocio

Captura de Mockaroo:

Field Name	Type	Options			
IdNegocio	Row Number	blank:	0 %	$\Sigma$	X
PrecioMinimo	Number	min:	10	max:	99
PrecioMaximo	Number	min:	100	max:	5000

Y así se ve la tabla poblada:



The screenshot shows the Mockaroo interface with the 'rangoprecionegocio' table populated with data. The table has four columns: idnegocio, preciominimo, preciomaximo, and a blank column. The data consists of 300 rows, each containing a unique ID from 272 to 300, and corresponding minimum and maximum price values.

idnegocio	preciominimo	preciomaximo	
272	86	448	
273	51	1.982	
274	30	1.771	
275	58	1.897	
276	74	2.318	
277	33	3.402	
278	18	291	
279	70	1.617	
280	70	1.299	
281	42	4.977	
282	29	1.843	
283	73	3.402	
284	42	742	
285	73	3.104	
286	44	3.038	
287	33	2.563	
288	51	3.582	
289	51	2.806	
290	12	954	
291	66	2.603	
292	19	730	
293	88	2.152	
294	13	4.631	
295	95	2.447	
296	79	4.477	
297	99	313	
298	10	1.937	
299	61	4.847	
300	26	3.572	

## Tabla Teléfono Negocio

Captura de Mockaroo:

Field Name	Type	Options
IdNegocio	Row Number	blank: 0 % $\Sigma$ X
Teléfono	Digit Sequence	##### blank: 0 % $\Sigma$ X

Y así se ve la tabla poblada:

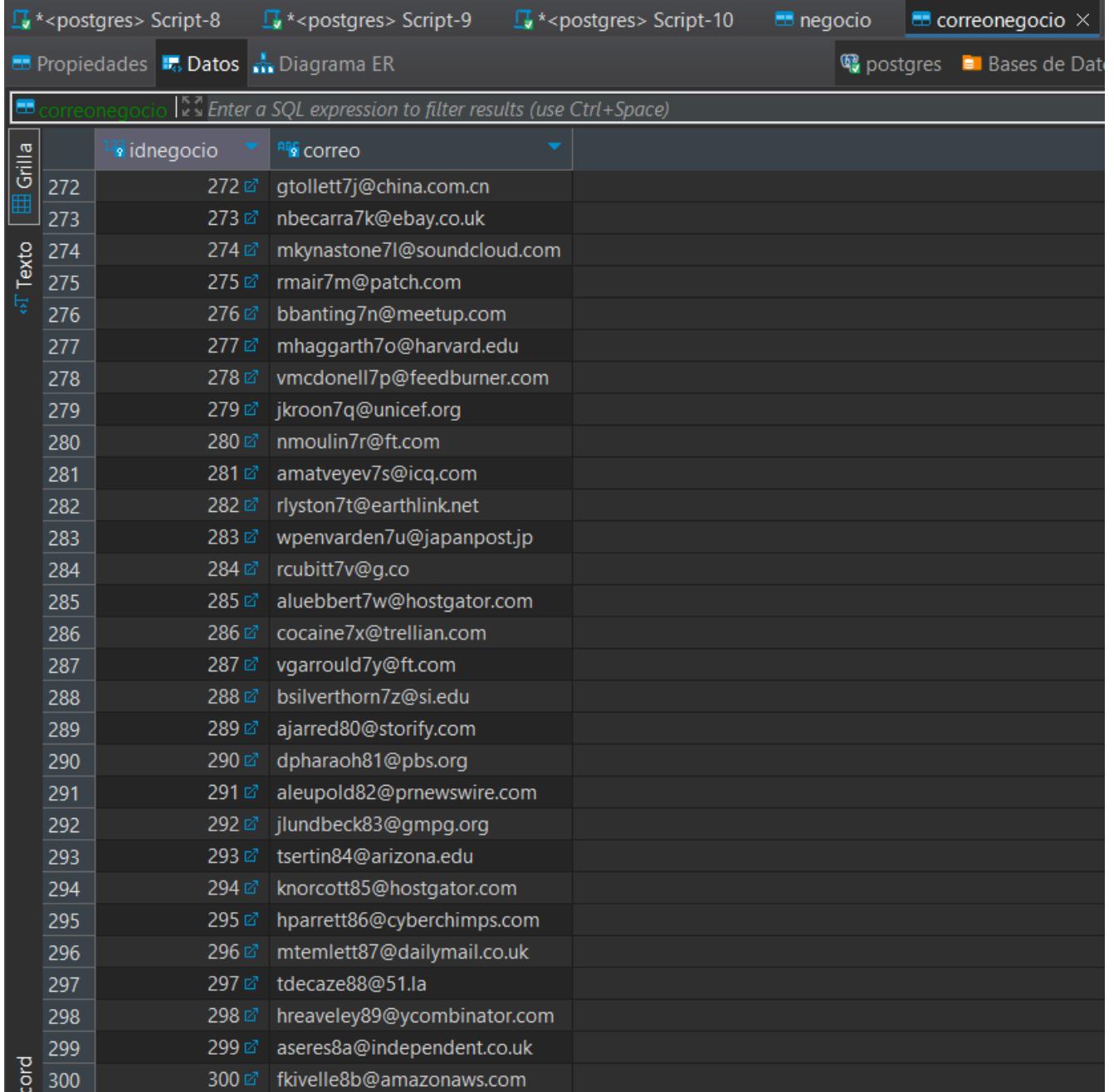
Grilla	123 idnegocio	ABC teléfono
272	272	9305414180
273	273	6097204146
274	274	4777902027
275	275	6136543752
276	276	8421592431
277	277	4668579709
278	278	2023902132
279	279	2824235988
280	280	9482528474
281	281	4238483270
282	282	0382561151
283	283	5430815935
284	284	6377020336
285	285	3904692279
286	286	8077360317
287	287	2515095736
288	288	9370291298
289	289	9872940948
290	290	9173810546
291	291	7170553216
292	292	0807999790
293	293	9111091991
294	294	3697942966
295	295	4785887315
296	296	6295796750
297	297	5319700073
298	298	4509776588
299	299	0281103095
300	300	8350564922

## Tabla Correo Negocio

Captura de Mockaroo:

Field Name	Type	Options
IdNegocio	Row Number	blank: 0 % $\Sigma$ X
Correo	Email Address	blank: 0 % $\Sigma$ X

Y así se ve la tabla poblada:



The screenshot shows the Mockaroo interface with a populated table named 'correo'. The table has two columns: 'idnegocio' (Row Number) and 'correo' (Email Address). The 'correo' column contains 300 entries, each starting with a blue link icon. The rows are numbered from 272 to 300. The interface includes a toolbar at the top with tabs for 'Propiedades', 'Datos', and 'Diagrama ER'. A search bar at the top right says 'Enter a SQL expression to filter results (use Ctrl+Space)'. The bottom right corner of the window says 'NAMEisNULL'.

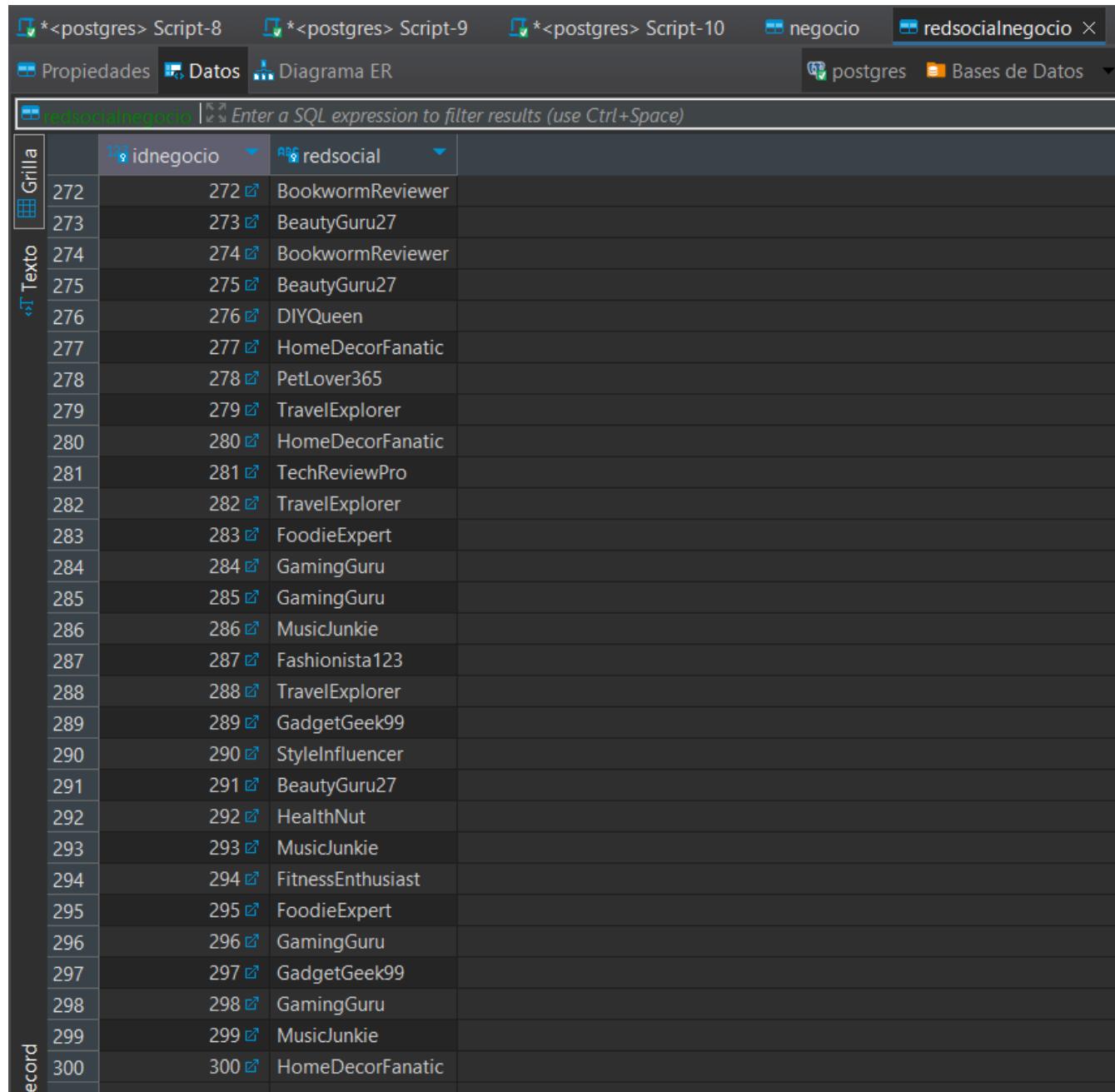
idnegocio	correo
272	gtollett7j@china.com.cn
273	nbecarra7k@ebay.co.uk
274	mkynastone7l@soundcloud.com
275	rmair7m@patch.com
276	bbanting7n@meetup.com
277	mhaggarth7o@harvard.edu
278	vmcdonell7p@feedburner.com
279	jkroon7q@unicef.org
280	nmoulin7r@ft.com
281	amatveyev7s@icq.com
282	rlyston7t@earthlink.net
283	wpenvarden7u@japanpost.jp
284	rcubitt7v@g.co
285	aluebbert7w@hostgator.com
286	cocaine7x@trellian.com
287	vgarrould7y@ft.com
288	bsilverthorn7z@si.edu
289	ajarred80@storify.com
290	dpharaoh81@pbs.org
291	aleupold82@prnewswire.com
292	jlundbeck83@gmpg.org
293	tsertin84@arizona.edu
294	knorcott85@hostgator.com
295	hparrett86@cyberchimps.com
296	mtemlett87@dailymail.co.uk
297	tdecaze88@51.la
298	hreaveley89@ycombinator.com
299	aseres8a@independent.co.uk
300	fkivelle8b@amazonaws.com

## Tabla Red Social Negocio

Captura de Mockaroo:

Field Name	Type	Options
IdNegocio	Row Number	blank: 0 % $\Sigma$ X
RedSocial	Custom List	Fashionista123, TechReviewPro, BeautyGuru27, FoodieExpert, GadgetGeek99, StyleInfluencer, HomeDecorFanatic, PetLover365, TravelExplorer, MusicJunkie, GamingGuru, GadgetGeek99, StyleInfluencer, BeautyGuru27, HealthNut, FitnessEnthusiast, FoodieExpert, GamingGuru, GadgetGeek99, GamingGuru, MusicJunkie, HomeDecorFanatic

Y así se ve la tabla poblada:



The screenshot shows a PostgreSQL database interface with the 'redsocialnegocio' table selected. The table has two columns: 'idnegocio' (Row Number) and 'redsocial'. The data consists of 300 rows, each containing a unique ID from 272 to 300 and a corresponding social media handle. The handles include various interests such as BookwormReviewer, BeautyGuru27, DIYQueen, HomeDecorFanatic, PetLover365, TravelExplorer, TechReviewPro, etc.

idnegocio	redsocial
272	BookwormReviewer
273	BeautyGuru27
274	BookwormReviewer
275	BeautyGuru27
276	DIYQueen
277	HomeDecorFanatic
278	PetLover365
279	TravelExplorer
280	HomeDecorFanatic
281	TechReviewPro
282	TravelExplorer
283	FoodieExpert
284	GamingGuru
285	GamingGuru
286	MusicJunkie
287	Fashionista123
288	TravelExplorer
289	GadgetGeek99
290	StyleInfluencer
291	BeautyGuru27
292	HealthNut
293	MusicJunkie
294	FitnessEnthusiast
295	FoodieExpert
296	GamingGuru
297	GadgetGeek99
298	GamingGuru
299	MusicJunkie
300	HomeDecorFanatic

Ahora, con las tablas **Bazar** y **Negocio** pobladas, podemos pasar a poblar **Agendar**:

### Tabla Agendar

Captura de Mockaroo:

Field Name	Type	Options
IdBazar	Number	min: 1 max: 300 decimals: 0 blank: 0 % $\Sigma$ X
IdNegocio	Number	min: 1 max: 300 decimals: 0 blank: 0 % $\Sigma$ X
FechaAsistencia	SQL Expression	example: DEFAULT blank: 0 % $\Sigma$ X

Y así queda la tabla poblada:

agendar   Enter a SQL expression to filter results (use Ctrl+Space)			
Grilla	123 idbazar	123 idnegocio	fechaasistencia
272	100	198	2024-06-06
273	146	250	2023-08-02
274	204	289	2023-09-12
275	264	114	2023-08-21
276	184	264	2024-01-23
277	3	30	2023-09-30
278	83	21	2024-09-29
279	174	37	2023-08-23
280	218	12	2023-09-28
281	113	204	2024-06-10
282	42	239	2023-10-18
283	106	93	2024-04-12
284	189	88	2024-01-15
285	276	63	2023-02-15
286	201	201	2024-05-20
287	245	119	2023-12-28
288	215	67	2023-11-06
289	52	154	2024-02-06
290	110	227	2023-07-01
291	65	288	2023-10-25
292	173	58	2024-02-27
293	58	205	2023-10-28
294	68	79	2024-01-31
295	25	31	2023-09-18
296	254	208	2023-11-01
297	42	39	2024-06-12
298	84	265	2023-12-13
299	295	230	2024-09-10
300	10	234	2024-04-04

Así que, pasemos a poblar **Producto** y **Servicio** que dependen de **Negocio** (porque eran entidades débiles).

### Tabla Producto

Captura de Mockaroo:

Field Name	Type	Options		
IdNegocio	Row Number	blank: 0 %	$\Sigma$	X
IdProducto	Row Number	blank: 0 %	$\Sigma$	X
NombreProducto	Product Name	blank: 0 %	$\Sigma$	X
Descripcion	Product Description	blank: 0 %	$\Sigma$	X
Tipo	Product Subcategory	blank: 0 %	$\Sigma$	X
Precio	Product Price	blank: 0 %	$\Sigma$	X
Presentacion	Custom List	bolsa, lata, botella, caja, frasco, envase, tubo, sobre, paquete, tarro	random ▾	blank: 0 % $\Sigma$ X
Stock	Number	min: 0 max: 200 decimals: 0 blank: 0 %	$\Sigma$	X

Y así queda la tabla poblada:

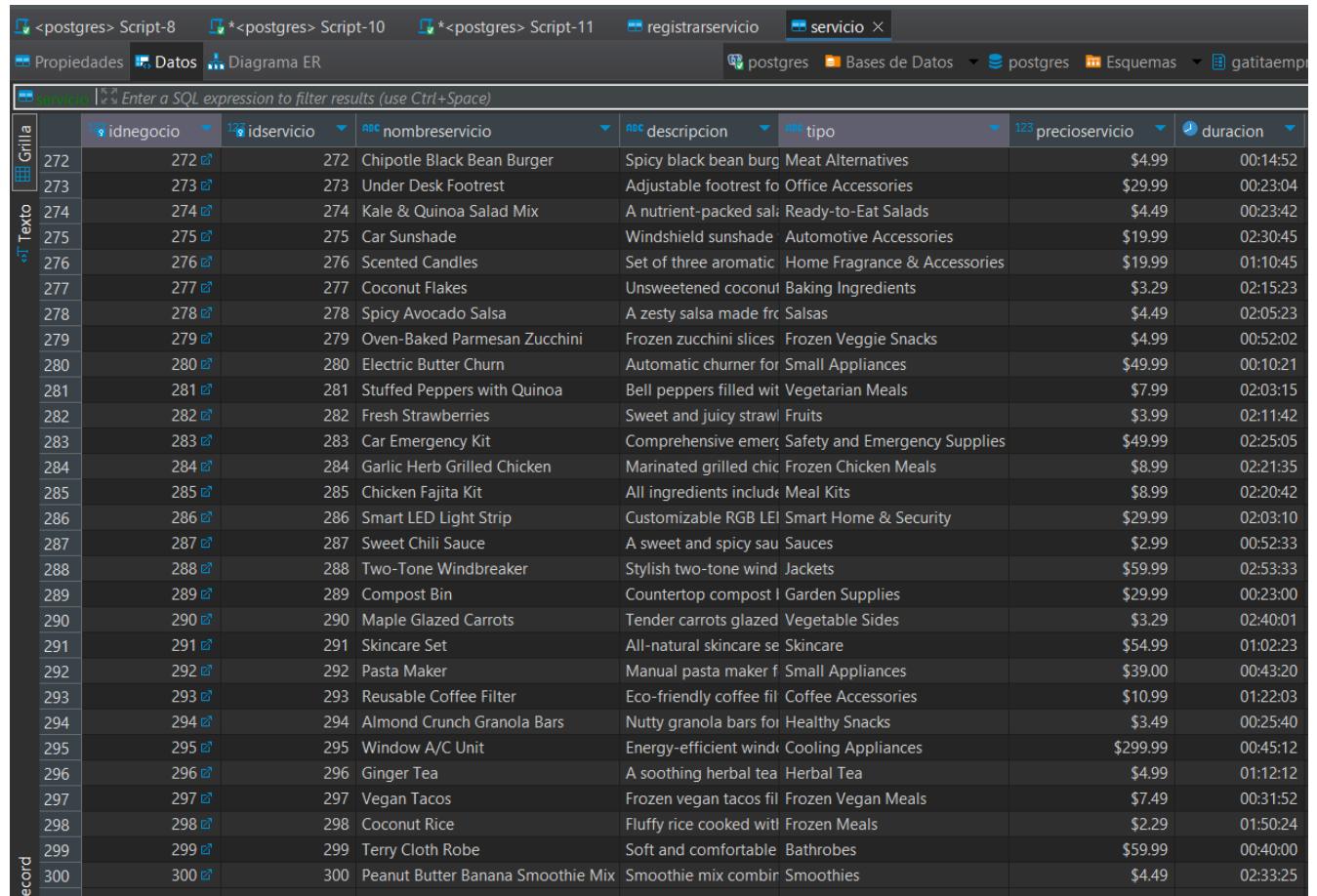
producto   Enter a SQL expression to filter results (use Ctrl+Space)						
Grilla	123 idnegocio	123 idproducto	A-Z nombreproducto	A-Z descripcion	A-Z tipo	123 precio
281	282	282	Tactical Backpack	Durable and versatile backpack for Camping Equipment		\$59.9
282	283	283	Artisan Bread Loaf	Fresh artisan bread, perfect for sar Breads		\$3.9
283	284	284	Ice Cream Scoop	Durable scoop for perfectly shape Kitchen Gadgets		\$12.9
284	285	285	Yoga Mat Carrier	Convenient carrier for transporting Yoga Equipment		\$12.9
285	286	286	Green Smoothie Mix	A convenient powder mix combin Health Drinks		\$5.9
286	287	287	Water Bottle	Insulated water bottle for keeping Hydration Gear		\$18.9
287	288	288	Wireless Car Charger	Convenient charging pad for wire Phone Accessories		\$24.9
288	289	289	Body Pillow Case	Soft and breathable pillowcase for Bedding & Linens		\$14.9
289	290	290	Maple Pecan Oatmeal Cookies	Soft oatmeal cookies with maple a Cookies		\$3.9
290	291	291	Maple Pecan Pancake Mix	A fluffy pancake mix infused with Pancakes & Waffles		\$4.9
291	292	292	Hand Mixer	Compact hand mixer for easy baki Blenders & Juicers		\$29.9
292	293	293	Coconut Milk	Rich coconut milk for curries and Coconut Products		\$2.4
293	294	294	Puff Pastry	Versatile puff pastry for pies and Pastry Products		\$5.4
294	295	295	Silicone Baking Mats	Reusable silicone mats for non-stick Baking Accessories		\$19.9
295	296	296	Thai Peanut Noodles	Noodles tossed in a spicy Thai pea Noodles		\$4.9
296	297	297	Digital Drawing Tablet	Tablet for digital drawing and illus Digital Art Supplies		\$79.9
297	298	298	Wireless Wi-Fi Extender	Boosts your Wi-Fi coverage for be Networking Equipment		\$49.9
298	299	299	Deep Tissue Massage Gun	Rechargeable massage gun for reli Injury Prevention & Recov		\$89.9
299	300	300	Bamboo Toothbrush Holder	Eco-friendly bamboo holder for Oral Care & Hygiene		\$14.9

## Tabla Servicio

Captura de Mockaroo:

Field Name	Type	Options
IdNegocio	Row Number	blank: 0 % $\Sigma$ X
IdServicio	Number	min: 1 max: 20 decimals: 0 blank: 10% $\Sigma$ X
NombreServicio	Custom List	barber, nails, spa, haircut, massage, facial, fast food, dancing classes, videogames, grooming pets, r $\Sigma$ random▼ blank: 0 % $\Sigma$ X
Descripcion	Product Description	blank: 0 % $\Sigma$ X
Tipo	Product Subcategory	blank: 0 % $\Sigma$ X
PrecioServicio	Money	between 0 and 50 in \$▼ blank: 0 % $\Sigma$ X
Duracion	Time	from 12:00 AM to 11:59 AM format: 24 Hour▼ blank: 0 % $\Sigma$ X

Y así queda la tabla poblada:



The screenshot shows a PostgreSQL database interface with the 'servicio' table selected. The table has 300 records and 8 columns: idnegocio, idservicio, nombrservicio, descripcion, tipo, precioservicio, and duracion. The data includes various items like Chipotle Black Bean Burger, Adjustable footrest, Kale & Quinoa Salad Mix, etc., with their descriptions, categories, prices, and durations.

idnegocio	idservicio	nombrservicio	descripcion	tipo	precioservicio	duracion
272	272	Chipotle Black Bean Burger	Spicy black bean burg	Meat Alternatives	\$4.99	00:14:52
273	273	Under Desk Footrest	Adjustable footrest fo	Office Accessories	\$29.99	00:23:04
274	274	Kale & Quinoa Salad Mix	A nutrient-packed sal	Ready-to-Eat Salads	\$4.49	00:23:42
275	275	Car Sunshade	Windshield sunshade	Automotive Accessories	\$19.99	02:30:45
276	276	Scented Candles	Set of three aromatic	Home Fragrance & Accessories	\$19.99	01:10:45
277	277	Coconut Flakes	Unsweetened coconut	Baking Ingredients	\$3.29	02:15:23
278	278	Spicy Avocado Salsa	A zesty salsa made fr	Salsas	\$4.49	02:05:23
279	279	Oven-Baked Parmesan Zucchini	Frozen zucchini slices	Frozen Veggie Snacks	\$4.99	00:52:02
280	280	Electric Butter Churn	Automatic churner for	Small Appliances	\$49.99	00:10:21
281	281	Stuffed Peppers with Quinoa	Bell peppers filled wit	Vegetarian Meals	\$7.99	02:03:15
282	282	Fresh Strawberries	Sweet and juicy straw	Fruits	\$3.99	02:11:42
283	283	Car Emergency Kit	Comprehensive emerg	Safety and Emergency Supplies	\$49.99	02:25:05
284	284	Garlic Herb Grilled Chicken	Marinated grilled chic	Frozen Chicken Meals	\$8.99	02:21:35
285	285	Chicken Fajita Kit	All ingredients include	Meal Kits	\$8.99	02:20:42
286	286	Smart LED Light Strip	Customizable RGB LEI	Smart Home & Security	\$29.99	02:03:10
287	287	Sweet Chili Sauce	A sweet and spicy sau	Sauces	\$2.99	00:52:33
288	288	Two-Tone Windbreaker	Stylish two-tone wind	Jackets	\$59.99	02:53:33
289	289	Compost Bin	Countertop compost I	Garden Supplies	\$29.99	00:23:00
290	290	Maple Glazed Carrots	Tender carrots glazed	Vegetable Sides	\$3.29	02:40:01
291	291	Skincare Set	All-natural skincare se	Skincare	\$54.99	01:02:23
292	292	Pasta Maker	Manual pasta maker f	Small Appliances	\$39.00	00:43:20
293	293	Reusable Coffee Filter	Eco-friendly coffee fil	Coffee Accessories	\$10.99	01:22:03
294	294	Almond Crunch Granola Bars	Nutty granola bars for	Healthy Snacks	\$3.49	00:25:40
295	295	Window A/C Unit	Energy-efficient windi	Cooling Appliances	\$299.99	00:45:12
296	296	Ginger Tea	A soothing herbal tea	Herbal Tea	\$4.99	01:12:12
297	297	Vegan Tacos	Frozen vegan tacos fil	Frozen Vegan Meals	\$7.49	00:31:52
298	298	Coconut Rice	Fluffy rice cooked with	Frozen Meals	\$2.29	01:50:24
299	299	Terry Cloth Robe	Soft and comfortable	Bathrobes	\$59.99	00:40:00
300	300	Peanut Butter Banana Smoothie Mix	Smoothie mix combir	Smoothies	\$4.49	02:33:25

## Tabla Emprendedor

Captura de Mockaroo:

The form includes the following fields and configurations:

- RFC:** Regular Expression [A-Z0-9] {13}, blank: 0%, Σ X
- NombreEmprendedor:** First Name, blank: 0%, Σ X
- APaternoEmprendedor:** Last Name, blank: 0%, Σ X
- AMaternoEmprendedor:** Last Name, blank: 0%, Σ X
- Calle:** Street Name, blank: 0%, Σ X
- NumeroExterior:** Street Number, blank: 0%, Σ X
- NumeroInterior:** Street Number, blank: 0%, Σ X
- Colonia:** City, blank: 0%, Σ X
- Estado:** State, restrict states..., Only US, blank: 0%, Σ X
- Genero:** Custom List F,M,B, random, blank: 0%, Σ X
- IdNegocio:** Row Number, blank: 0%, Σ X
- FechaNacimiento:** Datetime, 12/26/1996 to 06/14/2003, format: yyyy-mm-dd, blank: 0%, Σ X

Buttons: + ADD ANOTHER FIELD, GENERATE FIELDS USING AI...

Bottom controls: # Rows: 300, Format: SQL, Table Name: GatitaEmprendedor, include CREATE TABLE

Y así se ve la tabla poblada:

The table structure is as follows:

	rfc	nombreemprendedor	apaternoemprendedor	amaternoemprendedor	calle	numeroexterior	numerointerior	colonia	estado
272	SHAS1GGL1	Ambrosius	Nesbeth	Menis	Glendale	37200	2898	Oakland	California
273	A2252C395	Kaylil	Picker	Wysome	Shopko	4	7690	Lansing	Michigan
274	0050S058Z	Wang	Walstow	Figgures	Artisan	2262	50	Portland	Oregon
275	49V1109A7	Kinnie	Cluderay	Cockerill	Kinsman	6	82	Seattle	Washington
276	7BF8TASZ5	Isis	Trenaman	Solley	Cambridge	6	041	Peoria	Illinois
277	R6362K9R1	Fabe	Antonopoulos	Deary	Glendale	49	81	Tulsa	Oklahoma
278	MX5XA0VR	Alica	McGifford	Bricket	Clove	88347	65	Seattle	Washington
279	091ZY153U	Julita	Fownes	Blake	Mendota	5	93746	San Antonio	Texas
280	681D94880	Tannie	Brighouse	Ferneyhough	Blue Bill Park	4216	6	Wichita	Kansas
281	2877EM61k	Sada	Beccera	Mauditt	Westend	3321	607	Pasadena	California
282	FWZ65CN7	Rosalyn	Esterbrook	Clancey	Debra	3092	978	Saint Paul	Minnesota
283	NCXWIS9M	Mahalia	Witchard	Driscoll	Dayton	9	2140	Van Nuys	California
284	4822A7C60	Ethe	Wretham	Pylknyton	Gulseth	58181	5494	Winston Salem	North Carolina
285	4K4JKSXN9	Wallie	Menguy	Mularkey	Nancy	2547	438	Oklahoma City	Oklahoma
286	N16627X93	Jameson	O'Kelly	O' Ronan	Lawn	75	41	Phoenix	Arizona
287	IN7XUPKYR	Parnell	Kruger	Priel	Harbort	286	93	Des Moines	Iowa
288	P67345Q15	Consalve	Sabathier	Dunstone	Lotheville	485	400	Mobile	Alabama
289	7515ML6P51	Belicia	Silverlock	Mathivet	Ilene	81	23671	Jacksonville	Florida
290	L8QLSH6DT	Ranique	Cracoe	Raisher	Di Loreto	0707	8706	Abilene	Texas
291	6N1YD8E15	Abbie	Mocker	Pitkin	Bultman	4	81434	Trenton	New Jersey
292	WK9QL8A0	Wolfy	Reeve	Beadham	Ludington	0	68386	Des Moines	Iowa
293	7151X2HW	Corbett	Hercock	Chaplain	6th	4993	3	Salt Lake City	Utah
294	NYO95Ri81	Durward	Mowsdell	Hurford	Elmside	4781	19	Bakersfield	California
295	M849QBHV	Greer	Beardwood	Simoneau	Manley	3830	558	Knoxville	Tennessee
296	UTPSBD8R	Miguelita	Farndale	Griffe	Ramsey	6	694	Glendale	Arizona
297	17NIL2NH6	Brew	Simonnet	Irons	Longview	0	3201	Allentown	Pennsylvania
298	V93XYPDAl	Orbadiah	Fazzoli	Matignon	Cody	4	9995	El Paso	Texas
299	ZKP8R4GU6	Guinna	Pury	Stolli	Towne	1	27380	Ashburn	Virginia
300	KT4YMOKC	Sheffie	Readshall	McGannon	Mitchell	3398	8779	Syracuse	New York

Entonces, podemos poblar las tablas que dependen directamente de **Emprendedor**.

## Tabla Teléfono Emprendedor

Captura de Mockaroo:

Field Name	Type	Options
RFC	Blank	blank: 0% $\Sigma$ X
TeléfonoEmprendedor	Digit Sequence	##### $\Sigma$ X

Como se puede ver, generamos la columna de con valores null y esto es porque Mockaroo nos genera números aleatorios cada vez, por lo que necesitamos garantizar que haya en la BD. Con ayuda de un editor de código (Visual Studio Code) copiamos y pegamos de manera eficiente todos datos ya existentes de la tabla .

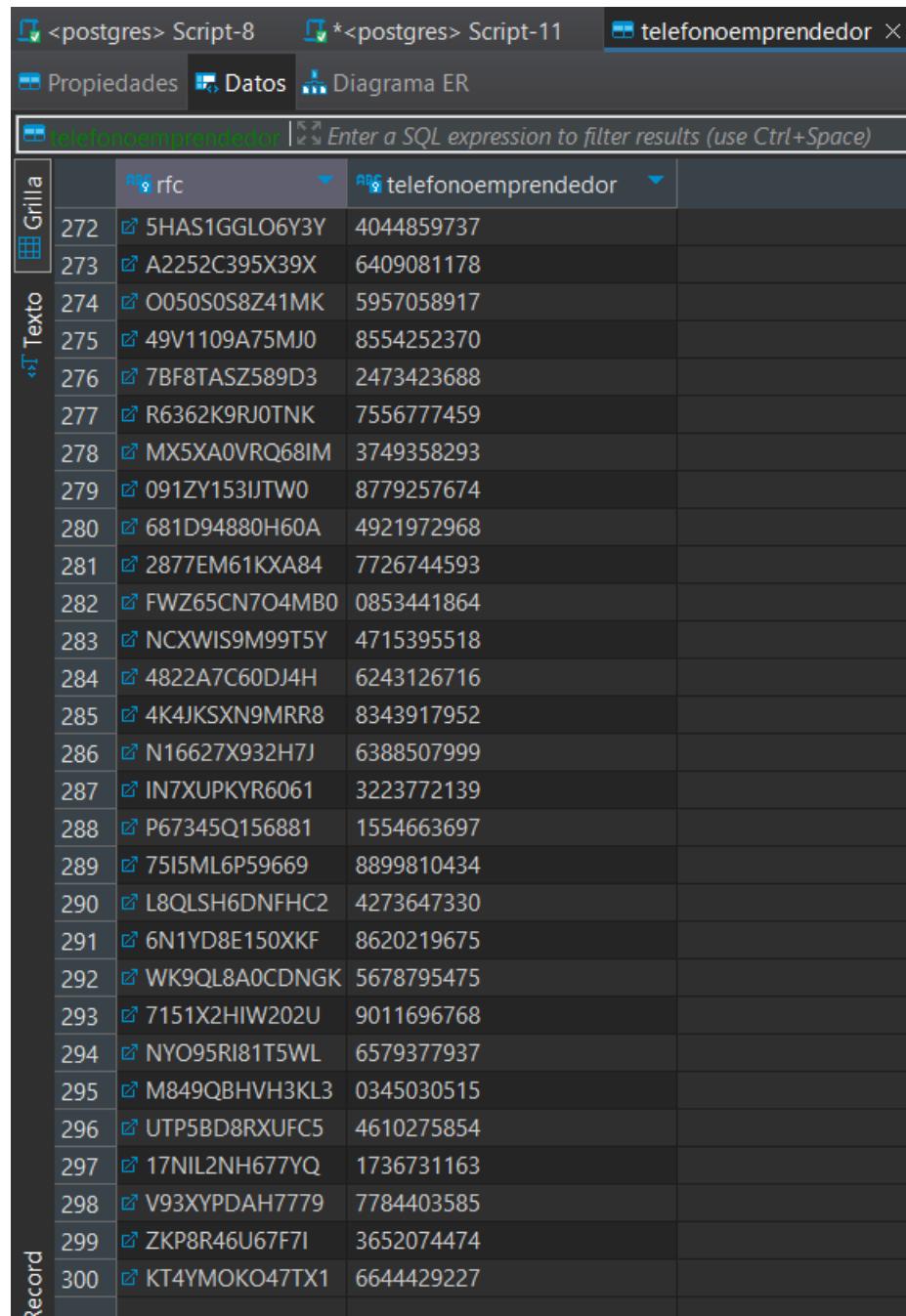
```

File Edit Selection View Go Run Terminal Help <- > FundamentosBD
EXPLORER DML(1).sql
C:\> Users > victo > Downloads > GatitaEmprendedora.TeléfonoEmprendedor.sql
> Prácticas
> ProyectoFinal_NAMEisNULL...
> Diagramas
> Docs
> SQL
> ConsultaDaniel.sql
> DDLsql
> disparadores_funcio...
> DMLsql
> DML(1).sql
> Correcciones.md
> Reparto.mnd
> Tareas
> Lineamientos_FBD.pdf
> Links.mnd
> README_NAMEisNULL...
> READMEEmd

File Edit Selection View Go Run Terminal Help <- > DML(1).sql
DML(1).sql
ProjectoFinal_NAMEisNULL > SQL > DML(1).sql
869 insert into GatitaEmprendedora.TeléfonoEmprendedor (RFC, TeléfonoEmprendedor) values ('CED0543040K63', '0222536151');
870 insert into GatitaEmprendedora.TeléfonoEmprendedor (RFC, TeléfonoEmprendedor) values ('5621AECK6LLV', '3468501727');
871 insert into GatitaEmprendedora.TeléfonoEmprendedor (RFC, TeléfonoEmprendedor) values ('6VGDURNL1BQD', '538096003');
872 insert into GatitaEmprendedora.TeléfonoEmprendedor (RFC, TeléfonoEmprendedor) values ('K4L1GKBN69Z2', '401805233');
873 insert into GatitaEmprendedora.TeléfonoEmprendedor (RFC, TeléfonoEmprendedor) values ('CP7KMA671W59', '2265879321');
874 insert into GatitaEmprendedora.TeléfonoEmprendedor (RFC, TeléfonoEmprendedor) values ('9WM0350690QA', '0168628392');
875 insert into GatitaEmprendedora.TeléfonoEmprendedor (RFC, TeléfonoEmprendedor) values ('5HAS1GG100Y3', '4044859737');
876 insert into GatitaEmprendedora.TeléfonoEmprendedor (RFC, TeléfonoEmprendedor) values ('A2252C39X93X', '640908178');
877 insert into GatitaEmprendedora.TeléfonoEmprendedor (RFC, TeléfonoEmprendedor) values ('0050805821M', '5957058917');
878 insert into GatitaEmprendedora.TeléfonoEmprendedor (RFC, TeléfonoEmprendedor) values ('49N110973N0', '854252370');
879 insert into GatitaEmprendedora.TeléfonoEmprendedor (RFC, TeléfonoEmprendedor) values ('7BF8TAS5B90D', '2473423688');
880 insert into GatitaEmprendedora.TeléfonoEmprendedor (RFC, TeléfonoEmprendedor) values ('R6362K9RJ0TM', '7556777459');
881 insert into GatitaEmprendedora.TeléfonoEmprendedor (RFC, TeléfonoEmprendedor) values ('MX5XABW0Q681M', '3749358293');
882 insert into GatitaEmprendedora.TeléfonoEmprendedor (RFC, TeléfonoEmprendedor) values ('0912Y15311TM0', '877925674');
883 insert into GatitaEmprendedora.TeléfonoEmprendedor (RFC, TeléfonoEmprendedor) values ('681094889160A', '49197968');
884 insert into GatitaEmprendedora.TeléfonoEmprendedor (RFC, TeléfonoEmprendedor) values ('2877PM61KX8A4', '7726744593');
885 insert into GatitaEmprendedora.TeléfonoEmprendedor (RFC, TeléfonoEmprendedor) values ('FM265CH704M80', '0853441864');
886 insert into GatitaEmprendedora.TeléfonoEmprendedor (RFC, TeléfonoEmprendedor) values ('MCXH7SM909TSY', '471539518');
887 insert into GatitaEmprendedora.TeléfonoEmprendedor (RFC, TeléfonoEmprendedor) values ('48227C680JAH', '6243126716');
888 insert into GatitaEmprendedora.TeléfonoEmprendedor (RFC, TeléfonoEmprendedor) values ('4K4K3SM9W9R88', '8343917952');
889 insert into GatitaEmprendedora.TeléfonoEmprendedor (RFC, TeléfonoEmprendedor) values ('N16627X93M7', '6388501990');
890 insert into GatitaEmprendedora.TeléfonoEmprendedor (RFC, TeléfonoEmprendedor) values ('INX0UPKVR0661', '3223772130');
891 insert into GatitaEmprendedora.TeléfonoEmprendedor (RFC, TeléfonoEmprendedor) values ('P67945Q156881', '1554665697');
892 insert into GatitaEmprendedora.TeléfonoEmprendedor (RFC, TeléfonoEmprendedor) values ('L8QJSHGWIFHC2', '4273647330');
893 insert into GatitaEmprendedora.TeléfonoEmprendedor (RFC, TeléfonoEmprendedor) values ('QNY8E150XRF', '862019675');
894 insert into GatitaEmprendedora.TeléfonoEmprendedor (RFC, TeléfonoEmprendedor) values ('W3QQL8ABCQNGK', '567879475');
895 insert into GatitaEmprendedora.TeléfonoEmprendedor (RFC, TeléfonoEmprendedor) values ('7151X2HTW2620', '901169676');
896 insert into GatitaEmprendedora.TeléfonoEmprendedor (RFC, TeléfonoEmprendedor) values ('IV095R1817M', '6579377937');
897 insert into GatitaEmprendedora.TeléfonoEmprendedor (RFC, TeléfonoEmprendedor) values ('W8490WH3KL3', '0345930515');
898 insert into GatitaEmprendedora.TeléfonoEmprendedor (RFC, TeléfonoEmprendedor) values ('U1P58DRXUFCS', '4610275854');
899 insert into GatitaEmprendedora.TeléfonoEmprendedor (RFC, TeléfonoEmprendedor) values ('17NLLZNH677YQ', '173673116');
900 insert into GatitaEmprendedora.TeléfonoEmprendedor (RFC, TeléfonoEmprendedor) values ('V9JXYPDAU7779', '7784403585');
901 insert into GatitaEmprendedora.TeléfonoEmprendedor (RFC, TeléfonoEmprendedor) values ('ZKP8R46U6F71', '3652074474');
902 insert into GatitaEmprendedora.TeléfonoEmprendedor (RFC, TeléfonoEmprendedor) values ('KT4YMK047X1', '664429227');

```

Y así se ve la tabla poblada:

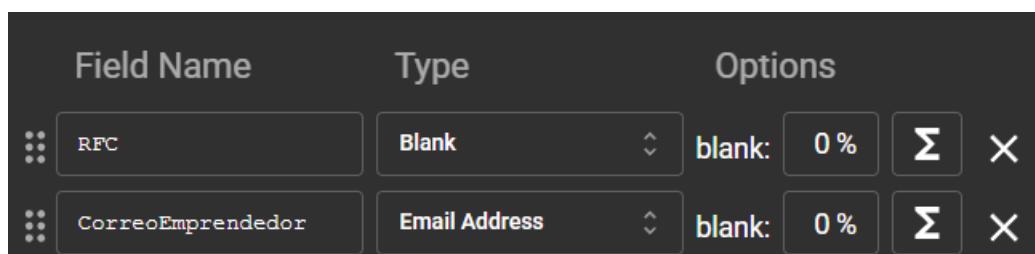


The screenshot shows the pgAdmin interface with the 'telefonoemprendedor' table selected. The table has two columns: 'rfc' and 'telefonoemprendedor'. The data consists of 300 records, each containing a unique string value for 'rfc' and a numeric value for 'telefonoemprendedor'. The records range from row 272 to 300.

	rfc	telefonoemprendedor
272	5HAS1GGLO6Y3Y	4044859737
273	A2252C395X39X	6409081178
274	O050S0S8Z41MK	5957058917
275	49V1109A75MJ0	8554252370
276	7BF8TASZ589D3	2473423688
277	R6362K9RJ0TNK	7556777459
278	MX5XA0VRQ68IM	3749358293
279	091ZY153IJTW0	8779257674
280	681D94880H60A	4921972968
281	2877EM61KXA84	7726744593
282	FWZ65CN7O4MB0	0853441864
283	NCXWIS9M99T5Y	4715395518
284	4822A7C60DJ4H	6243126716
285	4K4JKSXN9MRR8	8343917952
286	N16627X932H7J	6388507999
287	IN7XUPKYR6061	3223772139
288	P67345Q156881	1554663697
289	75I5ML6P59669	8899810434
290	L8QLSH6DNFHC2	4273647330
291	6N1YD8E150XKF	8620219675
292	WK9QL8A0CDNGK	5678795475
293	7151X2HIW202U	9011696768
294	NYO95RI81T5WL	6579377937
295	M849QBHVH3KL3	0345030515
296	UTP5BD8RXUFC5	4610275854
297	17NIL2NH677YQ	1736731163
298	V93XYPDAH7779	7784403585
299	ZKP8R46U67F7I	3652074474
300	KT4YMOKO47TX1	6644429227

### Tabla CorreoEmprendedor

Captura de Mockaroo:



The screenshot shows the Mockaroo field configuration interface for the 'CorreoEmprendedor' table. It includes columns for 'Field Name', 'Type', and 'Options'.

Field Name	Type	Options
RFC	Blank	blank: 0 % $\Sigma$ X
CorreoEmprendedor	Email Address	blank: 0 % $\Sigma$ X

Como se puede ver, generamos la columna de con valores null y esto es porque Mockaroo nos genera números aleatorios cada vez, por lo que necesitamos garantizar que haya en la BD. Con ayuda de un editor de código (Visual Studio Code) copiamos y pegamos de manera eficiente todos datos ya existentes de la tabla .



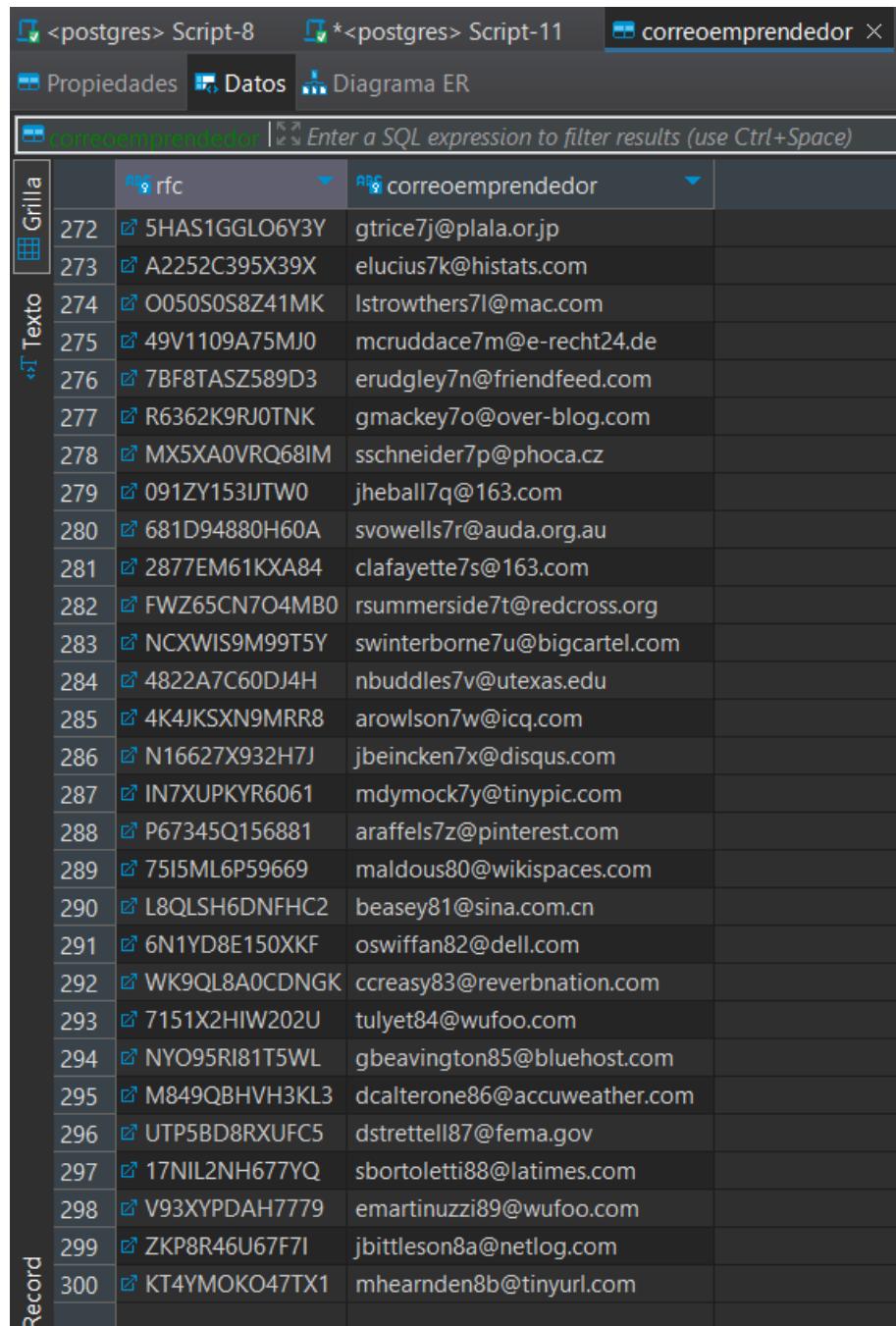
```

File Edit Selection View Go Run Terminal Help ← → 🔍 FundamentosBD
EXPLORER DML(1).sql GatitaEmprendedora.CorreoEmprendedor.sql
C: > Users > victo > Downloads > GatitaEmprendedora.CorreoEmprendedor.sql
267 insert into GatitaEmprendedora.CorreoEmprendedor (RFC, CorreoEmprendedor ) values (null, 'kolivetta7e@amazon.co.uk');
268 insert into GatitaEmprendedora.CorreoEmprendedor (RFC, CorreoEmprendedor ) values (null, 'fdevigne7f@free.fr');
269 insert into GatitaEmprendedora.CorreoEmprendedor (RFC, CorreoEmprendedor ) values (null, 'mpettingill7g@people.com.cn');
270 insert into GatitaEmprendedora.CorreoEmprendedor (RFC, CorreoEmprendedor ) values (null, 'bchiechio7h@mayoclinic.com');
271 insert into GatitaEmprendedora.CorreoEmprendedor (RFC, CorreoEmprendedor ) values (null, 'gdo7i@foxnews.com');
272 insert into GatitaEmprendedora.CorreoEmprendedor (RFC, CorreoEmprendedor ) values (null, 'grtrice7j@lala.or.jp');
273 insert into GatitaEmprendedora.CorreoEmprendedor (RFC, CorreoEmprendedor ) values (null, 'elucius7k@histats.com');
274 insert into GatitaEmprendedora.CorreoEmprendedor (RFC, CorreoEmprendedor ) values (null, 'lstromthers7l@mac.com');
275 insert into GatitaEmprendedora.CorreoEmprendedor (RFC, CorreoEmprendedor ) values (null, 'mcrudace7m@rechta24.de');
276 insert into GatitaEmprendedora.CorreoEmprendedor (RFC, CorreoEmprendedor ) values (null, 'erudgley7n@friendfeed.com');
277 insert into GatitaEmprendedora.CorreoEmprendedor (RFC, CorreoEmprendedor ) values (null, 'gmackey7o@over-blog.com');
278 insert into GatitaEmprendedora.CorreoEmprendedor (RFC, CorreoEmprendedor ) values (null, 'sschneider7p@phoca.cz');
279 insert into GatitaEmprendedora.CorreoEmprendedor (RFC, CorreoEmprendedor ) values (null, 'jheball7q@163.com');
280 insert into GatitaEmprendedora.CorreoEmprendedor (RFC, CorreoEmprendedor ) values (null, 'svowells7r@auda.org.au');
281 insert into GatitaEmprendedora.CorreoEmprendedor (RFC, CorreoEmprendedor ) values (null, 'clafayette7s@163.com');
282 insert into GatitaEmprendedora.CorreoEmprendedor (RFC, CorreoEmprendedor ) values (null, 'rsummerside7t@redcross.org');
283 insert into GatitaEmprendedora.CorreoEmprendedor (RFC, CorreoEmprendedor ) values (null, 'swinterborne7u@bigcartel.com');
284 insert into GatitaEmprendedora.CorreoEmprendedor (RFC, CorreoEmprendedor ) values (null, 'nbubbles7v@utexas.edu');
285 insert into GatitaEmprendedora.CorreoEmprendedor (RFC, CorreoEmprendedor ) values (null, 'arowlson7w@icq.com');
286 insert into GatitaEmprendedora.CorreoEmprendedor (RFC, CorreoEmprendedor ) values (null, 'jbeincken7x@disqus.com');
287 insert into GatitaEmprendedora.CorreoEmprendedor (RFC, CorreoEmprendedor ) values (null, 'mdymocky7y@tinypic.com');
288 insert into GatitaEmprendedora.CorreoEmprendedor (RFC, CorreoEmprendedor ) values (null, 'araffels7z@pinterest.com');
289 insert into GatitaEmprendedora.CorreoEmprendedor (RFC, CorreoEmprendedor ) values (null, 'malodus80@wikispaces.com');
290 insert into GatitaEmprendedora.CorreoEmprendedor (RFC, CorreoEmprendedor ) values (null, 'beasey81@sina.com.cn');
291 insert into GatitaEmprendedora.CorreoEmprendedor (RFC, CorreoEmprendedor ) values (null, 'oswiffan82@dell.com');
292 insert into GatitaEmprendedora.CorreoEmprendedor (RFC, CorreoEmprendedor ) values (null, 'ccreasy83@reverbnation.com');
293 insert into GatitaEmprendedora.CorreoEmprendedor (RFC, CorreoEmprendedor ) values (null, 'tulyet84@wufoo.com');
294 insert into GatitaEmprendedora.CorreoEmprendedor (RFC, CorreoEmprendedor ) values (null, 'gbeavington85@bluehost.com');
295 insert into GatitaEmprendedora.CorreoEmprendedor (RFC, CorreoEmprendedor ) values (null, 'dcalterone86@acuweather.com');
296 insert into GatitaEmprendedora.CorreoEmprendedor (RFC, CorreoEmprendedor ) values (null, 'dstrettell87@fema.gov');
297 insert into GatitaEmprendedora.CorreoEmprendedor (RFC, CorreoEmprendedor ) values (null, 'sbertoletti88@latimes.com');
298 insert into GatitaEmprendedora.CorreoEmprendedor (RFC, CorreoEmprendedor ) values (null, 'emartinuzzi89@wufoo.com');
299 insert into GatitaEmprendedora.CorreoEmprendedor (RFC, CorreoEmprendedor ) values (null, 'jbittleson8a@netlog.com');
300 insert into GatitaEmprendedora.CorreoEmprendedor (RFC, CorreoEmprendedor ) values (null, 'mhearnden8b@tinyurl.com');

File Edit Selection View Go Run Terminal Help ← → 🔍 FundamentosBD
EXPLORER DML(1).sql GatitaEmprendedora.CorreoEmprendedor.sql
C: > Users > victo > Downloads > GatitaEmprendedora.CorreoEmprendedor.sql
266 insert into GatitaEmprendedora.CorreoEmprendedor (RFC, CorreoEmprendedor ) values ('CEDGS43040KB5', 'wolliffe7d@devhub.com');
267 insert into GatitaEmprendedora.CorreoEmprendedor (RFC, CorreoEmprendedor ) values ('56Z1AECK6LLV', 'kolivetta7e@amazon.co.uk');
268 insert into GatitaEmprendedora.CorreoEmprendedor (RFC, CorreoEmprendedor ) values ('6VGDUMLW1RQD', 'fdevigne7f@free.fr');
269 insert into GatitaEmprendedora.CorreoEmprendedor (RFC, CorreoEmprendedor ) values ('K4LGLBKMGN2', 'mpettingill7g@people.com.cn');
270 insert into GatitaEmprendedora.CorreoEmprendedor (RFC, CorreoEmprendedor ) values ('CP7KMU671W5W', 'bchiechio7h@mayoclinic.com');
271 insert into GatitaEmprendedora.CorreoEmprendedor (RFC, CorreoEmprendedor ) values ('9WMO15D6U9QQA', 'gdo7i@foxnews.com');
272 insert into GatitaEmprendedora.CorreoEmprendedor (RFC, CorreoEmprendedor ) values ('5HAS1GGLO6Y3Y', 'grtrice7j@lala.or.jp');
273 insert into GatitaEmprendedora.CorreoEmprendedor (RFC, CorreoEmprendedor ) values ('A2252C395X39X', 'elucius7k@histats.com');
274 insert into GatitaEmprendedora.CorreoEmprendedor (RFC, CorreoEmprendedor ) values ('0959508741MK', 'lstromthers7l@mac.com');
275 insert into GatitaEmprendedora.CorreoEmprendedor (RFC, CorreoEmprendedor ) values ('49V1109A75M0', 'mcrudace7m@rechta24.de');
276 insert into GatitaEmprendedora.CorreoEmprendedor (RFC, CorreoEmprendedor ) values ('7BF8TASZ58D93', 'erudgley7n@friendfeed.com');
277 insert into GatitaEmprendedora.CorreoEmprendedor (RFC, CorreoEmprendedor ) values ('R6362K9R30TNK', 'gmackey7o@over-blog.com');
278 insert into GatitaEmprendedora.CorreoEmprendedor (RFC, CorreoEmprendedor ) values ('MX5XA0VR068IM', 'sschneider7p@phoca.cz');
279 insert into GatitaEmprendedora.CorreoEmprendedor (RFC, CorreoEmprendedor ) values ('091ZY1513IJTW0', 'jheball7q@163.com');
280 insert into GatitaEmprendedora.CorreoEmprendedor (RFC, CorreoEmprendedor ) values ('681D94880H6A', 'svowells7r@auda.org.au');
281 insert into GatitaEmprendedora.CorreoEmprendedor (RFC, CorreoEmprendedor ) values ('287EM61KXKA84', 'clafayette7s@163.com');
282 insert into GatitaEmprendedora.CorreoEmprendedor (RFC, CorreoEmprendedor ) values ('FNZ65CN704MB0', 'rsummerside7t@redcross.org');
283 insert into GatitaEmprendedora.CorreoEmprendedor (RFC, CorreoEmprendedor ) values ('NCWXA0VR068IM', 'swinterborne7u@bigcartel.com');
284 insert into GatitaEmprendedora.CorreoEmprendedor (RFC, CorreoEmprendedor ) values ('4822A7C60D4JH', 'nbubbles7v@utexas.edu');
285 insert into GatitaEmprendedora.CorreoEmprendedor (RFC, CorreoEmprendedor ) values ('4K4JKSXN9MR8', 'arowlson7w@icq.com');
286 insert into GatitaEmprendedora.CorreoEmprendedor (RFC, CorreoEmprendedor ) values ('N16627X932H7', 'jbeincken7x@disqus.com');
287 insert into GatitaEmprendedora.CorreoEmprendedor (RFC, CorreoEmprendedor ) values ('INTXUPKYR6061', 'mdymocky7y@tinypic.com');
288 insert into GatitaEmprendedora.CorreoEmprendedor (RFC, CorreoEmprendedor ) values ('P6734S5Q156881', 'araffels7z@pinterest.com');
289 insert into GatitaEmprendedora.CorreoEmprendedor (RFC, CorreoEmprendedor ) values ('7515MLGP59669', 'malodus80@wikispaces.com');
290 insert into GatitaEmprendedora.CorreoEmprendedor (RFC, CorreoEmprendedor ) values ('L8QLSH60NFHC2', 'beasey81@sina.com.cn');
291 insert into GatitaEmprendedora.CorreoEmprendedor (RFC, CorreoEmprendedor ) values ('6NYD8E150XKF', 'oswiffan82@dell.com');
292 insert into GatitaEmprendedora.CorreoEmprendedor (RFC, CorreoEmprendedor ) values ('WK9QL8A0CDNGK', 'ccreasy83@reverbnation.com');
293 insert into GatitaEmprendedora.CorreoEmprendedor (RFC, CorreoEmprendedor ) values ('7151X2HIW20U', 'tulyet84@wufoo.com');
294 insert into GatitaEmprendedora.CorreoEmprendedor (RFC, CorreoEmprendedor ) values ('NY095R181T5WL', 'gbeavington85@bluehost.com');
295 insert into GatitaEmprendedora.CorreoEmprendedor (RFC, CorreoEmprendedor ) values ('M849QBHM3KL3', 'dcalterone86@acuweather.com');
296 insert into GatitaEmprendedora.CorreoEmprendedor (RFC, CorreoEmprendedor ) values ('UTPSBD8RXUFCS', 'dstrettell87@fema.gov');
297 insert into GatitaEmprendedora.CorreoEmprendedor (RFC, CorreoEmprendedor ) values ('17N1L2H677YQ', 'sbertoletti88@latimes.com');
298 insert into GatitaEmprendedora.CorreoEmprendedor (RFC, CorreoEmprendedor ) values ('V93XYPDAAH7779', 'emartinuzzi89@wufoo.com');
299 insert into GatitaEmprendedora.CorreoEmprendedor (RFC, CorreoEmprendedor ) values ('ZKPR846U67F71', 'jbittleson8a@netlog.com');
300 insert into GatitaEmprendedora.CorreoEmprendedor (RFC, CorreoEmprendedor ) values ('KT4YAMOK047TX1', 'mhearnden8b@tinyurl.com');

```

Y así se ve la tabla poblada:



The screenshot shows the pgAdmin interface with the database 'correoemprendedor' selected. The table 'correoemprendedor' is displayed in a grid view. The columns are labeled 'rfc' and 'correoemprendedor'. The data consists of 300 records, each containing a unique identifier (row number) and a pair of values. The 'rfc' column contains various alphanumeric strings, and the 'correoemprendedor' column contains corresponding email addresses.

	rfc	correoemprendedor
272	5HAS1GGLO6Y3Y	gtrice7j@plala.or.jp
273	A2252C395X39X	elucius7k@histats.com
274	O050S0S8Z41MK	Istrowthers7l@mac.com
275	49V1109A75MJ0	mcruddace7m@e-recht24.de
276	7BF8TASZ589D3	erudgley7n@friendfeed.com
277	R6362K9RJ0TNK	gmackey7o@over-blog.com
278	MX5XA0VRQ68IM	sschneider7p@phoca.cz
279	091ZY153IJTW0	jheball7q@163.com
280	681D94880H60A	svowells7r@auda.org.au
281	2877EM61KXA84	clafayette7s@163.com
282	FWZ65CN7O4MB0	rsummerside7t@redcross.org
283	NCXWIS9M99T5Y	swinterborne7u@bigcartel.com
284	4822A7C60DJ4H	nbuddles7v@utexas.edu
285	4K4JKSXN9MRR8	arowlson7w@icq.com
286	N16627X932H7J	jbeincken7x@disqus.com
287	IN7XUPKXR6061	mdymock7y@tinypic.com
288	P67345Q156881	araffels7z@pinterest.com
289	75I5ML6P59669	maldous80@wikispaces.com
290	L8QLSH6DNFH2	beasey81@sina.com.cn
291	6N1YD8E150XKF	oswiffan82@dell.com
292	WK9QL8A0CDNGK	ccreasy83@reverbNation.com
293	7151X2HIW202U	tulyet84@wufoo.com
294	NYO95RI81T5WL	gbeavington85@bluehost.com
295	M849QBHVH3KL3	dcalterone86@accuweather.com
296	UTP5BD8RXUFC5	dstrettell87@fema.gov
297	17NIL2NH677YQ	sbortoletti88@latimes.com
298	V93XPDAH7779	emartinuzzi89@wufoo.com
299	ZKP8R46U67F7I	jbittleson8a@netlog.com
300	KT4YMOKO47TX1	mhearnden8b@tinyurl.com

Ahora bien, con las tablas **Cliente**, **Bazar**, **Negocio** y **Emprendedor** ya podemos poblar **Ticket**.

### Tabla Ticket

Captura de Mockaroo:

Field Name	Type	Options
IdTicket	Row Number	blank: 0 % $\Sigma$ X
IdCliente	Number	min: 1 max: 300 decimals: 0 blank: 0 % $\Sigma$ X
IdBazar	Number	min: 1 max: 300 decimals: 0 blank: 0 % $\Sigma$ X
IdNegocio	Number	min: 1 max: 300 decimals: 0 blank: 0 % $\Sigma$ X
RFC	SQL Expression	example: DEFAULT blank: 0 % $\Sigma$ X
ComisionBazar	Number	min: 50 max: 200 decimals: 1 blank: 0 % $\Sigma$ X
<a href="#">+ ADD ANOTHER FIELD</a> <a href="#">GENERATE FIELDS USING AI...</a>		

Como se puede ver, tenemos una restricción, de la siguiente manera:

```
concat(' (SELECT rfc FROM GATITAEMPRENDEDORA.EMPRENDEDOR WHERE IDNEGOCIO = ', field('IdNegocio'), ')')
```

Y así se ve la tabla poblada:

	Propiedades	Datos	Diagrama ER	postgres	Bases de Datos	postgres
	ticket	Enter a SQL expression to filter results (use Ctrl+Space)				
Grilla	idticket	idcliente	idbazar	idnegocio	rfc	comisionbazar
272	272	238	207	6	IR881G742MJ10	52,7
273	273	172	48	175	E18QR940Q83S6	141,5
274	274	82	151	106	129854MD78436	117,6
275	275	106	45	269	K4LLGLBKG9N2	123,5
276	276	52	175	116	5JAM598C1229X	126,5
277	277	52	159	113	CA9FH200AE155	125,9
278	278	193	169	192	IB0DCFO2TLMH6	186,8
279	279	29	83	265	C08OK0YX30X0K	154,9
280	280	38	33	48	G7QJ4U84HNC15	141
281	281	288	168	105	3D9G9IB8DT5D2	119
282	282	295	96	91	T6LC1E11SS536	144,1
283	283	16	248	75	SNRB4HZ3W3L11	111
284	284	248	211	56	Y3IO657R770PS	194,7
285	285	181	219	74	74W8SY91V1P1	146
286	286	166	113	278	MX5XA0VRQ68IM	84
287	287	259	53	299	ZKP8R46U67F7I	78,3
288	288	294	214	79	TO74OC52B34T	166,3
289	289	224	69	287	IN7XUPKYR6061	183,6
290	290	22	129	255	P7LG68430294	160,4
291	291	295	87	92	9ETG2KFLQ1K8I	54,8
292	292	159	175	64	ZD9Y478XPM15I	164
293	293	134	150	281	2877EM61KXA84	169
294	294	117	259	241	85HN70B19IGVK	199,5
295	295	229	67	96	D1E1578LAMCJM	101,9
296	296	143	263	294	NYO95RI81T5WL	115,4
297	297	179	29	194	3277WUR8G88Y5	123,5
298	298	17	215	193	T2M83J57N3A66	54,7
299	299	237	13	210	0H3WZ1574ZN9H	96,7
300	300	170	216	188	15J59PK9TB4QA	143

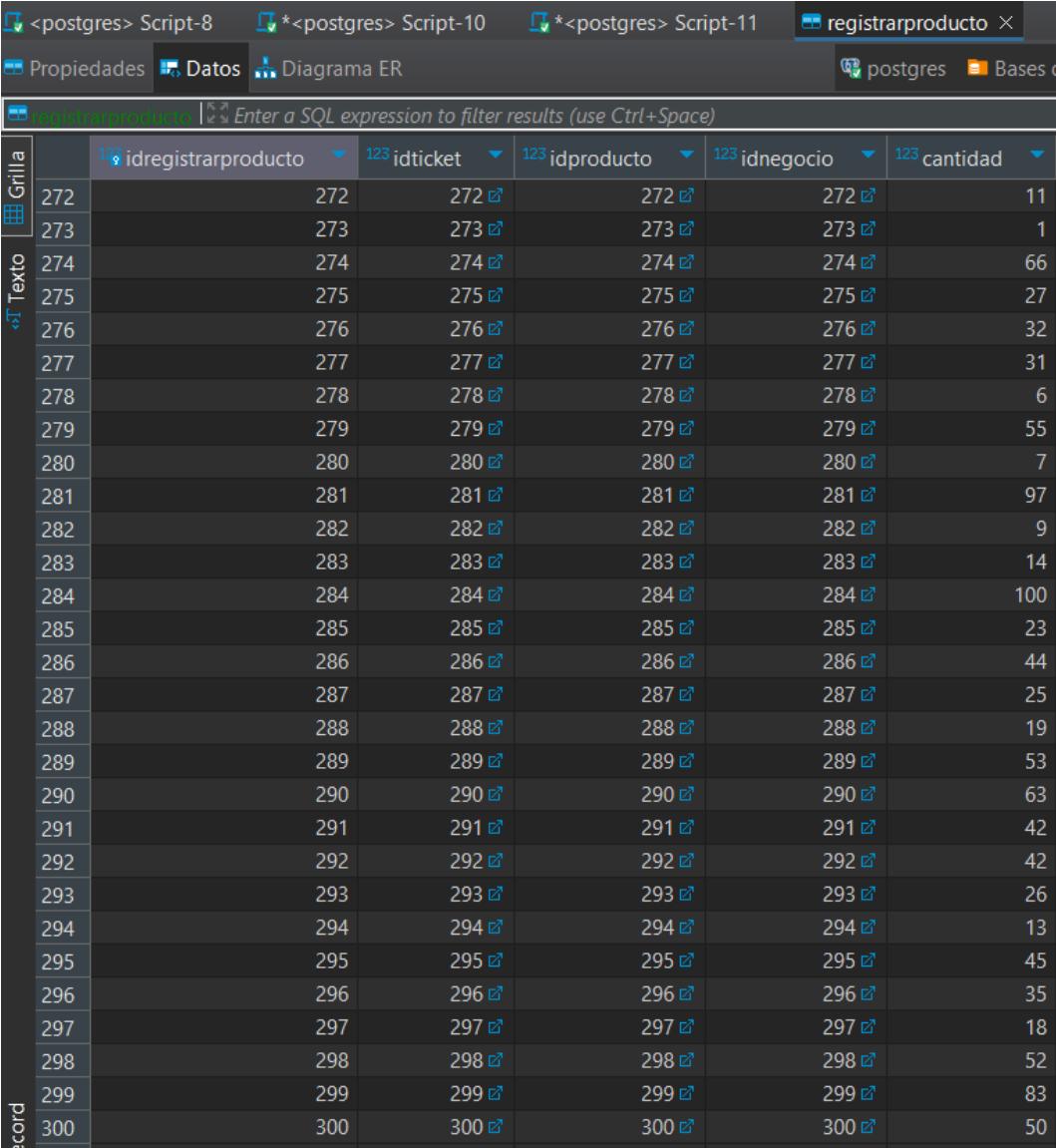
Entonces ya podemos pasar a poblar **Registrar Producto y Servicio**.

### Tabla Registrar Producto

Captura de Mockaroo:

Field Name	Type	Options		
IdRegistrarProducto	Row Number	blank: 0 %	$\Sigma$	X
IdTicket	Row Number	blank: 0 %	$\Sigma$	X
IdProducto	Row Number	blank: 0 %	$\Sigma$	X
IdNegocio	Row Number	blank: 0 %	$\Sigma$	X
Cantidad	Number	min: 1 max: 100 decimals: 0 blank: 0 %	$\Sigma$	X

Y así se ve la tabla poblada:



The screenshot shows the DBeaver database interface with the 'registrarproducto' table selected. The table has five columns: idregistrarproducto, idticket, idproducto, idnegocio, and cantidad. The data is populated with values ranging from 272 to 300. The 'Grilla' (grid) view is shown on the left, and the 'Texto' (text) view is on the right. The top navigation bar includes tabs for 'Script-8', 'Script-10', 'Script-11', and 'registrarproducto'. The bottom status bar shows 'postgres' and 'Bases d'.

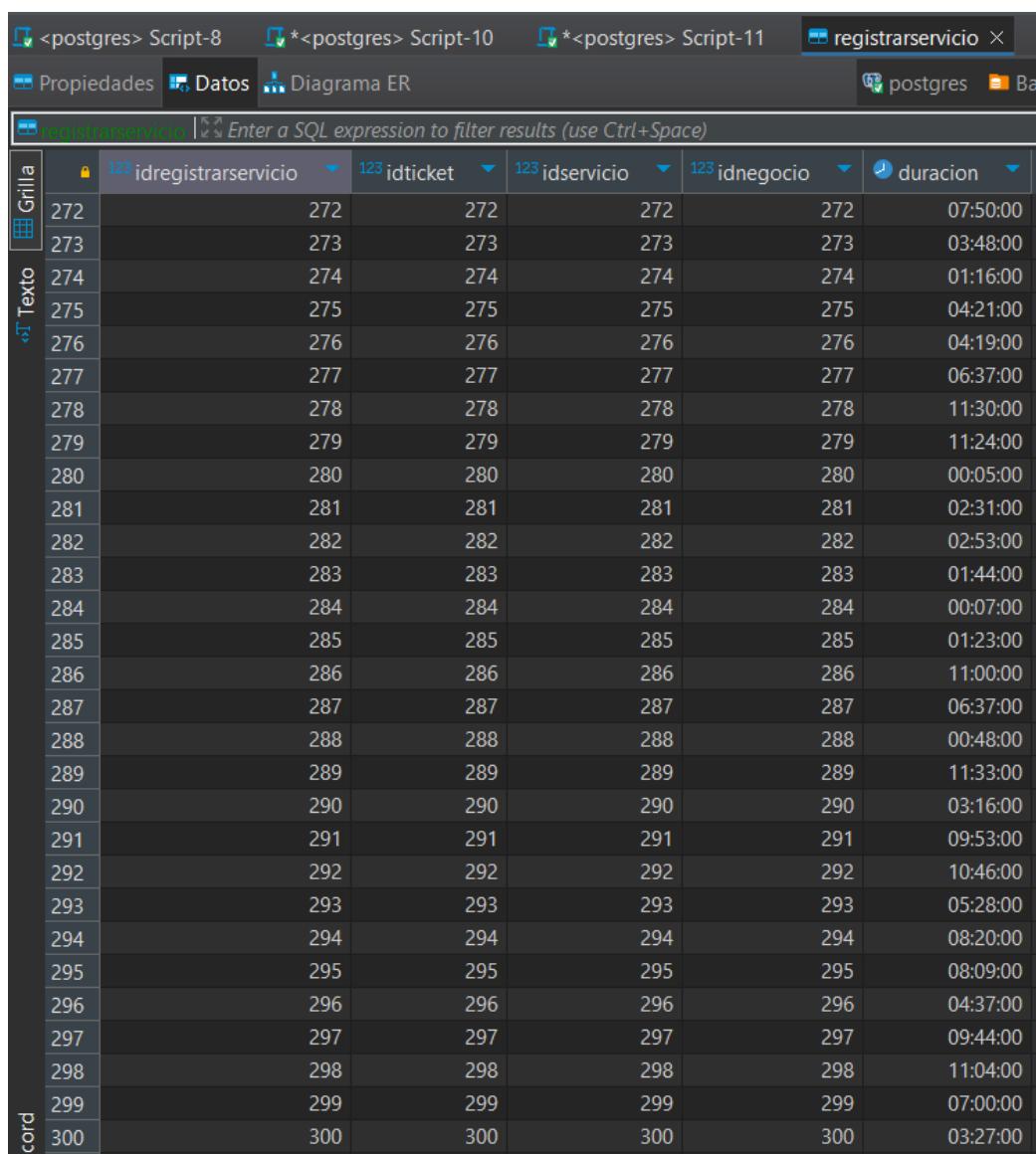
idregistrarproducto	idticket	idproducto	idnegocio	cantidad
272	272	272	272	11
273	273	273	273	1
274	274	274	274	66
275	275	275	275	27
276	276	276	276	32
277	277	277	277	31
278	278	278	278	6
279	279	279	279	55
280	280	280	280	7
281	281	281	281	97
282	282	282	282	9
283	283	283	283	14
284	284	284	284	100
285	285	285	285	23
286	286	286	286	44
287	287	287	287	25
288	288	288	288	19
289	289	289	289	53
290	290	290	290	63
291	291	291	291	42
292	292	292	292	42
293	293	293	293	26
294	294	294	294	13
295	295	295	295	45
296	296	296	296	35
297	297	297	297	18
298	298	298	298	52
299	299	299	299	83
300	300	300	300	50

## Tabla Registrar Servicio

Captura de Mockaroo:

Field Name	Type	Options					
IdRegistrarServicio	Row Number	blank:	0 %	$\Sigma$	X		
IdTicket	Row Number	blank:	0 %	$\Sigma$	X		
IdServicio	Row Number	blank:	0 %	$\Sigma$	X		
IdNegocio	Row Number	blank:	0 %	$\Sigma$	X		
Duracion	Time	from	12:00 AM	to	11:59 AM	format:	24 Hour
						blank:	0 %
						$\Sigma$	X

Y así se ve la tabla poblada:



The screenshot shows the DBeaver interface with the 'registrarservicio' table selected. The table has six columns: idregistrarServicio, idticket, idservicio, idnegocio, duracion, and a primary key column (idregistrarServicio) which is hidden. The data consists of 300 rows, each containing unique values for the first four columns and a duration value for the fifth column. The 'duracion' column contains values such as 07:50:00, 03:48:00, 01:16:00, etc.

Grilla	idregistrarServicio	idticket	idservicio	idnegocio	duracion
272		272	272	272	07:50:00
273		273	273	273	03:48:00
274		274	274	274	01:16:00
275		275	275	275	04:21:00
276		276	276	276	04:19:00
277		277	277	277	06:37:00
278		278	278	278	11:30:00
279		279	279	279	11:24:00
280		280	280	280	00:05:00
281		281	281	281	02:31:00
282		282	282	282	02:53:00
283		283	283	283	01:44:00
284		284	284	284	00:07:00
285		285	285	285	01:23:00
286		286	286	286	11:00:00
287		287	287	287	06:37:00
288		288	288	288	00:48:00
289		289	289	289	11:33:00
290		290	290	290	03:16:00
291		291	291	291	09:53:00
292		292	292	292	10:46:00
293		293	293	293	05:28:00
294		294	294	294	08:20:00
295		295	295	295	08:09:00
296		296	296	296	04:37:00
297		297	297	297	09:44:00
298		298	298	298	11:04:00
299		299	299	299	07:00:00
300		300	300	300	03:27:00

# Bibliography

- [1] PostgreSQL Global Development Group. *Documentation: Referential Integrity Constraints*. Disponible en: <https://www.postgresql.org/docs/current/ddl-constraints.html>
- [2] W3Schools. *SQL FOREIGN KEY Constraint*. Disponible en: [https://www.w3schools.com/sql/sql\\_foreignkey.asp](https://www.w3schools.com/sql/sql_foreignkey.asp)
- [3] DBA Stack Exchange. *Disadvantages of ON DELETE CASCADE*. Disponible en: <https://dba.stackexchange.com/questions/64710/disadvantages-of-on-delete-cascade>
- [4] TechTarget. *Foreign key constraints in PostgreSQL*. Disponible en: <https://www.techtarget.com/searchdatamanagement/definition/foreign-key>