**AUTORENTAL**

# 

**CAMILO MACHUCA VEGA**

**T2**

**PEDRO FELIPE GÓMEZ BONILLA**

**CAMPUSLANDS**

**SANTBOX**

**GRUPO: T2  
 MYSQL**

**TIBÚ**

**2024**

Tabla de Contenidos

[**Introducción 4**](#_xy776j9pw1un)

[**Caso de Estudio 5**](#_x5wjmxr9s7lx)

[**Planificación 5**](#_8muzr1pu3mlz)

[Construcción del Modelo Conceptual 5](#_akgvn7l12xs2)

[Descripción 5](#_qxuss0va0z8)

[Gráfica 5](#_u45t31dnlow2)

[Descripción Técnica 6](#_gww41npptmyg)

[Construcción del Modelo Lógico 6](#_nzoi4crbe63p)

[Descripción 6](#_46z7wrz43fjb)

[Gráfica 6](#_cx639fevpp57)

[Descripción Técnica 6](#_vemj55jop1dw)

[Normalización del Modelo Lógico 6](#_j975ki11pvw9)

[Primera Forma Normal (1FN) 7](#_ipsnvw5v2v0u)

[Descripción 7](#_yudex7qmv5l6)

[Gráfica 7](#_u1g2e96ymz30)

[Descripción Técnica 7](#_rjmpapoxneod)

[Segunda Forma Normal (2FN) 7](#_3fyocgo001fx)

[Descripción 8](#_n31b5bkefr23)

[Gráfica 8](#_mywvu5xf7mjo)

[Descripción Técnica 8](#_i9ctpni5mu37)

[Tercera Forma Normal (3FN) 8](#_aeip5fb7t3s5)

[Descripción 8](#_v6plqfy3484u)

[Gráfica 8](#_bcqznd6b310i)

[Descripción Técnica 8](#_qaz530fwxr49)

[Construcción del Modelo Físico 9](#_muq8ro9qbje6)

[Descripción 9](#_5g32igevz4r8)

[Código 9](#_ryjwvstap29b)

[Descripción Técnica 10](#_f9n2yx58hlja)

[Diagrama E-R 10](#_l2vl1lqjv6nt)

[Descripción 10](#_b542o9532ses)

[Gráfica 10](#_8wg77tt5rsnt)

[Descripción Técnica 10](#_whj2rpgnzs4v)

[Tablas 11](#_s3pvkylovom)

[Descripción 11](#_ye40r8z6ajh3)

[Gráfica 11](#_mgogeuideyfr)

[Descripción Técnica 11](#_4tau2idq5j2j)

[Relaciones entre Tablas 11](#_v02u5svki3o6)

[Descripción 11](#_mn4p1kzcs5d)

[Gráfica 12](#_ahrg4l6677cx)

[Descripción Técnica 12](#_v3u2pmnyvhxt)

[Inserción de Datos 12](#_khdudtfa7c4u)

[Descripción 12](#_r9p210nyg848)

[Gráfica 12](#_wpdzy6svxf2h)

[Descripción Técnica 12](#_cudxay80qe23)

[**Referencias 13**](https://docs.google.com/document/d/1SRuuADof0v8w4HWqaVgGDCH4xK5B4f5NYmcsIqbwk-s/edit?tab=t.0#heading=h.ywn691xazve)

# 

# Introducción

En el siguiente documento se presenta la realización de una base de datos la cual se relaciona con un sistema de información para una empresa de alquiler de vehículos llamada AutoRental para esto se comenzó construyendo el modelo conceptual en el cual se sacan las entidades y atributos que va a tener la base de datos en este caso se crearon cinco entidades con sus atributos y las relaciones entre ellas.

A partir del modelo conceptual realizamos el modelo lógico que consiste en la creación de tablas estas pasarían a representar las entidades del modelo conceptual, además este modelo tiene definido que tipo de datos va a tener cada atributo, así como también las llaves primarias y foráneas.

finalizamos con la creación del diagrama físico que consiste en la perfección y corrección de las tablas que realizamos en el modelo lógico con estos diagramas se busca ofrecer una vista detallada de cada proceso por el que pasa la creación de la base de datos para el sistema de información de alquiler de vehículos AutoRental.

# 

# 

# Caso de Estudio

La empresa donde usted trabaja ha sido contratada para desarrollar un sistema de

información para una empresa de alquiler de vehículos llamada AutoRental, y usted ha sido

designado para diseñar una base de datos para ese sistema de información.

AutoRental cuenta con 5 sucursales en diferentes ciudades y se proyecta a expandirse a otras

ciudades del país y cuenta con una flota propia de vehículos de diferentes tipos, modelos

(año), capacidad, etc.

Los clientes de AutoRental podrán alquilar un vehículo en una sucursal y entregarlo en otra

sucursal.

AutoRental ofrece descuentos sobre diferentes tipos de vehículos a lo largo del año.

Los valores de alquiler dependen del tipo de vehículo (sedán, compacto, camioneta platón,

camioneta lujo, deportivo, etc) y se cobran por días y/o semanas. Por ejemplo, si un alquila

un vehículo por 9 días, el valor cotizado será de 1 semana y 2 días.

Si un cliente entrega el vehículo pasado la fecha de entrega contratada, se cobrarán los días

adicionales con un incremento del 8%.

# Planificación

**Ejecución**

Una vez se analizó la información requerida para el alquiler de vehículos AutoRental, se inició la creación del modelo conceptual. Este modelo proporciona una descripción detallada de las necesidades de información que están detrás del diseño de una base de datos. Representa los conceptos principales de la base de datos y las relaciones entre ellos.

## Construcción del Modelo Conceptual

En este modelo, se identificaron las entidades, sus atributos y las relaciones entre ellas. Este modelo conceptual proporciona una visión detallada de cómo va a ser la base de datos además se tienen unas restricciones y relaciones.

### Descripción

Este paso tiene la estructura de la base de datos que está formada por cinco entidades y sus atributos las cuales son:

1. **Sucursales:**

* id\_sucursal: id único de sucursal.
* Celular: Numero de celular de la sucursal.
* dirección: dirección donde se encuentra la sucursal.
* ciudad: ciudad donde se encuentra la sucursal.
* Teléfono\_fijo: teléfono fijo de la sucursal.
* Correo\_electronico: correo electrónico de la sucursal.

1. **Empleados:**

* Id\_empleados: id único de empleados.
* Ciudad\_residencia: ciudad donde reside el empleado.
* cedula: número de cedula del empleado.
* nombre: nombre del empleado.
* Apellidos: apellidos del empleado.
* Dirección: dirección actual del empleado.
* Celular: numero del celular del empleado.
* Id\_sucursal: id de la sucursal que se relaciona con el empleado.
* Correo\_electronico: correo personal del empleado.

1. **Clientes:**

* Id\_cliente: id único de clientes.
* cedula: número de cedula del cliente.
* Nombres: nombres del cliente.
* apellidos: apellidos del cliente.
* direccion: dirección de residencia del cliente.
* Correo\_electronico: correo personal del cliente.
* celular: numero de celular del cliente.
* Ciudad\_residencia: ciudad donde reside el cliente.

1. **Vehiculos**

* Id\_vehiculos: id único de vehiculos.
* sunroof: sunroof del vehiculo.
* capacidad: capacidad máxima de personas.
* tipo: tipo de vehiculo.
* placa: numero de placa del vehiculo.
* Referencia: referencia del vehiculo.
* Modelo: nombre del modelo del vehiculo.
* Puertas: cantidad de puertas que dispone el vehiculo.
* Color: color del vehiculo.
* Motor: nombre del motor que posee el vehiculo.

1. **Alquileres**

* Id\_alquiler: id único de alquiler.
* Fecha\_llegada: fecha en que el cliente devolvió el vehiculo.
* Id\_cliente: id del cliente que se relaciona con el alquiler.
* Id\_empleado: id del empleado que se relaciona con el alquiler.
* Id\_vehiculo: id del vehiculo que se relaciona con el alquiler.
* Id\_sucursal: id de la sucursal que se relaciona con el alquiler.
* Fecha\_salida: fecha en que salió el cliente con el vehiculo.
* Fecha\_esperada\_llegada: fecha que se acordó con el cliente que entregaría el vehiculo.
* Valor\_alquiler\_dia: valor del alquiler por los días que pase el cliente con el vehiculo.
* Valor\_alquiler\_semana: valor alquiler por semanas que pase el cliente con el vehiculo.
* Valor\_cotizado: valor que debe cancelar el cliente por el alquiler del vehiculo.
* Valor\_pagado: valor que cancelo el cliente por el alquiler del vehiculo.
* Porcentaje\_descuento: porcentaje de descuento que se le aplica al alquiler.

1. **Descuento**

* Id\_descuento: id único de alquiler.
* Id\_vehiculo: id del vehículo que se relaciona con el Descuento.
* Fecha\_inicio: fecha en que se activa el descuento.
* Fecha\_fin: fecha en que se cancela el descuento.

**Relaciones y Cardinalidades**

Se realizó las relaciones y cardinalidades respectivas del modelo conceptual con sus entidades

para tener mejor visualización de la base de datos:

1. **Sucursales- Empleados:**

* Relación: “Tiene”, una sucursal puede tener muchos empleados y un empleado puede estar en una sucursal.
* Cardinalidad: 1: N (uno a muchos).

1. **Sucursales - Alquileres:**

* Relación: “Tiene”, una sucursal tiene muchos alquileres y un alquiler está en una sucursal.
* Cardinalidad: N:1 (muchos a uno).

1. **Alquileres - Vehiculos:**

* Relación: “Tiene”, muchos vehiculos pueden estar en un alquiler y un alquiler puede tener muchos vehiculos.
* Cardinalidad: 1: N (uno a muchos).

1. **Empleados- Alquileres:**

* Relación: “gestiona”, un empleado puede gestionar muchos alquileres y un alquiler puede estar gestionado por un empleado.
* Cardinalidad: 1: N (uno a muchos).

1. **Clientes - Alquileres:**

* Relación: “Realiza”, un cliente puede realizar muchos alquileres y muchos alquileres pueden pertenecer a un cliente.
* Cardinalidad: 1: N (uno a muchos).

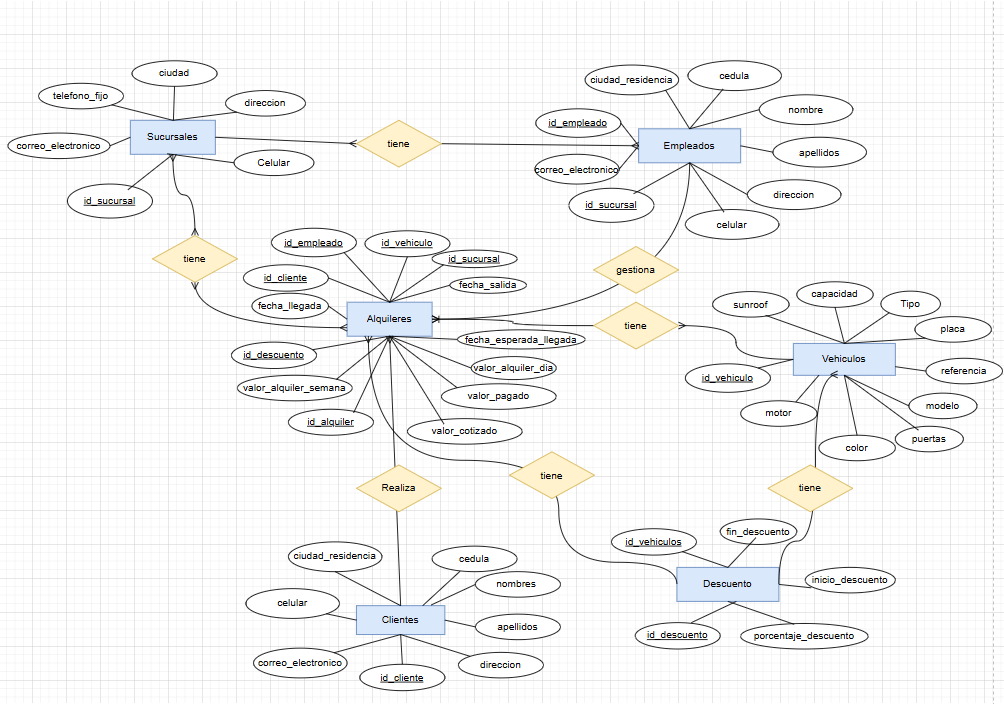
1. **Descuento – Vehiculo:**

* Relación: “Tiene” un tipo de vehículo tiene un descuento y un descuento tiene muchos vehículos.
* Cardinalidad: 1: N (uno a muchos).

1. **Alquileres – Descuento:**

* Relación: “Tiene” muchos alquileres tienen un descuento y un descuento tiene muchos alquileres

### Gráfica



## Construcción del Modelo Lógico

En este modelo se realiza la conversión de las entidades a tablas las cuales no pueden tener una relación de muchos a muchos por ende se realizan subtablas u añadieron nuevas entidades, además se incorporaron detalles más específicos como las características de cada atributo, incluidas las claves primarias, foráneas y las relaciones de cardinalidad.

### Descripción

Este paso tenemos las tablas de las entidades de la base de datos que está formada por cinco entidades y sus atributos las cuales son:

1. **Sucursales:**

* id\_sucursal int not null auto\_increment primary key,
* celular varchar(50),
* direccion varchar(50),
* ciudad varchar(50),
* telefono\_fijo varchar(50),
* correo\_electronico varchar(50)

1. **Empleados:**

* id\_empleado int not null auto\_increment primary key,
* ciudad\_residencia varchar(85),
* cedula varchar(50),
* nombre varchar(50),
* apellidos varchar(50),
* dirección varchar(50),
* celular varchar(50),
* id\_sucursal int,
* foreign key (id\_sucursal) references sucursales(id\_sucursal),
* correo\_electronico varchar(50)

1. **Clientes:**

* id\_cliente int not null auto\_increment primary key,
* cedula varchar(50),
* nombres varchar(50),
* apellidos varchar(50),
* direccion varchar(50),
* correo\_electronico varchar(50),
* celular varchar(50),
* ciudad\_residencia varchar(50)

1. **Vehiculos**

* id\_vehiculo int not null auto\_increment primary key,
* sunroof varchar(50),
* capacidad varchar(50),
* tipo varchar(50),
* placa varchar(50),
* referencia varchar(50),
* modelo varchar(50),
* puertas int,
* color varchar(50),
* motor varchar(50)

1. **Alquileres**

* id\_alquiler int not null auto\_increment primary key,
* fecha\_llegada date,
* id\_cliente int,
* foreign key(id\_cliente) references clientes(id\_cliente),
* id\_empleado int,
* foreign key(id\_empleado) references empleados(id\_empleado),
* id\_vehiculo int,
* foreign key(id\_vehiculo) references vehiculos(id\_vehiculo),
* id\_sucursal int,
* foreign key(id\_sucursal) references sucursales(id\_sucursal),
* fecha\_salida date,
* fecha\_esperada\_llegada date,
* valor\_alquiler\_dia decimal(10,2),
* valor\_alquiler\_semana decimal(10,2),
* valor\_cotizado decimal(10,2),
* valor\_pagado decimal(10,2),
* porcentaje\_descuento decimal(10,2)

1. **Descuento**

* Id\_descuento int not null auto\_increment primary key,
* Id\_vehiculo int,
* Foreign key (id\_vehiculo) references vehiculos(id\_vehiculo),
* fin\_descuento date,
* inicio\_descuento date,
* porcentaje\_descuento decimal (10,2)

**Relaciones y Cardinalidades**

Se realizó las relaciones y cardinalidades respectivas del modelo lógico con sus entidades para tener mejor visualización de la base de datos:

1. **Sucursales- Empleados:**

* una sucursal puede tener muchos empleados y un empleado puede estar en una sucursal. Cardinalidad: 1: N (uno a muchos).



1. **Sucursales - Alquileres:**

* una sucursal tiene muchos alquileres y un alquiler tiene una sucursal. Cardinalidad: N:1 (muchos a uno).



1. **Alquileres - Vehiculos:**

* muchos vehículos pueden estar en un alquiler y un alquiler puede tener muchos vehiculos. Cardinalidad: 1: N (uno a muchos).



1. **Empleados- Alquileres:**

* un empleado puede gestionar muchos alquileres y un alquiler puede estar gestionado por un empleado. Cardinalidad: 1: N (uno a muchos).



1. **Clientes - Alquileres:**

* Un cliente puede realizar muchos alquileres y muchos alquileres pueden pertenecer a un cliente. Cardinalidad: 1: N (uno a muchos).



1. **Vehiculos - Descuento:**

* Muchos vehículos pueden estar en un descuento y un descuento puede tener muchos vehículos. Cardinalidad: 1: N (uno a muchos).

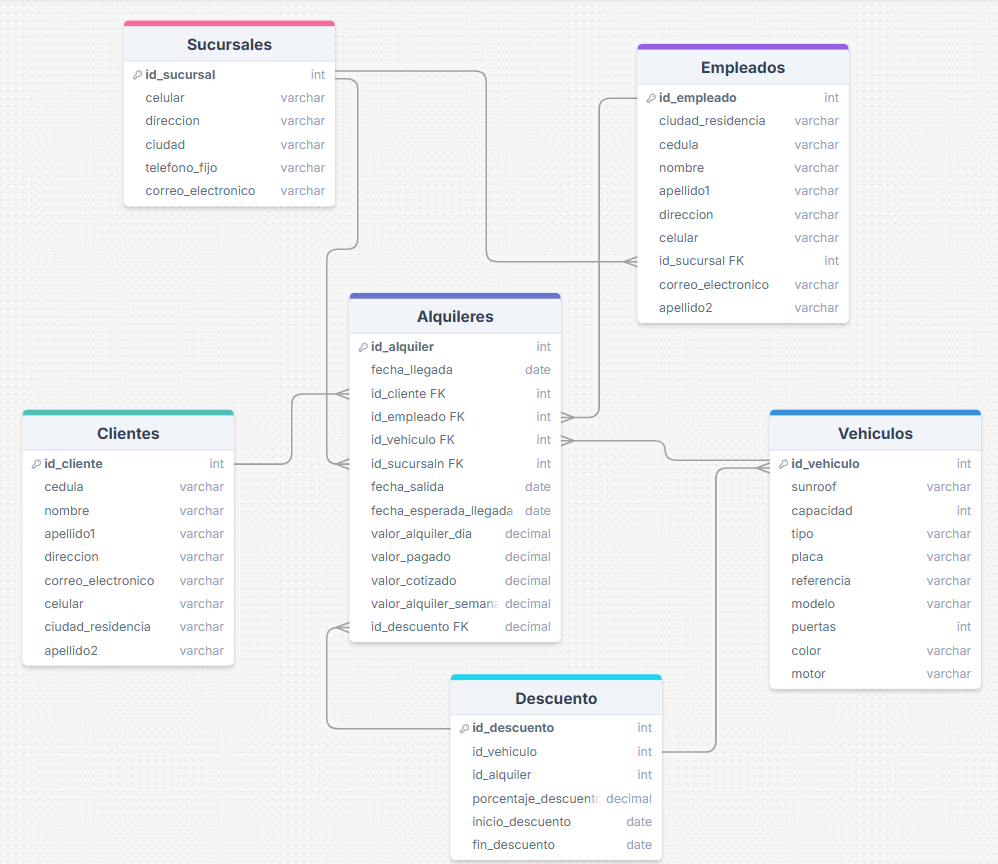


1. **Alquileres - Descuento:**

* Muchos alquileres pueden tener un descuento y un descuento puede estar en muchos alquileres.



Gráfica



## Normalización del Modelo Lógico

Se realizó el proceso de la normalización de las tablas anteriormente visualizadas para organizar sus datos de la manera más eficiente. En esta parte se organizaron los datos para reducir su redundancia y dependencias transitivas en la base de datos.

### Primera Forma Normal (1FN)

En esta forma se necesita que los atributos se encuentren atómicos en la mínima expresión que no haya más de dos datos en cada columna.

Una tabla está en 1FN si cumple con los siguiente reglas o criterios:

* Todos los atributos contienen valores atómicos (indivisibles).
* No debe haber grupos repetidos de columnas.
* Cada columna debe contener un solo valor en cada fila.

#### Descripción

Esta forma normal es el primer nivel de normalización en el diseño de la base de datos el cual se aplica a las tablas de la base de datos para organizar los datos de manera que evite redundancia y para asegurar la consistencia de la información de cada tabla.

#### Gráfica









#### Descripción Técnica

1. **Sucursales:**

* Se encuentra en 1FN, ya que cuenta con una clave primaria única y cada columna tiene valores únicos y no son repetitivos.

1. **Empleados:**

* Se encuentra en 1FN, ya que cuenta con una clave primaria única y cada columna tiene valores únicos y no son repetitivos.

1. **Clientes:**

* Se encuentra en 1FN, ya que cuenta con una clave primaria única y cada columna tiene valores únicos y no son repetitivos.

1. **Alquileres:**

* Se encuentra en 1FN, ya que cuenta con una clave primaria única y cada columna tiene valores únicos y no son repetitivos.

1. **Vehiculos:**

* Se encuentra en 1FN, ya que cuenta con una clave primaria única y cada columna tiene valores únicos y no son repetitivos.

1. **Descuento:**

* Se encuentra en 1FN, ya que cuenta con una clave primaria única y cada columna tiene valores únicos y no son repetitivos.

### Segunda Forma Normal (2FN)

Una tabla está en 2FN si cumple con las siguientes reglas o criterios:

* Está en 1FN.
* Todos los atributos no claves (no pertenecientes a una clave primaria compuesta dependen de la clave primaria u de la clave de la tabla principal.)

#### Descripción

La segunda forma normal, es el segundo nivel de normalización en el diseño de la base de datos que se aplicara a las tablas de la base de datos que ya cumplen con la primera forma normal y lleva a cabo la eliminación de dependencias parciales dentro de una table.

#### Gráfica









#### Descripción Técnica

1. **Sucursales:**

* Se encuentra en 2FN, ya que cuenta con una clave primaria única y cada columna depende completamente de esa clave primaria.

1. **Empleados:**

* Se encuentra en 2FN, ya que cuenta con una clave primaria única y cada columna depende completamente de esa clave primaria.

1. **Clientes:**

* Se encuentra en 2FN, ya que cuenta con una clave primaria única y cada columna depende completamente de esa clave primaria.

1. **Alquileres:**

* Se encuentra en 2FN, ya que cuenta con una clave primaria única y cada columna depende completamente de esa clave primaria.

1. **Vehiculos:**

* Se encuentra en 2FN, ya que cuenta con una clave primaria única y cada columna depende completamente de esa clave primaria.

1. **Descuento:**

* Se encuentra en 2FN, ya que cuenta con una clave primaria única y cada columna depende completamente de esa clave primaria.

### Tercera Forma Normal (3FN)

Una tabla está en 3NF si cumple con los siguientes criterios:

* Está en 2NF.
* No hay dependencias transitivas: ningún atributo no clave depende de otro atributo no clave.

En esta forma se verifica que cada tabla creada en la forma Normal 2FN sea independiente de la tabla principal ya que no deben existir atributos no principales que dependan transitivamente del atributo clave.

#### Descripción

La tercera forma normal, es el tercer nivel de normalización en el diseño de la base de datos que se aplicará a las tablas de una base de datos que ya cumplen con la segunda forma normal

En este caso las tablas tienen un id único o llave primaria para que no tenga una dependencia con la tabla principal como se muestra en las siguientes tablas:

#### Gráfica







#### 

#### Descripción Técnica

1. **Sucursales:**

* Se encuentra en 3FN, ya que esta en la 2FN y en cada columna no hay dependencias transitivas con la clave primaria.

1. **Empleados:**

* Se encuentra en 3FN, ya que está en la 2FN y en cada columna no hay dependencias transitivas con la clave primaria.

1. **Clientes:**

* Se encuentra en 3FN, ya que está en la 2FN y en cada columna no hay dependencias transitivas con la clave primaria.

1. **Alquileres:**

* Se encuentra en 3FN, ya que está en la 2FN y en cada columna no hay dependencias transitivas con la clave primaria.

1. **Vehiculos:**

* Se encuentra en 3FN, ya que está en la 2FN y en cada columna no hay dependencias transitivas con la clave primaria.

1. **Descuento:**

* Se encuentra en 3FN, ya que está en la 2FN y en cada columna no hay dependencias transitivas con la clave primaria.

## Construcción del Modelo Físico

Este modelo es la última etapa de la creación de diseño de la base de datos en el cual se tiene en consideración el modelo lógico, donde se verificaron las restricciones y se especificaron las estructuras de las tablas. Además, este modelo incorpora los tipos de datos de los atributos previamente definidos, los cuales fueron estructurados en tablas utilizando el lenguaje de un sistema de gestión de bases de datos (SGBD) compatible con la plataforma MySQL.

### Descripción

El modelo físico se diseñó para funcionar en MySQL, donde cada entidad se representa como una tabla compuesta por sus atributos correspondientes como también que tipo de dato contiene cada atributo ya sea INT, VARCHAR, DATE, entre otros.

### Código

Para crear la base de datos utilice el siguiente comando:

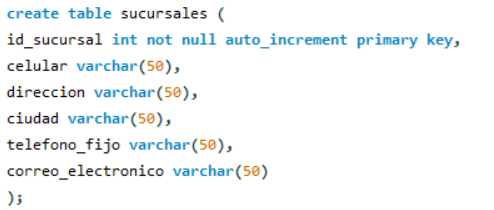


Para utilizar la base de datos ocupe el siguiente comando:

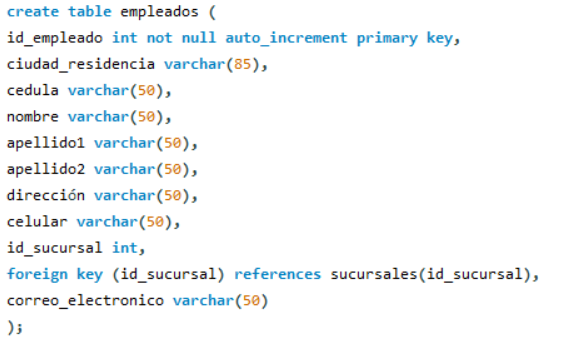


Comenzamos creando las tablas junto con sus tipos de datos correspondientes. Para esto, se utilizan los siguientes comandos:

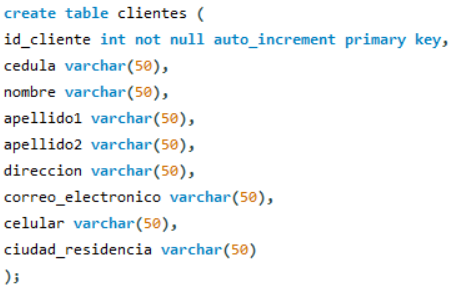
1. Creación de la tabla Sucursales



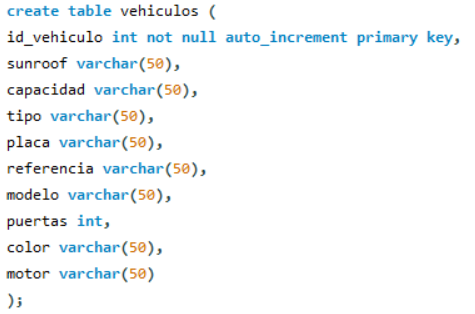
1. Creación de la tabla Empleados



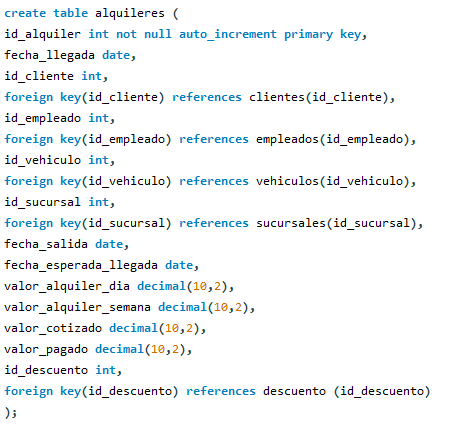
1. Creación de la tabla Clientes



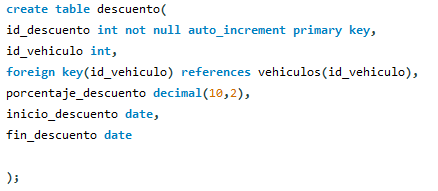
1. Creación de la tabla Vehiculos



1. Creación de la tabla Alquileres



1. Creación de la tabla Descuento



Finalmente, para listar las tablas anteriormente creadas utilizamos este comando:



**Construcción del Diagrama UML**

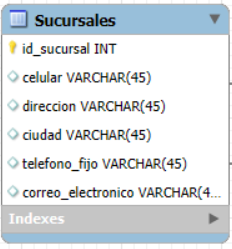
Se ha diseñado un diagrama UML tomando como referencia la normalización para entender mejor los diseños, la arquitectura del código y la implementación propuesta. Este enfoque nos permitirá tener una visión clara y detallada de cómo se manejarán cada una de las consultas, funcionalidades y los usuarios en la base de datos. De esta manera, podremos asegurar una implementación eficiente y coherente con los requisitos del sistema.

**Descripción**

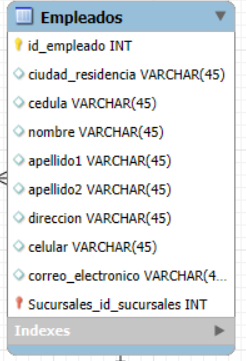
El diagrama UML se ha diseñado con el objetivo de representar detalladamente la estructura de cada tabla y sus relaciones. Este diagrama da a conocer claramente el tipo de dato correspondiente a cada atributo, así como la identificación de claves primarias (primary Keys) y claves foráneas (foreign Keys). Además, se especifica la obligatoriedad de los atributos, proporcionando una visión precisa y exhaustiva de la base de datos. Esto facilita el entendimiento y la implementación técnica, con esto se asegura que todos los componentes y sus interrelaciones estén correctamente definidos.

Comenzaremos creando las tablas junto con sus tipos de datos correspondientes:

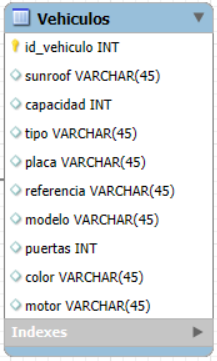
1. Tabla Sucursales



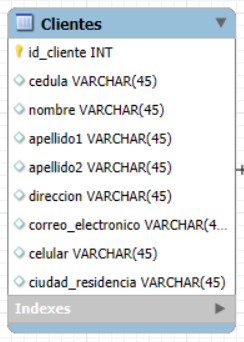
1. Tabla Empleados



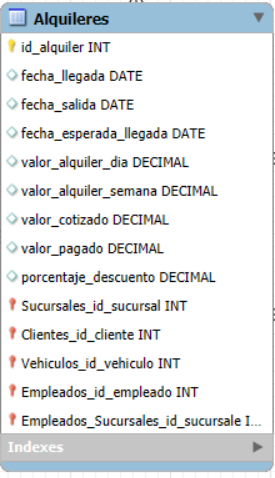
1. Tabla Vehiculos



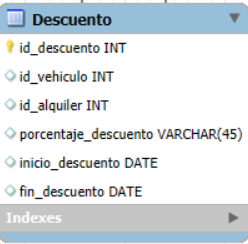
1. Tabla Clientes



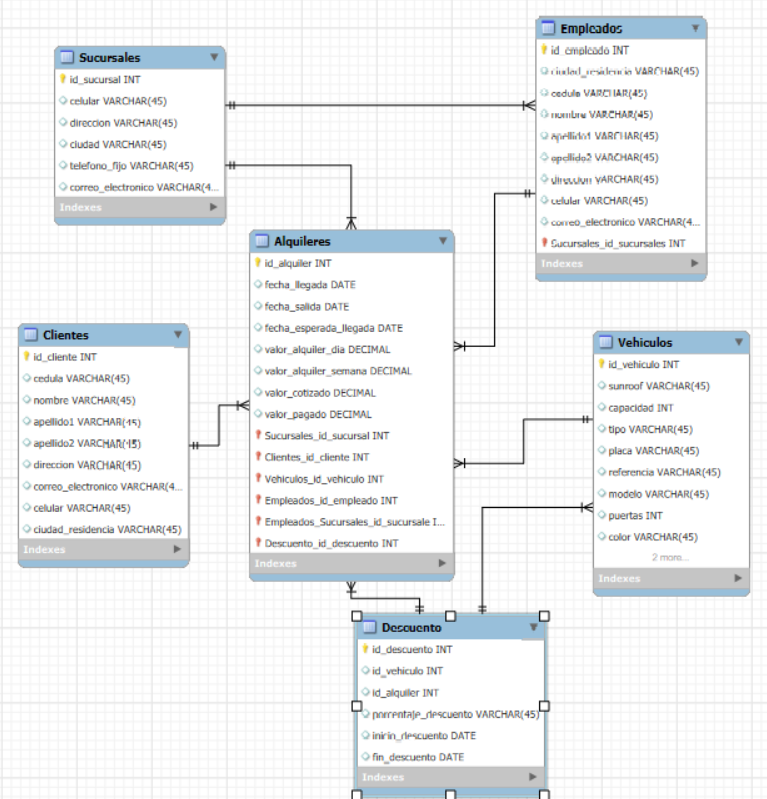
1. Tabla Alquileres



1. Tabla descuento



###### Gráfica



Inserciones de Datos

La inserción de datos en las tablas es una parte importante para poder realizar la prueba de operatividad.

Para insertar datos en una tabla en especifica se utiliza la siguiente sintaxis:



* **nombre\_de\_tabla**: El nombre de la tabla donde se insertarán los datos.
* **columna1, columna2, columna3**: Son los atributos de las tablas en los cuales se ingresarán los datos específicamente.
* **VALUES (valor1, valor2, valor3):** Se ingresan los valores que se desea en cada columna específica, los valores están en el mismo orden que las columnas.

**Descripción**

Para especificar el proceso de inserción de datos en las tablas de la base de datos ‘AutoRental, realizáremos la inserción de datos de cada tabla.

Para ingresar datos en la tabla ‘Sucursales’ la cual muestra el celular, dirección, ciudad, teléfono\_fijo, correo\_electronico, se emplea el siguiente comando:



Para ingresar datos en la tabla empleados la cual muestra la ciudad\_residencia, cedula del empleado, nombre, apellido1, apellido2, dirección, celular, id\_sucursal, correo electrónico del empleado, se emplea el siguiente comando:

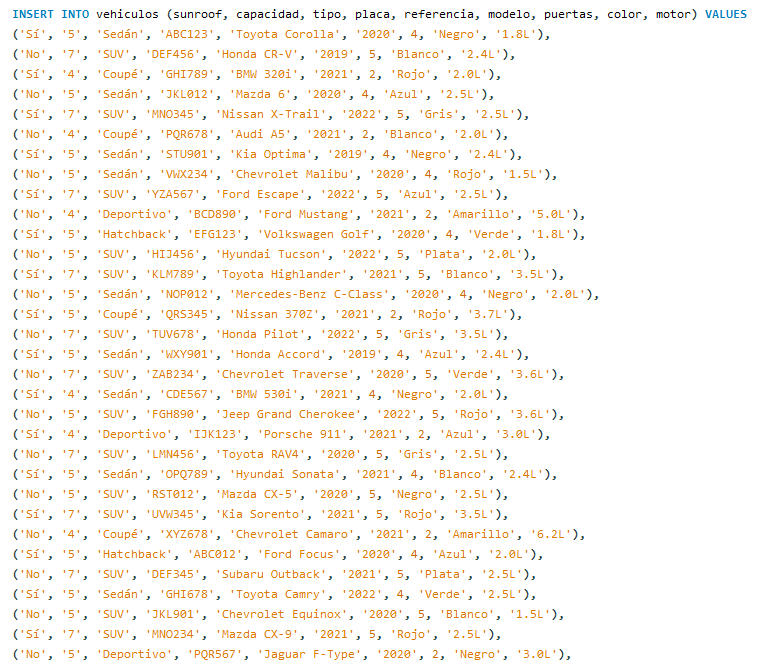


##### 

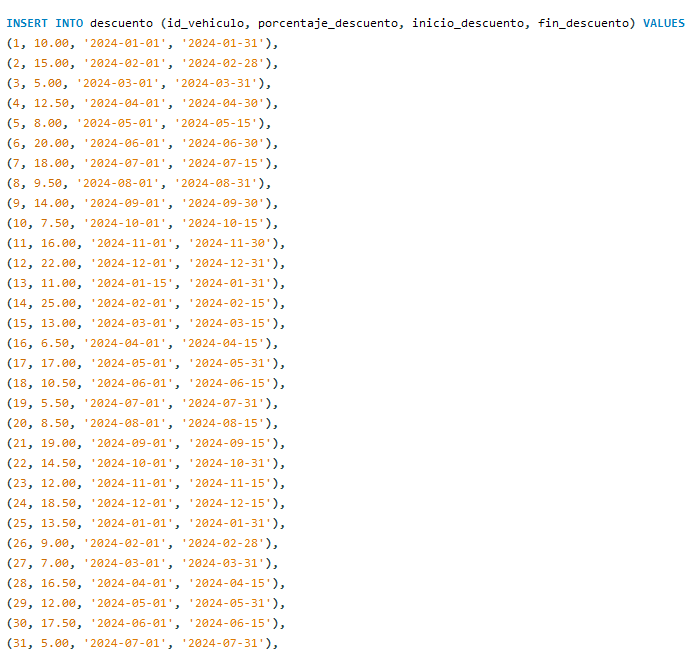
Para ingresar datos en la tabla clientes la cual muestra la cedula, nombre, apellido1, apellido2, dirección, correo\_electronico, celular, ciudad\_residencia, se emplea el siguiente comando:



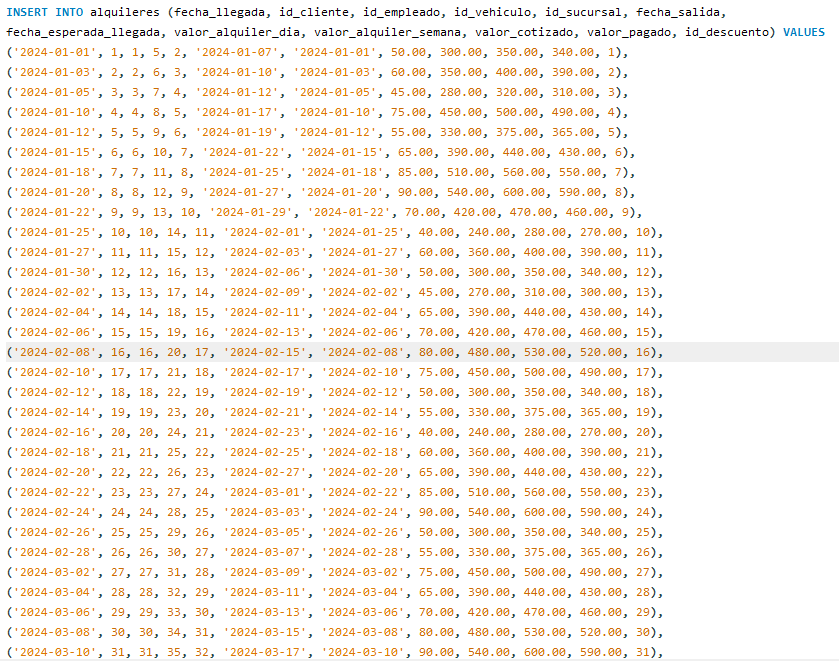
Para ingresar datos en la tabla ‘Vehiculos’ la cual muestra sunroof, capacidad, tipo, placa, referencia, modelo, puertas, color, motor, se emplea el siguiente comando:



Para ingresar datos en la tabla ‘Descuento’ la cual muestra el id\_vehiculo, porcentaje\_descuento, inicio\_descuento, fin\_descuento, se emplea el siguiente comando:



Para ingresar datos en la tabla de ‘Alquileres’ la cual muestra la fecha\_llegada, id\_cliente, id\_empleado, id\_vehiculo, id\_sucursal, fecha\_salida, fecha\_esperada\_llegada, valor\_alquiler\_dia, valor\_alquiler\_semana, valor\_cotizado, valor\_pagado, id\_descuento total a pagar, fecha de transacción y el id del pedido, se emplea el siguiente comando:



**Consultas de Datos**

Las consultas en una base de datos son indispensables, ya que facilitan el acceso y la recuperación de información almacenada. Además, permiten mantener la base de datos actualizada mediante la inserción, modificación y actualización de datos. Son fundamentales para almacenar, manipular y recuperar datos de manera eficiente y segura. Para realizar consultas básicas, se utiliza la siguiente sintaxis:

En esta consulta se muestran todos los datos ‘SELECT \*’ de una tabla en específico ‘FROM

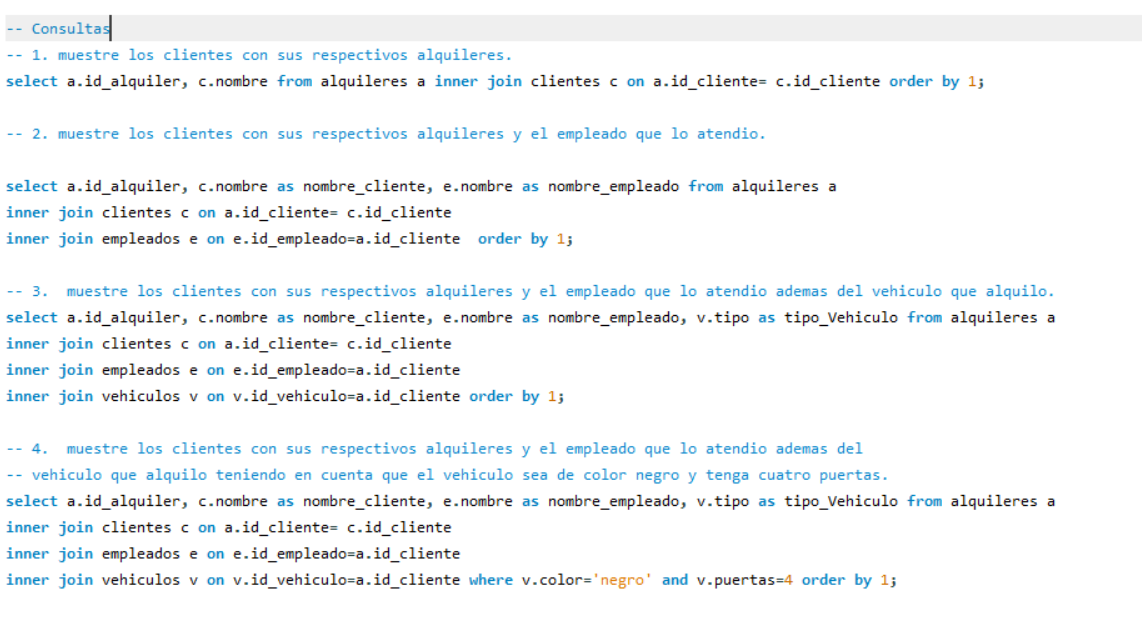
nombre\_tabla’:

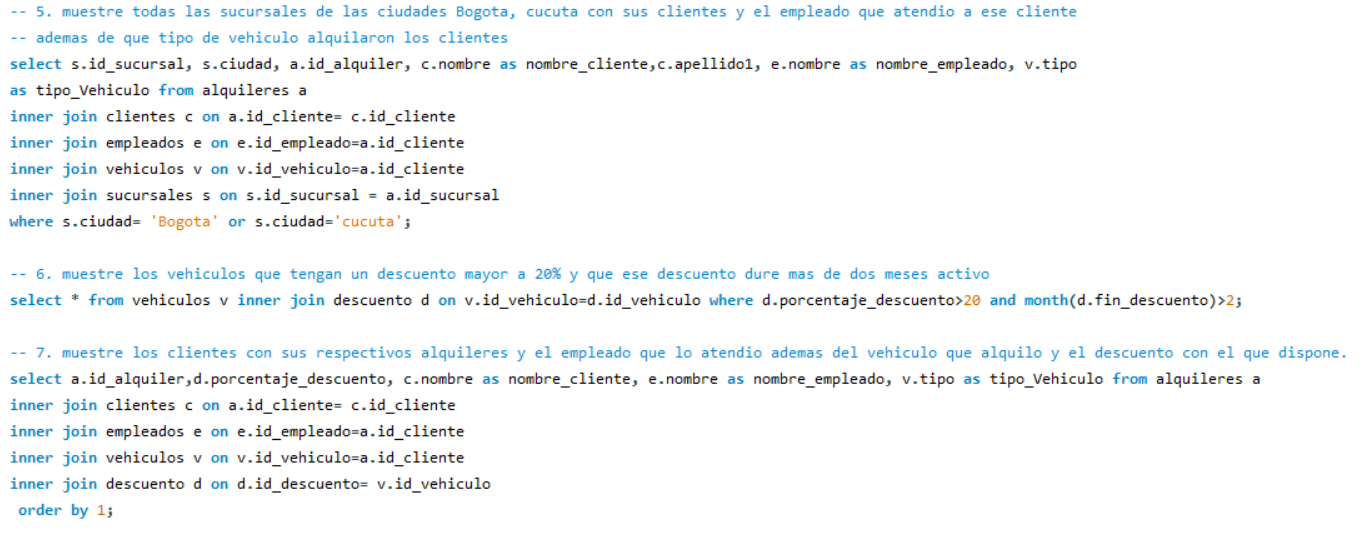


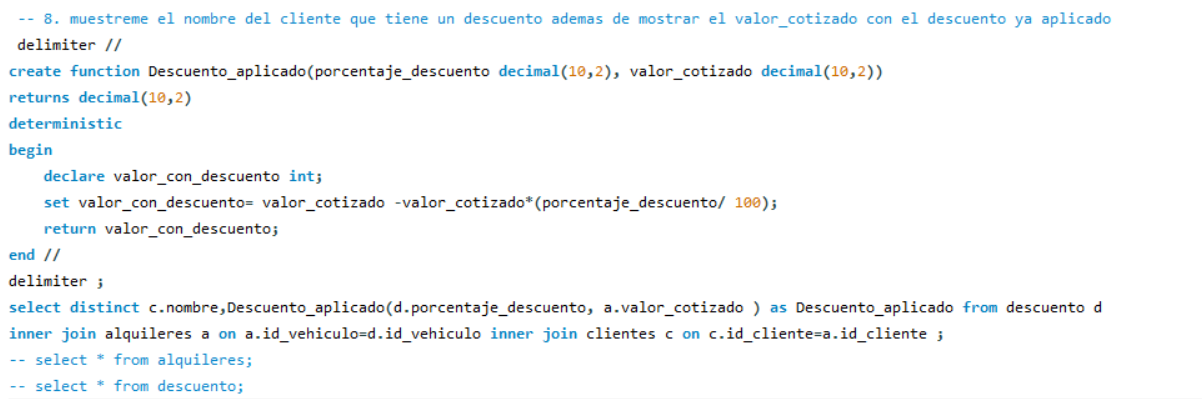
**Descripción**

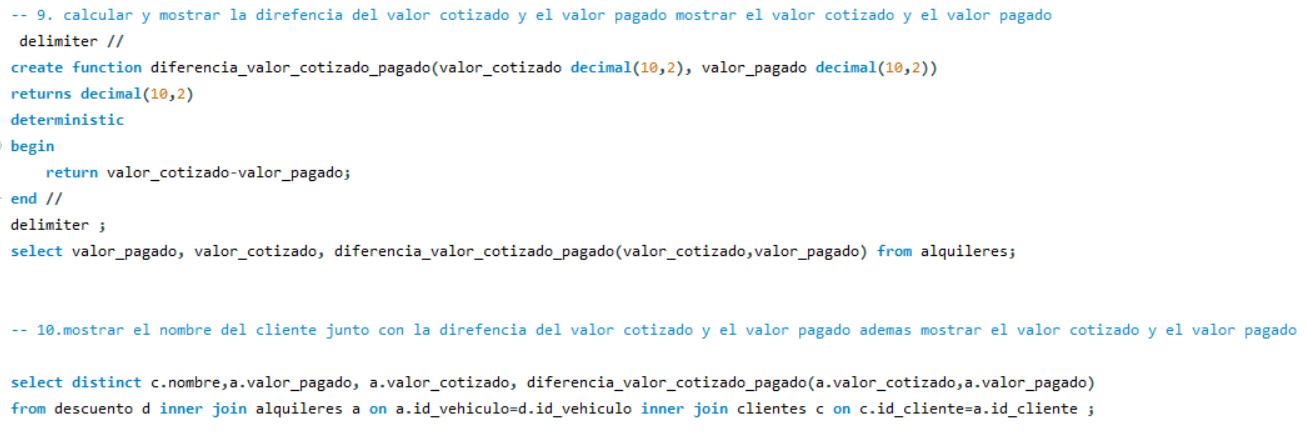
Para aclarar el proceso de consultas de datos en las tablas de la base de datos ‘AUTORENTAL’, veremos algunos ejemplos prácticos utilizando información que contiene cada tabla.

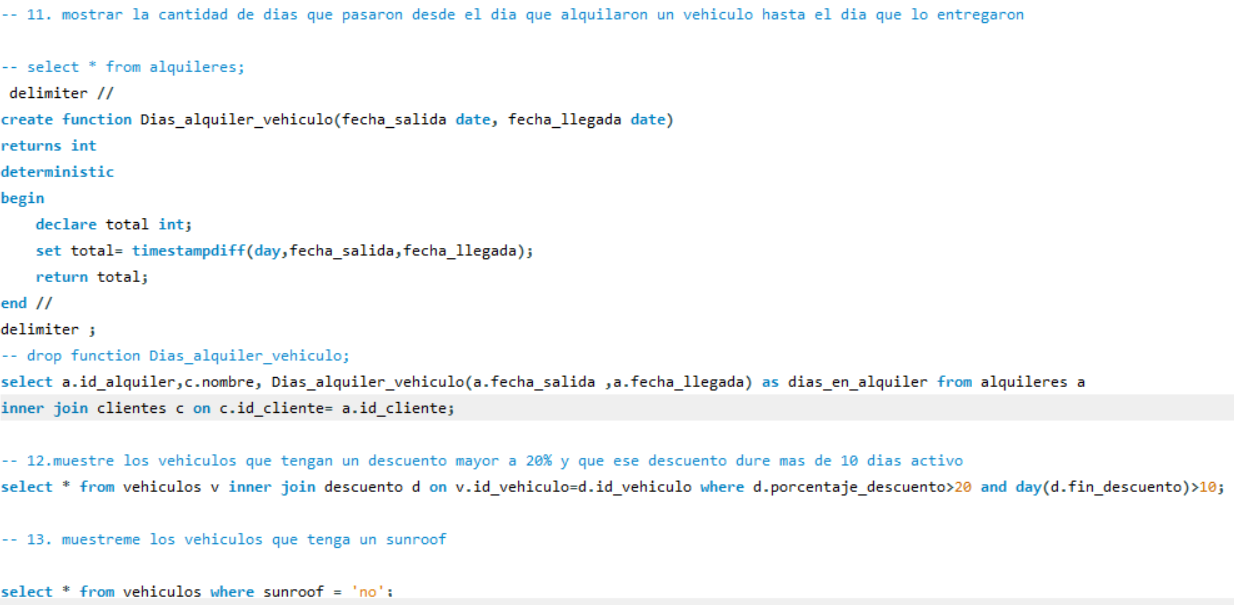
A continuación, tenemos unas consultas de la base de datos AUTORENTAL

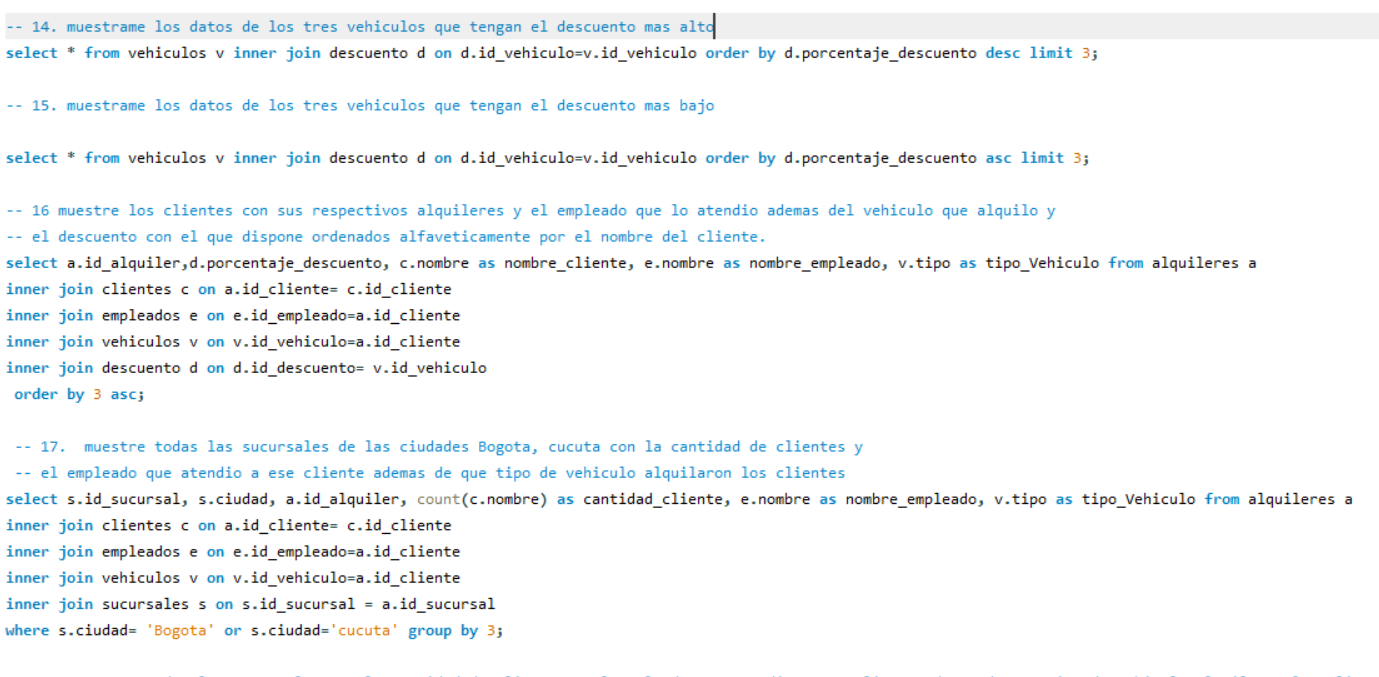


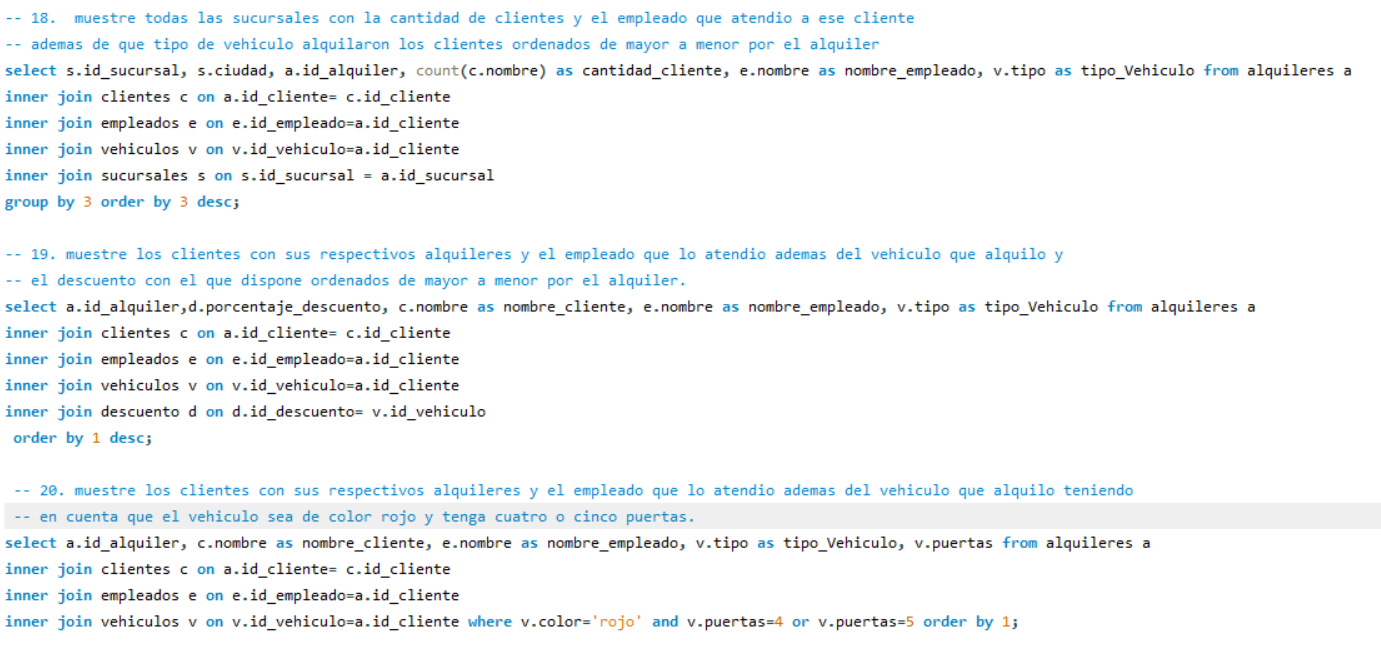


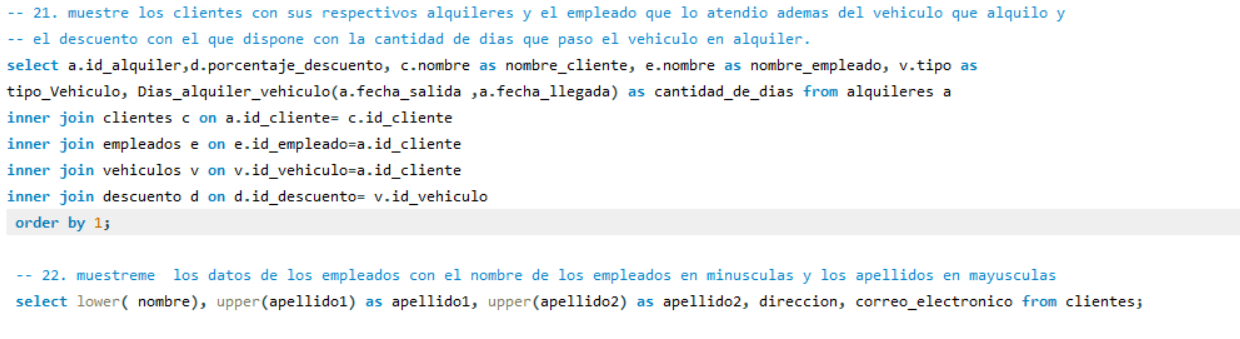


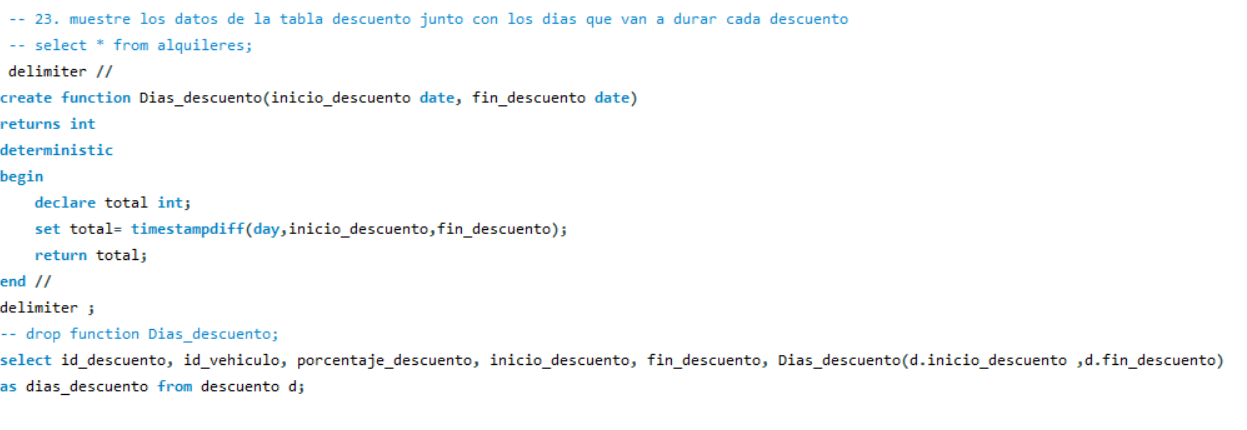


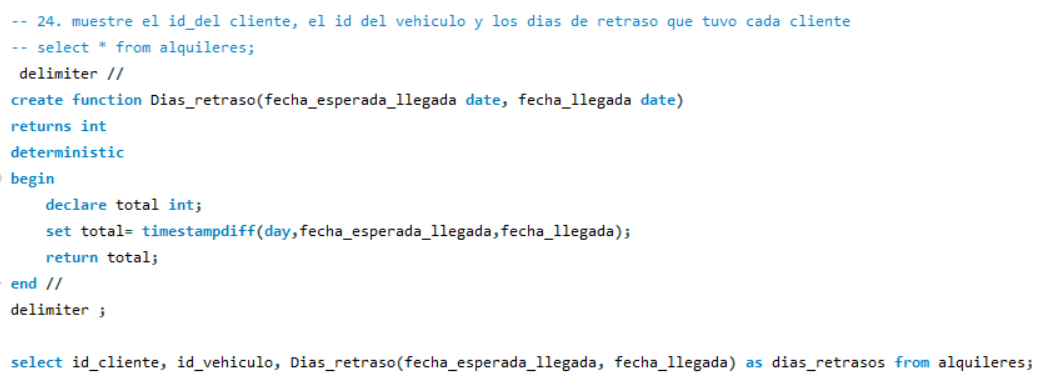


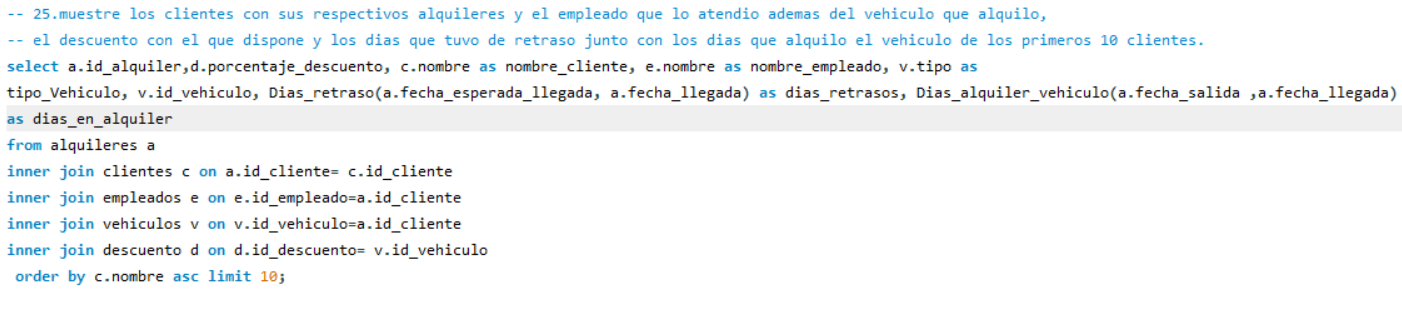




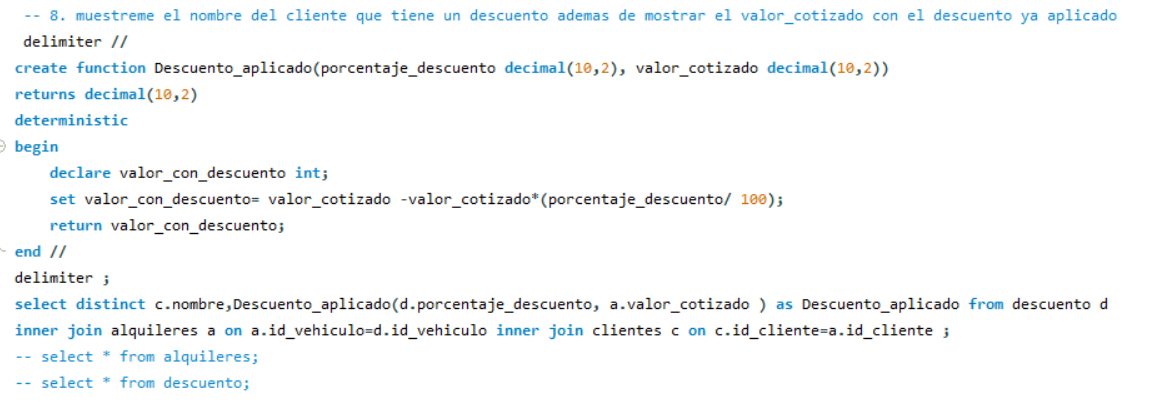


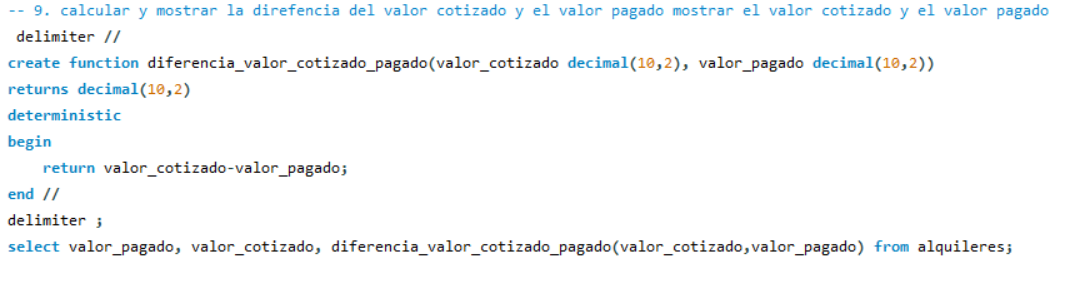


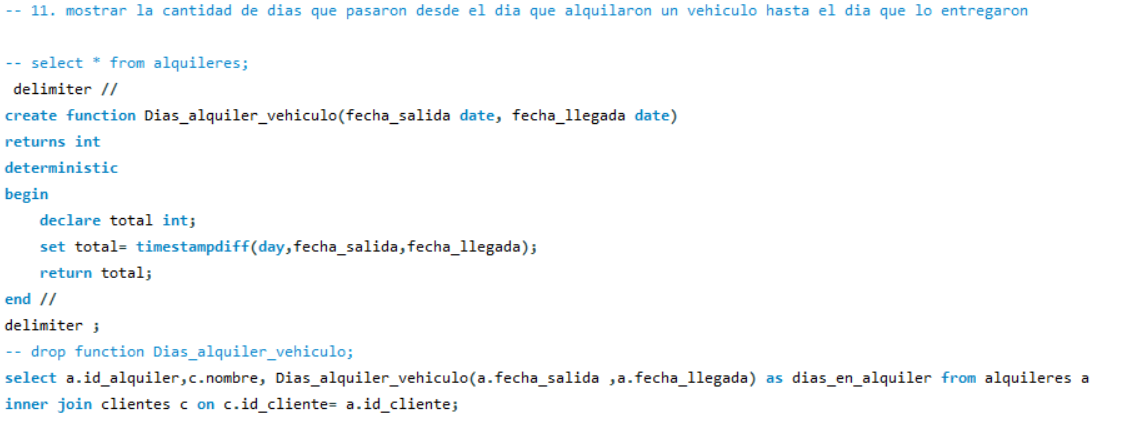


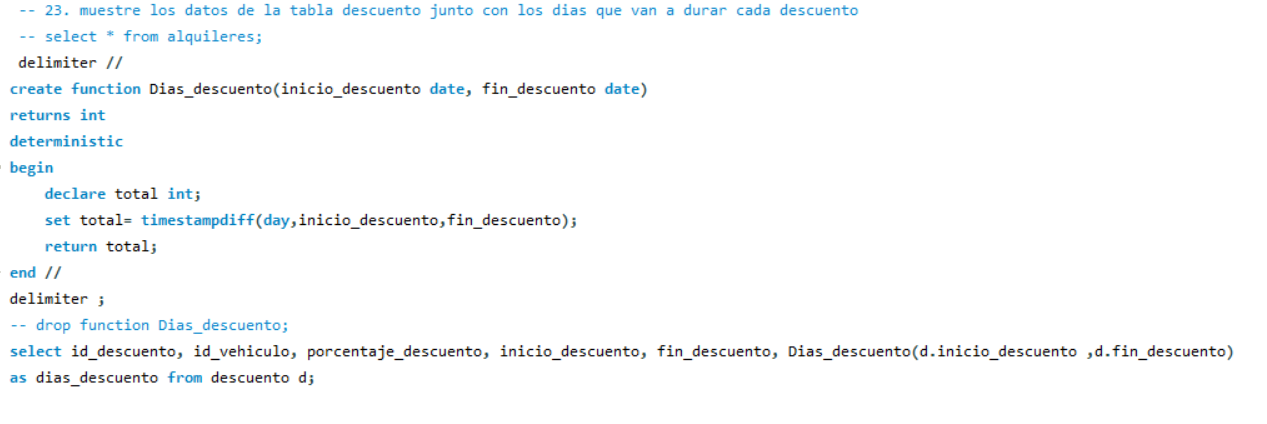


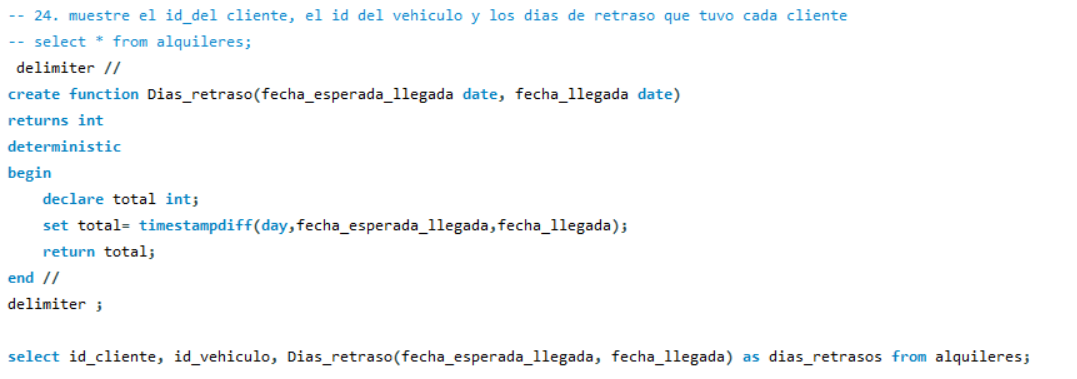
A continuación, se presentan unas funciones las cuales son importantes ya que realizan operaciones que nos ayudan a gestionar mejor los datos:









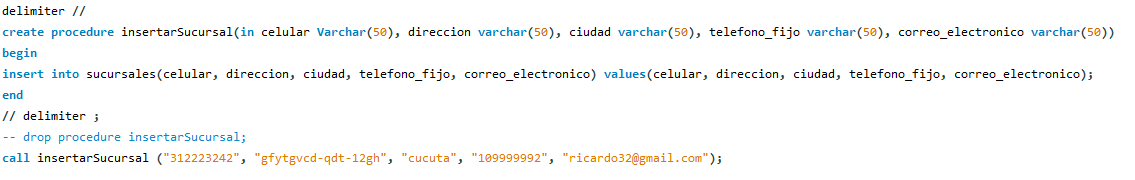


**Procedimientos**

A continuación, tenemos unos procedimientos los cuales permiten realizar unos procesos que facilitan el manejo de los datos de la base de datos.

1. Se realizó un procedimiento para poder añadir una nueva sucursal a la base de datos.

-- 1. Este procedimiento se diseñó para añadir una nueva sucursal



Código:

delimiter //

create procedure insertarSucursal(in celular Varchar(50), direccion varchar(50), ciudad varchar(50), telefono\_fijo varchar(50), correo\_electronico varchar(50))

begin

insert into sucursales(celular, direccion, ciudad, telefono\_fijo, correo\_electronico) values(celular, direccion, ciudad, telefono\_fijo, correo\_electronico);

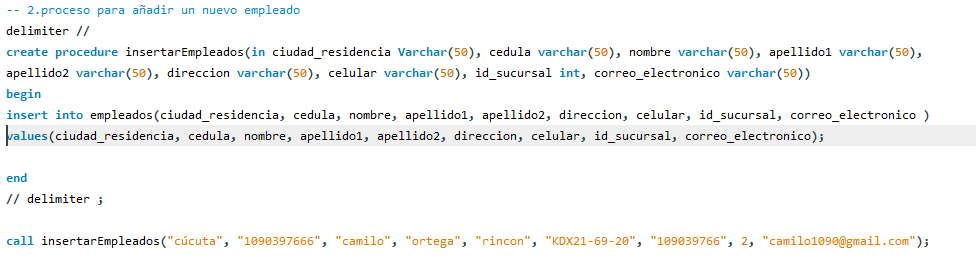
end

// delimiter ;

-- drop procedure insertarSucursal;

call insertarSucursal ("312223242", "gfytgvcd-qdt-12gh", "cucuta", "109999992", "ricardo32@gmail.com");

1. Este procedimiento permite añadir un nuevo empleado asignándole los valores requeridos en la parte de **call insertarEmpleados()**

****

-- 2.proceso para añadir un nuevo empleado

delimiter //

create procedure insertarEmpleados(in ciudad\_residencia Varchar(50), cedula varchar(50), nombre varchar(50), apellido1 varchar(50),

apellido2 varchar(50), direccion varchar(50), celular varchar(50), id\_sucursal int, correo\_electronico varchar(50))

begin

insert into empleados(ciudad\_residencia, cedula, nombre, apellido1, apellido2, direccion, celular, id\_sucursal, correo\_electronico )

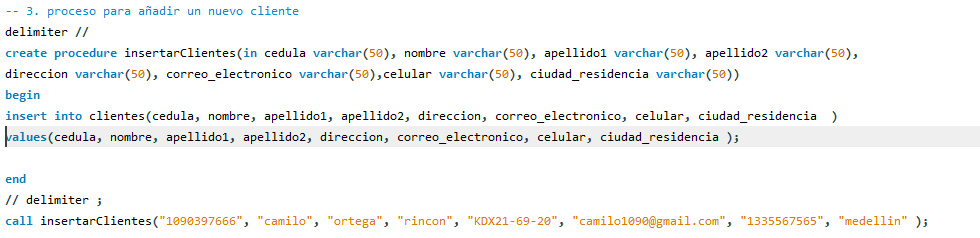
values(ciudad\_residencia, cedula, nombre, apellido1, apellido2, direccion, celular, id\_sucursal, correo\_electronico);

end

// delimiter ;

call insertarEmpleados("cúcuta", "1090397666", "camilo", "ortega", "rincon", "KDX21-69-20", "109039766", 2, "camilo1090@gmail.com");

1. Este procedimiento permite añadir un nuevo cliente asignándole los valores requeridos en la parte de call insertarClientes()



Código:

delimiter //

create procedure insertarClientes(in cedula varchar(50), nombre varchar(50), apellido1 varchar(50), apellido2 varchar(50),

direccion varchar(50), correo\_electronico varchar(50),celular varchar(50), ciudad\_residencia varchar(50))

begin

insert into clientes(cedula, nombre, apellido1, apellido2, direccion, correo\_electronico, celular, ciudad\_residencia )

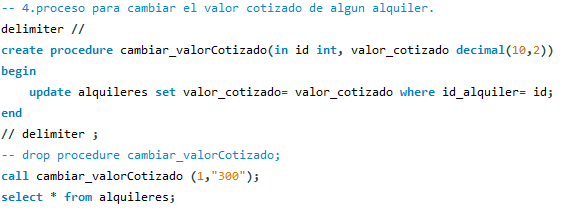
values(cedula, nombre, apellido1, apellido2, direccion, correo\_electronico, celular, ciudad\_residencia );

end

// delimiter ;

call insertarClientes("1090397666", "camilo", "ortega", "rincon", "KDX21-69-20", "camilo1090@gmail.com", "1335567565", "medellin" );

1. Este procedimiento permite editar o actualizar un nuevo valor de alquiler a un alquiler en específico que elija el usuario por medio del id.



Código:

delimiter //

create procedure cambiar\_valorCotizado(in id int, valor\_cotizado decimal(10,2))

begin

update alquileres set valor\_cotizado= valor\_cotizado where id\_alquiler= id;

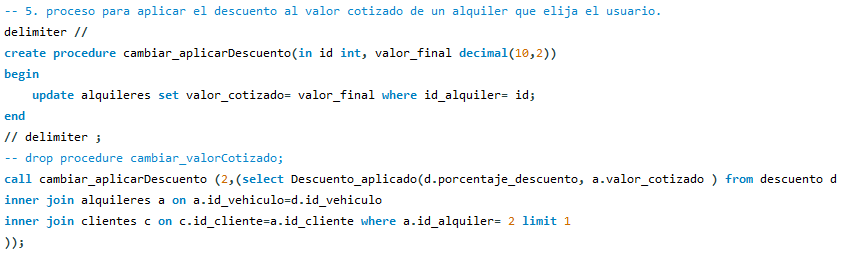
end

// delimiter ;

-- drop procedure cambiar\_valorCotizado;

call cambiar\_valorCotizado (1,"300");

1. Este procedimiento permite aplicar el descuento al valor cotizado de un alquiler que elija el usuario por medio del id del alquiler.



Código:

delimiter //

create procedure cambiar\_aplicarDescuento(in id int, valor\_final decimal(10,2))

begin

update alquileres set valor\_cotizado= valor\_final where id\_alquiler= id;

end

// delimiter ;

-- drop procedure cambiar\_valorCotizado;

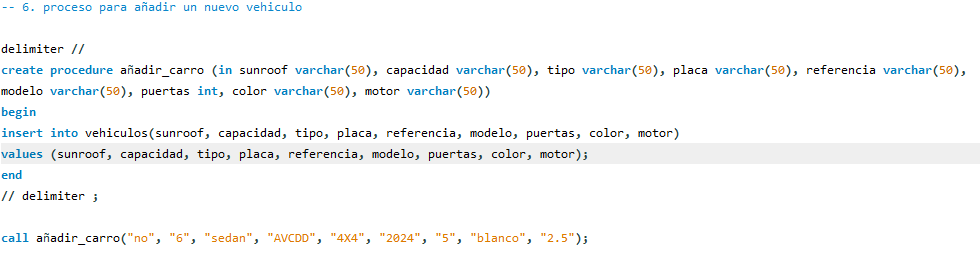
call cambiar\_aplicarDescuento (2,(select Descuento\_aplicado(d.porcentaje\_descuento, a.valor\_cotizado ) from descuento d

inner join alquileres a on a.id\_vehiculo=d.id\_vehiculo

inner join clientes c on c.id\_cliente=a.id\_cliente where a.id\_alquiler= 2 limit 1

));

1. Este procedimiento permite añadir un nuevo vehículo a la base de datos asignándole los valores requeridos en la parte de call añadir\_carro().



Código:

delimiter //

create procedure añadir\_carro (in sunroof varchar(50), capacidad varchar(50), tipo varchar(50), placa varchar(50), referencia varchar(50),

modelo varchar(50), puertas int, color varchar(50), motor varchar(50))

begin

insert into vehiculos(sunroof, capacidad, tipo, placa, referencia, modelo, puertas, color, motor)

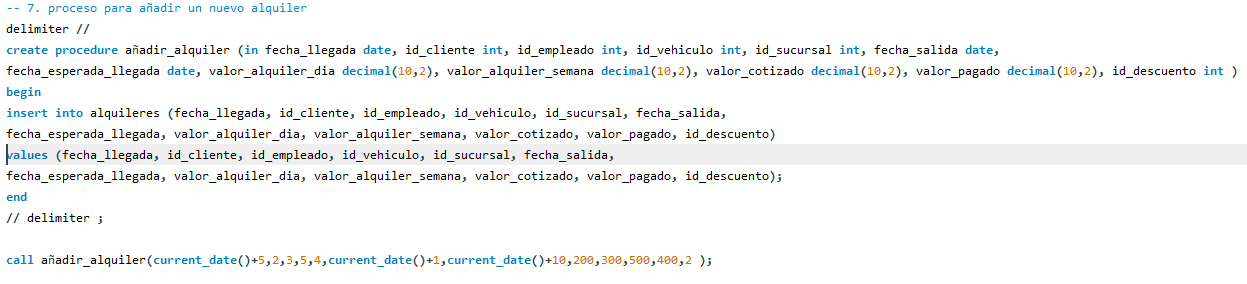
values (sunroof, capacidad, tipo, placa, referencia, modelo, puertas, color, motor);

end

// delimiter ;

call añadir\_carro("no", "6", "sedan", "AVCDD", "4X4", "2024", "5", "blanco", "2.5");

1. Este procedimiento permite añadir un nuevo alquiler a la base de datos asignándole los valores requeridos en la parte de call añadir\_alquiler(),



Código:

delimiter //

create procedure añadir\_alquiler (in fecha\_llegada date, id\_cliente int, id\_empleado int, id\_vehiculo int, id\_sucursal int, fecha\_salida date,

fecha\_esperada\_llegada date, valor\_alquiler\_dia decimal(10,2), valor\_alquiler\_semana decimal(10,2), valor\_cotizado decimal(10,2), valor\_pagado decimal(10,2), id\_descuento int )

begin

insert into alquileres (fecha\_llegada, id\_cliente, id\_empleado, id\_vehiculo, id\_sucursal, fecha\_salida,

fecha\_esperada\_llegada, valor\_alquiler\_dia, valor\_alquiler\_semana, valor\_cotizado, valor\_pagado, id\_descuento)

values (fecha\_llegada, id\_cliente, id\_empleado, id\_vehiculo, id\_sucursal, fecha\_salida,

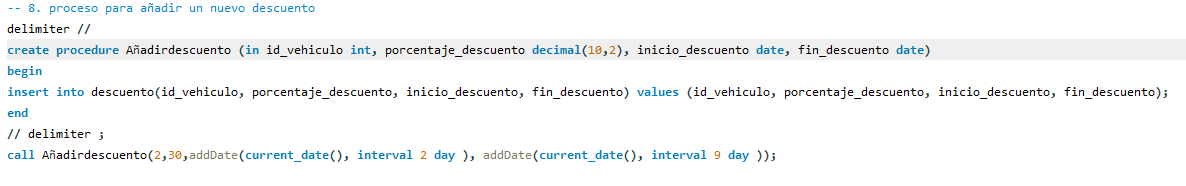
fecha\_esperada\_llegada, valor\_alquiler\_dia, valor\_alquiler\_semana, valor\_cotizado, valor\_pagado, id\_descuento);

end

// delimiter ;

call añadir\_alquiler(current\_date()+5,2,3,5,4,current\_date()+1,current\_date()+10,200,300,500,400,2 );

1. Procedimiento para añadir un nuevo descuento a la base de datos asignándole los valores requeridos en la parte de call Añadirdescuento().



Código:

delimiter //

create procedure Añadirdescuento (in id\_vehiculo int, porcentaje\_descuento decimal(10,2), inicio\_descuento date, fin\_descuento date)

begin

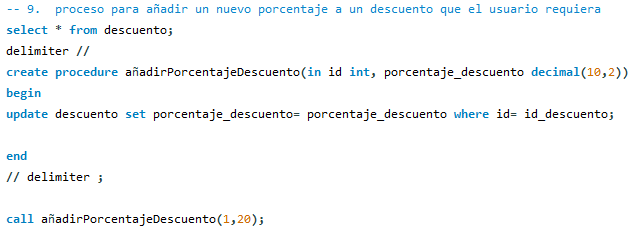
insert into descuento(id\_vehiculo, porcentaje\_descuento, inicio\_descuento, fin\_descuento) values (id\_vehiculo, porcentaje\_descuento, inicio\_descuento, fin\_descuento);

end

// delimiter ;

call Añadirdescuento(2,30,addDate(current\_date(), interval 2 day ), addDate(current\_date(), interval 9 day ));

1. Procedimiento para modificar un porcentaje de descuento en el descuento que el usuario elija por medio del id.



Código:

select \* from descuento;

delimiter //

create procedure añadirPorcentajeDescuento(in id int, porcentaje\_descuento decimal(10,2))

begin

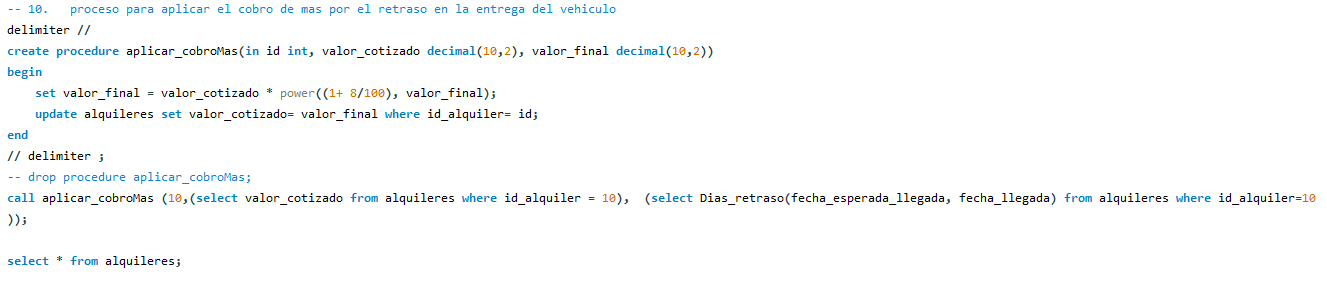
update descuento set porcentaje\_descuento= porcentaje\_descuento where id= id\_descuento;

end

// delimiter ;

call añadirPorcentajeDescuento(1,20);

1. Este procedimiento permite realizar u calcular el cobro de más, por los días de retraso que tuvo un cliente en específico y asignarle dicho valor.



Código:

delimiter //

create procedure aplicar\_cobroMas(in id int, valor\_cotizado decimal(10,2), valor\_final decimal(10,2))

begin

set valor\_final = valor\_cotizado \* power((1+ 8/100), valor\_final);

update alquileres set valor\_cotizado= valor\_final where id\_alquiler= id;

end

// delimiter ;

-- drop procedure aplicar\_cobroMas;

call aplicar\_cobroMas (10,(select valor\_cotizado from alquileres where id\_alquiler = 10), (select Dias\_retraso(fecha\_esperada\_llegada, fecha\_llegada) from alquileres where id\_alquiler=10

));