DISEÑO DE UNA BASE DE DATOS

LIBRERÍA EL MUNDO SOFÍA

**Valerie Michel Lasso Lizcano**

**T2**

**PEDRO FELIPE GÓMEZ BONILLA**

**CAMPUSLANDS  
INTRODUCCIÓN BASE DE DATOS**

**TIBÚ**

**2024**

Contenido

[**INTRODUCCIÓN** 3](#_Toc182984297)

[**Caso de Estudio** 4](#_Toc182984298)

[Instalación General 4](#_Toc182984299)

[**Planificación** 5](#_Toc182984300)

[Gráfica 5](#_Toc182984301)

[**Construcción del modelo Lógico** 6](#_Toc182984302)

[**Relaciones y Cardinalidad** 6](#_Toc182984303)

[Grafica 8](#_Toc182984304)

[**Normalización del modelo Lógico** 8](#_Toc182984305)

[**Primera forma Normal(1FN)** 8](#_Toc182984306)

[**Segunda forma Normal (2FN)** 9](#_Toc182984307)

[**Tercera forma Normal (3FN)** 10](#_Toc182984308)

[**Construcción del modelo físico** 10](#_Toc182984309)

[**Tablas** 11](#_Toc182984310)

[**Construcción del diagrama UML** 14](#_Toc182984311)

[**Grafica** 15](#_Toc182984312)

[**Inserción de datos** 15](#_Toc182984313)

[**Código para insertar datos en la tabla MYSQL** 16](#_Toc182984314)

[**Ejemplo** 16](#_Toc182984315)

[**Inserción de datos en MYSQL:** 16](#_Toc182984316)

[**Consultas** 16](#_Toc182984317)

[**Funciones** 24](#_Toc182984318)

[**Procedimientos** 28](#_Toc182984319)

[**Usar la base de datos con MySQL** 34](#_Toc182984320)

[**Creación de tablas** 34](#_Toc182984321)

[**Usar las Funciones en MySQL** 38](#_Toc182984322)

# 

# **INTRODUCCIÓN**

Este documento contiene una guía detallada del proceso de diseño de una base de datos para una empresa de renta de vehículos llamada *AutoRenta*. Esta empresa nos contrató con el fin de crear un sistema de información, para poder llevar un registro detallado de sus clientes, empleados, vehículos y alquileres.

Comenzamos con el análisis de requerimientos dados por la empresa y aplicamos un caso de estudio donde procederemos a identificar las entidades principales y a partir de esto creamos el modelo conceptual descriptivo de la estructura de la empresa *AutoRenta*.

Luego de construir el modelo conceptual procederemos a crear el modelo lógico, quien provee más información del proceso de construcción de la base de datos. En este paso aplicaremos la Normalización, que parte desde de la primera forma(1FN), segunda forma (2FN) y termina en la tercera(3FN), ofreciendo una descripción de cada una de ellas para mejor comprensión.

A continuación, se hará la conversión del modelo lógico al físico, en donde obtendremos su implementación real de todo el proceso antes realizado, añadiendo el tipo de dato y la cantidad de caracteres de cada elemento. También implementaremos un diagrama UML para visualizar la estructura de la base de datos.

# **Caso de Estudio**

La empresa AutoRenta nos ha pedido diseñar una base de datos que gestione las sucursales, clientes, empleados, vehículos y alquileres.

***Sucursales***:

ciudad y dirección donde se ubica, teléfono fijo, celular y correo electrónico.

***Empleados***:

sucursal donde labora, cédula, nombres, apellidos, dirección y ciudad de residencia, celular y correo electrónico.

***Clientes***:

cédula, nombres, apellidos, dirección y ciudad de residencia, celular y correo

electrónico.

***Vehículos***:

tipo de vehículo, placa, referencia, modelo, puertas, capacidad, sunroof, motor, color.

***Alquileres***:

vehículo, cliente, empleado, sucursal y fecha de salida, sucursal y fecha de llegada, fecha esperada de llegada, valor de alquiler por semana, valor de alquiler por día, porcentaje de descuento, valor cotizado y valor paga.

# Instalación General

El proceso de desarrollo de bases de datos para la empresa AutoRenta, se encuentra en la plataforma de GitHub en una carpeta que contiene este documento.

# **Planificación**

Ejecución: Después de analizar la información requerida por la empresa AutoRenta se comenzó la creación del modelo conceptual. Este modelo describe la estructura necesaria de la base de datos, dando una visualización de sus relaciones y entidades principales.

El modelo conceptual se identifica por sus formas y colores, a continuación, describiremos sus características visuales.

Entidad

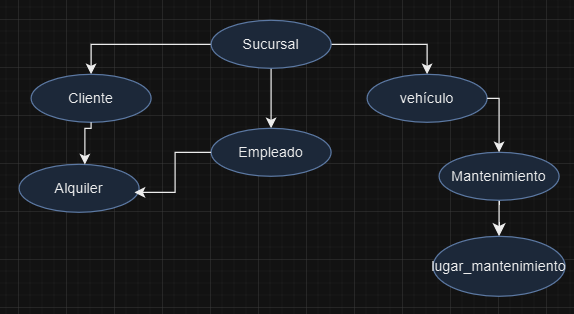
***Entidades***:

1. Sucursal
2. Cliente
3. Empleado
4. Alquiler
5. Vehículo
6. Mantenimiento
7. Lugar de mantenimiento

***Relación***:

* Una sucursal puede dar servicio a muchos clientes, y un cliente puede buscar estos servicios en varias sucursales.
* Una sucursal tiene muchos empleados, un empleado trabaja en una sucursal.
* Una sucursal ofrece muchos vehículos, y un vehículo puede ser ofrecido en muchas sucursales.
* Un vehículo puede tener muchos mantenimientos a lo largo de su servicio con la empresa y un mantenimiento es realizado a un vehículo.
* El lugar de mantenimiento puede ofrecer sus servicios a varios mantenimientos de vehículos y ese mantenimiento le pertenece a un lugar.
* Un cliente puede buscar los servicios de vehículos varias veces, y el alquiler se registra a un cliente.
* Un empleado puede gestionar un alquiler, y ese alquiler es gestionado por un empleado.

## Gráfica



# **Construcción del modelo Lógico**

La construcción del modelo lógico sigue la misma estructura del modelo conceptual. En este modelo se le añaden los atributos y se identifican las llaves primarias y foráneas, aplicando la cardinalidad.

Se crea una tabla conectora cuando la cardinalidad es de muchos a muchos

## **Relaciones y Cardinalidad**

1. *Sucursal- Cliente*

Una sucursal puede dar servicio a muchos clientes, y un cliente puede buscar estos servicios en varias sucursales.

**Cardinalidad: n-m (muchos a muchos)**



1. *Sucursal-Empleado*

Una sucursal tiene muchos empleados, un empleado trabaja en una sucursal.

**Cardinalidad: 1-n (uno a muchoos)**



1. *Sucursal-Vehículo*

Una sucursal ofrece muchos vehículos, y un vehículo es alquilado en una sucursal

**Cardinalidad: 1-n (uno a muchos)**



1. *Vehículo-Mantenimiento*

Un vehículo puede tener muchos mantenimientos a lo largo de su servicio con la empresa y un mantenimiento es realizado a un vehículo.

**Cardinalidad: 1-n (uno a muchos)**



1. *Lugar\_manteimiento-Mantenimiento*

El lugar de mantenimiento puede ofrecer sus servicios a varios mantenimientos de vehículos y ese mantenimiento le pertenece a un lugar.

**Cardinalidad: 1-n (uno a muchos)**



1. *Cliente-Alquiler*

Un cliente puede buscar los servicios de vehículos varias veces, y el alquiler se registra a un cliente.

**Cardinalidad: 1-n (uno a muchos)**



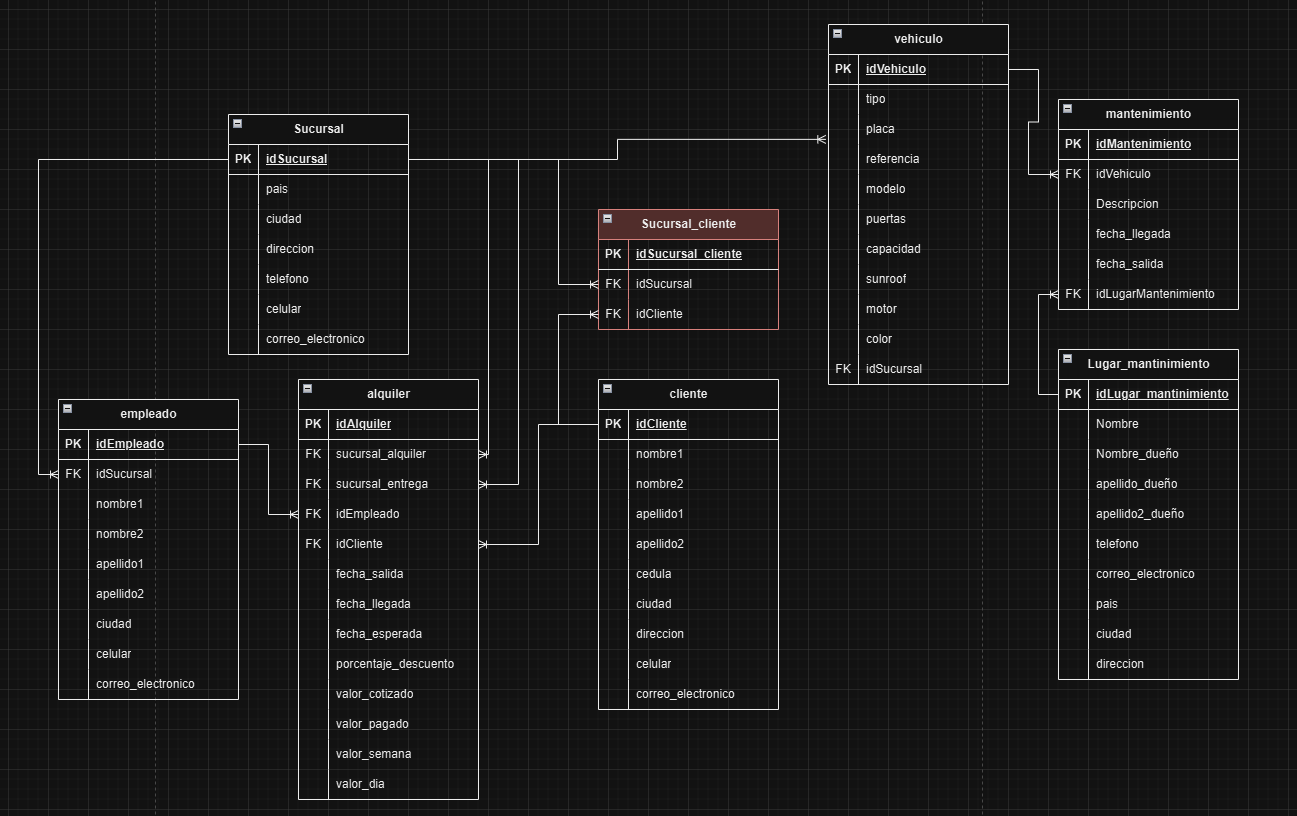
1. *Empleado-Alquiler*

Un empleado puede gestionar un alquiler, y ese alquiler es gestionado por un empleado.

**Cardinalidad: 1-n (uno a muchos)**



## Grafica



Las tablas rojas son las conectoras de la cardinalidad muchos a muchos.

# **Normalización del modelo Lógico**

Se realizó la normalización al modelo lógico con el fin de obtener mayor coherencia entre las tablas.

## **Primera forma Normal(1FN)**

* Cada casilla debe tener un dato.
* Cada atributo debe contener valores atómicos.
* Cada fila de la misma tabla debe contener datos diferentes.

**Descripción**: La tabla se encuentra en la primera forma normal después de cumplir con los requerimientos antes mencionados. Esta formal nos ayuda con el orden y la coherencia, proporcionando identificadores únicos, valores únicos y no repetidos.

**Sucursal**: se encuentra en 1FN, ya que cuenta con un identificador único y cada columna cuenta con valores únicos y no son repetitivos.

**Cliente**: se encuentra en 1FN, ya que cuenta con un identificador único y cada columna cuenta con valores únicos y no son repetitivos.

**Empleado**: se encuentra en 1FN, ya que cuenta con un identificador único y cada columna cuenta con valores únicos y no son repetitivos.

**Vehículo**: se encuentra en 1FN, ya que cuenta con un identificador único y cada columna cuenta con valores únicos y no son repetitivos.

**Alquiler**: se encuentra en 1FN, ya que cuenta con un identificador único y cada columna cuenta con valores únicos y no son repetitivos.

**Mantenimiento**: se encuentra en 1FN, ya que cuenta con un identificador único y cada columna cuenta con valores únicos y no son repetitivos.

**Lugar**\_**mantenimiento**: se encuentra en 1FN, ya que cuenta con un identificador único y cada columna cuenta con valores únicos y no son repetitivos.

***Relación de muchos a muchos:***

**Sucursal\_cliente**: se encuentra en 1FN, ya que cuenta con un identificador único y cada columna cuenta con valores únicos y no son repetitivos

## **Segunda forma Normal (2FN)**

* Debe estar en F1
* La relación debe tener una clave principal de preferencia simple
* Cada atributo de la tabla debe depender del atributo clave.

**Descripción**: Para estar en la segunda forma normal debemos haber cumplido con los requerimientos antes mencionados. En este paso verificamos que los datos dependan de la llave primaria.

**Sucursal**: se encuentra en 2FN, ya que cuenta con un identificador único (llave primaria) y cada columna depende de él.

**Cliente**: se encuentra en 2FN, ya que cuenta con un identificador único (llave primaria) y cada columna depende de él.

**Empleado**: se encuentra en 2FN, ya que cuenta con un identificador único (llave primaria) y cada columna depende de él.

**Vehículo**: se encuentra en 2FN, ya que cuenta con un identificador único (llave primaria) y cada columna depende de él.

**Alquiler**: se encuentra en 2FN, ya que cuenta con un identificador único (llave primaria) y cada columna depende de él.

**Mantenimiento**: se encuentra en 2FN, ya que cuenta con un identificador único (llave primaria) y cada columna depende de él.

**Lugar\_mantenimiento**: se encuentra en 2FN, ya que cuenta con un identificador único (llave primaria) y cada columna depende de él.

*Relación de muchos a muchos:*

**Sucursal\_cliente:** se encuentra en 2FN, ya que cuenta con un identificador único (llave primaria) y cada columna depende de él.

## **Tercera forma Normal (3FN)**

* Debe estar en F2
* se eliminan los datos repetidos

**Descripción**: En esta tercera forma normal tendremos en cuenta todos los requerimientos antes mencionados y eliminaremos datos repetidos que puedan dañar o generar una sobrecarga de datos en la base de datos, para esto podremos crear una nueva columna.

**Sucursal**: se encuentra en 3FN, ya que pasó por la segunda forma y no cuenta con datos repetidos.

**Cliente**: se encuentra en 3FN, ya que pasó por la segunda forma y no cuenta con datos repetidos.

**Empleado**: se encuentra en 3FN, ya que pasó por la segunda forma y no cuenta con datos repetidos.

**Vehículo**: se encuentra en 3FN, ya que pasó por la segunda forma y no cuenta con datos repetidos.

**Alquiler**: se encuentra en 3FN, ya que pasó por la segunda forma y no cuenta con datos repetidos.

**Mantenimiento**: se encuentra en 3FN, ya que pasó por la segunda forma y no cuenta con datos repetidos.

**Lugar\_mantenimiento**: se encuentra en 3FN, ya que pasó por la segunda forma y no cuenta con datos repetidos.

Relación de muchos a muchos:

**Sucursal\_cliente:** se encuentra en 3FN, ya que pasó por la segunda forma y no cuenta con datos repetidos.

# **Construcción del modelo físico**

Este modelo se implementa después de haber creado el lógico. En esta parte de la estructura de datos pasaremos todo lo antes realizado a un lenguaje de sistema de gestión de datos MYSQL, donde podremos crear tablas a partir de las entidades con sus atributos e implementar la inserción de datos.

Descripción: En el sistema de gestión de datos creamos tablas, estas tablas representan las entidades, y las entidades contienen sus respectivos atributos organizados en columnas, a quienes se le ingresa el tipo de dato con la cantidad de caracteres si es necesario.

## **Tablas**

* Código para crear la base de datos en MYSQL

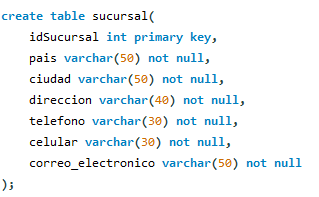


* Código para crear las tablas en MYSQL

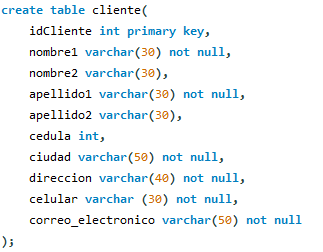


* En estas imágenes se puede ver la creación de las tablas basadas en las entidades con sus respectivos atributos, tipo de datos y cantidad de caracteres.

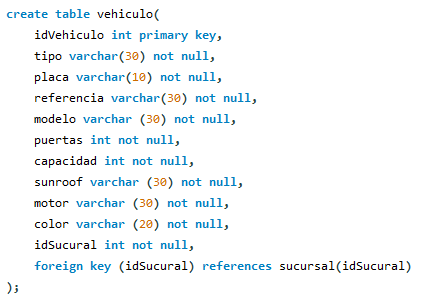
1. **Sucursal**



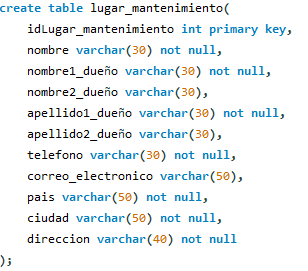
1. **Cliente**



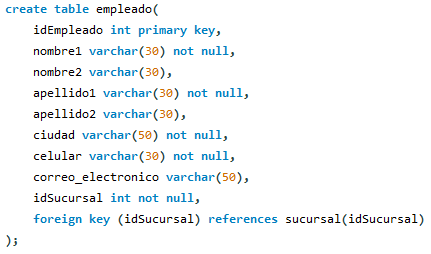
1. **Vehículo**



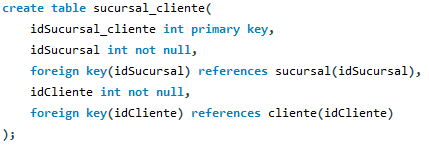
1. **Lugar de mantenimiento**



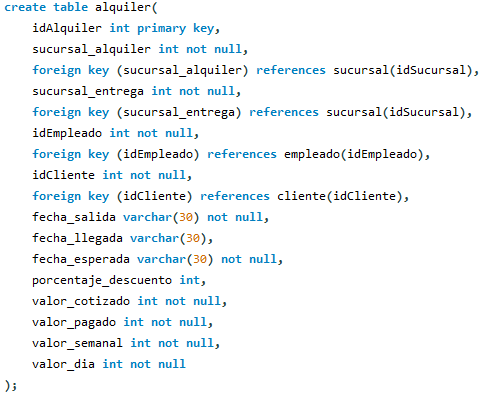
1. **Empleado**



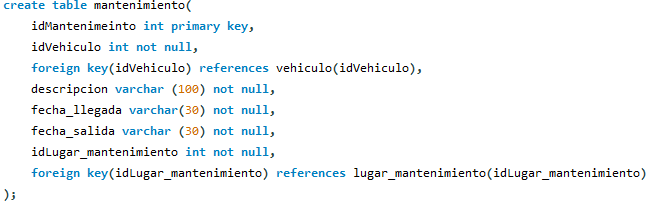
1. **Sucursal\_cliente**



1. **Alquiler**



1. **Mantenimiento**

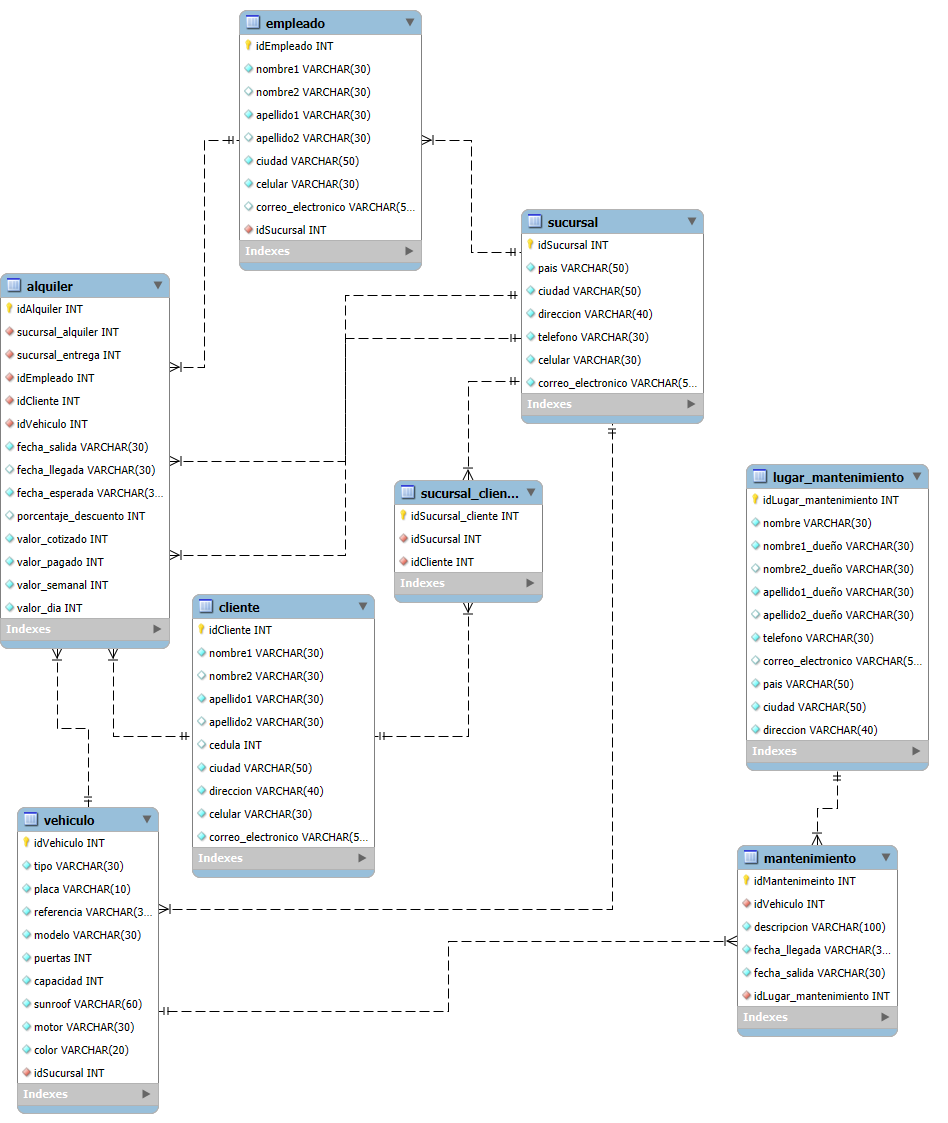


# **Construcción del diagrama UML**

En la construcción del diagrama UML toma como referencia la normalización para entender mejor la estructura de la base de datos. Este diagrama nos ayudará a visualizar y entender de una mejor manera su construcción e implementación de los requerimientos dados por la empresa ***AutoRenta***.

**Descripción**: El diagrama UML ha sido diseñado con el objetivo de representar detalladamente la estructura de cada tabla y sus relaciones. En este diagrama se puede visualizar el nombre de la tabla, sus atributos con su tipo de dato y el limitante de caracteres. Los tipos de datos más usados en este proceso son: primary key(llave primaria), foreing key(llave foránea), varchar(cadena) y int(números enteros).

## **Grafica**



# **Inserción de datos**

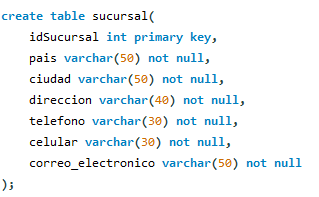
La inserción de datos es una parte muy importante del desarrollo de bases de datos, en esta parte agregamos los datos que almacenará la tabla.

## **Código para insertar datos en la tabla MYSQL**

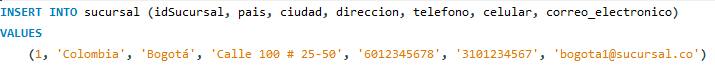


**Ejemplo**: Se insertaron los datos de la entidad sucursal siguiendo el orden de la tabla antes creada en MySQL

**Tabla sucursal:**

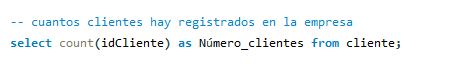


### **Inserción de datos en MYSQL:**

 se ingresaron los datos a los atributos de la tabla, siguiendo el orden de construcción antes creado. Si ejecuta la inserción y no cumplió con el orden y la cantidad de caracteres antes asignados nos dará un error.

# **Consultas**

Las consultas es una forma de interactuar con los datos de la base anteriormente creada, su uso se hace mediante comandos. A continuación, les mostrare una serie de comandos realizados en MySQL para interactuar con la base de datos de una manera que usted le podría interesar.

1. En esta primera consulta entramos a la tabla clientes y con el count() contamos las filas de la tabla con el identificador del cliente, con el fin de obtener el total de clientes registrados en la empresa autoRenta sin importar la sucursal.

* Resultado



1. Este segundo comando nos da como resultado el total de empleados de la empresa autoRenta sin importa la sucursal donde se encuentran trabajando.



* Resultado



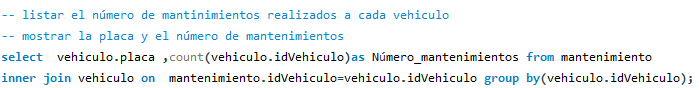
1. En este comando contamos los empleados que hay en cada sucursal. Hice uso de los inner join para entrar a la información de otra tabla, en este caso la tabla sucursal, con el fin de comparar los identificadores mediante la llave primaria de la tabla sucursal y la llave foránea de la tabla empleado que contiene el identificador de la sucursal y luego agrupar la suma de empleados según el id.



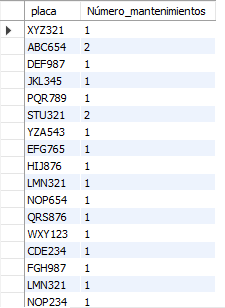
* Resultado



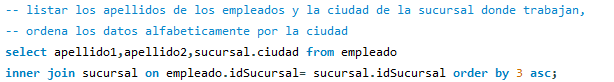
1. En esta consulta mostramos el número de mantenimientos realizados por cada vehículo. hice uso del inner join para ingresar a la información de otra tabla, en este caso la tabla vehículo, con el fin de comparar los identificadores mediante la llave primaria de la tabla vehículo y la llave foránea de la tabla mantenimiento que contiene el identificador del vehículo y luego agrupar la suma de vehículos según el id.



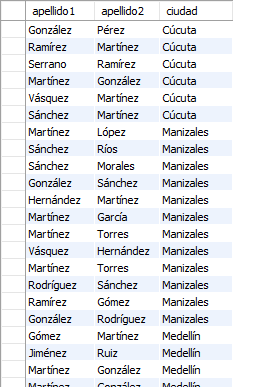
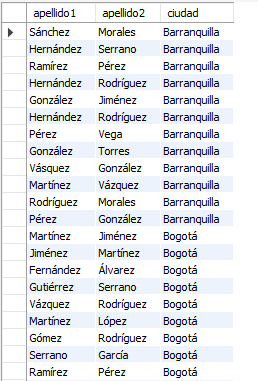
* Resultado



1. En esta consulta listamos datos del empleado y el lugar de la sucursal donde trabaja, dando orden alfabéticamente con el nombre de la ciudad. hice uso del inner join para ingresar a la información de otra tabla, en este caso la tabla sucursal, con el fin de comparar los identificadores mediante la llave primaria de la tabla sucursal y la llave foránea de la tabla empleado que contiene el identificador de la sucursal e hice el orden alfabético con el número 3, que en este caso representa la ciudad porque se encuentra en la posición número 3 después del select.



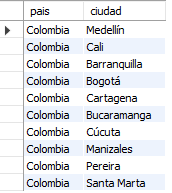
* Resultado



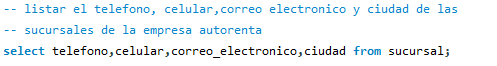
1. En este comando mostramos la información del lugar de las sucursales de la empresa autoRenta.



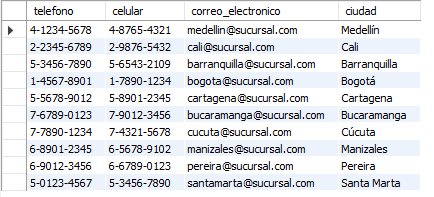
* Resultado



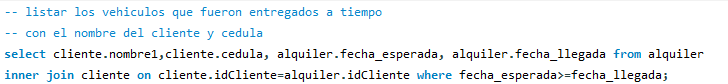
1. En este comando mostramos el contacto de las sucursales de la empresa autoRenta



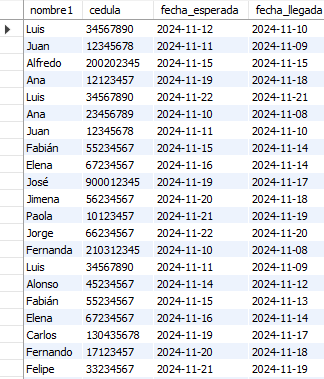
* Resultado



1. En esta consulta se hizo uso del inner join para ingresar a la información de otra tabla, en este caso la tabla cliente, con el fin de comparar los identificadores mediante la llave primaria de la tabla cliente y la llave foránea de la tabla alquiler que contiene el identificador del cliente e hice una condición usando el where para comparar las fechas y si la fecha esperada es mayor o igual a la de llegada muestra el resultado.



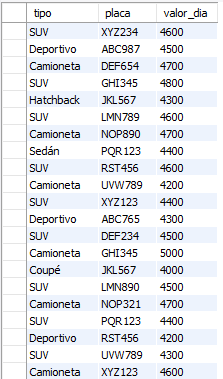
* Resultado



1. En esta consulta se hizo uso del inner join para ingresar a la información de otra tabla, en este caso la tabla vehículo, con el fin de comparar los identificadores mediante la llave primaria de la tabla vehículo y la llave foránea de la tabla alquiler que contiene el identificador del vehículo.



* Resultado

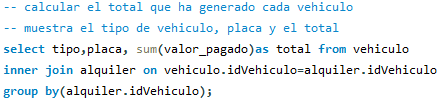
1. En esta consulta se hizo uso del sum, para sumar todo el registro de pagos.



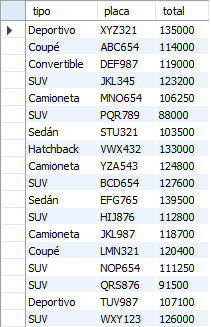
* Resultado



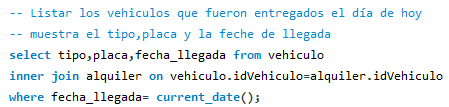
1. En esta consulta se hizo uso del sum, para sumar todo el registro de pagos por cada vehículo según su identificador. se hizo uso del inner join para ingresar a la información de otra tabla, en este caso la tabla alquiler con el fin de comparar los identificadores mediante la llave primaria de la tabla vehículo y la llave foránea de la tabla alquiler que contiene el identificador del vehículo.



* Resultado

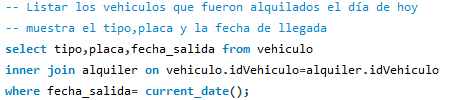
1. En esta consulta se hizo uso del inner join para ingresar a la información de otra tabla, en este caso la tabla alquiler con el fin de comparar los identificadores mediante la llave primaria de la tabla vehículo y la llave foránea de la tabla alquiler que contiene el identificador del vehículo e hice uso del where para poder identificar la fecha de llegada con la fecha actual.



* Resultado



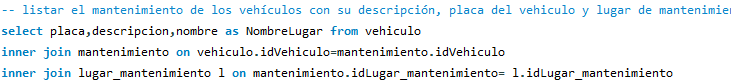
1. En esta consulta se hizo uso del inner join para ingresar a la información de otra tabla, en este caso la tabla alquiler con el fin de comparar los identificadores mediante la llave primaria de la tabla vehículo y la llave foránea de la tabla alquiler que contiene el identificador del vehículo e hice uso del where para poder identificar la fecha de salida con la fecha actual.



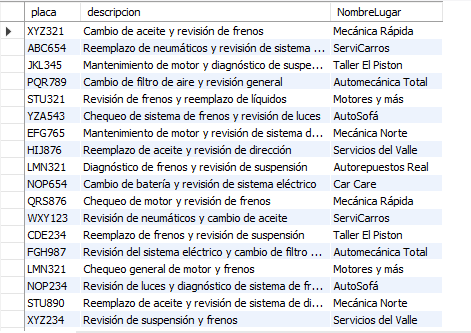
* Resultado



1. En esta consulta se hizo uso de 2 inner join para ingresar a la información de otra tabla, en este caso la tabla alquiler con el fin de comparar los identificadores mediante la llave primaria de la tabla vehículo y la llave foránea de la tabla mantenimiento que contiene el identificador del vehículo. Otra tabla utilizada es lugar\_mantenimiento con el fin de comparar los identificadores mediante la llave primaria de la tabla lugar\_mantenimiento y la llave foránea de la tabla mantenimiento que contiene el identificador del lugar\_mantenimiento.



* Resultado

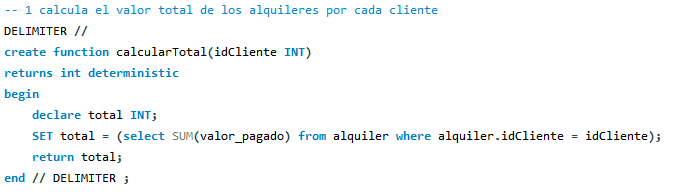


# **Funciones**

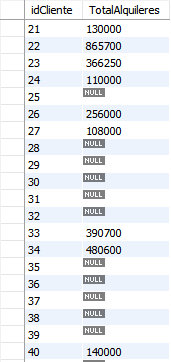
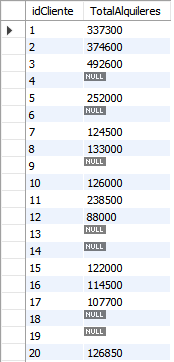
Las funciones creadas para esta base de datos nos ayudan a encapsular código que se puede utilizar múltiples veces . Estas funciones cumplen con tareas específicas hechas mediante consultas que podrás usar de manera sencilla.

A continuación, te proporcionare 5 funciones creadas para esta base de datos y te explicare su función.

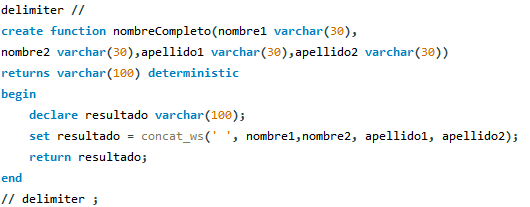
* Explicación
* Imagen de la función
* Imagen de como ejecutar la función
* Imagen de resultado

1. Se creó una función con el fin de calcular el valor total de los alquileres hechos por cada cliente

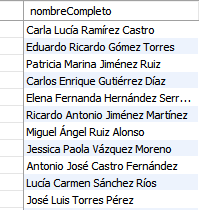
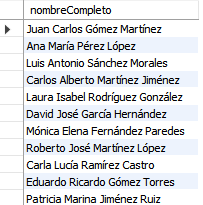
* 
* Resultado



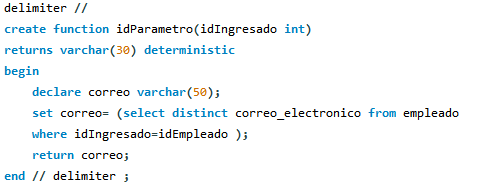
1. Se creó una función que cumple con el objetivo de mostrar el nombre completo de todos los empleados en una columna.



* 
* Resultado



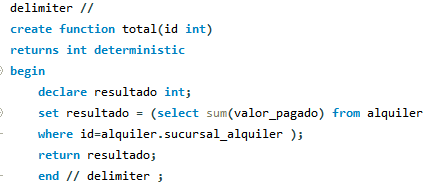
1. Esta función cumple con mostrar el correo electrónico del empleado según el id que tu ingresas, en este caso el id 23, si se desea ver el correo de otro empleado solo cambia el número 23 por otro.



* 
* Resultado



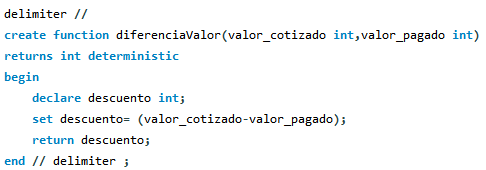
1. Esta función calcula el total de alquileres por el id de la sucursal. En este caso el resultado es de la sucursal que tiene el número 2 por id, si deseas ver el valor de otra sucursal cambia el id 2 por el que necesites revisar.

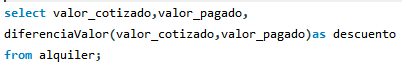


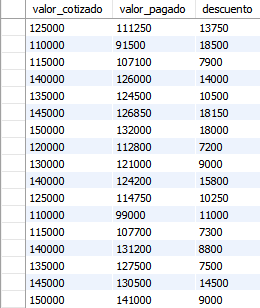
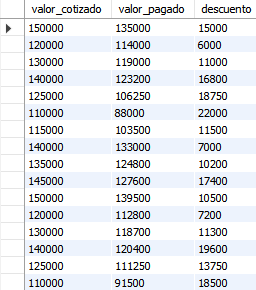
* 
* Resultado



1. Esta función calcula el descuento que se le ha hecho a cada alquiler y muestra una lista del valor cotizado, valor pagado y el descuento de cada alquiler.



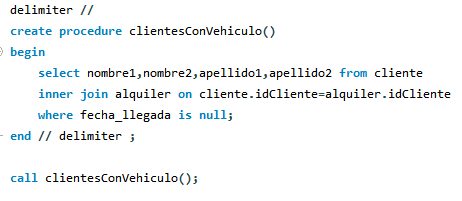
* 
* Resultado



# **Procedimientos**

* Explicación
* Imagen del procedimiento
* Imagen de como ejecutar el procedimiento
* Imagen de resultado

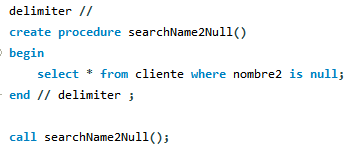
1. La empresa auntoRenta puede tener registros sin entregas de vehículos y el dato puede estar en nulo, se buscó realizar un procedimiento que facilite esa búsqueda y muestra una lista de todos los clientes que no han entregado el vehículo.



* Resultado



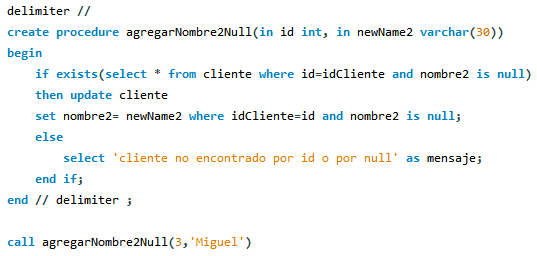
1. Esta función devuelve un listado de todos los clientes que no tienen un segundo nombre registrado en la base de datos.

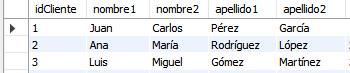


* Resultado

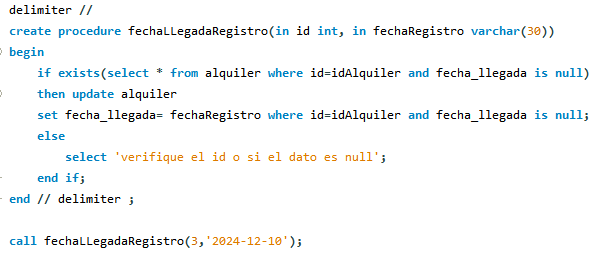


1. Este procedimiento agrega el segundo nombre a los clientes que tengan el dato nulo. Para poder usarlo se debe llamar el procedimiento con call y el nombre del procedimiento seguido del id del cliente e ingresar el segundo nombre entre comillas.

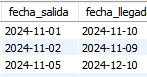
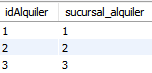


* 

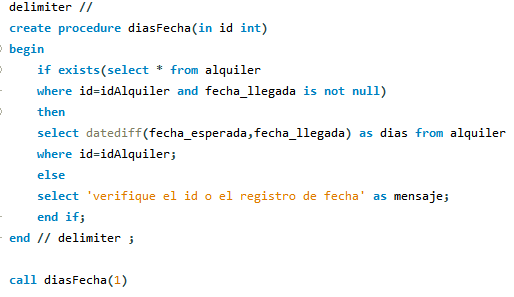
1. Este procedimiento le puede facilitar a la empresa autoRenta el registro de la fecha de llegada del vehículo que aún no se había sido entregado. Para poder registrar la fecha se debe ingresar el id del alquiler correspondiente al vehículo con la fecha de llegada cuando esta sea nula. Si no se encuentra el id y el dato no es nulo se muestra un mensaje.



* Resultado



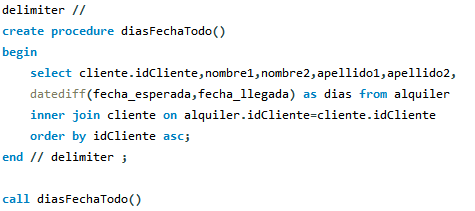
1. Este procedimiento calcula los días que hay entre la fecha esperada y la fecha de llegada del vehículo que desees revisar ingresando el id del alquiler.



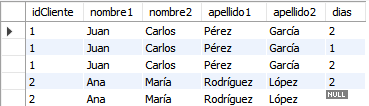
* Resultado



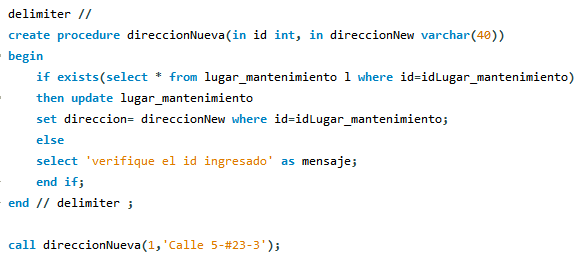
1. Este procedimiento calcula los días de todos los alquileres con retraso y el cliente que alquilo ese vehículo.



* Resultado



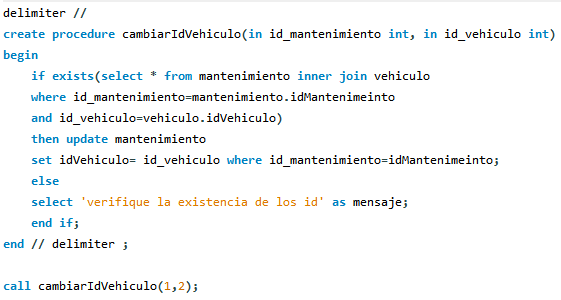
1. Si el lugar de mantenimiento muda su local, su dirección debe cambiar para mayor coherencia con los datos. Se creó un procedimiento que facilite este proceso ingresando el id del lugar que desea cambiar y la nueva dirección.



* Resultado

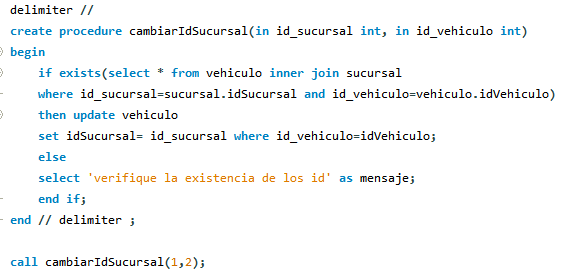
1. si hubo un error de registro en el mantenimiento de un vehículo, este procedimiento cambia el id del vehiculo ingresando el id del mantenimiento y el id del nuev vehiculo que desea registrar.



* Resutado



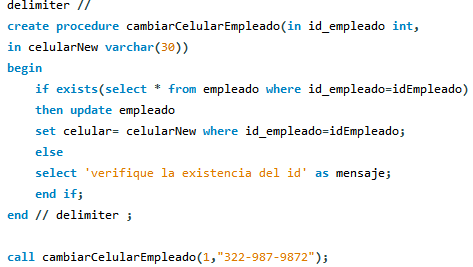
1. Si hubo un error al ingresar la sucursal de la tabla vehiculo, creamos un procedimiento que facilita el cambio del id de la sucursal . para poder usar el procedimiento debes ingresar de primero el id correcto de la sucursal seguido del id del vehículo donde esta el error.



* Resultado

1. Si un empleado cambió de celular y se debe ingresar el nuevo número, podras usar este procedimiento ingresando el id del empleado donde ira el nuevo número y separado por una coma ingresa el nuevo número como se muestra en el final de la primera imagen.



* Resultado

# **Usar la base de datos con MySQL**

Si deseas revisar la base de datos por tu cuenta, a continuación, te proporcionare un texto con las tablas ya creadas y los datos de inserción, solo debes copiar y pegar en un script de MySQL.

## **Creación de tablas**

## 

create database autoRenta;

use autoRenta;

create table sucursal(

idSucursal int primary key,

pais varchar(50) not null,

ciudad varchar(50) not null,

direccion varchar(40) not null,

telefono varchar(30) not null,

celular varchar(30) not null,

correo\_electronico varchar(50) not null

);

-- 2

create table cliente(

idCliente int primary key,

nombre1 varchar(30) not null,

nombre2 varchar(30),

apellido1 varchar(30) not null,

apellido2 varchar(30),

cedula int,

ciudad varchar(50) not null,

direccion varchar(40) not null,

celular varchar (30) not null,

correo\_electronico varchar(50) not null

);

-- 3

create table lugar\_mantenimiento(

idLugar\_mantenimiento int primary key,

nombre varchar(30) not null,

nombre1\_dueño varchar(30) not null,

nombre2\_dueño varchar(30),

apellido1\_dueño varchar(30) not null,

apellido2\_dueño varchar(30),

telefono varchar(30) not null,

correo\_electronico varchar(50),

pais varchar(50) not null,

ciudad varchar(50) not null,

direccion varchar(40) not null

);

-- 4

create table vehiculo(

idVehiculo int primary key,

tipo varchar(30) not null,

placa varchar(10) not null,

referencia varchar(30) not null,

modelo varchar (30) not null,

puertas int not null,

capacidad int not null,

sunroof varchar (60) not null,

motor varchar (30) not null,

color varchar (20) not null,

idSucursal int not null,

foreign key (idSucursal) references sucursal(idSucursal)

);

-- 5

create table empleado(

idEmpleado int primary key,

nombre1 varchar(30) not null,

nombre2 varchar(30),

apellido1 varchar(30) not null,

apellido2 varchar(30),

ciudad varchar(50) not null,

celular varchar(30) not null,

correo\_electronico varchar(50),

idSucursal int not null,

foreign key (idSucursal) references sucursal(idSucursal)

);

-- 6

create table sucursal\_cliente(

idSucursal\_cliente int primary key,

idSucursal int not null,

foreign key (idSucursal) references sucursal(idSucursal),

idCliente int not null,

foreign key(idCliente) references cliente(idCliente)

);

-- 7

create table alquiler(

idAlquiler int primary key,

sucursal\_alquiler int not null,

foreign key (sucursal\_alquiler) references sucursal(idSucursal),

sucursal\_entrega int not null,

foreign key (sucursal\_entrega) references sucursal(idSucursal),

idEmpleado int not null,

foreign key (idEmpleado) references empleado(idEmpleado),

idCliente int not null,

foreign key (idCliente) references cliente(idCliente),

idVehiculo int not null,

foreign key(idVehiculo) references vehiculo(idVehiculo),

fecha\_salida varchar(30) not null,

fecha\_llegada varchar(30),

fecha\_esperada varchar(30) not null,

porcentaje\_descuento int,

valor\_cotizado int not null,

valor\_pagado int not null,

valor\_semanal int not null,

valor\_dia int not null

);

-- 8

create table mantenimiento(

idMantenimeinto int primary key,

idVehiculo int not null,

foreign key(idVehiculo) references vehiculo(idVehiculo),

descripcion varchar (100) not null,

fecha\_llegada varchar(30) not null,

fecha\_salida varchar (30) not null,

idLugar\_mantenimiento int not null,

foreign key(idLugar\_mantenimiento) references lugar\_mantenimiento(idLugar\_mantenimiento)

);

# **Usar las Funciones en MySQL**

Si deseas revisar y utilizar las funciones anteriormente explicadas con imágenes puedes copiar y pegar en MySQL la función que deseas, recuerda que para dar uso de ellas debes tener las tablas e inserciones de datos y si ya tienes eso utiliza la base de datos con el Use autoRenta;

use autorenta;

-- Procedimiento autoRenta

-- 1. consultar los clientes que no han entregado el vehículo

delimiter //

create procedure clientesConVehiculo()

begin

select idAlquiler,nombre1,nombre2,apellido1,apellido2 from cliente

inner join alquiler on cliente.idCliente=alquiler.idCliente

where fecha\_llegada is null;

end // delimiter ;

call clientesConVehiculo();

-- 2. buscar el segundo nombre que tenga null en la tabla clientes

-- mostrar todos los datos

delimiter //

create procedure searchName2Null()

begin

select \* from cliente where nombre2 is null;

end // delimiter ;

call searchName2Null();

-- 3. agregarle el segundo nombre a los clientes que tengan null

-- ingresar el segundo nombre por id

delimiter //

create procedure agregarNombre2Null(in id int, in newName2 varchar(30))

begin

if exists(select \* from cliente where id=idCliente and nombre2 is null)

then update cliente

set nombre2= newName2 where idCliente=id and nombre2 is null;

else

select 'cliente no encontrado por id o por null' as mensaje;

end if;

end // delimiter ;

call agregarNombre2Null(3,'Miguel')

-- 4. registrar la fecha de llegada de los vehiculos

-- registrar por id

delimiter //

create procedure fechaLLegadaRegistro(in id int, in fechaRegistro varchar(30))

begin

if exists(select \* from alquiler where id=idAlquiler and fecha\_llegada is null)

then update alquiler

set fecha\_llegada= fechaRegistro where id=idAlquiler and fecha\_llegada is null;

else

select 'verifique el id o si el dato es null';

end if;

end // delimiter ;

call fechaLLegadaRegistro(3,'2024-12-10');

-- 5 calcular los días que hay entre la fecha esperada y la fecha de llegada

delimiter //

create procedure diasFecha(in id int)

begin

if exists(select \* from alquiler

where id=idAlquiler and fecha\_llegada is not null)

then

select datediff(fecha\_esperada,fecha\_llegada) as dias from alquiler

where id=idAlquiler;

else

select 'verifique el id o el registro de fecha' as mensaje;

end if;

end // delimiter ;

call diasFecha(1);

-- 6. mostrar una lista con la cantidad de dias

-- de todos los alquileres con retraso

-- mostrar los nombres y apellidos del cliente

delimiter //

create procedure diasFechaTodo()

begin

select cliente.idCliente,nombre1,nombre2,apellido1,apellido2,

datediff(fecha\_esperada,fecha\_llegada) as dias from alquiler

inner join cliente on alquiler.idCliente=cliente.idCliente

order by idCliente asc;

end // delimiter ;

call diasFechaTodo();

-- 7 cambiar la dirreción del lugar de mantenimiento

-- por el id ingresado

delimiter //

create procedure direccionNueva(in id int, in direccionNew varchar(40))

begin

if exists(select \* from lugar\_mantenimiento l where id=idLugar\_mantenimiento)

then update lugar\_mantenimiento

set direccion= direccionNew where id=idLugar\_mantenimiento;

else

select 'verifique el id ingresado' as mensaje;

end if;

end // delimiter ;

call direccionNueva(1,'Calle 5-#23-3');

-- 8. si hubo un error de registro en el mantenimiento de un vehículo

-- crea un procedimiento para cambiar el id del vehiculo

-- cambia el dato segun el id del mantenimiento ingresado

delimiter //

create procedure cambiarIdVehiculo(in id\_mantenimiento int, in id\_vehiculo int)

begin

if exists(select \* from mantenimiento inner join vehiculo

where id\_mantenimiento=mantenimiento.idMantenimeinto

and id\_vehiculo=vehiculo.idVehiculo)

then update mantenimiento

set idVehiculo= id\_vehiculo where id\_mantenimiento=idMantenimeinto;

else

select 'verifique la existencia de los id' as mensaje;

end if;

end // delimiter ;

call cambiarIdVehiculo(1,2);

-- 9. Hubo un error al ingresar un dato en la tabla vehiculo

-- crea un procedimiento para cambiar el id de la sucursal

-- cambia el dato según el id ingresado por el usuario

delimiter //

create procedure cambiarIdSucursal(in id\_sucursal int, in id\_vehiculo int)

begin

if exists(select \* from vehiculo inner join sucursal

where id\_sucursal=sucursal.idSucursal and id\_vehiculo=vehiculo.idVehiculo)

then update vehiculo

set idSucursal= id\_sucursal where id\_vehiculo=idVehiculo;

else

select 'verifique la existencia de los id' as mensaje;

end if;

end // delimiter ;

call cambiarIdSucursal(1,2);

-- 10. Un empleado cambió de celular

-- registra el nuevo dato del empleado

-- según el id ingresado

delimiter //

create procedure cambiarCelularEmpleado(in id\_empleado int,

in celularNew varchar(30))

begin

if exists(select \* from empleado where id\_empleado=idEmpleado)

then update empleado

set celular= celularNew where id\_empleado=idEmpleado;

else

select 'verifique la existencia del id' as mensaje;

end if;

end // delimiter ;

call cambiarCelularEmpleado(1,"322-987-9872");