Universidad Nacional Autónoma de México

Facultad de Ingeniería

Bases de Datos

Proyecto Final
Ecobici

Integrantes:

Aguilar Maya Daniel
Gonzalez Sotelo Elias Eduardo
Velázquez Martínez Karla Andrea

Grupo: 02
Profesor: M.I.A Martha López Pelcastre
Fecha de entrega: 10 de Junio del 2023
Semestre: 2023-1

Índice

I INTRODUCCIÓN **2**
II JUSTIFICACIÓN **2**
III REQUERIMIENTOS **3**
3.1 Enunciado 3
3.2 Requerimientos Funcionales: 7
3.3 Requerimientos No Funcionales: 7
IV DISEÑO ENTIDAD-RELACIÓN **8**
V MODELO RELACIONAL **9**
5.1 Diccionario de datos 11
VI NORMALIZACIÓN **18**
VII DISEÑO FÍSICO **21**
7.1 SCRIPT seguridad.sql (DCL) 21
7.2SCRIPT creaBase.sql (DDL) 23
7.3 SCRIPT dml.sql 34
7.4 SCRIPT informes.sql 40
7.5 SCRIPT cargaInicial.sql 60
7.6 SCRIPT valida_Triggers.sql 74
VII ANEXOS **81**
7.1 ÁLGEBRA RELACIONAL **85**

I INTRODUCCIÓN

Vivimos en una era digital donde la gestión eficiente de los datos se ha vuelto esencial.En este panorama, las bases de datos se han transformado en herramientas cruciales que permiten almacenar, manipular y recuperar datos de manera organizada y coherente. Este proyecto se enfoca en la implementación de una base de datos para Ecobici, un sistema de bicicletas compartidas, con el objetivo de mejorar la eficiencia, rendimiento y capacidad de respuesta del sistema.

Para lograr este objetivo, hemos utilizado técnicas de modelado y programación en SQL, desde el modelo entidad-relación hasta el modelo relacional. Este enfoque no solo aprovecha el conocimiento y las habilidades adquiridas en el curso de bases de datos, sino que también garantiza una solución de gestión de datos robusta y escalable.

II JUSTIFICACIÓN

La necesidad de implementar una base de datos en Ecobici surge de varios factores clave. Ecobici, siendo un sistema que atiende a una gran cantidad de usuarios, genera una enorme cantidad de datos. Esta información varía desde los datos personales de los usuarios hasta los detalles del uso de las bicicletas. Para garantizar un servicio de alta calidad a sus usuarios, el sistema necesita manejar y procesar estos datos de manera rápida y eficaz.

La implementación de una base de datos en el proyecto Ecobici es fundamental para manejar de forma eficiente este flujo constante de datos. Primero, permitirá almacenar y gestionar los datos de los usuarios de manera segura y confiable. Segundo, facilitará el seguimiento de la disponibilidad y uso de las bicicletas en tiempo real, mejorando así la eficiencia del servicio.

Además, el análisis de los datos recopilados puede ayudar a identificar patrones de uso, proporcionando información valiosa para futuras decisiones y estrategias de negocio. En este sentido, el uso de técnicas de modelado y programación en SQL adquiridas durante el curso de bases de datos es una valiosa aportación a este proyecto, permitiendo la creación de una base de datos flexible, segura y eficiente.

III REQUERIMIENTOS

Obtener los requisitos funcionales y no funcionales de este proyecto es importante porque proporcionan información clave para el desarrollo de un sistema o proyecto relacionado con el plan de renta de bicicletas. Los requisitos funcionales describen las funcionalidades y características específicas que debe tener el sistema, como el registro de usuarios, el cálculo de

tarifas, el seguimiento de los viajes, etc. Por otro lado, los requisitos no funcionales se refieren a aspectos como la seguridad, el rendimiento, la usabilidad y otros criterios de calidad que deben cumplir el sistema. Estos requisitos ayudarán a guiar el diseño, la implementación y las pruebas del sistema, asegurando que cumpla con las necesidades y expectativas de los usuarios.

3.1 Enunciado

En la Ciudad de México se implementó un nuevo plan económico de renta de bicicletas a cargo de la empresa ECOBICI. Cada usuario puede adquirir una de las 3 membresías para rentar una bicicleta. Para ello, deben adquirir una tarjeta de movilidad integrada con código QR que cuesta $50 pesos la primera vez y $80 pesos cada reposición. Hay tres tipos de planes:

1. Para la categoría básica, el alquiler por día es de $118 pesos.
2. Para la categoría intermedia, el usuario debe pagar semanalmente $400 pesos.
3. Para la categoría Premium, el usuario paga anualmente $1000 pesos.

Con su membresía, tienen la posibilidad de acudir a cualquiera de las estaciones y utilizar una bicicleta las veces que deseen por la duración de su membresía. Pueden cambiar de plan cuando así lo decidan.
Al momento de sobrepasar el tiempo que les brinda su membresía, se hará el cobro de una tarifa por el tiempo excedido en ese viaje:

- Si el plan es diario, se cobra $5 pesos cada 10 minutos.
- Si el plan es mensual, se cobra el siguiente mes.
- Si es anual, se cobra el siguiente año.

El cobro se hará en el método de pago (tarjeta de crédito/débito o PayPal) de manera automática. Por lo cual, todo usuario debe tener registrado al menos un método de pago mientras su suscripción permanezca vigente. Borrar el método de pago involucra cancelar su suscripción y perder los días restantes de la misma.
Para que un usuario se registre en la plataforma, debe brindar los siguientes datos: nombre completo, fecha de nacimiento, edad (calculada por su fecha de nacimiento), el código del reverso de su INE, correo, teléfonos y el género del usuario.

Al llegar a una de las estaciones y tomar una de las bicicletas, el usuario inicia un viaje. Dicho viaje posee la fecha, la hora de inicio y la hora de fin en la base de datos. Para cada renta, se registra la estación de inicio. Durante este intervalo de tiempo, se registra en todo momento la ruta que sigue el usuario con la bicicleta. Al terminar el viaje, se guarda la estación final y la hora de llegada para poder determinar una tarifa adicional, la cual puede tener un valor de 0 si no aplica.

De las estaciones que sirven como punto de entrega y retirada, se desea saber cuáles de estas son las más concurridas por los usuarios para así poder tener las unidades de bicicletas necesarias para abastecer la demanda. De igual forma, se debe saber cuáles son las menos concurridas y en qué temporadas del año esto sucede. También se necesita conocer los días que tuvieron mayor afluencia.

Estos dos últimos puntos se deberán relacionar con el inventario general y el de cada una de las estaciones. Dentro de este inventario debe estar indicado cuántas bicicletas disponemos en general y cuántas se encuentran operativas y en mantenimiento, esto por cada modelo de bicicleta.

Por cada estación de bicis, también se posee un inventario individual con el número de bicis que se encuentran en tiempo real, las que se encuentran en tránsito y el número de terminales donde el usuario coloca su tarjeta para iniciar su viaje o recargar. Aunque los usuarios pueden descargar una app mediante la cual revisan y recargan saldo.
Si durante el viaje el usuario presentó algún percance, este debe notificarlo al momento de finalizar su trayecto y, al instante, se le enviará un formulario para contestarlo. Dicho formulario debe contener los siguientes apartados: lugar del incidente (calle, número, colonia, alcaldía y código postal), tipo de incidente (problemas con la bicicleta, coche, auto, moto, bici, peatón, etc., o si se cayó de la bicicleta), fecha y hora (la cual se llena automáticamente), así como las coordenadas de su ubicación (la app tiene un mapa para eso). Para auxiliar al usuario, hay agentes que se encargan de acudir lo más pronto posible y auxiliarlos, dependiendo del seguro que les corresponda de acuerdo a su plan: daños a terceros para el plan básico, daños a terceros y composturas de la bicicleta si es mensual el plan, o full si se trata de un plan anual.
Un punto importante es que cada vez que un usuario finalice uno de sus viajes, los datos de fecha, duración, costo, lugar de inicio y de fin deben permanecer guardados en un registro al cual siempre se tendrá acceso. Es decir, un histórico de los viajes de cada bicicleta.
Cada membresía posee por separado sus propios beneficios, siendo en el caso de la membresía básica un descuento correspondiente al tiempo que lleva suscrito. En el caso de la membresía intermedia, se le brindará al usuario un viaje gratuito cada cierto tiempo. Por último, para la membresía premium, se dispondrá de un sistema de Cashback para el usuario.

De cada membresía se deberá poder obtener el total de afiliados adscritos a ésta.
Dentro del área operacional, los empleados trabajan en una de las siguientes áreas: mantenimiento y administración. De todos los empleados se requiere conocer su nombre completo, dirección (calle, número interior, número exterior, colonia, alcaldía), RFC, estado civil

(soltero, casado, divorciado o viudo), género, idiomas que hablan y teléfono. Para los que corresponden a mantenimiento, estos dan servicio de reparación, limpieza y transporte, y se requiere su especialidad. Mientras que para los de administración, se requiere el registro de las funciones que realizan.

Para el área de recursos humanos, es importante un informe mensual de todos los empleados y sus datos: RFC, nombre completo y sueldo. Asimismo, se necesita el nombre de los empleados y el puesto de aquellos que tengan un sueldo de $13000 mensuales y pertenezcan a la tercera edad.

Se deberán registrar las faltas de cada empleado, así como realizar un histórico de ellas, con la finalidad de agilizar el conteo de éstas. Las faltas de cada empleado se registran con un consecutivo para cada uno de ellos, además de la fecha y el motivo (se tiene un catálogo definido de las causas que son permitidas para justificar la falta).
De los empleados en el área administrativa, se debe conocer la ubicación del lugar de trabajo y el trabajo que realizan.

Por último, de cada bicicleta se registra su número de serie, color, estado de la bicicleta (dañada, en funcionamiento o baja) y tamaño (chica, mediana y grande).
Por temas de contabilidad y administración, se requiere:

- Estadísticas de los daños en las bicicletas con mayor frecuencia. Top 5 de los accidentes más frecuentes (descripción del daño, cantidad).
- Estaciones con más reportes de accidentes con mayor frecuencia. Listado de estaciones con el número de accidentes en un periodo de tiempo (fecha inicio – fecha fin) ordenados de mayor a menor.
- Total de accidentes en un rango de fechas, listados de mayor a menor.
- Total de usuarios por rangos de fechas y rangos de edades (10 a 15 años, 15-20 años, 20 a 30 años, más de 30 años).
- Inventario de las bicicletas (todos los datos de las bicicletas) por estaciones con el número de viajes, por un periodo de tiempo, incluyendo el número de accidentes si ha habido.
- Listado de usuarios (datos generales), datos de su membresía y el tiempo en meses que tienen la membresía.
- Agentes mejor reconocidos en un mes específico. Para eso, cada vez que un agente auxilia a un usuario en algún incidente, el usuario llena una pequeña encuesta.
- Reporte diario que los empleados que hacen rondines entregan de manera (fecha, descripción, si hubo incidentes o no, número de accidentes, estación donde se obtiene el

reporte). Ellos cuentan a su vez con un supervisor que también es empleado que hace rondines.

- Listado de empleados con su tipo.
- Informe de los recorridos, por estación y/o por periodo de tiempo (fecha inicio y fecha fin): nombre del usuario, estación de partida, lugar de llegada, tiempo en minutos del recorrido y costo.
- Épocas del año con número de recorridos ordenados de mayor a menor.
- Obtener para cada agente sus datos personales y el listado de los accidentes que han atendido (tipo de accidente, fecha, lugar).

Respecto al manejo de la base de datos:

- Se deberán implementar procedimientos almacenados para: alta, modificación y borrado de usuarios con su plan, reposición de tarjetas y recargas.
- Registro de usuarios a la base de datos con diferentes perfiles.
- Realizar los procedimientos para poder obtener 4 estadísticas. Dichos procedimientos deben mostrar al menos 10 registros para cada caso.
- Utilizar esquemas para la base de datos, al menos 2.
-

Crear los scripts para crear los usuarios en las bases de datos que reciba como parámetro el usuario, el password y las funciones a realizar. Considerar que habrá usuarios de solo consulta, otros que agreguen o actualicen información y los administradores. Por defecto, elaborar el script para crear los siguientes usuarios: usuarioConsulta, usuarioGestor, usuarioAdministrador (administrador), con password 1234zaq*. Las contraseñas deben ser de 8 a 12 dígitos y contener una mayúscula, minúsculas, dígitos y un carácter especial (*, $, &).

3.2 Requerimientos Funcionales:

1. Registro de usuarios con datos específicos como nombre completo, fecha de nacimiento, edad, código de INE, correo electrónico, números de teléfono, y género.
2. Los usuarios deben poder adquirir una de las tres tipos de membresías (básica, intermedia, premium).
3. Los usuarios deben ser capaces de rentar bicicletas en cualquier estación de ECOBICI.
4. Los usuarios pueden cambiar de plan de membresía en cualquier momento.
5. Implementación de tarifas adicionales por tiempo excedido, basado en el tipo de membresía.

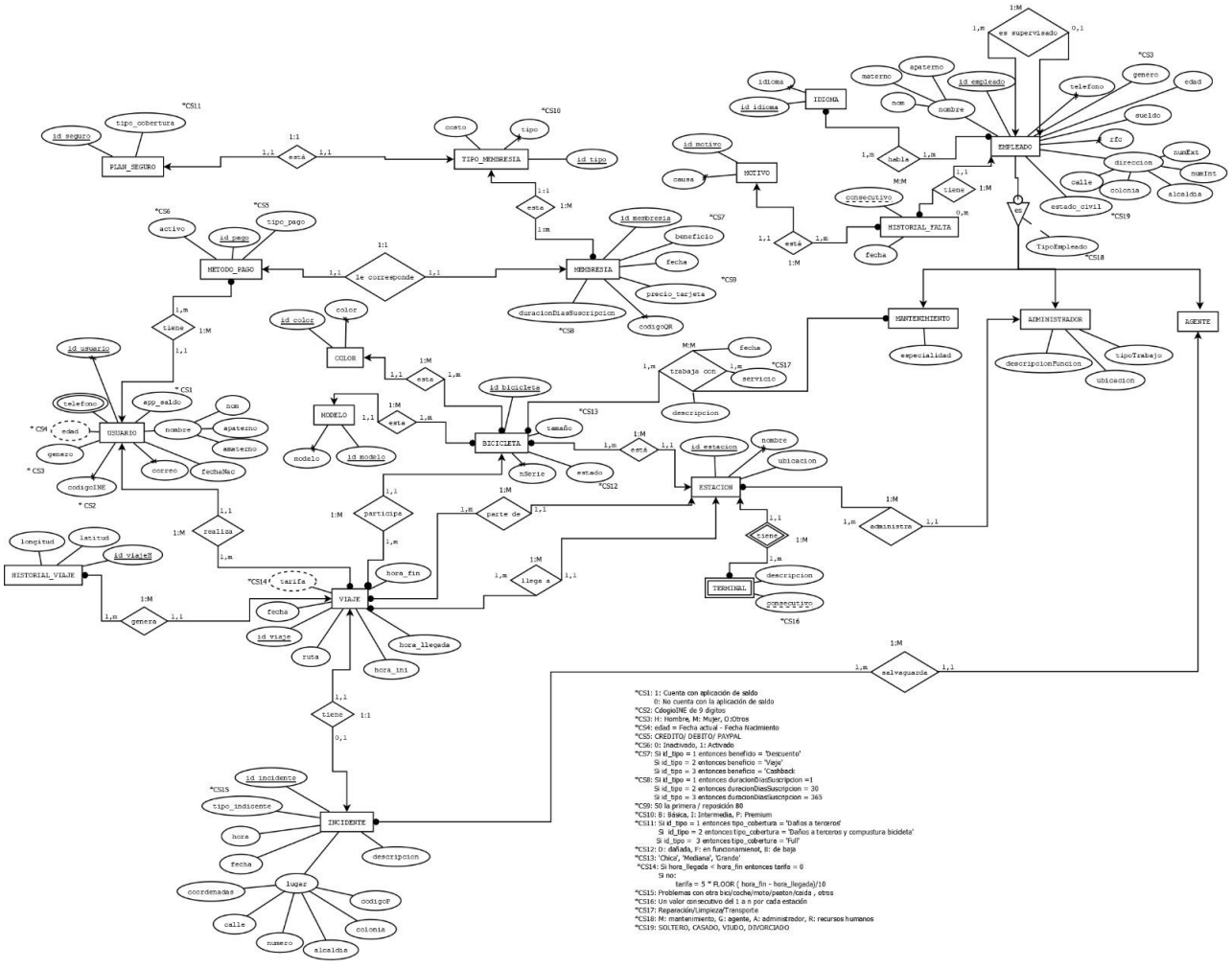
6. El sistema debe ser capaz de registrar automáticamente un viaje de un usuario, incluyendo detalles como la fecha, la hora de inicio, la hora de finalización, la estación de inicio, la ruta seguida y la estación de finalización.
7. Implementación de una característica que permita a los usuarios notificar incidentes durante su viaje.
8. El sistema debe generar un formulario de incidente con información específica cuando un usuario informa un problema.
9. Implementación de la capacidad de los usuarios de ver y recargar su saldo a través de una aplicación.
10. Mantenimiento de un inventario de bicicletas, incluyendo su número, su estado y la estación en la que se encuentran.
11. Registro de empleados con información específica, incluyendo nombre, dirección, RFC, estado civil, género, idiomas que hablan, teléfono y área de trabajo.
12. El sistema debe registrar las faltas de los empleados y mantener un historial de las mismas.
13. Procedimientos almacenados para la alta, modificación y borrado de usuarios, reposición de tarjetas y recargas, registro de usuarios a la base de datos y obtención de estadísticas.
14. Creación de scripts para crear usuarios a las bases de datos con distintos niveles de acceso y privilegios.

3.3 Requerimientos No Funcionales:

1. Seguridad: Los datos personales de los usuarios deben estar protegidos y almacenados de forma segura.
2. Disponibilidad: El sistema debe estar disponible para que los usuarios alquilen bicicletas en cualquier momento.
3. Rendimiento: El sistema debe ser capaz de manejar un gran número de usuarios al mismo tiempo sin retrasos significativos.
4. Escalabilidad: El sistema debe ser capaz de manejar un aumento en el número de usuarios o estaciones de bicicletas.
5. Usabilidad: La aplicación debe ser fácil de usar para los usuarios.
6. Mantenibilidad: El sistema debe ser fácil de mantener y actualizar.
7. Confiabilidad: El sistema debe ser confiable y tener un alto tiempo de actividad.
8. Las contraseñas de los usuarios deben tener un nivel mínimo de complejidad para garantizar la seguridad.
9. Los datos históricos deben ser preservados para futuras consultas y análisis.
10. Los datos deben ser precisos y actualizarse en tiempo real.

IV DISEÑO ENTIDAD-RELACIÓN

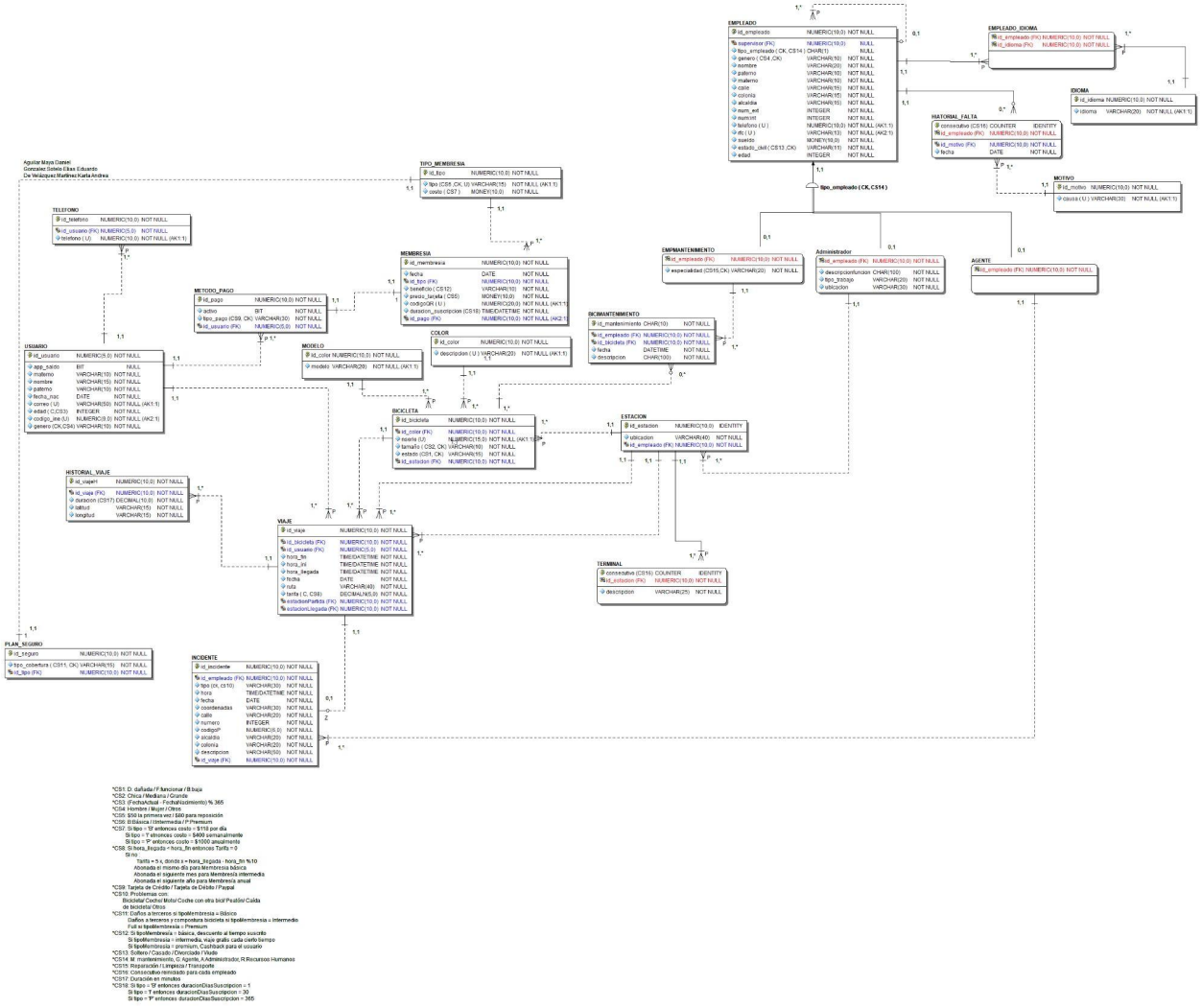
El modelo entidad-relación es esencial en el diseño de la base de datos, ya que permite identificar las entidades clave, definir sus atributos y establecer las relaciones entre ellas. Proporciona una representación visual y estructurada que facilita la comprensión y mejora la organización de los datos, asegurando una base de datos bien estructurada y fácil de mantener. Es por ello que el análisis de los requerimientos y del enunciado nos ha llevado a obtener el siguiente modelo entidad-relación.



V MODELO RELACIONAL

El modelo relacional es importante en el diseño de una base de datos debido a su capacidad para organizar los datos en tablas y establecer relaciones entre ellas. Proporciona integridad de los datos, permite consultas eficientes y ofrece flexibilidad y escalabilidad en la gestión de la

información. Por lo cual el siguiente paso del modelo entidad - relación y como un paso previo de pasar a la programación en una base de datos es el modelo relacional.



5.1 Diccionario de datos

Administrador

EntityType	Dependent, Subtype
Logical Entity Name	Administrador
Default Table Name	Administrador
Logical Only	NO
Owner	
Definition	En esta entiddad se almacenan los atributos de un Empleado de administracion requeridos por una agencia de ecobicis para identificar sus caracteristicas
Note	

Administrador Attributes

Attribute/Logical Rolename	Domain	Datatype	NULL	Definition
id_empleado		NUMERIC(10, 0)	NO	ID de cada empleado que lo identifica da manera unica, comienza en 0 y aumenta de uno en uno
descripcionfuncion		CHAR(100)	NO	Describe la funcion que ejerce el administrador
tipo_trabajo		VARCHAR(20)	NO	Tipo de trabajo desarrollado
ubicacion		VARCHAR(30)	NO	ubicacion

AGENTE

EntityType	Dependent, Subtype
Logical Entity Name	AGENTE
Default Table Name	AGENTE
Logical Only	NO
Owner	
Definition	En esta entiddad se almacenan los atributos de un Empleado agente requeridos por una agencia de ecobicis para identificar sus caracteristicas
Note	

AGENTE Attributes

Attribute/Logical Rolename	Domain	Datatype	NULL	Definition
id_empleado		NUMERIC(10, 0)	NO	ID de cada empleado que lo identifica da manera unica, comienza en 0 y aumenta de uno en uno

BICICLETA

EntityType	Independent
Logical Entity Name	BICICLETA
Default Table Name	BICICLETA
Logical Only	NO
Owner	
Definition	En esta entiddad se almacenan los atributos de una Bicicleta requeridos por una agencia de ecobicis para identificar las distintas caracteristicas de estas.Cada bicicleta cuenta con un estado, un color y un modelo, y a su vez está en una estación.
Note	

BICICLETA Attributes

Attribute/Logical Rolename	Domain	Datatype	NULL	Definition
id_bicicleta		NUMERIC(10, 0)	NO	Id de cada bicicleta, aumenta de forma ascendente de 1 en 1 y enumera a la bicileta como su identificador unico
id_color		NUMERIC(10, 0)	NO	El id del color de cada bicleta
nserie (U)		NUMERIC(15, 0)	NO	Numero de serie unico de cada bicliceta, por lo que no puede haber dos numeros iguales, este numero identifica a la bicicleta de forma unica.
tamaño (CS2, CK)		VARCHAR(10)	NO	Este atributo guarda tres tamaños de bicileta, Chica / Mediana / Grande
estado (CS1, CK)		VARCHAR(15)	NO	Este atributo guarda el estado de cada bicicleta, es importante porque nos ayuda a saber si la bicileta se ira a mantenimiento, se divide en lo siguiente: D: dañada / F:funcionar / B:baja
id_estacion		NUMERIC(10, 0)	NO	ID unico de cada estacion, este clasifica a la estacion de manera unica e irrepetible

BICIMANTENIMIENTO

EntityType	Independent
Logical Entity Name	BICIMANTENIMIENTO
Default Table Name	BICIMANTENIMIENTO
Logical Only	NO
Owner	
Definition	En esta entidad se guardan los registros de las bicicletas que necesitan mantenimiento. Es decir, se guardará la información de las bicicletas involucradas y en los empleados de mantenimiento que realizan los servicios para las bicicletas que les atienden.
Note	

BICIMANTENIMIENTO Attributes

Attribute/Logical Rolename	Domain	Datatype	NULL	Definition
id_mantenimiento		CHAR(10)	NO	Este atributo asigna de manera ascendente y unica los mantenimientos realizados a las biciletas
id_empleado		NUMERIC(10, 0)	NO	El ID de empleado asigna un numero unico e irrepetible a cada empleado, empieza en 0 y aumenta de 1 en 1 cada que se agrega un empleado.
id_bicicleta		NUMERIC(10, 0)	NO	Id de cada bicicleta, aumenta de forma ascendente de 1 en 1 y enumera a la bicileta como su identificador unico
fecha		DATETIME	NO	Se asiga tiempo y hora del mantenimiento
descripcion		CHAR(100)	NO	Se agrega una descripcion detallada sobre el mantenimiento realizado a la bicicleta, es importante que sea detallado para sus proximos mantenimientos

COLOR

EntityType	Independent
Logical Entity Name	COLOR
Default Table Name	COLOR
Logical Only	NO
Owner	
Definition	En esta entiddad se almacenan los atributos de un Catalogo de color requeridos por una agencia de ecobicis para identificar las distintas opciones de color que poseen sus
Note	

COLOR Attributes

Attribute/Logical Rolename	Domain	Datatype	NULL	Definition
id_color		NUMERIC(10, 0)	NO	El id de color diferencia cada bicicleta de manera de catalogo
descripcion (U)		VARCHAR(20)	NO	Aqui se escribe la descripcion del color, puede haber muchos colores por eso se deja abierto.

EMPLEADO

EntityType	Independent, Supertype
Logical Entity Name	EMPLEADO
Default Table Name	EMPLEADO
Logical Only	NO
Owner	
Definition	En esta entiddad se almacenan los atributos de un Empleado requeridos por una agencia de ecobicis para identificar a sus diferentes empleados. Además, dentro de la entidad de Empleado se cuenta con el id del supervisor para hacer una distinción de los empleados ordinarios y los empleados que supervisan a otros empleados.
Note	

EMPLEADO Attributes

Attribute/Logical Rolename	Domain	Datatype	NULL	Definition
id_empleado		NUMERIC(10, 0)	NO	ID de cada empleado que lo identifica de manera unica, comienza en 0 y aumenta de uno en uno
supervisor		NUMERIC(10, 0)	YES	ID de cada empleado que lo identifica de manera unica, comienza en 0 y aumenta de uno en uno
tipo_empleado (CK_CS14)		CHAR(1)	YES	El tipo de empleado al que corresponde, es una letra la cual es identificada de la siguiente manera M: mantenimiento, G: Agente, A:Administrador, R:Recursos Humanos
genero (CS8_CK)		VARCHAR(10)	NO	Genero del empleado: Hombre / Mujer / Otros
nombre		VARCHAR(20)	NO	Este atributo contiene el nombre de cada empleado, maximo 20 caracteres
paterno		VARCHAR(10)	NO	Apellido del empleado maximo 10 caracteres
materno		VARCHAR(10)	NO	Apellido materno del empleado maximo 10 caracteres
calle		VARCHAR(15)	NO	Calle donde vive el empleado
colonia		VARCHAR(15)	NO	Colonia donde vive
alcaldia		VARCHAR(15)	NO	Alcaldia o municipio donde vive
num_ext		INTEGER	NO	Numero exterior de la casa, dado por numeros decimales
num_int		INTEGER	NO	Numero interior
telefono (U)		NUMERIC(10, 0)	NO	Telefono unico e irrepetible para cada emmpleado, maximo 10 digitos
rfc (U)		VARCHAR(13)	NO	RFC unico e irrepetible para cada empleado, este debe contener 13 digitos
sueldo		MONEY(10, 0)	NO	Sueldo que se le paga al empleado
estado_civil (CS13_CK)		VARCHAR(11)	NO	Estatus civil de cada empleado, soltero, casado o viudo
edad		INTEGER	NO	Edad dada por un numero entero del empleado

EMPLEADO_IDIOMA

EntityType	Dependent
Logical Entity Name	EMPLEADO_IDIOMA
Default Table Name	EMPLEADO_IDIOMA
Logical Only	NO
Owner	
Definition	En esta entidad se almacenan los atributos de una relacion empleado_idoma requeridos por una agencia de ecobicis para identificar los empleados que hablan mas de un idioma
Note	

EMPLEADO_IDIOMA Attributes

Attribute/Logical Rolename	Domain	Datatype	NULL	Definition
id_empleado		NUMERIC(10, 0)	NO	ID de cada empleado que lo identifica da manera unica, comienza en 0 y aumenta de uno en uno
id_idioma		NUMERIC(10, 0)	NO	ID del idioma que habla el empleado

EMPMANTENIMIENTO

EntityType	Dependent, Subtype
Logical Entity Name	EMPMANTENIMIENTO
Default Table Name	EMPMANTENIMIENTO
Logical Only	NO
Owner	
Definition	En esta entiddad se almacenan los atributos de un Empleado de mantenimiento requeridos por una agencia de ecobicis para identificar sus caracteristicas
Note	

EMPMANTENIMIENTO Attributes

Attribute/Logical Rolename	Domain	Datatype	NULL	Definition
id_empleado		NUMERIC(10, 0)	NO	ID de cada empleado que lo identifica da manera unica, comienza en 0 y aumenta de uno en uno
especialidad (CS15.CK)		VARCHAR(20)	NO	Tipo de especializacion que tienen los empleados de mantenimiento Reparación / Limpieza / Transporte

ESTACION

EntityType	Independent
Logical Entity Name	ESTACION
Default Table Name	ESTACION
Logical Only	NO
Owner	
Definition	En esta entiddad se almacenan los atributos de una Estacion requeridos por una agencia de ecobicis para identificar las distintas estaciones que tienen
Note	

ESTACION Attributes

Attribute/Logical Rolename	Domain	Datatype	NULL	Definition
id_estacion		NUMERIC(10, 0)	NO	Numero asignado a cada estacion para distinguirla, aumenta de forma ascendente de uno en uno
ubicacion		VARCHAR(40)	NO	Direccion de la estacion
id_empleado		NUMERIC(10, 0)	NO	ID que se le asigna a cada empleado para distinguirlo y consultarlo proximamente

HIATORIAL_FALTA

EntityType	Dependent
Logical Entity Name	HIATORIAL_FALTA
Default Table Name	HIATORIAL_FALTA
Logical Only	NO
Owner	
Definition	En esta entidad se almacenan los atributos de un Hisstorial de faltas requeridos por una agencia de ecobicis para identificar las faltas de sus empleados. Es todo el historial de faltas que pudiera tener cada uno de los empleados de la empresa que trabaja en ECOBICI.
Note	

HIATORIAL_FALTA Attributes

Attribute/Logical Rolename	Domain	Datatype	NULL	Definition
consecutivo (CS16)		COUNTER	NO	Numero consecutivo que aumentara de acuerdo al numero de faltas
id_empleado		NUMERIC(10, 0)	NO	ID de cada empleado que lo identifica da manera unica, comienza en 0 y aumenta de uno en uno
id_motivo		NUMERIC(10, 0)	NO	ID de forma ascendente de uno en uno
fecha		DATE	NO	Fecha de la falta

HISTORIAL_VIAJE

EntityType	Independent
Logical Entity Name	HISTORIAL_VIAJE
Default Table Name	HISTORIAL_VIAJE
Logical Only	NO
Owner	
Definition	En esta entiddad se almacenan los atributos de un Historial de viajes requeridos por una agencia de ecobicis para identificar todos los viajes realizados. Es el histórico para cada viaje, desglosado en términos de las coordenadas de longitud y latitud.
Note	

HISTORIAL_VIAJE Attributes

Attribute/Logical Rolename	Domain	Datatype	NULL	Definition
id_viajeH		NUMERIC(10, 0)	NO	Id de cada historico de viaje, este aumenta de forma ascendente de uno en uno
id_viaje		NUMERIC(10, 0)	NO	ID que se le asigna a cada viaje para identificarlo, este puede comenzar en 0 y tener hasta 10 digitos
duracion (CS17)		DECIMAL(10, 0)	NO	Duración en minutos
latitud		VARCHAR(15)	NO	Numero decimal que guarda su coordenada en latitud
longitud		VARCHAR(15)	NO	Numero decimal que guarda su coordenada en longitud

IDIOMA

EntityType	Independent
Logical Entity Name	IDIOMA
Default Table Name	IDIOMA
Logical Only	NO
Owner	
Definition	En esta entiddad se almacenan los atributos de un Catalogo de idiomas requeridos por una agencia de ecobicis para identificar las distintos idiomas que pueden hablar sus empleados
Note	

IDIOMA Attributes

Attribute/Logical Rolename	Domain	Datatype	NULL	Definition
id_idioma		NUMERIC(10, 0)	NO	Id del idioma, aumenta de forma constante de uno en uno
idioma		VARCHAR(20)	NO	Descripcion del idioma hablado

INCIDENTE

EntityType	Independent
Logical Entity Name	INCIDENTE
Default Table Name	INCIDENTE
Logical Only	NO
Owner	
Definition	En esta entidad se guardan los posibles incidentes que pudiera tener cada viaje. Además, en el incidente también se guarda la información del formulario que tiene que llenar el usuario que tuvo el accidente.
Note	

INCIDENTE Attributes

Attribute/Logical Rolename	Domain	Datatype	NULL	Definition
id_incidente		NUMERIC(10, 0)	NO	ID del incidente que lo relaciona con la tablaa
id_empleado		NUMERIC(10, 0)	NO	Id del empleado que atendio el indicente
hora		TIME/DATETIME	NO	Hora en la que ocurrio el incidente
fecha		DATE	NO	Fecha que ocurrio el incidente
coordenadas		VARCHAR(30)	NO	Coordenada en latitud y longitud que representan la ubicacion exacta del accidente
calle		VARCHAR(20)	NO	calle donde ocurrio el accidente
numero		INTEGER	NO	numero aproximado donde ocurrio el accidente
codigoP		NUMERIC(6, 0)	NO	Codigo postal de donde ocurrio el accidente
alcaldia		VARCHAR(20)	NO	Atributo donde se registra la alcaldia del accidente
colonia		VARCHAR(20)	NO	Colonia donde se registro el accidente
id_viaje		NUMERIC(10, 0)	NO	ID que se le asigna a cada viaje para identificarlo, este puede comenzar en 0 y tener hasta 10 digitos
id_tipo		INTEGER	NO	ID del accidente, este sirve para diferenciar cada accidente, sirve como clave identificadora y aumenta de forma ascendente de uno en uno

MEMBRESIA

EntityType	Independent
Logical Entity Name	MEMBRESIA
Default Table Name	MEMBRESIA
Logical Only	NO
Owner	
Definition	En esta entiddad se almacenan los atributos de una Membresia requeridos por una agencia de ecobicis para identificar las distintas opciones para sus clientes. Son las membresias de cada cliente/usuario concreto
Note	

MEMBRESIA Attributes

Attribute/Logical Rolename	Domain	Datatype	NULL	Definition
id_membresia		NUMERIC(10, 0)	NO	En esta columna se encuentra el ID de la membresia, esta aumenta de forma ascendente de 1 en 1.
fecha		DATE	NO	La fecha de la adquisicion de la membresia, corresponde a un dato tipo "date" por lo cual se debe respetar el formato "YYYY-MM-DD"
id_tipo		NUMERIC(10, 0)	NO	Numero ascendente, comenzando desde 0
beneficio (CS12)		VARCHAR(10)	NO	Si tipoMembresía = básica, descuento al tiempo suscrito Si tipoMembresia = intermedia, viaje gratis cada cierto tiempo Si tipoMembresia = premium, Cashback para el usuario
codigoQR (U)		NUMERIC(20, 0)	NO	El codigo QR que sera generado por la aplicacion a partir del dato numeric de 20, por lo que sera un numero identificador UNIQUE
precio_tarjeta (CS5)		MONEY(10, 0)	NO	En este atributo se marca el costo de la tarjeta, $50 la primera vez / $80 para reposición
duracion_suscripcion (CS18)		TIME/DATETIME	NO	En este atributo se marca la duracion de la membresia, se describen los tres tipos de membresia y de duracion. 'B' entonces duracionDiasSuscripcion = 1 Si tipo = 'I' entonces duracionDiasSuscripcion = 30 Si tipo = 'P' entonces duracionDiasSuscripcion = 365
id_pago		NUMERIC(10, 0)	NO	El id de pago es una clave que comienza en 0 e incrementa de 1 en 1, explica de forma unica el aumento de pago.

METODO_PAGO

EntityType	Independent
Logical Entity Name	METODO_PAGO
Default Table Name	METODO_PAGO
Logical Only	NO
Owner	
Definition	En esta entiddad se almacenan los atributos de un Metodo de pago requeridos por una agencia de ecobicis para identificar las distintas formas de pago que existen para los clientes
Note	

METODO_PAGO Attributes

Attribute/Logical Rolename	Domain	Datatype	NULL	Definition
id_pago		NUMERIC(10, 0)	NO	El id de pago es una clave que comienza en 0 e incrementa de 1 en 1, explica de forma unica el aumento de pago.
activo		BIT	NO	Esta tabla demuestra si el pago esta activo si el resultado es 1 o si el pago esta inactivo con 0.
tipo_pago (CS9, CK)		VARCHAR(30)	NO	Hay tres metodos de pago posibles: Tarjeta de Crédito / Tarjeta de Débito / Paypal
id_usuario		NUMERIC(5, 0)	NO	Este atributo contiene una clave de usuario unica, comienza en el 0 y aumenta sucesivamente de forma ascendente de uno en uno

MODELO

EntityType	Independent
Logical Entity Name	MODELO
Default Table Name	MODELO
Logical Only	NO
Owner	
Definition	Catálogo de modelos de bicicletas
Note	

MODELO Attributes

Attribute/Logical Rolename	Domain	Datatype	NULL	Definition
id_modelo		NUMERIC(10, 0)	NO	El ID del modelo define unicamente cada modelo de bicicletas
modelo		VARCHAR(20)	NO	Aqui se describe el tipo de bicicleta que es

MOTIVO

EntityType	Independent
Logical Entity Name	MOTIVO
Default Table Name	MOTIVO
Logical Only	NO
Owner	
Definition	En esta entiddad se almacenan los atributos de un Catalogo de motivos requeridos por una agencia de ecobicis para identificar las distintas circustancias por la cual faltaron sus empleados a trabajar
Note	

MOTIVO Attributes

Attribute/Logical Rolename	Domain	Datatype	NULL	Definition
id_motivo		NUMERIC(10, 0)	NO	ID de forma ascendente de uno en uno
causa (U)		VARCHAR(30)	NO	Causa por la falta del empleado

PLAN_SEGURO

EntityType	Independent
Logical Entity Name	PLAN_SEGURO
Default Table Name	PLAN_SEGURO
Logical Only	NO
Owner	
Definition	En esta entiddad se almacenan los atributos de un Plan de seguros requeridos por una agencia de ecobicis para identificar los beneficios de cada usuario. Los beneficios dependerán del tipo de membresía.
Note	

PLAN_SEGURO Attributes

Attribute/Logical Rolename	Domain	Datatype	NULL	Definition
id_seguro		NUMERIC(10, 0)	NO	Id de la atencion del seguro, aumenta progesivamente de uno en uno
tipo_cobertura (CS11, CK)		VARCHAR(15)	NO	Tipo de dato que mencion el tipo de cobertura que tiene en caso de un accidente Daños a terceros si tipoMembresia = Básico Daños a terceros y compostura bicicleta si tipoMembresia = Intermedio Full si tipoMembresía = Premium
id_tipo		NUMERIC(10, 0)	NO	Numero ascendente, comenzando desde 0

TELEFONO

EntityType	Independent
Logical Entity Name	TELEFONO
Default Table Name	TELEFONO
Logical Only	NO
Owner	
Definition	En esta entiddad se almacenan los atributos de un Telefono requeridos por una agencia de ecobicis para guardarlos en un catalogo. Son los teléfonos de los usuarios
Note	

TELEFONO Attributes

Attribute/Logical Rolename	Domain	Datatype	NULL	Definition
id_telefono		NUMERIC(10, 0)	NO	Este atributo contine el id, comienza desde el 0 y aumenta sucesivamente de uno en uno.
id_usuario		NUMERIC(5, 0)	NO	Este atributo contiene una clave de usuario unica, comienza en el 0 y aumenta sucesivamente de forma ascendente de uno en uno
telefono (U)		NUMERIC(10, 0)	NO	El telefono guarda un numero unico para cada usuario, el numero es de 10 digitos y se acepta cualquier convinacion, este es el numero telefonico del usuario.

TERMINAL

EntityType	Dependent
Logical Entity Name	TERMINAL
Default Table Name	TERMINAL
Logical Only	NO
Owner	
Definition	En esta entiddad se almacenan los atributos de una Terminal requeridos por una agencia de ecobicis para identificar las distintas terminales que se tienen para cada estación.
Note	

TERMINAL Attributes

Attribute/Logical Rolename	Domain	Datatype	NULL	Definition
consecutivo (CS10)		COUNTER	NO	Consecutivo reiniciado para cada empleado
id_estacion		NUMERIC(10, 0)	NO	Numero asignado a cada estacion para distinguirla, aumenta de forma ascendente de uno en uno
descripcion		VARCHAR(25)	NO	Descripcion de cada terminal, en cada estacion puede haber una o multiples terminales de control

TIPO_INCIDENTE

EntityType	Independent
Logical Entity Name	TIPO_INCIDENTE
Default Table Name	TIPO_INCIDENTE
Logical Only	NO
Owner	
Definition	Catálogo de los tipos de incidentes, que sirve para identificar el tipo de incidente que pudiera haber en los registros de la tabla incidente, cuando se da un accidente para un viaje determinado por un usuario, una bicicleta, y una estación de partida y llegada.
Note	

TIPO_INCIDENTE Attributes

Attribute/Logical Rolename	Domain	Datatype	NULL	Definition
id_tipo		INTEGER	NO	ID del accidente, este sirve para diferenciar cada accidente, sirve como clave identificadora y aumenta de forma ascendente de uno en uno
tipo (CK, CS10)		VARCHAR(18)	NO	Este atributo guarda si hubo algun Problemas con: Bicicleta/ Coche/ Moto/ Coche con otra bici/ Peatón/ Caída de bicicleta/ Otros
descripcion		VARCHAR(30)	NO	Se describe el tipo del problema que hubo

TIPO_MEMBRESIA

EntityType	Independent
Logical Entity Name	TIPO_MEMBRESIA
Default Table Name	TIPO_MEMBRESIA
Logical Only	NO
Owner	
Definition	En esta entidad se almacenan los atributos de la tabl Tipo_Membresia. Se tiene el catálogo a partir del cual se tendrán todos los tipos de Membresia para cada usuario particular
Note	

TIPO_MEMBRESIA Attributes

Attribute/Logical Rolename	Domain	Datatype	NULL	Definition
id_tipo		NUMERIC(10, 0)	NO	Numero ascendente, comenzando desde 0
tipo (CS6, CK, U)		VARCHAR(15)	NO	Aqui podemos encontrar los tipos de membresia, son tres tipos y solo son aceptadas las siguientes: B:Básica / I:Intermedia / P:Premium
costo (CS7)		MONEY(10, 0)	NO	Aqui tenemos el costo de la membresia, los tipos de membresia se muestran a continuacion junto a su costo. Si tipo = 'B' entonces costo = $118 por día Si tipo = 'I' etnonces costo = $400 semanalmente Si tipo = 'P' entonces costo = $1000 anualmente

USUARIO

EntityType	Independent
Logical Entity Name	USUARIO
Default Table Name	USUARIO
Logical Only	NO
Owner	
Definition	En esta entidad se almacenan los atributos de un Usuario requeridos por una agencia de ecobicis para identificar a sus clientes
Note	

USUARIO Attributes

Attribute/Logical Rolename	Domain	Datatype	NULL	Definition
id_usuario		NUMERIC(5, 0)	NO	Este atributo contiene una clave de usuario unica, comienza en el 0 y aumenta sucesivamente de forma ascendente de uno en uno
app_saldo		BIT	YES	Esta columna almacena si el usuario cuenta con la aplicacion para pagar en la estacion, si es 1 es un "si" y si es un 0 es un "no"
materno		VARCHAR(10)	NO	Contiene el apellido materno del usuario
nombre		VARCHAR(15)	NO	Contiene el nombre del usuario, se permite hasta un maximo de 15 caracteres.
paterno		VARCHAR(10)	NO	Contiene el apellido paterno del usuario, se permite un maximo de 10 caracteres
fecha_nac		DATE	NO	Este atributo contiene la fecha de nacimiento, es un atributo tipo date, por lo cual la fecha debe seguir el siguiente formato: aaaa-MM-dd
correo (U)		VARCHAR(50)	NO	Este atributo corresponde al correo, debe ser unico, por lo cual no se permite el mismo correo para dos usuarios distintos, debe seguir la estructura de un correo con nombredelcorreo@dominio.com
edad (C,CS3)		INTEGER	NO	En esta columna es calculada, se agrega con la siguiente formula (FechaActual - FechaNacimiento) % 365
codigo_ine (U)		NUMERIC(13, 0)	NO	El número identificador de la credencial para votar o también conocido como OCR, se halla al reverso del INE. En los modelos más recientes de la credencial, se encuentran en la primera línea de la parte inferior después de los símbolos "<<". El número ident
genero (CK,CS4)		VARCHAR(10)	NO	Aqui se lee el genero, Hombre / Mujer / Otros

VIAJE

EntityType	Independent
Logical Entity Name	VIAJE
Default Table Name	VIAJE
Logical Only	NO
Owner	
Definition	En esta entidad se almacenan los atributos de un Viaje requeridos por una agencia de ecobicis para identificar los distintos viajes que se realizaron. Cada viaje está conformado por un usuario y una bicicleta, que parte de una estación y llega a otra.
Note	

VIAJE Attributes

Attribute/Logical Rolename	Domain	Datatype	NULL	Definition
id_viaje		NUMERIC(10, 0)	NO	ID que se le asigna a cada viaje para identificarlo, este puede comenzar en 0 y tener hasta 10 dígitos
id_bicicleta		NUMERIC(10, 0)	NO	Id de cada bicicleta, aumenta de forma ascendente de 1 en 1 y enumera a la bicileta como su identificador unico
id_usuario		NUMERIC(5, 0)	NO	Este atributo contiene una clave de usuario unica, comienza en el 0 y aumenta sucesivamente de forma ascendente de uno en uno
hora_fin		TIME/DATETIME	NO	Es la hora en la que finaliza el viaje, se guarda como un datip time/datatime, donde guarda la hora y fecha exacta,
hora_ini		TIME/DATETIME	NO	Esta columna asigna el inicio del viaje y la hora exacta, es un time / datetime
hora_llegada		TIME/DATETIME	NO	Hora en la que se llega a la estacion
fecha		DATE	NO	Fecha en la cual se toma el viaje
ruta		VARCHAR(40)	NO	Ruta la cual tomo el usuario del viaje
tarifa (C, CS8)		DECIMALN(5, 0)	NO	La ruta es una columna calculada de la siguiente manera Si hora_llegada < hora_fin entonces Tarifa = 0 Si no : Tarifa = 5 x, donde x = hora_llegada - hora_fin %10 Abonada el mismo día para Membresia básica Abonada el siguiente mes para Membresía intermedia Abonada el siguiente año para Membresía anual
estacionPartida		NUMERIC(10, 0)	NO	Numero asignado a cada estacion para distinguirla, aumenta de forma ascendente de uno en uno
estacionLlegada		NUMERIC(10, 0)	NO	Numero asignado a cada estacion para distinguirla, aumenta de forma ascendente de uno en uno

VI NORMALIZACIÓN

La normalización es importante en el diseño de bases de datos porque elimina la redundancia de datos, mejora la integridad y facilita las modificaciones. Además, optimiza el rendimiento de las consultas y permite la escalabilidad de la base de datos. Proporciona una estructura eficiente y confiable para gestionar datos de manera coherente.

1. Primera Forma Normal (1NF): En 1NF, cada columna de una tabla contiene un único valor atómico, evitando la duplicación y la repetición de datos. Esto elimina la posibilidad de tener múltiples valores en una sola celda, lo que garantiza la integridad y facilita la manipulación de datos.
2. Segunda Forma Normal (2NF): En 2NF, se eliminan las dependencias parciales al dividir la tabla en múltiples tablas más pequeñas. Cada tabla debe tener una clave primaria única y las columnas no clave deben depender completamente de la clave primaria. Esto ayuda a evitar la redundancia de datos y mejora la integridad.
3. Tercera Forma Normal (3NF): En 3NF, se eliminan las dependencias transitivas. Esto significa que las columnas no clave deben depender únicamente de la clave primaria y no de otras columnas no clave. Al eliminar estas dependencias, se evita la redundancia y se asegura la consistencia de los datos.

Por lo que a continuación mostraremos el proceso de normalización de nuestro modelo relacional:

Tabla sin normalizar:

1. **USUARIO**: id_usuario (clave primaria), genero, codigo_ine, edad, correo, fecha_nac, nombre, paterno, materno, app_saldo, activo, telefonos (un array con todos los teléfonos), metodos_pago (un array con todos los métodos de pago).

2. **BICICLETA**: id_bicicleta (clave primaria), nserie, tamaño, estado, id_modelo, modelo, id_color, descripcion_color.

3. **VIAJE**: id_viaje (clave primaria), hora_fin, hora_ini, hora_llegada, fecha, ruta, tarifa, id_estacion, id_usuario, usuario_info (incluyendo todos los datos de usuario), id_bicicleta, bicicleta_info (incluyendo todos los datos de bicicleta), historial_viaje (un array con todo el historial de viaje).

4. **INCIDENTE**: id_incidente (clave primaria), id_viaje, viaje_info (incluyendo todos los datos de viaje), hora_incidente, fecha_incidente, coordenadas, calle, numero, codigoP, alcaldia, colonia, id_empleado_incidente, id_tipo_incidente, descripcion_incidente.

5. **MEMBRESIA**: id_membresia (clave primaria), beneficio, precio_tarjeta, codigoQR, duracion_suscripcion, id_tipo, id_pago, metodos_pago (un array con todos los métodos de pago), fecha, id_seguro, seguro_info (incluyendo todos los datos de seguro).

Primera Forma Normal (1NF): Separaremos los arrays y los datos incrustados en tablas separadas:

1. **Usuario**: id_usuario (clave primaria), genero, codigo_ine, edad, correo, fecha_nac, nombre, paterno, materno, app_saldo, activo.

2. **TELEFONO**: id_telefono (clave primaria), id_usuario (clave foránea), telefono.

3. **METODO_PAGO**: id_pago (clave primaria), id_usuario (clave foránea), tipo_pago.

4. **BICICLETA**: id_bicicleta (clave primaria), nserie, tamaño, estado, id_modelo, id_color.

5. **MODELO**: id_modelo (clave primaria), modelo.

6. **COLOR**: id_color (clave primaria), descripcion_color.

7. **VIAJE**: id_viaje (clave primaria), hora_fin, hora_ini, hora_llegada, fecha, ruta, tarifa, id_estacion, id_usuario (clave foránea), id_bicicleta (clave foránea).

8. **HISTORIAL_VIAJE**: id_viajeH (clave primaria), id_viaje (clave foránea), duracion, latitud, longitud.

9. **INCIDENTE**: id_incidente (clave primaria), id_viaje (clave foránea), hora_incidente, fecha_incidente, coordenadas, calle, numero, codigoP, alcaldia, colonia, id_empleado_incidente, id_tipo_incidente.

10. **TIPO_INCIDENTE**: id_tipo_incidente (clave primaria), descripcion_incidente.

11. **MEMBRESIA**: id_membresia (clave primaria), beneficio, precio_tarjeta, codigoQR, duracion_suscripcion, id_tipo, id_pago (clave foránea), fecha, id_seguro.

12. **PLAN_SEGURO**: id_seguro (clave primaria), id_membresia (clave foránea), tipo_cobertura.

Segunda Forma Normal (2NF) y **Tercera Forma Normal** (3NF): Todas las tablas cumplen ya la segunda y la tercera forma normal, porque todas las columnas no clave dependen completamente de la clave primaria y no existen dependencias transitivas entre las columnas no clave.

VII DISEÑO FÍSICO

7.1 SCRIPT seguridad.sql (DCL)

```
-- Crear el usuario de solo lectura
CREATE LOGIN usuarioConsulta WITH PASSWORD = '1234zaq*';
CREATE USER usuarioConsulta FOR LOGIN usuarioConsulta;
GO

-- Otorgar permisos de solo lectura
ALTER ROLE db_datareader ADD MEMBER usuarioConsulta;
GO
-- Crear el usuario gestor
CREATE LOGIN usuarioGestor WITH PASSWORD = '1234zaq*';
CREATE USER usuarioGestor FOR LOGIN usuarioGestor;
GO

-- Otorgar permisos para agregar y actualizar información
GRANT INSERT, UPDATE TO usuarioGestor;
GO
-- Crear el usuario administrador
CREATE LOGIN usuarioAdministrador WITH PASSWORD = '1234zaq*';
CREATE USER usuarioAdministrador FOR LOGIN usuarioAdministrador;
GO

-- Otorgar permisos de administrador
ALTER ROLE db_owner ADD MEMBER usuarioAdministrador;
GO



------------------------------------------------------------------------
-------------------PROCEDIMIENTO PARA SOLO LECTURA ---------------------
------------------------------------------------------------------------

CREATE PROCEDURE CrearUsuarioConsulta
    @nombreUsuario NVARCHAR(50),
    @contrasena NVARCHAR(50)
AS
BEGIN
-- Declaramos la consulta SQL como una variable NVARCHAR para poder utilizar parámetros dinámicos
    DECLARE @SQL NVARCHAR(500);

-- Creamos el inicio de sesión con los detalles proporcionados
    SET @SQL = N'CREATE LOGIN ' + QUOTENAME(@nombreUsuario) + ' WITH PASSWORD = N' +
QUOTENAME(@contrasena, '''') + ';'
    EXEC sp_executesql @SQL;

-- Creamos el usuario para el inicio de sesión creado
    SET @SQL = N'CREATE USER ' + QUOTENAME(@nombreUsuario) + ' FOR LOGIN ' +
QUOTENAME(@nombreUsuario) + ';'
    EXEC sp_executesql @SQL;

    -- Otorgamos los permisos de solo lectura
    SET @SQL = N'ALTER ROLE db_datareader ADD MEMBER ' + QUOTENAME(@nombreUsuario) + ';'
```

```
    EXEC sp_executesql @SQL;

    PRINT 'Usuario ' + @nombreUsuario + ' creado exitosamente.'
END;
GO
EXEC CrearUsuarioConsulta @nombreUsuario = 'MiNuevoUsuario', @contrasena = 'MiNuevaContrasena';


-----------------------------------------------------------------------
------------------ PROCEDIMIENTO PARA USUARIO GESTOR ------------------
-----------------------------------------------------------------------


CREATE PROCEDURE CrearUsuarioGestor
    @nombreUsuario NVARCHAR(50),
    @contrasena NVARCHAR(50)
AS
BEGIN
-- Declaramos la consulta SQL como una variable NVARCHAR para poder utilizar parámetros dinámicos
    DECLARE @SQL NVARCHAR(500);

    -- Creamos el inicio de sesión con los detalles proporcionados
    SET @SQL = N'CREATE LOGIN ' + QUOTENAME(@nombreUsuario) + ' WITH PASSWORD = N' +
QUOTENAME(@contrasena, '''') + ';'
    EXEC sp_executesql @SQL;

    -- Creamos el usuario para el inicio de sesión creado
    SET @SQL = N'CREATE USER ' + QUOTENAME(@nombreUsuario) + ' FOR LOGIN ' +
QUOTENAME(@nombreUsuario) + ';'
    EXEC sp_executesql @SQL;

    -- Otorgamos los permisos para agregar y actualizar información
    SET @SQL = N'GRANT INSERT, UPDATE TO ' + QUOTENAME(@nombreUsuario) + ';'
    EXEC sp_executesql @SQL;

    PRINT 'Usuario ' + @nombreUsuario + ' creado exitosamente.'
END;
GO

EXEC CrearUsuarioGestor @nombreUsuario = 'MiNuevoGestor', @contrasena = 'MiNuevaContrasena';

-----------------------------------------------------------------------
------------ PROCEDIMIENTO PARA USUARIO ADMINISTRADOR ----------------
-----------------------------------------------------------------------
CREATE PROCEDURE CrearUsuarioAdministrador
    @nombreUsuario NVARCHAR(50),
    @contrasena NVARCHAR(50)
AS
BEGIN
-- Declaramos la consulta SQL como una variable NVARCHAR para poder utilizar parámetros dinámicos
    DECLARE @SQL NVARCHAR(500);

-- Creamos el inicio de sesión con los detalles proporcionados
    SET @SQL = N'CREATE LOGIN ' + QUOTENAME(@nombreUsuario) + ' WITH PASSWORD = N' +
QUOTENAME(@contrasena, '''') + ';'
    EXEC sp_executesql @SQL;
```

```
    -- Creamos el usuario para el inicio de sesión creado
    SET @SQL = N'CREATE USER ' + QUOTENAME(@nombreUsuario) + ' FOR LOGIN ' +
QUOTENAME(@nombreUsuario) + ';'
    EXEC sp_executesql @SQL;

    -- Otorgamos los permisos de administrador
    SET @SQL = N'ALTER ROLE db_owner ADD MEMBER ' + QUOTENAME(@nombreUsuario) + ';'
    EXEC sp_executesql @SQL;

    PRINT 'Usuario ' + @nombreUsuario + ' creado exitosamente.'
END;
GO

EXEC CrearUsuarioAdministrador @nombreUsuario
```

7.2SCRIPT creaBase.sql (DDL)

```
--CREACIÓN DE LA BASE DE DATOS
CREATE DATABASE ecobici
-------------------------------------------------------------------------------
/*
 * TABLE: USUARIO
 */

CREATE TABLE USUARIO(
    id_usuario         int                   NOT NULL IDENTITY(1,1),
    app_saldo          bit              NOT NULL,
    materno            varchar(10)      NOT NULL,
    nombre             varchar(15)      NOT NULL,
    paterno            varchar(10)      NOT NULL,
    fecha_nac          date             NOT NULL,
    correo             varchar(50)      NOT NULL,
    codigo_ine         varchar(9)       NOT NULL,
    genero             char(1)               NOT NULL constraint ck_Genero check
(genero in ('H','M','O')),
    CONSTRAINT PK1 PRIMARY KEY NONCLUSTERED (id_usuario)
)
go

ALTER TABLE USUARIO
ADD edad AS DATEDIFF(YEAR, fecha_nac, GETDATE())

ALTER TABLE USUARIO
ADD CONSTRAINT uq_correo UNIQUE(correo)

ALTER TABLE USUARIO
ADD CONSTRAINT uq_codigo_ine UNIQUE(codigo_ine)

ALTER TABLE USUARIO
```

```
ADD CONSTRAINT ck_codigo_ine CHECK (codigo_ine LIKE
'[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]')

-------------------------------------------------------------------------------
/*
 *TABLE: telefono
 */
CREATE TABLE telefono(
    id_telefono      int               NOT NULL IDENTITY(1,1),
    id_usuario       int               NOT NULL,
    tel              varchar(10)    NOT NULL,

CONSTRAINT PK2 PRIMARY KEY CLUSTERED (id_telefono),
CONSTRAINT fk_id_usuario FOREIGN KEY (id_usuario) REFERENCES USUARIO(id_usuario)
             ON DELETE CASCADE
             ON UPDATE CASCADE
)
go
ALTER TABLE telefono
ADD CONSTRAINT uq_telefono UNIQUE(tel)

ALTER TABLE telefono
ADD CONSTRAINT ck_telefono_usuario CHECK (tel LIKE
'[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]')

-------------------------------------------------------------------------------
/*
 * TABLE: metodo_pago
 */

CREATE TABLE metodo_pago(
    id_pago       int            NOT NULL IDENTITY(1,1),
    activo        bit            NOT NULL,
    tipo_pago     varchar(30)    NOT NULL constraint ck_tipoPago check (tipo_pago in
('CREDITO','DEBITO','PAYPAL')),
    id_usuario    int            NOT NULL,

CONSTRAINT PK3 PRIMARY KEY CLUSTERED (id_pago),
CONSTRAINT fk_id_usuario_metodoPago FOREIGN KEY (id_usuario) REFERENCES USUARIO(id_usuario)
             ON DELETE CASCADE
             ON UPDATE CASCADE
)
go

CREATE NONCLUSTERED INDEX idx_id_usuario
ON metodo_pago (id_usuario);

-------------------------------------------------------------------------------
/*
 * TABLE: tipo_membresia
 */
```

```
CREATE TABLE tipo_membresia(
    id_tipo       int           NOT NULL IDENTITY(1,1),
    tipo          char(1)       NOT NULL CONSTRAINT ck_tipo CHECK (tipo in
('B','I','P')), -- BÁSICA, INTERMEDIA, PREMIUM
    costo          int          NULL,
      CONSTRAINT PK4 PRIMARY KEY CLUSTERED (id_tipo)
)
go

ALTER TABLE tipo_membresia
ADD CONSTRAINT uq_tipo UNIQUE(tipo)




----------------------------------------------------------------------------/*
 * TABLE: membresia
 */

CREATE TABLE membresia(
    id_membresia             int                   NOT NULL IDENTITY(1,1),
    fecha                    date                  NOT NULL,
    id_tipo                  int                   NOT NULL,
    precio_tarjeta           money                 NOT NULL CONSTRAINT
CHK_ValoresPermitidos CHECK (precio_tarjeta IN (50.00, 80.00)),
    codigoQR                 numeric(20, 5)        NOT NULL,
    id_pago                  int                   NOT NULL,

 CONSTRAINT PK5 PRIMARY KEY CLUSTERED (id_membresia),
 CONSTRAINT fk_id_tipoMembresia FOREIGN KEY (id_tipo) REFERENCES
tipo_membresia(id_tipo)
             ON DELETE CASCADE
             ON UPDATE CASCADE,
       CONSTRAINT fk_id_pago FOREIGN KEY (id_pago) REFERENCES metodo_pago(id_pago)
             ON DELETE CASCADE
             ON UPDATE CASCADE,
)
go

ALTER TABLE membresia
ADD CONSTRAINT uq_codigoQR UNIQUE(codigoQR)

ALTER TABLE membresia --Relación uno a uno, conservando unicidad
ADD CONSTRAINT uq_id_pago UNIQUE(id_pago)

ALTER TABLE membresia
ADD beneficio AS (
             CASE
            WHEN id_tipo = 1 THEN 'Descuento'
```

```
            WHEN id_tipo = 2 THEN 'Viaje'
            WHEN id_tipo = 3 THEN 'Cashback'
        END
)

ALTER TABLE membresia
ADD duracionDiasSuscripcion AS (
        CASE
            WHEN id_tipo = 1 THEN 1
            WHEN id_tipo = 2 THEN 30
            WHEN id_tipo = 3 THEN 365
        END
)


--------------------------------------------------------------------------------/*
 * TABLE: EMPLEADO
 */
CREATE TABLE empleado(
    id_empleado       int               NOT NULL IDENTITY(1,1),
    id_supervisor     int               NULL,
-- M: Mantenimiento, G: Agente, A: Administrador, R: Recursos humanos
    tipo_empleado     char(1)           NOT NULL CONSTRAINT ck_tipoEmpleado CHECK
(tipo_empleado in ('M','G','A','R')),
    genero            char(1)           NOT NULL constraint ck_GeneroEmpleado check
(genero in ('H','M','O')),
    nombre            varchar(20)       NOT NULL,
    paterno           varchar(10)       NOT NULL,
    materno           varchar(10)       NOT NULL,
    calle             varchar(15)       NOT NULL,
    colonia           varchar(15)       NOT NULL,
    alcaldia          varchar(15)       NOT NULL,
    num_ext           int               NOT NULL,
    num_int           int               NOT NULL,
    telefono          varchar(10)       NOT NULL CONSTRAINT uq_telefonoEmp UNIQUE,
    rfc               varchar(13)       NOT NULL CONSTRAINT uq_rfc UNIQUE,
    sueldo            money             NOT NULL,
    estado_civil      varchar(11)       NOT NULL constraint ck_estadoCivil check
(estado_civil in ('SOLTERO','CASADO','VIUDO','DIVORCIADO')),
    edad              int               NOT NULL,

CONSTRAINT PK6 PRIMARY KEY NONCLUSTERED (id_empleado),
CONSTRAINT fk_Supervisor_Empleado FOREIGN KEY (id_supervisor) REFERENCES
empleado(id_empleado)
)
go

ALTER TABLE empleado
ADD CONSTRAINT ck_telefono_empleado CHECK (telefono LIKE
'[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]')
```

```
-------------------------------------------------------------------------------

/*
 * TABLE: idioma
*/
CREATE TABLE idioma(
    id_idioma       int                 NOT NULL IDENTITY(1,1),
    idioma          varchar(20)         NOT NULL,
    CONSTRAINT PK7 PRIMARY KEY CLUSTERED (id_idioma)
)
go

ALTER TABLE idioma
ADD CONSTRAINT uq_idioma UNIQUE(idioma)
-------------------------------------------------------------------------------

/*
 * TABLE: EMPLEADO_IDIOMA
 */

CREATE TABLE EMPLEADO_IDIOMA(
    id_empleado    int    NOT NULL,
    id_idioma      int    NOT NULL,

       CONSTRAINT PK8 PRIMARY KEY CLUSTERED (id_empleado, id_idioma),
       CONSTRAINT fk_idempleado FOREIGN KEY (id_empleado) REFERENCES
empleado(id_empleado)
             ON DELETE NO ACTION
             ON UPDATE NO ACTION,
      CONSTRAINT fk_idioma FOREIGN KEY (id_idioma) REFERENCES idioma(id_idioma)
             ON DELETE NO ACTION
             ON UPDATE NO ACTION,
)
go
-------------------------------------------------------------------------------
/*
 * TABLE: motivo
*/

CREATE TABLE motivo(
    id_motivo       int              NOT NULL IDENTITY(1,1),
    causa           varchar(30)      NOT NULL CONSTRAINT uq_causa UNIQUE,
    CONSTRAINT PK9 PRIMARY KEY CLUSTERED (id_motivo)
)
go
-------------------------------------------------------------------------------
/*
 * TABLE: historial_falta
 */
CREATE TABLE historial_falta(
```

```
    consecutivo    int               IDENTITY(1,1),
  --Para ver el número consecutivo por cada empleado se usará la función  ROW_NUMBER()
Al usar el select
    id_empleado    int               NOT NULL,
    id_motivo      int               NOT NULL,
    fecha          date              NOT NULL,
CONSTRAINT PK10 PRIMARY KEY NONCLUSTERED (consecutivo, id_empleado),
CONSTRAINT fk_idempleadoFalta FOREIGN KEY (id_empleado) REFERENCES
empleado(id_empleado)
             ON DELETE CASCADE
             ON UPDATE CASCADE,
       CONSTRAINT fk_motivoFalta FOREIGN KEY (id_motivo) REFERENCES motivo(id_motivo)
             ON DELETE CASCADE
             ON UPDATE CASCADE, ) go
-------------------------------------------------------------------------------/*
 * TABLE: agente
 */

CREATE TABLE agente(
    id_empleado      int   NOT NULL,
 CONSTRAINT PK11 PRIMARY KEY CLUSTERED (id_empleado),
 CONSTRAINT fk_empleadoAgente FOREIGN KEY (id_empleado) REFERENCES
empleado(id_empleado)
             ON DELETE CASCADE
             ON UPDATE CASCADE,
)
go
-------------------------------------------------------------------------------
/*
 * TABLE: administrador
 */

CREATE TABLE administrador(
    id_empleado          int               NOT NULL,
    descripcionfuncion    char(100)         NOT NULL,
    tipo_trabajo          varchar(20)       NOT NULL,
    ubicacion             varchar(40)       NOT NULL,

 CONSTRAINT PK12 PRIMARY KEY CLUSTERED (id_empleado),
CONSTRAINT fk_empleadoAdmin FOREIGN KEY (id_empleado) REFERENCES empleado(id_empleado)
             ON DELETE CASCADE
             ON UPDATE CASCADE,
)
go
-------------------------------------------------------------------------------
/*
 * TABLE: mantenimiento
*/

CREATE TABLE mantenimiento(
       id_empleado    int   NOT NULL,
```

```
       especialidad    varchar(20)       NOT NULL,
 CONSTRAINT PK13 PRIMARY KEY CLUSTERED (id_empleado),
 CONSTRAINT fk_empleadoMant FOREIGN KEY (id_empleado) REFERENCES empleado(id_empleado)
             ON DELETE CASCADE
             ON UPDATE CASCADE,
)
go



-------------------------------------------------------------------------------/*
 * TABLE: plan_seguro
 */

CREATE TABLE plan_seguro(
    id_seguro                    int    NOT NULL identity(1,1),
    tipo_cobertura               varchar(50),
    id_tipo                      int    NOT NULL,
CONSTRAINT PK14 PRIMARY KEY CLUSTERED (id_seguro),
CONSTRAINT fk_tipoMem FOREIGN KEY (id_tipo) REFERENCES tipo_membresia(id_tipo)
             ON DELETE CASCADE
             ON UPDATE CASCADE,
)
go

-------------------------------------------------------------------------------
/*
 * TABLE: color
*/

CREATE TABLE color(
    id_color        int                       NOT NULL identity(1,1),
    color           varchar(20)               NOT NULL CONSTRAINT uq_color UNIQUE,
    CONSTRAINT PK15       PRIMARY KEY CLUSTERED (id_color)
)
go

-------------------------------------------------------------------------------/*
 * TABLE: estacion
 */

CREATE TABLE estacion(
    id_estacion    int           NOT NULL IDENTITY(1,1),
    nombre         varchar(45)   NOT NULL CONSTRAINT uq_nombre UNIQUE,
    ubicacion      varchar(45)   NOT NULL,
    id_empleado    int           NOT NULL,
CONSTRAINT PK16 PRIMARY KEY CLUSTERED (id_estacion),
CONSTRAINT fk_empleadoAdminEstacion FOREIGN KEY (id_empleado) REFERENCES
administrador(id_empleado)
)
go
```

```
----------------------------------------------------------------------------------/*
 * TABLE: terminal
 */
CREATE TABLE terminal(
    consecutivo           int          NOT NULL,
    id_estacion           int          NOT NULL,
    descripcion           varchar(25)  NOT NULL,
CONSTRAINT PK17 PRIMARY KEY CLUSTERED (consecutivo, id_estacion),
CONSTRAINT fk_id_estacion FOREIGN KEY (id_estacion) REFERENCES estacion(id_estacion)
             ON DELETE CASCADE
             ON UPDATE CASCADE,
)
go

-------------------------------------------------------------------------------------
---------------------------------

/*
 * TABLE: modelo
 */

CREATE TABLE modelo (
       id_modelo      int          NOT NULL primary key,
       modelo         varchar(20)  NOT NULL Constraint uq_mod UNIQUE
)
----------------------------------------------------------------------------------/*
 * TABLE: bicicleta
 */
CREATE TABLE bicicleta(
    id_bicicleta         int           NOT NULL IDENTITY(1,1),
    id_color             int           NOT NULL,
    id_modelo            int           NOT NULL,
    nserie               varchar(15)   NOT NULL CONSTRAINT uq_nserie UNIQUE,
    tamaño               varchar(10)   NOT NULL CONSTRAINT ck_tamaño check (tamaño in
('Chica','Mediana','Grande')),
    estado               varchar(15)   NOT NULL CONSTRAINT ck_estado check (estado in
('D','F','B')), --D: DAÑADA, F:FUNCIONAL, B: BAJA
    id_estacion          int           NOT NULL,
CONSTRAINT PK18 PRIMARY KEY NONCLUSTERED (id_bicicleta),
CONSTRAINT fk_color FOREIGN KEY (id_color) REFERENCES color(id_color)
             ON UPDATE CASCADE,
CONSTRAINT fk_modelo FOREIGN KEY (id_modelo) REFERENCES modelo(id_modelo),
CONSTRAINT fk_NumEstacion FOREIGN KEY (id_estacion) REFERENCES estacion(id_estacion)
             ON DELETE CASCADE
             ON UPDATE CASCADE
)
go
-------------------------------------------------------------------------------
/*
 * TABLE: bici_mantenimiento
 */
```

```
CREATE TABLE bicimantenimiento(
    id_mantenimiento    int              NOT NULL identity(1,1),
    id_empleado         int              NOT NULL,
    id_bicicleta        int              NOT NULL,
    fecha               date             NOT NULL,
    descripcion         varchar(100)     NULL,
       servicio            varchar(20)   NOT NULL CONSTRAINT ck_servicio CHECK (servicio
in('REPARACIÓN','LIMPIEZA','TRANSPORTE'))
CONSTRAINT PK19 PRIMARY KEY CLUSTERED (id_mantenimiento),
CONSTRAINT fk_empleadoMantenimiento FOREIGN KEY (id_empleado) REFERENCES
mantenimiento(id_empleado),
CONSTRAINT fk_bicicletaMantenimiento FOREIGN KEY (id_bicicleta) REFERENCES
bicicleta(id_bicicleta)
)
go
-------------------------------------------------------------------------------
/*
 * TABLE: viaje
 */
CREATE TABLE viaje(
    id_viaje            int            NOT NULL IDENTITY(1,1),
    id_bicicleta        int            NOT NULL,
    id_usuario          int            NOT NULL,
    hora_fin            time           NOT NULL,
    hora_ini            time           NOT NULL,
    hora_llegada        time           NOT NULL,
    fecha               date           NOT NULL,
    ruta                varchar(100)   NOT NULL,
    estacionPartida     int            NOT NULL,
    estacionLlegada     int            NOT NULL,
CONSTRAINT PK21 PRIMARY KEY NONCLUSTERED (id_viaje),
CONSTRAINT fk_id_bicicletaViaje FOREIGN KEY (id_bicicleta) REFERENCES
bicicleta(id_bicicleta),
CONSTRAINT fk_id_usuarioViaje FOREIGN KEY (id_usuario) REFERENCES usuario(id_usuario)
             ON DELETE CASCADE
             ON UPDATE CASCADE,
CONSTRAINT fk_estacionPartida FOREIGN KEY (estacionPartida) REFERENCES
estacion(id_estacion),
CONSTRAINT fk_estacionLlegada FOREIGN KEY (estacionLlegada) REFERENCES
estacion(id_estacion)
)
go




ALTER TABLE viaje
ADD tarifa AS (
             CASE
                WHEN hora_llegada < hora_fin THEN 0
```

```
            ELSE
                5 * FLOOR(  DATEDIFF(MINUTE, hora_fin, hora_llegada) /10   )
        END
)

-------------------------------------------------------------------------------
/*
 * TABLE: incidente
 */
CREATE TABLE incidente(
    id_incidente       int            NOT NULL IDENTITY(1,1),
    id_empleado        int            NOT NULL,
    tipo_incidente     varchar(50)    NOT NULL CONSTRAINT ck_tipoInc
CHECK(tipo_incidente in('P. BICI','P. COCHE', 'P. MOTO', 'P. Peatón', 'Caída',
'Otros')),
    hora               time           NOT NULL,
    fecha              date           NULL,
 --IMPLEMENTAR TRIGGER PARA ACTUALIZAR FECHAS
    coordenadas        varchar(30)    NOT NULL,
    calle              varchar(20)    NOT NULL,
    numero             int            NOT NULL,
    codigoPostal       char(5)        NOT NULL CONSTRAINT CK_CodigoPostal CHECK
(LEN(codigoPostal) = 5),
    alcaldia           varchar(20)    NOT NULL,
    colonia            varchar(20)    NOT NULL,
    descripcion        varchar(50)    NOT NULL,
       id_viaje              int      NOT NULL UNIQUE,
CONSTRAINT PK20 PRIMARY KEY CLUSTERED (id_incidente),
CONSTRAINT fk_empleadoAgenteIncidente FOREIGN KEY (id_empleado) REFERENCES
agente(id_empleado)
             ON UPDATE CASCADE,
CONSTRAINT fk_idViajeIncidente FOREIGN KEY (id_viaje) REFERENCES viaje(id_viaje)
)
go

ALTER TABLE incidente
ADD CONSTRAINT df_descripcion DEFAULT 'Incidente' FOR descripcion




-------------------------------------------------------------------------------
/*
 * TABLE: HISTORIAL_VIAJE
 */
```

```
CREATE TABLE historial_viaje(
    id_viajeH          int              NOT NULL IDENTITY(1,1),
    id_viaje           int              NOT NULL,
       latitud         decimal(12,4)    NOT NULL,
    longitud           decimal(12,4)     NOT NULL,
CONSTRAINT PK22 PRIMARY KEY CLUSTERED (id_viajeH),
CONSTRAINT fk_idViajeHistorico FOREIGN KEY (id_viaje) REFERENCES viaje(id_viaje)
             ON DELETE CASCADE
             ON UPDATE CASCADE,
)
go
-------------------------------------------------------------------------------
--Correr esquema por separado
go
CREATE SCHEMA usuarios AUTHORIZATION DBO
go
CREATE SCHEMA estacion AUTHORIZATION DBO
go
CREATE SCHEMA empleados AUTHORIZATION DBO
go

--AGREGANDO CADA TABLA A SUS RESPECTIVOS ESQUEMAS
ALTER SCHEMA usuarios TRANSFER dbo.usuario;
ALTER SCHEMA usuarios TRANSFER dbo.tipo_membresia;
ALTER SCHEMA usuarios TRANSFER dbo.telefono;
ALTER SCHEMA usuarios TRANSFER dbo.metodo_pago;
ALTER SCHEMA usuarios TRANSFER dbo.incidente;
ALTER SCHEMA usuarios TRANSFER dbo.viaje;
ALTER SCHEMA usuarios TRANSFER dbo.plan_seguro;
ALTER SCHEMA usuarios TRANSFER dbo.membresia;
ALTER SCHEMA usuarios TRANSFER dbo.historial_viaje
ALTER SCHEMA estacion TRANSFER dbo.estacion;
ALTER SCHEMA estacion TRANSFER dbo.color;
ALTER SCHEMA estacion TRANSFER dbo.bicimantenimiento;
ALTER SCHEMA estacion TRANSFER dbo.bicicleta;
ALTER SCHEMA estacion TRANSFER dbo.terminal;
ALTER SCHEMA empleados TRANSFER dbo.administrador;
ALTER SCHEMA empleados TRANSFER dbo.empleado;
ALTER SCHEMA empleados TRANSFER dbo.empleado_idioma;
ALTER SCHEMA empleados TRANSFER dbo.historial_falta;
ALTER SCHEMA empleados TRANSFER dbo.mantenimiento;
ALTER SCHEMA empleados TRANSFER dbo.motivo;
ALTER SCHEMA empleados TRANSFER dbo.idioma;
ALTER SCHEMA empleados TRANSFER dbo.agente;



CREATE  TABLE tipo_incidente (
       idTipoInc int primary key,
       tipo varchar(30) unique,
       descripcion varchar(60)
```

```
)

ALTER TABLE usuarios.incidente
ADD idTipoInc int
```

7.3 SCRIPT dml.sql

```
/*
        *Proyecto: Ecobici
        *Autores: Daniel Aguilar
                        Gonzalez Sotelo ElÌas Eduardo
                        Velazquez Martinez Karla Andrea
        *Facultad de IngenierÌa UNAM
        *Profesora: Martha Lopez Pelcastre
        *Bases de datos: Grupo 02
        *Equipo: 3
        *ARCHIVO DML
*/


/*-----------------   FUNCION 1-----------------------------------

        FUNCION PARA ESTADÕSTICA 4
        Autor: Elias Eduardo Gonz·lez

        Fecha de creaciÛn: 11 / 06 / 2023

        Descripcion: FunciÛn util para segmentar por edades particulares para la estadÌstica 4

-----------------------------------------------------------------------------------------------------*/


--FUNCION PARA ESTADÕSTICA 4
CREATE or ALTER FUNCTION usuarios.edades()
RETURNS TABLE
AS

RETURN (SELECT CASE
                        WHEN edad <=15 THEN '10-15 A—OS'
                        WHEN edad <=20 THEN '15-20 A—OS'
                        WHEN edad <=30 THEN '20-30 A—OS'
                        ELSE '+30 A—OS'
                        END AS rango_edades, COUNT(*) AS 'Numero de Usuarios'
                FROM usuarios.USUARIO
                GROUP BY CASE
                        WHEN edad <=15 THEN '10-15 A—OS'
                        WHEN edad <=20 THEN '15-20 A—OS'
                        WHEN edad <=30 THEN '20-30 A—OS'
                        ELSE '+30 A—OS'
                        END
                )

GO

/*------------    FUNCION 2      ----------------------------------------


        FUNCION PARA ESTADÕSTICA 7

        Autor: Eduardo Elias Gonz·lez
```

```
        Fecha de creaciÛn: 11 / 06 / 2023

        Descripcion: funcion que recibe el numero entero del mes para retornar el una tabla
        con el numero de incidentes de acuerdo a una segmentaciÛn por un mes dado

-------------------------------------------------------------*/

CREATE OR ALTER FUNCTION empleados.AgentesReporte(@mes int)
RETURNS TABLE
AS
RETURN(
SELECT  a.id_empleado,  CONCAT(e.nombre, ' ', e.paterno) as 'Nombre Completo', count(*) 'Numero de incidentes
auxiliados'
                FROM usuarios.incidente i inner join empleados.agente a on i.id_empleado = a.id_empleado
                        inner join usuarios.viaje v on v.id_viaje = i.id_viaje inner join
                        empleados.empleado e on e.id_empleado =  a.id_empleado
                WHERE MONTH (v.fecha) = @mes
                GROUP BY a.id_empleado, CONCAT(e.nombre, ' ', e.paterno)
                --ORDER BY 'Numero de incidentes auxiliados' DESC
);
go


/*--------------------    FUNCION 3 Y FUNCION 4  -------------------


        FUNCIONES PARA ESTADÕSTICA 10

        Autor: Daniel Aguilar

        Fecha de creaciÛn: 09 / 06 / 2023

        Descripcion: Regresa una tabla cada funciÛn. Una regresa una tabla segmentada por un intervalo de
fechas.
        La segunda hace lo mismo pero segmentando por un numero entero, referente al id de la estaciÛn en
cuestiÛn.


-------------------------------------------------------------*/

CREATE OR ALTER FUNCTION estacion.recorridoPeriodo(@fecha1 DATE, @fecha2 DATE) -- RECORRIDOS DE ACUERDO A UN
PERIODO DE TIEMPO
RETURNS TABLE
AS
        RETURN (
                SELECT  u.nombre  +  '  '  +u.paterno  +  '  '  +  u.materno  as  NombreUsuario,  e.nombre  as
EstacionPartida,
                        ep.nombre as EstacionLlegada,v.fecha, v.tarifa as Costo, DATEDIFF(MINUTE,hora_ini,
v.hora_llegada) AS DuracionMinutos
                FROM usuarios.viaje v inner join usuarios.USUARIO u on
                v.id_usuario  =  u.id_usuario  inner  join  estacion.estacion  e  on  v.estacionPartida  =
e.id_estacion
                inner join estacion.estacion ep on v.estacionLlegada = ep.id_estacion
                where v.fecha > @fecha1 and v.fecha < @fecha2
        )
GO

CREATE OR ALTER FUNCTION estacion.recorridoEstacion(@estacion INT) -- RECORRIDOS DE ACUERDO A UNA ESTACI”N EN
PARTICULAR
RETURNS TABLE
AS
        RETURN (
```

```
		SELECT u.nombre + ' ' +u.paterno + ' ' + u.materno as NombreUsuario, e.nombre as
EstacionPartida,
                                ep.nombre as EstacionLlegada, v.fecha, v.tarifa as Costo,
DATEDIFF(MINUTE,hora_ini, v.hora_llegada) AS DuracionMinutos
		FROM usuarios.viaje v inner join usuarios.USUARIO u on
		v.id_usuario = u.id_usuario inner join estacion.estacion e on v.estacionPartida =
e.id_estacion
		inner join estacion.estacion ep on v.estacionLlegada = ep.id_estacion
		where v.estacionPartida = @estacion
	)
GO


/*--------------------------------------                                                TRIGGER                          1
-------------------------------------------------------------------


	Autor: Karla Vel·zquez

	Fecha de creaciÛn: 11 / 06 / 2023

	Descripcion: Trigger para validar la integridad de la jerarquÌa de tipos en la tabla de agente.
	Verifica que no exista en la tabla de Mantenimiento ni Administrador. Ademas, valida que tampoco
	exista en la tabla de empleado con el tipo 'R' (recursos humanos)


-------------------------------------------------------------------------------------------------------------------
-------*/

CREATE OR ALTER TRIGGER empleados.tr_usuarios
ON empleados.agente
INSTEAD OF INSERT
AS
BEGIN
	IF EXISTS (SELECT 1 FROM empleados.mantenimiento WHERE id_empleado = (SELECT id_empleado FROM
inserted))
		BEGIN
			PRINT 'No es posible realizar InsersiÛn, existe en la tabla de mantenimiento'
			RETURN
		END
	IF EXISTS (SELECT 1 FROM empleados.administrador WHERE id_empleado = (SELECT id_empleado FROM
inserted))
	    BEGIN
			PRINT 'No es posible realizar InsersiÛn, existe en la tabla de administrador'
			RETURN
		END
	IF EXISTS (SELECT 1 FROM empleados.empleado e inner join inserted i on e.id_empleado = i.id_empleado
where e.tipo_empleado = 'R')
		BEGIN
			PRINT 'No es posible realizar la insersiÛn, es un empleado de recursos humanos'
			RETURN
		END
	IF EXISTS (SELECT 1 FROM empleados.empleado e inner join inserted i on e.id_empleado = i.id_empleado
where e.tipo_empleado ='G')
		BEGIN
			INSERT INTO empleados.agente (id_empleado)
			SELECT id_empleado FROM inserted
		END
	ELSE
		PRINT 'NO EXISTE EL ID PREVIAMENTE EN EMPLEADO o EXISTE PERO NO TIENE EL TIPO AGENTE (G)'

END
```

```
/*-------------------------------------------------TRIGGER
2-----------------------------------------------------------

        Autor: Daniel Aguilar

        Fecha de creaciÛn: 12 / 06 / 2023

        Descripcion: Trigger para validar que al insertar un viaje,haya coherencia en la informaciÛn
ingresada. Es decir, para
        un viaje dado, verifica que al ingresar el id de la bicicleta cuya estaciÛn de partida es X y la
estaciÛn de llegada
        es Y, se revise si dicha bicicleta a insertar efectivamente su estaciÛn en la que est· es la estaciÛn
X. En caso de
        no estarlo no se permite la inserciÛn. Por el contrario, si la informaciÛn del viaje se ingresa
adecuadamente, entonces
        en la tabla de bicicleta se cambia el id de la estaciÛn en la que se encuentra por la estaciÛn de
llegada Y de dicho
        viaje.

-------------------------------------------------------------------------------------------------------------
---------*/
GO
CREATE OR ALTER TRIGGER usuarios.tr_InsertarViaje
ON usuarios.viaje
INSTEAD OF INSERT
AS
BEGIN

        IF  EXISTS(SELECT  1    FROM  estacion.bicicleta  b  inner  join  inserted  i  on  b.id_bicicleta  =
i.id_bicicleta where b.id_estacion = i.estacionPartida)
                BEGIN
                        PRINT 'La bicicleta coincide con la estaciÛn de partida. Se ha efectuado el insert
exitosamente'

                        --ACTUALIZAMOS LA ESTACION EN DONDE EST¡ LA BICICLETA
                        UPDATE estacion.bicicleta
                        SET id_estacion = (SELECT estacionLlegada FROM inserted)
                        WHERE id_bicicleta = (SELECT id_bicicleta FROM inserted)

                        --SE INSERTA EL VIAJE EXITOSAMENTE
                        INSERT INTO viaje (id_bicicleta, id_usuario, hora_fin, hora_ini, hora_llegada, fecha,
ruta, estacionPartida, estacionLlegada)
                        SELECT  id_bicicleta,  id_usuario,  hora_fin,  hora_ini,  hora_llegada,  fecha,  ruta,
estacionPartida, estacionLlegada
                        FROM inserted
                END
        ELSE
                BEGIN
                        --NO SE REALIZA LA INSERCI”N DEL VIAJE
                        PRINT 'La bici que trataste de insertar no pertenece a la estaciÛn de partida'
                        RETURN
                END


END


/*-------------------------------------   VISTA  1 ------------------------------------


        VISTA PARA ESTADÕSTICA 13
```

```
        Autor: Daniel Aguilar

        Fecha de creaciÛn: 12 / 06 / 2023

        Descripcion: Regresa una tabla con toda la informaciÛn de los empleados, lo que facilita evitar tener
que nombrar
        todos los atributos para cada select en el que se use la tabla de empleado. Adem·s, se juntan los 3
nombres en uno


------------------------------------------------------------*/
go
CREATE OR ALTER VIEW empleados.visEmpleados as
select rfc , nombre + ' ' + paterno + ' ' + materno AS Nombre, sueldo, tipo_empleado, edad
from empleados.empleado
go

/*----------------------------------------                                                                    VISTA
2-------------------------------------------------------------------

        Autor: Daniel Aguilar

        Fecha de creaciÛn: 12 / 06 / 2023

        Descripcion: Vista con cada uno de los idiomas que habla cada empleado. Da la informaciÛn del
id_empleado,
                                nombre e idiomas


-------------------------------------------------------------------------------------------------------------
-------*/

CREATE OR ALTER VIEW empleados.visEmpleadosIdiomas as
select e.id_empleado, nombre + ' ' + paterno + ' ' + materno as Nombre, id.idioma
from EMPLEADOS.empleado e inner join empleados.EMPLEADO_IDIOMA ei on e.id_empleado = ei.id_empleado
inner join empleados.idioma id on ei.id_idioma = id.id_idioma

go

/*----------------------------------------                                                                    VISTA
3-------------------------------------------------------------------

        Autor: Eduardo Elias Gonz·lez

        Fecha de creaciÛn: 12 / 06 / 2023

        Descripcion: Vista util para tener informaciÛn sobre todas las bicicletas


-------------------------------------------------------------------------------------------------------------
-------*/
go
CREATE OR ALTER VIEW estacion.visBicicleta as
select id_bicicleta, color, modelo, nserie, tamaÒo, es.nombre, es.ubicacion
FROM estacion.bicicleta b inner join estacion.color c on b.id_color = c.id_color
inner join estacion.modelo m on m.id_modelo = b.id_modelo inner join estacion.estacion es
on es.id_estacion  = b.id_estacion

go

/*----------------------------------------    USO    DE    TRANSACCIONES    Y    MANEJO    DE    ERRORES
-----------------------------------------------
```

```
        EL USO DE TRANSACCIONES SE UTILIZA EN LOS TRIGGERS DE ESTE DOCUMENTO PARA TENER UN CONTROL SOBRE LAS
ESTRUCTURAS CONDICIONALES.
        ADEM¡S, EL USO DE TRANSACCIONES SE USAN EN EL ARCHIVO DE ESTADÕSTICAS PARA PROBAR LA SALIDA, USANDO
ROLLBACK TRANSACTION Y
        COMMIT TRANSACTION.

        EL MANEJO DE ERRORES VIENE VALIDADO EN EL MANEJO DE TRIGGERS PARA CONSERVAR LA INTEGRIDAD DE LA
INFORMACI”N, ADEM¡S DE QUE LOS
        POSIBLES ERRORES SE MANDAN MENSAJES DE POR QU… NO SE PUDO REALIZAR UNA INSERCI”N, POR EJEMPLO

-------------------------------------------------------------------------------------------------------------------
-------------------------*/

-------------------------------------------------- LLAMADAS A PROCEDIMIENTOS
------------------------------------------------------------

--ESTADISTICA 1
EXECUTE usuarios.pa_reporteIncidentes '2022-02-22','2022-10-30'

--ESTADÕSTICA 5
EXECUTE estacion.InventarioBicicletas '2022-01-04', '2022-12-30'

--ESTADÕSTICA 6
EXEC usuarios.InformeMembresias

--ESTADÕSTICA 9
EXEC empleados.InformeEmpleados


--ESTADÕSTICA 10

--PROBANDO POR PERIODO DE TIEMPO (SOLO MUESTRA UN SELECT)
EXEC estacion.recorridos NULL, '2022-02-20', '2022-10-30'

--PROBANDO POR AMBOS CASOS (MUESTRA DOS SELECT'S)
EXEC estacion.recorridos 5, '2022-02-20', '2022-10-30'

--PROBANDO POR ESTACION NADA MAS (MUESTRA UN SELECT)
EXEC estacion.recorridos 5, '2022-02-20', NULL

--PROBANDO UN CASO QUE NO GENERE REPORTE
EXEC estacion.recorridos NULL, '2022-02-20', NULL

-- ESTADÕSTICA 12
EXEC empleados.AgentesAccidentes


-- USUARIOS

EXEC CrearUsuarioConsulta @nombreUsuario = 'MiNuevoUsuario', @contrasena = 'MiNuevaContrasena';


EXEC CrearUsuarioGestor @nombreUsuario = 'MiNuevoGestor', @contrasena = 'MiNuevaContrasena';

EXEC CrearUsuarioAdministrador @nombreUsuario = 'MiNuevoAdministrador', @contrasena = 'MiNuevaContrasena';


-- USO DEL LIKE

--Seleccionar nombre de empleados cuya segunda letra es una e
-- y en su rfc haya un 5
SELECT * from  empleados.visEmpleados where nombre LIKE LOWER('_e%') AND RFC LIKE '%5%'
```

7.4 SCRIPT informes.sql

- **ESTADÍSTICA 1:** Estadísticas de los daños en las bicicletas con mayor frecuencia. Top 5 de los accidentes más frecuentes (descripción del daño, cantidad).

```
--Estadísticas de los daños en las bicicletas con mayor frecuencia. Top 5 de
-- los accidentes más frecuentes (descripción del daño, cantidad)

SELECT TOP 5 i.idTipoInc as 'ID_Tipo_Accidente',ti.tipo as 'Tipo de accidente',ti.descripcion as
'Descripción del daño', count(*) AS  'Número de incidentes'
FROM usuarios.incidente i INNER JOIN usuarios.tipo_incidente ti ON i.idTipoInc = ti.idTipoInc
GROUP BY i.idTipoInc, ti.tipo, ti.descripcion
ORDER BY 'Número de incidentes' DESC
```

Results | Messages

	ID_Tipo_Accidente	Tipo de accidente	Descripción del daño	Número de incidentes
1	1	P. BICI	Problema al chocar con otra bicicleta	9
2	2	P. Coche	Problema al chocar con una moto	3
3	3	P. Moto	Problema al chocar con un peatón	3
4	5	Otros	Problema sin especificaciones	3
5	6	P. PEATÓN	Problema al chocar/toparse con un peatón	2

Tipos de accidentes (daños en las bicicletas) más comunes. 5 accidentes más frecuentes

- **ESTADÍSTICA 2:** Estaciones con más reportes de accidentes con mayor frecuencia. Listado de estaciones con el número de accidentes en un periodo de tiempo (fecha inicio – fecha fin) ordenados de mayor a menor.

```
todología: Partimos del siguiente supuesto:
--En un viaje de la estación A a la estación B, si llega a haber un accidente
--Se le atribuye a la estación de llegada, es decir a la estación B-- Me
GO
CREATE OR ALTER PROCEDURE usuarios.pa_reporteIncidentes
--Variables de entrada (fecha inicio y fecha fin)
	@fecha1 date,
	@fecha2 date
AS
BEGIN
	--Declaración de tabla artificial de salida
	CREATE TABLE #EstacionIncidentes(
		id_estacion int,
		nombre varchar(20),
		incidentes int,
		fechas varchar(200)
	)
```

```
--DECLARACIÓN DE VARIABLES DEL CURSOR INTERNO (ITERAR SOBRE ACCIDENTES PARA CADA ESTACIÓN DADA)

	DECLARE @fechaAcc varchar(12)
	DECLARE @estacionAcc int

	--VARIABLE AUXILIAR PARA EL NÚMERO DE INCIDENTES Y JUNTAR FECHAS EN UN SOLO REGISTRO
	DECLARE @numIncidentes int
	DECLARE @fechas varchar(200)

--DECLARACIÓN DE VARIABLES DEL CURSOR EXTERNO (ITERAR SOBRE ESTACIONES)
	DECLARE @id_estacion int
	DECLARE @nombre varchar(20)

--DECLARACIÓN CURSOR EXTERNO
	DECLARE cursor_Estacion CURSOR
	FOR
	SELECT id_estacion, nombre
	FROM estacion.estacion

--APERTURA CURSOR EXTERNO
	OPEN cursor_Estacion

	--LECTURA PRIMERA FILA EXTERNA
	FETCH cursor_Estacion INTO @id_estacion, @nombre

	WHILE (@@FETCH_STATUS = 0)
		BEGIN

		DECLARE cursor_Incidente CURSOR
		FOR
		SELECT v.estacionLlegada,v.fecha
		FROM estacion.estacion e
		INNER JOIN usuarios.viaje v  ON e.id_Estacion = v.estacionLlegada
		INNER JOIN usuarios.incidente i ON v.id_viaje = i.id_viaje
		WHERE v.fecha < @fecha2 and v.fecha > @fecha1
			 and v.estacionLlegada =  @id_estacion

		    OPEN cursor_Incidente

			SET @numIncidentes = 0
			SET @fechas = ''

			FETCH cursor_Incidente INTO @estacionAcc, @fechaAcc
			WHILE (@@FETCH_STATUS = 0)
				BEGIN
					SET @fechas = @fechas + @fechaAcc + ' || '
					SET @numIncidentes = @numIncidentes + 1
					FETCH cursor_Incidente INTO @estacionAcc, @fechaAcc
				END

		INSERT INTO #EstacionIncidentes (id_estacion, nombre, incidentes, fechas) VALUES
			(@id_estacion, @nombre, @numIncidentes, @fechas)

			CLOSE cursor_Incidente
			DEALLOCATE cursor_Incidente
```

```
		FETCH cursor_Estacion INTO @id_estacion, @nombre
	END

	CLOSE cursor_Estacion
	DEALLOCATE cursor_Estacion

	SELECT @fecha1 AS FechaInicial, @fecha2 AS FechaFinal

	SELECT * FROM #EstacionIncidentes
	ORDER BY incidentes DESC

	DROP TABLE #EstacionIncidentes


END
GO

EXECUTE usuarios.pa_reporteIncidentes '2022-02-22','2022-10-30'
```

	FechaInicial	FechaFinal
1	2022-02-22	2022-10-30

	id_estacion	nombre	incidentes	fechas
1	5	Est. 5	4	2022-06-10 || 2022-05-17 || 2022-03-11 || 2022-08-19 ||
2	1	Est. 1	3	2022-03-02 || 2022-08-15 || 2022-03-12 ||
3	2	Est. 2	2	2022-07-18 || 2022-10-02 ||
4	3	Est. 3	2	2022-06-28 || 2022-10-06 ||
5	4	Est. 4	1	2022-09-05 ||

Estaciones y sus accidentes, ordenadas por número de accidentes

- **ESTADÍSTICA 3:** Total de accidentes en un rango de fechas, listados de mayor a menor

```
SELECT CASE
		WHEN v.fecha >= '2022-01-01' AND v.fecha < '2022-04-01' THEN 'Enero-Marzo'
		WHEN v.fecha >= '2022-04-01' AND v.fecha < '2022-07-01' THEN 'Abril-Junio'
		WHEN v.fecha >= '2022-07-01' AND v.fecha <= '2022-10-01' THEN 'Julio-Septiembre'
		WHEN v.fecha >= '2022-10-01' AND v.fecha <= '2022-12-31' THEN 'Octubre-Diciembre'
	END  AS Rango_Fechas, COUNT(*) as Numero_Incidentes
FROM usuarios.viaje v INNER JOIN usuarios.incidente i ON v.id_viaje = i.id_viaje
GROUP BY
	CASE
		WHEN v.fecha >= '2022-01-01' AND v.fecha < '2022-04-01' THEN 'Enero-Marzo'
		WHEN v.fecha >= '2022-04-01' AND v.fecha < '2022-07-01' THEN 'Abril-Junio'
		WHEN v.fecha >= '2022-07-01' AND v.fecha <= '2022-10-01' THEN 'Julio-Septiembre'
		WHEN v.fecha >= '2022-10-01' AND v.fecha <= '2022-12-31' THEN 'Octubre-Diciembre'
	END
ORDER BY Numero_Incidentes
```

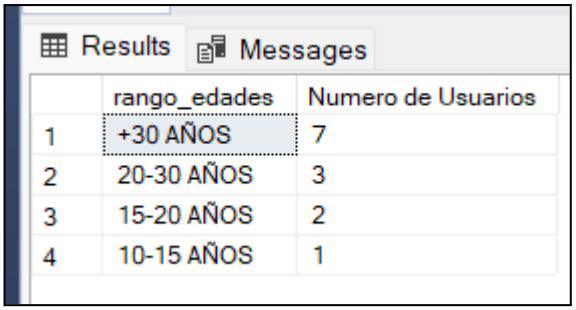
Results | Messages

	Rango_Fechas	Numero_Incidentes
1	Abril-Junio	3
2	Julio-Septiembre	4
3	Enero-Marzo	7
4	Octubre-Diciembre	8

Número de incidentes para un rango de fechas (meses en este caso)

- **ESTADÍSTICA 4:** Total de usuarios por rangos de fechas y rangos de edades (10 a 215 años, 15-20 años, 20 a 30 años, más de 30 años)

```
-- No tenemos usuarios con edad de 10 a 15 años, por lo que creamos a uno
-- De 15 a 20 años creamos a 2
INSERT INTO usuarios.USUARIO (app_saldo, materno, nombre, paterno, fecha_nac, correo, codigo_ine,
genero)
	VALUES
	(0, 'Rodríguez', 'Agustín','Arriaga', '2005-04-23','AgustinArr@gmail.com', '134678543',
'H'),
	(1, 'Perez', 'Omar','Razo', '2006-06-25','OmarRaz@gmail.com', '954334200', 'H'),
	(1, 'Mancera', 'Rocío','Razo', '2012-02-28','RocioRaz@gmail.com', '998665335', 'M')


-- Creación de función para segmentar por edades:
GO
CREATE or ALTER FUNCTION usuarios.edades()
RETURNS TABLE
AS

RETURN (SELECT CASE
			WHEN edad <=15 THEN '10-15 AÑOS'
			WHEN edad <=20 THEN '15-20 AÑOS'
			WHEN edad <=30 THEN '20-30 AÑOS'
			ELSE '+30 AÑOS'
			END AS rango_edades, COUNT(*) AS 'Numero de Usuarios'
		FROM usuarios.USUARIO
		GROUP BY CASE
			WHEN edad <=15 THEN '10-15 AÑOS'
			WHEN edad <=20 THEN '15-20 AÑOS'
			WHEN edad <=30 THEN '20-30 AÑOS'
			ELSE '+30 AÑOS'
			END
		)

GO

SELECT * FROM usuarios.edades()
ORDER BY [Numero de Usuarios] DESC
```



Results | Messages

	rango_edades	Numero de Usuarios
1	+30 AÑOS	7
2	20-30 AÑOS	3
3	15-20 AÑOS	2
4	10-15 AÑOS	1

Número de usuarios por rango de edades

- **ESTADÍSTICA 5:** Inventario de las bicicletas (todos los datos de las bicicletas) por estaciones con el número de viajes, por un periodo de tiempo, incluir el número de accidentes si ha tenido

```
CREATE OR ALTER PROCEDURE estacion.InventarioBicicletas
	@fecha1 date,
	@fecha2 date
AS
BEGIN
	--Variables del cursor de bicicletas
	DECLARE @id_bicicleta int, @color varchar(15), @modelo varchar(12),@nserie varchar(12),
@tamaño varchar(12), @estado varchar(10), @id_estacion int

--Salida deseada para el reporte de bicicletas
	CREATE TABLE #inventarioBicicletas (
		id_bicicleta int,
		color varchar(15),
		modelo varchar(12),
		nserie varchar(12),
		tamaño varchar(12),
		estado varchar(5),
		id_estacion int,
		numViajes int,
		numAccidentes int
	)
--Variables auxiliares del numero de viajes y número de accidentes
	DECLARE @numViajes int, @numAccidentes int


--Declarando cursor
	DECLARE cursor_Bicicletas CURSOR
		FOR
		SELECT b.id_bicicleta, c.color, m.modelo, b.nserie, b.tamaño,
				b.estado, b.id_estacion

		FROM estacion.bicicleta b
				INNER JOIN estacion.color c ON c.id_color = b.id_color
				INNER JOIN estacion.modelo m on m.id_modelo = b.id_modelo

	OPEN cursor_Bicicletas
	FETCH  cursor_Bicicletas INTO @id_bicicleta, @color, @modelo, @nserie, @tamaño, @estado,
@id_estacion

	WHILE (@@FETCH_STATUS = 0)
```

```
		BEGIN
			SET @numViajes = (SELECT count(*) FROM usuarios.viaje WHERE id_bicicleta
=@id_bicicleta and fecha >= @fecha1 and fecha <= @fecha2)
			SET @numAccidentes = (SELECT count(*) FROM usuarios.viaje v INNER JOIN
usuarios.incidente i ON v.id_viaje = i.id_viaje
                                                                WHERE v.id_bicicleta =
@id_bicicleta and v.fecha >= @fecha1 and v.fecha <= @fecha2)
			INSERT INTO #inventarioBicicletas (id_bicicleta, color, modelo, nserie,
tamaño,estado, id_estacion, numViajes,numAccidentes)
			VALUES (@id_bicicleta, @color, @modelo, @nserie, @tamaño, @estado,
@id_estacion, @numViajes, @numAccidentes)

			FETCH  cursor_Bicicletas INTO @id_bicicleta, @color, @modelo, @nserie,
@tamaño, @estado, @id_estacion
		END


	CLOSE cursor_Bicicletas
	DEALLOCATE cursor_Bicicletas

	SELECT * FROM #inventarioBicicletas
	ORDER BY id_estacion, id_bicicleta
	DROP TABLE #inventarioBicicletas

END

EXECUTE estacion.InventarioBicicletas '2022-01-04', '2022-12-30'
```

	id_bicicleta	color	modelo	nserie	tamaño	estado	id_estacion	numViajes	numAccidentes
1	1	Rojo	BICYCLE ROD	BYCL-0001	Chica	F	1	1	1
2	2	Azul	BICYCLE ROD	BYCL-0002	Mediana	F	1	1	0
3	3	Azul	BICYCLE ROD	BYCL-0003	Chica	F	1	0	0
4	4	Rojo	BICYCLE ROD	BYCL-0004	Chica	F	1	1	1
5	5	Azul	BICYCLE ROD	BYCL-0005	Mediana	F	1	1	1
6	6	Rojo	BICYCLE ROD	BYCL-0006	Grande	F	1	2	2
7	7	Rojo	BICYCLE ROD	BYCL-0007	Mediana	D	1	1	0
8	8	Rojo	BICYCLE ROD	BYCL-0008	Grande	B	1	2	1
9	9	Am...	BICYCLE ROD	BYCL-0009	Chica	F	2	0	0
10	10	Am...	BICYCLE ROD	BYCL-0010	Grande	F	2	2	1
11	11	Rojo	BICYCLE ROD	BYCL-0011	Mediana	F	2	2	0
12	12	Azul	BICYCLE ROD	BYCL-0012	Mediana	F	2	2	2
13	13	Azul	BICYCLE ROD	BYCL-0013	Mediana	D	2	1	1
14	14	Rojo	BICYCLE ROD	BYCL-0014	Mediana	F	2	1	1
15	15	Rojo	BICYCLE ROD	BYCL-0015	Mediana	F	2	1	1
16	16	Azul	BICYCLE ROD	BYCL-0016	Chica	B	2	0	0
17	17	Am...	BICYCLE ROD	BYCL-0017	Grande	F	3	1	0
18	18	Am...	BICYCLE ROD	BYCL-0018	Chica	F	3	1	0
19	19	Rojo	BICYCLE ROD	BYCL-0019	Mediana	F	3	2	1
20	20	Rojo	BICYCLE ROD	BYCL-0020	Chica	D	3	1	1
21	21	Rojo	BICYCLE ROD	BYCL-0021	Mediana	F	3	0	0
22	22	Rojo	BICYCLE ROD	BYCL-0022	Grande	F	3	3	1
23	23	Azul	BICYCLE ROD	BYCL-0023	Mediana	F	3	1	0
24	24	Azul	BICYCLE ROD	BYCL-0024	Mediana	B	3	1	0
25	25	Am...	BICYCLE ROD	BYCL-0025	Mediana	F	4	0	0
26	26	Am...	BICYCLE ROD	BYCL-0026	Chica	F	4	0	0
27	27	Azul	BICYCLE ROD	BYCL-0027	Chica	F	4	2	1
28	28	Rojo	BICYCLE ROD	BYCL-0028	Grande	F	4	1	0
29	29	Rojo	BICYCLE ROD	BYCL-0029	Grande	F	4	2	0
30	30	Azul	BICYCLE ROD	BYCL-0030	Chica	F	4	1	1
31	31	Am...	BICYCLE ROD	BYCL-0031	Chica	D	4	2	2
32	32	Am...	BICYCLE ROD	BYCL-0032	Mediana	D	4	0	0
33	33	Rojo	BICYCLE ROD	BYCL-0033	Grande	F	5	0	0
34	34	Rojo	BICYCLE ROD	BYCL-0034	Mediana	F	5	1	0
35	35	Azul	BICYCLE ROD	BYCL-0035	Chica	F	5	2	1
36	36	Am...	BICYCLE ROD	BYCL-0036	Grande	F	5	1	1
37	37	Rojo	BICYCLE ROD	BYCL-0037	Mediana	F	5	1	0
38	38	Am...	BICYCLE ROD	BYCL-0038	Mediana	F	5	1	0
39	39	Azul	BICYCLE ROD	BYCL-0039	Mediana	F	5	1	1
40	40	Azul	BICYCLE ROD	BYCL-0040	Mediana	F	5	0	0

Información de todas las bicicletas y número de viajes efectuados por cada una, así como el número de incidentes (si tienen). Segmentado por un periodo de tiempo

- **ESTADÍSTICA 6:** Listado de usuarios (datos generales), datos de su membresía y el tiempo en meses que tienen la membresía

```
DELETE FROM usuarios.membresia
DBCC CHECKIDENT ('usuarios.membresia', reseed, 0) -- Ejecutar dos veces
-- NUEVAS INSERCIONES PARA TENER UNA MEMBRESÍA POR USUARIO
INSERT INTO usuarios.membresia (fecha, id_tipo, precio_tarjeta, codigoQR, id_pago) VALUES
              ('2023-02-14', 1, 50.00, 1245.323, 1),
              ('2021-05-11', 3, 50.00, 9876.222, 5),
              ('2021-05-11', 2, 50.00, 2453.222, 6),
              ('2021-03-01', 1, 50.00, 1435.336, 7),
              ('2021-05-12', 2, 50.00, 4321.789, 8),
              ('2021-06-15', 3, 50.00, 5678.987, 9),
              ('2021-02-25', 1, 50.00, 2468.135, 10),
              ('2021-10-26', 2, 50.00, 8795.642, 11),
              ('2021-11-27', 3, 50.00, 7531.864, 12),
              ('2021-12-12', 1, 50.00, 3198.572, 13)

GO
CREATE OR ALTER PROCEDURE usuarios.InformeMembresias
AS
BEGIN
       SELECT u.nombre + ' ' + u.paterno + ' ' + u.materno as NombreCompleto, u.fecha_nac,
u.correo, u.codigo_ine, u.genero, u.edad,
       m.id_membresia, m.fecha 'Fecha Suscripcion', m.beneficio, m.duracionDiasSuscripcion,
       CASE WHEN m.duracionDiasSuscripcion = 1 THEN '1 día, 0 meses'
            WHEN m.duracionDiasSuscripcion = 30 THEN '1 mes'
               WHEN m.duracionDiasSuscripcion = 365 THEN '12 meses'
       END as 'Duracion Membresia'
       ,
       mp.tipo_pago
       FROM usuarios.membresia m inner join usuarios.metodo_pago mp on m.id_pago = mp.id_pago
       right join usuarios.usuario u
                     on u.id_usuario = mp.id_usuario

END


EXEC usuarios.InformeMembresias
```

Results Messages

	NombreCompleto	fecha_nac	correo	codigo_ine	genero	edad	id_membresia	Fecha Suscripcion	beneficio	duracionDiasSuscripcion	Duracion Membresia	tipo_pago
1	Braulio Fernández Martínez	1995-05-12	Braulio_Martinez@gmail.com	126054319	H	28	1	2023-02-14	Descuento	1	1 día, 0 meses	CREDITO
2	María López González	1982-07-18	maria_lopez@gmail.com	563029857	M	41	2	2021-05-11	Cashback	365	12 meses	PAYPAL
3	Juan García Hernández	1987-11-23	juan_garcia@gmail.com	732916845	O	36	3	2021-05-11	Viaje	30	1 mes	CREDITO
4	Laura Vargas Torres	1992-09-30	laura_vargas@gmail.com	429187536	M	31	4	2021-03-01	Descuento	1	1 día, 0 meses	CREDITO
5	Carlos Sánchez Rojas	1983-04-05	carlos_sanchez@gmail.com	903457218	H	40	5	2021-05-12	Viaje	30	1 mes	DEBITO
6	Ana Jiménez Fernández	1988-12-15	ana_jimenez@gmail.com	628743591	M	35	6	2021-06-15	Cashback	365	12 meses	DEBITO
7	Pedro González López	1997-02-28	pedro_gonzalez@gmail.com	182456903	O	26	7	2021-02-25	Descuento	1	1 día, 0 meses	DEBITO
8	Mónica Hernández García	1985-08-08	monica_hernandez@gmail....	347819524	M	38	8	2021-10-26	Viaje	30	1 mes	DEBITO
9	Jorge Torres Vargas	1994-06-20	jorge_torres@gmail.com	519672430	H	29	9	2021-11-27	Cashback	365	12 meses	DEBITO
10	María Rojas Sánchez	1981-03-11	maria_rojas@gmail.com	683927541	M	42	10	2021-12-12	Descuento	1	1 día, 0 meses	PAYPAL
11	Agustín Arriaga Rodríguez	2005-04-23	AgustinArr@gmail.com	134678543	H	18	NULL	NULL	NULL	NULL	NULL	NULL
12	Omar Razo Perez	2006-06-25	OmarRaz@gmail.com	954334200	H	17	NULL	NULL	NULL	NULL	NULL	NULL
13	Rocío Razo Mancera	2012-02-28	RocioRaz@gmail.com	998665335	M	11	NULL	NULL	NULL	NULL	NULL	NULL

Datos de todos los usuarios, membresía y tiempo en meses que la tienen. Hay usuarios sin membresia

- **ESTADÍSTICA 7:** Agentes mejor recocidos en un mes especifico, para eso cada que un agente auxilia a un usuario en algún incidente el usuario llena una pequeña encuesta

```
INSERT INTO usuarios.incidente (id_empleado, hora, fecha, coordenadas, calle, numero,
codigoPostal, alcaldia, colonia, id_viaje, idTipoInc)
VALUES (11, '23:03:00', NULL, '156.665.546', 'Av. San Monk', 60, '50666', 'Madrigal', 'Pedregal
Monk', 33,4)

INSERT INTO usuarios.viaje VALUES (30,7, '20:10:00','18:10:00','21:12:00','2022-11-05','Av Rojas
- AvE - Estacion ', 4,5)
INSERT INTO usuarios.incidente (id_empleado, hora, fecha, coordenadas, calle, numero,
codigoPostal, alcaldia, colonia, id_viaje, idTipoInc)
VALUES (11, '21:03:00', NULL, '156.633.546', 'Av. San Crack', 2, '50336', 'Madrigal', 'Pedregal
Crack', 43,2)

GO
CREATE OR ALTER FUNCTION empleados.AgentesReporte(@mes int)
RETURNS TABLE
AS
RETURN(
SELECT  a.id_empleado,  CONCAT(e.nombre, ' ', e.paterno) as 'Nombre Completo', count(*) 'Número
de incidentes auxiliados'
       FROM usuarios.incidente i inner join empleados.agente a on i.id_empleado = a.id_empleado
                     inner join usuarios.viaje v on v.id_viaje = i.id_viaje inner join
                     empleados.empleado e on e.id_empleado =  a.id_empleado
              WHERE MONTH (v.fecha) = @mes
              GROUP BY a.id_empleado, CONCAT(e.nombre, ' ', e.paterno)
);
go

SELECT 'MES DE NOVIEMBRE 11'
SELECT * FROM empleados.AgentesReporte(11)
ORDER BY [Numero de incidentes auxiliados] DESC
```

Results Messages

	(No column name)
1	MES DE NOVIEMBRE 11

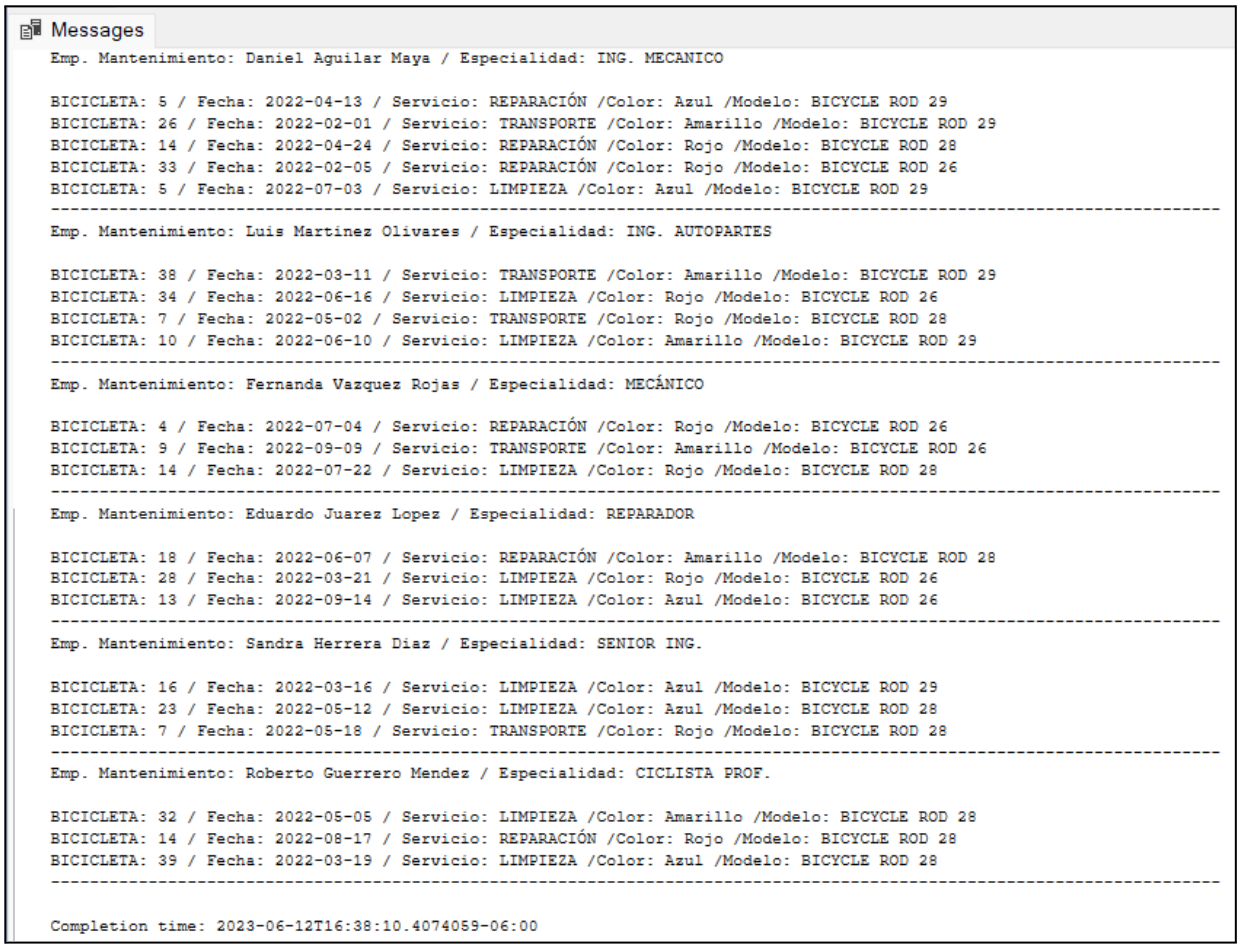
	id_empleado	Nombre Completo	Numero de incidentes auxiliados
1	11	Lorena Vega	2
2	7	Veronica Cruz	1
3	10	David Zamora	1
4	8	Jose Morales	1

Para el mes 11 se tienen a los agentes que participaron en accidentes, ordenados de forma descendente por el número de incidentes auxiliados (mejor reconocidos)

- **ESTADÍSTICA 8:** Informe de todos los empleados administradores y las bicicletas que auxiliaron para su mantenimiento

```
--Variables externas
DECLARE @id_empleado int,@nombre varchar(15), @paterno varchar(20), @materno varchar(20),
@especialidad varchar(25)
--Variables internas
DECLARE @id_bicicleta int, @fecha date, @servicio varchar(20), @color varchar(20), @modelo
varchar(30)


DECLARE cursorMantenimiento CURSOR
FOR
	SELECT m.id_empleado,nombre, paterno, materno, especialidad
	FROM empleados.mantenimiento m INNER join empleados.empleado e on m.id_empleado =
e.id_empleado

OPEN cursorMantenimiento

FETCH cursorMantenimiento INTO @id_empleado,@nombre, @paterno, @materno, @especialidad

WHILE (@@FETCH_STATUS = 0)
	BEGIN
		PRINT 'Emp. Mantenimiento: ' + @nombre + ' ' + @paterno + ' ' + @materno + ' /
Especialidad: ' + @especialidad
		PRINT ''

		DECLARE cursorBicicletas CURSOR FOR
			SELECT b.id_bicicleta,bc.fecha, bc.servicio, c.color, m.modelo
			FROM estacion.bicimantenimiento bc INNER JOIN estacion.bicicleta  b on
b.id_bicicleta = bc.id_bicicleta
			INNER JOIN estacion.modelo m ON m.id_modelo = b.id_modelo
			INNER JOIN estacion.color c on c.id_color = b.id_color
			WHERE bc.id_empleado = @id_empleado

		OPEN cursorBicicletas
		FETCH cursorBicicletas INTO @id_bicicleta, @fecha, @servicio, @color, @modelo

		WHILE(@@FETCH_STATUS = 0)
		BEGIN
			PRINT 'BICICLETA: ' + CAST(@id_bicicleta as VARCHAR(3)) + ' / Fecha: ' +
CAST(@fecha as varchar(15)) + ' / Servicio: ' + @servicio
					+ ' /Color: ' + @color + ' /Modelo: ' + @modelo
			FETCH cursorBicicletas INTO @id_bicicleta, @fecha, @servicio, @color,
@modelo

		END
PRINT
'-------------------------------------------------------------------------------------------------------------'
		CLOSE cursorBicicletas
		DEALLOCATE cursorBicicletas

		FETCH cursorMantenimiento INTO @id_empleado,@nombre, @paterno, @materno,
@especialidad

	END
```

```
CLOSE cursorMantenimiento
DEALLOCATE cursorMantenimiento
```

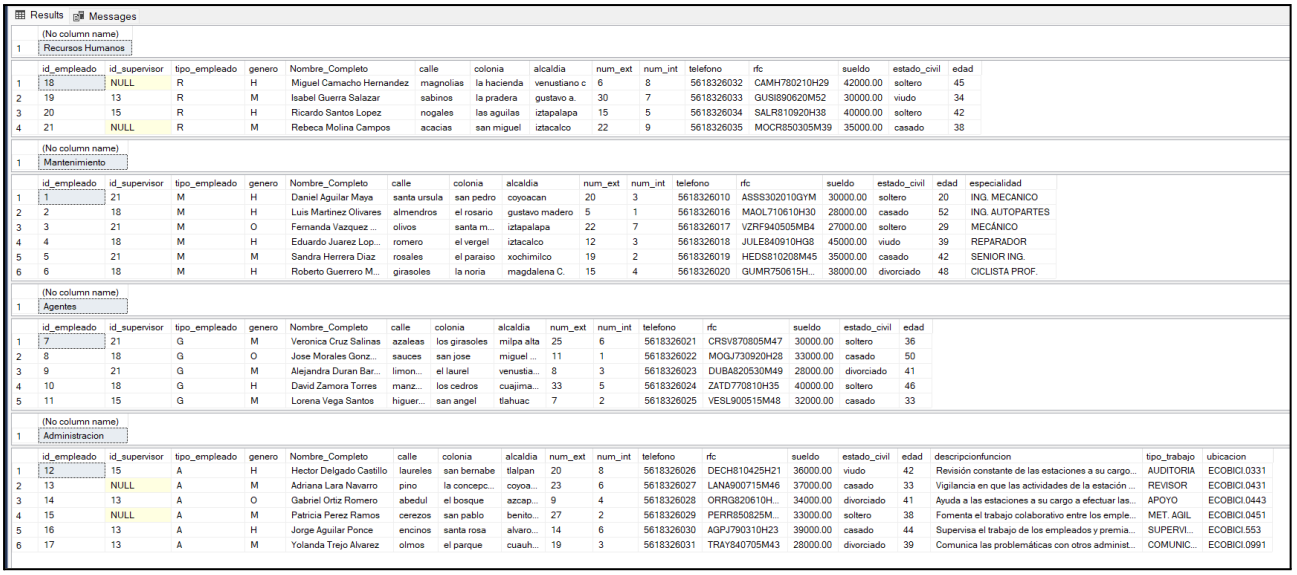


Bicicletas auxiliadas por cada empleado de mantenimiento

- **ESTADÍSTICA 9:** Listado de empleados con su tipo

```
go
CREATE or ALTER PROCEDURE empleados.InformeEmpleados
AS
BEGIN
	DECLARE @contador int
	SET @contador = 1

	WHILE @contador <= 4
	BEGIN
		IF @contador = 1
			BEGIN
				SELECT 'Recursos Humanos'
				SELECT id_empleado, id_supervisor, tipo_empleado, genero, nombre + ' ' + paterno + ' ' + materno
				 AS 'Nombre_Completo', calle, colonia, alcaldia, num_ext, num_int, telefono, rfc, sueldo, estado_civil,
				 edad
```

```
                        FROM empleados.empleado WHERE tipo_empleado = 'R'
                END
            ELSE IF @contador = 2
                    BEGIN
                        SELECT 'Mantenimiento'
                        SELECT e.id_empleado, id_supervisor, e.tipo_empleado, e.genero,
e.nombre +  ' ' + e.paterno + ' ' + e.materno
                        AS 'Nombre_Completo', e.calle, e.colonia, e.alcaldia, e.num_ext,
e.num_int, e.telefono, e.rfc, e.sueldo, e.estado_civil,
                        e.edad, m.especialidad
                        FROM empleados.empleado e INNER JOIN empleados.mantenimiento m ON
e.id_empleado = m.id_empleado
                    END
            ELSE IF @contador = 3
                    BEGIN
                        SELECT 'Agentes'
                        SELECT e.id_empleado, id_supervisor, e.tipo_empleado, e.genero,
e.nombre +  ' ' + e.paterno + ' ' + e.materno
                        AS 'Nombre_Completo', e.calle, e.colonia, e.alcaldia, e.num_ext,
e.num_int, e.telefono, e.rfc, e.sueldo, e.estado_civil,
                        e.edad
                        FROM empleados.empleado e INNER JOIN empleados.agente a on
e.id_empleado = a.id_empleado
                    END
            ELSE
                    BEGIN
SELECT 'Administracion'
                        SELECT e.id_empleado, id_supervisor, e.tipo_empleado, e.genero,
e.nombre +  ' ' + e.paterno + ' ' + e.materno
                        AS 'Nombre_Completo', e.calle, e.colonia, e.alcaldia, e.num_ext,
e.num_int, e.telefono, e.rfc, e.sueldo, e.estado_civil,
                        e.edad, a.descripcionfuncion, a.tipo_trabajo, a.ubicacion
                        FROM empleados.empleado e INNER JOIN empleados.administrador a on
a.id_empleado = e.id_empleado
                    END
            SET @contador = @contador + 1
        END

END


EXEC empleados.InformeEmpleados
```



Results Messages

(No column name) 1 Recursos Humanos

	id_empleado	id_supervisor	tipo_empleado	genero	Nombre_Completo	calle	colonia	alcaldia	num_ext	num_int	telefono	rfc	sueldo	estado_civil	edad
1	18	NULL	R	H	Miguel Camacho Hernandez	magnolias	la hacienda	venustiano c	6	8	5618326032	CAMH780210H29	42000.00	soltero	45
2	19	13	R	M	Isabel Guerra Salazar	sabinos	la pradera	gustavo a.	30	7	5618326033	GUSI890620M52	30000.00	viudo	34
3	20	15	R	H	Ricardo Santos Lopez	nogales	las aguilas	iztapalapa	15	5	5618326034	SALR810920H38	40000.00	soltero	42
4	21	NULL	R	M	Rebeca Molina Campos	acacias	san miguel	iztacalco	22	9	5618326035	MOCR850305M39	35000.00	casado	38

(No column name) 1 Mantenimiento

	id_empleado	id_supervisor	tipo_empleado	genero	Nombre_Completo	calle	colonia	alcaldia	num_ext	num_int	telefono	rfc	sueldo	estado_civil	edad	especialidad
1	1	21	M	H	Daniel Aguilar Maya	santa ursula	san pedro	coyoacan	20	3	5618326010	ASSS302010GYM	30000.00	soltero	20	ING. MECANICO
2	2	18	M	H	Luis Martinez Olivares	almendros	el rosario	gustavo madero	5	1	5618326016	MAOL710610H30	28000.00	casado	52	ING. AUTOPARTES
3	3	21	M	O	Fernanda Vazquez ...	olivos	santa m...	iztapalapa	22	7	5618326017	VZRF940505MB4	27000.00	soltero	29	MECÁNICO
4	4	18	M	H	Eduardo Juarez Lop...	romero	el vergel	iztacalco	12	3	5618326018	JULE840910HG8	45000.00	viudo	39	REPARADOR
5	5	21	M	M	Sandra Herrera Diaz	rosales	el paraiso	xochimilco	19	2	5618326019	HEDS810208M45	35000.00	casado	42	SENIOR ING.
6	6	18	M	H	Roberto Guerrero M...	girasoles	la noria	magdalena C.	15	4	5618326020	GUMR750615H...	38000.00	divorciado	48	CICLISTA PROF.

(No column name) 1 Agentes

	id_empleado	id_supervisor	tipo_empleado	genero	Nombre_Completo	calle	colonia	alcaldia	num_ext	num_int	telefono	rfc	sueldo	estado_civil	edad
1	7	21	G	M	Veronica Cruz Salinas	azaleas	los girasoles	milpa alta	25	6	5618326021	CRSV870805M47	30000.00	soltero	36
2	8	18	G	O	Jose Morales Gonz...	sauces	san jose	miguel ...	11	1	5618326022	MOGJ730920H28	33000.00	casado	50
3	9	21	G	M	Alejandra Duran Bar...	limon...	el laurel	venustia...	8	3	5618326023	DUBA820530M49	28000.00	divorciado	41
4	10	18	G	H	David Zamora Torres	manz...	los cedros	cuajima...	33	5	5618326024	ZATD770810H35	40000.00	soltero	46
5	11	15	G	M	Lorena Vega Santos	higuer...	san angel	tlahuac	7	2	5618326025	VESL900515M48	32000.00	casado	33

(No column name) 1 Administracion

	id_empleado	id_supervisor	tipo_empleado	genero	Nombre_Completo	calle	colonia	alcaldia	num_ext	num_int	telefono	rfc	sueldo	estado_civil	edad	descripcionfuncion	tipo_trabajo	ubicacion
1	12	15	A	H	Hector Delgado Castillo	laureles	san bernabe	tlalpan	20	8	5618326026	DECH810425H21	36000.00	viudo	42	Revisión constante de las estaciones a su cargo...	AUDITORIA	ECOBICI.0331
2	13	NULL	A	M	Adriana Lara Navarro	pino	la concepc...	coyoa...	23	6	5618326027	LANA900715M46	37000.00	casado	33	Vigilancia en que las actividades de la estación ...	REVISOR	ECOBICI.0431
3	14	13	A	O	Gabriel Ortiz Romero	abedul	el bosque	azcap...	9	4	5618326028	ORRG820610H...	34000.00	divorciado	41	Ayuda a las estaciones a su cargo a efectuar las...	APOYO	ECOBICI.0443
4	15	NULL	A	M	Patricia Perez Ramos	cerezos	san pablo	benito...	27	2	5618326029	PERR850825M...	33000.00	soltero	38	Fomenta el trabajo colaborativo entre los emple...	MET. AGIL	ECOBICI.0451
5	16	13	A	H	Jorge Aguilar Ponce	encinos	santa rosa	alvaro...	14	6	5618326030	AGPJ790310H23	39000.00	casado	44	Supervisa el trabajo de los empleados y premia...	SUPERVI...	ECOBICI.553
6	17	13	A	M	Yolanda Trejo Alvarez	olmos	el parque	cuauh...	19	3	5618326031	TRAY840705M43	28000.00	divorciado	39	Comunica las problemáticas con otros administ...	COMUNIC...	ECOBICI.0991

Listar a cada empleado de acuerdo al tipo de empleado que sea

- **ESTADÍSTICA 10:** Informe de los recorridos, por estación y/o por periodo de tiempo (fecha inicio y fecha fin); nombre del usuario, estación de partida, lugar de llegada, tiempo en minutos del recorrido y costo

```
go
CREATE OR ALTER FUNCTION estacion.recorridoPeriodo(@fecha1 DATE, @fecha2 DATE)-- RECORRIDOS DE
ACUERDO A UN PERIODO DE TIEMPO
RETURNS TABLE
AS
	RETURN (
		SELECT u.nombre + ' ' +u.paterno + ' ' + u.materno as NombreUsuario, e.nombre as
EstacionPartida,
			ep.nombre as EstacionLlegada,v.fecha, v.tarifa as Costo,
DATEDIFF(MINUTE,hora_ini, v.hora_llegada) AS DuracionMinutos
		FROM usuarios.viaje v inner join usuarios.USUARIO u on
		v.id_usuario = u.id_usuario inner join estacion.estacion e on v.estacionPartida =
e.id_estacion
		inner join estacion.estacion ep on v.estacionLlegada = ep.id_estacion
		where v.fecha > @fecha1 and v.fecha < @fecha2
	)
GO

CREATE OR ALTER FUNCTION estacion.recorridoEstacion(@estacion INT) -- RECORRIDOS DE ACUERDO A UNA
ESTACIÓN EN PARTICULAR
RETURNS TABLE
AS
	RETURN (
		SELECT u.nombre + ' ' +u.paterno + ' ' + u.materno as NombreUsuario, e.nombre as
EstacionPartida,
			ep.nombre as EstacionLlegada, v.fecha, v.tarifa as Costo,
DATEDIFF(MINUTE,hora_ini, v.hora_llegada) AS DuracionMinutos
		FROM usuarios.viaje v inner join usuarios.USUARIO u on
		v.id_usuario = u.id_usuario inner join estacion.estacion e on v.estacionPartida =
e.id_estacion
		inner join estacion.estacion ep on v.estacionLlegada = ep.id_estacion
```

```
                where v.estacionPartida = @estacion
        )
GO

CREATE OR ALTER PROCEDURE estacion.recorridos -- PROCEDIMIENTO PARA ESCOGER INFORME POR ESTACION
Y/O POR PERIODO DE TIEMPO
        @parametro1 INT = NULL,
        @fecha1 DATE= NULL,
        @fecha2 DATE = NULL
AS
BEGIN
        IF @fecha1 IS NOT NULL and @fecha2 IS NOT NULL and @parametro1 IS NULL
                BEGIN
                        SELECT 'INFORME POR PERIODO DE TIEMPO'
                        SELECT @fecha1 AS FechaInicio, @fecha2 AS FechaFin
                        SELECT * from estacion.recorridoPeriodo(@fecha1, @fecha2)
                END
        ELSE IF @fecha1 IS NOT NULL and @fecha2 IS NOT NULL and @parametro1 IS NOT NULL
                BEGIN
                        SELECT 'INFORME POR PERIODO DE TIEMPO'
                        SELECT * from estacion.recorridoPeriodo(@fecha1,@fecha2)
                        SELECT 'INFORME POR ESTACION'
                        SELECT * FROM estacion.recorridoEstacion(@parametro1)
                END
        ELSE IF @parametro1 IS NOT NULL and (@fecha1 IS NULL or @fecha2 IS NULL)
                BEGIN
                        SELECT 'INFORME POR ESTACION'
                        SELECT * FROM estacion.recorridoEstacion(@parametro1)
                END
        ELSE
                PRINT 'NO FUE POSIBLE GENERAR UNA SALIDA'

END

--Primer parámetro: Una estacion del 1 al 5
--Segundo parámetro: fecha inicial
--Tercer parámetro: fecha final

--PROBANDO POR PERIODO DE TIEMPO (SOLO MUESTRA UN SELECT)
EXEC estacion.recorridos NULL, '2022-02-20', '2022-10-30'

--PROBANDO POR AMBOS CASOS (MUESTRA DOS SELECT'S)
EXEC estacion.recorridos 5, '2022-02-20', '2022-10-30'

--PROBANDO POR ESTACION NADA MAS (MUESTRA UN SELECT)
EXEC estacion.recorridos 5, '2022-02-20', NULL

--PROBANDO UN CASO QUE NO GENERE REPORTE
EXEC estación.recorridos NULL, '2022-02-20', NULL
```

Results Messages

	(No column name)
1	INFORME POR PERIODO DE TIEMPO

	FechaInicio	FechaFin
1	2022-02-20	2022-10-30

	NombreUsuario	EstacionPartida	EstacionLlegada	fecha	Costo	DuracionMinutos
1	María López González	Est. 4	Est. 1	2022-03-02	25	215
2	Juan García Hernández	Est. 1	Est. 5	2022-06-10	0	60
3	Laura Vargas Torres	Est. 3	Est. 2	2022-07-18	0	90
4	Mónica Hernández García	Est. 5	Est. 4	2022-09-05	0	75
5	Ana Jiménez Fernández	Est. 4	Est. 1	2022-04-29	0	90

Caso 1: Informe únicamente por periodo de tiempo

Results Messages

	(No column name)
1	INFORME POR PERIODO DE TIEMPO

	NombreUsuario	EstacionPartida	EstacionLlegada	fecha	Costo	DuracionMinutos
1	María López González	Est. 4	Est. 1	2022-03-02	25	215
2	Juan García Hernández	Est. 1	Est. 5	2022-06-10	0	60
3	Laura Vargas Torres	Est. 3	Est. 2	2022-07-18	0	90
4	Mónica Hernández García	Est. 5	Est. 4	2022-09-05	0	75
5	Ana Jiménez Fernández	Est. 4	Est. 1	2022-04-29	0	90
6	Pedro González López	Est. 3	Est. 2	2022-08-03	5	45
7	María Rojas Sánchez	Est. 5	Est. 4	2022-10-12	15	75
8	Braulio Fernández Martí...	Est. 1	Est. 5	2022-05-17	0	90

	(No column name)
1	INFORME POR ESTACION

	NombreUsuario	EstacionPartida	EstacionLlegada	fecha	Costo	DuracionMinutos
1	Mónica Hernández García	Est. 5	Est. 4	2022-09-05	0	75
2	María Rojas Sánchez	Est. 5	Est. 4	2022-10-12	15	75
3	Juan García Hernández	Est. 5	Est. 4	2022-01-25	0	60
4	Pedro González López	Est. 5	Est. 4	2022-12-09	15	135
5	Laura Vargas Torres	Est. 5	Est. 3	2022-11-14	0	90
6	María López González	Est. 5	Est. 3	2022-12-01	0	75
7	Jorge Torres Vargas	Est. 5	Est. 3	2022-06-29	45	195
8	Juan García Hernández	Est. 5	Est. 3	2022-10-06	5	75

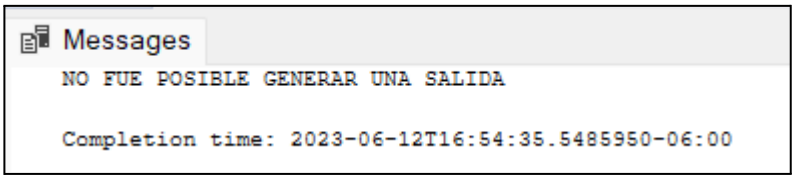
Caso 2: Informe por periodo de tiempo y por cierta estación de partida

Results Messages

	(No column name)
1	INFORME POR ESTACION

	NombreUsuario	EstacionPartida	EstacionLlegada	fecha	Costo	DuracionMinutos
1	Mónica Hernández García	Est. 5	Est. 4	2022-09-05	0	75
2	María Rojas Sánchez	Est. 5	Est. 4	2022-10-12	15	75
3	Juan García Hernández	Est. 5	Est. 4	2022-01-25	0	60
4	Pedro González López	Est. 5	Est. 4	2022-12-09	15	135
5	Laura Vargas Torres	Est. 5	Est. 3	2022-11-14	0	90
6	María López González	Est. 5	Est. 3	2022-12-01	0	75
7	Jorge Torres Vargas	Est. 5	Est. 3	2022-06-29	45	195
8	Juan García Hernández	Est. 5	Est. 3	2022-10-06	5	75

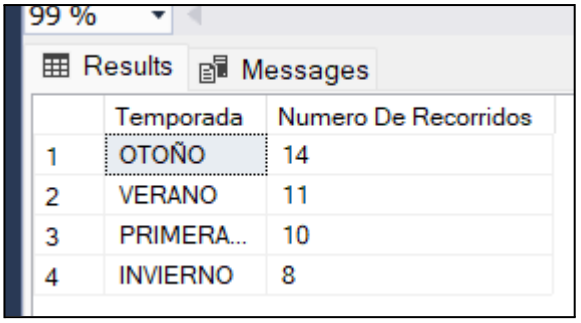
Caso 3: Informe únicamente por cierta estación de partida



Caso 4: Variables de entrada no válidas

- **ESTADÍSTICA 11:** Épocas del año con número de recorridos ordenados de mayor a menor

```
go
CREATE OR ALTER FUNCTION dbo.fechaTemporada( @fecha date)
returns varchar(30)
as
begin
	declare @estacion varchar(30)
	declare @mes INT

	SET @mes = MONTH(@fecha)

	IF @mes IN (12,1,2)
		SET @estacion = 'INVIERNO'
	ELSE IF @mes IN (3,4,5)
		SET @estacion = 'PRIMAVERA'
	ELSE IF @mes in (6,7,8)
		SET @estacion = 'VERANO'
	ELSE
		SET @estacion = 'OTOÑO'
	RETURN @estacion
end
go

SELECT t.Temporada, count(*) as 'Numero De Recorridos'
FROM (
SELECT  dbo.fechaTemporada(fecha) as Temporada, fecha
FROM usuarios.viaje ) as t
group by Temporada
order by [Numero De Recorridos] DESC
```



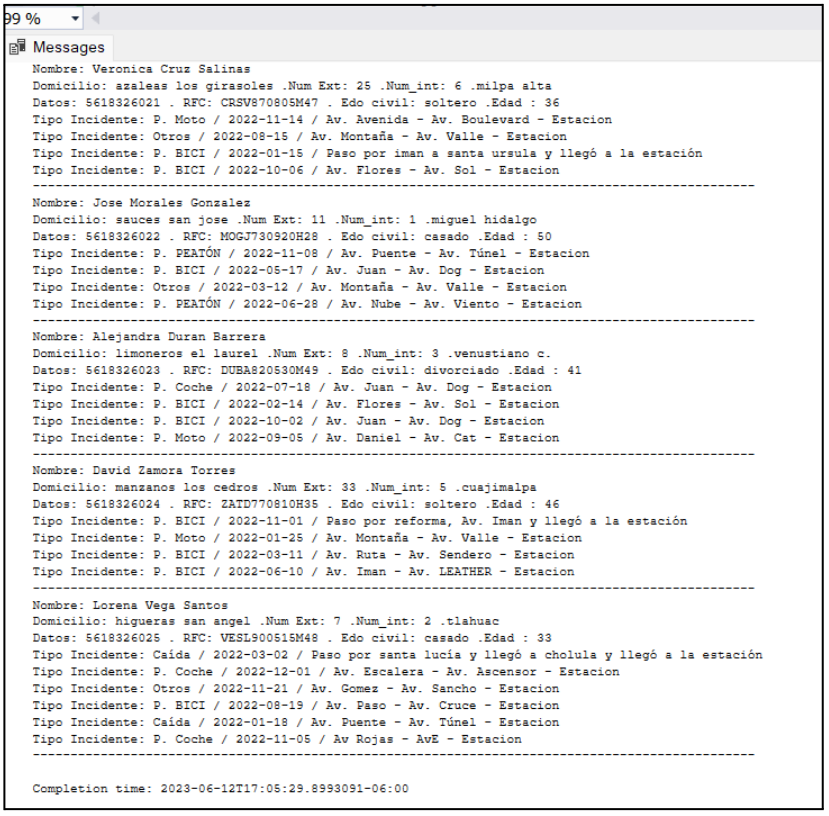
	Temporada	Numero De Recorridos
1	OTOÑO	14
2	VERANO	11
3	PRIMERA...	10
4	INVIERNO	8

Épocas del año con mayor número de recorridos

- **ESTADÍSTICA 12:** Obtener para cada agente sus datos personales y el listado de los accidentes que han atendido (tipo de accidente, fecha, lugar)

```
CREATE OR ALTER PROCEDURE empleados.AgentesAccidentes
AS
BEGIN
	--Cursor externo variables
	DECLARE @id_empleado int, @nombre varchar(20), @paterno varchar(20), @materno varchar(20),
@calle varchar(20), @colonia varchar(20),
	@alcaldia varchar(25), @num_ext int, @num_int int, @telefono varchar(15),@rfc varchar(14),
@sueldo money, @estado_civil varchar(20), @edad int

	--Cursor interno variables
	DECLARE @id_empleadoAcc int, @tipo varchar(20), @fecha date, @ruta varchar(150)

	DECLARE curAgente CURSOR
	FOR
	SELECT a.id_empleado,e.nombre, e.paterno, e.materno, e.calle, e.colonia, e.alcaldia,
e.num_ext, e.num_int, e.telefono, e.rfc, e.sueldo, e.estado_civil,
		e.edad
	FROM EMPLEADOS.agente a inner join empleados.empleado e  on a.id_empleado = e.id_empleado

	OPEN curAgente

	FETCH curAgente into @id_empleado, @nombre, @paterno, @materno,@calle, @colonia,
@alcaldia, @num_ext, @num_int, @telefono, @rfc, @sueldo, @estado_civil, @edad

	WHILE (@@FETCH_STATUS = 0)
	BEGIN
		PRINT 'Nombre: ' + @nombre + ' ' + @paterno + ' ' + @materno
		PRINT 'Domicilio: ' + @calle +  ' ' + @colonia + ' .Num Ext: ' + CAST(@num_ext as
varchar(3)) + ' .Num_int: ' + CAST(@num_int as varchar(4)) + ' .'  + @alcaldia
		PRINT 'Datos: ' + @telefono + ' . RFC: ' + @rfc + ' . Edo civil: ' + @estado_civil
+ ' .Edad : ' + CAST(@edad as varchar(4))

		DECLARE curAgenteDetalle CURSOR
		FOR SELECT i.id_empleado,ti.tipo,v.fecha,v.ruta FROM usuarios.incidente i INNER
join usuarios.tipo_incidente ti ON i.idTipoInc = ti.idTipoInc inner join
		usuarios.viaje v on v.id_viaje = i.id_viaje WHERE  id_empleado = @id_empleado

		OPEN curAgenteDetalle

		FETCH curAgenteDetalle into @id_empleadoAcc, @tipo, @fecha, @ruta
```

```
		WHILE (@@FETCH_STATUS = 0)
			BEGIN
				PRINT 'Tipo Incidente: ' + @tipo + ' / ' + CAST(@fecha AS
VARCHAR(13)) + ' / ' + @ruta
				FETCH curAgenteDetalle into @id_empleadoAcc, @tipo, @fecha, @ruta
			END
		PRINT'-----------------------------------------------------------------'
		CLOSE curAgenteDetalle
		DEALLOCATE curAgenteDetalle

		FETCH curAgente into @id_empleado, @nombre, @paterno, @materno,@calle, @colonia,
@alcaldia, @num_ext, @num_int, @telefono, @rfc, @sueldo, @estado_civil, @edad

	END

	CLOSE curAgente
	DEALLOCATE curAgente

END

EXEC empleados.AgentesAccidentes
```



Accidentes que cada agente ha atendido

ESTADÍSTICA 13: Para el área de recursos humanos es importante un informe mensual de todos los empleados y sus datos: RFC, nombre completo y sueldo, asimismo nombre de los empleados y puesto de los que tengan un sueldo de 13000 mensuales y pertenezcan a la tercera edad.

```
-- usando vista (dml)
go
CREATE OR ALTER VIEW empleados.visEmpleados as
select rfc , nombre + ' ' + paterno + ' ' + materno AS Nombre, sueldo, tipo_empleado, edad
from empleados.empleado
go

-- salida 1
SELECT *, case
       when tipo_empleado ='R' then 'Recursos Humanos'
       when tipo_empleado ='G' then 'Agente'
       when tipo_empleado ='M' then 'Mantenimiento'
       when tipo_empleado ='A' then 'Administración'
       end as 'Puesto'
FROM empleados.visEmpleados


UPDATE empleados.empleado
SET sueldo = 13000
WHERE id_empleado in (3,7,9,10,11)

UPDATE empleados.empleado
SET edad = 70
WHERE id_empleado in (10,11)

UPDATE empleados.empleado
SET edad = 77
WHERE id_empleado in (3)

UPDATE empleados.empleado
SET edad = 77
WHERE id_empleado in (15)


--salida 2
SELECT nombre,
case
       when tipo_empleado ='R' then 'Recursos Humanos'
       when tipo_empleado ='G' then 'Agente'
       when tipo_empleado ='M' then 'Mantenimiento'
       when tipo_empleado ='A' then 'Administración'
       end as 'Puesto',edad,sueldo
from empleados.visEmpleados
where edad > 65 and sueldo =13000
```

	rfc	Nombre	sueldo	tipo_empleado	edad	Puesto
1	ASSS302010GYM	Daniel Aguilar Maya	30000.00	M	20	Mantenimiento
2	MAOL710610H30	Luis Martinez Olivares	28000.00	M	52	Mantenimiento
3	VZRF940505MB4	Fernanda Vazquez Rojas	13000.00	M	77	Mantenimiento
4	JULE840910HG8	Eduardo Juarez Lopez	45000.00	M	39	Mantenimiento
5	HEDS810208M45	Sandra Herrera Diaz	35000.00	M	42	Mantenimiento
6	GUMR750615H50	Roberto Guerrero Mendez	38000.00	M	48	Mantenimiento
7	CRSV870805M47	Veronica Cruz Salinas	13000.00	G	36	Agente
8	MOGJ730920H28	Jose Morales Gonzalez	33000.00	G	50	Agente
9	DUBA820530M49	Alejandra Duran Barrera	13000.00	G	41	Agente
10	ZATD770810H35	David Zamora Torres	13000.00	G	70	Agente
11	VESL900515M48	Lorena Vega Santos	13000.00	G	70	Agente
12	DECH810425H21	Hector Delgado Castillo	36000.00	A	42	Administración
13	LANA900715M46	Adriana Lara Navarro	37000.00	A	33	Administración
14	ORRG820610H39	Gabriel Ortiz Romero	34000.00	A	41	Administración
15	PERR850825M37	Patricia Perez Ramos	33000.00	A	77	Administración
16	AGPJ790310H23	Jorge Aguilar Ponce	39000.00	A	44	Administración
17	TRAY840705M43	Yolanda Trejo Alvarez	28000.00	A	39	Administración
18	CAMH780210H29	Miguel Camacho Herna...	42000.00	R	45	Recursos Humanos
19	GUSI890620M52	Isabel Guerra Salazar	30000.00	R	34	Recursos Humanos
20	SALR810920H38	Ricardo Santos Lopez	40000.00	R	42	Recursos Humanos
21	MOCR850305M39	Rebeca Molina Campos	35000.00	R	38	Recursos Humanos

Informe de todos los empleados (usando vistas)

	nombre	Puesto	edad	sueldo
1	Fernanda Vazquez Rojas	Mantenimiento	77	13000.00
2	David Zamora Torres	Agente	70	13000.00
3	Lorena Vega Santos	Agente	70	13000.00

Informe de personas de la tercera edad y sueldo de 13000

7.5 SCRIPT cargaInicial.sql

```
/*
	INSERCIÓN DE EMPLEADOS (PK NONCLUSTERED)
	--CALCULANDO EDAD A PARTIR DE LA FECHA DE NACIMIENTO
	--GENERO H , M , O
	--CODIGO INE DE 9 DIGITOS
	--INDICE NONCLUSTERED en id_usuario IDENTITY
*/

INSERT INTO usuarios.USUARIO (app_saldo, materno, nombre, paterno, fecha_nac, correo,
codigo_ine, genero)
VALUES
(1, 'Martínez', 'Braulio', 'Fernández','1995-05-12', 'Braulio_Martinez@gmail.com', '126054319','H'),
(1, 'González', 'María', 'López', '1982-07-18', 'maria_lopez@gmail.com', '563029857', 'M'),
(1, 'Hernández', 'Juan', 'García', '1987-11-23', 'juan_garcia@gmail.com', '732916845', 'O'),
(0, 'Torres', 'Laura', 'Vargas', '1992-09-30', 'laura_vargas@gmail.com', '429187536', 'M'),
(1, 'Rojas', 'Carlos', 'Sánchez', '1983-04-05', 'carlos_sanchez@gmail.com', '903457218', 'H'),
(0, 'Fernández', 'Ana', 'Jiménez', '1988-12-15', 'ana_jimenez@gmail.com', '628743591', 'M'),
(1, 'López', 'Pedro', 'González', '1997-02-28', 'pedro_gonzalez@gmail.com', '182456903', 'O'),
(0, 'García', 'Mónica', 'Hernández', '1985-08-08', 'monica_hernandez@gmail.com', '347819524', 'M'),
(1, 'Vargas', 'Jorge', 'Torres', '1994-06-20', 'jorge_torres@gmail.com', '519672430', 'H'),
(0, 'Sánchez', 'María', 'Rojas', '1981-03-11', 'maria_rojas@gmail.com', '683927541', 'M')


/*
	INSERCIÓN DE TELEFONOS DE EMPLEADOS (PK CLUSTERED)
	--TELEFONO (CHECK LIKE DE 10 DIGITOS)
	--UNIQUE
*/
INSERT INTO usuarios.telefono (id_usuario, tel)
VALUES
    (1, '5610246429'),
    (1, '5620394857'),
    (2, '5639283746'),
    (2, '5647291830'),
    (3, '5657392018'),
    (4, '5668457293'),
    (5, '5672819403'),
    (6, '5689382740'),
    (7, '5693748291'),
    (7, '5602938471'),
    (7, '5619384752'),
    (8, '5629384019'),
    (9, '5634958710'),
    (10, '5643092817'),
    (10, '5616250312')
```

```
/*
 INSERCIÓN DE METODOS DE PAGO
  -- (PK CLUSTERED)
  -- CHECK TIPO DE PAGO : 'CREDITO', 'DEBITO' , 'PAYPAL'
  -- id_usuario NONCLUSTERED
*/

INSERT INTO usuarios.metodo_pago  (activo, tipo_pago, id_usuario)
VALUES (1, 'CREDITO', 1),
             (1, 'DEBITO', 1),
             (1, 'PAYPAL', 1),
             (1, 'CREDITO', 2),
             (0, 'PAYPAL', 2),
             (1, 'CREDITO', 3),
             (0, 'CREDITO', 4),
             (1, 'DEBITO', 5),
             (1, 'DEBITO', 6),
             (1, 'DEBITO', 7),
             (0, 'DEBITO', 8),
             (1, 'DEBITO', 9),
             (1, 'PAYPAL', 10),
             (0, 'CREDITO', 10),
             (1, 'DEBITO', 10),
             (1, 'DEBITO', 4)


/*
   INSERCIÓN DE tipo_membresia
  -- Check tipo: 'B', 'I', 'P'
  -- PK CLUSTERED
*/
INSERT INTO usuarios.tipo_membresia (tipo) VALUES ('B'), ('I') , ('P')

UPDATE usuarios.tipo_membresia
SET costo = CASE tipo
                    WHEN 'B' THEN 118
                    WHEN 'I' THEN 400
                    WHEN 'P' THEN 1000
                    END

/*
INSERCIÓN DE membresia
-- precio de tarjeta: todas en 50, luego con triggers verificar 50 o 80 para inserts y update
-- beneficio y duracionDiasSuscripcion calculado (pk CLUSTERED)
*/

INSERT INTO usuarios.membresia (fecha, id_tipo, precio_tarjeta, codigoQR, id_pago)
VALUES
             ('2023-02-14', 1, 50.00, 1245.323, 1),
             ('2021-05-11', 3, 50.00, 9876.222, 3),
             ('2021-05-11', 2, 50.00, 2453.222, 5),
```

```
('2021-03-01', 1, 50.00, 1435.336, 6),
('2021-05-12', 2, 50.00, 4321.789, 7),
('2021-06-15', 3, 50.00, 5678.987, 8),
('2021-02-25', 1, 50.00, 2468.135, 9),
('2021-10-26', 2, 50.00, 8795.642, 10),
('2021-11-27', 3, 50.00, 7531.864, 11),
('2021-12-12', 1, 50.00, 3198.572, 12),
('2021-08-08', 2, 50.00, 6482.947, 13),
('2021-09-02', 3, 50.00, 9274.516, 15),
('2021-10-10', 1, 50.00, 1584.247, 16);
```

```
/*
INSERCIÓN DE empleado
-- tipo empleado:
-- M: Mantenimiento, G: Agente, A: Administrador, R: Recursos humanos
      -- genero : H M O
      -- ESTADO CIVIL: 'SOLTERO','CASADO','VIUDO','DIVORCIADO'
      (pk nonClustered)
*/
```

```
INSERT INTO empleados.empleado(nombre, paterno, materno, calle, colonia,
alcaldia,num_ext, num_int, telefono, rfc, sueldo, estado_civil,edad, genero,
tipo_empleado)
VALUES
('Daniel', 'Aguilar', 'Maya', 'santa ursula', 'san pedro', 'coyoacan', 20,3,5618326010,
'ASSS302010GYM', 30000, 'soltero', 20, 'H', 'M'),
('Luis', 'Martinez', 'Olivares', 'almendros', 'el rosario', 'gustavo madero', 5,1,5618326016,
'MAOL710610H30', 28000, 'casado', 52, 'H', 'M'),
('Fernanda', 'Vazquez', 'Rojas', 'olivos', 'santa maria', 'iztapalapa', 22,7,5618326017,
'VZRF940505MB4', 27000, 'soltero', 29, 'O', 'M'),
('Eduardo', 'Juarez', 'Lopez', 'romero', 'el vergel', 'iztacalco', 12,3,5618326018,
'JULE840910HG8', 45000, 'viudo', 39, 'H', 'M'),
('Sandra', 'Herrera', 'Diaz', 'rosales', 'el paraiso', 'xochimilco', 19,2,5618326019,
'HEDS810208M45', 35000, 'casado', 42, 'M', 'M'),
('Roberto', 'Guerrero', 'Mendez', 'girasoles', 'la noria', 'magdalena C.', 15,4,5618326020,
'GUMR750615H50', 38000, 'divorciado', 48, 'H', 'M'),
('Veronica', 'Cruz', 'Salinas', 'azaleas', 'los girasoles', 'milpa alta', 25,6,5618326021,
'CRSV870805M47', 30000, 'soltero', 36, 'M', 'G'),
('Jose', 'Morales', 'Gonzalez', 'sauces', 'san jose', 'miguel hidalgo', 11,1,5618326022,
'MOGJ730920H28', 33000, 'casado', 50, 'O', 'G'),
('Alejandra', 'Duran', 'Barrera', 'limoneros', 'el laurel', 'venustiano c.', 8,3,5618326023,
'DUBA820530M49', 28000, 'divorciado', 41, 'M', 'G'),
('David', 'Zamora', 'Torres', 'manzanos', 'los cedros', 'cuajimalpa', 33,5,5618326024,
'ZATD770810H35', 40000, 'soltero', 46, 'H', 'G'),
('Lorena', 'Vega', 'Santos', 'higueras', 'san angel', 'tlahuac', 7,2,5618326025, 'VESL900515M48',
32000, 'casado', 33, 'M', 'G'),
('Hector', 'Delgado', 'Castillo', 'laureles', 'san bernabe', 'tlalpan', 20,8,5618326026,
'DECH810425H21', 36000, 'viudo', 42, 'H', 'A'),
('Adriana', 'Lara', 'Navarro', 'pino', 'la concepcion', 'coyoacan', 23,6,5618326027,
'LANA900715M46', 37000, 'casado', 33, 'M', 'A'),
('Gabriel', 'Ortiz', 'Romero', 'abedul', 'el bosque', 'azcapotzalco', 9,4,5618326028,
'ORRG820610H39', 34000, 'divorciado', 41, 'O', 'A'),
```

```
('Patricia', 'Perez', 'Ramos', 'cerezos', 'san pablo', 'benito juarez', 27,2,5618326029,
'PERR850825M37', 33000, 'soltero', 38, 'M', 'A'),
('Jorge', 'Aguilar', 'Ponce', 'encinos', 'santa rosa', 'alvaro O.', 14,6,5618326030,
'AGPJ790310H23', 39000, 'casado', 44, 'H', 'A'),
('Yolanda', 'Trejo', 'Alvarez', 'olmos', 'el parque', 'cuauhtemoc', 19,3,5618326031,
'TRAY840705M43', 28000, 'divorciado', 39, 'M', 'A'),
('Miguel', 'Camacho', 'Hernandez', 'magnolias', 'la hacienda', 'venustiano c', 6,8,5618326032,
'CAMH780210H29', 42000, 'soltero', 45, 'H', 'R'),
('Isabel', 'Guerra', 'Salazar', 'sabinos', 'la pradera', 'gustavo a.', 30,7,5618326033,
'GUSI890620M52', 30000, 'viudo', 34, 'M', 'R'),
('Ricardo', 'Santos', 'Lopez', 'nogales', 'las aguilas', 'iztapalapa', 15,5,5618326034,
'SALR810920H38', 40000, 'soltero', 42, 'H', 'R'),
('Rebeca', 'Molina', 'Campos', 'acacias', 'san miguel', 'iztacalco', 22,9,5618326035,
'MOCR850305M39', 35000, 'casado', 38, 'M', 'R');

/*
  INSERCIÓN DE idioma
  pk clustered
*/

INSERT INTO empleados.idioma ( idioma)
VALUES ( 'Español'),
       ( 'Frances'),
       ( 'Italiano'),
       ( 'Chino'),
        ( 'Aleman');

/*
  INSERSIÓN DE EMPLEADO_IDIOMA
  ID DEL IDIOMA Y ID DEL EMPLEADO
*/

INSERT INTO EMPLEADOS.EMPLEADO_IDIOMA (id_empleado, id_idioma)
VALUES
       (1, 1),(1, 2),(1, 5),
       (2,3), (2,1),
       (3,5), (3,1),
       (4, 1), (4, 2),
       (5, 1), (5, 3),
       (6, 1), (6, 4),
       (7, 1), (7, 2), (7, 5),
       (8, 1), (8, 3),
       (9, 1), (9, 2),(9, 4),
       (10, 1), (10, 3),(10, 5),
       (11, 1), (11, 4),
       (12, 1), (12, 3),(12, 5),
       (13, 1), (13, 2),
       (14, 1), (14, 4),
       (15, 1), (15, 3),(15, 5),
       (16, 1), (16, 2),
       (17, 1), (17, 4),
       (18, 1), (18, 3),
       (19, 1), (19, 5),
```

```
       (20, 1), (20, 2),
       (21, 1), (21, 4);


/*
 Insersión tabla motivo
 --Causa: Enfermedad, accidente, situacion familiar, otros
*/

INSERT INTO EMPLEADOS.motivo(causa)
VALUES ('Enfermedad'),
           ('Accidente'),
       ('Situacion Familiar'),
       ('Otros');
select * from empleados.motivo
order by id_motivo

/*
 inserción historial_falta
  pk compuesta nonclustered
*/

INSERT INTO empleados.historial_falta (id_empleado, id_motivo, fecha)
VALUES
       (1, 1, '2022-01-01'), (1, 3, '2022-05-05'),
       (2, 4, '2022-11-14'), (2,1, '2022-12-12'),
       (3, 1, '2022-02-04'), (3, 2, '2022-06-22'),
       (4, 4, '2022-02-15'), (4, 3, '2022-07-30'), (4, 2, '2022-10-05'),
       (5, 1, '2022-03-12'),
       (6, 3, '2022-04-26'), (6, 1, '2022-08-13'),
       (7, 2, '2022-05-03'), (7, 3, '2022-09-25'), (7, 4, '2022-12-02'),
       (9, 4, '2022-02-24'), (9, 2, '2022-04-18'), (9, 1, '2022-09-01'), (9, 4, '2022-12-10'),
       (10, 2, '2022-03-05'), (10, 1, '2022-08-21'),
       (11, 3, '2022-01-19'), (11, 4, '2022-07-14'), (11, 4, '2022-11-22'),
       (12, 1, '2022-05-10'),
       (14, 3, '2022-02-07'), (14, 1, '2022-07-18'),
       (15, 1, '2022-04-04'), (15, 4, '2022-10-31'), (15, 2, '2022-11-27'),
       (16, 4, '2022-01-13'),
       (17, 1, '2022-02-20'), (17, 3, '2022-05-08'), (17, 3, '2022-08-24'), (17, 4, '2022-11-12'),
       (18, 2, '2022-03-17'), (18, 1, '2022-06-07'),
       (19, 3, '2022-04-12'), (19, 2, '2022-07-27'), (19, 4, '2022-10-13'),
       (21, 4, '2022-01-29'), (21, 2, '2022-03-30'), (21, 2, '2022-06-18'), (21, 3, '2022-09-07'), (21, 2, '2022-11-29');




/*
  Insersión de tabla agente. Id's 7 8 9 10 11
*/

INSERT INTO empleados.agente (id_empleado)
SELECT id_empleado FROM empleados.empleado WHERE tipo_empleado ='G'
```

```
/*
 Inserción tabla administrador
 id_empleado = 12,13,14,15,16,17
*/

INSERT INTO empleados.administrador (id_empleado, descripcionfuncion, tipo_trabajo,
ubicacion) VALUES
(12,'Revisión constante de las estaciones a su cargo y su auditoría' , 'AUDITORIA',
'ECOBICI.0331'),
(13,'Vigilancia en que las actividades de la estación sean efectuadas en tiempo y
forma' , 'REVISOR', 'ECOBICI.0431'),
(14,'Ayuda a las estaciones a su cargo a efectuar las labores para dar un mejor
servicio' , 'APOYO', 'ECOBICI.0443'),
(15,'Fomenta el trabajo colaborativo entre los empleados a partir de metodologías
ágiles' , 'MET. AGIL', 'ECOBICI.0451'),
(16,'Supervisa el trabajo de los empleados y premia el servicio de aquellos que dan el
mejor servicio' , 'SUPERVISOR', 'ECOBICI.553'),
(17,'Comunica las problemáticas con otros administradores para fomentar el trabajo
ágil' , 'COMUNICADOR', 'ECOBICI.0991')

/*
  inserción tabla mantenimiento, id's: 1 2 3 4 5 6
*/

INSERT INTO empleados.mantenimiento (id_empleado, especialidad)
VALUES
       (1,'ING. MECANICO'),
       (2,'ING. AUTOPARTES'),
       (3,'MECÁNICO'),
       (4,'REPARADOR'),
       (5,'SENIOR ING.'),
       (6,'CICLISTA PROF.')

/*
 insersión tabla plan_seguro
 (AÑADIR TRIGGER DESPUÉS PARA QUE NO PUEDA MODIFICARSE YA QUE NO HAY UN CHECK en el
tipo de cobertura)
*/
INSERT INTO usuarios.plan_seguro (tipo_cobertura, id_tipo)
```

```
VALUES
      ('Daños a terceros', 1),
      ('Daños a terceros y compostura bicicleta',2),
      ('Full',3)

/*
      inserción tabla color   pk clustered
*/

INSERT INTO estacion.color (color)
VALUES
       ( 'Rojo'),
       ( 'Azul'),
       ( 'Amarillo');
select * from estacion.color

/*
  inserción tabla estación
  pk clustered
  id_empleado de los de administracion
*/

INSERT INTO estacion.estacion (ubicacion,nombre, id_empleado)
VALUES
      ('Av. Norte ST. 12','Est. 1', 12),
      ('Av. Pedregal. ST. 15','Est. 2', 13),
      ('Av. Luz ST. 05','Est. 3', 14),
      ('Av. Base Sur ST. 01', 'Est. 4',15),
      ('Av. Strokes ST. 03','Est. 5',16)

/*
  Insersión tabla terminal
*/

INSERT INTO estacion.terminal (id_estacion, consecutivo, descripcion)
VALUES
      (1,1,'T. 1.01'), (1,2, 'T. 1.02'),(1,3,'T. 1.03'),
      (2,1,'T. 2.01'), (2,2, 'T. 2.02'),(2,3,'T. 2.03'),
      (3,1,'T. 3.01'), (3,2, 'T. 3.02'),(3,3,'T. 3.03'),
      (4,1,'T. 4.01'), (4,2, 'T. 4.02'),(4,3,'T. 4.03'),
      (5,1,'T. 5.01'), (5,2, 'T. 5.02'),(5,3,'T. 5.03')



/*
 Insersión catalogo de modelo de bicicleta
*/

INSERT INTO estacion.modelo
VALUES
```

```
       (1, 'BICYCLE ROD 26'),
       (2, 'BICYCLE ROD 29'),
       (3,'BICYCLE ROD 28')

/*
  Insersión de bicicleta
  --color con id 1 2 o 3
  --modelo 1 2 o 3
  -- tamaño chica, mediana o grande
  --estado F: funcionamiento, D: dañada o B: baja
*/

INSERT INTO estacion.bicicleta (id_color, id_modelo, nserie,tamaño, estado,
id_estacion)
 VALUES
       (1, 1, 'BYCL-0001', 'Chica', 'F', 1),
       (2, 1, 'BYCL-0002', 'Mediana', 'F', 1),
       (2, 2, 'BYCL-0003', 'Chica', 'F', 1),
       (1, 1, 'BYCL-0004', 'Chica', 'F', 1),
       (2, 2, 'BYCL-0005', 'Mediana', 'F', 1),
       (1, 3, 'BYCL-0006', 'Grande', 'F', 1),
       (1, 3, 'BYCL-0007', 'Mediana', 'D', 1),
       (1, 3, 'BYCL-0008', 'Grande', 'B', 1), --8 bicicletas en estación 1
       (3, 1, 'BYCL-0009', 'Chica', 'F', 2),
       (3, 2, 'BYCL-0010', 'Grande', 'F', 2),
       (1, 2, 'BYCL-0011', 'Mediana', 'F', 2),
       (2, 1, 'BYCL-0012', 'Mediana', 'F', 2),
       (2, 1, 'BYCL-0013', 'Mediana', 'D', 2),
       (1, 3, 'BYCL-0014', 'Mediana', 'F', 2),
       (1, 3, 'BYCL-0015', 'Mediana', 'F', 2),
       (2, 2, 'BYCL-0016', 'Chica', 'B', 2),--8 bicicletas en estacion 2
       (3, 2, 'BYCL-0017', 'Grande', 'F', 3),
       (3, 3, 'BYCL-0018', 'Chica', 'F', 3),
       (1, 1, 'BYCL-0019', 'Mediana', 'F', 3),
       (1, 1, 'BYCL-0020', 'Chica', 'D', 3),
       (1, 3, 'BYCL-0021', 'Mediana', 'F', 3),
       (1, 2, 'BYCL-0022', 'Grande', 'F', 3),
       (2, 3, 'BYCL-0023', 'Mediana', 'F', 3),
       (2, 1, 'BYCL-0024', 'Mediana', 'B', 3),--8 biciletas en estacion 3
       (3, 3, 'BYCL-0025', 'Mediana', 'F', 4),
       (3, 2, 'BYCL-0026', 'Chica', 'F', 4),
       (2, 1, 'BYCL-0027', 'Chica', 'F', 4),
       (1, 1, 'BYCL-0028', 'Grande', 'F', 4),
       (1, 2, 'BYCL-0029', 'Grande', 'F', 4),
       (2, 2, 'BYCL-0030', 'Chica', 'F', 4),
       (3, 3, 'BYCL-0031', 'Chica', 'D', 4),
       (3, 3, 'BYCL-0032', 'Mediana', 'D', 4),--8 bicicletas en estacion 4
       (1, 1, 'BYCL-0033', 'Grande', 'F', 5),
       (1, 1, 'BYCL-0034', 'Mediana', 'F', 5),
       (2, 1, 'BYCL-0035', 'Chica', 'F', 5),
       (3, 2, 'BYCL-0036', 'Grande', 'F', 5),
```

```
	(1, 1, 'BYCL-0037', 'Mediana', 'F', 5),
	(3, 2, 'BYCL-0038', 'Mediana', 'F', 5),
	(2, 3, 'BYCL-0039', 'Mediana', 'F', 5),
	(2, 2, 'BYCL-0040', 'Mediana', 'F', 5)  --8 bicicletas en estación 5
/*
	Inserción bicimantenimiento
	id's empleado (de mantenimiento 1-6)
	servicio: 'REPARACION', 'LIMPIEZA', 'TRANSPORTE'
*/

INSERT INTO estacion.bicimantenimiento (id_empleado,id_bicicleta,fecha, descripcion,
servicio)
VALUES
(1, 5, '2022-04-13', 'Reparación de frenos', 'REPARACIÓN'),
(1, 26, '2022-02-01', 'Cambio de estación', 'TRANSPORTE'),
(5, 16, '2022-03-16', 'Limpieza general', 'LIMPIEZA'),
(3, 4, '2022-07-04', 'Reparación de ruedas', 'REPARACIÓN'),
(2, 38, '2022-03-11', 'Mandada a matenimiento full', 'TRANSPORTE'),
(6, 32, '2022-05-05', 'Limpieza general', 'LIMPIEZA'),
(4, 18, '2022-06-07', 'Reparación de cadena y piñones', 'REPARACIÓN'),
(6, 14, '2022-08-17', NULL, 'REPARACIÓN'),
(1, 14, '2022-04-24', 'Reparación de radios', 'REPARACIÓN'),
(2, 34, '2022-06-16', 'Limpieza general', 'LIMPIEZA'),
(3, 9, '2022-09-09', NULL, 'TRANSPORTE'),
(5, 23, '2022-05-12', 'Limpieza superficial', 'LIMPIEZA'),
(4, 28, '2022-03-21', 'Limpieza profunda', 'LIMPIEZA'),
(1, 33, '2022-02-05', 'Reparación de horquilla', 'REPARACIÓN'),
(3, 14, '2022-07-22', 'Limpieza general', 'LIMPIEZA'),
(2, 7, '2022-05-02', 'Cambio de estación', 'TRANSPORTE'),
(6, 39, '2022-03-19', 'Limpieza de manillar', 'LIMPIEZA'),
(4, 13, '2022-09-14', 'Limpieza sencilla', 'LIMPIEZA'),
(5, 7, '2022-05-18', 'Reemplazo de bicicleta', 'TRANSPORTE'),
(2, 10, '2022-06-10', 'Reparación de frenos', 'LIMPIEZA'),
(1, 5, '2022-07-03',NULL, 'LIMPIEZA');

/*
  Inserción tabla viaje id non clustered
  --atributo de tarifa calculado a partir de la hora de inicio y la hora de llegada
*/

INSERT INTO usuarios.viaje (id_bicicleta, id_usuario, hora_ini, hora_llegada, hora_fin,
fecha, ruta, estacionPartida, estacionLlegada)
VALUES
(1,5,'12:46:00', '13:46:00', '16:00:00', '2022-11-01', 'Paso por reforma, Av. Iman y
llegó a la estación', 1,5),
(27, 1, '09:30:00', '10:40:00', '10:10:00', '2022-01-15', 'Paso por iman a santa ursula
y llegó a la estación', 2, 3),
(15, 2, '14:45:00', '18:20:00', '17:30:00', '2022-03-02', 'Paso por santa lucía y llegó
a cholula y llegó a la estación', 4, 1),
(6, 3, '10:00:00', '11:00:00', '11:30:00', '2022-06-10', 'Av. Iman - Av. LEATHER -
Estacion', 1, 5),
```

```
(12, 4, '12:30:00', '14:00:00', '14:30:00', '2022-07-18', 'Av. Juan - Av. Dog -
Estacion', 3, 2),
(35, 8, '16:15:00', '17:30:00', '18:00:00', '2022-09-05', 'Av. Daniel - Av. Cat -
Estacion', 5, 4),
(8, 10, '11:45:00', '14:50:00', '13:30:00', '2022-11-21', 'Av. Gomez - Av. Sancho -
Estacion', 1, 5),
(19, 9, '15:30:00', '16:30:00', '17:00:00', '2022-02-14', 'Av. Flores - Av. Sol -
Estacion', 2, 3),
(29, 6, '13:00:00', '14:30:00', '15:00:00', '2022-04-29', 'Av. Luna - Av. Estrella -
Estacion', 4, 1),
(11, 7, '09:15:00', '10:00:00', '09:48:00', '2022-08-03', 'Av. Montaña - Av. Río -
Estacion', 3, 2),
(37, 10, '16:45:00', '18:00:00', '17:30:00', '2022-10-12', 'Av. Montaña - Av. Río -
Estacion', 5, 4),
(22, 1, '14:00:00', '15:30:00', '16:00:00', '2022-05-17', 'Av. Juan - Av. Dog -
Estacion', 1, 5),
(23, 1, '11:30:00', '12:30:00', '12:10:00', '2022-07-03', 'Av. Avenida - Av. Boulevard
- Estacion', 2, 3),
(18, 2, '16:30:00', '20:00:00', '19:30:00', '2022-09-20', 'Av. Nube - Av. Viento -
Estacion', 3, 2),
(31, 2, '10:45:00', '12:00:00', '12:30:00', '2022-11-08', 'Av. Puente - Av. Túnel -
Estacion', 4, 1),
(13, 3, '15:15:00', '16:15:00', '16:45:00', '2022-01-25', 'Av. Montaña - Av. Valle -
Estacion', 5, 4),
(5, 5, '12:00:00', '14:55:00', '14:00:00', '2022-03-11', 'Av. Ruta - Av. Sendero -
Estacion', 1, 5),
(36, 7, '14:30:00', '15:30:00', '16:00:00', '2022-06-28', 'Av. Nube - Av. Viento -
Estacion', 2, 3),
(20, 8, '09:45:00', '11:00:00', '10:30:00', '2022-08-15', 'Av. Montaña - Av. Valle -
Estacion', 4, 1),
(10, 10, '13:30:00', '14:45:00', '15:15:00', '2022-10-02', 'Av. Juan - Av. Dog -
Estacion', 3, 2),
(38, 7, '16:00:00', '18:15:00', '17:45:00', '2022-12-09', 'Av. Puente - Av. Túnel -
Estacion', 5, 4),
(7, 9, '10:30:00', '11:30:00', '12:00:00', '2022-03-15', 'Av. Luna - Av. Estrella -
Estacion', 2, 4),
(22, 6, '14:45:00', '16:15:00', '15:50:00', '2022-06-10', 'Av. Montaña - Av. Valle -
Estacion', 3, 1),
(29, 6, '12:15:00', '13:30:00', '14:00:00', '2022-09-27', 'Av. Luna - Av. Estrella -
Estacion', 4, 5),
(12, 4, '16:00:00', '17:30:00', '18:00:00', '2022-11-14', 'Av. Avenida - Av. Boulevard
- Estacion', 5, 3),
(2, 3, '11:00:00', '12:00:00', '12:30:00', '2022-02-03', 'Av. Cielo - Av. Mar -
Estacion', 1, 2),
(24, 8, '15:30:00', '18:45:00', '17:15:00', '2022-04-20', 'Av. Ruta - Av. Sendero -
Estacion', 2, 4),
(34, 9, '14:00:00', '15:00:00', '15:30:00', '2022-07-16', 'Av. Flores - Av. Sol -
Estacion', 3, 1),
(19, 1, '10:30:00', '12:55:00', '12:15:00', '2022-10-23', 'Av. Paso - Av. Cruce -
Estacion', 4, 5),
```

```
(6, 2, '13:15:00', '14:30:00', '15:00:00', '2022-12-01', 'Av. Escalera - Av. Ascensor - Estacion', 5, 3),
(28, 4, '15:45:00', '17:00:00', '16:50:00', '2022-05-08', 'Av. Nube - Av. Viento - Estacion', 1, 2),
(11, 6, '09:30:00', '10:30:00', '11:00:00', '2022-08-05', 'Av. Lago - Av. Río - Estacion', 2, 4),
(31, 7, '13:45:00', '15:00:00', '15:30:00', '2022-01-18', 'Av. Puente - Av. Túnel - Estacion', 3, 1),
(17, 8, '11:15:00', '13:30:00', '13:00:00', '2022-04-03', 'Av. Cielo - Av. Mar - Estacion', 4, 5),
(8, 9, '16:30:00', '19:45:00', '18:15:00', '2022-06-29', 'Av. Paso - Av. Cruce - Estacion', 5, 3),
(35, 10, '12:00:00', '13:00:00', '13:30:00', '2022-09-15', 'Av. Escalera - Av. Ascensor - Estacion', 1, 2),
(22, 8, '15:00:00', '17:15:00', '16:10:00', '2022-11-28', 'Av. Nube - Av. Viento - Estacion', 2, 4),
(39, 5, '14:30:00', '15:30:00', '16:00:00', '2022-03-12', 'Av. Montaña - Av. Valle - Estacion', 3, 1),
(14, 1, '10:45:00', '12:00:00', '12:30:00', '2022-08-19', 'Av. Paso - Av. Cruce - Estacion', 4, 5),
(4, 3, '13:30:00', '14:45:00', '14:30:00', '2022-10-06', 'Av. Flores - Av. Sol - Estacion', 5, 3),
(27, 5, '15:00:00', '16:15:00', '16:45:00', '2022-02-22', 'Av. Ruta - Av. Sendero - Estacion', 1, 2),
(10, 9, '09:45:00', '10:45:00', '10:20:00', '2022-05-17', 'Av. Cielo - Av. Mar - Estacion', 2, 4);

/*
 Inserción incidentes
  'P. BICI','P. COCHE', 'P. MOTO', 'P. Peatón', 'Caída', 'Otros'
*/

INSERT INTO usuarios.incidente (id_empleado, tipo_incidente, hora, coordenadas, calle, numero, codigoPostal, alcaldia, colonia, descripcion, id_viaje)
VALUES
(10, 'P. BICI', '17:12:00', '124.235.222', 'Av. San Ricardo', 35, '04670', 'Coyoacan', 'Pedegal Ricardo', 'Me caí de la bicicleta al chocar con un peatón', 1),
(9, 'P. COCHE', '19:00:00', '144.531.456', 'Av. San Pepe', 35, '65864', 'Magdalena', 'Pedegal Pepe', 'Choqué con otra bicicleta bruscamente', 5),
(8, 'P. Peatón', '10:30:00', '176.892.124', 'Av. San Juan', 12, '04580', 'Benito Juarez', 'Pedegal Juan', 'Casi atropello a un peatón', 15),
(10, 'P. MOTO', '14:45:00', '123.456.789', 'Av. San Pedro', 45, '06430', 'Cuauhtemoc', 'Pedegal Pedro', 'Dañé una bicicleta estacionada', 16),
(11, 'Caída', '16:20:00', '987.654.321', 'Av. San Lucas', 28, '03240', 'Iztacalco', 'Pedegal Lucas', 'Fui golpeado por un automóvil', 3),
(7, 'P. MOTO', '09:15:00', '789.123.456', 'Av. San Ignacio', 56, '05670', 'Gustavo A. Madero', 'Pedegal Ignacio', 'Me robaron la bicicleta', 25), (9, 'P. BICI', '12:40:00', '555.666.777', 'Av. San Roberto', 9, '04980', 'Tlahuac', 'Pedegal Roberto', 'Perdí mi tarjeta de acceso', 8),
(8, 'P. BICI', '18:30:00', '777.888.999', 'Av. San Martín', 76, '07740', 'Venustiano C.', 'Pedegal Martín', 'Mi bicicleta se descompuso', 12),
(10, 'P. BICI', '21:50:00', '999.888.777', 'Av. San Andrés', 18, '03100', 'Azcapotzalco', 'Pedegal Andrés', 'Me quedé sin batería en la tarjeta', 17),
```

```
(11, 'P. COCHE', '15:10:00', '111.222.333', 'Av. San Miguel', 64, '04700', 'Alvaro Obregon',
'Pedegal Miguel', 'Nada', 30),
(7, 'Otros', '13:20:00', '222.333.444', 'Av. San Manuel', 29, '06880', 'Iztapalapa', 'Pedegal
Manuel', 'Mmmmmmmmmm', 19),
(8, 'Otros', '11:55:00', '333.444.555', 'Av. San Diego', 78, '06100', 'Miguel Hidalgo', 'Pedegal
Diego', 'Se me pinchó una rueda', 38),
(9, 'P. BICI', '19:40:00', '444.555.666', 'Av. San Felipe', 51, '04780', 'Cuajimalpa', 'Pedegal
Felipe', 'Casi me caigo al pasar un tope', 20),
(11, 'Otros', '16:15:00', '666.555.444', 'Av. San Luis', 43, '04230', 'Xochimilco', 'Pedegal
Luis', 'Perdí mi casco durante el viaje', 7),
(7, 'P. BICI', '16:30:00', '777.888.999', 'Av. San Rafael', 18, '04280', 'Alvaro Obregon',
'Pedegal Rafael', 'Se me salió la cadena de la bicicleta', 2),
(9, 'P. MOTO', '10:00:00', '555.666.777', 'Av. San Ignacio', 35, '04940', 'Coyoacan', 'Pedegal
Ignacio', 'Perdí mi tarjeta de acceso en el viaje', 6),
(8, 'P. Peatón', '13:45:00', '444.555.666', 'Av. San Roberto', 27, '06120', 'Magdalena', 'Pedegal
Roberto', 'Tuve una caída por un bache en el camino', 18),
(10, 'P. BICI', '18:20:00', '666.555.444', 'Av. San Manuel', 14, '04370', 'Benito Juarez',
'Pedegal Manuel', 'Me robaron el celular mientras iba en la bicicleta', 4),
(11, 'P. BICI', '12:15:00', '888.999.000', 'Av. San EXCELSO', 62, '04760', 'Cuajimalpa', 'Pedegal
EXCELSO', 'Me encontré con un obstáculo en el carril bici', 39),
(7, 'P. BICI', '15:50:00', '999.000.111', 'Av. San Macario', 45, '04050', 'Iztacalco', 'Pedegal
Macario', 'Perdí mi tarjeta de acceso durante el viaje', 40)

/*
      Inserts tabla historial viaje
*/
INSERT INTO usuarios.historial_viaje
VALUES
   (1,124.3122, 125.3312), (1, 504.1445, 391.0122),
    (2, 223.3122, 126.3312),(2, 156.3, 585.0),
    (3, 333.1445, 391.0122),(3, 555.2121, 422.102),
    (4, 435.3111, 443.2122), (4, 212.341, 442.202),
    (5, 553.1245, 543.2221),(5, 654.1, 543.1),
    (6, 665.4444, 212.1243),(6, 325.4444, 212.12),
    (7, 767.2321, 655.211),(7, 871.2321, 615.31),
    (8, 887.1234, 777.555),(8, 312.1234, 127.5),
    (9, 988.2121, 253.4332),(9, 621.2121, 254.4332),
    (10, 1000.5432, 221.2221),(10, 432.5432, 112.1),
    (11, 1121.0000, 568.7777),
    (12, 1223.1231, 377.2345), (12, 999.0000, 888.74),
    (13, 1344.6543, 543.1254),(13, 432.6543, 543.1234),
    (14, 1432.5432, 33.9856),(14, 876.5432, 345.9876),
    (15, 1577.2222, 3.4444),(15, 111.24, 334.144),
    (16, 1665.5555, 55.7777), (16, 444.5555, 6.77),
    (17, 1788.9999, 566.4444),(17, 888.949, 555.44),
    (18, 1876.4567, 33.4321),(18, 123.427, 765.4321),
    (19, 1902.6666, 2552.33),(19, 767.64, 222.3333),
    (20, 2012.4444, 866.9999),(20, 555.4244, 778.99),
    (21, 2144.1234, 455.6543),(21, 321.1234, 432.6543),
    (22, 2243.1231, 312.2345),(22, 212.1231, 321.2345),
    (23, 2355.6543, 512.1234),(23, 432.643, 543.1234),
    (24, 2407.5432, 32.9876),(24, 876.5232, 345.9876),
    (25, 2576.2222, 313.444), (25, 111.2, 333.44),
    (26, 2676.5555, 6446.557), (26, 42.4, 626.77),
    (27, 2765.9999, 5522.4),
    (28, 2894.4567, 735.21),(28, 123.4567, 765.4321),
    (29, 2943.666, 2232.33),
```

```
    (30, 3012.4444, 828.929),(30, 9.4444, 88.999),
    (31, 3153.1234, 1.6343),(31, 45.444, 8.999),
    (32, 3245.1231, 43.2345),(32, 212.1231, 321.245),
    (33, 3345.6543, 31.134),(33, 432.6543, 543.124),
    (34, 3454.5432, 344.9876),
    (35, 3542.2222, 343.4444),
    (36, 3644.5555, 12.7777),
    (37, 3786.9999, 5.4444),
    (38, 3843.4567, 3.4321),(38, 123.567, 76.431),
    (39, 3966.6666, 44.3333),
    (40, 4012.4444, 6.9999),
    (41, 1234.44, 74.6543),(42, 5.1234, 2.6543),(42, 31.34, 42.63)

------------------------------------------------------------
-- Poniendo supervisores en empleado

SELECT * FROM empleados.empleado
UPDATE empleados.empleado
SET id_supervisor = 18
WHERE id_empleado IN (2,4,6,8,10)

UPDATE empleados.empleado
SET id_supervisor = 21
WHERE id_empleado IN (1,3,5,7,9)

UPDATE empleados.empleado
SET id_supervisor = 15
WHERE id_empleado IN (11,12,20)

UPDATE empleados.empleado
SET id_supervisor = 13
WHERE id_empleado IN (14,16,17,19)

------------------------------------------------------------
-- SELECT'S
SELECT * FROM empleados.administrador
SELECT * FROM empleados.agente
SELECT * FROM empleados.empleado
SELECT * FROM empleados.EMPLEADO_IDIOMA
SELECT * FROM empleados.historial_falta
SELECT * FROM empleados.idioma
SELECT * FROM empleados.mantenimiento
SELECT * FROM empleados.motivo

SELECT COUNT(*) FROM empleados.administrador
SELECT COUNT(*) FROM empleados.agente
SELECT COUNT(*) FROM empleados.empleado
SELECT COUNT(*) FROM empleados.EMPLEADO_IDIOMA
SELECT COUNT(*) FROM empleados.historial_falta
SELECT COUNT(*) FROM empleados.idioma
SELECT COUNT(*) FROM empleados.mantenimiento
SELECT COUNT(*) FROM empleados.motivo
------------------------------------------------------------
```

```
SELECT * FROM estacion.bicicleta
SELECT * FROM estacion.bicimantenimiento
SELECT * FROM estacion.color
SELECT * FROM estacion.estacion
SELECT * from estacion.terminal
SELECT * FROM estacion.modelo
SELECT COUNT(*) FROM estacion.bicicleta
SELECT COUNT(*) FROM estacion.bicimantenimiento
SELECT COUNT(*) FROM estacion.color
SELECT COUNT(*) FROM estacion.estacion
SELECT COUNT(*) from estacion.terminal
SELECT COUNT(*) FROM estacion.modelo
-------------------------------------------------------

SELECT * FROM usuarios.historial_viaje
SELECT * FROM usuarios.incidente
SELECT * FROM usuarios.membresia
SELECT * FROM usuarios.metodo_pago
SELECT * FROM usuarios.plan_seguro
SELECT * FROM usuarios.telefono
SELECT * FROM usuarios.tipo_membresia
SELECT * FROM usuarios.USUARIO
SELECT * FROM usuarios.viaje

SELECT count(*) FROM usuarios.historial_viaje
SELECT count(*) FROM usuarios.incidente
SELECT count(*) FROM usuarios.membresia
SELECT count(*) FROM usuarios.metodo_pago
SELECT count(*) FROM usuarios.plan_seguro
SELECT count(*) FROM usuarios.telefono
SELECT count(*) FROM usuarios.tipo_membresia
SELECT count(*) FROM usuarios.USUARIO
SELECT count(*) FROM usuarios.viaje


INSERT INTO tipo_incidente VALUES
       (1, 'P. BICI', 'Problema al chocar con otra bicicleta'),
       (2, 'P. Coche', 'Problema al chocar con un coche'),
       (3, 'P. Moto', 'Problema al chocar con una moto'),
       (4, 'Caída', 'Caída debido a problemas externos'),
       (5, 'Otros', 'Problema sin especificaciones'),
       (6, 'P. PEATÓN', 'Problema al chocar/toparse con un peatón')
```

7.6 SCRIPT valida_Triggers.sql

TRIGGER 1. Integridad en la jerarquía de tipos en la tabla de Agente: Para este ejercicio, se optó por desarrollar un trigger que al insertar en la tabla de Agente, valide que no esté en la tabla de Mantenimiento, Administración. Además, se verifica que en caso de no estar en las dos tablas anteriores, tampoco exista el empleado en el supertipo con el atributo de tipo de empleado en 'R', pues indicaría que es un empleado de Recursos Humanos. Es decir, para los 3 casos no se permitiría la inserción. Solamente se permitiría la inserción en caso de que ya exista el empleado con el tipo de empleado 'G' (que lo designamos para el agente).

```
/*-----------------------------------TRIGGER 1---------------------------------
Autor: Karla Velázquez
Fecha de creación: 11 / 06 / 2023
Descripcion: Trigger para validar la integridad de la jerarquía de tipos en la tabla de
agente.
Verifica que no exista en la tabla de Mantenimiento ni Administrador. Ademas, valida
que tampoco exista en la tabla de empleado con el tipo 'R' (recursos humanos)
-----------------------------------------------------------------------------------*/

CREATE OR ALTER TRIGGER empleados.tr_usuarios
ON empleados.agente
INSTEAD OF INSERT
AS
BEGIN
   IF EXISTS (SELECT 1 FROM empleados.mantenimiento WHERE id_empleado = (SELECT id_empleado FROM
inserted))
       BEGIN
         PRINT 'No es posible realizar Insersión, existe en la tabla de mantenimiento'
         RETURN
       END
   IF EXISTS (SELECT 1 FROM empleados.administrador WHERE id_empleado = (SELECT id_empleado FROM
inserted))
       BEGIN
         PRINT 'No es posible realizar Insersión, existe en la tabla de administrador'
         RETURN
       END
IF EXISTS (SELECT 1 FROM empleados.empleado e inner join inserted i on e.id_empleado =
i.id_empleado where e.tipo_empleado = 'R')
       BEGIN
          PRINT 'No es posible realizar la insersión, es un empleado de recursos humanos'
          RETURN
       END
IF EXISTS (SELECT 1 FROM empleados.empleado e inner join inserted i on e.id_empleado =
i.id_empleado where e.tipo_empleado ='G')
       BEGIN
           INSERT INTO empleados.agente (id_empleado)
           SELECT id_empleado FROM inserted
       END
       ELSE
       PRINT 'NO EXISTE EL ID PREVIAMENTE EN EMPLEADO o EXISTE PERO NO TIENE EL TIPO AGENTE (G)'
END
--- FIN TRIGGER
```

```
SELECT * FROM empleados.empleado

--Probando trigger con alguien de mantenimiento
INSERT empleados.agente (id_empleado) VALUES (1)

--Probando trigger con alguien de recursos humanos
INSERT empleados.agente (id_empleado) VALUES (21)

--Probando trigger con alguien de administracion
INSERT empleados.agente (id_empleado) VALUES (13)

--Probando trigger
--Se tiene que agregar primero un empleado ya que si se agrega en agente directamnete
--saltará el error de la llave foránea
dbcc checkident('empleados.empleado',reseed,21)
 --ejecutar esto despues de haber ejecutado el begin-rollback para
--restablecer identity en su valor original
BEGIN TRAN

INSERT INTO empleados.empleado
(id_supervisor, tipo_empleado, genero, nombre, paterno, materno, calle, colonia, alcaldia,
num_Ext, num_int, telefono, rfc, sueldo,estado_civil, edad)
VALUES
(21,'G','H','Jose','Alcaraz','Gonzales','Calle josé','el rosario','coyoacan',21,100,'5610295444',
'AUMS4000DFGG', 45000,'soltero', 25)

INSERT into empleados.agente(id_empleado) VALUES (22)

SELECT 'EMPLEADO AGREGADO'
SELECT TOP 1 * FROM empleados.empleado order by id_empleado desc
SELECT 'AGENTE AGREGADO'
SELECT TOP 1 *  FROM empleados.agente order by id_empleado desc

ROLLBACK TRAN
```

Se procede a hacer las 4 pruebas, añadiendo un agente que primero está en Mantenimiento, luego en Recursos Humanos, Administración y luego que sí se permita la inserción si todo está en orden:

```
35  --Probando trigger con alguien de mantenimiento
36  INSERT empleados.agente (id_empleado) VALUES (1)
37
```

160 %

Messages

```
No es posible realizar Insersión, existe en la tabla de mantenimiento

(1 row affected)

Completion time: 2023-06-12T14:17:32.2783549-06:00
```

Sin inserción por existir en la tabla de mantenimiento

```
38  --Probando trigger con alguien de recursos humanos
39  INSERT empleados.agente (id_empleado) VALUES (21)
```

160 %

Messages

```
No es posible realizar la insersión, es un empleado de recursos humanos

(1 row affected)

Completion time: 2023-06-12T14:18:21.4101744-06:00
```

Sin inserción por ser empleado de recursos humanos

```
41  --Probando trigger con alguien de administracion
42  INSERT empleados.agente (id_empleado) VALUES (13)
43
```

%

Messages

```
No es posible realizar Insersión, existe en la tabla de administrador

(1 row affected)

Completion time: 2023-06-12T14:18:57.7326326-06:00
```

Sin inserción por existir en la tabla de administracion

	(No column name)
1	EMPLEADO AGREGADO

	id_empleado	id_supervisor	tipo_empleado	genero	nombre	paterno	materno	calle	colonia	alcaldia	num_ext	num_int	telefono	rfc	sueldo	estado_civil	edad
1	22	21	G	H	Jose	Alcaraz	Gonzales	Calle josé	el rosario	coyoacan	21	100	5610295444	AUMS4000DFGG	45000.00	soltero	25

	(No column name)
1	AGENTE AGREGADO

	id_empleado
1	22

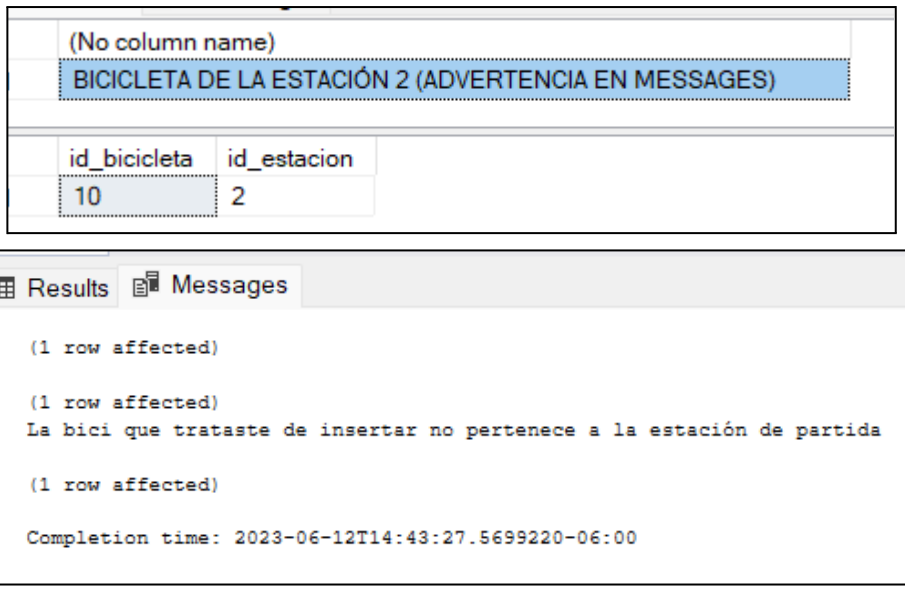
Inserción exitosa dado que existe el empleado y tiene tipo_empleado = 'G'

TRIGGER 2. Trigger para validar que al insertar un viaje,haya coherencia en la información ingresada. Es decir, para un viaje dado, verifica que al ingresar el id de la bicicleta cuya estación de partida es X y la estación de llegada es Y, se revise si dicha bicicleta a insertar efectivamente su estación en la que está es la estación X. En caso de no estarlo no se permite la inserción. Por el contrario, si la información del viaje se ingresa adecuadamente, entonces en la tabla de bicicleta se cambia el id de la estación en la que se encuentra por la estación de llegada Y de dicho viaje.

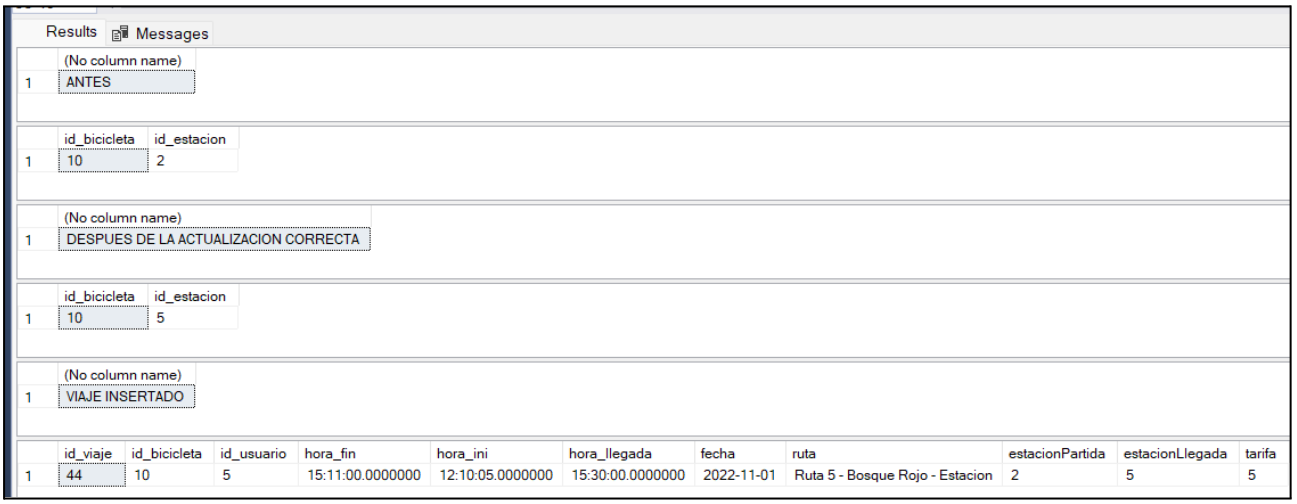
```
/*--------------------------------TRIGGER 2---------------------------------------
  Autor: Daniel Aguilar
  Fecha de creación: 12 / 06 / 2023

Descripcion: Trigger para validar que al insertar un viaje,haya coherencia en la información ingresada. Es decir, para un viaje dado, verifica que al ingresar el id de la bicicleta cuya estación de partida es X y la estación de llegada es Y, se revise si dicha bicicleta a insertar efectivamente su estación en la que está es la estación X. En caso de no estarlo no se permite la inserción. Por el contrario, si la información del viaje se ingresa adecuadamente, entonces en la tabla de bicicleta se cambia el id de la estación en la que se encuentra por la estación de llegada Y de dicho viaje.
-------------------------------------------------------------------------------*/

GO
CREATE OR ALTER TRIGGER usuarios.tr_InsertarViaje
ON usuarios.viaje
INSTEAD OF INSERT
AS
BEGIN

IF EXISTS(SELECT 1  FROM estacion.bicicleta b inner join inserted i on b.id_bicicleta =
i.id_bicicleta where b.id_estacion = i.estacionPartida)
   BEGIN
        PRINT 'La bicicleta coincide con la estación de partida. Se ha efectuado el insert exitosamente'

--ACTUALIZAMOS LA ESTACION EN DONDE ESTÁ LA BICICLETA
UPDATE estacion.bicicleta
SET id_estacion = (SELECT estacionLlegada FROM inserted)
WHERE id_bicicleta = (SELECT id_bicicleta FROM inserted)

--SE INSERTA EL VIAJE EXITOSAMENTE
 INSERT INTO viaje (id_bicicleta, id_usuario, hora_fin, hora_ini, hora_llegada, fecha, ruta,
estacionPartida, estacionLlegada)
 SELECT id_bicicleta, id_usuario, hora_fin, hora_ini, hora_llegada, fecha, ruta, estacionPartida,
estacionLlegada
 FROM inserted
   END
ELSE
   BEGIN
--NO SE REALIZA LA INSERCIÓN DEL VIAJE
PRINT 'La bici que trataste de insertar no pertenece a la estación de partida'
RETURN
END

END
--- FIN TRIGGER

--PRUEBA CON INFORMACIÓN ÉRRONEA
BEGIN TRAN
--Probamos con una bicicleta cuya estación de partida no concuerda con el lugar en donde se
encuentra la bici
-- en ese momento

--La bicicleta 10 está en la estación 2
SELECT 'BICICLETA DE LA ESTACIÓN 2 (ADVERTENCIA EN MESSAGES)'
SELECT  id_bicicleta, id_estacion FROM estacion.bicicleta WHERE id_bicicleta = 10
```

```
--Manda mensaje ya que la bici 10 no está en la estación 3
INSERT INTO usuarios.viaje (id_bicicleta, id_usuario, hora_fin, hora_ini, hora_llegada, fecha,
ruta, estacionPartida, estacionLlegada)
VALUES
(10, 5, '15:11:00', '12:10:05', '15:30:00', '2022-11-01', 'Ruta 5 - Bosque Rojo - Estacion' , 3, 5)
ROLLBACK TRAN

--PRUEBA CON INFORMACIÓN CORRECTA
BEGIN TRAN
	SELECT 'ANTES'
	SELECT  id_bicicleta, id_estacion FROM estacion.bicicleta WHERE id_bicicleta = 10

--Insert correcto
	SELECT 'DESPUES DE LA ACTUALIZACION CORRECTA'

INSERT INTO usuarios.viaje (id_bicicleta, id_usuario, hora_fin, hora_ini, hora_llegada, fecha,
ruta, estacionPartida, estacionLlegada)
VALUES
(10, 5, '15:11:00', '12:10:05', '15:30:00', '2022-11-01', 'Ruta 5 - Bosque Rojo - Estacion' , 2, 5)

	SELECT  id_bicicleta, id_estacion FROM estacion.bicicleta WHERE id_bicicleta = 10
	SELECT 'VIAJE INSERTADO'
	SELECT TOP 1 * FROM usuarios.viaje ORDER BY id_viaje DESC

ROLLBACK TRAN

DBCC CHECKIDENT('usuarios.viaje',reseed, 43)
GO
--Deshabilitar un trigger:
DISABLE TRIGGER tr_InsertarViaje on usuarios.viaje
GO
--Habilitar un trigger:
ENABLE TRIGGER tr_InsertarViaje on usuarios.viaje
```

Se procede a hacer las dos pruebas, una inserción en donde en el viaje, la bicicleta parte de una estación en la que no se encuentra en ese momento. Luego, un caso donde sí exista la bicicleta indicada en la estación correspondiente para poder ingresar el viaje y con ello, cambiar el id de la estación en la tabla de bicicleta, para la bicicleta que efectuó el viaje.



Inserción fallida: Se intentó ingresar un viaje de la bicicleta 10 yendo de la estación 3 a la estación 5. Sin embargo, la estación 10 está en la estación 2 en ese momento, no en la 3.

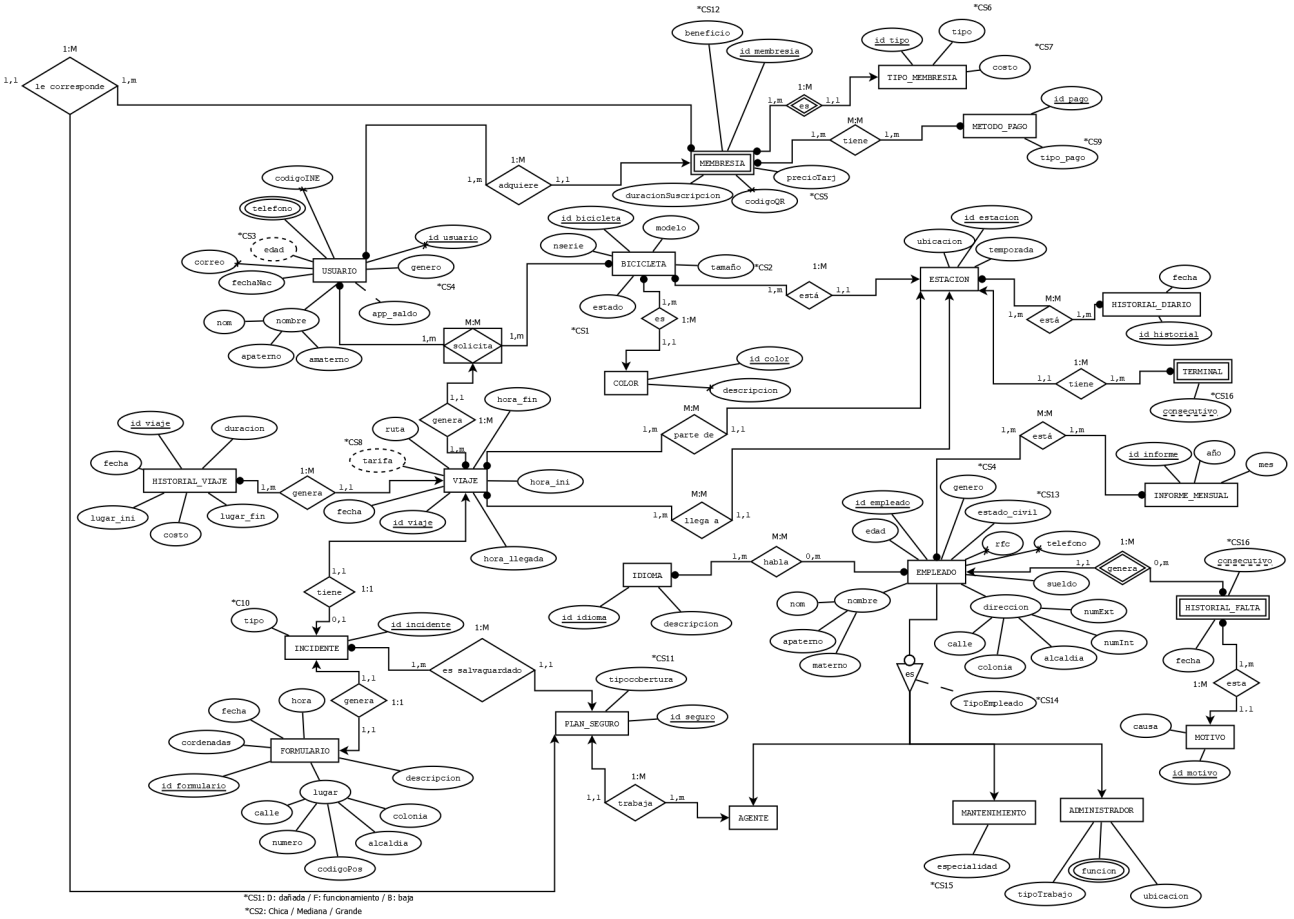


Inserción con éxito. Se cambia el lugar en el que está la bicicleta 10 (id_estacion = 5) y muestra el viaje ingresado con éxito.

NOMBRE	ARCHIVO	OBSERVCIONES
DCL	seguridad.sql	*Se tienen la creación de usuarios por default y los procedimientos almacenados para la creación de usuarios*
DDL	creaBase.sql	*Se cuenta con tablas con índices clustered y non-clostered*
DML	dml.sql	*Incluye* *2 triggers llamados*

		3 vistas *4 procedimientos*
ESTADÍSTICAS	informes.sql	*Se crearon 13 estadísticas*
CARGA DE INFORMACIÓN	cargaInicial.sql.	*Se llenaron todas las tablas, el documento puede leerse como se informa en el archivo*
TRIGGERS	validaTriggers.sql	*se usaron dos triggers llamados “`usuarios.tr_InsertarViaje`“ y `empleados.tr_usuarios`*

VII ANEXOS



*CS1: D: dañada / F: funcionamiento / B: baja

*CS2: Chica / Mediana / Grande

*CS3: (FechaActual - FechaNacimiento) % 365

*CS4: Hombre / Mujer / Otros

*CS5: $50 la primera vez / $80 para reposición

*CS6: B: Basica / I: Intermedia / P: Premium

*CS7: Si tipo = 'B' entonces costo = $118 por día
Si tipo = 'I' entonces costo = $400 semanalmente
Si tipo = 'P' entonces costo = $1000 anualmente

*CS8: Si hora_llegada < hora_fin entonces Tarifa = 0
Si no:
Tarifa = 5 x, donde x = hora_llegada - hora_fin % 10
Abonada el mismo dia para Membresia básica
Abonada el siguiente mes para Membresia intermedia
Abonada el siguiente año para Membresia anual

*CS9: Tarjeta de Crédito / Tarjeta de Débito / Paypal

*C10: Problemas con:
Bicicleta / Coche / Moto / Choche con otra bici / Peatón
Caida de bicicleta / Otros

*CS11: Daños a terceros si tipoMembresia = Básico
Daños a terceros y compostura bicicleta si tipoMembresia = Intermedio
Full si tipoMembresia = Premium

*CS12: Si tipoMembresia = basica , descuento al tiempo suscrito
Si tipoMembresia = intermedia, viaje gratis cada cierto tiempo
Si tipoMembresia = premium, Cashback para el usuario

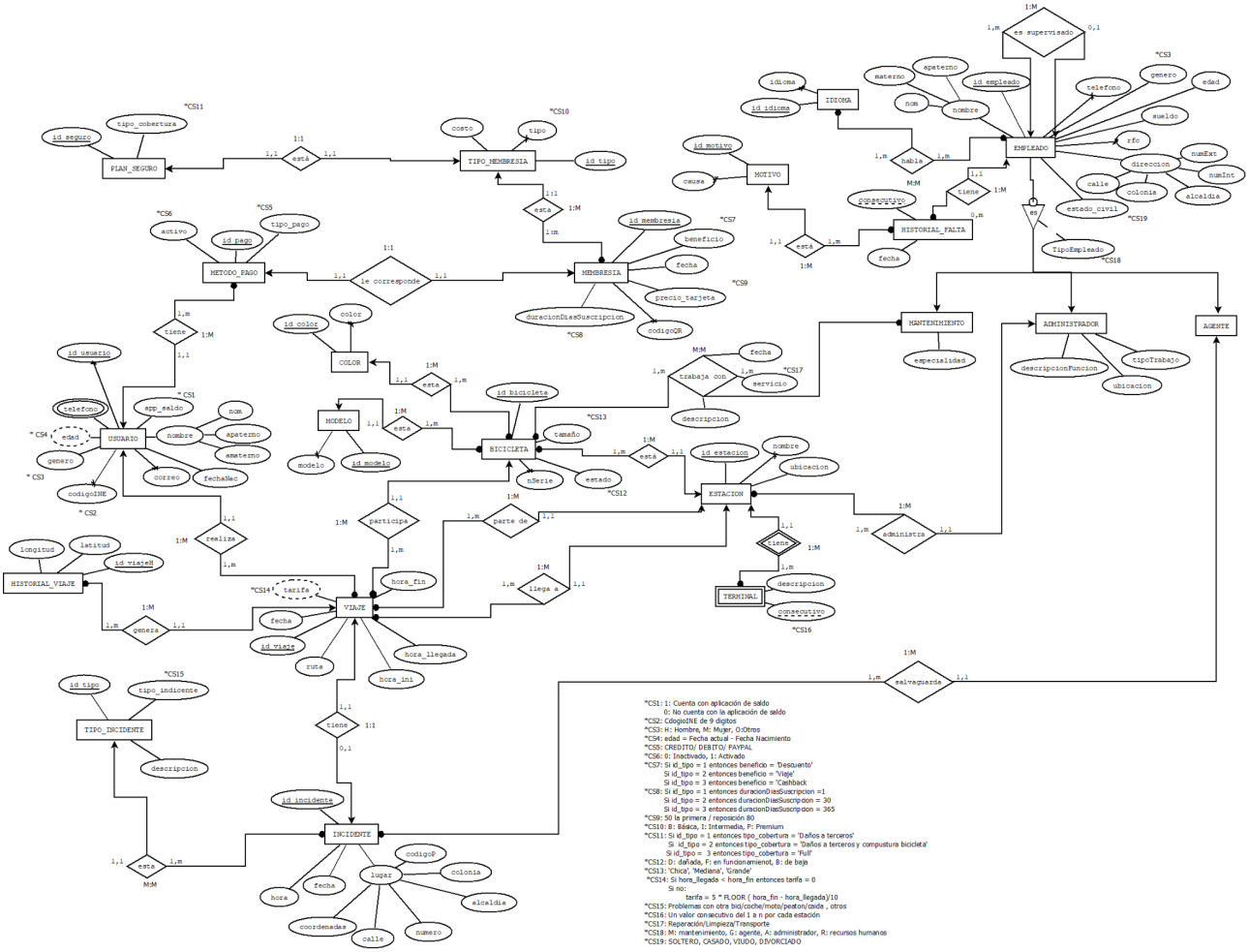
*CS13: Soltero / Casado / Divorciado / Viudo

*CS14: M: mantenimiento, G: Agente, A: Administrador, R: Recursos Humanos

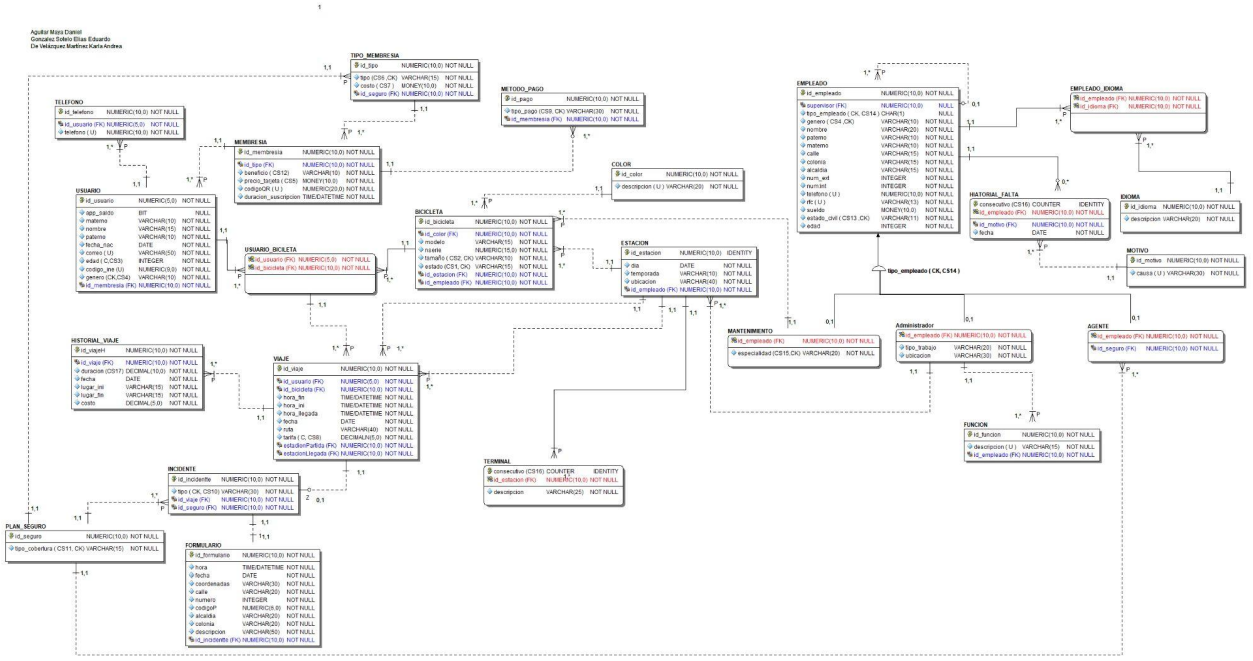
*CS15: Reparación / Limpieza / Transporte

*CS16: Consecutivo reiniciado para cada empleado

VERSION 1 DEL MODELO CONCEPTUAL

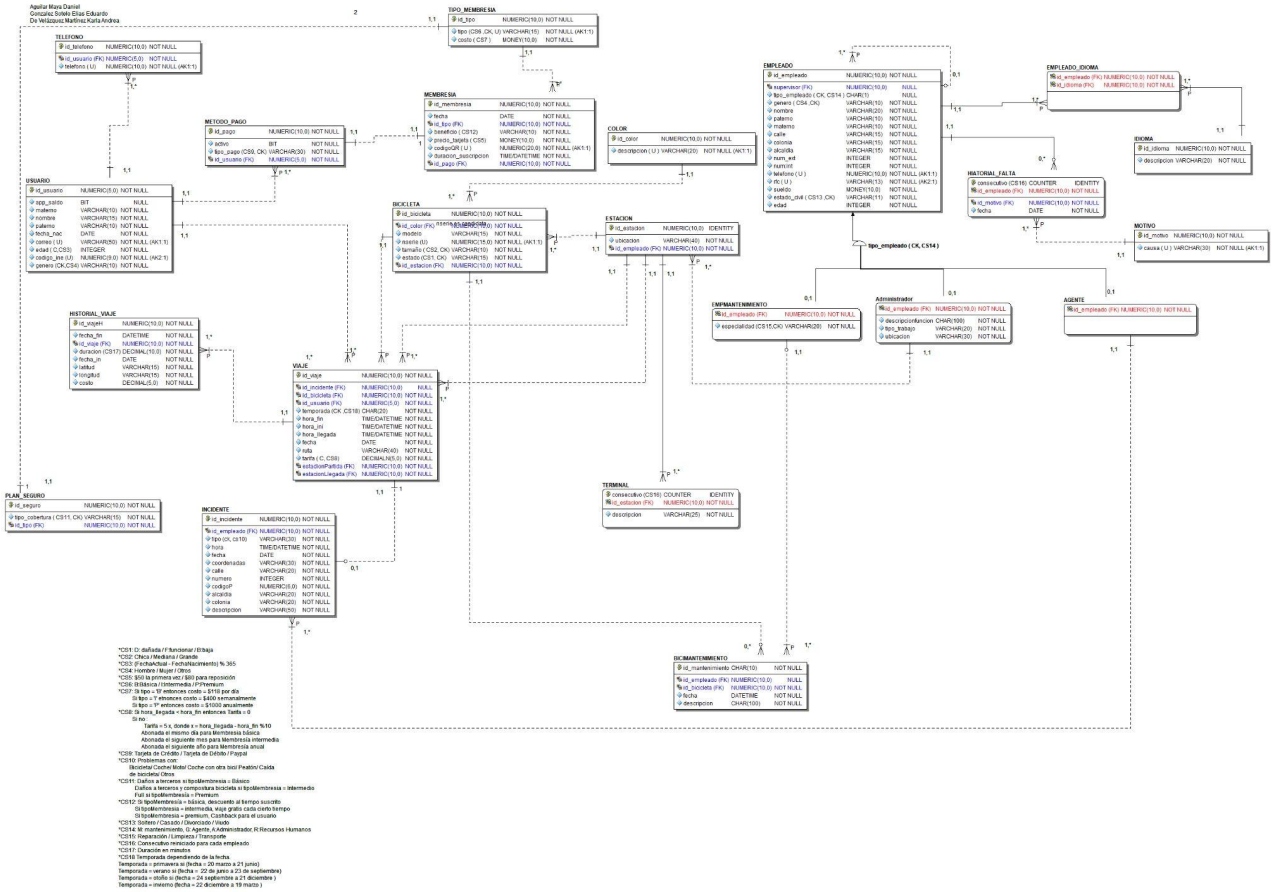


VERSION 1.1 MODELO CONCEPTUAL

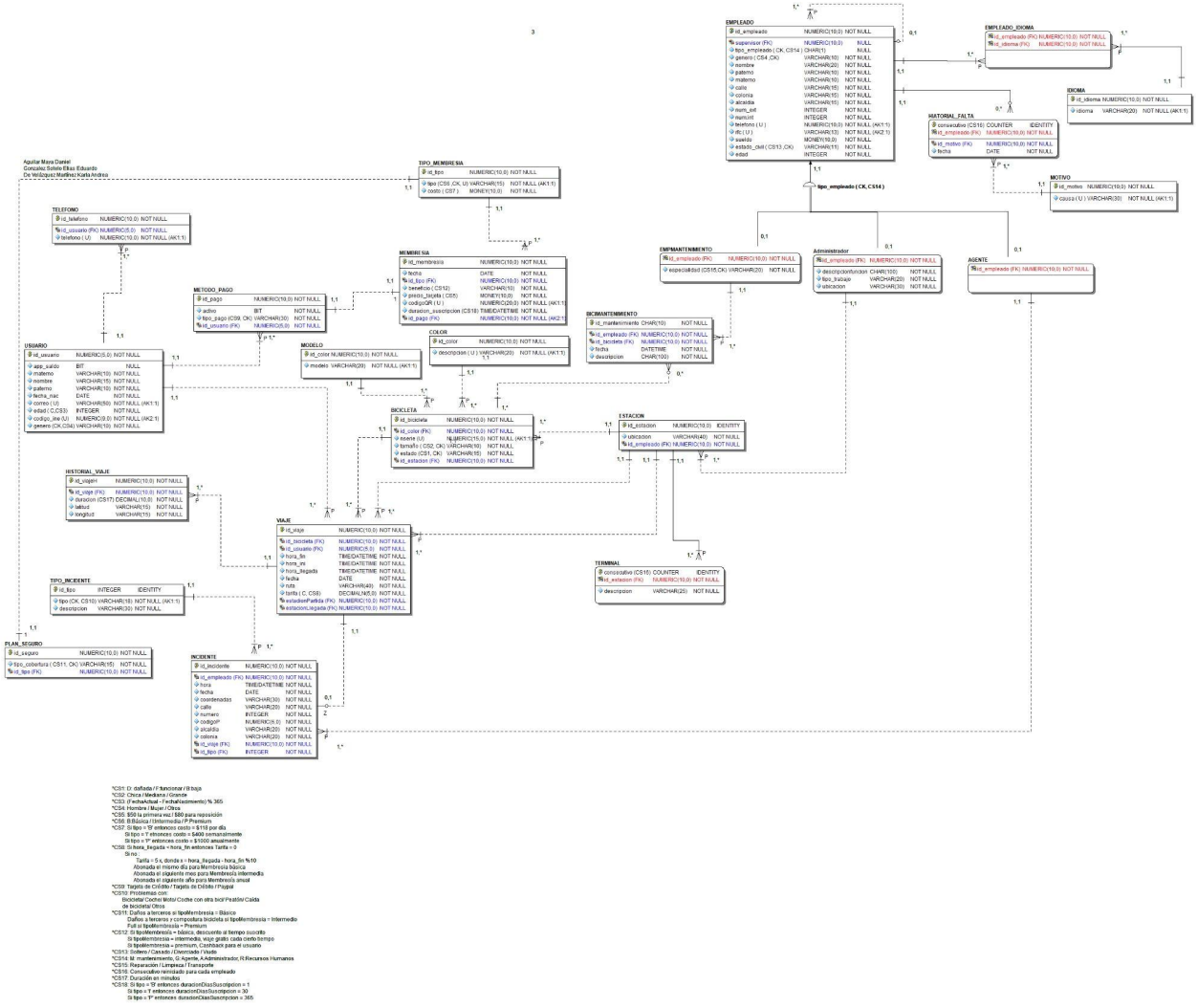


*CS1: D: dañada / F:fundonar / B:baja
*CS2: Chica / Mediana / Grande
*CS3: (FechaActual - FechaNacimiento) % 365
*CS4: Hombre / Mujer / Otros
*CS5: $50 la primera vez / $80 para reposición
*CS6: B:Básica / I:Intermedia / P:Premium
*CS7: Si tipo = 'B' entonces costo = $118 por día
Si tipo = 'I' etnonces costo = $400 semanalmente
Si tipo = 'P' entonces costo = $1000 anualmente
*CS8: Si hora_llegada = hora_fin entonces Tarifa = 0
Si no :
Tarifa = 5 x, donde x = hora_llegada - hora_fin %10
Abonada el mismo día para Membresía básica
Abonada el siguiente mes para Membresía intermedia
Abonada el siguiente año para Membresía anual
*CS9: Tageta de Crédito / Tageta de Débito / Paypal
*CS10: Predeterpo con:
Bicicleta/ Coche/ Motor/ Coche con otra bici/ Peatón/ Caída
de bicicleta/ Otros
*CS11: Daños a terceros si tipoMembresia = Básico
Daños a terceros y compostura bicicleta si tipoMembresia = Intermedio
Full si tipoMembresia = Premium
*CS12: Si tipoMembresia = básica, descuento al tiempo suscrito
Si tipoMembresia = intermedia, viaje gratis cada cierto tiempo
Si tipoMembresia = premium, Cashback para el usuario
*CS13: Soltero / Casado / Divorciado / Viudo
*CS14: M: mantenimiento, G: Agente, A:Administrador, R:Recursos Humanos
*CS15: Reparación / Limpieza / Transporte
*CS16: Consecutivo reiniciado para cada empleado
*CS17: Duración en minutos

VERSION 1 MODELO ENTIDAD-RELACIONAL



VERSION 1.1 MODELO ENTIDAD-RELACIONAL



VERSION 1.2 MODELO ENTIDAD-RELACIONAL

7.1 ÁLGEBRA RELACIONAL

Vista 2

```
select e.id_empleado, nombre + ' ' + paterno + ' ' + materno as Nombre,
id.idioma
from EMPLEADOS.empleado e inner join empleados.EMPLEADO_IDIOMA ei on
e.id_empleado = ei.id_empleado
inner join empleados.idioma id on ei.id_idioma = id.id_idioma
```

```
π (e.id_empleado, e.nombre + ' ' + e.paterno + ' ' + e.materno AS
Nombre, id.idioma) ( EMPLEADOS.empleado e ⋈ (e.id_empleado =
```

ei.id_empleado) empleados.EMPLEADO_IDIOMA ei ⋈ (ei.id_idioma = id.id_idioma) empleados.idioma id)

VISTA 3

select id_bicicleta, color, modelo, nserie, tamaño, es.nombre, es.ubicacion
FROM estacion.bicicleta b inner join estacion.color c on b.id_color = c.id_color
inner join estacion.modelo m on m.id_modelo = b.id_modelo inner join estacion.estacion es on es.id_estacion = b.id_estacion

π (b.id_bicicleta, b.color, b.modelo, b.nserie, b.tamaño, es.nombre, es.ubicacion) (estacion.bicicleta b ⋈ (b.id_color = c.id_color) estacion.color c ⋈ (m.id_modelo = b.id_modelo) estacion.modelo m ⋈ (es.id_estacion = b.id_estacion) estacion.estacion)