

Proyecto final: Desarrollo de una Base de Datos a partir de Conjuntos de Datos

Abiertos

Gabriel Fernando Castillo Mendieta

Juan David Ochoa Pinilla

Escuela de ingeniería de sistemas y computación, Universidad Pedagógica y Tecnológica de

Colombia

Bases de Datos II

Ing. Juan Sebastián González Sanabria

3 de junio de 2024

Descripción General de los Datos Utilizados

Para el desarrollo del proyecto, se utilizaron datos públicos del gobierno colombiano titulados "Costos de la atención hospitalaria en Bucaramanga por accidentes de tránsito enero 2018 a noviembre 2021". Estos datos permiten analizar el costo promedio de un accidente de tránsito para las aseguradoras, tomando en cuenta diversos factores como el tipo de vehículo involucrado, la relación del conductor con el accidente, el historial de salud del paciente, el sexo de los involucrados, el mes del año en que ocurrió el accidente y el costo total de la atención médica (Datos Abiertos. 2018).

El conjunto de datos empleado es detallado e incluye variables como: edad y sexo de las víctimas, tipo de vehículo involucrado, relación de la persona con el accidente (conductor, pasajero, peatón, etc.), fechas y horas del accidente y de la atención médica, tiempos de respuesta de los servicios de emergencia, necesidad de cirugía, referencias a otras instituciones de salud y estado al egreso del paciente. Además, el conjunto de datos contiene información sobre el tipo de afiliación del paciente al sistema de salud, la entidad de salud responsable de la atención, los costos desglosados de la atención médica y detalles específicos del accidente como el mes, día de la semana y la administradora de salud a la que estaba afiliado el paciente.

Herramientas Empleadas

PowerDesigner

PowerDesigner es una herramienta líder en el modelado de datos por su amplia oferta en técnicas de modelización, PowerDesigner facilitó la creación del modelo conceptual y lógico de la base de datos, definiendo entidades, atributos, relaciones y generando el modelo lógico de manera eficiente.

DataGrip

DataGrip es una herramienta que ofrece diferentes beneficios, como: ahorrar tiempo en la automatización de tareas rutinarias, localizar y arreglar errores, además de ofrecer resaltado de sintaxis preciso en MySQL y ser compatible con todos los tipos importantes de objetos de tu base de datos de MySQL. (DataGrip herramienta de GUI para MySQL, s/f.). Por lo anterior, su uso permitió la conexión y gestión de la base de datos, automatizando tareas, localizando errores, ofreciendo resaltado de sintaxis preciso en MySQL y compatibilidad con objetos de la base de datos. Adicionalmente, DataGrip facilitó la importación de datos CSV, la manipulación de datos mediante filtros y búsquedas, la detección y corrección de errores en consultas y la revisión de sintaxis, optimizando el tiempo y la productividad en el desarrollo del proyecto.

Arquitectura de Almacenamiento Propuesta

Se empleó MySQL como base de datos relacional para almacenar la atención hospitalaria de las personas que tuvieron accidentes de tránsito. Principalmente, porque “MySQL es rápido, confiable, ampliable y fácil de utilizar. Debido a que fue desarrollado para manejar rápidamente grandes bases de datos. Por ello, se ha utilizado durante muchos años en entornos de producción altamente exigentes” (Oracle, 2023).

Luego de la selección de bases de datos se evaluaron los siguientes aspectos:

Estructura y Organización de Datos

- Naturaleza relacional de los datos hospitalarios: los datos hospitalarios son inherentemente relacionales porque cada persona está asociada a un accidente específico y a su respectiva atención médica. Tanto los datos del accidente como los de la atención médica están relacionados con entidades como la EPS, la IPS, el tipo de vehículo, y las administradoras. La utilización de una base de datos relacional como MySQL permite estructurar estos datos de manera organizada y eficiente, estableciendo relaciones claras y definidas entre las distintas entidades.

- Prevención de duplicidad de datos: según Amazon Web Services, (AWS, 2024), la integridad de los datos es la totalidad, precisión y coherencia general de los datos. Las bases de datos relacionales utilizan un conjunto de restricciones para aplicar la integridad de los datos en la base de datos. Esto incluye la clave principal, la clave externa y las restricciones “Not NULL”, “Unique”, “Default” y “Check”. MySQL, mediante el uso de claves primarias y foráneas, asegura la integridad referencial y evita la duplicación de datos. Esto garantiza que cada registro sea único y que las relaciones entre las tablas sean coherentes, facilitando la gestión de los datos y manteniendo la consistencia de la información.

Integridad de los Datos

- Implementación de restricciones y claves foráneas: MySQL permite definir restricciones y claves foráneas que aseguran la validez de las relaciones entre las tablas. Esto es crucial en un sistema de atención hospitalaria, donde la precisión y coherencia de los datos son esenciales para proporcionar una atención médica de calidad.
- Validaciones y restricciones de verificación: las restricciones de verificación (CHECK) permiten mantener la calidad y consistencia de los datos. Por ejemplo, las restricciones de rango en las edades, las validaciones de género y los tipos de accidentes garantizan que solo se ingresen datos válidos y coherentes.

Consulta y Análisis de Datos

- Flexibilidad del lenguaje SQL: MySQL proporciona un lenguaje de consulta estructurado (SQL) que permite realizar consultas complejas, limpieza y transformación de datos, eliminación de duplicados, normalización de formatos y corrección de inconsistencias. Esto es fundamental para mantener la calidad de los datos y para obtener información precisa y útil.

- Automatización mediante procedimientos almacenados y Triggers: MySQL permite la creación de procedimientos almacenados y triggers que automatizan la validación de datos y aseguran la integridad y calidad de los mismos. Esto facilita la implementación de reglas de negocio y la realización de tareas repetitivas de manera eficiente y confiable.

Consistencia y Fiabilidad de los Datos

- Soporte para las propiedades Atomicidad, Consistencia, Aislamiento y Durabilidad (ACID: de acuerdo con AWS, ACID es la opción ideal para aplicaciones empresariales que requieren coherencia de datos, fiabilidad y previsibilidad. Por ejemplo, los bancos utilizan una base de datos ACID para almacenar las transacciones de los clientes porque la integridad de los datos es la máxima prioridad. MySQL soporta transacciones que cumplen con las propiedades ACID, asegurando que las operaciones de la base de datos sean confiables y que los datos permanezcan consistentes incluso en caso de fallos o errores.
- Transacciones seguras y confiables: la capacidad de realizar transacciones seguras es fundamental en un entorno hospitalario, donde la precisión y la integridad de los datos pueden tener un impacto directo en la atención al paciente.

Escalabilidad y Rendimiento

- Manejo de Grandes Volúmenes de Datos: MySQL está diseñado para gestionar grandes volúmenes de datos y soportar múltiples usuarios concurrentes, lo cual es esencial para una base de datos de atención hospitalaria que puede crecer con el tiempo.
- Optimización del Rendimiento: De acuerdo con Oracle (2023) “MySQL se amplía para satisfacer las demandas de las aplicaciones más accesibles. La arquitectura de replicación nativa de MySQL permite a organizaciones como Facebook escalar

aplicaciones para admitir miles de millones de usuarios”. MySQL ofrece diversas herramientas y técnicas para optimizar el rendimiento, como índices, particionamiento de tablas y ajustes de configuración, lo que permite mantener tiempos de respuesta rápidos incluso con grandes conjuntos de datos y cargas de trabajo intensivas.

Modelo Conceptual

Para el modelo conceptual (Ver figura 1), podemos identificar las siguientes entidades y atributos:

1. Tipo administradora
 - a. ID tipo administradora.
 - b. Tipo administradora.
2. Administradora
 - a. ID administradora.
 - b. Nombre administradora.
3. EPS
 - a. ID eps
 - b. Nombre Eps
4. Informe accidente
 - a. ID accidente
 - b. Género (Femenino, Masculino)
 - c. Año de nacimiento
 - d. Actor vial (ciclista, conductor, pasajero, peatón, sin información)
 - e. fecha de accidente
 - f. hora de accidente
5. Grupo etario
 - a. ID grupo etario

- b. curso de vida
- c. rango mínimo
- d. rango máximo

6. Afiliación

- a. ID afiliación.
- b. Tipo de afiliación

7. Vehículo

- a. ID vehículo
- b. Tipo de vehículo

8. IPS

- a. ID IPS
- b. nombre de IPS

9. Registro individual de prestación de servicios de salud:

- a. Cod RIPS
- b. Descripción RIPS

10. Procedimiento

- a. id diagnóstico
- b. requerimiento cx
- c. politraumatismo

11. Atención Hospitalaria

- a. ID atención
- b. fecha ingreso ips
- c. hora_ingreso_ips
- d. fecha atención médica
- e. hora_atencion_medica

- f. duración atención
- g. costo
- h. condición egreso (vivo, muerto, sin información)
- i. triage (nivel I, nivel 2, nivel 3, nivel 4, sin información)
- j. referido ips

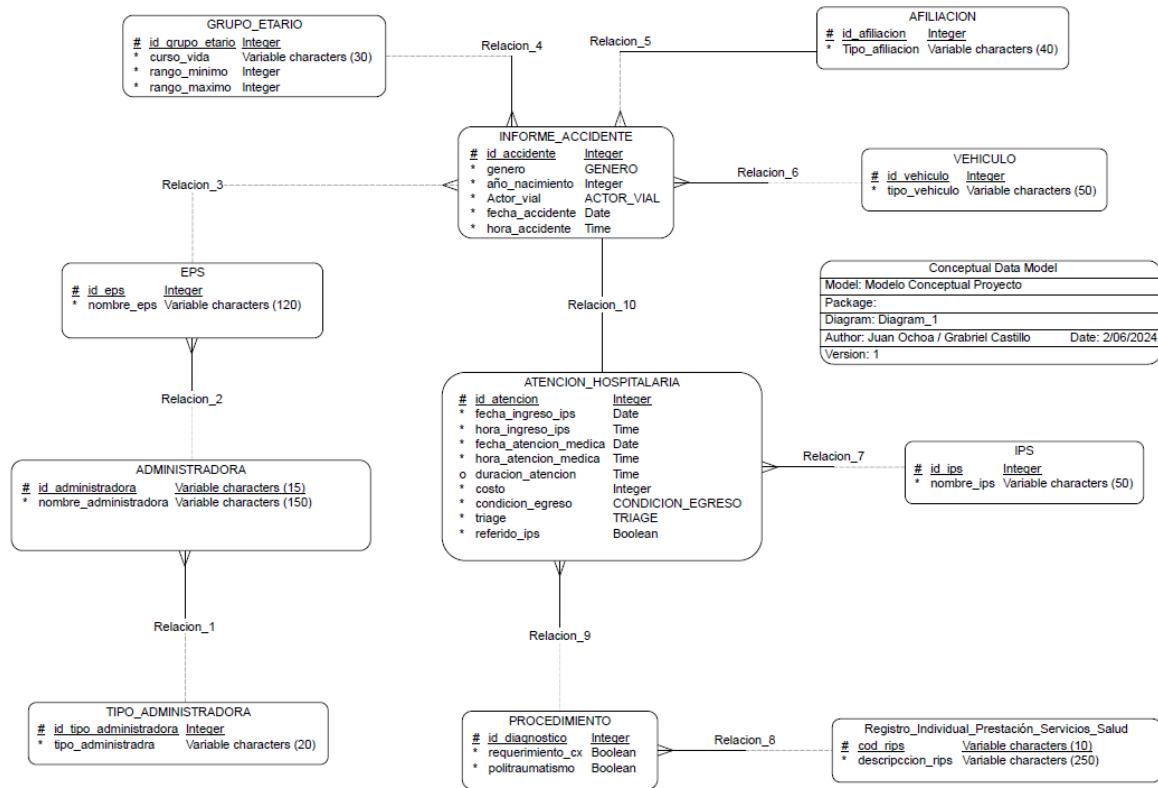
Relacionamiento

1. Una administradora pertenece a un tipo de administradora y a un tipo de administradora la pueden tener una o varias administradoras.
2. Una EPS tiene una administradora asignada y una administradora puede tener a su cargo una o varias EPS.
3. Una EPS puede estar registrada en uno o varios informes de accidente y a su vez un informe de accidente puede asociar una EPS.
4. Para cada informe de accidente solo se establece un grupo etario, sin embargo, un grupo etario se puede registrar en varios informes de accidentes.
5. En un informe de accidente se puede asignar una afiliación y una afiliación estará registrada en uno o varios informes.
6. Un tipo de vehículo puede estar involucrado en uno o varios informes de accidentes y cada informe debe estar asociado a un vehículo.
7. Una atención hospitalaria se debe realizar en una IPS y cada IPS puede atender varias atenciones hospitalarias.
8. El registro individual de prestación de servicios de salud puede estar implicado en uno o varios procedimientos y un procedimiento siempre va a tener asociado un código RIPS.
9. Cada atención hospitalaria va a tener relacionado un procedimiento y el procedimiento puede tener varias atenciones hospitalarias.

10. Para cada informe de accidente se debe generar una atención hospitalaria.

Figura 1.

Modelo conceptual



Modelo Lógico

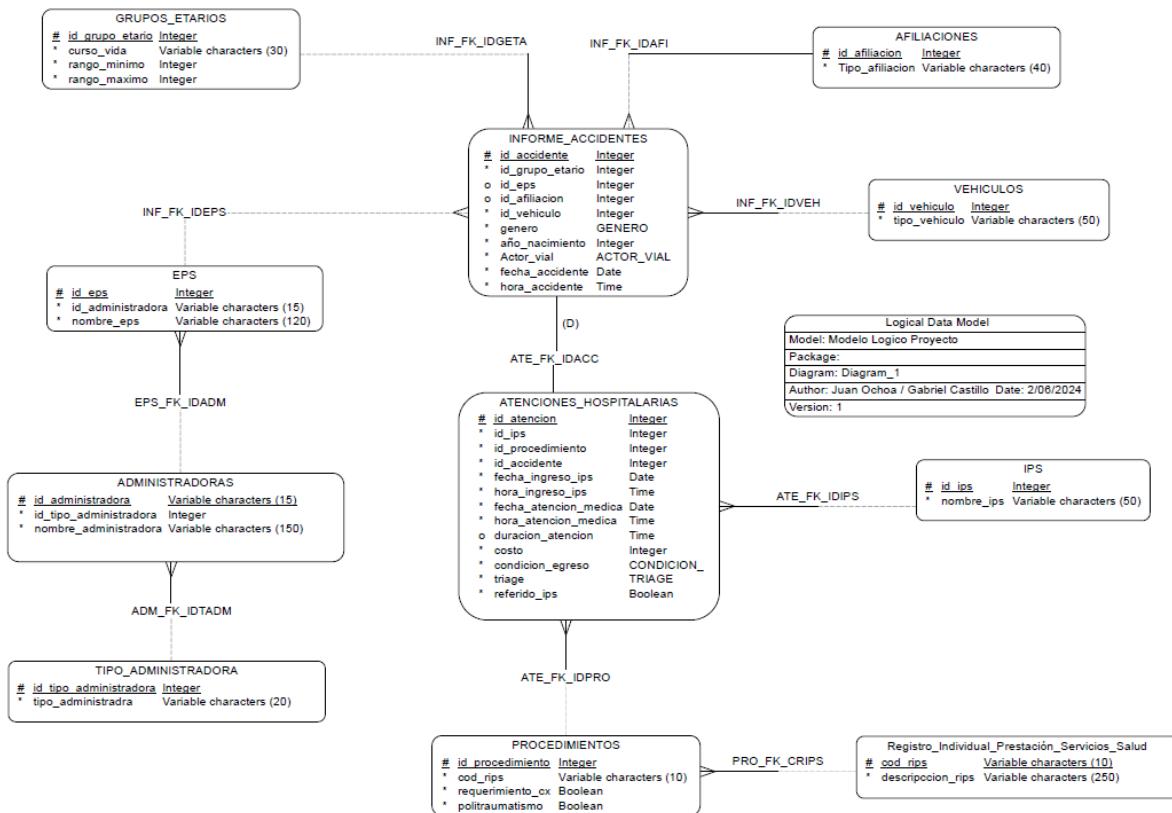
Los nombres de las tablas y las llaves foránea ,si corresponden, quedan de la siguiente manera, (Ver figura 2):

1. Tipo de administradora.
2. Administradoras recibe la llave primaria como una llave foránea (fk) de tipo administradora (ADM_FK_ID TADM).
3. EPS recibe la llave primaria como una llave foránea (fk) de administradoras (EPS_FK_IDADM)
4. Grupos etarios.
5. Afiliaciones.

6. Vehículos.
7. Informe Accidentes recibe las llaves primarias como llaves foráneas (fk) de:
 - a. EPS (*INF_FK_IDEPS*)
 - b. Grupos etarios (*INF_FK_IDGETA*)
 - c. Afiliaciones (*INF_FK_IDAFI*)
 - d. Vehículos (*INF_FK_IDVEH*)
8. Registro individual de prestación de servicios de salud.
9. IPS.
10. Procedimientos recibe la llave primaria como una llave foránea (fk) del Registro individual de prestación de servicios de salud (*PRO_FK_CRIPS*)
11. Atenciones hospitalarias recibe las llaves primarias como llaves foráneas (fk) de:
 - a. IPS (*ATE_FK_IDIPS*)
 - b. Procedimientos (*ATE_FK_IDPRO*)
 - c. Informe Accidentes (*ATE_FK_IDACC*)

Figura 2.

Modelo lógico



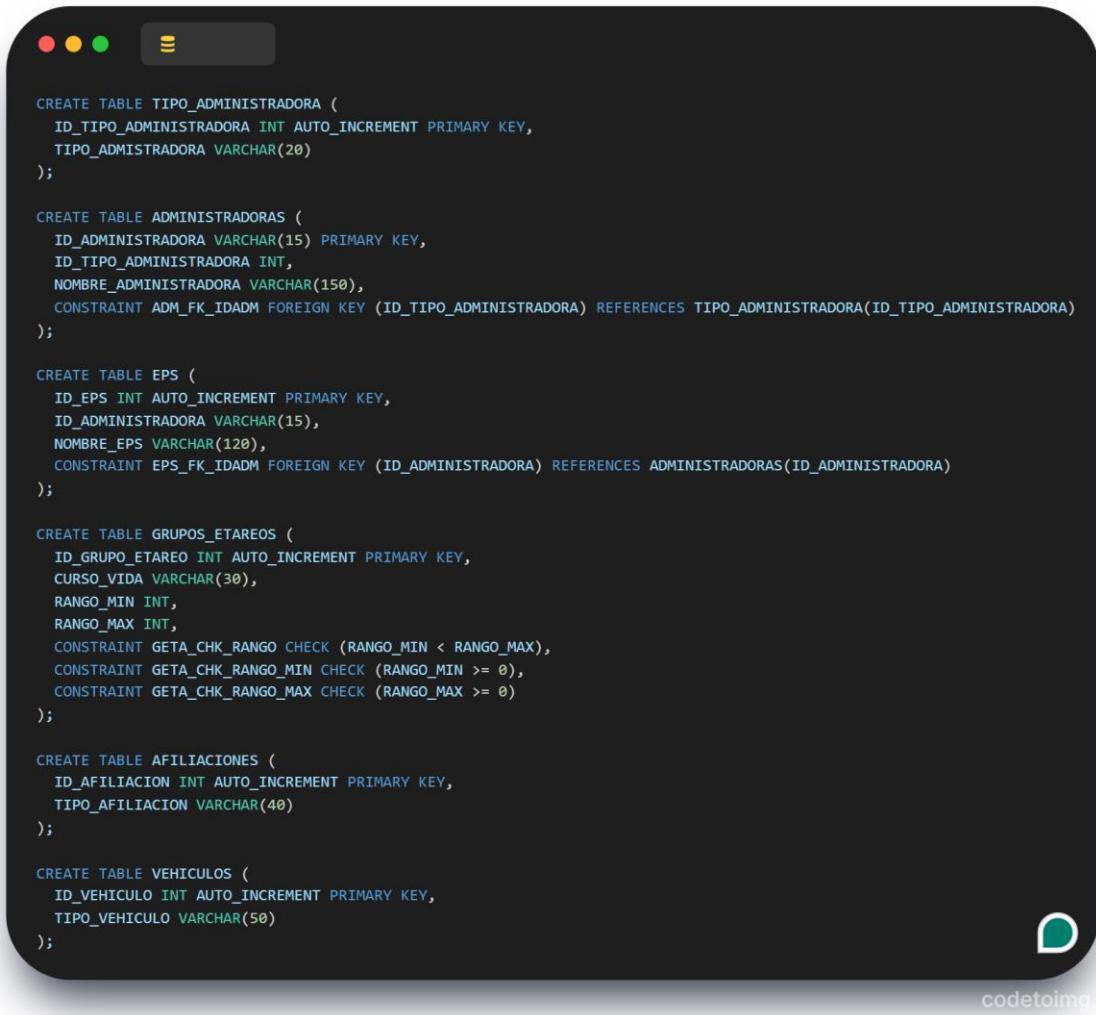
Scripts de Implementación de la Base de Datos

Se realizó la importación de los datos por medio del sistema de importación de DataGrip desde un CSV a la tabla ‘test_table’. Esta tabla mantuvo todas las columnas que tenía el archivo, para posteriormente migrar la información limpia a las tablas correspondientes. Cabe resaltar que el documento contiene alrededor de treinta y dos mil registros, que fueron insertados en su totalidad dentro del modelo.

En primer lugar, se realizó la creación de las tablas, conforme al modelo propuesto y teniendo en cuenta múltiples restricciones, así como los dominios propuestos que darían mejor estabilidad a la base de datos, ver figura 3 y 4.

Figura 3.

Implementación de las tablas de la base de datos



```
CREATE TABLE TIPO_ADMINISTRADORA (
    ID_TIPO_ADMINISTRADORA INT AUTO_INCREMENT PRIMARY KEY,
    TIPO_ADMINISTRADORA VARCHAR(20)
);

CREATE TABLE ADMINISTRADORAS (
    ID_ADMINISTRADORA VARCHAR(15) PRIMARY KEY,
    ID_TIPO_ADMINISTRADORA INT,
    NOMBRE_ADMINISTRADORA VARCHAR(150),
    CONSTRAINT ADM_FK_IDADM FOREIGN KEY (ID_TIPO_ADMINISTRADORA) REFERENCES TIPO_ADMINISTRADORA(ID_TIPO_ADMINISTRADORA)
);

CREATE TABLE EPS (
    ID_EPS INT AUTO_INCREMENT PRIMARY KEY,
    ID_ADMINISTRADORA VARCHAR(15),
    NOMBRE_EPS VARCHAR(120),
    CONSTRAINT EPS_FK_IDADM FOREIGN KEY (ID_ADMINISTRADORA) REFERENCES ADMINISTRADORAS(ID_ADMINISTRADORA)
);

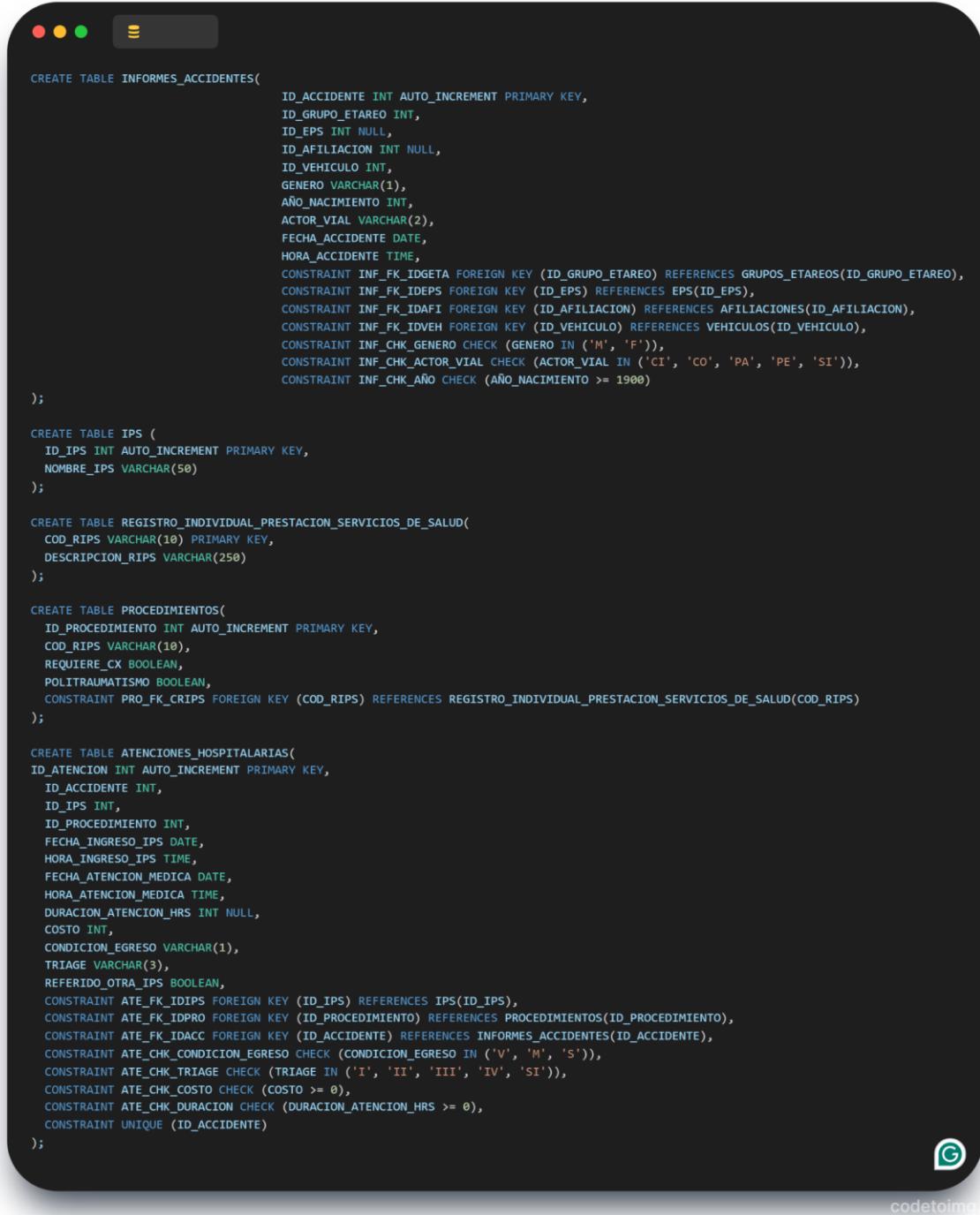
CREATE TABLE GRUPOS_ETAREOS (
    ID_GRUPO_ETAREO INT AUTO_INCREMENT PRIMARY KEY,
    CURSO_VIDA VARCHAR(30),
    RANGO_MIN INT,
    RANGO_MAX INT,
    CONSTRAINT GETA_CHK_RANGO CHECK (RANGO_MIN < RANGO_MAX),
    CONSTRAINT GETA_CHK_RANGO_MIN CHECK (RANGO_MIN >= 0),
    CONSTRAINT GETA_CHK_RANGO_MAX CHECK (RANGO_MAX >= 0)
);

CREATE TABLE AFILIACIONES (
    ID_AFILIACION INT AUTO_INCREMENT PRIMARY KEY,
    TIPO_AFILIACION VARCHAR(40)
);

CREATE TABLE VEHICULOS (
    ID_VEHICULO INT AUTO_INCREMENT PRIMARY KEY,
    TIPO_VEHICULO VARCHAR(50)
);
```

Figura 4.

Implementación de las tablas de la base de datos - II



```
CREATE TABLE INFORMES_ACCIDENTES(
    ID_ACCIDENTE INT AUTO_INCREMENT PRIMARY KEY,
    ID_GRUPO_ETAREO INT,
    ID_EPS INT NULL,
    ID_AFILIACION INT NULL,
    ID_VEHICULO INT,
    GENERO VARCHAR(1),
    AÑO_NACIMIENTO INT,
    ACTOR_VIAL VARCHAR(2),
    FECHA_ACCIDENTE DATE,
    HORA_ACCIDENTE TIME,
    CONSTRAINT INF_FK_IDGETA FOREIGN KEY (ID_GRUPO_ETAREO) REFERENCES GRUPOS_ETAREOS(ID_GRUPO_ETAREO),
    CONSTRAINT INF_FK_IDEPS FOREIGN KEY (ID_EPS) REFERENCES EPS(ID_EPS),
    CONSTRAINT INF_FK_IDAFI FOREIGN KEY (ID_AFILIACION) REFERENCES AFILIACIONES(ID_AFILIACION),
    CONSTRAINT INF_FK_IDVEH FOREIGN KEY (ID_VEHICULO) REFERENCES VEHICULOS(ID_VEHICULO),
    CONSTRAINT INF_CHK_GENERO CHECK (GENERO IN ('M', 'F')),
    CONSTRAINT INF_CHK_ACTOR_VIAL CHECK (ACTOR_VIAL IN ('CI', 'CO', 'PA', 'PE', 'SI')),
    CONSTRAINT INF_CHK_AÑO CHECK (AÑO_NACIMIENTO >= 1900)
);

CREATE TABLE IPS (
    ID_IPS INT AUTO_INCREMENT PRIMARY KEY,
    NOMBRE_IPS VARCHAR(50)
);

CREATE TABLE REGISTRO_INDIVIDUAL_PRESTACION_SERVICIOS_DE_SALUD(
    COD_RIPS VARCHAR(10) PRIMARY KEY,
    DESCRIPCION_RIPS VARCHAR(250)
);

CREATE TABLE PROCEDIMIENTOS(
    ID_PROCEDIMIENTO INT AUTO_INCREMENT PRIMARY KEY,
    COD_RIPS VARCHAR(10),
    REQUIERE_CX BOOLEAN,
    POLITRAUMATISMO BOOLEAN,
    CONSTRAINT PRO_FK_CRIPS FOREIGN KEY (COD_RIPS) REFERENCES REGISTRO_INDIVIDUAL_PRESTACION_SERVICIOS_DE_SALUD(COD_RIPS)
);

CREATE TABLE ATENCIONES_HOSPITALARIAS(
    ID_ATENCION INT AUTO_INCREMENT PRIMARY KEY,
    ID_ACCIDENTE INT,
    ID_IPS INT,
    ID_PROCEDIMIENTO INT,
    FECHA_INGRESO_IPS DATE,
    HORA_INGRESO_IPS TIME,
    FECHA_ATENCION_MEDICA DATE,
    HORA_ATENCION_MEDICA TIME,
    DURACION_ATENCION_HRS INT NULL,
    COSTO INT,
    CONDICION_EGRESO VARCHAR(1),
    TRIAGE VARCHAR(3),
    REFERIDO_OTRA_IPS BOOLEAN,
    CONSTRAINT ATE_FK_IDIPS FOREIGN KEY (ID_IPS) REFERENCES IPS(ID_IPS),
    CONSTRAINT ATE_FK_IDPRO FOREIGN KEY (ID_PROCEDIMIENTO) REFERENCES PROCEDIMIENTOS(ID_PROCEDIMIENTO),
    CONSTRAINT ATE_FK_IDACC FOREIGN KEY (ID_ACCIDENTE) REFERENCES INFORMES_ACCIDENTES(ID_ACCIDENTE),
    CONSTRAINT ATE_CHK_CONDICION_EGRESO CHECK (CONDICION_EGRESO IN ('V', 'M', 'S')),
    CONSTRAINT ATE_CHK_TRIAGE CHECK (TRIAGE IN ('I', 'II', 'III', 'IV', 'SI')),
    CONSTRAINT ATE_CHK_COSTO CHECK (COSTO >= 0),
    CONSTRAINT ATE_CHK_DURACION CHECK (DURACION_ATENCION_HRS >= 0),
    CONSTRAINT UNIQUE (ID_ACCIDENTE)
);
```

Teniendo estos datos en la tabla inicial, se procede a realizar una limpieza general de los errores que son propios del documento, por ejemplo, aquellas columnas con valores,

como: '#N/D', '# ¡NUM!', 'ERROR: #N/A', '# ¡VALOR!'. Columnas que adquirieron un valor predeterminado dependiendo de la situación/contexto, ver figura 5.

Figura 5.

Limpieza general de las columnas.

```

UPDATE
    test_table
SET
    Numero = IF(Numero IN ('#N/D', '#¡NUM!', 'ERROR: #N/A', '#¡VALOR!'), '0', Numero),
    EDAD = IF(EDAD IN ('#N/D', '#¡NUM!', 'ERROR: #N/A', '#¡VALOR!'), '0', EDAD),
    SEXO = IF(SEXO IN ('#N/D', '#¡NUM!', 'ERROR: #N/A', '#¡VALOR!'), 'SIN INFORMACION', SEXO),
    `TIPO DE VEHÍCULO` = IF(`TIPO DE VEHÍCULO` IN ('#N/D', '#¡NUM!', 'ERROR: #N/A', '#¡VALOR!'), 'SIN INFORMACION',
                           `TIPO DE VEHÍCULO`),
    `RELACION USUARIO/ACCIDENTE` = IF(`RELACION USUARIO/ACCIDENTE` IN ('#N/D', '#¡NUM!', 'ERROR: #N/A', '#¡VALOR!'),
                                         'SIN INFORMACION', `RELACION USUARIO/ACCIDENTE`),
    `FECHA DE ACCIDENTE` = IF(`FECHA DE ACCIDENTE` IN ('#N/D', '#¡NUM!', 'ERROR: #N/A', '#¡VALOR!'), '01-01-1900',
                               `FECHA DE ACCIDENTE`),
    `FECHA DE INGRESO IPS` = IF(`FECHA DE INGRESO IPS` IN ('#N/D', '#¡NUM!', 'ERROR: #N/A', '#¡VALOR!'), '01-01-1900',
                                `FECHA DE INGRESO IPS`),
    `HORA DE ACCIDENTE` = IF(`HORA DE ACCIDENTE` IN ('#N/D', '#¡NUM!', 'ERROR: #N/A', '#¡VALOR!'), '12:00:00 PM',
                             `HORA DE ACCIDENTE`),
    `HORA DE INGRESO IPS` = IF(`HORA DE INGRESO IPS` IN ('#N/D', '#¡NUM!', 'ERROR: #N/A', '#¡VALOR!'), '12:00:00 PM',
                               `HORA DE INGRESO IPS`),
    `FECHA DE ATENCION MÉDICA` = IF(`FECHA DE ATENCION MÉDICA` IN ('#N/D', '#¡NUM!', 'ERROR: #N/A', '#¡VALOR!'),
                                      '01-01-1900', `FECHA DE ATENCION MÉDICA`),
    `HORA DE ATENCION MÉDICA` = IF(`HORA DE ATENCION MÉDICA` IN ('#N/D', '#¡NUM!', 'ERROR: #N/A', '#¡VALOR!'),
                                   '12:00:00 PM', `HORA DE ATENCION MÉDICA`),
    `OPORTUNIDAD DE LA ATENCIÓN EN HORAS` = IF(
        `OPORTUNIDAD DE LA ATENCIÓN EN HORAS` IN ('#N/D', '#¡NUM!', 'ERROR: #N/A', '#¡VALOR!'), 'SIN INFORMACION',
        `OPORTUNIDAD DE LA ATENCIÓN EN HORAS`),
    `OPORTUNIDAD DE LA ATENCIÓN EN MINUTOS` = IF(
        `OPORTUNIDAD DE LA ATENCIÓN EN MINUTOS` IN ('#N/D', '#¡NUM!', 'ERROR: #N/A', '#¡VALOR!'), '0',
        `OPORTUNIDAD DE LA ATENCIÓN EN MINUTOS`),
    `REQUERIMIENTO DE CX` = IF(`REQUERIMIENTO DE CX` IN ('#N/D', '#¡NUM!', 'ERROR: #N/A', '#¡VALOR!'), '0',
                               `REQUERIMIENTO DE CX`),
    `REFERIDO A OTRA IPS` = IF(`REFERIDO A OTRA IPS` IN ('#N/D', '#¡NUM!', 'ERROR: #N/A', '#¡VALOR!'), '0',
                               `REFERIDO A OTRA IPS`),
    `REGION ANATÓMICA MÁS AFECTADA` = IF(
        `REGION ANATÓMICA MÁS AFECTADA` IN ('#N/D', '#¡NUM!', 'ERROR: #N/A', '#¡VALOR!'), 'SIN INFORMACION',
        `REGION ANATÓMICA MÁS AFECTADA`),
    POLITRAUMATISMO = IF(POLITRAUMATISMO IN ('#N/D', '#¡NUM!', 'ERROR: #N/A', '#¡VALOR!'), '0', POLITICOUMATISMO),
    `TIPO AFILIACIÓN` = IF(`TIPO AFILIACIÓN` IN ('#N/D', '#¡NUM!', 'ERROR: #N/A', '#¡VALOR!'), 'SIN INFORMACION',
                           `TIPO AFILIACIÓN`),
    `EPS USUARIO` = IF(`EPS USUARIO` IN ('#N/D', '#¡NUM!', 'ERROR: #N/A', '#¡VALOR!'), 'SIN INFORMACION', `EPS USUARIO`),
    `CONDICION EGRESO` = IF(`CONDICION EGRESO` IN ('#N/D', '#¡NUM!', 'ERROR: #N/A', '#¡VALOR!'), 'SIN INFORMACION',
                           `CONDICION EGRESO`),
    `OPORTUNIDAD CITAS DE CONTROL` = IF(`OPORTUNIDAD CITAS DE CONTROL` IN ('#N/D', '#¡NUM!', 'ERROR: #N/A', '#¡VALOR!'),
                                         '0', `OPORTUNIDAD CITAS DE CONTROL`),
    COSTOS = IF(COSTOS IN ('#N/D', '#¡NUM!', 'ERROR: #N/A', '#¡VALOR!'), '0', COSTOS),
    MES = IF(MES IN ('#N/D', '#¡NUM!', 'ERROR: #N/A', '#¡VALOR!'), '01. ENERO', MES),
    IPS = IF(IPS IN ('#N/D', '#¡NUM!', 'ERROR: #N/A', '#¡VALOR!'), 'SIN INFORMACION', IPS),
    `GRUPO ETAREO` = IF(`GRUPO ETAREO` IN ('#N/D', '#¡NUM!', 'ERROR: #N/A', '#¡VALOR!'), 'SIN INFORMACION',
                        `GRUPO ETAREO`),
    `CURSO DE VIDA` = IF(`CURSO DE VIDA` IN ('#N/D', '#¡NUM!', 'ERROR: #N/A', '#¡VALOR!'), 'SIN INFORMACION',
                         `CURSO DE VIDA`),
    AÑO = IF(AÑO IN ('#N/D', '#¡NUM!', 'ERROR: #N/A', '#¡VALOR!'), '1900', AÑO),
    `DIA SEMANA` = IF(`DIA SEMANA` IN ('#N/D', '#¡NUM!', 'ERROR: #N/A', '#¡VALOR!'), '1. LUNES', `DIA SEMANA`),
    `Código Administradora` = IF(`Código Administradora` IN ('#N/D', '#¡NUM!', 'ERROR: #N/A', '#¡VALOR!'),
                                 'SIN INFORMACION', `Código Administradora`),
    Administradora = IF(Administradora IN ('#N/D', '#¡NUM!', 'ERROR: #N/A', '#¡VALOR!'), 'SIN INFORMACION',
                        Administradora),
    `Tipo Administradora` = IF(`Tipo Administradora` IN ('#N/D', '#¡NUM!', 'ERROR: #N/A', '#¡VALOR!'),
                              'SIN INFORMACION', `Tipo Administradora`),
    `Codigo RIPS` = IF(`Codigo RIPS` IN ('#N/D', '#¡NUM!', 'ERROR: #N/A', '#¡VALOR!'), 'SIN INFORMACION', `Codigo RIPS`),
    `Descripcion RIPS` = IF(`Descripcion RIPS` IN ('#N/D', '#¡NUM!', 'ERROR: #N/A', '#¡VALOR!'), 'SIN INFORMACION',
                           `Descripcion RIPS`);

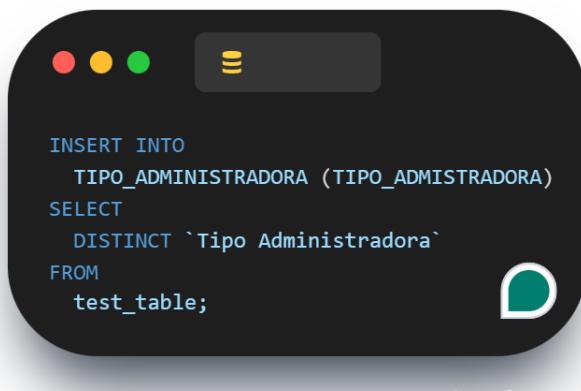
```

Manteniendo esta limpieza general, se procede a la inserción de los datos dentro de las tablas correspondientes, iniciando por aquellas que tienen una menor cantidad de relaciones, con el fin de evitar problemas en el relacionamiento. En ese sentido se mantiene el siguiente orden:

- **Tabla “TIPO_ADMINISTRADORA”:** se aplica un script inserta valores únicos en la columna TIPO_ADMINISTRADORA de la tabla TIPO_ADMINISTRADORA tomando los datos distintos de la columna Tipo Administradora de la tabla test_table, ver figura 6.

Figura 6.

Inserción de datos en la tabla TIPO_ADMINISTRADORA.

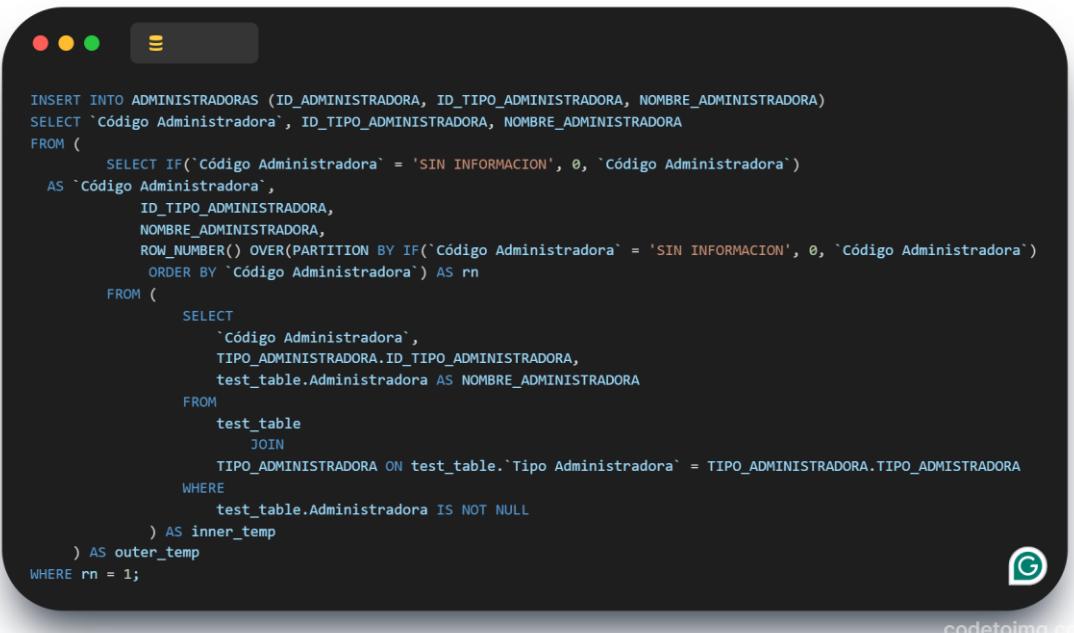


```
INSERT INTO
    TIPO_ADMINISTRADORA (TIPO_ADMINISTRADORA)
SELECT
    DISTINCT `Tipo Administradora`
FROM
    test_table;
```

- **Tabla “ADMINISTRADORAS”:** el script inserta datos en la tabla ADMINISTRADORAS, seleccionando Código Administradora, ID_TIPO_ADMINISTRADORA y NOMBRE_ADMINISTRADORA de una tabla temporal. Filtra y transforma datos para asegurar que Código Administradora tenga un valor único, reemplazando 'SIN INFORMACION' con 0 y utilizando ROW_NUMBER () para eliminar duplicados. Solo se insertan registros donde Administradora no es nulo y rn es 1, garantizando la unicidad de cada administradora, ver figura 7.

Figura 7.

Inserción de datos en la tabla ADMINISTRADORAS



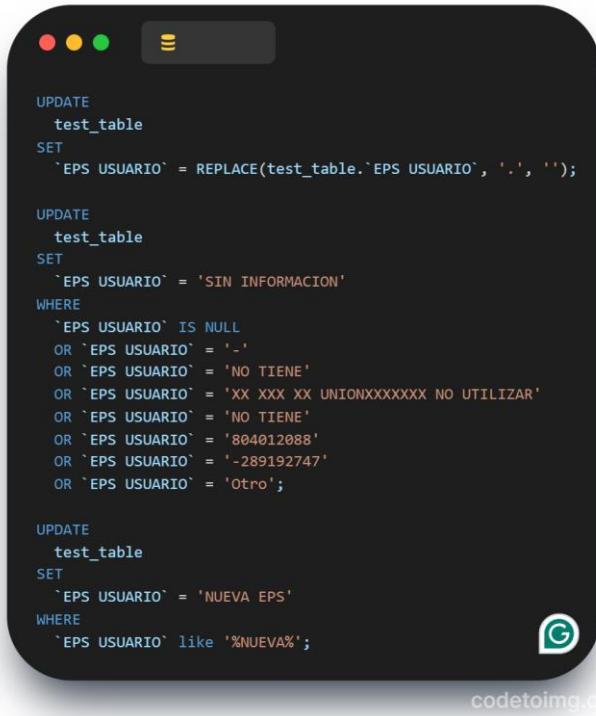
```
INSERT INTO ADMINISTRADORAS (ID_ADMINISTRADORA, ID_TIPO_ADMINISTRADORA, NOMBRE_ADMINISTRADORA)
SELECT `Código Administradora`, ID_TIPO_ADMINISTRADORA, NOMBRE_ADMINISTRADORA
FROM (
    SELECT IF(`Código Administradora` = 'SIN INFORMACION', 0, `Código Administradora`)
    AS `Código Administradora`,
    ID_TIPO_ADMINISTRADORA,
    NOMBRE_ADMINISTRADORA,
    ROW_NUMBER() OVER(PARTITION BY IF(`Código Administradora` = 'SIN INFORMACION', 0, `Código Administradora`)
    ORDER BY `Código Administradora`) AS rn
    FROM (
        SELECT
            `Código Administradora`,
            TIPO_ADMINISTRADORA.ID_TIPO_ADMINISTRADORA,
            test_table.Administradora AS NOMBRE_ADMINISTRADORA
        FROM
            test_table
            JOIN
            TIPO_ADMINISTRADORA ON test_table.`Tipo Administradora` = TIPO_ADMINISTRADORA.TIPO_ADMINISTRADORA
        WHERE
            test_table.Administradora IS NOT NULL
        ) AS inner_temp
    ) AS outer_temp
WHERE rn = 1;
```

codetomsg.co

- **Tabla “EPS”:** se implementó un conjunto de scripts que realiza operaciones de limpieza y normalización de datos en la columna ‘EPS USUARIO’ de la tabla ‘test_table’, seguido de esto se realizó la inserción de datos únicos en la tabla ‘EPS’. Para la limpieza de datos en primer lugar se eliminó los puntos (‘.’) en todos los valores, al igual que valores nulos e información problemática. De igual forma se reemplazaron varios nombres dado que su diferencia está radicada en uno o dos caracteres, únicamente, ver figura 8.

Figura 8.

Normalización de datos en la columna EPS USUARIO



```
UPDATE
    test_table
SET
    `EPS USUARIO` = REPLACE(test_table.`EPS USUARIO`, '.', '');

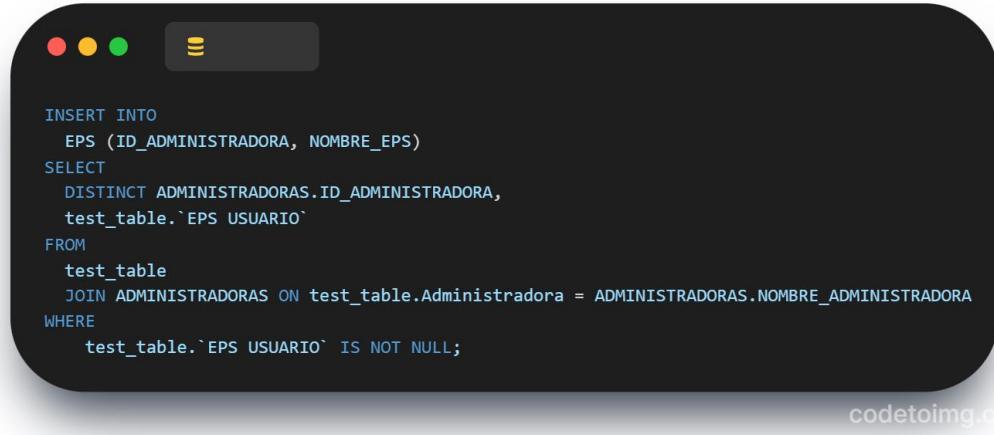
UPDATE
    test_table
SET
    `EPS USUARIO` = 'SIN INFORMACION'
WHERE
    `EPS USUARIO` IS NULL
    OR `EPS USUARIO` = '_'
    OR `EPS USUARIO` = 'NO TIENE'
    OR `EPS USUARIO` = 'XX XXX XX UNIONXXXXXX NO UTILIZAR'
    OR `EPS USUARIO` = 'NO TIENE'
    OR `EPS USUARIO` = '804012088'
    OR `EPS USUARIO` = '-289192747'
    OR `EPS USUARIO` = 'Otro';

UPDATE
    test_table
SET
    `EPS USUARIO` = 'NUEVA EPS'
WHERE
    `EPS USUARIO` like '%NUEVA%';
```

Para la inserción de la información limpia y corregida, en primer lugar, se restablece el valor de autoincremento de la tabla EPS y posteriormente se insertan valores únicos de EPS USUARIO y ID_ADMINISTRADORA en la tabla EPS desde test_table, realizando una unión con la tabla ADMINISTRADORAS para obtener el ID_ADMINISTRADORA correspondiente, ver figura 9.

Figura 9.

Inserción de datos en la tabla EPS



```
INSERT INTO
  EPS (ID_ADMINISTRADORA, NOMBRE_EPS)
SELECT
  DISTINCT ADMINISTRADORAS.ID_ADMINISTRADORA,
  test_table.`EPS USUARIO`
FROM
  test_table
  JOIN ADMINISTRADORAS ON test_table.Administradora = ADMINISTRADORAS.NOMBRE_ADMINISTRADORA
WHERE
  test_table.`EPS USUARIO` IS NOT NULL;
```

codetoimg.co

- **Tabla “GRUPOS_ETAREOS”:** en primer lugar, se realiza una limpieza básica para actualizar los valores '0' en las columnas GRUPO ETAREO y CURSO DE VIDA a 'SIN INFORMACION'. (Ver figura 10)

Figura 10.

Normalización de datos en la columna CURSO DE VIDA



```
UPDATE test_table
SET `GRUPO ETAREO` = 'SIN INFORMACION'
WHERE `GRUPO ETAREO` = '0';

UPDATE test_table
SET `CURSO DE VIDA` = 'SIN INFORMACION'
WHERE `CURSO DE VIDA` = '0';
```

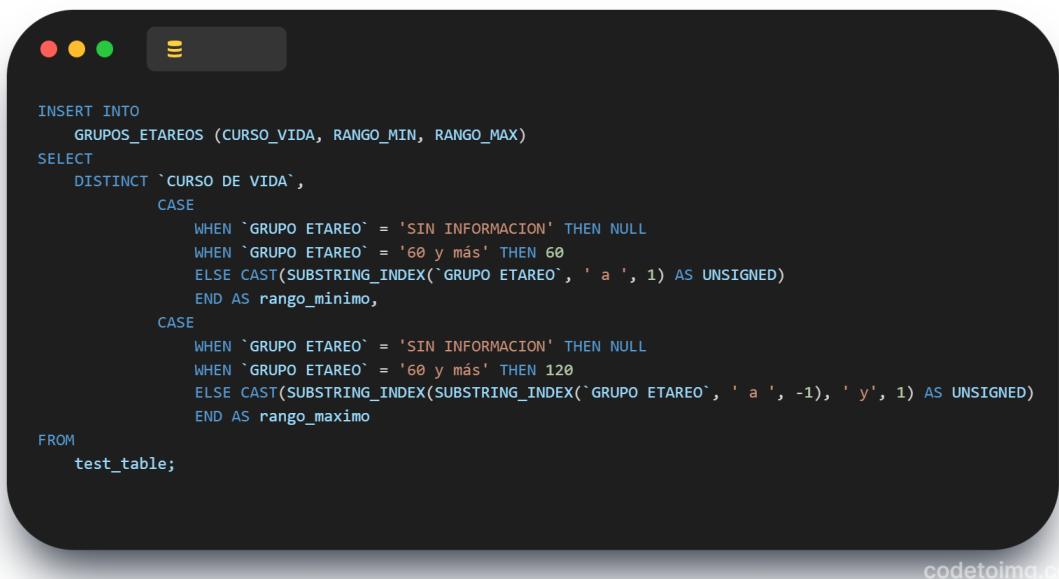
codetoimg.co

Posteriormente se realiza la inserción de los datos donde se restablece el valor de autoincremento de la tabla GRUPOS_ETAREOS. Posteriormente, se insertan valores únicos de CURSO DE VIDA, RANGO_MIN, y RANGO_MAX en la tabla GRUPOS_ETAREOS

desde test_table. RANGO_MIN se calcula a partir del primer número en GRUPO ETAREO o se asigna NULL para 'SIN INFORMACION' y 60 para '60 y más'. RANGO_MAX se calcula a partir del último número en GRUPO ETAREO o se asigna NULL para 'SIN INFORMACION' y 120 para '60 y más'. Este conjunto de operaciones asegura que los datos de las columnas GRUPO ETAREO y CURSO DE VIDA estén limpios y estructurados antes de ser insertados en la tabla GRUPOS_ETAREOS de manera organizada y sin duplicados.

Figura 11.

Inserción de datos en la tabla GRUPOS_ETAREOS



```

INSERT INTO
    GRUPOS_ETAREOS (CURSO_VIDA, RANGO_MIN, RANGO_MAX)
SELECT
    DISTINCT `CURSO DE VIDA`,
    CASE
        WHEN `GRUPO ETAREO` = 'SIN INFORMACION' THEN NULL
        WHEN `GRUPO ETAREO` = '60 y más' THEN 60
        ELSE CAST(SUBSTRING_INDEX(`GRUPO ETAREO`, ' a ', 1) AS UNSIGNED)
        END AS rango_minimo,
    CASE
        WHEN `GRUPO ETAREO` = 'SIN INFORMACION' THEN NULL
        WHEN `GRUPO ETAREO` = '60 y más' THEN 120
        ELSE CAST(SUBSTRING_INDEX(SUBSTRING_INDEX(`GRUPO ETAREO`, ' a ', -1), ' y', 1) AS UNSIGNED)
        END AS rango_maximo
FROM
    test_table;

```

codetomsg.com

- **Tabla “AFILIACIONES”:** en este caso, el script reinicia el valor de autoincremento en la tabla AFILIACIONES y luego inserta valores únicos de la columna TIPO AFILIACIÓN de la tabla test_table en la tabla AFILIACIONES, asegurando que solo se registren tipos de afiliación distintos y sin duplicados, ver figura 12.

Figura 12.

Inserción de datos en la tabla AFILIACIONES



```
ALTER TABLE
    AFILIACIONES AUTO_INCREMENT = 1;

INSERT INTO
    AFILIACIONES (TIPO_AFILIACION)
SELECT
    DISTINCT `TIPO AFILIACIÓN`
FROM
    test_table;
```

codetoimg.co

- **Tabla “VEHICULOS”:** similar al anterior, el script comienza reiniciando el valor de autoincremento en la tabla VEHICULOS y luego procede a insertar valores únicos de la columna TIPO DE VEHÍCULO de la tabla test_table en la tabla VEHICULOS, garantizando que solo se registren tipos de vehículos distintos y sin duplicados, ver figura 13.

Figura 13.

Inserción de datos en la tabla VEHICULOS



```
ALTER TABLE
    VEHICULOS AUTO_INCREMENT = 1;

INSERT INTO
    VEHICULOS (TIPO_VEHICULO)
SELECT
    DISTINCT `TIPO DE VEHÍCULO`
FROM
    test_table;
```

codetoimg.co

- **Tabla “IPS”:** de igual manera, el script inicia reiniciando el valor de autoincremento en la tabla IPS, luego inserta valores únicos de la columna IPS de la tabla test_table en la tabla IPS, asegurando que solo se registren nombres de IPS distintos y sin duplicados, ver figura 14:

Figura 14.

Inserción de datos en la tabla IPS



```

ALTER TABLE
    IPS AUTO_INCREMENT = 1;

INSERT INTO
    IPS (NOMBRE_IPS)
SELECT
    DISTINCT IPS
FROM
    test_table;
  
```

The screenshot shows a terminal window with a dark theme. At the top, there are three colored window control buttons (red, yellow, green) and a menu icon. The main area contains the following MySQL script:

```

ALTER TABLE
    IPS AUTO_INCREMENT = 1;

INSERT INTO
    IPS (NOMBRE_IPS)
SELECT
    DISTINCT IPS
FROM
    test_table;
  
```

At the bottom right of the terminal window, the watermark "codetomg.com" is visible.

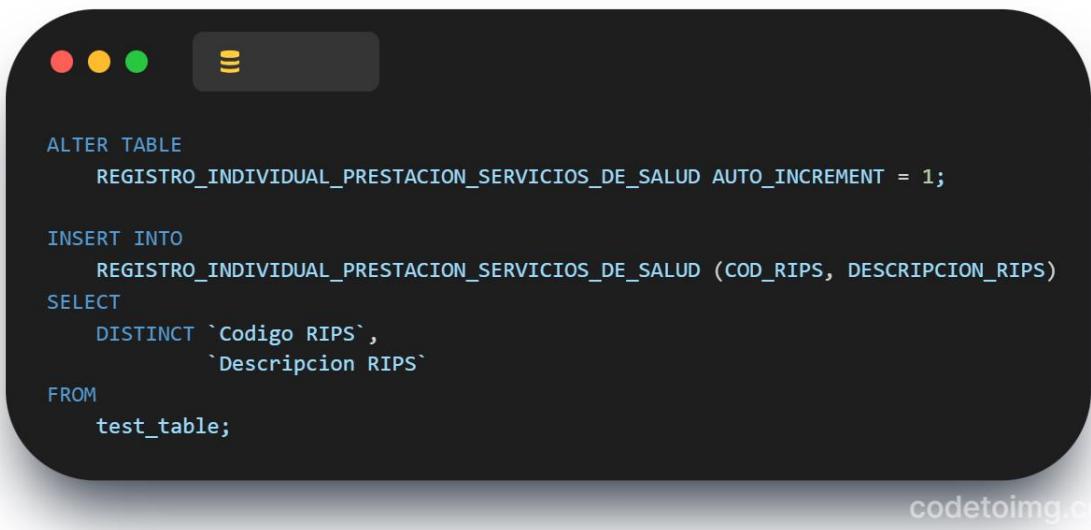
- **Tabla**

“REGISTRO_INDIVIDUAL_PRESTACION_SERVICIOS_DE_SALUD”: el script primero reinicia el valor de autoincremento en la tabla REGISTRO_INDIVIDUAL_PRESTACION_SERVICIOS_DE_SALUD, luego inserta valores únicos de las columnas Código RIPS y Descripción RIPS de la tabla test_table en la tabla REGISTRO_INDIVIDUAL_PRESTACION_SERVICIOS_DE_SALUD, asegurando que solo se registren códigos RIPS y descripciones únicas sin duplicados, ver figura 15.

Figura 15.

Inserción de datos en la tabla

REGISTRO_INDIVIDUAL_PRESTACION_SERVICIOS_DE_SALUD



```
ALTER TABLE
    REGISTRO_INDIVIDUAL_PRESTACION_SERVICIOS_DE_SALUD AUTO_INCREMENT = 1;

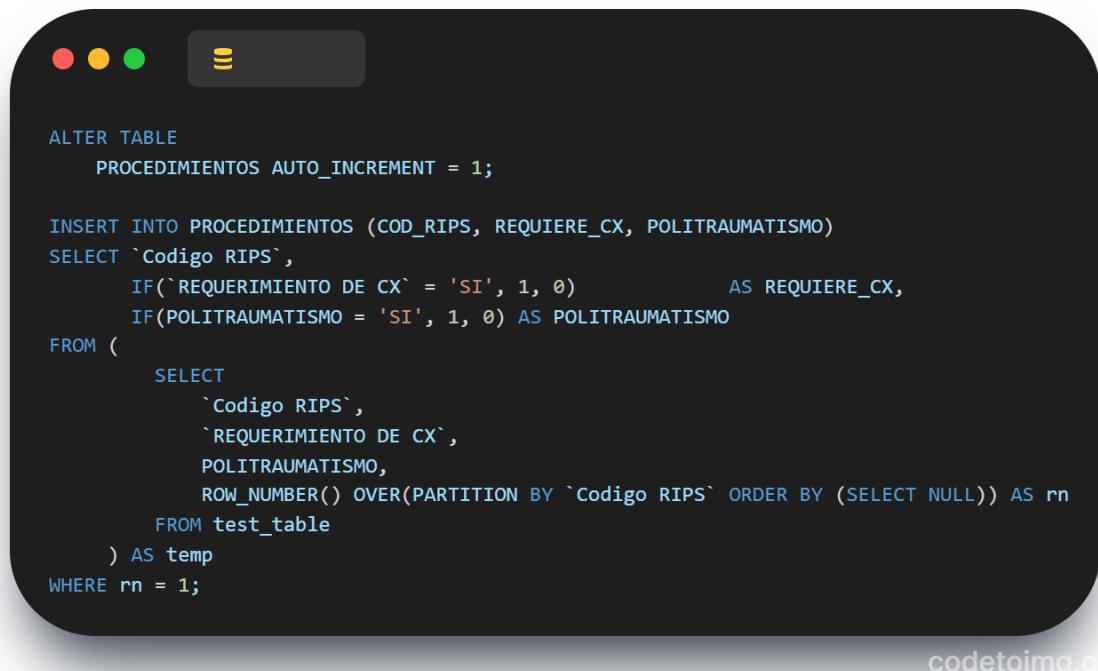
INSERT INTO
    REGISTRO_INDIVIDUAL_PRESTACION_SERVICIOS_DE_SALUD (COD_RIPS, DESCRIPCION_RIPS)
SELECT
    DISTINCT `Codigo RIPS`,
    `Descripcion RIPS`
FROM
    test_table;
```

codetoimg.co

- **Tabla “PROCEDIMIENTOS”:** en este caso, el script comienza reiniciando el valor de autoincremento en la tabla PROCEDIMIENTOS. Luego, inserta valores únicos de las columnas Código RIPS, REQUERIMIENTO DE CX y POLITRAUMATISMO de la tabla test_table en la tabla PROCEDIMIENTOS, donde se transforman los valores de REQUERIMIENTO DE CX y POLITRAUMATISMO en valores binarios (0 o 1) y se asegura que solo se inserte una fila por cada código RIPS único, ver figura 16.

Figura 16.

Inserción de datos en la tabla PROCEDIMIENTOS



```
ALTER TABLE
PROCEDIMIENTOS AUTO_INCREMENT = 1;

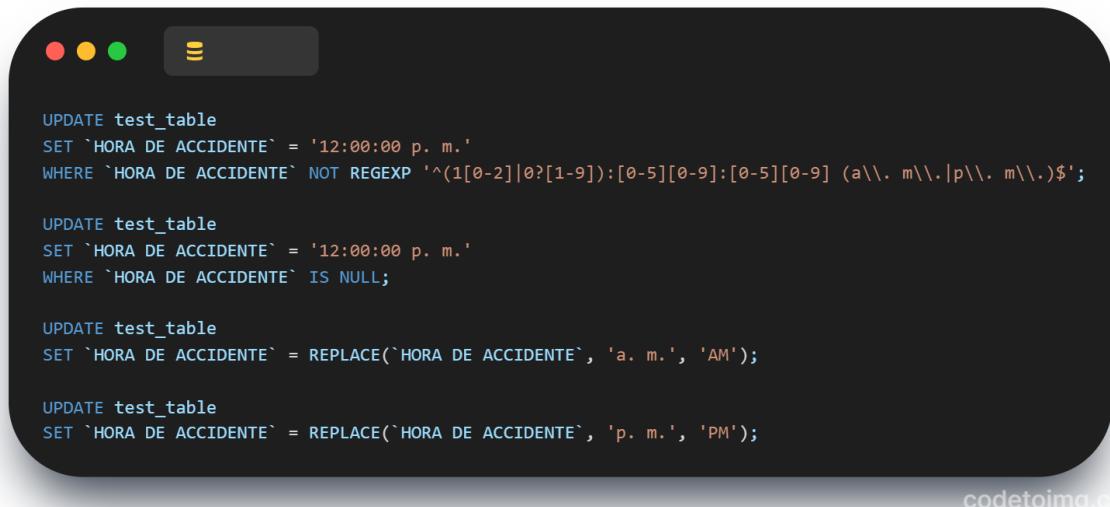
INSERT INTO PROCEDIMIENTOS (COD_RIPS, REQUIERE_CX, POLITRAUMATISMO)
SELECT `Codigo RIPS`,
       IF(`REQUERIMIENTO DE CX` = 'SI', 1, 0) AS REQUIERE_CX,
       IF(POLITRAUMATISMO = 'SI', 1, 0) AS POLITRAUMATISMO
FROM (
    SELECT
        `Codigo RIPS`,
        `REQUERIMIENTO DE CX`,
        POLITRAUMATISMO,
        ROW_NUMBER() OVER(PARTITION BY `Codigo RIPS` ORDER BY (SELECT NULL)) AS rn
    FROM test_table
) AS temp
WHERE rn = 1;
```

codetomsg.co

- **Tabla “INFORMES_ACCIDENTES”:** el script presenta operaciones para normalizar y transferir datos de la tabla test_table a la tabla INFORMES_ACCIDENTES, junto con la inicialización del valor de autoincremento en la tabla de destino, de manera general se presentan los siguientes cambios:
Normalización de datos de hora de accidente: se actualizan los valores de la columna HORA DE ACCIDENTE que no cumplen con el formato establecido, transformando cualquier formato de hora incorrecto en '12:00:00 p. m.' para mantener la consistencia. Se reemplazan los indicadores de mañana y tarde ('a. m.' y 'p. m.') por 'AM' y 'PM', respectivamente, para estandarizar el formato de las horas. (Ver figura 17)

Figura 17.

Normalización de la columna HORA DE ACCIDENTE



```
UPDATE test_table
SET `HORA DE ACCIDENTE` = '12:00:00 p. m.'
WHERE `HORA DE ACCIDENTE` NOT REGEXP '^([0-2]|0?[1-9]):[0-5][0-9]:[0-5] ([a\\. m\\\\.|p\\\\. m\\\\.])$';

UPDATE test_table
SET `HORA DE ACCIDENTE` = '12:00:00 p. m.'
WHERE `HORA DE ACCIDENTE` IS NULL;

UPDATE test_table
SET `HORA DE ACCIDENTE` = REPLACE(`HORA DE ACCIDENTE`, 'a. m.', 'AM');

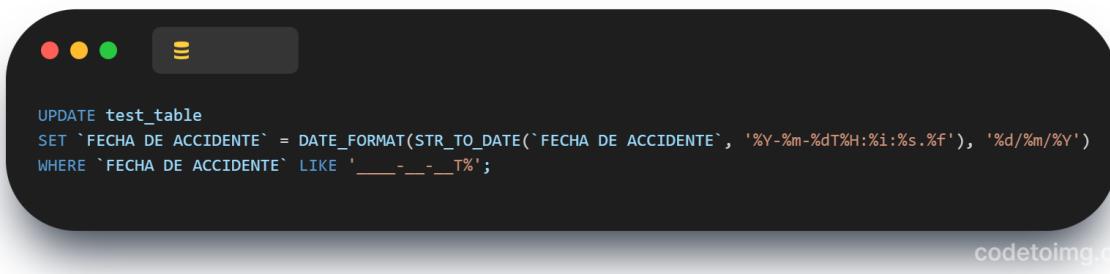
UPDATE test_table
SET `HORA DE ACCIDENTE` = REPLACE(`HORA DE ACCIDENTE`, 'p. m.', 'PM');
```

codetomg.co

Normalización de datos de fecha de accidente: se convierten los valores de la columna FECHA DE ACCIDENTE en el formato deseado ('dd/mm/aaaa'), extrayendo la fecha de los datos originales, que podrían estar en formato ISO8601 ('YYYY-MM-DDTHH: MM.SSS'). (Ver figura 18)

Figura 18.

Normalización de la columna FECHA DE ACCIDENTE



```
UPDATE test_table
SET `FECHA DE ACCIDENTE` = DATE_FORMAT(STR_TO_DATE(`FECHA DE ACCIDENTE`, '%Y-%m-%dT%H:%i:%s.%f'), '%d/%m/%Y')
WHERE `FECHA DE ACCIDENTE` LIKE '__-__-__T%';
```

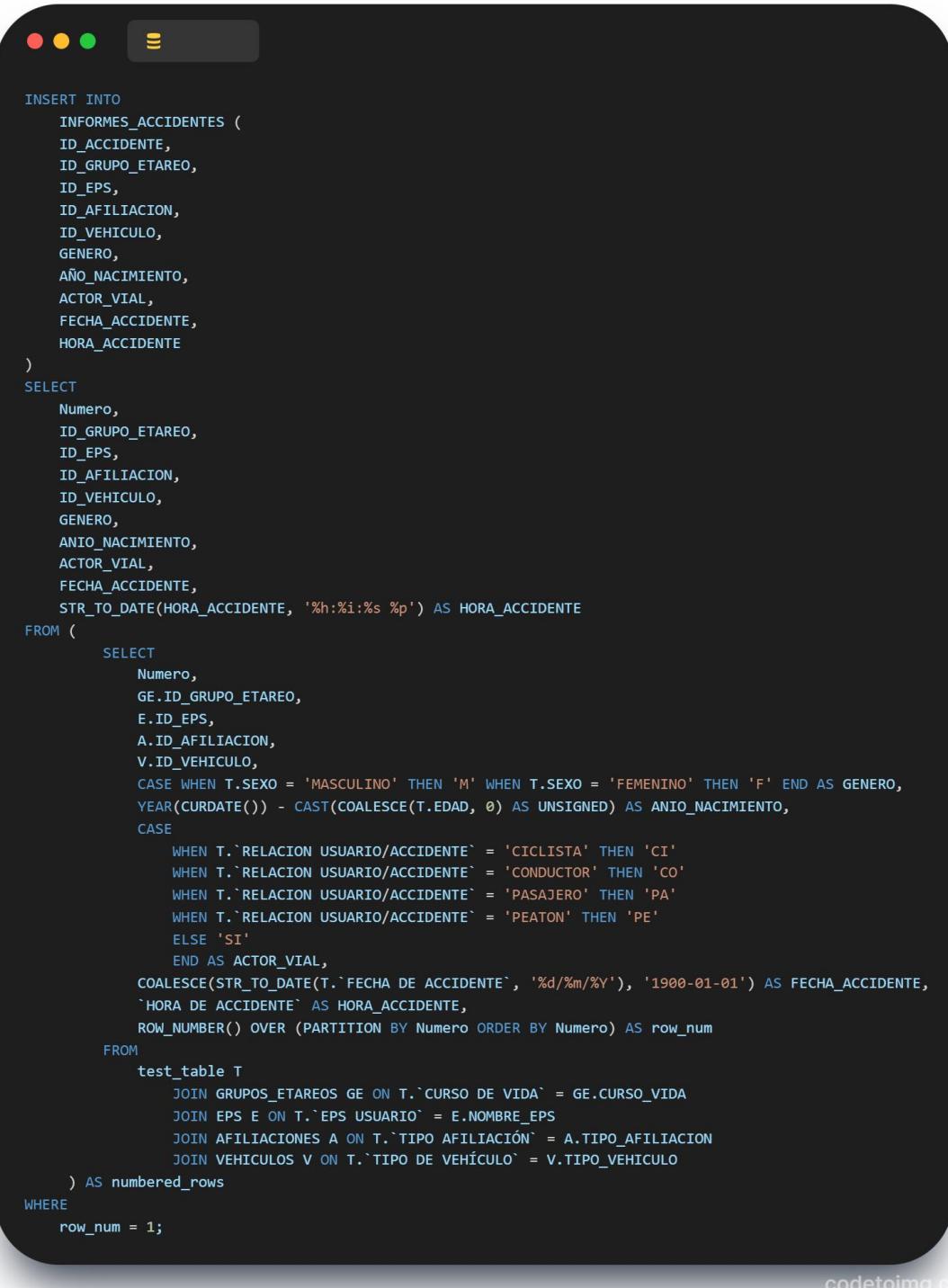
codetomg.co

- **Inserción de Datos en la Tabla de Destino:** se realiza una inserción en la tabla INFORMES_ACCIDENTES seleccionando valores únicos de las columnas requeridas de la tabla test_table. Se realiza un filtrado para garantizar que solo se inserte una fila por cada número de accidente único. Los datos seleccionados incluyen: número de accidente, identificación del grupo etario, EPS, tipo de afiliación,

tipo de vehículo, género, año de nacimiento, actor vial, fecha de accidente (normalizada) y hora de accidente (normalizada).

Figura 19.

Inserción de datos en la tabla INFORMES_ACCIDENTES



```
INSERT INTO
    INFORMES_ACCIDENTES (
        ID_ACCIDENTE,
        ID_GRUPO_ETAREO,
        ID_EPS,
        ID_AFILIACION,
        ID_VEHICULO,
        GENERO,
        AÑO_NACIMIENTO,
        ACTOR_VIAL,
        FECHA_ACCIDENTE,
        HORA_ACCIDENTE
    )
SELECT
    Numero,
    ID_GRUPO_ETAREO,
    ID_EPS,
    ID_AFILIACION,
    ID_VEHICULO,
    GENERO,
    ANIO_NACIMIENTO,
    ACTOR_VIAL,
    FECHA_ACCIDENTE,
    STR_TO_DATE(HORA_ACCIDENTE, '%h:%i:%s %p') AS HORA_ACCIDENTE
FROM (
    SELECT
        Numero,
        GE.ID_GRUPO_ETAREO,
        E.ID_EPS,
        A.ID_AFILIACION,
        V.ID_VEHICULO,
        CASE WHEN T.SEXO = 'MASCULINO' THEN 'M' WHEN T.SEXO = 'FEMENINO' THEN 'F' END AS GENERO,
        YEAR(CURDATE()) - CAST(COALESCE(T.EDAD, 0) AS UNSIGNED) AS ANIO_NACIMIENTO,
        CASE
            WHEN T.`RELACION USUARIO/ACCIDENTE` = 'CICLISTA' THEN 'CI'
            WHEN T.`RELACION USUARIO/ACCIDENTE` = 'CONDUTOR' THEN 'CO'
            WHEN T.`RELACION USUARIO/ACCIDENTE` = 'PASAJERO' THEN 'PA'
            WHEN T.`RELACION USUARIO/ACCIDENTE` = 'PEATON' THEN 'PE'
            ELSE 'SI'
        END AS ACTOR_VIAL,
        COALESCE(STR_TO_DATE(T.`FECHA DE ACCIDENTE`, '%d/%m/%Y'), '1900-01-01') AS FECHA_ACCIDENTE,
        `HORA DE ACCIDENTE` AS HORA_ACCIDENTE,
        ROW_NUMBER() OVER (PARTITION BY Numero ORDER BY Numero) AS row_num
    FROM
        test_table T
        JOIN GRUPOS_ETAREOS GE ON T.`CURSO DE VIDA` = GE.CURSO_VIDA
        JOIN EPS E ON T.`EPS USUARIO` = E.NOMBRE_EPS
        JOIN AFILIACIONES A ON T.TIPO_AFILIACION = A.TIPO_AFILIACION
        JOIN VEHICULOS V ON T.TIPO_DE_VEHICULO = V.TIPO_VEHICULO
    ) AS numbered_rows
WHERE
    row_num = 1;
```

- **Tabla “ATENCIONES_HOSPITALARIAS”:** siendo esta la tabla final, el script realiza una serie de operaciones para preparar los datos de la tabla test_table y luego insertarlos en la tabla ATENCIONES_HOSPITALARIAS, como se detalla a continuación:

Normalización de Fechas y Horas: Se normalizan las fechas de ingreso a la IPS (FECHA DE INGRESO IPS) y de atención médica (FECHA DE ATENCIÓN MÉDICA) al formato 'dd/mm/aaaa'. Se establece la fecha de ingreso a la IPS como '01/01/1900' para los casos en los que está ausente. Se normalizan las horas de ingreso a la IPS (HORA DE INGRESO IPS) al formato 'hh:mm AM/PM'.

Figura 20.

Normalización de las columnas FECHA DE INGRESO IPS y HORA DE INGRESO IPS

```

ALTER TABLE
    ATENCIONES_HOSPITALARIAS AUTO_INCREMENT = 1;
UPDATE test_table
SET `FECHA DE INGRESO IPS` = DATE_FORMAT(STR_TO_DATE(`FECHA DE INGRESO IPS`, '%Y-%m-%dT%H:%i:%s.%f'), '%d/%m/%Y')
WHERE `FECHA DE INGRESO IPS` LIKE '____-__-_T%';

UPDATE test_table
SET `FECHA DE INGRESO IPS` = '01/01/1900'
WHERE `FECHA DE INGRESO IPS` IS NULL;

UPDATE test_table
SET `FECHA DE INGRESO IPS` = STR_TO_DATE(`FECHA DE INGRESO IPS`, '%d/%m/%Y');

UPDATE test_table
SET `HORA DE INGRESO IPS` = STR_TO_DATE(REPLACE(test_table.`HORA DE INGRESO IPS`, 'a. m.', 'AM'), '%h:%i:%s %p')
WHERE test_table.`HORA DE INGRESO IPS` LIKE '%a. m.';

UPDATE test_table
SET `HORA DE INGRESO IPS` = REPLACE(test_table.`HORA DE INGRESO IPS`, 'p. m.', 'PM')
WHERE test_table.`HORA DE INGRESO IPS` LIKE '%p. m.';

UPDATE test_table
SET `HORA DE INGRESO IPS` = '12:00:00 PM'
WHERE test_table.`HORA DE INGRESO IPS` IS NULL;

UPDATE test_table
SET `HORA DE INGRESO IPS` = REPLACE(test_table.`HORA DE INGRESO IPS`, 'AM', ':00 AM')
WHERE test_table.`HORA DE INGRESO IPS` REGEXP '^[0-9]{1,2}:[0-9]{2}[AP]M$';

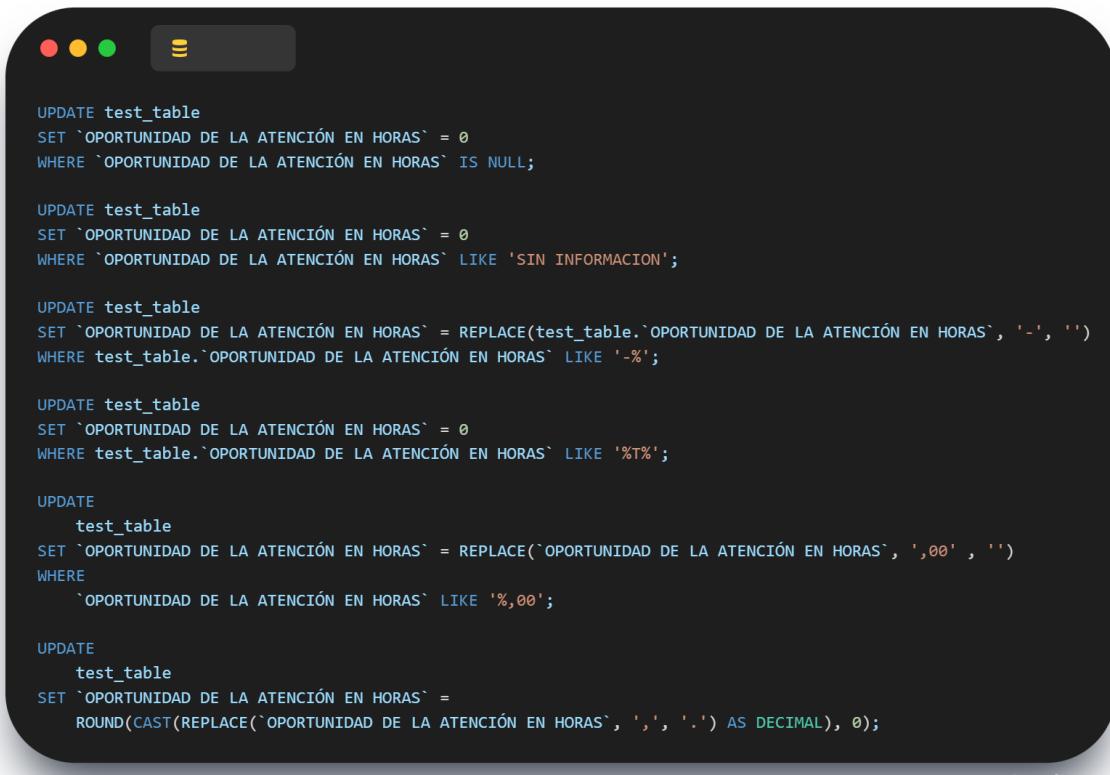
UPDATE test_table
SET `HORA DE INGRESO IPS` = REPLACE(test_table.`HORA DE INGRESO IPS`, 'PM', ':00 PM')
WHERE test_table.`HORA DE INGRESO IPS` REGEXP '^[0-9]{1,2}:[0-9]{2}[AP]M$';

```

- **Normalización y Conversión de Datos de Duración de Atención:** se establece la duración de la atención en horas (OPORTUNIDAD DE LA ATENCIÓN EN HORAS) como 0 para los valores nulos o con texto 'SIN INFORMACIÓN'. Se eliminan los caracteres innecesarios y se convierten los valores numéricos de duración de atención de coma flotante a entero.

Figura 20.

Normalización de la columna OPORTUNIDAD DE ATENCIÓN EN HORAS



```

UPDATE test_table
SET `OPORTUNIDAD DE LA ATENCIÓN EN HORAS` = 0
WHERE `OPORTUNIDAD DE LA ATENCIÓN EN HORAS` IS NULL;

UPDATE test_table
SET `OPORTUNIDAD DE LA ATENCIÓN EN HORAS` = 0
WHERE `OPORTUNIDAD DE LA ATENCIÓN EN HORAS` LIKE 'SIN INFORMACION';

UPDATE test_table
SET `OPORTUNIDAD DE LA ATENCIÓN EN HORAS` = REPLACE(test_table.`OPORTUNIDAD DE LA ATENCIÓN EN HORAS`, ',', '')
WHERE test_table.`OPORTUNIDAD DE LA ATENCIÓN EN HORAS` LIKE '-%';

UPDATE test_table
SET `OPORTUNIDAD DE LA ATENCIÓN EN HORAS` = 0
WHERE test_table.`OPORTUNIDAD DE LA ATENCIÓN EN HORAS` LIKE '%T%';

UPDATE
    test_table
SET `OPORTUNIDAD DE LA ATENCIÓN EN HORAS` = REPLACE(`OPORTUNIDAD DE LA ATENCIÓN EN HORAS`, ',00', '')
WHERE
    `OPORTUNIDAD DE LA ATENCIÓN EN HORAS` LIKE '%,00';

UPDATE
    test_table
SET `OPORTUNIDAD DE LA ATENCIÓN EN HORAS` =
    ROUND(CAST(REPLACE(`OPORTUNIDAD DE LA ATENCIÓN EN HORAS`, ',', '.') AS DECIMAL), 0);

```

codetoiimg.co

- **Normalización de la Condición de Egreso:** se convierten las diferentes condiciones de egreso en códigos estandarizados ('V', 'M', 'S').
- **Normalización del Triage:** se convierten las categorías de triage en códigos estandarizados ('I', 'II', 'III', 'IV', 'SI').
- **Determinación de Referencia a Otra IPS:** se convierte la información sobre si el paciente fue referido a otra IPS en valores binarios (0 o 1).

- **Inserción de Datos en la Tabla de Destino:** se insertan los datos normalizados y preparados de la tabla test_table en la tabla ATENCIONES_HOSPITALARIAS, relacionando las claves foráneas con las tablas IPS, PROCEDIMIENTOS e INFORMES_ACCIDENTES.

Figura 21.

Inserción de datos en la tabla ATENCIONES HOSPITALARIAS

```

INSERT INTO ATENCIONES_HOSPITALARIAS (
    ID_ACCIDENTE,
    ID_IPS,
    ID_PROCEDIMIENTO,
    FECHA_INGRESO_IPS,
    HORA_INGRESO_IPS,
    FECHA_ATENCION_MEDICA,
    HORA_ATENCION_MEDICA,
    DURACION_ATENCION_HRS,
    COSTO,
   CONDICION_EGRESO,
    TRIAGE,
    REFERIDO_OTRA_IPS
)
SELECT ID_ACCIDENTE,
    ID_IPS,
    ID_PROCEDIMIENTO,
    `FECHA DE INGRESO IPS`,
    `HORA DE INGRESO IPS`,
    `FECHA DE ATENCION MÉDICA`,
    `HORA DE ATENCION MÉDICA`,
    `OPORTUNIDAD DE LA ATENCIÓN EN HORAS` AS DURACION_ATENCION_HRS,
    COSTOS,
    CASE
        WHEN `CONDICION EGRESO` = 'VIVO' THEN 'V'
        WHEN `CONDICION EGRESO` = 'MUERTO' THEN 'M'
        ELSE 'S'
    END AS CONDICION_EGRESO,
    CASE
        WHEN `OPORTUNIDAD CITAS DE CONTROL` = 'I' THEN 'I'
        WHEN `OPORTUNIDAD CITAS DE CONTROL` = 'II' THEN 'II'
        WHEN `OPORTUNIDAD CITAS DE CONTROL` = 'III' THEN 'III'
        WHEN `OPORTUNIDAD CITAS DE CONTROL` = 'IV' THEN 'IV'
        ELSE 'SI'
    END AS TRIAGE,
    IF(`REFERIDO A OTRA IPS` = 'SI', 1, 0) AS REFERIDO_OTRA_IPS
FROM test_table t
JOIN IPS i ON t.IPS = i.NOMBRE_IPS
JOIN PROCEDIMIENTOS p ON t.`Codigo RIPS` = p.COD_RIPS
JOIN INFORMES_ACCIDENTES ia ON t.Numero = ia.ID_ACCIDENTE;

```

Funciones y Operadores en MySQL

Al desarrollar este proyecto fue necesario realizar una investigación adicional ya que algunas de las funciones y operadores cambian en comparación con Oracle Database, teniendo en cuenta lo visto en clase a lo largo de este semestre. Los conceptos que fueron necesarios investigar son:

- **auto_increment:** es una propiedad de una columna que genera automáticamente un valor único y secuencial cada vez que se inserta un nuevo registro en la tabla.
- **Date:** Tipo de dato que almacena sólo la fecha en el formato YYYY-MM-DD
- **Time:** Tipo de dato que almacena solo la hora en el formato HH:MM: SS
- **Boolean:** Tipo de dato que almacena valores verdaderos (1) o falsos (0).
- **IF:** es una función que permite realizar evaluaciones condicionales dentro de una consulta SQL, su estructura es: IF(condición, valor_si_verdadero, valor_si_falso)
- **Not regexp:** se utiliza para filtrar filas que no coinciden con un patrón específico de expresiones regulares.
- **Unsigned:** convierte un valor a un tipo de dato sin signo; Estructura: cast (valor as unsigned).
- **str_to_date:** convierte una cadena de texto en una fecha; Estructura:
`str_to_date(cadena_fecha, formato)`
- **Date_format:** Manejar diferentes formatos de fecha para conversión y comparación.
- **Coalesce:** devuelve el primer valor no nulo en una lista de argumentos, evita valores nulos retornando el primer valor no nulo disponible; Estructura: Coalesce(valor1, valor2, ..., valorN).
- **Curdate():** devuelve la fecha actual del sistema.
- **Time_Format:** se utiliza para formatear valores de tiempo; Estructura:
`Time_Format(hora, formato).`

- **Substring_Index:** extrae una parte de una cadena de texto antes o después de un delimitador específico; Estructura: Substring_Index(cadena, delimitador, cuenta)
- **CAST:** convierte un valor de un tipo de dato a otro; Estructura: CAST(valor AS tipo_dato)
- **NOW():** devuelve la fecha y hora actual del sistema.
- **LPAD:** es una función que se utiliza para llenar una cadena con un conjunto específico de caracteres, agregándolos al inicio de la cadena hasta alcanzar una longitud deseada. LPAD(cadena, longitud, relleno)

Para la parte de funciones, procedimientos implementamos las estructuras correspondientes:

- **Triggers:** ver figura 22.

Figura 22.

Estructura implementada para los triggers.



```

CREATE TRIGGER nombre_trigger
{BEFORE | AFTER} {INSERT | UPDATE | DELETE}
ON nombre_tabla
FOR EACH ROW
BEGIN
    -- Código a ejecutar
END;

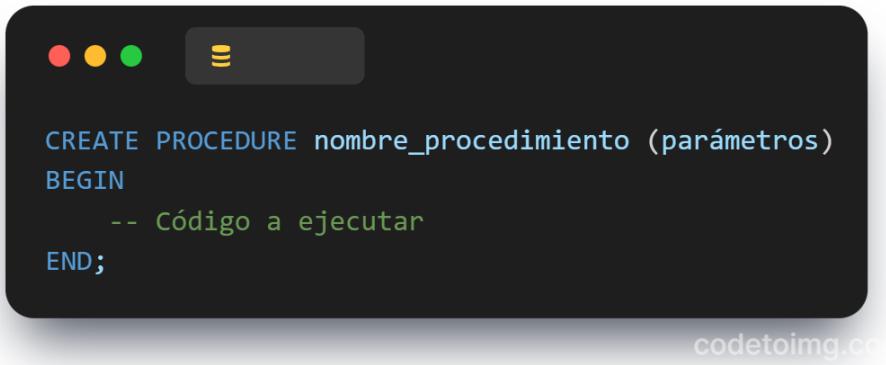
```

codetomsg.com

- **Procedimientos:** ver figura 23.

Figura 23.

Estructura implementada para los procedimientos.



A screenshot of a Mac OS X application window. The title bar has three colored window control buttons (red, yellow, green) and a close button. Below the title bar is a toolbar with a single icon. The main area contains the following MySQL code:

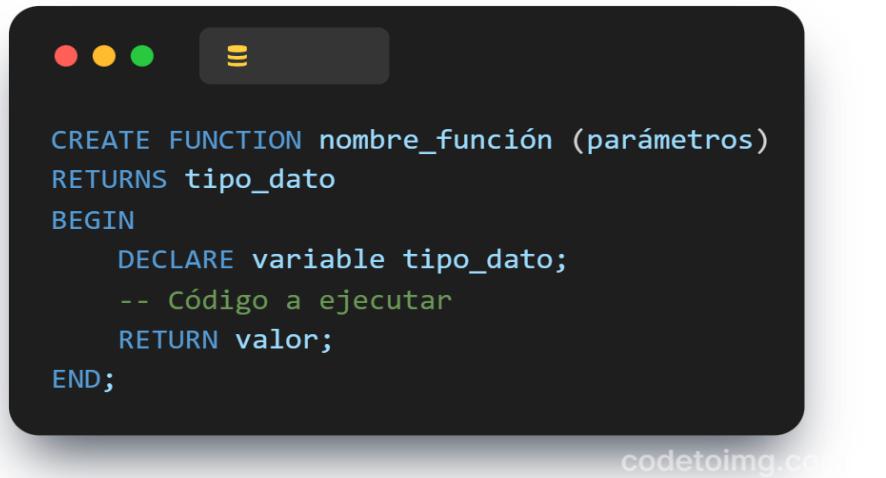
```
CREATE PROCEDURE nombre_procedimiento (parámetros)
BEGIN
    -- Código a ejecutar
END;
```

In the bottom right corner of the window, there is a watermark that reads "codetomsg.com".

- **Funciones:** ver figura 24.

Figura 24.

Estructura implementada para las funciones.



A screenshot of a Mac OS X application window. The title bar has three colored window control buttons (red, yellow, green) and a close button. Below the title bar is a toolbar with a single icon. The main area contains the following MySQL code:

```
CREATE FUNCTION nombre_función (parámetros)
RETURNS tipo_dato
BEGIN
    DECLARE variable tipo_dato;
    -- Código a ejecutar
    RETURN valor;
END;
```

In the bottom right corner of the window, there is a watermark that reads "codetomsg.com".

Triggers, Funciones y Procedimientos en MySQL

Triggers

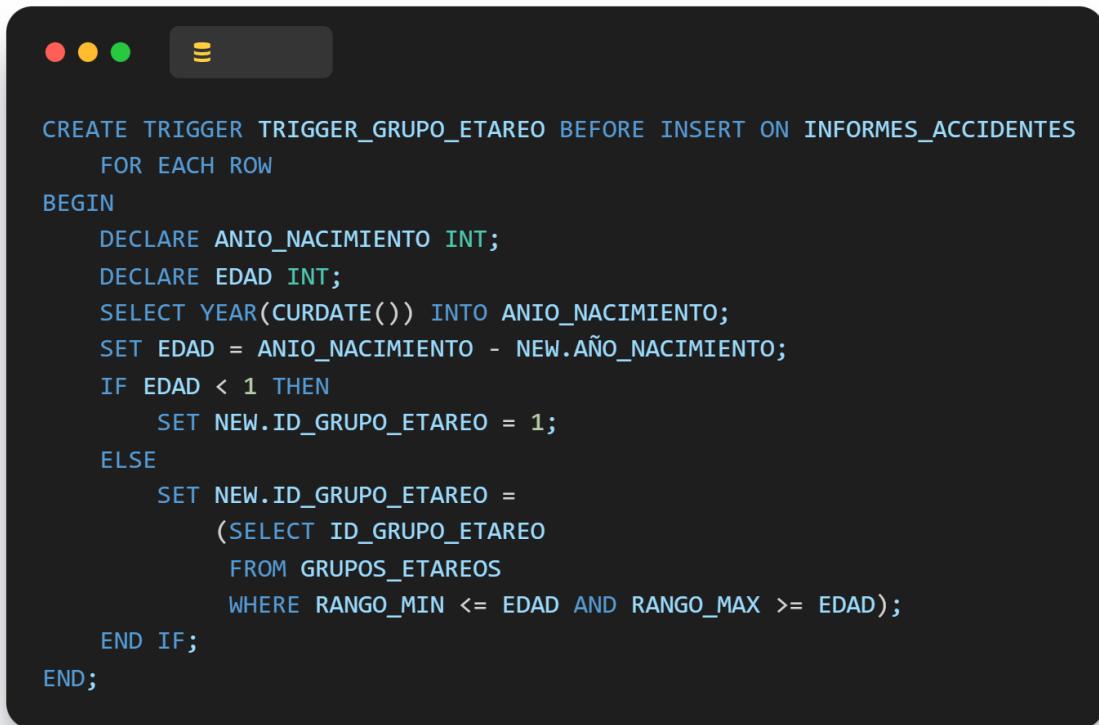
Calcular grupo etario

Calcula y asigna el grupo etario basado en el año de nacimiento del paciente antes de insertar un nuevo registro en la tabla informes_accidentes. Además, automatiza la asignación

del grupo etario, asegurando consistencia y precisión sin intervención manual. (Ver figura 25).

Figura 25.

contenido del trigger para calcular el grupo etario.



```
CREATE TRIGGER TRIGGER_GRUPO_ETAREO BEFORE INSERT ON INFORMES_ACCIDENTES
FOR EACH ROW
BEGIN
    DECLARE ANIO_NACIMIENTO INT;
    DECLARE EDAD INT;
    SELECT YEAR(CURDATE()) INTO ANIO_NACIMIENTO;
    SET EDAD = ANIO_NACIMIENTO - NEW.AÑO_NACIMIENTO;
    IF EDAD < 1 THEN
        SET NEW.ID_GRUPO_ETAREO = 1;
    ELSE
        SET NEW.ID_GRUPO_ETAREO =
            (SELECT ID_GRUPO_ETAREO
            FROM GRUPOS_ETAREOS
            WHERE RANGO_MIN <= EDAD AND RANGO_MAX >= EDAD);
    END IF;
END;
```

codetomg.com

Validar fechas atenciones hospitalarias

Valida que la fecha de ingreso a la IPS no sea posterior a la fecha de atención médica antes de insertar un nuevo registro en la tabla de atenciones hospitalarias. Garantiza la integridad de los datos asegurando que las fechas sean lógicas. (Ver figura 26)

Figura 26.

contenido del trigger para validar fechas de atenciones hospitalarias

```
CREATE TRIGGER VALIDAR_FECHAS_ATENCIONES_HOSPITALARIAS BEFORE INSERT ON ATENCIONES_HOSPITALARIAS
FOR EACH ROW
BEGIN
    IF NEW.FECHA_INGRESO_IPS > NEW.FECHA_ATENCION_MEDICA THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'La fecha de ingreso a la IPS no puede ser posterior a la fecha de atención médica.';
    END IF;
END;
```

codetom

Logs de cambios en informes_accidentes

Registra las operaciones de inserción, actualización y eliminación en la tabla

INFORMES_ACCIDENTES en una tabla de logs. Mantiene un historial de cambios para auditorías y seguimiento de modificaciones. (Ver figura 27)

Figura 27.

Contenido del trigger para logs de cambios en informes_accidentes.

```
CREATE TABLE LOG_INFORMES_ACCIDENTES (
    ID_LOG INT AUTO_INCREMENT PRIMARY KEY,
    ID_ACCIDENTE INT,
    ACCION VARCHAR(10),
    FECHA_HORA TIMESTAMP
);

CREATE TRIGGER LOG_INSERT_INFORME_ACCIDENTE
AFTER INSERT ON INFORMES_ACCIDENTES
FOR EACH ROW
BEGIN
    INSERT INTO LOG_INFORMES_ACCIDENTES (ID_ACCIDENTE, ACCION, FECHA_HORA)
    VALUES (NEW.ID_ACCIDENTE, 'INSERT', NOW());
END;

CREATE TRIGGER LOG_UPDATE_INFORME_ACCIDENTE
AFTER UPDATE ON INFORMES_ACCIDENTES
FOR EACH ROW
BEGIN
    INSERT INTO LOG_INFORMES_ACCIDENTES (ID_ACCIDENTE, ACCION, FECHA_HORA)
    VALUES (NEW.ID_ACCIDENTE, 'UPDATE', NOW());
END;

CREATE TRIGGER LOG_DELETE_INFORME_ACCIDENTE
AFTER DELETE ON INFORMES_ACCIDENTES
FOR EACH ROW
BEGIN
    INSERT INTO LOG_INFORMES_ACCIDENTES (ID_ACCIDENTE, ACCION, FECHA_HORA)
    VALUES (OLD.ID_ACCIDENTE, 'DELETE', NOW());
END;
```

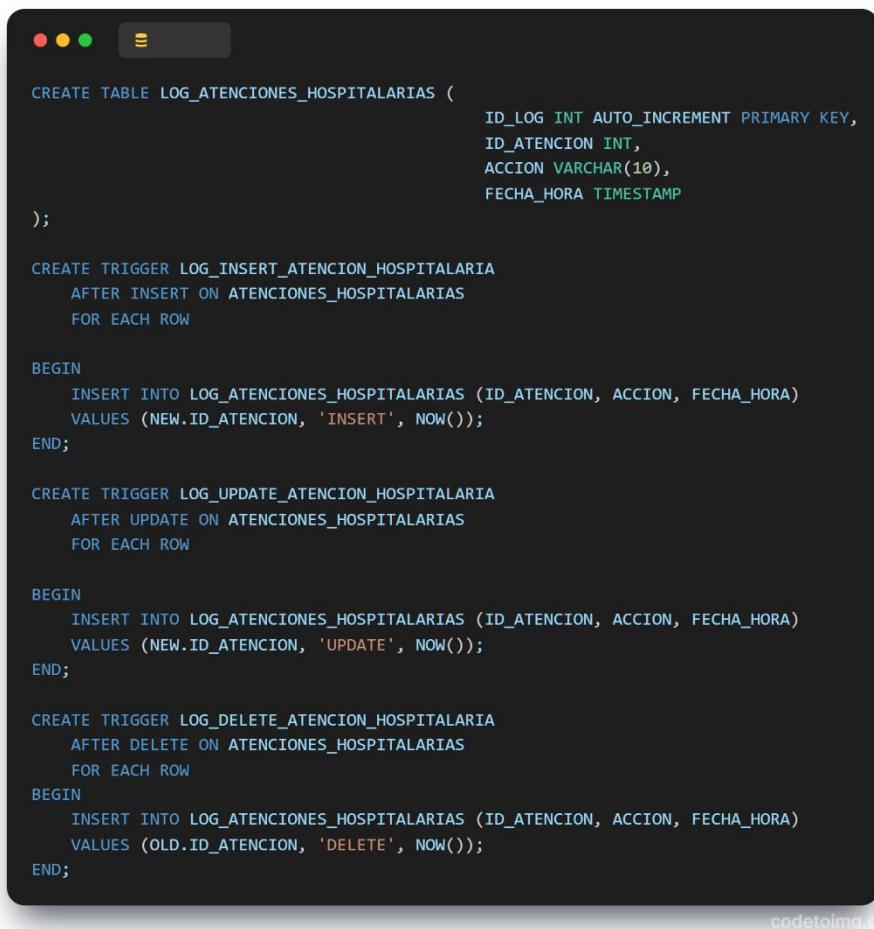
codetom

Logs de cambios en atenciones_hospitalarias

Registra las operaciones de inserción, actualización y eliminación en la tabla ATENCIONES_HOSPITALARIAS en una tabla de logs. Mantiene un historial de cambios para auditorías y seguimiento de modificaciones. (Ver figura 28)

Figura 28.

Contenido del trigger para logs de cambios en atenciones_hospitalarias.



```
CREATE TABLE LOG_ATENCIONES_HOSPITALARIAS (
    ID_LOG INT AUTO_INCREMENT PRIMARY KEY,
    ID_ATENCION INT,
    ACCION VARCHAR(10),
    FECHA_HORA TIMESTAMP
);

CREATE TRIGGER LOG_INSERT_ATENCION_HOSPITALARIA
    AFTER INSERT ON ATENCIONES_HOSPITALARIAS
    FOR EACH ROW
BEGIN
    INSERT INTO LOG_ATENCIONES_HOSPITALARIAS (ID_ATENCION, ACCION, FECHA_HORA)
    VALUES (NEW.ID_ATENCION, 'INSERT', NOW());
END;

CREATE TRIGGER LOG_UPDATE_ATENCION_HOSPITALARIA
    AFTER UPDATE ON ATENCIONES_HOSPITALARIAS
    FOR EACH ROW
BEGIN
    INSERT INTO LOG_ATENCIONES_HOSPITALARIAS (ID_ATENCION, ACCION, FECHA_HORA)
    VALUES (NEW.ID_ATENCION, 'UPDATE', NOW());
END;

CREATE TRIGGER LOG_DELETE_ATENCION_HOSPITALARIA
    AFTER DELETE ON ATENCIONES_HOSPITALARIAS
    FOR EACH ROW
BEGIN
    INSERT INTO LOG_ATENCIONES_HOSPITALARIAS (ID_ATENCION, ACCION, FECHA_HORA)
    VALUES (OLD.ID_ATENCION, 'DELETE', NOW());
END;
```

codetomsg.com

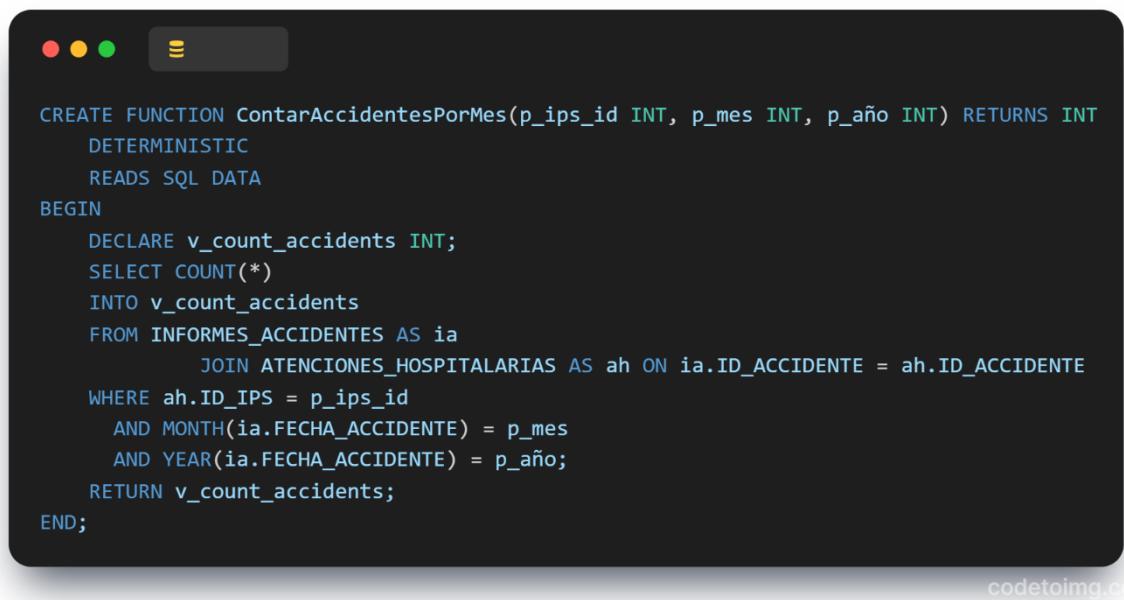
Funciones

Contar accidentes por mes

Cuenta el número de accidentes ocurridos en un mes y año específicos para una IPS dada. Proporciona estadísticas de accidentes para análisis mensual. (Ver figura 29)

Figura 29.

Contenido de la función para contar accidentes por mes.



```
CREATE FUNCTION ContarAccidentesPorMes(p_ips_id INT, p_mes INT, p_año INT) RETURNS INT
  DETERMINISTIC
  READS SQL DATA
BEGIN
  DECLARE v_count_accidents INT;
  SELECT COUNT(*)
    INTO v_count_accidents
   FROM INFORMES_ACCIDENTES AS ia
        JOIN ATENCIONES_HOSPITALARIAS AS ah ON ia.ID_ACCIDENTE = ah.ID_ACCIDENTE
  WHERE ah.ID_IPS = p_ips_id
    AND MONTH(ia.FECHA_ACCIDENTE) = p_mes
    AND YEAR(ia.FECHA_ACCIDENTE) = p_año;
  RETURN v_count_accidents;
END;
```

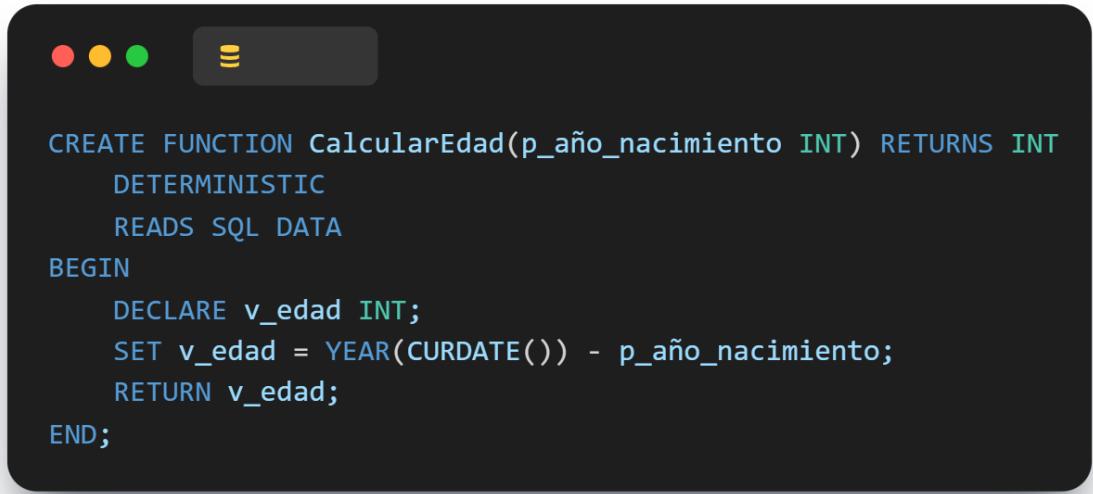
codetoimg.co

Calcular edad

Calcula la edad de una persona basada en su año de nacimiento. Facilita el cálculo de la edad para su uso en análisis y reportes. (Ver figura 30)

Figura 30.

Contenido de la función para calcular la edad.



```
CREATE FUNCTION CalcularEdad(p_año_nacimiento INT) RETURNS INT
  DETERMINISTIC
  READS SQL DATA
BEGIN
  DECLARE v_edad INT;
  SET v_edad = YEAR(CURDATE()) - p_año_nacimiento;
  RETURN v_edad;
END;
```

codetoimg.co

Procedimientos

Obtener costo total por tipo procedimiento

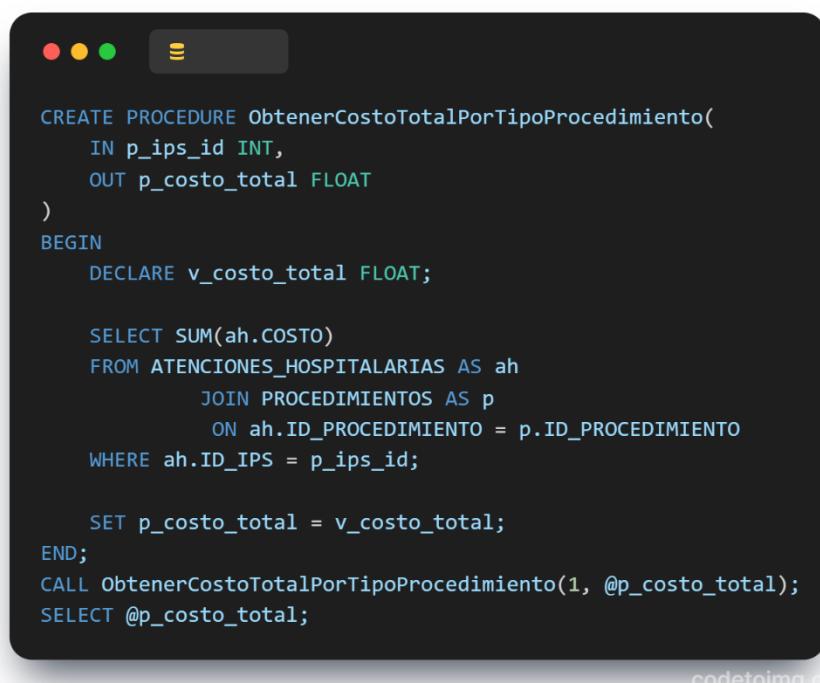
Calcula el costo total de los procedimientos realizados en una IPS específica.

Proporciona un resumen financiero de los costos asociados a los procedimientos médicos.

(Ver figura 31)

Figura 31.

Contenido del procedimiento para obtener el costo total por tipo de procedimiento.



```
CREATE PROCEDURE ObtenerCostoTotalPorTipoProcedimiento(
    IN p_ips_id INT,
    OUT p_costo_total FLOAT
)
BEGIN
    DECLARE v_costo_total FLOAT;

    SELECT SUM(ah.COSTO)
    FROM ATENCIONES_HOSPITALARIAS AS ah
        JOIN PROCEDIMIENTOS AS p
            ON ah.ID_PROCEDIMIENTO = p.ID_PROCEDIMIENTO
    WHERE ah.ID_IPS = p_ips_id;

    SET p_costo_total = v_costo_total;
END;
CALL ObtenerCostoTotalPorTipoProcedimiento(1, @p_costo_total);
SELECT @p_costo_total;
```

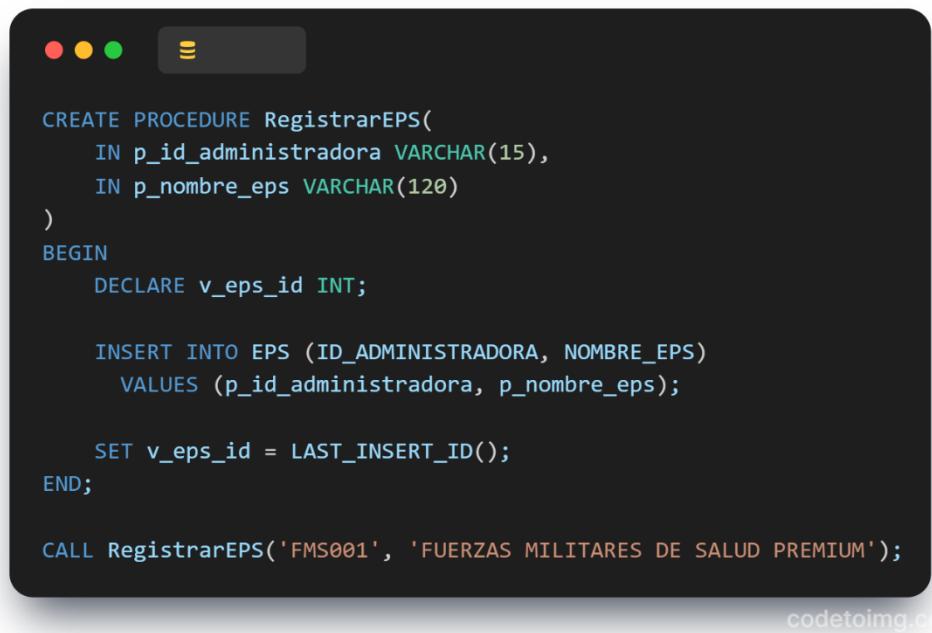
codetomsg.com

Procedimiento: registrar eps

Inserta un nuevo registro en la tabla de EPS. Automatiza el registro de nuevas EPS en el sistema. (Ver figura 32)

Figura 32.

Contenido del procedimiento para registrar una nueva eps.



```
CREATE PROCEDURE RegistrarEPS(
    IN p_id_administradora VARCHAR(15),
    IN p_nombre_eps VARCHAR(120)
)
BEGIN
    DECLARE v_eps_id INT;

    INSERT INTO EPS (ID_ADMINISTRADORA, NOMBRE_EPS)
    VALUES (p_id_administradora, p_nombre_eps);

    SET v_eps_id = LAST_INSERT_ID();
END;

CALL RegistrarEPS('FMS001', 'FUERZAS MILITARES DE SALUD PREMIUM');
```

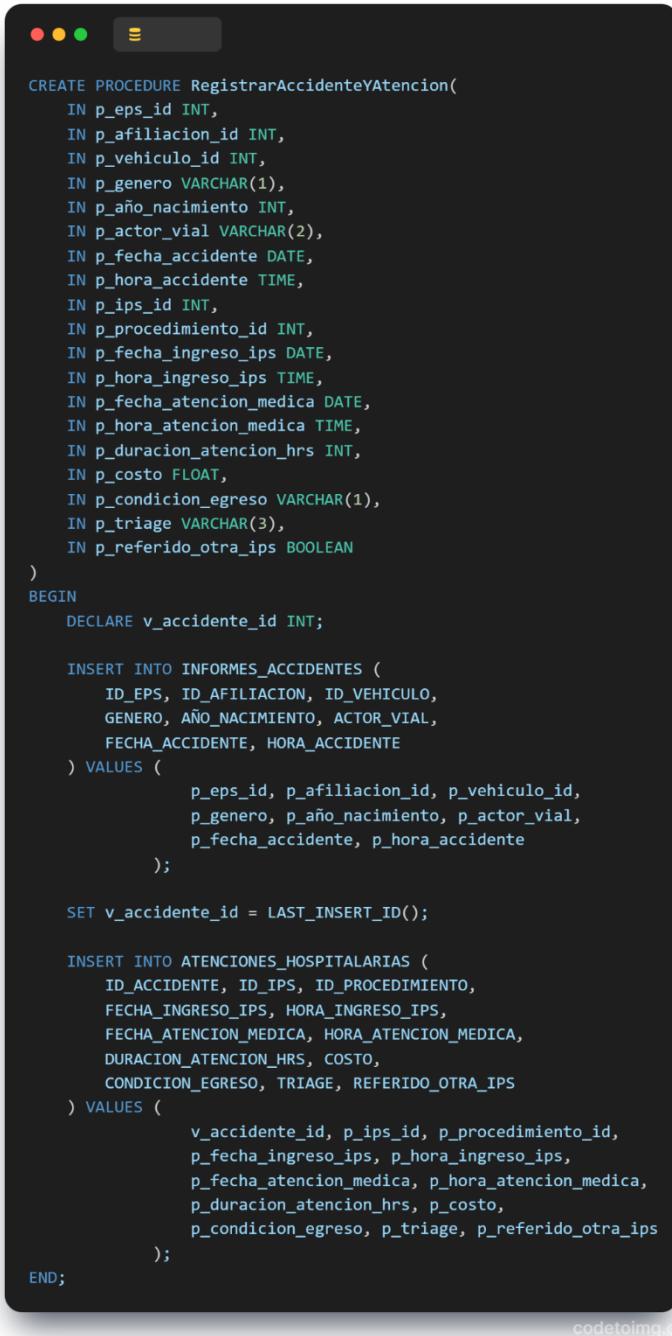
codetoimg.co

Registrar accidente y atención

Inserta registros en INFORMES_ACCIDENTES y ATENCIONES_HOSPITALARIAS para un accidente y su atención médica. Facilita el registro de nuevos accidentes y sus atenciones correspondientes en una sola transacción. (Ver figura 33)

Figura 33.

Contenido del procedimiento para registrar un accidente y una atención.



```
CREATE PROCEDURE RegistrarAccidenteYAtencion(
    IN p_eps_id INT,
    IN p_afiliacion_id INT,
    IN p_vehiculo_id INT,
    IN p_genero VARCHAR(1),
    IN p_año_nacimiento INT,
    IN p_actor_vial VARCHAR(2),
    IN p_fecha_accidente DATE,
    IN p_hora_accidente TIME,
    IN p_ips_id INT,
    IN p_procedimiento_id INT,
    IN p_fecha_ingreso_ips DATE,
    IN p_hora_ingreso_ips TIME,
    IN p_fecha_atencion_medica DATE,
    IN p_hora_atencion_medica TIME,
    IN p_duracion_atencion_hrs INT,
    IN p_costo FLOAT,
    IN p_condicion_egreso VARCHAR(1),
    IN p_triage VARCHAR(3),
    IN p_referido_otra_ips BOOLEAN
)
BEGIN
    DECLARE v_accidente_id INT;

    INSERT INTO INFORMES_ACCIDENTES (
        ID_EPS, ID_AFILIACION, ID_VEHICULO,
        GENERO, AÑO_NACIMIENTO, ACTOR_VIAL,
        FECHA_ACCIDENTE, HORA_ACCIDENTE
    ) VALUES (
        p_eps_id, p_afiliacion_id, p_vehiculo_id,
        p_genero, p_año_nacimiento, p_actor_vial,
        p_fecha_accidente, p_hora_accidente
    );

    SET v_accidente_id = LAST_INSERT_ID();

    INSERT INTO ATENCIONES_HOSPITALARIAS (
        ID_ACCIDENTE, ID_IPS, ID_PROCEDIMIENTO,
        FECHA_INGRESO_IPS, HORA_INGRESO_IPS,
        FECHA_ATENCION_MEDICA, HORA_ATENCION_MEDICA,
        DURACION_ATENCION_HRS, COSTO,
        CONDICION_EGRESO, TRIAGE, REFERIDO_OTRA_IPS
    ) VALUES (
        v_accidente_id, p_ips_id, p_procedimiento_id,
        p_fecha_ingreso_ips, p_hora_ingreso_ips,
        p_fecha_atencion_medica, p_hora_atencion_medica,
        p_duracion_atencion_hrs, p_costo,
        p_condicion_egreso, p_triage, p_referido_otra_ips
    );
END;
```

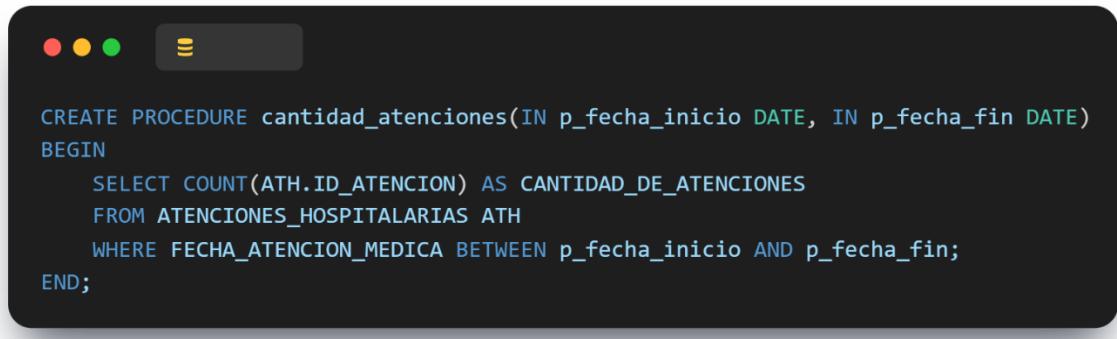
codetoimg.co

Cantidad de atenciones en un rango de fechas

Cuenta el número de atenciones hospitalarias en un rango de fechas. Proporciona estadísticas de atenciones en un periodo específico. (Ver figura 34)

Figura 34.

Contenido del procedimiento para mostrar la cantidad de atenciones realizadas en un rango de fechas.



```
CREATE PROCEDURE cantidad_atenciones(IN p_fecha_inicio DATE, IN p_fecha_fin DATE)
BEGIN
    SELECT COUNT(ATH.ID_ATENCION) AS CANTIDAD_DE_ATENCIONES
    FROM ATENCIONES_HOSPITALARIAS ATH
    WHERE FECHA_ATENCION_MEDICA BETWEEN p_fecha_inicio AND p_fecha_fin;
END;
```

codetomsg.com

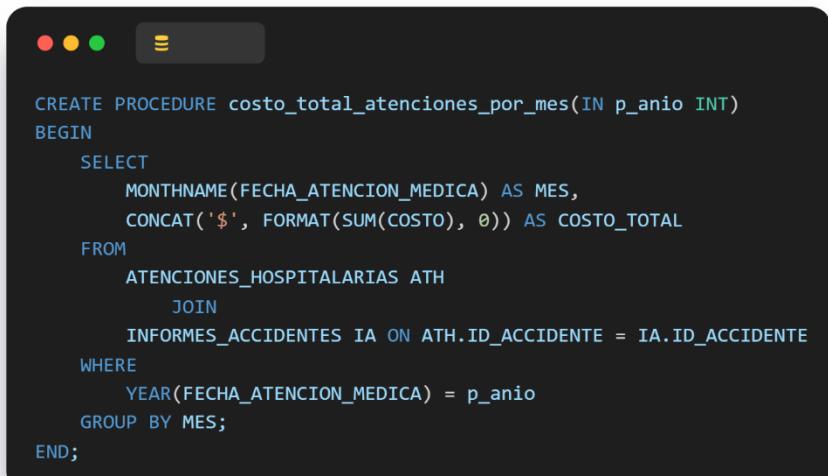
Costo total de las atenciones por mes en un año específico

Calcula el costo total de atenciones hospitalarias por mes en un año específico.

Proporciona un resumen financiero mensual de los costos asociados a las atenciones. (Ver figura 35)

Figura 35.

Contenido del procedimiento para mostrar el costo total de las atenciones por mes en un año.



```
CREATE PROCEDURE costo_total_atenciones_por_mes(IN p_anio INT)
BEGIN
    SELECT
        MONTHNAME(FECHA_ATENCION_MEDICA) AS MES,
        CONCAT('$', FORMAT(SUM(COSTO), 0)) AS COSTO_TOTAL
    FROM
        ATENCIONES_HOSPITALARIAS ATH
        JOIN
        INFORMES_ACCIDENTES IA ON ATH.ID_ACCIDENTE = IA.ID_ACCIDENTE
    WHERE
        YEAR(FECHA_ATENCION_MEDICA) = p_anio
    GROUP BY MES;
END;
```

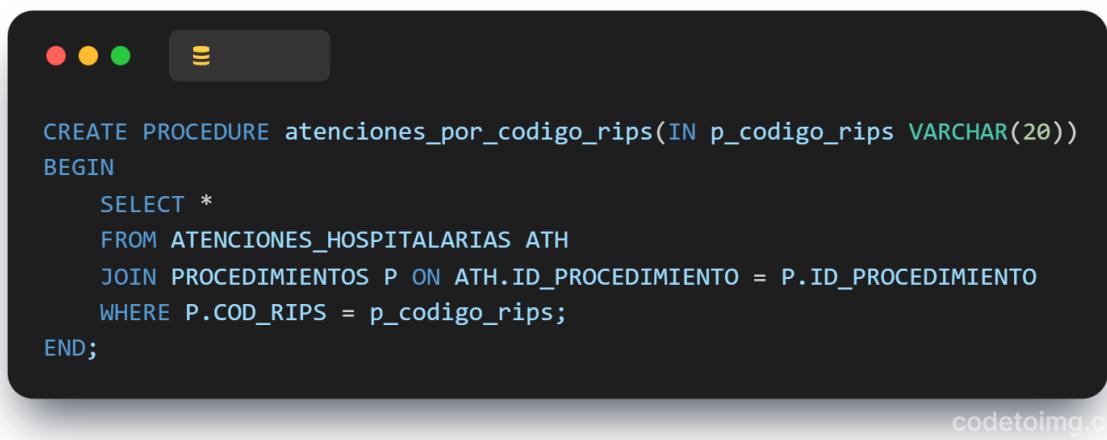
codetomsg.com

Atenciones por código rips

Obtiene las atenciones hospitalarias asociadas a un código RIPS específico. Permite consultar atenciones basadas en el código RIPS de los procedimientos. (Ver figura 36)

Figura 36.

Contenido del procedimiento que filtra las atenciones por código RIPS.



```
CREATE PROCEDURE atenciones_por_codigo_rips(IN p_codigo_rips VARCHAR(20))
BEGIN
    SELECT *
    FROM ATENCIONES_HOSPITALARIAS ATH
    JOIN PROCEDIMIENTOS P ON ATH.ID_PROCEDIMIENTO = P.ID_PROCEDIMIENTO
    WHERE P.COD_RIPS = p_codigo_rips;
END;
```

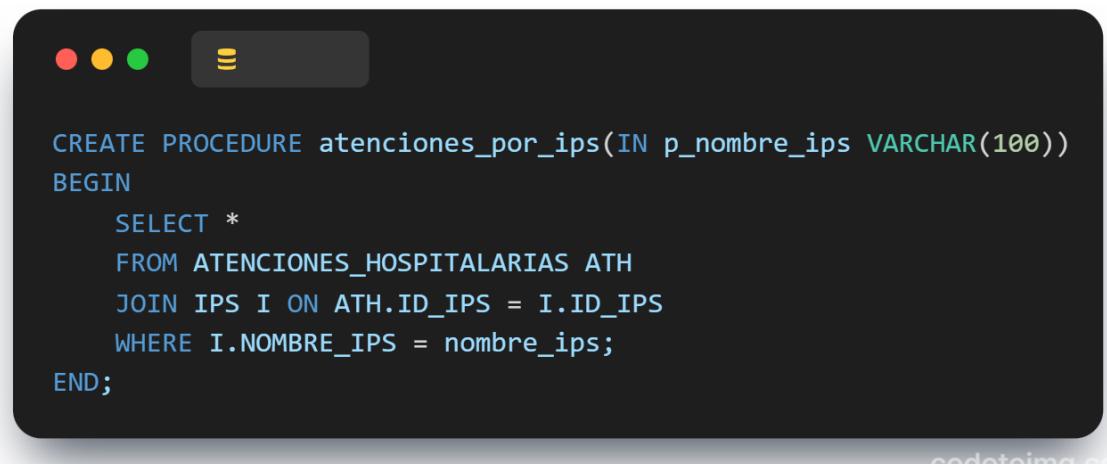
codetomsg.co

Atenciones dadas por una ips en particular

Obtiene las atenciones hospitalarias realizadas por una IPS específica. Permite consultar las atenciones basadas en el nombre de la IPS. (Ver figura 37)

Figura 37.

Contenido del procedimiento que muestra las atenciones dadas por una IPS.



```
CREATE PROCEDURE atenciones_por_ips(IN p_nombre_ips VARCHAR(100))
BEGIN
    SELECT *
    FROM ATENCIONES_HOSPITALARIAS ATH
    JOIN IPS I ON ATH.ID_IPS = I.ID_IPS
    WHERE I.NOMBRE_IPS = nombre_ips;
END;
```

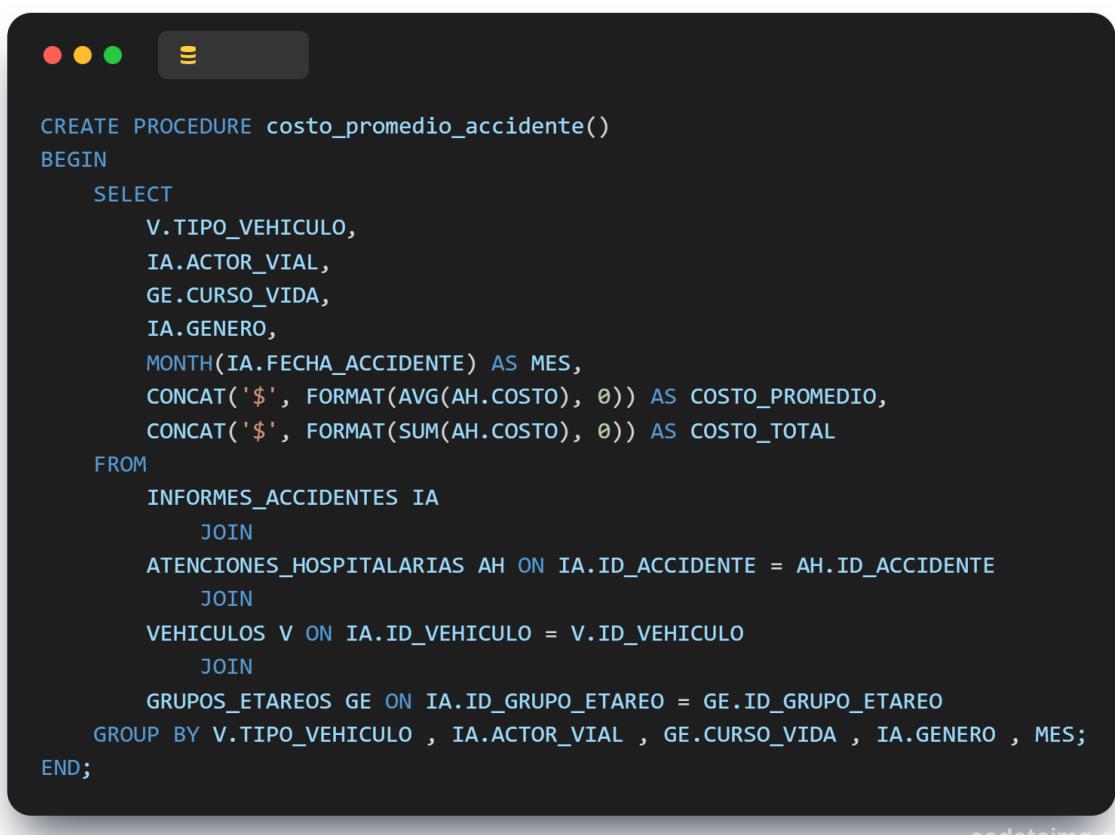
codetomsg.co

Costo promedio de accidentes de tránsito

Calcula el costo promedio y total de accidentes de tránsito basándose en diversos factores como tipo de vehículo, relación conductora, curso de vida, sexo, mes del año, etc. Proporciona análisis detallados del costo de accidentes para las aseguradoras. (Ver figura 38)

Figura 38.

Contenido del procedimiento para obtener el promedio de accidentes de tránsito.



```
CREATE PROCEDURE costo_promedio_accidente()
BEGIN
    SELECT
        V.TIPO_VEHICULO,
        IA.ACTOR_VIAL,
        GE.CURSO_VIDA,
        IA.GENERO,
        MONTH(IA.FECHA_ACCIDENTE) AS MES,
        CONCAT('$', FORMAT(AVG(AH.COSTO), 0)) AS COSTO_PROMEDIO,
        CONCAT('$', FORMAT(SUM(AH.COSTO), 0)) AS COSTO_TOTAL
    FROM
        INFORMES_ACCIDENTES IA
        JOIN
        ATENCIONES_HOSPITALARIAS AH ON IA.ID_ACCIDENTE = AH.ID_ACCIDENTE
        JOIN
        VEHICULOS V ON IA.ID_VEHICULO = V.ID_VEHICULO
        JOIN
        GRUPOS_ETAREOS GE ON IA.ID_GRUPO_ETAREO = GE.ID_GRUPO_ETAREO
    GROUP BY V.TIPO_VEHICULO , IA.ACTOR_VIAL , GE.CURSO_VIDA , IA.GENERO , MES;
END;
```

codetoimg.co

Costo promedio de accidentes por tipo de vehículo

Calcula el costo promedio y total de accidentes de tránsito basándose en el tipo de vehículo. Proporciona un análisis específico del costo de accidentes por tipo de vehículo para las aseguradoras. (Ver figura 39)

Figura 39.

Contenido del procedimiento para obtener el promedio de accidentes por tipo de vehículo.

```
CREATE PROCEDURE costo_promedio_accidente_por_vehiculo(in p_tipo_vehiculo VARCHAR(100))
BEGIN
    SELECT
        V.TIPO_VEHICULO,
        IA.ACTOR_VIAL,
        GE.CURSO_VIDA,
        IA.GENERO,
        MONTH(IA.FECHA_ACCIDENTE) AS MES,
        CONCAT('$', FORMAT(AVG(AH.COSTO), 0)) AS COSTO_PROMEDIO,
        CONCAT('$', FORMAT(SUM(AH.COSTO), 0)) AS COSTO_TOTAL
    FROM
        INFORMES_ACCIDENTES IA
        JOIN
        ATENCIONES_HOSPITALARIAS AH ON IA.ID_ACCIDENTE = AH.ID_ACCIDENTE
        JOIN
        VEHICULOS V ON IA.ID_VEHICULO = V.ID_VEHICULO
        JOIN
        GRUPOS_ETAREOS GE ON IA.ID_GRUPO_ETAREO = GE.ID_GRUPO_ETAREO
    WHERE V.TIPO_VEHICULO = p_tipo_vehiculo
    GROUP BY V.TIPO_VEHICULO , IA.ACTOR_VIAL , GE.CURSO_VIDA , IA.GENERO , MES;
END;
```

codetoimg.co

Referencias

Bases de datos ACID en comparación con las BASE: Diferencia entre bases de datos:
AWS. (2023). Amazon Web Services, Inc. <https://aws.amazon.com/es/compare/the-difference-between-acid-and-base-database/>

DataGrip: herramienta de GUI para MySQL. (s/f). JetBrains. Recuperado el 2 de junio de 2024, de <https://www.jetbrains.com/es-es/datagrip/features/mysql.html>

Datos Abiertos. (2018). *Costos de la atención hospitalaria en Bucaramanga por accidentes de tránsito enero 2018 a noviembre 2021.* Datos Abiertos.
https://www.datos.gov.co/Salud-y-Protección-Social/Costos-de-la-atención-hospitalaria-en-Bucaramanga-/g4vd-w4ip/about_data

Funcionalidades Principales de PowerDesigner. (s/f). Powerdesigner.biz.
Recuperado el 2 de junio de 2024, de
<https://www.powerdesigner.biz/ES/powerdesigner/powerdesigner-features.html>

¿Qué es MySQL? (2023). Oracle.com. <https://www.oracle.com/co/mysql/what-is-mysql/>

Profundizar en Amazon Aurora y sus innovaciones. (2024). Amazon Web Services, Inc. <https://aws.amazon.com/es relational-database/>

Apéndice

- Apéndice A: Materiales Adicionales

Este documento se acompaña de los siguientes materiales adicionales:

Diagramas:

Diagrama del Modelo Lógico

Diagrama del Modelo Conceptual

Código SQL:

Código SQL utilizado para la creación y manipulación de la base de datos.

Datos en Formato CSV:

Conjunto de datos utilizados en el estudio en formato CSV.

Estos materiales están disponibles en el anexo adjunto a este documento. Para acceder a ellos, por favor consulte los archivos correspondientes que se encuentran en el mismo paquete de entrega.