Proyecto de Bases de datos para siniestros víales en Bogotá D.C

Santiago Monsalve, Iván Pinzón

Dpto. de Matemáticas, Dpto. de Matemáticas,
Universidad Uexternado
Pregrado Ciencia de Datos
Curso de Bases de Datos
Bogotá, Colombia
miguel.monsalve1@est.uexternado.edu.co, ivan.pinzon3@est.uexternado.edu.co

Índice

1. Introducción (Max 250 Palabras) - (*Primera entrega*) **3**

2. Características del proyecto de investigación que hace uso de Bases de Datos (Max 500 Palabras) - (*Primera entrega*) **3**
2.1. Titulo del proyecto de investigación (Max 100 Palabras) - (*Primera entrega*) 3
2.2. Objetivo general (Max 100 Palabras) - (*Primera entrega*) 4
2.2.1. Objetivos especificos (Max 100 Palabras) - (*Primera entrega*) 4
2.3. Alcance (Max 200 Palabras) - (*Primera entrega*) 4
2.4. Pregunta de investigación (Max 100 Palabras) - (*Primera entrega*) . 5
2.5. Hipotesis (Max 100 Palabras) - (*Primera entrega*) 5

3. Reflexiones sobre el origen de datos e información (Max 400 Palabras) - (*Primera entrega*) **6**
3.1. ¿Cuál es el origen de los datos e información? (Max 100 Palabras) - (*Primera entrega*) 6
3.2. ¿Cuáles son las consideraciones legales o eticas del uso de la información? (Max 100 Palabras) - (*Primera entrega*) 6
3.3. ¿Cuáles son los retos de la información y los datos que utilizara en la base de datos en terminos de la calidad y la consolidación? (Max 100 Palabras) - (*Primera entrega*) 7
3.4. ¿Qué espera de la utilización de un sistema de Bases de Datos para su proyecto? (Max 100 Palabras) - (*Primera entrega*) 7

4. Diseño del Modelo de Datos del SMBD (Sistema Manejador de Bases de Datos)(*Primera entrega*) 8
4.1. Características del SMBD (Sistema Manejador de Bases de Datos) para el proyecto (*Primera entrega*) 8
4.2. Diagrama modelo de datos (*Primera entrega*) 8
4.3. Imágenes de la Base de Datos (*Primera entrega*) 8
4.4. Código SQL - Lenguaje de Definición de Datos (DDL) (*Primera entrega*) 9
4.5. Código SQL - Manipulación de datos (DML) (*Primera entrega*) . . 10
4.6. Código SQL + Resultados: Vistas (*Primera entrega*) 14
4.7. Código SQL + Resultados: Triggers (*Primera entrega*) 15
4.8. Código SQL + Resultados: Funciones (*Primera entrega*) 15
4.9. Código SQL + Resultados: procedimientos almacenados (*Primera entrega*) 16

5. Bases de Datos No-SQL (*Segunda entrega*) 17
5.1. Diagrama Bases de Datos No-SQL (*Segunda entrega*) 17
5.2. SMBD utilizado para la Base de Datos No-SQL (*Segunda entrega*) 17

6. Aplicación de ETL (Extract, Transform, Load) y Bodega de Datos (*Tercera entrega*) 20
6.1. Ejemplo de aplicación de ETL y Bodega de Datos (*Tercera entrega*) 20
6.2. Automatización de Datos (*Tercera entrega*) 21
6.3. Integración de Datos (*Tercera entrega*) 22

7. Proximos pasos (*Tercera entrega*) 23

8. Lecciones aprendidas (*Tercera entrega*) 24

9. Bibliografía 25

1. Introducción (Max 250 Palabras) - (*Primera entrega*)

La seguridad vial en Bogotá es motivo creciente de preocupación. Los accidentes de tránsito impactan la vida de los ciudadanos y la ciudad misma. Este proyecto emplea la Base de Datos Abiertos de Bogotá y SQL para descifrar patrones y factores detrás de estos incidentes. Se explorará la relación entre congestión, infraestructura vial y comportamientos de conductores, peatones y ciclistas. Los datos abiertos son cruciales para decisiones informadas y estrategias de seguridad. El estudio no solo busca comprender, sino también proponer recomendaciones concretas para abordar el problema y mejorar la seguridad vial. Análisis detallados y conclusiones se presentarán en las siguientes secciones, con el objetivo de impactar positivamente en la seguridad vial. El enfoque es construir un futuro donde los accidentes sean escasos y los ciudadanos puedan transitar con confianza.

2. Características del proyecto de investigación que hace uso de Bases de Datos (Max 500 Palabras) - (*Primera entrega*)

Este proyecto de investigación se basa en la obtención de datos de fuentes fundamentales: la página de Datos Abiertos Bogotá y la página de Movilidad de Bogotá. La autenticidad y actualización constante de estos datos garantizan la solidez de la investigación.

En el proceso de análisis, Oracle Developer desempeñará un papel esencial. Su capacidad avanzada para manipular bases de datos y realizar consultas complejas permitirá una comprensión profunda de los datos recopilados.

El enfoque principal del proyecto se centra en la extracción de datos relacionados con accidentes viales. Esto implica una exploración detallada de variables clave, como el tipo de accidente, la localidad específica en la que tuvo lugar, la clase de accidente y la presencia de heridos o fallecidos. Esta desglose minucioso permitirá identificar patrones y tendencias, arrojando luz sobre las circunstancias detrás de los siniestros.

Un aspecto central de la investigación es la identificación y análisis de los distintos tipos de accidentes. Colisiones, atropellos, volcamientos y otros incidentes serán examinados en profundidad. Esta segmentación permitirá una comprensión precisa de las situaciones de mayor riesgo en las vías de Bogotá.

Además, se investigará la relación entre la localidad específica y los accidentes. Al examinar en qué áreas de la ciudad ocurren con mayor frecuencia, se podrán identificar patrones geográficos y factores contextuales que contribuyen a la accidentalidad vial.

2.1. Titulo del proyecto de investigación (Max 100 Palabras) - (*Primera entrega*)

Análisis de Siniestros Viales en Bogotá: Extracción y Exploración de Datos desde Fuentes Abiertas Utilizando Oracle Developer

2.2. Objetivo general (Max 100 Palabras) - (*Primera entrega*)

Desarrollar una base de datos en Oracle que integre y analice datos de accidentes de tráfico en Bogotá desde fuentes abiertas, aplicando técnicas de ETL y análisis de datos para identificar patrones, tendencias y factores de riesgo. Los insights generados se utilizarán para informar políticas y programas que mejoren la seguridad vial en la ciudad. El proyecto empleará Oracle Developer para la ingesta y transformación de datos, y técnicas como machine learning para un análisis avanzado que permita comprender las causas raíz de los accidentes y formular recomendaciones concretas para su prevención y reducción.

2.2.1. Objetivos especificos (Max 100 Palabras) - (*Primera entrega*)

- Explorar la relación entre las variables para comprender cómo factores como el tipo de accidente y la localidad pueden estar relacionados con la gravedad y la presencia de heridos o fallecidos.
- Identificar horarios de mayor riesgo analizando los datos en función del tiempo, lo que permitirá detectar momentos específicos del día con una incidencia más alta de accidentes.
- Utilizar los resultados del análisis para generar recomendaciones concretas que puedan contribuir a mejorar la seguridad vial en Bogotá, proporcionando sugerencias específicas para reducir los riesgos identificados.
- Evaluar la aplicabilidad de las recomendaciones, considerando factores como la factibilidad de implementación y el impacto potencial en la seguridad vial de la ciudad.

2.3. Alcance (Max 200 Palabras) - (*Primera entrega*)

El alcance de este proyecto se enfoca en analizar la problemática de los accidentes de tráfico en Bogotá a través de la exploración de datos disponibles en línea y la aplicación de herramientas tecnológicas. Nuestro enfoque abarca la identificación de patrones y tendencias en variables clave, como el tipo de accidente, la ubicación, la gravedad y la presencia de heridos o fallecidos. Para llevar a cabo este análisis, empleamos Oracle Developer, una plataforma de manejo de datos.

Nos centraremos en descubrir relaciones espaciales y temporales, investigando si hay áreas específicas de la ciudad con una mayor incidencia de accidentes y si existen momentos del día más propensos a este tipo de incidentes. También se realizará un análisis predictivo para anticipar factores de riesgo.

Es importante destacar que, aunque proporcionaremos recomendaciones basadas en los resultados obtenidos, no nos adentraremos en la implementación precisa de políticas de seguridad vial. El propósito es proporcionar información valiosa para respaldar la toma de decisiones informadas en esta área. Cabe mencionar

que el análisis se limita a los datos obtenidos de fuentes públicas en línea y no involucrará la obtención de información adicional de otras fuentes externas.

2.4. Pregunta de investigación (Max 100 Palabras) - (*Primera entrega*)

¿Cuáles son los patrones y factores que influyen en los accidentes viales en Bogotá, utilizando la Base de Datos Abiertos de Bogotá y la herramienta Oracle Developer, y cómo se pueden utilizar estos hallazgos para proponer recomendaciones concretas que contribuyan a mejorar la seguridad vial en la ciudad?

2.5. Hipotesis (Max 100 Palabras) - (*Primera entrega*)

Se espera que al analizar los datos de accidentes de tráfico en Bogotá utilizando la Base de Datos Abiertos de Bogotá y herramientas como Oracle Developer, se identificarán patrones geográficos y temporales en la ocurrencia de siniestros viales. Además, se prevé que se encontrarán relaciones entre variables como el tipo de accidente, la localidad y la presencia de heridos o fallecidos. A partir de estos hallazgos, será posible formular recomendaciones concretas que contribuyan a mejorar la seguridad vial en la ciudad.

3. Reflexiones sobre el origen de datos e información
(Max 400 Palabras) - (*Primera entrega*)

El origen de los datos sobre siniestros viales en Bogotá es crucial para garantizar análisis precisos y éticos. Es fundamental preguntarse sobre la fiabilidad y precisión de estos datos. Si provienen de fuentes oficiales, podrían ser más confiables, pero no están exentos de errores o desajustes. La actualidad es otro aspecto esencial: los datos antiguos pueden no reflejar el estado actual del tránsito o las normativas vigentes en Bogotá. Además, la granularidad y complejidad de los datos nos llevan a cuestionar qué información podría faltar. Por ejemplo, las condiciones climáticas y el estado de las carreteras son variables relevantes en siniestros viales que pueden no estar presentes en nuestra base de datos. Por último, pero no menos importante, es la ética y privacidad. Aunque este conjunto no parece contener datos personales directos, siempre es primordial ser consciente de la privacidad. En resumen, esta base de datos es una herramienta valiosa para entender la situación vial en Bogotá, pero es vital acercarse con un enfoque crítico, considerando su origen y las posibles limitaciones.

3.1. ¿Cuál es el origen de los datos e información? (Max 100 Palabras) - (*Primera entrega*)

El origen de nuestros datos e información sobre siniestros viales proviene directamente de las bases de datos del portal oficial de la Secretaría Distrital de Movilidad (datos.movilidadbogota.gov.co). Esta fuente, siendo un ente gubernamental, proporciona una perspectiva oficial y confiable sobre la situación del tránsito en Bogotá, reflejando registros basados en reportes y estadísticas gestionadas por la administración de la capital.

3.2. ¿Cuáles son las consideraciones legales o eticas del uso de la información? (Max 100 Palabras) - (*Primera entrega*)

La utilización de datos sobre siniestros viales en Bogotá, aunque provienen de una fuente oficial, conlleva consideraciones éticas y legales. Esencialmente, debe asegurarse de que no se infrinjan derechos de privacidad, evitando la divulgación de información personal. Además, el uso incorrecto o la manipulación de datos puede conducir a interpretaciones erróneas, potencialmente afectando políticas públicas o decisiones basadas en dicha información. Es fundamental reconocer y citar la fuente, garantizando que el propósito del uso no difiere de la intención original del conjunto de datos. Finalmente, cualquier interpretación o conclusión debe ser comunicada con responsabilidad y transparencia.

3.3. ¿Cuáles son los retos de la información y los datos que utilizara en la base de datos en terminos de la calidad y la consolidación? (Max 100 Palabras) - (*Primera entrega*)

El manejo de datos sobre siniestros viales implica retos en cuanto a calidad y consolidación. Primero, la precisión y actualidad de la información son cruciales; registros desactualizados o inexactos pueden llevar a conclusiones erróneas. Segundo, la consolidación puede enfrentar incongruencias debido a distintos formatos o criterios de registro a lo largo del tiempo. Adicionalmente, la falta de datos específicos o la presencia de valores nulos pueden afectar el análisis. Es vital, por ende, someter a la base de datos a rigurosos procesos de limpieza y validación, buscando asegurar su confiabilidad y utilidad en investigaciones y decisiones.

3.4. ¿Qué espera de la utilización de un sistema de Bases de Datos para su proyecto? (Max 100 Palabras) - (*Primera entrega*)

De la implementación de un sistema de Bases de Datos para el proyecto, se espera una gestión eficiente, segura y estructurada de la información sobre siniestros viales. Dicha estructura permitirá realizar análisis complejos, facilitar la toma de decisiones y extraer insights valiosos. Además, se busca garantizar la integridad y consistencia de los datos, minimizando errores. El sistema también proporcionará flexibilidad para adaptarse a futuras necesidades, como la incorporación de nuevos datos o la integración con otras plataformas. En pocas palabras se espera que la base de datos sea la columna vertebral que soporte y potencie la investigación.

4. Diseño del Modelo de Datos del SMBD (Sistema Manejador de Bases de Datos)(*Primera entrega*)

4.1. Características del SMBD (Sistema Manejador de Bases de Datos) para el proyecto (*Primera entrega*)

El Sistema de Gestión de Bases de Datos (SMBD), en este caso, Oracle Developer, desempeña un papel crucial al permitir la extracción, transformación y análisis de datos de accidentes viales en Bogotá. Sus características clave incluyen la capacidad de gestionar grandes volúmenes de datos, ejecutar consultas SQL para explorar patrones y relaciones en los datos, y ofrecer herramientas para garantizar la integridad y seguridad de la información. Además, facilita la generación de informes y visualizaciones que respaldan el análisis de tendencias y la formulación de recomendaciones para mejorar la seguridad vial en la ciudad.

4.2. Diagrama modelo de datos (*Primera entrega*)

A continuación se muestra el diagrama del modelo de datos:

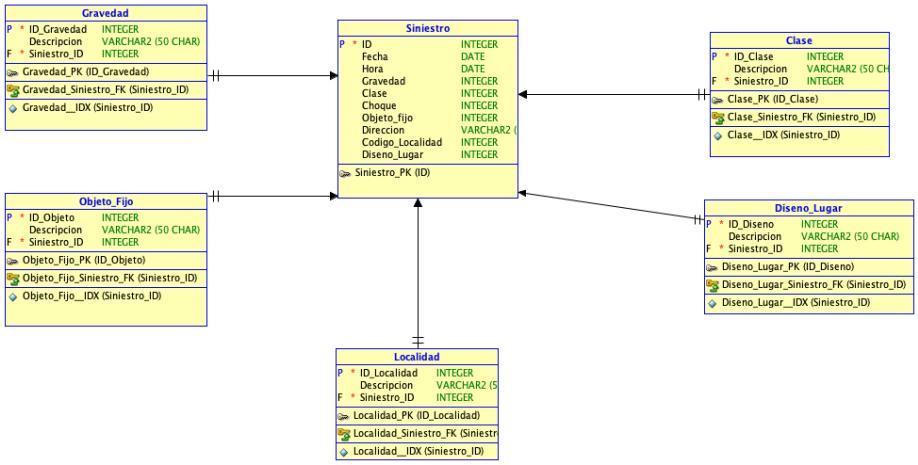
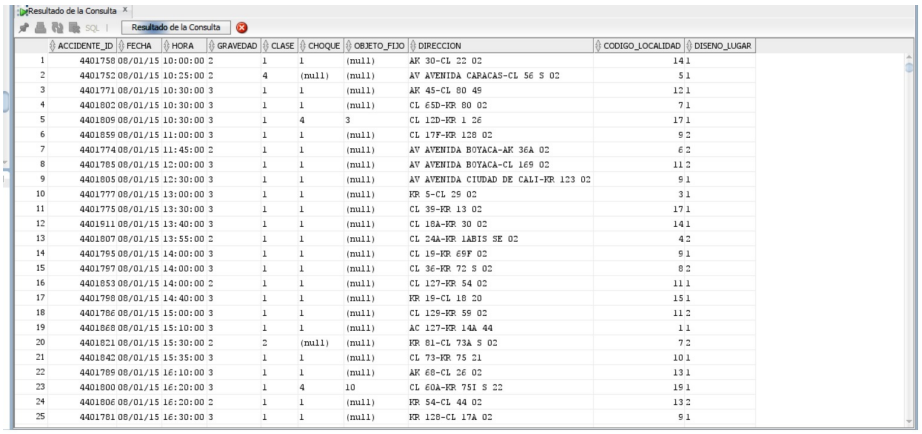


Figura 1: Modelo entidad-relación

4.3. Imágenes de la Base de Datos (*Primera entrega*)

A continuación, la presentación de cómo es la base de datos original, en la cual se basará este proyecto y su profundización.



Resultado de la Consulta

	ACCIDENTE_ID	FECHA	HORA	GRAVEDAD	CLASE	CHOQUE	OBJETO_FIJO	DIRECCION	CODIGO_LOCALIDAD	DISENO_LUGAR
1	4401758	08/01/15	10:00:00	2	1	1	(null)	AK 30-CL 22 02	14	1
2	4401752	08/01/15	10:25:00	2	4	(null)	(null)	AV AVENIDA CARACAS-CL 56 S 02	5	1
3	4401771	08/01/15	10:30:00	3	1	1	(null)	AK 45-CL 80 48	12	1
4	4401802	08/01/15	10:30:00	3	1	1	(null)	CL 65D-KR 80 02	7	1
5	4401809	08/01/15	10:30:00	3	1	4	3	CL 12D-KR 1 26	17	1
6	4401859	08/01/15	11:00:00	3	1	1	(null)	CL 17F-KR 128 02	9	2
7	4401774	08/01/15	11:45:00	2	1	1	(null)	AV AVENIDA BOYACA-AK 36A 02	6	2
8	4401785	08/01/15	12:00:00	3	1	1	(null)	AV AVENIDA BOYACA-CL 169 02	11	2
9	4401805	08/01/15	12:30:00	3	1	1	(null)	AV AVENIDA CIUDAD DE CALI-KR 123 02	9	1
10	4401777	08/01/15	13:00:00	3	1	1	(null)	KR 5-CL 29 02	3	1
11	4401775	08/01/15	13:30:00	3	1	1	(null)	CL 39-KR 13 02	17	1
12	4401911	08/01/15	13:40:00	3	1	1	(null)	CL 18A-KR 30 02	14	1
13	4401807	08/01/15	13:55:00	2	1	1	(null)	CL 24A-KR 1ABIS SE 02	4	2
14	4401795	08/01/15	14:00:00	3	1	1	(null)	CL 19-KR 69F 02	9	1
15	4401797	08/01/15	14:00:00	3	1	1	(null)	CL 36-KR 72 S 02	8	2
16	4401853	08/01/15	14:00:00	2	1	1	(null)	CL 127-KR 54 02	11	1
17	4401798	08/01/15	14:40:00	3	1	1	(null)	KR 19-CL 18 20	15	1
18	4401786	08/01/15	15:00:00	3	1	1	(null)	CL 129-KR 59 02	11	2
19	4401868	08/01/15	15:10:00	3	1	1	(null)	AC 127-KR 14A 44	11	
20	4401821	08/01/15	15:30:00	2	2	(null)	(null)	KR 81-CL 73A S 02	7	2
21	4401842	08/01/15	15:35:00	3	1	1	(null)	CL 73-KR 75 21	10	1
22	4401789	08/01/15	16:10:00	3	1	1	(null)	AK 68-CL 26 02	13	1
23	4401800	08/01/15	16:20:00	3	1	4	10	CL 60A-KR 75I S 22	19	1
24	4401806	08/01/15	16:20:00	2	1	1	(null)	KR 54-CL 44 02	13	2
25	4401781	08/01/15	16:30:00	3	1	1	(null)	KR 128-CL 17A 02	9	1

Figura 2: Base de datos Siniestros viales Bogotá

4.4. Código SQL - Lenguaje de Definición de Datos (DDL) (*Primera entrega*)

El código utilizado para la creación de las distintas tablas creadas con base en el modelo entidad-relación es el siguiente:

```
CREATE TABLE clase (
    id_clase     INTEGER NOT NULL PRIMARY KEY,
    descripcion  VARCHAR2(50 CHAR),
    siniestro_id INTEGER NOT NULL,
    FOREIGN KEY (siniestro_id) REFERENCES siniestro (id)
);

CREATE TABLE diseno_lugar (
    id_diseno    INTEGER NOT NULL PRIMARY KEY,
    descripcion  VARCHAR2(50 CHAR),
    siniestro_id INTEGER NOT NULL,
    FOREIGN KEY (siniestro_id) REFERENCES siniestro (id)
);

CREATE TABLE gravedad (
    id_gravedad  INTEGER NOT NULL PRIMARY KEY,
    descripcion  VARCHAR2(50 CHAR),
    siniestro_id INTEGER NOT NULL,
    FOREIGN KEY (siniestro_id) REFERENCES siniestro (id)
);

CREATE TABLE localidad (
    id_localidad INTEGER NOT NULL PRIMARY KEY,
    descripcion  VARCHAR2(50 CHAR),
    siniestro_id INTEGER NOT NULL,
    FOREIGN KEY (siniestro_id) REFERENCES siniestro (id)
```

```
);

CREATE TABLE objeto_fijo (
    id_objeto    INTEGER NOT NULL PRIMARY KEY,
    descripcion  VARCHAR2(50 CHAR),
    siniestro_id INTEGER NOT NULL,
    FOREIGN KEY (siniestro_id) REFERENCES siniestro (id)
);

CREATE TABLE siniestro (
    id               INTEGER NOT NULL PRIMARY KEY,
    fecha            DATE,
    hora             DATE,
    gravedad         INTEGER,
    clase            INTEGER,
    choque           INTEGER,
    objeto_fijo      INTEGER,
    direccion        VARCHAR2(100 CHAR),
    codigo_localidad INTEGER,
    diseno_lugar     INTEGER
);
```

4.5. Código SQL - Manipulación de datos (DML) (*Primera entrega*)

Como objetivo principal se tenía ingresar datos en las tablas. En primer lugar se importaron datos aleatorios mediante código de Python, el cual proporciona 100 datos de manera pseudoaleatoria para cada una de las tablas proporcionadas.

A continuación, se muestra el código Python que genera registros ficticios y los guarda en un archivo SQL:

```
import random
from faker import Faker
from google.colab import files

# Crear un objeto Faker
faker = Faker()

# Generar registros ficticios y escribir en un archivo SQL
with open('registros.sql', 'w') as archivo:
    # Generar registros para la tabla Siniestros
    for i in range(101):
        id_suceso = i
        fecha = faker.date()
        hora = faker.time()
        gravedad = random.randint(1, 3)
        clase = random.randint(1, 3)
        objeto_fijo = random.randint(1, 3)
        direccion = faker.address()
```

```
19          codigo_localidad = random.randint(100000, 199999)
20          diseno_lugar = random.randint(1, 6)
21          archivo.write(f"INSERT INTO siniestro (id, fecha,
22          hora, gravedad, clase, objeto_fijo,
23          direccion, codigo_localidad, diseno_lugar)
24          VALUES ({id_suceso}, '{fecha}', '{hora}', {gravedad},
25          {clase}, {objeto_fijo}, '{direccion}',
26          {codigo_localidad},{diseno_lugar});")
27
28      # Generar registros para la tabla Gravedades
29      for i in range(101):
30          id_gravedad = i
31          descripcion = faker.word()
32          archivo.write(f"INSERT INTO gravedad (id_gravedad,
33          descripcion)
34          VALUES ({id_gravedad}, '{descripcion}');")
35
36      # Generar registros para la tabla Clases
37      for i in range(101):
38          id_clase = i
39          descripcion = faker.word()
40          archivo.write(f"INSERT INTO clase (id_clase,
41          descripcion)
42          VALUES ({id_clase}, '{descripcion}');")
43
44      # Generar registros para la tabla Objetos Fijos
45      for i in range(101):
46          id_objeto = i
47          descripcion = faker.word()
48          archivo.write(f"INSERT INTO objeto_fijo (id_objeto,
49          descripcion)
50          VALUES ({id_objeto}, '{descripcion}');")
51
52      # Generar registros para la tabla Localidades
53      for i in range(101):
54          id_localidad = i
55          descripcion = faker.city()
56          archivo.write(f"INSERT INTO localidad (id_localidad,
57          descripcion)
58          VALUES ({id_localidad}, '{descripcion}');")
59
60      # Generar registros para la tabla Dise os de Lugares
61      for i in range(101):
62          id_diseno = i
63          descripcion = faker.word()
64          archivo.write(f"INSERT INTO diseno_lugar (id_diseno,
65          descripcion)
66          VALUES ({id_diseno}, '{descripcion}');")
67
68  print("Registros generados y guardados en 'registros.sql'")
```

```
69
70 # Descargar el archivo SQL generado
71 files.download('registros.sql')
72 print("Archivo 'registros.sql' descargado")
```

También se hicieron distintas consultas a las distintas tablas establecidas, para profundizar poco a poco la información que estaba ingresando dentro de las mismas, esto se hizo mediante SQL developer, directamente.

A continuación el código utilizado:

Tabla çlase":

1. Seleccionar todas las clases:

```
1 SELECT * FROM clase;
```

2. Seleccionar la clase con id_clase igual a 1:

```
1 SELECT * FROM clase WHERE id_clase = 1;
```

3. Seleccionar las clases con descripción que contenga .ªccidente":

```
1 SELECT * FROM clase WHERE descripcion LIKE '%accidente%';
```

4. Contar la cantidad de clases:

```
1 SELECT COUNT(*) FROM clase;
```

5. Seleccionar las clases ordenadas por id_clase de forma descendente:

```
1 SELECT * FROM clase ORDER BY id_clase DESC;
```

Tabla "diseno_lugar":

1. Seleccionar todos los diseños de lugar:

```
1 SELECT * FROM diseno_lugar;
```

2. Seleccionar el diseño de lugar con id_diseno igual a 1:

```
1 SELECT * FROM diseno_lugar WHERE id_diseno = 1;
```

3. Seleccionar los diseños de lugar con descripción que contenga çruce":

```
1 SELECT * FROM diseno_lugar WHERE descripcion LIKE '%cruce%';
```

4. Contar la cantidad de diseños de lugar:

```
1 SELECT COUNT(*) FROM diseno_lugar;
```

5. Seleccionar los diseños de lugar ordenados por id_diseno de forma descendente:

```
SELECT * FROM diseno_lugar ORDER BY id_diseno DESC;
```

Tabla "gravedad":

1. Seleccionar todas las gravedades:

```
SELECT * FROM gravedad;
```

2. Seleccionar la gravedad con id_gravedad igual a 1:

```
SELECT * FROM gravedad WHERE id_gravedad = 1;
```

3. Seleccionar las gravedades con descripción que contenga "grave":

```
SELECT * FROM gravedad WHERE descripcion LIKE '%grave%';
```

4. Contar la cantidad de gravedades:

```
SELECT COUNT(*) FROM gravedad;
```

5. Seleccionar las gravedades ordenadas por id_gravedad de forma descendente:

```
SELECT * FROM gravedad ORDER BY id_gravedad DESC;
```

Tabla "localidad":

1. Seleccionar todas las localidades:

```
SELECT * FROM localidad;
```

2. Seleccionar la localidad con id_localidad igual a 1:

```
SELECT * FROM localidad WHERE id_localidad = 1;
```

3. Seleccionar las localidades con descripción que contenga çentro":

```
SELECT * FROM localidad WHERE descripcion LIKE '%centro%';
```

4. Contar la cantidad de localidades:

```
SELECT COUNT(*) FROM localidad;
```

5. Seleccionar las localidades ordenadas por id_localidad de forma descendente:

```
SELECT * FROM localidad ORDER BY id_localidad DESC;
```

Tabla .ºbjeto_fijo":

1. Seleccionar todos los objetos fijos:

```
SELECT * FROM objeto_fijo;
```

2. Seleccionar el objeto fijo con id_objeto igual a 1:

```
SELECT * FROM objeto_fijo WHERE id_objeto = 1;
```

3. Seleccionar los objetos fijos con descripción que contenga "semáforo":

```
SELECT * FROM objeto_fijo WHERE descripcion LIKE '%sem foro%';
```

4. Contar la cantidad de objetos fijos:

```
SELECT COUNT(*) FROM objeto_fijo;
```

5. Seleccionar los objetos fijos ordenados por id_objeto de forma descendente:

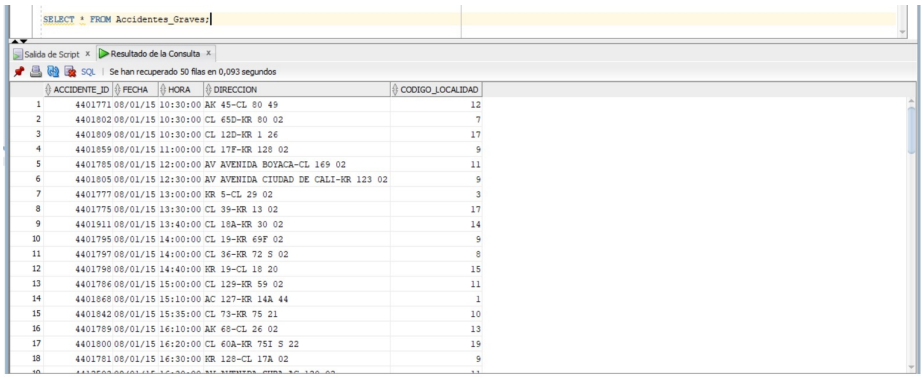
```
SELECT * FROM objeto_fijo ORDER BY id_objeto DESC;
```

4.6. Código SQL + Resultados: Vistas (*Primera entrega*)

Esta vista ha sido creada específicamente para mostrar solo aquellos accidentes que tienen una gravedad clasificada como 'grave'. Su propósito principal es facilitar consultas y análisis enfocados exclusivamente en los accidentes de mayor gravedad, permitiendo a los usuarios y aplicaciones acceder a esta información filtrada sin necesidad de realizar filtrados adicionales en la tabla principal de accidentes.

```
CREATE VIEW Accidentes_Graves AS
SELECT Accidente_ID, Fecha, Hora, Direccion, Codigo_localidad
FROM Siniestro
WHERE Gravedad = 3;
```

Como resultado de la vista, tenemos la siguiente tabla:



```
SELECT * FROM Accidentes_Graves;
```

Salida de Script | Resultado de la Consulta

SQL | Se han recuperado 50 filas en 0,093 segundos

	ACCIDENTE_ID	FECHA	HORA	DIRECCION	CODIGO_LOCALIDAD
1	4401771	08/01/15	10:30:00	AK 45-CL 80 49	12
2	4401802	08/01/15	10:30:00	CL 65D-KR 80 02	7
3	4401809	08/01/15	10:30:00	CL 12D-KR 1 26	17
4	4401859	08/01/15	11:00:00	CL 17F-KR 128 02	9
5	4401785	08/01/15	12:00:00	AV AVENIDA BOYACA-CL 169 02	11
6	4401805	08/01/15	12:30:00	AV AVENIDA CIUDAD DE CALI-KR 123 02	9
7	4401777	08/01/15	13:00:00	KR 5-CL 29 02	3
8	4401775	08/01/15	13:30:00	CL 39-KR 13 02	17
9	4401911	08/01/15	13:40:00	CL 18A-KR 30 02	14
10	4401795	08/01/15	14:00:00	CL 19-KR 69F 02	9
11	4401797	08/01/15	14:00:00	CL 36-KR 72 S 02	8
12	4401798	08/01/15	14:40:00	KR 19-CL 18 20	15
13	4401786	08/01/15	15:00:00	CL 129-KR 59 02	11
14	4401868	08/01/15	15:10:00	AC 127-KR 14A 44	1
15	4401842	08/01/15	15:35:00	CL 73-KR 75 21	10
16	4401789	08/01/15	16:10:00	AK 68-CL 26 02	13
17	4401800	08/01/15	16:20:00	CL 60A-KR 75I S 22	19
18	4401781	08/01/15	16:30:00	KR 128-CL 17A 02	9

Figura 3: Vista Accidentes Graves

4.7. Código SQL + Resultados: Triggers (*Primera entrega*)

Este trigger ha sido implementado para auditar automáticamente cada nuevo registro de accidente que se añade a la base de datos. Cada vez que se inserta un nuevo accidente en la tabla principal de accidentes (Siniestro), el trigger se activa y crea un registro en la tabla Tabla_Auditoria. Esta acción permite llevar un registro histórico de todas las adiciones a la base de datos y ayuda en el monitoreo y análisis de las actividades de inserción.

```
CREATE TRIGGER tr_auditoria_insert_accidente
AFTER INSERT ON Siniestro
FOR EACH ROW
BEGIN
  INSERT INTO Tabla_Auditoria (Accidente_ID, Fecha, Accion)
  VALUES
  (:NEW.Accidente_ID, SYSDATE, 'INSERT');
END;
```

4.8. Código SQL + Resultados: Funciones (*Primera entrega*)

El objetivo de esta función es determinar rápidamente la cantidad de accidentes que han ocurrido en una localidad específica. Al proporcionar un código de localidad como parámetro, la función devolverá el número total de accidentes registrados para esa localidad.

```
CREATE FUNCTION contar_accidentes(p_localidad NUMBER)
RETURN NUMBER AS
  v_count NUMBER;
BEGIN
  SELECT COUNT(*) INTO v_count
  FROM Siniestro
  WHERE Codigo_localidad = p_localidad;

```

```
  RETURN v_count;
END;
```

Como podemos observar a continuación, ejecutando la función tenemos que para la localidad número 7 tenemos 9684 accidentes:

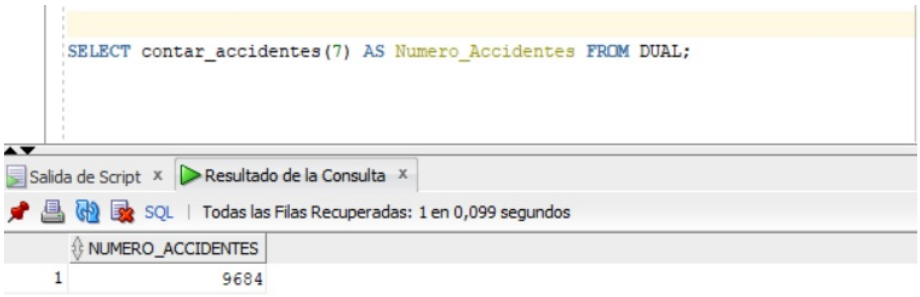


Figura 4: Número de Accidentes Para la Localidad 7

4.9. Código SQL + Resultados: procedimientos almacenados (*Primera entrega*)

Creamos este procedimeinto con el fin de actualizar la gravedad de un accidente en la base de datos. El procedimiento requiere el Accidente_ID y el nuevo valor de Gravedad para realizar la actualización. Este procedimiento simplifica la tarea de actualizar registros específicos sin tener que escribir la sentencia SQL de actualización cada vez.

```
CREATE OR REPLACE PROCEDURE cambiar_gravedad
(p_accidente_id NUMBER, p_nueva_gravedad NUMBER) AS
BEGIN
  UPDATE Siniestro
  SET Gravedad = p_nueva_gravedad
  WHERE Accidente_ID = p_accidente_id;

  COMMIT;
END;
```

5. Bases de Datos No-SQL (*Segunda entrega*)

5.1. Diagrama Bases de Datos No-SQL (*Segunda entrega*)

A continuación se muestra el diagrama del modelo de datos propuesto para trabajar en un ambiente NoSQL.

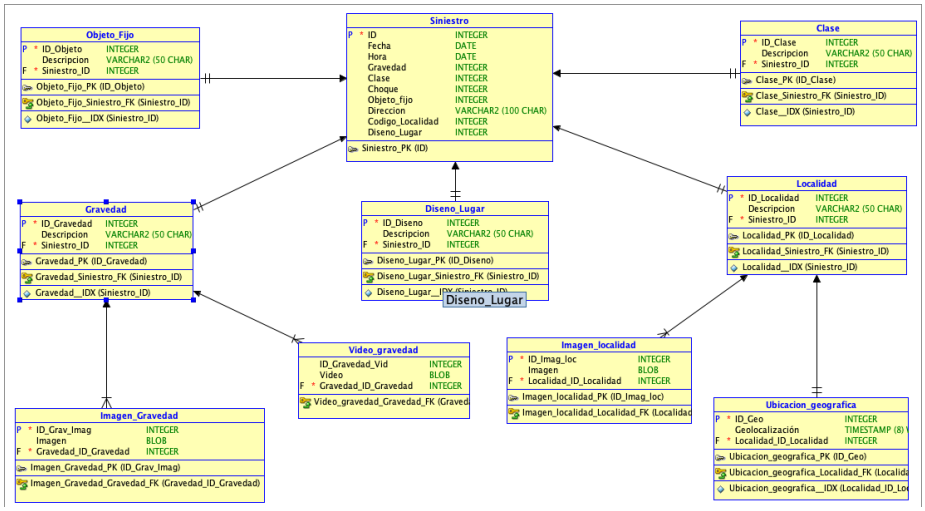


Figura 5: Modelo entidad-relación NoSql

5.2. SMBD utilizado para la Base de Datos No-SQL (*Segunda entrega*)

Para integrar datos en un sistema NoSQL como MongoDB, se puede emplear un script de Python que automatiza el proceso de lectura de un archivo CSV, transforma la información en un diccionario de datos y luego, utilizando la librería 'pymongo', establece una conexión con MongoDB para cargar los datos. Este método aprovecha la capacidad de Python para manipular datos y su interacción fluida con MongoDB, permitiendo que cada registro del archivo CSV se inserte como un documento independiente en la base de datos, facilitando así la gestión y análisis de la información en un formato no relacional.

El código utilizado fue el siguiente:

```
from pymongo import MongoClient
import pandas as pd

# Ruta al archivo en tu sistema local o Google Drive
ruta_al_archivo = '/content/drive/My Drive/Colab Notebooks/
Siniestros_viales.csv'

# Intenta abrir el archivo CSV
try:
```

```
    # Leer el archivo CSV con el delimitador correcto
    df = pd.read_csv(ruta_al_archivo, delimiter=';',
    error_bad_lines=False)
    # Convertir el DataFrame a una lista de diccionarios
    datos = df.to_dict('records')
except FileNotFoundError:
    print(f"El archivo no se encontr  en la ruta
    especificada:{ruta_al_archivo}")

# Obtener los datos de conexi n
client = MongoClient("mongodb+srv://Santiago:HasPzdz6QEsq2ohE
@cluster0.bqbpkf3.mongodb.net/?retryWrites=true&w=majority")

# Conectarse a la base de datos
db = client["Siniestros_viales"]

# Crear la colecci n
coleccion = db["Siniestros viales"]

# Importar los datos en la colecci n de MongoDB
coleccion.insert_many(datos)

print("Datos importados exitosamente a MongoDB")
```

Tras la ejecución exitosa del código y el establecimiento de una conexión efectiva entre Python y MongoDB, podemos observar que la base de datos ha sido importada en MongoDB, donde ahora se presenta como una colección no estructurada de tipo clave-valor, característica de las bases de datos NoSQL.

Figura 6: Vista 1, base de datos en MongoDB

Figura 7: Vista 2, base de datos en MongoDB

Para confirmar la carga completa de datos en MongoDB, revisamos el último registro insertado, y si este coincide con el del archivo CSV, entonces la importación ha sido exitosa.

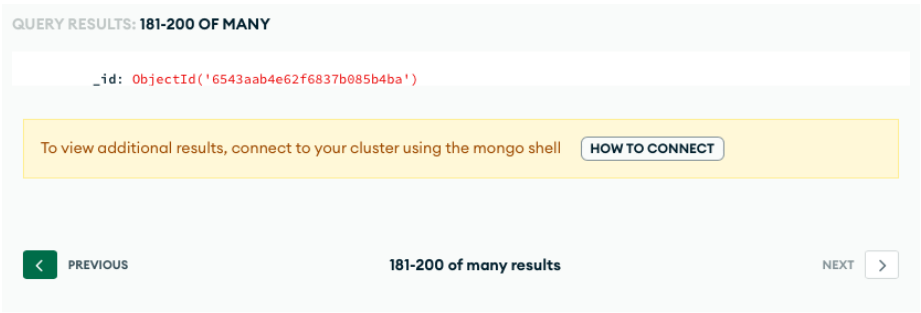


Figura 8: Vista 3, base de datos en MongoDB

6. Aplicación de ETL (Extract, Transform, Load) y Bodega de Datos (*Tercera entrega*)

6.1. Ejemplo de aplicación de ETL y Bodega de Datos (*Tercera entrega*)

El proceso de Extracción de datos es fundamental en el análisis de datos. Para nuestro estudio, recopilamos datos de la base de Datos Abiertos de Bogotá, una fuente confiable y actualizada. Esta información, accesible a través del enlace Datos de Siniestros Viales, fue esencial para garantizar la calidad y relevancia de nuestro análisis.

Durante el proceso de ETL, no necesitamos transformar los datos. Ya que estaban listos para usarse tal como los obtuvimos de Datos Abiertos de Bogotá. Así que pasamos directo a la etapa de Carga, lo que hizo todo más fácil y rápido.

Tras obtener los datos, el siguiente paso fue cargarlos en nuestra base de datos para poder trabajar con ellos. Este proceso de carga comenzó con la creación de una tabla específica en la base de datos, utilizando el código SQL:

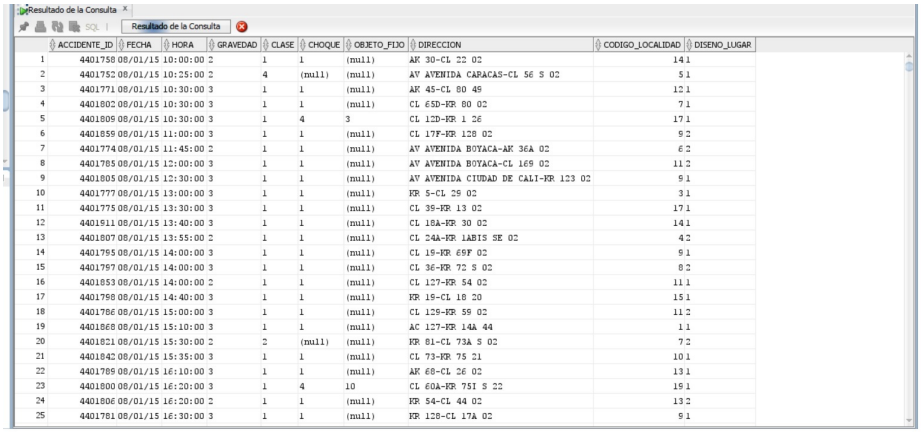
```
CREATE TABLE siniestros_viales (
    codigo_accidente NUMBER,
    fecha DATE,
    hora VARCHAR2(8),
    gravedad NUMBER,
    clase NUMBER,
    choque NUMBER,
    objeto_fijo VARCHAR2(100),
    direccion VARCHAR2(255),
    codigo_localidad NUMBER,
    diseno_lugar VARCHAR2(255)
);
```

Luego, para transferir los datos del archivo CSV a la tabla recién creada, usamos un comando de SQL*Loader. Este comando fue:

```
LOAD DATA
INFILE '/Users/macbook/Documentos/Siniestros_viales.csv'
INTO TABLE siniestros_viales
FIELDS TERMINATED BY ';'
OPTIONALLY ENCLOSED BY '"'
TRAILING NULLCOLS
(
    CODIGO_ACCIDENTE,
    FECHA DATE "DD/MM/YYYY",
    HORA,
    GRAVEDAD,
    CLASE,
```

```
    CHOQUE,
    OBJETO_FIJO,
    DIRECCION,
    CODIGO_LOCALIDAD,
    DISENO_LUGAR
)
```

Lo que nos asegura una carga exitosa al software SQL Developer, a continuación la representación de la carga de datos:



	ACCIDENTE_ID	FECHA	HORA	GRAVEDAD	CLASE	CHOQUE	OBJETO_FIJO	DIRECCION	CODIGO_LOCALIDAD	DISENO_LUGAR
1	4401758	08/01/15	10:00:00	2	1	1	(null)	AK 30-CL 22 02	14	1
2	4401752	08/01/15	10:25:00	2	4	(null)	(null)	AV AVENIDA CARACAS-CL 56 S 02	5	1
3	4401771	08/01/15	10:30:00	3	1	1	(null)	AK 45-CL 80 49	12	1
4	4401802	08/01/15	10:30:00	3	1	1	(null)	CL 65D-KR 80 02	7	1
5	4401809	08/01/15	10:30:00	3	1	4	3	CL 12D-KR 1 26	17	1
6	4401859	08/01/15	11:00:00	3	1	1	(null)	CL 17F-KR 128 02	9	2
7	4401774	08/01/15	11:45:00	2	1	1	(null)	AV AVENIDA BOYACA-AK 36A 02	6	2
8	4401785	08/01/15	12:00:00	3	1	1	(null)	AV AVENIDA BOYACA-CL 169 02	11	2
9	4401805	08/01/15	12:30:00	3	1	1	(null)	AV AVENIDA CIUDAD DE CALI-KR 123 02	9	1
10	4401777	08/01/15	13:00:00	3	1	1	(null)	KR 5-CL 25 02	3	1
11	4401775	08/01/15	13:30:00	3	1	1	(null)	CL 39-KR 13 02	17	1
12	4401911	08/01/15	13:40:00	3	1	1	(null)	CL 18A-KR 30 02	14	1
13	4401807	08/01/15	13:55:00	2	1	1	(null)	CL 24A-KR 1ABIS SE 02	4	2
14	4401795	08/01/15	14:00:00	3	1	1	(null)	CL 19-KR 69F 02	9	1
15	4401797	08/01/15	14:00:00	3	1	1	(null)	CL 36-KR 72 S 02	8	2
16	4401853	08/01/15	14:00:00	2	1	1	(null)	CL 127-KR 54 02	11	1
17	4401798	08/01/15	14:40:00	3	1	1	(null)	KR 19-CL 18 20	15	1
18	4401786	08/01/15	15:00:00	3	1	1	(null)	CL 129-KR 59 02	11	2
19	4401868	08/01/15	15:10:00	3	1	1	(null)	AC 127-KR 14A 44	1	1
20	4401821	08/01/15	15:30:00	2	2	(null)	(null)	KR 81-CL 73A S 02	7	2
21	4401842	08/01/15	15:35:00	3	1	1	(null)	CL 73-KR 75 21	10	1
22	4401789	08/01/15	16:10:00	3	1	1	(null)	AK 68-CL 26 02	13	1
23	4401800	08/01/15	16:20:00	3	1	4	10	CL 60A-KR 75I S 22	19	1
24	4401806	08/01/15	16:20:00	2	1	1	(null)	KR 54-CL 44 02	13	2
25	4401781	08/01/15	16:30:00	3	1	1	(null)	KR 128-CL 17A 02	9	1

Figura 9: Base de datos Siniestros viales Bogotá

6.2. Automatización de Datos (*Tercera entrega*)

En nuestro proyecto, una parte clave del manejo de datos involucra asegurarse de que la tabla `siniestros_viales` esté actualizada y libre de registros obsoletos. Para ello, hemos implementado un procedimiento automatizado en la base de datos que se encarga de eliminar los registros de más de un año de antigüedad. Este proceso automatizado no solo ahorra tiempo sino que también mantiene la eficiencia y relevancia de los datos para el análisis.

El siguiente es el código SQL para crear el procedimiento almacenado y programar su ejecución automática:

```
CREATE OR REPLACE PROCEDURE LimpiarRegistrosAntiguos AS
BEGIN
    DELETE FROM siniestros_viales
    WHERE fecha < ADD_MONTHS(SYSDATE, -12);
END;


BEGIN
    DBMS_SCHEDULER.CREATE_JOB (
```

```
        job_name        => 'Job_LimpiarRegistrosAntiguos',
        job_type        => 'PLSQL_BLOCK',
        job_action      => 'BEGIN LimpiarRegistrosAntiguos;
        END;',
        start_date      => SYSTIMESTAMP,
        repeat_interval => 'FREQ=MONTHLY',
        enabled         => TRUE);
END;
```

Este código realiza dos acciones principales. Primero, crea un procedimiento almacenado llamado `LimpiarRegistrosAntiguos`, que elimina todos los registros en la tabla `siniestros_viales` que tienen más de un año de antigüedad. Luego, programa este procedimiento para que se ejecute automáticamente cada mes, utilizando Oracle DBMS Scheduler. De esta manera, nos aseguramos de que la tabla contenga siempre datos relevantes y actualizados.

6.3. Integración de Datos (*Tercera entrega*)

Una tarea crucial en el análisis de datos es la integración de información de diferentes fuentes. En nuestro proyecto, esto involucra combinar detalles de siniestros viales con información adicional disponible en otra tabla. Realizamos esta integración creando una nueva tabla que une los datos relevantes de ambas fuentes. Este enfoque nos permite tener una vista más completa y detallada de cada incidente.

Aquí está el código SQL que utilizamos para esta integración de datos:

```
CREATE TABLE siniestros_integrados AS
SELECT
    v.codigo_accidente,
    v.fecha,
    v.hora,
    v.gravedad,
    a.informacion_detallada
FROM
    siniestros_viales v
JOIN
    informacion_adicional a
    ON v.codigo_accidente = a.codigo_accidente;
```

Este código crea una nueva tabla `siniestros_integrados`, seleccionando y uniendo datos de las tablas `siniestros_viales` y `informacion_adicional`. La integración se realiza mediante un *JOIN*, asegurando que sólo los registros que tienen un código de accidente común en ambas tablas sean incluidos.

7. Proximos pasos (*Tercera entrega*)

1. **Análisis Exploratorio Profundo**: Continuar explorando los datos para descubrir tendencias y patrones ocultos.

2. **Dashboards y Visualización**: Desarrollar dashboards interactivos que faciliten la comprensión de los patrones de siniestros viales.

3. **Modelado Predictivo**: Implementar modelos estadísticos o de machine learning para prever tendencias futuras en siniestros viales.

4. **Integración de Más Datos**: Considerar la integración de datos adicionales como condiciones climáticas o tráfico en tiempo real.

5. **Automatización y Optimización Continua**: Mejorar continuamente los procesos de ETL y automatización para aumentar la eficiencia.

8. Lecciones aprendidas (*Tercera entrega*)

1. **Importancia de la Calidad de Datos**: La buena calidad de los datos es clave para análisis precisos y confiables.

2. **Valor de la Automatización**: La automatización de procesos ahorra tiempo y permite enfocarse en el análisis.

3. **Flexibilidad en la Resolución de Problemas**: Los desafíos enfrentados enseñaron la importancia de ser flexible y creativo en la solución de problemas.

4. **Colaboración y Comunicación**: La colaboración efectiva y una buena comunicación son cruciales al trabajar con grandes conjuntos de datos.

9. Bibliografía

1. Oracle. (2023). *Oracle SQL Developer User's Guide*. Disponible en Documentación de Oracle SQL Developer.

2. Elmasri, R., & Navathe, S.B. (2020). *Fundamentals of Database Systems* (8ª ed.). Pearson.

3. Data School. (2023). *Blog sobre Análisis de Datos*. Visitar Data School Blog.

4. Perkovic, L. (2021). *SQL for Data Analysis: Weekender Crash Course for Beginners*. Independently published.

5. Oracle Learning Library. (2023). *Oracle Database Tutorials for Beginners*. Acceder a Tutoriales de Oracle Database.