

**Accidentabilidad Barranquilla 2021- 2025**

**Candy Manrique**

**Gina Ortega**

**Karen Sanjuan**

**Lina Piocuda**

**Martha Coronel**

**Marcela Luna**

**Sharon Varelo**

**Stefany Lozano**

**Tina Varela**

**Vanessa Gonzalez**

**Análisis de Datos – Conexión Mujeres TIC**

**Tutor: Santiago Giraldo**

**Fecha de entrega: Junio 13 del 2025**

**Barranquilla, Atlántico**

## Índice

- 1. Introducción**
- 2. Objetivos del análisis**
- 3. Hipótesis de trabajo**
- 4. Metodología**
  - 4.1. Fuentes de datos
  - 4.2. Herramientas utilizadas
  - 4.3. Técnicas de análisis aplicadas
- 5. Pasos preliminares realizados**
  - 5.1. Obtención de datos
  - 5.2. Carga de datos en MySQL
  - 5.3. Conexión con Visual Studio Code (Python)
  - 5.4. Exploración y limpieza de datos
- 6. Análisis de datos, resultados y hallazgos**
- 7. Interpretaciones de la evidencia y conclusiones**
  - 7.1. Hipótesis 1 (H1)
  - 7.2. Hipótesis 2 (H2)
- 8. Random forest**
  - 8.1 Modelo K-means**
- 9. Referencias**

## 1. Introducción

Durante el periodo comprendido entre 2021 y 2025, Barranquilla ha experimentado fluctuaciones significativas en las tasas de accidentabilidad, motivadas por múltiples factores socio-demográficos y urbanos. En este análisis, se examinan variables clave como el sexo y la edad de las personas involucradas, el tipo de accidente como colisiones vehiculares, atropellos, caídas, accidentes de motos o bicicletas, el día de la semana en que ocurren y las zonas geográficas de la ciudad con mayor incidencia. Esta combinación de variables permite identificar patrones y correlaciones, esenciales para comprender quiénes están más expuestos y en qué sectores se concentran los siniestros.

La integración de estas variables brinda una visión integral de la siniestralidad vial en Barranquilla. El sexo y la edad permiten desagregar la población en grupos vulnerables, mientras que el tipo de accidente ayuda a diseñar intervenciones específicas.

A lo largo del estudio se presentan los principales hallazgos derivados del análisis de datos, con el fin de ofrecer una visión más clara del contexto vial de la ciudad y evaluar la efectividad de las medidas de seguridad implementadas en los últimos años.

## **2. Objetivo del análisis**

Investigar y representar visualmente la dinámica de los accidentes de tránsito ocurridos en Barranquilla entre 2021 y 2025, con el propósito de identificar los factores que condicionan su gravedad y sustentar, con base en la evidencia, acciones que disminuyan sus consecuencias sobre la población.

### **3. Hipótesis principal**

La severidad de los accidentes viales en Barranquilla se ve influenciada por variables sociodemográficas y contextuales, tales como el sexo, la edad, el tipo de siniestro, el día de la semana y la ubicación geográfica del incidente.

#### **Hipótesis 1:**

Los accidentes viales con mayor número de víctimas mortales se concentran en zonas específicas de Barranquilla.

#### **Hipótesis 2:**

La gravedad de los accidentes de tránsito está fuertemente relacionada con la edad y el rol de la víctima, donde los motociclistas jóvenes (18-30 años) y peatones mayores (más de 60 años) presentan un riesgo significativamente mayor de resultar con lesiones graves o fatales.

## 4. Metodología

### 4.1 Fuente de datos

El análisis se basa en información proveniente de una fuente pública externa: el conjunto de datos titulado "*Accidentalidad Barranquilla - Detalle de víctimas*", disponible en el portal oficial del Gobierno de Colombia ([datos.gov.co](https://www.datos.gov.co)). Este conjunto recopila registros de accidentes de tránsito en el distrito de Barranquilla, según los informes policiales de accidentes de tránsito (IPAT).

Para este proyecto, se seleccionaron exclusivamente los datos correspondientes a la ciudad de Barranquilla, abarcando el período 2021–2025. Cabe destacar que los datos correspondientes a la vigencia actual son preliminares y están sujetos a actualizaciones.

### 4.2 Herramientas utilizadas

- a. **MySQL:** Utilizado para la carga y estructuración de la base de datos, permitiendo la importación masiva de datos y la organización adecuada de la información.
- b. **Python en Visual Studio Code (Pandas, Matplotlib, Seaborn):** Empleado procesamiento, limpieza, transformación, análisis estadístico y visualización de datos, así como la creación de visualizaciones y gráficos.
- c. **Power BI:** Utilizado para la creación de dashboards interactivos y presentación de resultados del análisis.
- d. **maching, modelos para predecir....**

### 4.3 Técnicas de análisis aplicadas

- a. **Análisis de datos descriptivo:** Se calcularon estadísticas básicas (media, mediana) para entender la distribución de los datos.
- b. **Análisis temporal:** Se analizó la evolución de las multas a lo largo del tiempo, realizando agrupaciones por año y mes.
- c. **Segmentación y agrupación:** Se agruparon datos por variables para identificar patrones y diferencias significativas entre años, además del tipo de vehículo y evaluar diferencias entre los distintos grupos.

## 5. Pasos preliminares realizados

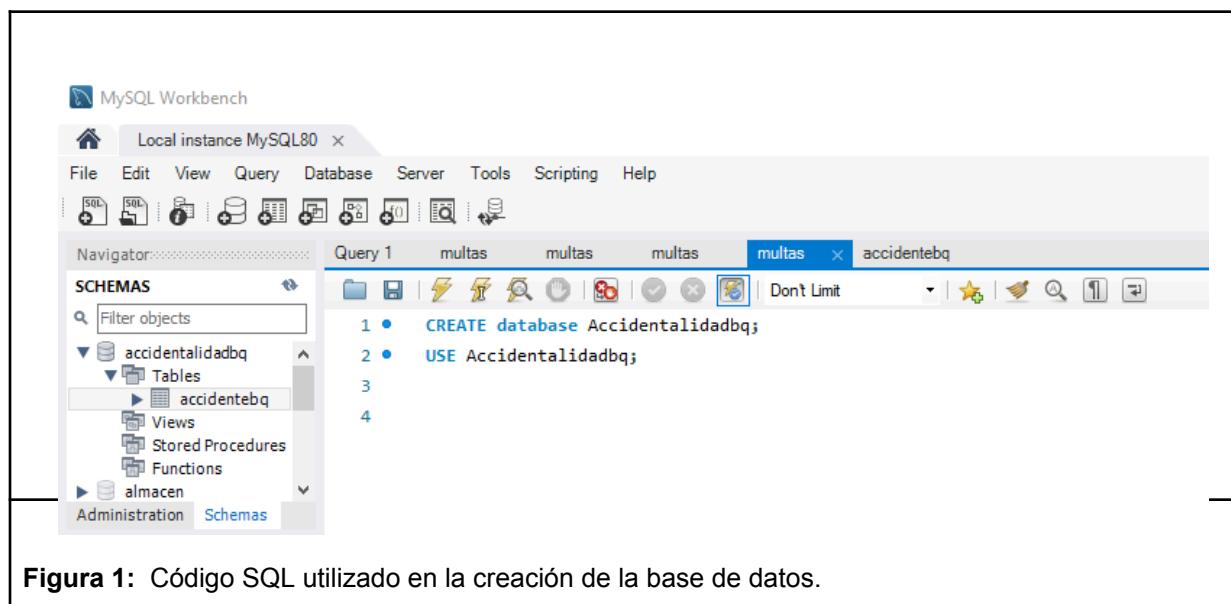
### 5.1 Obtención de datos:

Se descargaron los datos en formato .csv desde el portal oficial datos.gov.co, específicamente el conjunto de datos titulado “Accidentalidad en Barranquilla - Víctimas”.

La información contenida incluía variables como: Fecha\_Accidente, Direccion\_Accidente, Condicion\_Victima, Gravedad\_Accidente, Clase\_Accidente, Sexo\_Victima, Edad\_victima, Cantidad\_Victimas.

### 5.2 Carga de datos en MySQL:

Para estructurar los datos, se creó una base de datos en MySQL llamada Accidentalidadbq. A continuación, se utilizó la herramienta de importación de archivos CSV de MySQL Workbench para cargar los datos directamente en una tabla denominada accidentebq.



The screenshot shows the MySQL Workbench interface. The title bar says "MySQL Workbench" and "Local instance MySQL80". The menu bar includes File, Edit, View, Query, Database, Server, Tools, Scripting, and Help. The toolbar has various icons for database management. The Navigator pane on the left shows the "SCHEMAS" section with "accidentalidadbq" expanded, revealing "Tables", "accidentebq", "Views", "Stored Procedures", and "Functions". The "Tables" section also lists "almacen". The "Query 1" tab is active, displaying the following SQL code:

```
1 • CREATE database Accidentalidadbq;
2 • USE Accidentalidadbq;
3
4
```

**Figura 1:** Código SQL utilizado en la creación de la base de datos.

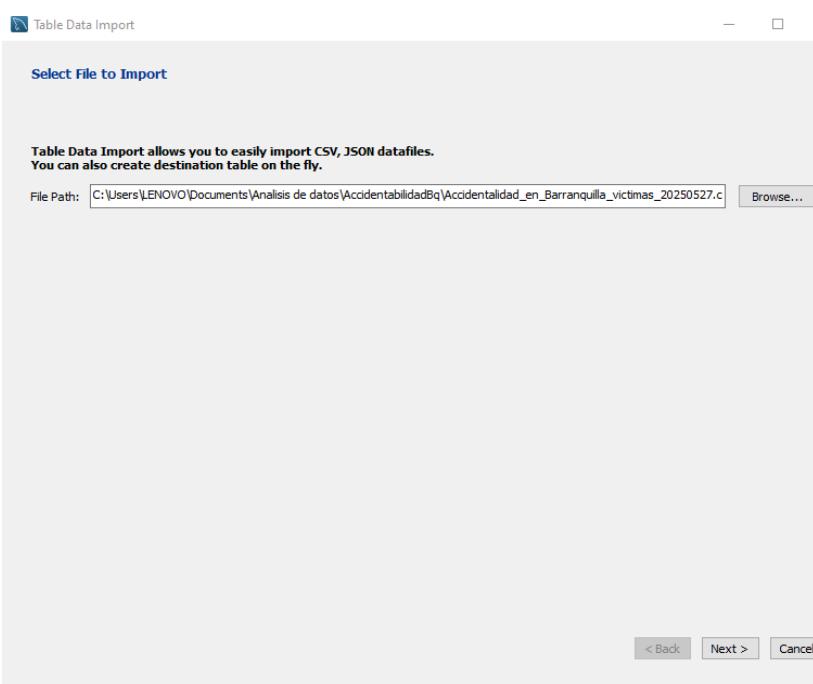


Figura 2: Proceso de importación del archivo .csv.

A screenshot of the MySQL Workbench interface. The title bar says "MySQL Workbench" and "Local instance MySQL80". The main area shows a database named "accidentalidadBq" with a single table "accidenteBq". A query editor window is open with the SQL command "SELECT \* FROM accidentalidadBq.accidenteBq;". The results grid displays data from the table, including columns like Fecha\_Accidente, DIRECCION\_ACCIDENTE, CONDICION\_VICTIMA, GRAVEDAD\_ACCIDENTE, CLASE\_ACCIDENTE, SEXO\_VICTIMA, EDAD\_VICTIMA, and CANTIDAD\_VICTIMAS. The data shows various accident details such as Peaton, herido, Atropello, etc., across different dates and locations.

Figura 3: Visualización de los datos cargados en la tabla accidentebq.

### 5.3. Conexión de base de datos desde MySQL con Visual Studio Code

Se realizó la conexión a la base de datos MySQL desde Python utilizando la librería pymysql. Esto permitió extraer los datos necesarios desde la tabla accidentebq para su posterior análisis. La conexión y consulta se llevaron a cabo en el entorno de desarrollo Visual Studio Code, como se muestra en la **Figura 5**.

```
# Conexion a una base de datos MySQL
import pymysql

conn = pymysql.connect(
    host="localhost",
    user="root",
    password="Mysql123#",
    database="Accidentalidadbq"
)

cursor = conn.cursor()

cursor.execute("SELECT * FROM accidentebq")

resultados = cursor.fetchall()
```

**Figura 5.** Conexión de la base de datos MySQL con Visual Studio Code mediante Python e importación de la librerías.

### 5.4. Exploración y limpieza de datos con Python

Se llevó a cabo un proceso de exploración y limpieza de datos utilizando la librería **pandas**. Inicialmente se revisaron los tipos de datos y se realizaron las conversiones necesarias para garantizar un tratamiento adecuado de la información. Se separaron las columnas de fecha y hora, y se filtraron los registros correspondientes al periodo 2021-2025. Además, se identificaron y eliminaron registros duplicados, así como aquellos con valores nulos o inconsistentes. Como parte del enriquecimiento de los datos, se generaron nuevas columnas para determinar el día de la semana del accidente y el rango de edad de la víctima. Al finalizar este proceso, la información tratada fue exportada a un archivo .csv para su posterior análisis. Ver código en la Figura 6 y resultados en la Figura 7.

```

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

print(" Análisis de accidentalidad vial en la ciudad de Barranquilla 2022-2025")

columnas = ["Fecha_Accidente", "DIRECCION_ACCIDENTE", "CONDICION_VICTIMA", "GRAVEDAD_ACCIDENTE", "CLASE_ACCIDENTE",
            "SEXO_VICTIMA", "EDAD_VICTIMA", "CANTIDAD_VICTIMA"]
df = pd.DataFrame(resultados, columns=columnas)

print(df)
print(df.shape)
print(df.dtypes)

#Separar la columna Fecha_Accidente en dos columnas: FECHA y HORA
df[['FECHA', 'HORA']] = df['Fecha_Accidente'].str.split(' ', n=1, expand=True)
df['FECHA'] = pd.to_datetime(df['FECHA'], format='%m/%d/%Y', errors='coerce')
#Filtrar los años 2021 a 2025
df = df[(df['FECHA'].dt.year >= 2021) & (df['FECHA'].dt.year <= 2025)]

df.drop('HORA', axis=1, inplace=True)
df.drop('Fecha_Accidente', axis=1, inplace=True)

df = df.astype (
    {
        'DIRECCION_ACCIDENTE': 'string',
        'CONDICION_VICTIMA': 'string',
        'GRAVEDAD_ACCIDENTE': 'string',
        'CLASE_ACCIDENTE': 'string',
        'SEXO_VICTIMA': 'string',
    }
)
print(df.info())
#Antes de eliminar
print(f"\n Duplicados antes de limpiar: {df.duplicated().sum()}")
print(df.duplicated())

# Eliminar duplicados
df = df.drop_duplicates()
print(f"\n Duplicados después de limpiar: {df.duplicated().sum()}")

print("Valores nulos antes de limpieza:\n", df.isnull().sum())

df['EDAD_VICTIMA'] = pd.to_numeric(df['EDAD_VICTIMA'], errors='coerce')
df['CANTIDAD_VICTIMA'] = pd.to_numeric(df['CANTIDAD_VICTIMA'], errors='coerce')

# Eliminar registros con valores cero o nulos o vacíos
df = df.dropna()
df = df[df['CANTIDAD_VICTIMA'] > 0]
df = df[(df['EDAD_VICTIMA'] > 0) & (df['EDAD_VICTIMA'] <= 100)]
df['SEXO_VICTIMA'] = df['SEXO_VICTIMA'].str.strip()
df = df[df['SEXO_VICTIMA'] != '']
df = df.dropna(subset=['SEXO_VICTIMA'])

print("Valores nulos después de limpieza:\n", df.isnull().sum())
print(f"Filas después de limpiar: {len(df)}")

# Nueva columna: Día de la semana
df['DIA_SEMANA'] = df['FECHA'].dt.day_name()

# Nueva columna: Rango de edad
bins = [0, 17, 30, 45, 60, 200]
labels = ['<18', '18-30', '31-45', '46-60', '60+']
df['RANGO_EDAD'] = pd.cut(df['EDAD_VICTIMA'], bins=bins, labels=labels)

#Guardar base de datos limpia con los cambios aplicados
df.to_csv('accidentesbq_limpio.csv', index=False)

```

**Figura 6. Proceso de exploración y limpieza de datos en Python.**

Análisis de accidentalidad vial en la ciudad de Barranquilla 2022-2025							
	Fecha_Accidente	DIRECCION ACCIDENTE \			EDAD_VICTIMA	CANTIDAD_VICTIMA	
0	01/01/2018 12:00:00 AM	CL 87 9H 24			0	57	1
1	01/01/2018 12:00:00 AM	CLLE 119B CRA 11B			1	13	1
2	01/01/2018 12:00:00 AM	CR 8 CL 41			2	32	1
3	01/01/2018 12:00:00 AM	CR 8 CL 41			3	26	1
4	01/01/2018 12:00:00 AM	CR 8 CL 41			4	53	1
...	...	...			...	...	...
16639	04/29/2025 12:00:00 AM	CALLE 110 CON CARRERA 44			16639	27	1
16640	04/29/2025 12:00:00 AM	CALLE 110 CON CARRERA 44			16640	25	1
16641	04/29/2025 12:00:00 AM	CALLE 110 CRA 43			16641	35	1
16642	04/30/2025 12:00:00 AM	CALLE 48 CARRERA 46			16642	29	1
16643	04/30/2025 12:00:00 AM	CALLE 87 CARRERA 73 BARRANQUILLA			16643	25	1
[16644 rows x 8 columns]							
(16644, 8)							
Fecha_Accidente object							
DIRECCION_ACCIDENTE object							
CONDICION_VICTIMA object							
GRAVEDAD_ACCIDENTE object							
CLASE_ACCIDENTE object							
SEXO_VICTIMA object							
EDAD_VICTIMA int64							
CANTIDAD_VICTIMA int64							
dtype: object							
<class 'pandas.core.frame.DataFrame'>							
Index: 10325 entries, 6319 to 16643							
Data columns (total 8 columns):							
# Column Non-Null Count Dtype							
---							
0	DIRECCION_ACCIDENTE	10325	non-null	string			
1	CONDICION_VICTIMA	10325	non-null	string			
2	GRAVEDAD_ACCIDENTE	10325	non-null	string			
3	CLASE_ACCIDENTE	10325	non-null	string			
4	SEXO_VICTIMA	10325	non-null	string			
5	EDAD_VICTIMA	10325	non-null	int64			
6	CANTIDAD_VICTIMA	10325	non-null	int64			
7	FECHA	10325	non-null	datetime64[ns]			
dtypes: datetime64[ns](1), int64(2), string(5)							
memory usage: 726.0 KB							
None							
Duplicados antes de limpiar: 0							
5319	False						
5320	False						
5321	False						
5322	False						
5323	False						
...							
16639	False						
16640	False						
16641	False						
16642	False						
16643	False						
length: 10325, dtype: bool							

**Figura 7. Resultados preliminares de la exploración y limpieza de datos: duplicados, nulos y tipos de datos corregidos.**

## 6. Análisis de datos, resultados y hallazgos

Una vez finalizado el proceso de limpieza, se procedió al análisis estadístico de los datos de accidentalidad con el objetivo de responder las preguntas de investigación y comprobar las hipótesis planteadas. Entre los cálculos realizados se encuentran: el conteo de muertes por tipo de accidente, el cálculo de la tasa de mortalidad por tipo de accidente, y la identificación de las direcciones con mayor número de muertes y accidentes. Asimismo, se determinó el top 10 de zonas con mayor tasa de mortalidad (considerando aquellas con al menos cinco accidentes registrados). Este análisis permitió establecer patrones geográficos y tipológicos en los accidentes de tránsito. **Ver código en la Figura 8.**

```

# Calculos
print("\nConteo de muertes por tipo de accidente")
muertes_por_clase = df[df['GRAVEDAD_ACCIDENTE'] == 'muerto']['CLASE_ACCIDENTE'].value_counts()  Undefined name `df`  Undefined name `df`
print(muertes_por_clase)

print("\nTasa de mortalidad por tipo de accidente - compara cuántas muertes hay en proporción al total de accidentes de cada tipo")
# Total de accidentes por tipo
total_por_clase = df['CLASE_ACCIDENTE'].value_counts()  Undefined name `df`
print(total_por_clase)

# Tasa de mortalidad
tasa_mortalidad = (muertes_por_clase / total_por_clase).sort_values(ascending=False)
print(tasa_mortalidad)

# Asegurar formato uniforme
df['GRAVEDAD_ACCIDENTE'] = df['GRAVEDAD_ACCIDENTE'].str.strip().str.lower() # herido / muerto  Undefined name `df`  Undefined name `df`
df['DIRECCION ACCIDENTE'] = df['DIRECCION ACCIDENTE'].str.strip().str.upper()  Undefined name `df`  Undefined name `df`
# Agrupar por dirección y tipo de gravedad (muerto/herido)
zonas = df.groupby(['DIRECCION ACCIDENTE', 'GRAVEDAD_ACCIDENTE']).size().unstack(fill_value=0)  Undefined name `df`

# Asegurar que las columnas 'muerto' y 'herido' existen (por si no hay datos)
for col in ['muerto', 'herido']:
    if col not in zonas.columns:
        zonas[col] = 0

# Total y tasa de mortalidad:
zonas['TOTAL_ACCIDENTES'] = zonas['muerto'] + zonas['herido'] #TOTAL_ACCIDENTES: suma de muertos y heridos por dirección
zonas['TASA_MORTALIDAD_%'] = (zonas['muerto'] / zonas['TOTAL_ACCIDENTES']) * 100
zonas['TASA_MORTALIDAD_%'] = zonas['TASA_MORTALIDAD_%'].round(2)

# Top 10 direcciones con más muertes
top_muertos = zonas.sort_values('muerto', ascending=False).head(10)
print("\n Zonas con más víctimas fatales:")
print(top_muertos[['muerto', 'TOTAL_ACCIDENTES']])

# Top 10 direcciones con más accidentes (totales)
top_accidentes = zonas.sort_values('TOTAL_ACCIDENTES', ascending=False).head(10)
print("\n Zonas con más accidentes totales:")
print(top_accidentes[['TOTAL_ACCIDENTES', 'muerto']])

# Top zonas con mayor tasa de mortalidad (%)
top_tasa = zonas[zonas['TOTAL_ACCIDENTES'] >= 5].sort_values('TASA_MORTALIDAD_%', ascending=False).head(10)
print("\n Zonas con mayor tasa de mortalidad (%):")
print(top_tasa[['TASA_MORTALIDAD_%', 'TOTAL_ACCIDENTES']])

```

**Figura 8. Análisis de datos y cálculos estadísticos realizados en Python.**

## 7. Interpretaciones de la evidencia y conclusiones:

A partir de los resultados obtenidos durante el análisis de los datos de accidentalidad vial en Barranquilla, se realizaron las siguientes interpretaciones en relación con las hipótesis planteadas:

**7.1. Hipótesis 1 (H1):** “Los accidentes viales con mayor número de víctimas mortales se concentran en zonas específicas de Barranquilla.”

Para comprobar esta hipótesis, se abordaron preguntas clave como:

- ¿Cuántas muertes se registran por tipo de accidente?
- ¿Qué zonas concentran mayor cantidad de víctimas mortales?
- ¿Cuáles son las direcciones con más accidentes y mayor tasa de mortalidad?

Se desarrolló un análisis agrupando los accidentes por dirección y gravedad (muerto/herido), lo que permitió identificar las zonas con mayor número de muertes y calcular la tasa de mortalidad por ubicación. Esto facilitó la identificación de las áreas críticas con alta incidencia de fatalidades visualizadas en las figuras **Figura 9 y 10**.

El análisis fue desarrollado mediante scripts implementados en Visual Studio Code y la posterior visualización de los resultados en power bi, lo que permitió representar las siguientes gráficas

falta visualizaciones power bi

```

# Visualización para Top 10 zonas con más muertes:
plt.figure(figsize=(12, 6))
bars = plt.barh(top_muertos.index, top_muertos['muerto'], color='pink')
plt.xlabel('Cantidad de fallecidos')
plt.title('Zonas con mayor cantidad de víctimas fatales')
plt.gca().invert_yaxis()
plt.tight_layout()

# Etiquetas numéricas al final de cada barra
for bar in bars:
    width = bar.get_width()
    plt.text(width + 1, bar.get_y() + bar.get_height()/2,
             str(int(width)), va='center')

plt.show()

# Visualización para la cantidad total de accidentes:
plt.figure(figsize=(12, 6))
bars_total = plt.barh(top_accidentes.index, top_accidentes['TOTAL_ACCIDENTES'], color='skyblue')
plt.xlabel('Cantidad total de accidentes')
plt.title('Zonas con mayor cantidad total de accidentes')
plt.gca().invert_yaxis()
plt.tight_layout()

# Etiquetas numéricas
for bar in bars_total:
    width = bar.get_width()
    plt.text(width + 1, bar.get_y() + bar.get_height()/2,
             str(int(width)), va='center')

plt.show()

# Visualización de la tasa de mortalidad:
plt.figure(figsize=(12, 6))
bars_tasa = plt.barh(top_tasa.index, top_tasa['TASA_MORTALIDAD_%'], color='orange')
plt.xlabel('Tasa de mortalidad (%)')
plt.title('Zonas con mayor tasa de mortalidad (con al menos 5 accidentes)')
plt.gca().invert_yaxis()
plt.tight_layout()

# Etiquetas con porcentaje
for bar in bars_tasa:
    width = bar.get_width()
    plt.text(width + 0.5, bar.get_y() + bar.get_height()/2, f'{width:.2f}%', va='center')

plt.show()

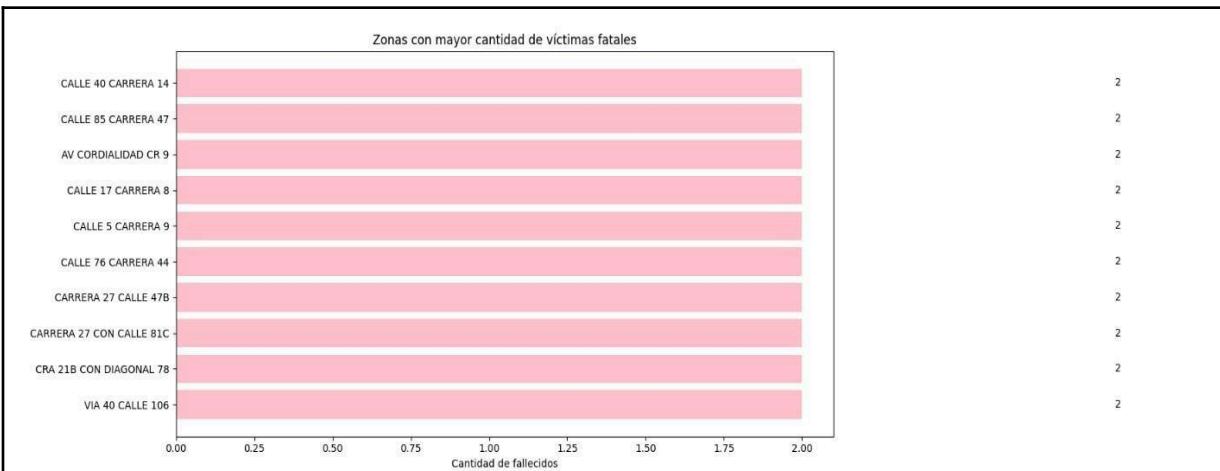
```

**Figura 9. Código para visualización de la Hipótesis 1.**

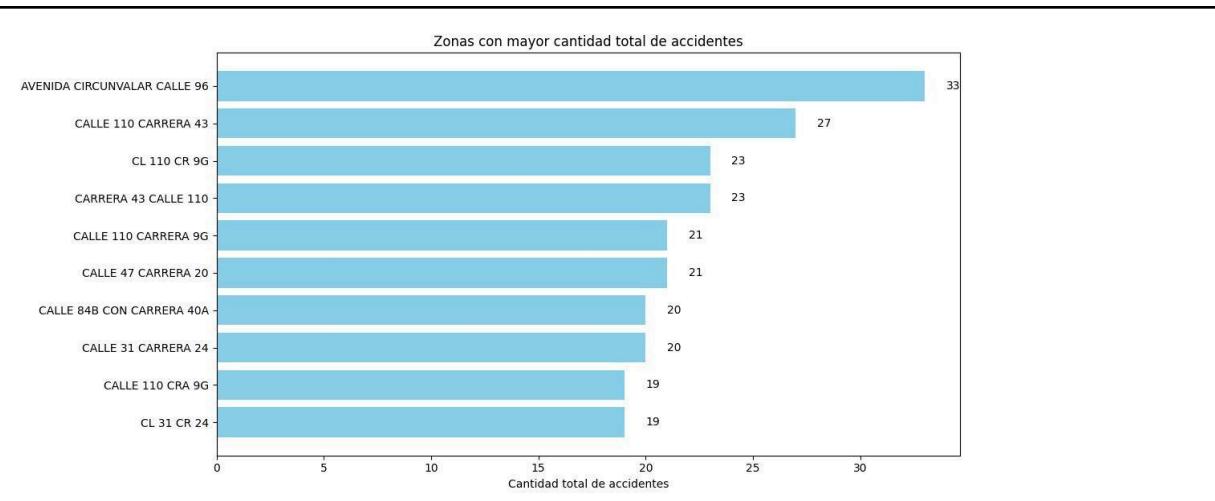
Conteo de muertes por tipo de accidente		
CLASE_ACCIDENTE		
Choque	115	
Atropello	53	
Otro	7	
Caida Ocupante	3	
Volcamiento	3	
Name: count, dtype: Int64		
Tasa de mortalidad por tipo de accidente - compara cuántas muertes hay en proporción al total de accidentes de cada tipo		
CLASE_ACCIDENTE		
Otro	0.107692	Zonas con mayor tasa de mortalidad (%):
Atropello	0.048007	GRAVEDAD_ACCIDENTE TASA_MORTALIDAD_% TOTAL_ACCIDENTES
Volcamiento	0.026549	DIRECCION ACCIDENTE
Caida Ocupante	0.018868	CALLE 40 CARRERA 14 28.57 7
Choque	0.013002	CALLE 5 CARRERA 9 22.22 9
Incendio	<NA>	CALLE 17 CARRERA 8 22.22 9
Name: count, dtype: Float64		CALLE 30 CARRERA 36 20.00 5
Zonas con más víctimas fatales:		
GRAVEDAD_ACCIDENTE	muerto	TOTAL_ACCIDENTES
DIRECCION ACCIDENTE		
CALLE 40 CARRERA 14	2	7
CALLE 85 CARRERA 47	2	12
AV CORDIALIDAD CR 9	2	2
CALLE 17 CARRERA 8	2	9
CALLE 5 CARRERA 9	2	9
CALLE 76 CARRERA 44	2	10
CARRERA 27 CALLE 47B	2	2
CARRERA 27 CON CALLE 81C	2	3
CRA 21B CON DIAGONAL 78	2	14
VIA 40 CALLE 106	2	4
Zonas con más accidentes totales:		
GRAVEDAD_ACCIDENTE		TOTAL_ACCIDENTES
DIRECCION ACCIDENTE		
AVENIDA CIRCUNVALAR CALLE 96	33	0
CALLE 110 CARRERA 43	27	1
CL 110 CR 9G	23	1
CARRERA 43 CALLE 110	23	0
CALLE 110 CARRERA 9G	21	0
CALLE 47 CARRERA 20	21	0
CALLE 84B CON CARRERA 40A	20	0
CALLE 31 CARRERA 24	20	0
CALLE 110 CRA 9G	19	0
CL 31 CR 24	19	0

**Figura 10. Análisis de Datos y Hallazgos de H1**

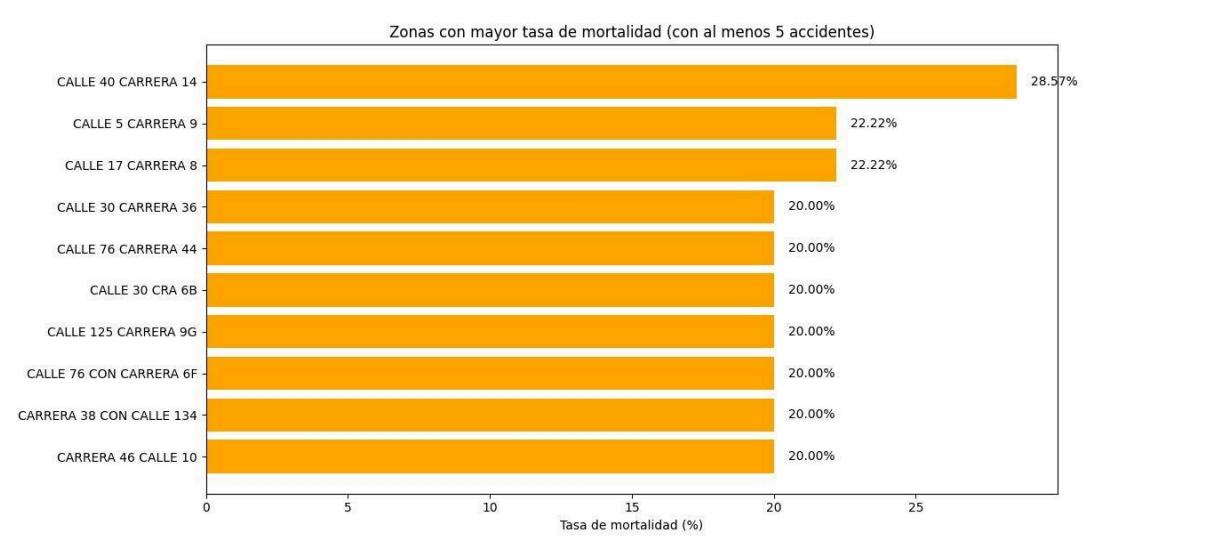
A continuación, se muestran las gráficas elaboradas en función de las preguntas planteadas en la **hipótesis 1**



**Figura 12. Gráfico de Muertes Registradas por tipo de accidente.**



**Figura 13. Gráfico de las zonas con mayor cantidad de accidentes.**



**Figura 14. Gráfico de las zonas con mayor tasa de mortalidad.**

**Conclusión de la hipótesis 1 (H1): “Los accidentes viales con mayor número de víctimas mortales se concentran en zonas específicas de Barranquilla.”**

Con base en el análisis de los datos, se obtuvieron las siguientes evidencias:

- Se observaron incrementos de datos, significativos en los siguientes años (por ejemplo +108%, +98%, +68%).
- Se registraron caídas en otros períodos (por ejemplo. -27%, -39%).
- Solo 7 de 14 años se evidenció un aumento en la cantidad de multas superior al 50%.
- En general, el comportamiento no refleja tendencia de crecimiento constante, sino un patrón fluctuante e irregular

Este hallazgo justifica la pertinencia de la investigación, ya que permite comprender el comportamiento de las infracciones de tránsito en la ciudad de Barranquilla, lo cual estaría influenciado por los siguientes factores:

- Aumento o reducción de operaciones de control vial.
- Cambios en el comportamiento de los conductores.
- Variaciones en la normativa o mecanismos de transporte.

**7.2 Hipótesis (H2) "La gravedad de los accidentes de tránsito está fuertemente relacionada con la edad y el rol de la víctima, donde los motociclistas jóvenes (18-30 años) y peatones mayores (más de 60 años) presentan un riesgo significativamente mayor de resultar con lesiones graves o fatales. ."**

Además del análisis por zonas y tipos de accidente, se realizó un estudio detallado sobre la variable edad de la víctima, con el fin de evaluar su relación con la gravedad del accidente. Para ello, se formularon las siguientes preguntas clave:

- ¿Cuál es la edad promedio de las víctimas de accidentes?
- ¿Cómo varía la edad entre víctimas mortales y heridas?
- ¿La edad de las víctimas sigue una distribución normal?

El análisis se desarrolló mediante scripts en Python implementados en Visual Studio Code. Se calcularon medidas estadísticas generales (media, mediana, moda, desviación estándar y cuartiles) para toda la muestra, y luego se compararon los valores entre víctimas heridas y fallecidas. Adicionalmente, se generó un histograma con distribución KDE diferenciando entre niveles de gravedad, y se aplicaron pruebas estadísticas (Shapiro-Wilk) para evaluar si los datos se ajustaban a una distribución normal.

Este análisis permitió evidenciar patrones de edad asociados a mayor letalidad, como una mayor concentración de muertes en ciertos rangos de edades, especialmente en motociclistas jóvenes y peatones mayores, apoyando así la hipótesis planteada.

Ver código en la Figura 15 y resultados en la Figura 16.

```
# 1. Medidas generales
# Estadísticas generales de EDAD_VICTIMA
print("Media:", df['EDAD_VICTIMA'].mean())
print("Mediana:", df['EDAD_VICTIMA'].median())
print("Moda:", df['EDAD_VICTIMA'].mode()[0])
print("Desviación estándar:", df['EDAD_VICTIMA'].std())
print("Cuartiles:\n", df['EDAD_VICTIMA'].quantile([0.25, 0.5, 0.75]))
# 2. Comparar estadísticas entre muertos y heridos:
df.groupby('GRAVEDAD_ACCIDENTE')['EDAD_VICTIMA'].describe()

# Análisis de distribuciones
# Histograma de edades de las víctimas según gravedad del accidente
import seaborn as sns
import matplotlib.pyplot as plt
sns.histplot(data=df, x='EDAD_VICTIMA', hue='GRAVEDAD_ACCIDENTE', kde=True)
plt.title("Distribución de edades según gravedad")
plt.show()

# Se ajusta a una distribución normal?
from scipy.stats import shapiro, normaltest

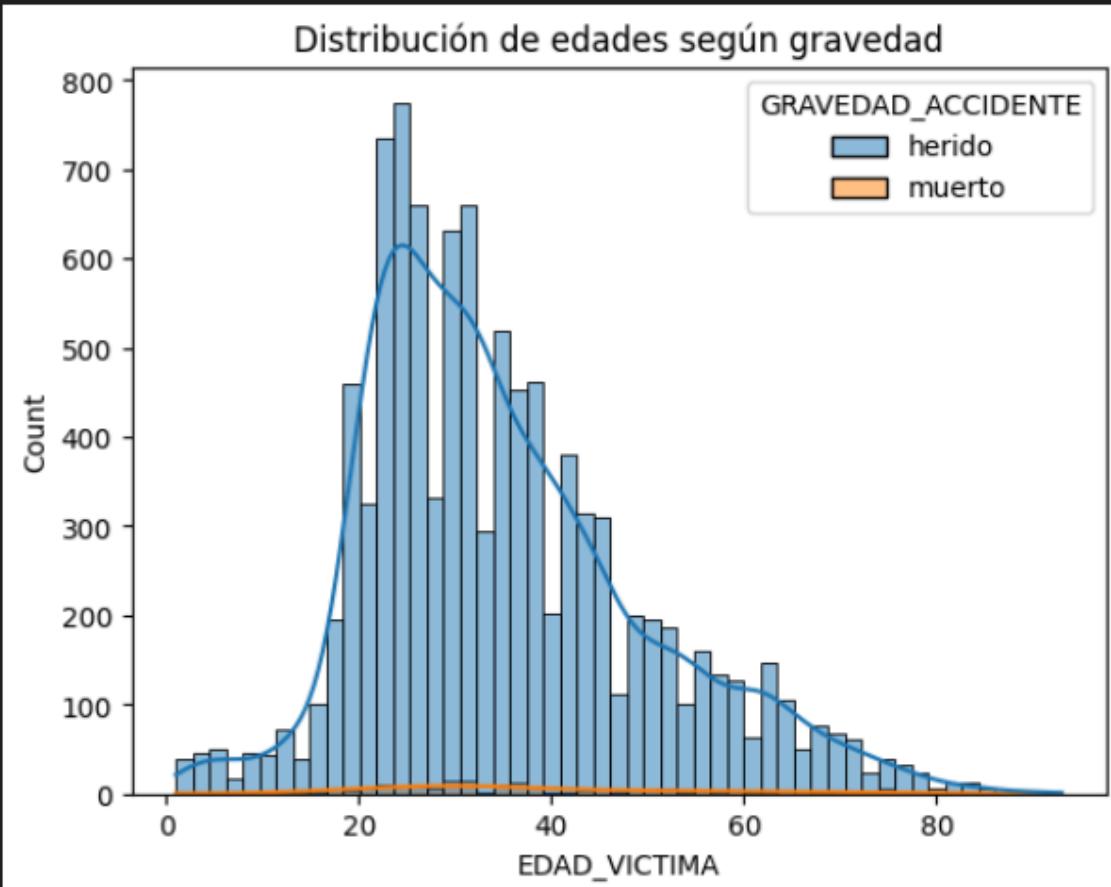
# Solo edades
stat, p = shapiro(df['EDAD_VICTIMA'])
print("Shapiro-Wilk p-value:", p)
```

**Figura 16. Código para analizar la relación entre edad y gravedad del accidente..**

Se calcularon estadísticas descriptivas de la edad de las víctimas, se visualiza su distribución según la gravedad del accidente y se aplica la prueba de normalidad Shapiro-Wilk.

### Análisis de Datos y Hallazgos H2:

```
Media: 34.99727811801303
Mediana: 32.0
Moda: 24
Desviación estándar: 14.618068200895955
Cuartiles:
  0.25    24.0
  0.50    32.0
  0.75    43.0
Name: EDAD_VICTIMA, dtype: float64
```



Shapiro-Wilk p-value: 8.13400190474322e-49

Figura 17. Resultado obtenido en Visual Studio Code para la H2.

```

# Visualización de datos

# 1 Distribución de gravedad
plt.figure()
sns.countplot(data=df, x='GRAVEDAD_ACCIDENTE')
plt.title('Distribución de Gravedad del Accidente')
plt.show()

# 2 Gravedad vs Rango de Edad
plt.figure(figsize=(8,5))
sns.countplot(data=df, x='RANGO_EDAD', hue='GRAVEDAD_ACCIDENTE', palette='Set2')
plt.title("Gravedad del accidente por rango de edad")
plt.xlabel("Rango de edad")
plt.ylabel("Cantidad")
plt.legend(title='Gravedad')
plt.tight_layout()
plt.show()

# 3 Gravedad vs Condición de la víctima
plt.figure(figsize=(10,5))
sns.countplot(data=df, x='CONDICION_VICTIMA', hue='GRAVEDAD_ACCIDENTE', palette='Set1')
plt.title("Gravedad del accidente por tipo de víctima")
plt.xlabel("Condición de la víctima")
plt.ylabel("Cantidad")
plt.xticks(rotation=45)
plt.legend(title='Gravedad')
plt.tight_layout()
plt.show()

# 4 Gravedad según edad y condición
plt.figure()
sns.boxplot(data=df[df['GRAVEDAD_ACCIDENTE'] != 'ilesos'], x='CONDICION_VICTIMA', y='EDAD_VICTIMA', hue='GRAVEDAD_ACCIDENTE')
plt.title("Edad y Gravedad según condición")
plt.xticks(rotation=45)
plt.show()

# 5 Mapa de calor
heatmap_data = df.pivot_table(
    values='GRAVEDAD_ACCIDENTE',
    index='CONDICION_VICTIMA',
    columns='RANGO_EDAD',
    aggfunc=lambda x: (x == 'muerto').mean()
)

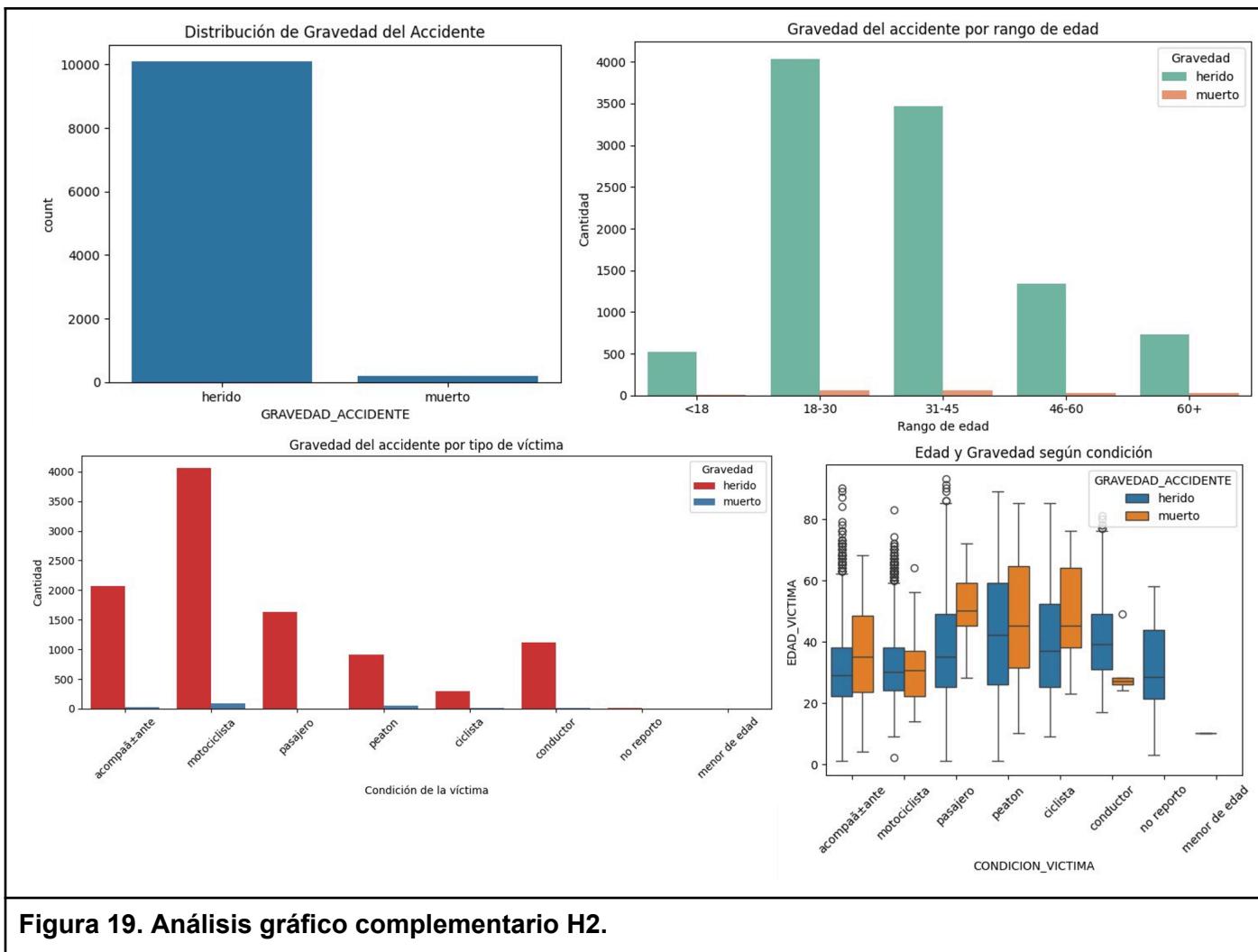
plt.figure(figsize=(10, 6))
sns.heatmap(heatmap_data, annot=True, cmap='Reds', fmt=".2f")
plt.title('Proporción de fallecidos por condición y rango de edad')
plt.ylabel('Condición de la víctima')
plt.xlabel('Rango de edad')
plt.tight_layout()
plt.show()

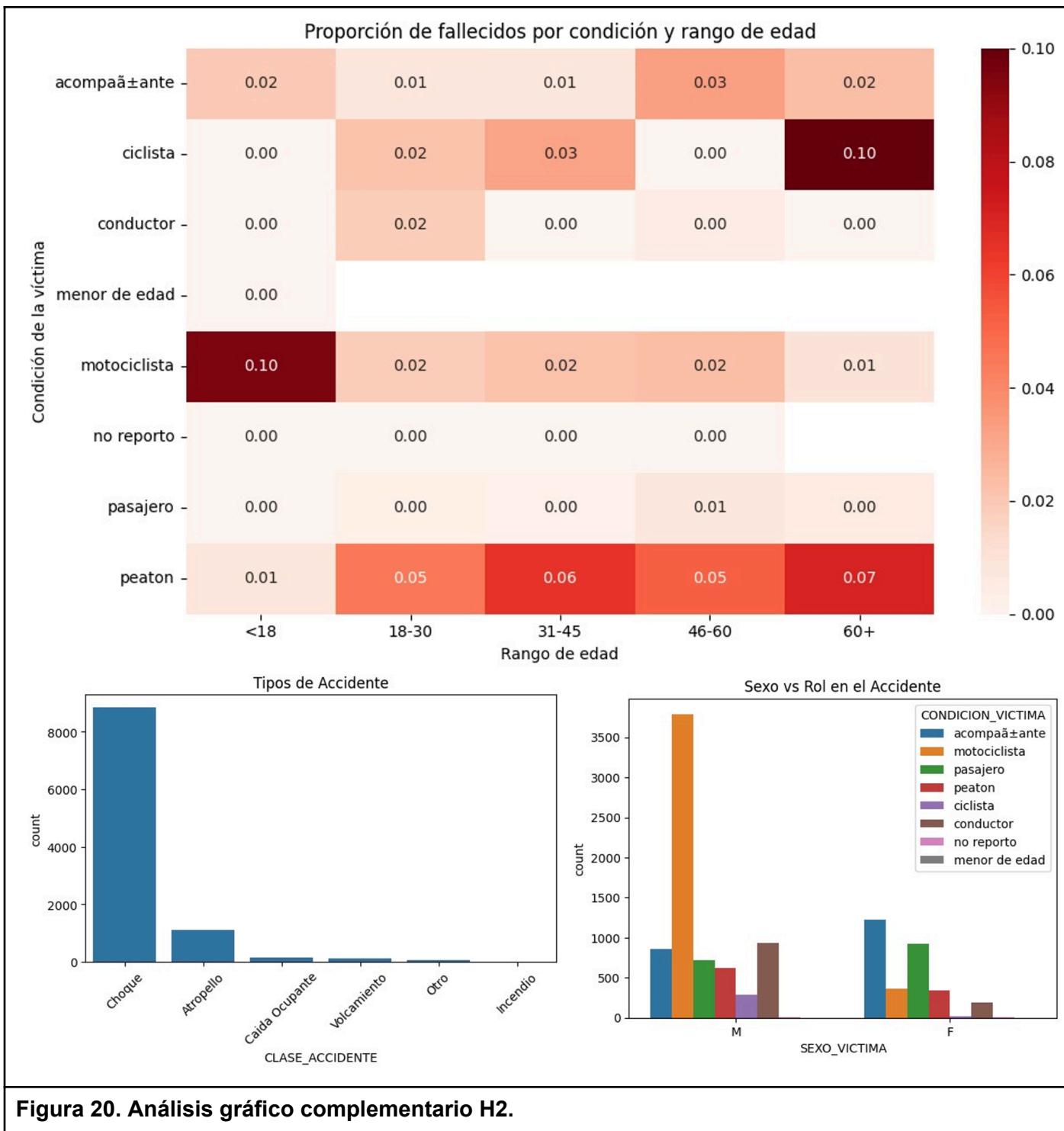
# 6 Distribución por sexo y condición
plt.figure()
sns.countplot(data=df, x='SEXO_VICTIMA', hue='CONDICION_VICTIMA')
plt.title('Sexo vs Rol en el Accidente')
plt.show()

# 7 Distribución por tipo de accidente
plt.figure()
sns.countplot(data=df, x='CLASE_ACCIDENTE', order=df['CLASE_ACCIDENTE'].value_counts().index)
plt.title('Tipos de Accidente')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

```

**Figura 18.** Script para obtener gráficas de la H2.





## **8. Modelo Random Forest.**

Se implementa un modelo de clasificación supervisada utilizando Random Forest para predecir la gravedad de los accidentes de tránsito. Se transforman variables categóricas, se dividen los datos en entrenamiento y prueba, y se evalúa el desempeño del modelo con métricas como accuracy y reporte de clasificación como se muestra en la Figura 21.

```

# Modelo 1 Random Forest para predecir la gravedad del accidente (modelo de clasificación supervisada)

# Importar librerías necesarias
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

#Cargar y limpiar datos,
df['GRAVEDAD_BINARIA'] = df['GRAVEDAD_ACCIDENTE'].apply(lambda x: 1 if x == 'muerto' else 0)

#Variables predictoras (puedes ajustar según tus datos),
variables = ['SEXO_VICTIMA', 'EDAD_VICTIMA', 'CONDICION_VICTIMA', 'CLASE_ACCIDENTE', 'DIA_SEMANA']

#Quitar filas con valores faltantes,
df_modelo = df[variables + ['GRAVEDAD_BINARIA']].dropna()

#convertimos las variables de texto a numero (columna binaria 0 o 1),
df_modelo = pd.get_dummies(df_modelo, columns=['SEXO_VICTIMA', 'CONDICION_VICTIMA', 'CLASE_ACCIDENTE', 'DIA_SEMANA'], drop_first=True)

#Separar X e y,
X = df_modelo.drop('GRAVEDAD_BINARIA', axis=1) #todo menos la columna de gravedad
y = df_modelo['GRAVEDAD_BINARIA']

#Dividir en conjunto de entrenamiento y prueba,
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

#entrenamiento del modelo
modelo = RandomForestClassifier(n_estimators=100, random_state=42)
modelo.fit(X_train, y_train)

#Hacer predicciones y evaluar el modelo
y_pred = modelo.predict(X_test)

print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nReporte de clasificación:\n", classification_report(y_test, y_pred))

```

**Figura 22. Modelo Random Forest para predecir la gravedad del accidente.**

Reporte de clasificación:				
	precision	recall	f1-score	support
0	0.98	1.00	0.99	2025
1	0.11	0.03	0.05	33
accuracy			0.98	2058
macro avg	0.55	0.51	0.52	2058
weighted avg	0.97	0.98	0.98	2058

**Figura 23. Resultados random forest**

```
from sklearn.metrics import ConfusionMatrixDisplay
ConfusionMatrixDisplay.from_estimator(modelo, X_test, y_test, display_labels=["No Muerto", "Muerto"], cmap="Reds")
plt.title("Matriz de Confusión")
plt.show()
```



**Figura 24. Matriz de confusión**

## 8.1 Modelo K-means

Este código implementa un modelo de clustering K-means que agrupa zonas de accidentes según la frecuencia y el día de la semana, con el objetivo de identificar patrones temporales y espaciales en la ocurrencia de siniestros viales. Utiliza datos preprocesados y visualiza los resultados en un gráfico de dispersión coloreado por clúster.

```

# Modelo 2 K-means: Clustering de zonas por frecuencia de accidentes

from sklearn.cluster import KMeans
import numpy as np

# Preparamos los datos: agrupamos por dirección y día de la semana
zonas_frecuencia = df.groupby(['DIRECCION ACCIDENTE', 'DIA_SEMANA']).size().reset_index(name='FRECUENCIA')

# Convertimos los días a números para el modelo
dias_map = {'Monday':0, 'Tuesday':1, 'Wednesday':2, 'Thursday':3, 'Friday':4, 'Saturday':5, 'Sunday':6}
zonas_frecuencia['DIA_NUM'] = zonas_frecuencia['DIA_SEMANA'].map(dias_map)

# Seleccionamos las variables para clustering
X_cluster = zonas_frecuencia[['FRECUENCIA', 'DIA_NUM']]

# Elegimos el número de clusters (puedes ajustar este valor)
kmeans = KMeans(n_clusters=4, random_state=42)
zonas_frecuencia['CLUSTER'] = kmeans.fit_predict(X_cluster)

# Visualización de los clusters
plt.figure(figsize=(10,6))
sns.scatterplot(data=zonas_frecuencia, x='DIA_NUM', y='FRECUENCIA', hue='CLUSTER', palette='Set2')
plt.title('Clusters de frecuencia de accidentes por día y zona')
plt.xlabel('Día de la semana (0=Lunes)')
plt.ylabel('Frecuencia de accidentes')
plt.show()

# Mostrar las zonas más críticas de cada cluster
for c in zonas_frecuencia['CLUSTER'].unique():
    top_zonas = zonas_frecuencia[zonas_frecuencia['CLUSTER'] == c].sort_values('FRECUENCIA', ascending=False).head(3)
    print(f"\nCluster {c} - Zonas y días más críticos:")
    print(top_zonas[['DIRECCION ACCIDENTE', 'DIA_SEMANA', 'FRECUENCIA']])

```

**Figura 25. Modelo K-means**

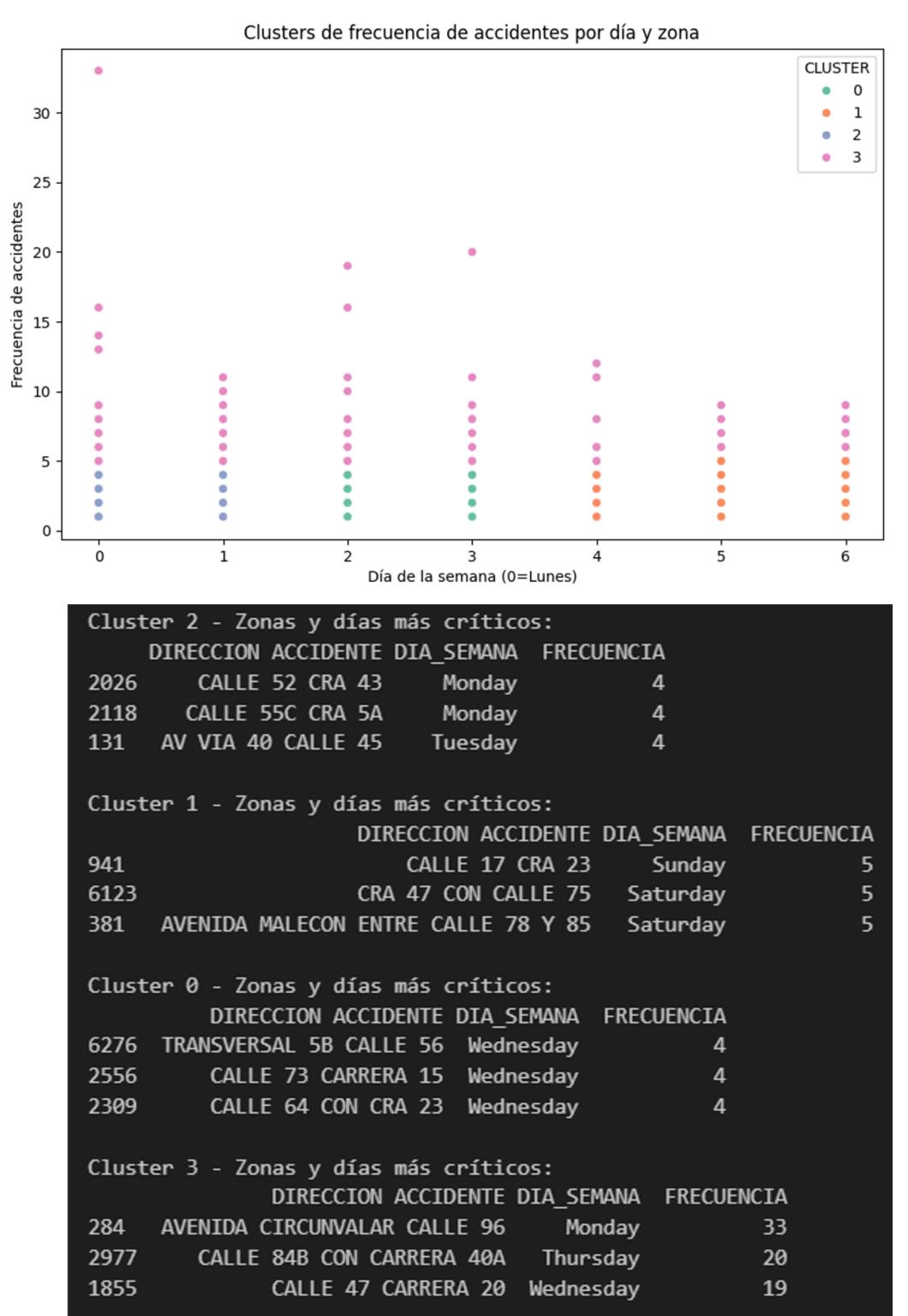


Figura 26. Hallazgos Kmeans

## 9. Referencias

Contraloría Distrital de Medellín. (2021). *Informe definitivo auditoría de cumplimiento: Gestión cobro de multas de tránsito.*  
<https://www.cdm.gov.co/cgm/Paginaweb/IP/Informes%20de%20Auditora%20PVCFT%202020/2/Informe%20Definitivo%20Auditor%C3%ADa%20de%20Cumplimiento%20Gesti%C3%B3n%20Cobro%20de%20Multas%20de%20Tr%C3%A1nsito.pdf>

Contraloría General de la República. (2020). *Informe de auditoría de cumplimiento No. 013 de 2020: Federación Colombiana de Municipios – SIMIT.*  
<https://www.fcm.org.co/wp-content/uploads/2021/02/Informe-CGR-CDSI-No.-013-de-2020-Auditoria-de-Cumplimiento-2019-FCM-SIMIT-1.pdf>

El Espectador. (2025, enero). *Hallazgos fiscales en Secretaría de Movilidad por ineeficiencia en cobros de multas.*  
<https://www.elespectador.com/bogota/hallazgos-fiscales-en-secretaria-de-movilidad-por-ineficiencia-en-cobros-de-multas/>

Federación Colombiana de Municipios. (s. f.). *Multas SIMIT – Historial de multas reportados en el Sistema Integrado de Información sobre Multas y Sanciones por Infracciones de Tránsito – SIMIT.* Datos.gov.co.  
<https://www.datos.gov.co/Funcion-publica/Multas-SIMIT/bgfy-53qq>

A partir de estos análisis se obtuvo:

- **Tipos de accidentes más letales.**
- **Tasa de mortalidad por tipo de accidente.**
- **Zonas con mayor número de muertes.**
- **Zonas con más accidentes registrados.**
- **Zonas con mayor tasa de mortalidad,** considerando al menos cinco casos por ubicación.

Estos resultados son clave para focalizar intervenciones en los puntos más vulnerables de la ciudad.

