Reporte proyecto PCD

El siguiente reporte presenta un análisis detallado de los datos de accidentes de tráfico, abarcando la limpieza de datos, análisis exploratorio y visualizaciones mediante gráficas.

1. Importación de Bibliotecas y Carga de Datos

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns #Importamos librerias

# Cargar datos
dtype_dict = {
    'Vehicle Make': 'str',
    'Vehicle Model': 'str',
    'Vehicle Year': 'str'  # Usamos 'str' para tratar columnas de
manera uniforme y no se detecta otro valor
}

# Ajusta la ruta al archivo CSV según sea necesario
crash_df = pd.read_csv('Crash_Reporting_-_Drivers_Data.csv',
dtype=dtype_dict, low_memory=False)
```

Aquí solo importamos la librerías que vamos a ocupar durante la manipulación del dataset y sus diferentes apartados para cambiar columnas y darnos a entender que no se detecten de otra manera

2. Exploración Inicial de Datos

```
print("           ¿Qué es lo que más afecta en los choques?
") #Pregunta general de mi caso

print("1.- Imprimimos todos los primeros 10 valores")
print(crash_df.head(10))
print("\n")
print("Para seguir pulsa enter")
input()

print("2.- Imprimimos todos los últimos 10 valores")
print(crash_df.tail(10))
print("\n")
print("Para seguir pulsa enter")
input()
```

```
print("3.- Imprimimos el tamaño de nuestro dataset")
print(crash_df.shape)
print("\n")
print("Para seguir pulsa enter")
input()
```

```
1.- Imprimimos todos los primeros 10 valores
  report number local case number                agency name       acrs report type      crash date/time        route type  ...   latitude     longitude                  location        date      crash_datetime hour
0   MCP3170003V          240000438   Montgomery County Police  Property Damage Crash  2024-01-03 14:55:00               NaN  ...  39.165005    -77.24931     (39.16500483, -77.24931)  2024-01-03  2024-01-03 14:55:00   14
1   MCP3254003K          230072050   Montgomery County Police           Injury Crash  2023-12-16 12:36:00  Maryland (State)  ...  39.178776  -77.26718974  (39.17877577, -77.26718974)  2023-12-16  2023-12-16 12:36:00   12
2    EJ7887003Q          230074270  Gaithersburg Police Depar           Injury Crash  2023-12-29 12:00:00  Maryland (State)  ...  39.123574    -77.231769    (39.12357374, -77.231769)  2023-12-29  2023-12-29 12:00:00   12
3   MCP2674004J          230064598   Montgomery County Police  Property Damage Crash  2023-11-05 21:07:00  Maryland (State)  ...  39.211742  -77.17146065  (39.21174219, -77.17146065)  2023-11-05  2023-11-05 21:07:00   21
4   MCP2528000B          230067019   Montgomery County Police  Property Damage Crash  2023-11-18 00:40:00  Maryland (State)  ...  39.228915  -77.28909117  (39.22891483, -77.28909117)  2023-11-18  2023-11-18 00:40:00    0
5    EJ7887003Q          230074270  Gaithersburg Police Depar           Injury Crash  2023-12-29 12:00:00  Maryland (State)  ...  39.123574    -77.231769    (39.12357374, -77.231769)  2023-12-29  2023-12-29 12:00:00   12
6   MCP3039006P          230072243   Montgomery County Police           Injury Crash  2023-12-17 17:43:00  Maryland (State)  ...  38.995899  -77.02818753  (38.99589921, -77.02818753)  2023-12-17  2023-12-17 17:43:00   17
7   MCP2239002M          230065616   Montgomery County Police  Property Damage Crash  2023-11-10 14:04:00               NaN  ...  39.063839  -77.16083517    (39.063839, -77.16083517)  2023-11-10  2023-11-10 14:04:00   14
8   MCP2667008Z          230073137   Montgomery County Police           Injury Crash  2023-12-22 13:00:00  Maryland (State)  ...  39.159079  -77.16054149  (39.15907918, -77.16054149)  2023-12-22  2023-12-22 13:00:00   13
9   MCP3170003X          240001395   Montgomery County Police  Property Damage Crash  2023-12-12 05:00:00               NaN  ...  39.180337   -77.2730184   (39.18033651, -77.2730184)  2023-12-12  2023-12-12 05:00:00    5

[10 rows x 46 columns]
```

```
2.- Imprimimos todos los últimos 10 valores
       report number local case number                agency name       acrs report type      crash date/time        route type  ...   latitude     longitude                  location        date      crash_datetime hour
172095   MCP3019005J          220033189   Montgomery County Police  Property Damage Crash  2022-08-02 17:30:00  Maryland (State)  ...  39.031183  -77.12605333  (39.03118333, -77.12605333)  2022-08-02  2022-08-02 17:30:00   17
172096   MCP2656001Q          170506656   Montgomery County Police           Injury Crash  2017-05-08 17:12:00        US (State)  ...  39.090168    -76.94216    (39.09016833, -76.94216)  2017-05-08  2017-05-08 17:12:00   17
172097    DD5583005P          190036993  Rockville Police Departme  Property Damage Crash  2019-08-03 10:16:00               NaN  ...  39.091113  -77.17802483  (39.091113, -77.17802483)  2019-08-03  2019-08-03 10:16:00   10
172098   MCP3258002R          220047494   Montgomery County Police  Property Damage Crash  2022-10-29 06:01:00  Maryland (State)  ...  39.041368  -77.05192333  (39.04136833, -77.05192333)  2022-10-29  2022-10-29 06:01:00    6
172099    DD5623001G          180014691  Rockville Police Departme  Property Damage Crash  2018-03-25 14:10:00       Municipality  ...  39.077970    -77.122989     (39.07797, -77.122989)  2018-03-25  2018-03-25 14:10:00   14
172100    DM8338000C            16010228  Takoma Park Police Depart  Property Damage Crash  2016-03-01 10:01:00       Municipality  ...  38.972560  -76.99746689  (38.97255976, -76.99746689)  2016-03-01  2016-03-01 10:01:00   10
172101   MCP1182001S          170519976   Montgomery County Police  Property Damage Crash  2017-07-19 14:22:00             County  ...  39.004640   -77.10850167    (39.00464, -77.10850167)  2017-07-19  2017-07-19 14:22:00   14
172102   MCP1453008X          200046217   Montgomery County Police  Property Damage Crash  2020-11-23 07:37:00  Maryland (State)  ...  39.228963   -77.2367567  (39.22896333, -77.2367567)  2020-11-23  2020-11-23 07:37:00    7
172103   MCP2568000M          190056701   Montgomery County Police  Property Damage Crash  2019-11-23 23:23:00             County  ...  39.120440  -77.18004738  (39.12043995, -77.18004738)  2019-11-23  2019-11-23 23:23:00   23
172104   MCP1048000P            15003337   Montgomery County Police  Property Damage Crash  2015-01-21 09:02:00  Maryland (State)  ...  39.106847     -77.15838     (39.10684667, -77.15838)  2015-01-21  2015-01-21 09:02:00    9

[10 rows x 46 columns]
```

```
3.- Imprimimos el tamaño de nuestro dataset
(172105, 46)
```

Aquí en este paso mostramos las primeras 10 y las últimas diez líneas para que entiendan de que vamos a observar, tambien el tama;o de nuestro data set

3. Limpieza de Datos

```
def limpiar_datos(df):
    marcas_raras = ['N/A', 'UNKNOWN', 'NaN']
    modelos_raros = ['N/A', 'UNKNOWN', 'NaN']
    # Eliminar filas con marcas de vehículos raros
    df_clean = df[~df['vehicle make'].isin(marcas_raras)]
    # Convertir la columna 'vehicle year' a tipo numérico
    df_clean['vehicle year'] = pd.to_numeric(df_clean['vehicle year'],
errors='coerce')
    # Eliminar filas con modelos de vehículos raros
    df_clean = df_clean[~df_clean['vehicle model'].isin(modelos_raros)]
    # Eliminar filas con valores NaN en la columna 'vehicle year'
    df_clean = df_clean.dropna(subset=['vehicle year'])
    # Filtrar vehículos por año entre 1980 y 2024
    df_clean = df_clean[df_clean['vehicle year'].between(1980, 2024)]
    return df_clean

# Aplicar limpieza de datos
crash_clean_df = limpiar_datos(crash_df)
```

Limpiamos los datos para quitar los valores nulos con la función dropna y con el .isin para explicar si esta contenido con ese valor en el data frame Para comprobar si values es no en el DataFrame, use el ~ operador

4. Exploración Detallada de Datos

```
print("4.- Imprimimos todas las columnas")
print(crash_df.columns.sort_values())
print("\n")
print("Para seguir pulsa enter")
input()

print("5.- Imprimimos las filas")
print(crash_df.index.values)
print("\n")
print("Para seguir pulsa enter")
input()

print("6.- Imprimimos de la columna route type ordenamos")
conteo_tipos_ruta = crash_df['route
type'].value_counts().sort_values(ascending=False)

print(conteo_tipos_ruta)
print("\n")
print("Para seguir pulsa enter")
input()

print("7.- Imprimimos de la columna road name ordenamos")
conteo_carreteras = crash_df['road
name'].value_counts().sort_values(ascending=False)

print(conteo_carreteras)
print("\n")
print("Para seguir pulsa enter")
input()

print("8.- Imprimimos una función que saca los valores más recurrentes
de la columna que queremos")
def contar_valores():
    """
```

```
    Esta función permite al usuario contar los valores únicos de una
columna específica en el DataFrame crash_df.
    """
    while True:
        print("Algunos ejemplos: route type, road name, collision type,
speed limit, vehicle make")
        columna = str(input("Dame la columna (o escribe 'salir' para
finalizar): ")).lower()

        if columna == 'salir':
            break

        if columna not in crash_df.columns:
            print("¡La columna ingresada no existe en el DataFrame!")
            continue

        valores =
crash_df[columna].value_counts().sort_values(ascending=False)
        print(valores)

contar_valores()
print("\n")
print("Para seguir pulsa enter")
input()
```

```
4.- Imprimimos todas las columnas
Index(['acrs report type', 'agency name', 'circumstance', 'collision type',
       'crash date/time', 'crash_datetime', 'cross-street name',
       'cross-street type', 'date', 'driver at fault', 'driver distracted by',
       'driver substance abuse', 'driverless vehicle', 'drivers license state',
       'equipment problems', 'hour', 'injury severity', 'latitude', 'light',
       'local case number', 'location', 'longitude', 'municipality',
       'non-motorist substance abuse', 'off-road description',
       'parked vehicle', 'person id', 'related non-motorist', 'report number',
       'road name', 'route type', 'speed limit', 'surface condition',
       'traffic control', 'vehicle body type', 'vehicle continuing dir',
       'vehicle damage extent', 'vehicle first impact location',
       'vehicle going dir', 'vehicle id', 'vehicle make', 'vehicle model',
       'vehicle movement', 'vehicle second impact location', 'vehicle year',
       'weather'],
      dtype='object')
```

```
5.- Imprimimos las filas
[     0      1      2 ... 172102 172103 172104]
```

```
6.- Imprimimos de la columna route type ordenamos
route type
Maryland (State)        77074
County                  55568
Municipality             9379
US (State)               7567
Interstate (State)       3149
Other Public Roadway     1129
Government                627
Ramp                      579
Service Road               40
Unknown                    20
Name: count, dtype: int64
```

```
7.- Imprimimos de la columna road name ordenamos
road name
GEORGIA AVE          10861
NEW HAMPSHIRE AVE     6832
FREDERICK RD          5768
ROCKVILLE PIKE        4839
CONNECTICUT AVE       4093
                     ...
HAWTHORN PL              1
LAWNSBERRY PL            1
PLAYFORD LA              1
GODWIT STREET            1
HARVEST SCENE CT         1
Name: count, Length: 3803, dtype: int64
```

```
Algunos ejemplos: route type, road name, collision type, speed limit, vehicle make
Dame la columna (o escribe 'salir' para finalizar): speed limit
speed limit
35    50479
40    33589
25    24045
30    23377
45    12454
15     6125
0      4748
50     4669
```

Aquí imprimimos valores comunes en el dataset de todas las columnas para visualizar qué podemos hacer en el dataset, ahora después de limpiar lineas, osea no quito ninguna porque no hay valores desconocidos en las tablas que limpiamos, pero esto no quiere decir que no existan valores nulos.

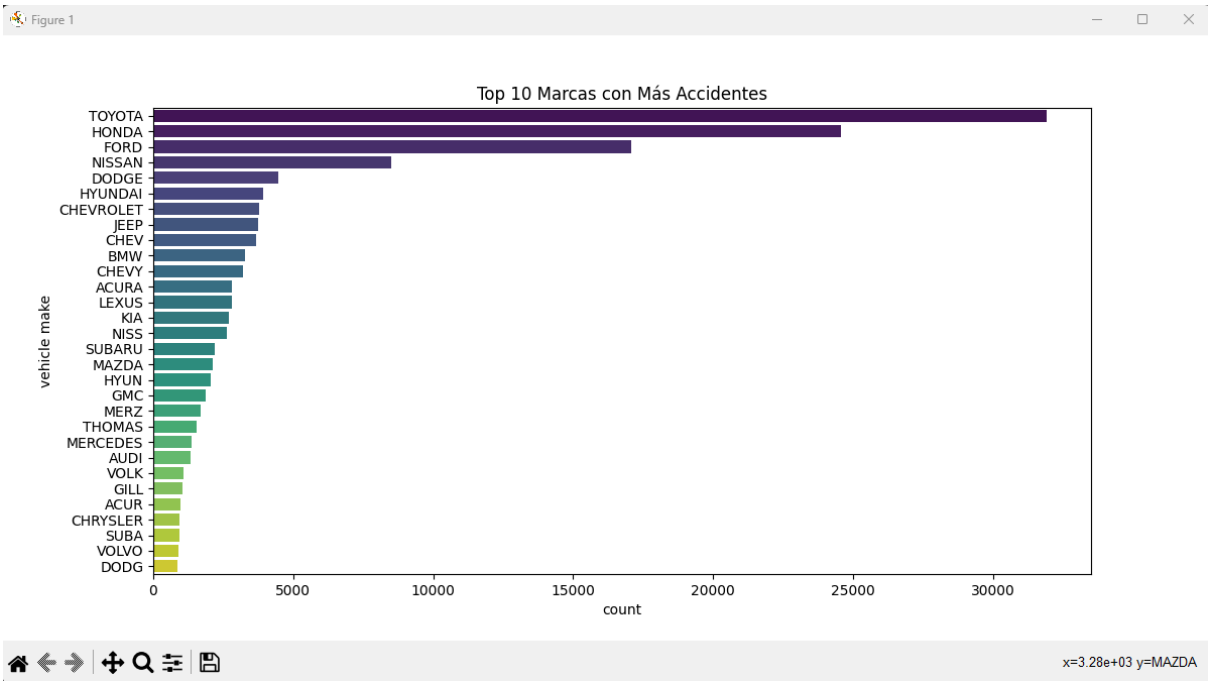
Mostramos en que estado y en que ruta chocan más frecuentemente las personas de este Dataset, aparte que la velocidad es la que comúnmente chocan que es a 35 km. Esto quiere decir que no necesariamente ocupas ir a velocidad fuerte para chocar.

5. Análisis Exploratorio y Visualización

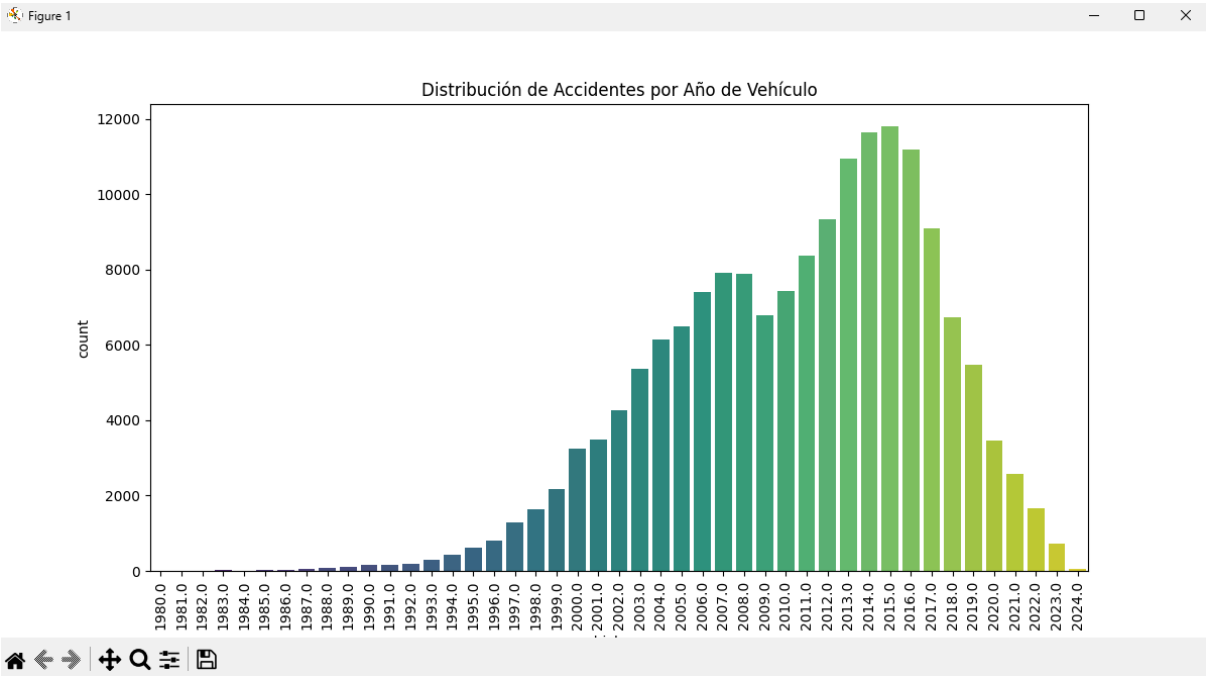
```
def analisis_exploratorio(df):

    # Análisis de las marcas más comunes en accidentes
    top_marcas = df['vehicle make'].value_counts().head(30).index
    df_top_marcas = df[df['vehicle make'].isin(top_marcas)]

    plt.figure(figsize=(12, 6))
    sns.countplot(data=df_top_marcas, y='vehicle make',
order=top_marcas, palette='viridis')
    plt.title('Top 10 Marcas con Más Accidentes')
    plt.show()

    # Distribución de accidentes por año del vehículo
    plt.figure(figsize=(12, 6))
    sns.countplot(data=df, x='vehicle year', palette='viridis')
    plt.title('Distribución de Accidentes por Año de Vehículo')
    plt.xticks(rotation=90)
    plt.show()

    # Distribución de accidentes por condiciones climáticas
    plt.figure(figsize=(10, 6))
    sns.countplot(y='weather', data=crash_df, palette='viridis')
    plt.xlabel('Frecuencia')
    plt.ylabel('Condiciones Climáticas')
    plt.title('Distribución de Accidentes por Condiciones Climáticas')
    plt.tight_layout()
    plt.show()

    # Distribución de accidentes a lo largo del tiempo
    plt.figure(figsize=(12, 6))
    crash_df['crash date/time'] = pd.to_datetime(crash_df['crash
date/time'])
    crash_df['date'] = crash_df['crash date/time'].dt.date
    crash_df['date'].value_counts().sort_index().plot()
    plt.title('Distribución de Accidentes a lo largo del Tiempo')
    plt.xlabel('Fecha')
    plt.ylabel('Número de Accidentes')
    plt.grid(True)
```

```
    plt.show()

    # Análisis de frecuencia: Frecuencia de diferentes tipos de
colisión
    plt.figure(figsize=(10, 6))
    sns.countplot(data=crash_df, x='collision type', palette='viridis',
order=crash_df['collision type'].value_counts().index)
    plt.title('Frecuencia de Diferentes Tipos de Colisión')
    plt.xlabel('Tipo de Colisión')
    plt.ylabel('Número de Accidentes')
    plt.xticks(rotation=45, ha='right')
    plt.tight_layout()
    plt.show()

    crash_df['crash_datetime'] = pd.to_datetime(crash_df['crash
date/time'])

    # Extraer la hora del día de la columna de fecha y hora
    crash_df['hour'] = crash_df['crash_datetime'].dt.hour

    # Gráfico de distribución de accidentes por hora del día
    plt.figure(figsize=(10, 6))
    sns.histplot(data=crash_df, x='hour', bins=24, kde=True,
color='skyblue')
    plt.xlabel('Hora del Día')
    plt.ylabel('Frecuencia de Accidentes')
    plt.title('Distribución de Accidentes por Hora del Día')
    plt.xticks(range(0, 24))
    plt.grid(axis='y')
    plt.tight_layout()
    plt.show()

    while True:
        # Distribución de accidentes por modelo de una marca específica
        marca = str(input("Dame la marca (o escribe 'salir' para
finalizar): ")).upper().strip()

        if marca == 'SALIR':
            break

        df_marca = df[df['vehicle make'].str.upper().str.strip() ==
marca]
```

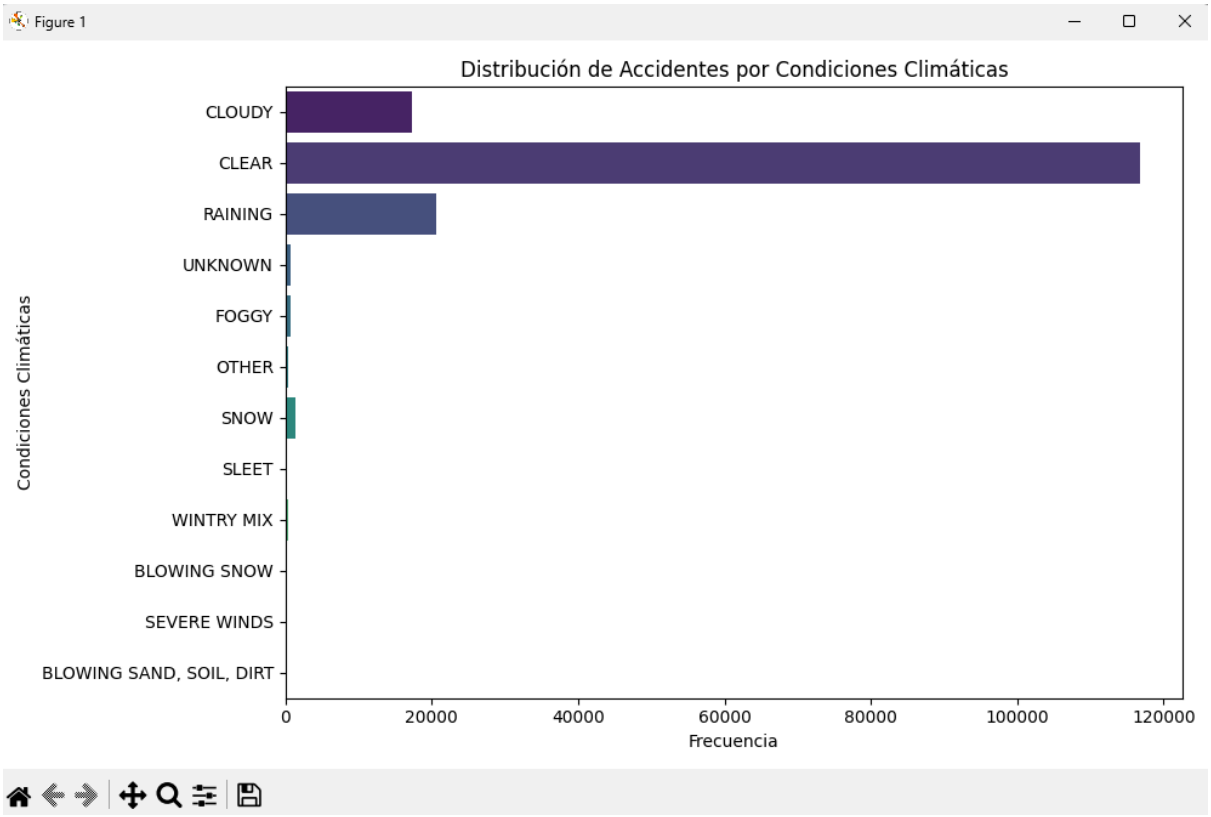
```
        if not df_marca.empty:
            top_models = df_marca['vehicle
model'].value_counts().head(10).index
            df_top_models = df_marca[df_marca['vehicle
model'].isin(top_models)]

            plt.figure(figsize=(12, 6))
            sns.countplot(data=df_top_models, y='vehicle model',
order=top_models, palette='viridis')
            plt.title(f'Top 10 Modelos de {marca} con Más Accidentes')
            plt.show()
        else:
            print(f"No se encontraron datos para la marca {marca}")

# Ejecutar análisis exploratorio
analisis_exploratorio(crash_clean_df)
```



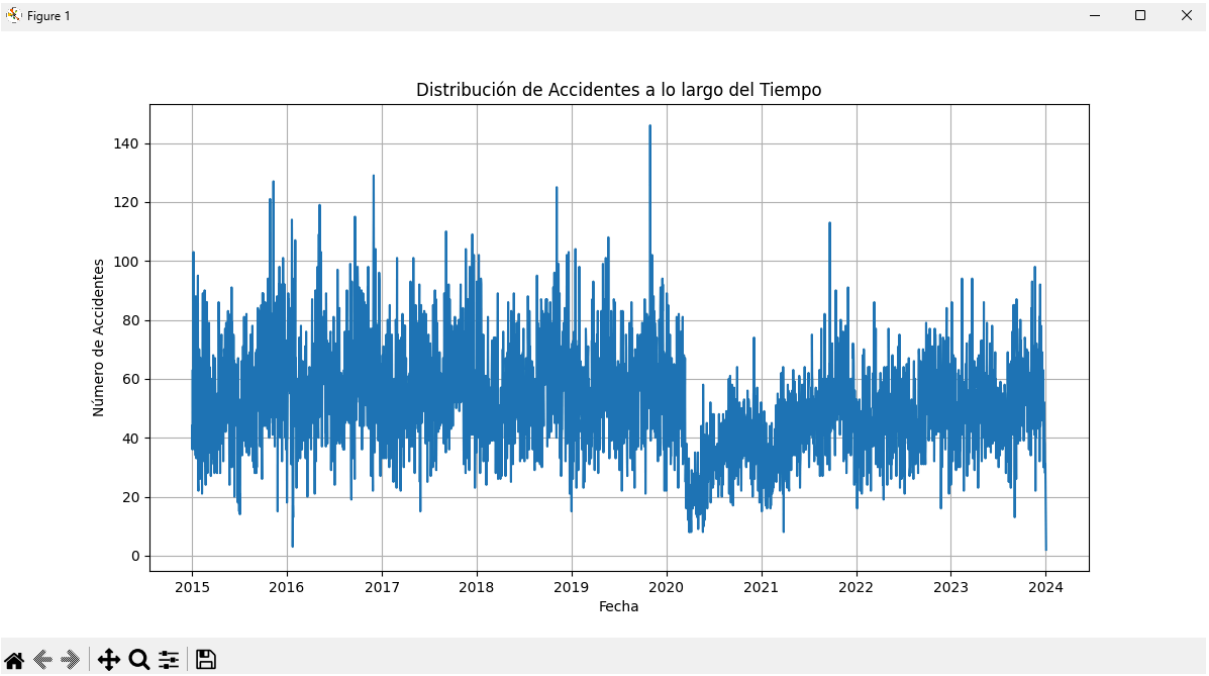
En esta gráfica nos muestra los tipo de marcas con más accidentes lo que podemos concluir que son de lo más usados o vienen defectuosos los toyotas.



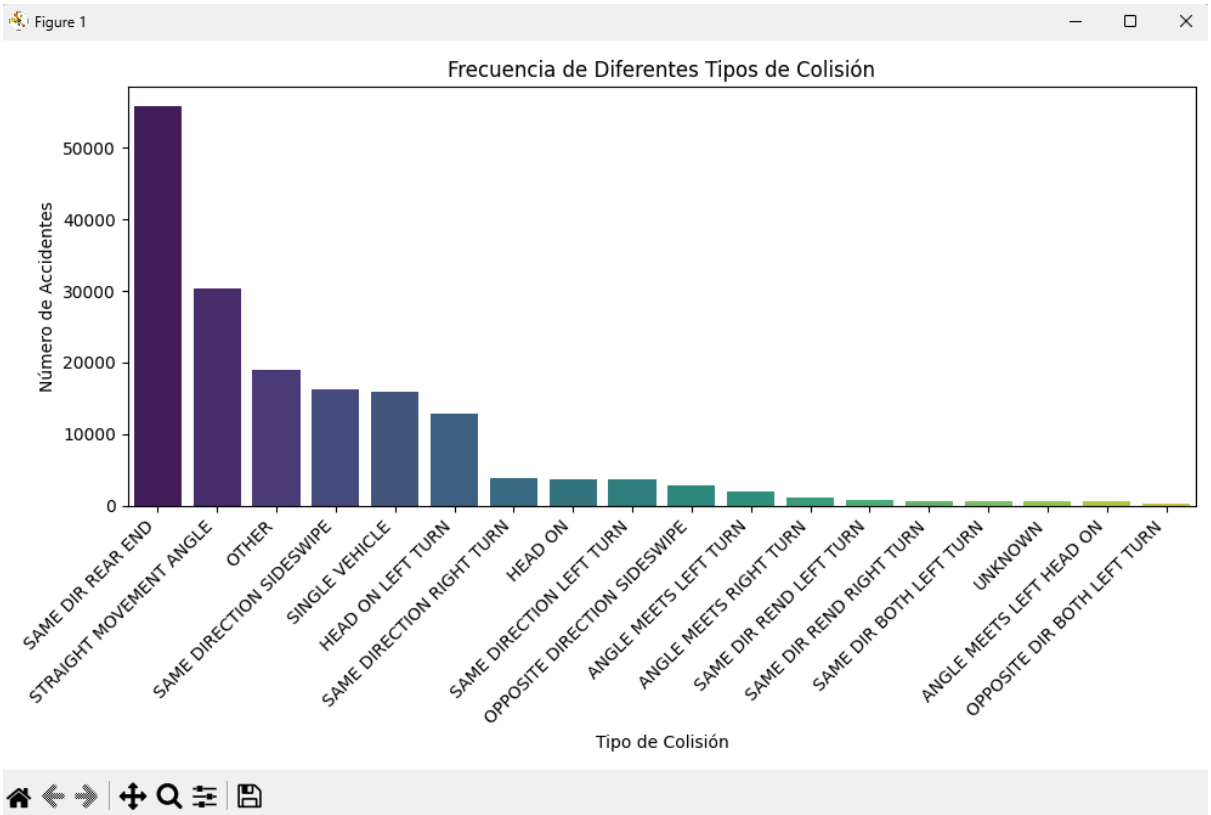
Aquí mostramos una gráfica que donde en general nos muestra los carros distribuidos por sus año de creación donde vemos que son los del 2012 donde hubo más choques y en los años de pandemia 2021 se muestra que son muy poquitos



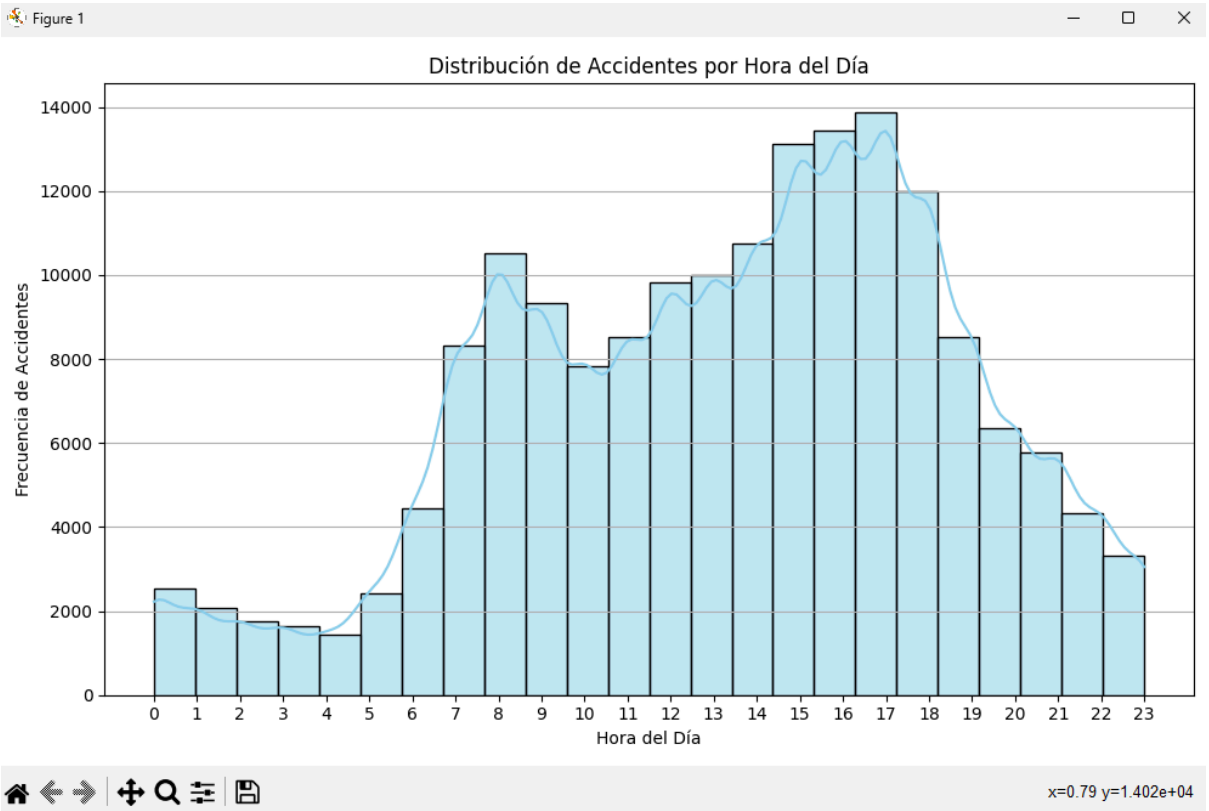
Aquí mostramos una gráfica donde nos muestra los diferentes tipos de clima donde predomina en seco osea que nos están factor la lluvia en los choques en estados unidos.



En esta gráfica podemos predecir los choques a lo largo del tiempo como en pandemia bajo demasiado y poco a poco va subiendo



En esta gráfica nos muestra solo como son las colisiones de los carros entre choques y predomina los choques traseros

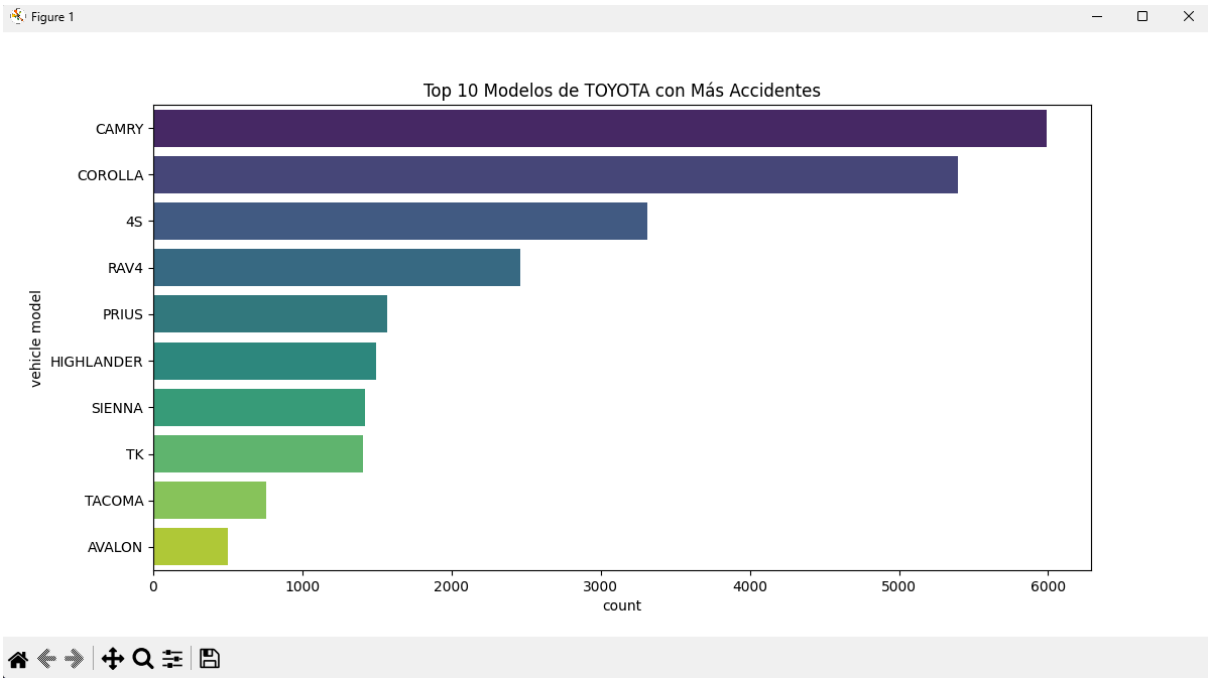


En esta gráfica vemos los choques a lo largo del tiempo mirando que al salir y al entrar al trabajo vemos que hay más choques

6. Interacción con el Usuario

```
def mostrar_modelos_por_marca(df):
    marca = input("Por favor ingresa la marca del vehículo:
").strip().upper()
    df_marca = df[df['vehicle make'] == marca]

    if not df_marca.empty:
        modelos_unicos = df_marca['vehicle model'].unique()
        print(f"Modelos de {marca}:")
        for modelo in modelos_unicos:
            print(modelo)
        return marca  # Devuelve la marca ingresada por el usuario
    else:
        print(f"No se encontraron modelos para la marca {marca}")
        return None

def mostrar_años_por_modelo(df, marca):
```

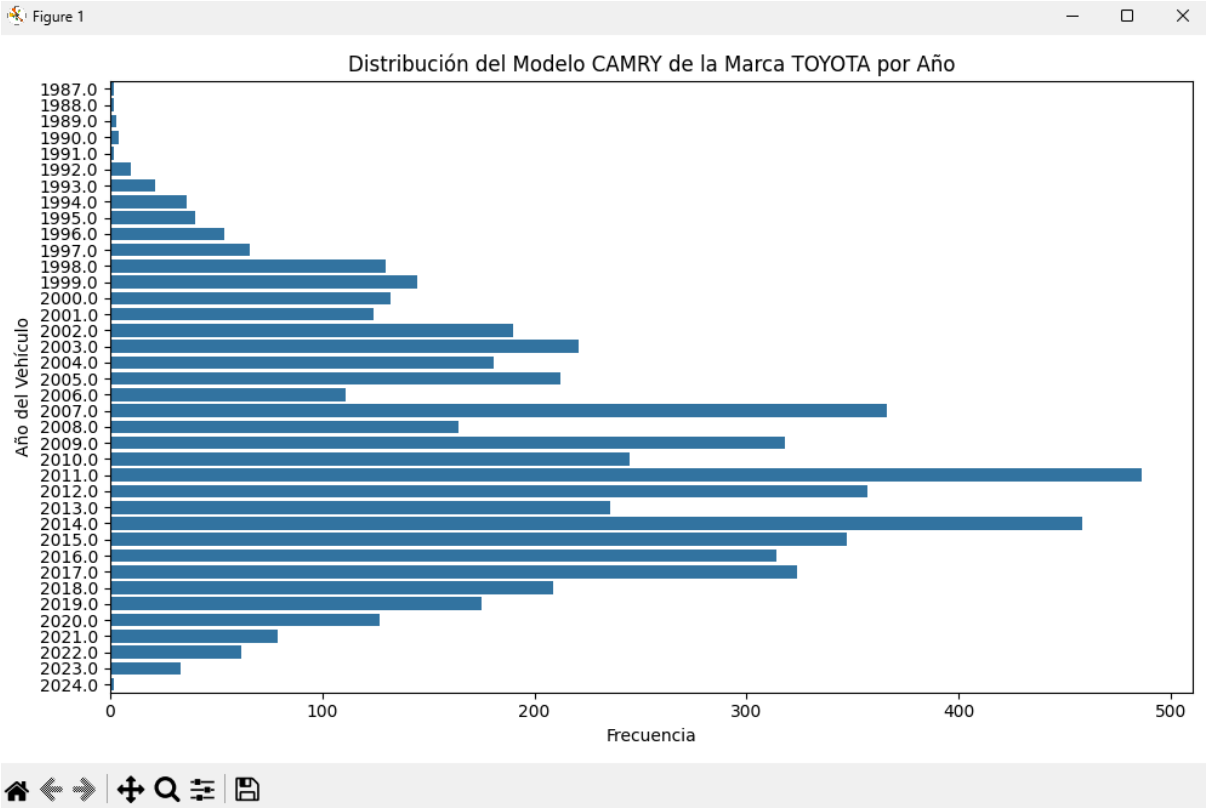
```
    modelo = input(f"Por favor ingresa el modelo del vehículo de la
marca {marca}: ").strip().upper()
    df_modelo = df[(df['vehicle make'] == marca) & (df['vehicle model']
== modelo)]

    if not df_modelo.empty:
        # Crear un gráfico de barras horizontal
        plt.figure(figsize=(10, 6))
        sns.countplot(y='vehicle year', data=df_modelo)
        plt.xlabel('Frecuencia')
        plt.ylabel('Año del Vehículo')
        plt.title(f'Distribución del Modelo {modelo} de la Marca
{marca} por Año')
        plt.tight_layout()
        plt.show()
    else:
        print(f"No se encontraron datos para el modelo {modelo} de la
marca {marca}")

# Ejecutar la función para mostrar modelos por marca y guardar la marca
marca_seleccionada = mostrar_modelos_por_marca(crash_clean_df)

# Ejecutar la función para mostrar años por modelo y pasarle la marca
seleccionada
if marca_seleccionada:
    mostrar_años_por_modelo(crash_clean_df, marca_seleccionada)
```



En esta gráfica nos muestra de la marca que te pregunta te muestra cual es el carro cual es que más choca de esa marca

```
Dame la marca (o escribe 'salir' para finalizar): salir
Por favor ingresa la marca del vehículo: toyota
Modelos de TOYOTA:
PRIUS
COROLLA
CP
RAV4
TK
CAMRY
4S
RAV 4
SD
UT
HIGHLANDER
SIENNA
TACOMA
CARROLA
TUNDRA
SUV
```

Aquí preguntamos qué marca y muestra todas las marca que existen en esa marca de todo el dataset y vemos que existen errores por lo que le falta una manipulación y limpieza de datos



Aqui despues de ver todas las marcas y te interesa alguna te muestra por años de ese carro los choques que tiene

7. Corrección de Datos

```
def corregir_marca(df):
    seguir = str(input("¿Quieres modificar algo?: "))
    if seguir.lower() == 'si':
        columna = input("Por favor ingresa el nombre de la columna que
deseas modificar: ").strip().lower()
        valor_buscar = input(f"Por favor ingresa el valor que deseas
buscar en la columna '{columna}': ").strip().upper()
        nuevo_valor = input("Por favor ingresa el nuevo valor:
").strip().upper()

        pregunta = input(f"¿Estás seguro de que deseas cambiar
'{valor_buscar}' a '{nuevo_valor}' en la columna '{columna}'? (s/n):
").strip().lower()
        if pregunta == 's':
            df[columna] = df[columna].str.replace(valor_buscar,
nuevo_valor, case=False)
            df.to_csv('Crash_Reporting_-_Drivers_Data.csv',
index=False)
            print("Se han guardado los cambios en el archivo CSV.")
        elif pregunta == 'n':
            print("No se han realizado cambios en el archivo CSV.")
        else:
            print("Respuesta no válida. No se han realizado cambios en
el archivo CSV.")
    else:
        return

# Llama a la función para corregir la marca en tu DataFrame y guardar
los cambios en el archivo CSV si el usuario lo desea
corregir_marca(crash_df)
```

En esta te pregunta una columna a modificar y solo la reemplaza por lo que tu quieras ejemplo todas las toyotas le puedes poner hola y se cambia