Global Terrorism Analysis - Documentación (segunda entrega)

Martín García Chagüezá

David Melo Valbuena

Juan Andrés Ruiz Muñoz

Sobre el proyecto

En este caso decidimos utilizar un conjunto de datos que incluye información sobre atentados terroristas en todo el mundo desde 1970 hasta 2017. El conjunto de datos utilizado (https://www.kaggle.com/datasets/START-UMD/gtd/data) tiene 181.691 filas y 135 columnas: lo sometimos a procesos de carga, limpieza y transformación para encontrar *insights* interesantes sobre los diferentes atentados terroristas allí registrados.

Las herramientas usadas son:

- *Python 3.10* ➜ Página de descarga
- *Jupyter Notebook* ➜ Herramienta de VS Code para notebooks
- *PostgreSQL* ➜ Página de descarga
- *Power BI Desktop* ➜ Página de descarga

Las dependencias necesarias para Python son:

- Apache Airflow
- Dotenv
- Pandas
- Matplotlib
- Seaborn
- SQLAlchemy

⚠ Apache Airflow sólo funciona correctamente en **entornos Linux**. Si tiene Windows, le recomendamos que utilice una máquina virtual o WSL.

Estas dependencias se incluyen en el archivo `requirements.txt` del proyecto Python. La instalación paso a paso se describe en el archivo README.

Objetivos del proyecto

Obtener un *dataset* limpio para la creación de un informe analítico o *dashboard* utilizando herramientas de BI como Power BI, que se conectará a través de una base de datos PostgreSQL que contendrá el conjunto de datos transformados.

El *dashboard* contendrá gráficas sobre:

- Evolución de atentados por grupos terroristas por año (*gráfico de barras horizontales*).
- Atentados terroristas por grupos armados (*gráfico de torta*).

- Número de atentados terroristas por país (*mapa coroplético*).

Información sobre el dataset

Tras un riguroso proceso de limpieza y transformación, nuestro conjunto de datos presenta las siguientes columnas dispuestas para el análisis y la visualización de sus datos:

- **eventid**: ID del incidente en la GTD. Es un número de 12 dígitos que indica la fecha del incidente (8 primeros dígitos) y un número de caso secuencial (4 últimos dígitos).
- **iyear**: Año del incidente.
- **imonth**: Mes del incidente.
- **iday**: Día del incidente.
- **extended**: Indica si la duración de un incidente se extendió más de 24 horas. Valores: 1 para «Sí», 0 para «No».
- ***country_txt***: Nombre del país donde se produjo el incidente.
- **country**: Código del país donde se produjo el incidente.
- ***region_txt***: Nombre de la región donde se produjo el incidente.
- **region**: Código de la región donde se produjo el incidente.
- **city**: Ciudad donde se produjo el incidente.
- **latitude**: Latitud geográfica del incidente.
- **longitude**: Longitud geográfica del incidente.
- **vicinity**: Indica si el incidente se produjo en las proximidades de una ciudad o localidad. Valores: 1 para «Sí, cerca», 0 para «No, en el lugar exacto», 9 para «Desconocido».
- **crit1, crit2, crit3**: Criterios de inclusión en la base de datos, evaluando si el incidente cumple determinadas normas de terrorismo. Valores: 1 para «Sí», 0 para «No».
- **doubtterr**: Indica si existen dudas sobre si el incidente es un acto de terrorismo. Valores: 1 para «Sí», 0 para «No (Esencialmente, no hay dudas de que sea un acto terrorista)».
- **multiple**: Indica si el incidente forma parte de varios incidentes relacionados. Valores: 1 para «Sí», 0 para «No».
- **success**: Indica si el atentado terrorista fue un éxito. Valores: 1 para «Sí», 0 para «No».
- **suicide**: Indica si el atentado fue suicida. Valores: 1 para «Sí», 0 para «No».
- ***attacktype1_txt***: Descripción textual del tipo de atentado.
- **attacktype1**: Código del tipo de ataque.
- ***targtype1_txt***: Descripción textual del tipo de objetivo.
- **targtype1**: Código del tipo de objetivo.
- ***natlty1_txt***: Descripción textual de la nacionalidad del objetivo.
- **natlty1**: Código de la nacionalidad del objetivo.
- **gname**: Nombre del grupo que perpetró el atentado.

- **guncertain1**: Indica si el grupo autor del atentado es sospechoso o no confirmado. Valores: 1 para «Sí», 0 para «No».
- **individual**: Indica si los autores fueron individuos no afiliados a un grupo. Valores: 1 para «Sí», 0 para «No».
- **nperps**: Número de perpetradores.
- **nperpcap**: Número de autores capturados.
- **claimed**: Indica si alguien reivindicó la autoría del atentado. Valores: 1 para «Sí», 0 para «No».
- ***weaptype1_txt***: Descripción textual del tipo de arma.
- **weaptype1**: Código del tipo de arma.
- **nkill**: Número de personas asesinadas.
- **nwound:** Número de heridos por atentado terrorista
- **property**: Indica si hubo daños materiales. Valores: 1 para «Sí», 0 para «No», 9 para «Desconocido».
- **ishostkid**: Indica si hubo secuestro de rehenes. Valores: 1 para «Sí», 0 para «No», 9 para «Desconocido».
- **INT_ANY**: Indica la participación internacional. Valores: 1 para «Sí», 0 para «No», 9 para «Desconocido».

Información sobre la API

https://acleddata.com/

Para esta entrega del proyecto, además, debemos utilizar una API que debe ser fusionada con el dataset principal mediante un ***merge***. Después de pasar dichos datos por procesos de limpieza y transformación, nuestro conjunto de datos presenta las siguientes columnas dispuestas para el análisis y la visualización de sus datos:

Column Name	Type	Description
event_id_cnty	string	A unique alphanumeric event identifier by number and country acronym. This identifier remains constant even when the event details are updated.
event_date	date	The date on which the event took place. Recorded as Year-Month-Day.
year	int	The year in which the event took place.
time_precision	int	A numeric code between 1 and 3 indicating the level of precision of the date recorded for the event. The higher the number, the lower the precision.
disorder_type	string	The disorder category an event belongs to.
event_type	string	The type of event; further specifies the nature of the event.
sub_event_type	string	A subcategory of the event type.
actor1	string	One of two main actors involved in the event (does not necessarily indicate the aggressor).
assoc_actor_1	string	Actor(s) involved in the event alongside 'Actor 1' or actor designations that further identify 'Actor 1'
inter1	int	A numeric code between 0 and 8 indicating the type of 'Actor 1'.
actor2	string	One of two main actors involved in the event (does not necessarily indicate the target or victim).

assoc_actor_2	string	Actor(s) involved in the event alongside ‘Actor 2’ or actor designation further identifying ‘Actor 2’.
inter2	int	A numeric code between 0 to 8 indicating the type of ‘Actor 2’.
interaction	int	A two-digit numeric code (combination of ‘Inter 1’ and ‘Inter 2’) indicating the two actor types interacting in the event.
civilian_targeting	String	This column indicates whether the event involved civilian targeting.
iso	int	A unique three-digit numeric code assigned to each country or territory according to ISO 3166.
region	string	The region of the world where the event took place.
country	string	The country or territory in which the event took place.
admin1	string	The largest sub-national administrative region in which the event took place.
admin2	string	The second largest sub-national administrative region in which the event took place.
admin3	string	The third largest sub-national administrative region in which the event took place.
location	string	The name of the location at which the event took place.
latitude	decimal	The latitude of the location in four decimal degrees notation (EPSG:3857).
longitude	decimal	The longitude of the location in four decimal degrees notation (EPSG:3857).
geo_precision	int	A numeric code between 1 and 3 indicating the level of certainty of the location recorded for the event. The higher the number, the lower the precision.
source	string	The sources used to record the event. Separated by a semicolon.
source_scale	string	An indication of the geographic closeness of the used sources to the event
notes	string	A short description of the event.
fatalities	int	The number of reported fatalities arising from an event. When there are conflicting reports, the most conservative estimate is recorded
tags	string	Additional structured information about the event. Separated by a semicolon.
timestamp	int/date	An automatically generated Unix timestamp that represents the exact date and time an event was last uploaded to the ACLED API.
population_1km	int	Estimated population in a 1km radius (only returned if population=full)
population_2km	int	Estimated population in a 2km radius (only returned if population=full)
population_5km	int	Estimated population in a 5km radius (only returned if population=full)
population_best	int	Best estimate of affected population (only returned if population=full OR population=TRUE)

De lo anterior, se decidió quedarse con columnas que apoyaran al dataset original, como lo fue *disorder_type*, *population_1km*, *population_2km* y *population_5km*.

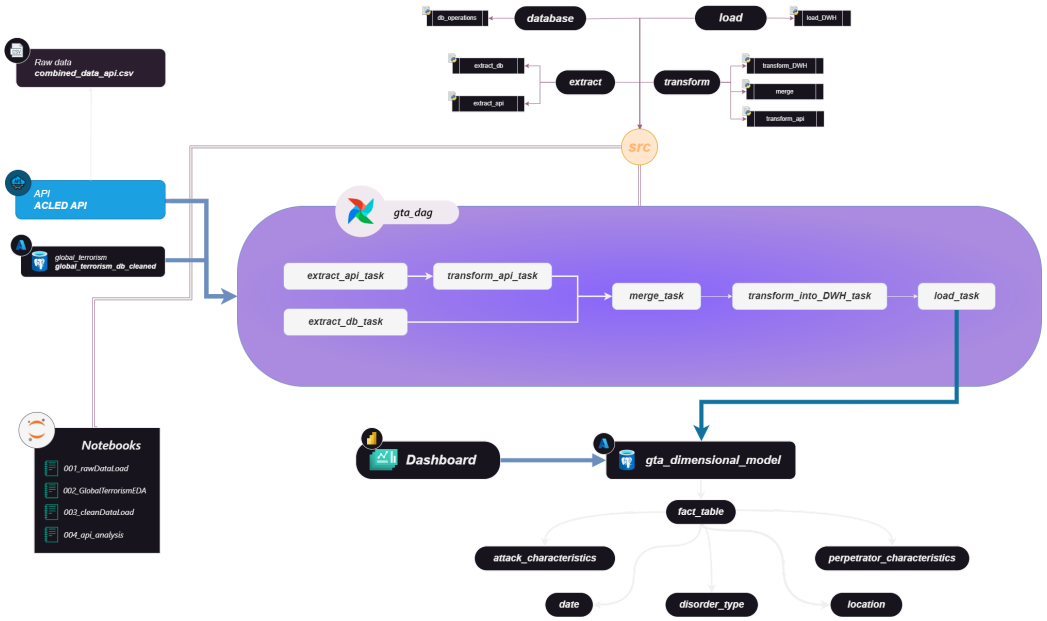
Sin embargo, de las anteriores, las *population* depende de un `population=full`, y de los datos que consumimos de la API, ninguno traía condición, por lo que se descartan todas las columnas population, sea cual sea sus tipos, y se mantiene *disorder_type*, qué es importante al analizar el tipo de conflicto y darnos algunos hechos y curiosidades sobre el ataque terrorista, como si es una estrategia política, manifestaciones, entre otras.

Recordemos que *disorder_type* tiene las siguientes características:

disorder_type
Political Violence
Political Violence; demonstrations
demonstrations

Strategic developments

Flujo de los datos



Proceso

Extracción de datos: base de datos

Utilizaremos la base de datos transformada proveniente del corte del anterior proyecto. Esta base de datos se encuentra alojada en una instancia de PostgreSQL en Azure.

	eventid [PK] bigint	iyear integer	imonth integer	iday integer	extended integer	country_txt character varying (500)	country integer	region_txt character varying (500)	region integer	city character varying (500)	latitude double precision	longitude double precision
1	197001010002	1970	1	1	0	United States	217	North America	1	Cairo	37.005105	-89.176269
2	197001020001	1970	1	2	0	Uruguay	218	South America	3	Montevideo	-34.891151	-56.187214
3	197001020003	1970	1	2	0	United States	217	North America	1	Madison	43.076592	-89.412488
4	197001030001	1970	1	3	0	United States	217	North America	1	Madison	43.07295	-89.386694
5	197001090001	1970	1	9	0	United States	217	North America	1	Detroit	42.331685	-83.047924
6	197001120001	1970	1	12	0	United States	217	North America	1	New York City	40.697132	-73.931351
7	197001120002	1970	1	12	0	United States	217	North America	1	Rio Grande	18.379998	-65.830948
8	197001130001	1970	1	13	0	United States	217	North America	1	Seattle	47.610786	-122.331306
9	197001140001	1970	1	14	0	United States	217	North America	1	Champaign	40.116748	-88.23927
10	197001150001	1970	1	15	0	Uruguay	218	South America	3	Montevideo	-34.891151	-56.187214
11	197001190002	1970	1	19	0	United States	217	North America	1	Seattle	47.610786	-122.331306
12	197001190003	1970	1	19	0	United States	217	North America	1	Seattle	47.610786	-122.331306
13	197001190004	1970	1	19	0	United States	217	North America	1	Jersey City	40.717892	-74.067467
14	197001200001	1970	1	20	0	Guatemala	83	Central America & Caribbean	2	Guatemala City	14.622869	-90.529068
15	197001220001	1970	1	22	0	Venezuela	222	South America	3	Caracas	10.482834	-66.962128
16	197001220002	1970	1	22	0	United States	217	North America	1	South Sioux City	42.47031	-96.413949
17	197001250002	1970	1	25	0	United States	217	North America	1	New York City	40.697132	-73.931351
18	197001260001	1970	1	26	0	United States	217	North America	1	West Point	33.60651	-88.650419
19	197001260003	1970	1	26	0	United States	217	North America	1	New York City	40.697132	-73.931351
20	197001270002	1970	1	27	0	United States	217	North America	1	Norwalk	41.241996	-82.615241
21	197001300001	1970	1	30	0	United States	217	North America	1	South Sioux City	42.479999	-96.413046

```
from database.db_operations import creating_engine, disposing_engine

import pandas as pd
import logging

logging.basicConfig(level=logging.INFO, format="%(asctime)s %(message)s", datefmt="%d/%

## ----- DB Extract ----- ##
```

```
def extracting_db_data():
    """
    Extracting data from the GTA table and return it as a DataFrame.

    """
    engine = creating_engine()

    try:
        logging.info("Starting to extract the data from the Global Terrorism table.")
        df = pd.read_sql_table("global_terrorism_db_cleaned", engine)
        logging.info("Data extracted from the Global Terrorism table.")

        return df
    except Exception as e:
        logging.error(f"Error extracting data from the Global Terrorism table: {e}.")
    finally:
        disposing_engine(engine)
```

Extracción de datos: API

https://apidocs.acleddata.com/acled_endpoint.html

La extracción de la API cuenta con métodos comunes, utilizando requests, y contiene la estructura:

```
import requests
import pandas as pd

url = 'https://api.acleddata.com/acled/read.json'
params = {
    'key': 'KEY',
    'email': 'EMAIL',
    'fields': 'event_date|country|disorder_type|actor1',
    'year': '1989|2017',
    'year_where': 'BETWEEN',
    'limit': 5000,
    'page': 1
}

all_records = []

while True:
    response = requests.get(url, params=params)
    data = response.json()

    if 'data' not in data or not data['data']:
```

```
        break

    all_records.extend(data['data'])
    params['page'] += 1

df = pd.DataFrame(all_records)
```

Por documentación de la API, debemos realizar un ciclo que nos permita pasar de páginas para obtener todos los posibles datos, en los que filtramos en los años objetivos de nuestro dataset original.

EDA: API

Importación de Librerías: Se importa la librería pandas para manipular y analizar datos.

Carga del Archivo CSV: Se carga un archivo CSV llamado globalterrorismdb_0718dist.csv que contiene datos sobre ataques terroristas. Se especifica la codificación como iso-8859-1 y se desactiva el uso de memoria baja.

Selección de Columnas: Se define una lista de columnas seleccionadas, llamada columns_choice, que incluye datos relevantes como: el año, país, ciudad, tipo de ataque, grupo responsable, número de víctimas, entre otros.

Asignación de Valores por Defecto: Se crea un diccionario defect_values para asignar valores predeterminados a las columnas con valores faltantes. Por ejemplo, el número de perpetradores capturados se asigna como 0, y para la columna claimed, se asigna el valor 999.

Relleno de Valores Faltantes: Se utiliza el método fillna() para rellenar los valores nulos en el DataFrame con los valores definidos en defect_values.

Filtrado de Datos: Se eliminan las filas donde la columna doubtterr tiene un valor de 1, indicando que el ataque está en duda de ser terrorista.

Fusión de Datos: Se combinan los datos del DataFrame df_terrorism con otro DataFrame (denominado api_terrorism) utilizando una fusión de tipo outer. Esta fusión asegura que todas las filas de ambos DataFrames se incluyan, y los valores faltantes se manejan como NaN.

Revisión de los Datos: Se revisan los datos resultantes mediante los comandos data.info() y data.head(5), que muestran un resumen del DataFrame y las primeras cinco filas, respectivamente.

Eliminación de Filas con Valores Nulos: Se eliminan todas las filas que contienen valores nulos utilizando dropna().

Verificación de los Datos: Se imprime la primera fila del DataFrame resultante después de haber eliminado los valores nulos, utilizando head(1) para asegurarse de que los datos se vean correctos.

CONCLUSIONES:

Selección de columnas clave: Se ha filtrado la información más relevante del archivo original, centrándose en atributos importantes como el año, país, tipo de ataque, y el número de víctimas. Esto permite reducir la complejidad del análisis y enfocarse en aspectos más significativos.

Relleno de valores faltantes: Se han imputado valores por defecto para algunos campos, lo cual es una práctica común para evitar que datos incompletos afecten los análisis futuros. Este paso mejora la calidad de los datos para su posterior uso, especialmente en columnas con datos numéricos.

Eliminación de ataques dudosos: Al eliminar las filas con dudas sobre si el ataque fue realmente terrorista, los datos se hacen más confiables para su análisis.

Fusión con datos adicionales: La combinación de este DataFrame con otro, mediante una fusión de tipo `outer`, ha permitido ampliar el conjunto de datos sin perder registros, aunque haya valores nulos en alguna de las

tablas.

Limpieza final: La eliminación de filas con valores nulos asegura que los datos finales sean más homogéneos, lo cual facilita su análisis.

Transformación de los datos: merge

Teniendo como referencia la API y nuestro dataset original, creamos una llave compuesta del dataset y la api creada con `date+country+actor`, con la que podemos ser precisos en el merge dado que el dataset y la api tienen estas columnas.

Lo anterior nos da como resultado un merge con más de 13 mil registros, que al manejar los datos nulos (como lo realizamos en la tabla *disorder_type*), nos da como resultado los mismos registros que el dataset original, manejando los nulos correctamente. El merge lo realizamos a través de un left join, persistiendo los datos del dataset original con el complemento de la api.

Transformación de los datos: base de datos - modelo dimensional

Teniendo en cuenta las columnas disponibles deseadas, ya previamente realizando un merge, empezamos a pensar en un modelo dimensional tipo estrella, debido a su alto performance.

- **La granularidad**, en nuestro caso, se refiere a **cada hecho por atentado**. Tomamos como unidad **cada atentado.** Ejemplo: "*Los heridos por **el atentado** a la primera de las Torres Gemelas fueron aproximadamente 2,500 personas.*"
- Separamos las *facts* y las *dimensions,* por lo que solamente nos enfocamos para las *facts* medidas en las que fueran faciles sacar conclusiones por si solas. Por lo que manejaramos las siguientes tablas:
  - *Fact* table

eventid	property	ishostkid	nwound	multiple	success	suicide

  - Tabla Location (Dimension)

id_location	region	country	city
country_x+region+city	region_txt	country_txt	city

  - Tabla Date (Dimension)

id_date	date
yyyymmdd	date (yyyy-mm-dd)

  - Tabla attackCharacteristics (Dimension)

id_attack	attack_type	target_type	nationa
attacktype1+targtype1+natlty1_txt+weaptype1+crit1+crit2+crit3+property	attacktype1_txt	targtype1_txt	natlty1_

  - Tabla perpetratorCharacteristics (Dimension)

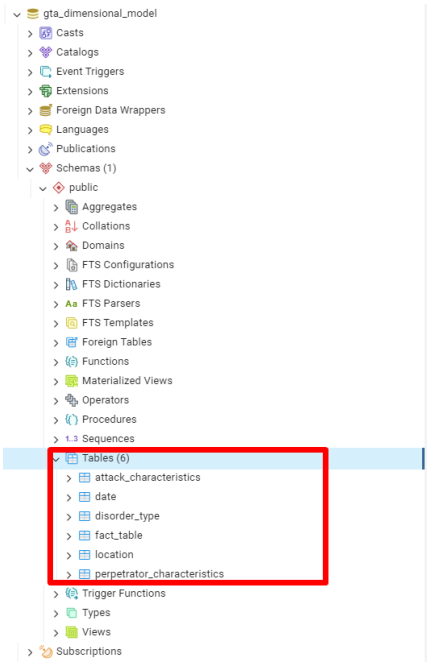
id_perpetrator	group_name	individual	number_perpetratores	number_perps_cap	perpetrator_claime
eventid+gname	gname	individual	nperps	nperpcap	claimed

  - tabla disorderType (Dimension)

id_disorder	disorder_type
1	Political Violence
2	Political Violence; demonstrations
3	demonstrations
4	Strategic developments
5	Desconocido

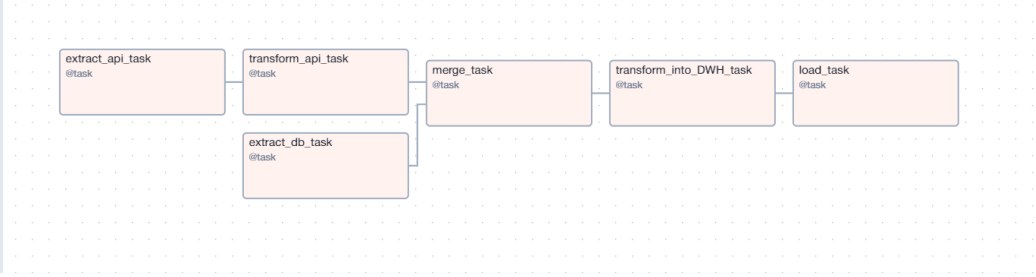
Carga de los datos

La carga del Data Warehouse se realiza con destino a una base de datos alojada en Azure, aunque el usuario puede decidir donde quiere que se le carguen los datos.



Data Pipeline en Airflow

Evidenciamos la estructura de las tasks de nuestro Data Pipeline en Airflow. Dentro de todas estas tasks, tenemos que destacar una en particular.



El asunto con *transform_into_DWH_task*

En nuestro modulo `etl` encontramos que devolvemos los 6 dataframes provenientes desde el modulo encargado de la transformación en formato JSON.

```python
def transform_into_DWH(df_json):      You, hace 14 horas via  PR #11 • Final touches: Airflow data pipeline
    """
    Function to define the DWH schema.

    """
    try:
        json_data = json.loads(df_json)
        raw_data = pd.DataFrame(json_data)
        location, date, attackCharacteristics, perpetratorCharacteristics, disorderType, factTable = transforming_into_DWH(raw_data)

        return location.to_json(orient="records"), date.to_json(orient="records"), attackCharacteristics.to_json(orient="records"), per
    except Exception as e:
        logging.error(f"Error defining DWH schema from JSON: {e}")
        return None, None, None, None, None, None
```

Sin embargo, al momento de manejar estos outputs en el script del DAG vemos que todos estos resultados vienen en una lista. Para poder manejarlos y transferirlos a las otras tareas, los establecemos en un formato JSON donde la clave va a ser el nombre de la tabla (`df` siendo la ***fact table***).

Luego, en `load_task`, recibimos este JSON y lo descomponemos dependiendo de su clave. La función `load` ya se encarga de convertirlos a DataFrame de nuevo y realizar el proceso de carga a la base de datos.

```python
@task
def transform_into_DWH_task(df_json):
    result = transform_into_DWH(df_json)              You, hace 15 horas via  PR #11 • Final touches:
    return {
        'location': result[0],
        'date': result[1],
        'attackCharacteristics': result[2],
        'perpetratorCharacteristics': result[3],
        'disorderType': result[4],
        'df': result[5],
    }

@task
def load_task(db_data):
    location_json = db_data["location"]
    date_json = db_data["date"]
    attackCharacteristics_json = db_data["attackCharacteristics"]
    perpetratorCharacteristics_json = db_data["perpetratorCharacteristics"]
    disorderType_json = db_data["disorderType"]
    df_json = db_data["df"]
    return load(location_json, date_json, attackCharacteristics_json, perpetratorCharacteristics_json,
```

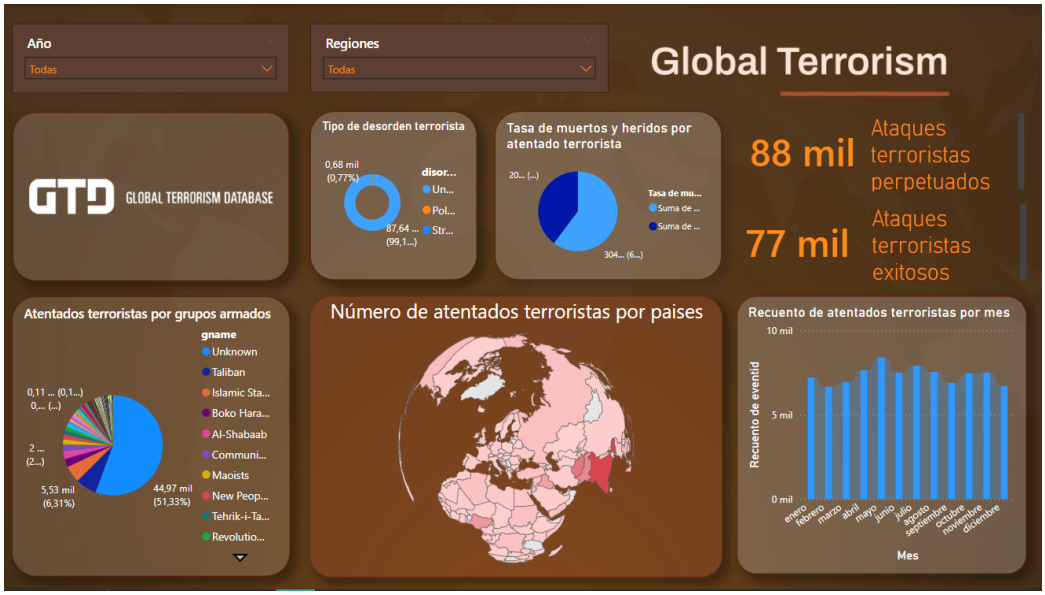
```python
transformed_data_api = transform_api_task(data_api)

merge_data = merge_task(data_db, transformed_data_api)

dimensional_model = transform_into_DWH_task(merge_data)

load_task(dimensional_model)
```

Visualización de los datos



Resumen de los cambios:

- Se realizan nuevos diseños en la página 1, en la que se incluyen nuevas variables como tipo de desorden terrorista, y la tasa de muertos y heridos en ataques terroristas. También se cambian el color de los filtros y las tarjetas, al igual que el logo, y también se modifican algunos errores en el mapa. Se cambian los colores en general de las gráficas para que se vea mucho más vivo.
- Se realizan un nuevo cambio en la página 2, en la cual se agrega como nueva variable *nwound*, para calcular las variables de mortalidad en los atentados terroristas.

Conclusiones

- Se minimiza el tiempo de respuesta de las bases de datos con el nuevo modelo estrella, al filtrar los datos duplicados, y tomando protagonismo el modelo a través de sus hechos y dimensiones.

- Se incluye almacenamiento de datos en la nube en la que se aprovecha la herramienta, aumenta la seguridad de datos y minimiza los errores que existen en local con el uso de las máquinas virtuales y postgreSQL.
- Se realiza un poderoso de análisis de datos con la complementación de la API, que nos permiten ver más allá de los simples hechos, y enfocarnos más en los motivos y curiosidades que traen los atentados terroristas en sí.
- La región con más actos terroristas cometidos a través de la violencia política es el sur de Asía, en la que también, curiosamente, es la segunda región con mayor número de actos terroristas registrados, solo por detrás de la región del norte de África y Europa medio.