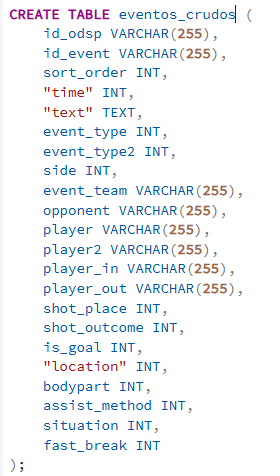
***Proyecto final de análisis de datos Eventos Deportivos***

***PASO DEL DATASET A PGADMIN4 (POSTGRE)***

1. Como primer arranque, se ha seleccionado el tema de eventos deportivos a nivel mundial del siguiente dataset: <https://www.kaggle.com/datasets/secareanualin/football-events/data>
2. Se realiza la carga de datos en pgAdmin4, de momento en forma local, donde primero creamos una tabla que tenga las mismas columnas del dataset

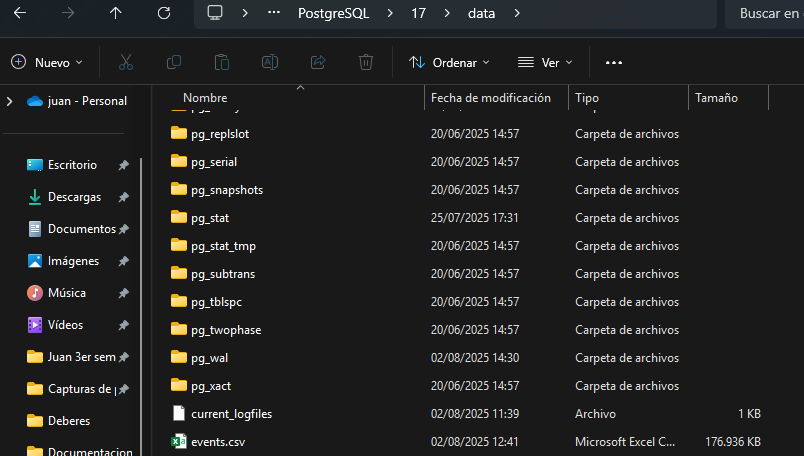


1. Una vez descargado el dataset, lo descomprimimos



1. Lo movemos a la siguiente el csv a la siguiente ruta: C:\Program Files\PostgreSQL\17\data

Esto con el objetivo de que Postgre (pgAdmin4) pueda leerlo e importarlo



1. En pgAdmin4 damos deberemos ejecutar el siguiente script

CREATE TEMP TABLE eventos\_temp (

id\_odsp VARCHAR(255),

id\_event VARCHAR(255),

sort\_order INT,

"time" INT,

"text" TEXT,

event\_type INT,

event\_type2 INT,

side INT,

event\_team VARCHAR(255),

opponent VARCHAR(255),

player VARCHAR(255),

player2 VARCHAR(255),

player\_in VARCHAR(255),

player\_out VARCHAR(255),

shot\_place INT,

shot\_outcome INT,

is\_goal INT,

"location" INT,

bodypart INT,

assist\_method INT,

situation INT,

fast\_break INT

);

COPY eventos\_temp

FROM 'C:/Program Files/PostgreSQL/17/data/events.csv'

WITH (

FORMAT csv,

HEADER true,

DELIMITER ',',

QUOTE '"',

ESCAPE '"',

NULL 'NA',

ENCODING 'UTF8'

);

Esto hará que los datos que están en el csv pasen a la tabla eventos

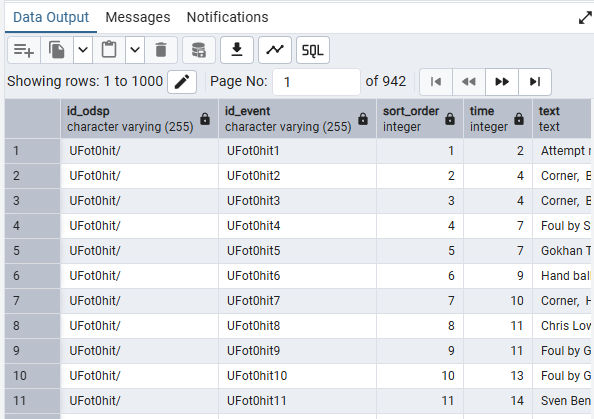
1. Insertamos los datos de la tabla temporal en la tabla que está en la database usando el siguiente script, esto tardará un poco

INSERT INTO eventos\_crudos

SELECT \*

FROM eventos\_temp;

1. La tabla debería verse asi:



***LIMPIEZA DE DATOS EN JUPYTER USANDO PYTHON***

1. Usamos el siguiente comando que nos dará las herramientas necesarias para continuar

pip install pandas sqlalchemy psycopg2-binary pymongo textblob

1. Importamos pandas, librería importante para limpieza



1. El siguiente código nos ayudará con la conexión entre pgAdmin4 y jupyter

import pandas as pd

from sqlalchemy import create\_engine

# --- 1. Configurar los detalles de la conexión

# Formato: "postgresql://usuario:contraseña@host:puerto/nombre\_db"

db\_user = "postgres"

db\_pass = "12345"

db\_host = "localhost"

db\_port = "5432"

db\_name = "proyecto\_futbol"

# Creamos la cadena de conexión

db\_string = f"postgresql://{db\_user}:{db\_pass}@{db\_host}:{db\_port}/{db\_name}"

# --- 2. Crear el "motor" de conexión y extraer los datos ---

try:

engine = create\_engine(db\_string)

print("Conexión a PostgreSQL local exitosa")

query = "SELECT \* FROM eventos\_crudos;"

df = pd.read\_sql\_query(query, engine)

print(f"Se cargaron {len(df)} registros desde PostgreSQL.")

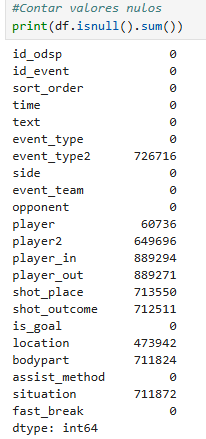
# Verificamos las primeras filas

print(df.head())

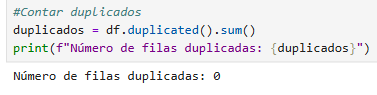
except Exception as e:

print(f"Error al conectar a PostgreSQL: {e}")

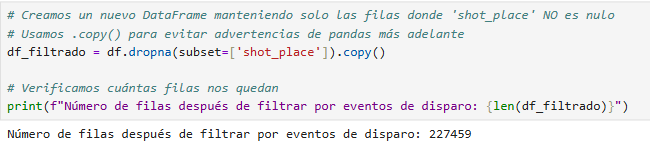
1. Contamos cuantos valores nulos tenemos



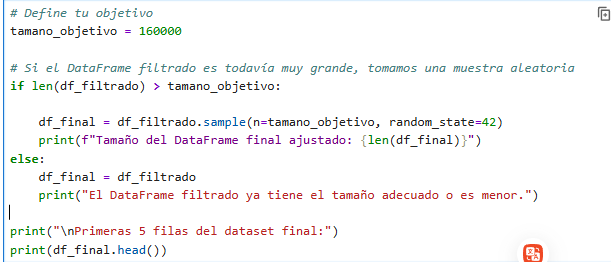
1. Contamos valores duplicados



1. Aquí, decidimos eliminar los valores nulos de shot\_place

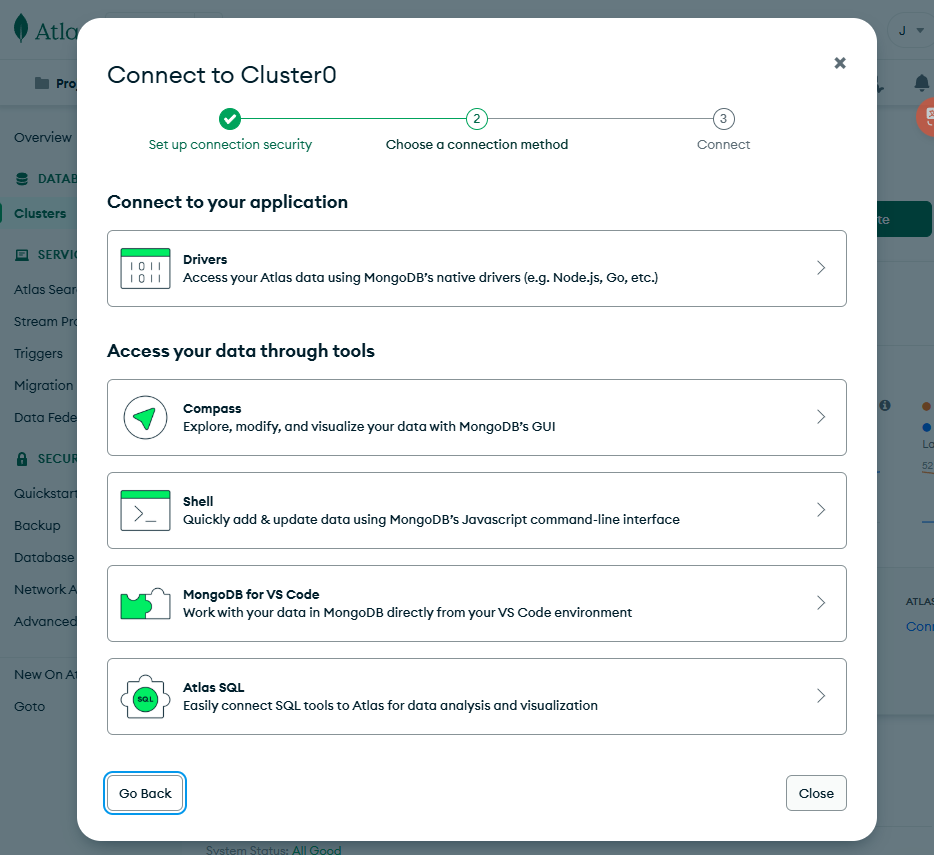


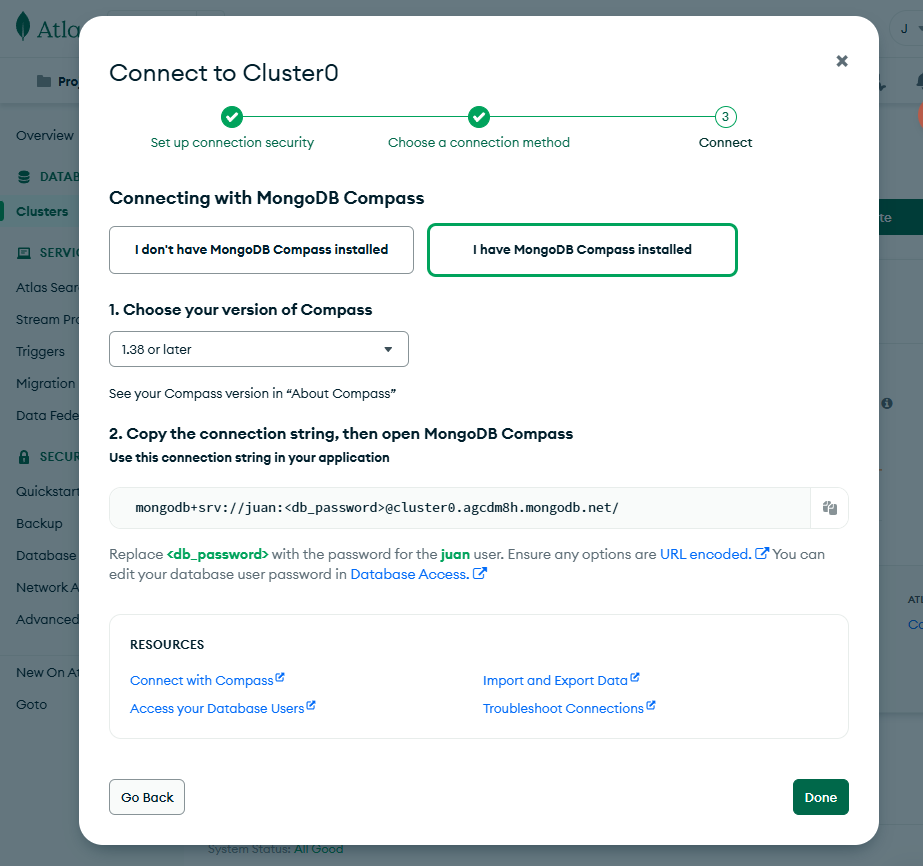
1. Aun tenemos varios datos, asi que eliminaremos algunos de forma al azar, esto hasta tener 160.000 datos



**PASAR EL DATAFRAME A MONGO DB ATLAS**

1. Seleccionar nuestro cluster en Mongo Atlas, para obtener el URI de conexión





Con esto, podremos tener acceso al cluster

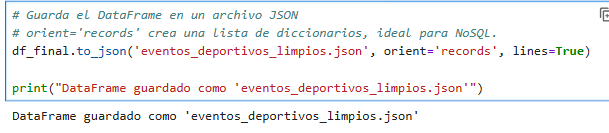
1. Utiliza un código similar a este para convertir el dataframe limpio en un archivo JSON, para hacer el correcto paso de datos

# Guarda el DataFrame en un archivo JSON

# orient='records' crea una lista de diccionarios, ideal para NoSQL.

df\_final.to\_json('eventos\_deportivos\_limpios.json', orient='records', lines=True)

print("DataFrame guardado como 'eventos\_deportivos\_limpios.json'")



1. El siguiente código, ayudará a enviarlo al cluster de MONGO Atlas, la inserción lo haremos mediante lotes, porque cuando lo hacemos todos de una sola vez, la conexión se perderá

import pymongo

import pandas as pd

# --- 1. Configuración de la Conexión ---

MONGO\_URI = "mongodb+srv://juan:12345@cluster0.agcdm8h.mongodb.net/"

try:

# --- 2. Conectar al Cliente de MongoDB ---

client = pymongo.MongoClient(MONGO\_URI)

client.admin.command('ping')

print("¡Conexión a MongoDB Atlas exitosa!")

# --- 3. Seleccionar Base de Datos y Colección ---

db = client['proyecto\_futbol']

collection = db['eventos\_procesados']

# --- 4. Preparar los Datos ---

datos\_para\_mongo = df\_final.to\_dict(orient='records')

# --- 5. Borrar Datos Antiguos ---

print("Limpiando colección existente...")

collection.delete\_many({})

# --- 6. Insertar los Datos en Lotes ---

batch\_size = 10000 # Tamaño de cada lote

total\_docs = len(datos\_para\_mongo)

print(f"Iniciando inserción de {total\_docs} documentos en lotes de {batch\_size}...")

for i in range(0, total\_docs, batch\_size):

# Creamos el lote (batch)

batch = datos\_para\_mongo[i:i + batch\_size]

# Insertamos el lote

collection.insert\_many(batch)

print(f" -> Lote {i//batch\_size + 1} insertado ({i + len(batch)}/{total\_docs} documentos)")

print("¡Carga de datos a MongoDB Atlas completada!")

except Exception as e:

print(f"Error durante el proceso: {e}")

finally:

if 'client' in locals():

client.close()

Deberíamos obtener algo asi:

