



ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS



PROYECTO DE FIN DE SEMESTRE

ANÁLISIS DE DATOS

PROFESOR: Ing. Lorena Chulde / Ing. Juan Pablo Zaldumbide

PERÍODO ACADÉMICO: 2024-B

FECHA DE ENTREGA: 04 / 02 / 2025

PROYECTO

TÍTULO:

ARQUITECTURA DE ANÁLISIS



ESTUDIANTES

Kevin Muñoz, Kevin Almeida

2024 - B

Objetivo General

Realizar una arquitectura donde se manejen al menos 1 millón de registros provenientes de DataSets, usando 4 bases de datos, donde 2 son SQL y 2 NoSQL, mostrando el paso de información entre estas.

Objetivos Específicos

- Realizar el paso de información entre archivos usando Python.
- Generar Dashboards explicativos con gráficas usando Power BI.
- Analizar la información que se vaya obteniendo.

Fuentes de Datos

Para la realización del Proyecto se sacó información de la siguiente página: <https://www.kaggle.com/> la cual fue de gran ayuda para la obtención de los respectivos DataSets.

Temáticas

Se escogieron las siguientes temáticas para estructurar el proyecto:

1. Eventos deportivos a nivel mundial
2. Actividades y hobbies
3. Tema definido por el grupo (Autos)
4. Restaurantes
5. Noticias mundiales

Casos de Estudio

Para este proyecto se han definido 15 casos de estudio en base a los temas elegidos, los cuales son:

1. Conocer el mes con la mayor cantidad de noticias publicadas.
2. Identificar la cantidad de noticias que tratan sobre temas relacionados a Ucrania.
3. Qué restaurantes tiene una buena calificación del público.
4. ¿Cuál es la ciudad con la mayor cantidad de restaurantes en el mundo?
5. Qué moneda es la más usada por los restaurantes.
6. Deporte con la mayor cantidad de equipos registrados.
7. Estatura con más registros de jugadores olímpicos.
8. Qué tipo de vehículo es más concurrente en el mercado.
9. Año con la mayor cantidad de vehículos registrados.
10. Película con la mayor popularidad.
11. Tipo de obra dominante en la industria.
12. Conocer la cantidad de encargados que existen por puesto de dirección.
13. Año con la mayor cantidad de personal de dirección fallecidos.
14. Restaurantes con servicio delivery online.
15. Peso (Kg) que registra más jugadores olímpicos.

Equipo de Trabajo

Kevin Muñoz: Encargado del diseño de la arquitectura del proyecto y generación de los Dashedboards.

Kevin Almeida: Encargado de la elección de la página, buscar los DataSets en formato JSON de la página escogida y la generación de videos.

Recurso y herramientas utilizadas

Para la elaboración del proyecto se usó Jupyter con Python en el paso de información entre tipos de archivos y entre bases de datos SQL y NoSQL.

Para las bases SQL se usó SQLite y SQL Server.

Para las bases NoSQL se usó MongoDB y CouchDB.

Para la generación de los Dashboards se usó Power BI por su facilidad de generación de gráficas.

Arquitectura de la solución

Primero, las librerías usadas para este proyecto fueron:

```
import sqlite3
import pyodbc
import pymongo
import pandas as pd
import couchdb
import json
import csv
from textblob import TextBlob
from pymongo import MongoClient
from datetime import datetime
```

Para el paso entre diferentes tipos de archivos (JSON a CSV) se logró usando código como el siguiente:


```
# PASO DE ARCHIVOS (DE JSON A CSV)

with open('BBC_NOTICIAS.json', 'r', encoding='utf-8') as json_file:
    data = json.load(json_file)

with open("BBC_NOTICIAS.csv", "w", newline='', encoding='utf-8') as csv_file:
    # Crear el escritor de CSV
    writer = csv.DictWriter(csv_file, fieldnames=data[0].keys())

    # Escribir la cabecera (nombres de columnas)
    writer.writeheader()

    # Escribir los datos
    for row in data:
        writer.writerow(row)
```


☐  BBC_NOTICIAS.csv

Para el paso entre archivos (CSV a JSON) se usó:

```
# PASO DE ARCHIVOS (DE CSV A JSON)

with open("BBC_NOTICIAS.csv", "r", encoding='utf-8') as csv_file:
    csv_reader = csv.DictReader(csv_file)
    data = [row for row in csv_reader]

with open("BBC_NOTI_JSON.json", "w", encoding='utf-8') as file:
    json.dump(data, file, indent=4, ensure_ascii=False)
```

☐  BBC_NOTI_JSON.json

También se hizo un paso de JSON a .db (extensión usada por SQLite), pero antes se realizó una revisión de los archivos, en este caso de los CSV provenientes de la conversión de cada JSON y confirmar que no haya valores vacíos o nulos.

```
df = pd.read_csv('BBC_NOTICIAS.csv')

nulos = df.isnull().sum()

print(nulos)

if nulos.sum() > 0:
    print("El archivo tiene valores nulos.")
else:
    print("El archivo no tiene valores nulos.")

_id            0
title          0
pubDate       0
guid          0
link          0
description    0
dtype: int64
El archivo no tiene valores nulos.
```

```
df = pd.read_csv('Peli.csv')

nulos = df.isnull().sum()

print(nulos)

if nulos.sum() > 0:
    print("El archivo tiene valores nulos.")
else:
    print("El archivo no tiene valores nulos.")
```

```
_id      0
Unnamed: 1  0
id        0
original_title  0
original_language  0
release_date  0
popularity    0
vote_average  0
vote_count    0
media_type    0
adult         0
dtype: int64
El archivo no tiene valores nulos.
```

```
df = pd.read_csv('Olimpicos.csv')

nulos = df.isnull().sum()

print(nulos)

if nulos.sum() > 0:
    print("El archivo tiene valores nulos.")
else:
    print("El archivo no tiene valores nulos.")
```

```
_id      0
ID        0
Name      0
Sex       0
Age       0
Height    0
Weight    0
Team      0
NOC       0
Games     0
Year      0
Season    0
City      0
Sport     0
Event     0
Medal     0
dtype: int64
El archivo no tiene valores nulos.
```

```
df = pd.read_csv('VentasYPopularidadDeAutosE.csv')

nulos = df.isnull().sum()

print(nulos)

if nulos.sum() > 0:
    print("El archivo tiene valores nulos.")
else:
    print("El archivo no tiene valores nulos.")
```

```
_id      0
region    0
category  0
parameter 0
mode      0
powertrain 0
year      0
unit      0
value     0
dtype: int64
El archivo no tiene valores nulos.
```

```
df = pd.read_csv('Restaurantes_ranking.csv')

nulos = df.isnull().sum()

print(nulos)

if nulos.sum() > 0:
    print("El archivo tiene valores nulos.")
else:
    print("El archivo no tiene valores nulos.")
```

```
_id      0
Restaurant ID  0
Restaurant Name  0
Country Code  0
City          0
Address        0
Locality       0
Locality Verbose  0
Longitude      0
Latitude       0
Cuisines       0
Average Cost for two  0
Currency       0
Has Table booking  0
Has Online delivery  0
Is delivering now  0
Switch to order menu  0
Price range     0
Aggregate rating  0
Rating color    0
Rating text     0
Votes          0
dtype: int64
El archivo no tiene valores nulos.
```


Estos archivos fueron los elegidos para el paso entre bases de datos, ya estaban listos para empezar el proceso. Primero se hizo el movimiento de JSON a .db

```
with open ("BBC_NOTICIAS.json", "r", encoding='utf-8') as file:
    data = json.load(file)

conexion = sqlite3.connect("BBC_NOTICIAS.db")
cursor = conexion.cursor()
cursor.execute("PRAGMA journal_mode=WAL;")
cursor.execute("""
CREATE TABLE IF NOT EXISTS noticias (
    id TEXT PRIMARY KEY,
    title TEXT NOT NULL,
    pubDate TEXT NOT NULL,
    guid TEXT NOT NULL,
    link TEXT NOT NULL,
    description TEXT NOT NULL
)
""")

# Insertar datos en la tabla
for item in data:
    cursor.execute("""
INSERT OR IGNORE INTO noticias (id, title, pubDate, guid, link, description)
VALUES (?, ?, ?, ?, ?, ?)
""", (
        item["_id"]["$oid"], # Convertir el campo _id.$oid a texto
        item["title"],
        item["pubDate"],
        item["guid"],
        item["link"],
        item["description"]
    ))

# Guardar los cambios y cerrar la conexión
conexion.commit()
conexion.close()
```

 BBC_NOTICIAS

El mismo procedimiento se siguió con los demás archivos hasta tenerlos en la

base de datos SQLite, y se presente la información como:

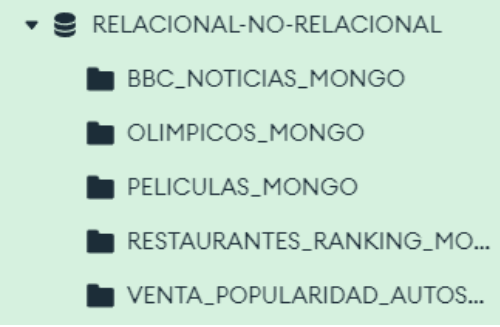
```
select * from noticias
```

id	title
66baad7caaeae7997f9ae929	Ukraine: Angry Zelensky vows to ...
66baad7caaeae7997f9ae931	Covid: Fourth jab for Scotland's ...
66baad7caaeae7997f9ae932	Protests across Russia see thousands...
66baad7caaeae7997f9ae933	Ukraine conflict: Your guide to ...
66baad7caaeae7997f9ae934	Russian gymnast investigated for ...
66baad7caaeae7997f9ae935	Ukraine: With placards and tears, ...
66baad7caaeae7997f9ae936	Twitter is part of our war effort - ...
66baad7caaeae7997f9ae937	Ukraine invasion: Volunteers 'workin...
66baad7caaeae7997f9ae938	Ukraine crisis: The West fights back...
66baad7caaeae7997f9ae939	Mariupol: Fires, no water, and bodie...
66baad7caaeae7997f9ae93a	The young Ukrainians battling pro-...
66baad7caaeae7997f9ae93b	Ukraine maps: New agreed ceasefire ...
66baad7caaeae7997f9ae93c	Five ways the Ukraine war could push...
66baad7caaeae7997f9ae93d	Ukraine: 'We try to tell them the ...
66baad7caaeae7997f9ae93e	War in Ukraine: Russian helicopter ...

De SQL a NoSQL

Ahora empiezan los movimientos entre bases de datos. Primero desde SQLite a MongoDB:

Primero, en MongoDB creamos la base y las diferentes colecciones.



Luego se hace el movimiento de información desde Jupyter:

```
sqlite_conn = sqlite3.connect("BBC_NOTICIAS.db")
sqlite_cursor = sqlite_conn.cursor()

client = MongoClient("mongodb://localhost:27017/")
mongodb = client["RELACIONAL-NO-RELACIONAL"]
coleccion = mongodb["BBC_NOTICIAS_MONGO"]
sqlite_cursor.execute("SELECT * FROM noticias")
columnas = [desc[0] for desc in sqlite_cursor.description]

documentos = []
for fila in sqlite_cursor.fetchall():
    documento = {columnas[i]: fila[i] for i in range(len(columnas))}
    documentos.append(documento)

if documentos: # Insertar solo si hay datos
    coleccion.insert_many(documentos)

# Cerrar conexiones
sqlite_conn.close()
client.close()
```

En MongoDB la información se presentará así:

```
{ "_id": ObjectId('6797f7cfbaa95726cac7dd19'),
  "id": "66baad7caaeae7997f9ae929",
  "title": "Ukraine: Angry Zelensky vows to punish Russian atrocities",
  "pubDate": "Mon, 07 Mar 2022 08:01:56 GMT",
  "guid": "https://www.bbc.co.uk/news/world-europe-60638042",
  "link": "https://www.bbc.co.uk/news/world-europe-60638042?at_medium=RSS&at_campaign=",
  "description": "The Ukrainian president says the country will not forgive or forget th..." }
```

Lo mismo pasa con los demás movimientos, todos siguen el mismo proceso.

También se hizo el paso de SQLite a CouchDB:

```
sqlite_conn = sqlite3.connect("BBC_NOTICIAS.db")
sqlite_cursor = sqlite_conn.cursor()

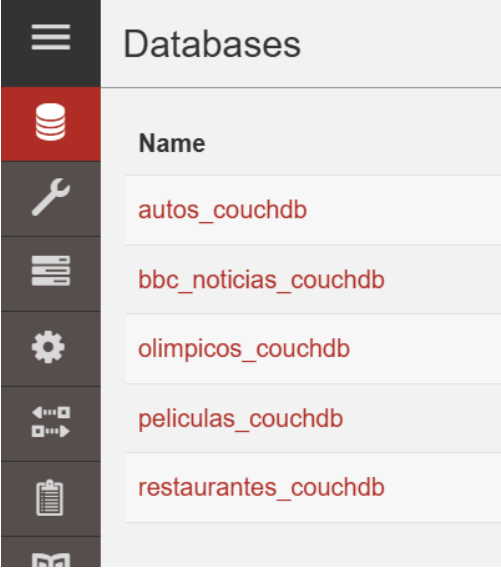
couch = couchdb.Server("http://localhost:5984/")
usuario = "admin"
contrasena = "123456"
couch.resource.credentials = (usuario, contrasena)

db_nombre = "bbc_noticias_couchdb"
if db_nombre in couch:
    db = couch[db_nombre]
else:
    db = couch.create(db_nombre)

sqlite_cursor.execute("SELECT * FROM noticias")
columnas = [desc[0] for desc in sqlite_cursor.description]

for fila in sqlite_cursor.fetchall():
    documento = {columnas[i]: fila[i] for i in range(len(columnas))}
    db.save(documento)

# Cerrar conexiones
sqlite_conn.close()
```



	_id	category	mode
<input type="checkbox"/>		Od1f6a2c6cbbdbfe17b8587...	Historical
<input type="checkbox"/>		Od1f6a2c6cbbdbfe17b8587...	Cars
<input type="checkbox"/>		Od1f6a2c6cbbdbfe17b8587...	Historical
<input type="checkbox"/>		Od1f6a2c6cbbdbfe17b8587...	Cars
<input type="checkbox"/>		Od1f6a2c6cbbdbfe17b8587...	Historical
<input type="checkbox"/>		Od1f6a2c6cbbdbfe17b8587...	Cars
<input type="checkbox"/>		Od1f6a2c6cbbdbfe17b8587...	Historical
<input type="checkbox"/>		Od1f6a2c6cbbdbfe17b8587...	Cars
<input type="checkbox"/>		Od1f6a2c6cbbdbfe17b8587...	Historical
<input type="checkbox"/>		Od1f6a2c6cbbdbfe17b8587...	Cars
<input type="checkbox"/>		Od1f6a2c6cbbdbfe17b8587...	Historical
<input type="checkbox"/>		Od1f6a2c6cbbdbfe17b8587...	Cars

De NoSQL a SQL

Se hizo el movimiento desde MongoDB a SQL Server, el cual es el repositorio final. Al hacer el paso de información desde MongoDB, primero se debe crear la base de datos en SQL Server:

```
client = pymongo.MongoClient("mongodb://localhost:27017/")
db = client["RELACIONAL-NO-RELACIONAL"]
collection = db["BBC_NOTICIAS_MONGO"]

conn = pyodbc.connect('DRIVER={ODBC Driver 17 for SQL Server};'
                      'SERVER=DESKTOP-CPD75BF\\SQLSERVER;'
                      'DATABASE=De_NoSQL_SQL;'
                      'trusted_connection=yes')

cursor = conn.cursor()

cursor.execute("""
IF NOT EXISTS (SELECT * FROM sysobjects WHERE name='noticias' AND xtype='U')
CREATE TABLE noticias (
    id NVARCHAR(255) PRIMARY KEY,
    title NVARCHAR(255),
    pubDate DATETIME,
    guid NVARCHAR(255),
    link NVARCHAR(255),
    description TEXT
)
""")
conn.commit()

def insertar_lote(lote):
    try:
        for item in lote:
            cursor.execute("""
MERGE INTO noticias AS target
USING (VALUES (?, ?, ?, ?, ?)) AS source (id, title, pubDate, guid, link,
ON target.id = source.id
WHEN NOT MATCHED THEN
INSERT (id, title, pubDate, guid, link, description)
VALUES (source.id, source.title, source.pubDate, source.guid, source.lin
            """, item[0], *item[1:])

            conn.commit()
        except Exception as e:
            print(f"Error al insertar lote: {e}")
            conn.rollback()

batch_size = 100000
lote = []
```

```
for document in collection.find({}, no_cursor_timeout=True).batch_size(batch_size):
    try:
        pubDate = document["pubDate"]
        pubDate = datetime.strptime(pubDate, "%a, %d %b %Y %H:%M:%S GMT")

        lote.append((
            document["id"],
            document["title"],
            pubDate,
            document["guid"],
            document["link"],
            document["description"]
        ))
        if len(lote) == batch_size:
            insertar_lote(lote)
            lote = []

    except Exception as e:
        print(f"Error al procesar el documento con id {document.get('id', 'desconocido')}: {e}")

if lote:
    insertar_lote(lote)

# Cerrar Las conexiones
cursor.close()
conn.close()
```

Las tablas se crearán desde Jupyter, si todo tiene éxito se visualizará la información en SQL Server:

- De_NoSQL_SQL
 - Database Diagrams
 - Tables
 - System Tables
 - FileTables
 - External Tables
 - Graph Tables
 - dbo.deportistas
 - dbo.electric_vehicles
 - dbo.noticias
 - dbo.peliculas
 - dbo.restaurantes

```
select * from noticias
```

id	title
66baad7caaeae7997f9ae929	Ukraine: Angry Zelensky vows to punish Russian atr...
66baad7caaeae7997f9ae931	Covid: Fourth jab for Scotland's vulnerable, and testi...
66baad7caaeae7997f9ae932	Protests across Russia see thousands detained
66baad7caaeae7997f9ae933	Ukraine conflict: Your guide to understanding day 11
66baad7caaeae7997f9ae934	Russian gymnast investigated for wearing pro-war s...
66baad7caaeae7997f9ae935	Ukraine: With placards and tears, Poles are greetin...
66baad7caaeae7997f9ae936	Twitter is part of our war effort - Ukraine minister
66baad7caaeae7997f9ae937	Ukraine invasion: Volunteers 'working on autopilot'
66baad7caaeae7997f9ae938	Ukraine crisis: The West fights back against Putin th...
66baad7caaeae7997f9ae939	Mariupol: Fires, no water, and bodies in the street
66baad7caaeae7997f9ae93a	The young Ukrainians battling pro-Russian trolls

También se hizo el paso de información desde CouchDB:

```

couch = couchdb.Server("http://localhost:5984/")
usuario = "admin"
contrasena = "123456"
couch.resource.credentials = (usuario, contrasena)

db = couch['bbc_noticias_couchdb']

conn = pyodbc.connect('DRIVER={ODBC Driver 17 for SQL Server};
                      'SERVER=DESKTOP-CPD7SBF\\SQLEXPRESS;'
                      'DATABASE=couchdb_sqlserver;'
                      'trusted_connection=yes')

cursor = conn.cursor()

cursor.execute("""
IF NOT EXISTS (SELECT * FROM sysobjects WHERE name='noticias' AND xtype='U')
CREATE TABLE noticias (
    id NVARCHAR(255) PRIMARY KEY,
    title NVARCHAR(255),
    pubDate DATETIME,
    guid NVARCHAR(255),
    link NVARCHAR(255),
    description TEXT
)
""")
conn.commit()

def insertar_lote(lote):
    try:
        for item in lote:
            cursor.execute("""
MERGE INTO noticias AS target
USING (VALUES (?, ?, ?, ?, ?)) AS source (id, title, pubDate, guid, link, description)
ON target.id = source.id
WHEN NOT MATCHED THEN
INSERT (id, title, pubDate, guid, link, description)
VALUES (source.id, source.title, source.pubDate, source.guid, source.link, source.description);
""", item[0], *item[1:])

            conn.commit()
    except Exception as e:
        print(f"Error al insertar lote: {e}")
        conn.rollback()

batch_size = 7000
lote = []

for doc_id in db:
    try:
        doc = db[doc_id]
        pubDate = doc.get("pubDate")
        if pubDate:
            pubDate = datetime.strptime(pubDate, "%a, %d %b %Y %H:%M:%S GMT")

            lote.append((
                doc["id"],
                doc.get("title", None),
                pubDate,
                doc.get("guid", None),
                doc.get("link", None),
                doc.get("description", None)
            ))

            if len(lote) >= batch_size:
                insertar_lote(lote)
                lote = []

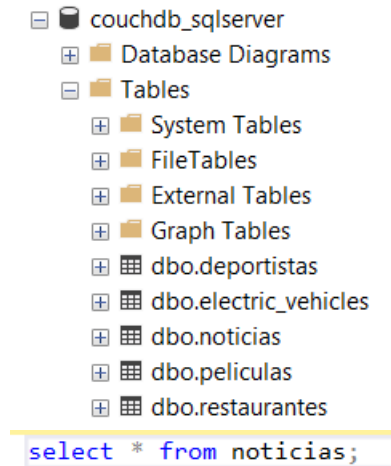
    except Exception as e:
        print(f"Error al procesar el documento con id {doc_id}: {e}")

if lote:
    insertar_lote(lote)

# Cerrar las conexiones
cursor.close()
conn.close()

```

Con un proceso exitoso, se verá la creación en SQL Server:



	title	pubDate
ibaa7caaeae7997f9ae929	Ukraine: Angry Zelensky vows to punish Russian atr...	2022-03-
ibaa7caaeae7997f9ae931	Covid: Fourth jab for Scotland's vulnerable, and testi...	2022-03-
ibaa7caaeae7997f9ae932	Protests across Russia see thousands detained	2022-03-
ibaa7caaeae7997f9ae933	Ukraine conflict: Your guide to understanding day 11	2022-03-
ibaa7caaeae7997f9ae934	Russian gymnast investigated for wearing pro-war s...	2022-03-
ibaa7caaeae7997f9ae935	Ukraine: With placards and tears, Poles are greetin...	2022-03-
ibaa7caaeae7997f9ae936	Twitter is part of our war effort - Ukraine minister	2022-03-

Análisis de Sentimientos

Para el análisis de sentimientos se trabajó con los archivos CSV usando la librería TextBlob.

```

df = pd.read_csv('BBC_NOTICIAS.csv')

def obtener_sentimiento(texto):
    blob = TextBlob(texto)
    return blob.sentiment.polarity

def clasificar_sentimiento(polaridad):
    if polaridad > 0:
        return 'Positivo'
    elif polaridad < 0:
        return 'Negativo'
    else:
        return 'Neutral'

df['sentimiento'] = df['description'].apply(obtener_sentimiento)
df['sentimiento_clasificado'] = df['sentimiento'].apply(clasificar_sentimiento)

conteo_sentimientos = df['sentimiento_clasificado'].value_counts()

conteo_df = conteo_sentimientos.reset_index()
conteo_df.columns = ['Sentimiento', 'Cantidad']

conteo_df.to_csv('conteo_sentimientos_BBC_NOTICIAS.csv', index=False)

print(conteo_sentimientos)

```

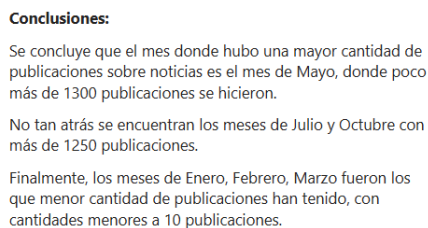
```

sentimiento_clasificado
Neutral      4420
Positivo     4143
Negativo     2014
Name: count, dtype: int64

```

Resultados a los casos de Estudio

1. Conocer el mes con la mayor cantidad de noticias publicadas.



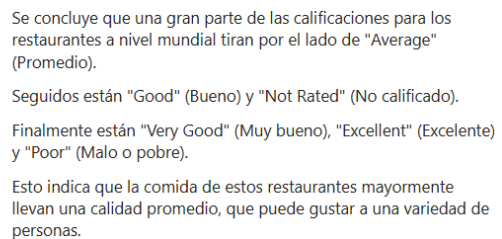
2. Identificar la cantidad de noticias que tratan sobre temas relacionados a Ucrania.

Cantidad de Noticias por Noticias sobre Ucrania

Topic	Quantity
Ukraine war in maps. Tr	32
Ukraine conflict: What	5
Ukraine conflict: What	4
Ukraine war: How rela	3
Ukraine: rich student	3
Ukraine conflict: Perso	2
Ukraine crisis: Why	2
Ukraine conflict: Perso	2
Ukraine war: Battle of	2
Ukraine war: Covid Res	2
Ukraine war: My sold c	2
Ukraine war: PM in hol	2
Ukraine war: Putin's gr	2
Ukraine War: Putin's ca	2
Ukraine War: UK pledg	2
Ukraine weapons: Wha	2
Ukraine: fugitive Putin	2
Ukraine: top medical	2
Ukraine: technology ZK	2
Ukraine: Zelenskyy's	2
Ukraine: Zelenskyy's	2
Ukraine: Zelenskyy's	2
Ukraine: Zelenskyy's	2
Ukraine ambassador ra	2
Ukraine anger as Mar	2
Ukraine attention thro	2
Ukraine host Scotland	2
Ukraine children: Killed	2
Ukraine confirms Russ	2
Ukraine conflict round	2
Ukraine conflict: Biden	2
Ukraine conflict: Biden	2
Ukraine conflict: Biden	2

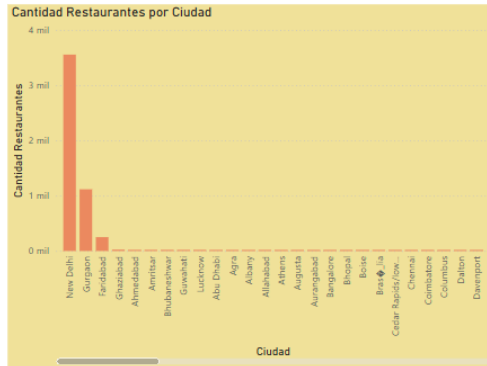
Se concluye que las noticias que tratan temas relacionados a Ucrania son demasiadas, en el gráfico no se logra observar todos los tipos de noticias relacionadas a ese tema.

3. Qué restaurantes tiene una buena calificación del público.



4. ¿Cuál es la ciudad con la mayor cantidad de restaurantes en el mundo?

Ciudad con la mayor cantidad de Restaurantes



Conclusiones:

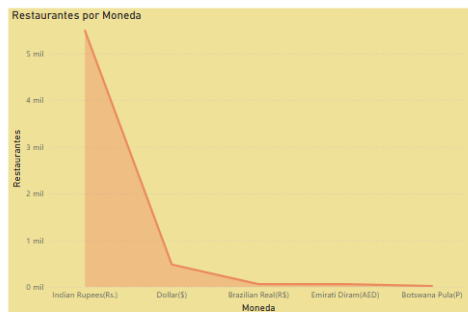
Se concluye que la ciudad en el mundo con la mayor cantidad de restaurantes de este dataset, es New Delhi, ubicado en la India.

Seguidos están Gurgaon y Faridabad, por una amplia diferencia, más de 2000 restaurantes menos que New Delhi.

Finalmente, la India es el país con la ciudad más abundante de restaurantes.

5. Qué moneda es la más usada por los restaurantes.

Moneda más usada por restaurantes



Conclusiones:

Se puede concluir que la moneda "Rupees" es la más usada en estos registros, esto debido a la gran cantidad de restaurantes de la India que existen en el dataset.

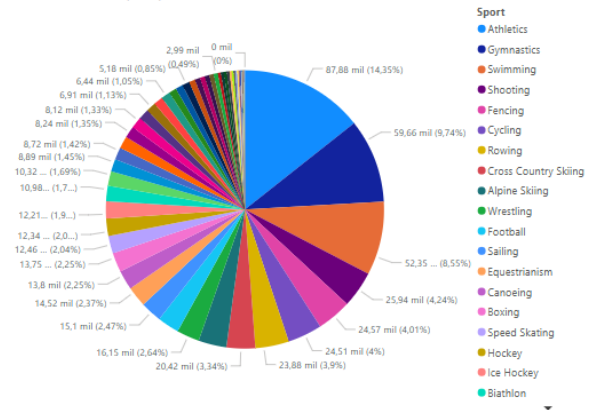
Muy por detrás, monedas como el Dólar y el Real Brasileño acompañan la gráfica, separados del primer lugar por una amplia diferencia. Estas monedas no llegan a ser usadas ni en mil restaurantes.

Finalmente, monedas como el Diram y Pula no son las más usadas en los restaurantes.

6. Deporte con la mayor cantidad de equipos registrados.

Deporte con la mayor cantidad de equipos

Recuento de Team por Sport



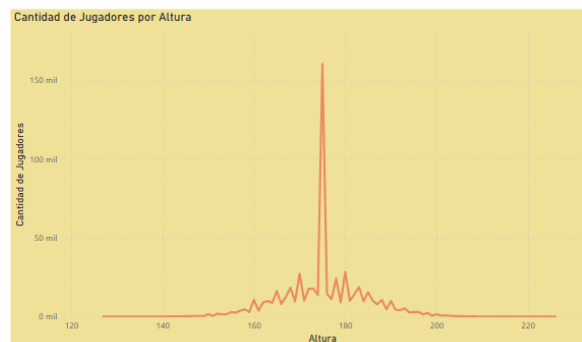
Conclusiones:

Se puede concluir que el deporte con la mayor cantidad de equipos registrados es el de Atletismo con más de 80000 equipos activos en la competición.

Gimnasia y Natación siguen un tanto lejos al deporte con la mayor cantidad de equipos, con más de 50000 equipos activos en cada uno.

7. Estatura con más registros de jugadores olímpicos.

Cantidad de jugadores por altura



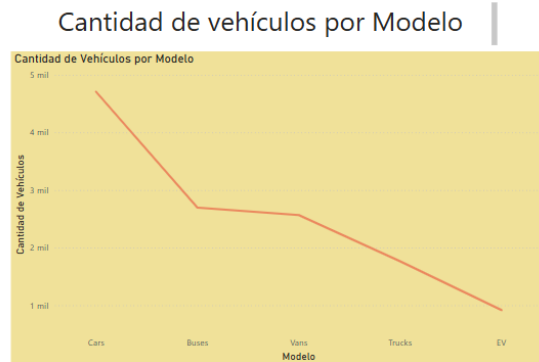
Conclusiones:

Se puede concluir que la mayor cantidad de participantes tiene una estatura de 175 cm, esta cantidad ronda más de 150000 jugadores.

De resto, la altura más "cercana" a esa cifra es 180 cm, con más de 25000 jugadores que mide esa cantidad.

Finalmente, alturas que van de los 200 cm tienen existencias de jugadores, aunque no son varios los que miden eso, hay registros de más de 1000 participantes.

8. Qué tipo de vehículo es más concurrente en el mercado.



Conclusiones:

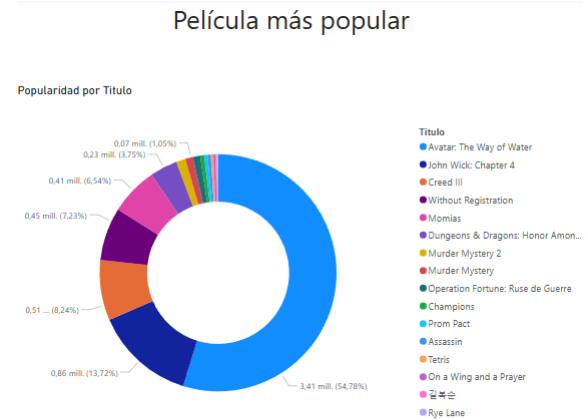
Los vehículos del tipo "Carro" (Común y Corriente) dominan la lista con cantidades por encima de las 4000 unidades registradas en esa categoría.

Más abajo están los de tipo "Buses", el cual tiene más de 2000 unidades registradas.

Luego se tienen "Camiones" y "Vehículos Eléctricos" por debajo de la cifra anteriormente mencionada.

Finalmente, en estos registros parece haber cierta popularidad por vehículos del tipo "Carro" los cuales son vehículos que muy cotidianamente se observan por la calle.

10. Película con la mayor popularidad.



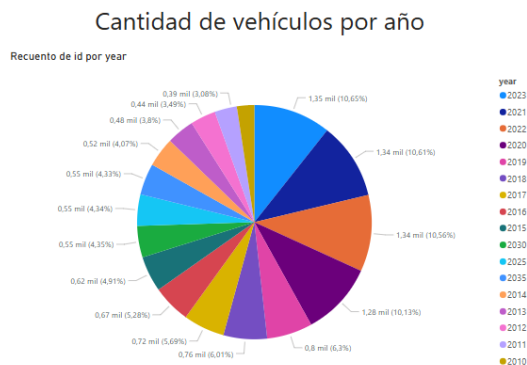
Conclusiones:

De este gráfico se puede concluir que la película más popular es Avatar, con una gran diferencia sobre los demás, sumando más de 3 millones de puntos de popularidad.

La película más "cercana" a esa cifra es una donde John Wick es el protagonista, sumando más de 800000 puntos de popularidad.

Finalmente, la película que no destaca en este dataset es Rye Line con poco más de 10000 puntos de popularidad.

9. Año con la mayor cantidad de vehículos registrados.



Conclusiones:

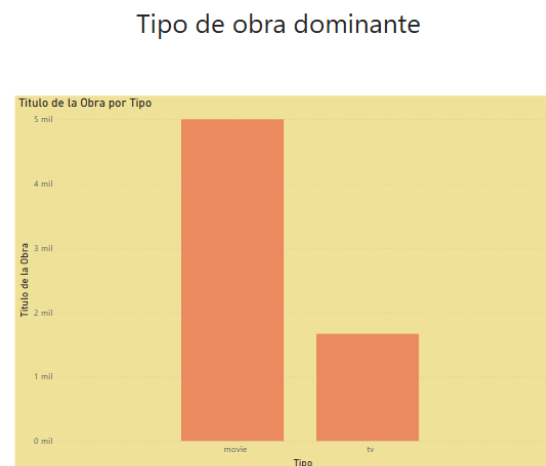
Se puede concluir que en este dataset hay más existencias de los vehículos del 2023, donde existen más de 1340 registros.

Le siguen vehículos del 2021 y 2022 con poco más de 1300 registros en cada uno.

También hay existencias de vehículos planeados para un futuro desde el 2025 en adelante.

Finalmente, se puede decir que los vehículos del 2023, tienden a ser más usados, y dejando en claro que vehículos de años pasados ya no tienen tanta popularidad.

11. Tipo de obra dominante en la industria.



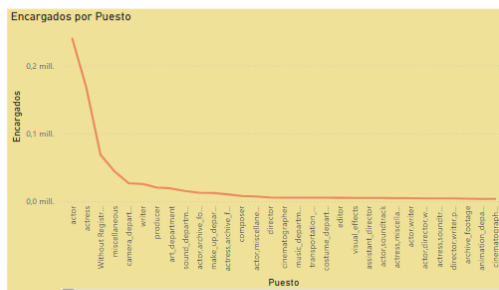
Conclusiones:

Se concluye que las obras del tipo Película abundan más en los registros, lo que indica una clara ventaja de estas sobre las obras que se pasan por TV.

La diferencia entre ambas es más de 3000 obras, 4999 son de Películas y 1665 son de propias de TV.

12. Conocer la cantidad de encargados que existen por puesto de dirección.

Cantidad de encargados por
puesto de producción



Conclusiones:

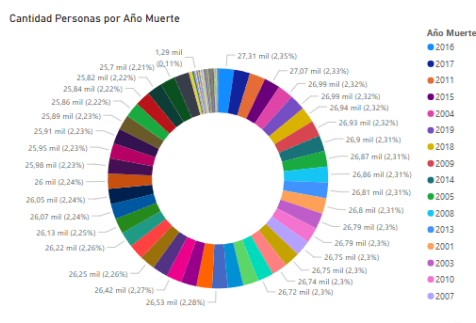
De este gráfico, se puede concluir que el puesto de "actor" es el más ocupado dentro de los puestos de dirección de alguna obra, llegando a tener más de 200000 registros en ese puesto, incluso con otro roles, es decir, compartidos.

Le sigue el rol de "actress" con más de 100000 registros existentes en ese puesto.

Finalmente, de este gráfico también se puede decir que los roles compartidos tiene fuerza dentro de la industria, las personas que los cumplen, llegan a ser polivalentes.

13. Año con la mayor cantidad de personal de dirección fallecidos.

Año con la mayor cantidad de personal fallecidos



Conclusiones:

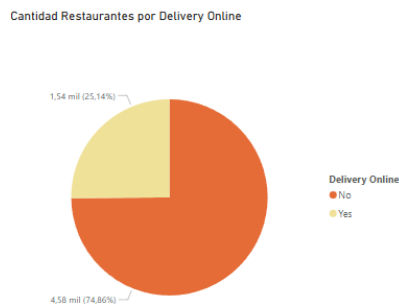
Se concluye que el año donde hubo una mayor cantidad de fallecidos fue en el 2016, existiendo más de 20000 registros para ese entonces.

No tan alejada de esa cifra, está el año 2017, con casi la misma cifra que en 2016, separadas por una mínima diferencia.

Finalmente, esto da paso a ciertas teorías que entre esos años pudo haber ocurrido algún incidente, o simplemente se trate de alguna casualidad en el conteo de estos registros.

14. Restaurantes con servicio delivery online.

Cantidad de restaurantes con
delivery online



Conclusiones:

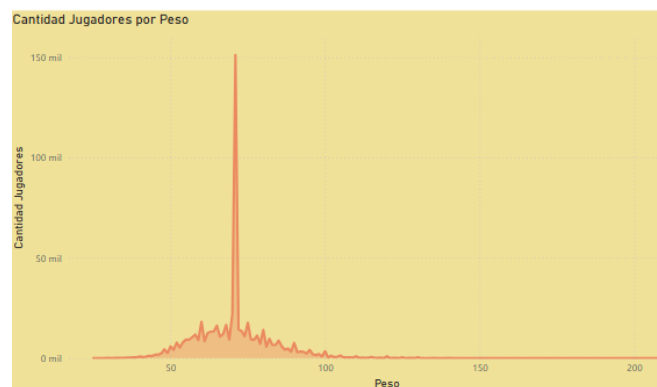
Se concluye que, son muchos los restaurantes que no incluyen este tipo de servicio, teniendo más de 4000 registros existentes para este campo.

Los que si cuentan con este servicio son poco más de 1500 restaurantes.

Finalmente, se puede decir que aquellos restaurantes que no incluyen este servicio son más propensos a tener ciertas opiniones negativas de los clientes.

15. Peso (Kg) que registra más jugadores olímpicos.

Cantidad de jugadores por Peso



Conclusions:

Se puede concluir que la mayor cantidad de jugadores tiene un peso que va de los 71 Kg, teniendo registros de más de 100000 existentes en esa cifra.

Muy alejada de esa cifra están los jugadores que pesan 60 Kg, teniendo poco más de 18000 registros existentes.

Finalmente, se puede decir que la mayor parte de jugadores lleva un peso en un rango promedio que se concentra alrededor de los 71 Kg, lo que representa el pico más significativo en los datos analizados

Conclusiones

- Con este proyecto se ha logrado dar respuestas a los distintos casos de estudio planteados en el grupo.
- Se ha podido aprender más sobre la herramienta Power BI.
- SQL Server es el más difícil cuando se trata de pasar información desde Python.

Recomendaciones

- Es importante hacer manejo de excepciones desde Python, porque nunca se sabe qué puede fallar.
- En SQL Server es importante crear primero la base de datos, es decir manualmente porque desde el código no se creará automáticamente.
- Hay que tener en cuenta la cantidad de registros que se insertarán en alguna base de datos desde Jupyter, ya que existe un límite que puede soportar.

Desafíos y Problemas

Durante el desarrollo del proyecto, el mayor problema encontrado fue el límite de registros que se pueden insertar desde Jupyter. Con DataSets que llevan 1 millón de registros, otros con 600 mil, no son ideales para hacer estos movimientos ya que superan un límite permitido para la salida de datos que se genera, el error específico es:

```
IOPub data rate exceeded.
The Jupyter server will temporarily stop sending output
to the client in order to avoid crashing it.
To change this limit, set the config variable
`--ServerApp.iopub_data_rate_limit`.
```

No obstante, al cambiar el límite permitido, seguía dando error, por lo que se tuvo que reducir la cantidad de datos que se pasarán a las bases de datos.

Cantidad de Datos Recopilada

✓	Import completed. 1162544 documents imported.
✓	Import completed. 612232 documents imported.
✓	Import completed. 12653 documents imported.
✓	Import completed. 11367 documents imported.
✓	Import completed. 6664 documents imported.
✓	Import completed. 6118 documents imported.

Con esos registros que se ha trabajado para el movimiento de información entre tipos de archivos, entre bases SQL a NoSQL y viceversa, análisis de sentimientos, análisis de información generada para la resolución de cada caso de estudio, se han recopilado más de:

1.8 millones de registros

Enlace a los videos

Video 1:

<https://youtu.be/DHKfFfz3WHw>

Video 2:

<https://www.youtube.com/watch?v=Z2c7-sC1qzA>

Enlace GitHub

<https://github.com/KevinMunoz15112004/Proyecto-Analisis-de-Datos>