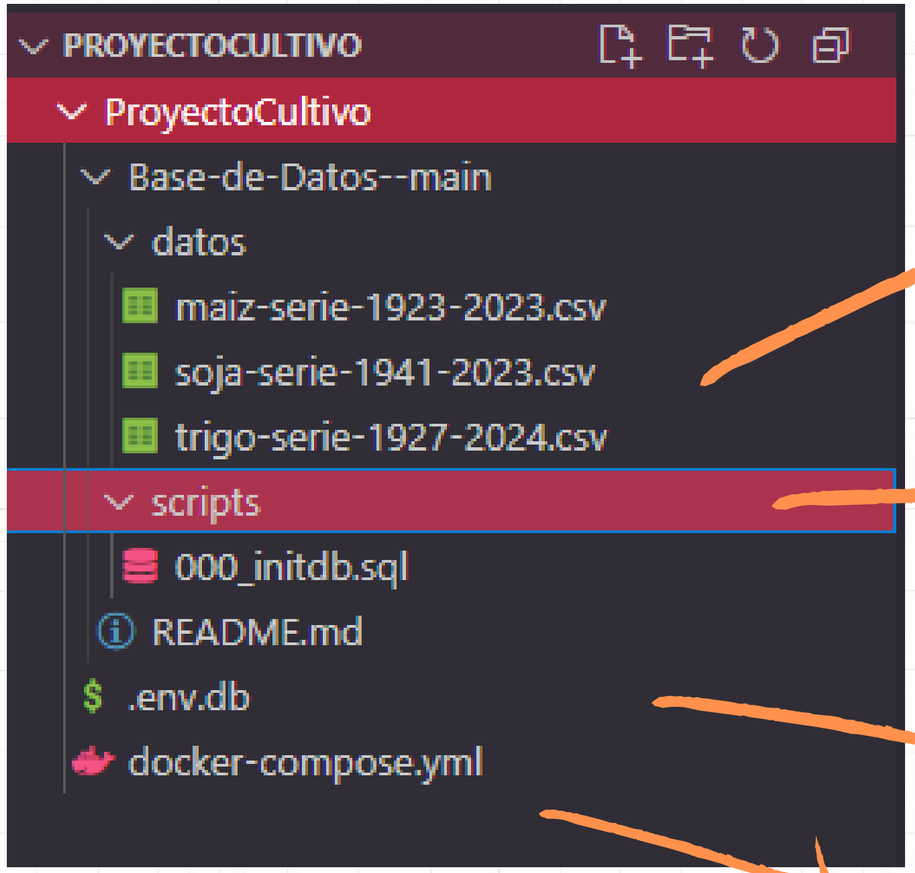
ETL-DATASETS

ANÁLISIS DE SIEMBRA, COSECHA Y PRODUCCIÓN DE CULTIVOS PRINCIPALES

Bergas Victoria, Corti Elba, Giovanardi Blanco Felipe, Lattanzi Simona,
Peliza Matías, Petrucci Bianca, Porporatto Lázaro, Rubio Carolina

GRUPO 3

Estructura del Proyecto



Cargamos los DataSets

Hacemos el Script

Archivo env.db

Archivo docker

Archivo de entorno (environment file)

```
#Definimos cada variable
DATABASE_HOST=db
DATABASE_PORT=5432
DATABASE_NAME=postgres
DATABASE_USER=postgres
DATABASE_PASSWORD=postgres
POSTGRES_INITDB_ARGS="--auth-host=scram-sha-256 --auth-local=trust"
# Configuracion para inicializar postgres
POSTGRES_PASSWORD=${DATABASE_PASSWORD}
PGUSER=${DATABASE_USER}
```

Archivo docker-compose.yml

```yaml
services:
  db:
    image: postgres:alpine
    env_file: .env.db
    container_name: postgres_cultivos
    environment:
      POSTGRES_DB: cultivos
      POSTGRES_USER: postgres
      POSTGRES_PASSWORD: postgres
    ports:
      - "5432:5432"
    volumes:
      - ./Base-de-Datos--main/Datos:/csv
```

Imagen PostgreSQL basada en Alpine

Cargamos las variables de entorno desde el archivo .env.db

Contenedor

Variables de entorno para PostgreSQL

Puerto

Guardamos los datos

Script

```
DROP DATABASE IF EXISTS cultivos;

▷Run
CREATE DATABASE cultivos;

▷Run
\c cultivos;
```

Eliminación de la base de datos

Creación de la base de datos

Conexión a la base de datos

```
▷Run
DROP TABLE IF EXISTS public.produccion_agricola CASCADE;
▷Run
DROP TABLE IF EXISTS public.cultivo CASCADE;
▷Run
DROP TABLE IF EXISTS public.campania CASCADE;
▷Run
DROP TABLE IF EXISTS public.departamento CASCADE;
▷Run
DROP TABLE IF EXISTS public.provincia CASCADE;
```

Eliminación de las tablas

Creación de las tablas

```sql
CREATE TABLE public.cultivo (
    id_cultivo INT PRIMARY KEY,
    nombre VARCHAR(100),
    anio INT,
    campania VARCHAR(100),
    id_provincia INT,
    id_departamento INT,
    superficie_sembrada DECIMAL(10, 2),
    superficie_cosechada DECIMAL(10, 2),
    produccion_toneladas DECIMAL(10, 2),
    rendimiento DECIMAL(10, 2),
    FOREIGN KEY (id_provincia) REFERENCES provincia (id_provincia),
    FOREIGN KEY (id_departamento) REFERENCES departamento (id_departamento)
);
```

Tablas temporales

```
Run | Select | Ask AI
CREATE TEMP TABLE temp_todo (
    cultivo_nombre VARCHAR,
    anio INT,
    campania VARCHAR,
    provincia_nombre VARCHAR,
    provincia_id VARCHAR,
    departamento_nombre VARCHAR,
    departamento_id VARCHAR,
    superficie_sembrada_ha FLOAT,
    superficie_cosechada_ha FLOAT,
    produccion_tm FLOAT,
    rendimiento_kgxha FLOAT
);
```

Solo existe durante la sesión actual y nos permite cargar y almacenar los datos crudos (temporalemente)

Las columnas coinciden con los campos del archico CSV

Facilita la detección y corrección de errores antes de afectar las tablas principales

Carga de archivos CSV en las tablas temporales

```
COPY temp_todo FROM '/csv/maiz-serie-1923-2023.csv' DELIMITER ',' CSV HEADER ENCODING
'UTF8';
▷Run
COPY temp_todo FROM '/csv/soja-serie-1941-2023.csv' DELIMITER ',' CSV HEADER ENCODING
'UTF8';
▷Run
COPY temp_todo FROM '/csv/trigo-serie-1927-2024.csv' DELIMITER ',' CSV HEADER ENCODING
'UTF8';
```

Nos aseguramos que se tomen los caracteres especiales

Indicamos que la primera fila contiene los nombres

Inserción de datos únicos y normalizados

```sql
-- Producción agrícola
▷ Run | Select | Ask AI
INSERT INTO public.produccion_agricola (
    id_cultivo, id_campania,
    superficie_sembrada, superficie_cosechada, produccion_toneladas
)
SELECT DISTINCT
    departamento_id::INT, anio,
    superficie_sembrada_ha, superficie_cosechada_ha, produccion_tm
FROM temp_todo;
```

```sql
-- Provincias
▷ Run | Select | Ask AI
INSERT INTO public.provincia (id_provincia, nombre)
SELECT DISTINCT provincia_id::INT, provincia_nombre
FROM temp_todo
WHERE provincia_id IS NOT NULL
ON CONFLICT (id_provincia) DO NOTHING;
```

```sql
-- Departamentos
▷ Run | Select | Ask AI
INSERT INTO public.departamento (id_departamento, nombre, id_provincia)
SELECT DISTINCT departamento_id::INT, departamento_nombre, provincia_id::INT
FROM temp_todo
WHERE departamento_id IS NOT NULL
ON CONFLICT (id_departamento) DO NOTHING;
```

```sql
-- Campañas
▷ Run | Select | Ask AI
INSERT INTO public.campania (id_campania, anio, nombre)
SELECT DISTINCT anio, anio, campania
FROM temp_todo
WHERE anio IS NOT NULL
ON CONFLICT (id_campania) DO NOTHING;
```

```sql
-- Cultivos
▷ Run | Select | Ask AI
INSERT INTO public.cultivo (
    id_cultivo, nombre, anio, campania,
    id_provincia, id_departamento,
    superficie_sembrada, superficie_cosechada, produccion_toneladas, rendimiento
)
SELECT DISTINCT
    departamento_id::INT, cultivo_nombre, anio, campania,
    provincia_id::INT, departamento_id::INT,
    superficie_sembrada_ha, superficie_cosechada_ha, produccion_tm, rendimiento_kgxha
FROM temp_todo
WHERE departamento_id IS NOT NULL
ON CONFLICT (id_cultivo) DO NOTHING;
```

Consultas SQL

1. Producción total por cultivo y año
2. Rendimiento promedio por cultivo (últimos 20 años)
3. Comparativa entre soja y maíz en los últimos 10 años

Primer Consulta

```
cultivos=# SELECT
cultivos-#     c.nombre AS cultivo,
cultivos-#     ca.anio,
cultivos-#     SUM(p.produccion_toneladas) AS produccion_total
cultivos-# FROM
cultivos-#     produccion_agricola p
cultivos-#     JOIN cultivo c ON p.id_cultivo = c.id_cultivo
cultivos-#     JOIN campania ca ON p.id_campania = ca.id_campania
cultivos-# GROUP BY
cultivos-#     c.nombre, ca.anio
cultivos-# ORDER BY
cultivos-#     ca.anio;
```

Primer Respuesta

cultivo	anio	produccion_total
trigo	1923	170420.00
ma◆z	1923	3743153.00
soja	1923	2443457.00
soja	1924	1547024.00
ma◆z	1924	2667589.00
trigo	1924	127813.00
trigo	1925	216718.00
ma◆z	1925	4853305.00
soja	1925	2575944.00
soja	1926	2582523.00
trigo	1926	220006.00
ma◆z	1926	4647371.00
soja	1927	4420944.00
trigo	1927	434875.00
ma◆z	1927	10188396.00

--More--

Podemos ver la producción total de cada cultivo por año, en toneladas

Segunda Consulta

```
cultivos=# SELECT
cultivos-#     c.nombre AS cultivo,
cultivos-#     AVG(c.rendimiento) AS rendimiento_promedio
cultivos-# FROM
cultivos-#     cultivo c
cultivos-# WHERE
cultivos-#     c.anio >= EXTRACT(YEAR FROM CURRENT_DATE) - 19
cultivos-# GROUP BY
cultivos-#     c.nombre
cultivos-# ORDER BY
cultivos-#     c.nombre;
```

Segunda Respuesta

```
 cultivo | rendimiento_promedio
---------+-------------------------
 ma?z    | 6001.6666666666666667
 soja    | 2376.0256410256410256
 trigo   | 2000.0000000000000000
(3 rows)

cultivos=#
```

Podemos ver el rendimiento promedio de cada cultivo, en KG x Ha

Tercer Consulta

```
cultivos=# SELECT
cultivos-#     ca.anio,
cultivos-#     c.nombre AS cultivo,
cultivos-#     SUM(p.produccion_toneladas) AS produccion_total
cultivos-# FROM
cultivos-#     produccion_agricola p
cultivos-#     JOIN cultivo c ON p.id_cultivo = c.id_cultivo
cultivos-#     JOIN campania ca ON p.id_campania = ca.id_campania
cultivos-# WHERE
cultivos-#     (c.nombre ILIKE 'soja' OR c.nombre ILIKE 'maiz' OR c.nombre ILIKE 'maíz')
cultivos-#     AND ca.anio >= EXTRACT(YEAR FROM CURRENT_DATE) - 9
cultivos-# GROUP BY
cultivos-#     ca.anio, c.nombre
cultivos-# ORDER BY
cultivos-#     ca.anio, c.nombre;
```

Tercer Respuesta

```
 anio | cultivo | produccion_total
------+---------+------------------
 2016 | soja    |       33711249.00
 2017 | soja    |       25978075.00
 2018 | soja    |       36248421.00
 2019 | soja    |       34418079.00
 2020 | soja    |       32805942.00
 2021 | soja    |       33013761.00
 2022 | soja    |       18905112.00
 2023 | soja    |       33018153.00
 2024 | soja    |        5360477.00
(9 rows)

cultivos=#
```

Esta consulta compara únicamente soja y maíz en términos de producción en los últimos 10 años, en toneladas

Muchas Gracias