# SQL Moderno para Ingeniería Agroindustrial

Integración con Bash, Python y Pandas

Curso de IA Aplicada al Agro

11 de enero de 2026

## Índice

# Introducción: El Flujo de Datos

En la Semana 03 aprendiste a limpiar datos en memoria con Pandas. Pero ¿y si los datos son demasiado grandes? ¿O si necesitas acceder a ellos desde múltiples scripts simultáneamente?

La respuesta es: “almacénalos en una base de datos”.

**Arquitectura del Pipeline**

**Sensores (CSV)** $\xrightarrow{\text{Bash}}$ **SQLite (DB)** $\xrightarrow{\text{Python}}$ **Pandas (DF)**

Figura 1: Flujo de datos desde la recolección hasta el análisis.

SQL (Structured Query Language) es el estándar porque:

- Es **declarativo**: describes *qué* quieres, no *cómo* hacerlo.
- Es **eficiente**: filtra millones de registros antes de cargarlos en la RAM de Python.
- Es **universal**: lo que aprendas aquí sirve para PostgreSQL, BigQuery o Spark SQL.

# 1. Capítulo I: Primeros Pasos con SQLite

SQLite es una base de datos ligera, sin servidor, ideal para entornos frugales (Raspberry Pi, laptops rurales) donde la conectividad es limitada.

## 1.1. Crear y Poblar una Tabla

Imagina una tabla que registra la actividad de sensores de humedad en diferentes lotes de café.

```sql
-- Crear tabla de sensores
CREATE TABLE sensores (
    id INTEGER PRIMARY KEY,
    zona TEXT NOT NULL,
    fecha DATE,
    temperatura REAL,
    humedad REAL
);

-- Insertar datos manuales (solo para prueba)
INSERT INTO sensores (zona, fecha, temperatura, humedad)
VALUES ('Norte', '2026-01-10', 24.5, 78.2);
```

Listing 1: DDL para crear la tabla

## 1.2. Consultas Básicas (Queries)

```sql
-- 1. Ver todos los registros
SELECT * FROM sensores;

-- 2. Filtrar zonas críticas (Humedad alta = riesgo de hongos)
SELECT zona, fecha
FROM sensores
WHERE humedad > 80;

-- 3. Ordenar por temperatura
SELECT * FROM sensores ORDER BY temperatura DESC;
```

Listing 2: Consultas de exploración

**Caso de Uso: Alerta de Roya**

La consulta número 2 es vital. Si la humedad supera el 80 % y la temperatura está entre 20-25°C, las condiciones son ideales para la roya del café. SQL nos permite detectar esto en milisegundos.

# 2. Capítulo II: Ingesta Masiva desde Bash

Como ingenieros de datos, no insertamos datos uno por uno. Automatizamos la ingesta de archivos CSV generados por los dataloggers.

## 2.1. Importar CSV a SQLite

Supón que tienes `sensores_2026.csv` con miles de registros.

```
# Comando en terminal (Bash)
# .mode csv le dice a SQLite que espere comas
# .import archivo tabla
sqlite3 finca.db << EOF
.mode csv
.import sensores_2026.csv sensores
EOF
```

Listing 3: Script de Bash para Ingesta

**⚠ Advertencia: Integridad de Datos**

Asegúrate de que la primera fila del CSV (encabezados) coincida con los nombres de las columnas en la tabla, o usa la opción `--skip 1` si vas a mapear columnas manualmente.

## 2.2. Exportar Resultados a CSV

Después de procesar los datos, el agrónomo necesita un Excel.

```
sqlite3 finca.db << EOF
.headers on
.mode csv
.output reporte_sequia.csv
SELECT zona, AVG(humedad) as prom_hum
FROM sensores
GROUP BY zona
HAVING prom_hum < 40;
EOF
```

# 3. Capítulo III: SQL + Python = Poder Total

Pandas delega el trabajo pesado a SQL y se encarga del análisis fino y la visualización.

## 3.1. Conectar y Consultar

```python
import sqlite3
import pandas as pd

# 1. Conexión a la base de datos (archivo local)
conn = sqlite3.connect('finca.db')

# 2. Ejecutar consulta optimizada
# Traemos SOLO los datos necesarios, no toda la base
query = '''
    SELECT zona, AVG(temperatura) as temp_prom
    FROM sensores
    WHERE fecha >= '2026-01-01'
    GROUP BY zona
'''

df = pd.read_sql(query, conn)

print("--- Promedios por Zona ---")
print(df)

conn.close()
```

Listing 4: Python leyendo SQL

**Concepto Clave: Eficiencia de Memoria**

Filtrar en SQL (`WHERE`) es 10x más eficiente que cargar todo el CSV en Pandas y filtrar después (`df[df['col'] >x]`).

# 4. Capítulo IV: Ética y Gobernanza

Una base de datos no es neutral. Refleja decisiones de diseño que impactan a las personas.

- **¿Qué se mide?** Si solo medimos producción y no salud del suelo, optimizaremos la explotación a corto plazo.
- **¿Quién accede?** El pequeño agricultor debe ser dueño de sus métricas, no solo la empresa que vende los insumos.

**Soberanía Tecnológica**

En contextos rurales, evita depender 100 % de la nube. SQLite permite mantener los datos **locales, controlables y auditables** incluso si se corta el internet satelital.

# Reto Final: Monitor de Cultivos

1. Ejecuta `generar_sensores.py` para crear `datos_crudos.csv`.
2. Crea la base de datos `finca.db` y la tabla `lecturas` usando Bash.
3. Escribe una consulta SQL que identifique "Días Críticos": * Humedad <30 % (Estrés hídrico) * **O** Temperatura >35°C (Golpe de calor)
4. Carga esos datos en Python y genera un gráfico de barras.
5. Exporta los IDs de los lotes afectados a `alertas.txt`.