

⌚ Python & PostgreSQL Audit Toolkit

Este repositorio contiene un conjunto de herramientas modulares en **Python** diseñadas para conectar, auditar y explorar bases de datos **PostgreSQL** de manera segura y eficiente.

El objetivo es automatizar la exploración de datos (Data Discovery) y la ingeniería inversa de esquemas, reemplazando consultas SQL manuales repetitivas con scripts de Python robustos.



📁 Arquitectura del Proyecto

```
📦 proyecto-postgres-python
├── .venv/                                # Entorno virtual (Ignorado por Git)
├── .env                                    # Variables de entorno (Ignorado por Git)
├── .gitignore                             # Configuración de exclusiones
├── connexion.py                           # 🚗 Core: Motor de conexión seguro
├── ver_bases.py                          # 🌐 Server: Listado de bases de datos
├── mapear_db.py                           # 📄 Analysis: Diccionario de tablas
├── inspector_avanzado.py                 # 🌟 Engineering: Auditoría de PK/FKs
├── README.md                               # Documentación
└── requirements.txt                      # Dependencias
```

Archivo	Nivel	Responsabilidad
connexion.py	Core	Gestiona la conexión a la BD usando SQLAlchemy . Implementa seguridad vía variables de entorno (.env) para no exponer credenciales.
ver_bases.py	Server	Se conecta a la base maestra para listar todas las bases de datos existentes en el servidor y su tamaño.
mapear_db.py	Analysis	Genera un "Diccionario de Datos" legible (Dataframe) de una base específica. Ideal para ver tablas y tipos de datos rápidamente.
inspector_avanzado.py	Eng	Utiliza SQLAlchemy Inspector para auditar relaciones complejas. Detecta Claves Primarias (PK) y Claves Foráneas (FK) automáticamente.

🔧 Instalación y Configuración

1. Clonar el repositorio

```
git clone https://github.com/DavidHuamanRoman/proyecto-postgres-python.git
cd proyecto-postgres-python
```

2. Preparar el entorno

Se recomienda usar un entorno virtual para mantener las dependencias aisladas.

```
# Crear entorno virtual (Windows)
python -m venv .venv

# Activar entorno
.venv\Scripts\activate

# Instalar librerías
pip install -r requirements.txt
```

3. Configuración de Seguridad (.env)

Este proyecto no "hardcodea" contraseñas. Debes crear un archivo llamado .env en la raíz del proyecto y definir tus credenciales:

```
DB_USER=postgres
DB_PASS=tu_password_secreto
DB_HOST=localhost
DB_PORT=5432
DB_NAME=postgres
```

🛠 Uso de las Herramientas

A. Auditoría de Servidor

Para ver qué bases de datos existen en tu instancia de Postgres:

```
python ver_bases.py
```

Salida: Tabla con nombres de DBs y su tamaño en disco

B. Mapeo de una Base de Datos

Genera un reporte limpio de las tablas para análisis.

Tip: Puedes editar mapear_db.py para cambiar la base de datos objetivo si no quieres usar la default.

```
python mapear_db.py
```

Salida: Dataframe con tablas y tipos de datos

C. Auditoría Avanzada de Esquema

Detecta PKs y FKs automáticamente.

```
python inspector_avanzado.py
```

Salida: Reporte detallado de relaciones entre tablas

🤝 Contribuciones

¡Las contribuciones son bienvenidas! Si deseas mejorar las herramientas o agregar nuevas funcionalidades, por favor sigue estos pasos:

1. Haz un fork del repositorio.
2. Crea una nueva rama (`git checkout -b feature/nueva-funcionalidad`).
3. Realiza tus cambios y haz commit (`git commit -m 'Agrega nueva funcionalidad'`).
4. Haz push a la rama (`git push origin feature/nueva-funcionalidad`).
5. Abre un Pull Request.

📄 Licencia

Este proyecto está bajo la Licencia MIT. Consulta el archivo LICENSE para más detalles.

Hecho con ❤️ por David Huamán Román