Cómo instalar DuckDB para usar SQL en Python

Índice

Requisitos previos

Tanto si estás en Linux o en Windows con una máquina virtual o con anaconda, necesitás tener instalado:

- **Python 3.6 o superior**
- **pip** (el gestor de paquetes de Python)

Instalación según sistema operativo

Windows + Anaconda

1. Abrí **Anaconda Prompt**
2. *(Recomendado) Activá el entorno donde vas a trabajar, por ejemplo:*

```
conda activate base
```

3. Instalá **DuckDB**:

```
conda install duckdb
```

*De no activar el entorno virtual, se puede utilizar el mismo comando "**`conda install duckdb`**" y se estaría instalando el paquete en el Python del sistema. Ver más sobre entornos acá*

Linux (Debian, Ubuntu, Mint, etc.)

1. Abrí una terminal.
2. *(Recomendado) Activá tu entorno virtual o conda:*

```
source venv/bin/activate
```

o

```
conda activate mi_entorno
```

3. Instalá DuckDB:

```
pip install duckdb
```

*De no activar el entorno virtual, probablemente sea necesario ejecutar en terminal el comando "**`sudo pip install duckdb`**". Ver más sobre entornos acá*

Google Colab

Colab corre en la nube, así que no necesitas preocuparte por entornos locales. Para usar DuckDB en Google Colab, necesitás instalarlo la primera vez que abras el notebook:

```
!pip install duckdb
```

Uso de duckdb

Si fue correctamente instalada, deberías poder importar la librería sin obtener ningún tipo de error.

```
import duckdb
```

Para realizar consultas de SQL, es necesario agregar el comando "**`duckdb.query(…)`**" que entre los paréntesis contiene la consulta a realizar.

Ejemplo:

```
consultaSQL = duckdb.query("""
    SELECT DISTINCT Columna1, Columna2
    FROM tabla
""")
```

Diferencias de uso con inline_sql

El formato para realizar consultas de sql con esta librería es ligeramente distinta a **inline_sql**. A continuación podemos ver las diferencias:

	inline_sql	**duckdb**
Consulta	`consultaSQL = """ SELECT DISTINCT Columna1, Columna2 FROM tabla; """`	`consultaSQL = duckdb.query(""" SELECT DISTINCT Columna1, Columna2 FROM tabla """)`
***Guardar resultado de consulta como* dataframe**	`consultaSQL = """ SELECT DISTINCT Columna1, Columna2 FROM tabla; """ dataframeResultado = sql^ consultaSQL`	`dataframeResultado = duckdb.query(""" SELECT DISTINCT Columna1, Columna2 FROM tabla """).df()`

Chequeo de funcionamiento y errores comunes:

Probar que funciona

Abrí Python o un Jupyter Notebook y escribí este código:

```
import duckdb
duckdb.sql("SELECT 'DuckDB funciona!'").show()
```

Deberías ver una pequeña tabla con el texto "DuckDB funciona!".

ModuleNotFoundError: No module named 'duckdb'

- Asegurate de estar trabajando en el **mismo entorno** donde lo instalaste.
- Verificá el path con:

```
import sys
print(sys.executable)
```

Nota sobre entornos de instalación:

Activar el entorno no es estrictamente obligatorio, pero es **altamente recomendable**. Te permite mantener los paquetes organizados por proyecto y evitar conflictos con el Python del sistema.

Al activar un entorno:

- Instalás paquetes como duckdb aisladamente, sin mezclar con el sistema.
- Mantenés limpio el Python del sistema operativo.
- Podés tener versiones distintas de paquetes para distintos proyectos.
- Evitás conflictos y errores por versiones incompatibles.

Si no se activa el entorno, se está usando el **Python del sistema**. Es implica:

- Puede necesitar permisos de superusuario (sudo)
- Puede romper paquetes del sistema si no se usa con cuidado
- Hace más difícil desinstalar o actualizar paquetes sin conflictos

¿Cómo crear un entorno y activarlo?

Podés tener entornos de desarrollo tanto con **conda** como con **virtualenv (o venv)**, que es el sistema de entornos virtuales nativo de **Python**. Ambas opciones son válidas y tienen sus ventajas.

OPCIÓN 1: Crear un entorno con Conda

Ventajas:

- Maneja tanto paquetes de Python como librerías del sistema (como C, R, etc.).
- Ideal si usás Anaconda o Miniconda.
- Facilita el manejo de dependencias complejas.

Pasos:

1. Crear el entorno

```
conda create --name mi_entorno python=3.10
```

2. Activar el entorno

```
conda activate mi_entorno
```

3. Instalar duckdb (desde conda-forge si no está en defaults)

```
conda install -c conda-forge duckdb
```

4. Salir del entorno

```
conda deactivate
```

OPCIÓN 2: Crear un entorno con venv (entorno virtual de Python)

Ventajas

- Ligero, sin necesidad de instalar Anaconda.
- Usás solo pip para instalar paquetes.
- Viene incluido en Python (desde 3.3).

Pasos

1. Crear una carpeta para tu entorno virtual

```
python3 -m venv mi_entorno
```

2. Activar el entorno

```
source mi_entorno/bin/activate
```

3. Instalar duckdb

```
pip install duckdb
```

4. Verificar instalación

```
python -c "import duckdb; print(duckdb.__version__)"
```

5. Salir del entorno

```
deactivate
```

¿Cuál conviene?

Criterio	*conda*	*venv + pip*
Facilidad con paquetes complejos	mejor	a veces falla con paquetes con C/C++
Ligero y minimalista	requiere instalación de conda	viene con Python
Comunidad y compatibilidad	enorme (especialmente ciencia de datos)	muy compatible también
Gestión de dependencias del sistema	excelente	sólo maneja Python

¿Cómo ver todos los entornos creados?

CONDA:

Si en algún momento quieres ver qué entornos tienes creados en tu sistema, puedes usar el siguiente comando:

```
conda env list
```

Este comando te mostrará una lista de todos los entornos de conda disponibles.

venv + pip:

Tenés que recordar dónde los creaste.

Recomendación para venv

Guardar todos tus entornos virtuales en una carpeta fija como **~/venvs/** es súper útil. Por ejemplo:

```
mkdir ~/venvs
cd ~/venvs
python3 -m venv entorno_proyecto1
```

Así después es más fácil verlos:

```
ls ~/venvs
```

Por convención, a veces se crea una carpeta ~/envs/ o ~/proyectos/mi_proyecto/venv/, pero no es obligatorio.

⚠️ *Si **no fuiste tan organizado** y querés buscar todos los entornos posibles en tu sistema (opcional):*

```
find ~ -type d -name "bin" -exec test -e "{}/activate" \; -
print
```

⚠️ Este comando puede tardar un poco porque escanea todo tu home.