External-Drive-Catalog_V01

Yahya Nazer

Table of contents

External Drive Catalog - Setup & Usage Guide

A single tool that catalogs all of your external drives into one searchable database, so you can answer *"which drive is that file on?"* without plugging anything in. Works on **Windows** and **Mac**. Handles **UTF-8** (Farsi, French, German, Chinese filenames all index and search correctly).

It replaces the three old scripts (index-to-CSV + the two PostgreSQL export scripts). There is **no database server to install** - it uses **SQLite**, which is built into Python. The whole catalog is one portable file you can back up.

---

1. Folder structure

Create one top folder anywhere (Desktop, Dropbox, etc.) with this layout:

```
External-Drive-Catalog/        <- name it whatever you like
    A-Data/                    <- the database + all exports go here
    B-Engines/
        External-Drive-Catalog_V01.py
```

Put the `.py` file in **B-Engines**. You do **not** need to create `A-Data` by hand - the program creates it automatically the first time it runs. The script finds `A-Data` by going one level up from wherever it sits, so as long as the two folders share the same parent, it just works.

---

2. Install (one time)

The database

Nothing to install. SQLite ships inside Python. The catalog file (`A-Data/drive-catalog.sqlite3`) is created automatically on first launch.

Python

You only need Python 3 with Tk (the GUI toolkit). Both come together in the official installers.

Windows 1. Download Python 3 from https://www.python.org/downloads/windows/ 2. Run the installer and **tick "Add python.exe to PATH"**. Leave "tcl/tk" checked (default). 3. Verify in a Command Prompt: `python --version`

Mac 1. Download Python 3 from https://www.python.org/downloads/macos/ (the python.org build includes Tk; the system Python often does not). 2. Verify in Terminal: `python3 --version` `python3 -m tkinter` The second command should pop up a small test window. If it does, Tk works. - If you use Homebrew Python instead and Tk is missing: `brew install python-tk`

Optional: Excel export

Only needed for the "Export Drive Summary (Excel)" button. Everything else works without it.

```
pip install openpyxl            (Windows)
pip3 install openpyxl           (Mac)
```

---

3. Run it

Windows - double-click the `.py`, or from a Command Prompt:

```
cd path\to\External-Drive-Catalog\B-Engines
python External-Drive-Catalog_V01.py
```

Mac - from Terminal:

```
cd path/to/External-Drive-Catalog/B-Engines
python3 External-Drive-Catalog_V01.py
```

You can also run it from VS Code with Code Runner, same as your other tools.

---

4. Step-by-step: cataloging your ~20 drives

You do this once per drive. After that the data lives in the catalog forever (until you re-index or delete it).

1. Plug in **one** drive.
2. Open the app and go to the **"1. Index a Drive"** tab.
3. Click **Browse...** and pick the drive root.
   - Windows: the drive letter, e.g. `E:\`
   - Mac: `/Volumes/<DriveName>`
4. The **Disk label** fills in automatically (the volume name on Windows, the `/Volumes` name on Mac). This is how the drive will be named in the catalog. Edit it if you want a cleaner name (e.g. `YN-4T`, `Photos-2019`, `Backup-A`). **Use a unique, consistent label per physical drive** - that label is the key.
5. Click **Start Indexing**. The status line counts files as it goes (every 25,000). A big drive can take a few minutes; the window stays responsive and you can **Cancel** at any time.
6. When it finishes, eject that drive, plug in the next one, and repeat.

Skip hidden / dot-files (checkbox, on by default): leave this on for Mac drives. macOS scatters hidden clutter across external drives - a `._Name` AppleDouble companion next to almost every file, plus `.DS_Store`, `.Spotlight-V100`, `.Trashes`, etc. These are never things you would search for, and the `._*` files alone can nearly double your row count. With the box on, that clutter is skipped (the log tells you how many). The unambiguous junk (`._*`, `.DS_Store`, `Thumbs.db`, system folders) is always skipped; the checkbox only also controls *other* dot-files. Uncheck it for a drive where you actually want hidden files like `.gitignore` catalogued.

Do that for all 20. The **"3. Drives"** tab shows every drive on record with its file count, total size, and when it was last indexed.

Re-indexing: just index a drive again using the **same label**. The old records for that label are replaced cleanly, so the count never doubles up.

---

5. Step-by-step: searching (the payoff)

This works with the drives **unplugged** - you are searching the catalog, not the disks.

1. Go to the **"2. Search"** tab.
2. Fill any combination of:
   - **Name contains** - partial filename, e.g. `invoice` or `Ferdowsi`
   - **Ext** - extension without the dot, e.g. `jpg`, `pdf`, `mp4`
   - **Min MB** - only files at least this big (handy for finding large videos)
3. Click **Search** (or press Enter in the name box).
4. Results show **which drive**, the name, size, modified date, and full path.
5. **Double-click any row to copy its full path** to the clipboard.

Find Duplicates lists files that appear on **two or more drives** with the same name and size - useful before you reclaim space or consolidate the 20 drives down to fewer.

---

6. Step-by-step: exporting

Go to the **"4. Export"** tab. Everything is written into **A-Data** with a timestamped filename (`YYYY-MM-DD--HH-MM`):

- **Export Selected Drive (CSV)** - one drive's file list.
- **Export ENTIRE Catalog (CSV)** - every file on every drive in one file.
- **Export Drive Summary (Excel)** - one row per drive (counts + sizes). Requires `openpyxl`; the button is disabled until it is installed.

All CSVs are written with a UTF-8 BOM so Excel on Windows opens them with Farsi/accented characters intact.

7b. Migrating from your old PostgreSQL database (one time)

If you already have file data in PostgreSQL (the `public.files` table from your old scripts), use `Postgres-To-SQLite-Converter_V01.py` to load it straight into this catalog. It lives in `B-Engines` next to the main app.

1. Install the PostgreSQL driver once:

```
pip install psycopg2-binary        (Windows)
pip3 install psycopg2-binary       (Mac)
```

2. Run the converter (same way you run the main app).
3. Fill in Host / Port / Database / User / Password (it pre-fills from your `PGHOST`/`PGUSER`/etc. environment variables if you set them).
4. Leave **Table** as `public.files` unless yours differs.
5. Click **Test Connection** - it lists the columns it found and shows how it mapped them to `disk_name / file_name / file_path / size / modified`.
    - If your table has no `disk_name` column, type one label in the "Disk label" box and every migrated row gets that label.
6. **Replace** is on by default: it clears any existing catalog rows for the drives being migrated first, so re-running never doubles your counts. Uncheck it to append instead. **Skip macOS/Windows junk rows** is also on by default - it drops `._*`, `.DS_Store`, and `Thumbs.db` rows that your old indexer captured, so the junk does not carry over into the clean catalog.
7. Click **Start Migration**. Progress prints every 250,000 rows; it streams with a server-side cursor so even millions of rows stay light on memory.

When it finishes, open the catalog app's **"3. Drives"** tab to confirm the counts, then you can retire PostgreSQL entirely.

7c. Backups

Your entire catalog is the single file `A-Data/drive-catalog.sqlite3`. To back it up, copy that one file (e.g. onto your YN-4T drive). To move the tool to another computer, copy the whole top folder. That is the whole "database backup" - no server, no dumps.

> You may also see `drive-catalog.sqlite3-wal` and `-shm` files appear while the app is open. Those are normal SQLite working files. Close the app before copying the database and they fold back into the main file.

---

8. Troubleshooting

Mac: nothing happens / "no module named tkinter" The system Python lacks Tk. Install the python.org build (Section 2) and run with `python3`. Test with `python3 -m tkinter`.

Mac: the indexer skips folders or sees nothing on an external drive macOS privacy can block access. Open **System Settings -> Privacy & Security -> Full Disk Access** and add **Terminal** (or VS Code, whichever you launch it from). This is the same permission area that blocked your Photos backup before.

Mac: buttons look gray instead of colored The colors are applied in a cross-platform way, but some macOS themes render buttons with the system style. The function is identical - it is cosmetic only.

Windows: "python is not recognized" Python was not added to PATH during install. Re-run the installer and tick "Add python.exe to PATH", or launch via the Start-menu "Python" entry.

A drive shows fewer files than expected Files that error during scanning (permission denied, locked) are skipped rather than crashing the run. System/hidden files are included; locked OS files may not be.

I want to start completely fresh Close the app and delete `A-Data/drive-catalog.sqlite3` (and the `-wal`/`-shm` files if present). A new empty catalog is created next launch.

What changed from the old scripts

- **One GUI app** instead of three command-line scripts.
- **SQLite** instead of PostgreSQL - zero install, one portable file, no passwords, no `pg_hba.conf`.
- **Indexing writes straight into the database** - the CSV middle-step is gone (CSV is now an *export* option, not a required stage).
- **Cross-platform** - the Windows-only `ctypes` volume lookup is now guarded and there is a Mac `/Volumes` path, with a manual label fallback everywhere.
- **Search across all drives at once**, plus cross-drive duplicate detection.
- **UTF-8 throughout**, ASCII-only console/log messages, your `[DEBUG]` prefixes, timestamped outputs, and the A-Data / B-Engines layout.