Mini-Chapter — Practical Data Storage for Reproducible Analysis

Why Storage Matters

Your models are only as trustworthy as your ability to **reload the exact same data** later. Reproducible storage reduces “it worked on my machine” incidents and helps teammates follow your steps.

Folder Conventions

- `data/raw/` — immutable inputs exactly as acquired. Don’t edit by hand; add new versions with new filenames.
- `data/processed/` — derived tables after cleaning/joins/feature steps. These should be reproducible from code.
- `src/`, `notebooks/`, `README.md` — code, experiments, and documentation.

A clear structure is a contract for your future self and collaborators.

CSV vs Parquet (Tradeoffs)

- **CSV**
  - Pros: human-readable, ubiquitous, easy diffs in git.
  - Cons: larger files, no schema, type loss (dates/numbers as text).
- **Parquet**
  - Pros: columnar, compressed, preserves types, very fast reads; supports efficient querying (column pruning, predicate pushdown).
  - Cons: binary format; requires an engine (`pyarrow`/`fastparquet`).

Rule of thumb: default to CSV for tiny/exchange files; Parquet for analysis-ready tables and anything bigger or repeatedly queried.

Environment-Driven Paths

Hardcoded absolute paths break on other machines. Use a `.env` file:

```
DATA_DIR_RAW=data/raw
DATA_DIR_PROCESSED=data/processed
```

And in Python:

```
from dotenv import load_dotenv
load_dotenv()
from pathlib import Path
import os
```

```
RAW = Path(os.getenv("DATA_DIR_RAW", "data/raw"))
PROC = Path(os.getenv("DATA_DIR_PROCESSED", "data/processed"))
```

Basic IO Patterns (pandas)

```
df.to_csv(RAW / "prices_20240101.csv", index=False)
df.to_parquet(PROC / "prices_20240101.parquet")   # needs pyarrow/fastparquet

df2 = pd.read_csv(RAW / "prices_20240101.csv", parse_dates=['date'])
df3 = pd.read_parquet(PROC / "prices_20240101.parquet")
```

Validate after reload: shapes, key dtypes, and expected columns.

Versioning & Metadata (Preview)

- Timestamped filenames (`prices_YYYYMMDD-HHMM.csv`)
- A small manifest (YAML/JSON) that records source, schema, and code version (optional at this stage)
- For large datasets and teams: data lakes (e.g., S3) with partitioning by date/ticker enable efficient queries (concept only today).

Common Pitfalls

- Saving processed outputs into `data/raw/`
- Silent type drift (dates become strings) when reloading
- Missing Parquet engine causing hidden CSV-only workflows
- Hardcoded OS-specific paths instead of `pathlib`

Takeaway

Reproducibility = (organized folders) × (env-driven paths) × (explicit save/load) × (validation). Master these now to accelerate every later stage.
