1

Git Commands for Version Control in Data Projects (Complete Table)

Category	Git Command	What It Does	Why It Matters in Data Projects
Basics	`git init`	Initialize a new repository	Start version control for ETL scripts, SQL, notebooks
	`git clone <url>`	Clone a remote repo	Standard way to pull shared data repos
	`git status`	Show repo state	Prevents accidental commits of data or configs
	`git add <file>`	Stage a file	Track specific scripts instead of entire folders
	`git add .`	Stage all changes	Quick staging when changes are clean
	`git commit -m "msg"`	Commit changes	Captures pipeline evolution
	`git commit -am "msg"`	Commit tracked files only	Avoids committing new data files accidentally

---

2

Branching and Collaboration

Category	Git Command	What It Does	Why It Matters
Branching	`git branch`	List branches	Track active development streams
	`git branch <name>`	Create branch	Feature isolation for pipelines
	`git checkout <branch>`	Switch branches	Legacy but common
	`git checkout -b <branch>`	Create + switch	Rapid experimentation
	`git switch <branch>`	Switch branches	Safer, clearer alternative
	`git switch -c <branch>`	Create + switch	Preferred modern approach
	`git merge <branch>`	Merge branches	Integrate pipeline changes
	`git merge --no-ff`	Preserve branch history	Helpful for audit trails

3

Remote Operations

Category	Git Command	What It Does	Why It Matters
Remote	`git push`	Push commits	Sync pipelines with team
	`git pull`	Fetch + merge	Risky without review
	`git fetch`	Download changes only	Safer for production branches

4

File Management and Repo Hygiene

Category	Git Command	What It Does	Why It Matters
Cleanup	`.gitignore`	Ignore files	Prevents leaking data files
	`git rm <file>`	Remove tracked file	Clean removals
	`git rm -r --cached <dir>`	Stop tracking folder	Recover from bad commits
	`git clean -fd`	Remove untracked files	Cleans notebook temp files

5

Large Files and Data Artifacts

Category	Git Command	What It Does	Why It Matters
Large Files	`git lfs install`	Enable LFS	Prevents bloated repos
	`git lfs track "*.csv"`	Track large files	Safer dataset handling
	`git lfs track "*.parquet"`	Track parquet	Common in data pipelines
	`git lfs track "*.ipynb"`	Track notebooks	Optional but useful

6

Commit Hygiene (Often Overlooked)

Category	Git Command	What It Does	Why Senior Engineers Use It
Hygiene	`git add -p`	Partial staging	Commit only relevant changes
	`git commit --amend`	Edit last commit	Clean history before PR
	`git rebase -i`	Rewrite commits	Professional commit logs

7

Undo, Restore, and Recovery

Category	Git Command	What It Does	Real-World Data Scenario
Recovery	`git restore <file>`	Restore file	Revert broken notebook
	`git diff --staged`	Review staged changes	Inspect pipeline edits
	`git reset --soft HEAD~1`	Undo commit, keep changes	Fix bad commit message

Category	Git Command	What It Does	Real-World Data Scenario
	`git reset --mixed HEAD~1`	Undo + unstage	Rework changes
	`git reset --hard HEAD~1`	Full discard	Emergency rollback
	`git reflog`	Track HEAD history	Recover lost work

8

Advanced Workflow Control

Category	Git Command	What It Does	Why It’s Powerful
Advanced	`git stash`	Save work temporarily	Notebook experimentation
	`git stash apply/pop`	Restore stash	Resume work later
	`git cherry-pick <hash>`	Apply specific commit	Bugfix reuse
	`git merge --abort`	Cancel merge	Fast conflict exit
	`git diff <b1>..<b2>`	Compare branches	Pre-merge validation

9

Auditing and Debugging (Senior-Level)

Category	Git Command	What It Does	Why It Matters
Audit	`git blame <file>`	Track line ownership	Debug pipeline regressions
	`git bisect`	Find breaking commit	Root-cause failures
	`git log --graph --oneline`	Visual history	Understand branch flow
	`git shortlog -sn`	Contributor summary	Ownership clarity

10

Production and Releases

Category	Git Command	What It Does	Use Case
Releases	`git tag <tag>`	Version release	Track deployed pipelines
	`git archive`	Export snapshot	Deploy stable code
	`git worktree add`	Multiple working trees	Parallel hotfix work
	`git filter-repo`	Rewrite history	Remove leaked data

11

Jupyter and Notebook-Specific

Category	Tool	Purpose	Why It’s Critical
Notebooks	`nbdime diff`	Notebook diffs	Readable comparisons
	`nbdime merge`	Notebook merges	Conflict resolution
	`.gitattributes`	Custom diff rules	Cleaner Git history