Collaboration Assignment · Team Workflow with Git & GitHub

Why this matters
Data-science work is rarely solo. This mini-project shows you how software teams share code smoothly—skills that translate directly to labs, internships, and jobs.

Team Formation

Create teams of 2–3 classmates during lab or in the Canvas "Team Signup" page.
If you are unmatched by **Wednesday 5 PM**, the TA will auto-assign partners.

Objectives

By the end of the assignment, your team will:

1. Use **fork → clone** to start from a common template.
2. **Create a new branch**, add or edit files, and **push** changes.
3. Open a **pull request (PR)** and assign a teammate as reviewer.
4. **Pull**, **merge**, and **resolve** at least one conflict.
5. Produce a clean **commit history** showing contributions from *each* member.

Starter Template

We've posted a small repo with three placeholder files here:

`https://github.com/osuds101/collab-template`

It contains:

```
data/           # CSV already provided
scripts/        # two .py & two .R stubs
README.md       # task list
```

Task Checklist

Step	Who	Tool	Notes
1. **Fork** `collab-template`	*One* member	GitHub Web	Creates team-specific copy.
2. **Add teammates** as collaborators	Owner	Settings → Collaborators	Give push access.
3. **Clone** fork to local machines	Everyone	GitHub Desktop *or* Git Bash	`git clone <fork-url>`
4. **Create feature branch**	Everyone	Desktop: Branch → NewCLI: `git checkout -b feature/<name>`	Work only on your branch.
5. **Edit a file**	Everyone	VS Code / RStudio	Each member modifies **different** file.
6. `add` → `commit` → `push`	Everyone	Desktop or CLI	Use descriptive commit messages.
7. **Open Pull Request**	Everyone	GitHub Web	Assign a teammate to review.
8. **Review & merge** PR	Reviewer	GitHub Web	Use *Squash & merge*.
9. **Pull** merged `main`	Everyone	`git pull origin main`	Keeps local copy up-to-date.

Step	Who	Tool	Notes
10. **Intentional conflict**	Any member	Edit **same line** in `README.md`, push branch, open PR → resolve conflict in web editor.	Demonstrates conflict handling.

What to Submit

- **One** team member submits a short **response file** named `submission.md` (place it in the repo root and push):

```
# Collaboration Assignment Submission
Team Members: Jane Doe (jdoe), Alex Lee (alee)

Repo URL: https://github.com/<team>/collab-template

Conflict file resolved: README.md
```

- Submit the **repo URL** in Canvas (Assignment Website URL).

Grading Rubric (40 pts)

Criterion	Excellent ( )	Pts
Repo forked & public	URL opens; template structure intact	5
Each member has 1 commit	`git log --author` shows activity	10
Feature branch used & PR merged	Branch name seen; PR conversation in "Closed" tab	5
Conflict demonstrated & resolved	Merge commit or "Resolved via web editor" visible	10
Repo clones without errors	TA can `git clone` successfully	5
Commit messages clear	Messages describe change, not "update"	5

Total = 40 points

Hints & Troubleshooting

- **Pull first. Push later.** 90 % of merge headaches vanish if you stay updated.
- **PR reviewers**: leave at least *one* comment (even "LGTM") before merging.
- If PAT prompts repeat on Windows, open **Git Credential Manager** and delete stale entries, then push again.
- Stuck? Post in **#git-help** Slack channel—include error text or screenshot.

Due Date

Sunday 11:59 PM (Pacific) — Week 3.

Late work: −10 % per day, up to 3 days.

Submissions after 3 days will be marked as 0.

When you finish, you'll have practiced the exact workflow used by countless open-source and professional teams.