Git Advanced Commands Exercise

Scenario 1: Collaborating on Feature Branches

You're working on a team project where multiple features are being developed simultaneously. This will help with branching, merging, and conflict resolution.

Exercise:

1. **Create a New Branch:** Create a branch for a feature (e.g., feature/login).
2. git checkout -b feature/login
3. **Make Some Changes:** Add files and make a few commits on this branch.
4. echo "Login feature" >> login.txt
5. git add login.txt
6. git commit -m "Add login feature"
7. **Simulate a Colleague's Changes:** Switch to main and create another feature branch, feature/dashboard. Make changes and merge feature/dashboard into main.
8. **Merge Main into Your Branch:** While on feature/login, merge the latest main branch changes into it. Resolve any conflicts.
9. git merge main
10. **Advanced Merge Conflicts:** Try using git mergetool to open a merge tool to resolve conflicts.

Scenario 2: Interactive Rebase for Clean Commit History

You have a series of commits on a feature branch that you want to clean up before merging to main.

Exercise:

1. **Create a Series of Commits:** Make multiple small, related commits on a branch (e.g., feature/refactor).
2. **Interactive Rebase:** Start an interactive rebase to squash, reorder, or edit commits.
3. git rebase -i HEAD~3

4. **Resolve Any Conflicts During Rebase:** If there are conflicts, resolve them and use git rebase --continue.

Scenario 3: Cherry-picking Commits

You're developing a hotfix and need to apply some specific commits from a feature branch to main.

Exercise:

1. **Create Commits on a Feature Branch:** Create a branch feature/hotfix and add a few commits.
2. **Cherry-pick Specific Commits:** Switch to main and use git cherry-pick to bring in specific commits from feature/hotfix.
3. git cherry-pick <commit-hash>
4. **Handle Conflicts During Cherry-picking:** Resolve any conflicts that arise.

Scenario 4: Stashing Work and Applying Stash

You need to switch tasks quickly but don't want to commit half-done work.

Exercise:

1. **Make Changes Without Committing:** Modify files but don't commit them.
2. **Stash the Changes:** Use git stash to temporarily set changes aside.
3. git stash
4. **List, Apply, and Drop Stashes:** List your stashes, apply one, and then remove it from the stash.
5. git stash list
6. git stash apply stash@{0}
7. git stash drop stash@{0}

Scenario 5: Bisecting to Find a Bug

A bug was introduced, but it’s unclear which commit caused it.

Exercise:

1. **Use Git Bisect:** Start with git bisect and mark a known good and bad commit to find where the bug was introduced.
2. git bisect start
3. git bisect bad HEAD
4. git bisect good <commit-hash>
5. **Mark Commits During Bisect:** As you go through commits, mark them as good or bad until Git identifies the problematic commit.

Scenario 6: Advanced Conflict Resolution in Rebase and Merge

You have complex changes in both main and feature/advanced that require careful conflict resolution.

Exercise:

1. **Simulate a Conflict:** Create and merge branches that will cause conflicts.
2. **Resolve Using Different Strategies:** Experiment with git rebase and git merge --strategy-option flags, and try using git rerere to record and reuse resolutions.