Github Project: Git Commands Documentation Template

Programming for Data Science Nanodegree Program

In this project, I show the commands that I used to create a GitHub repository and your local repository. The commands that I wrote on a repo for my BikeShare project, made multiple branches, made edits to files on those branches (e.g., a readme file and my Python code), and then I have committed the changes to your remote repository.

This report is my submission for the third project of my nano degree.

Contents

GitHub Repository Link
https://github.com/sondosaabed/Nano-Degree-3rd-Project

1. Setting Up My Repository

In the following steps I create my Git repository, add my Python code, and post my files on GitHub.

	Tasks	Git Commands
A.	Clone the GitHub repository to your local repository.	$ git clone https://github.com/sondosaabed/Nano-Degree-3rd-Project
B.	Move your bikeshare.py and data files into your local repository.	
C.	Create a .gitignore file containing the name of your data file.	
D.	List the file names associated with the data files you added to your .gitignore	
E.	Check the status of your files to make sure your files are not being tracked	$ git status
F.	Stage your changes.	$ git add
G.	Commit your changes with a descriptive message.	$ git commit -m “ignore data & add bikeshare.py”
H.	Push your commit to your remote repository.	$ git push origin master

2. Improve Documentation

In this section, I am working in my local repository, on the BikeShare python file and the README.md file. I have repeated steps C through E three times to make at least three commits as I am working on my documentation improvements.

	Tasks	**Git Commands**
A.	Create a branch named *documentation* on your local repository.	$ git branch documentation
B.	Switch to the *documentation* branch.	$ git checkout documentation
C.	Update your README.md file.	
D.	Stage your changes.	$ git status $ git add
E.	Commit your work with a descriptive message.	$ git commit -m "update README.md"
F.	Push your commit to your remote repository branch.	$ git push origin documentation
G.	Switch back to the master b ranch.	$ git checkout master $ git brach

3. Additional Changes to Documentation

	Tasks	**Git Commands**
A.	Switch to the *documentation* branch.	$ git checkout documentation $ git branch
B.	Make at least 2 additional changes to the documentation	Changed README.md Changed Explore BikeShare.py

C.	After each change, stage and commit your changes. When you commit your work, you should use a descriptive message of the changes made. Your changes should be small and aligned with your commit message.	$ git add README.md
		$ git commit -m "Update README.md"
		$ git add '.\Explore BikeShare.py'
		$ git commit -m "Edit imports of Explore Bikeshare.py"

D.	Push your changes to the remote repository branch.	$ git push origin documentation
E.	Switch back to the *master* branch.	$ git checkout master $ git branch
F.	Check the local repository log to see how ***all the branches*** have changed.	$ git log $ git log --oneline --graph --all
G.	Go to Github. Notice that you now have two branches available for your project, and when you change branches the README changes.	

4. Refactor Code

In this section, I'm working in my local repository, on the code in my BikeShare python file to make improvements to its efficiency and readability. I have repeated steps C - E 3 times to make at least 3 commits as I refactor.

	Tasks	**Git Commands**
A.	Create a branch named *refactoring* on your local repository.	$ git branch refactoring
B.	Switch to the *refactoring* branch.	$ git checkout refactoring $ git branch
C.	Similar to the process you used in making the documentation changes, make 2 or more changes in refactoring your code.	

D.	*For each change,* stage and commit your work with a descriptive message of the changes made.	$ git add README.md
		$ git commit -m "Update README.md"
		$ git add '.\Explore BikeShare.py'
		$ git commit -m "Edit comments spelling on code"

E.	Push your commits to your remote repository branch.	$ git push origin refactoring
F.	Switch back to the *master* branch.	$ git checkout master
G.	Check the local repository log to see how *all the branches* have changed.	$ git log $ git log --oneline --graph --all
H.	Go to GitHub. Notice that I have 3 branches. Notice the files change.	

5. Merge Branches

	Tasks	**Git Commands**
A.	Switch to the *master* branch.	$ git checkout master
B.	Pull the changes you and your coworkers might have made in the passing days (in this case, you won't have any updates, but pulling changes is often the first thing you do each day).	$ git fetch origin $ git status

C.	Since your changes are all ready to go, merge all the branches into the master. Address any merge conflicts. If you split up your work among your branches correctly, you should have no merge conflicts.	$ git branch (on master)
		$ git merge documentation refactoring
		— I had a conflict on readme.md
		– I have resolved it using the editor
		$ git status
		$ git commit -m "merge branches"

D.	You should see a message that shows the changes to the files, insertions, and deletions.	
E.	Push the repository to your remote repository.	$ git push origin master
F.	Go to GitHub. Notice that your master branch has all of the changes.	