Github Project:

Git Commands Documentation

Programming for Data Science Nanodegree Program

INTRODUCTION:

We will have to copy and paste the git commands that we used to complete all tasks on our local and remote git repository for this project. This file will serve as our submission for the GitHub project.

1.Set Up Your Repository

The following are the steps you will take to create your git repository, add your python code, and post your files on GitHub.

Step 1. Create a GitHub profile (if you don't already have one).

So, I have a Github account and no need to create new one.

Github Profile Link:
https://github.com/kpkrishan

Step 2. Fork a repository from Udacity's GitHub Project repository and provide a link to your forked GitHub repository here:

Forked Repository Link:
https://github.com/kpkrishan/pdsnd_github

Step 3. Complete the tasks outlined in the table below and copy and paste your git commands into the "Git Commands" column. The first git command is partially filled out for you.

	Tasks	**Git Commands**
A.	Clone the GitHub repository to your local repository.	git clone 'https://github.com/kpkrishan/pdsnd_github.git'
B.	Move your bikeshare.py and data files into your local repository.	**No git command needed (you can use cp or a GUI)**
C.	Create a .gitignore file containing the name of your data file.	touch .gitignore
D.	List the file names associated with the data files you added to your .gitignore	**No git command needed (add the file names into your .gitignore file)**

E.	Check the status of your files to make sure your files are not being tracked	git status
F.	Stage your changes.	git add ‘bikeshare.py’
G.	Commit your changes with a descriptive message.	git commit -m "Initiate project with bikeshare.py file”
H.	Push your commit to your remote repository.	git push -u origin master

2.Improve Documentation

Now you will be working in your local repository, on the BikeShare python file and the README.md file. You should repeat steps C through E three times to make at least three commits as you work on your documentation improvement.

	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.	**No git command needed (edit the** **text in your README.md file)**
D.	Stage your changes.	git add README.md
E.	Commit your work with a descriptive message.	git commit -m "docs: Update README.md file ”
F.	Push your commit to your remote repository branch.	git push origin documentation
G.	Switch back to the master branch.	git checkout master

3. Additional Changes to Documentation

In a real world situation, you or other members of your team would likely be making other changes to documentation on the documentation branch. To simulate this follow the tasks below.

	Tasks	Git Commands
A.	Switch to the *documentation* branch.	git checkout documentation
B.	Make at least 2 additional changes to the documentation - this might be additional changes to the README or changes to the document strings and line comments of the bikeshare file.	1. $ git diff **diff --git a/README.md b/README.md** **index 9d3988f..e6413c7 100644** **--- a/README.md** **+++ b/README.md** @@ -4,7 +4,7 @@ ### Project Title Github:Udacity-Python-For_Data-Science-Nanodegree-Program -### Description +#### Description This Project is all about the learning of version control using the previous Project In the previous project we did ananlysis of three files of bikeshare

```
data.

 And using those
files and output
file we will be
working on version
control i.e Github

@@ -13,21 +13,16 @@
on github
repository.



 ### Files used

-In this project I
dealt with the
previous project
files and names of
file are given
below:

-

-Name:



2. $ git diff

diff --git
a/README.md
b/README.md

index
e6413c7..b62a320
100644

--- a/README.md

+++ b/README.md

@@ -5,12 +5,25 @@


Github:Udacity-Pyth
on-For_Data-Science
-Nanodegree-Program


 #### Description

-This Project is all
about the learning
of version control
```

		using the previous Project +This Project is all about the learning of version control using the Bikeshare Project. +### Project Overview +In this project, you will make use of Python to explore data related to bike share systems for three major cities in the United States—Chicago, New York City, and Washington… . . . +A terminal application (Terminal on Mac and Linux or Cygwin on Windows). 3. For third modification in README.md "The headers are modified" I did not mention the output from '$ git diff' command, to make the git command output section clean and short. we can check it through git log --oneline --graph --all Command.

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.	1. git add README.md git commit -m "docs: Change README.md documentation to improve readability" 2. git add README.md $ git commit -m "docs: Add Project overview in README.md as new subheading." 3. git add "README.md" git commit -m "docs: Change heading and subheading font size."
D.	Push your changes to the remote repository branch.	git push origin documentation
E.	Switch back to the *master* branch.	git checkout master
F.	Check the local repository log to see how *all the branches* have changed.	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.	**No git command needed**

4. Refactor Code

Now you will be working in your local repository, on the code in your BikeShare python file to make improvements to its efficiency and readability. You should repeat steps C through E three times to make at least three commits as you refactor.

	Tasks	**Git commands**
A	Create a branch named *refactoring* on your local repository.	git checkout -b refactoring
B	Switch to the *refactoring* branch.	git checkout -b refactoring
C	Similar to the process you used in making the documentation changes, make 2 or more changes in refactoring your code.	**No git command needed (edit the code in your python file)**

D	*For each change,* stage and commit your work with a descriptive message of the changes made.	git add “bikeshare.py”
		`$ git commit -m "docs: Add comment to understand main function."`
		
		git add “bikeshare.py”
		`$ git commit -m "docs: Add comment to improve documentation."`
		
		git add “bikeshare.py”
		`$ git commit -m "style: Add`

		`missing parenthesis"`
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 --graph --all --oneline
H	Go to GitHub. Notice that you now have 3 branches. Notice how the files change as you move through the branches.	**No git command needed**

5. Merge Branches

	Tasks	Git commands
A	Switch to the *master* branch.	git checkout master
B	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 pull origin
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 merge refactoring git merge documentation
D	You should see a message that shows the changes to the files, insertions, and deletions.	**No git command needed**
E	Push the repository to your remote repository.	git push origin
F	Go to GitHub. Notice that	**No git command needed**

	your master branch has all of the changes.	

Submission:

This concludes the project.

Please review this document to make sure you entered all the required response fields in all four sections.

Download this document as a PDF file.

Submit the PDF file on the Project Submission.