# DSI: Unix Shell, Git and GitHub

## Assignment 2 & Quiz: Git and GitHub

### Part 1

Part 1 of Assignment 2 is a quiz. Please complete to the best of your ability. Notes are permitted. Please email your responses to the TA with Instructor CC'd.

1. Check all that are TRUE about version control:

**True** - [ ] Can revert files to a previous state

**True** - [ ] Can compare changes over time

**True** - [ ] Can see who modified something last

**True** - [ ] Can recover lost files

2. What is the difference between centralized version control systems and distributed version control systems?

```

**Your answer here...**

In a centralized version control, everything is stored to a central server, access to which is limited, which eventually makes it difficult to work collaboratively.

In a central version control system, developers clone a working copy of the entire repository from the server and changes are directly committed to the server. This raises the risk of dependency over the central server i.e. if the central server goes down then many operations will be unavailable.

While in Distributed Server offers more flexibility and opportunity to collaborate, as developers can work independently, make changes and commit on local repository, and push local repository later.

```

3. What are the three states that files can reside in?

- [ ] a) committed, changed, waiting

- [ ] b) saved, changed, staged

**True** - [ ] c) committed, modified, staged

- [ ] d) saved, modified, staged

4. What command initializes a new repository?

- [ ] a) `git clone`

- [ ] b) `git branch`

- [ ] c) `git fork`

**True** - [ ] d) `git init`

5. What does `git diff` do?

- [ ] a) compares the differences between the home directory and staging area

- [ ] b) compares the differences between the working directory and staging area

**True** - [ ] c) compares the differences between the working directory and what’s been committed

- [ ] d) compares the differences between the staging area and what’s been committed

6. How do you add a message to your commit? (select all that apply)

**True** - [ ] a) `git commit -m`

- [ ] b) `git commit --messages`

- [ ] c) `git commit`

- [ ] d) `git commit -message`

7. How do you add a remote repo? (select all that apply)

**True** - [ ] a) `git remote add`

- [ ] b) `git add remote`

- [ ] c) `git clone`

- [ ] d) `git add clone`

8. How do you add a remote repo? (select all that apply)

**True** - [ ] a) `git remote`

- [ ] b) `git add remote`

- [ ] c) `git clone`

- [ ] d) `git add clone`

9. What is the difference between `git pull` and `git fetch`?

```

Your answer here...

Git “Fetch” is a command used to fetch / review changes initiated on remote repository. However, Git “Pull” is kind of a combination of Git “Fetch” & Git “Merge”. It does not only fetch changes from the remote repository but also update the local repository accordingly.

```

9. How do you switch branches?

**True** - [ ] a) `git checkout`

- [ ] b) `git checkout -b`

- [ ] c) `git branch -c`

- [ ] d) `git branch`

10. Why are messages important? What would make a good commit message?

```

Your answer here...

Messages plays a very crucial role as it provides details of the code / change / etc. A “Commit Message” should be detailed enough to communicate necessary information about the committed batch, so the reviewer / collaborator / user does not need to spend reading all of the code / changes to understand the motive.

```

11. Please correct the merge shown below (both codes are suitable, neither has errors):

```

<<<<<<< HEAD

df.loc[df['sex'] == 'f', 'age'].mean()

=======

df.loc[df['sex'] == 'm', 'age'].mean()

>>>>>>> branch\_1

```

```

**Your corrected merge here...**

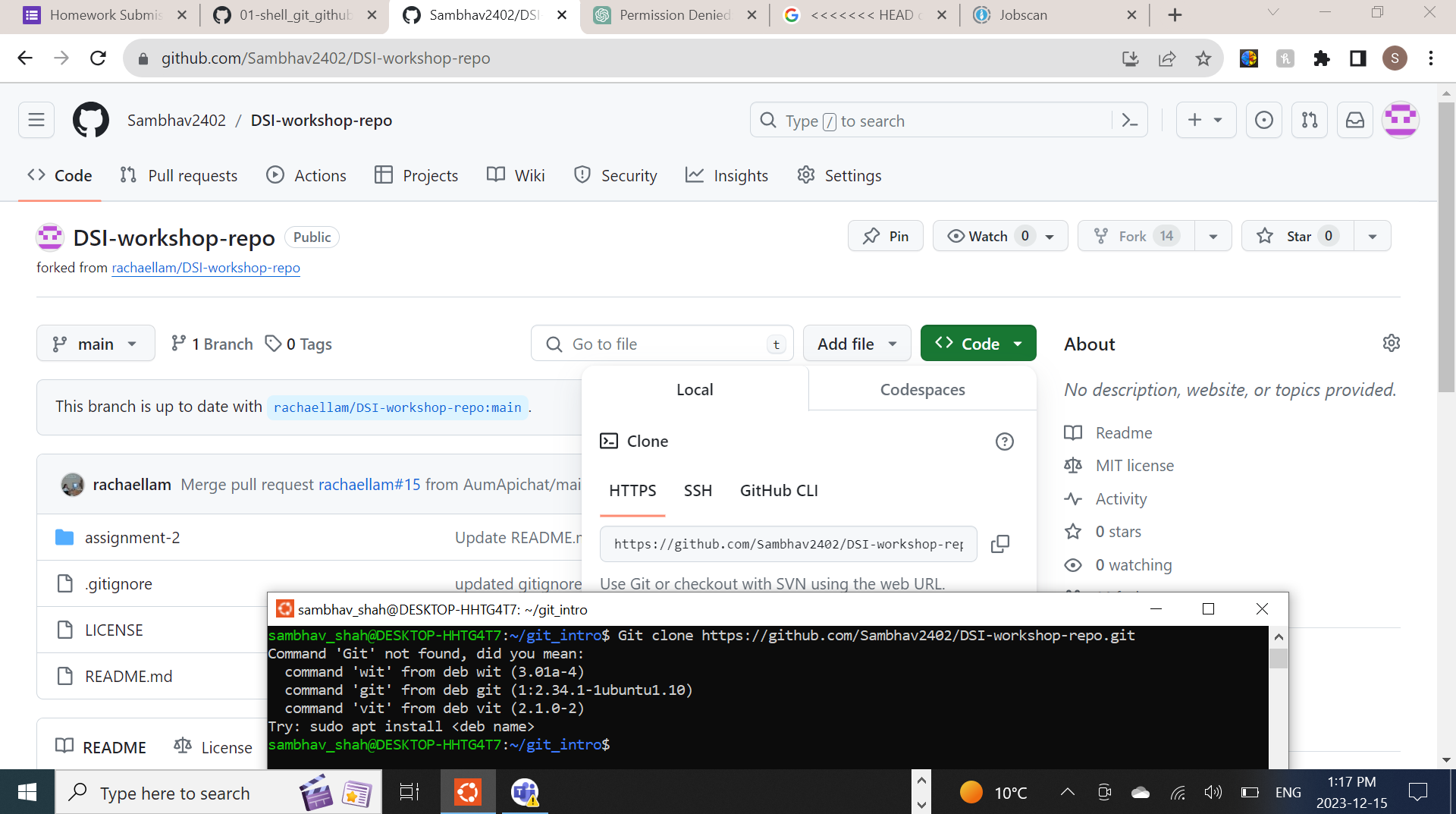
**df.loc[df['sex'] == 'f', 'age'].mean()**

**df.loc[df['sex'] == 'm', 'age'].mean()**

```

# Part 2

1. `fork` and `clone` [this class GitHub repo](<https://github.com/rachaellam/DSI-workshop-repo>).



2. `push` your Assignment 1 to the folder labelled "assignment-2." Your additions should include...

- All components necessary to run Assignment 1

- Proper folder structure (inputs, outputs, scripts)

- A README.md file. The README should include components discussed in the workshop. Feel free to research good READMEs and add anything that you believe will add value to your README

3. Create a `pull request` to add your additions to the class repo.

## Rubric:

| Component | 1 | 2 | 3 | 4 | 5 |

| --------------------------------------------------------------------------------------------------------------------------- | --- | --- | --- | --- | --- |

| Repo contains all necessary components to run Shell script and has the correct folder structure | | | | | |

| README is comprehensive and includes components discussed in class plus at least one component learned from outside sources | | | | | |

| Pull request has been successfully requested without any merge errors | | | | | |

\*\*Total:\*\* /15

\*\*Quiz Total:\*\* /11

\*\*Final:\*\* /26