CS108 Project - Bash Grader

Abhineet Majety (Roll no: 23B0923)

April 28, 2024

Contents

1 Overview **2**

2 Features and Working of Code **2**

2.1 csv File Manager 2

2.2 Version Control System 3

3 Customization **5**

4 Demonstration **7**

4.1 csv File Manager 7

4.1.1 combine 7

4.1.2 upload 8

4.1.3 total 9

4.1.4 update 9

4.1.5 stats 9

4.1.6 graph 10

4.1.7 view 11

4.2 Version Control 11

4.2.1 git_init 11

4.2.2 git_commit 12

4.2.3 git_checkout and git_log 12

4.2.4 git_checkout -b and git_branch 13

5 Modularisation **14**

1 Overview

The objective of this project is to create a csv File Manager and Interpreter, and a basic Version Control System. The bash file `submission.sh` implements the functions. Most of the program files are written in bash script, while some use awk and python.

2 Features and Working of Code

The code for each utility is present in a different file. `submission.sh` contains commands to execute the utilities based on the arguments given to it.
The various commands can be run using:
"`bash submission.sh <command> [option] [argument(s)]`".

2.1 csv File Manager

The csv file manager maintains the records of marks scored by different students in different exams. Individual files in the folder contain the marks of various students in a particular exam. Each file should have the three columns in the format `<Roll_Number,Name,Marks>`. A command is provided to combine all the data into a single file. The code has commands to update marks, upload new files, find various statistics, etc.
The basic commands are described below:

1. `combine`
   Combines all the csv files present in the working directory to a single `main.csv`.
   If we have two files quiz1.csv and midsem.csv, then main.csv will have the columns `<Roll_Number,Name,midsem,quiz1>`. Files are ordered lexicographically.
   If `combine` was already run before, then `main.csv` will not be made afresh. Instead, the files which were added after the previous combine will be appended to main.csv.
   The list of all files newly combined is printed to standard output.

   - `combine -f <files_list>`
     To define the exams and order of exams in main.csv, a file containing list of csv files must be created, one file per line. Then, this command can be run with this file as an argument to give the desired order.

   The code handles the following cases:

   - If `total` command was run before, the total column is updated by itself.
   - If a person is absent in a particular exam, the entry in main.csv is marked "a".
   - If combine -f is used and a file in the list is not present, it is indicated in the output.
   - If a file is in improper format, it is indicated in the output.

2. `upload <file_path>`
   Copies the file from the given path. When `combine` is run again, details of this file are also updated.
   The code handles the following cases:

   - When the given file is not present, an error message is displayed.
   - When a file with the same name is present in the current directory, it asks for confirmation whether to replace the existing file. If replacement is done, the existing main.csv must be removed and combined again to see the changes.

3. `total`
   Inserts a "total" column at the end of main.csv. The code handles various cases such as:
   - If main.csv is not present, an error message is given.
   - If main.csv already has a "total" column, it asks for confirmation whether to redo it.

4. `update`
   Updates main.csv and the corresponding file with updated marks. User input of exam name, roll number and marks is taken. If main.csv has a total column, that is updated too. If a student is absent for an exam, updating the student's marks for the exam creates a new record for the student in the exam file. The code gives error messages if the file or roll number does not exist.
   If main.csv is not present, combine needs to be run first.

5. `stats <file>`
   Calculates some statistics, i.e., number of students, mean, median, third quartile and standard deviation of the marks in the argument file. If main.csv is the file, total command must be run first. If main.csv does not have a total column, an error message is printed.

6. `graph <file>`
   Displays a frequency-marks graph. Takes the maximum and minimum possible scores in the exam as input. If main.csv does not have a total column, an error message is printed. The graph is generated using the python `matplotlib` library.

7. `view <roll number>`
   A roll number should be given as input to this command. For each column in main.csv, the column heading and the corresponding value for the given roll number is printed. If the roll number is not present in main.csv, an error message is printed. If main.csv is not present, an error message is printed to run combine command first.

2.2 Version Control System

A Version Control System (VCS) is used to manage changes in a directory. Git is a popular VCS. This is a smaller version of git, which tracks the entire working directory. The details of features are given below:

1. `git_init <folder_path>`
   The path of the intended VCS repository (a directory) should be given as an argument. If the directory is not already present, a directory is created by the command in the given path along with the required structure. A file named `git_config.txt` is created in the working directory. This file stores the repository location.
   The repository structure created is mentioned below:
   - `commits`: This directory will have subdirectories, each of which will store the details of a commit.
   - `commits/master`: This directory will be used to store the head of master branch during `git_checkout`.
   - `branches`: This directory will have a file for each branch. Each filename is the name of a branch. The file with a branch name will store hashes of all the commits of the branch in chronological order.
   - `branches/master.txt`: This file will store all the commit hashes of master branch, in chronological order.
   - `git_log`: This directory will have a log file for each branch. Each filename is the name of a branch. The file with a branch name will store the commit log of the branch.

- `git_log/master.txt`: This file will store the commit log of master branch.
- `current_head.txt`: This file stores data of the branch we are working in and the exact commit at which the working directory is checked out.

2. `git_commit`
   This command is used to commit the working directory.
   On running this command, the user is prompted to enter a commit message. The message can be mentioned as an argument by running "`git_commit -m <message>`".
   `git_commit` is only allowed at the end of a branch. If the user tries this command from any other commit of the branch, an error message is printed asking the user to create a branch from there first.
   A 16-digit random number hash is created for the commit using `git_hash.sh`.
   A directory with the hash as its name is created in the "commits" directory of the repository. The structure of this directory is detailed below:

   - `changes`: This directory contains files. For each file in the working directory, a file with the same name is present in this directory.
   - `message.txt`: This file stores the commit message.
   - `branch.txt`: This file stores the branch(es) to which the commit belongs. The first entry in this file is the branch in which the commit was created. The other branches start from this commit.
   - `prev_commit.txt`: This file contains the hash of the previous commit.

   Each of the files in `commits/changes` stores the changes in the file from the previous commit.This is implemented using diff and patch commands using `git_patch.sh`. If the file was not present in the previous commit, the entire file is copied from the working directory.

3. `git_checkout <hash>`
   `<hash>` is the first few digits of the hash of the commit to which the user wants to check out. Typing the hash can be avoided by running "`git_checkout -m <message>`". If the hash or message have multiple or no matching commits, suitable error messages are printed.
   If the working directory is at the end of a branch "b", the entire directory is copied to `commits/b`. If it is not at the end, the user is warned that any changes made will be lost, and is asked for confirmation.
   The desired commit is created using the patch command.

4. `git_checkout -b <branch>`
   `<branch>` is the name of the branch to which the user wants to checkout to. If the branch does not already exist, a new branch is created from the current commit after confirmation. If the branch exists, the working directory is checked out to the head of the branch. (stored in `commits/<branch>`).

5. `git_log`
   The contents of the git log file of the current branch till the current commit is printed. For each commit, the hash and the commit message are printed.

6. `git_branch`
   A list of all the branches is printed. The current branch is indicated by "***" at the end.

3 Customization

Various customizations are included, which are as follows:

1. **combine and combine -f:**
   Combine -f option gives the user an option to define the exams present in main.csv and the order of exams. If the filename of the list of files given as argument does not exist, a close match is suggested if present.
   If main.csv has a total column, it is auto-updated if combine is used again.

2. **upload:**
   If a file in the directory has the same name as the input file, user confirmation is requested whether to replace it or not.

3. **update:**
   The total column in main.csv is updated automatically. If a student is absent for an exam, updating the student's marks for the exam creates a new record for the student in the exam file after user confirmation.

4. **Statistics and Graphs:**
   `stats <file>` command takes the name of a csv file as input (it can be main.csv or any other marks file) and prints statistics, i.e., number of students, mean, median, third quartile and standard deviation of the marks in the argument file. If main.csv is the file, total command must be run first.

5. **View a student:**
   `view <roll number>` command takes a roll number as an argument. For each column in main.csv, the column heading and the corresponding value for the given roll number is printed. If the roll number is not present in main.csv, an error message is printed. `combine` command must be run before running this command.

6. **Git commit:**
   The user is prompted for a commit message if "commit -m message" is not run. At each commit, only the changes over the previous commit are stored.

7. **Git log:**
   `git_log` command prints the commit log of the current branch until the current commit. For each commit, the hash and the commit message are printed. Commits are shown in reverse-chronological order, i.e., the latest commit is shown first.

8. **Saving uncommited changes during git checkout:**
   If checkout is done from the head of a branch, the uncommited changes are also retained. If it is done from intermediary commits of a branch, checkout is done after user confirmation.

9. **Git Branching:**
   `git_checkout -b <branch>` command takes a branch name as argument. If the branch does not already exist, a new branch is created from the current commit after confirmation. If the branch exists, the working directory is checked out to the head of the branch.
   `git_branch` command prints a list of all branches. The current branch is indicated by "***" at the end.

10. **Error Handling and Suggestions**
    The code handles several cases of usage of commands. In some cases, when the argument does not have an exact match among the list of possible arguments, a suggestion of closest match is provided. This is implemented using `suggestions.sh` which uses the python `difflib` library. The cases handled by each command are broadly listed below :

- If a command is typed wrong, an error message is given along with a suggestion for closest match.
- For each command, wrong format of arguments gives an error message stating the correct usage.
- When combine -f <files_list> is executed, if the argument file does not exist, an error message is given along with a suggestion. If a file in the list is not present in the directory, it is indicated in the output as shown in figure 2.
- If a csv file in the directory is not in the correct format, it is indicated in the output of combine command.
- When upload command is run, if the file mentioned is not present in the mentioned path, an error message is printed with suggestions for file name.
- If update, total, stats, graph or view commands are run without main.csv, an error message is printed asking to run combine first.
- If stats and graph commands are run without total column in main.csv, an error message is printed.
- If git commit, checkout, log or branch commands are run without running git_init, an error message is printed.
- If git_commit is run from intermediary commits of a branch, an error message is given asking to create a new branch from there.
- If the part of hash given to git_checkout has multiple matches, an error message is printed asking for more digits. If there is no match, an error message is printed. If the commit message given to git_checkout -m is the same for multiple commits or does not have any commit, an error message is printed.
- If git_checkout or git_checkout -b is tried without a single commit, an error message is printed.

4 Demonstration

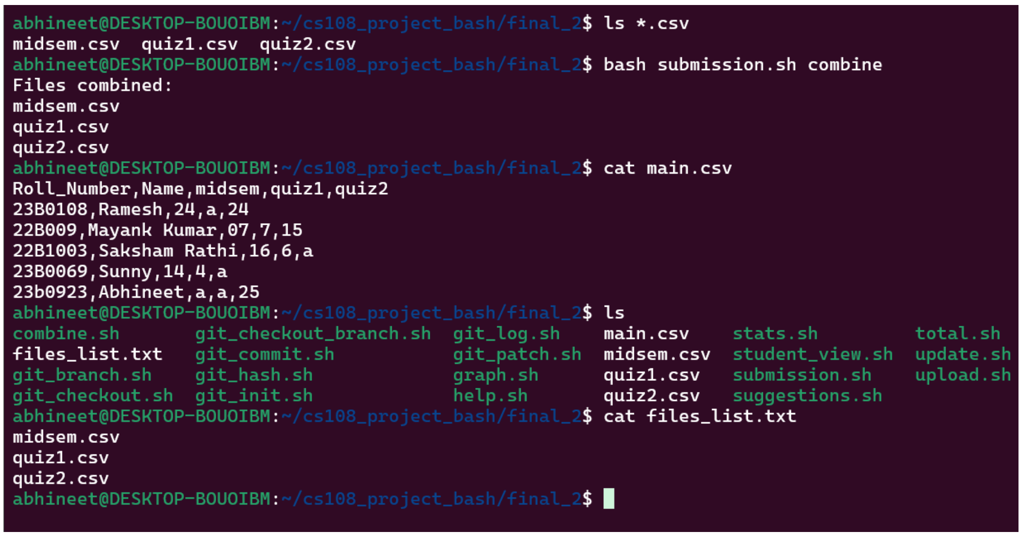
In this section, execution of each command in various cases is demonstrated.

4.1 csv File Manager

4.1.1 combine

The directory has quiz1.csv, midsem.csv and quiz2.csv.

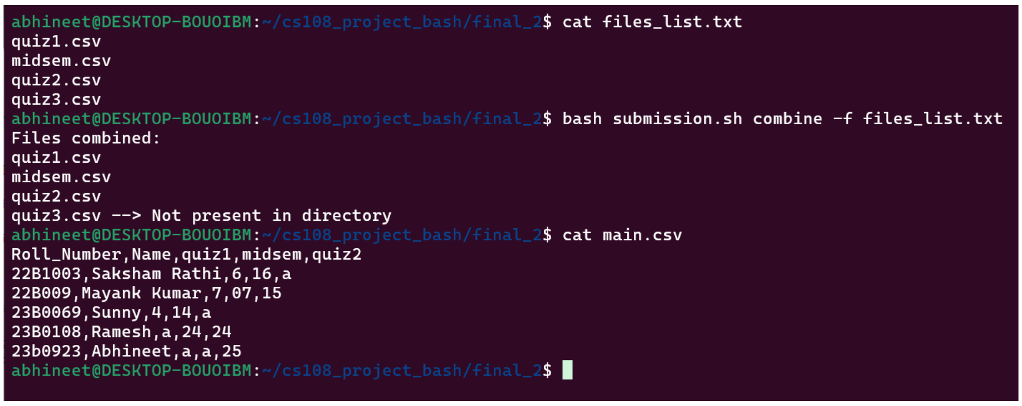
- Command: `bash submission.sh combine`
  Output: Shown below (Files are ordered lexicographically.)



```
abhineet@DESKTOP-BOUOIBM:~/cs108_project_bash/final_2$ ls *.csv
midsem.csv  quiz1.csv  quiz2.csv
abhineet@DESKTOP-BOUOIBM:~/cs108_project_bash/final_2$ bash submission.sh combine
Files combined:
midsem.csv
quiz1.csv
quiz2.csv
abhineet@DESKTOP-BOUOIBM:~/cs108_project_bash/final_2$ cat main.csv
Roll_Number,Name,midsem,quiz1,quiz2
23B0108,Ramesh,24,a,24
22B009,Mayank Kumar,07,7,15
22B1003,Saksham Rathi,16,6,a
23B0069,Sunny,14,4,a
23b0923,Abhineet,a,a,25
abhineet@DESKTOP-BOUOIBM:~/cs108_project_bash/final_2$ ls
combine.sh       git_checkout_branch.sh  git_log.sh    main.csv    stats.sh         total.sh
files_list.txt   git_commit.sh           git_patch.sh  midsem.csv  student_view.sh  update.sh
git_branch.sh    git_hash.sh             graph.sh      quiz1.csv   submission.sh    upload.sh
git_checkout.sh  git_init.sh             help.sh       quiz2.csv   suggestions.sh
abhineet@DESKTOP-BOUOIBM:~/cs108_project_bash/final_2$ cat files_list.txt
midsem.csv
quiz1.csv
quiz2.csv
abhineet@DESKTOP-BOUOIBM:~/cs108_project_bash/final_2$ 
```

Figure 1: combine

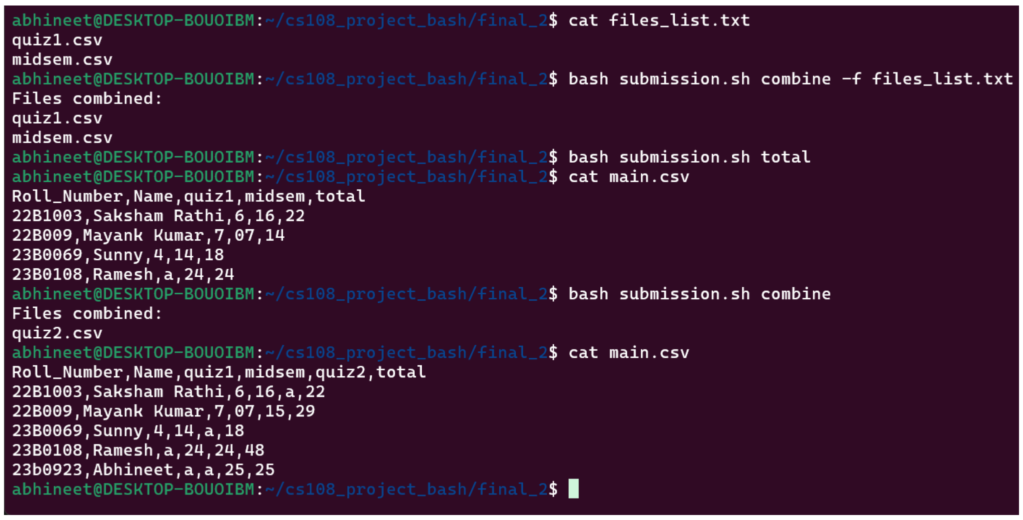
- Command: `rm main.csv; bash submission.sh combine -f files_list.txt`
  Output: (The directory only has quiz1.csv, midsem.csv and quiz2.csv.)



```
abhineet@DESKTOP-BOUOIBM:~/cs108_project_bash/final_2$ cat files_list.txt
quiz1.csv
midsem.csv
quiz2.csv
quiz3.csv
abhineet@DESKTOP-BOUOIBM:~/cs108_project_bash/final_2$ bash submission.sh combine -f files_list.txt
Files combined:
quiz1.csv
midsem.csv
quiz2.csv
quiz3.csv --> Not present in directory
abhineet@DESKTOP-BOUOIBM:~/cs108_project_bash/final_2$ cat main.csv
Roll_Number,Name,quiz1,midsem,quiz2
22B1003,Saksham Rathi,6,16,a
22B009,Mayank Kumar,7,07,15
23B0069,Sunny,4,14,a
23B0108,Ramesh,a,24,24
23b0923,Abhineet,a,a,25
abhineet@DESKTOP-BOUOIBM:~/cs108_project_bash/final_2$ 
```

Figure 2: combine -f

- If a new column is added in main.csv after running `total` command, the total gets updated.



```
abhineet@DESKTOP-BOUOIBM:~/cs108_project_bash/final_2$ cat files_list.txt
quiz1.csv
midsem.csv
abhineet@DESKTOP-BOUOIBM:~/cs108_project_bash/final_2$ bash submission.sh combine -f files_list.txt
Files combined:
quiz1.csv
midsem.csv
abhineet@DESKTOP-BOUOIBM:~/cs108_project_bash/final_2$ bash submission.sh total
abhineet@DESKTOP-BOUOIBM:~/cs108_project_bash/final_2$ cat main.csv
Roll_Number,Name,quiz1,midsem,total
22B1003,Saksham Rathi,6,16,22
22B009,Mayank Kumar,7,07,14
23B0069,Sunny,4,14,18
23B0108,Ramesh,a,24,24
abhineet@DESKTOP-BOUOIBM:~/cs108_project_bash/final_2$ bash submission.sh combine
Files combined:
quiz2.csv
abhineet@DESKTOP-BOUOIBM:~/cs108_project_bash/final_2$ cat main.csv
Roll_Number,Name,quiz1,midsem,quiz2,total
22B1003,Saksham Rathi,6,16,a,22
22B009,Mayank Kumar,7,07,15,29
23B0069,Sunny,4,14,a,18
23B0108,Ramesh,a,24,24,48
23b0923,Abhineet,a,a,25,25
abhineet@DESKTOP-BOUOIBM:~/cs108_project_bash/final_2$ 
```

Figure 3: combine after total

- Command: `bash submission.sh combine -f fils_list.txt`
  Output: The mentioned file is not present. Did you mean "files_list.txt"?

- Command:
  `bash submission.sh combine abc`
  Output: (Similar error messages are given for other commands)
  Wrong format. Usage: bash submission.sh combine (or) bash submission.sh combine -f <files_list>

4.1.2 upload

- Command: `bash submission.sh upload`
  Output:
  Wrong format. Usage: bash submission.sh upload < file_path>

- Command:
  `bash submission.sh upload /home/abhineet/cs108_project_bash/testcases/quz1.csv`
  Output:
  quz1.csv was not found in /home/abhineet/cs108_project_bash/testcases/quz1.csv. Did you mean "quiz1.csv"?

- Command:
  `bash submission.sh upload /home/abhineet/cs108_project_bash/testcases/quiz1.csv`
  Output:
  quiz1.csv uploaded from /home/abhineet/cs108_project_bash/testcases/quiz1.csv

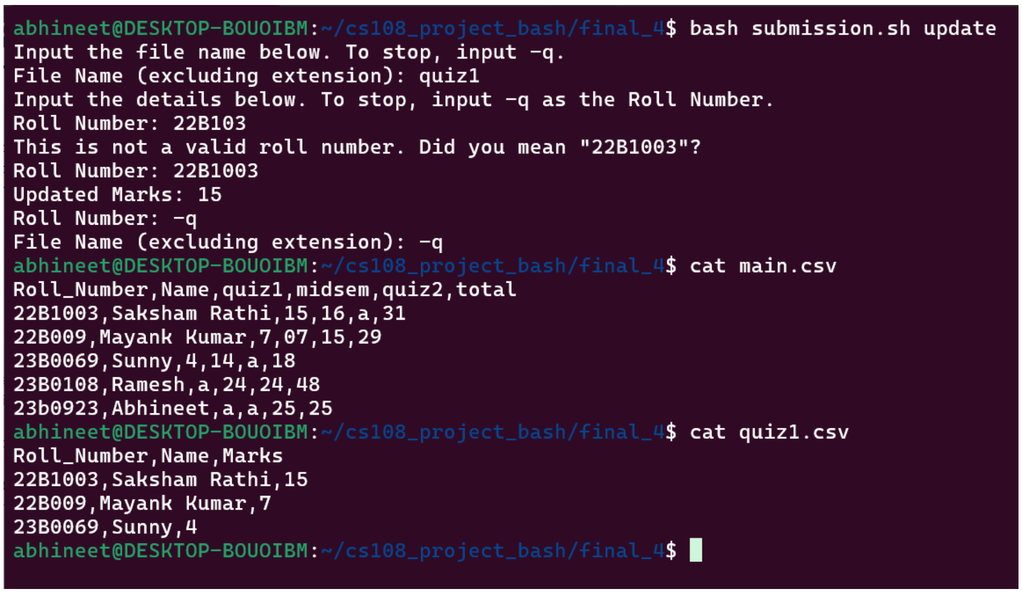
- Command: (if user tries to upload a file with the same name as an existing file)
  `bash submission.sh upload /home/abhineet/cs108_project_bash/testcases/quiz1.csv`
  Output:
  This file is already in the working directory. Do you want to replace it?(y/n)
  - If user inputs "y", output:
    quiz1.csv uploaded from /home/abhineet/cs108_project_bash/testcases/quiz1.csv
    To see changes in main.csv, delete the existing main.csv and run bash submission.sh combine
  - If user inputs "n", there is no output.

4.1.3 total

- Command: `bash submission.sh total`
  Output: Refer figure 3
- If main.csv already has a total column, user is asked whether to calculate it
- If main.csv is not present, on running `bash submission.sh total`,
  the output is : Please run bash submission.sh combine first.

4.1.4 update

- Command: `bash submission.sh update`

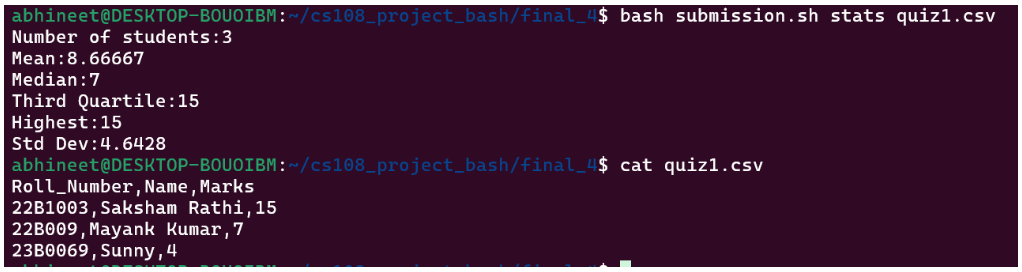


```
abhineet@DESKTOP-BOUOIBM:~/cs108_project_bash/final_4$ bash submission.sh update
Input the file name below. To stop, input -q.
File Name (excluding extension): quiz1
Input the details below. To stop, input -q as the Roll Number.
Roll Number: 22B103
This is not a valid roll number. Did you mean "22B1003"?
Roll Number: 22B1003
Updated Marks: 15
Roll Number: -q
File Name (excluding extension): -q
abhineet@DESKTOP-BOUOIBM:~/cs108_project_bash/final_4$ cat main.csv
Roll_Number,Name,quiz1,midsem,quiz2,total
22B1003,Saksham Rathi,15,16,a,31
22B009,Mayank Kumar,7,07,15,29
23B0069,Sunny,4,14,a,18
23B0108,Ramesh,a,24,24,48
23b0923,Abhineet,a,a,25,25
abhineet@DESKTOP-BOUOIBM:~/cs108_project_bash/final_4$ cat quiz1.csv
Roll_Number,Name,Marks
22B1003,Saksham Rathi,15
22B009,Mayank Kumar,7
23B0069,Sunny,4
abhineet@DESKTOP-BOUOIBM:~/cs108_project_bash/final_4$
```

Figure 4: update

4.1.5 stats

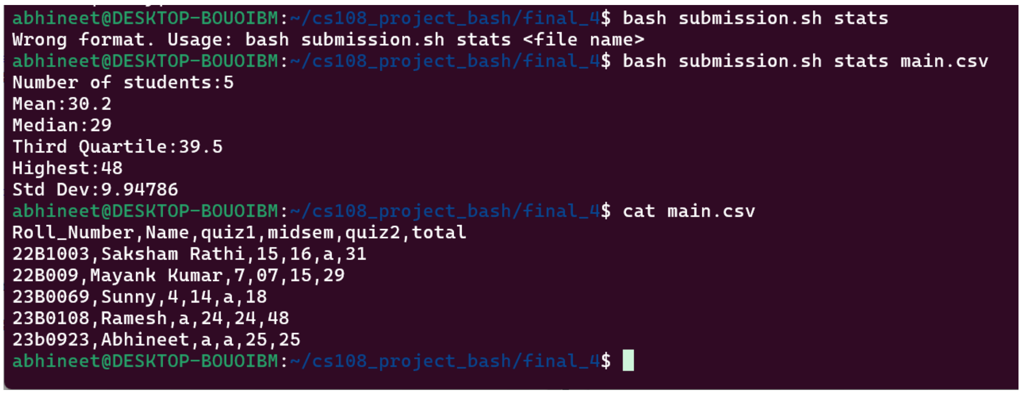
- Command: `bash submission.sh stats quiz1.csv`



```
abhineet@DESKTOP-BOUOIBM:~/cs108_project_bash/final_4$ bash submission.sh stats quiz1.csv
Number of students:3
Mean:8.66667
Median:7
Third Quartile:15
Highest:15
Std Dev:4.6428
abhineet@DESKTOP-BOUOIBM:~/cs108_project_bash/final_4$ cat quiz1.csv
Roll_Number,Name,Marks
22B1003,Saksham Rathi,15
22B009,Mayank Kumar,7
23B0069,Sunny,4
```

Figure 5: stats quiz1.csv

- Command: `bash submission.sh stats main.csv`

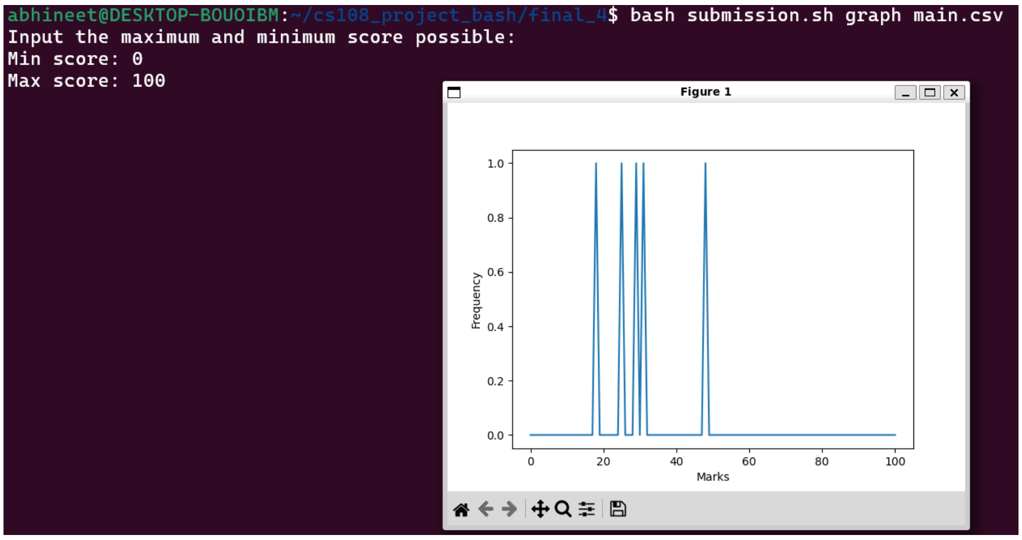


```
abhineet@DESKTOP-BOUOIBM:~/cs108_project_bash/final_4$ bash submission.sh stats
Wrong format. Usage: bash submission.sh stats <file name>
abhineet@DESKTOP-BOUOIBM:~/cs108_project_bash/final_4$ bash submission.sh stats main.csv
Number of students:5
Mean:30.2
Median:29
Third Quartile:39.5
Highest:48
Std Dev:9.94786
abhineet@DESKTOP-BOUOIBM:~/cs108_project_bash/final_4$ cat main.csv
Roll_Number,Name,quiz1,midsem,quiz2,total
22B1003,Saksham Rathi,15,16,a,31
22B009,Mayank Kumar,7,07,15,29
23B0069,Sunny,4,14,a,18
23B0108,Ramesh,a,24,24,48
23b0923,Abhineet,a,a,25,25
abhineet@DESKTOP-BOUOIBM:~/cs108_project_bash/final_4$
```

Figure 6: stats main.csv

4.1.6 graph

- Command: `bash submission.sh graph main.csv`

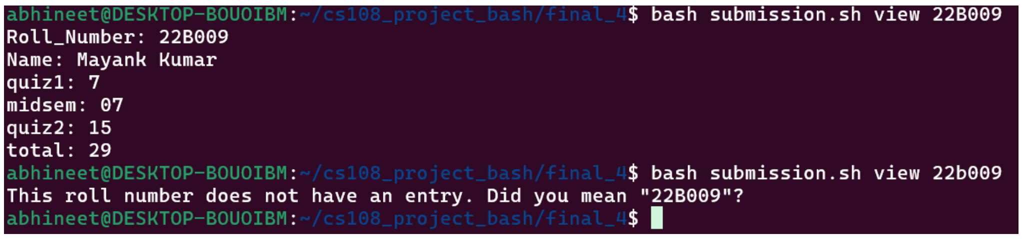


```
abhineet@DESKTOP-BOUOIBM:~/cs108_project_bash/final_4$ bash submission.sh graph main.csv
Input the maximum and minimum score possible:
Min score: 0
Max score: 100
```

Figure 7: graph

4.1.7 view

- Command: `bash submission.sh view 22B009`



```
abhineet@DESKTOP-BOUOIBM:~/cs108_project_bash/final_4$ bash submission.sh view 22B009
Roll_Number: 22B009
Name: Mayank Kumar
quiz1: 7
midsem: 07
quiz2: 15
total: 29
abhineet@DESKTOP-BOUOIBM:~/cs108_project_bash/final_4$ bash submission.sh view 22b009
This roll number does not have an entry. Did you mean "22B009"?
abhineet@DESKTOP-BOUOIBM:~/cs108_project_bash/final_4$
```

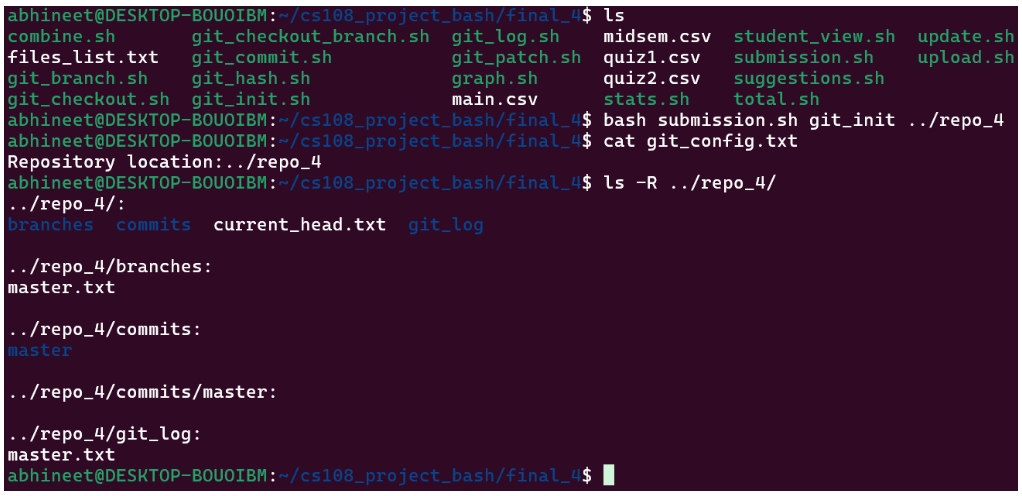
Figure 8: view

4.2 Version Control

In this section, the outputs printed for various commands of VCS are shown.

4.2.1 git_init

Command: `bash submission.sh git_init`

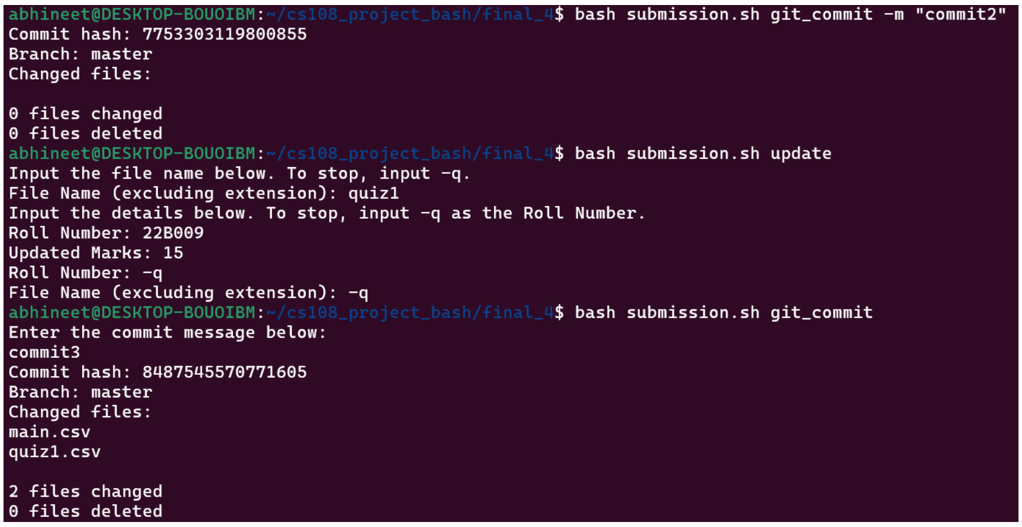


```
abhineet@DESKTOP-BOUOIBM:~/cs108_project_bash/final_4$ ls
combine.sh       git_checkout_branch.sh  git_log.sh    midsem.csv  student_view.sh  update.sh
files_list.txt   git_commit.sh           git_patch.sh  quiz1.csv   submission.sh    upload.sh
git_branch.sh    git_hash.sh             graph.sh      quiz2.csv   suggestions.sh
git_checkout.sh  git_init.sh             main.csv      stats.sh    total.sh
abhineet@DESKTOP-BOUOIBM:~/cs108_project_bash/final_4$ bash submission.sh git_init ../repo_4
abhineet@DESKTOP-BOUOIBM:~/cs108_project_bash/final_4$ cat git_config.txt
Repository location:../repo_4
abhineet@DESKTOP-BOUOIBM:~/cs108_project_bash/final_4$ ls -R ../repo_4/
../repo_4/:
branches  commits  current_head.txt  git_log

../repo_4/branches:
master.txt

../repo_4/commits:
master

../repo_4/commits/master:

../repo_4/git_log:
master.txt
abhineet@DESKTOP-BOUOIBM:~/cs108_project_bash/final_4$
```

Figure 9: git_init

4.2.2 git_commit

Commands: `bash submission.sh git_commit -m "commit2"` and
`bash submission.sh git_commit`

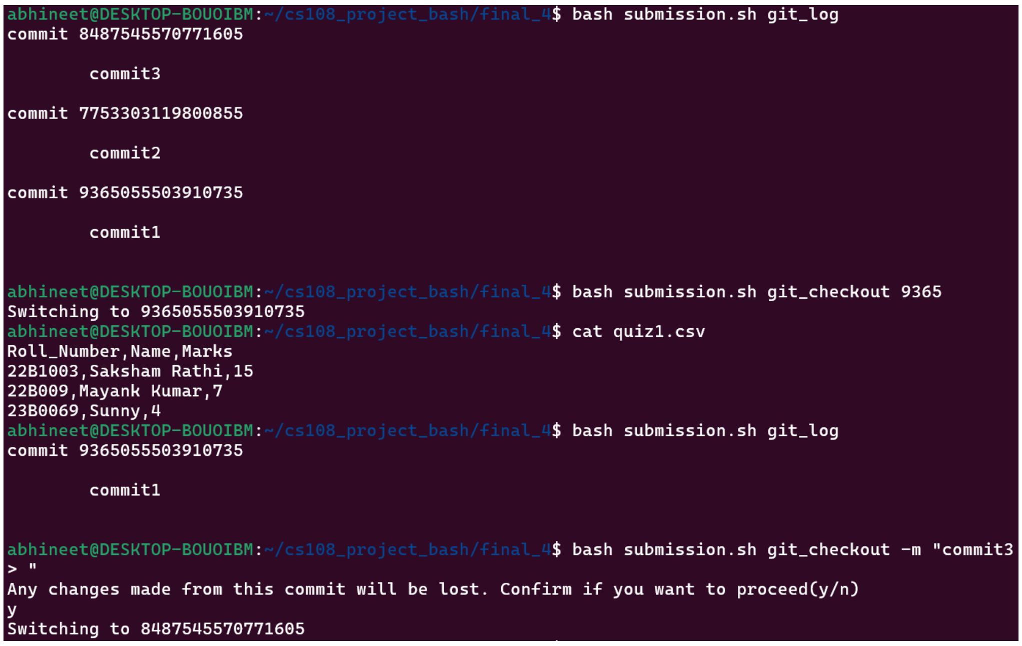


```
abhineet@DESKTOP-BOUOIBM:~/cs108_project_bash/final_4$ bash submission.sh git_commit -m "commit2"
Commit hash: 7753303119800855
Branch: master
Changed files:

0 files changed
0 files deleted
abhineet@DESKTOP-BOUOIBM:~/cs108_project_bash/final_4$ bash submission.sh update
Input the file name below. To stop, input -q.
File Name (excluding extension): quiz1
Input the details below. To stop, input -q as the Roll Number.
Roll Number: 22B009
Updated Marks: 15
Roll Number: -q
File Name (excluding extension): -q
abhineet@DESKTOP-BOUOIBM:~/cs108_project_bash/final_4$ bash submission.sh git_commit
Enter the commit message below:
commit3
Commit hash: 8487545570771605
Branch: master
Changed files:
main.csv
quiz1.csv

2 files changed
0 files deleted
```

Figure 10: git_commit

4.2.3 git_checkout and git_log

Commands: `bash submission.sh git_checkout`,
`bash submission.sh git_log` and
`bash submission.sh git_checkout -m message`

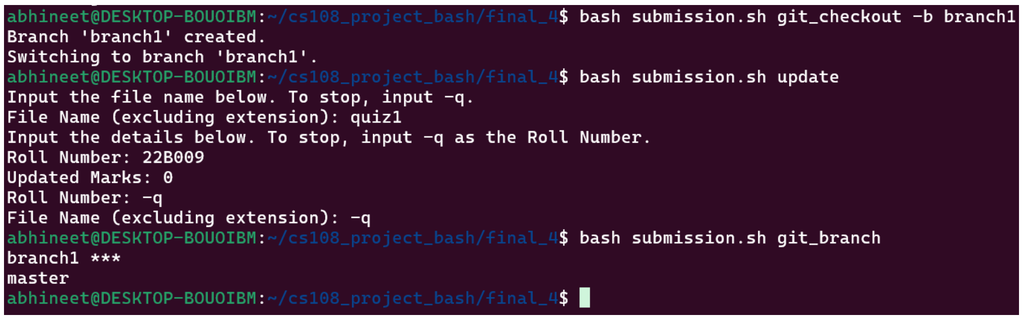


```
abhineet@DESKTOP-BOUOIBM:~/cs108_project_bash/final_4$ bash submission.sh git_log
commit 8487545570771605

        commit3

commit 7753303119800855

        commit2

commit 9365055503910735

        commit1


abhineet@DESKTOP-BOUOIBM:~/cs108_project_bash/final_4$ bash submission.sh git_checkout 9365
Switching to 9365055503910735
abhineet@DESKTOP-BOUOIBM:~/cs108_project_bash/final_4$ cat quiz1.csv
Roll_Number,Name,Marks
22B1003,Saksham Rathi,15
22B009,Mayank Kumar,7
23B0069,Sunny,4
abhineet@DESKTOP-BOUOIBM:~/cs108_project_bash/final_4$ bash submission.sh git_log
commit 9365055503910735

        commit1


abhineet@DESKTOP-BOUOIBM:~/cs108_project_bash/final_4$ bash submission.sh git_checkout -m "commit3
> "
Any changes made from this commit will be lost. Confirm if you want to proceed(y/n)
y
Switching to 8487545570771605
```

Figure 11: git_checkout and git_log

4.2.4 git_checkout -b and git_branch

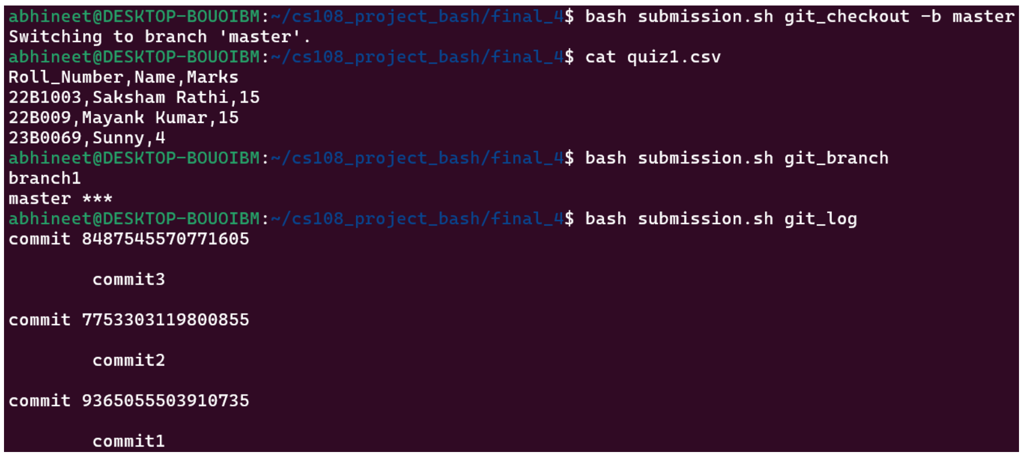
- Creating a branch:
  Commands: `bash submission.sh git_checkout -b branch1` and
  `bash submission.sh git_branch`



```
abhineet@DESKTOP-BOUOIBM:~/cs108_project_bash/final_4$ bash submission.sh git_checkout -b branch1
Branch 'branch1' created.
Switching to branch 'branch1'.
abhineet@DESKTOP-BOUOIBM:~/cs108_project_bash/final_4$ bash submission.sh update
Input the file name below. To stop, input -q.
File Name (excluding extension): quiz1
Input the details below. To stop, input -q as the Roll Number.
Roll Number: 22B009
Updated Marks: 0
Roll Number: -q
File Name (excluding extension): -q
abhineet@DESKTOP-BOUOIBM:~/cs108_project_bash/final_4$ bash submission.sh git_branch
branch1 ***
master
abhineet@DESKTOP-BOUOIBM:~/cs108_project_bash/final_4$
```

Figure 12: Creating a branch

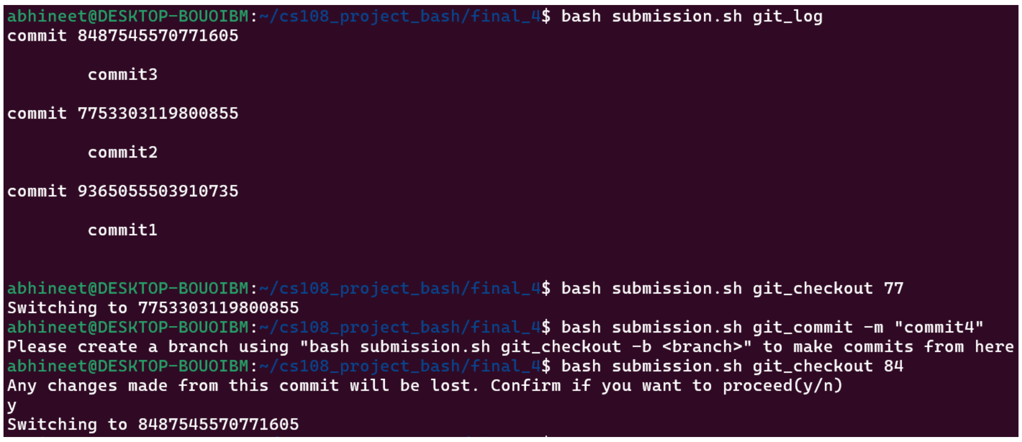
- Checking out to a branch:
  Commands: `bash submission.sh git_checkout -b master`



```
abhineet@DESKTOP-BOUOIBM:~/cs108_project_bash/final_4$ bash submission.sh git_checkout -b master
Switching to branch 'master'.
abhineet@DESKTOP-BOUOIBM:~/cs108_project_bash/final_4$ cat quiz1.csv
Roll_Number,Name,Marks
22B1003,Saksham Rathi,15
22B009,Mayank Kumar,15
23B0069,Sunny,4
abhineet@DESKTOP-BOUOIBM:~/cs108_project_bash/final_4$ bash submission.sh git_branch
branch1
master ***
abhineet@DESKTOP-BOUOIBM:~/cs108_project_bash/final_4$ bash submission.sh git_log
commit 8487545570771605

        commit3

commit 7753303119800855

        commit2

commit 9365055503910735

        commit1
```

Figure 13: git_checkout -b

- Commit and Checkout from a intermediary commit:



```
abhineet@DESKTOP-BOUOIBM:~/cs108_project_bash/final_4$ bash submission.sh git_log
commit 8487545570771605

        commit3

commit 7753303119800855

        commit2

commit 9365055503910735

        commit1


abhineet@DESKTOP-BOUOIBM:~/cs108_project_bash/final_4$ bash submission.sh git_checkout 77
Switching to 7753303119800855
abhineet@DESKTOP-BOUOIBM:~/cs108_project_bash/final_4$ bash submission.sh git_commit -m "commit4"
Please create a branch using "bash submission.sh git_checkout -b <branch>" to make commits from here
abhineet@DESKTOP-BOUOIBM:~/cs108_project_bash/final_4$ bash submission.sh git_checkout 84
Any changes made from this commit will be lost. Confirm if you want to proceed(y/n)
y
Switching to 8487545570771605
```

5 Modularisation

The code has been modularised by using a seperate file for each command. The code for individual commands is run via `submission.sh`. Below is a table of the code file for each command:

Command	Program file
combine	combine.sh
upload	upload.sh
total	total.sh
update	update.sh
stats	stats.sh
graph	graph.sh
view	student_view.sh
git_init	git_init.sh
git_commit	git_commit.sh
git_checkout	git_checkout.sh
git_checkout -b	git_checkout_branch.sh
git_log	git_log.sh
git_branch	git_branch.sh

`git_patch.sh` is used by `git_commit.sh` to create files of previous commits by patching.
`git_hash.sh` is used to create commit hashes.
`suggestions.sh` is used to give close-matches for some arguments in error messages.

References

[1] https://www.gnu.org/software/bash/manual/bash.html.

[2] https://www.gnu.org/software/diffutils/manual/html_node/diff-Options.html.

[3] https://www.geeksforgeeks.org/bash-scripting-for-loop/.

[4] https://www.gnu.org/software/sed/manual/html_node/Addresses.html.

[5] https://stackoverflow.com/questions/1728683/case-insensitive-comparison-of-strings-in-shell-script.

[6] https://www.gnu.org/software/gawk/manual/html_node/Index.html.

[7] https://stackoverflow.com/questions/3362920/get-just-the-filename-from-a-path-in-a-bash-script.

[8] https://stackoverflow.com/questions/16831429/when-setting-ifs-to-split-on-newlines-why-is-it-necessary-to-include-a-backspac.

[9] https://stackoverflow.com/questions/18488651/how-to-break-out-of-a-loop-in-bash.

[10] https://stackoverflow.com/questions/26283715/how-to-find-the-most-similar-word-in-a-list-in-python.

[11] https://unix.stackexchange.com/questions/496179/how-to-change-a-file-in-place-using-awk-as-with-sed-i.

[12] https://unix.stackexchange.com/questions/225943/except-the-1st-argument.

[13] https://unix.stackexchange.com/questions/87772/add-lines-to-the-beginning-and-end-of-the-huge-file.

[14] Bash manual pages.

[15] Kameswari Chebrolu. Cs 104 course slides - bash,sed,awk,matplotlib, 2024.