BASH GRADING

Somanath Vasanth

April 28, 2024

Contents

1 Introduction

The BASH GRADER project simplifies managing and interpreting CSV files using Bash scripting. It helps combine data, update totals, and track versions easily. With commands like combine, upload, and git functions, it's user-friendly. Customizations, like calculating statistics and creating graphs, make it even more useful.

2 Files

The main code is written in bash script in the file submission.sh.And codes for customization are written in python.

3 Usage of Basic commands

3.1 bash submission.sh combine

It combines all marks of all the students into main.csv.

3.2 bash submission.sh total

It creates a total column in main.csv and adds all the marks of each student.

3.3 bash submission.sh upload < path of the file you want to upload >

This uploads the given file in the present working directory.

3.4 bash submission.sh update

This command prompts you to enter the roll number and exam name for which you want to change marks, and then updates the marks in main.csv and individual files accordingly.

3.5 bash submission.sh git_init <path of the directory>

This command marks the given directory as remote directory which is used in the git_commit and git_checkout

3.6 bash submission.sh git_commit -m <commit message>

this creates a copy of original directory and stores it and adds its hash value and commit message to gitlog file.

3.7 bash submission.sh git_checkout <hashvalue or hash prefix>

this changes original directory to the given hash directory on the basis of given hash or hash prefix.

3.8 bash submission.sh git_checkout -m <commit message>

this changes original directory to the given hash directory on the basis of given commit message.

4 Usage of Customization commands

4.1 bash submission.sh search

This prompts you to enter the roll number of a student and returns the marks of that student in the terminal.

4.2 bash submission.sh statistics

This prompts you to enter the exam for which you want statistics and prints the statistics in the terminal. If you want statistics of all exams, you need to type ”all”.

4.3 bash submission.sh graphs

This prompts you to choose whether you want the graph student-wise or exam-wise, and then asks for other details. Finally, it generates a bar graph.

4.4 bash submission.sh git_diff <commit hashprefix1> <commit hashprefix2− >

this shows diff between the given two commits similar to the actual git diff command.

4.5 bash submission.sh git_log

this gives information about all previous commits.

4.6 bash submission.sh git_clean

it will remove the assignment of remotedirectory and this removes the commits and .git log from the remote directory . this uninitialize the remote directory.

4.7 bash submission.sh add_a_student

this asks the user to enter the exam name in which you want to add the student and details of the student and adds the student if there is no such exam given by user and if student already exists in the file it will give corresponding errors.

4.8 bash submission.sh remove_a_student

this asks the user to enter the exam name in which you want to remove the student and details of the student and removes the student if there is no such exam given by user and if student doesnot exists in the file it will give corresponding errors.

4.9 bash submission.sh top_performer

this asks the user to enter the name of exam for which he want to know the to top performer.Then It prints the top performer details in the terminal.This gives error if the exam doesnt exist.

4.10 bash submission.sh leaderboard

this asks the user to enter the name of the exam for which he wants the rankings.Then it prints the students ranks in the terminal with their details.This gives error if the exam doesnt exist.

4.11 bash submission.sh overview

this asks the user to enter the name of the exam for which he wants to know the overview.then it prints the names of the students who are performing well and the names of the student who are not performing well and need to inprove.This gives error if the exam doesnt exist.

5 How the Basic code works

5.1 Combine Function

– **Check for Existing Main CSV File:** First, it checks if there's already a file called main.csv in the current folder. If it exists, it looks inside to see if there's a column named "total".

– **Extract Exam Names:** It looks at the names of all the CSV files in the folder to figure out the names of the exams.

– **Extract Student Roll Numbers:** It goes through all the CSV files and makes a list of all the unique student roll numbers.

– **Iterate Over Students and Exams:** For each student, it finds their name and their marks for each exam. If a student didn't take a particular exam, it marks their score as "a".

– **Append Data to Main CSV:** It adds all this student information (roll number, name, marks) to the main.csv file.

5.2 Total Function

– **Check for Total Column:** It looks at the last column of main.csv to see if it's named "total". If it's missing, it adds a new total column.

– **Calculate Total Marks:** For each student in main.csv, it adds up all their exam marks to find their total.

– **Update Main CSV:** It updates main.csv with the new total marks for each student.

5.3 Upload Function

– **File Existence Check:** Upon invocation, the function first verifies if the specified file exists at the provided path. If the file is not found, an error message is displayed, and the function terminates with an exit code of 1.

– **File Copying:** If the file exists, the function proceeds to copy it into the current working directory using the cp command. This ensures that the uploaded file is readily accessible for subsequent processing steps.

– **Confirmation Message:** Upon successful completion of the file transfer, the function outputs a confirmation message indicating that the file has been uploaded to the current directory.

5.4 Update Function

– **Prompting for Input:**It asks the user for the student's roll number and the exam name to update marks.

– **Input Validation:**It checks if the specified exam file exists. If not, it terminates with an error message.

- **Reading Updated Marks:** After confirming the exam file's existence, it prompts the user for updated marks.
- **Updating Marks:** It iterates through the exam file, updating marks for the specified student if found.
- **Saving Changes:** After updating, it saves changes to a temporary file and replaces the original.
- **Error Handling:** It notifies the user if the provided roll number doesn't match any student.file and replaces the original.
- **Integration with Combine Function:** If main.csv exists, it invokes the combine function to sync changes.

5.5 git_init function

- **Purpose:** Sets up a remote directory for Git version control.
- **Directory Creation:** If the specified directory doesn't exist, it creates it along with any necessary parent directories.
- **Storing Remote Directory Path:** Saves the path of the remote directory in a hidden file named `$HOME/.git_repo_data`.
- **Recording Initialization:** Appends a line "initial" to the `$HOME/.git_repo_data` file to indicate that no commits have been made yet.
- **Error Handling:** If the directory path is not provided, it displays an error message and exits.

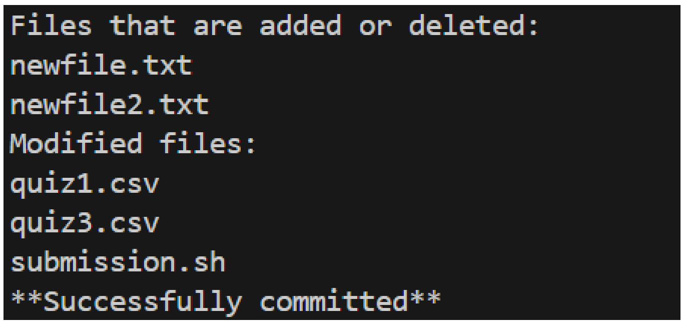
5.6 git_commit function

- **checking if git is initialized:** if git is not initialised it gives an error
- **Retrieving Remote Directory Path and Previous Commit Hash:**
  - Reads the remote repository directory path and the previous commit hash from the file stored at `$REPO_DATA_FILE`.
- **Validating Commit Message:**
  - Checks if a commit message is provided as an argument. If not, it displays an error message and exits.
- **Generating Commit Hash:**
  - Generates a unique hash using the `openssl rand -hex 8` command.
- **Creating Commit Directory:**
  - Creates a directory named with the generated hash in the `commits` folder within the remote directory to store the current directory snapshot.
- **Copying Current Directory:**
  - Recursively copies the contents of the current directory to the newly created commit directory.
- **Logging Commit Information:**

  - Appends the commit hash and message to the `.git_log` file in the remote directory to track commit history.

- **Identifying Modified Files :**
  - If it's not the first commit (`prevhash != "initial"`), compares the current commit with the previous one to identify modified files.
  - Uses the `diff` command to compare directories and extracts modified file names.
  - Handles different formats of `diff` output to extract modified files.

- **Updating Previous Hash:**
  - Updates the previous hash in the `$REPO_DATA_FILE` by adding the latest hash at the end of the file.



```
Files that are added or deleted:
newfile.txt
newfile2.txt
Modified files:
quiz1.csv
quiz3.csv
submission.sh
**Successfully committed**
```

Figure 1: the output of git commit function

5.7 git_checkout function

- **Initialization Check:**
  - Verifies whether the Git repository has been initialized by checking for the presence of the `$REPO_DATA_FILE`.

- **Retrieving Remote Directory:**
  - Obtains the remote repository directory path from the `$REPO_DATA_FILE`.

- **Input Validation:**
  - Validates the input provided to the function, ensuring either a commit hash or commit message is provided.

- **Identifying Commit ID:**
  - Searches for the provided commit hash or commit message in the `.git_log` file to find the corresponding commit ID.

- **Error Handling for Commit ID:**
  - Handles scenarios where no commit ID matches the provided prefix or multiple matches are found.

- **Setting Commit Directory:**

  - Determines the directory corresponding to the identified commit ID for checkout.
- **Performing Checkout:**
  - Removes the current directory contents and copies the contents of the identified commit directory to perform the checkout operation.
- **Success Message:**
  - Displays a success message indicating that the checkout operation was completed successfully.
- **Error Handling:**
  - Ensures that the Git repository has been initialized before attempting to perform the checkout operation.
  - Provides informative error messages and exits in case of missing commit IDs or multiple matches found.

6 How the Customization code works

6.1 graphs Function

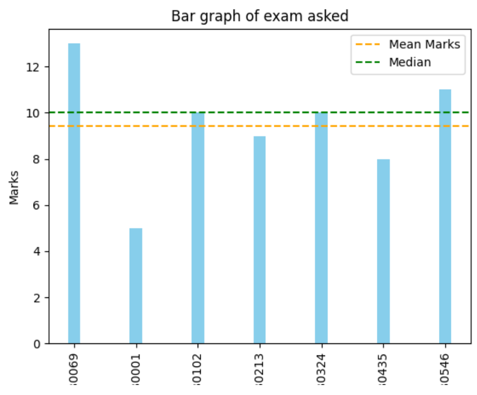
- **Input Prompt:** Asks the user whether they want the graph with respect to the student or the exam.
- **Exam Graph Generation:** If the user chooses "exam," prompts for the specific exam name, checks its existence, and generates a graph displaying student-wise marks for the chosen exam.
- **Student Graph Generation:** If the user chooses "student," prompts for the student's name, calculates the mean marks of each exam from the main CSV file, extracts the student's marks for each exam, and generates a graph comparing the student's actual marks to the class mean for each exam.

graphsexam.py Script

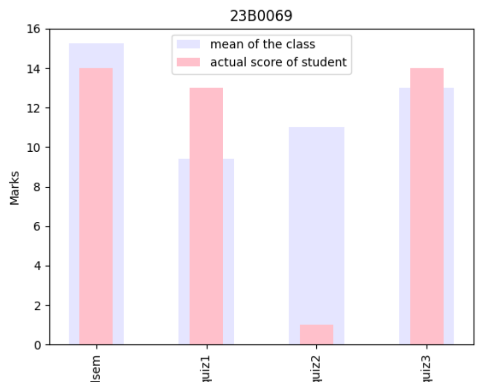
- **Input Handling:** Accepts the exam CSV filename as a command-line argument.
- **Data Processing:** Reads the CSV contents, extracts student names and marks, calculates mean and median marks, and generates a bar graph displaying student-wise marks for the exam, along with mean and median lines.

graphsstudent.py Script

- **Input Handling:** Accepts the student's name and the total number of exams as command-line arguments.
- **Data Processing:** Reads the main CSV file to obtain exam names and mean marks, extracts the student's marks for each exam, and generates a bar graph comparing the student's actual marks to the class mean for each exam.
- **The graphs will be displayed like this:**



(a) graph with respect to exam



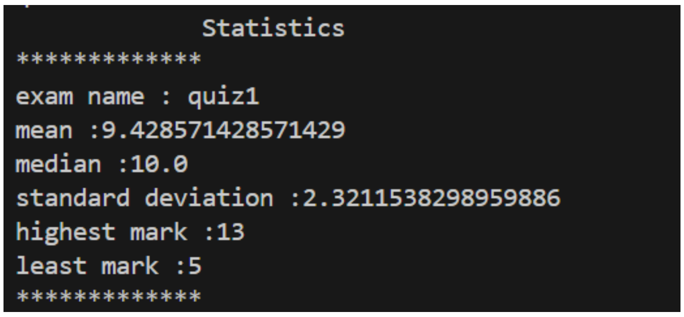
(b) graph with respect to exam

6.2 Search Function

- **Prompt User Input:** The function prompts the user to enter the roll number of the student they want to search for.
- **File Existence Check:** If the "main.csv" file does not exist, the function creates it by calling the "combine" and "total" functions.
- **Extract Student Information:** Using the "awk" command, the function extracts the exam names and marks of the student with the provided roll number from the main CSV file.
- **Error Handling:** If the roll number is not found in the main CSV file, it displays an error message and exits the function.
- **Display Student Information:** If the roll number is found, the function displays the student's information, including exam names and marks.
- **Cleanup:** Finally, if the "main.csv" file was created during the search process, it is deleted to maintain cleanliness.

6.3 Statistics Function

- **Prompt for Input:** Ask the user to input the exam name or "all" to get statistics.
- **Check Exam Existence:** Make sure the specified exam CSV file exists.
- **Extract Data and Run Python Script:** Read CSV files, get marks data, and run the "statistics.py" script.
- **Display Statistics:** Show the calculated statistics for the specified exam or all exams.
- **about statistics.py:**
  Purpose: Calculate statistics for marks in CSV files.
  Usage: Takes filename and exam name as arguments.
  Functionality: Reads CSV file, gets marks data, and calculates statistics using NumPy.
  utput: Prints exam name and statistics to the terminal.



```
              Statistics
*************
exam name : quiz1
mean :9.428571428571429
median :10.0
standard deviation :2.3211538298959886
highest mark :13
least mark :5
*************
```

Figure 3: the output of statistics function

6.4 git_clean function

- **Initialization Check:**
  - Checks if the Git repository has been initialized.
- **Removing Files:**
  - Removes the commit history, log file, and repository data file.
- **Success Message:**
  - Displays a message indicating the completion of the clean operation.

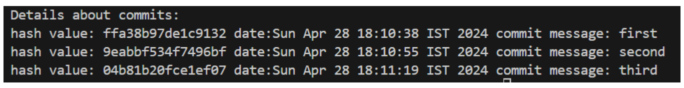
6.5 git_diff Function

- Input Validation: The function checks if the repository data file exists and verifies whether both commit IDs are provided as arguments. If not, it displays the correct usage syntax and exits.
- Commit ID Retrieval: It retrieves the remote directory path from the repository data file and searches for the full commit hash corresponding to the provided commit IDs by parsing the '.git_log' file within the repository. Error messages are displayed if no or multiple matches are found for a commit ID.
- Unified Diff Generation:Once the full commit hashes for both commits are obtained, the function generates a unified diff using the 'diff' command, providing a comprehensive view of the changes made between the two commits.
- Error Handling: Robust error handling ensures smooth execution, detecting and handling scenarios such as missing repository data file, incomplete or incorrect commit IDs, and multiple matches for a given commit ID prefix.

6.6 git_log Function

- Input Validation: The function checks if the repository data file exists. If not, it displays an error message indicating that the remote directory is not initialized.
- Remote Directory Retrieval: If the repository data file exists, the function retrieves the remote directory path from the file.
- Git Log File Check: It checks if the '.git_log' file exists in the remote directory. If the file exists, it displays the commit details.

- Commit Details Display: The function prints a message indicating that details about commits are about to be displayed. It then reads and displays the contents of the '.git_log' file, which typically includes commit hashes, commit messages, timestamps, and authors.
- Error Handling: Robust error handling ensures that appropriate messages are displayed in case of missing repository data files or '.git_log' files.



```
Details about commits:
hash value: ffa38b97de1c9132 date:Sun Apr 28 18:10:38 IST 2024 commit message: first
hash value: 9eabbf534f7496bf date:Sun Apr 28 18:10:55 IST 2024 commit message: second
hash value: 04b81b20fce1ef07 date:Sun Apr 28 18:11:19 IST 2024 commit message: third
```

Figure 4: the output of gitlog function

6.7 add_a_student Function

- **Prompt for Exam Name:** The function begins by asking the user to input the name of the exam to which they want to add a student.
- **Check Exam Existence:** It checks if the CSV file corresponding to the specified exam exists. If the file does not exist, an error message is displayed, and the function terminates.
- **Input Student Details:** The function prompts the user to input the student's roll number, name, and marks.
- **Check Existing Roll Number:** It ensures that no student with the same roll number already exists in the exam file. If a student with the same roll number is found, the function prompts the user to decide whether to update their marks instead.
- **Add Student:** If there are no existing students with the same roll number, the function adds the new student's details (roll number, name, marks) to the exam file.

6.8 remove_a_student Function

- **Prompt for Exam Name:** The function begins by asking the user to input the name of the exam from which they want to remove a student.
- **Check Exam Existence:** It checks if the CSV file corresponding to the specified exam exists. If the file does not exist, an error message is displayed, and the function terminates.
- **Input Student's Roll Number:** The function prompts the user to input the roll number of the student they want to remove.
- **Search and Remove:** It iterates through the CSV file, searching for the student with the specified roll number. If found, it removes the student's record from the file.
- **Update CSV File:** After removing the student's record, it updates the original CSV file accordingly.
- **Output:** If the specified student is not found, it notifies the user that no such student exists for removal.

6.9 top_performer Function

– **Input:** Prompts the user to input the exam name.

– **Validation:** Checks if the CSV file for the specified exam exists. If not, displays an error message and exits.

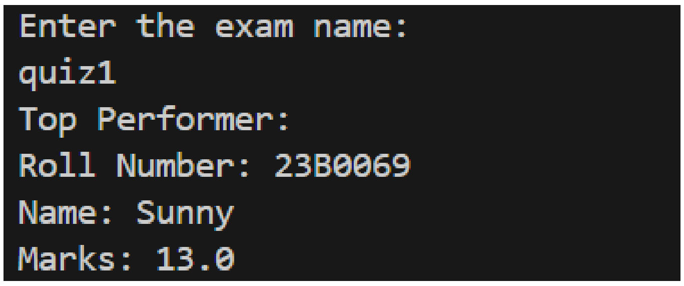
– **Execution:** Calls a Python script named "top_performer.py" with the exam CSV file as an argument.

top_performer.py Script

– **Input:** Receives the exam CSV file as a command-line argument.

– **Processing:** Reads the CSV file, skips the header, and finds the student with the highest marks.

– **Output:** Prints the details (roll number, name, and marks) of the top performer.



```
Enter the exam name:
quiz1
Top Performer:
Roll Number: 23B0069
Name: Sunny
Marks: 13.0
```

Figure 5: the output of statistics function

6.10 Leaderboard Function

– **Input Prompt:** Asks for the exam name.

– **Exam Existence Check:** Verifies the existence of the exam's CSV file. If not found, displays an error and exits.

– **Data Sorting:** Extracts and sorts student data by marks, saving it to a temporary file.

– **Information Display:** Prints exam details and a leaderboard header.

– **Python Script Invocation:** Calls "rank.py" to calculate ranks and display the leaderboard.

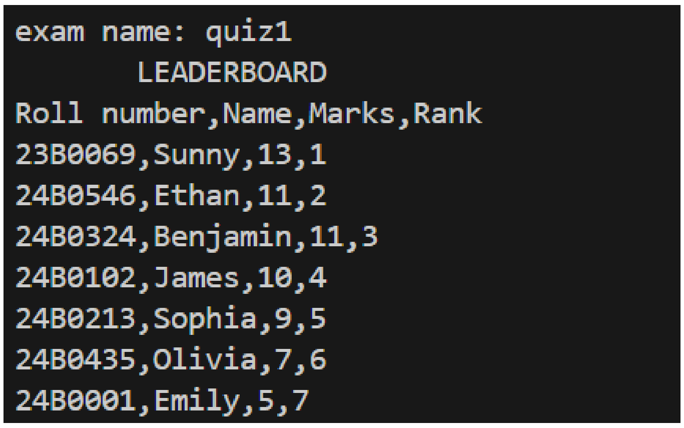
– **Cleanup:** Removes the temporary file after processing.

rank.py Script

– **Input Handling:** Accepts the CSV filename as a command-line argument.

– **File Reading:** Reads the CSV contents.

– **Data Processing:** Parses the data, add ranks, and prints the leaderboard.



```
exam name: quiz1
       LEADERBOARD
Roll number,Name,Marks,Rank
23B0069,Sunny,13,1
24B0546,Ethan,11,2
24B0324,Benjamin,11,3
24B0102,James,10,4
24B0213,Sophia,9,5
24B0435,Olivia,7,6
24B0001,Emily,5,7
```

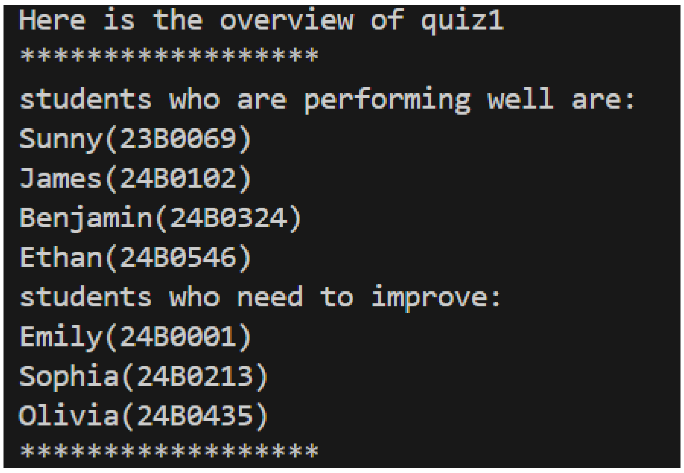
Figure 6: Output of the leaderboard function

6.11 Overview Function

- **Input Prompt:** Asks for the exam name.
- **Exam Existence Check:** Verifies the existence of the exam's CSV file. If not found, displays an error and exits.
- **Display Overview:** Prints an overview of the exam using the "overview.py" script.

overview.py Script

- **Input Handling:** Accepts the CSV filename as a command-line argument.
- **Data Processing:** Reads the CSV contents, calculates mean marks, and identifies good and bad performing students.
- **Output:**
  - Prints the list of good performing students.
  - Prints the list of bad performing students.



```
Here is the overview of quiz1
******************
students who are performing well are:
Sunny(23B0069)
James(24B0102)
Benjamin(24B0324)
Ethan(24B0546)
students who need to improve:
Emily(24B0001)
Sophia(24B0213)
Olivia(24B0435)
******************
```

Figure 7: output of overview function