Relational Database - Build Five Programs using Bash Scripting - freeCodeCamp

1. https://gitpod.io/workspaces
2. echo random_text
3. sh <filename>.sh - to run the file
    a. sh stands for shell
    b. Using sh to run your script uses the shell interpreter
4. bash <filename>.sh - to run the file
    a. bash stands for bourne-again shell
5. which bash - find the bash interpreter where it is located by entering this command
    a. e.g, - /usr/bin/bash
    b. The absolute path to the bash interpreter
    c. shebang = #!/bin/bash - add this command on the very top of the file
    d. You can run it by executing the file and it will default to bash
6. List what is in the project - ls -l
    a. - describes permissions different users have with the file
    b. r means read - means accept input from a user
        i. read VARIABLE_NAME - this will get user input and store it into a new variable
    c. w means write
    d. x execute
    e. chmod +x <filename>.sh - give everyone executable permissions
7. Bash has variables, functions, and other things you might be familiar with
    a. VARIABLE_NAME=VALUE
    b. There cannot be any spaces around the equal(=) sign, if a variable has any spaces in it, place double quotes around it.
    c. To **use** a variable, please place **$** in front of it like this: $VARIABLE_NAME
        i. echo $VARIABLE_NAME
8. man - manual
    a. man <command> or man echo
9. echo $* printed all the argument passed to your script
10. if condition
    a. if
    b. then
        i. echo true
    c. else
        i. echo false
    d. fi
11. Compare data type inside the bracket
    a. -eq (equal)
    b. -ne (not equal)
    c. -lt (less than)
    d. -gt (greater than)
    e. -ge (greater than or equal)
12. $? - View the exit status of the last command
13. help <command> - e.g. help test
14. NOTE: 0 - true | 1 - false
15. bin - stands for binary

16. echo -e “\n<message>\n”

17. While condition in bash
    a. While [[CONDITION]]
    b. do
        i. STATEMENTS
    c. done
18. printenv - print all environment variables
19. p - stands for print
20. $RANDOM - a variable will generate a random number from 0 to 32767.
    a. modulus
        i. operator to make it in the range you want
        ii. $RANDOM%75
        iii. So, as a reminder, `(( ... ))` will perform a calculation or operation and output nothing.
21. ARRAY = ARR(“a” “b” “c”)
    a. echo ${ARR[1]}
    b. Similarly, you can use the `*` or `@` to print your whole array. In the terminal, use `echo` to print all the items in your array.
22. FUNCTIONS
    a. FUNCTION_NAME(){
        i. STATEMENTS
    b. }
    c. No $ needed when calling the function
23. until loop
    a. The `until` loop is very similar to the `while` loop you used. It will execute the loop until a condition is met.
    b. until [[CONDITION]]
    c. do
        i. STATEMENTS
    d. done