-Faculty of Engineering and Technology-

-LINUX LABORATORY-

(ENCS3130)

-Shell Scripting Project-

Student's Name: Qays Safa

Student's No: 1190880

Partner Name: Tariq Odeh

Partner No: 1190699

Sec: 1

Instructor: Dr. Mohammad Jubran

Assistant: Eng. Shahd Al-Masri

Date: 12-8-2021

Table of contact:

Theory:

We will discuss and clarify all the operations that took place in the project and clarify all the necessary points.

1) Test the data file if exit or not.

A- Code:

```
#detecting file is exist or not
while true                          #loop to keep read filename from user and >
do
  read filename                     #read the file name from user
  if [ -e "$filename" ]             #check if file exist
  then
     echo "--> File Readed successful"
     break
   else                              #if the file not exist
      echo "**> No Such file"
      echo "--> Please enter the filename again"
      continue                       #to back read drom user
   fi
done

#clear the  empty line from data file and put the data in tempfile
grep -v '^$' $filename > tempfile
```

B- Code results:

True case:

```
qayssafa@qayssafa-VirtualBox:~$ ./datasetprocessing
    [Welcome to Data set preprocessing and manipulation]
--> Please enter the filename:
data.csv
--> File Readed successful
```

False case:

```
    [Welcome to Data set preprocessing and manipulation]
--> Please enter the filename:
test.csv
**> No Such file
--> Please enter the filename again
gxf
**> No Such file
--> Please enter the filename again
data.csv
--> File Readed successful
```

C- Discuss results and code:

In this case, the program asks the user to enter the name of the file, and after entering it, the program reads it and checks it. If it is existing, the program prints that the reading was completed successfully and moves to the next step. In the event of an incorrect reading, the program asks the user to enter a new name, and the process is repeated until the user enters a valid name.

2) Test the format of data file.

Format checking depends on several things:

1- File extension is .csv

A- Code:

```
lastfour=$(echo -n $filename | tail -c 4)
if [ $lastfour = ".csv" ]    #check the file extension is it .csv or not
then
    echo "--> The file extension is correct"
else
   echo "**> The file extension is not correct !!"
   exit
fi
```

B- Code results:

True case:

```
    [Welcome to Data set preprocessing and manipulation]
--> Please enter the filename:
data.csv
--> File Readed successful
--> The file extension is correct
```

False case:

```
qayssafa@qayssafa-VirtualBox:~$ touch data.txt
qayssafa@qayssafa-VirtualBox:~$ ./datasetprocessing
    [Welcome to Data set preprocessing and manipulation]
--> Please enter the filename:
data.txt
--> File Readed successful
**> The file extension is not correct !!
```

C- Discuss results and code:

In this case, we extracted the last 4 char, which are supposed to be the file extension that the user entered, and then compare them with the correct extension (.csv) to move to other processes or close the program.

2- The number of commas.

A- Code:

```
truecomma=$(echo "$rows * ($cols -1)" | bc)
testcomma=$(cat tempfile | awk -F ',' '{print NF-1}' | awk '{n+=$1} END{print n}')
if [ $truecomma -eq $testcomma ]    #check if the numb
then
   echo "--> The number of comma is correct"
else
   echo "**> The number of comma is not correct !!"
   exit
fi
```

B- Code results:

True case:

```
qayssafa@qayssafa-VirtualBox:~$ ./datasetprocessing
    [Welcome to Data set preprocessing and manipulation]
--> Please enter the filename:
data.csv
--> File Readed successful
--> The file extension is correct
--> The number of comma is correct
```

```
sepal.length,sepal.w
5.1,3.5,1.4,0.2
4.9,3.35,1.4,0.2
4.7,3.2,1.3,0.2
4.6,3.1,1.5,0.2
5,3.6,1.4,0.2
```

False case:

```
    [Welcome to Data set preprocessing and manipulation]
--> Please enter the filename:
data.csv
--> File Readed successful
--> The file extension is correct
**> The number of comma is not correct !!
```

```
sepal.length,sepal.wi
5.13.5,1.4,0.2
4.9,3.35,1.4,0.2
4.7,3.2,1.3,0.2
4.6,3.1,1.5,0.2
5,3.6,1.4,0.2
```

C- Discuss results and code:

In this case, we first calculated the number of commas that should be, and then we calculated the actual number of commas currently in the file and compared them to each other to move to the next step or exit the program.

3- Number of columns.

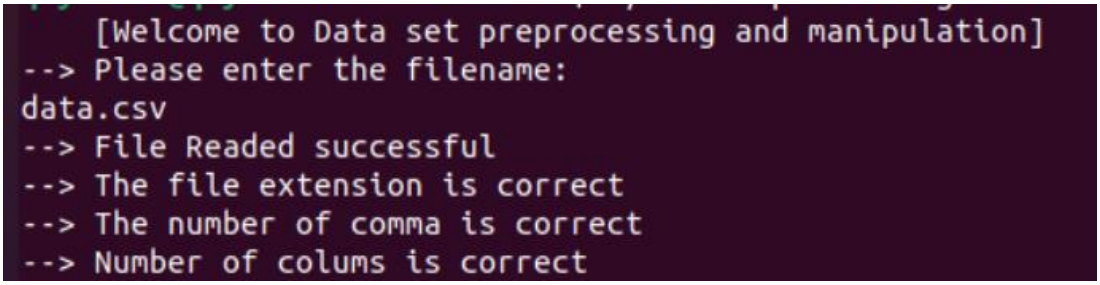
A- Code:

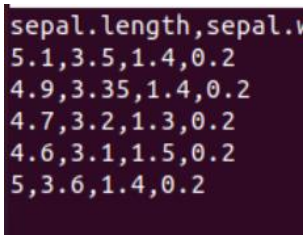
```
cols=$(echo $firstline | tr ',' '\12' | wc -l)
```

```
if [ $cols -eq 4 ]                    #check the numbe
then
   echo "--> Number of colums is correct"
else
   echo "**> Number of colums is not correct !!"
   exit
fi
```

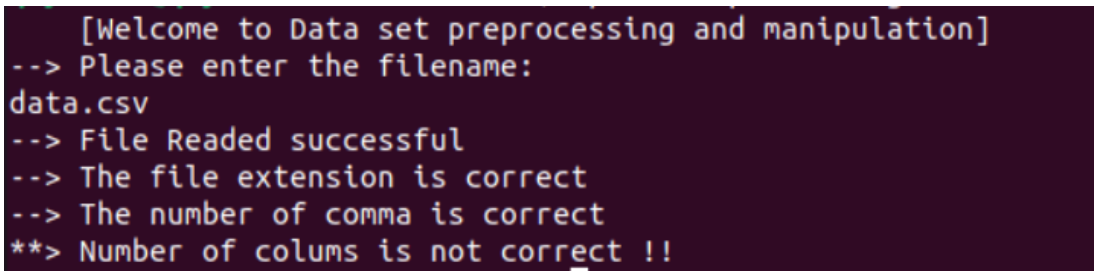
B- Code results:

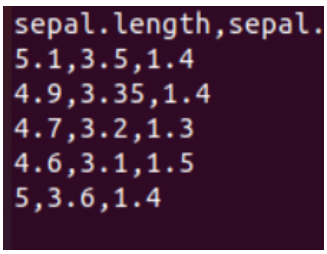
True case:





False case:





C- Discuss results and code:

In this case, we have calculated the number of columns and then compared it with the imposed number of columns (4). If it is correct, it will move to the next step, otherwise it will exit

4- <u>All data is numbers.</u>

A- Code:

```
truenumber=$(sed '1d' tempfile | grep '[^.,0-9]' | wc -w)
if [ $truenumber -eq 0 ]          #check if the
then
   echo "--> The all data is numbers"
else
   echo "**> The data have character !!"
   exit
fi
```

B- Code results:

True case:

```
    [Welcome to Data set preprocessing and manipulation]
--> Please enter the filename:
data.csv
--> File Readed successful
--> The file extension is correct
--> The number of comma is correct
--> The all data is numbers
```

```
sepal.length,sepal.w
5.1,3.5,1.4,0.2
4.9,3.35,1.4,0.2
4.7,3.2,1.3,0.2
4.6,3.1,1.5,0.2
5,3.6,1.4,0.2
```

False case:

```
    [Welcome to Data set preprocessing and manipulation]
--> Please enter the filename:
data.csv
--> File Readed successful
--> The file extension is correct
--> The number of comma is correct
**> The data have character !!
```

```
sepal.length,sepal.w
5.1,3.5,1.4,0.2
4.9,3.35,1.4,0.2
4.7,3.2,1.3,0.2
4.6,3.1,1.5,0.2
a,3.6,1.4,0.2
```

C- Discuss results and code:

In this case, we removed the first line and all numbers and commas from the file, so the letters remain, and then calculate the number of words that remained in the file. If the result is zero, this indicates that the file does not contain any letter or word, The comparison is done on this basis to move to the next step or exit the program.

3) Calculate the dimension.

Code:

```
cols=$(echo $firstline | tr ',' '\12' | wc -l)
rows=$(($(cat tempfile | wc -l)))
```

```
#function to get dimensions of dataset
dimensions() {
    #count number of colums
    cols=$(echo $firstline | tr ',' '\12' | wc -l)
    #count the number of rows
    rows=$(cat tempfile | wc -l)

    echo "Dimensions is $((rows - 1)) X $cols"
    echo "Rows = $((rows - 1)), Columns = $cols"
}
```

Code results:

```
D: for dimension
C: for compute statistics
S: for substitution
E: Exit
D
Dimensions is 5 X 4
Rows = 5, Columns = 4
```

Discuss results and code:

In this case, we calculated the number of columns and the number of rows and printed them on the screen.

4) Basic statistics.

Code:

```
statistics() {
   #we do the substites on the data and save it in another file to use it
in statistics
   #we use the substites here to fill any empty index in table before find
the statistics
   #because when find mean if the table have any empty index the output is
inaccurate

   echo "[!! IF the file contains any empty value]"
   echo "[!! It automatically replaces it with the arithmetic mean of the
column values]"

   cols=$(echo $firstline | tr ',' '\12' | wc -l)
   rows=$(($(cat tempfile | wc -l) - 1 ))

   #find the total of first colum, first we use grep to get ride of first
line (features)
   #then use awk to extract first coulm, and then use awk again to find
sum of numbers in colum field by field
   total=$(grep [0-9] tempfile | awk -F, '{print $1}' | awk '{ total +=
$1} END{print total}')
   #find the number of empty index in colum, fist use grep to get ride of
first line

   ##then use awk to extract first coulm, and then use awk again to count
number of empty index
   empty1=$(grep [0-9] tempfile | awk -F, '{print $1}' | awk
'{if($1=="")empty1 +=1}; END{print empty1 +=0}')
   #find mean value by dividing total by number of rows subtracted form it
number of empty index
   mean1=$(echo "scale=3;$total/($rows-$empty1)" | bc)

   #We use the same thing in other colums
   total=$(grep [0-9] tempfile | awk -F, '{print $2}' | awk '{ total +=
$1} END{print total}')
   empty2=$(grep [0-9] tempfile | awk -F, '{print $2}' | awk
'{if($1=="")empty2 += 1} END{print empty2 +=0}')
   mean2=$(echo "scale=3;$total/($rows-$empty2)" | bc)

   total=$(grep [0-9] tempfile | awk -F, '{print $3}' | awk '{ total +=
$1} END{print total}')
   empty3=$(grep [0-9] tempfile | awk -F, '{print $3}' | awk
'{if($1=="")empty3 += 1} END{print empty3 +=0}')
   mean3=$(echo "scale=3;$total/($rows-$empty3)" | bc)

   total=$(grep [0-9] tempfile | awk -F, '{print $4}' | awk '{ total +=
$1} END{print total}')
   empty4=$(grep [0-9] tempfile | awk -F, '{print $4}' | awk
'{if($1=="")empty4 += 1} END{print empty4 +=0}')

  mean4=$(echo "scale=3;$total/($rows-$empty4)" | bc)

  #to replace the empty index with mean value, first get the data from
tempfile
  #then use awk to check every colums if it has any empty index to
replace it with mean value for the colum which it is located
  #then replace all space with comma to maintain format then save data in
datafile to use it to find statistics
  cat tempfile | awk -F, '{if($1=="")$1='"$mean1"'}; {if($2=="")-
$2='"$mean2"'}; {if($3=="")$3='"$mean3"'}; {if($4=="")$4='"$mean4"'};
{print $0}' | tr ' ' ',' > datafile

  printf "\nMin: "
  for i in $(seq 1 $cols);    #loop to pass on all colums
  do
    #count min of the columns value
    #use grep to get ride of first line, then use awk to get colum by
colum
    #then sort the colum descending and take the first value which it min
    min=$(grep [0-9] datafile | awk -F, '{print $'$i'}' | sort -n | head
-1)
    printf "%5g" $min
  done

  printf "\nMax: "
  for i in $(seq 1 $cols);    #loop to pass on all colums
  do
    #count max of the columns value
    #use grep to get ride of first line, then use awk to get colum by
colum
    #then sort the colum ascending and take the first value which it max
    max=$(grep [0-9] datafile | awk -F, '{print $'"$i"'}' | grep -v '^$' |
sort -n | tail -1)
    printf "%5g" $max
  done

  printf "\nMean: "
  for i in $(seq 1 $cols);    #loop to pass on all colums
  do
    #count mean of the columns value
    #find the total of first colum, first we use grep to get ride of
first line (features)
    #then use awk to extract first coulm, and then use awk again to find
sum of numbers in colum field by field
    total=$(grep [0-9] datafile | awk -F, '{print $'"$i"'}' | awk
'{ total += $1} END{print total}')
    #find mean value by dividing total by number of rows
    mean=$(echo "scale=5;$total/$rows" | bc)
    printf "%5g" $mean
  done

  #count STDEV of the columns value
  printf "\nSTDEV "

  for i in $(seq 1 $cols)     #loop to pass on all colums
  do
     v1=0          #intialize variable and save in it the output of
opration (number - mean)^2 and
     v2=0          #intialize variable and save in it sum of v1 for each
colum
     sd=0          #intialize variable nd save in it the output of
opration (sqrt($v2 / ($rows - 1))
     #find sum for each colum to find mean as it was previously explained
     total=$(grep [0-9] datafile | awk -F, '{print $'"$i"'}' | awk
'{ total += $1} END{print total}')
     mean=$(echo "scale=5;$total/$rows" | bc)

     for j in $(seq 1 $rows)   #loop to pass on each index
     do
        #we store each elemant in n to use it to find stdev
        #delete first line and print each colum and take the elmeant by
elemant from cloum
        n=$(sed '1d' datafile | awk -F ',' '{print $'$i'}' | sed -n
''$j'p' | bc)
        #find (x-x')^2 and save it in v1
        v1=$(echo "($n - $mean) * ($n - $mean)" | bc)
        #find the sum v1 for each colum
        v2=$(echo "$v2 + $v1" | bc)
     done

  sd=$( echo "sqrt($v2 / ($rows - 1))" | bc -l )   #final answer
  printf "%15.7g" $sd
  done

  printf "\n"
}
```

Datafile:

```
sepal.length,sepal.wig
5.1,3.5,1.4,0.2
4.9,3.35,1.4,0.2
4.7,3.2,1.3,0.2
4.6,3.1,1.5,0.2
5,3.6,1.4,0.2
```

Code results:

```
D: for dimension
C: for compute statistics
S: for substitution
E: Exit
C
[!! IF the file contains any empty value]
[!! It automatically replaces it with the arithmetic mean of the column values]

Min:   4.6  3.1  1.3  0.2
Max:   5.1  3.6  1.5  0.2
Mean:  4.86 3.35  1.4  0.2
STDEV       0.2073644      0.2061553     0.07071068             0
```

Discuss results and code:

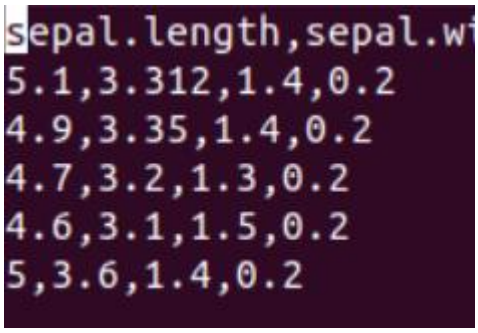
In this case, we will perform some arithmetic operations on each column separately, and before performing these operations, to avoid any problem, we compensate for any space in any column with the value of the mean of that column, and then we calculate the min and max value, the mean and the STDV for each column.

5) **Substitute.**

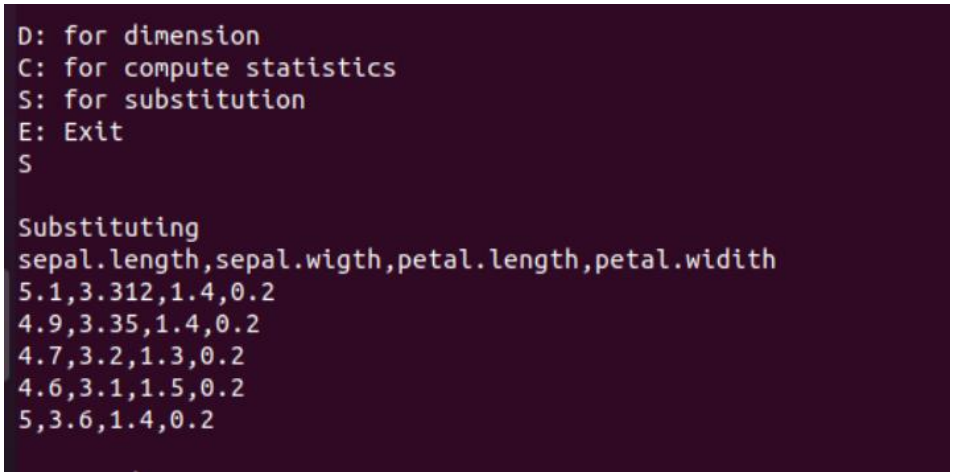
Code:

```
#function to substitute
substitute() {

   printf "\nSubstituting\n"
   cols=$(echo $firstline | tr ',' '\12' | wc -l)
   rows=$(($(cat tempfile | wc -l) - 1 ))

   #find the total of first colum, first we use grep to get ride of first
line (features)
   #then use awk to extract first coulm, and then use awk again to find
sum of numbers in colum field by field
   total=$(grep [0-9] tempfile | awk -F, '{print $1}' | awk '{ total +=
$1} END{print total}')
   #find the number of empty index in colum, fist use grep to get ride of
first line
   ##then use awk to extract first coulm, and then use awk again to count
number of empty index
   empty1=$(grep [0-9] tempfile | awk -F, '{print $1}' | awk
'{if($1=="")empty1 +=1}; END{print empty1 +=0}')
   #find mean value by dividing total by number of rows subtracted form it
number of empty index
   mean1=$(echo "scale=3;$total/($rows-$empty1)" | bc)

   #We use the same thing in other colums
   total=$(grep [0-9] tempfile | awk -F, '{print $2}' | awk '{ total +=
$1} END{print total}')
   empty2=$(grep [0-9] tempfile | awk -F, '{print $2}' | awk
'{if($1=="")empty2 += 1} END{print empty2 +=0}')
   mean2=$(echo "scale=3;$total/($rows-$empty2)" | bc)

   total=$(grep [0-9] tempfile | awk -F, '{print $3}' | awk '{ total +=
$1} END{print total}')
   empty3=$(grep [0-9] tempfile | awk -F, '{print $3}' | awk
'{if($1=="")empty3 += 1} END{print empty3 +=0}')
   mean3=$(echo "scale=3;$total/($rows-$empty3)" | bc)

   total=$(grep [0-9] tempfile | awk -F, '{print $4}' | awk '{ total +=
$1} END{print total}')
   empty4=$(grep [0-9] tempfile | awk -F, '{print $4}' | awk
'{if($1=="")empty4 += 1} END{print empty4 +=0}')
   mean4=$(echo "scale=3;$total/($rows-$empty4)" | bc)

   #to replace the empty index with mean value, first get the data from
tempfile
   #then use awk to check every colums if it has any empty index to
replace it with mean value for the colum which it is located

   #then replace all space with comma to maintain format then save data in
ccfile
   cat tempfile | awk -F, '{if($1=="")$1='"$mean1"'}; {if($2=="")-
$2='"$mean2"'}; {if($3=="")$3='"$mean3"'}; {if($4=="")$4='"$mean4"'};
{print $0}' | tr ' ' ',' > ccfile
   cat ccfile   #print the output of substitute
   cp ccfile $filename    #save the result in the orginal file
}
```

Datafile:



Code results:



Discuss results and code:

In this case, we checked the columns and each column contains empty value, we calculate the mean and replace it in its empty value.

6) **menu.**

Code:

```
#menu to choose opration
while true
do
   printf "\nD: for dimension\nC: for compute statistics\nS: for substitution\>
   read choice           #read opration from user

   case $choice in       #case to do the opration which the user enter
     "D") dimensions;;   #call function to do opration
     "C") statistics;;
     "S") substitute;;
     "E") exit 0;;
     *) echo "Enter a valid choice";;    #defult input
   esac
done
```

Discuss results and code:

Here, the operation icon is read from the user and executed, and the operation code is repeated until the exit icon is entered

References:

1. https://www.youtube.com/watch?v=n8qz0wZ8Z0c
2. https://www.youtube.com/watch?v=tQ-oKC6KbiY&fbclid=IwAR3ivMCi8liYlkSPj0jaAD6H1HQHOR1lwVSA3Dq0hHCzHPmQRZE-TEVRGto
3. https://unix.stackexchange.com/questions/265119/how-to-format-floating-point-number-with-exactly-2-significant-digits-in-bash?fbclid=IwAR2mADLW6PKpJUEG_PzpFIVe3rjym0bT7jtC4oH0EH5OVH2zI3EwlPwqAO8
4. https://www.youtube.com/watch?v=yqpY-Wk-i9k
5. https://www.geekpills.com/automation/awk/awk-if-statement-examples?fbclid=IwAR07e3vI0ai74m0B9c8AQInl79LwOg2sarwZWni2-7witNJUeMmuBMYfV90
6. https://www.cyberciti.biz/faq/bash-for-loop/?fbclid=IwAR1fZzldZzoWmX6aFZn7FklbIchlumURm4X343Dm8fzuo5uXWqeLHumk3hQ