1. Print all the First Names.

Awk ‘{print $1}’ AwkLab.data

A screen shot of a computer

Description automatically generated

1a. Awk handles spaces are breaks in the fields, which is why im using ‘{print $1}. The $ would pull the field of the number next to it. $1 is the first field, $2 the second and so on. As the first field for this file is the first name, we use $1 which in turn provides us with the first names for these users.

2. Print phone numbers for Tom and Frodo after their names

Awk -F’:’ ‘/Tom|Frodo/ {print $1, $2} ‘ AwkLab.data

A black screen with white text

Description automatically generated

2a. By using the -F option, I can tell awk to set the field separator as colon. To search for either tom or Frodo, I insert a | which instructs awk to find the two entries between the forward slashes. In simple terms, | is an OR operator and I’m using it with Awk to match either name (tom or Frodo) against the query. /Tom|Frodo/ is stating that I am searching the file specifically for instances where the field tom or Frodo are present and asking for the first and second fields to be printed ($1 $2). The comma after the $1 is simply to create more clarity as it provides a line break (inserts a space), making the output more readable.

3. Print Peregrin’s full name and phone number area code only.

awk -F‘:’ ‘/Peregrin/ {print $1, $2}’ AwkLab.data

A black and white screen with text

Description automatically generated

3a. Similar to question 2, I targeted the first and second fields, separating the fields by a manually set parameter of a colon. The /Peregrin/ entry allows me to focus my search only to lines that contain the text that matches Peregrin, so in this case it only prints out one line. The $1 and $2 grabs just the first and second field, discarding the rest of the data in the line which prints just the Full name + phone number.

4. Print all phone numbers (full number) in the 123 area code along with the names

awk ‘/(123)/ {print}’ AwkLab.data

A screenshot of a computer

Description automatically generated

4a. This time we are targeting all 123 area codes. To achieve this, I filter out the lines that don’t contain a 123 area code with /(123)/, which searches for keyword matches and discards any lines that don’t contain a match, then print the lines with {print}. Formatting is extremely important in awk as queries only match to what they are instructed to match. In this case, the area code is three numbers surrounded b

y parentheses, so we need to ensure that our query matches that format (xxx)

5. Print all Last names beginning with either a T or D (careful of middle names!)

awk -F ':' '{split($1, name, " "); if (name[length(name)] ~ /^[TD]/) print name[length(name)]}' AwkLab.data

A computer screen with white text and green squares

Description automatically generated

5a. To find names where the last name starts with T or D, you use the key pattern /^[TD]/, which matches words starting with "T" or "D." Setting the field separator to : isolates the names, and the split function ensures the last word in the name segment is checked.

6. Print all first names containing four or less characters.

Awk ‘length($1) <=4 {print $1}’ AwkLab.data

A black screen with white text

Description automatically generated

6a. The length function takes the numerical value that is passed to it (in this case <=4) and applies it to the first field ($1). The function then scans the file and finds and lines that have a first field with 4 or less characters.

7. Print the first names and area codes of all those in the 916 area code.

Awk -F’[: ]’ ‘/(916)/{print $1 $3}’ AwkLab.data

A black screen with white text

Description automatically generated

7a. To find just the first names you need to filter by the specific keyword which in this instance is the area code 916 which we accomplish with /(916)/. Then we set our field separators to both an empty space and a colon to filter out all text after the first name, excluding our area code, and all text after our area code which we accomplish with -F ‘[: ]. Then we print out our results by looking at the two fields we need, $1 which is the first name and $3 which is the area code.

8. Print Sacharissa’s campaign contributions following her name. Each value should be printed with a leading dollar sign; e.g., $250 $100 $175.

awk -F':' '/Sacharissa/ {print $1 " $" $3 " $" $4 " $" $5}' AwkLab.data

A black screen with white text

Description automatically generated

8a. To add dollar symbols next to the value for Sacharissa’s contributions, we must first filter for lines with her contributions via /Sacharrisa/. Once found, we set our field separator to : via -F’:’ and print fields 1 ($1 her name), 3 ($3 for $250), 4 ($4 for $100), and 5 ($5 for $175) while attaching a “ character to the front of each field as we’re adding text in front of it. This allows us to print each field but with the required $ sign infront.

9. Print last names followed by a comma and the phone number. Be careful of the last names’s format.

awk -F ':' '{split($1, name, " "); print name[length(name)] "," $2 }' AwkLab.dataA black screen with white text

Description automatically generated

9a. To accomplish this task we creation two functions, one splits the first field into an array called name with the strings separated by a space. The other function prints the last string of the name array [length(name)]. Because the string is always the last value in the array, the last name is always printed. After we simply add a comma and print the second field which provides us with the phone number to attach.

10. Print the first and last names of those who contributed more than $110 in the last month. Make sure to include their last month contribution amount after the name

awk -F ':' '$5 > 110 {print $1, $5}' AwkLab.data

A screenshot of a computer

Description automatically generated

10a. To solve this, we must first set our field separator to colon. Once set, we want the last donation which in this case is field 5 ($5). So we search all lines fifth field for values larger than 110 and if found they will be included in the print output. The output contains $1 which is the first and last name field and $5 which is the final donation field.

11. Print the last names, phone numbers, and first month contribution of those who contributed less than $150 in the first month.

awk -F '[: ]' '$5 < 150 {print $2, $5}' AwkLab.data

A black screen with white text

Description automatically generated

11a. To solve this ,we must first set our field separators to both colon and a space so that we can split out this data into fields. Once set, we search our fifth field which is the first contribution field ($5) for values less than 150. Finally, we print the second field field which newly contains the last names along with the first contribution value.

12. Print the first names and contribution of those who contributed between $10 and $200 in the first month.

awk -F ':' '$3 >= 10 && $3 <= 200 {print $1, $3}' AwkLab.data

A black screen with white text

Description automatically generated

12a. To solve this we must first set our field separator to colon. Once set, we search all lines where our third field (initial contributions) is both higher than 10 but also lower than 200. To facilitate the and aspect of this query, we included && which represents and. After this, we simply print $1 which is the name field and $3 which is the initial donation field.

13. Print the first name, last names and total contributions of those who contributed less than $700 over the three-month period.

awk -F ':' '($3 + $4 + $5) < 700 {print $1, $3 + $4 + $5}' AwkLab.data

A black screen with white text

Description automatically generated

13a. First set the field separator, then prepare our query where we add $3+$4+$5 into a value for each line which then gets checked if it’s less than 700. For printing, we print the initial field (name) then we combine fields 3,4, and 5 into one field to represent the total contributions.

14. Print the first names and first letter of the last name, and average contribution of those who had an average contribution of more then $300

awk -F ':' '($3 + $4 + $5) / 3 > 300 {split($1, name, " "); print name[1], substr(name[2], 1, 1), ",", ($3 + $4 + $5) / 3}' AwkLab.data

A black screen with a black background

Description automatically generated

14a. First we set the field separator, then combine fields $3, $4, $5 for each line individually, averaging, then determining if that line’s average contribution is greater than 300. Then we use the split function to split the first field (full name) into an array called ‘name’ with two different strings (String 1 is first name & String 2 last name) separated by a space, then printing the first string of the array. Using the substr function, we extract just the first character from the second string (name[2], which is the last name) by specifying the starting position and length.

15. Print the last name and area code of those not in the 916 area code.

awk -F '[:()]' '($3 != "916") {split($1, name, " "); print name[2], $3}' AwkLab.data

A black screen with white text

Description automatically generated

15a. First we set the field separator which for this case is both colon and parentheses because of the format of the area code, as they are wrapped in (). Then we check the third field (which is the area code field) to make sure that its not a match of 916 (!=). After, we split the first field into an array called name, separated by the space character (“ “ means the space between characters). As we’re using two functions it requires a break so we insert a semi-colon and move onto printing the second string in our name array and 3 characters length (enough for a area code).

16. Print each record preceded by the number of the record.

awk ‘{print NR, $0}’ AwkLab.data

A screenshot of a computer

Description automatically generated

16a. To complete this task we first print the number of records (NR) which will list the number of line entries and include a $0 to print the entire line worth of data. Combining these, you get the number of records + the line data.

17. Print the name and total contribution of each person.

awk -F ':' '{print $1, $3 + $4 + $5}' AwkLab.data

A screenshot of a computer

Description automatically generated

17a. To complete the task we need to set our field separator and print the first field (full name). After we can use fields 3, 4, and 5 to generate our total contributions. To do so, we simply add $3, $4, and $5.

18. Add $10 to Tiffany Aching’s first contribution and print her full name and first contribution.

awk -F ':' '/Tiffany Aching/ {print $1, $3 + 10}' AwkLab.data



18a. First we set our field separator and search for our keyword /Tiffany Aching/. Then we simply print our first field containing the name of the user (print $1) and add 10 dollars to her first contribution with $3+10.

19. Change Samwise Gamgee’s name to Sean Astin

awk '{sub(/Samwise Gamgee/, "Sean Astin"); print}' AwkLab.data

A screenshot of a computer

Description automatically generated

19a. Using the sub function, we can replace text in a file with the following command {sub/oldtext/, “newtext”}. With this, we can take Samwise Gamgee and replace their name with Sean astin.

20. Write an awk script to do the following (MUST be an awk script not just a bash script or commands on the commandline)

(a) Prints first name of the all the Tooks followed by their total campaign contributions .

(b) Print the full names and contributions of anyone who contributed

between $10 and $200 in the last contribution

(c) Prints the full names and average contribution of those who contributed less than $300 on average

#!/usr/bin/awk -f

BEGIN { FS=":" }

{

split($1, name\_parts, " ")

#Names of the Tooks and combines their total contributions

if (name\_parts[2] == "Took" || name\_parts[3] == "Took") {

print "a: ", name\_parts[1], "Total: $" ($3 + $4 + $5)

}

#Names of contributors if they're greater or equal to 10 but less or equal to 200

if ($5 >= 10 && $5 <= 200) {

print "b: ", $1, "Last Contribution: $" $5

}

#Names of anyone who's average contribution is less than 300

total = $3 + $4 + $5

average = total / 3

if (average < 300) {

print "c: ", $1, "Average Contribution: $" average

}

}

A computer screen with text and numbers

Description automatically generated

A screenshot of a computer

Description automatically generated

20a. The first line splits the first field and then stores the contents of that split field into nameparts. After it checks the second and third part of that split field to see if it matches tooks. If matched, it prints the first field (their name) and then combine their contributions ($3+$4+$5) into on value.

20b. The second line checks if the last contribution (represented by $5) is greater (>=) than 10 but less than 200. If it matches, it prints the attaches B.) infront of the first field (name) then Last contribution with the value of the last contribution ($5).

20c. This entry checks each line to see IF the average of the contributions (calculated by summing and dividing $3, $4, and $5) is less than 300. If this condition is true, it prints the first field ($1) along with the calculated average. Since there is no else or else if statement, the print only happens if the condition is met; otherwise, nothing is printed.

Work Cited

<https://www.computerworld.com/video/509010/how-to-use-the-awk-command-2-minute-linux-tips.html>

<https://www.youtube.com/watch?v=9YOZmI-zWok>

<https://www.grymoire.com/Unix/Awk.html#uh-45>

<https://www.gnu.org/software/gawk/manual/html_node/Options.html>

<https://flylib.com/books/en/4.356.1.52/1/>

https://www.aholdengouveia.name/LinuxAdmin/Awk.html