**[awk](http://unix-school.blogspot.in/search/label/awk" \t "_blank)**is very powerful when it comes for file formatting.  In this article, we will discuss some wonderful grouping features of awk. awk can group a data based on a column or field , or on a set of columns. It uses the powerful associative array for grouping. If you are new to awk, this article will be easier to understand if you can go over the article how to [parse a simple CSV file using awk](http://unix-school.blogspot.in/2011/05/awk-read-file-and-split-contents.html).

Let us take a sample CSV file with the below contents. The file is kind of an expense report containing items and their prices. As seen, some expense items  have multiple entries.

$ cat file

Item1,200

Item2,500

Item3,900

Item2,800

Item1,600

**1. To find the total of all numbers in second column**. i.e, to find the sum of all the prices.

$ awk -F"," '{x+=$2}END{print x}' file

3000

    The delimiter(-F) used is comma since its a comma separated file. *x+=$2* stands for *x=x+$2*. When a line is parsed, the second column($2) which is the price, is added to the variable x. At the end, the variable x contains the sum. This example is same as discussed in the awk example of finding the [sum of all numbers in a file](http://unix-school.blogspot.in/2011/09/5-different-ways-to-add-all-numbers-in.html).

   If your input file is a text file with the only difference being the comma not present in the above file, all you need to make is one change. Remove this part from the above command: -F","  . This is because the default delimiter in awk is whitespace.

**2. To find the total sum of particular group entry alone.** i.e, in this case, of "Item1":

$ awk -F, '$1=="Item1"{x+=$2;}END{print x}' file

800

  This gives us the total sum of all the items pertaining to "Item1". In the earlier example, no condition was specified since we wanted awk to work on every line or record. In this case, we want awk to work on only the records whose first column($1) is equal to *Item1*.

**3**. If the data to be worked upon is present in a shell variable:

$ VAR="Item1"

$ awk -F, -v inp=$VAR '$1==inp{x+=$2;}END{print x}' file

800

   -v is used to [pass the shell variable to awk](http://unix-school.blogspot.in/2011/09/awk-passing-arguments-or-shell.html), and the rest is same as the last one.  
  
**4. To find unique values of first column**

$ awk -F, '{a[$1];}END{for (i in a)print i;}' file

Item1

Item2

Item3

**Arrays in awk** are associative and is a very powerful feature. Associate arrays have an index and a corresponding value. Example: *a["Jan"]=30* meaning in the array *a,* "Jan" is an index with value 30. In our case here, we use only the index without values. So, the command *a[$1]* works like this: When the first record is processed, in the array named *a,* an index value "Item1" is stored. During the second record, a new index "Item2", during third "Item3" and so on. During the 4th record, since the "Item1" index is already there, no new index is added and the same continues.

  Now, once the file is processed completely, the control goes to the END label where we print all the index items. for loop in awk comes in 2 variants: 1. The C language kind of for loop,  Second being the one used for associate arrays.

*for i in a* : This means for every index in the array *a* . The variable "i" holds the index value. In place of "i", it can be any variable name. Since there are 3 elements in the array, the loop will run for 3 times, each time holding the value of an index in the "i". And by printing "i", we get the index values printed.

 To understand the for loop better, look at this:

for (i in a)

{

print i;

}

Note: The order of the output in the above command may vary from system to system. Associative arrays do not store the indexes in sequence and hence the order of the output need not be the same in which it is entered.  
  
**5. To find the sum of individual group records.** i.e, to sum all records pertaining to Item1 alone, Item2 alone, and so on.

$ awk -F, '{a[$1]+=$2;}END{for(i in a)print i", "a[i];}' file

Item1, 800

Item2, 1300

Item3, 900

   a[$1]+=$2 . This can be written as a[$1]=a[$1]+$2. This works like this: When the first record is processed, a["Item1"] is assigned 200(a["Item1"]=200). During second "Item1" record, a["Item1"]=800 (200+600) and so on. In this way, every index item in the array is stored with the appropriate value associated to it which is the sum of the group.

   And in the END label, we print both the index(i) and the value(a[i]) which is nothing but the sum.

**6. To find the sum of all entries in second column and add it as the last record**.

$ awk -F"," '{x+=$2;print}END{print "Total,"x}' file

Item1,200

Item2,500

Item3,900

Item2,800

Item1,600

Total,3000

   This is same as the first example except that along with adding the value every time, every record is also printed, and at the end, the "Total" record is also printed.  
  
**7. To print the maximum or the biggest record of every group**:

$ awk -F, '{if (a[$1] < $2)a[$1]=$2;}END{for(i in a){print i,a[i];}}' OFS=, file

Item1,600

Item2,800

Item3,900

     Before storing the value($2) in the array,  the current second column value is compared with the existing value and stored only if the value in the current record is bigger. And finally, the array will contain only the maximum values against every group. In the same way, just by changing the "lesser than(<)" symbol to greater than(>), we can find the smallest element in the group.

The syntax for if in awk is, similar to the C language syntax:

*if (condition)*  
*{*   
*<code for true condition* >  
*}else{*   
*<code for false condition>*  
 *}*  
  
**8. To find the count of entries against every group**:

$ awk -F, '{a[$1]++;}END{for (i in a)print i, a[i];}' file

Item1 2

Item2 2

Item3 1

    a[$1]++ : This can be put as a[$1]=a[$1]+1. When the first "Item1" record is parsed, a["Item1"]=1 and every item on encountering "Item1" record, this count is incremented, and the same follows for other entries as well. This code simply increments the count by 1 for the respective index on encountering a record. And finally on printing the array, we get the item entries and their respective counts.

**9. To print only the first record of every group**:

$ awk -F, '!a[$1]++' file

Item1,200

Item2,500

Item3,900

    A little tricky this one. In this awk command, there is only condition, no action statement. As a result, if the condition is true, the current record gets printed by default.

 !a[$1]++ : When the first record of a group is encountered, a[$1] remains 0 since ++ is post-fix, and not(!) of 0 is 1 which is true, and hence the first record gets printed. Now,  when the second records of "Item1" is parsed, a[$1] is 1 (will become 2 after the command since its a post-fix). Not(!) of 1 is 0 which is false, and the record does not get printed. In this way, the first record of every group gets printed.

   Simply by removing '!' operator, the above command will print all records other than the first record of the group.

**10. To join or concatenate the values of all group items**. Join the values of the second column with a colon separator:

$ awk -F, '{if(a[$1])a[$1]=a[$1]":"$2; else a[$1]=$2;}END{for (i in a)print i, a[i];}' OFS=, file

Item1,200:600

Item2,500:800

Item3,900

     This if condition is pretty simple: If there is some value in a[$1], then append or concatenate the current value using a colon delimiter, else just assign it to a[$1] since this is the first value.

To make the above if block clear, let me put it this way:  "if (a[$1])"  means "if a[$1] has some value".

if(a[$1])

 a[$1]=a[$1]":"$2;

else

 a[$1]=$2

  The same can be achieved using the awk ternary operator as well which is same as in the C language.

$ awk -F, '{a[$1]=a[$1]?a[$1]":"$2:$2;}END{for (i in a)print i, a[i];}' OFS=, file

Item1,200:600

Item2,500:800

Item3,900

Ternary operator is a short form of if-else condition. An example of ternary operator is: x=x>10?"Yes":"No"  means if x is greater than 10, assign "Yes" to x, else assign "No".

In the same way: a[$1]=a[$1]?a[$1]":"$2:$2  means if a[$1] has some value assign a[$1]":"$2 to a[$1] , else simply assign $2 to a[$1].

**Concatenate variables in awk**:  
One more thing to notice is the way string concatenation is done in awk. To concatenate 2 variables in awk, use a space in-between.  
Examples:

z=x y    #to concatenate x and y

z=x":"y  #to concatenate x and y with a colon separator.