**Text Processing Commands:**

1. **Searching and replacing text in a file**

Using sed command to search for and replace specific text in a text file

**Scenario 1:**

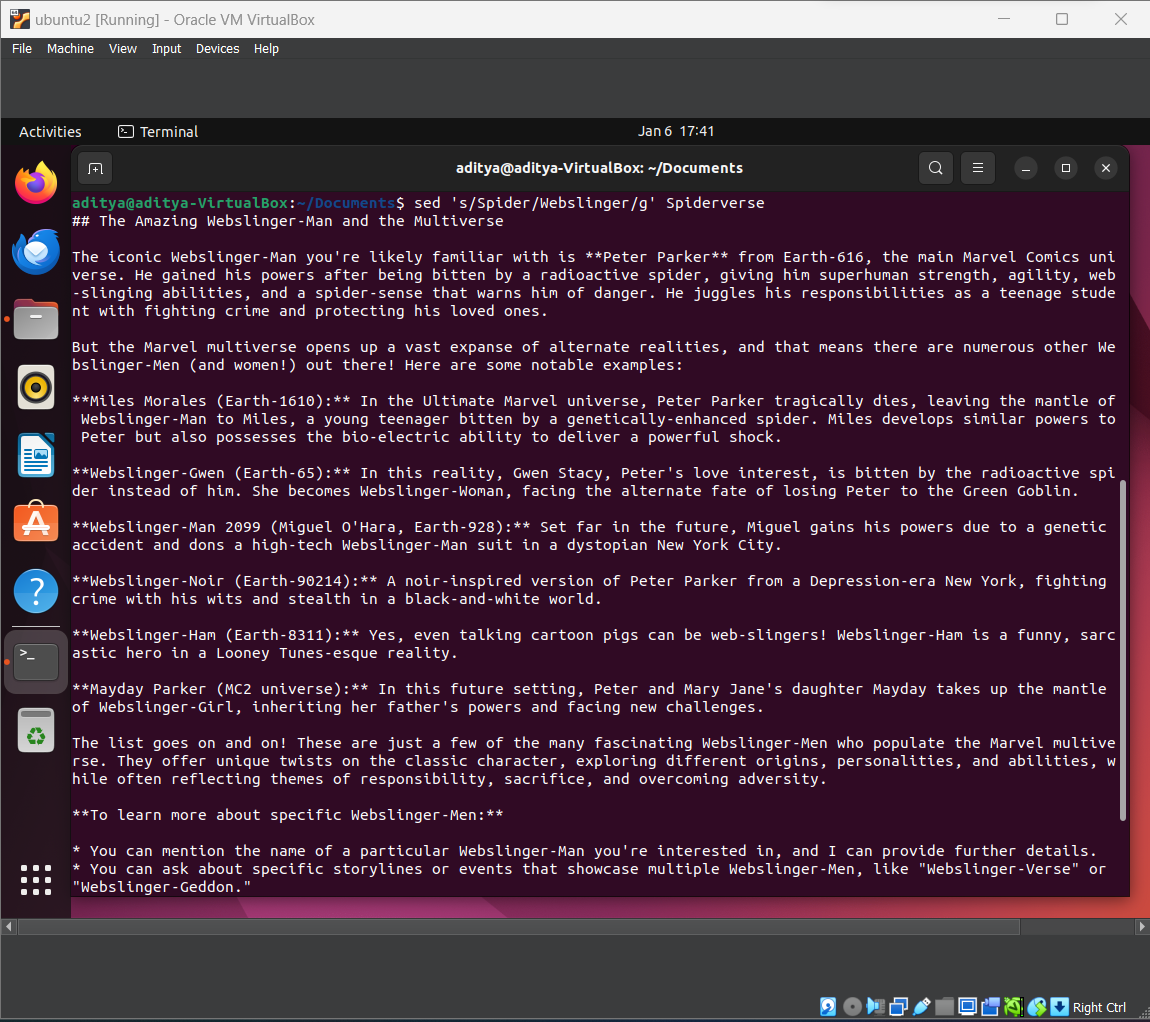
* Search pattern: "Spider"
* Replacement text: "Webslinger"
* Filename: Spiderverse.txt

Command:

sed 's/ Spider / Webslinger /g' Spiderverse.txt

This command will replace every occurrence of the word "Spider" with "Webslinger" in the file "Spiderverse.txt".

Now, the output:



**Scenario 2:**

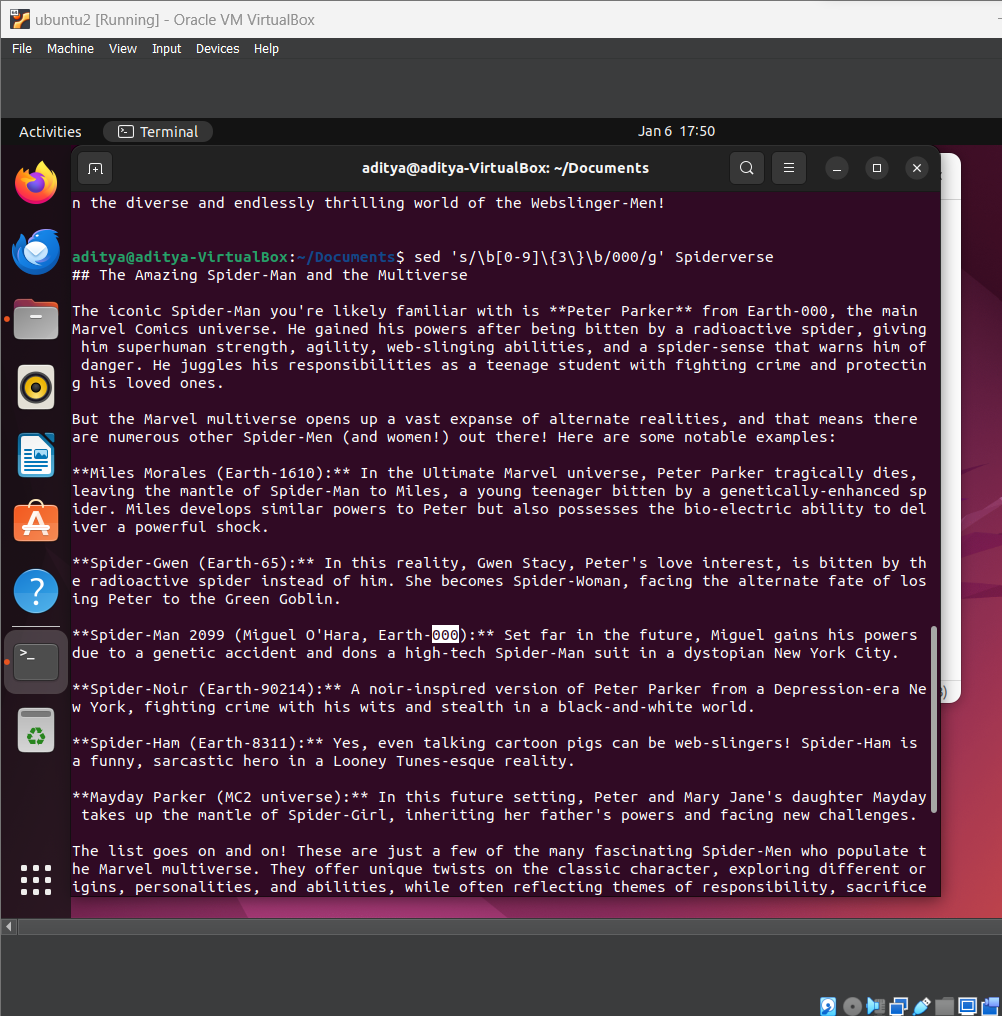
* Search pattern: Any three-digit number (using a regular expression)
* Replacement text: "000"
* Filename: Spiderverse.txt

Command:

sed 's/\b[0-9]\{3\}\b/000/g' Spiderverse.txt

This command would replace all three-digit numbers in the file "Spiderverse.txt" with "000".

Output:



**Regular expressions** (regex) are a powerful tool for searching, matching, and manipulating text patterns. They provide a concise way to describe complex patterns using special characters and metacharacters.

In the above scenario the regular expression in the sed command:

1. sed 's/.../.../g' Spiderverse.txt:

* sed: Stream editor for modifying text files.
* s: Substitute command for search and replace.
* /.../.../g: Delimiters for the search pattern and replacement text, with g for global replacement (all occurrences).
* Spiderverse.txt: The name of the file to modify.

2. \b[0-9]\{3\}\b: The regular expression search pattern:

* \b: Word boundary, ensuring a complete three-digit number, not part of a larger word.
* [0-9]: Character class matching any single digit (0-9).
* \{3\}: Quantifier specifying exactly three repetitions of the preceding digit.
* \b: Word boundary again, ensuring a distinct number.

3. 000: The replacement text.

Together, the command replaces all standalone three-digit numbers in the file "Spiderverse.txt" with "000".

Key points:

* Word boundaries: \b ensures only whole numbers are replaced, not partial numbers within words.
* Character class: [0-9] matches any digit.
* Quantifier: \{3\} specifies exactly three digits.
* Global replacement: g flag replaces all occurrences in each line.

Example:

* If a line in "Spiderverse.txt" contains "Miles lives in Universe 610", it would become "Miles lives in Universe 000" after the replacement.

1. **Extracting and manipulating data from a CSV file:**

Key Concepts:

* Field Separator: awk uses the default field separator of spaces. For CSV files, set it to the comma (,):

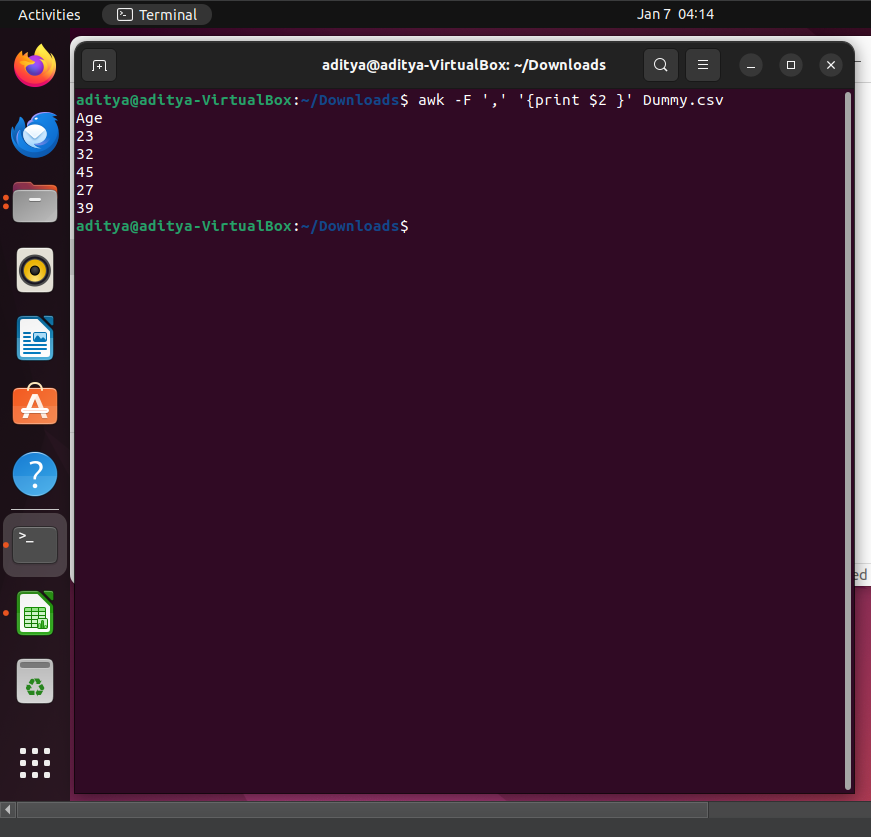
awk -F ',' '{ ... }' filename.csv

* Accessing Fields: $1, $2, $3, etc., represent fields in each line.
* Actions: Enclose actions within curly braces {}.
* Built-in Variables: NR (number of records), NF (number of fields), more.

Examples:

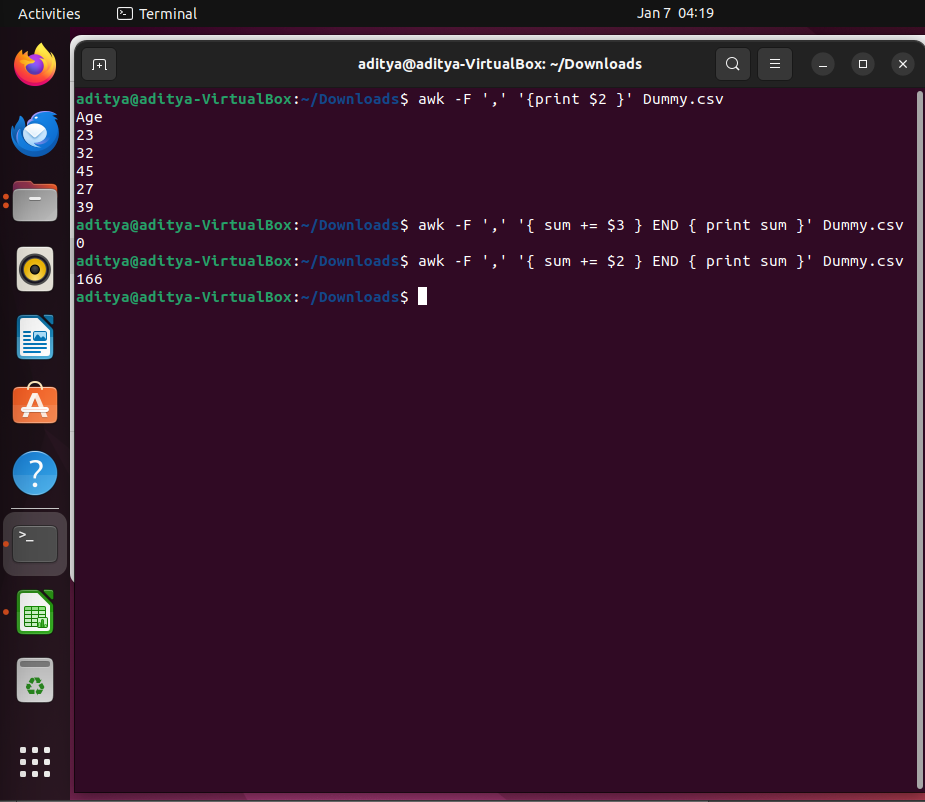
1. Print the second column:

awk -F ',' '{ print $2 }' Dummy.csv



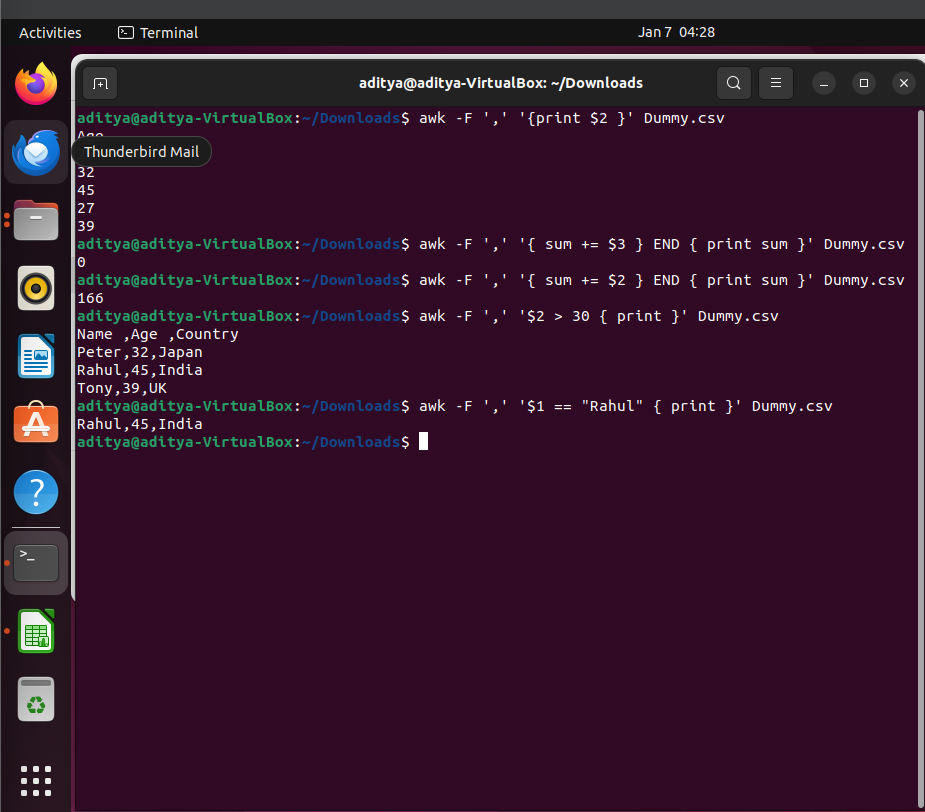
1. Sum the values in the second column:

awk -F ',' '{ sum += $2 } END { print sum }' Dummy.csv



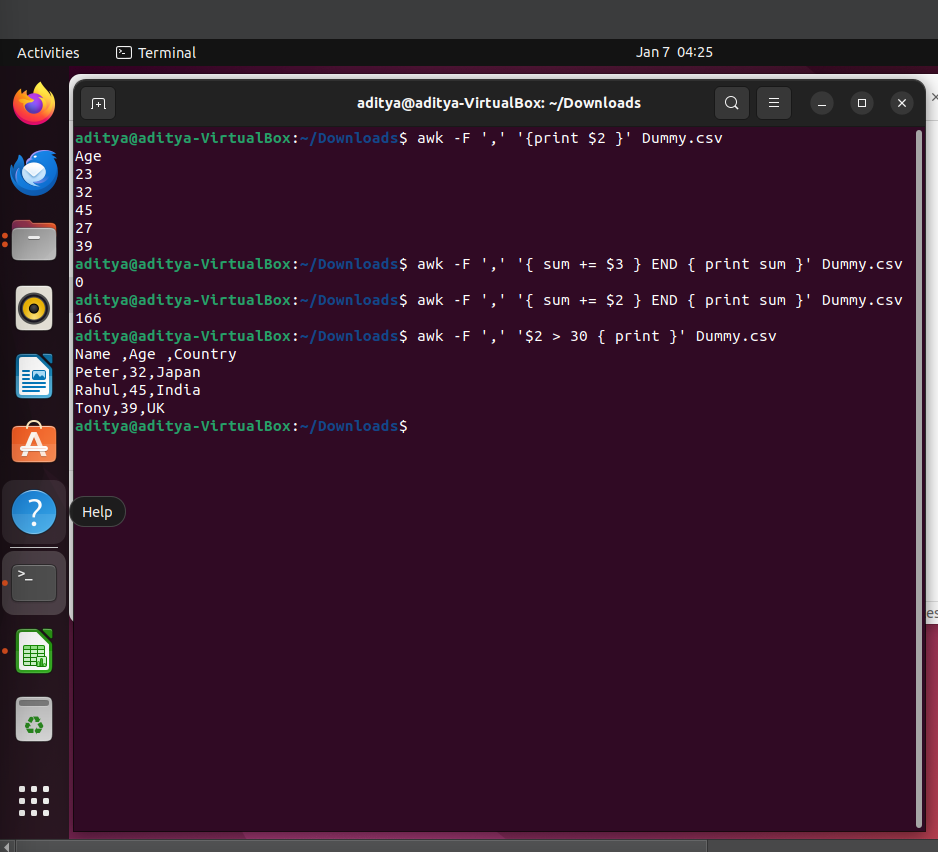
1. Filter lines where the first column is "Rahul":

awk -F ',' '$1 == "Rahul" { print }' Dummy.csv



1. Print lines where the second column is greater than 30:

awk -F ',' '$2 > 30 { print }' Dummy.csv



1. Print the first and last fields, separated by a colon:

awk -F ',' '{ print $1 ":" $NF }' Dummy.csv

