

Unix Assignment

Name: Taklakoglou Chidiroglou Argyrios

Student ID: f2822114

What is the data's field separator character?

1 point

Result

,

Πίσω

Επόμενο

Εκκαθάριση



Σελίδα 2 από 25

What happens when you try to load the data into a spreadsheet?

1 point

Result

oasa-history.xls (this dataset is too large for the Excel grid. If you save the workbook, you will lose data that wasn't loaded.)

The command `cat oasa-history >>oasa-history.xls` display the data-records from oasa-history file and creates a xls file with name oasa-history.xls that contains the data-records from oasa-history.

Spreadsheet program(s) used

`$ cat oasa-history >>oasa-history.xls`

Πίσω

Επόμενο

Εκκαθάριση



Σελίδα 3 από 25

How many records are provided with the data?

1 point

Answer

8699446


Command used

`$ wc -l oasa-history`

The command `wc -l` counts the lines-records in `oasa-history` file

[Πίσω](#)

[Επόμενο](#)

 Σελίδα 4 από 25

What is the data acquisition time stamp of the last record in the stream you fetched?

1 point

Answer *

2022-03-08T12:32:16

Command used

```
$ tail -1 oasa-history | awk '{print $1}'
```

The command `tail -1` returns as output the last record from `oasa-history`. Through the pipeline, the output of the 1st command is connected via a pipe to the standard input of `awk '{print $1}'` command. The `awk '{print $1}'` choose and display the 1st field from the last record which is the last time stamp.

Πίσω

Επόμενο

Εκκαθάριση

Σελίδα 5 από 25

How many different buses appear in the data?

1 point

Answer

1585

Command used

```
$ awk -F, '{print $3}' oasa-history | sort -u | uniq | wc -l
```

The AWK language is useful for manipulation of data files, text retrieval and processing. The symbol - is the interpreter for the AWK Programming Language. The -F, option tells awk to use comma (,) as field separator. The command '{print \$3}' means print the third field (the fields being separated by ,) from the oasa-history file. The output of awk -F, '{print \$3}' oasa-history is connected via a pipe to the standard input of sort -u command. This command sorts and remove duplicates. So in this command duplicates are removed. The output of sort -u is connected via a pipe to the standard input of uniq command which is optional because I will get the same output. I used uniq command to ensure that have been kept only the unique buses.(Uniq is essential command if i use sort and not sort -u. So I should use sort | uniq or sort -u but if I use sort -u | uniq doesn't make any difference in the output) The output of uniq is connected via a pipe to the standard input of wc -l command. The wc -l command counts the lines-unique buses.

[Πίσω](#)

[Επόμενο](#)

[Εκκαθάριση](#)

Σελίδα 6 από 25

How many different routes are covered by the data?

1 point

Answer

471

Command used

```
awk -F, '{print $2}' oasa-history | sort -u | uniq | wc -l
```

The AWK language is useful for manipulation of data files, text retrieval and processing. The symbol - is the interpreter for the AWK Programming Language. The option -F, tells awk to use comma (,) as field separator. The command '{print \$2}' means print the second field (the fields being separated by ,) from the oasa-history file. The output of awk -F, '{print \$2}' oasa-history is connected via a pipe to the standard input of sort -u command. This command sorts and remove duplicates. The output of sort -u is connected via a pipe to the standard input of uniq command. I used uniq to ensure that have been kept only the unique routes. (Uniq is essential command if i use sort and not sort -u. So I should use sort | uniq or sort -u but if I use sort -u | uniq doesn't make any difference in the output). The output of uniq is connected via a pipe to the standard input of wc -l command. The wc -l command counts the lines-unique routes.

Πίσω

Επόμενο

Εκκαθάριση

Σελίδα 7 από 25

How many dates are covered by the data?

2 points

Answer

67

Command used

```
$ cut -b -10 oasa-history | sort | uniq | wc -l
```

The cut command is a command-line utility that allows to cut parts of lines from files. The option -b select by specifying a byte, a set of bytes, or a range of bytes when cutting out selected portions. In this case is a range of bytes. -10 means that I want to cut from 1st up to 10th byte from oasa-history. The output of cut -b -10 oasa-history is connected via a pipe to the standard input of sort command. This command sorts the output from the 1st command. The output of sort is connected via a pipe to the standard input of uniq command. (Uniq is essential command if I use sort and not sort -u. So I should use sort | uniq or sort -u but if I use sort -u | uniq doesn't make any difference in the output). The output of uniq is connected via a pipe to the standard input of wc -l command. The wc -l command counts the lines- unique dates.

Πίσω

Επόμενο

Εκκαθάριση

Σελίδα 8 από 25

Which route is associated with the most position reports?

2 points

Answer

3609

Command used

```
$ awk -F, '{print $2}' oasa-history | sort | uniq -c | sort -r | head -1
```

-c – -count : It tells how many times a line was repeated by displaying a number as a prefix with the line

The AWK language is useful for manipulation of data files, text retrieval and processing. The symbol `awk` is the interpreter for the AWK Programming Language. The `-F`, tells `awk` to use comma (,) as field separator. The command `'{print $2}'` means print the second field (the fields being separated by ,) from the `oasa-history` file. The output of `awk -F, '{print $2}' oasa-history` is connected via a pipe to the standard input of `sort` command. This command sorts the output from the 1st command. The output of `sort` is connected via a pipe to the standard input of `uniq -c` command. `-c` option tells how many times a line was repeated by displaying a number as a prefix with the line. The output of `uniq -c` is connected via a pipe to the standard input of `sort -r` command. The `sort -r` command sort the input in descending order(`-r` option in `sort` means reverse the sorting order). The output of `sort -r` is connected via a pipe to the standard input of `head -1` command. The `head -1` command choose the 1st line from the input.

125506(times appeared) 3609(route)

Πίσω

Επόμενο

Σελίδα 9 από 25

Εκκαθάριση

How many position reports appear in the data more than once?

2 points

Answer

2278527

Command used

```
awk -F, '{print $4}' oasa-history | sort | uniq -c | awk '($1>1)' | wc -l
```

The AWK language is useful for manipulation of data files, text retrieval and processing. The symbol - is the interpreter for the AWK Programming Language. The option -F, tells awk to use comma (,) as field separator. The command '{print \$4}' means print the fourth field (the fields being separated by ,) from the oasa-history file. The output of awk -F, '{print \$4}' oasa-history is connected via a pipe to the standard input of sort command. This command sorts the output from the 1st command. The output of sort is connected via a pipe to the standard input of uniq -c command. -c option tells how many times a line was repeated by displaying a number as a prefix with the line. The output of uniq -c is connected via a pipe to the standard input of awk '(\$1>1)' command. This command check if the 1st field of the input is greater than 1 or not. This means if this line have been found in the reports more than 1 time. If the line is duplicate (exist more than 1 time through the data) is kept anotherwise is exluded. So the output of awk '(\$1>1)' will contain the duplicate position reports. The output of awk '(\$1>1)' is connected via a pipe to the standard input of wc -l command. wc -l command counts the lines. Or equivalent counts how many position reports appear in the data more than once.

Πίσω

Επόμενο

Εκκαθάρις

Σελίδα 10 από 25

Which is the most frequent first two-digit sequence in numbers assigned to buses?

3 points

First two digits

30

Number of bus numbers starting with these digits

247

Command used

1)First two digits:

```
$ awk -F, '{print $3}' oasa-history | cut -b -2 | sort | uniq -c | sort -r
```

The AWK language is useful for manipulation of data files, text retrieval and processing. The symbol `-` is the interpreter for the AWK Programming Language. The option `-F`, tells awk to use comma (,) as field separator. The command `'{print $3}'` means print the third field (the fields being separated by ,) from the oasa-history file. The output of `awk -F, '{print $3}' oasa-history` is connected via a pipe to the standard input of `cut -b -2` command. This command cuts the first 2 bytes from the output lines of the 1st command in order to have the first two-digit sequence in the numbers assigned to the buses. The output of `cut -b -2` is connected via a pipe to the standard input of `sort` command. This command sorts the output from the 2nd command. The output of `sort` is connected via a pipe to the standard

output from the 2nd command. The output of sort is connected via a pipe to the standard input of uniq -c command. -c option tells how many times each two-digits numbers was repeated by displaying a number as a prefix with the line. The output of uniq -c is connected via a pipe to the standard input of sort -r command. The sort -r command sort the input in descending order(-r option in sort means reverse the sorting order)

OR equivalent:

```
$ cut -b 26,27 oasa-history | sort | uniq -c | sort -nr
```

It cuts only the 2 bytes corresponds to the 2 two-digits in the bus number instead of take only the 3rd field first and then cut the first two digits.

2)Number of bus numbers starting with these digits:

```
$ awk -F, '{print $3}' oasa-history | grep '^30' | sort | uniq -c | wc -l
```

The AWK language is useful for manipulation of data files, text retrieval and processing. The symbol - is the interpreter for the AWK Programming Language. The option -F, tells awk to use comma (,) as field separator. The command '{print \$3}' means print the third field (the fields being separated by ,) from the oasa-history file. The output of awk -F, '{print \$3}' oasa-history is connected via a pipe to the standard input of grep '^30' command. This command return the buses in which the bus number starts with the two digits '30' in order to choose the buses we want. The output of grep '^30' is connected via a pipe to the standard input of sort command. The sort command sort the output of the 2nd command. The output of sort is connected via a pipe to the standard input of uniq -c command. -c option tells how many times a bus was repeated by displaying a number as a prefix with the line. So the output of this command contains the unique buses that starts with 30. The output of uniq -c is connected via a pipe to the standard input of wc -l command. wc -l command counts the number of bus numbers starting with these digits.

OR equivalent:

```
$awk -F, '{print $3}' oasa-history | grep '^30' | sort -u | wc -l
```

How many buses did not travel on this year's February 6th?

4 points

Answer

930

Command used

```
$ awk -F, '{print $3}' oasa-history | sort -u > unique_buses
```

This command choose the 3rd field (contains the number of the buses) with `awk -F, '{print $3}' oasa-history`, then sort the result of the 1st command and keeps only the unique buses and we save the output of this command to a file with name `unique_buses`. So the file `unique_buses` contains all the buses in the `oasa-history` file.

```
$ grep '02-06' oasa-history | awk -F, '{print $3}' | sort -u > unique_buses_02-06
```

This command keep the lines that contain the '02-06' (with `grep '02-06' oasa-history`). That means that it keeps only the reports from 6th February. Then we keep only the 3rd field containing the 5-digit buses number. That means that contains the buses that traveled on this year's February 6th. Because I want the unique buses that traveled in February 6th so I use `sort -u` command. The output of this line is kept in a file with name `unique_buses_02-06`. So the file `unique_buses_02-06` contain the unique buses that traveled on this year's February 6th.

```
$ comm -23 unique_buses unique_buses_02-06 | wc -l
```

In order to obtain how many buses didn't travel I subtract the total unique buses in `oasa-history` file with the unique buses that traveled in February 6th by using the command `comm -23`.

Option -2: suppresses lines that appear only in `unique_buses_02-06`.

Option -3: suppresses lines that appear both in `unique_buses` and `unique_buses_02-06`.

Then I use `wc -l` to count the buses that didn't match. These buses are the buses that didn't travel on this year's February 6th.

Πίσω

Επόμενο

Εκκαθάριση

Σελίδα 12 από 25

On which date were the most buses on the road?

3 points

Answer

Provide the date in ISO format (YYYY-MM-DD)

2022-02-03

Command used

```
$ cut -d ',' -f 1,3 oasa-history | cut -b -10,20- | sort -u | awk 'BEGIN { FS=OFS=","; }{arr[$1]+=1 }END {for (i in arr) print i,arr[i]} | tr -s ',' $'\t' | sort -k 2n | tail -1
```

The `cut -d, -f 1,3 oasa-history` command, cuts the 1st and 3rd field from `oasa-history`. That means that we cut the data acquisition time stamp field and the bus number field. `-d` option specifies a delimiter that will be used instead of the default "TAB" delimiter. So `-d,` means that the delimiter will be the comma. The output of `cut -d ',' -f 1,3 oasa-history` is connected via a pipe to the standard input of `cut -b -10,20-` command. This command cuts the bytes 1 up to 10 and the bytes 20 up to the end. In other words, it keeps the date in YYYY-MM-DD format in the 1st field, the comma and the bus number field. The output of `cut -b -10,20-` is connected via a pipe to the standard input of `sort -u` command. `Sort -u` command keeps only the unique values. That means that if there are many reports for a specific bus in a specific day, this bus will be kept only once. In other words, this command keeps only the unique buses number for each day. The output of `sort -u` is connected via a pipe to the standard input of `awk 'BEGIN { FS=OFS=","; }{arr[$1]+=1 }END {for (i in arr) print i,arr[i]}'` command. `Awk FS` variable is used to set the field separator for each record. `Awk OFS` is an output equivalent of `awk FS` variable. By default `awk OFS` is a single space character. `BEGIN` means that `awk` will execute the action specified in `BEGIN` once before any input lines are read. `END` means that `awk` will execute the action specified in `END` before it actually exits. This command counts how many buses was on the road in each date. (it counts how many

This command counts how many buses was on the road in each date.(it counts how many times each date appears and it's time add 1). In END it depicts the dates and the number of buses that were on the road in each date which is the output of this command. The output of `awk 'BEGIN { FS=OFS="," }{arr[$1]+=1 }END {for (i in arr) print i,arr[i]}'` is connected via a pipe to the standard input of `tr -s ',' '\t'` command. This command separates the dates and the number of buses that were on the road in each date with a tab instead of comma that was previously. The output of `tr -s ',' '\t'` is connected via a pipe to the standard input of `sort -k 2n` command. `Sort -k 2n` sort the output based on the 2nd field(the number of buses that were on the road in each date) in ascending order. The output of `sort -k 2n` is connected via a pipe to the standard input of `tail -1` command. `Tail -1` command keeps only the last row from the input which is the result. (2022-02-03 1224)

[Πίσω](#)

[Επόμενο](#)

[Εκκαθάριση](#)

Σελίδα 13 από 25

Which route has been served by the highest number of different buses?

3 points

Route number

3566

Number of different buses on that route

226

Command used

```
$ cut -d ',' -f 2,3 oasa-history | sort -u | awk 'BEGIN { FS=OFS="," } {arr[$1]+=1 } END {for (i in arr) print i,arr[i]} ' | tr -s ',' '\t' | sort -k 2n | tail -1
```

The `cut -d, -f 2,3 oasa-history` command, cuts the 2nd and 3rd field from `oasa-history`. That means that we cut the route field and the bus number field. `-d` option specifies a delimiter that will be used instead of the default "TAB" delimiter. So `-d,` means that the delimiter will be the comma. The output of `cut -d ',' -f 2,3 oasa-history` is connected via a pipe to the standard input of `sort -u` command. This command keeps only the unique values and sort based on the 1st field. That means that if there are many reports for a specific bus in a specific route, this bus will be kept only once for this route. In other words, this command keeps only the unique buses number for each route. The output of `sort -u` is connected via a pipe to the standard input of `awk 'BEGIN { FS=OFS="," } {arr[$1]+=1 } END {for (i in arr) print i,arr[i]}'` command. `Awk FS` variable is used to set the field separator for each record. `Awk OFS` is an output equivalent of `awk FS` variable. By default `awk OFS` is a single space character. `BEGIN` means that `awk` will execute the action specified in `BEGIN` once before any

character. BEGIN means that awk will execute the action specified in BEGIN once before any input lines are read. END means that awk will execute the action specified in END before it actually exits. This command counts how many unique buses serves its route. Or in other words the number of unique buses that "use" each route. The output of `awk 'BEGIN { FS=OFS="," } {arr[$1]+=1 } END {for (i in arr) print i,arr[i]}'` is connected via a pipe to the standard input of `tr -s ',' '\t'` command. This command separates the dates and the number of buses that were on the road in each date with a tab instead of comma that was previously. The output of `tr -s ',' '\t'` is connected via a pipe to the standard input of `sort -k 2n` command. `Sort -k 2n` sort the output based on the 2nd field(the number of buses that serve its route) in ascending order. The output of `sort -k 2n` is connected via a pipe to the standard input of `tail -1` command. `Tail -1` command keeps only the last row from the input which is the result.

On which hour of the day (e.g. 09) are there overall the most buses on the road?

3 points

Hour

08

Number of buses

1584

Command used

```
$ cut -d ',' -f 1,3 oasa-history | cut -b 12-13,20- | sort -u | awk 'BEGIN { FS=OFS="," }{arr[$1]+=1 }END {for (i in arr) print i,arr[i]}' | tr -s ',' '\t' | sort -k 2n | tail -2
```

The `cut -d, -f 1,3 oasa-history` command, cuts the 1st and 3rd field from `oasa-history`. That means that we cut the data acquisition time stamp field and the bus number field. `-d` option specifies a delimiter that will be used instead of the default "TAB" delimiter. So `-d,` means that the delimiter will be the comma. The output of `cut -d ',' -f 1,3 oasa-history` is connected via a pipe to the standard input of `cut -b 12-13,20-` command. This command cuts the bytes 12,13 and the bytes 20 up to the end. In other words, it keeps the hours from the 1st field, the comma and the bus number field. The output of `cut -b 12-13,20-` is connected via a pipe to the standard input of `sort -u` command. `Sort -u` command keeps only the unique values. That means that if there are many reports for a specific bus in a specific hour, this bus will be kept only once for this hour. In other words, this command keeps only the unique buses number for each hour. The output of `sort -u` is connected via a pipe to the standard input of `awk 'BEGIN { FS=OFS="," }{arr[$1]+=1 }END {for (i in arr) print i,arr[i]}'` command. `Awk`

input of awk 'BEGIN { FS=OFS=","}{arr[\$1]+=1 }END {for (i in arr) print i,arr[i]}' command. Awk FS variable is used to set the field separator for each record. Awk OFS is an output equivalent of awk FS variable. By default awk OFS is a single space character. BEGIN means that awk will execute the action specified in BEGIN once before any input lines are read. END means that awk will execute the action specified in END before it actually exits. This command counts how many buses are on the road each hour. In END it depicts the hours and the number of buses that were on the road in each hour which is the output of this command. The output of awk 'BEGIN { FS=OFS=","}{arr[\$1]+=1 }END {for (i in arr) print i,arr[i]}' is connected via a pipe to the standard input of tr -s ',' '\t' command. This command separates the dates and the number of buses that were on the road in each date with a tab instead of comma that was previously. The output of tr -s ',' '\t' is connected via a pipe to the standard input of sort -k 2n command. Sort -k 2n sort the output based on the 2nd field(the number of buses that were on the road in each hour) in ascending order. The output of sort -k 2n is connected via a pipe to the standard input of tail -2 command. Tail -2 command keeps only the two last row from the input which is the result. I kept the 2 last rows and not only the last because I observed that the hour 07 and 08 there are the same number of buses on the road. (1584).|

On which hour of the day (e.g. 23) are there overall the fewest buses on the road?

3 points

Hour

03

Number of buses

728

Command used

```
$ cut -d ',' -f 1,3 oasa-history | cut -b 12-13,20- | sort -u | awk 'BEGIN { FS=OFS="," } {arr[$1]+=1 } END {for (i in arr) print i,arr[i]} ' | tr -s ',' '\t' | sort -k 2n | head -1
```

I executed the same command with the previous question but instead of taking the last row, I took the 1st row because I want the hour with the fewest buses.

For which weekday (e.g. Wednesday) does your data set contain the most records?

Consider using the (GNU) awk strftime and mktime functions to convert between date formats. (5 bonus points — you can get a 10/10 without answering this question.)

Weekday

Η απάντησή σας

Number of records in the data set matching the corresponding weekday

Η απάντησή σας

Command used

Η απάντησή σας

Πίσω

Επόμενο

Σελίδα 17 από 25

Εκκαθάριση

φύλλου

What are the bounding box geographic coordinates of the area served by the buses?

3 points

Bounding box's most northern latitude (degrees north)

38.1081390

Bounding box's most southern latitude (degrees north)

37.8003100

Bounding box's most western longitude (degrees east)

23.4817260

Bounding box's most eastern longitude (degrees east)

23.9514340

Commands used

```
$ cut -d ';' -f 5 oasa-history | uniq | sort -r | (head -1; tail -1)
```

```
$ cut -d ';' -f 6 oasa-history | uniq | sort -r | (head -1; tail -1)
```

The `cut -d, -f 5 oasa-history` command in the 1st one and the `cut -d, -f 6 oasa-history` command in the second. So we cut the 5th field from `oasa-history` in the first one and the 6th field from `oasa-history` in the second. That means that we cut bus position latitude, bus position longitude field correspondingly. `-d` option specifies a delimiter that will be used instead of the default "TAB" delimiter. So `-d,` means that the delimiter will be the comma. The output of `cut -d ';' -f 5 oasa-history` and the output of `cut -d ';' -f 6 oasa-history` is connected via a pipe to the standard input of `uniq` command. This command keeps only the unique bus position latitude, bus position longitude correspondingly. Or equivalent it removes duplicates. The output of `uniq` is connected via a pipe to the standard input of `sort -r` command. This command `sort -r` sort the input in descending order (`-r`: reverse the sorting order). The output of `sort -r` is connected via a pipe to the standard input of `(head -1; tail -1)` command. This command returns the max and min from 5th and 6th field correspondingly. (the max and min from bus position latitude, bus position longitude).

Which bus has appeared closest to your favorite location?

Obtain the coordinates of your favorite location (e.g. workplace, home, restaurant, bar, club, cinema, café, gym) using e.g. GPS or an online map. For a small area, such as Athens, you can use the simple Euclidean distance formula for calculating the distance between your workplace and the bus's location. (4 points)
If you cannot obtain an answer for this question based on distance, choose a bus at random, using e.g. the shuf command. (1 point)
The answer to this question will become the "chosen bus" specified in the next questions.

Favorite location

Name the location, describing its type and why you chose it.

The location I chose is my house. Is my favorit

Favorite location's latitude (degrees north)

38.046989

Favorite location's longitude (degrees east)

23.756307

Bus number *

70198

Bus location at the shortest distance from your favorite location latitude
(degrees north)

38.0474430

Bus location at the shortest distance from your favorite location longitude
(degrees east)

23.7565820

Command used

```
$ cut -d ',' -f 3,5,6 oasa-history | awk '{ FS=OFS=","}{x=sqrt(($2-38.046989)^2 + ($3-23.756307)^2)}{print $1,$2,$3,x}' | tr -s ',' $'\t' | sort -k 4n | head -1
```

The cut -d, -f 3,5,6 oasa-history command, cuts the 3rd, the 5th and 6th field from oasa-history.

That means that we cut the bus number, the bus position latitude and bus position longitude fields. -d option specifies a delimiter that will be used instead of the default "TAB" delimiter. So -d, means that the delimiter will be the comma. The output of cut -d ',' -f 3,5,6 oasa-history is connected via a pipe to the standard input of awk '{ FS=OFS=","}{x=sqrt((\$2-38.046989)^2 + (\$3-23.756307)^2)}{print \$1,\$2,\$3,x}' command. Awk FS variable is used to set the field separator for each record. Awk OFS is an output equivalent of awk FS variable. By default awk OFS is a single space character. BEGIN means that awk will execute the action specified in BEGIN once before any input lines are read. END means that awk will execute the action specified in END before it actually exits. This command calculate the Euclidean distance between the latitude and longitude corresponds to my house and the bus position latitude and bus position longitude from each line (each bus report). Then it prints the bus number, the bus position latitude, bus position longitude and the result of the Euclidean distance. The output of awk '{ FS=OFS=","}{x=sqrt((\$2-38.046989)^2 + (\$3-23.756307)^2)}

distance. The output of `awk '{ FS=OFS=","}{x=sqrt(($2-38.046989)^2 + ($3-23.756307)^2)}{print $1,$2,$3,x}'` is connected via a pipe to the standard input of `tr -s ',' '\t'` command. This command separates each of these 4 fields with a tab instead of comma that was previously. The output of `tr -s ',' '\t'` is connected via a pipe to the standard input of `sort -k 4n` command. `Sort -k 4n` sort the output based on the 4th field (the Euclidean distance) in ascending order. The output of `sort -k 4n` is connected via a pipe to the standard input of `head -1` command. `Head -1` command keeps only the first row from the input which is the result. The result contains the bus number, the bus position latitude, bus position longitude and the Euclidean distance(which is the minimum Euclidean distance from my house).

Πίσω

Επόμενο

Εκκαθάριση

Σελίδα 19 από 25

How many position reports have been sent by the chosen bus?

1 point

Answer

7588

Command used

```
$ awk -F, ' $3=="70198" oasa-history | uniq | wc -l
```

The symbol `-` is the interpreter for the AWK Programming Language. The AWK language is useful for manipulation of data files, text retrieval and processing.

`-F <value>` - tells awk what field separator to use. In your case, `-F,` means that the separator is `,` (comma).

It returns the reports-lines where in 3rd field(bus number) contains 70198. This means that it returns the lines where the bus number = 70198. The output of `awk -F, ' $3=="70198" oasa-history` is connected via a pipe to the standard input of `uniq` command. I used `uniq` command to be sure that there aren't duplicates. The output of `uniq` is connected via a pipe to the standard input of `wc -l` command. `Wc -l` command counts the lines-reports for the bus with number 70198.

Πίσω

Επόμενο

Εκκαθάριση

Σελίδα 20 από 25

What was the chosen bus's last position in the obtained data stream?

2 points

Latitude (degrees north)

38.0471130

Longitude (degrees east)

23.7595530

Command used

```
$ awk -F, '$3=="70198"' oasa-history | cut -d ';' -f 5,6 | tail -1
```

The symbol `-` is the interpreter for the AWK Programming Language. The AWK language is useful for manipulation of data files, text retrieval and processing.

`-F <value>` - tells awk what field separator to use. In your case, `-F,` means that the separator is `,` (comma). It returns the reports-lines where in 3rd field(bus number) contains 70198. This means that it returns the lines where the bus number = 70198. The output of `awk -F, '$3=="70198"' oasa-history` is connected via a pipe to the standard input of `cut -d ';' -f 5,6` command. That means that we cut the bus position latitude and bus position longitude fields where bus number is equal to 70198. `-d` option specifies a delimiter that will be used instead of the default "TAB" delimiter. So `-d ';'` means that the delimiter will be the comma. The output of `cut -d ';' -f 5,6` is connected via a pipe to the standard input of `tail -1` command. The `tail -1` returns the last row because we want to have the last position for the bus with number 70198.

Πίσω

Επόμενο

Σελίδα 21 από 25

Εκκαθάριση

On which date has the chosen bus given the most position reports?

3 points

Answer

2022-01-05

Command used

```
$ awk -F, '$3=="70198"' oasa-history | cut -d ',' -f 1,3 | cut -b -10,20- | sort | awk 'BEGIN { FS=OFS="," } {arr[$1]+=1 } END {for (i in arr) print i,arr[i]} ' | tr -s ' ' '\t' | sort -k 2n | tail -1
```

The symbol - is the interpreter for the AWK Programming Language. The AWK language is useful for manipulation of data files, text retrieval and processing.

-F <value> - tells awk what field separator to use. In your case, -F, means that the separator is , (comma). It returns the reports-lines where in 3rd field(bus number) contains 70198. This means that it returns the lines where the bus number = 70198. The output of awk -F, '\$3=="70198"' oasa-history is connected via a pipe to the standard input of cut -d ',' -f 1,3 command, cuts the 1st and 3rd field from oasa-history. That means that we cut the data acquisition time stamp field and the bus number field(which now contains only the bus number 70198). -d option specifies a delimiter that will be used instead of the default "TAB" delimiter. So -d, means that the delimiter will be the comma. The output of cut -d ',' -f 1,3 oasa-history is connected via a pipe to the standard input of cut -b -10,20- command. This command cuts the bytes 1 up to 10 and the bytes 20 up to the end. In other words, it keeps the date in YYYY-MM-DD format in the 1st field, the comma and the bus number field(which now has only the bus number 70198). The output of cut -b -10,20- is connected via a pipe to the standard input of sort command. Sort command sort the dates(it's optional. It doesn't make any difference in the output). The output of sort is connected via a pipe to the standard input of awk 'BEGIN { FS=OFS="," } {arr[\$1]+=1 } END {for (i in arr) print i,arr[i]} ' command. Awk FS variable is used to set the field separator for each record. Awk OFS is an output equivalent of awk FS variable. By default awk OFS is a single space character. BEGIN means that awk will execute the action specified in BEGIN once before any input lines are

command. Awk FS variable is used to set the field separator for each record. Awk OFS is an output equivalent of awk FS variable. By default awk OFS is a single space character. BEGIN means that awk will execute the action specified in BEGIN once before any input lines are read. END means that awk will execute the action specified in END before it actually exits. This command counts how many buses was on the road in each date.(it counts how many times each date appears and it's time add 1). In END it depicts the dates and the number of reports from bus with number 70198 in each date which is the output of this command. The output of `awk 'BEGIN { FS=OFS=","}{arr[$1]+=1 }END {for (i in arr) print i,arr[i]}'` is connected via a pipe to the standard input of `tr -s ',' '\t'` command. This command separates the dates and the number of buses that were on the road in each date with a tab instead of comma that was previously. The output of `tr -s ',' '\t'` is connected via a pipe to the standard input of `sort -k 2n` command. `Sort -k 2n` sort the output based on the 2nd field(the number of position reports of bus with number 70198 in each date) in ascending order. The output of `sort -k 2n` is connected via a pipe to the standard input of `tail -1` command. `Tail -1` command keeps only the last row from the input which is the result. (2022-01-05 197)

On how many routes has the chosen bus traveled?

2 points

Answer

19

Command used

```
awk -F, '$3=="70198"' oasa-history | cut -d ';' -f 2 | sort -u | wc -l
```

This means that it returns the lines where the bus number = 70198. The output of `awk -F, '$3=="70198"' oasa-history` is connected via a pipe to the standard input of `cut -d ';' -f 2` command, cuts the 2nd field from oasa-history. That means that we cut the route field. The output of `cut -d ';' -f 3,2 oasa-history` is connected via a pipe to the standard input of `sort -u` command. This command remove duplicates and sort the 1st field(the routes) in ascending order. The output of this command is connected via a pipe to the standard input of `wc -l` command. This command counts the lines. The different routes that the bus with number 70198 has traveled.

OR equivalent

```
awk -F, '$3=="70198"' oasa-history | cut -d ';' -f 3,2 | sort -u | awk 'BEGIN { FS=OFS="," } {arr[$1]+1 } END {for (i in arr) print i,arr[i]} ' | tr -s ',' '\t' | wc -l
```

Πίσω

Επόμενο

Εκκαθάριση

Σελίδα 23 από 25

How many buses have shared at least one route with the chosen bus?

4 points

Answer

98

Command used

1st command:

```
$ awk -F, ' $3=="70198" oasa-history | cut -d ',' -f 3,2 | sort -u | awk 'BEGIN { FS=OFS="," } {arr[$1]+=1 }END {for (i in arr) print i,arr[i]}' | tr -s ',' '\t' | cut -b -4 > routes
```

I execute this command in order to have the routes for the bus with number 70198. The command cut -b -4 cuts only the 4 first bytes which are the routes. The output is saved to a file named routes.

2nd command:

```
$ cut -d ',' -f 2,3 oasa-history | tr -s ',' '\t' | awk ' $2 !~ 70198' > routes_full
```

I execute this command in order to have the routes for all the other buses except the chosen bus(70198). That's why I use the command awk '\$2 !~ 70198' because I don't want the row containing the number 70198 in the 2nd field(2nd field because in the beginning I cutted the 2nd and 3rd field. So in the next commands this fields are in 1st and 2nd field correspondingly). The output is saved to a file named routes_full.

3rd command:

```
$ awk 'NR==FNR{val[$1]=$1; next} $1 in val {print $2}' routes routes_full | sort | uniq | wc -l
```

NR and FNR are awk built-in variables.

FNR refers to the record number in the current file

3rd command:

```
$ awk 'NR==FNR{val[$1]=$1; next} $1 in val {print $2}' routes routes_full | sort | uniq | wc -l
```

NR and FNR are awk built-in variables.

FNR refers to the record number in the current file,

NR refers to the total record number.

NR==FNR will be TRUE when first file (routes) is being read.

val[\$1]=\$1; Create an array named val with index of field 1 and have value as first field.

next is a built-in function in awk so as to skip all next statements.

\$1 in val Checks if first field of routes_full is present in array val, this will be checked only when routes_full is being read.

{print \$2} prints the second field. The buses from routes_full file where their routes that have been matched with the routes from routes files. Because there are many duplicates, I use the command sort and uniq because I want only the unique buses that have shared at least one route with the chosen bus(70198).

Πίσω

Επόμενο

Σελίδα 24 από 25

Εκκαθάριση
φόρμας



Mission Possible: Data Engineering with Unix Tools

Η απάντησή σας καταγράφηκε.

[Υποβολή νέας απάντησης](#)

```
sort -k3,3
```

```
a$1
```

```
a[1]
```

```
###a[$1]
```

```
awk -F"#" '{if($1==123) print $2}' test-1.txt
```

I also use awk to find the intersect (overlap) of two files. For example, to find the SNPs in file1.txt and file2.txt, you can first store Column 1 of the first file (in the variable arr), then test whether elements in Column 1 of the second file are in arr

```
$awk '{if(NR==FNR){arr[$1];next}}($1 in arr){print $1}' file1.txt file2.txt
```

Here, FNR is the row count for the file being read, if testing whether NR equals FNR is asking whether awk is reading the first file.