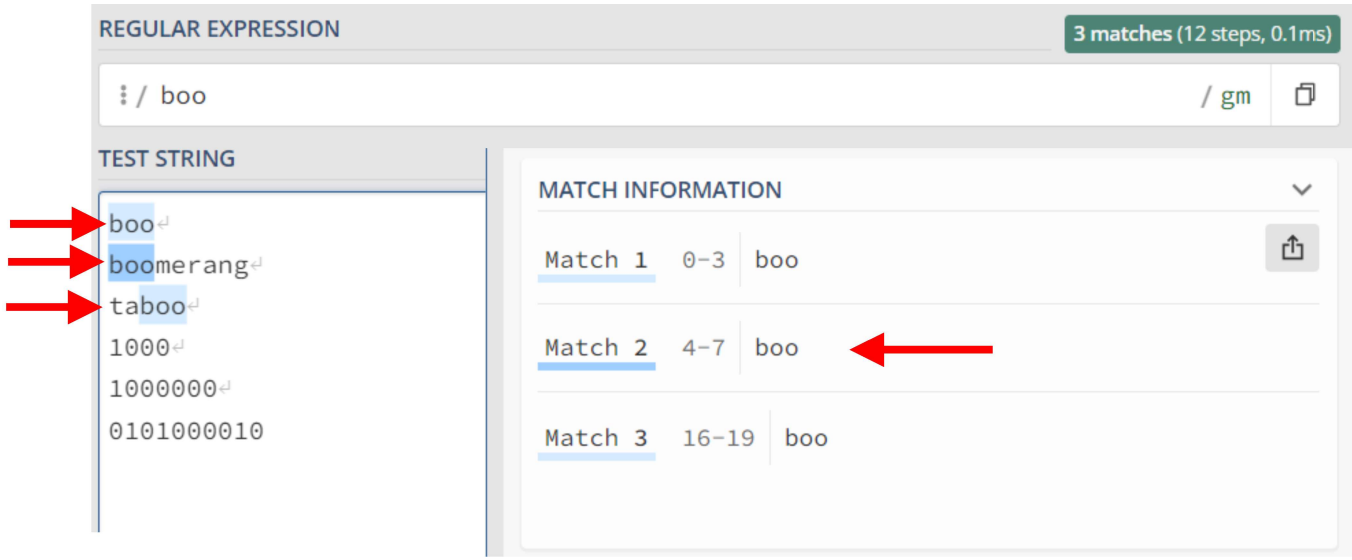
Regex Basics and Sanity

By the end hopefully we have both...

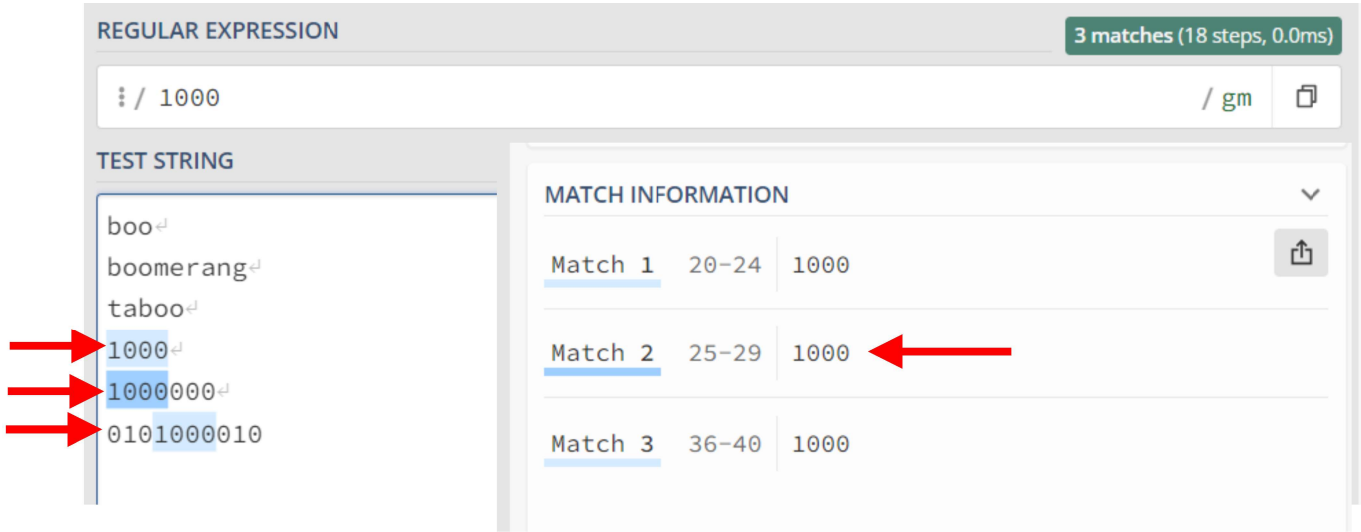
Ok, so lets break into this easy and test out using literals. This is exactly as it sounds and matches literally what we type (to a degree as we will see later)

Here we have matched “boo”



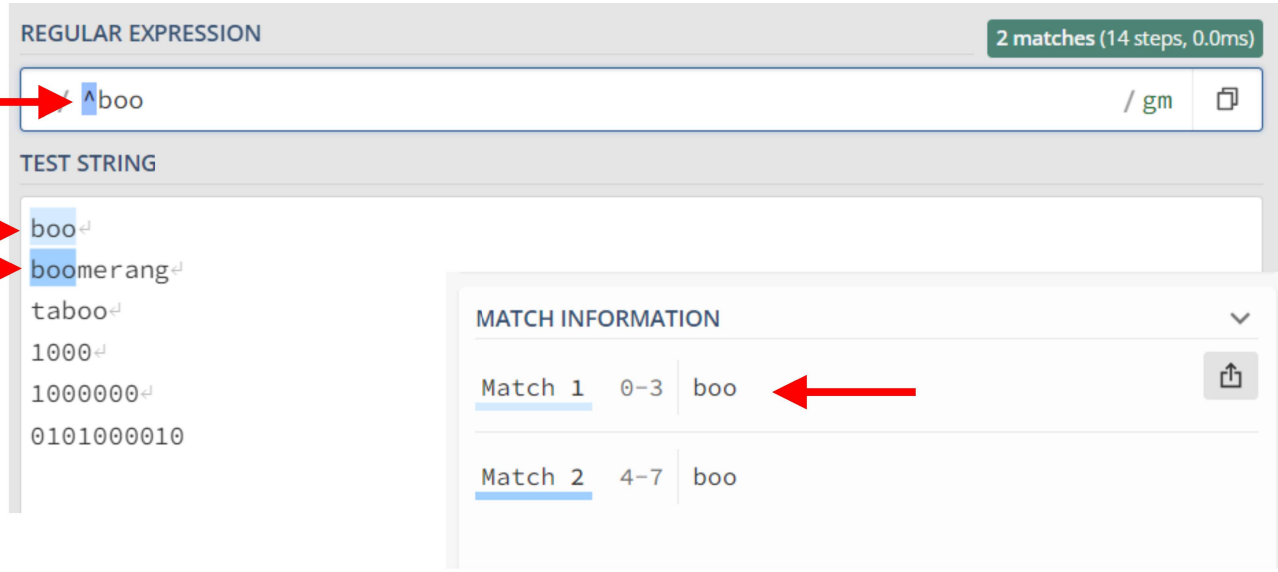
Now, lets do it again but with numbers. Notice, just like “boo”, it finds it anywhere and matches.

Here we have matched “1000”



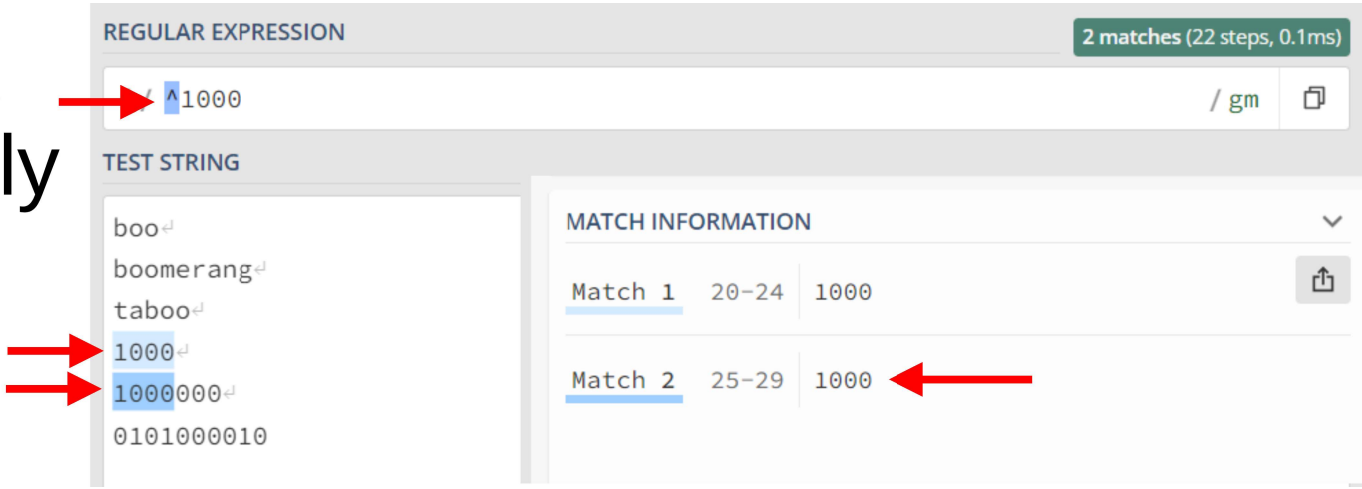
Lets now match only rows that start with “boo”.

We use “^” to say match whats next but it must be at the beginning.



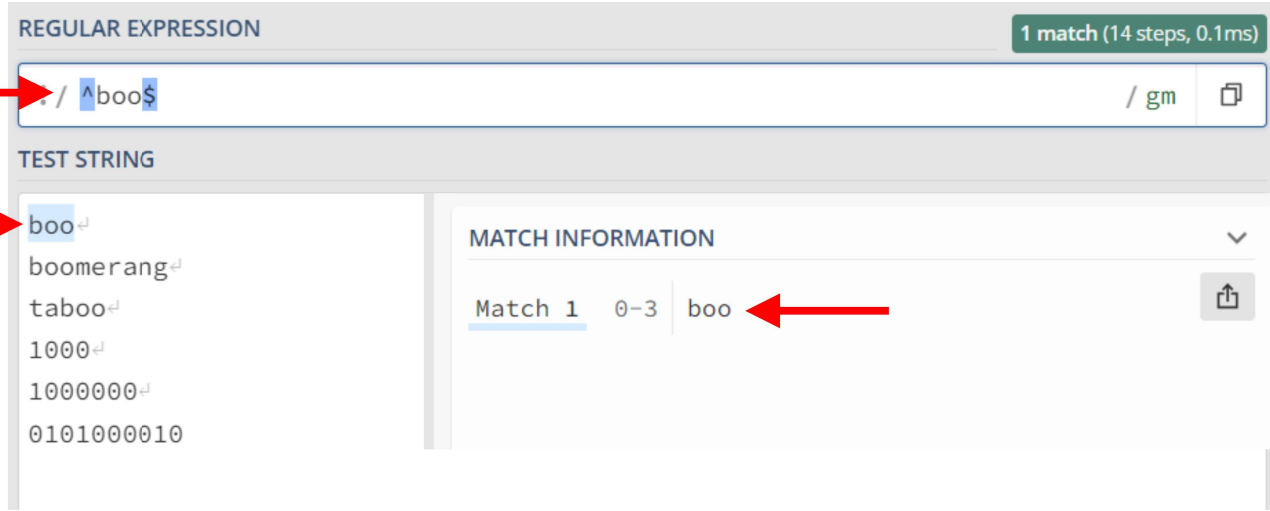
Lets do it again and switch to numbers this time.

Again, use “^” and see it finds 1000 only when its the beginning.



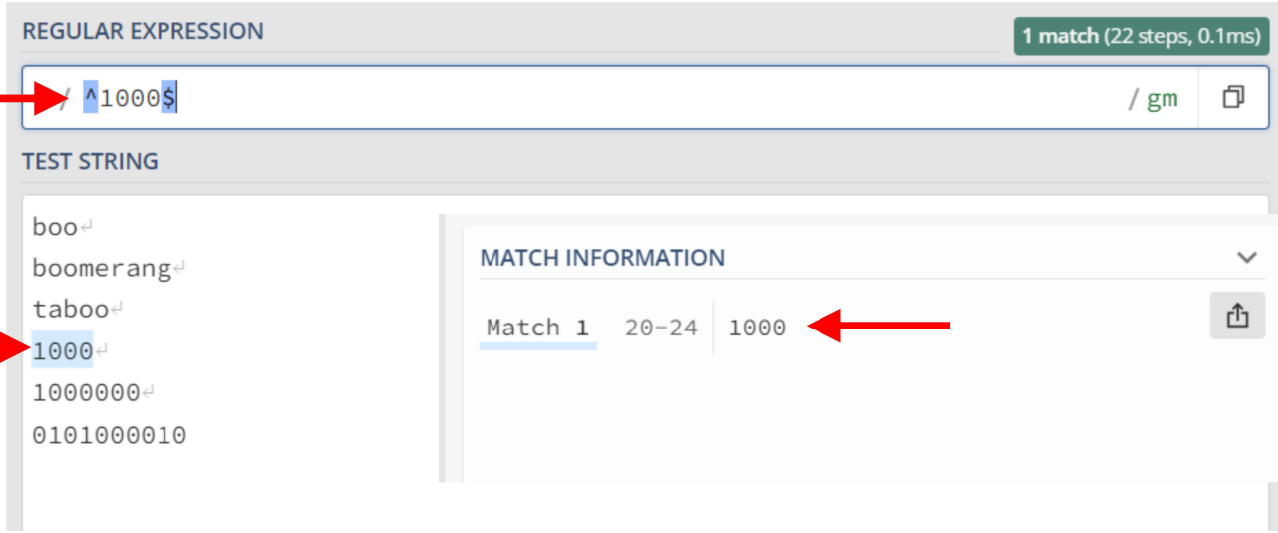
This time lets add “$” to the end and see what we get. Notice, we only have one match now. That is because $ says match whatever is before but only at the end.

So, ^boo$ says match me but only if I start and end the row with boo.



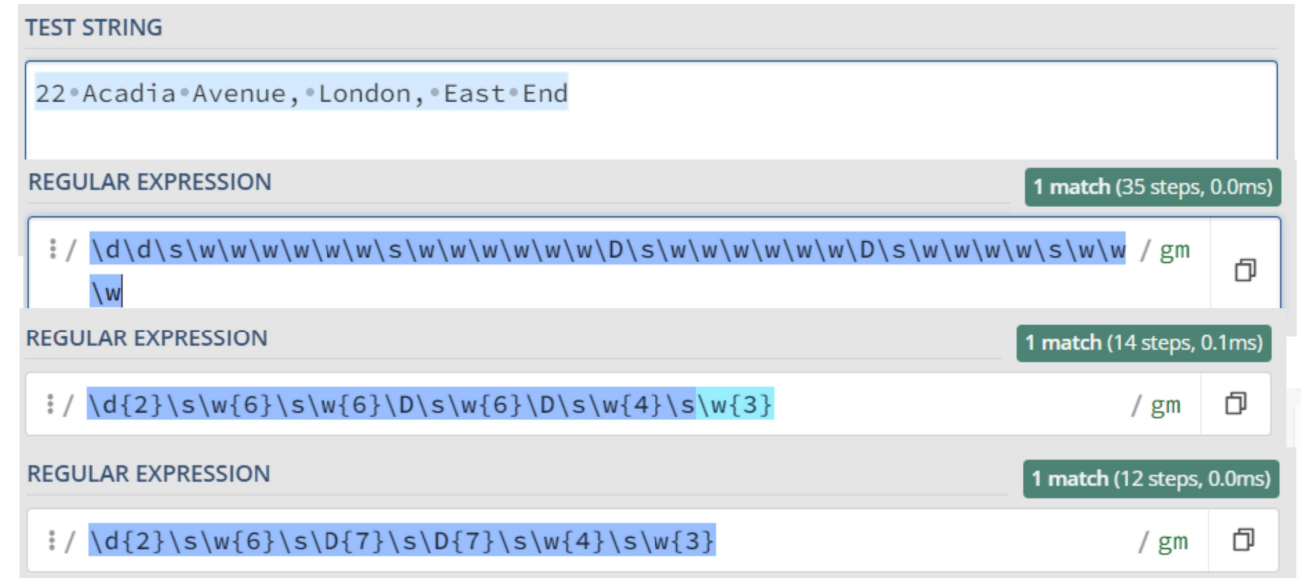
Lets check on numbers and see if the rules change.

Looks good and again 1 match because it must start and end with 1000



Ok, lets kick it up a notch…

Lets try part of an address but don't use literals. There are many ways to do this and here are a few examples. Notice how we continue to refine what took 35 steps to start ended at only taking 3.



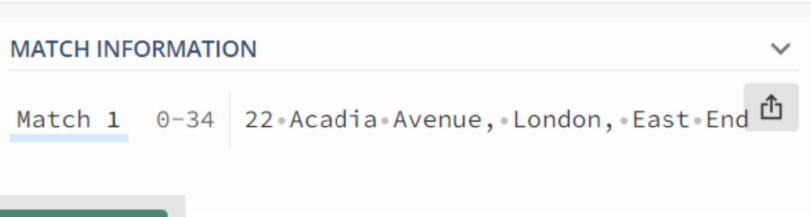
Helpful predefined classes

\d – digits

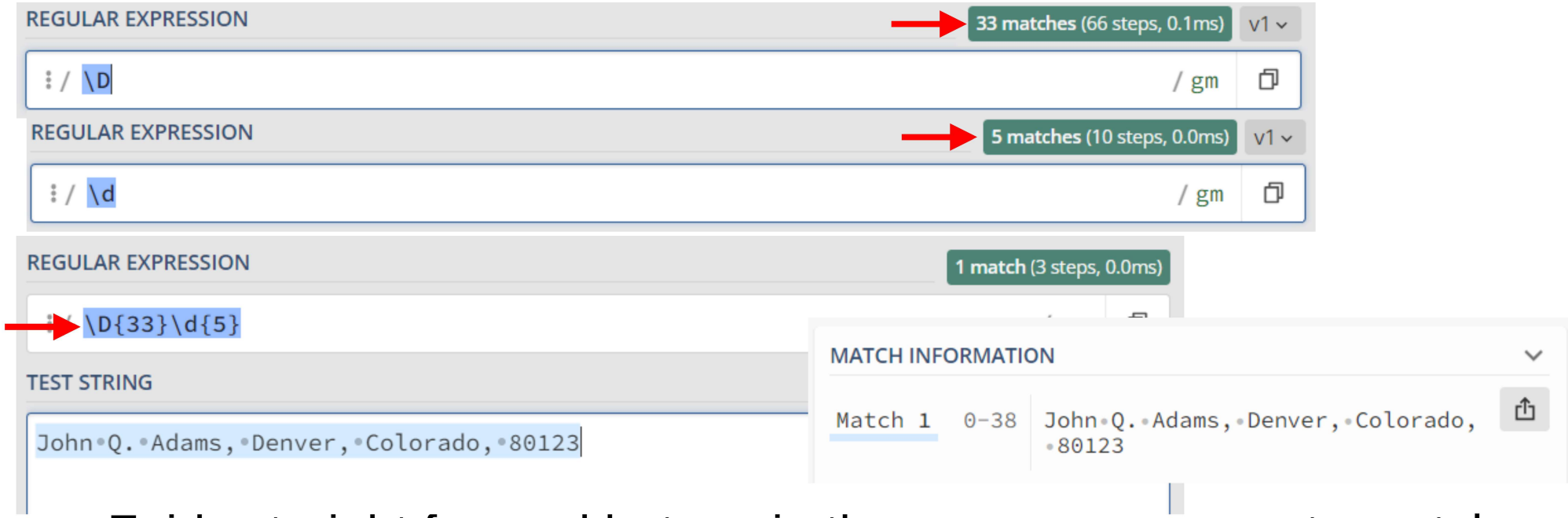
\s – space

\w – letter and digit

\D – anything not a digit

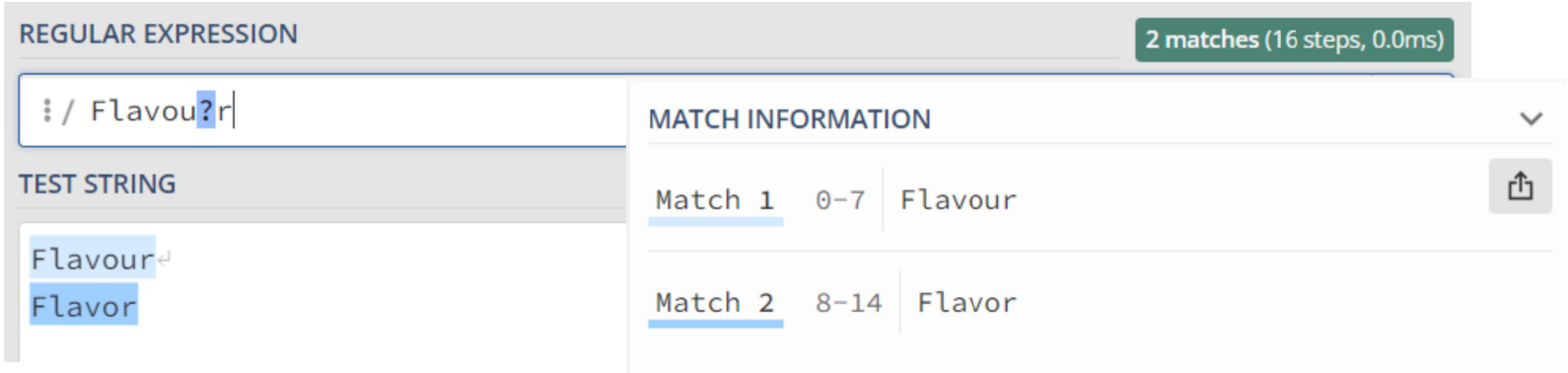


So, I liked how strong \D was and for the next string I'm going to jump right into using it.
First I'll input \D and note 33 matches
Then \d and 5 matches
This helps because I don't want to count characters so I now know exactly what numbers to use to select the entire string

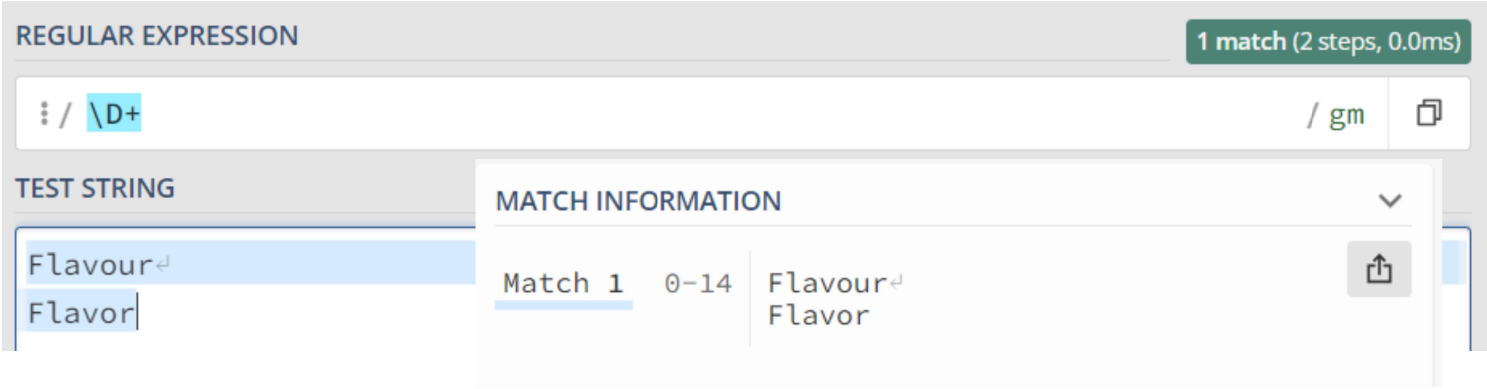


Fairly straight forward but again there are many ways to match.

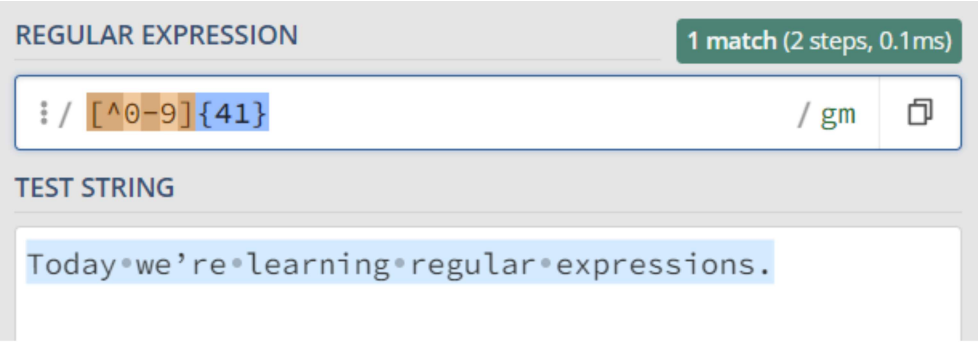
Lets gets weird and try spelling variation selection. The key here is “?” which says match the previous item 0-1 times. Basically, I’m ok if its there or not and keep matching.



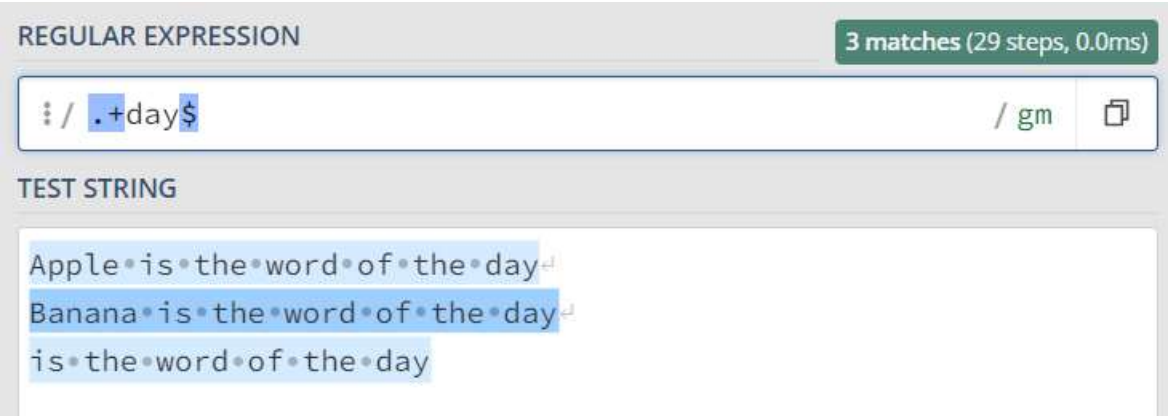
However, if we just want to match everything in one fell swoop I will introduce you to “+”. This says match the previous until you can’t. Powerful but remember with great power comes great responsibility...



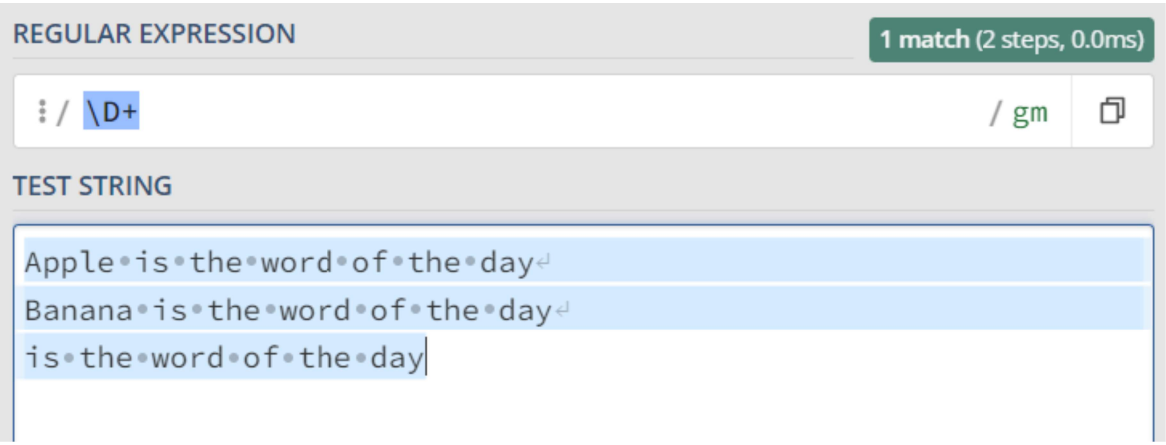
Ok, next up lets create a character class. We do that with “[ ]” and below I said match anything but numbers.

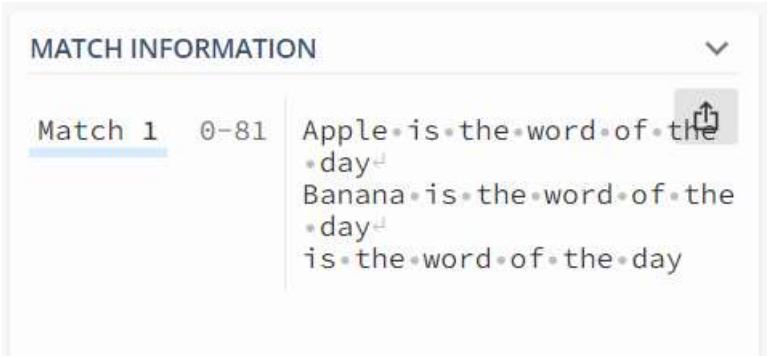


Lets start to select individual strings now. Here we see each string ends (hint) in day. So, lets say I want to match any string that ends ($) in day.



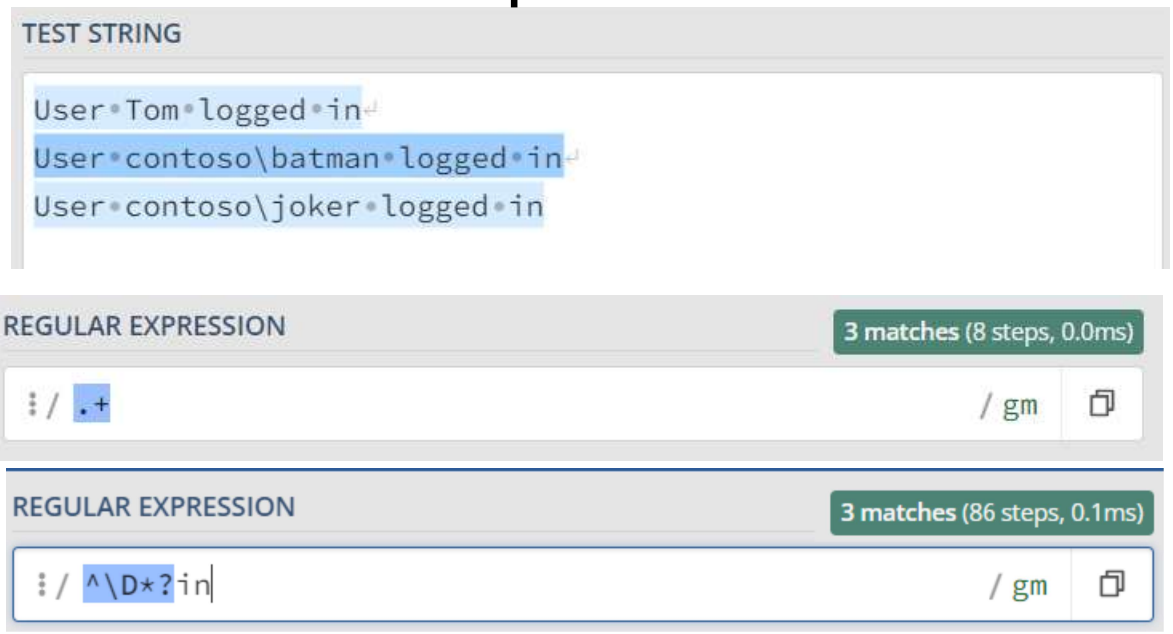
To select the entire string lets use \D+ and call it a day.

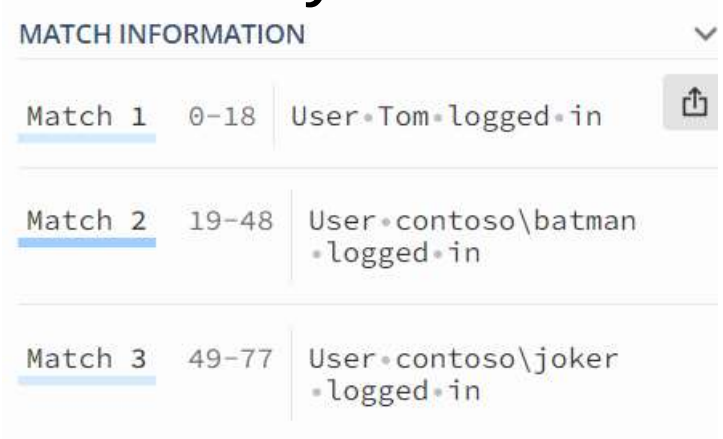




Now, lets start to match strings that you may find in a log. Sure we can use a classic .+ but that has limitation. Lets think a bit more…

Ok, so I went with match anything (no numbers) at start and stop at “in” because I used it literally.



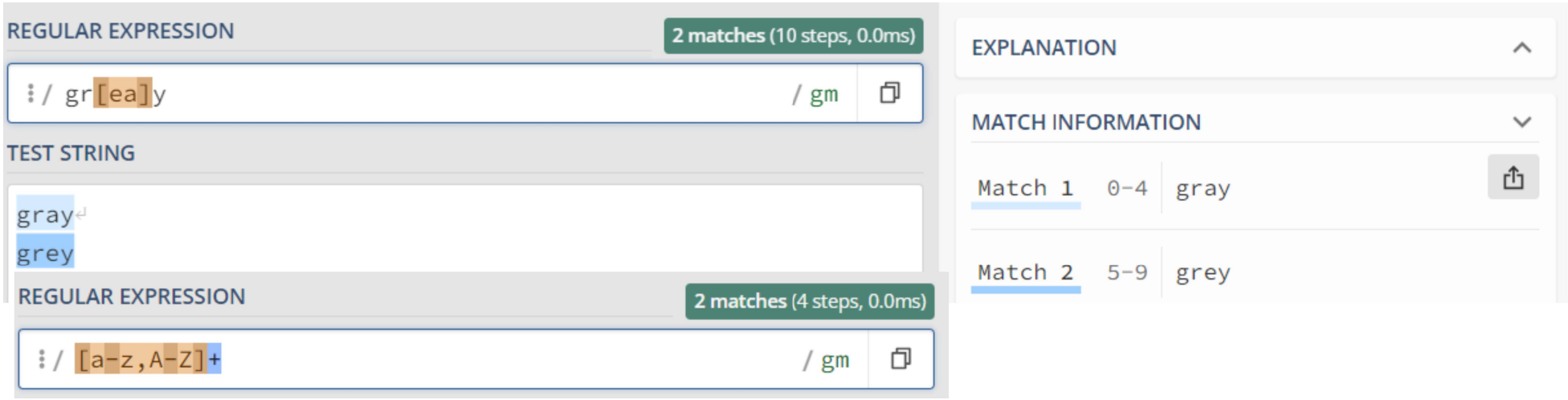


I wasn't happy so I had to try again. This time lets toss some groups into play. Sure, very similar to before but seeing it this way feels better. Lets keep learning and see if this develops.

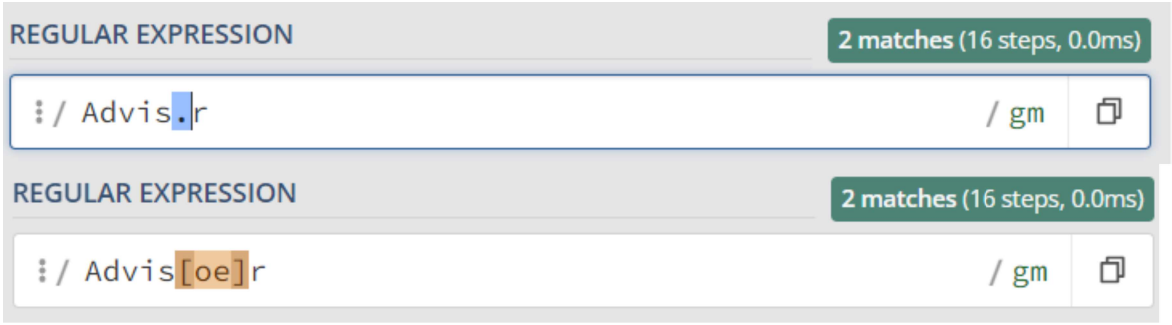
MATCH INFORMATION

Match 1	0-18	User Tom logged in
Group 1	0-4	User
Group 2	4-16	Tom logged
Group 3	16-18	in
Match 2	19-48	User contoso\batman logged in
Group 1	19-23	User
Group 2	23-46	contoso\batman logged
Group 3	46-48	in
Match 3	49-77	User contoso\joker logged in
Group 1	49-53	User
Group 2	53-75	contoso\joker logged
Group 3	75-77	in

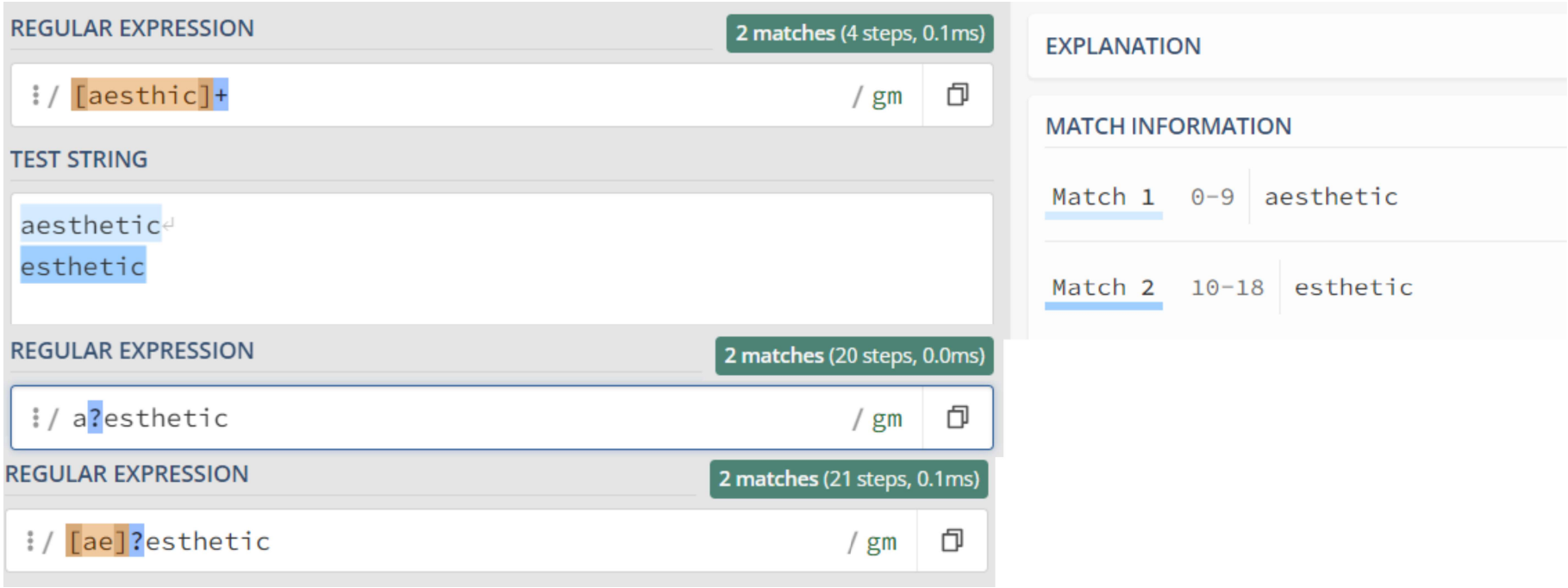
Ok, lets work on spelling variations again. This time lets build a character. As we learn, think about ways to apply in a work setting. Then go mess with it yourself.



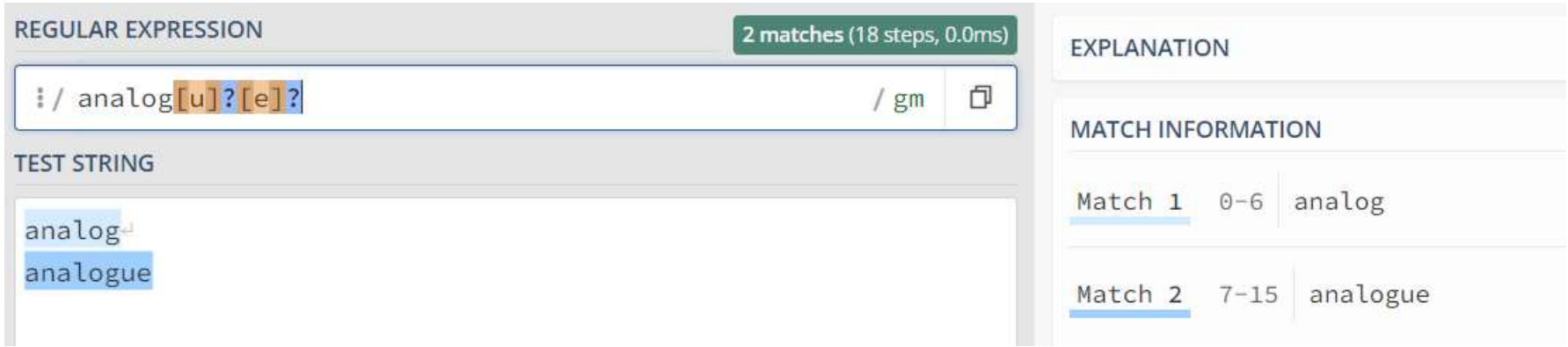
Again, here the class says upper and lower case a-z matches. Some optional ways below are using the wildcard “.” or insert class string in the word as well.



Depending, yes we can use “\D” or “.” followed by “+” and it works, but remember will this work in an actual log? It may. Here, I found using “?” worked well. Remember the “?” says optional and will come in handy.



Well that came sooner than expected. Remeber “?”? Now is a good time as we say match analog and then what follows optional.



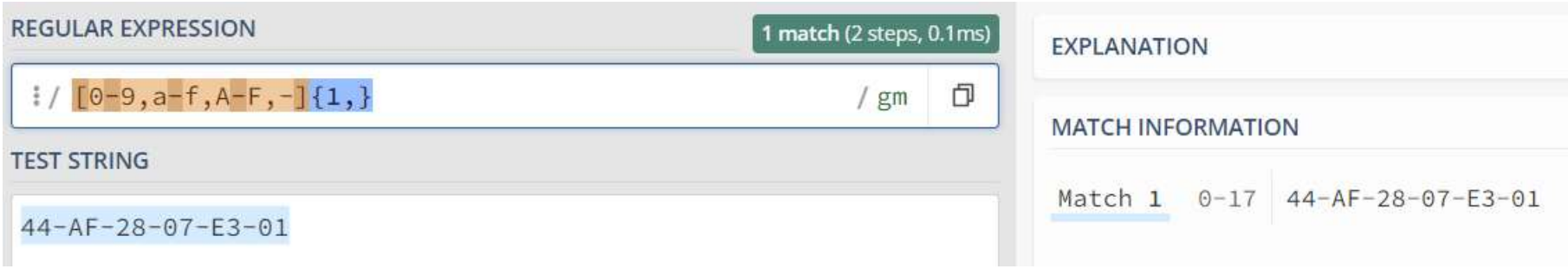
Ok, now this seems work related so lets build a class for hexadecimal. Fairly straight forward but lets keep going.

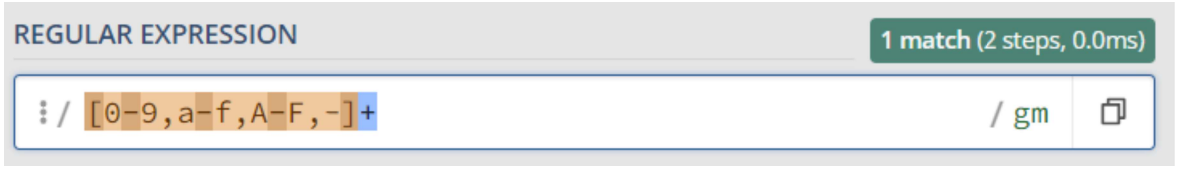
Lets grab our MAC and see if we can regex that. First hit that CLI and ipconfig /all and go ahead and copy/paste.

```
C:\Users\Chris>ipconfig /all

   Connection-specific DNS Suffix  . : hsd1.co.comcast.net
   Description . . . . . . . . . . . : Intel(R) Wi-Fi 6 AX201 160MHz
   Physical Address. . . . . . . . . : 44-AF-28-07-E3-01
```

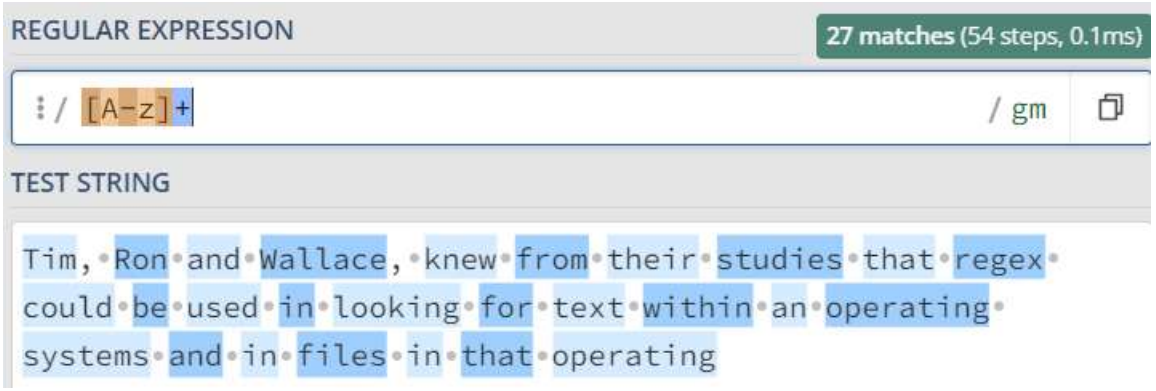
Then lets regex it. Again, hexadecimal and some options to grab as a string are the classic “+” or I initially went with “{1,}” which really is the same as “+” in this sense.





This time lets just grab the numbers. I’m sure you remember “\d” and its a zipcode so lets tell it 5x using “{5}” and there we go.

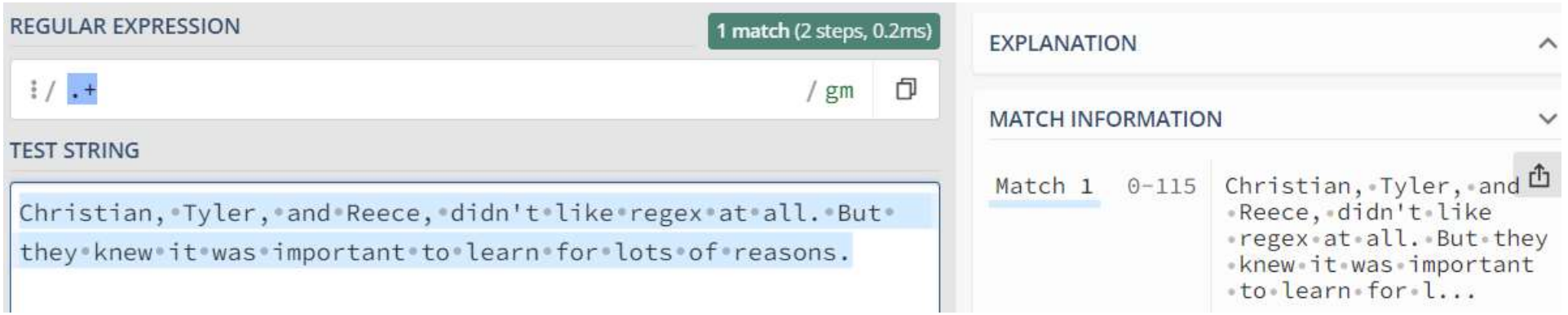
Individual words? No problem. Character class and away we go.



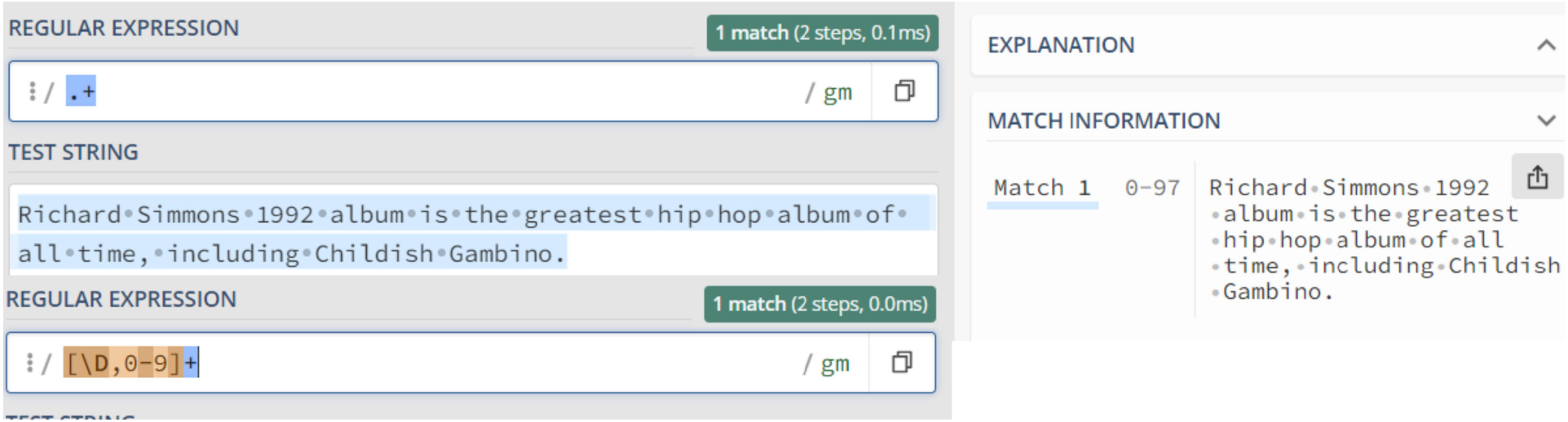
MATCH INFORMATION

Match 1	0-3	Tim
Match 2	5-8	Ron
Match 3	9-12	and
Match 4	13-20	Wallace
Match 5	22-26	knew
Match 6	27-31	from
Match 7	32-37	their
Match 8	38-45	studies
Match 9	46-50	that

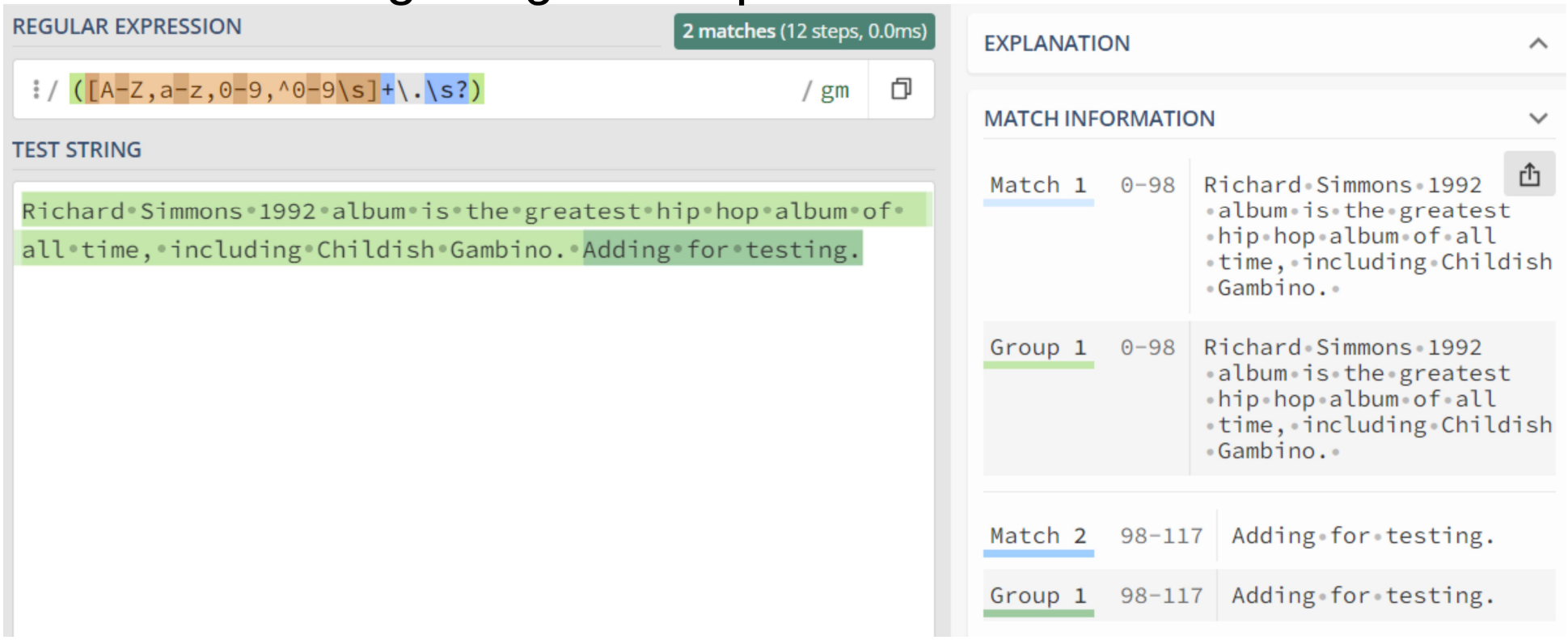
Can we grab the entire string in one stroke? Of course, but I took the easy way out. What did you come up with?



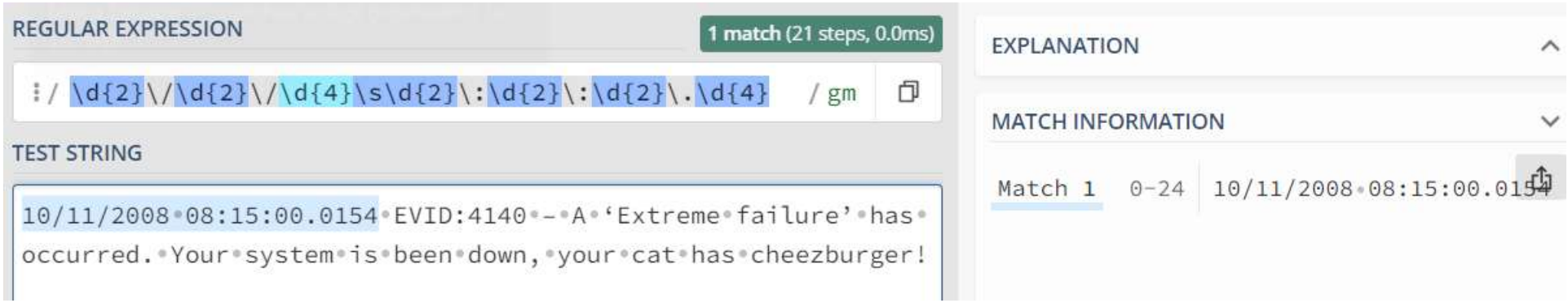
Again? Ok, this time I also included another way. I said lets catch everything but is this really different than ".+"?



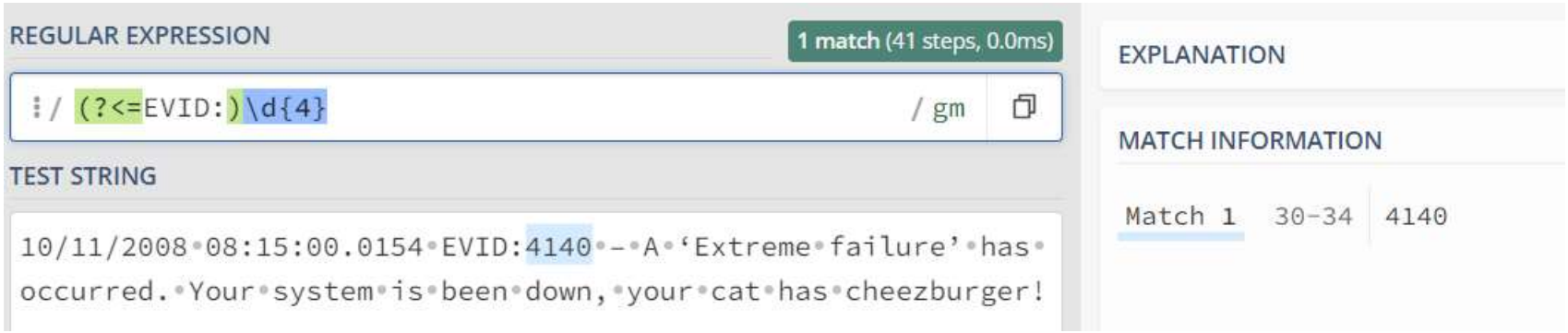
I was disappointed in me too. I wanted to try to catch sentences so added a short test sentence at the end. It looks like I did it...but did I? The literal "." is going to trip up my regex if ever a doctor (Dr.) enters the room. Also, many other reasons this would break, but fun getting to this point. Whats next??



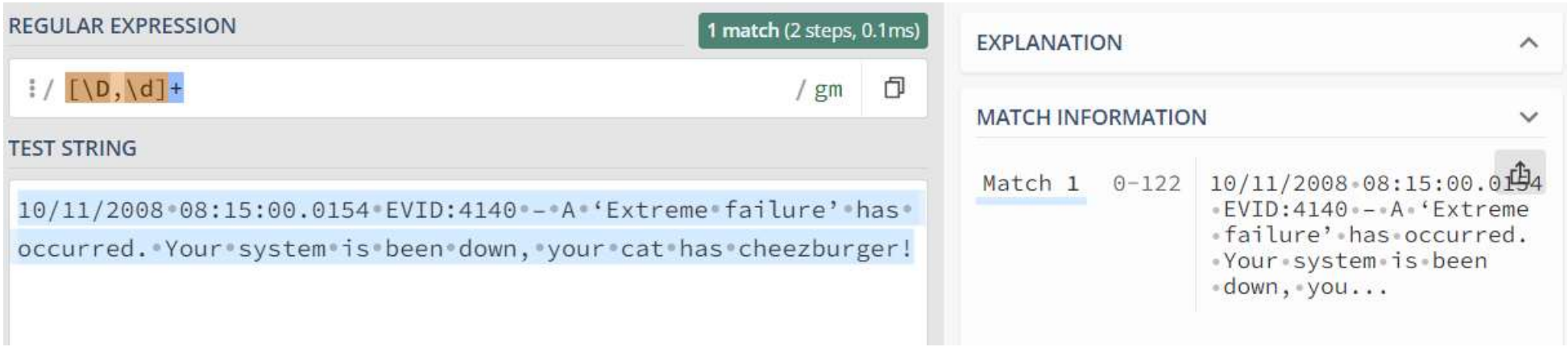
Looks like we have a possible log string now. Lets tackle the date/time. Getting easier? It may not be the most efficient but a quick combination of \d, \s, and literals and we have the dtg selected.



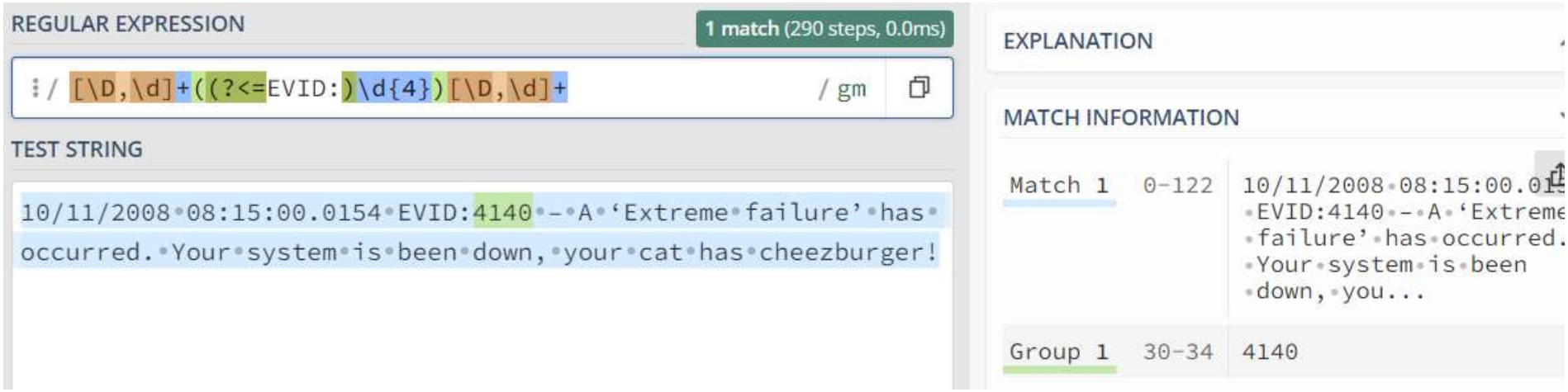
Can we pull out just the EVID number? Doing some reading and testing I found a way to look for EVID and then just provide me what follows. However, regex has multiple ways and play around with it and see what you get.



Entire string? Sure, lets snag a \D, \d and away we go.



Entire string and the EVID number? Lets group up EVID and we have it completed.



These are some of the basics of regex and how to apply them. There is so much more. Keep on learning.

Bonus: can you match and group the following log section

2024-02-10 00:01:56, User4, open file, success, Document1.docx
2024-02-23 05:40:33, User1, login, success,
2024-02-19 21:09:17, User2, open file, success, Document1.docx
2024-02-12 20:35:58, User3, logout, success,
2024-02-10 19:25:07, User2, login, success,
2024-02-14 08:40:18, User4, login, success,
2024-02-26 07:37:04, User4, delete file, success, Presentation.pptx

Continue for answer

REGULAR EXPRESSION

1 000 matches (42 174 steps, 29.2ms)

```
/ (?<date>\d{4}\-\d{2}\-\d{2})(?<time>\s\d{2}\:\d{2}\:\d{2})\,\s(?<UserID>User\d)\,\s(?<action>\w{1,}\s?\w{1,})\,\s(?<result>\w+)\,\s(?<target>.+)? / gm
```

TEST STRING

```
2024-02-27 13:42:25, User1, logout, success,
2024-02-20 06:20:01, User2, login, success,
2024-03-04 02:43:50, User1, open file, success, Document1.docx
2024-02-10 21:19:19, User1, logout, success,
2024-02-27 00:24:23, User2, open file, success, Report.pdf
2024-02-25 08:48:41, User1, login, success,
2024-02-10 00:22:49, User1, password failure, failed,
2024-02-09 00:53:03, User2, logout, success,
2024-03-01 12:03:40, User3, delete file, success, Presentation.pptx
2024-02-25 11:26:19, User4, delete file, success, Report.pdf
2024-02-11 05:32:23, User2, edit file, success, Spreadsheet.xlsx
2024-02-19 10:08:08, User2, password failure, failed,
2024-02-08 13:14:11, User1, password failure, failed,
2024-02-10 01:49:33, User4, password failure, failed,
2024-02-24 13:06:58, User2, logout, success,
```

EXPLANATION

MATCH INFORMATION

Match 1	0-65	2024-03-05 08:44:39, User3, edit file, success, Presentation.pptx
Group date	0-10	2024-03-05
Group time	10-19	08:44:39
Group UserID	21-26	User3
Group action	28-37	edit file
Group result	39-46	success
Group target	48-65	Presentation.pptx
Match 2	66-120	2024-02-18 06:40:31, User2, password failure, failed,