Create Hash Values using Linux

Project description

In this project I am tasked with creating hash values for two files and use linux commands to manually examine the differences.

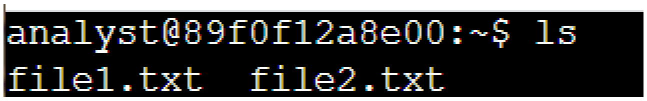
In this project, I need to display the contents of each file. I will then generate a hash value for each of these files and send the values to new files, which I will use to examine the differences in these values later.

I will use Linux to:

- List the contents of the home directory
- Compare the plain text of the two files presented for hashing
- Compute the *sha256sum* hash of the two separate files
- Compare the hashes provided to identify the differences

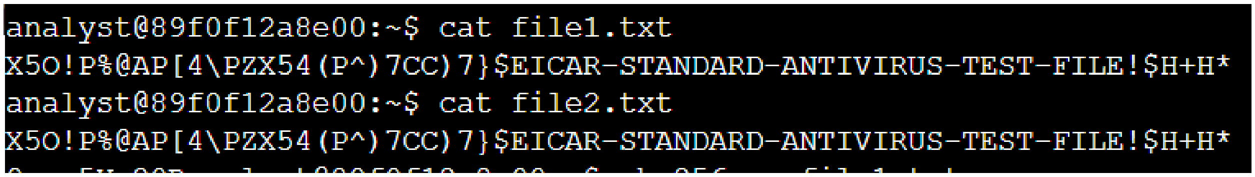
Generate hashes for files

To begin, I used the `ls` command to view the directory content,



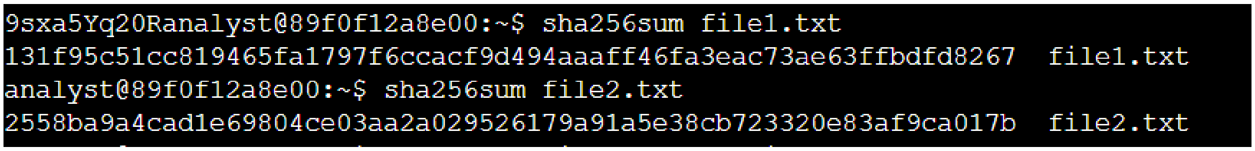
```
analyst@89f0f12a8e00:~$ ls
file1.txt  file2.txt
```

I then used the `cat` command to display each of the files contents



```
analyst@89f0f12a8e00:~$ cat file1.txt
X5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*
analyst@89f0f12a8e00:~$ cat file2.txt
X5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*
```

After this I used the `sha256sum` command to generate the hash of the files

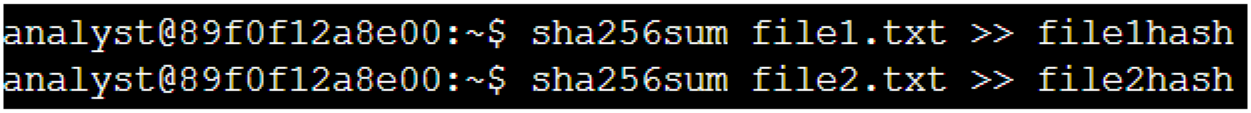


```
9sxa5Yq20Ranalyst@89f0f12a8e00:~$ sha256sum file1.txt
131f95c51cc819465fa1797f6ccacf9d494aaaff46fa3eac73ae63ffbdfd8267  file1.txt
analyst@89f0f12a8e00:~$ sha256sum file2.txt
2558ba9a4cad1e69804ce03aa2a029526179a91a5e38cb723320e83af9ca017b  file2.txt
```

I then reviewed the hash values and concluded they were different, however I wanted to compare and find where each difference is.

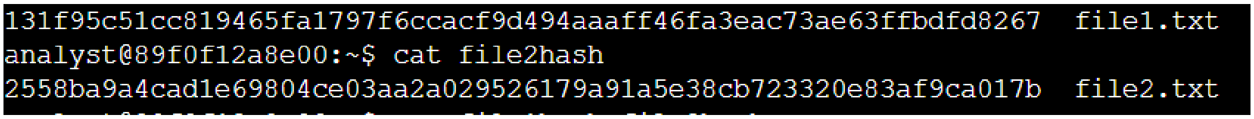
Compare hashes

I used the following command to generate the hash of the files and send the output to a new file. `sha256sum file1.txt >> file1hash`



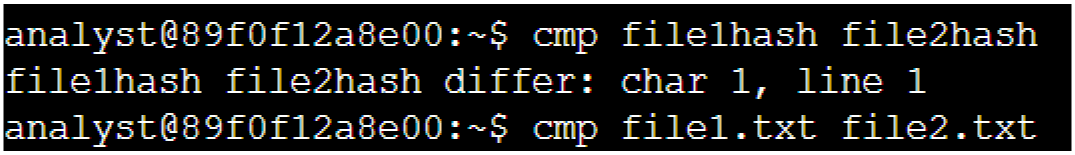
```
analyst@89f0f12a8e00:~$ sha256sum file1.txt >> file1hash
analyst@89f0f12a8e00:~$ sha256sum file2.txt >> file2hash
```

I then confirmed the contents of the new file with the `cat` command



```
131f95c51cc819465fa1797f6ccacf9d494aaaff46fa3eac73ae63ffbdfd8267  file1.txt
analyst@89f0f12a8e00:~$ cat file2hash
2558ba9a4cad1e69804ce03aa2a029526179a91a5e38cb723320e83af9ca017b  file2.txt
```

Lastly, I used the `cmp` command to highlight the differences between the two files I had sent the generated hashes too.



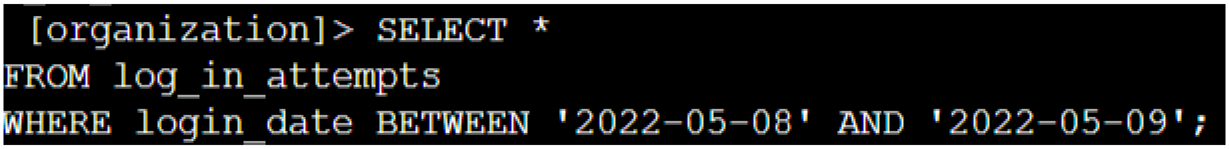
```
analyst@89f0f12a8e00:~$ cmp file1hash file2hash
file1hash file2hash differ: char 1, line 1
analyst@89f0f12a8e00:~$ cmp file1.txt file2.txt
```

I then reviewed the two files, which reports the first difference between the two files.

From the output of the cmp command, I can the the hashes differ at the first character in the first line

Retrieve login attempts on specific dates

My team is investigating a suspicious event that occurred on `'2022-05-09'`. I want to retrieve all login attempts that occurred on this day and the day before (`'2022-05-08'`).



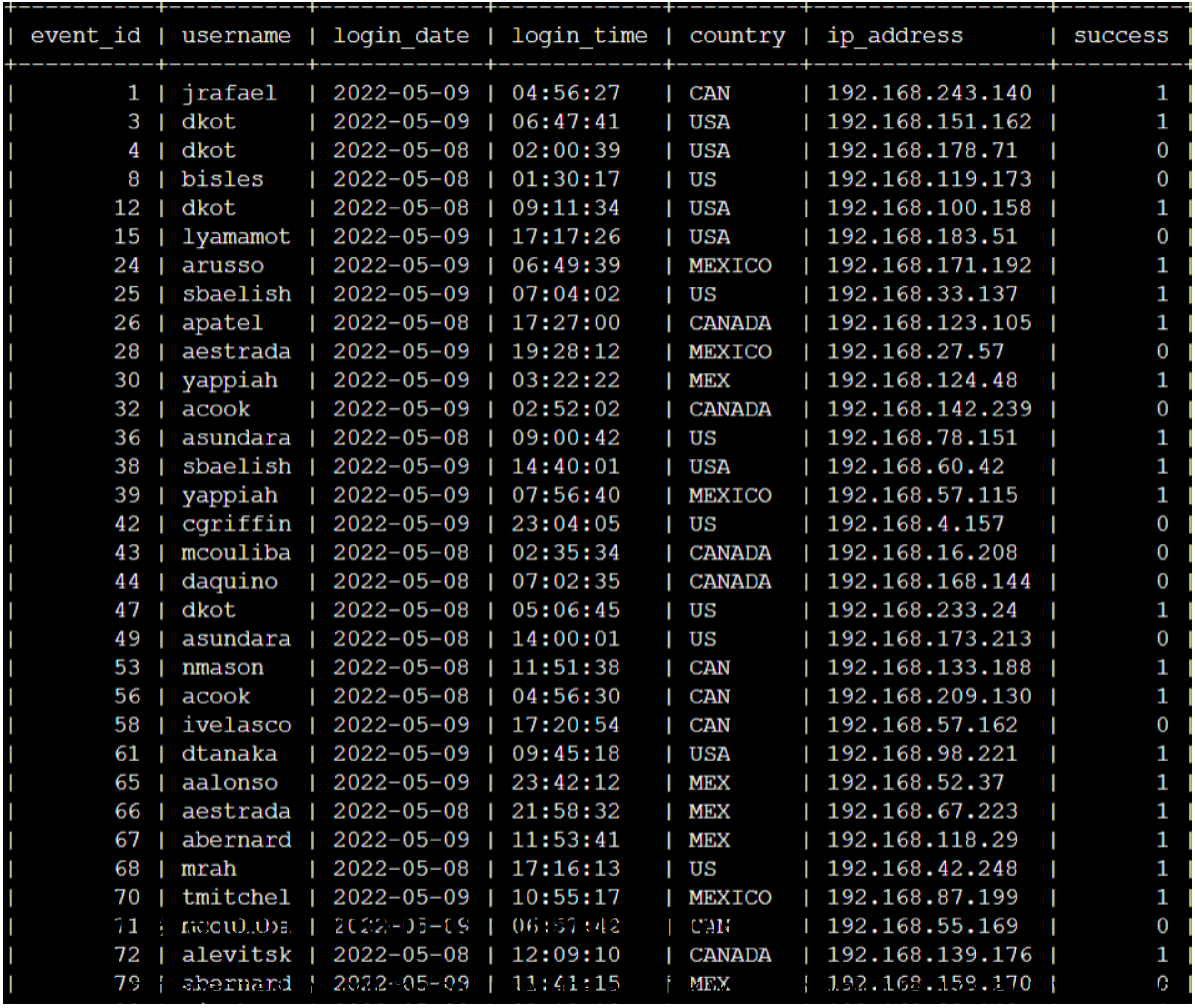
```
 [organization]> SELECT *
FROM log_in_attempts
WHERE login_date BETWEEN '2022-05-08' AND '2022-05-09';
```

I had chosen to return all database columns as I want to know all the events that occured during this time frame.

To specify this I used the BETWEEN operator and the AND operator to filter between two dates.

The WHERE operator always comes before as this indicates the column I want to apply the filter to

This has returned a large list, partial below.



```
+----------+----------+------------+------------+---------+-----------------+---------+
| event_id | username | login_date | login_time | country | ip_address      | success |
+----------+----------+------------+------------+---------+-----------------+---------+
|        1 | jrafael  | 2022-05-09 | 04:56:27   | CAN     | 192.168.243.140 |       1 |
|        3 | dkot     | 2022-05-09 | 06:47:41   | USA     | 192.168.151.162 |       1 |
|        4 | dkot     | 2022-05-08 | 02:00:39   | USA     | 192.168.178.71  |       0 |
|        8 | bisles   | 2022-05-08 | 01:30:17   | US      | 192.168.119.173 |       0 |
|       12 | dkot     | 2022-05-08 | 09:11:34   | USA     | 192.168.100.158 |       1 |
|       15 | lyamamot | 2022-05-09 | 17:17:26   | USA     | 192.168.183.51  |       0 |
|       24 | arusso   | 2022-05-09 | 06:49:39   | MEXICO  | 192.168.171.192 |       1 |
|       25 | sbaelish | 2022-05-09 | 07:04:02   | US      | 192.168.33.137  |       1 |
|       26 | apatel   | 2022-05-08 | 17:27:00   | CANADA  | 192.168.123.105 |       1 |
|       28 | aestrada | 2022-05-09 | 19:28:12   | MEXICO  | 192.168.27.57   |       0 |
|       30 | yappiah  | 2022-05-09 | 03:22:22   | MEX     | 192.168.124.48  |       1 |
|       32 | acook    | 2022-05-09 | 02:52:02   | CANADA  | 192.168.142.239 |       0 |
|       36 | asundara | 2022-05-08 | 09:00:42   | US      | 192.168.78.151  |       1 |
|       38 | sbaelish | 2022-05-09 | 14:40:01   | USA     | 192.168.60.42   |       1 |
|       39 | yappiah  | 2022-05-09 | 07:56:40   | MEXICO  | 192.168.57.115  |       1 |
|       42 | cgriffin | 2022-05-09 | 23:04:05   | US      | 192.168.4.157   |       0 |
|       43 | mcouliba | 2022-05-08 | 02:35:34   | CANADA  | 192.168.16.208  |       0 |
|       44 | daquino  | 2022-05-08 | 07:02:35   | CANADA  | 192.168.168.144 |       0 |
|       47 | dkot     | 2022-05-08 | 05:06:45   | US      | 192.168.233.24  |       1 |
|       49 | asundara | 2022-05-08 | 14:00:01   | US      | 192.168.173.213 |       0 |
|       53 | nmason   | 2022-05-08 | 11:51:38   | CAN     | 192.168.133.188 |       1 |
|       56 | acook    | 2022-05-08 | 04:56:30   | CAN     | 192.168.209.130 |       1 |
|       58 | ivelasco | 2022-05-09 | 17:20:54   | CAN     | 192.168.57.162  |       0 |
|       61 | dtanaka  | 2022-05-09 | 09:45:18   | USA     | 192.168.98.221  |       1 |
|       65 | aalonso  | 2022-05-09 | 23:42:12   | MEX     | 192.168.52.37   |       1 |
|       66 | aestrada | 2022-05-08 | 21:58:32   | MEX     | 192.168.67.223  |       1 |
|       67 | abernard | 2022-05-09 | 11:53:41   | MEX     | 192.168.118.29  |       1 |
|       68 | mrah     | 2022-05-08 | 17:16:13   | US      | 192.168.42.248  |       1 |
|       70 | tmitchel | 2022-05-09 | 10:55:17   | MEXICO  | 192.168.87.199  |       1 |
|       71 | mcouliba | 2022-05-09 | 06:57:42   | CAN     | 192.168.55.169  |       0 |
|       72 | alevitsk | 2022-05-08 | 12:09:10   | CANADA  | 192.168.139.176 |       1 |
|       79 | abernard | 2022-05-09 | 11:41:15   | MEX     | 192.168.158.170 |       0 |
```

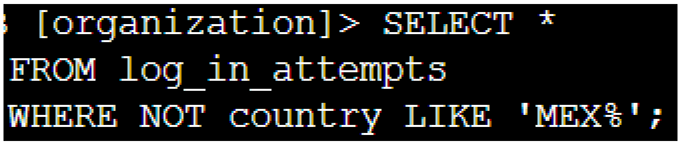
specified the database to be used.

Lastly, to filter for all login attempts apart from Mexico I had used the NOT and LIKE operator with the percentage sign wildcard.

The NOT operator will filter out the specified condition after it,

The LIKE operator will search for a matching pattern of the specified string.

The issue was there are countries listed as ‘MEX’ and ‘MEXICO’ so to make the query as efficient as possible I used the percent wildcard which tells the database to match the pattern of anything beginning with ‘MEX’.



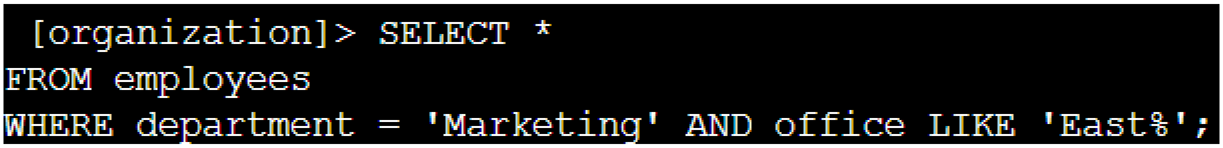
```
 [organization]> SELECT *
FROM log_in_attempts
WHERE NOT country LIKE 'MEX%';
```

Retrieve employees in Marketing

In the following query I had to search for all employee information who work in the east building marketing department.

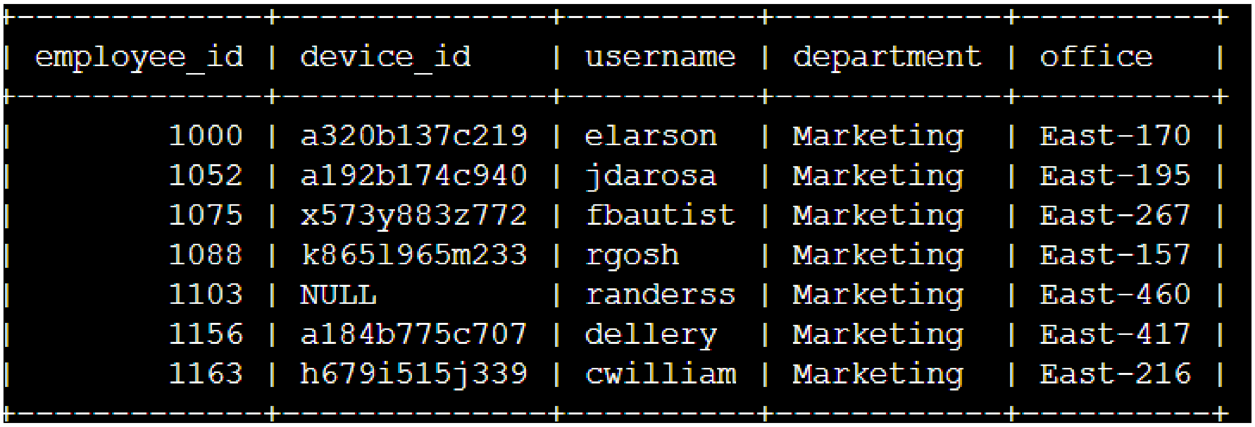
I again specified to return all columns in the employees table.

I then used the WHERE operator to specify the exact columns I want to filter against, specific the marketing department and used the LIKE operator with the percent wildcard to return all the devices in the east office building.



```
 [organization]> SELECT *
FROM employees
WHERE department = 'Marketing' AND office LIKE 'East%';
```

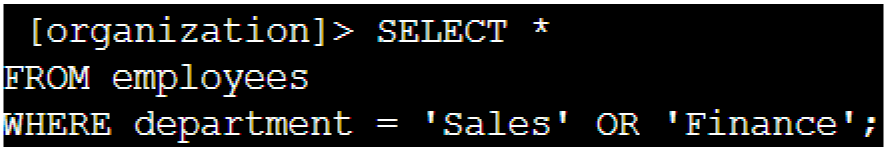
The results look like this:



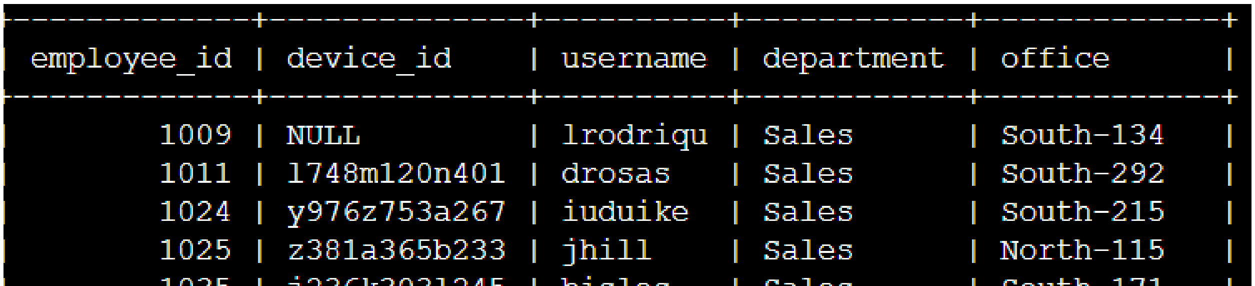
```
+-------------+-------------+----------+------------+----------+
| employee_id | device_id   | username | department | office   |
+-------------+-------------+----------+------------+----------+
|        1000 | a320b137c219 | elarson | Marketing  | East-170 |
|        1052 | a192b174c940 | jdarosa | Marketing  | East-195 |
|        1075 | x573y883z772 | fbautist | Marketing | East-267 |
|        1088 | k8651965m233 | rgosh   | Marketing  | East-157 |
|        1103 | NULL        | randerss | Marketing  | East-460 |
|        1156 | a184b775c707 | dellery | Marketing  | East-417 |
|        1163 | h679i515j339 | cwilliam | Marketing | East-216 |
+-------------+-------------+----------+------------+----------+
```

Retrieve employees in Finance or Sales

In this query I had used the OR operator to filter for two specific departments.



```
 [organization]> SELECT *
FROM employees
WHERE department = 'Sales' OR 'Finance';
```



```
+-------------+-------------+----------+------------+---------------+
| employee_id | device_id   | username | department | office        |
+-------------+-------------+----------+------------+---------------+
|        1009 | NULL        | lrodriqu | Sales      | South-134     |
|        1011 | 1748m120n401 | drosas  | Sales      | South-292     |
|        1024 | y976z753a267 | iuduike | Sales      | South-215     |
|        1025 | z381a365b233 | jhill   | Sales      | North-115     |
```

Summary

In Summary, I was able to retrieve Logs using SQL to filter for information I needed from the database by applying AND, OR and NOT operators to SQL queries as well as utilizing wildcards to return accurate information.