DIGITAL FORENSICS LAB

Exercise 9	
Name	S Shyam Sundaram
Registration Number	19BCE1560
Slot	L39+L40
Faculty	Dr. Seshu Babu Pulagara
Date	12th October, 2021

AIM

Comparing file types and signatures with Hex editors to identify what type of file they are originally.

PROCEDURE AND OBSERVATIONS

Q

Download at least two files with each of the following extensions from the Internet and keep them in a folder: jpg, png, bmp, gif, pdf

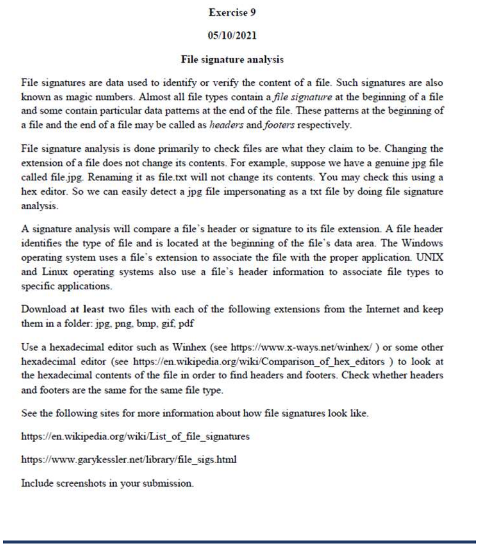
Use a hexadecimal editor such as Winhex (see https://www.x-ways.net/winhex/) or some other hexadecimal editor (see https://en.wikipedia.org/wiki/Comparison_of_hex_editors) to look at the hexadecimal contents of the file in order to find headers and footers. Check whether headers and footers are the same for the same file type.

A

We use 4 files: one.jfif, two.png, three.pdf and four.gif. They are shown below:

One.jfif

two.png



Exercise 9

05/10/2021

File signature analysis

File signatures are data used to identify or verify the content of a file. Such signatures are also known as magic numbers. Almost all file types contain a *file signature* at the beginning of a file and some contain particular data patterns at the end of the file. These patterns at the beginning of a file and the end of a file may be called as *headers* and *footers* respectively.

File signature analysis is done primarily to check files are what they claim to be. Changing the extension of a file does not change its contents. For example, suppose we have a genuine jpg file called file.jpg. Renaming it as file.txt will not change its contents. You may check this using a hex editor. So we can easily detect a jpg file impersonating as a txt file by doing file signature analysis.

A signature analysis will compare a file's header or signature to its file extension. A file header identifies the type of file and is located at the beginning of the file's data area. The Windows operating system uses a file's extension to associate the file with the proper application. UNIX and Linux operating systems also use a file's header information to associate file types to specific applications.

Download **at least** two files with each of the following extensions from the Internet and keep them in a folder: jpg, png, bmp, gif, pdf

Use a hexadecimal editor such as Winhex (see https://www.x-ways.net/winhex/) or some other hexadecimal editor (see https://en.wikipedia.org/wiki/Comparison_of_hex_editors) to look at the hexadecimal contents of the file in order to find headers and footers. Check whether headers and footers are the same for the same file type.

See the following sites for more information about how file signatures look like.

https://en.wikipedia.org/wiki/List_of_file_signatures

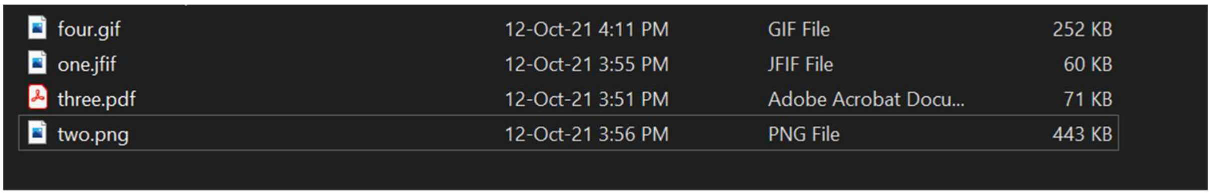
https://www.garykessler.net/library/file_sigs.html

Include screenshots in your submission.

three.pdf

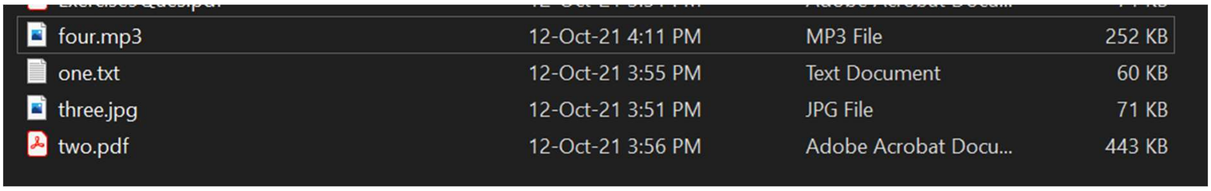
four.gif

We now change their extensions to: one.txt, two.pdf, three.jpg and four.mp3.



```
four.gif     12-Oct-21 4:11 PM   GIF File                  252 KB
one.jfif     12-Oct-21 3:55 PM   JFIF File                  60 KB
three.pdf    12-Oct-21 3:51 PM   Adobe Acrobat Docu...      71 KB
two.png      12-Oct-21 3:56 PM   PNG File                  443 KB
```

Before

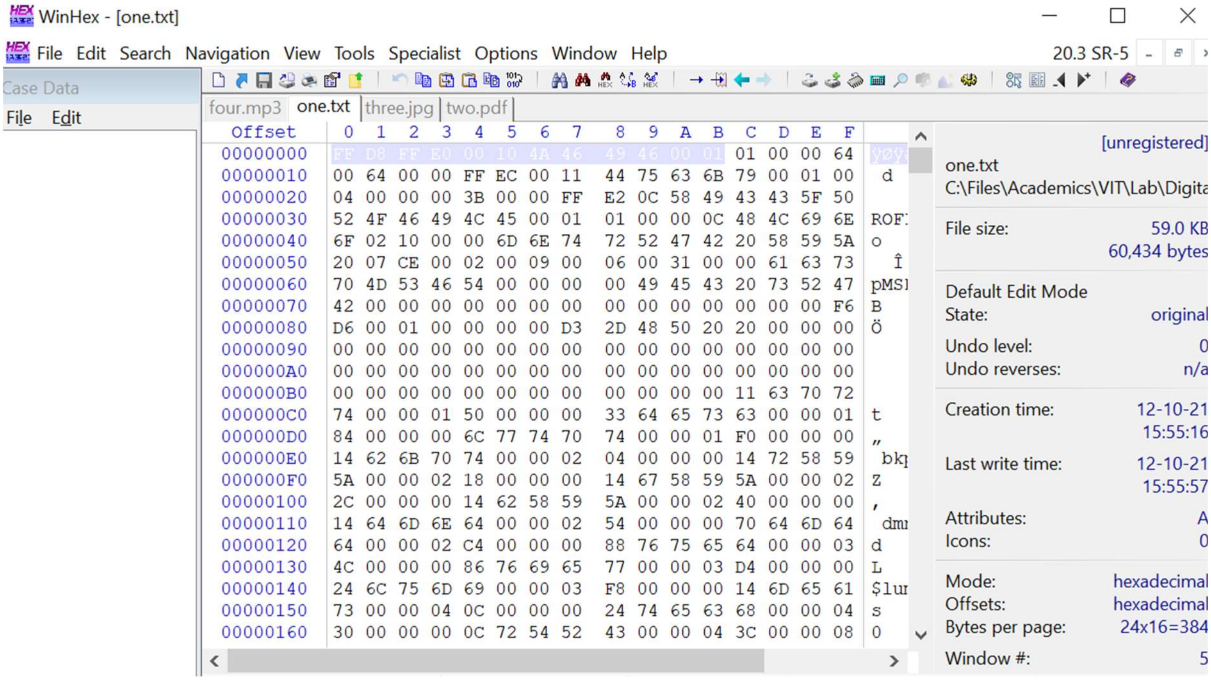


```
four.mp3     12-Oct-21 4:11 PM   MP3 File                  252 KB
one.txt      12-Oct-21 3:55 PM   Text Document              60 KB
three.jpg    12-Oct-21 3:51 PM   JPG File                   71 KB
two.pdf      12-Oct-21 3:56 PM   Adobe Acrobat Docu...     443 KB
```

After

We now open these files in WinHex and see their contents.

One.jfif/.txt



```
  Offset    0  1  2  3  4  5  6  7   8  9  A  B  C  D  E  F
00000000   FF D8 FF E0 00 10 4A 46  49 46 00 01 01 00 00 64  ÿØÿà
00000010   00 64 00 00 FF EC 00 11  44 75 63 6B 79 00 01 00   d
00000020   04 00 00 00 3B 00 00 FF  E2 0C 58 49 43 43 5F 50
00000030   52 4F 46 49 4C 45 00 01  01 00 00 0C 48 4C 69 6E  ROFI
00000040   6F 02 10 00 00 6D 6E 74  72 52 47 42 20 58 59 5A  o
00000050   20 07 CE 00 02 00 09 00  06 00 31 00 00 61 63 73    Î
00000060   70 4D 53 46 54 00 00 00  00 49 45 43 20 73 52 47  pMSI
00000070   42 00 00 00 00 00 00 00  00 00 00 00 00 00 00 F6  B
00000080   D6 00 01 00 00 00 00 D3  2D 48 50 20 20 00 00 00  Ö
00000090   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
000000A0   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
000000B0   00 00 00 00 00 00 00 00  00 00 00 00 11 63 70 72
000000C0   74 00 00 01 50 00 00 00  33 64 65 73 63 00 00 01  t
000000D0   84 00 00 00 6C 77 74 70  74 00 00 01 F0 00 00 00  „
000000E0   14 62 6B 70 74 00 00 02  04 00 00 00 14 72 58 59   bkp
000000F0   5A 00 00 02 18 00 00 00  14 67 58 59 5A 00 00 02  Z
00000100   2C 00 00 00 14 62 58 59  5A 00 00 02 40 00 00 00  ,
00000110   14 64 6D 6E 64 00 00 02  54 00 00 00 70 64 6D 64   dmn
00000120   64 00 00 02 C4 00 00 00  88 76 75 65 64 00 00 03  d
00000130   4C 00 00 00 86 76 69 65  77 00 00 03 D4 00 00 00  L
00000140   24 6C 75 6D 69 00 00 03  F8 00 00 00 14 6D 65 61  $lum
00000150   73 00 00 04 0C 00 00 00  24 74 65 63 68 00 00 04  s
00000160   30 00 00 00 0C 72 54 52  43 00 00 04 3C 00 00 08  0
```

[unregistered]
one.txt
C:\Files\Academics\VIT\Lab\Digita

File size: 59.0 KB
60,434 bytes

Default Edit Mode
State: original
Undo level: 0
Undo reverses: n/a

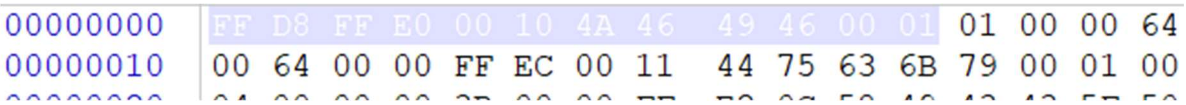
Creation time: 12-10-21
15:55:16

Last write time: 12-10-21
15:55:57

Attributes: A
Icons: 0

Mode: hexadecimal
Offsets: hexadecimal
Bytes per page: 24x16=384

Window #: 5



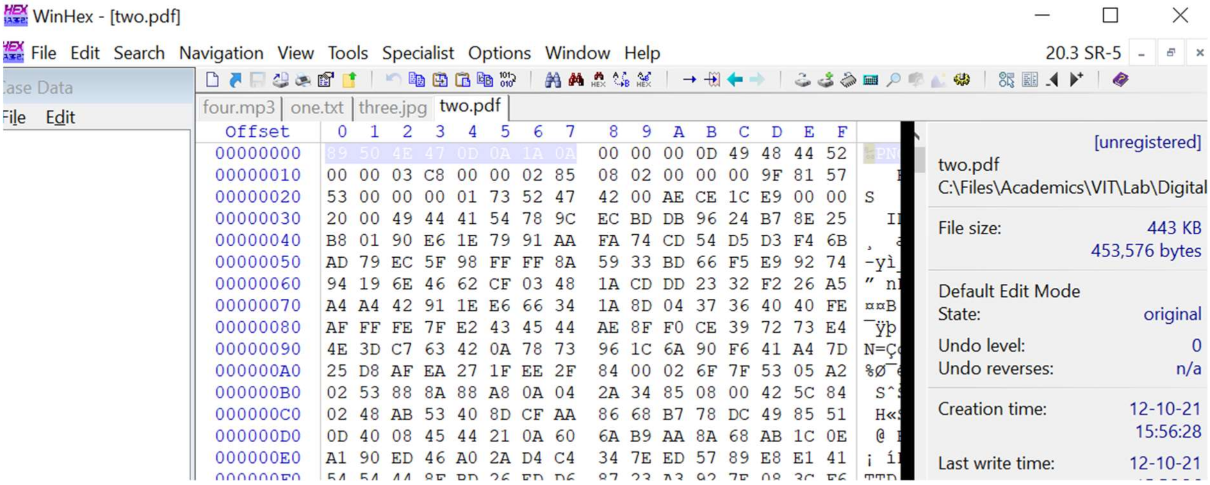
```
00000000   FF D8 FF E0 00 10 4A 46  49 46 00 01 01 00 00 64
00000010   00 64 00 00 FF EC 00 11  44 75 63 6B 79 00 01 00
```

Header of another JFIF file

The file opened in the editor is named ‘one.txt’ and the File Explorer recognises it as a text file. But, when we open it with a hex editor, we see the header to have this Hex signature, (highlighted in the image above) which reads: `FF D8 FF E0 00 10 4A 46 49 46 00 01`.

This is the signature of a JFIF file. Hence, we now know that the file is actually a JFIF file. When checked with another JFIF file’s header they are the same, but the footers are different. This may be due to the fact that they have different content.

two.png/.pdf


WinHex - [two.pdf]
File Edit Search Navigation View Tools Specialist Options Window Help
20.3 SR-5
Case Data
File Edit
four.mp3 | one.txt | three.jpg | two.pdf

```
Offset     0  1  2  3  4  5  6  7   8  9  A  B  C  D  E  F
00000000  89 50 4E 47 0D 0A 1A 0A  00 00 00 0D 49 48 44 52
00000010  00 00 03 C8 00 00 02 85  08 02 00 00 00 9F 81 57
00000020  53 00 00 00 01 73 52 47  42 00 AE CE 1C E9 00 00
00000030  20 00 49 44 41 54 78 9C  EC BD DB 96 24 B7 8E 25
00000040  B8 01 90 E6 1E 79 91 AA  FA 74 CD 54 D5 D3 F4 6B
00000050  AD 79 EC 5F 98 FF FF 8A  59 33 BD 66 F5 E9 92 74
00000060  94 19 6E 46 62 CF 03 48  1A CD DD 23 32 F2 26 A5
00000070  A4 A4 42 91 1E E6 66 34  1A 8D 04 37 36 40 40 FE
00000080  AF FF FE 7F E2 43 45 44  AE 8F F0 CE 39 72 73 E4
00000090  4E 3D C7 63 42 0A 78 73  96 1C 6A 90 F6 41 A4 7D
000000A0  25 D8 AF EA 27 1F EE 2F  84 00 02 6F 7F 53 05 A2
000000B0  02 53 88 8A 88 A8 0A 04  2A 34 85 08 00 42 5C 84
000000C0  02 48 AB 53 40 8D CF AA  86 68 B7 78 DC 49 85 51
000000D0  0D 40 08 45 44 21 0A 60  6A B9 AA 8A 68 AB 1C 0E
000000E0  A1 90 ED 46 A0 2A D4 C4  34 7E ED 57 89 E8 E1 41
```

[unregistered]
two.pdf
C:\Files\Academics\VIT\Lab\Digital

File size: 443 KB
453,576 bytes

Default Edit Mode
State: original

Undo level: 0
Undo reverses: n/a

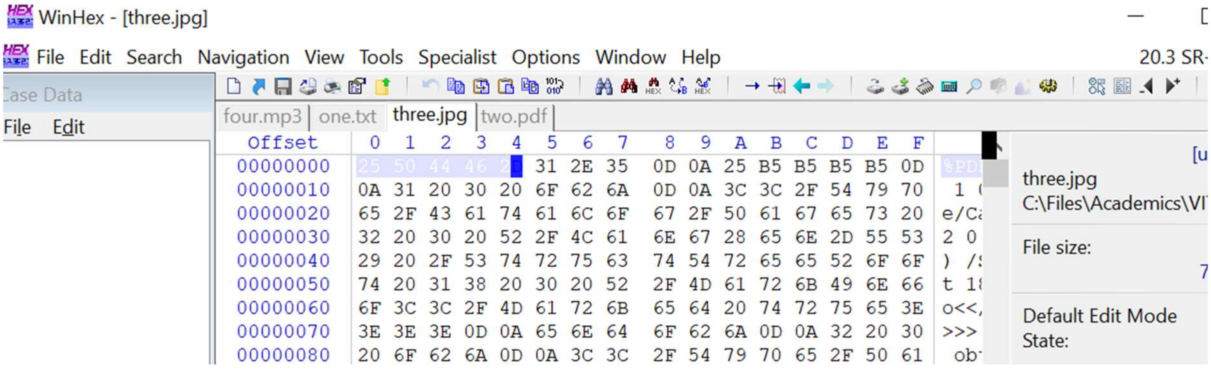
Creation time: 12-10-21
15:56:28

Last write time: 12-10-21

The file opened in the editor is named ‘two.pdf’ and the File Explorer recognises it as a PDF file. But, when we open it with a Hex editor, we see the header to have this Hex signature, (highlighted in the image above) which reads: `89 50 4E 47 0D 0A 1A 0A`.

This is the signature of a PNG file and PDF has a different hex signature as we will see in a following output. Hence, we now know that the file is actually a PNG image file. When compared to the header of another PNG’s header, we see that they are matching.

three.pdf/.jpg


WinHex - [three.jpg]
File Edit Search Navigation View Tools Specialist Options Window Help
20.3 SR-
Case Data
File Edit
four.mp3 | one.txt | three.jpg | two.pdf

```
Offset     0  1  2  3  4  5  6  7   8  9  A  B  C  D  E  F
00000000  25 50 44 46 2D 31 2E 35  0D 0A 25 B5 B5 B5 B5 0D
00000010  0A 31 20 30 20 6F 62 6A  0D 0A 3C 3C 2F 54 79 70
00000020  65 2F 43 61 74 61 6C 6F  67 2F 50 61 67 65 73 20
00000030  32 20 30 20 52 2F 4C 61  6E 67 28 65 6E 2D 55 53
00000040  29 20 2F 53 74 72 75 63  74 54 72 65 65 52 6F 6F
00000050  74 20 31 38 20 30 20 52  2F 4D 61 72 6B 49 6E 66
00000060  6F 3C 3C 2F 4D 61 72 6B  65 64 20 74 72 75 65 3E
00000070  3E 3E 3E 0D 0A 65 6E 64  6F 62 6A 0D 0A 32 20 30
00000080  20 6F 62 6A 0D 0A 3C 3C  2F 54 79 70 65 2F 50 61
```

three.jpg
C:\Files\Academics\VI

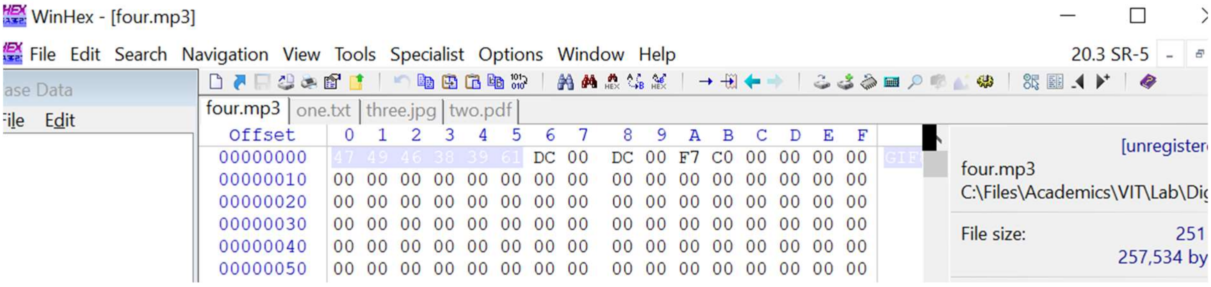
File size:

Default Edit Mode
State:

The file opened in the editor is named ‘three.jpg’ and the File Explorer recognises it as a JPG file. But, when we open it with a hex editor, we see the header to have this Hex signature, (highlighted in the image above) which reads: `25 50 44 46 2D`.

This is the signature of a PDF file. Hence, we now know that the file is actually a PDF file.

four.gif/.mp3



WinHex - [four.mp3]
File Edit Search Navigation View Tools Specialist Options Window Help
20.3 SR-5
ase Data
ile Edit
four.mp3 | one.txt | three.jpg | two.pdf
[unregister
four.mp3
C:\Files\Academics\VIT\Lab\Dig
File size: 251
257,534 by

```
Offset     0  1  2  3  4  5  6  7   8  9  A  B  C  D  E  F
00000000  47 49 46 38 39 61 DC 00  DC 00 F7 C0 00 00 00 00  GIF
00000010  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00000020  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00000030  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00000040  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00000050  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
```

The file opened in the editor is named ‘four.mp3’ and the File Explorer recognises it as an MP3 file. But, when opened with an MP3 player, it doesn’t play the file and closes due to corrupt data. When we open it with a hex editor, we see the header to have this Hex signature, (highlighted in the image above) which reads: `47 49 46 38 37 61.`

This is the signature of a GIF and MP3 has a different hex signature. Hence, we now know that the file is actually a GIF image.

OBSERVATIONS

Files of the same type always have the same file signatures in their header. This doesn’t change if the file’s extension is changed as their contents remain intact. The rest of the content excluding the header may vary for different files of the same type.

CONCLUSION

We now know how to identify file types with their header content which consists of their file signature. This is done with the help of a Hex editor such as win hex.

DIGITAL FORENSICS LAB

Exercise 10	
Name	S Shyam Sundaram
Registration Number	19BCE1560
Slot	L39+L40
Faculty	Dr. Seshu Babu Pulagara
Date	12th October, 2021

AIM

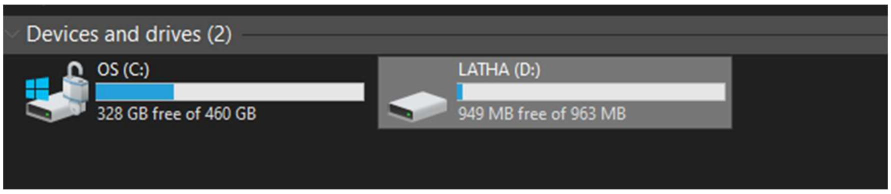
Working with TestDisk to recover deleted partitions and drives.

EXERCISE 1 - TestDisk

Identification of lost or deleted partitions.

A

First, a partition is created in the USB shown below.

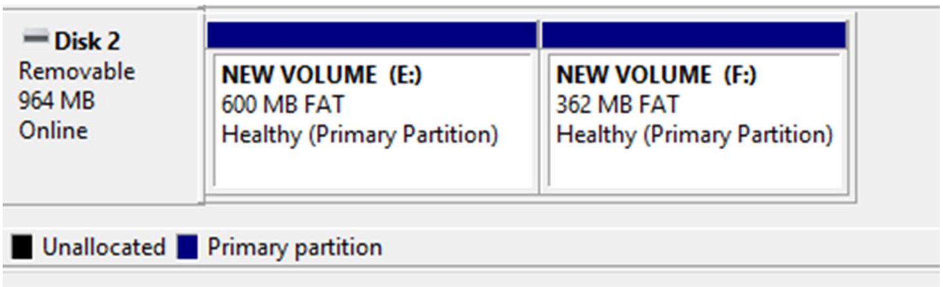


Devices and drives (2)

OS (C:)
328 GB free of 460 GB

LATHA (D:)
949 MB free of 963 MB

Next, two partitions were created on the USB drive. (Reference: https://www.windowscentral.com/how-set-usb-flash-drive-multiple-partitions-windows-10)

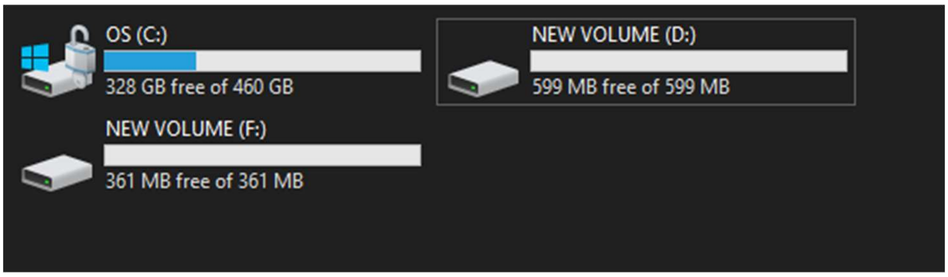


Disk 2
Removable
964 MB
Online

NEW VOLUME (E:)
600 MB FAT
Healthy (Primary Partition)

NEW VOLUME (F:)
362 MB FAT
Healthy (Primary Partition)

■ Unallocated ■ Primary partition

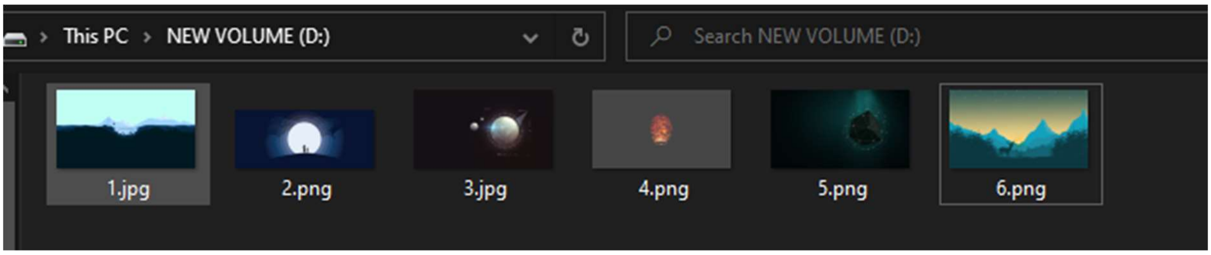


OS (C:)
328 GB free of 460 GB

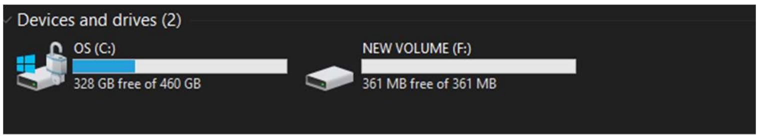
NEW VOLUME (D:)
599 MB free of 599 MB

NEW VOLUME (F:)
361 MB free of 361 MB

Partition D: was populated by a few JPG and PNG files.



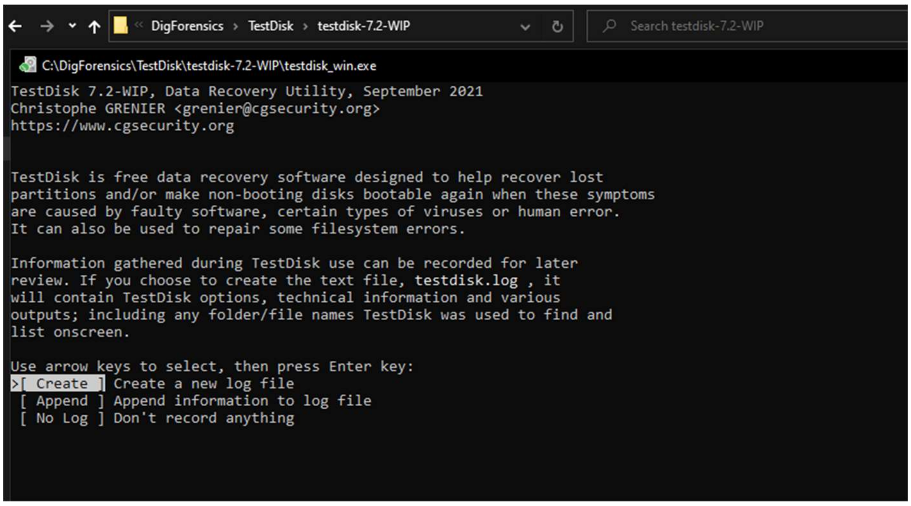
Now, that partition is deleted.



We now use TestDisk to see if we can identify the deleted partition.

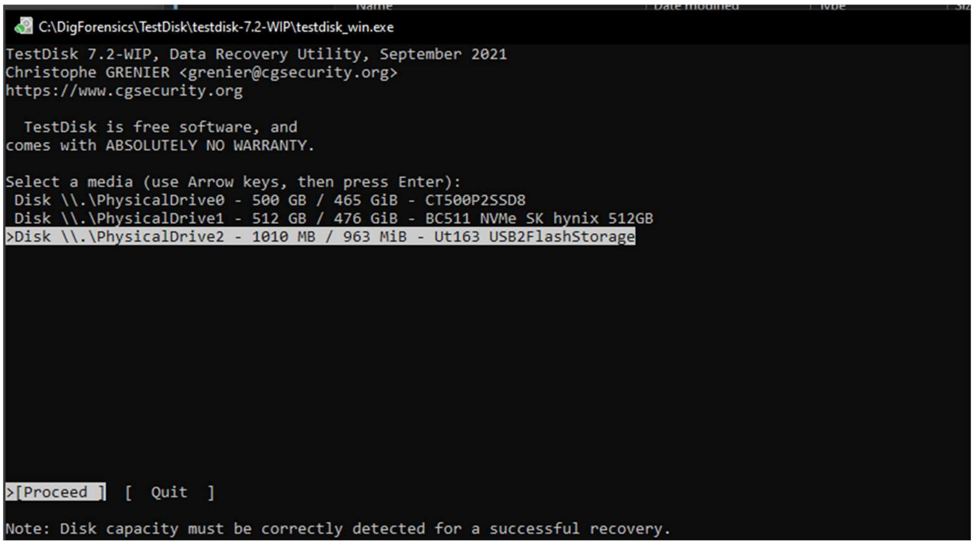
STEPS

1. Open testdisk_win.exe in the test-disk-7.2-WIP folder.
2. Select 'Create'


DigForensics > TestDisk > testdisk-7.2-WIP
Search testdisk-7.2-WIP
C:\DigForensics\TestDisk\testdisk-7.2-WIP\testdisk_win.exe

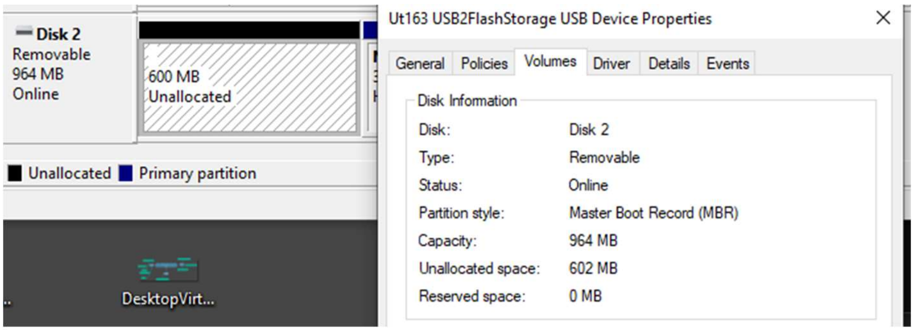
```
TestDisk 7.2-WIP, Data Recovery Utility, September 2021
Christophe GRENIER <grenier@cgsecurity.org>
https://www.cgsecurity.org


TestDisk is free data recovery software designed to help recover lost
partitions and/or make non-booting disks bootable again when these symptoms
are caused by faulty software, certain types of viruses or human error.
It can also be used to repair some filesystem errors.

Information gathered during TestDisk use can be recorded for later
review. If you choose to create the text file, testdisk.log , it
will contain TestDisk options, technical information and various
outputs; including any folder/file names TestDisk was used to find and
list onscreen.

Use arrow keys to select, then press Enter key:
>[ Create ] Create a new log file
 [ Append ] Append information to log file
 [ No Log ] Don't record anything
```

3. Select the disk of interest. Here, we choose the USB of 1 GB we had partitioned above.

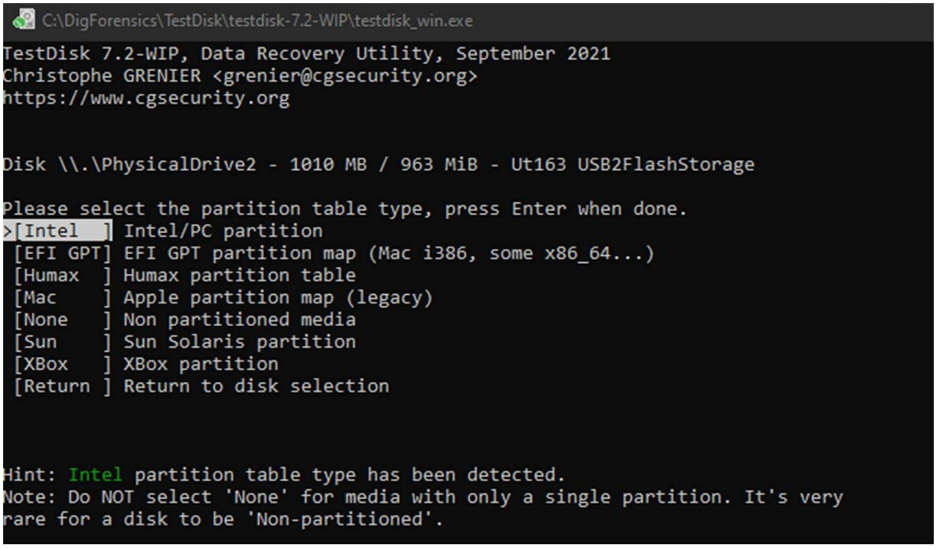


```
TestDisk 7.2-WIP, Data Recovery Utility, September 2021
Christophe GRENIER <grenier@cgsecurity.org>
https://www.cgsecurity.org

  TestDisk is free software, and
comes with ABSOLUTELY NO WARRANTY.

Select a media (use Arrow keys, then press Enter):
 Disk \\.\PhysicalDrive0 - 500 GB / 465 GiB - CT500P2SSD8
 Disk \\.\PhysicalDrive1 - 512 GB / 476 GiB - BC511 NVMe SK hynix 512GB
>Disk \\.\PhysicalDrive2 - 1010 MB / 963 MiB - Ut163 USB2FlashStorage













>[Proceed ]  [  Quit  ]

Note: Disk capacity must be correctly detected for a successful recovery.
```

4. Now, select the partition table type. We can see this by checking out the partition properties in the Disk Management program under Volumes tab. We see that this USB partition style is MBR. This means we select ‘Intel/PC partition’ option.

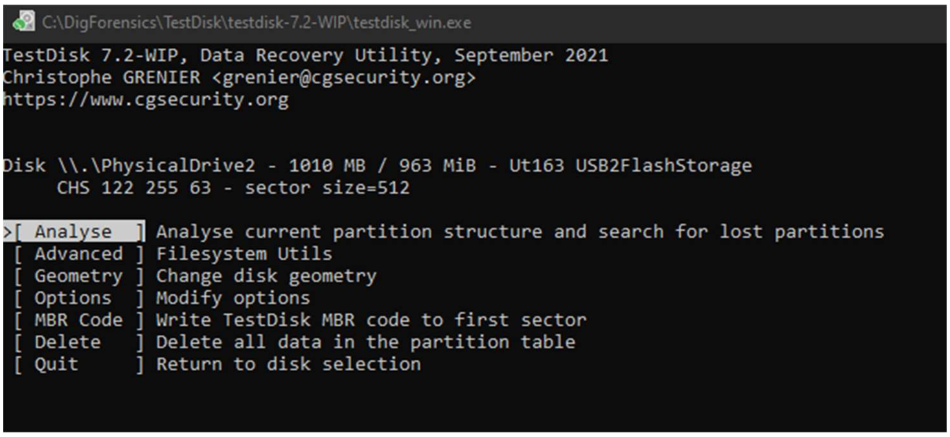


Disk:	Disk 2
Type:	Removable
Status:	Online
Partition style:	Master Boot Record (MBR)
Capacity:	964 MB
Unallocated space:	602 MB
Reserved space:	0 MB



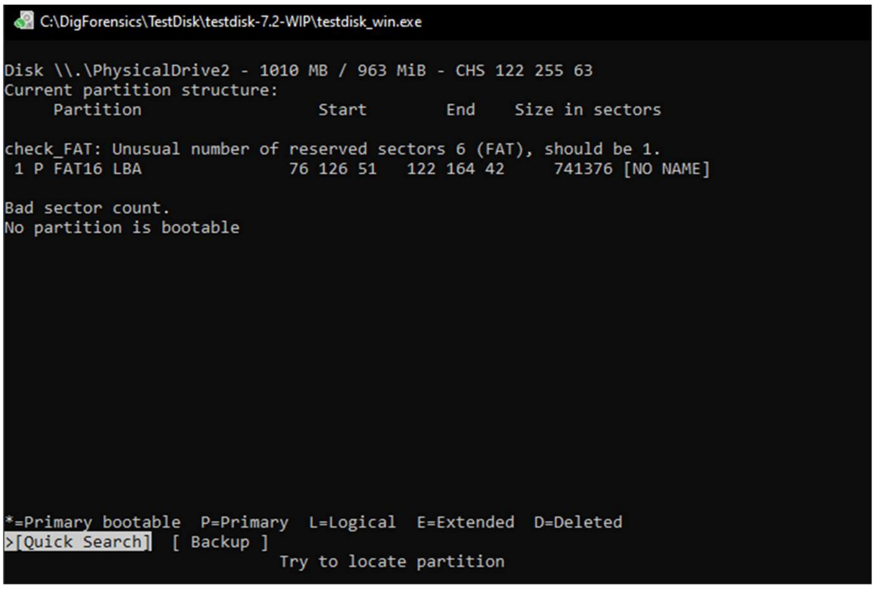
```
TestDisk 7.2-WIP, Data Recovery Utility, September 2021
Christophe GRENIER <grenier@cgsecurity.org>
https://www.cgsecurity.org


Disk \\.\PhysicalDrive2 - 1010 MB / 963 MiB - Ut163 USB2FlashStorage

Please select the partition table type, press Enter when done.
>[Intel  ] Intel/PC partition
 [EFI GPT] EFI GPT partition map (Mac i386, some x86_64...)
 [Humax  ] Humax partition table
 [Mac    ] Apple partition map (legacy)
 [None   ] Non partitioned media
 [Sun    ] Sun Solaris partition
 [XBox   ] XBox partition
 [Return ] Return to disk selection



Hint: Intel partition table type has been detected.
Note: Do NOT select 'None' for media with only a single partition. It's very
rare for a disk to be 'Non-partitioned'.
```

5. Select ‘Analyse’



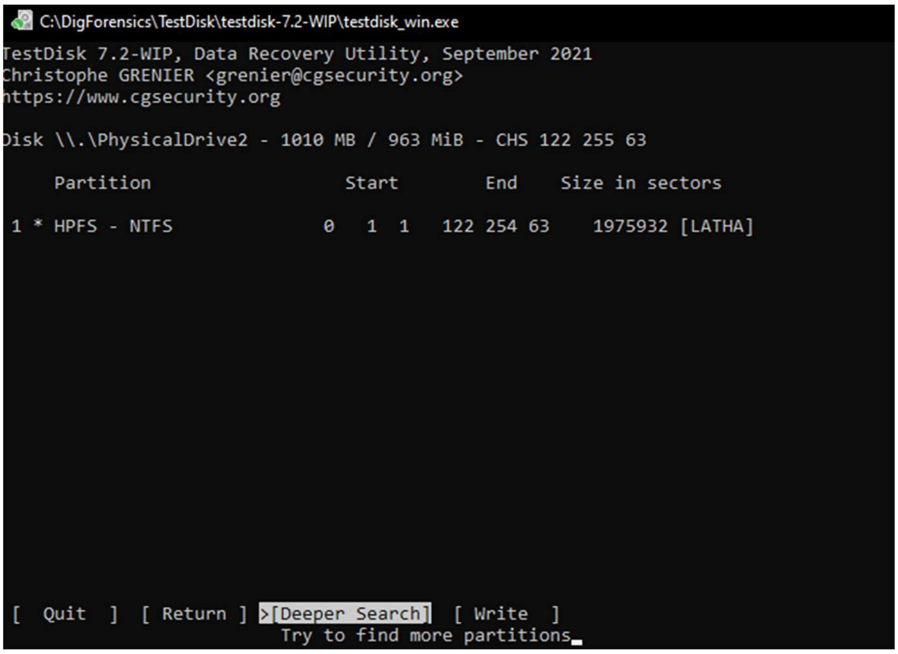
```
TestDisk 7.2-WIP, Data Recovery Utility, September 2021
Christophe GRENIER <grenier@cgsecurity.org>
https://www.cgsecurity.org


Disk \\.\PhysicalDrive2 - 1010 MB / 963 MiB - Ut163 USB2FlashStorage
     CHS 122 255 63 - sector size=512

>[ Analyse  ] Analyse current partition structure and search for lost partitions
 [ Advanced ] Filesystem Utils
 [ Geometry ] Change disk geometry
 [ Options  ] Modify options
 [ MBR Code ] Write TestDisk MBR code to first sector
 [ Delete   ] Delete all data in the partition table
 [ Quit     ] Return to disk selection
```

6. Select ‘Quick Search’.



```
Disk \\.\PhysicalDrive2 - 1010 MB / 963 MiB - CHS 122 255 63
Current partition structure:
     Partition                  Start        End    Size in sectors

check_FAT: Unusual number of reserved sectors 6 (FAT), should be 1.
 1 P FAT16 LBA               76 126 51   122 164 42     741376 [NO NAME]

Bad sector count.
No partition is bootable






*=Primary bootable  P=Primary  L=Logical  E=Extended  D=Deleted
>[Quick Search]  [ Backup ]
                            Try to locate partition
```

7. Select ‘Deeper Search’.



```
TestDisk 7.2-WIP, Data Recovery Utility, September 2021
Christophe GRENIER <grenier@cgsecurity.org>
https://www.cgsecurity.org

Disk \\.\PhysicalDrive2 - 1010 MB / 963 MiB - CHS 122 255 63

     Partition                  Start        End    Size in sectors

 1 * HPFS - NTFS              0   1  1   122 254 63    1975932 [LATHA]






[  Quit  ]  [ Return ] >[Deeper Search]  [ Write  ]
                          Try to find more partitions
```

8. Upon deeper Search, the deleted partition is not visible.

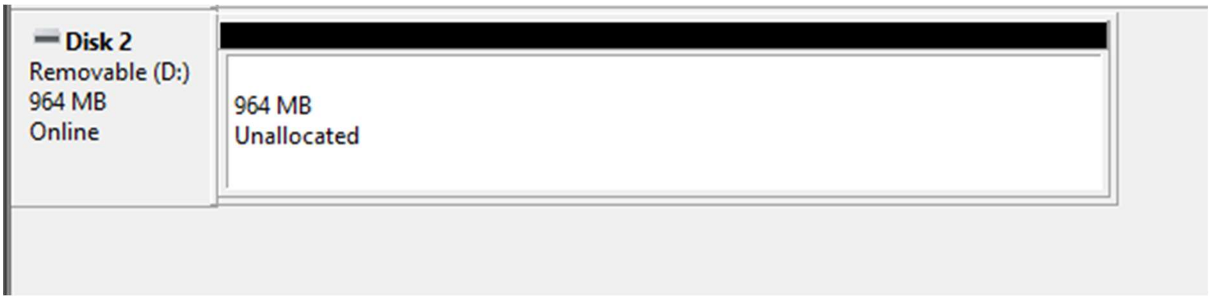
Note: As this tool did not detect deleted Partition and Undelete requires registration with payment to be used. For this exercise, the third suggested tool, EaseUs is used.

EXERCISE 2 and 3 - EaseUS

Q Identification and recovery of deleted/lost partitions and files.

A

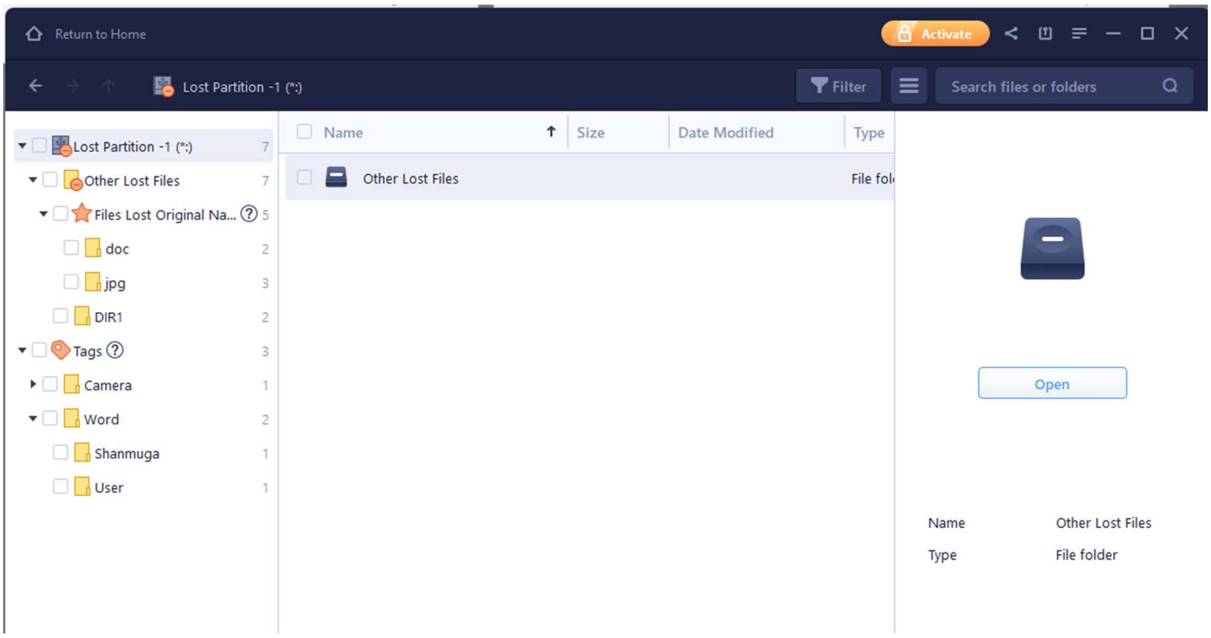
For this exercise, the same USB is used and is populated with a JPG file. Then, the whole partition is deleted as seen below.



Disk 2
Removable (D:)
964 MB
Online

964 MB
Unallocated

Next, we open EaseUS and scan the drive. Upon scanning, we see the following:

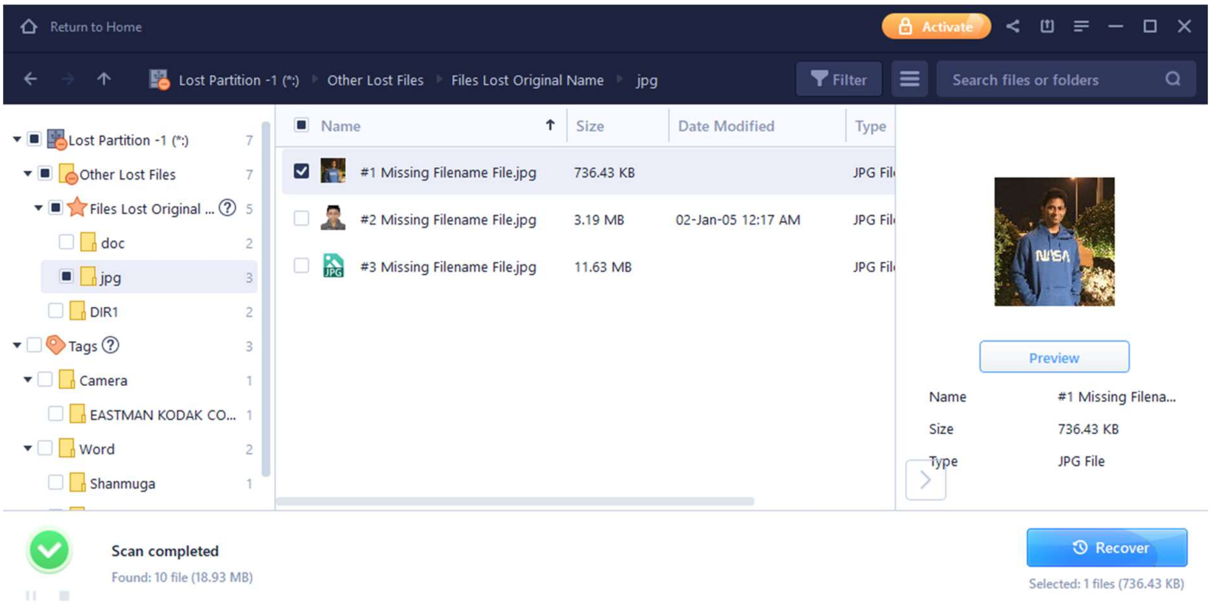


- Lost Partition -1 (*:) 7
  - Other Lost Files 7
    - Files Lost Original Na... 5
      - doc 2
      - jpg 3
    - DIR1 2
- Tags 3
  - Camera 1
  - Word 2
    - Shanmuga 1
    - User 1

Name	Size	Date Modified	Type
Other Lost Files			File fol

Name Other Lost Files
Type File folder

All the files from the deleted partition are found. Even certain older files are discovered.



■ Name	Size	Date Modified	Type
☑ #1 Missing Filename File.jpg	736.43 KB		JPG Fil
□ #2 Missing Filename File.jpg	3.19 MB	02-Jan-05 12:17 AM	JPG Fil
□ #3 Missing Filename File.jpg	11.63 MB		JPG Fil

Name #1 Missing Filena...
Size 736.43 KB
Type JPG File

Scan completed
Found: 10 file (18.93 MB)

Selected: 1 files (736.43 KB)

The images from the deleted partition are also discovered! To recover, we click ‘Recover’. This recovery will happen only after payment to use the software is done.

OBSERVATIONS

We see that partitions and files deleted can indeed be recovered. These need not be recent files necessarily. Even much older data deleted from drives can be recovered with tools. Some tools are more effective than others (as seen, EaseUS was able to detect and recover partitions and files, whereas TestDisk could not discover the deleted partition itself) but may be proprietary and require purchase of license.

CONCLUSION

Data and partitions deleted can be recovered from drives.