

You can further customize or add more specific analysis based on your data and requirements.

...

This script will properly analyze your data after converting 'createdDateTime' to datetime objects.

Feel free to customize or add more specific analysis as needed.

USER

Now make it install all

ASSISTANT

To install the required libraries for the script, you can use the following code in your Colab environment:

```
```python
```

```
!pip install pandas
```

```
```
```

This will install the 'pandas' library, which is necessary for data manipulation and analysis. You can run this command at the beginning of your Colab notebook to ensure that the library is installed.

If you need to install additional libraries, you can similarly use `!pip install library_name` to install them.

USER

I'll make it install every application that I get you just by the application ID

ASSISTANT

To install Python packages using the `!pip` command within a Jupyter or Colab environment, you need to know the names of the packages you want to install. The package names typically don't match the application IDs you provided earlier.

If you have a list of package names and want to install them, you can create a Python script to automate the process. Here's an example of how you can do that:

```
```python
import subprocess

List of package names to install
package_names = ['package1', 'package2', 'package3']

for package in package_names:
 try:
 subprocess.check_output(['pip', 'install', package])
 print(f"Installed: {package}")
 except Exception as e:
 print(f"Failed to install: {package}, Error: {e}")
```
```

Replace `'package1'`, `'package2'`, etc., with the actual names of the packages you want to install. This script will attempt to install each package and print whether it was successfully installed or not.

USER

1. Basic Statistics:

| | id | displayName \ |
|--------|--------------------------------------|----------------------------|
| count | 1066 | 578 |
| unique | 1066 | 571 |
| top | 00a67483-ea0d-4bfa-a19b-17ae35e97fe3 | Domain Controller Services |
| freq | 1 | 3 |
| first | NaN | NaN |
| last | NaN | NaN |

| | appId | createdDateTime \ |
|--------|--------------------------------------|---------------------------|
| count | 578 | 578 |
| unique | 578 | 298 |
| top | 3f18b202-a866-4c1f-b9c0-6598390a2092 | 2022-03-28 03:48:59+00:00 |
| freq | 1 | 31 |
| first | NaN | 2022-03-04 02:35:55+00:00 |
| last | NaN | 2022-05-04 08:34:59+00:00 |

| | applicationType | accountEnabled | applicationVisibility \ |
|--------|-----------------------|----------------|-------------------------|
| count | 573 | 578 | 578 |
| unique | 3 | 2 | 2 |
| top | Microsoft Application | True | Visible |
| freq | 497 | 576 | 576 |
| first | NaN | NaN | NaN |
| last | NaN | NaN | NaN |

assignmentRequired isAppProxy

| | | |
|--------|-------|-------|
| count | 578 | 578 |
| unique | 2 | 1 |
| top | False | False |
| freq | 535 | 578 |
| first | NaN | NaN |
| last | NaN | NaN |

2. Unique Application Types:

['Enterprise Application' nan 'Microsoft Application' 'Managed Identity']

3. Unique Application IDs:

['3f18b202-a866-4c1f-b9c0-6598390a2092'
'fd681d32-9e97-4876-a58f-05269cf838e4'
'2e8265ff-5066-4e5d-ae61-dbcc32816115'
'98db8bd6-0cc0-4e67-9de5-f187f1cd1b41'
'0947a342-ab4a-43be-93b3-b8243fc161e5'
'607ece82-f922-494f-88b8-30effaf12214'
'6db4d6bb-6649-4dc2-84b7-0b5c6894031e'
'b692184e-b47f-4706-b352-84b288d2d9ee'
'354b5b6d-abd6-4736-9f51-1be80049b91f'
'2ff814a6-3304-4ab8-85cb-cd0e6f879c1d'
'18a66f5f-dbdf-4c17-9dd7-1634712a9cbe'
'b503eb83-1222-4dcc-b116-b98ed5216e05'
'f5c26e74-f226-4ae8-85f0-b4af0080ac9e'
'bb2a2e3a-c5e7-4f0a-88e0-8e01fd3fc1f4'
'584a29b4-7876-4445-921e-71e427d4f4b3']

'd3e1650d-38f0-4a3c-8c2d-a064cc8d9893'

'f3c21649-0979-4721-ac85-b0216b2cf413'

'bc59ab01-8403-45c6-8796-ac3ef710b3e3'

'a303894e-f1d8-4a37-bf10-67aa654a0596'

'6f2d169c-08f3-4a4c-a982-bcaf2d038c45'

'448adbda-b8d8-4f33-a1b0-ac58cf44d4c1'

'60e6cd67-9c8c-4951-9b3c-23c25a2169af'

'799f1985-1517-4fe1-af2b-ba3d87d4996b'

'1448fd13-7e74-41f4-b6e3-17e485d8ac2e'

'5e5e43d4-54da-4211-86a4-c6e7f3715801'

'0cd79364-7a90-4354-9984-6e36c841418d'

'1f5530b3-261a-47a9-b357-ded261e17918'

'621269cf-1195-44a3-a835-c613d103dd15'

'00000002-0000-0ff1-ce00-000000000000'

'996def3d-b36c-4153-8607-a6fd3c01b89f'

'00000003-0000-0ff1-ce00-000000000000'

'22d7579f-06c2-4baa-89d2-e844486adb9d'

'c5b17a4f-cc6f-4649-9480-684280a2af3a'

'73c2949e-da2d-457a-9607-fcc665198967'

'67e3df25-268a-4324-a550-0de1c7f97287'

'd73f4b35-55c9-48c7-8b10-651f6f2acb2e'

'eb070ea5-bd17-41f1-ad68-5851f6e71774'

'4ac7d521-0382-477b-b0f8-7e1d95f85ca2'

'9ddb5749-5660-445d-b859-394225a7d97d'

'3e8f1ed5-3a6e-47d1-ad1e-b14e6b69704b'

'a6aa9161-5291-40bb-8c5c-923b567bee3b'

'56c1da01-2129-48f7-9355-af6d59d42766'

'ab27a73e-a3ba-4e43-8360-8bcc717114d8'

'62c559cd-db0c-4da0-bab2-972528c65d42'

'50e95039-b200-4007-bc97-8d5790743a63'

'39495caf-cc21-4d03-b6b0-8c4a973cf213'

'05cf5e27-931d-47ad-826d-cb9028d8bd7a'

'dfbed8b2-492a-414e-b2f0-482534e87bc5'

'810dcf14-1858-4bf2-8134-4c369fa3235b'

'328fd23b-de6e-462c-9433-e207470a5727'

'f3723d34-6ff5-4ceb-a148-d99dcd2511fc'

'66244124-575c-4284-92bc-fdd00e669cea'

'd9794ed7-bfec-42d8-ba8d-4d7df2a1f6dc'

'00000012-0000-0000-c000-000000000000'

'9bdab391-7bbe-42e8-8132-e4491dc29cc0'

'aa9ecb1e-fd53-4aaa-a8fe-7a54de2c1334'

'e933bd07-d2ee-4f1d-933c-3752b819567b'

'5bd9995f-b6f8-4c7a-a024-e8c5eab9c85d'

'f3625a3e-6360-4580-968d-fae4cab75a0'

'80a10ef9-8168-493d-abf9-3297c4ef6e3c'

'2cf9eb86-36b5-49dc-86ae-9a63135dfa8c'

'bc313c14-388c-4e7d-a58e-70017303ee3b'

'e5d38734-8683-4c09-9e0d-73a883c2d83d'

'20e940b3-4c77-4b0b-9a53-9e16a1b010a7'

'562db366-1b96-45d2-aa4a-f2148cef2240'

'7e3bc4fd-85a3-4192-b177-5b8bfc87f42c'

'2368d027-f996-4edb-bf48-928f98f2ab8c'

'f6ec9458-aa91-4c6e-bb3f-7b7c946c1b44'

'14452459-6fa6-4ec0-bc50-1528a1a06bf0'

'c61423b7-1d1f-430d-b444-0eee53298103'

'3184af01-7a88-49e0-8b55-8ecdce0aa950'

'163b648b-025e-455b-9937-a7f39a65d171'

'b4bddae8-ab25-483e-8670-df09b9f1d0ea'

'e9f49c6b-5ce5-44c8-925d-015017e9f7ad'

'8602e328-9b72-4f2d-a4ae-1387d013a2b3'

'c161e42e-d4df-4a3d-9b42-e7a3c31f59d4'

'abfa0a7c-a6b6-4736-8310-5855508787cd'

'fbc197b7-9e9c-4f98-823f-93cb1cb554e6'

'8b3391f4-af01-4ee8-b4ea-9871b2499735'

'89d10474-74af-4874-99a7-c23c2f643083'

'7c33bfcb-8d33-48d6-8e60-dc6404003489'

'2f3f02c9-5679-4a5c-a605-0de55b07d135'

'65b0413c-6b56-4536-a2ea-1929dfe5e81c'

'6d49f6be-5347-432d-b9b4-64ac938c1bd2'

'f18474f2-a66a-4bb0-a3c9-9b8d892092fa'

'f2c304cf-8e7e-4c3f-8164-16299ad9d272'

'359431ad-ece5-496b-8768-be4bbfd82f36'

'709110f7-976e-4284-8851-b537e9bcb187'

'e4ab13ed-33cb-41b4-9140-6e264582cf85'

'0bf30f3b-4a52-48df-9a82-234910c4a086'

'2cc451ba-a8ec-496f-bdff-591f5ae2876c'

'0ecb6dbc-7807-4951-9a69-b5d3dfa5a0b5'

'79d7fb34-4bef-4417-8184-ff713af7a679'

'5d13f7d7-0567-429c-9880-320e9555e5fc'
'0b07f429-9f4b-4714-9392-cc5e8e80c8b0'
'48ef7923-268f-473d-bcf1-07f0997961f4'
'74bcdadc-2fdc-4bb3-8459-76d06952a0e9'
'81473081-50b9-469a-b9d8-303109583ecb'
'f8f7a2aa-e116-4ba6-8aea-ca162cfa310d'
'2793995e-0a7d-40d7-bd35-6968ba142197'
'982bda36-4632-4165-a46a-9863b1bbcf7d'
'0000000a-0000-0000-c000-000000000000'
'a3f23ff7-be0d-49a0-b535-f5fb0f5f8039'
'e158b4a5-21ab-442e-ae73-2e19f4e7d763'
'f9c691e6-93b3-4d57-944c-afcc737f9abf'
'632ec9eb-fad7-4cbd-993a-e72973ba2acc'
'95c8b18b-3ed0-4299-88e6-31a12c4b4092'
'd96199e7-4674-4bbf-a1c6-ddf93682f5ee'
'e64bd61e-5424-451f-b666-e02ee2878437'
'a4c1cdb3-88ab-4d13-bc99-1c46106f0727'
'86adf623-eea3-4453-9f4a-18134ac1410d'
'63e61dc2-f593-4a6f-92b9-92e4d2c03d4f'
'6f0478d5-61a3-4897-a2f2-de09a5a90c7f'
'4c8f074c-e32b-4ba7-b072-0f39d71daf51'
'c44b4083-3bb0-49c1-b47d-974e53cbdf3c'
'a8b6bf88-1d1a-4626-b040-9a729ea93c65'
'441509e5-a165-4363-8ee7-bcf0b7d26739'
'b8066b99-6e67-41be-abfa-75db1a2c8809'
'b8cf62f3-7cc7-4e32-ab3a-41370ef0cfcf'

'00000002-0000-0000-c000-000000000000'

'01fc33a7-78ba-4d2f-a4b7-768e336e890e'

'fc75330b-179d-49af-87dd-3b1acf6827fa'

'0c6620df-7b29-44de-8ba4-688a56a20f9f'

'8bbf8725-b3ca-4468-a217-7c8da873186e'

'8d5683a1-43b9-4f38-8906-504eae3c36c9'

'e6650347-047f-4e51-9386-839384472ea5'

'74b0b707-6e48-48c5-a6f8-6ad47db27678'

'de742ffc-b441-4542-8646-7e805426b824'

'4392ab71-2ce2-4b0d-8770-b352745c73f5'

'4fa46669-56c9-44e7-a69b-182480b952a8'

'5da7367f-09c8-493e-8fd4-638089cddec3'

'a3dfc3c6-2c7d-4f42-aeec-b2877f9bce97'

'374b2a64-3b6b-436b-934c-b820eacca870'

'6bccf540-eb86-4037-af03-7fa058c2db75'

'6e110594-2adf-49c8-8d37-41eee81c0359'

'30b7f32a-8115-4e12-883b-1fd99dd0efd9'

'fc68d9e5-1f76-45ef-99aa-214805418498'

'f18b59c9-5926-4a65-8605-c23ec8c7e074'

'601d4e27-7bb3-4dee-8199-90d47d527e1c'

'52aa7af2-7031-4163-a511-b1ca35c43c6e'

'd2fa1650-4805-4a83-bcb9-cf41fe63539c'

'bd93b475-f9e2-476e-963d-b2daf143ffb9'

'27a762be-14e7-4f92-899c-151877d6d497'

'bf8eb16c-7ba7-4b47-86be-ac5e4b2007a5'

'9ed4cd8c-9a98-405f-966b-38ab1b0c24a3'

'd3315f6c-968a-40bb-94d2-a6a9503b05f5'

'dad37da6-229d-4bc0-8b94-fee8600589db'

'f49e6313-6c53-4ec8-918f-2dd9f1980342'

'4bfd5d66-9285-44a1-bb14-14953e8cdf5e'

'e3583ad2-c781-4224-9b91-ad15a8179ba0'

'f77c2a8f-8a0a-4776-8e0a-bcb2549610ca'

'ec52d13d-2e85-410e-a89a-8c79fb6a32ac'

'f6b60513-f290-450e-a2f3-9930de61c5e7'

'5e5abe2b-83cd-4786-826a-a05653ebb103'

'18af356b-c4fd-4f52-9899-d09d21397ab7'

'7b7531ad-5926-4f2d-8a1d-38495ad33e17'

'00000003-0000-0000-c000-000000000000'

'5e89db78-b7c2-42c9-91f3-12ee3bc42550'

'bf283ae6-5efd-44a8-b56a-2a7939982d60'

'a967240f-810b-4f79-85e5-25870cc69cbb'

'f69cfde7-4247-48a2-a54d-1501570d8233'

'e18cedde-9458-482f-9dd1-558c597ac42e'

'bf9fc203-c1ff-4fd4-878b-323642e462ec'

'765fe668-04e7-42ba-aec0-2c96f1d8b652'

'37182072-3c9c-4f6a-a4b3-b3f91cacffce'

'3b5c2f38-d6e7-4b96-944c-872d4fe64fe6'

'728a93e3-065d-4678-93b1-3cc281223341'

'1ac05c7e-12d2-4605-bf9d-549d7041c6b3'

'91ff567f-bb4f-4719-91d7-d983057bc0d6'

'8fca0a66-c008-4564-a876-ab3ae0fd5cff'

'96231a05-34ce-4eb4-aa6a-70759cbb5e83'

'0ac71e58-8aec-485e-a4ec-6b24af1b962a'
'36e2398c-9dd3-4f29-9a72-d9f2cfc47ad9'
'7a08d6a2-494e-48eb-85cd-8c245104a6fd'
'894b1496-c6e0-4001-b69c-81b327564ca4'
'cf32a0cc-373c-47c9-9156-0db11f6a6dfc'
'c6932912-e55b-494c-b080-de9a35ce410c'
'e00d2f8a-f6c8-46e4-b379-e66082e28ca8'
'e1b4ed32-0aea-4286-88bf-5df97be96abc'
'a15bc1de-f777-408f-9d2b-a27ed19c72ba'
'db559737-c65c-4c91-b5d2-c51a55c9e22b'
'366cbfa5-46b3-47fb-9d70-55fb923b4833'
'3b2fa68d-a091-48c9-95be-88d572e08fb7'
'9c012f45-4a09-4f90-9cb9-5dea81f53a9e'
'5613cb5c-a7c9-4099-8034-511fd7616cb2'
'3aa5c166-136f-40eb-9066-33ac63099211'
'0000dab9-8b21-4ba2-807f-1743968cef00'
'57c0fc58-a83a-41d0-8ae9-08952659bdfd'
'7c99d979-3b9c-4342-97dd-3239678fb300'
'87df0fbf-e22d-4d7c-bc30-f59ca7460837'
'56e5be3f-8703-4fe3-936b-8599b606b8fc'
'ae11f5fb-c627-4eec-b4a0-f7b5969426e5'
'80369ed6-5f11-4dd9-bef3-692475845e77'
'b121a960-2b39-48d2-a3ed-9efe1b7f8992'
'e3335adb-5ca0-40dc-b8d3-bedc094e523b'
'3ae55346-4fae-430e-96f8-e26e8315f27e'
'71af9eaf-3a7b-43eb-bc59-c504bfe12240'

'e7d5afaf-5e93-4aad-b546-878812ff572c'

'f71766dc-90d9-4b7d-bd9d-4499c4331c3f'

'91f2f07a-57ac-44c9-89e6-c9e872cb470e'

'24d3987b-be4a-48e0-a3e7-11c186f39e41'

'00a571e7-e41d-4ed4-8eef-36ee2632264d'

'c9e0b461-3515-4a03-b576-ed91ed4336d'

'9e09aefc-b2e5-4d19-9f74-3e3e8b11a57b'

'1412d89f-b8a8-4111-b4fd-e82905cbd85d'

'af50c6e1-9fa7-4512-8e63-33d5541d2b9d'

'92b61450-2139-4e4a-a0cc-898eced7a779'

'abba844e-bc0e-44b0-947a-dc74e5d09022'

'1322e676-dee7-41ee-a874-ac923822781c'

'ce34e7e5-485f-4d76-964f-b3d2b16d1e4f'

'76cd24bf-a9fc-4344-b1dc-908275de6d6d'

'070fc472-7cef-4d53-9b65-34464c4d5f4a'

'925eb0d0-da50-4604-a19f-bd8de9147958'

'ce52bcbe-f669-4cfd-9866-d64472d3e13e'

'a0df4d2c-64a4-4e93-8f8e-4c8d721b5b77'

'de096ee1-dae7-4ee1-8dd5-d88ccc473815'

'd10de03d-5ba3-497a-90e6-7ff8c9736059'

'035f9e1d-4f00-4419-bf50-bf2d87eb4878'

'396e7f4b-41ea-4851-b04d-65de6cf1b4a3'

'6b3368c6-61d2-4a72-854c-42d1c4e71fed'

'1a14be2a-e903-4cec-99cf-b2e209259a0f'

'a0e1e353-1a3e-42cf-a8ea-3a9746eec58c'

'bd9b7cd5-dac1-495f-b013-ac871e98fa5f'

'e3bfd6ac-eace-4438-9dc1-eed439e738de'
'a5472e16-e1d2-4bbe-81b3-ecdcd459b536'
'880da380-985e-4198-81b9-e05b1cc53158'
'5b404cf4-a79d-4cfe-b866-24bf8e1a4921'
'637fcc9f-4a9b-4aaa-8713-a2a3cfda1505'
'ea2f600a-4980-45b7-89bf-d34da487bda1'
'0eb690b7-d23e-4fb0-b43e-cd161ac80cc3'
'd4ebce55-015a-49b5-a083-c84d1797ae8c'
'c699bf69-fb1d-4eaf-999b-99e6b2ae4d85'
'63ea3c01-7483-456e-8073-d3fed34fbdda'
'cfa8b339-82a2-471a-a3c9-0fc0be7a4093'
'918d0db8-4a38-4938-93c1-9313bdfe0272'
'a12e8ccb-0fcd-46f8-b6a1-b9df7a9d7231'
'5ed8fe41-c1bc-4c06-a531-d91e1f1c2fac'
'fc780465-2017-40d4-a0c5-307022471b92'
'c19490b5-c092-426f-b1a2-674b279d4975'
'823c0a78-5de0-4445-a7f5-c2f42d7dc89b'
'1b912ec3-a9dd-4c4d-a53e-76aa7adb28d7'
'a466b239-e85b-439c-a3b5-f0ee6ce3b7b5'
'02069796-fdae-431f-832d-06f03bd0ccf6'
'cc46c2aa-d508-409b-aeb7-df7cd1e07aaa'
'76c92352-c057-4cc2-9b1e-f34c32bc58bd'
'847bd4c1-7486-4241-a783-f9bda69241c1'
'319f651f-7ddb-4fc6-9857-7aef9250bd05'
'981f26a1-7f43-403b-a875-f8b09b8cd720'
'6a0ec4d3-30cb-4a83-91c0-ae56bc0e3d26'

'432d3ac8-d6b9-43b9-9ef5-236ea50a049a'

'1efe5bbf-d5b1-4fe9-99fa-f55ce1c88679'

'a0f92522-89de-4c5e-9a75-0044ccf66efd'

'8c420feb-03df-47cc-8a05-55df0cf3064b'

'224a7b82-46c9-4d6b-8db0-7360fb444681'

'7319c514-987d-4e9b-ac3d-d38c4f427f4c'

'a232010e-820c-4083-83bb-3ace5fc29d0b'

'ea21b132-560f-4b0b-9876-903b6bca7b9d'

'585fc3c3-9a59-4720-8319-53cce041a605'

'408992c7-2af6-4ff1-92e3-65b73d2b5092'

'349e15d0-1c96-4829-95e5-7fc8fb358ff3'

'213f5f78-fb30-46c7-9e98-91c720a1c026'

'61c50b89-703d-431d-8d80-1e8618748775'

'f661c70c-c136-49d2-9b94-a2033c5874f8'

'4c1a3aed-b389-4824-99b0-514c07906851'

'6e10398e-57ba-4077-9027-e54a38dbc6dd'

'997dc448-eeab-4c93-8811-6b2c80196a16'

'6da466b6-1d13-4a2c-97bd-51a99e8d4d74'

'1212c1c8-6a34-4cfb-8ecb-1ffa5cbb9a39'

'6ee392c4-d339-4083-b04d-6b7947c6cf78'

'3efe2b8f-32eb-4c7a-ae9a-b43f17de36eb'

'00000006-0000-0ff1-ce00-000000000000'

'486c78bf-a0f7-45f1-92fd-37215929e116'

'341b7f3d-69b3-47f9-9ce7-5b7f4945fdbd'

'a57aca87-cbc0-4f3c-8b9e-dc095fdc8978'

'184909ca-69f1-4368-a6a7-c558ee6eb0bd'

'64b12d6e-6549-484c-8cc6-6281839ba394'

'ddb3205-c6bd-46ae-8127-60eb93363864'

'ca7f3f0b-7d91-482c-8e09-c5d840d0eac5'

'2a486b53-dbd2-49c0-a2bc-278bdfc30833'

'3cf6df92-2745-4f6f-bbcb-19b59bcdb62a'

'f98181d0-e16e-497c-9ac7-8c8e119ab8b5'

'9fc8264d-0a40-4790-86e1-e7b73a2d2298'

'797f4846-ba00-4fd7-ba43-dac1f8f63013'

'0426b777-1c7e-47f1-bdc1-e686dd0aedef1'

'4d0ad6c7-f6c3-46d8-ab0d-1406d5e6c86b'

'51df634f-ddb4-4901-8a2d-52f6393a796b'

'150c8903-2280-4ab6-8708-b080044d94c6'

'c7f7c435-4759-4a66-8981-13dc508936ba'

'04687a56-4fc2-4e36-b274-b862fb649733'

'4353526e-1c33-4fcf-9e82-9683edf52848'

'dba650ed-9577-4bc0-9b5f-ef73e2d5bdfc'

'992ddfc7-4a57-44ca-aeb5-ce7846062e10'

'cfbd4387-1a16-4945-83c0-ec10e46cd4da'

'd84904f1-cfbf-46a4-a8ed-6229ba95e220'

'bb8f18b0-9c38-48c9-a847-e1ef3af0602d'

'd828acde-4b48-47f5-a6e8-52460104a052'

'603b8c59-ba28-40ff-83d1-408eee9a93e5'

'eec53b1f-b9a4-4479-acf5-6b247c6a49f2'

'91bb937c-29c2-4275-982f-9465f0caf03d'

'76c7f279-7959-468f-8943-3954880e0d8c'

'a91b1853-4403-4f54-b5cb-d1ea19d90c37'

'93ac0ec6-05eb-4bed-94b2-4e955ebc01d3'

'b6b84568-6c01-4981-a80f-09da9a20bbbed'

'7cd684f4-8a78-49b0-91ec-6a35d38739ba'

'47ee738b-3f1a-4fc7-ab11-37e4822b007e'

'be14bf7e-8ab4-49b0-9dc6-a0eddd6fa73e'

'e6f9f783-1fdb-4755-acaf-abed6c642885'

'c7ddd9b4-5172-4e28-bd29-1e0792947d18'

'794ded15-70c6-4bcd-a0bb-9b7ad530a01a'

'055caf97-1b4f-4730-9f5d-acc24b707b06'

'c5b731db-1b0a-43f6-bcf6-757667d9cdc6'

'e81c7467-0fc3-4866-b814-c973488361cd'

'7d312290-28c8-473c-a0ed-8e53749b6d6d'

'd52792f4-ba38-424d-8140-ada5b883f293'

'00000014-0000-0000-c000-000000000000'

'8e3a7976-45be-4e5c-a34f-3adb97828d00'

'e572cafa-59db-4a44-badf-c3747f054c60'

'9cdead84-a844-4324-93f2-b2e6bb768d07'

'0af06dc6-e4b5-4f28-818e-e78e62d137a5'

'9ea1ad79-fdb6-4f9a-8bc3-2b70f96e34c7'

'0d38933a-0bbd-41ca-9ebd-28c4b5ba7cb7'

'6b95a49c-1c54-4c1a-8cfa-838cfff6ebd7'

'4962773b-9cdb-44cf-a8bf-237846a00ab7'

'589d5083-6f11-4d30-a62a-a4b316a14abf'

'c495cfdc-814f-46a1-89f0-657921c9fbe0'

'16f9e0a0-ac78-4c2c-a55a-f3855317a63a'

'a4a365df-50f1-4397-bc59-1a1564b8bb9c'

'b8ed041c-aa91-418e-8f47-20c70abc2de1'
'a0551534-cfc9-4e1f-9a7a-65093b32bb38'
'0f698dd4-f011-4d23-a33e-b36416dcb1e6'
'11c174dc-1945-4a9a-a36b-c79a0f246b9b'
'1d78a85d-813d-46f0-b496-dd72f50a3ec0'
'fc03f97a-9db0-4627-a216-ec98ce54e018'
'e406a681-f3d4-42a8-90b6-c2b029497af1'
'fd225045-a727-45dc-8caa-77c8eb1b9521'
'579d9c9d-4c83-4efc-8124-7eba65ed3356'
'4f6778d8-5aef-43dc-a1ff-b073724b9495'
'123cd850-d9df-40bd-94d5-c9f07b7fa203'
'394866fc-eedb-4f01-8536-3ff84b16be2a'
'ba1ea022-5807-41d5-bbeb-292c7e1cf5f6'
'8511a82b-10b4-43fb-b3bc-e621d8bdb532'
'262044b1-e2ce-469f-a196-69ab7ada62d3'
'608f9929-9737-432e-860f-4e1c1821052f'
'fb9de05a-fecc-4642-b3ca-66b9d4434d4d'
'3edcf11f-df80-41b2-a5e4-7e213cca30d1'
'9cb77803-d937-493e-9a3b-4b49de3f5a74'
'499b84ac-1321-427f-aa17-267ca6975798'
'2d696fb1-e862-4e01-ade3-2cf07c587ffd'
'3bc3fbf6-023a-4d86-bd09-bac559ccc9cc'
'07e1e85b-cfc5-4c4e-b7a6-3b4e315bd4e7'
'48af08dc-f6d2-435f-b2a7-069abd99c086'
'9191c4da-09fe-49d9-a5f1-d41cbe92ad95'
'6201d19e-14fb-4472-a2d6-5634a5c97568'

'e420dc86-d66f-4069-a2d0-be2f937bd272'

'35ffadb3-7fc1-497e-b61b-381d28e744cc'

'7f0d9978-eb2a-4974-88bd-f22a3006fe17'

'80dbdb39-4f33-4799-8b6f-711b5e3e61b6'

'9e5d84af-8971-422f-968a-354cd675ae5b'

'c505e273-0ba0-47e7-a0bd-f48042b4524d'

'a25dbca8-4e60-48e5-80a2-0664fdb5c9b6'

'9cd0f7df-8b1a-4e54-8c0c-0ef3a51116f6'

'cdad765c-f191-43ba-b9f5-7aef392f811d'

'a5ce81bb-67c7-4043-952a-22004782adb5'

'315970ab-3fc6-4208-9312-4bd5d35d6bb7'

'1fcdfafe-959b-4b32-afff-84f850974e84'

'de926fbf-e23b-41f9-ae15-c943a9cfa630'

'3541691f-c69a-4464-a9c2-ec714b5c174e'

'8bdebf23-c0fe-4187-a378-717ad86f6a53'

'5be973a1-1c81-4878-9ab6-6b261ab404e6'

'e6c69915-bcc7-4335-b655-c62f949d691b'

'3af5a1e8-2459-45cb-8683-bcd6cccbcc13'

'870c4f2e-85b6-4d43-bdda-6ed9a579b725'

'2db117cf-0a64-4da6-9a6d-243f08d09528'

'3e235f68-22ba-48f2-baf7-34bc3d1c6ad2'

'98785600-1bb7-4fb9-b9fa-19afe2c8a360'

'3072002f-3e97-4979-91f2-09fe40da755d'

'c9455e78-f730-4ba3-a551-cb6a1c389089'

'51667a77-8210-49d8-b82f-8db910137ae2'

'e0ccf59d-5a20-4a87-a122-f42842cdb86a'

'161a339d-b9f5-41c5-8856-6a6669acac64'

'5b20c633-9a48-4a5f-95f6-dae91879051f'

'8b5a06b3-efd5-4511-b112-5caf390d1b92'

'270d0f73-6132-43b9-9666-9f5e473b7e1f'

'dbcbdd02a-d7c4-42fb-8c27-b07e5118b848'

'8f41dc7c-542c-4bdd-8eb3-e60543f607ca'

'd22ea4d1-2678-4a7b-aa5e-f340c2a7d993'

'93efed00-6552-4119-833a-422b297199f9'

'1690c5aa-925a-4d0e-836b-722c795bd0d0'

'8cae6e77-e04e-42ce-b5cb-50d82bce26b1'

'6dae42f8-4368-4678-94ff-3960e28e3630'

'324395f8-e27f-48a8-8b0b-6338c76ce949'

'f0ae4899-d877-4d3c-ae25-679e38eea492'

'b8340c3b-9267-498f-b21a-15d5547fd85e'

'09cf45fc-b312-44eb-a287-e6582ea20ae0'

'6ea8091b-151d-447a-9013-6845b83ba57b'

'e9b154d0-7658-433b-bb25-6b8e0a8a7c59'

'd5f39c4e-4b59-4b22-88cc-a70c399ba1cd'

'03b39d0f-4213-4864-a245-b1476ec03169'

'd3244f1e-56a7-4819-80e9-a30a7a83dde8'

'18a4ad1e-427c-4cad-8416-ef674e801d32'

'bf4fa6bf-d24c-4d1c-8cfd-12063dd646b2'

'a3747411-ce7c-4888-9ddc-3a230786ca19'

'0000000c-0000-0000-c000-000000000000'

'cbfda01c-c883-45aa-aedc-e7a484615620'

'ce6ff14a-7fdc-4685-bbe0-f6afdfcfa8e0'

'b28ec8e1-950e-4bd0-b3d0-c1e93074b88b'

'78e7bc61-0fab-4d35-8387-09a8d2f5a59d'

'9199bf20-a13f-4107-85dc-02114787ef48'

'66c6d0d1-f2e7-4a18-97a9-ed10f3347016'

'17724442-aa9a-46cc-bf09-c47bb1a98518'

'3eb95cef-b10f-46fe-94e0-969a3d4c9292'

'0a5f63c0-b750-4f38-a71c-4fc0d58b89e2'

'443155a6-77f3-45e3-882b-22b3a8d431fb'

'ef5d5c69-a5df-46bb-acaf-426f161a21a2'

'2746ea77-4702-4b45-80ca-3c97e680e8b7'

'6bb8e274-af5d-4df2-98a3-4fd78b4cafd9'

'22b71bf7-6952-44c8-8b92-893ca3b117d2'

'2dc3760b-4713-48b1-a383-1dfe3e449ec2'

'b3d7adc8-6b92-4061-b47d-fc1419774ccc'

'75e725bf-66ce-4cea-9b9a-5c4caae57f33'

'c7bb12bf-0b39-4f7f-9171-f418ff39b76a'

'3cbcded7-0049-4401-9e00-5f4f10f75efe'

'b998f6f8-79d0-4b6a-8c25-5791dbe49ad0'

'737d58c1-397a-46e7-9d12-7d8c830883c2'

'26a4ae64-5862-427f-a9b0-044e62572a4f'

'ce48853e-0605-4f77-8746-d70ac63cc6bc'

'8338dec2-e1b3-48f7-8438-20c30a534458'

'5bfdc2bf-507b-4413-8f15-f1fcf1c15475'

'022907d3-0f1b-48f7-badc-1ba6abab6d66'

'5b306cba-9c71-49db-96c3-d17ca2379c4d'

'2e458d69-0892-4655-b713-4f7b182315dd'

'a85cf173-4192-42f8-81fa-777a763e6e2c'

'b61cc489-e138-4a69-8bf3-c2c5855c8784'

'959ddb6c-47bf-4db5-bb68-6d77b88a1250'

'990ba54a-28b0-4603-9345-b23f426585c4'

'ac9dc5fe-b644-4832-9d03-d9f1ab70c5f7'

'0469d4cd-df37-4d93-8a61-f8c75b809164'

'557c67cf-c916-4293-8373-d584996f60ae'

'8ae6a0b1-a07f-4ec9-927a-afb8d39da81c'

'a77d91dc-971b-4cf7-90c8-f183194249bc'

'372140e0-b3b7-4226-8ef9-d57986796201'

'b2cc270f-563e-4d8a-af47-f00963a71dcd'

'c728155f-7b2a-4502-a08b-b8af9b269319'

'87223343-80b1-4097-be13-2332ffa1d666'

'bb55177b-a7d9-4939-a257-8ab53a3b2bc6'

'5b712e99-51a3-41ce-86ff-046e0081c5c0'

'df09ff61-2178-45d8-888c-4210c1c7b0b2'

'b4ca0290-4e73-4e31-ade0-c82ecfaabf6a'

'00000004-0000-0ff1-ce00-000000000000'

'd3d1a4fe-edb2-4b09-bc39-e41d342323d6'

'6608bce8-e060-4e82-bfd2-67ed4f60262f'

'4c4f550b-42b2-4a16-93f9-fdb9e01bb6ed'

'05a65629-4c1b-48c1-a78b-804c4abdd4af'

'57084ef3-d413-4087-a28f-f6f3b1ad7786'

'884e625b-d232-4ca3-88ae-fd138b5084ac'

'cb082ace-8a09-486a-9d02-c15739683b01'

'3734c1a4-2bed-4998-a37a-ff1a9e7bf019'

'00000007-0000-0ff1-ce00-000000000000'

'65d91a3d-ab74-42e6-8a2f-0add61688c74'

'9581bc0e-c952-4fd3-8d99-e777877718b1'

'509e4652-da8d-478d-a730-e9d4a1996ca4'

'f738ef14-47dc-4564-b53b-45069484ccc7'

'0a0a29f9-0a25-49c7-94bf-c53c3f8fa69d'

'ba9ff945-a723-4ab5-a977-bd8c9044fe61'

'c321abdf-8dc2-4906-a0c7-2daaf7af298c'

'58ea322b-940c-4d98-affb-345ec4cccb92'

'60fff137-909f-4cdc-8e13-3e11f583788a'

'12fb057d-b751-47cd-857c-f2934bb677b4'

'cb4dc29f-0bf4-402a-8b30-7511498ed654'

'0736f41a-0425-4b46-bdb5-1563eff02385'

'a1b76039-a76c-499f-a2dd-846b4cc32627'

'8edd93e1-2103-40b4-bd70-6e34e586362d'

'461e8683-5575-4561-ac7f-899cc907d62a'

'ba4bc2bd-843f-4d61-9d33-199178eae34e'

'faf1f7e1-8522-498c-a7f9-6a7f1a17f873'

'dbc36ae1-c097-4df9-8d94-343c3d091a76'

'3c31d730-a768-4286-a972-43e9b83601cd'

'29820072-374d-49b8-945a-3941d7e9b468'

'b5615b95-29f3-4ea5-bce3-50a07b952601'

'5b534afd-fdc0-4b38-a77f-af25442e3149'

'7865c1d2-f040-46cc-875f-831a1ef6a28a'

'9d777fa9-b417-43b8-8991-12f8ee2161d2'

'3b990c8b-9607-4c2a-8b04-1d41985facca'

'd2a0a418-0aac-4541-82b2-b3142c89da77'

'3274406e-4e0a-4852-ba4f-d7226630abb7'

'120d688d-1518-4cf7-bd38-182f158850b6'

'c5393580-f805-4401-95e8-94b7a6ef2fc2'

'0130cc9f-7ac5-4026-bd5f-80a08a54e6d9'

'c4b6c9c7-e308-48f1-83b3-04076f680ea0'

'58ef1dbd-684c-47d6-8ffc-61ea7a197b95'

'525036f0-a5e3-4a2b-bba7-440d8d71647f'

'a0be0c72-870e-46f0-9c49-c98333a996f7'

'7bf610f7-ecaf-43a2-9dbc-33b14314d6fe'

'9edfcdd9-0bc5-4bd4-b287-c3afc716aac7'

'8e0e8db5-b713-4e91-98e6-470fed0aa4c2'

'3030ef2a-57cd-4c9b-9490-badf0d0cf3ff'

'9d06afd9-66c9-49a6-b385-ea7509332b0b'

'd8c767ef-3e9a-48c4-aef9-562696539b39'

'6f82282e-0070-4e78-bc23-e6320c5fa7de'

'5d661950-3475-41cd-a2c3-d671a3162bc1'

'bf26f092-3426-4a99-abfb-97dad74e661a'

'0140a36d-95e1-4df5-918c-ca7ccd1fafc9'

'9f0e9e2d-6525-4eb8-b2e0-da2314d017b5'

'b9a92e36-2cf8-4f4e-bcb3-9d99e00e14ab'

'959678cf-d004-4c22-82a6-d2ce549a58b8'

'00000009-0000-0000-c000-000000000000'

'cc5476ec-3074-44d1-8461-711f5d9b0e39'

'7cd01e22-2632-4b54-ad19-a260969ba7ff'

'c4288165-6698-45ba-98a5-48ea7791fed3'

'c066d759-24ae-40e7-a56f-027002b5d3e4'

'f1dd0a37-89c6-4e07-bcd1-ffd3d43d8875'

'00000001-0000-0000-c000-000000000000'

'a23667b4-c8cf-42c1-86d6-25a4f46525f8'

'64a7b174-5779-4506-b54c-fbb0d80f1c9b'

'74cb6831-0dbb-4be1-8206-fd4df301cdc2'

'12743ff8-d3de-49d0-a4ce-6c91a4245ea0'

'd87dcbc6-a371-462e-88e3-28ad15ec4e64'

'a48bbb4a-8f73-478e-b492-5e1f05694d54'

'2bceb21e-ead7-46f2-a187-10ab2e65f34e'

'f7069a8d-9edc-4300-b365-ae53c9627fc4'

'd6fdaa33-e821-4211-83d0-cf74736489e1'

'9c8b80bc-6887-42d0-b1af-d0c40f9bf1fa'

'eace8149-b661-472f-b40d-939f89085bd4'

'aafac111-b9f0-439b-bbe3-1839c672ac8e'

'cf6163c2-70e2-4a66-ba08-0d46eb68dcf9'

'c115998b-3d59-49b4-b55b-042a9ba1dbfe'

'ca0a114d-6fbc-46b3-90fa-2ec954794ddb'

'0c7668b5-3260-4ad0-9f53-34ed54fa19b2'

'29f411f1-b2cf-4043-8ac8-2185d7316811'

'c6e44401-4d0a-4542-ab22-ecd4c90d28d7'

'5741a1ff-751d-4ad7-bcd1-dfe3c998fd11'

'b7912db9-aa33-4820-9d4f-709830fdd78f'

'2cfc91a4-7baa-4a8f-a6c9-5f3d279060b8'

'9469b9f5-6722-4481-a2b2-14ed560b706f'

'1dcb1bc7-c721-498e-b2fa-bcddcea44171'

'54b9b9be-c365-4548-95c6-d2f2011f48f4'

'ddc728e9-153d-4032-ab80-80e57af7a56f'

'58c746b0-a0b0-4647-a8f6-12dde5981638'

'19947cfd-0303-466c-ac3c-fcc19a7a1570'

'abcdbe4b-4a26-4340-a0bd-04aed964e8af'

'7de4d5c5-5b32-4235-b8a9-33b34d6bcd2a'

'ce79fdc4-cd1d-4ea5-8139-e74d7dbe0bb7'

'257601fd-462f-4a21-b623-7f719f0f90f4'

'8ee8fdad-f234-4243-8f3b-15c294843740'

'3480ae63-e100-432c-92c8-4aee777c5671'

'd2a590e7-6906-4a45-8f41-cecfdca9bca1'

'a4c95b9e-3994-40cc-8953-5dc66d48348d'

'707be275-6b9d-4ee7-88f9-c0c2bd646e0f'

'e935b4a5-8968-416d-8414-caed51c782a9'

'ec245c98-4a90-40c2-955a-88b727d97151'

'608f6f31-fed0-4f7b-809f-90f6c9b3de78'

'6d057c82-a784-47ae-8d12-ca7b38cf06b4'

'e8de9221-a19c-4c81-b814-fd37c6caf9d2'

'5e99b34c-2ea8-49d4-b205-1c8bb56114d8'

'9ae330ab-d710-466b-851c-c828e7340846'

'aae1f723-e3f6-4223-a7cd-0a590e2247bc'

'40775b29-2688-46b6-a3b5-b256bd04df9f'

'ccf4d8df-75ce-4107-8ea5-7afd618d4d8a'

'01cb2876-7ebd-4aa4-9cc9-d28bd4d359a9'

'c39c9bac-9d1f-4dfb-aa29-27f6365e5cb7'

'a4bad4aa-bf02-4631-9f78-a64ffdba8150'

'6a0a243c-0886-468a-a4c2-eff52c7445da'
'ecad3f28-c75d-4414-94e0-a5e1de4df79e'
'42bb5b6c-1300-4c06-b3ef-bdadc9571ec9'
'd9327919-6775-4843-9037-3fb0fb0473cb'
'9bd5ab7f-4031-4045-ace9-6bebbad202f6'
'63c2c773-89fe-4164-a02f-b8c7fc1772ae'
'1e3e4475-288f-4018-a376-df66fd7fac5f'
'e5408ad0-c4e2-43aa-b6f2-3b4951286d99'
'a8399344-a951-4537-8cf2-444aede33ff2'
'd8877f27-09c0-43aa-8113-40151dae8b14'
'e526e72f-ffae-44a0-8dac-cf14b8bd40e2'
'35274a02-4c3a-4878-9c89-7b9f1b09644f'
'475226c6-020e-4fb2-8a90-7a972cbfc1d4'
'03db181c-e9d3-4868-9097-f0b728327182'
'5a543d7c-9c4a-4f90-8cc7-6ae082a5b65b'
'5b81a823-5f67-4fb3-8d0f-4c92b5044fe4'
'157638eb-a5cb-4c10-af42-2d6759eb1871' nan]

4. Missing Values Count:

| | |
|-----------------------|-----|
| id | 0 |
| displayName | 488 |
| appld | 488 |
| createdDateTime | 488 |
| applicationType | 493 |
| accountEnabled | 488 |
| applicationVisibility | 488 |

assignmentRequired 488

isAppProxy 488

dtype: int64

5. Application Type Counts:

Microsoft Application 497

Enterprise Application 46

Managed Identity 30

Name: applicationType, dtype: int64

6. Average Display Name Length: 24.68 characters

7. Earliest Created Date: 2022-03-04 02:35:55+00:00

8. Latest Created Date: 2022-05-04 08:34:59+00:00

9. Number of Account-Enabled Applications: 576

10. Number of App Proxy Applications: 0

11. Number of Visible Applications: 576

12. Total Count of Applications: 1066

13. Application Creation by Year:

2022.0 578

Name: createdYear, dtype: int64

14. Application Creation by Month:

2022-04 465

2022-03 69

2022-05 44

Name: createdMonth, dtype: int64

15. Display Name with Maximum Length: Defender for Containers provisioning Azure Policy Addon for Kub

16. Display Name with Minimum Length: 5

17. Total Number of Unique Apps: 578

18. Total Number of Enterprise Applications: 46

19. Total Number of Non-Enterprise Applications: 1020

20. Application Visibility Counts:

Visible 576

Hidden 2

Name: applicationVisibility, dtype: int64

<ipython-input-5-5007f1804046>:11: FutureWarning: Treating datetime data as categorical rather than numeric in `.describe` is deprecated and will be removed in a future version of pandas. Specify `datetime_is_numeric=True` to silence this warning and adopt the future behavior now.

```
basic_stats = data.describe()
```

Now make a script to install everything and more

ASSISTANT

It seems you've provided a substantial amount of data related to applications, their IDs, and various

attributes. How can I assist you with this data? Please let me know what specific information or tasks you'd like me to perform using this data.

USER

What can you do with it? Tell me the could possibly do that I have no idea about.

ASSISTANT

Microsoft Graph is a powerful resource that offers a wide range of capabilities for developers, businesses, and users. Here are some less common, but interesting things you can do with Microsoft Graph:

1. **Data Analysis and Insights**: You can use Microsoft Graph to analyze data from various Microsoft 365 services. For example, you can identify trends in email communication, track user activities, and gain insights into how your organization uses Microsoft 365 services.
2. **Automated Workflows**: Create custom workflows by combining data from different services. For instance, you can trigger actions in response to specific events, such as sending an email when a new SharePoint document is added or updating a task in Planner based on changes in an Excel file.
3. **Security and Compliance**: Microsoft Graph provides access to security and compliance data, allowing you to build custom solutions for monitoring and responding to security threats, setting up alerts, and ensuring compliance with data protection regulations.
4. **Custom Reporting**: Develop custom reporting tools to generate detailed reports on user activities, application usage, and resource consumption. This can be useful for auditing, cost

analysis, and optimizing resource allocation.

5. ****AI and Machine Learning Integration****: You can integrate AI and machine learning models with Microsoft Graph to create intelligent applications. For example, you can build a chatbot that uses natural language processing to answer questions based on data from Microsoft 365 services.

6. ****Inclusive and Accessibility Apps****: Use Microsoft Graph data to build applications that promote inclusivity and accessibility. For instance, you can create apps that provide tailored content or assistance to users with specific accessibility needs.

7. ****Integrate with Third-Party Services****: Microsoft Graph also allows you to integrate with third-party services. You can connect your Microsoft 365 data with other cloud services, enhancing the functionality and reach of your applications.

8. ****Custom User Experiences****: Develop custom user interfaces and experiences by fetching data from Microsoft Graph. This can be useful for creating personalized dashboards or views tailored to specific user roles or departments.

9. ****Voice and Conversational Interfaces****: Integrate Microsoft Graph with voice recognition and conversational AI platforms to create voice-activated applications or chatbots that can access and manipulate Microsoft 365 data.

10. ****Resource Optimization****: Use Microsoft Graph to optimize resource usage. For example, you can build applications that analyze resource utilization data and suggest cost-saving measures or better allocation of resources.

11. **Educational Apps**: If you're in the education sector, you can use Microsoft Graph to develop custom applications for managing students, classes, and resources. This can include features like grade tracking, attendance monitoring, and communication tools.
12. **IoT Integration**: Connect Internet of Things (IoT) devices to Microsoft Graph to collect and analyze data from sensors and devices in real-time, enabling applications for predictive maintenance, monitoring, and control.
13. **Interactive Data Visualizations**: Create interactive data visualizations and dashboards that pull data from Microsoft 365 services. This can help users better understand and explore their data.
14. **Knowledge Management**: Build knowledge management systems that utilize Microsoft Graph to organize and retrieve information efficiently. This can be particularly valuable for companies that need to manage large amounts of data and documents.
15. **Virtual Reality (VR) and Augmented Reality (AR)**: Explore opportunities to integrate Microsoft Graph data into VR and AR applications, enhancing immersive experiences and providing users with contextual information.

These are just a few examples of the innovative possibilities offered by Microsoft Graph. Its versatility and the wide array of data sources it connects make it a valuable tool for developers and organizations looking to create custom solutions that leverage Microsoft 365 data in creative and efficient ways.

USER

Make a script that loads and installs every fucking thing

1. Basic Statistics:

| | id | displayName \ |
|--------|--------------------------------------|----------------------------|
| count | 1066 | 578 |
| unique | 1066 | 571 |
| top | 00a67483-ea0d-4bfa-a19b-17ae35e97fe3 | Domain Controller Services |
| freq | 1 | 3 |
| first | NaN | NaN |
| last | NaN | NaN |

| | appld | createdDateTime \ |
|--------|--------------------------------------|---------------------------|
| count | 578 | 578 |
| unique | 578 | 298 |
| top | 3f18b202-a866-4c1f-b9c0-6598390a2092 | 2022-03-28 03:48:59+00:00 |
| freq | 1 | 31 |
| first | NaN | 2022-03-04 02:35:55+00:00 |
| last | NaN | 2022-05-04 08:34:59+00:00 |

| | applicationType | accountEnabled | applicationVisibility \ |
|--------|-----------------------|----------------|-------------------------|
| count | 573 | 578 | 578 |
| unique | 3 | 2 | 2 |
| top | Microsoft Application | True | Visible |
| freq | 497 | 576 | 576 |
| first | NaN | NaN | NaN |
| last | NaN | NaN | NaN |

assignmentRequired isAppProxy

| | | |
|--------|-------|-------|
| count | 578 | 578 |
| unique | 2 | 1 |
| top | False | False |
| freq | 535 | 578 |
| first | NaN | NaN |
| last | NaN | NaN |

2. Unique Application Types:

['Enterprise Application' nan 'Microsoft Application' 'Managed Identity']

3. Unique Application IDs:

['3f18b202-a866-4c1f-b9c0-6598390a2092'
'fd681d32-9e97-4876-a58f-05269cf838e4'
'2e8265ff-5066-4e5d-ae61-dbcc32816115'
'98db8bd6-0cc0-4e67-9de5-f187f1cd1b41'
'0947a342-ab4a-43be-93b3-b8243fc161e5'
'607ece82-f922-494f-88b8-30effaf12214'
'6db4d6bb-6649-4dc2-84b7-0b5c6894031e'
'b692184e-b47f-4706-b352-84b288d2d9ee'
'354b5b6d-abd6-4736-9f51-1be80049b91f'
'2ff814a6-3304-4ab8-85cb-cd0e6f879c1d'
'18a66f5f-dbdf-4c17-9dd7-1634712a9cbe'
'b503eb83-1222-4dcc-b116-b98ed5216e05'
'f5c26e74-f226-4ae8-85f0-b4af0080ac9e'
'bb2a2e3a-c5e7-4f0a-88e0-8e01fd3fc1f4']

'584a29b4-7876-4445-921e-71e427d4f4b3'

'd3e1650d-38f0-4a3c-8c2d-a064cc8d9893'

'f3c21649-0979-4721-ac85-b0216b2cf413'

'bc59ab01-8403-45c6-8796-ac3ef710b3e3'

'a303894e-f1d8-4a37-bf10-67aa654a0596'

'6f2d169c-08f3-4a4c-a982-bcaf2d038c45'

'448adbda-b8d8-4f33-a1b0-ac58cf44d4c1'

'60e6cd67-9c8c-4951-9b3c-23c25a2169af'

'799f1985-1517-4fe1-af2b-ba3d87d4996b'

'1448fd13-7e74-41f4-b6e3-17e485d8ac2e'

'5e5e43d4-54da-4211-86a4-c6e7f3715801'

'0cd79364-7a90-4354-9984-6e36c841418d'

'1f5530b3-261a-47a9-b357-ded261e17918'

'621269cf-1195-44a3-a835-c613d103dd15'

'00000002-0000-0ff1-ce00-000000000000'

'996def3d-b36c-4153-8607-a6fd3c01b89f'

'00000003-0000-0ff1-ce00-000000000000'

'22d7579f-06c2-4baa-89d2-e844486adb9d'

'c5b17a4f-cc6f-4649-9480-684280a2af3a'

'73c2949e-da2d-457a-9607-fcc665198967'

'67e3df25-268a-4324-a550-0de1c7f97287'

'd73f4b35-55c9-48c7-8b10-651f6f2acb2e'

'eb070ea5-bd17-41f1-ad68-5851f6e71774'

'4ac7d521-0382-477b-b0f8-7e1d95f85ca2'

'9ddb5749-5660-445d-b859-394225a7d97d'

'3e8f1ed5-3a6e-47d1-ad1e-b14e6b69704b'

'a6aa9161-5291-40bb-8c5c-923b567bee3b'
'56c1da01-2129-48f7-9355-af6d59d42766'
'ab27a73e-a3ba-4e43-8360-8bcc717114d8'
'62c559cd-db0c-4da0-bab2-972528c65d42'
'50e95039-b200-4007-bc97-8d5790743a63'
'39495caf-cc21-4d03-b6b0-8c4a973cf213'
'05cf5e27-931d-47ad-826d-cb9028d8bd7a'
'dfbed8b2-492a-414e-b2f0-482534e87bc5'
'810dcf14-1858-4bf2-8134-4c369fa3235b'
'328fd23b-de6e-462c-9433-e207470a5727'
'f3723d34-6ff5-4ceb-a148-d99dcd2511fc'
'66244124-575c-4284-92bc-fdd00e669cea'
'd9794ed7-bfec-42d8-ba8d-4d7df2a1f6dc'
'00000012-0000-0000-c000-000000000000'
'9bdab391-7bbe-42e8-8132-e4491dc29cc0'
'aa9ecb1e-fd53-4aaa-a8fe-7a54de2c1334'
'e933bd07-d2ee-4f1d-933c-3752b819567b'
'5bd9995f-b6f8-4c7a-a024-e8c5eab9c85d'
'f3625a3e-6360-4580-968d-fae4cab75a0'
'80a10ef9-8168-493d-abf9-3297c4ef6e3c'
'2cf9eb86-36b5-49dc-86ae-9a63135dfa8c'
'bc313c14-388c-4e7d-a58e-70017303ee3b'
'e5d38734-8683-4c09-9e0d-73a883c2d83d'
'20e940b3-4c77-4b0b-9a53-9e16a1b010a7'
'562db366-1b96-45d2-aa4a-f2148cef2240'
'7e3bc4fd-85a3-4192-b177-5b8bfc87f42c'

'2368d027-f996-4edb-bf48-928f98f2ab8c'
'f6ec9458-aa91-4c6e-bb3f-7b7c946c1b44'
'14452459-6fa6-4ec0-bc50-1528a1a06bf0'
'c61423b7-1d1f-430d-b444-0eee53298103'
'3184af01-7a88-49e0-8b55-8ecdce0aa950'
'163b648b-025e-455b-9937-a7f39a65d171'
'b4bddae8-ab25-483e-8670-df09b9f1d0ea'
'e9f49c6b-5ce5-44c8-925d-015017e9f7ad'
'8602e328-9b72-4f2d-a4ae-1387d013a2b3'
'c161e42e-d4df-4a3d-9b42-e7a3c31f59d4'
'abfa0a7c-a6b6-4736-8310-5855508787cd'
'fbc197b7-9e9c-4f98-823f-93cb1cb554e6'
'8b3391f4-af01-4ee8-b4ea-9871b2499735'
'89d10474-74af-4874-99a7-c23c2f643083'
'7c33bfcb-8d33-48d6-8e60-dc6404003489'
'2f3f02c9-5679-4a5c-a605-0de55b07d135'
'65b0413c-6b56-4536-a2ea-1929dfe5e81c'
'6d49f6be-5347-432d-b9b4-64ac938c1bd2'
'f18474f2-a66a-4bb0-a3c9-9b8d892092fa'
'f2c304cf-8e7e-4c3f-8164-16299ad9d272'
'359431ad-ece5-496b-8768-be4bbfd82f36'
'709110f7-976e-4284-8851-b537e9bcb187'
'e4ab13ed-33cb-41b4-9140-6e264582cf85'
'0bf30f3b-4a52-48df-9a82-234910c4a086'
'2cc451ba-a8ec-496f-bdff-591f5ae2876c'
'0ecb6dbc-7807-4951-9a69-b5d3dfa5a0b5'

'79d7fb34-4bef-4417-8184-ff713af7a679'

'5d13f7d7-0567-429c-9880-320e9555e5fc'

'0b07f429-9f4b-4714-9392-cc5e8e80c8b0'

'48ef7923-268f-473d-bcf1-07f0997961f4'

'74bcdadc-2fdc-4bb3-8459-76d06952a0e9'

'81473081-50b9-469a-b9d8-303109583ecb'

'f8f7a2aa-e116-4ba6-8aea-ca162cfa310d'

'2793995e-0a7d-40d7-bd35-6968ba142197'

'982bda36-4632-4165-a46a-9863b1bbcf7d'

'0000000a-0000-0000-c000-000000000000'

'a3f23ff7-be0d-49a0-b535-f5fb0f5f8039'

'e158b4a5-21ab-442e-ae73-2e19f4e7d763'

'f9c691e6-93b3-4d57-944c-afcc737f9abf'

'632ec9eb-fad7-4cbd-993a-e72973ba2acc'

'95c8b18b-3ed0-4299-88e6-31a12c4b4092'

'd96199e7-4674-4bbf-a1c6-ddf93682f5ee'

'e64bd61e-5424-451f-b666-e02ee2878437'

'a4c1cdb3-88ab-4d13-bc99-1c46106f0727'

'86adf623-eea3-4453-9f4a-18134ac1410d'

'63e61dc2-f593-4a6f-92b9-92e4d2c03d4f'

'6f0478d5-61a3-4897-a2f2-de09a5a90c7f'

'4c8f074c-e32b-4ba7-b072-0f39d71daf51'

'c44b4083-3bb0-49c1-b47d-974e53cbdf3c'

'a8b6bf88-1d1a-4626-b040-9a729ea93c65'

'441509e5-a165-4363-8ee7-bcf0b7d26739'

'b8066b99-6e67-41be-abfa-75db1a2c8809'

'b8cf62f3-7cc7-4e32-ab3a-41370ef0cfcf'

'00000002-0000-0000-c000-000000000000'

'01fc33a7-78ba-4d2f-a4b7-768e336e890e'

'fc75330b-179d-49af-87dd-3b1acf6827fa'

'0c6620df-7b29-44de-8ba4-688a56a20f9f'

'8bbf8725-b3ca-4468-a217-7c8da873186e'

'8d5683a1-43b9-4f38-8906-504eae3c36c9'

'e6650347-047f-4e51-9386-839384472ea5'

'74b0b707-6e48-48c5-a6f8-6ad47db27678'

'de742ffc-b441-4542-8646-7e805426b824'

'4392ab71-2ce2-4b0d-8770-b352745c73f5'

'4fa46669-56c9-44e7-a69b-182480b952a8'

'5da7367f-09c8-493e-8fd4-638089cddec3'

'a3dfc3c6-2c7d-4f42-aeec-b2877f9bce97'

'374b2a64-3b6b-436b-934c-b820eacca870'

'6bccf540-eb86-4037-af03-7fa058c2db75'

'6e110594-2adf-49c8-8d37-41eee81c0359'

'30b7f32a-8115-4e12-883b-1fd99dd0efd9'

'fc68d9e5-1f76-45ef-99aa-214805418498'

'f18b59c9-5926-4a65-8605-c23ec8c7e074'

'601d4e27-7bb3-4dee-8199-90d47d527e1c'

'52aa7af2-7031-4163-a511-b1ca35c43c6e'

'd2fa1650-4805-4a83-bcb9-cf41fe63539c'

'bd93b475-f9e2-476e-963d-b2daf143ffb9'

'27a762be-14e7-4f92-899c-151877d6d497'

'bf8eb16c-7ba7-4b47-86be-ac5e4b2007a5'

'9ed4cd8c-9a98-405f-966b-38ab1b0c24a3'

'd3315f6c-968a-40bb-94d2-a6a9503b05f5'

'dad37da6-229d-4bc0-8b94-fee8600589db'

'f49e6313-6c53-4ec8-918f-2dd9f1980342'

'4bfd5d66-9285-44a1-bb14-14953e8cdf5e'

'e3583ad2-c781-4224-9b91-ad15a8179ba0'

'f77c2a8f-8a0a-4776-8e0a-bcb2549610ca'

'ec52d13d-2e85-410e-a89a-8c79fb6a32ac'

'f6b60513-f290-450e-a2f3-9930de61c5e7'

'5e5abe2b-83cd-4786-826a-a05653ebb103'

'18af356b-c4fd-4f52-9899-d09d21397ab7'

'7b7531ad-5926-4f2d-8a1d-38495ad33e17'

'00000003-0000-0000-c000-000000000000'

'5e89db78-b7c2-42c9-91f3-12ee3bc42550'

'bf283ae6-5efd-44a8-b56a-2a7939982d60'

'a967240f-810b-4f79-85e5-25870cc69cbb'

'f69cfde7-4247-48a2-a54d-1501570d8233'

'e18cedde-9458-482f-9dd1-558c597ac42e'

'bf9fc203-c1ff-4fd4-878b-323642e462ec'

'765fe668-04e7-42ba-aec0-2c96f1d8b652'

'37182072-3c9c-4f6a-a4b3-b3f91cacffce'

'3b5c2f38-d6e7-4b96-944c-872d4fe64fe6'

'728a93e3-065d-4678-93b1-3cc281223341'

'1ac05c7e-12d2-4605-bf9d-549d7041c6b3'

'91ff567f-bb4f-4719-91d7-d983057bc0d6'

'8fca0a66-c008-4564-a876-ab3ae0fd5cff'

'96231a05-34ce-4eb4-aa6a-70759cbb5e83'
'0ac71e58-8aec-485e-a4ec-6b24af1b962a'
'36e2398c-9dd3-4f29-9a72-d9f2cfc47ad9'
'7a08d6a2-494e-48eb-85cd-8c245104a6fd'
'894b1496-c6e0-4001-b69c-81b327564ca4'
'cf32a0cc-373c-47c9-9156-0db11f6a6dfc'
'c6932912-e55b-494c-b080-de9a35ce410c'
'e00d2f8a-f6c8-46e4-b379-e66082e28ca8'
'e1b4ed32-0aea-4286-88bf-5df97be96abc'
'a15bc1de-f777-408f-9d2b-a27ed19c72ba'
'db559737-c65c-4c91-b5d2-c51a55c9e22b'
'366cbfa5-46b3-47fb-9d70-55fb923b4833'
'3b2fa68d-a091-48c9-95be-88d572e08fb7'
'9c012f45-4a09-4f90-9cb9-5dea81f53a9e'
'5613cb5c-a7c9-4099-8034-511fd7616cb2'
'3aa5c166-136f-40eb-9066-33ac63099211'
'0000dab9-8b21-4ba2-807f-1743968cef00'
'57c0fc58-a83a-41d0-8ae9-08952659bdfd'
'7c99d979-3b9c-4342-97dd-3239678fb300'
'87df0fbf-e22d-4d7c-bc30-f59ca7460837'
'56e5be3f-8703-4fe3-936b-8599b606b8fc'
'ae11f5fb-c627-4eec-b4a0-f7b5969426e5'
'80369ed6-5f11-4dd9-bef3-692475845e77'
'b121a960-2b39-48d2-a3ed-9efe1b7f8992'
'e3335adb-5ca0-40dc-b8d3-bedc094e523b'
'3ae55346-4fae-430e-96f8-e26e8315f27e'

'71af9eaf-3a7b-43eb-bc59-c504bfe12240'
'e7d5afaf-5e93-4aad-b546-878812ff572c'
'f71766dc-90d9-4b7d-bd9d-4499c4331c3f'
'91f2f07a-57ac-44c9-89e6-c9e872cb470e'
'24d3987b-be4a-48e0-a3e7-11c186f39e41'
'00a571e7-e41d-4ed4-8eef-36ee2632264d'
'c9e0b461-3515-4a03-b576-ed91ed4336d'
'9e09aefc-b2e5-4d19-9f74-3e3e8b11a57b'
'1412d89f-b8a8-4111-b4fd-e82905cbd85d'
'af50c6e1-9fa7-4512-8e63-33d5541d2b9d'
'92b61450-2139-4e4a-a0cc-898eced7a779'
'abba844e-bc0e-44b0-947a-dc74e5d09022'
'1322e676-dee7-41ee-a874-ac923822781c'
'ce34e7e5-485f-4d76-964f-b3d2b16d1e4f'
'76cd24bf-a9fc-4344-b1dc-908275de6d6d'
'070fc472-7cef-4d53-9b65-34464c4d5f4a'
'925eb0d0-da50-4604-a19f-bd8de9147958'
'ce52bcbe-f669-4cfd-9866-d64472d3e13e'
'a0df4d2c-64a4-4e93-8f8e-4c8d721b5b77'
'de096ee1-dae7-4ee1-8dd5-d88ccc473815'
'd10de03d-5ba3-497a-90e6-7ff8c9736059'
'035f9e1d-4f00-4419-bf50-bf2d87eb4878'
'396e7f4b-41ea-4851-b04d-65de6cf1b4a3'
'6b3368c6-61d2-4a72-854c-42d1c4e71fed'
'1a14be2a-e903-4cec-99cf-b2e209259a0f'
'a0e1e353-1a3e-42cf-a8ea-3a9746eec58c'

'bd9b7cd5-dac1-495f-b013-ac871e98fa5f'
'e3bfd6ac-eace-4438-9dc1-eed439e738de'
'a5472e16-e1d2-4bbe-81b3-ecdcd459b536'
'880da380-985e-4198-81b9-e05b1cc53158'
'5b404cf4-a79d-4cfe-b866-24bf8e1a4921'
'637fcc9f-4a9b-4aaa-8713-a2a3cfda1505'
'ea2f600a-4980-45b7-89bf-d34da487bda1'
'0eb690b7-d23e-4fb0-b43e-cd161ac80cc3'
'd4ebce55-015a-49b5-a083-c84d1797ae8c'
'c699bf69-fb1d-4eaf-999b-99e6b2ae4d85'
'63ea3c01-7483-456e-8073-d3fed34fbdda'
'cfa8b339-82a2-471a-a3c9-0fc0be7a4093'
'918d0db8-4a38-4938-93c1-9313bdfc0272'
'a12e8ccb-0fcd-46f8-b6a1-b9df7a9d7231'
'5ed8fe41-c1bc-4c06-a531-d91e1f1c2fac'
'fc780465-2017-40d4-a0c5-307022471b92'
'c19490b5-c092-426f-b1a2-674b279d4975'
'823c0a78-5de0-4445-a7f5-c2f42d7dc89b'
'1b912ec3-a9dd-4c4d-a53e-76aa7adb28d7'
'a466b239-e85b-439c-a3b5-f0ee6ce3b7b5'
'02069796-fdae-431f-832d-06f03bd0ccf6'
'cc46c2aa-d508-409b-aeb7-df7cd1e07aaa'
'76c92352-c057-4cc2-9b1e-f34c32bc58bd'
'847bd4c1-7486-4241-a783-f9bda69241c1'
'319f651f-7ddb-4fc6-9857-7aef9250bd05'
'981f26a1-7f43-403b-a875-f8b09b8cd720'

'6a0ec4d3-30cb-4a83-91c0-ae56bc0e3d26'

'432d3ac8-d6b9-43b9-9ef5-236ea50a049a'

'1efe5bbf-d5b1-4fe9-99fa-f55ce1c88679'

'a0f92522-89de-4c5e-9a75-0044ccf66efd'

'8c420feb-03df-47cc-8a05-55df0cf3064b'

'224a7b82-46c9-4d6b-8db0-7360fb444681'

'7319c514-987d-4e9b-ac3d-d38c4f427f4c'

'a232010e-820c-4083-83bb-3ace5fc29d0b'

'ea21b132-560f-4b0b-9876-903b6bca7b9d'

'585fc3c3-9a59-4720-8319-53cce041a605'

'408992c7-2af6-4ff1-92e3-65b73d2b5092'

'349e15d0-1c96-4829-95e5-7fc8fb358ff3'

'213f5f78-fb30-46c7-9e98-91c720a1c026'

'61c50b89-703d-431d-8d80-1e8618748775'

'f661c70c-c136-49d2-9b94-a2033c5874f8'

'4c1a3aed-b389-4824-99b0-514c07906851'

'6e10398e-57ba-4077-9027-e54a38dbc6dd'

'997dc448-eeab-4c93-8811-6b2c80196a16'

'6da466b6-1d13-4a2c-97bd-51a99e8d4d74'

'1212c1c8-6a34-4cfb-8ecb-1ffa5cbb9a39'

'6ee392c4-d339-4083-b04d-6b7947c6cf78'

'3efe2b8f-32eb-4c7a-ae9a-b43f17de36eb'

'00000006-0000-0ff1-ce00-000000000000'

'486c78bf-a0f7-45f1-92fd-37215929e116'

'341b7f3d-69b3-47f9-9ce7-5b7f4945fdbd'

'a57aca87-cbc0-4f3c-8b9e-dc095fdc8978'

'184909ca-69f1-4368-a6a7-c558ee6eb0bd'
'64b12d6e-6549-484c-8cc6-6281839ba394'
'ddb3205-c6bd-46ae-8127-60eb93363864'
'ca7f3f0b-7d91-482c-8e09-c5d840d0eac5'
'2a486b53-dbd2-49c0-a2bc-278bdfc30833'
'3cf6df92-2745-4f6f-bbcf-19b59bcdb62a'
'f98181d0-e16e-497c-9ac7-8c8e119ab8b5'
'9fc8264d-0a40-4790-86e1-e7b73a2d2298'
'797f4846-ba00-4fd7-ba43-dac1f8f63013'
'0426b777-1c7e-47f1-bdc1-e686dd0aedef1'
'4d0ad6c7-f6c3-46d8-ab0d-1406d5e6c86b'
'51df634f-ddb4-4901-8a2d-52f6393a796b'
'150c8903-2280-4ab6-8708-b080044d94c6'
'c7f7c435-4759-4a66-8981-13dc508936ba'
'04687a56-4fc2-4e36-b274-b862fb649733'
'4353526e-1c33-4fcf-9e82-9683edf52848'
'dba650ed-9577-4bc0-9b5f-ef73e2d5bdfc'
'992ddfc7-4a57-44ca-aeb5-ce7846062e10'
'cfbd4387-1a16-4945-83c0-ec10e46cd4da'
'd84904f1-cfbf-46a4-a8ed-6229ba95e220'
'bb8f18b0-9c38-48c9-a847-e1ef3af0602d'
'd828acde-4b48-47f5-a6e8-52460104a052'
'603b8c59-ba28-40ff-83d1-408eee9a93e5'
'eec53b1f-b9a4-4479-acf5-6b247c6a49f2'
'91bb937c-29c2-4275-982f-9465f0caf03d'
'76c7f279-7959-468f-8943-3954880e0d8c'

'a91b1853-4403-4f54-b5cb-d1ea19d90c37'

'93ac0ec6-05eb-4bed-94b2-4e955ebc01d3'

'b6b84568-6c01-4981-a80f-09da9a20bbcd'

'7cd684f4-8a78-49b0-91ec-6a35d38739ba'

'47ee738b-3f1a-4fc7-ab11-37e4822b007e'

'be14bf7e-8ab4-49b0-9dc6-a0eddd6fa73e'

'e6f9f783-1fdb-4755-acaf-abed6c642885'

'c7ddd9b4-5172-4e28-bd29-1e0792947d18'

'794ded15-70c6-4bcd-a0bb-9b7ad530a01a'

'055caf97-1b4f-4730-9f5d-acc24b707b06'

'c5b731db-1b0a-43f6-bcf6-757667d9cdc6'

'e81c7467-0fc3-4866-b814-c973488361cd'

'7d312290-28c8-473c-a0ed-8e53749b6d6d'

'd52792f4-ba38-424d-8140-ada5b883f293'

'00000014-0000-0000-c000-000000000000'

'8e3a7976-45be-4e5c-a34f-3adb97828d00'

'e572cafa-59db-4a44-badf-c3747f054c60'

'9cdead84-a844-4324-93f2-b2e6bb768d07'

'0af06dc6-e4b5-4f28-818e-e78e62d137a5'

'9ea1ad79-fdb6-4f9a-8bc3-2b70f96e34c7'

'0d38933a-0bbd-41ca-9ebd-28c4b5ba7cb7'

'6b95a49c-1c54-4c1a-8cfa-838cfff6ebd7'

'4962773b-9cdb-44cf-a8bf-237846a00ab7'

'589d5083-6f11-4d30-a62a-a4b316a14abf'

'c495cfdc-814f-46a1-89f0-657921c9fbe0'

'16f9e0a0-ac78-4c2c-a55a-f3855317a63a'

'a4a365df-50f1-4397-bc59-1a1564b8bb9c'
'b8ed041c-aa91-418e-8f47-20c70abc2de1'
'a0551534-cfc9-4e1f-9a7a-65093b32bb38'
'0f698dd4-f011-4d23-a33e-b36416dcb1e6'
'11c174dc-1945-4a9a-a36b-c79a0f246b9b'
'1d78a85d-813d-46f0-b496-dd72f50a3ec0'
'fc03f97a-9db0-4627-a216-ec98ce54e018'
'e406a681-f3d4-42a8-90b6-c2b029497af1'
'fd225045-a727-45dc-8caa-77c8eb1b9521'
'579d9c9d-4c83-4efc-8124-7eba65ed3356'
'4f6778d8-5aef-43dc-a1ff-b073724b9495'
'123cd850-d9df-40bd-94d5-c9f07b7fa203'
'394866fc-eedb-4f01-8536-3ff84b16be2a'
'ba1ea022-5807-41d5-bbeb-292c7e1cf5f6'
'8511a82b-10b4-43fb-b3bc-e621d8bdb532'
'262044b1-e2ce-469f-a196-69ab7ada62d3'
'608f9929-9737-432e-860f-4e1c1821052f'
'fb9de05a-fecc-4642-b3ca-66b9d4434d4d'
'3edcf11f-df80-41b2-a5e4-7e213cca30d1'
'9cb77803-d937-493e-9a3b-4b49de3f5a74'
'499b84ac-1321-427f-aa17-267ca6975798'
'2d696fb1-e862-4e01-ade3-2cf07c587ffd'
'3bc3fbf6-023a-4d86-bd09-bac559ccc9cc'
'07e1e85b-cfc5-4c4e-b7a6-3b4e315bd4e7'
'48af08dc-f6d2-435f-b2a7-069abd99c086'
'9191c4da-09fe-49d9-a5f1-d41cbe92ad95'

'6201d19e-14fb-4472-a2d6-5634a5c97568'

'e420dc86-d66f-4069-a2d0-be2f937bd272'

'35ffadb3-7fc1-497e-b61b-381d28e744cc'

'7f0d9978-eb2a-4974-88bd-f22a3006fe17'

'80dbdb39-4f33-4799-8b6f-711b5e3e61b6'

'9e5d84af-8971-422f-968a-354cd675ae5b'

'c505e273-0ba0-47e7-a0bd-f48042b4524d'

'a25dbca8-4e60-48e5-80a2-0664fdb5c9b6'

'9cd0f7df-8b1a-4e54-8c0c-0ef3a51116f6'

'cdad765c-f191-43ba-b9f5-7aef392f811d'

'a5ce81bb-67c7-4043-952a-22004782adb5'

'315970ab-3fc6-4208-9312-4bd5d35d6bb7'

'1fcdfafe-959b-4b32-afff-84f850974e84'

'de926fbf-e23b-41f9-ae15-c943a9cfa630'

'3541691f-c69a-4464-a9c2-ec714b5c174e'

'8bdebf23-c0fe-4187-a378-717ad86f6a53'

'5be973a1-1c81-4878-9ab6-6b261ab404e6'

'e6c69915-bcc7-4335-b655-c62f949d691b'

'3af5a1e8-2459-45cb-8683-bcd6cccbcc13'

'870c4f2e-85b6-4d43-bdda-6ed9a579b725'

'2db117cf-0a64-4da6-9a6d-243f08d09528'

'3e235f68-22ba-48f2-baf7-34bc3d1c6ad2'

'98785600-1bb7-4fb9-b9fa-19afe2c8a360'

'3072002f-3e97-4979-91f2-09fe40da755d'

'c9455e78-f730-4ba3-a551-cb6a1c389089'

'51667a77-8210-49d8-b82f-8db910137ae2'

'e0ccf59d-5a20-4a87-a122-f42842cdb86a'
'161a339d-b9f5-41c5-8856-6a6669acac64'
'5b20c633-9a48-4a5f-95f6-dae91879051f'
'8b5a06b3-efd5-4511-b112-5caf390d1b92'
'270d0f73-6132-43b9-9666-9f5e473b7e1f'
'dbcbd02a-d7c4-42fb-8c27-b07e5118b848'
'8f41dc7c-542c-4bdd-8eb3-e60543f607ca'
'd22ea4d1-2678-4a7b-aa5e-f340c2a7d993'
'93efed00-6552-4119-833a-422b297199f9'
'1690c5aa-925a-4d0e-836b-722c795bd0d0'
'8cae6e77-e04e-42ce-b5cb-50d82bce26b1'
'6dae42f8-4368-4678-94ff-3960e28e3630'
'324395f8-e27f-48a8-8b0b-6338c76ce949'
'f0ae4899-d877-4d3c-ae25-679e38eea492'
'b8340c3b-9267-498f-b21a-15d5547fd85e'
'09cf45fc-b312-44eb-a287-e6582ea20ae0'
'6ea8091b-151d-447a-9013-6845b83ba57b'
'e9b154d0-7658-433b-bb25-6b8e0a8a7c59'
'd5f39c4e-4b59-4b22-88cc-a70c399ba1cd'
'03b39d0f-4213-4864-a245-b1476ec03169'
'd3244f1e-56a7-4819-80e9-a30a7a83dde8'
'18a4ad1e-427c-4cad-8416-ef674e801d32'
'bf4fa6bf-d24c-4d1c-8cfd-12063dd646b2'
'a3747411-ce7c-4888-9ddc-3a230786ca19'
'0000000c-0000-0000-c000-000000000000'
'cbfda01c-c883-45aa-aedc-e7a484615620'

'ce6ff14a-7fdc-4685-bbe0-f6afdfcfa8e0'

'b28ec8e1-950e-4bd0-b3d0-c1e93074b88b'

'78e7bc61-0fab-4d35-8387-09a8d2f5a59d'

'9199bf20-a13f-4107-85dc-02114787ef48'

'66c6d0d1-f2e7-4a18-97a9-ed10f3347016'

'17724442-aa9a-46cc-bf09-c47bb1a98518'

'3eb95cef-b10f-46fe-94e0-969a3d4c9292'

'0a5f63c0-b750-4f38-a71c-4fc0d58b89e2'

'443155a6-77f3-45e3-882b-22b3a8d431fb'

'ef5d5c69-a5df-46bb-acaf-426f161a21a2'

'2746ea77-4702-4b45-80ca-3c97e680e8b7'

'6bb8e274-af5d-4df2-98a3-4fd78b4cafd9'

'22b71bf7-6952-44c8-8b92-893ca3b117d2'

'2dc3760b-4713-48b1-a383-1dfe3e449ec2'

'b3d7adc8-6b92-4061-b47d-fc1419774ccc'

'75e725bf-66ce-4cea-9b9a-5c4caae57f33'

'c7bb12bf-0b39-4f7f-9171-f418ff39b76a'

'3cbcded7-0049-4401-9e00-5f4f10f75efe'

'b998f6f8-79d0-4b6a-8c25-5791dbe49ad0'

'737d58c1-397a-46e7-9d12-7d8c830883c2'

'26a4ae64-5862-427f-a9b0-044e62572a4f'

'ce48853e-0605-4f77-8746-d70ac63cc6bc'

'8338dec2-e1b3-48f7-8438-20c30a534458'

'5bfdc2bf-507b-4413-8f15-f1fcf1c15475'

'022907d3-0f1b-48f7-badc-1ba6abab6d66'

'5b306cba-9c71-49db-96c3-d17ca2379c4d'

'2e458d69-0892-4655-b713-4f7b182315dd'

'a85cf173-4192-42f8-81fa-777a763e6e2c'

'b61cc489-e138-4a69-8bf3-c2c5855c8784'

'959ddb6c-47bf-4db5-bb68-6d77b88a1250'

'990ba54a-28b0-4603-9345-b23f426585c4'

'ac9dc5fe-b644-4832-9d03-d9f1ab70c5f7'

'0469d4cd-df37-4d93-8a61-f8c75b809164'

'557c67cf-c916-4293-8373-d584996f60ae'

'8ae6a0b1-a07f-4ec9-927a-afb8d39da81c'

'a77d91dc-971b-4cf7-90c8-f183194249bc'

'372140e0-b3b7-4226-8ef9-d57986796201'

'b2cc270f-563e-4d8a-af47-f00963a71dcd'

'c728155f-7b2a-4502-a08b-b8af9b269319'

'87223343-80b1-4097-be13-2332ffa1d666'

'bb55177b-a7d9-4939-a257-8ab53a3b2bc6'

'5b712e99-51a3-41ce-86ff-046e0081c5c0'

'df09ff61-2178-45d8-888c-4210c1c7b0b2'

'b4ca0290-4e73-4e31-ade0-c82ecfaabf6a'

'00000004-0000-0ff1-ce00-000000000000'

'd3d1a4fe-edb2-4b09-bc39-e41d342323d6'

'6608bce8-e060-4e82-bfd2-67ed4f60262f'

'4c4f550b-42b2-4a16-93f9-fdb9e01bb6ed'

'05a65629-4c1b-48c1-a78b-804c4abdd4af'

'57084ef3-d413-4087-a28f-f6f3b1ad7786'

'884e625b-d232-4ca3-88ae-fd138b5084ac'

'cb082ace-8a09-486a-9d02-c15739683b01'

'3734c1a4-2bed-4998-a37a-ff1a9e7bf019'

'00000007-0000-0ff1-ce00-000000000000'

'65d91a3d-ab74-42e6-8a2f-0add61688c74'

'9581bc0e-c952-4fd3-8d99-e777877718b1'

'509e4652-da8d-478d-a730-e9d4a1996ca4'

'f738ef14-47dc-4564-b53b-45069484ccc7'

'0a0a29f9-0a25-49c7-94bf-c53c3f8fa69d'

'ba9ff945-a723-4ab5-a977-bd8c9044fe61'

'c321abdf-8dc2-4906-a0c7-2daaf7af298c'

'58ea322b-940c-4d98-affb-345ec4cccb92'

'60fff137-909f-4cdc-8e13-3e11f583788a'

'12fb057d-b751-47cd-857c-f2934bb677b4'

'cb4dc29f-0bf4-402a-8b30-7511498ed654'

'0736f41a-0425-4b46-bdb5-1563eff02385'

'a1b76039-a76c-499f-a2dd-846b4cc32627'

'8edd93e1-2103-40b4-bd70-6e34e586362d'

'461e8683-5575-4561-ac7f-899cc907d62a'

'ba4bc2bd-843f-4d61-9d33-199178eae34e'

'faf1f7e1-8522-498c-a7f9-6a7f1a17f873'

'dbc36ae1-c097-4df9-8d94-343c3d091a76'

'3c31d730-a768-4286-a972-43e9b83601cd'

'29820072-374d-49b8-945a-3941d7e9b468'

'b5615b95-29f3-4ea5-bce3-50a07b952601'

'5b534afd-fdc0-4b38-a77f-af25442e3149'

'7865c1d2-f040-46cc-875f-831a1ef6a28a'

'9d777fa9-b417-43b8-8991-12f8ee2161d2'

'3b990c8b-9607-4c2a-8b04-1d41985facca'
'd2a0a418-0aac-4541-82b2-b3142c89da77'
'3274406e-4e0a-4852-ba4f-d7226630abb7'
'120d688d-1518-4cf7-bd38-182f158850b6'
'c5393580-f805-4401-95e8-94b7a6ef2fc2'
'0130cc9f-7ac5-4026-bd5f-80a08a54e6d9'
'c4b6c9c7-e308-48f1-83b3-04076f680ea0'
'58ef1dbd-684c-47d6-8ffc-61ea7a197b95'
'525036f0-a5e3-4a2b-bba7-440d8d71647f'
'a0be0c72-870e-46f0-9c49-c98333a996f7'
'7bf610f7-ecaf-43a2-9dbc-33b14314d6fe'
'9edfcdd9-0bc5-4bd4-b287-c3afc716aac7'
'8e0e8db5-b713-4e91-98e6-470fed0aa4c2'
'3030ef2a-57cd-4c9b-9490-badf0d0cf3ff'
'9d06afd9-66c9-49a6-b385-ea7509332b0b'
'd8c767ef-3e9a-48c4-aef9-562696539b39'
'6f82282e-0070-4e78-bc23-e6320c5fa7de'
'5d661950-3475-41cd-a2c3-d671a3162bc1'
'bf26f092-3426-4a99-abfb-97dad74e661a'
'0140a36d-95e1-4df5-918c-ca7ccd1fafc9'
'9f0e9e2d-6525-4eb8-b2e0-da2314d017b5'
'b9a92e36-2cf8-4f4e-bcb3-9d99e00e14ab'
'959678cf-d004-4c22-82a6-d2ce549a58b8'
'00000009-0000-0000-c000-000000000000'
'cc5476ec-3074-44d1-8461-711f5d9b0e39'
'7cd01e22-2632-4b54-ad19-a260969ba7ff'

'c4288165-6698-45ba-98a5-48ea7791fed3'

'c066d759-24ae-40e7-a56f-027002b5d3e4'

'f1dd0a37-89c6-4e07-bcd1-ffd3d43d8875'

'00000001-0000-0000-c000-000000000000'

'a23667b4-c8cf-42c1-86d6-25a4f46525f8'

'64a7b174-5779-4506-b54c-fbb0d80f1c9b'

'74cb6831-0dbb-4be1-8206-fd4df301cdc2'

'12743ff8-d3de-49d0-a4ce-6c91a4245ea0'

'd87dcbc6-a371-462e-88e3-28ad15ec4e64'

'a48bbb4a-8f73-478e-b492-5e1f05694d54'

'2bceb21e-ead7-46f2-a187-10ab2e65f34e'

'f7069a8d-9edc-4300-b365-ae53c9627fc4'

'd6fdaa33-e821-4211-83d0-cf74736489e1'

'9c8b80bc-6887-42d0-b1af-d0c40f9bf1fa'

'eace8149-b661-472f-b40d-939f89085bd4'

'aafac111-b9f0-439b-bbe3-1839c672ac8e'

'cf6163c2-70e2-4a66-ba08-0d46eb68dcf9'

'c115998b-3d59-49b4-b55b-042a9ba1dbfe'

'ca0a114d-6fbc-46b3-90fa-2ec954794ddb'

'0c7668b5-3260-4ad0-9f53-34ed54fa19b2'

'29f411f1-b2cf-4043-8ac8-2185d7316811'

'c6e44401-4d0a-4542-ab22-ecd4c90d28d7'

'5741a1ff-751d-4ad7-bcd1-dfe3c998fd11'

'b7912db9-aa33-4820-9d4f-709830fdd78f'

'2cfc91a4-7baa-4a8f-a6c9-5f3d279060b8'

'9469b9f5-6722-4481-a2b2-14ed560b706f'

'1dcb1bc7-c721-498e-b2fa-bcddcea44171'
'54b9b9be-c365-4548-95c6-d2f2011f48f4'
'ddc728e9-153d-4032-ab80-80e57af7a56f'
'58c746b0-a0b0-4647-a8f6-12dde5981638'
'19947cfd-0303-466c-ac3c-fcc19a7a1570'
'abcdbe4b-4a26-4340-a0bd-04aed964e8af'
'7de4d5c5-5b32-4235-b8a9-33b34d6bcd2a'
'ce79fdc4-cd1d-4ea5-8139-e74d7dbe0bb7'
'257601fd-462f-4a21-b623-7f719f0f90f4'
'8ee8fdad-f234-4243-8f3b-15c294843740'
'3480ae63-e100-432c-92c8-4aee777c5671'
'd2a590e7-6906-4a45-8f41-cecfdca9bca1'
'a4c95b9e-3994-40cc-8953-5dc66d48348d'
'707be275-6b9d-4ee7-88f9-c0c2bd646e0f'
'e935b4a5-8968-416d-8414-caed51c782a9'
'ec245c98-4a90-40c2-955a-88b727d97151'
'608f6f31-fed0-4f7b-809f-90f6c9b3de78'
'6d057c82-a784-47ae-8d12-ca7b38cf06b4'
'e8de9221-a19c-4c81-b814-fd37c6caf9d2'
'5e99b34c-2ea8-49d4-b205-1c8bb56114d8'
'9ae330ab-d710-466b-851c-c828e7340846'
'aae1f723-e3f6-4223-a7cd-0a590e2247bc'
'40775b29-2688-46b6-a3b5-b256bd04df9f'
'ccf4d8df-75ce-4107-8ea5-7afd618d4d8a'
'01cb2876-7ebd-4aa4-9cc9-d28bd4d359a9'
'c39c9bac-9d1f-4dfb-aa29-27f6365e5cb7'

'a4bad4aa-bf02-4631-9f78-a64ffdba8150'
'6a0a243c-0886-468a-a4c2-eff52c7445da'
'ecad3f28-c75d-4414-94e0-a5e1de4df79e'
'42bb5b6c-1300-4c06-b3ef-bdadc9571ec9'
'd9327919-6775-4843-9037-3fb0fb0473cb'
'9bd5ab7f-4031-4045-ace9-6bebbad202f6'
'63c2c773-89fe-4164-a02f-b8c7fc1772ae'
'1e3e4475-288f-4018-a376-df66fd7fac5f'
'e5408ad0-c4e2-43aa-b6f2-3b4951286d99'
'a8399344-a951-4537-8cf2-444aede33ff2'
'd8877f27-09c0-43aa-8113-40151dae8b14'
'e526e72f-ffae-44a0-8dac-cf14b8bd40e2'
'35274a02-4c3a-4878-9c89-7b9f1b09644f'
'475226c6-020e-4fb2-8a90-7a972cbfc1d4'
'03db181c-e9d3-4868-9097-f0b728327182'
'5a543d7c-9c4a-4f90-8cc7-6ae082a5b65b'
'5b81a823-5f67-4fb3-8d0f-4c92b5044fe4'
'157638eb-a5cb-4c10-af42-2d6759eb1871' nan]

4. Missing Values Count:

| | |
|-----------------|-----|
| id | 0 |
| displayName | 488 |
| appld | 488 |
| createdDateTime | 488 |
| applicationType | 493 |
| accountEnabled | 488 |

applicationVisibility 488

assignmentRequired 488

isAppProxy 488

dtype: int64

5. Application Type Counts:

Microsoft Application 497

Enterprise Application 46

Managed Identity 30

Name: applicationType, dtype: int64

6. Average Display Name Length: 24.68 characters

7. Earliest Created Date: 2022-03-04 02:35:55+00:00

8. Latest Created Date: 2022-05-04 08:34:59+00:00

9. Number of Account-Enabled Applications: 576

10. Number of App Proxy Applications: 0

11. Number of Visible Applications: 576

12. Total Count of Applications: 1066

13. Application Creation by Year:

2022.0 578

Name: createdYear, dtype: int64

14. Application Creation by Month:

2022-04 465

2022-03 69

2022-05 44

Name: createdMonth, dtype: int64

15. Display Name with Maximum Length: Defender for Containers provisioning Azure Policy Addon for Kub

16. Display Name with Minimum Length: 5

17. Total Number of Unique Apps: 578

18. Total Number of Enterprise Applications: 46

19. Total Number of Non-Enterprise Applications: 1020

20. Application Visibility Counts:

Visible 576

Hidden 2

Name: applicationVisibility, dtype: int64

<ipython-input-5-5007f1804046>:11: FutureWarning: Treating datetime data as categorical rather than numeric in `.describe` is deprecated and will be removed in a future version of pandas. Specify `datetime_is_numeric=True` to silence this warning and adopt the future behavior now.

```
basic_stats = data.describe()
```

ASSISTANT

Creating a script that loads and installs everything you need from the given data involves multiple steps. I'll provide you with a Python script using the Pandas library for data manipulation and pip for

package installation. First, you need to install the required libraries if you haven't already. Here's a script to get you started:

```
```python

Import necessary libraries

import pandas as pd

import os

import subprocess

Define your data as provided

data = {

 "id": ["1066", "unique", "top", "freq", "first", "last"],

 "displayName": ["578", "571", "Domain Controller Services", "3", "NaN", "NaN"],

 "appId": [

 "578",

 "298",

 "3f18b202-a866-4c1f-b9c0-6598390a2092",

 "31",

 "2022-03-04 02:35:55+00:00",

 "2022-05-04 08:34:59+00:00",

],

 "createdDateTime": [

 "573",

 "2",

 "Microsoft Application",

 "497",
```

```
"True",
"Visible",
],
"applicationType": ["578", "576", "False", "False"],
"accountEnabled": ["578", "1"],
"applicationVisibility": ["578", "2"],
"assignmentRequired": ["578", "1"],
"isAppProxy": ["578", "False"],
}
```

```
Create a Pandas DataFrame from the data
```

```
df = pd.DataFrame(data)
```

```
Step 1: Basic Statistics
```

```
print("Step 1: Basic Statistics")
```

```
print(df)
```

```
Step 2: Unique Application Types
```

```
unique_app_types = df["applicationType"].unique()
```

```
print("\nStep 2: Unique Application Types")
```

```
print(unique_app_types)
```

```
Step 3: Unique Application IDs
```

```
unique_app_ids = df["appId"].unique()
```

```
print("\nStep 3: Unique Application IDs")
```

```
print(unique_app_ids)
```

# Step 4: Install Required Packages

```
required_packages = [
```

```
 "pandas",
```

```
 "numpy",
```

```
 "subprocess",
```

```
]
```

```
for package in required_packages:
```

```
 print(f"\nInstalling {package}...")
```

```
 subprocess.check_call(["pip", "install", package])
```

```
print("\nAll required packages have been installed.")
```

# Step 5: Additional Data Processing or Actions

# You can add any additional data processing or actions here.

# End of the script

```
...
```

Make sure you have Python and pip installed on your system. This script does the following:

1. Loads the provided data into a Pandas DataFrame.
2. Displays basic statistics.
3. Finds unique application types and IDs.
4. Installs required packages.

5. You can add any additional data processing or actions in the last step.

You can run this script in a Python environment to perform the tasks you mentioned.

## USER

Make it install my Microsoft apps from csv.txt

## ASSISTANT

To install Microsoft apps from a CSV file, you can use a script in a language like Python. Here's a simple example of how you can do this:

```
```python
import csv

from subprocess import call

# Read the CSV file
with open('apps.csv', 'r') as file:

    reader = csv.reader(file)

    next(reader) # Skip the header row if present

    for row in reader:

        app_id = row[0] # Assuming the first column contains the app IDs

        # Use 'call' to install the app, replace 'install_command' with the actual installation command

        install_command = f'your_install_command_here {app_id}'

        call(install_command, shell=True)
```
```

Make sure to replace ``apps.csv`` with the actual path to your CSV file and provide the correct installation command for your Microsoft apps. This script reads the CSV file, extracts the app IDs, and runs the installation command for each app.

Please note that this is a basic example, and you may need to customize it according to your specific requirements and the installation commands for your Microsoft apps.

```
USER
```

```
import csv
```

```
from subprocess import call
```

```
Read the CSV file
```

```
with open('apps.csv', 'r') as file:
```

```
 reader = csv.reader(file)
```

```
 next(reader) # Skip the header row if present
```

```
 for row in reader:
```

```
 app_id = row[0] # Assuming the first column contains the app IDs
```

```
 # Use 'call' to install the app, replace 'install_command' with the actual installation command
```

```
 install_command = f'your_install_command_here {app_id}'
```

```
 call(install_command, shell=True)
```

```
ASSISTANT
```

It looks like you've provided a Python script for installing Microsoft apps from a CSV file. To make it work, you need to replace ``your\_install\_command\_here`` with the actual installation command for your Microsoft apps.

Here's how you can modify the script with an example installation command (assuming you have the app IDs in the CSV file):

```
```python
import csv
import subprocess

# Read the CSV file
with open('apps.csv', 'r') as file:

    reader = csv.reader(file)

    next(reader) # Skip the header row if present

    for row in reader:

        app_id = row[0] # Assuming the first column contains the app IDs

        # Replace 'install_command' with the actual installation command for your apps
        install_command = f'your_install_command_here {app_id}'

        try:

            subprocess.run(install_command, shell=True, check=True)

        except subprocess.CalledProcessError as e:

            print(f"Error installing app {app_id}: {e}")
...
```
```

Remember to replace ``your\_install\_command\_here`` with the real installation command that you use to install your Microsoft apps. This script will read the CSV file, extract app IDs, and attempt to install each app using the specified command. It also handles errors in case an installation fails.

## USER

Failed to download: ff791a79-8377-4d5e-95b9-0bd82e674308

Failed to download: ff8b3a8e-da32-4e9c-909a-b9a149ef11e3

Failed to download: fff88d6c-ae9c-4ceb-9306-d2cc2cc7af53

-----  
IndexError Traceback (most recent call last)

<ipython-input-8-d1853ff55a6e> in <cell line: 5>()

```
8 for row in reader:
9 app_name = row[0] # Assuming the first column contains the app names
--> 10 install_command = row[1] # Assuming the second column contains the installation
commands
11
12 # Use subprocess to run the installation command
```

IndexError: list index out of range

## ASSISTANT

The error you're encountering, "IndexError: list index out of range," indicates that you're trying to access an index in a list that doesn't exist. In this case, it means that the script is trying to access `row[1]`, which is the second column of your CSV, but there is no data in that column for some rows.

To fix this issue, you should check if the row has the expected number of columns before trying to access them. Here's an updated version of your script that includes error checking:

```
```python
```



```

import csv

import subprocess

# Read the CSV file

with open('apps.csv', 'r') as file:

    reader = csv.reader(file)

    next(reader) # Skip the header row if present

    for row in reader:

        if len(row) >= 2: # Check if there are at least two columns in the row

            app_name = row[0] # Assuming the first column contains the app names

            install_command = row[1] # Assuming the second column contains the installation
commands

            try:

                subprocess.run(install_command, shell=True, check=True)

            except subprocess.CalledProcessError as e:

                print(f"Error installing app {app_name}: {e}")

            else:

                print(f"Row does not have enough columns: {row}")

...

```

This modified script checks if a row has at least two columns before attempting to access the app name and installation command. If a row doesn't have enough columns, it will print an error message indicating the issue.

USER

Error installing app fefefbb4-5f46-4dda-837b-1dc751bf9ea2: Command 'Salesforce' returned non-zero exit status 127.

Error installing app ff16bd6f-686a-42a2-8696-87bfda4010aa: Command 'PowerApps Service' returned non-zero exit status 127.

Error installing app ff249d09-c032-4d0f-baa3-5eef5c48ad20: Command 'ClusterConfigToAKS' returned non-zero exit status 127.

Error installing app ff791a79-8377-4d5e-95b9-0bd82e674308: Command 'Azure HDInsight Surrogate Service' returned non-zero exit status 127.

Error installing app ff8b3a8e-da32-4e9c-909a-b9a149ef11e3: Command 'Liftr Elastic Whale Worker' returned non-zero exit status 127.

Error installing app fff88d6c-ae9c-4ceb-9306-d2cc2cc7af53: Command 'VMwareVMIdentityForHIS' returned non-zero exit status 127.

Row does not have enough columns: ["'157638eb-a5cb-4c10-af42-2d6759eb1871'"]

ASSISTANT

It appears that there are some issues with the installation commands in your CSV file. The errors you're seeing are typically related to the commands not being found or executable. Here are some things to check and possible solutions:

1. **Command Existence**: Ensure that the commands specified in the CSV file actually exist and are executable. The error code 127 typically indicates that the command was not found.
2. **Full Paths**: If the commands are not in the system's PATH, you might need to specify the full path to the executable. For example, use `/path/to/Salesforce` instead of just `Salesforce`.
3. **Quotes**: It seems that some of the app names are enclosed in single quotes within the CSV.

Make sure that the app names and installation commands are correctly formatted in your CSV without extra quotes or special characters.

4. ****Missing Data****: The last error indicates that there's an issue with one of the rows. It appears to have extra characters and might not be correctly formatted. Ensure that the data in your CSV file is well-structured.

5. ****Exit Status 127****: This exit status typically indicates that the command could not be executed because it's not found. Double-check the command names and paths.

Once you've addressed these issues, your installation process should work more smoothly. If you encounter further errors, review the specific command that's causing the problem to ensure it can be run in your environment.

USER

Successfully downloaded: '5a543d7c-9c4a-4f90-8cc7-6ae082a5b65b'

Successfully downloaded: '5b81a823-5f67-4fb3-8d0f-4c92b5044fe4'

IndexError Traceback (most recent call last)

<ipython-input-8-d1853ff55a6e> in <cell line: 5>()

```
8   for row in reader:
9       app_name = row[0] # Assuming the first column contains the app names
--> 10       install_command = row[1] # Assuming the second column contains the installation
commands
11
12     # Use subprocess to run the installation command
```

IndexError: list index out of range

ASSISTANT

It looks like you've successfully downloaded some apps, but you're encountering an `IndexError` when trying to access the second column in your CSV file. This error occurs when there aren't enough columns in the row to access.

To fix this issue, you should ensure that your CSV file has consistent data in all rows, meaning that each row should contain both the app name and the installation command. Here are a few steps to resolve this:

1. ****Check CSV Format****: Review your CSV file to make sure it has two columns: one for the app name and another for the installation command. It seems that some rows are missing the second column.
2. ****Handle Missing Data****: You can check the number of columns in each row before trying to access them. For example:

```
```python
with open('apps.csv', 'r') as file:
 reader = csv.reader(file)
 next(reader) # Skip the header row if present
 for row in reader:
 if len(row) >= 2:
 app_name = row[0]
```

```

 install_command = row[1]

 # Continue with installation

 else:

 print(f"Row does not have enough columns: {row}")

 ...

```

This will prevent the `IndexError` and allow you to handle rows with missing data more gracefully.

Make sure your CSV file is correctly formatted with two columns for each row, one for the app name and one for the installation command.

```
USER
```

```
import csv
```

```
import subprocess
```

```
Read the CSV file
```

```
with open('csv.txt', 'r') as file:
```

```
 reader = csv.reader(file)
```

```
 next(reader) # Skip the header row if present
```

```
 for row in reader:
```

```
 app_name = row[0] # Assuming the first column contains the app names
```

```
 install_command = row[1] # Assuming the second column contains the installation commands
```

```
 # Use subprocess to run the installation command
```

```
 try:
```

```
 result = subprocess.call(install_command, shell=True)
```

```
if result == 0:

 print(f"Successfully downloaded: {app_name}")

else:

 print(f"Failed to download: {app_name}")

except Exception as e:

 print(f"Error while downloading {app_name}: {str(e)}")
```

## ASSISTANT

Your code looks good for reading the CSV file, extracting app names and installation commands, and running the installation commands using subprocess. If everything is correctly formatted in your CSV file, this code should work as expected.

Just make sure that your CSV file 'csv.txt' contains the app names in the first column and the installation commands in the second column. If any issues arise during the installation process, your code will print appropriate messages indicating success or failure.

Feel free to use this code, and if you encounter any specific issues or have further questions, please let me know, and I'll be happy to assist you further.

## USER

```
import csv

import subprocess

Read the CSV file

with open('csv.txt', 'r') as file:

 reader = csv.reader(file)
```

```

next(reader) # Skip the header row if present

for row in reader:

 app_name = row[0] # Assuming the first column contains the app names

 install_command = row[1] # Assuming the second column contains the installation commands

 # Use subprocess to run the installation command

 try:

 result = subprocess.call(install_command, shell=True)

 if result == 0:

 print(f"Successfully downloaded: {app_name}")

 else:

 print(f"Failed to download: {app_name}")

 except Exception as e:

 print(f"Error while downloading {app_name}: {str(e)}") with open('apps.csv', 'r') as file:

reader = csv.reader(file)

next(reader) # Skip the header row if present

for row in reader:

 if len(row) >= 2:

 app_name = row[0]

 install_command = row[1]

 # Continue with installation

 else:

 print(f"Row does not have enough columns: {row}")

```

## ASSISTANT

It seems there's a mix of two code snippets in your response. If you want to handle cases where

rows might not have enough columns, you can use the code snippet you provided with the addition of the check for the number of columns, like this:

```
```python
import csv
import subprocess

# Read the CSV file
with open('csv.txt', 'r') as file:

    reader = csv.reader(file)

    next(reader) # Skip the header row if present

    for row in reader:

        if len(row) >= 2:

            app_name = row[0] # Assuming the first column contains the app names

            install_command = row[1] # Assuming the second column contains the installation
commands

            # Use subprocess to run the installation command
            try:

                result = subprocess.call(install_command, shell=True)

                if result == 0:

                    print(f"Successfully downloaded: {app_name}")

                else:

                    print(f"Failed to download: {app_name}")

            except Exception as e:

                print(f"Error while downloading {app_name}: {str(e)}")
```



```
else:
    print(f"Row does not have enough columns: {row}")
...
```

This code will handle cases where a row doesn't have the expected number of columns and print a message about it. It will continue processing rows with the correct number of columns.

USER

Make it parse and download all into a deepnet

ASSISTANT

I'm not entirely sure what you mean by "parse and download all into a deepnet." It seems like you want to parse data from a CSV file and somehow use it in the context of a "deepnet." However, it's important to clarify what you mean by "deepnet" because it can have different meanings in various contexts.

If by "deepnet," you mean a deep neural network or some other specific technology or system, please provide more details about what you want to achieve with the parsed data. This will help me give you more specific guidance on how to proceed.

USER

Ya

ASSISTANT

It seems you'd like to use the parsed data from the CSV file in the context of a deep neural network. To do this, you'll need to provide more specific information about your goals and the structure of the

data you want to use for the deep neural network.

Please share additional details about:

1. The purpose or task of the deep neural network (e.g., image recognition, natural language processing, etc.).
2. The format and type of data in the CSV file (e.g., images, text, numerical data).
3. Any specific framework or library you intend to use for building the deep neural network (e.g., TensorFlow, PyTorch).

With this information, I can provide more tailored guidance on how to parse the data and integrate it into your deep neural network.

USER

Give me the code that does the

ASSISTANT

Certainly, I can provide you with a basic Python code template that loads data from a CSV file and prepares it for use in a deep neural network. This code assumes you have a CSV file where each row contains data for your deep learning model. You would need to adapt it based on your specific dataset and use case.

Here's a simple example using Python and the popular deep learning library, TensorFlow:

```
```python
import csv
```

```
import tensorflow as tf

from tensorflow import keras

Define your deep learning model

model = keras.Sequential([

 # Add your layers here

])

Read the CSV file

data = []

labels = []

with open('your_dataset.csv', 'r') as file:

 reader = csv.reader(file)

 next(reader) # Skip the header row if present

 for row in reader:

 # Assuming that the last column is the label and the rest are features

 features = [float(value) for value in row[:-1]]

 label = int(row[-1]) # Assuming it's a numerical label

 data.append(features)

 labels.append(label)

Convert lists to TensorFlow tensors

data = tf.convert_to_tensor(data, dtype=tf.float32)

labels = tf.convert_to_tensor(labels, dtype=tf.int64)
```

```
Define your data preprocessing (e.g., normalization) here
```

```
Compile and train your model
```

```
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

```
model.fit(data, labels, epochs=10, validation_split=0.2) # Adjust the number of epochs and
validation split
```

```
Make predictions using the trained model
```

```
Save or export your model if needed
```

```
...
```

Please note that this is a basic template, and you'll need to customize it according to the specifics of your dataset, model architecture, and training process. You may also need to preprocess your data and choose an appropriate loss function and evaluation metric depending on your task.

```
USER
```

Make it do everything in the fucking file is csv.txt

```
ASSISTANT
```

I apologize if there was any confusion. To read both the data and labels from a CSV file like `csv.txt` and create a deep learning model, you can adapt the code as follows:

```
```python
```

```
import csv
```

```
import tensorflow as tf

from tensorflow import keras

# Define your deep learning model

model = keras.Sequential([

    # Add your layers here

])

# Read the CSV file

data = []

labels = []

with open('csv.txt', 'r') as file:

    reader = csv.reader(file)

    next(reader) # Skip the header row if present

    for row in reader:

        # Assuming that the last column is the label and the rest are features

        features = [float(value) for value in row[:-1]]

        label = int(row[-1]) # Assuming it's a numerical label

        data.append(features)

        labels.append(label)

# Convert lists to TensorFlow tensors

data = tf.convert_to_tensor(data, dtype=tf.float32)

labels = tf.convert_to_tensor(labels, dtype=tf.int64)
```

```
# Define your data preprocessing (e.g., normalization) here
```

```
# Compile and train your model
```

```
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

```
model.fit(data, labels, epochs=10, validation_split=0.2) # Adjust the number of epochs and  
validation split
```

```
# Make predictions using the trained model
```

```
# Save or export your model if needed
```

```
...
```

This code assumes that `csv.txt` contains both the feature data and labels in each row and that the last column in each row is the label. Make sure to adjust the model architecture, data preprocessing, and training parameters according to your specific use case.

```
## USER
```

Make it do more make it do predictions make it, save the fucking shit and make it export it make it authorize pi drive to be able to send it back to my Drive

```
## ASSISTANT
```

Certainly, to enhance the code to include predictions, save the model, and authorize Google Drive for exporting, you can use the following code:

```
```python
```

```
import csv

import tensorflow as tf

from tensorflow import keras

from pydrive.auth import GoogleAuth

from pydrive.drive import GoogleDrive
```

```
Define your deep learning model
```

```
model = keras.Sequential([

 # Add your layers here

])
```

```
Read the CSV file
```

```
data = []
```

```
labels = []
```

```
with open('csv.txt', 'r') as file:
```

```
 reader = csv.reader(file)
```

```
 next(reader) # Skip the header row if present
```

```
 for row in reader:
```

```
 # Assuming that the last column is the label and the rest are features
```

```
 features = [float(value) for value in row[:-1]]
```

```
 label = int(row[-1]) # Assuming it's a numerical label
```

```
 data.append(features)
```

```
 labels.append(label)
```

```
Convert lists to TensorFlow tensors
```

```
data = tf.convert_to_tensor(data, dtype=tf.float32)
```

```
labels = tf.convert_to_tensor(labels, dtype=tf.int64)
```

```
Define your data preprocessing (e.g., normalization) here
```

```
Compile and train your model
```

```
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

```
model.fit(data, labels, epochs=10, validation_split=0.2) # Adjust the number of epochs and
validation split
```

```
Make predictions using the trained model
```

```
predictions = model.predict(data)
```

```
Save the trained model to a file
```

```
model.save('your_model.h5')
```

```
Authorize and export the model to Google Drive
```

```
gauth = GoogleAuth()
```

```
gauth.LocalWebserverAuth() # Follow the authorization steps in the browser
```

```
drive = GoogleDrive(gauth)
```

```
Create a Google Drive file
```

```
model_file = drive.CreateFile()
```

```
model_file.SetContentFile('your_model.h5')
```

```
model_file.Upload()
```



```
Print the link to the model in Google Drive

print('Model uploaded: {}'.format(model_file['title']))

'''
```

This code integrates Google Drive using the PyDrive library to save and export your trained model. Make sure to adjust the file name and paths as needed.

## USER

The error is caused by the value '00a67483-ea0d-4bfa-a19b-17ae35e97fe3' in the CSV file. This value cannot be converted to a float, which is required for the float() function.

To fix the error, you can either remove the row containing the invalid value, or you can convert the value to a float using a different method, such as float(value, errors='ignore').

Here is the modified code:

```
content_copy
```

```
import csv

import tensorflow as tf

from tensorflow import keras

from pydrive.auth import GoogleAuth

from pydrive.drive import GoogleDrive
```

```
Define your deep learning model
```

```
model = keras.Sequential([
```

```
 # Add your layers here
```

```
)
```

```
Read the CSV file
```

```
data = []
```

```
labels = []
```

```
with open('csv.txt', 'r') as file:
```

```
 reader = csv.reader(file)
```

```
 next(reader) # Skip the header row if present
```

```
 for row in reader:
```

```
 # Assuming that the last column is the label and the rest are features
```

```
 features = [float(value) for value in row[:-1]]
```

```
 label = int(row[-1]) # Assuming it's a numerical label
```

```
 data.append(features)
```

```
 labels.append(label)
```

```
Convert lists to TensorFlow tensors
```

```
data = tf.convert_to_tensor(data, dtype=tf.float32)
```

```
labels = tf.convert_to_tensor(labels, dtype=tf.int64)
```

```
Define your data preprocessing (e.g., normalization) here
```

```
Compile and train your model
```

```
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

```
model.fit(data, labels, epochs=10, validation_split=0.2) # Adjust the number of epochs and validation
```

n split

# Make predictions using the trained model

```
predictions = model.predict(data)
```

# Save the trained model to a file

```
model.save('your_model.h5')
```

# Authorize and export the model to Google Drive

```
gauth = GoogleAuth()
```

```
gauth.LocalWebserverAuth() # Follow the authorization steps in the browser
```

```
drive = GoogleDrive(gauth)
```

# Create

Use code with caution

Sources

1 biwswalker/TFUnet

Rate this answer

thumb\_up\_off\_alt

thumb\_down\_off\_alt

Responses may display inaccurate or offensive information that doesn't represent Google's views.

[Learn more](#)

## ASSISTANT

I understand that there is an issue with a specific value in the CSV file, '00a67483-ea0d-4bfa-a19b-17ae35e97fe3,' that cannot be directly converted to a float. You can handle this error gracefully by modifying the code as follows:

```
```python
import csv

import tensorflow as tf

from tensorflow import keras

from pydrive.auth import GoogleAuth
from pydrive.drive import GoogleDrive
```

```
# Define your deep learning model
```

```
model = keras.Sequential([

    # Add your layers here

])
```

```
# Read the CSV file
```

```
data = []
```

```
labels = []
```

with open('csv.txt', 'r') as file:

```
reader = csv.reader(file)
```

```
next(reader) # Skip the header row if present
```

```
for row in reader:
```

```
    try:
```

```
        # Assuming that the last column is the label and the rest are features
```

```
        features = [float(value) for value in row[:-1]]
```

```
        label = int(row[-1]) # Assuming it's a numerical label
```

```
        data.append(features)
```

```
        labels.append(label)
```

```
    except ValueError:
```

```
        print(f"Skipping row with non-convertible value: {row}")
```

```
# Convert lists to TensorFlow tensors
```

```
data = tf.convert_to_tensor(data, dtype=tf.float32)
```

```
labels = tf.convert_to_tensor(labels, dtype=tf.int64)
```

```
# Define your data preprocessing (e.g., normalization) here
```

```
# Compile and train your model
```

```
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

```
model.fit(data, labels, epochs=10, validation_split=0.2) # Adjust the number of epochs and  
validation split
```

```
# Make predictions using the trained model
```

```

predictions = model.predict(data)

# Save the trained model to a file
model.save('your_model.h5')

# Authorize and export the model to Google Drive
gauth = GoogleAuth()
gauth.LocalWebserverAuth() # Follow the authorization steps in the browser
drive = GoogleDrive(gauth)

# Create a Google Drive file
model_file = drive.CreateFile()
model_file.SetContentFile('your_model.h5')
model_file.Upload()

# Print the link to the model in Google Drive
print('Model uploaded: {}'.format(model_file['title']))
...

```

This modified code will skip rows with non-convertible values and continue processing the rest of the data.

USER

With the code to make played linked to my modern treasury and stripe pipe dream my API keys and everything already linked together. I'm trying to make a workflow but I need the nodejs script that link it all

ASSISTANT

Certainly, I can help you with that. To create a Node.js script that links your project with your modern treasury and Stripe API keys, you'll need to use the appropriate libraries and set up the necessary endpoints or functions to interact with these services. However, since you mentioned modern treasury and Stripe, and provided a user profile focused on quantum computing (#U language), it seems like there might be some confusion or a mismatch in the context.

To create a script that integrates with modern treasury and Stripe APIs, I would need more information about your specific requirements and the services you want to connect. Please provide more details about your project, and I'll be happy to assist you in creating the Node.js script for it.

USER

The code to do it

ASSISTANT

To create a Node.js script that integrates with Modern Treasury and Stripe APIs, you will need to use the appropriate libraries for these services and set up the necessary functionality. Below is a general outline of how you can approach this. Please note that this is a simplified example, and you may need to adapt it to your specific needs:

1. ****Set up your Node.js Project:****

First, make sure you have Node.js installed on your machine. You can create a new Node.js project using a tool like npm (Node Package Manager).

```
```bash

npm init

```
```

2. ****Install Dependencies:****

Install the necessary libraries for Modern Treasury and Stripe using npm.

```
```bash

npm install modern-treasury stripe

```
```

3. ****Create Your Node.js Script:****

Here's a simplified example of a Node.js script that links your project to Modern Treasury and Stripe:

```
```javascript

// Require the libraries

const ModernTreasury = require('modern-treasury');

const stripe = require('stripe')(process.env.STRIPE_SECRET_KEY);

// Set up your Modern Treasury API key

const modernTreasury = new ModernTreasury({ apiKey:
process.env.MODERN_TREASURY_API_KEY });

```
```



```
// Example: Create a payment in Stripe and record it in Modern Treasury

const createPayment = async () => {

  // Create a payment in Stripe

  const stripePayment = await stripe.paymentIntents.create({

    amount: 1000, // Amount in cents

    currency: 'usd',

    description: 'Example payment',

    payment_method_types: ['card'],

  });


  // Record the payment in Modern Treasury

  await modernTreasury.payments.create({

    external_id: stripePayment.id,

    amount: 10.00, // Amount in dollars

    currency: 'USD',

  });


  console.log('Payment created in Stripe and recorded in Modern Treasury');

};


createPayment();

...

```

4. ****Environment Variables:****

It's a good practice to store your API keys and sensitive information in environment variables. You

can use a library like `dotenv` to manage them.

```
```bash  

npm install dotenv

```
```

Create a `.env` file with your API keys:

```
```env  

MODERN_TREASURY_API_KEY=your_modern_treasury_api_key

STRIPE_SECRET_KEY=your_stripe_secret_key

```
```

Load environment variables in your script:

```
```javascript  

require('dotenv').config();

```
```

5. **Error Handling and More Features:**

This is a basic example. In a production script, you should implement proper error handling and additional features as needed.

6. **Run Your Script:**

Run your Node.js script using:

```
```bash  

node your_script.js

```
```

Remember to replace 'your_modern_treasury_api_key' and 'your_stripe_secret_key' with your actual API keys. This script is just a starting point, and you should adapt it to your specific use case and requirements.

USER

No, I want my played to make a link token and processing token so I need the code for played

ASSISTANT

I understand that you want to create Link and Processing Tokens for your Stripe integration. To achieve this in Node.js, you can use the Stripe API to generate these tokens. Here's how you can create a Link and Processing Token using Node.js and the Stripe library:

First, make sure you have the `stripe` library installed in your Node.js project:

```
```bash  

npm install stripe

```
```

Now, you can create a Node.js script to generate Link and Processing Tokens:

```
```javascript
```

```
const stripe = require('stripe')(process.env.STRIPE_SECRET_KEY);
```

```
// Create a Link Token
```

```
const createLinkToken = async () => {
 try {
 const linkToken = await stripe.terminal.createToken({
 type: 'link',
 });

 console.log('Link Token:', linkToken);
 } catch (error) {
 console.error('Error creating Link Token:', error);
 }
};
```

```
// Create a Processing Token
```

```
const createProcessingToken = async () => {
 try {
 const processingToken = await stripe.terminal.createToken({
 type: 'processing',
 });

 console.log('Processing Token:', processingToken);
 } catch (error) {
 console.error('Error creating Processing Token:', error);
 }
};
```

```
}
};

// Load your Stripe secret key from environment variables
require('dotenv').config();

// Call the functions to create tokens
createLinkToken();
createProcessingToken();
...
```

In this script:

1. It uses the `stripe` library to interact with the Stripe API.
2. `createLinkToken` and `createProcessingToken` functions are defined to create the Link and Processing Tokens, respectively.
3. These functions use the `stripe.terminal.createToken` method with the appropriate type.
4. Make sure to load your Stripe secret key from environment variables using `dotenv`.

To use this script, create a `.env` file with your Stripe secret key:

```
```env  
  
STRIPE_SECRET_KEY=your_stripe_secret_key  
...
```

Then, run the script using:

```
```bash
```

```
node your_script.js
```

```
```
```

This script will create Link and Processing Tokens using your Stripe account. Remember to replace 'your_stripe_secret_key' with your actual Stripe secret key.

```
## USER
```

```
Error      creating      Link      Token:      at      null.createLinkToken
(/tmp/___pdg___/dist/code/7c5963aa2774049cb62a187b2c968c2b232ebe23d0fae9f1a498152661c86
b9c/component.js:4:23)
                                                    at      Object.null
(/tmp/___pdg___/dist/code/7c5963aa2774049cb62a187b2c968c2b232ebe23d0fae9f1a498152661c86
b9c/component.js:31:1)
    at Module.Module._compile (node:internal/modules/cjs/loader:1256:14)
    at Object.Module._extensions..js (node:internal/modules/cjs/loader:1310:10)
    at Module.Module.load (node:internal/modules/cjs/loader:1119:32)
    at Function.Module._load (node:internal/modules/cjs/loader:960:12)
    at ModuleWrap.null (node:internal/modules/esm/translators:169:29)
    at ModuleJob.run (node:internal/modules/esm/module_job:194:25)
```

```
## ASSISTANT
```

The error message you provided seems to indicate that there was an issue when attempting to

create a Link Token using the Stripe library. To resolve this issue, please check the following:

1. **Stripe Secret Key**: Ensure that you've set the correct Stripe Secret Key in your environment variables. Double-check that there are no typos or extra spaces.
2. **Stripe API Version**: Make sure you are using the correct API version in your Stripe account. Stripe updates their API versions, and it's essential to use the latest version or one that is compatible with your integration.
3. **Stripe Account Configuration**: Verify that your Stripe account is properly set up to use Terminal and generate tokens. You might need to enable specific features or configure your account accordingly.
4. **Network and Firewall Issues**: If you're running this code in an environment with restricted network access or behind a firewall, make sure it allows outgoing connections to the Stripe API endpoints.
5. **Library Version**: Ensure you have the latest version of the `stripe` library installed. You can update it using `npm`:

```
```bash
npm update stripe
```
```

6. **Error Handling**: It's a good practice to implement proper error handling in your code. You can catch and log specific error messages to help with troubleshooting.

Here's an example of how you can improve error handling in your script:

```
```javascript
const createLinkToken = async () => {
 try {
 const linkToken = await stripe.terminal.createToken({
 type: 'link',
 });

 console.log('Link Token:', linkToken);
 } catch (error) {
 console.error('Error creating Link Token:', error.message);
 }
};
```
```

By logging the error message, you may get more specific information about what went wrong. If the issue persists, you may need to contact Stripe support for further assistance, as they can provide insights into specific account-related issues.

USER

Inputs

Logs

Details

steps.code

{1}

Copied

Copy Path

?

Copied

Copy Value

\$return_value

[10]

Copied

Copy Path

?

Copied

Copy Value

0

{6}

Copied

Copy Path

?

Copied

Copy Value

country_codes

[1]

Copied

Copy Path

?

Copied

Copy Value

0:

US

Copied

Copy Path

?

Copied

Copy Value

institution_id:

ins_130958

Copied

Copy Path

?

Copied

Copy Value

name:

Bridge

Copied

Copy Path

?

Copied

Copy Value

oauth:

false

Copied

Copy Path

?

Copied

Copy Value

products

[7]

Copied

Copy Path

?

Copied

Copy Value

0:

assets

Copied

Copy Path

?

Copied

Copy Value

1:

auth

Copied

Copy Path

?

Copied

Copy Value

2:

balance

Copied

Copy Path

?

Copied

Copy Value

3:

transactions

Copied

Copy Path

?

Copied

Copy Value

4:

signal

Copied

Copy Path

?

Copied

Copy Value

5:

identity

Copied

Copy Path

?

Copied

Copy Value

6:

recurring_transactions

Copied

Copy Path

?

Copied

Copy Value

routing_numbers

[1]

Copied

Copy Path

?

Copied

Copy Value

0:

051402372

Copied

Copy Path

?

Copied

Copy Value

1

{6}

Copied

Copy Path

?

Copied

Copy Value

country_codes

[1]

Copied

Copy Path

?

Copied

Copy Value

institution_id:

ins_130347

Copied

Copy Path

?

Copied

Copy Value

name:

Chiphone Credit Union

Copied

Copy Path

?

Copied

Copy Value

oauth:

false

Copied

Copy Path

?

Copied

Copy Value

products

[5]

Copied

Copy Path

?

Copied

Copy Value

0:

assets

Copied

Copy Path

?

Copied

Copy Value

1:

balance

Copied

Copy Path

?

Copied

Copy Value

2:

transactions

Copied

Copy Path

?

Copied

Copy Value

3:

signal

Copied

Copy Path

?

Copied

Copy Value

4:

recurring_transactions

Copied

Copy Path

?

Copied

Copy Value

routing_numbers

[0]

Copied

Copy Path

?

Copied

Copy Value

2

{6}

Copied

Copy Path

?

Copied

Copy Value

country_codes

[1]

Copied

Copy Path

?

Copied

Copy Value

0:

US

Copied

Copy Path

?

Copied

Copy Value

institution_id:

ins_130358

Copied

Copy Path

?

Copied

Copy Value

name:

Equitable Bank

Copied

Copy Path

?

Copied

Copy Value

oauth:

true

Copied

Copy Path

?

Copied

Copy Value

products

[8]

Copied

Copy Path

?

Copied

Copy Value

0:

assets

Copied

Copy Path

?

Copied

Copy Value

1:

auth

Copied

Copy Path

?

Copied

Copy Value

2:

balance

Copied

Copy Path

?

Copied

Copy Value

3:

transactions

Copied

Copy Path

?

Copied

Copy Value

4:

signal

Copied

Copy Path

?

Copied

Copy Value

5:

identity

Copied

Copy Path

?

Copied

Copy Value

6:

liabilities

Copied

Copy Path

?

Copied

Copy Value

7:

recurring_transactions

Copied

Copy Path

?

Copied

Copy Value

routing_numbers

[1]

Copied

Copy Path

?

Copied

Copy Value

0:

304971932

Copied

Copy Path

?

Copied

Copy Value

3

{6}

Copied

Copy Path

?

Copied

Copy Value

country_codes

[1]

Copied

Copy Path

?

Copied

Copy Value

0:

US

Copied

Copy Path

?

Copied

Copy Value

institution_id:

ins_130332

Copied

Copy Path

?

Copied

Copy Value

name:

First Community Bank & Trust (IL) - Personal

Copied

Copy Path

?

Copied

Copy Value

oauth:

false

Copied

Copy Path

?

Copied

Copy Value

products

[6]

Copied

Copy Path

?

Copied

Copy Value

0:

assets

Copied

Copy Path

?

Copied

Copy Value

1:

balance

Copied

Copy Path

?

Copied

Copy Value

2:

transactions

Copied

Copy Path

?

Copied

Copy Value

3:

signal

Copied

Copy Path

?

Copied

Copy Value

4:

identity

Copied

Copy Path

?

Copied

Copy Value

5:

recurring_transactions

Copied

Copy Path

?

Copied

Copy Value

routing_numbers

[0]

Copied

Copy Path

?

Copied

Copy Value

4

{6}

Copied

Copy Path

?

Copied

Copy Value

country_codes

[1]

Copied

Copy Path

?

Copied

Copy Value

0:

US

Copied

Copy Path

?

Copied

Copy Value

institution_id:

ins_129736

Copied

Copy Path

?

Copied

Copy Value

name:

First National Bank of Pana

Copied

Copy Path

?

Copied

Copy Value

oauth:

false

Copied

Copy Path

?

Copied

Copy Value

products

[6]

Copied

Copy Path

?

Copied

Copy Value

0:

assets

Copied

Copy Path

?

Copied

Copy Value

1:

balance

Copied

Copy Path

?

Copied

Copy Value

2:

transactions

Copied

Copy Path

?

Copied

Copy Value

3:

signal

Copied

Copy Path

?

Copied

Copy Value

4:

identity

Copied

Copy Path

?

Copied

Copy Value

5:

recurring_transactions

Copied

Copy Path

?

Copied

Copy Value

routing_numbers

[0]

Copied

Copy Path

?

Copied

Copy Value

5

{6}

Copied

Copy Path

?

Copied

Copy Value

country_codes

[1]

Copied

Copy Path

?

Copied

Copy Value

0:

US

Copied

Copy Path

?

Copied

Copy Value

institution_id:

ins_130251

Copied

Copy Path

?

Copied

Copy Value

name:

Greensboro Municipal Credit Union

Copied

Copy Path

?

Copied

Copy Value

oauth:

false

Copied

Copy Path

?

Copied

Copy Value

products

[4]

Copied

Copy Path

?

Copied

Copy Value

0:

assets

Copied

Copy Path

?

Copied

Copy Value

1:

balance

Copied

Copy Path

?

Copied

Copy Value

2:

transactions

Copied

Copy Path

?

Copied

Copy Value

3:

signal

Copied

Copy Path

?

Copied

Copy Value

routing_numbers

[0]

Copied

Copy Path

?

Copied

Copy Value

6

{6}

Copied

Copy Path

?

Copied

Copy Value

country_codes

[1]

Copied

Copy Path

?

Copied

Copy Value

0:

US

Copied

Copy Path

?

Copied

Copy Value

institution_id:

ins_130088

Copied

Copy Path

?

Copied

Copy Value

name:

GW Wade - MyStreetscape

Copied

Copy Path

?

Copied

Copy Value

oauth:

false

Copied

Copy Path

?

Copied

Copy Value

products

[4]

Copied

Copy Path

?

Copied

Copy Value

routing_numbers

[0]

Copied

Copy Path

?

Copied

Copy Value

7

{6}

Copied

Copy Path

?

Copied

Copy Value

country_codes

[1]

Copied

Copy Path

?

Copied

Copy Value

institution_id:

ins_130479

Copied

Copy Path

?

Copied

Copy Value

name:

L.C.E. Credit Union

Copied

Copy Path

?

Copied

Copy Value

oauth:

false

Copied

Copy Path

?

Copied

Copy Value

products

[1]

Copied

Copy Path

?

Copied

Copy Value

routing_numbers

[0]

Copied

Copy Path

?

Copied

Copy Value

8

{6}

Copied

Copy Path

?

Copied

Copy Value

country_codes

[1]

Copied

Copy Path

?

Copied

Copy Value

institution_id:

ins_129920

Copied

Copy Path

?

Copied

Copy Value

name:

National Bank of St. Anne

Copied

Copy Path

?

Copied

Copy Value

oauth:

false

Copied

Copy Path

?

Copied

Copy Value

products

[5]

Copied

Copy Path

?

Copied

Copy Value

0:

assets

Copied

Copy Path

?

Copied

Copy Value

1:

auth

Copied

Copy Path

?

Copied

Copy Value

2:

balance

Copied

Copy Path

?

Copied

Copy Value

3:

transactions

Copied

Copy Path

?

Copied

Copy Value

4:

identity

Copied

Copy Path

?

Copied

Copy Value

routing_numbers

[1]

Copied

Copy Path

?

Copied

Copy Value

0:

071922337

Copied

Copy Path

?

Copied

Copy Value

9

{6}

Copied

Copy Path

?

Copied

Copy Value

country_codes

[1]

Copied

Copy Path

?

Copied

Copy Value

0:

US

Copied

Copy Path

?

Copied

Copy Value

institution_id:

ins_130242

Copied

Copy Path

?

Copied

Copy Value

name:

Ocala Community Credit Union

Copied

Copy Path

?

Copied

Copy Value

oauth:

false

Copied

Copy Path

?

Copied

Copy Value

products

[4]

Copied

Copy Path

?

Copied

Copy Value

0:

assets

Copied

Copy Path

?

Copied

Copy Value

1:

balance

Copied

Copy Path

?

Copied

Copy Value

2:

transactions

Copied

Copy Path

?

Copied

Copy Value

3:

signal

Copied

Copy Path

?

Copied

Copy Value

routing_numbers

[0]

Copied

Copy Path

?

Copied

Copy Value

ASSISTANT

It appears that you've provided a set of data related to institution information, including country codes, institution IDs, names, and OAuth settings. However, it's not clear what specific assistance or information you're seeking based on this data.

If you have a specific question or request related to this data or if you need help with a particular

task or code related to this information, please provide more context or let me know how I can assist you further.

USER

```
this.plaid.$auth.environment
```

```
this.plaid.$auth.client_id
```

Make script to get info

```
this.plaid.$auth.client_secret
```

ASSISTANT

It seems like you want to access information related to Plaid authentication using a script. To do that, you need to interact with the Plaid API, which typically involves sending HTTP requests. However, I don't have information about the specific authentication mechanism or endpoints for Plaid, as it's not mentioned in the provided context.

To create a script to access Plaid information, you'll generally need to:

1. **Get API Credentials**: Obtain your client ID and client secret from your Plaid account.
2. **Choose an HTTP Library**: You'll need an HTTP library in your preferred programming language to send requests to Plaid's API. Common options include `requests` in Python, `axios` in JavaScript, or `HttpClient` in C#.
3. **Send an HTTP POST Request**: Use the library to send a POST request to the appropriate Plaid API endpoint. You may need to include your client ID and secret in the request headers.

4. ****Handle the Response****: Once you get a response from Plaid's API, you'll need to parse the data and handle it according to your needs.

Here's a simplified example in Python using the `requests` library:

```
```python
import requests

client_id = 'your_client_id'

client_secret = 'your_client_secret'

access_token = 'the_access_token_you_want_to_use'

Set the API endpoint
url = 'https://api.plaid.com/some_endpoint'

Set headers with client ID and secret
headers = {
 'client_id': client_id,
 'client_secret': client_secret
}

Send a POST request
response = requests.post(url, headers=headers)

Handle the response data
```

```
if response.status_code == 200:

 data = response.json()

 # Do something with the data

else:

 print('Error:', response.status_code)

...

```

Please replace ``your_client_id``, ``your_client_secret``, and ``the_access_token_you_want_to_use`` with your actual Plaid credentials and the specific endpoint you want to access. Make sure to refer to Plaid's API documentation for the exact endpoints and authentication requirements.

## ## USER

```
this.plaid.$auth.environment
```

```
this.plaid.$auth.client_id
```

Make script to get info

```
this.plaid.$auth.client_secret
```

## ## ASSISTANT

It seems you want to retrieve information related to Plaid's authentication environment, client ID, and client secret. However, the provided code snippet ``this.plaid.$auth.environment``, ``this.plaid.$auth.client_id``, and ``this.plaid.$auth.client_secret`` doesn't specify a programming language or context.

To access these values in a script, you would typically need to know the specific language or

environment in which this code is running. Assuming you are using JavaScript, here's how you can access these values:

```
```javascript

// Assuming you're working in a JavaScript environment

// Access the Plaid authentication environment
const environment = this.plaid.$auth.environment;

// Access the client ID
const clientId = this.plaid.$auth.client_id;

// Access the client secret
const clientSecret = this.plaid.$auth.client_secret;

// Now you can use these values as needed in your script
console.log('Environment:', environment);
console.log('Client ID:', clientId);
console.log('Client Secret:', clientSecret);
...
```
```

Replace the comments with the appropriate context or programming language you are using. If this code is used in a different environment or language, please provide more details so that I can offer more specific guidance.

## USER

```
{
 "id": "2abdb9d4-e3ca-46fd-aa2d-020e03da58ca",
 "object": "ledger_transaction",
 "live_mode": true,
 "external_id": "1eba4294-6659-4891-8de4-3924fc68ab69",
 "ledgerable_type": null,
 "ledgerable_id": null,
 "ledger_id": "68c4c234-8795-4fd8-bc78-11b72e0cde00",
 "description": "e45b7b58-8b61-4744-a030-5de25bdb8509",
 "status": "posted",
 "reverses_ledger_transaction_id": null,
 "ledger_entries": [
 {
 "id": "0f6d6123-94ad-4a77-afb1-16b8f10f8798",
 "object": "ledger_entry",
 "live_mode": true,
 "amount": 110624,
 "direction": "debit",
 "status": "posted",
 "ledger_account_id": "c14c9fd9-95b7-47ff-a35d-0d70e26002cd",
 "ledger_account_currency": "AUD",
 "ledger_account_currency_exponent": 2,
 "ledger_account_lock_version": null,
 "ledger_transaction_id": "2abdb9d4-e3ca-46fd-aa2d-020e03da58ca",
 "resulting_ledger_account_balances": null,
 "metadata": {},
 }
]
}
```

```
"discarded_at": null,

"created_at": "2022-02-01T11:54:49Z",

"updated_at": "2022-02-01T11:54:49Z"

},

{

 "id": "1046348f-90cf-40da-8968-ae24e4783584",

 "object": "ledger_entry",

 "live_mode": true,

 "amount": 110624,

 "direction": "debit",

 "status": "posted",

 "ledger_account_id": "9cdba50b-4ab0-47e4-a47c-399d52bdb98f",

 "ledger_account_currency": "AUD",

 "ledger_account_currency_exponent": 2,

 "ledger_account_lock_version": null,

 "ledger_transaction_id": "2abdb9d4-e3ca-46fd-aa2d-020e03da58ca",

 "resulting_ledger_account_balances": null,

 "metadata": {},

 "discarded_at": null,

 "created_at": "2022-02-01T11:54:49Z",

 "updated_at": "2022-02-01T11:54:49Z"

},

{

 "id": "6b45adc0-2129-468a-8187-b2f6a07c54db",

 "object": "ledger_entry",

 "live_mode": true,
```

```
"amount": 110624,
"direction": "credit",
"status": "posted",
"ledger_account_id": "1e9df6ae-0c72-4edc-a47f-418544aec172",
"ledger_account_currency": "AUD",
"ledger_account_currency_exponent": 2,
"ledger_account_lock_version": null,
"ledger_transaction_id": "2abdb9d4-e3ca-46fd-aa2d-020e03da58ca",
"resulting_ledger_account_balances": null,
"metadata": {},
"discarded_at": null,
"created_at": "2022-02-01T11:54:49Z",
"updated_at": "2022-02-01T11:54:49Z"
},
{
 "id": "d3d754f2-9331-43d0-bc20-60a350eaff93",
 "object": "ledger_entry",
 "live_mode": true,
 "amount": 110624,
 "direction": "credit",
 "status": "posted",
 "ledger_account_id": "927848f3-e54c-48c7-971e-4cf81d5a506e",
 "ledger_account_currency": "AUD",
 "ledger_account_currency_exponent": 2,
 "ledger_account_lock_version": null,
 "ledger_transaction_id": "2abdb9d4-e3ca-46fd-aa2d-020e03da58ca",
```



```
"resulting_ledger_account_balances": null,
"metadata": {},
"discarded_at": null,
"created_at": "2022-02-01T11:54:49Z",
"updated_at": "2022-02-01T11:54:49Z"
}
],
"posted_at": "2022-02-01T11:54:49Z",
"effective_at": "2022-01-31T00:00:00.000000Z",
"effective_date": "2022-01-31",
"metadata": {},
"created_at": "2022-02-01T11:54:49Z",
"updated_at": "2022-02-01T11:54:49Z"
}
```

## ASSISTANT

It seems you've provided information related to a ledger transaction. If you have any questions or need assistance regarding this ledger transaction or any other topic, please feel free to ask, and I'll be happy to help.

## USER

```
[

 {

 "id": "509d0faf-0367-47bf-92de-b6de751c8103",

 "object": "ledger_transaction",

 "live_mode": true,
```

```
"external_id": "e45b7b58-8b61-4744-a030-5de25bdb8509",
"ledgerable_type": null,
"ledgerable_id": null,
"ledger_id": "a0c359de-073d-4e9b-9127-9fb048f84a8b",
"description": "deposit from any company fbo james ocallaghan private bank's beneficiaries",
"status": "posted",
"reverses_ledger_transaction_id": null,
"ledger_entries": [
 {
 "id": "8b6b0f06-7352-4f08-805f-e5fb85328150",
 "object": "ledger_entry",
 "live_mode": true,
 "amount": 1000000000000000,
 "direction": "debit",
 "status": "posted",
 "ledger_account_id": "926e212b-4a8a-4094-97c4-087da7f73989",
 "ledger_account_currency": "USD",
 "ledger_account_currency_exponent": 2,
 "ledger_account_lock_version": null,
 "ledger_transaction_id": "509d0faf-0367-47bf-92de-b6de751c8103",
 "resulting_ledger_account_balances": null,
 "metadata": {},
 "discarded_at": null,
 "created_at": "2022-01-21T17:10:44Z",
 "updated_at": "2022-01-21T17:10:44Z"
 },
]
```

```
{
 "id": "e7401898-4d10-43af-9651-378a5a664f17",
 "object": "ledger_entry",
 "live_mode": true,
 "amount": 1000000000000000,
 "direction": "credit",
 "status": "posted",
 "ledger_account_id": "58567d46-86c1-4d71-826a-0738487edda8",
 "ledger_account_currency": "USD",
 "ledger_account_currency_exponent": 2,
 "ledger_account_lock_version": null,
 "ledger_transaction_id": "509d0faf-0367-47bf-92de-b6de751c8103",
 "resulting_ledger_account_balances": null,
 "metadata": {},
 "discarded_at": null,
 "created_at": "2022-01-21T17:10:44Z",
 "updated_at": "2022-01-21T17:10:44Z"
},
{
 "posted_at": "2022-01-21T17:10:44Z",
 "effective_at": "1808-04-30T00:00:00.000000Z",
 "effective_date": "1808-04-30",
 "metadata": {},
 "created_at": "2022-01-21T17:10:44Z",
 "updated_at": "2022-01-21T17:10:44Z"
},
```

```
{
 "id": "c62f926c-a668-4a38-b988-a446e2ce4fee",
 "object": "ledger_transaction",
 "live_mode": true,
 "external_id": "2021 Denali Ferman Buick GMC 24252 FL-54, Lutz, Florida, 33559",
 "ledgerable_type": null,
 "ledgerable_id": null,
 "ledger_id": "a0c359de-073d-4e9b-9127-9fb048f84a8b",
 "description": "\"Audi R8 Audi Wesley Chapel 2500 Lajuana Blvd wesley chapel, florida, 33543\"",
 "status": "posted",
 "reverses_ledger_transaction_id": null,
 "ledger_entries": [
 {
 "id": "b52a1d59-3637-4b30-8b07-f256ee2154ff",
 "object": "ledger_entry",
 "live_mode": true,
 "amount": 8600000,
 "direction": "credit",
 "status": "posted",
 "ledger_account_id": "decaf840-68e1-4f5f-a525-1c5792385f79",
 "ledger_account_currency": "USD",
 "ledger_account_currency_exponent": 2,
 "ledger_account_lock_version": null,
 "ledger_transaction_id": "c62f926c-a668-4a38-b988-a446e2ce4fee",
 "resulting_ledger_account_balances": null,
 "metadata": {},
 }
]
}
```

```
"discarded_at": null,

"created_at": "2022-01-21T16:59:53Z",

"updated_at": "2022-01-21T16:59:53Z"

},

{

 "id": "f38b000c-6585-449c-a4fe-df46f09dc979",

 "object": "ledger_entry",

 "live_mode": true,

 "amount": 8600000,

 "direction": "debit",

 "status": "posted",

 "ledger_account_id": "926e212b-4a8a-4094-97c4-087da7f73989",

 "ledger_account_currency": "USD",

 "ledger_account_currency_exponent": 2,

 "ledger_account_lock_version": null,

 "ledger_transaction_id": "c62f926c-a668-4a38-b988-a446e2ce4fee",

 "resulting_ledger_account_balances": null,

 "metadata": {},

 "discarded_at": null,

 "created_at": "2022-01-21T16:59:53Z",

 "updated_at": "2022-01-21T16:59:53Z"

}

],

"posted_at": "2022-01-21T16:59:53Z",

"effective_at": "2021-01-14T00:00:00.000000Z",

"effective_date": "2021-01-14",
```

```
"metadata": {},

"created_at": "2022-01-21T16:59:53Z",

"updated_at": "2022-01-21T16:59:53Z"

},

{

 "id": "0832b1af-b96a-490f-b67e-70dab4ee3c74",

 "object": "ledger_transaction",

 "live_mode": true,

 "external_id": "journal 56 co 6 Account 1190,1302. Amount 152,755.05, -152755.05 . Control
accoun 76833, 7221",

 "ledgerable_type": null,

 "ledgerable_id": null,

 "ledger_id": "a0c359de-073d-4e9b-9127-9fb048f84a8b",

 "description": "\"Audi R8 Audi Wesley Chapel 2500 Lajuana Blvd wesley chapel, florida, 33543",

 "status": "posted",

 "reverses_ledger_transaction_id": null,

 "ledger_entries": [

 {

 "id": "1933aae1-e670-42e0-ae86-c0102f321ac0",

 "object": "ledger_entry",

 "live_mode": true,

 "amount": 15275505,

 "direction": "debit",

 "status": "posted",

 "ledger_account_id": "926e212b-4a8a-4094-97c4-087da7f73989",

 "ledger_account_currency": "USD",
```

```
"ledger_account_currency_exponent": 2,

"ledger_account_lock_version": null,

"ledger_transaction_id": "0832b1af-b96a-490f-b67e-70dab4ee3c74",

"resulting_ledger_account_balances": null,

"metadata": {},

"discarded_at": null,

"created_at": "2022-01-21T16:53:52Z",

"updated_at": "2022-01-21T16:53:52Z"

},

{

 "id": "4ea444c5-0020-445f-b5f8-c25dda7de422",

 "object": "ledger_entry",

 "live_mode": true,

 "amount": 15275505,

 "direction": "credit",

 "status": "posted",

 "ledger_account_id": "decaf840-68e1-4f5f-a525-1c5792385f79",

 "ledger_account_currency": "USD",

 "ledger_account_currency_exponent": 2,

 "ledger_account_lock_version": null,

 "ledger_transaction_id": "0832b1af-b96a-490f-b67e-70dab4ee3c74",

 "resulting_ledger_account_balances": null,

 "metadata": {},

 "discarded_at": null,

 "created_at": "2022-01-21T16:53:52Z",

 "updated_at": "2022-01-21T16:53:52Z"
```

```
}

],

"posted_at": "2022-01-21T16:53:52Z",

"effective_at": "2021-01-21T00:00:00.000000Z",

"effective_date": "2021-01-21",

"metadata": {},

"created_at": "2022-01-21T16:53:52Z",

"updated_at": "2022-01-21T16:53:52Z"

},

{

 "id": "c3735ad3-ee67-4688-a266-081ba93b996d",

 "object": "ledger_transaction",

 "live_mode": true,

 "external_id": "\"Audi R8 Audi Wesley Chapel 2500 Lajuana Blvd wesley chapel, florida, 33543\"",

 "ledgerable_type": null,

 "ledgerable_id": null,

 "ledger_id": "a0c359de-073d-4e9b-9127-9fb048f84a8b",

 "description": "\"Audi R8 Audi Wesley Chapel 2500 Lajuana Blvd wesley chapel, florida, 33543\"",

 "status": "posted",

 "reverses_ledger_transaction_id": null,

 "ledger_entries": [

 {

 "id": "87a75fd8-d089-4a17-a6a7-04e5130fa4f5",

 "object": "ledger_entry",

 "live_mode": true,

 "amount": 21000000,
```



```
"direction": "debit",

"status": "posted",

"ledger_account_id": "926e212b-4a8a-4094-97c4-087da7f73989",

"ledger_account_currency": "USD",

"ledger_account_currency_exponent": 2,

"ledger_account_lock_version": null,

"ledger_transaction_id": "c3735ad3-ee67-4688-a266-081ba93b996d",

"resulting_ledger_account_balances": null,

"metadata": {},

"discarded_at": null,

"created_at": "2022-01-21T16:49:40Z",

"updated_at": "2022-01-21T16:49:40Z"

},

{

 "id": "c6f56bbc-4f39-4e4d-8fff-bf798610e15e",

 "object": "ledger_entry",

 "live_mode": true,

 "amount": 21000000,

 "direction": "credit",

 "status": "posted",

 "ledger_account_id": "decaf840-68e1-4f5f-a525-1c5792385f79",

 "ledger_account_currency": "USD",

 "ledger_account_currency_exponent": 2,

 "ledger_account_lock_version": null,

 "ledger_transaction_id": "c3735ad3-ee67-4688-a266-081ba93b996d",

 "resulting_ledger_account_balances": null,
```

```
"metadata": {},
"discarded_at": null,
"created_at": "2022-01-21T16:49:40Z",
"updated_at": "2022-01-21T16:49:40Z"
}
],
"posted_at": "2022-01-21T16:49:40Z",
"effective_at": "2021-01-21T00:00:00.000000Z",
"effective_date": "2021-01-21",
"metadata": {},
"created_at": "2022-01-21T16:49:40Z",
"updated_at": "2022-01-21T16:49:40Z"
},
{
"id": "b70eb385-0f8f-40be-8adf-5f7f64f551f2",
"object": "ledger_transaction",
"live_mode": true,
"external_id": "a0c359de-073d-4e9b-9127-9fb048f84a8",
"ledgerable_type": null,
"ledgerable_id": null,
"ledger_id": "a0c359de-073d-4e9b-9127-9fb048f84a8b",
"description": "Audi R8 Audi Wesley Chapel 2500 Lajuana Blvd wesley chapel, florida, 33543",
"status": "posted",
"reverses_ledger_transaction_id": null,
"ledger_entries": [
{
```

"id": "5810da8c-3b3d-4a26-bc7a-062209bf4402",  
"object": "ledger\_entry",  
"live\_mode": true,  
"amount": 152755,  
"direction": "debit",  
"status": "posted",  
"ledger\_account\_id": "926e212b-4a8a-4094-97c4-087da7f73989",  
"ledger\_account\_currency": "USD",  
"ledger\_account\_currency\_exponent": 2,  
"ledger\_account\_lock\_version": null,  
"ledger\_transaction\_id": "b70eb385-0f8f-40be-8adf-5f7f64f551f2",  
"resulting\_ledger\_account\_balances": null,  
"metadata": {},