Bing

Sure, I can help you with that. Here's a high-level overview of how you might approach this task:

1. **Extract Data**: Use the `Get-MgDevice` command to extract device information. This will return a list of devices and their properties.

```
$devices = Get-MgDevice
```

1. **Map Fields**: Define a mapping between the fields returned by `Get-MgDevice` and the fields in your `cmdb` table. This could be a hashtable where the keys are the `Get-MgDevice` field names and the values are the corresponding `cmdb` field names.

```
$fieldMapping = @{
    "DeviceId" = "device_id";
    "DeviceName" = "device_name";
    # Add more mappings as needed
}
```

1. **Transform Data**: For each device, create a new object where the properties align with the `cmdb` table fields. Use the `$fieldMapping` hashtable to achieve this.

```
$transformedDevices = $devices | ForEach-Object {
    $device = $_
    $transformedDevice = New-Object PSObject
    $fieldMapping.GetEnumerator() | ForEach-Object {
        $transformedDevice | Add-Member -NotePropertyName $_.Value -NotePropertyValue $device.($_.Key)
    }
    $transformedDevice
}
```

1. **Store Data**: Connect to your MySQL database and insert the transformed data into the `cmdb` table. You can use the `MySql.Data` PowerShell module for this.

```
$connectionString = "Server=myServerAddress;Database=myDataBase;Uid=myUsername;Pwd=myPassword;"
$conn = New-Object MySql.Data.MySqlClient.MySqlConnection
$conn.ConnectionString = $connectionString
$conn.Open()

$transformedDevices | ForEach-Object {
    $device = $_
    $command = $conn.CreateCommand()
    $command.CommandText = "INSERT INTO cmdb (device_id, device_name) VALUES (@deviceId, @deviceName)" # Update this based on your actual table structure
    $command.Parameters.AddWithValue("@deviceId", $device.device_id)
    $command.Parameters.AddWithValue("@deviceName", $device.device_name)
    $command.ExecuteNonQuery()
}

$conn.Close()
```

Please replace the placeholders with your actual server address, database name, username, and password. Also, update the SQL query based on your actual table structure and the fields you are interested in.

Remember to install the necessary modules and libraries before running the script. You can install the Microsoft Graph PowerShell Module using the command `Install-Module -Name MSGraph`. For the MySQL connection, you might need to install the `MySql.Data` module using the command `Install-Package -Name MySql.Data`.

This is a basic script and might need to be adjusted based on your specific requirements and environment. Always ensure to follow best practices for scripting and database management, including error handling and secure credential management. If you have any more questions or need further clarification, feel free to ask!.