# Work Portfolio

*contains no confidential information

## Contents



# Automation Scripts

## Consolidation of 10,000 cost center files, utilizing pandas and parallelization of CPU cores

```
import pandas as pd
import os
import time
import datetime
from joblib import Parallel, delayed

start = time.time()

paths = pd.read_csv("Path.csv")
user_name_str = 'jesse.curran'
```

```
df_lis = []

def process_file(row):
    folder = row["Path"][17:]
    entity = row["Path"].split("\\")[3][:4]
    file = f'{entity} Budget Metadata Review.xlsx'
    # Create the full path
    p = os.path.join('C:', os.sep, 'Users', user_name_str, 'Providence St. Joseph
Health', '2025 Budget - Documents', folder, file)
    #df = pd.read_excel(p, sheet_name="Update Log")
    try:
        df = pd.read_excel(p, sheet_name="6. New Cost Centers",
skiprows=range(5))
        #print("Sheet read successfully!")
    except FileNotFoundError:
        print(f"The {file} was not found. Please check the file path {p}.")
    except ValueError as e:
        print(f"ValueError: {e}. File: {file}, Path: {p}")
        #print("This error may occur if the 'Update Log' sheet does not exist.
Please check the sheet name.")
    except Exception as e:
        print(f"An error occurred: {e}")

    df["Path"] = p[17:].split("\\")[-1]
    return df


df_lis = Parallel(n_jobs=-1, verbose=10)(delayed(process_file)(row) for _, row in
paths.iterrows())

cons_df = pd.concat(df_lis)

S_File = 'Admin\PBI\LOADS\Consolidated_New_CC.csv'
s_path = os.path.join('C:', os.sep, 'Users', user_name_str, 'Providence St.
Joseph Health', '2025 Budget - Documents', S_File)
cons_df.to_csv(s_path, index=False)

stop = time.time()

file = open(fr'C:\Users\{user_name_str}\Providence St. Joseph Health\2025 Budget
- Documents\Test\Consolidation_Scripts\running_log.txt', 'a')

file.write(f'{datetime.datetime.now()} - The UPDATE LOG script ran in {(stop -
start):.02f}\n')
```

```
file.close()
```

## Compare.py script to find daily changes in SharePoint permissions

```
import pandas as pd

# Load the two Excel files
df1 =
pd.read_csv('Daily_Accountibility_Matrix_Access\CC_User_Add\Today_Accountability_
CC_Output.csv')
df2 =
pd.read_csv('Daily_Accountibility_Matrix_Access\CC_User_Add\Yesterday_Accountabil
ity_CC_Output.csv')

# Ensure that the dataframes are sorted in the same way
df1.sort_values(by=list(df1.columns), inplace=True)
df2.sort_values(by=list(df2.columns), inplace=True)

df2["rm"] = "rm"


# Find the rows which are different between the dataframes

df1 = df1.drop_duplicates()
df2 = df2.drop_duplicates()

diff =
pd.concat([df1,df2]).drop_duplicates(subset=['Division','Region','Entity','Minist
ry','Permission_Type','User','Full_Path'], keep=False)

#Rename and columns and create output files
diff = diff.rename(columns={"Full_Path": "Path", "User": "Email", "Division":
"Division"})
diff = diff.fillna("")
diff[diff['rm'] == ""].loc[:,["Path", "Email",
"Division"]].to_csv('Daily_Accountibility_Matrix_Access\CC_User_Add\daily_matrix_
add.csv', index=False)
diff[diff['rm'] != ""].loc[:,["Path", "Email", "Division",
"rm"]].to_csv('Daily_Accountibility_Matrix_Access\CC_User_Add\daily_matrix_rm.csv
', index=False)
```

## Update 350 Budget File Instructions

```
import os
import xlwings as xw
import pandas as pd
import time

def write_ex(template_sheet, path):
    """
    details
    """
    # Define the path to the template and the target workbook
    target_path = path

    # Open both workbooks
    target_wb = xw.Book(target_path)

    # Define the sheets from the template and the target workbook
    target_sheet = target_wb.sheets['Instruction']

    # Copy over the sheet instructions in specified cell range

    template_sheet.range('B7:K27').copy(target_sheet.range('B7:K27'))

    # Save and close the target workbook
    target_wb.save()
    target_wb.close()


def main():
    """
    Drive instruction update function write_ex.
    Loops through each file path and updates the instructions.
    """
    # open template instructions, pass into loop to copy/paste
    template_path = 'Template Pre-Budget Workforce Tagging.xlsx'
    template_wb = xw.Book(template_path)
    template_sheet = template_wb.sheets['Instruction']

    # update user name as needed
    user_name_str = 'ahsin.saleem'

    # read path of files to copy/paste
    paths = pd.read_csv("Path.csv")
    paths = paths[paths["Path"].str.contains("Texas")]
```

```
    for index, row in paths.iterrows():
        folder = row["Path"][17:]
        entity = row["Path"].split("\\")[3][:4]
        file = f'{entity} Pre-Budget Workforce Tagging.xlsx'
        p = os.path.join('C:', os.sep, 'Users', user_name_str,
                         'Providence St. Joseph Health', '2025 Budget -
Documents', folder, file)
        print(p)
        write_ex(template_sheet, p)

    template_wb.app.quit()

start = time.time()
main()
stop = time.time()
print(f'The program ran in {stop - start:.02f} seconds')
```

## Make Path.csv using prior excel ledger of Division, Region, Entity

```
import os
import pandas as pd

df = pd.read_excel("Python_scripts\Entity Listing_Model Flag.xlsx")

def create_path(directory, folder_name):
    path = os.path.join(directory, folder_name)
    return f"Shared Documents{path[23:]}"


directory = f"2025 Budget - Documents"

Region_dic = (df.groupby("Division")["Region"].apply(set)).to_dict()

entity_dic = (df.groupby("Region")["ENTITY DESCRIPTION"].apply(set)).to_dict()
path = []

for divis, regions in Region_dic.items():
    path.append(create_path(directory, divis))
    directory1 = f"2025 Budget - Documents\{divis}"
    for region in regions:
        path.append(create_path(directory1, region))
        directory2 = f"2025 Budget - Documents\{divis}\{region}"
        for entity in entity_dic[region]:
            path.append(create_path(directory2, entity))
```

```
# Create a DataFrame from the list of paths
df_paths = pd.DataFrame(path, columns=['Path'])

# Write the DataFrame to a CSV file
df_paths.to_csv("Python_scripts\Path.csv", index=False)
```

## 1-Extract SharePoint Folder Permissions

```
#Config Variables
$SiteURL = "https://providence4.sharepoint.com/sites/2025Budget"
$CSVFile = "User_extract\all_current_matrix_permissions.csv"
$FolderPathFile = "User_extract\Path.csv"

#Connect to PnP Online
Connect-PnPOnline -Url $SiteURL -UseWebLogin

#Import folder paths from the CSV file
$IncludeFolders = Import-Csv -Path $FolderPathFile | ForEach-Object { $_.Path }

#Create an array to hold the output
$Output = @()

#Loop through each folder
ForEach($FolderURL in $IncludeFolders) {
    #Get the folder
    try {
        $Folder = Get-PnPFolder -Url $FolderURL -Includes
ListItemAllFields.RoleAssignments
    }
    catch {
        Write-Host -f Red "Error processing folder '$($FolderURL)"
    }
    #Get the permissions for the folder
    $RoleAssignments = $Folder.ListItemAllFields.RoleAssignments

    #Loop through each role assignment
    ForEach($RoleAssignment in $RoleAssignments) {
        $RoleDefinitionBindings = Get-PnPProperty -ClientObject $RoleAssignment -
Property RoleDefinitionBindings
        $Member = Get-PnPProperty -ClientObject $RoleAssignment -Property Member

        #Add the permission to the output
        $Output += New-Object PSObject -Property @{
```

```
            "Path" = $FolderURL
            "User/Group" = $Member.Email
            "Permission" = $RoleDefinitionBindings.Name
        }
        Write-host -f Green "Extracted Permissions on Path '$($FolderURL)' to
'$($Member.Email)'"
    }
}

#Export the output to a CSV file
$Output | Export-Csv -Path $CSVFile -NoTypeInformation
```

## Clean extracted permissions in Jupyter notebook

```
#
import pandas as pd
import os
import openpyxl

# Purpose of this Python file is to clean the list of current permissions.
# We take all the current permissions, extracted from the PowerShell extract,
# then we split into divisions, filter out internal emails, compare the emails of
# today vs. yesterday, output those differences in xlsx.


#Grab all permissions and put into a dataframe
df = pd.read_csv('all_current_matrix_permissions.csv')
df.shape
df.head()

#
len(df["User/Group"].unique())

#
df.isnull().sum()

#
df = df.dropna()
```

```
df.head()

#
df.duplicated().sum()

#
central = df[df["Path"].str.contains('Central')]
central.head()


#
hcc = df[df["Path"].str.contains('HCC')]
hcc.head()

#
managed = df[df["Path"].str.contains('Managed')]
managed.head()

#
north = df[df["Path"].str.contains('North')]
north.head()

#
south = df[df["Path"].str.contains('South')]
south.head()

#
south.groupby("Path")["User/Group"].apply(list)
south.head()

#
#Filter out of our permission list of each region our internal emails (think core
FP&A + emails)
emails = “”
filter_lis = emails.split(";")
filter_lis = [i.strip().lower() for i in filter_lis]

#Create a region list
region = []

#
# updating now each division dataframe to not include internal emails.


filtered_rows = []
```

```
for index, row in central.iterrows():
    if row["User/Group"].lower() not in filter_lis:
        filtered_rows.append(row)

# central represents the df filtered for path central prior to this
central = pd.DataFrame(filtered_rows)
region.append(central)
central.head()


#
filtered_rows = []

for index, row in hcc.iterrows():
    if row["User/Group"].lower() not in filter_lis:
        filtered_rows.append(row)

hcc = pd.DataFrame(filtered_rows)
region.append(hcc)
hcc.head()

#
filtered_rows = []

for index, row in managed.iterrows():
    if row["User/Group"].lower() not in filter_lis:
        filtered_rows.append(row)

managed = pd.DataFrame(filtered_rows)
region.append(managed)
managed.head()

#
filtered_rows = []

for index, row in north.iterrows():
    if row["User/Group"].lower() not in filter_lis:
        filtered_rows.append(row)

north = pd.DataFrame(filtered_rows)
region.append(north)
north.head()

#
```

```
filtered_rows = []

for index, row in south.iterrows():
    if row["User/Group"].lower() not in filter_lis:
        filtered_rows.append(row)

south = pd.DataFrame(filtered_rows)
region.append(south)
south.head()

#
# setting up environment to compare matrix of yesterday to today and find emails
to update

try:
    os.remove("matrix_yesterday.xlsx")
except:
    print("Error in deleting yesterday file")
    #raise
try:
    os.rename("matrix_today.xlsx", "matrix_yesterday.xlsx")
except:
    print("Error in renaming file")
    #raise


wb = openpyxl.Workbook()
wb.save(filename='matrix_today.xlsx')


#
#create full dataframe of division sheet permissions, folder, emails
#matrix_today.xlsx will show all current matrix permissions
#--> QUESTION -> why are we using xlsx and not csv?
with pd.ExcelWriter('matrix_today.xlsx', engine='openpyxl', mode='a',
if_sheet_exists='replace') as writer:
    central.to_excel(writer, sheet_name="Central", index=False)
    managed.to_excel(writer, sheet_name='Managed', index=False)
    north.to_excel(writer, sheet_name="North", index=False)
    south.to_excel(writer, sheet_name="South", index=False)
    hcc.to_excel(writer, sheet_name="HCC", index=False)


#
# represents division dataframes of previous day, to compare
```

```
central_y = pd.read_excel("matrix_yesterday.xlsx", sheet_name="Central")
central_y["rm"] = "rm"
managed_y = pd.read_excel("matrix_yesterday.xlsx", sheet_name="Managed")
managed_y["rm"] = "rm"
north_y = pd.read_excel("matrix_yesterday.xlsx", sheet_name="North")
north_y["rm"] = "rm"
south_y = pd.read_excel("matrix_yesterday.xlsx", sheet_name="South")
south_y["rm"] = "rm"
hcc_y = pd.read_excel("matrix_yesterday.xlsx", sheet_name="HCC")
hcc_y["rm"] = "rm"


#
# combining all dataframes into one that only contains unique values (i.e. add /
delete)
lis = [central, central_y, managed, managed_y, north, north_y, south, south_y,
hcc, hcc_y]
for i in lis:
    i = i.drop_duplicates()
diff_df =
pd.concat(lis).drop_duplicates(subset=["Permission","Path","User/Group"]
,keep=False)
diff_df

#
#diff_central = pd.concat([central, central_y]).drop_duplicates(keep=False)

#

# here we are creating a new division column, first pointing to path column,
stripping out the Division from path,
# for each row instance using pandas apply(function)
# [1] is used to extract the second item of path column. ["shared docs",
"Central"]

# adding division column to the unique dataframe
diff_df["Divisons"] = diff_df["Path"].apply(lambda x: x.split("\\")[1])
team_df = diff_df[diff_df["rm"] != "rm"].loc[:,["Path","User/Group",
"Divisons"]].drop_duplicates()
diff_df = diff_df[diff_df["rm"] != "rm"].loc[:,["User/Group", "Divisons",
"rm"]].drop_duplicates()
diff_df = diff_df.rename(columns={"User/Group": "Email"})

#
diff_df.to_csv('daily_matrix_add.csv', index=False)
```

```
# Define the path to the directory where you want to save team_df
dir_path = os.path.join('..\\'*3, 'Admin', 'Accountability Matrix-2025 Ministry
Level')

filename_team = f'2025_Budget_Scrape.csv'

# Check if the directory exists
if not os.path.exists(dir_path):
    print(f"Directory does not exist: {dir_path}")
else:
    # Save the team_df DataFrame
    team_df.to_csv(os.path.join(dir_path, filename_team), index=False)

diff_df


#
t = df["Path"].apply(lambda x : x.split("\\")[-1][:4])
t=t.drop_duplicates()
paths = df["Path"]
paths = "Shared Documents\Admin\Volume Export Test\\" + t + "Global Volumes.xlsx"
paths

#
```

## Add the permissions to each folder in SharePoint using PowerShell

- See Compare.py for daily_matrix_add.csv

```
#Config Variables
$SiteURL = "https://providence4.sharepoint.com/sites/2025Budget"
Connect-PnPOnline -Url $SiteURL -UseWebLogin
$CSVFile = "User_extract\daily_matrix_add.csv"

Try {
    #Connect to PnP Online

    #Get the CSV file
    $CSVData = Import-CSV $CSVFile

    ForEach($Row in $CSVData) {
        Try {

            #Get the Folder
```

```
            #$Folder = Get-PnPFolder -Url $Row.Path -Includes
ListItemAllFields.ParentList -ErrorAction Stop
            #$List =  $Folder.ListItemAllFields.ParentList

            #Sharepoint site 'visitor' group access
            Add-PnPUserToGroup -LoginName $Row.Email -Identity $Row.Divisons -
ErrorAction Stop

            #Grant Permission to the Folder
            #Set-PnPFolderPermission -List $List -Identity
$Folder.ServerRelativeUrl -User $Row.Email -AddRole "Edit" -ErrorAction Stop
            Write-host -f Green "Ensured Permissions on Group '$($Row.Divisons)'
for '$($Row.Email)'"

        } Catch {
            Write-host "Error: in the for loop $($_.Exception.Message)" -
foregroundcolor Red
        }
    }
} Catch {
    write-host "Error: in the csv $($_.Exception.Message)" -foregroundcolor Red
}
```

# Finance DB Tool

## SQL Query -> Excel & PBI Insert, Pivot Tables and PBI Visuals.

```
SELECT A.*, B.*

FROM ADMINTECH_DB.DMS_FINANCE.VW_FACT_GENESIS as A

LEFT JOIN ADMINTECH_DB.DMS_FINANCE.VW_FACT_GENESIS_BUDGET as B

ON A.EDL = B.EDL

WHERE A.MONTH >= '2024-01-01'

AND A.ENTITY IN
('2700','2701','2702','2703','2704','2705','2709','2710','2711','2712','2713','2714','2715','2716','2717','2718','
2942');
```

# Office Scripts Automation - TypeScript

- Inserts a button on a worksheet that automates updating a long range of values, from one tab to another.

```
function main(workbook: ExcelScript.Workbook) {

  // grab driver tagging tab
  let driverTaggingTab = workbook.getWorksheet("3. Driver Tagging");

  // variable for active sheet
  let maintenanceTab = workbook.getWorksheet("5. Maintenance")

  // navigate to cell to paste values: col A range edge + down + one
  let pasteCell = driverTaggingTab.getRange("A7").getRangeEdge(ExcelScript.KeyboardDire
ction.down).getOffsetRange(1, 0);

  // copy values from Maintenance usedRange to Driver Tagging pasteCell col A
  pasteCell.copyFrom(maintenanceTab.getRange("F10").getExtendedRange(ExcelScript.Key
boardDirection.down),ExcelScript.RangeCopyType.values, false, false);

}
```