# **Slide Automation Tool Documentation**

## Slides Sections

* [Landscape](#landscape-section)
* [Pricing](#pricing-section)
* [PPA](#ppa-section)
* [Mix&Assortment](#mixassortment-section)
* [Promotion](#promotion-section)
* [Financials](#Financials-section)
* [Pricing\_CBC](#Pricing_CBC-section)
* [Innovation\_CBC](#Innovation_CBC-section)

## Landscape Section

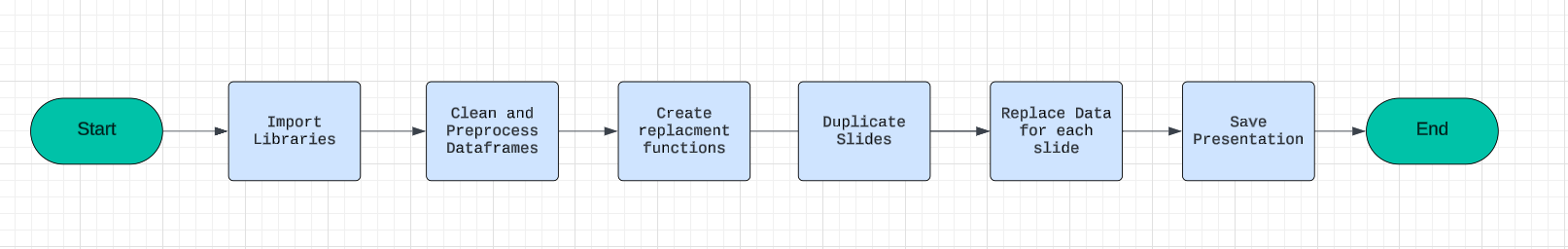
### Introduction

In the slide automation landscape: from13 slide base we create 5 decks

![1732020538845](image/Slidesdocumantion/1732020538845.png)

1. Landscape Market(trend-Concent-growth)&ValAvgPrice Slides:
   * Market Trends Analysis
   * Market Concentration
   * Market growth contributors
   * Value Sales & Avg Price
2. Landscape ShareGrowth Slides:
   * Share and Growth by Manufacturer/Brands
   * Share and Growth By Manufacturer
   * Momentum Analysis
3. Landscape Category(Trend-Overview)Slides :
   * Category Trends
   * Share Evolution index analysis
   * Category Overview
4. Landscape Output WO CB Slides:
   * Market Trends Analysis
   * Market Concentration
   * Share and Growth by Manufacturer/Brands
   * Share and Growth By Manufacturer
5. Landscape RPVM Slides:
   * Revenue by Price vs. Volume vs. Mix analysis

### Project Steps

* Project Flow
* 
* [Step 1: Import Libraries we use](#step-1-import-libraries-we-use)
* [Step 2: modified Data frames: cleaning and preprocessing the data frames](#step-2-modified-data-frames)
* [Step 3: Write Functions to Create Slides: Define functions to Automatically generate slides based on the base slides](#step-3-write-functions-to-create-slides)
* [Step 4: Duplicate Slides: Use functions or methods to duplicate existing slides as needed for the presentation.](#step-4-duplicate-slides)
* [Step 5: Replace Data in Slides: update information from the cleaned data frames to slides](#step-5-replace-data-in-slides)
* [Step 6: Save Presentation](#step-6-save-presentation)

#### [Step 1: Import Libraries we use](https://github.com/khaledSeifEleslam/Slide-Automate/blob/main/general_functions/generalFunctions.ipynb)

##### Ex: Libraries we use

* This script sets up an environment for working with PowerPoint presentations, data manipulation, filesystem operations, and COM (Component Object Model) object access.
* It imports necessary modules such as 'pptx' for PowerPoint automation, 'win32com' for COM object access and Windows automation, 'pandas' and 'numpy' for data manipulation,
* 'pathlib' for working with filesystem paths, 're' for regular expression operations, and various other modules for general-purpose tasks like file operations and timing functions.
* By importing these modules, the script prepares itself for tasks such as creating or modifying PowerPoint presentations, analyzing data using pandas and numpy, interacting
* with the Windows environment using win32com, and performing filesystem operations using shutil and os. Overall, this script provides a comprehensive setup for automating tasks
* related to PowerPoint presentations and general-purpose Python programming.

# Import necessary module for working with PowerPoint presentationsfrom pptx import Presentation # Import the win32com.client module, aliasing it as win32 for convenienceimport win32com.client as win32 # Import pandas for data manipulation and analysisimport pandas as pd # Import numpy for numerical computingimport numpy as np # Import the Path class from pathlib for working with filesystem pathsfrom pathlib import Path # Import re for regular expression operationsimport re # Import sys for access to interpreter-related functionsimport sys # Import time for various time-related functionsimport time # Assign win32.constants to a shorter alias win32c for easier access win32c = win32.constants # Import shutil for high-level file operationsimport shutil # Import os for operating system dependent functionalityimport os # Import win32com.client again for COM object and functions accessimport win32com.client # Import warnings for warning control functionalityimport warnings

#### [Step 2: modified Data frames](https://github.com/khaledSeifEleslam/Slide-Automate/blob/main/Landscape%20slide%20duplicate/Landscape%20duplicate.ipynb)

* This function takes a dictionary of dataframes and a category type as input.
* It iterates over each dataframe in the dictionary and performs cleaning operations,
* such as renaming columns, removing unwanted rows, converting data types, and
* separating totals from the main data. The cleaned dataframes and totals are
* stored in separate dictionaries. Finally, it returns two dictionaries:
* one containing cleaned sector segment data and the other containing totals.

##### Parameters:

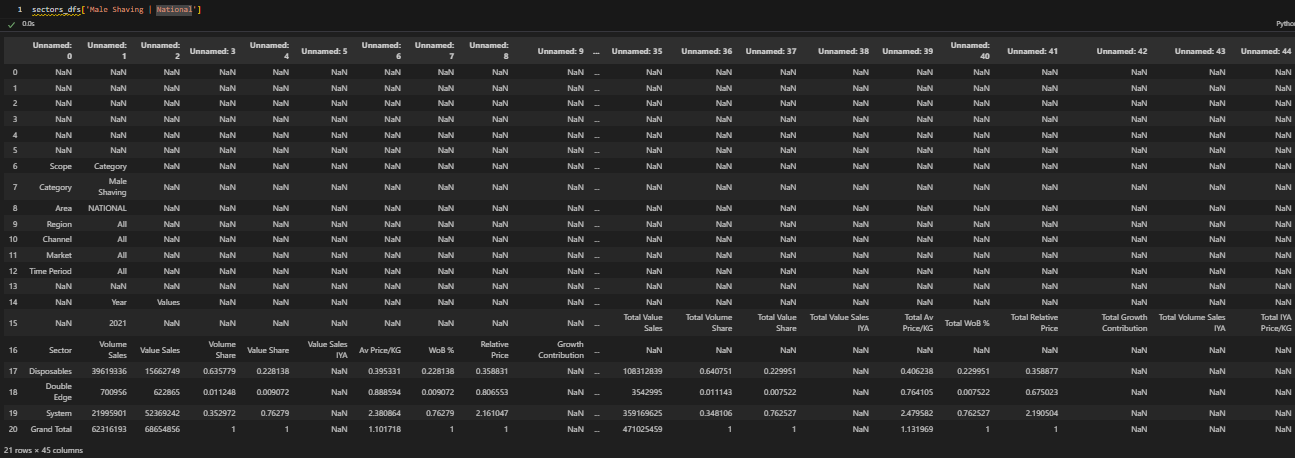
* + inputdic: A dictionary of dataframes where each dataframe represents data.

##### Returns:

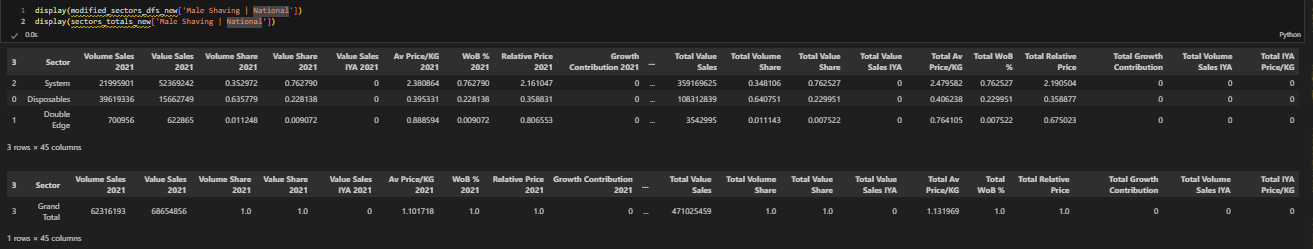
* + outputdic: A dictionary containing cleaned sector segment dataframes for each sector.
  + totaloutputdic: A dictionary containing totals dataframes for each sector.

defsecsegclean(inputdic): outputdic={} totaloutputdic={} for s in inputdic.keys(): t = inputdic[s].copy() t=DetectHeader(t).fillna(0) mod = t[(~t[t.columns[0]].astype(str).str.contains('Grand Total'))] mod = mod.sort\_values([col for col in mod.columns if'Value Share'in col], ascending=False) tot = t[(t[t.columns[0]].astype(str).str.contains('Grand Total'))] ifnot mod.empty: outputdic[s] = mod ifnot tot.empty: totaloutputdic[s] = tot return outputdic,totaloutputdic

##### Example: input dataframe before clean

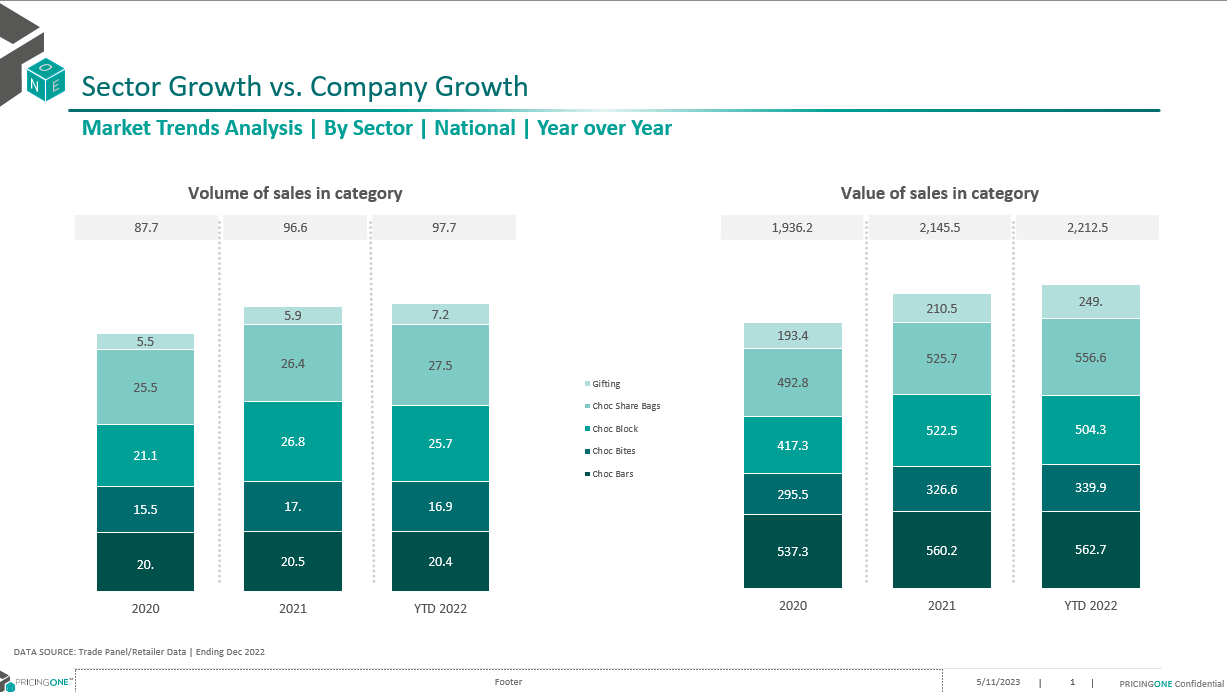


##### Example: How to call the function & show the DataFrame output



#### [Step 3: Write Functions to Create Slides](https://github.com/khaledSeifEleslam/Slide-Automate/blob/main/general_functions/Landscape%20Replacement%20Function.ipynb)

To create slides we need some function Example:Market Trends slides



* [Totals\_Table\_Fill](https://github.com/khaledSeifEleslam/Slide-Automate/blob/main/Landscape%20slide%20duplicate/Landscape%20Replacement%20Function.ipynb): function populates a table on a slide with totals data from a specified dataframe. It formats the text in the cells, including font size, font name, and alignment.
  + It takes Parameters: table (Table): Table shape in the slide. list\_duplicates (list): List of duplicate names for identifying slides. df\_totals (dict): Dictionary of total DataFrames for each duplicate name. cols (list): Columns in the DataFrame. slidenum (int): Slide number.
  + Returns: Table: Updated table shape.

defTotals\_Table\_Fill(table, list\_duplicates, df\_totals, cols, slidenum):for i, row inenumerate(table.rows): if i != 0: for j, cell inenumerate(row.cells): cell.text = str(round(df\_totals[list\_duplicates[slidenum]][cols].iloc[0, 1:4][j] / 1000000, 1 )) cell.text\_frame.paragraphs[0].runs[0].font.size = Pt(10) cell.text\_frame.paragraphs[0].runs[0].font.name = 'Nexa Book' cell.text\_frame.paragraphs[0].alignment = PP\_ALIGN.CENTER return table

* [Column\_Chart\_Fill](https://github.com/khaledSeifEleslam/Slide-Automate/blob/main/general_functions/Landscape%20Replacement%20Function.ipynb):function customizes a column chart on a slide by filling series with specific colors based on their names and adding formatted data labels to each point in the series
  + It takes Parameters: chart (Chart): Chart shape in the slide. scope (list): List of scope names.

defColumn\_Chart\_Fill(chart, scope): client\_colors = [RGBColor(0, 80, 75), RGBColor(0, 108, 109), RGBColor(0, 160, 151), RGBColor(126, 202, 196), RGBColor(153, 199, 197), RGBColor(178, 223, 220)] gray\_colors = [RGBColor(217, 217, 217), RGBColor(191, 191, 191), RGBColor(166, 166, 166), RGBColor(155, 152, 152), RGBColor(127, 127, 127)] for i, series inenumerate(chart.series): if series.name in scope: series.format.fill.solid() series.format.fill.fore\_color.rgb = client\_colors[i if i < len(client\_colors) else -1] else: series.format.fill.solid() series.format.fill.fore\_color.rgb = gray\_colors[i if i < len(gray\_colors) else -1] for j, point inenumerate(series.points): data\_label = point.data\_label data\_label.has\_text\_frame = True data\_label.text\_frame.text = str(round(series.values[j], 1)) data\_label.text\_frame.paragraphs[0].runs[0].font.color.rgb = RGBColor(255, 255, 255)

* [Markete\_Trends function](https://github.com/khaledSeifEleslam/Slide-Automate/blob/main/Landscape%20slide%20duplicate/Landscape%20Replacement%20Function.ipynb), which automates the creation and updating of market trend analysis slides in a PowerPoint presentation. It fills the slides with charts and tables using data from provided dataframes. The function processes each slide based on a list of duplicate names, adding relevant data and formatting to charts and tables. It supports customization through parameters like position offset and slide grouping criteria. This function is useful for generating detailed, data-driven presentations on market trends, enhancing efficiency and consistency in reporting.
  + It takes Parameters: prs (Presentation): PowerPoint presentation object. list\_duplicates (list): List of duplicate names for identifying slides. modified\_df (dict): Dictionary of modified DataFrames for each duplicate name. df\_totals (dict): Dictionary of total DataFrames for each duplicate name. scope (list): List of scope names. position (int, optional): Position offset for slides. Defaults to 0. slide\_by (str, optional): Slide grouping criteria. Defaults to ''.

defMarket\_Trends(prs, list\_duplicates, modified\_df, df\_totals, scope, position=0, slide\_by=''):for slidenum inrange(len(list\_duplicates)): shapes = prs.slides[slidenum + position].shapes charts = [] tables = [] title = shapes.title.text shapes[4].text = data\_source shapes[5].text = f'Market Trends Analysis | By {slide\_by} | ' + list\_duplicates[slidenum] + ' | Year over Year' shapes[5].text\_frame.paragraphs[0].font.bold = Truefor shape in shapes: if shape.has\_chart: charts.append(shape) if shape.has\_table: tables.append(shape) for chartnum inrange(2): chart = charts[chartnum].chart table = tables[chartnum].table chart\_data = CategoryChartData() chart\_data.categories = ['2021', '2022', 'YTD 2023'] volume\_cols = [c for c in modified\_df[list\_duplicates[slidenum]].columns[modified\_df[list\_duplicates[slidenum]].columns.str.contains(f'{slide\_by}|Volume Sales')]] value\_cols = [c for c in modified\_df[list\_duplicates[slidenum]].columns[(modified\_df[list\_duplicates[slidenum]].columns.str.contains(f'{slide\_by}|Value Sales')) & ~(modified\_df[list\_duplicates[slidenum]].columns.str.contains('IYA'))]] if chartnum == 0: for i inrange(modified\_df[list\_duplicates[slidenum]].shape[0]): series\_name = modified\_df[list\_duplicates[slidenum]][volume\_cols].iloc[i, 0] number = modified\_df[list\_duplicates[slidenum]][volume\_cols].iloc[i, 1:4] / 1000000 series = chart\_data.add\_series(series\_name, number) chart.replace\_data(chart\_data) Column\_Chart\_Fill(chart, scope) Totals\_Table\_Fill(table, list\_duplicates, df\_totals, volume\_cols, slidenum) elif chartnum == 1: value\_cols = [c for c in modified\_df[list\_duplicates[slidenum]].columns[(modified\_df[list\_duplicates[slidenum]].columns.str.contains(f'{slide\_by}|Value Sales')) & ~(modified\_df[list\_duplicates[slidenum]].columns.str.contains('IYA'))]] for i inrange(modified\_df[list\_duplicates[slidenum]].shape[0]): series\_name = modified\_df[list\_duplicates[slidenum]][value\_cols].iloc[i, 0] number = modified\_df[list\_duplicates[slidenum]][value\_cols].iloc[i, 1:4] / 1000000 series = chart\_data.add\_series(series\_name, number) chart.replace\_data(chart\_data) Column\_Chart\_Fill(chart, scope) Totals\_Table\_Fill(table, list\_duplicates, df\_totals, value\_cols, slidenum)

#### [Step 4: Duplicate Slides](https://github.com/khaledSeifEleslam/Slide-Automate/blob/main/Landscape%20slide%20duplicate/Landscape%20duplicate.ipynb)

* prepares data and configurations for generating market analysis slides in a PowerPoint presentation. It creates index and duplication lists dynamically based on the presence of segment data and the number of data keys in various dictionaries. These lists are used to control the slide generation process. The script also defines a comprehensive list of section names for organizing slide titles, ensuring each slide is labeled appropriately based on its content. This setup allows for automated, consistent, and dynamic creation of market trend analysis slides, which is particularly useful for large presentations with multiple sections and varying data inputs.

# This script prepares index and duplication lists for generating PowerPoint slides# with various market trends and growth analysis. It dynamically adjusts based on the presence# of segment data and compiles a list of section names for slide titles.# Generate index list based on the presence of segments and section numbers index = [ \*[0] \* (4if segments else3), # Index for manufacturer and brand slides \*[1] \* (4if segments else3), # Index for sector slides \*[2] \* (2if segments else1), # Index for segment slides \*[2] \* section\_number, # Index for additional sections \*[3] \* (2if segments else1), # Index for additional segment slides \*[3] \* section\_number, # Index for additional sections4, # Index for specific slide \*sectorIndex \* 2, # Index for sector details \*segmentIndex \* 2, # Index for segment details8, 9, 10# Fixed indices for specific slides ] # Generate duplication list based on the presence of segments and data keys duplication = [ len(modified\_manuf\_dfs\_new.keys()), # Number of manufacturer data keyslen(modified\_brands\_share\_new.keys()), # Number of brand share data keyslen(modified\_sectors\_dfs\_new.keys()), # Number of sector data keyslen(modified\_segment\_dfs\_new.keys()) if segments else0, # Number of segment data keys if segments existlen(modified\_manuf\_dfs\_new.keys()), # Repeated for next sectionlen(modified\_brands\_share\_new.keys()), # Repeated for next sectionlen(modified\_sectors\_dfs\_new.keys()), # Repeated for next sectionlen(modified\_segment\_dfs\_new.keys()) if segments else0, # Repeated for next section if segments existlen(modified\_sectors\_P12M\_new.keys()), # Number of sectors data keys for P12Mlen(modified\_segment\_P12M\_new.keys()) if segments else0, # Number of segments data keys for P12M if segments exist \*duplication\_num, # Additional duplication numberslen(modified\_sectors\_clients\_new.keys()), # Number of sector client data keyslen(modified\_segment\_clients\_new.keys()) if segments else0, # Number of segment client data keys if segments exist \*duplication\_num\_Avg, # Additional average duplication numberslen(modified\_manuf\_P12M\_new.keys()), # Number of manufacturer data keys for P12Mlen(sharGrowthDf\_sec.keys()), # Number of sector share growth data keyslen(sharGrowthDf\_sec.keys()), # Repeated for next sectionlen(sharGrowthDf\_seg.keys()) if segments else0, # Number of segment share growth data keys if segments existlen(sharGrowthDf\_seg.keys()) if segments else0, # Repeated if segments existlen(modified\_calendar\_new.keys()), # Number of calendar data keyslen(modified\_brands\_evolution\_sorted\_new.keys()), # Number of sorted brand evolution data keyslen(categories\_overview\_dfs\_new.keys()) # Number of category overview data keys ] # Remove zeros from duplication list duplication = [item for item in duplication if item != 0] # Define section names for each slide group section\_names\_slide1 = ["Market Trends by Manufacturer", "Market Trends by Brands", "Market Trends by Sectors"] + (["Market Trends by Segments"] iflen(segments) > 0else []) section\_names\_slide2 = ["Market Concentraion By Manufacturer", "Market Concentration By Brands", "Market Concentration By Sectors"] + (["Market Concentration By Segments"] iflen(segments) > 0else []) section\_names\_slide3 = ["Market Growth By Sectors"] + (["Market Growth By Segments"] iflen(segments) > 0else []) + [\*section\_name\_Growth] section\_names\_slide4 = ["Value Vs AvgPrice By Sectors"] + (["Value Vs AvgPrice By Segments"] iflen(segments) > 0else []) + [\*section\_name\_Avg] section\_names\_slide5 = ['Share and Growth By Manufacturer/Brands'] section\_names\_slide6 = ['Share And Growth By Manufacturer By Sector', 'Share And Growth By Brands By Sector'] + (['Share And Growth By Manufacturer By Segment', 'Share And Growth By Brands By Segment'] iflen(segments) > 0else []) section\_names\_slide7 = ["Category Trends"] section\_names\_slide8 = ['Share Evolution By Brand'] section\_names\_slide9 = ["Category Overview"] # Combine all section names into a single list section\_names = [ \*section\_names\_slide1, \*section\_names\_slide2, \*section\_names\_slide3, \*section\_names\_slide4, \*section\_names\_slide5, \*section\_names\_slide6, \*section\_names\_slide7, \*section\_names\_slide8, \*section\_names\_slide9 ]

#### [Step 5: Replace Data in Slides](https://github.com/khaledSeifEleslam/Slide-Automate/blob/main/Landscape%20slide%20duplicate/Landscape%20duplicate.ipynb)

* This script uses the Market\_Trends function to generate market trend analysis slides in a PowerPoint presentation for various categories such as Top Companies, Top Brands, Sectors, and Segments. It initializes a position counter p and increments it after each call to ensure the slides are added sequentially. The calculate\_position function is used to determine the correct position for each set of slides, allowing for dynamic and organized slide generation based on the provided data dictionaries and scope lists. This approach enables efficient creation of comprehensive market analysis presentations.

# calls the Market\_Trends function to generate PowerPoint slides# for different categories (Top Companies, Top Brands, Sector, Segment) and# updates the position variable for each call to ensure slides are generated in the correct order. p=0 Market\_Trends(prs, list(modified\_manuf\_dfs\_new.keys()), modified\_manuf\_dfs\_new, modified\_manuf\_totals\_new, client\_manuf ,position = calculate\_position(p), slide\_by = 'Top Companies') p+=1 Market\_Trends(prs, list(modified\_brands\_share\_new.keys()), modified\_brands\_share\_new, modified\_brands\_totals\_new, client\_brands ,position =calculate\_position(p), slide\_by = 'Top Brands') p+=1 Market\_Trends(prs, list(modified\_sectors\_dfs\_new.keys()), modified\_sectors\_dfs\_new, sectors\_totals\_new, sectors ,position = calculate\_position(p), slide\_by = 'Sector') p+=1iflen(segments)!=0: Market\_Trends(prs, list(modified\_segment\_dfs\_new.keys()), modified\_segment\_dfs\_new, segment\_totals\_new, segments ,position = calculate\_position(p), slide\_by = 'Segment') p+=1

#### [Step 6: Save Presentation](https://github.com/khaledSeifEleslam/Slide-Automate/blob/main/Landscape%20slide%20duplicate/Landscape%20duplicate.ipynb)

* performs two main tasks: saving the current PowerPoint presentation to a file and opening that file using the PowerPoint application. The outputPath variable is constructed using the current working directory, ensuring the presentation is saved in the correct location. After saving the presentation, the script uses win32com.client to dispatch the PowerPoint application and open the saved presentation. This automation allows for seamless transition from generating the presentation to viewing or editing it in PowerPoint, streamlining the workflow for creating market analysis slides.

# This script saves the generated PowerPoint presentation to a specified path# and then opens the saved presentation using the PowerPoint application.# Define the output path for the PowerPoint presentation outputPath = os.getcwd() + "\\Landscape output.pptx"# Save the PowerPoint presentation to the specified output path prs.save(outputPath) # Initialize the PowerPoint application using win32com client app = win32.Dispatch("PowerPoint.Application") # Open the saved PowerPoint presentation presentation = app.Presentations.Open(outputPath)

##### Ex:Market Trends Slide OutPut After Replacement Data

![Alt text](C:\Users\aleaa\Documents\Slide-Automate\Slides Documentation\market trends output.png)

## Pricing Section

### Introduction

In the slide automation pricing : from12 slide base we create 5 decks

![1732021356685](image/Slidesdocumantion/1732021356685.png)

1. Pricing Output Slides:
   * Price Positioning Analysis
   * Share and Growth By Brands(**Leadership Table**)
   * Value Sales Vs Avg Price
2. Pricing Avg&Shelf Price Output Slides:
   * Avg Price/Vol
   * Shelf Price/Vol
3. Pricing Price Point Output Slides :
   * Price Point Distribution Analysis by product
   * Price Point Comparison Analysis by Product
   * Price Point Distribution Analysis by brand
   * Price Point Distribution by brand by Sector
4. Pricing Correlation Output Slides:
   * Price Correlation Analysis P3Y
   * Price Correlation Analysis P12M
5. Pricing not\_forcing\_CB\_Output Slides
   * Price Positioning Analysis
   * Share and Growth By Brands(**Leadership Table**)
   * Price Point Distribution Analysis by brand

### Project Steps

* Project Flow ![Project Flow](../Slides Documantion/duplication\_Steps.PNG)
* [Step 1: Import Libraries we use](#step-1-import-libraries)
* [Step 2: modified Data frames: cleaning and preprocessing the data frames](#step-2-modified-data-frame)
* [Step 3: Write Functions to Create Slides: Define functions to dynamically generate slides based on the base slides](#step-3-write-functions-to-create-slide)
* [Step 4: Duplicate Slides: Use functions or methods to duplicate existing slides as needed for the presentation.](#step-4-duplicate-slides)
* [Step 5: Replace Data in Slide: update information from the cleaned data frames to slides](#step-5-replace-data-in-slide)
* [Step 6: Save Presentation](#step-6-save-presentation)

#### [Step 1: Import Libraries we use](https://github.com/khaledSeifEleslam/Slide-Automate/blob/main/general_functions/generalFunctions.ipynb)

##### Ex: Libraries we use

* This script sets up an environment for working with PowerPoint presentations, data manipulation, filesystem operations, and COM (Component Object Model) object access.
* It imports necessary modules such as 'pptx' for PowerPoint automation, 'win32com' for COM object access and Windows automation, 'pandas' and 'numpy' for data manipulation,
* 'pathlib' for working with filesystem paths, 're' for regular expression operations, and various other modules for general-purpose tasks like file operations and timing functions.
* By importing these modules, the script prepares itself for tasks such as creating or modifying PowerPoint presentations, analyzing data using pandas and numpy, interacting
* with the Windows environment using win32com, and performing filesystem operations using shutil and os. Overall, this script provides a comprehensive setup for automating tasks
* related to PowerPoint presentations and general-purpose Python programming.

# Import necessary module for working with PowerPoint presentationsfrom pptx import Presentation # Import the win32com.client module, aliasing it as win32 for convenienceimport win32com.client as win32 # Import pandas for data manipulation and analysisimport pandas as pd # Import numpy for numerical computingimport numpy as np # Import the Path class from pathlib for working with filesystem pathsfrom pathlib import Path # Import re for regular expression operationsimport re # Import sys for access to interpreter-related functionsimport sys # Import time for various time-related functionsimport time # Assign win32.constants to a shorter alias win32c for easier access win32c = win32.constants # Import shutil for high-level file operationsimport shutil # Import os for operating system dependent functionalityimport os # Import win32com.client again for COM object and functions accessimport win32com.client # Import warnings for warning control functionalityimport warnings

#### [Step 2: modified Data frame](https://github.com/khaledSeifEleslam/Slide-Automate/blob/main/Pricing%20slide%20duplicate/Pricing%20duplicate.ipynb)

##### EX: input dataframes before cleaning

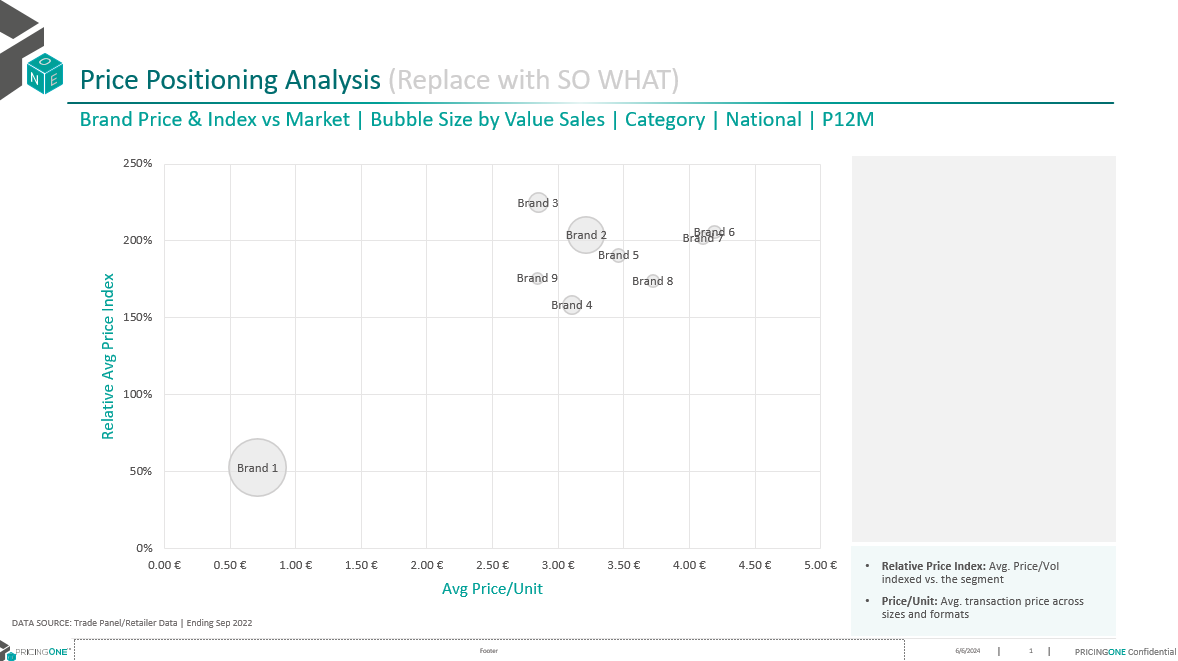
![data frame before cleaning](../Slides Documentation/Pricing dataframe input.png)

###### cleaning Code

* This code processes a dictionary of DataFrames, modified\_price\_positioning\_sorted, by performing a series of operations on each DataFrame. Specifically, it iterates over each key in the dictionary, makes a copy of the DataFrame to avoid altering the original, filters out rows where the 'Top Brands' column has the value 'Others', replaces all NaN values with 0, and then updates the dictionary with the modified DataFrame. This ensures that the DataFrames only include data from specified brands and that missing values are handled appropriately.

# Iterate over each key in the dictionary 'modified\_price\_positioning\_sorted'for k in modified\_price\_positioning\_sorted.keys(): # Create a copy of the DataFrame associated with the current key to avoid modifying the original data df = modified\_price\_positioning\_sorted[k].copy() # Filter out rows where the 'Top Brands' column has the value 'Others' df = df[df['Top Brands'] != 'Others'] # Replace all NaN values in the DataFrame with 0 df = df.replace(np.nan, 0) # Update the dictionary with the modified DataFrame modified\_price\_positioning\_sorted[k] = df

###### Data frame after cleaning



#### [Step 3: Write Functions to Create Slide](https://github.com/khaledSeifEleslam/Slide-Automate/blob/main/general_functions/Pricing%20Replacement%20Function.ipynb)

##### Example slide : Price Positioning Analysis Slide

![Price Positioning Analysis Slide](../Slides Documentation/price positioning slide .png)

* [PricePositioning Function](https://github.com/khaledSeifEleslam/Slide-Automate/blob/main/general_functions/Pricing%20Replacement%20Function.ipynb): generates slides for a PowerPoint presentation, focusing on price positioning analysis with bubble chart visualizations. It iterates through a specified number of slides (numOfDuplicates)
  + parameters : prs: PowerPoint presentation object. modified\_price\_positioning\_sorted: Dictionary containing sorted price positioning dataframes. numOfDuplicates: Number of duplicate slides to generate. position: Position index to start adding slides (default is 0).

defpricePositioning(prs,modified\_price\_positioning\_sorted,numOfDuplicates,position=0):for slidenum inrange(numOfDuplicates): # Extract market and corresponding dataframe market=list(modified\_price\_positioning\_sorted.keys())[slidenum] df=modified\_price\_positioning\_sorted[market].reset\_index(drop=True) # Access shapes in the slide shapes = prs.slides[slidenum+position].shapes charts = [] tables = [] title = shapes.title.text # Update text boxes in the slide shapes[4].text = data\_source shapes[5].text = 'Brand Price & Index vs Market | Bubble Size by Value Sales | '+market+' | P12M' shapes[5].text\_frame.paragraphs[0].font.bold = Truefor shape in shapes: if shape.has\_chart: shape\_id = shape.shape\_id charts.append(shape) chart = charts[0].chart charts[0].left = Inches(0.57) # Adjust left position chart\_name = charts[0].name chart\_type = chart.chart\_type # Add bubble chart data chart\_data = BubbleChartData() chart\_data.categories = df['Av Price/Unit'].unique().tolist() series = chart\_data.add\_series("Relative Price Index") series.has\_data\_labels = True# Add data points to the bubble chartfor i inrange(df.shape[0]): series.add\_data\_point(df['Av Price/Unit'].iloc[i], df['Relative Price'].iloc[i], df['Value Sales'].iloc[i]) chart.replace\_data(chart\_data) # Update chart formatting xlsx\_file=BytesIO() with chart\_data.\_workbook\_writer.\_open\_worksheet(xlsx\_file) as (workbook, worksheet): chart\_data.\_workbook\_writer.\_populate\_worksheet(workbook, worksheet) worksheet.write(0, 4, "labels") worksheet.write\_column(1, 4, df['Top Brands'], None) chart.\_workbook.update\_from\_xlsx\_blob(xlsx\_file.getvalue()) category\_axis = chart.category\_axis if sign == 'Before': category\_axis.tick\_labels.number\_format = f'{currency}#,##0.00'if decimals == 2elsef'{currency}#,##0'else: category\_axis.tick\_labels.number\_format = f'#,##0.00{currency}'if decimals == 2elsef'#,##0{currency}' category\_axis.auto\_axis = True value\_axis = chart.value\_axis value\_axis.tick\_labels.number\_format = '0%' value\_axis.auto\_axis = True# Customize data labels for each point in the chartfor i,point inenumerate(chart.series[0].points): if df['Top Brands'].iloc[i]=="Others": point.format.fill.background() point.data\_label.text\_frame.text='' point.format.line.width = Pt(0) else: data\_label = point.data\_label data\_label.has\_text\_frame=True data\_label.text\_frame.text=df['Top Brands'].iloc[i] data\_label.text\_frame.paragraphs[0].runs[0].font.size = Pt(10) data\_label.position = XL\_LABEL\_POSITION.CENTER point.format.fill.solid() point.format.fill.fore\_color.rgb = RGBColor(245,245,245) point.format.line.color.rgb = RGBColor(207,206,206) # Set the desired RGB color value point.format.line.width = Pt(1)

#### [Step 4: Duplicate Slides](https://github.com/khaledSeifEleslam/Slide-Automate/blob/main/Pricing%20slide%20duplicate/Pricing%20duplicate.ipynb)

* This code is preparing data and configurations for generating a PowerPoint presentation with multiple sections, each requiring a different number of slides based on various price and distribution analyses. It includes:
  + Index List: Specifies the starting slide positions for different sections.
  + Duplication List: Indicates the number of slides to be generated for each section, based on the length of different datasets.
  + Section Names: Provides names for each section in the presentation.
  + Paths: Defines the file paths for the base PowerPoint template and the new duplicated presentation.

# Define the index list for slide positions index = [0, 1, 2, 3, 4, 3, 4, 5, 5, 5, 5, 5, 5, 6, 7, 7] # Define the duplication list representing the number of slides to be generated for each section duplication = [ len(modified\_price\_positioning\_sorted.keys()), # Number of price positioning slideslen(modified\_brands\_segments\_leadership.keys()), # Number of segments leadership slideslen(modified\_brands\_sector\_leadership.keys()), # Number of sectors leadership slideslen(all\_brands\_sector.keys()), # Number of sector avg price/vol comparison slideslen(all\_brands\_sector.keys()), # Number of sector shelf price/vol comparison slideslen(all\_brands\_segment.keys()), # Number of segment avg price/vol comparison slideslen(all\_brands\_segment.keys()), # Number of segment shelf price/vol comparison slideslen(sectorP3mPD.keys()), # Number of category price point distribution analysis P3M slideslen(sectorP12mPD.keys()), # Number of category price point distribution analysis P12M slideslen(segmentP3mPD.keys()), # Number of sector price point distribution analysis P3M slideslen(segmentP12mPD.keys()), # Number of sector price point distribution analysis P12M slideslen(sub\_segmentP3mPD.keys()), # Number of segment price point distribution analysis P3M slideslen(sub\_segmentP12mPD.keys()), # Number of segment price point distribution analysis P12M slideslen(modified\_brandPriceDistribution.keys()), # Number of price point distribution analysis by brand slideslen(modified\_sectorsPriceDistribution.keys()), # Number of price point distribution by brand by sector slideslen(modified\_segmentPriceDistribution.keys()) # Number of price point distribution by brand by segment slides ] # Define the section names to be used in the presentation section\_names = [ "Price Positioning Analysis", "Segments Leadership Analysis", "Sectors Leadership Analysis", "Sector Avg Price/Vol Comparison", "Sector Shelf Price/Vol Comparison", "Segment Avg Price/Vol Comparison", "Segment Shelf Price/Vol Comparison", "Category Price Point Distribution Analysis P3M", "Category Price Point Distribution Analysis P12M", "Sector Price Point Distribution Analysis P3M", "Sector Price Point Distribution Analysis P12M", "Segment Price Point Distribution Analysis P3M", "Segment Price Point Distribution Analysis P12M", "Price Point Distribution Analysis By Brand", "Price Point Distribution By Brand By Sector", "Price Point Distribution By Brand By Segment" ] # Define paths for the base PowerPoint template and the new duplicated presentation path = os.getcwd() + '\Pricing slide base.pptx' new\_pre = os.getcwd() + '\Pricing duplicated.pptx'

#### [Step 5: Replace Data in Slide](https://github.com/khaledSeifEleslam/Slide-Automate/blob/main/Pricing%20slide%20duplicate/Pricing%20duplicate.ipynb)

* This part of the code calls the pricePositioning function to generate slides for the "Price Positioning Analysis" section of the presentation. It uses the prs PowerPoint presentation object, the dictionary modified\_price\_positioning\_sorted containing the sorted price positioning dataframes, and the first element of the duplication list to determine the number of slides to generate. The position variable is set to 0, indicating that the slides should be added starting from the first position.

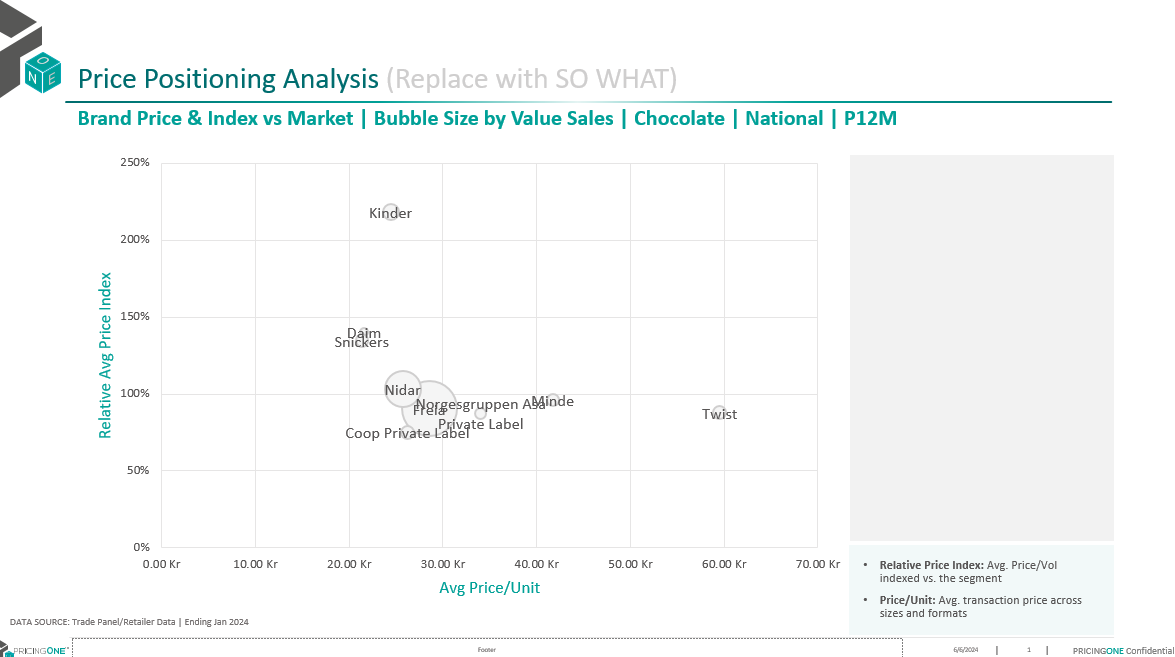
# Set the initial position for slide insertion to 0 position = 0# Call the pricePositioning function to generate slides for price positioning analysis pricePositioning(prs, modified\_price\_positioning\_sorted, duplication[0], position)

#### [Step 6: Save Presentation](https://github.com/khaledSeifEleslam/Slide-Automate/blob/main/Pricing%20slide%20duplicate/Pricing%20duplicate.ipynb)

* This code is responsible for finalizing the creation of a PowerPoint presentation by saving it to a specified file path and then opening it using Microsoft PowerPoint. Initially, it defines the output path for the new PowerPoint presentation by combining the current working directory with the filename Pricing output.pptx. The script then saves the modified presentation (prs object) to this specified path. After saving, it uses the win32com.client.Dispatch method to create an instance of the PowerPoint application, and then it opens the saved presentation within this application. This process ensures that the newly created presentation is both saved and immediately available for viewing or further editing in Microsoft PowerPoint.

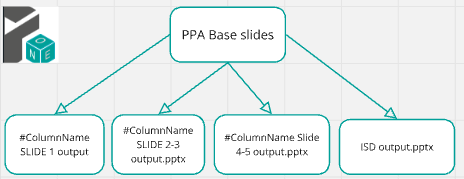
# Define the output path for the new PowerPoint presentation outputPath = os.getcwd() + "\\Pricing output.pptx"# Save the modified presentation to the specified output path prs.save(outputPath) # Open the saved PowerPoint presentation using the PowerPoint application app = win32.Dispatch("PowerPoint.Application") presentation = app.Presentations.Open(outputPath)

##### Example: OutPut Slide After Replacement Data "PricePositioning Slide OutPut"



## PPA Section

### Introduction

In the slide automation PPA: from 5 slide base we create 4 decks  ColumnName= ['Size Bracket']#Run per bracket

1. #ColumnName SLIDE 1output Slides:
   * Brand Share Topline
2. #ColumnName SLIDE 2-3output Slides:
   * #ColumnName by Sector/Segment
3. #ColumnName Slide4-5 output Slides:
   * Brackets Analysis By Sector
   * BracketsAnalysis By Segment
4. ISD output Slides :
   * Inter-size Discount Analysis

### Project Steps

* Project Flow ![Project Flow](../Slides Documantion/duplication\_Steps.PNG)
* [Step 1: Import Libraries we use](#step-1-import-libraries)
* [Step 2: modified Data frames: cleaning and preprocessing the data frames](#step-2-modified-data-frame)
* [Step 3: Write Functions to Create Slides: Define functions to dynamically generate slides based on the base slides](#step-3-write-functions-to-create-slide)
* [Step 4: Duplicate Slides: Use functions or methods to duplicate existing slides as needed for the presentation.](#step-4-duplicate-slides)
* [Step 5: Replace Data in Slide: update information from the cleaned data frames to slides](#step-5-replace-data-in-slide)
* [Step 6: Save Presentation](#step-6-save-presentation)

#### [Step 1: Import Libraries we use](https://github.com/khaledSeifEleslam/Slide-Automate/blob/main/general_functions/generalFunctions.ipynb)

##### Ex: Libraries we use

* This script sets up an environment for working with PowerPoint presentations, data manipulation, filesystem operations, and COM (Component Object Model) object access.
* It imports necessary modules such as 'pptx' for PowerPoint automation, 'win32com' for COM object access and Windows automation, 'pandas' and 'numpy' for data manipulation,
* 'pathlib' for working with filesystem paths, 're' for regular expression operations, and various other modules for general-purpose tasks like file operations and timing functions.
* By importing these modules, the script prepares itself for tasks such as creating or modifying PowerPoint presentations, analyzing data using pandas and numpy, interacting
* with the Windows environment using win32com, and performing filesystem operations using shutil and os. Overall, this script provides a comprehensive setup for automating tasks
* related to PowerPoint presentations and general-purpose Python programming.

# Import necessary module for working with PowerPoint presentationsfrom pptx import Presentation # Import the win32com.client module, aliasing it as win32 for convenienceimport win32com.client as win32 # Import pandas for data manipulation and analysisimport pandas as pd # Import numpy for numerical computingimport numpy as np # Import the Path class from pathlib for working with filesystem pathsfrom pathlib import Path # Import re for regular expression operationsimport re # Import sys for access to interpreter-related functionsimport sys # Import time for various time-related functionsimport time # Assign win32.constants to a shorter alias win32c for easier access win32c = win32.constants # Import shutil for high-level file operationsimport shutil # Import os for operating system dependent functionalityimport os # Import win32com.client again for COM object and functions accessimport win32com.client # Import warnings for warning control functionalityimport warnings

#### [Step 2: modified Data frame](https://github.com/khaledSeifEleslam/Slide-Automate/blob/main/PPA%20slide%20duplicate/PPA%20Duplicate.ipynb)

##### EX: input dataframes before cleaning

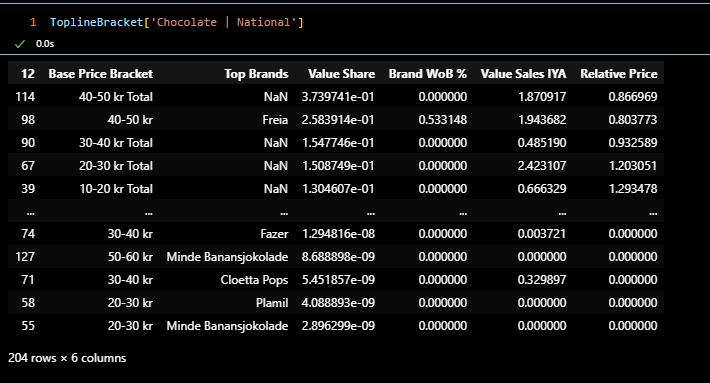
![1732186335029](image/Slidesdocumantion/1732186335029.png)

##### Cleaning Data Frame

* This code is part of a data processing pipeline for cleaning and preparing bracket-related data. It selects the appropriate DataFrame based on columnName, processes elements in a brackets list, and iterates over ppaDf keys to rename columns, remove rows, forward-fill missing values, replace NaNs, and sort by 'Value Share'. Cleaned DataFrames are stored in ToplineBracket. The script methodically handles data preparation, crucial for accurate analysis and reporting, but the use of brackets and commented-out lines suggest the code may still be in development or require additional context.

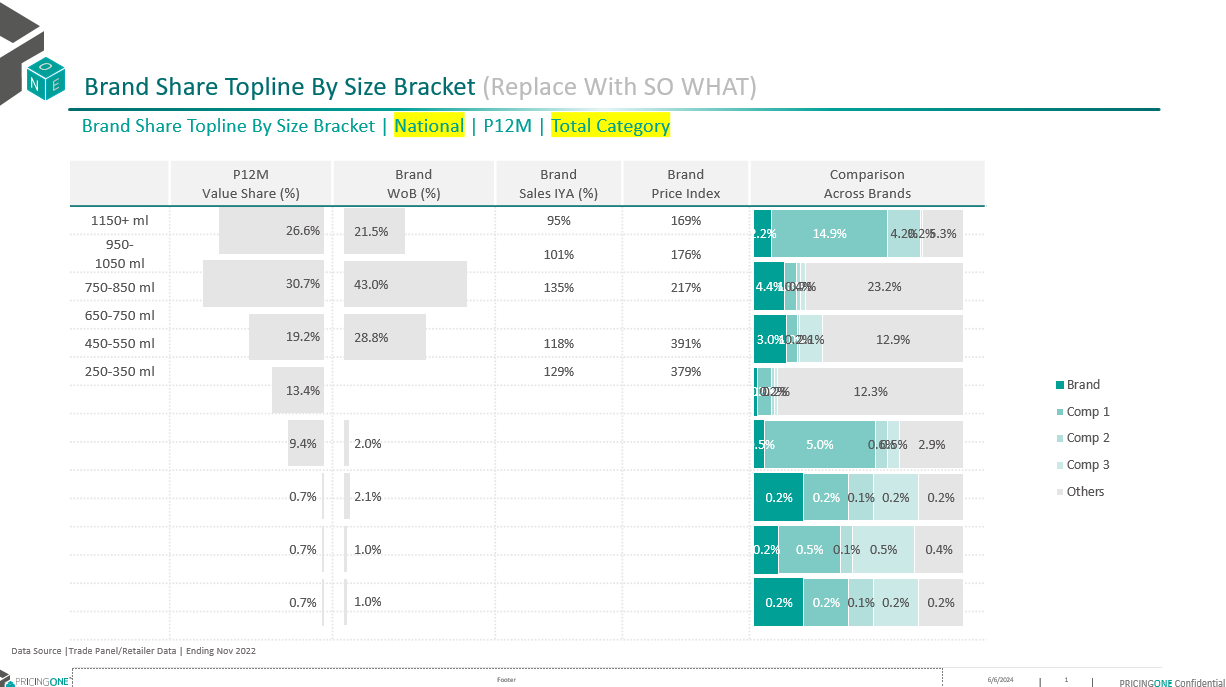
# Assign the appropriate DataFrame based on the value of columnName ppaDf = share\_topline\_base\_price\_bracket if columnName == "Base Price Bracket"else share\_topline\_size\_bracket # The variable 'brackets' is used but not defined in this snippet; it should be defined beforehand.# Iterate over the elements in 'brackets'for i in brackets: # Create a formatted string based on the elements of 'brackets', replacing special characters and converting to lowercasef"share\_topline\_{i.split('.[')[1]}".replace("]", "").replace(" ","\_").lower() # Initialize dictionaries to store processed DataFrames ToplineBracket = {} ToplineBracketTotal = {} # Iterate over the keys in ppaDffor key in ppaDf.keys(): df = ppaDf[key].copy() # Create a copy of the DataFrame to avoid modifying the original df.columns = df.iloc[12].str.replace('\xa0',' ') # Rename columns by replacing non-breaking spaces with regular spaces df = df.iloc[13:-1] # Remove the first 13 rows and the last row df[columnName] = df[columnName].ffill() # Forward-fill missing values in the specified column# df\_tot = df[df['Size Bracket'].str.contains("Total")] # (Optional) Filter rows that contain "Total" in 'Size Bracket'# df\_brands = df[~df["Size Bracket"].isin(df\_tot['Size Bracket'])] # (Optional) Exclude rows that match 'df\_tot' df[["Value Share", "Brand WoB %", "Value Sales IYA", "Relative Price"]] = df[["Value Share", "Brand WoB %", "Value Sales IYA", "Relative Price"]].replace(np.nan, 0).astype(float) # Replace NaN values with 0 and convert to float df = df.sort\_values('Value Share', ascending=False) # Sort the DataFrame by 'Value Share' in descending order# Check if the DataFrame is empty and print the key if it isif df.shape[0] == 0: print(key) else: # ToplineBracketTotal[key] = df\_tot # (Optional) Store the 'Total' filtered DataFrame ToplineBracket[key] = df # Store the processed DataFrame in the dictionary# Extract unique values from the processed DataFrames, removing ' Total' and collecting them into a list bracketsValue = list(set([value.replace(' Total','') for val in ToplineBracket.values() for value in val[columnName].unique() if'Total'in value]))

##### Data Frame After cleaning



#### [Step 3: Write Functions to Create Slides](https://github.com/khaledSeifEleslam/Slide-Automate/blob/main/general_functions/PPA%20Replacement%20Function.ipynb)

##### Example slide : Brand Share Topline By Size Bracket



* [brandShareToplin](https://github.com/khaledSeifEleslam/Slide-Automate/blob/main/general_functions/PPA%20Replacement%20Function.ipynb)function:Generate a PowerPoint slide presentation with data visualizations and tables showing brand share topline metrics by various brackets for a specified number of slides.
  + Parameters: prs (pptx.presentation.Presentation): The PowerPoint presentation object where slides will be added or modified. modifiedShareToplineBracket (dict): Dictionary containing data frames for different markets. bracketsValue (list): List of size brackets to be considered. clientElement (str): Name of the client brand to highlight in the presentation.

defbrandShareTopline(prs, modifiedShareToplineBracket, bracketsValue, numOfDuplicates, slide\_by, clientElement, position=0):for slidenum inrange(numOfDuplicates): # Get the market name and its corresponding data frame market = list(modifiedShareToplineBracket.keys())[slidenum] df = modifiedShareToplineBracket[market].copy() # Extract and sort 'Size' from the slide\_by column df['Size'] = df[f"{slide\_by}"].apply(lambda x: x.split('-')[1].split(' ')[0] if'-'in x else9999).astype(float) df = df.sort\_values(by=['Size'], ascending=False) # Filter total brand data and clean up column values dfTotalBrand = df[df[f"{slide\_by}"].str.contains('Total')] dfTotalBrand[f"{slide\_by}"] = dfTotalBrand[f"{slide\_by}"].str.replace(' Total', '') dfTotalBrand = dfTotalBrand[dfTotalBrand['Value Share'] > .01] # Filter the main data frame based on size df = df[df['Size'].isin(dfTotalBrand['Size'].unique())].sort\_values(by='Value Share', ascending=False) # Get the top 3 brands excluding the client element dfTopSales = df[(df['Top Brands'].notna()) & (df['Top Brands'] != clientElement)].drop\_duplicates(subset='Top Brands')['Top Brands'].iloc[:3].to\_list() dfBrandInScope = df[df['Top Brands'].isin(dfTopSales)] # Calculate the 'Other' category for the data frame dfOther = df[(~df['Top Brands'].isin(dfTopSales + [clientElement])) & (~df[f"{slide\_by}"].str.contains('Total'))].groupby([f"{slide\_by}", 'Size'])['Value Share'].sum().reset\_index().sort\_values(by='Size', ascending=False) missingOtherBracket = list(set(bracketsValue) - set(dfTotalBrand[f"{slide\_by}"].unique())) missingOtherBracket = pd.DataFrame({f"{slide\_by}": missingOtherBracket, 'Size': [float(x.split('-')[1].split(' ')[0]) if'-'in x else9999for x in missingOtherBracket]}) dfOther = pd.concat([dfOther, missingOtherBracket]).sort\_values(by='Size', ascending=False) dfTotalBrand = pd.concat([dfTotalBrand, missingOtherBracket]).sort\_values(by='Size', ascending=False) # Filter the client's brand data dfClientBrand = df[df['Top Brands'] == clientElement] # Access slide shapes to update text and formatting shapes = prs.slides[slidenum + position].shapes shapes[4].text = data\_source shapes[5].text = f'Brand Share Topline By {slide\_by} | {market} | P12M'# Format text as bold and set font size shapes[5].text\_frame.paragraphs[0].font.bold = Truefor p inrange(len(shapes[5].text\_frame.paragraphs)): shapes[5].text\_frame.paragraphs[p].font.size = Pt(12) shapes[6].text\_frame.paragraphs[0].runs[0].text = shapes[6].text\_frame.paragraphs[0].runs[0].text.replace('Size Bracket', slide\_by) shapes[6].text\_frame.paragraphs[0].font.size = Pt(16) # Create tables and charts tables, charts = createTableAndChart(shapes) # Adjust table row numbers table = tables[0].table num\_rows\_to\_remove = len(table.rows) - dfTotalBrand[f"{slide\_by}"].nunique() - 1for \_ inrange(num\_rows\_to\_remove): iflen(table.rows) > 1: # Skip removing the first row if there is more than one row row = table.rows[1] remove\_row(table, row) # Set table row height table\_height = Inches(3.81) # Specify the desired table height total\_row\_height = table\_height - table.rows[0].height num\_rows = len(table.rows) - 1# Exclude the first rowif num\_rows > 0: cell\_height = total\_row\_height / num\_rows for row inrange(1, len(table.rows)): table.rows[row].height = int(cell\_height) # Replace the table datafor i, row inenumerate(table.rows): for j, cell inenumerate(row.cells): if i == 0: # Update header cellsif j in [2, 3, 4]: cell.text = cell.text.replace('Brand', clientElement) for paragraph in cell.text\_frame.paragraphs: paragraph.font.name = 'Nexa Bold' paragraph.font.size = Pt(9) paragraph.alignment = PP\_ALIGN.CENTER paragraph.font.color.rgb = RGBColor(87, 85, 85) paragraph.font.bold = Falsecontinue# Update data cells sizeBracket = dfTotalBrand[f"{slide\_by}"].unique()[i - 1] if j == 0: cell.text = sizeBracket cell.text\_frame.paragraphs[0].font.name = 'Nexa Bold' cell.text\_frame.paragraphs[0].font.size = Pt(9) cell.text\_frame.paragraphs[0].alignment = PP\_ALIGN.CENTER if j == 3or j == 4: if j == 3: value = dfClientBrand[dfClientBrand[f"{slide\_by}"] == sizeBracket]['Value Sales IYA'].unique() # Exclude Brand 'Brand WoB %' < 5%if value and dfClientBrand[dfClientBrand[f"{slide\_by}"] == sizeBracket]['Brand WoB %'].unique()[0] < .0005: value = [0] cell.text = ''if (len(value) == 0) or (int(round(float(value[0]) \* 100, 0)) == 0) else (str(int(round(float(value[0]) \* 100, 0))) + '%'ifint(round(float(value[0]) \* 100, 0)) <= 1000else'Large') else: value = dfClientBrand[dfClientBrand[f"{slide\_by}"] == sizeBracket]['Relative Price'].unique() # Exclude Brand 'Brand WoB %' < 5%if value and dfClientBrand[dfClientBrand[f"{slide\_by}"] == sizeBracket]['Brand WoB %'].unique()[0] < .0005: value = [0] cell.text = ''iflen(value) == 0or (int(round(float(value[0]) \* 100, 0)) == 0) elsestr(int(round(float(value[0]) \* 100, 0))) + '%' cell.text\_frame.paragraphs[0].font.name = 'Nexa Book' cell.text\_frame.paragraphs[0].font.size = Pt(8) cell.text\_frame.paragraphs[0].alignment = PP\_ALIGN.CENTER # Update chart datafor chartNum in [0, 1]: chart = charts[chartNum].chart chart\_data = CategoryChartData() chart\_data.categories = [''] if chartNum == 0: missingBrandBracket = list(set(dfTotalBrand[f"{slide\_by}"].unique()) - set(dfClientBrand[f"{slide\_by}"].unique())) missingBrandBracket = pd.DataFrame({'Top Brands': clientElement, f"{slide\_by}": missingBrandBracket, 'Size': [float(x.split('-')[1].split(' ')[0]) if'-'in x else9999for x in missingBrandBracket]}) dfClientBrand2 = pd.concat([dfClientBrand, missingBrandBracket]).sort\_values(by='Size', ascending=False).replace(np.nan, None) # Exclude Value Share less than 5% dfClientBrand2['Brand WoB %'] = np.where(dfClientBrand2['Brand WoB %'] < .0005, None, dfClientBrand2['Brand WoB %']) brandWob = dfClientBrand2['Brand WoB %'].to\_list() chart\_data.add\_series('Brand WoB %', brandWob) else: valueShare = dfTotalBrand['Value Share'].replace(np.nan, None).to\_list() chart\_data.add\_series('Value Share', valueShare) chart.replace\_data(chart\_data) # Update the comparison chart chart2 = charts[2].chart chart\_data2 = CategoryChartData() chart\_data2.categories = dfTotalBrand[f"{slide\_by}"].unique() missingBrandBracket = list(set(dfTotalBrand[f"{slide\_by}"].unique()) - set(dfClientBrand[dfClientBrand['Top Brands'] == clientElement][f"{slide\_by}"].unique())) missingBrandBracket = pd.DataFrame({'Top Brands': clientElement, f"{slide\_by}": missingBrandBracket, 'Size': [float(x.split('-')[1].split(' ')[0]) if'-'in x else9999for x in missingBrandBracket]}) dfClientBrand2 = pd.concat([dfClientBrand[dfClientBrand['Top Brands'] == clientElement], missingBrandBracket]).sort\_values(by='Size', ascending=False) valueShare = dfClientBrand2['Value Share'].replace(np.nan, None).to\_list() chart\_data2.add\_series(clientElement, valueShare) for brand in dfBrandInScope['Top Brands'].unique(): missingBrandBracket = list(set(dfTotalBrand[f"{slide\_by}"].unique()) - set(dfBrandInScope[dfBrandInScope['Top Brands'] == brand][f"{slide\_by}"].unique())) missingBrandBracket = pd.DataFrame({'Top Brands': brand, f"{slide\_by}": missingBrandBracket, 'Size': [float(x.split('-')[1].split(' ')[0]) if'-'in x else9999for x in missingBrandBracket]}) dfClientBrand2 = pd.concat([dfBrandInScope[dfBrandInScope['Top Brands'] == brand], missingBrandBracket]).sort\_values(by='Size', ascending=False) valueShare = dfClientBrand2['Value Share'].replace(np.nan, None).to\_list() chart\_data2.add\_series(brand, valueShare) valueShare = dfOther['Value Share'].replace(np.nan, None).to\_list() chart\_data2.add\_series('Others', valueShare) chart2.replace\_data(chart\_data2)

#### [Step 4: Duplicate Slides](https://github.com/khaledSeifEleslam/Slide-Automate/blob/main/PPA%20slide%20duplicate/PPA%20Duplicate.ipynb)

[Duplicate Slides](../PPA slide duplicate/PPA Duplicate.ipynb): this part of code calculate duplication values, and define section names for generating or updating a PowerPoint presentation.

# Generate a list of indices based on various category, sector, segment, and channel indices index = [0, \*categoryIndex, \*sectorIndex, \*segmentIndex, \*channelRepeat] # Calculate the number of duplications for each section duplication = [ len(ToplineBracket.keys()) \* len(clientBrand), # Duplication for the main topline section \*[len(dfLis) \* len(clientBrand)] \* (3iflen(segmentInScope) > 0else2), # Duplication for sector/segmentlen(categories) \* len(clientBrand), # Duplication for categorieslen(sectorInScope) \* len(clientBrand), # Duplication for sectorslen(segmentInScope) \* len(clientBrand) # Duplication for segments ] # Remove any zero duplication values from the list duplication = [item for item in duplication if item != 0] # Define section names for the slides section\_names\_slide1 = ["Brand Share Topline By " + columnName] section\_names\_slide2 = [ columnName + " By Sector/Segment By Category", columnName + " By Sector/Segment By Sector" ] + ([columnName + " By Sector/Segment By Segment"] iflen(segmentInScope) > 0else []) section\_names\_slide3 = [ columnName + " By Channel/Retailer By Category", columnName + " By Channel/Retailer By Sector" ] + ([columnName + " By Channel/Retailer By Segment"] iflen(segmentInScope) > 0else []) # Combine all section names into one list section\_names = [\*section\_names\_slide1, \*section\_names\_slide2, \*section\_names\_slide3] # Define the path to the base PowerPoint slide and the new duplicated slide path = os.getcwd() + '\\PPA slide base.pptx' new\_pre = os.getcwd() + '\\PPA slide duplicate.pptx'

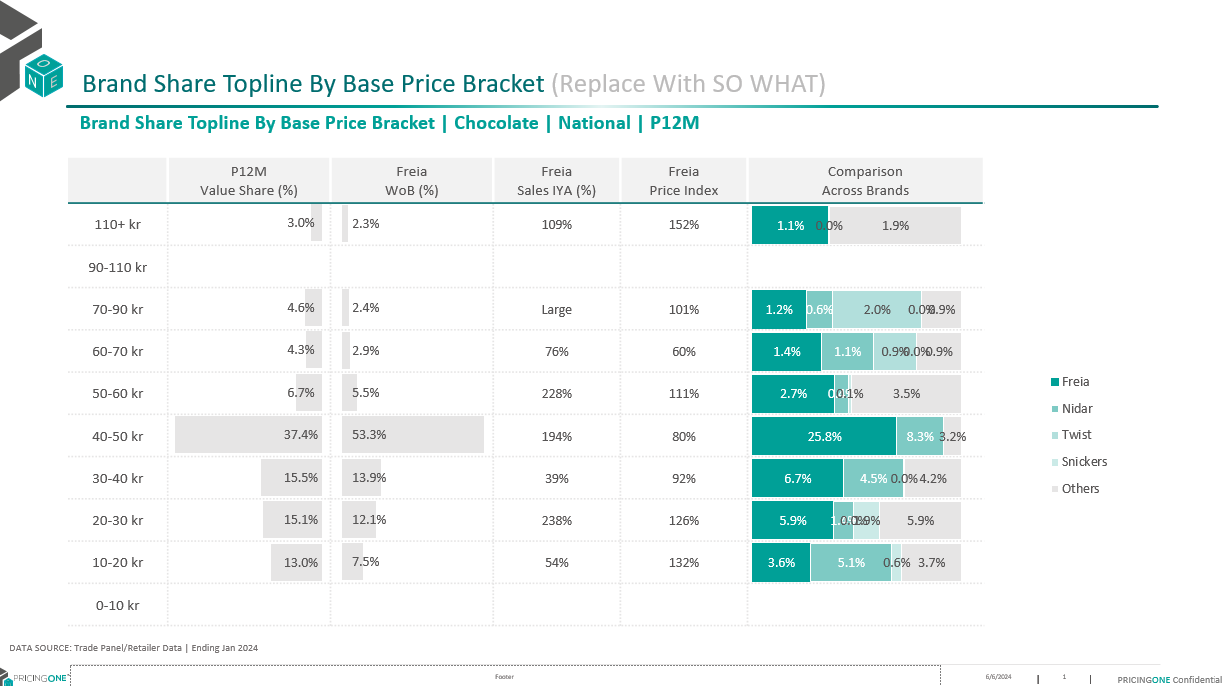
#### [Step 5: Replace Data in Slide](https://github.com/khaledSeifEleslam/Slide-Automate/blob/main/PPA%20slide%20duplicate/PPA%20Duplicate.ipynb)

# Initialize the starting position for slides position = 0# Determine the number of duplicates for the current section numOfDuplicates = duplication[sectionPosition] # Iterate over each client brandfor i, clientElement inenumerate(clientBrand): # Call the brandShareTopline function to generate slides for each client brand# Parameters:# - prs: The PowerPoint presentation object# - ToplineBracket: Dictionary containing data frames for different markets# - bracketsValue: List of size brackets to be considered# - numOfDuplicates/len(clientBrand): Number of slides to create or modify for each brand# - slide\_by: Column name used to determine the bracket type (specified by columnName)# - clientElement: Name of the client brand to highlight in the presentation# - position: Position offset for the slides (updated in each iteration) brandShareTopline(prs, ToplineBracket, bracketsValue, int(numOfDuplicates / len(clientBrand)), slide\_by=columnName, clientElement=clientElement, position=position) # Update the position for the next set of slides# Dividing numOfDuplicates by 3 to spread out the positions evenly position = int(position + numOfDuplicates / 3) # Move to the next section position sectionPosition += 1

#### [Step 6: Save Presentation](https://github.com/khaledSeifEleslam/Slide-Automate/blob/main/PPA%20slide%20duplicate/PPA%20Duplicate.ipynb)

# Define the path for saving the output presentation# This combines the current working directory with the column name and "output.pptx" outputPath = os.getcwd() + '\\' + columnName + " output.pptx"# Save the PowerPoint presentation to the specified output path prs.save(outputPath) # Use the win32 library to open the saved PowerPoint presentation# This dispatches the PowerPoint application and opens the presentation app = win32.Dispatch("PowerPoint.Application") presentation = app.Presentations.Open(outputPath)

##### Example: OutPut Slide After Replacement Data "Brand Share Topline By Size Bracket Slide OutPut"



## Mix&Assortment Section

### Introduction

In slide automation Mix&Assortment Section using 7 basic slides we have created 7 sections

* SKU Share By Brand
* Cumulative Product Shares
* Top 50% cumulative share
* Brand Cumulative Product Share
* Top 20 cumulative share
* SKU Productivity Analysis with TM%
* SKU Productivity Analysis with WD

### Project Steps

* Project Flow ![Project Flow](../Slides Documantion/duplication\_Steps.PNG)
* [Step 1: Import Libraries we use](#step-1-import-libraries)
* [Step 2: modified Data frames: cleaning and preprocessing the data frames](#step-2-modified-data-frame)
* [Step 3: Write Functions to Create Slides: Define functions to dynamically generate slides based on the base slides](#step-3-write-functions-to-create-slide)
* [Step 4: Duplicate Slides: Use functions or methods to duplicate existing slides as needed for the presentation.](#step-4-duplicate-slides)
* [Step 5: Replace Data in Slide: update information from the cleaned data frames to slides](#step-5-replace-data-in-slide)
* [Step 6: Save Presentation](#step-6-save-presentation)

#### [Step 1: Import Libraries we use](https://github.com/khaledSeifEleslam/Slide-Automate/blob/main/general_functions/generalFunctions.ipynb)

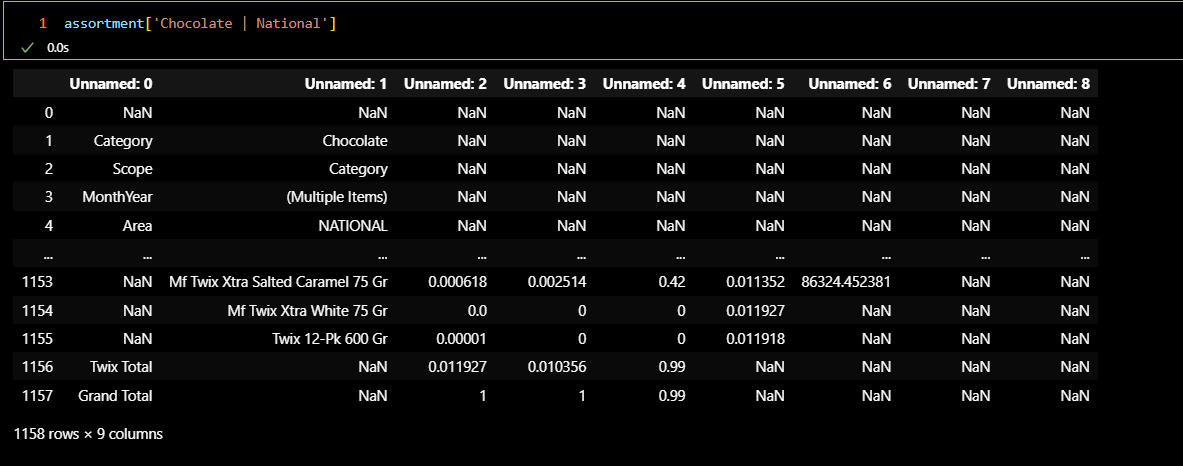
##### Ex: Libraries we use

* This script sets up an environment for working with PowerPoint presentations, data manipulation, filesystem operations, and COM (Component Object Model) object access.
* It imports necessary modules such as 'pptx' for PowerPoint automation, 'win32com' for COM object access and Windows automation, 'pandas' and 'numpy' for data manipulation,
* 'pathlib' for working with filesystem paths, 're' for regular expression operations, and various other modules for general-purpose tasks like file operations and timing functions.
* By importing these modules, the script prepares itself for tasks such as creating or modifying PowerPoint presentations, analyzing data using pandas and numpy, interacting
* with the Windows environment using win32com, and performing filesystem operations using shutil and os. Overall, this script provides a comprehensive setup for automating tasks
* related to PowerPoint presentations and general-purpose Python programming.

# Import necessary module for working with PowerPoint presentationsfrom pptx import Presentation # Import the win32com.client module, aliasing it as win32 for convenienceimport win32com.client as win32 # Import pandas for data manipulation and analysisimport pandas as pd # Import numpy for numerical computingimport numpy as np # Import the Path class from pathlib for working with filesystem pathsfrom pathlib import Path # Import re for regular expression operationsimport re # Import sys for access to interpreter-related functionsimport sys # Import time for various time-related functionsimport time # Assign win32.constants to a shorter alias win32c for easier access win32c = win32.constants # Import shutil for high-level file operationsimport shutil # Import os for operating system dependent functionalityimport os # Import win32com.client again for COM object and functions accessimport win32com.client # Import warnings for warning control functionalityimport warnings

#### [Step 2: modified Data frames](https://github.com/khaledSeifEleslam/Slide-Automate/blob/main/Assortment%20Slide%20Duplicate/Assortment%20Duplicate.ipynb)

##### EX: input dataframes before cleaning

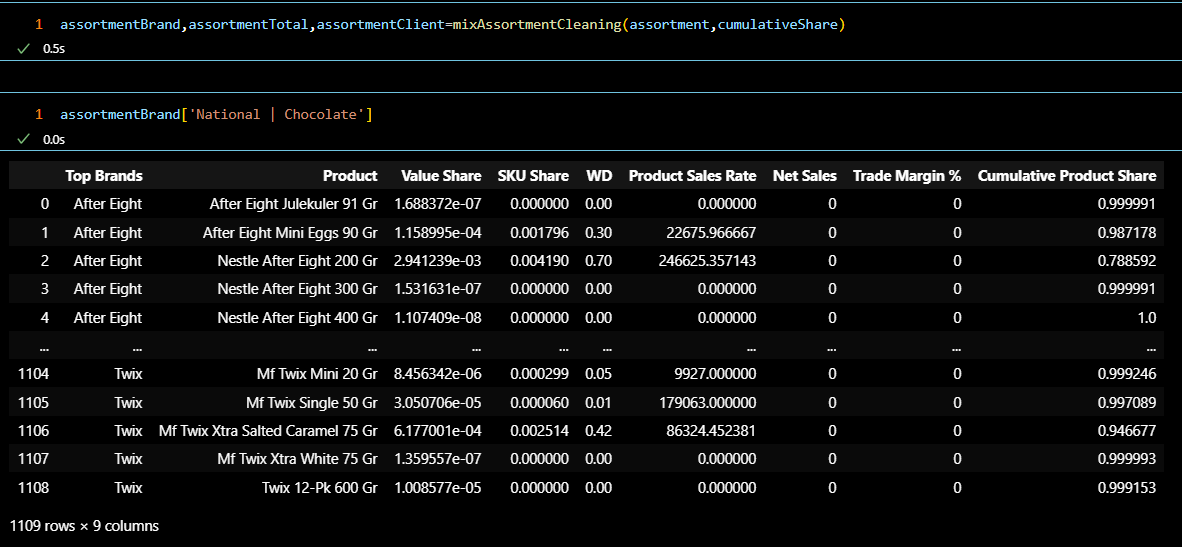


##### cleaning Code

* [mixAssortmentCleaning function](https://github.com/khaledSeifEleslam/Slide-Automate/blob/main/Assortment%20Slide%20Duplicate/Assortment%20Duplicate.ipynb): Cleans and processes assortment and cumulative share data to provide modified data for brand-specific analysis.
  + Args: assortment (dict): Dictionary containing assortment data. cumulativeShare (dict): Dictionary containing cumulative share data.
  + Returns: tuple:
    - assortmentModifiedBrand (dict): Dictionary containing cleaned and modified assortment data for brands.
    - assortmentModifiedTotal (dict): Dictionary containing cleaned and modified total assortment data.
    - assortmentClient (dict): Dictionary containing client-specific cleaned and modified assortment data.

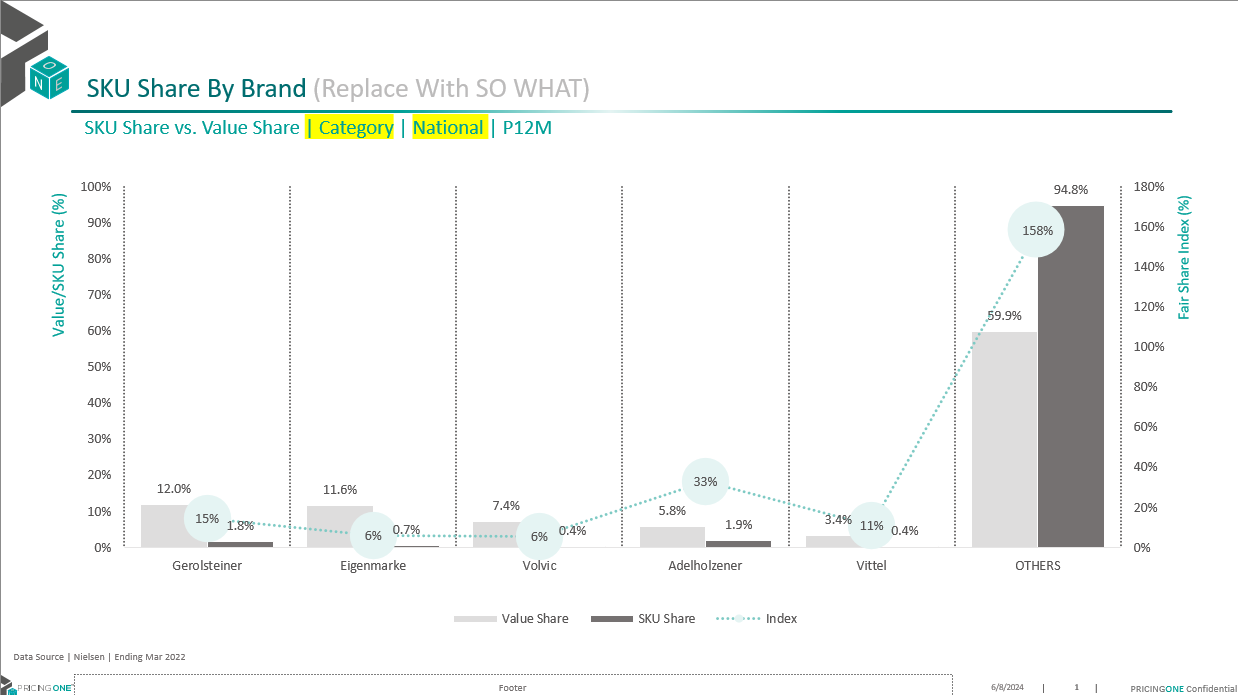
defmixAssortmentCleaning(assortment, cumulativeShare):# Initialize dictionaries to store modified data cumulativeShareModifiedBrand = {} assortmentModified = {} assortmentModifiedBrand = {} assortmentModifiedTotal = {} assortmentClient = {} # Process cumulative share datafor key, value in cumulativeShare.items(): dfcumulative = value.iloc[11:].reset\_index(drop=True) # Adjust the dataframe to remove unnecessary rows dfcumulative.columns = dfcumulative.iloc[0] # Set the first row as the column headers dfcumulative = dfcumulative.iloc[1:] # Remove the row used for headersif dfcumulative.shape[0] != 0: # If the dataframe is not empty newKey = key if key.split(' | ')[0] notin categories: # Adjust key if it does not match category format newKey = key.split(' | ')[1] + ' | ' + key.split(' | ')[0] cumulativeShareModifiedBrand[newKey] = dfcumulative.replace(np.nan, 0) # # Process assortment datafor key, value in assortment.items(): df = value.iloc[12:].reset\_index(drop=True) # Adjust the dataframe to remove unnecessary rows df.columns = df.iloc[0] # Set the first row as the column headers df = df.iloc[1:] # Remove the row used for headers df['Top Brands'] = df['Top Brands'].ffill() # Forward fill 'Top Brands' colum# Replace specific values in 'Top Brands' as per 'valueToReplace' dictionaryfor val, replacer in valueToReplace.items(): df['Top Brands'] = df['Top Brands'].str.replace(val, replacer) dfBrand = df[~df['Top Brands'].str.contains('Total')] # Filter out rows containing 'Total' in 'Top Brands' dfTotal = df[df['Top Brands'].str.contains('Total') & (df['Top Brands'] != 'Grand Total')].reset\_index(drop=True) dfTotal['Top Brands'] = dfTotal['Top Brands'].str.replace(' Total', '') # Adjust 'Top Brands' column for total rowsif df.shape[0] != 0: # If the dataframe is not empty newKey = key if key.split(' | ')[0] notin categories: # Adjust key if it does not match category format newKey = key.split(' | ')[1] + ' | ' + key.split(' | ')[0] # Process client-specific data for each brandfor brand in client\_brands\_competitor: if df[df['Top Brands'] == brand].shape[0] > 0: # Check if brand data exists in dataframe assortmentClient[newKey + ' | ' + brand] = df[df['Top Brands'] == brand].replace(np.nan, 0) assortmentClient[newKey + ' | ' + brand] = assortmentClient[newKey + ' | ' + brand].merge( cumulativeShareModifiedBrand[newKey], how='left', on='Product') # Store modified data for total, brand, and overall assortment assortmentModified[newKey] = df.replace(np.nan, 0) assortmentModifiedBrand[newKey] = dfBrand.replace(np.nan, 0).drop(columns=['Cumulative Product Share']) assortmentModifiedBrand[newKey] = assortmentModifiedBrand[newKey].merge( cumulativeShareModifiedBrand[newKey], how='left', on='Product') assortmentModifiedTotal[newKey] = dfTotal.replace(np.nan, 0) return assortmentModifiedBrand, assortmentModifiedTotal, assortmentClient

##### Calling function & data frame After cleaning



#### [Step 3: Write Functions to Create Slides](https://github.com/khaledSeifEleslam/Slide-Automate/blob/main/general_functions/Assortment%20Replacement%20Function.ipynb)

##### Example Slide : SKU Share By Brand



##### Replacement function

* [SkuShareByBrand function](https://github.com/khaledSeifEleslam/Slide-Automate/blob/main/general_functions/Assortment%20Replacement%20Function.ipynb):Updates PowerPoint slides with SKU and Value Share data by brand for a given market.
  + Args: prs (Presentation): The PowerPoint presentation object. assortmentTotalSorted (dict): Dictionary containing sorted assortment data by market. numOfDuplicates (int): Number of slides to duplicate and update. position (int, optional): Starting position for slide updates. Default is 0.

defSkuShareByBrand(prs, assortmentTotalSorted, numOfDuplicates, position=0):for slidenum inrange(numOfDuplicates): market = list(assortmentTotalSorted.keys())[slidenum] df = assortmentTotalSorted[market].copy() # Sort the dataframe by 'Value Share' in descending order df = df.sort\_values('Value Share', ascending=False) # Ensure that 'Others' is the last row in the dataframe df = pd.concat([df[df['Top Brands'] != 'Others'], df[df['Top Brands'] == 'Others']]).reset\_index(drop=True) # Get the shapes in the current slide shapes = prs.slides[slidenum + position].shapes # Update text in specific shapes shapes[4].text = data\_source shapes[5].text = shapes[5].text.replace('National', market.split(' | ')[1]).replace('Category', market.split(' | ')[0]) # Format the text in the shapes shapes[5].text\_frame.paragraphs[0].font.size = Pt(12) shapes[5].text\_frame.paragraphs[0].font.name = 'Nexa Bold (Headings)' shapes[6].text\_frame.paragraphs[0].font.size = Pt(16) shapes[6].text\_frame.paragraphs[0].font.name = 'Nexa Bold (Headings)'# Create tables and charts from the shapes tables, charts = createTableAndChart(shapes) chart = charts[0].chart # Prepare chart data chart\_data = CategoryChartData() chart\_data.categories = df['Top Brands'].tolist() chart\_data.add\_series('Value Share', df['Value Share']) chart\_data.add\_series('SKU Share', df['SKU Share']) # Calculate the index (SKU Share / Value Share) and handle division by zero chart\_data.add\_series('Index', df['SKU Share'] / df['Value Share'].replace(0, 1)) # Replace the chart data with the prepared data chart.replace\_data(chart\_data)

#### [Step 4: Duplicate Slides](https://github.com/khaledSeifEleslam/Slide-Automate/blob/main/Assortment%20Slide%20Duplicate/Assortment%20Duplicate.ipynb)

[Duplicate Slides](https://github.com/khaledSeifEleslam/Slide-Automate/blob/main/Assortment%20Slide%20Duplicate/Assortment%20Duplicate.ipynb):generate a PowerPoint presentation with multiple sections, each containing data visualizations and tables

# Define the indexes for different sections of the presentation index = [0, 1, 2, 3, 4, 5, 6] # Calculate the number of duplicates for each section based on the length of their respective data sets duplication = [ len(assortmentTotalSorted.keys()), len(assortmentBrand.keys()), len(cumulativeShareTop50.keys()), len(assortmentClient.keys()), len(assortmentBrand.keys()), len(assortmentClientBrand.keys()), len(assortmentClientBrand.keys()) ] # Define the names for each section of the presentation section\_names = [ "SKU Share By Brand", "Cumulative Product Shares", "Top 50% cumulative share", "Brand Cumulative Product Share", "Top 20 cumulative share", "SKU Productivity Analysis with TM%", "SKU Productivity Analysis with WD" ] # Define paths for the base slide template and the new presentation to be created path = os.getcwd() + '//slide base.pptx' new\_pre = os.getcwd() + '//slide duplicated.pptx'# Initialize the position and the number of duplicates for each section position = 0# Loop through each client brand and create slides based on the provided datafor i, clientElement inenumerate(clientBrand): # Generate the required slides for each brand based on the calculated duplication brandShareTopline( prs, ToplineBracket, bracketsValue, int(numOfDuplicates / len(clientBrand)), slide\_by=columnName, clientElement=clientElement, position=position ) # Update the position for the next set of slides position += int(numOfDuplicates / 3) # Increment the section position sectionPosition += 1# Define the path for saving the output presentation outputPath = os.getcwd() + '\\' + columnName + " output.pptx"# Save the PowerPoint presentation to the specified output path prs.save(outputPath) # Use the win32 library to open the saved PowerPoint presentation app = win32.Dispatch("PowerPoint.Application") presentation = app.Presentations.Open(outputPath)

#### [Step 5: Replace Data in Slide](https://github.com/khaledSeifEleslam/Slide-Automate/blob/main/Assortment%20Slide%20Duplicate/Assortment%20Duplicate.ipynb)

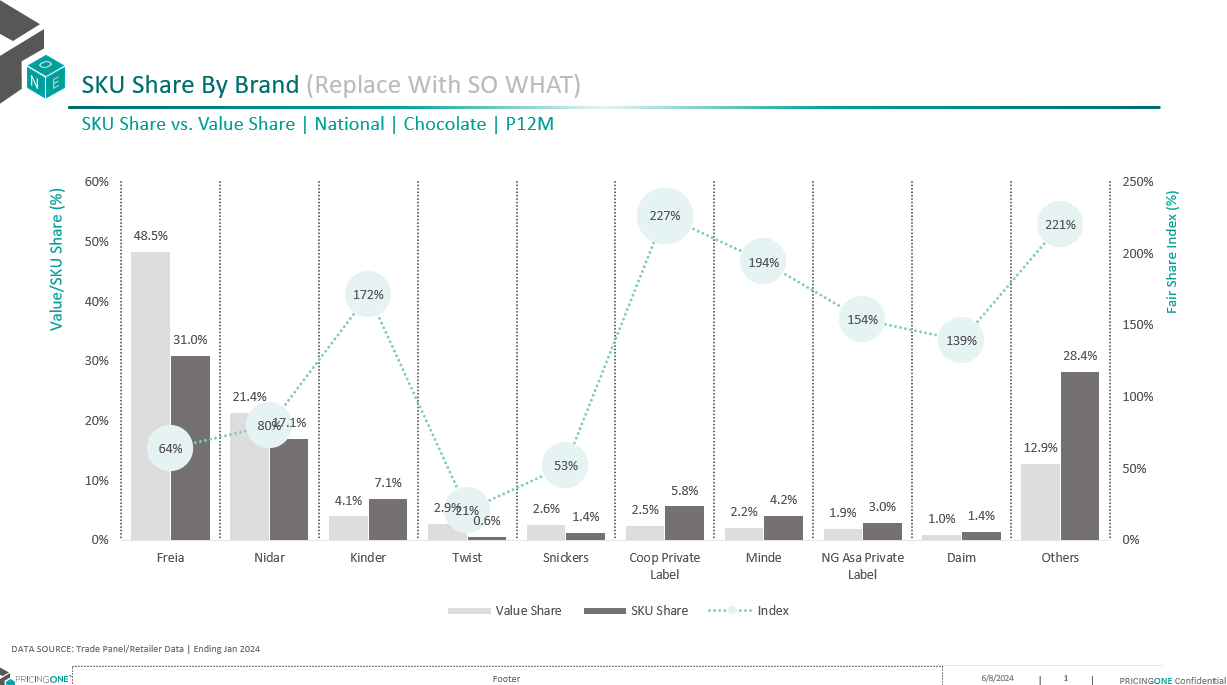
* Generate the slides for SKU Share By Brand using the provided data.
  + prs: The PowerPoint presentation object where slides will be added or modified.
  + assortmentTotalSorted: The data set containing SKU share information, sorted accordingly.
  + duplication[0]: The number of duplicates to create for this section, specified by the first element in the duplication list.
  + position=0: The starting position for slide creation in this section.

SkuShareByBrand(prs, assortmentTotalSorted, duplication[0], position=0)

#### [Step 6: Save Presentation](https://github.com/khaledSeifEleslam/Slide-Automate/blob/main/Assortment%20Slide%20Duplicate/Assortment%20Duplicate.ipynb)

# Set the output path for the PowerPoint presentation outputPath = os.getcwd() + "\\Mix and assortment doc output.pptx"# Save the current PowerPoint presentation to the specified path prs.save(outputPath) # Open the saved PowerPoint presentation using the PowerPoint application app = win32.Dispatch("PowerPoint.Application") presentation = app.Presentations.Open(outputPath)

##### Example: OutPut Slide After Replacement Data "SKU Share By Brand"



## Promotion Section

### Introduction

In slide automation Promotion Section using 20 basic slides we have created 17 sections

* Promo Value Sales
* Promo Evolution
* VSOD Summary
* Value uplift by retailer by brand
* Volume Uplift vs discount depth
* Value Uplift vs Promo Efficiency Quadrant
* Top 20 promotions
* Top 20 promotions CLIENT ONLY
* Bottom 20 promotions CLIENT ONLY
* Volume Sold on Deal
* Promo share vs Value Share
* Promo Sales by total size
* Promo Frequency learnings
* Category Promo sales per retailer
* Sector Promo sales per retailer
* Segment Promo sales per retailer
* Value Uplift vs discount depth

### Project Steps

* Project Flow ![Project Flow](../Slides Documantion/duplication\_Steps.PNG)
* [Step 1: Import Libraries we use](#step-1-import-libraries)
* [Step 2: modified Data frames: cleaning and preprocessing the data frames](#step-2-modified-data-frame)
* [Step 3: Write Functions to Create Slides: Define functions to dynamically generate slides based on the base slides](#step-3-write-functions-to-create-slide)
* [Step 4: Duplicate Slides: Use functions or methods to duplicate existing slides as needed for the presentation.](#step-4-duplicate-slides)
* [Step 5: Replace Data in Slide: update information from the cleaned data frames to slides](#step-5-replace-data-in-slide)
* [Step 6: Save Presentation](#step-6-save-presentation)

#### [Step 1: Import Libraries we use](https://github.com/khaledSeifEleslam/Slide-Automate/blob/main/general_functions/generalFunctions.ipynb)

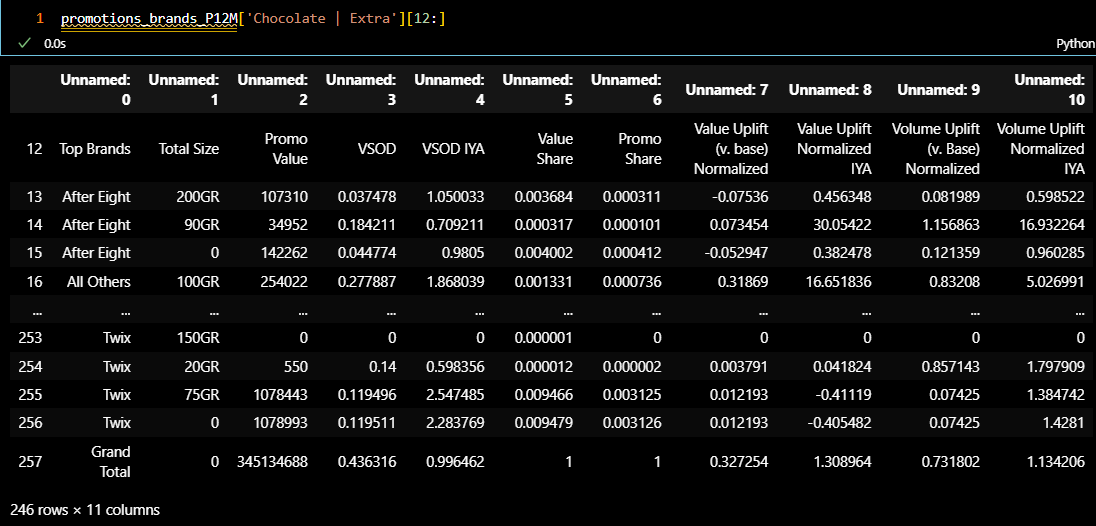
##### Ex: Libraries we use

* This script sets up an environment for working with PowerPoint presentations, data manipulation, filesystem operations, and COM (Component Object Model) object access.
* It imports necessary modules such as 'pptx' for PowerPoint automation, 'win32com' for COM object access and Windows automation, 'pandas' and 'numpy' for data manipulation,
* 'pathlib' for working with filesystem paths, 're' for regular expression operations, and various other modules for general-purpose tasks like file operations and timing functions.
* By importing these modules, the script prepares itself for tasks such as creating or modifying PowerPoint presentations, analyzing data using pandas and numpy, interacting
* with the Windows environment using win32com, and performing filesystem operations using shutil and os. Overall, this script provides a comprehensive setup for automating tasks
* related to PowerPoint presentations and general-purpose Python programming.

# Import necessary module for working with PowerPoint presentationsfrom pptx import Presentation # Import the win32com.client module, aliasing it as win32 for convenienceimport win32com.client as win32 # Import pandas for data manipulation and analysisimport pandas as pd # Import numpy for numerical computingimport numpy as np # Import the Path class from pathlib for working with filesystem pathsfrom pathlib import Path # Import re for regular expression operationsimport re # Import sys for access to interpreter-related functionsimport sys # Import time for various time-related functionsimport time # Assign win32.constants to a shorter alias win32c for easier access win32c = win32.constants # Import shutil for high-level file operationsimport shutil # Import os for operating system dependent functionalityimport os # Import win32com.client again for COM object and functions accessimport win32com.client # Import warnings for warning control functionalityimport warnings

#### [Step 2: modified Data frame](https://github.com/khaledSeifEleslam/Slide-Automate/blob/main/Promotion%20Slide%20Duplicate/Promotion%20Duplicate.ipynb)

##### EX: input dataframes before cleaning

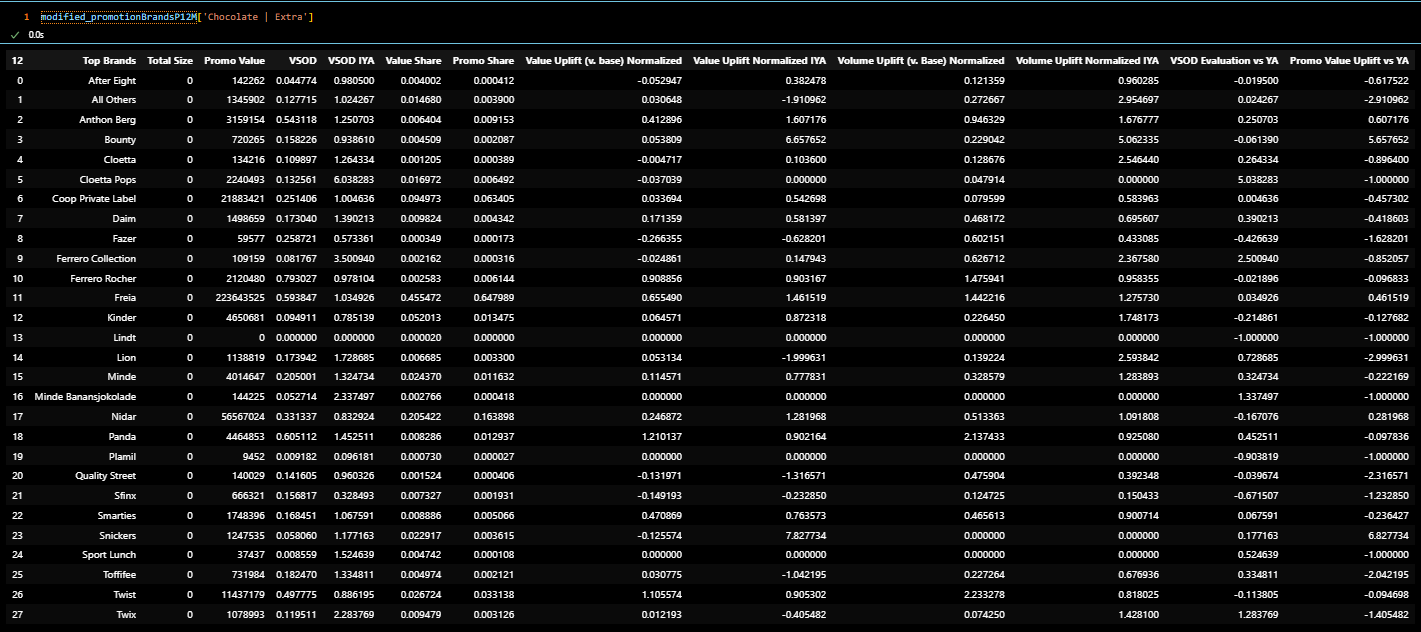


##### Cleaning Data Frame

* [cleaningData function](https://github.com/khaledSeifEleslam/Slide-Automate/blob/main/Promotion%20Slide%20Duplicate/Promotion%20Duplicate.ipynb): Clean and preprocess data in a dictionary of DataFrames.
  + Parameters: data (dict): Dictionary containing DataFrames.
  + Returns: dict: Dictionary containing cleaned DataFrames.

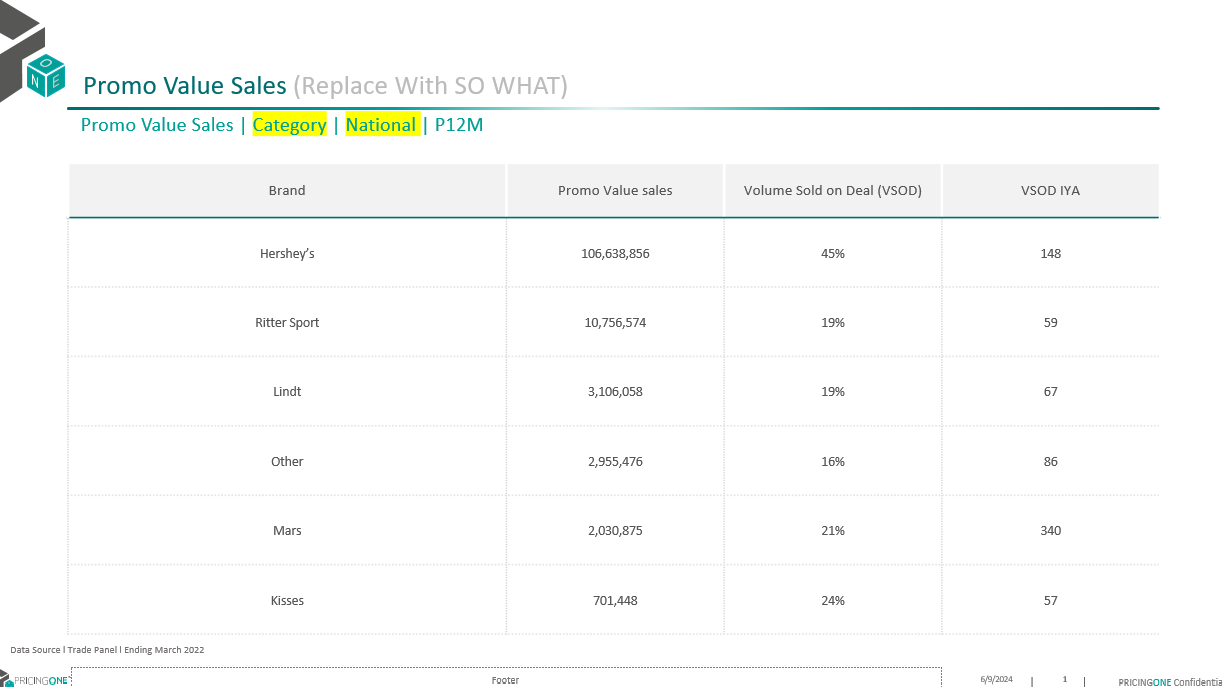
defcleaningData(data): cleaned\_data = {} # Iterate over each key-value pair in the input dictionaryfor key in data: # Skip the first 11 rows if there are NaN values df = data[key].iloc[11:] if data[key].iloc[11,:].isna().any(): df = data[key].iloc[12:] # Set column names and skip the first row df.columns = df.iloc[0] df = df.iloc[1:] # Perform specific cleaning operations based on the DataFrame columns and keyif df.shape[0] > 0andnot'National'in key: if'Top Brands'in df.columns and'Product'in df.columns: df['Top Brands'] = df['Top Brands'].fillna(method='ffill') df['Product'].fillna('', inplace=True) df.fillna(0, inplace=True) df['Top Brands'] = df['Top Brands'].apply(lambda x: 'Grand Total'if'Grand Total'in x else x.replace('Total', '').strip()) elif'Top Brands'in df.columns: df['Top Brands'] = df['Top Brands'].fillna(method='ffill') df.fillna(0, inplace=True) df['Top Brands'] = df['Top Brands'].apply(lambda x: 'Grand Total'if'Grand Total'in x else x.replace('Total', '').strip()) df = df[~df['Top Brands'].str.contains('Total', case=False)] df = df[df['Total Size'] == 0].reset\_index(drop=True) df['VSOD Evaluation vs YA']=df['VSOD IYA']-1 df['Promo Value Uplift vs YA']=df['Value Uplift Normalized IYA']-1elif'End of Week'in df.columns and'Product'in df.columns: df['Product'] = df['Product'].fillna(method='ffill') df = df[(df['End of Week'].str.contains('2023|2024')) & (df['End of Week'].notna())] df['End of Week'] = pd.to\_datetime(df['End of Week']) df = df[(df['End of Week'] >= start\_date) & (df['End of Week'] <= end\_date)] df = df[~df['Product'].str.contains('Total', case=False)].reset\_index(drop=True) df = df[df['Promo Sales'] > 10000] df = df.dropna(subset=['Value Uplift (v. base) Normalized']) df.fillna(0, inplace=True) df = df.reset\_index(drop=True) elif'End of Week'in df.columns: df['End of Week'] = df['End of Week'].astype(str) df = df[~df['End of Week'].str.contains('Total', case=False)].reset\_index(drop=True) df['End of Week'] = pd.to\_datetime(df['End of Week']) df['End of Week'] = df['End of Week'].dt.strftime("%d-%b-%y") df = df[(df['End of Week'].str.contains('-21|-22|-23|Jan-24')) & (df['End of Week'].notna())] df['End of Week'] = pd.to\_datetime(df['End of Week']) df = df[(df['End of Week'] >= start\_date) & (df['End of Week'] <= end\_date)] df = df.dropna() elif'Grand Total'in df.columns: df.fillna(0, inplace=True) # Check if the key matches specific categories and modify the key accordinglyif key.split(' | ')[0] in categories andlen(key.split(' | ')) == 3: modified\_key = key.split(' | ')[1] + ' | ' + key.split(' | ')[2] + ' | ' + key.split(' | ')[0] if df.shape[0] > 0: cleaned\_data[modified\_key] = df else: if df.shape[0] > 0: cleaned\_data[key] = df return cleaned\_data

##### Example: out put data frame after cleaning



#### [Step 3: Write Functions to Create Slides](https://github.com/khaledSeifEleslam/Slide-Automate/blob/main/Promotion%20Slide%20Duplicate/Promotion%20Duplicate.ipynb)

##### Example slide : Promo Value Sales base slide



##### Replacement function

* promoValueSales function:Generate PowerPoint slides for promo value sales

defpromoValueSales(prs, promotionsBrandDF, numOfDuplicates, position=0):# Loop through each slide numberfor slidenum inrange(numOfDuplicates): # Get market from promotionsBrandDF keys market = list(promotionsBrandDF.keys())[slidenum] # Retrieve DataFrame for the current market df = promotionsBrandDF[market].reset\_index(drop=True) # Remove rows with 'Others' in 'Top Brands' column and filter by 'Value Share' df = df[~df['Top Brands'].str.contains('Others', case=False)] df = df[df['Value Share'] > 0.01] # Select client brands df\_client = selectClientBrands(promotionsBrandDF[market],'Top Brands', 'Promo Value') number\_of\_brands\_needed = 5 - len(df\_client) # Filter top brands and concatenate with client brands df = df[~df['Top Brands'].isin(client\_brands)] df = df.sort\_values(by='Promo Value', ascending=False).head(number\_of\_brands\_needed) df = pd.concat([df, df\_client], ignore\_index=True) df = df.sort\_values(by='Promo Value', ascending=False) # Update title shapes = prs.slides[slidenum + position].shapes titlNumber = get\_shape\_number(shapes, "Promo Value Sales | Category | National | P12M") shapes[titlNumber - 1].text = data\_source shapes[titlNumber + 1].text\_frame.paragraphs[0].font.size = Pt(16) shapes[titlNumber + 1].text\_frame.paragraphs[0].font.name = 'Nexa Bold (Headings)' shapes[titlNumber].text = shapes[titlNumber].text.replace('Category', market.split(' | ')[0]).replace( 'National', market.split(' | ')[1]) shapes[titlNumber].text\_frame.paragraphs[0].font.size = Pt(12) shapes[titlNumber].text\_frame.paragraphs[0].font.name = 'Nexa Bold (Headings)'# Create table and chart tables, charts = createTableAndChart(shapes) table = tables[0].table # Remove unnecessary rows num\_rows\_to\_remove = len(table.rows) - df['Top Brands'].nunique() - 1 table\_height = get\_table\_height(table) for \_ inrange(num\_rows\_to\_remove): iflen(table.rows) > 1: row = table.rows[1] remove\_row(table, row) # Adjust row heights total\_row\_height = table\_height - table.rows[0].height num\_rows = len(table.rows) - 1if num\_rows > 0: cell\_height = total\_row\_height / num\_rows for row inrange(1, table.rows.\_\_len\_\_()): table.rows[row].height = int(cell\_height) # Populate table cellsfor i, row inenumerate(table.rows): for j, cell inenumerate(row.cells): if i == 0: # Header rowcontinueif j == 0: # Brand column cell.text = df['Top Brands'].iloc[i - 1] cell.text\_frame.paragraphs[0].font.name = 'Nexa Bold'elif j == 1: # Promo Value sales column value = df['Promo Value'].iloc[i - 1] iflen(str(value)) > 3: formatted\_value = '{:,}'.format(int(value)) cell.text = str(formatted\_value) cell.text\_frame.paragraphs[0].font.name = 'Nexa Book'else: cell.text = str(df['Promo Value'].iloc[i - 1]) cell.text\_frame.paragraphs[0].font.name = 'Nexa Book'elif j == 2: # Volume Sold on Deal (VSOD) column cell.text = str(int(round(df['VSOD'].replace(np.nan, 0).iloc[i - 1] \* 100, 0))) + '%' cell.text\_frame.paragraphs[0].font.name = 'Nexa Book'else: # VSOD IYA column cell.text = str(int(round(df['VSOD IYA'].replace(np.nan, 0).iloc[i - 1] \* 100, 0))) cell.text\_frame.paragraphs[0].font.name = 'Nexa Book'# Set font size and alignment cell.text\_frame.paragraphs[0].font.size = Pt(8) cell.text\_frame.paragraphs[0].alignment = PP\_ALIGN.CENTER

#### [Step 4: Duplicate Slides](https://github.com/khaledSeifEleslam/Slide-Automate/blob/main/general_functions/Promotion%20Replacement%20Function.ipynb):

* This code generates indices, duplication factors, and section names for PowerPoint slides
* based on different promotional data sources. It sets up paths for the base and duplicated slides,
* and ensures the correct indices and duplication values for each section of the presentation.

# Generate indices for slides containing promo value data for different categories, sectors, and segments slidePromoValueIndex = [ [i + 15for i in catDuplication.values()], # Adjust category duplication indices by adding 15 [i + 15for i in secDuplication.values()], # Adjust sector duplication indices by adding 15 [i + 15for i in segDuplication.values()] # Adjust segment duplication indices by adding 15 ] # Create a list of slide indices, conditional on the presence of promo type, feature share, and display share data index = [ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12if promo\_type != FalseelseNone, # Conditional index for promo type13if feature\_share != FalseelseNone, # Conditional index for feature share14if display\_share != FalseelseNone, # Conditional index for display share15, \*slidePromoValueIndex, 20# Base index 15, adjusted promo value indices, and final index 20 ] # Remove None values from the index list index = [x for x in index if x isnotNone] # Calculate the lengths of various datasets len\_brands = len(modified\_promotionBrandsP12M) len\_Prod = len(modified\_promotionProductsP12M) len\_modified\_prod = len(new\_modified\_promotionProductsP12M) len\_client\_market = len(client\_brands) \* len(regions\_RET) # Define duplication factors for each section based on the lengths of relevant datasets duplication = [ len\_brands, len(promotionsBrandSortedTotal), len(promotionsBrandsWithMarket), len(concated), len\_Prod, len\_modified\_prod, len\_modified\_prod, len(top20clientonly), len(bottom20clientonly), len\_client\_market, len\_brands, len(newModifiedBrands), len(newModifiedBrands) if promo\_type != FalseelseNone, # Conditional duplication factor for promo type len\_brands if feature\_share != FalseelseNone, # Conditional duplication factor for feature share len\_brands if display\_share != FalseelseNone, # Conditional duplication factor for display sharelen(modified\_promotionEndOfWeek), 1, 1, 1iflen(segments) > 0elseNone, # Conditional duplication factor for segmentslen(modified\_valueUplift) ] # Remove None values from the duplication list duplication = [x for x in duplication if x isnotNone] # Define section names for each part of the presentation section\_names = [ "Promo Value Sales", "Promo Evolution", "VSOD Summary", "Value uplift by retailer by brand", "Volume Uplift vs discount depth", "Value Uplift vs Promo Efficiency Quadrant", "Top 20 promotions", "Top 20 promotions CLIENT ONLY", "Bottom 20 promotions CLIENT ONLY", "Volume Sold on Deal", "Promo share vs Value Share", "Promo Sales by total size", "Promo Sales by promo type"if promo\_type != FalseelseNone, # Conditional section name for promo type"Feature Share vs Fair Share"if feature\_share != FalseelseNone, # Conditional section name for feature share"Display Share vs Fair Share"if display\_share != FalseelseNone, # Conditional section name for display share"Promo Frequency learnings", "Category Promo sales per retailer", "Sector Promo sales per retailer", "Segment Promo sales per retailer"iflen(segments) > 0elseNone, # Conditional section name for segments"Value Uplift vs discount depth" ] # Remove None values from the section names list section\_names = [x for x in section\_names if x isnotNone] # Define paths for the base PowerPoint file and the duplicated PowerPoint file path = os.getcwd() + '//slide base.pptx' new\_pre = os.getcwd() + '//slide duplicated.pptx'# Define the data source string to be used in the presentation data\_source = "DATA SOURCE: Trade Panel/Retailer Data | Ending Jan 2024"

#### [Step 5: Replace Data in Slides](https://github.com/khaledSeifEleslam/Slide-Automate/blob/main/general_functions/Promotion%20Replacement%20Function.ipynb)

* Call the promoValueSales function to generate slides for promotional value sales
* prs: PowerPoint presentation object
* modified\_promotionBrandsP12M: Dictionary containing promotion data for different markets
* duplication[posItr]: Number of slides to duplicate for the current market
* position=posItr: Starting position to add slides in the presentation

promoValueSales(prs, modified\_promotionBrandsP12M, duplication[posItr], position=posItr) # Increment the position iterator by 1 to move to the next section for the next function call posItr += 1

#### [Step 6: Save Presentation](https://github.com/khaledSeifEleslam/Slide-Automate/blob/main/general_functions/Promotion%20Replacement%20Function.ipynb)

* performs two main tasks: saving the current PowerPoint presentation to a file and opening that file using the PowerPoint application. The outputPath variable is constructed using the current working directory, ensuring the presentation is saved in the correct location. After saving the presentation, the script uses win32com.client to dispatch the PowerPoint application and open the saved presentation. This automation allows for seamless transition from generating the presentation to viewing or editing it in PowerPoint, streamlining the workflow for creating market analysis slides.

# This script saves the generated PowerPoint presentation to a specified path# and then opens the saved presentation using the PowerPoint application.# Define the output path for the PowerPoint presentation outputPath=os.getcwd() + "\\Promotion doc output.pptx"# Save the PowerPoint presentation to the specified output path prs.save(outputPath) # Initialize the PowerPoint application using win32com client app = win32.Dispatch("PowerPoint.Application") # Open the saved PowerPoint presentation presentation = app.Presentations.Open(outputPath)

##### Example slide : Promo Value Sales slide after replacement

