# Analysis of the Streamlit Script for Presentation Generation

## General Overview

This Python script, designed to run in a Streamlit environment, automates the creation of presentations (both in PPTX and PDF formats) using a PowerPoint template and data from an Excel file. It allows users to customize report generation by selecting specific row ranges or IDs and configuring the names of the resulting files.

## Need

This tool was built to automate the process of generating customized presentations for reporting, reducing manual effort and improving accuracy. Its main goal is to have a user-friendly tool that can scale and will make the team save time.

## Pilot

The first version of the tool will be used by the customer success team to do the reporting of target by store campaigns to clients in France and Portugal.

## Main Features

* **Automated Presentation Generation:** The script uses a PPTX file as a template and an Excel file as a data source to generate multiple customized presentations.
* **Flexible Data Selection:** It enables filtering of Excel data by specific row ranges or a list of "Store IDs," providing flexibility in data selection.
* **Customizable File Names:** Users can personalize the output file names by selecting columns from the Excel file to construct the filename.
* **Versatile Output Format:** Supports the creation of files in both PPTX and PDF formats, allowing users to choose the most suitable format for their needs.
* **Preserving Excel Formatting:** The script maintains the original formatting of Excel cells (numbers, dates, percentages, currencies) when inserting data into presentations.
* **Integration with Streamlit Cloud:** The script is designed to work seamlessly in Streamlit Cloud, including PPTX to PDF conversion using LibreOffice.
* **User-Friendly Interface:** Built with Streamlit, the interface enables easy interaction through file uploads, option selections, and buttons.

## Key Ideas and Facts

* **Core Functionality:** The script utilizes the pptx library to manipulate PowerPoint files, pandas for Excel data processing, and openpyxl to read the specific formatting of Excel cells.
* **Text Replacement in PPTX:** The update\_text\_of\_textbox function searches for text patterns such as {A}, {B}, etc., inside the text boxes of the PPTX template and replaces them with the corresponding values from the Excel columns.
  + *Quote:* *"Searches and replaces text inside text boxes formatted as {A}, {B}, etc., while preserving the PPTX formatting."*
* **PDF Conversion (Linux):** The convert\_pptx\_to\_pdf function uses LibreOffice to convert PPTX files to PDF. This is crucial for the script's functionality in Streamlit Cloud.
  + *Quote:* *"Converts a PPTX file to PDF on Linux using LibreOffice (works in Streamlit Cloud)."*
* **File Name Generation:** The get\_filename\_from\_selection function generates filenames by concatenating selected Excel column values.
  + *Quote:* *"Generates the file name based on the selected columns."*
* **Row-by-Row Processing:** The process\_row function iterates through the selected DataFrame rows, replaces text in the PPTX template, and saves each presentation as an individual PPTX or PDF file.
  + *Quote:* *"Processes a row and generates a PPTX or PDF file while preserving the original Excel formatting."*
* **Cell Formatting:** The format\_cell\_value function retrieves the value of an Excel cell and applies the correct formatting (number, date, percentage, currency) before inserting it into the presentation.
  + *Quote:* *"Formats and rounds the cell value according to its type and format in Excel."*
* **User Interface:** The Streamlit-based interface includes components such as st.file\_uploader, st.radio, st.number\_input, st.text\_input, and st.multiselect to enable user interaction.
* **ZIP Output:** The create\_zip\_of\_presentations function generates a ZIP file containing all generated presentations, simplifying the download of multiple files.
  + *Quote:* *"Creates a ZIP file with all the generated PPTX files in the folder."*
* **Error Handling:** The script includes error handling for Excel file reading and PDF conversion, informing the user in case of failures.
* **Progress Bar:** The use of st.progress provides a visual progress bar during file generation, which is particularly useful when processing multiple reports.

## Current Status

At present, the tool is functional and enables:

* Uploading an Excel file that follows a specific rule.
* Selecting 2 data filters; row range and Store IDs.
* Generating PPTX presentations based on a template.
* Converting PPTX files into PDFs.
* Downloading the generated reports as a ZIP file.

## Planned Improvements

* Optimizing performance when creating under PDF format
* Reducing the number of “rules” to follow to allow more flexibility
  + First one would be to make the code find Store\_ID no matter its position. Nowadays we rely on Store ID being Column A.
* Gathering user feedback from the pilot phase to optimize its current capabilities and to expand the tool's features.
* Add a way to store files in the cloud using a cloud service such as AWS and automatically share the link of each report via email with each corresponding store owner.