PDPs Analysis Automation: User Guide

Version 1.0

Welcome to your new PDPs Analysis Automation tool! This guide will walk you through everything you need to know, from the initial technical setup to strategic use, all in simple, easy-to-understand language.

Note: This automation involves a local setup on your computer using Python and requires a one-time configuration with Google Cloud. Please follow the steps carefully.

Sheet Link: PDPs Analysis

Table of Contents

- Your Regular Workflow
- How to Add New Queries to Track
- Understanding the Output in Your Sheet
- Handling CAPTCHAs and Login Issues

Chapter 4: Strategies for Better Analysis

- How to Use the Data Strategically
- Use Cases for Deeper Insights
- How to Customize the Automation

Chapter 5: How the Magic Happens (A Simple Look at the Script)

- What are Python and Selenium?
- The Journey of a Query: The Script's Logic
- Important Timings to Know

Appendix: Troubleshooting & Frequently Asked Questions (FAQ)

- Common Problems and Solutions

Chapter 1: Introduction to the Tool

What is this Automation Tool?

Imagine you have a list of 100 important search queries, and you want to know how many search results Google shows for each one. Manually searching for each query and copying the result count is incredibly time-consuming and tedious. If you want to track this data over time to see trends, you'd have to repeat this process every week or month.

This tool solves that problem.

It is a smart system that runs on your computer. It automatically reads a list of search queries from a Google Sheet, performs a Google search for each one, and records the total number of results back into the same sheet. It creates a new column with the current date for each run, allowing you to build a historical record of search result volumes effortlessly.

Who is this Tool For?

This tool is designed for anyone who needs to track the volume of search results for specific keywords over time. It's perfect for:

- **SEO Specialists:** To monitor the SERP (Search Engine Results Page) landscape for target keywords.
- **Digital Marketers:** To analyze market size and competition for specific topics or products.
- **Market Researchers:** To track trends and the volume of online content related to certain subjects.
- **Content Strategists:** To identify changes in content volume around key pillars.

Key Features at a Glance

- **Fully Automated Scraping:** Just add your keywords to the Google Sheet and run the script.
- **Historical Data Logging:** Automatically creates a new, date-stamped column for each run, building a valuable historical dataset.
- **Human-Like Behavior:** Uses realistic delays and a dedicated browser profile to reduce the chance of being blocked by Google.
- **Smart CAPTCHA Handling:** If a CAPTCHA appears, the script pauses and waits for you to solve it manually before continuing.
- **Error Notifications:** Can be configured to send you an email alert if the script crashes, so you always know its status.

- **Centralized Management:** Your entire list of queries and all historical data are stored in one convenient Google Sheet.

How It Works: A Simple Overview

The process is straightforward. Think of it as giving a set of instructions to a very efficient robot assistant on your computer.

1. **You Provide the Queries:** You add the search terms you want to track into a specific column in your Google Sheet.
2. **You Configure the Settings:** In a simple configuration file, you tell the script where to find your project files and your Google Sheet.
3. **The Automation Kicks In:** When you run the script, it "wakes up" and starts its work.
4. **It Prepares the Sheet:** The script looks at your Google Sheet, finds the last column with data, and inserts a new column next to it, adding today's date to the header.
5. **It Checks Each Query:** For every query on your list, the script:
    - Opens a Chrome browser window.
    - Types the query into Google and searches.
    - Clicks the "Tools" button to reveal the result count.
    - Reads and copies the number of results (e.g., "71,60,000").
6.
7. **It Records Its Work:** The script pastes the result count into the new, date-stamped column in the correct row for that query.
8. **It Repeats:** It moves to the next query and repeats the process until the entire list is complete.

This cycle repeats every time you run the script, adding a new snapshot of data to your sheet.

Chapter 2: Getting Started: The Full Setup Process

This chapter will guide you through the one-time setup of your automation. This process is more technical than a simple sheet add-on, but following these steps carefully will ensure everything works perfectly.

Understanding Your Workspace: The Project Files

First, you will receive a folder containing all the necessary project files. Place this folder in a permanent location on your computer, like your `Documents` folder.

- `pdp_checker.py`: The main script that runs the automation.
- `config.py`: The central settings file where you'll add your specific paths and IDs.
- `create_master_profile.py`: A one-time script to set up your dedicated Chrome browser profile.
- `refresh_profile.py`: A script to refresh your Chrome profile if you encounter login issues.
- `gcp_credentials.json`: Your private key from Google Cloud (you will generate this).
- `requirements.txt`: A list of all the Python libraries the script needs to function.

Part A: Your Computer Setup

Step 1: Install Python

If you don't have Python installed, you'll need to install it.

1. Go to the official Python website: `https://www.python.org/downloads/`
2. Download the latest stable version (e.g., Python 3.10 or newer).
3. Run the installer. **Important:** On the first screen of the installer, make sure to check the box that says **"Add Python to PATH"**. This is crucial.
4. Follow the rest of the installation steps.

Step 2: Set Up the Project Folder & Virtual Environment (venv)

A virtual environment is a private, isolated space for our project's libraries, which prevents conflicts with other Python projects.

1. Open your computer's command prompt (CMD) or terminal.

- On Windows, press `Win + R`, type `cmd`, and press Enter.

2. Navigate to your project folder using the `cd` command. For example:
   `cd C:\Users\YourName\Documents\PDPs_Analysis_Project`
3. Once inside the folder, run the following command to create a virtual environment named `venv`:
   `python -m venv venv`
4. Now, activate the environment:
   - **On Windows:** `venv\Scripts\activate`
   - **On Mac/Linux:** `source venv/bin/activate`
     You will know it's active because you'll see `(venv)` at the beginning of your command prompt line.

Step 3: Install Required Libraries

While the virtual environment is active, install all the necessary libraries using the `requirements.txt` file.

1. Make sure you are in the project folder in your command prompt and that `(venv)` is visible.
2. Run this command:
   `pip install -r requirements.txt`
   This will automatically download and install Selenium, gspread, and all other dependencies.

Part B: Google Cloud & Sheets Setup

This part connects your script to your Google account securely.

Step 4: Configure Google Cloud Platform (GCP) & APIs

The script needs permission to read and write to your Google Sheet. We will create a "service account" (like a robot user) to grant these permissions safely.

1. **Create a GCP Project:**
   - Go to the Google Cloud Console.
   - Click the project dropdown at the top of the page and click **"New Project"**.
   - Give it a name (e.g., "Sheets Automation Project") and click **"Create"**.

2. **Enable APIs:**
    - Make sure your new project is selected.
    - In the search bar at the top, search for and enable the **"Google Drive API"**.
    - Search again and enable the **"Google Sheets API"**.
3. **Create a Service Account:**
    - In the search bar, search for **"Service Accounts"** and go to that page.
    - Click **"+ Create Service Account"**.
    - Give it a name (e.g., "pdp-sheet-manager") and a description. Click **"Create and Continue"**.
    - For the "Role", select **"Editor"**. This gives it permission to edit files. Click **"Continue"**, then **"Done"**.
4. **Generate JSON Key:**
    - You will now see your new service account in the list. Click on the three dots under "Actions" and select **"Manage keys"**.
    - Click **"Add Key"** -> **"Create new key"**.
    - Choose **JSON** as the key type and click **"Create"**.
    - A `.json` file will be downloaded to your computer. **This file is very important and private.**
5. **Final Step:** Rename the downloaded file to `gcp_credentials.json` and move it into your main project folder.

Step 5: Prepare Your Google Sheet

1. Open the provided Google Sheet template link: https://docs.google.com/spreadsheets/d/1MMiZkAm8I8jumzuYghTZFgCdqylZppEURq_2ulT8aey8/edit
2. Click **File > Make a copy** to create your own version.
3. **Share the Sheet with the Robot:**
    - Open your `gcp_credentials.json` file with a text editor. Find the line that says `"client_email"`. It will look something like `pdp-sheet-manager@...iam.gserviceaccount.com`. Copy this entire email address.
    - In your Google Sheet, click the **"Share"** button.
    - Paste the robot's email address into the sharing box, make sure it has **"Editor"** access, and click **"Send"**.

Your sheet is now ready. You can add your search queries in **Column D**, starting from **Row 4**. We have included sample CSV files to help you understand the required format.

Part C: Final Configuration & Activation

Step 6: Configure the

Open the `config.py` file in a text editor. You need to update the paths and settings to match your setup.

- `PROJECT_ROOT`: Change the path to the absolute path of your project folder.
    - Example: `r"C:\Users\YourName\Documents\PDPs_Analysis_Project"`
-
- `CHROME_PROFILE_PATH`: This is set automatically based on `PROJECT_ROOT`. You don't need to change it.
- `GCP_CREDENTIALS_PATH`: This is also set automatically. No change needed.
- **Email Notification Settings (Optional):**
    - If you want error alerts, set `ENABLE_EMAIL_NOTIFICATIONS = True`.
    - Fill in your `SENDER_EMAIL`, `SENDER_PASSWORD`, and `RECIPIENT_EMAIL`.

Step 7: Create the Master Chrome Profile

This step creates a dedicated Chrome profile that will stay logged into Google, which helps prevent CAPTCHAs.

1. Make sure your virtual environment is active (`(venv)` is visible).
2. In your command prompt, run this script:
   `python create_master_profile.py`
3. A new Chrome window will open. You have 90 seconds to:
    - Go to `google.com`.
    - **Sign in** to a Google account.
    - Accept any cookie pop-ups.
    - When you are done, **manually close the browser window**.

Step 8: Running the Automation for the First Time

Everything is now set up! To run the automation:

1. Make sure your virtual environment is active (`(venv)` is visible).

2. In your command prompt, run the main script:
   ```
   python pdp_checker.py
   ```

The script will now start. You will see its progress being printed in the command prompt window.

Congratulations! Your PDPs Analysis Automation tool is now fully set up and ready to go.

Chapter 3: Using the Automation Day-to-Day

Your Regular Workflow: Using the tool is simple once the setup is complete.

1. **Add/Update Queries:** Open your Google Sheet and add any new search queries to **Column D**.
2. **Activate Environment:** Open a command prompt, navigate to your project folder, and activate the virtual environment (`venv\Scripts\activate`).
3. **Run the Script:** Execute the command `python pdp_checker.py`.
4. **Check the Results:** Once the script finishes, open your Google Sheet. You will see a new column with today's date filled with the latest search result counts.

How to Add New Queries to Track

Simply open your Google Sheet and add new queries to the bottom of the list in **Column D**. The script is configured to process rows 4 to 32 by default, but this can be changed (see Chapter 4).

Understanding the Output in Your Sheet

- **A Number (e.g.,** This is a successful scrape. It's the number of results Google found.
- **:** This means Google explicitly returned "No results found" for the query.
- **:** The script could not find the result count. This might be due to a temporary network issue or a change in Google's page layout.
- **:** A CAPTCHA was detected, but it was not solved in time.

Handling CAPTCHAs and Login Issues

Over time, Google might log you out or present more frequent CAPTCHAs.

- **If the script pauses for a CAPTCHA:** The command prompt will alert you. Simply switch to the open Chrome window, solve the puzzle, and the script will automatically continue.
- **If you consistently get errors or CAPTCHAs:** Your Chrome profile may be stale. To fix this, run the refresh script:
    1. Activate your virtual environment.
    2. Run `python refresh_profile.py`.
    3. A new Chrome window will open. Log in to your Google account again.
    4. Close the window manually. This creates a fresh, logged-in profile.

Chapter 4: Strategies for Better Analysis

How to Use the Data Strategically: This tool is powerful, but using it strategically will give you the best results.

- **Track Competitor Brand Mentions:** Add queries like `"Competitor Name" reviews` or `site:news.google.com "Competitor Name"` to track their media presence.
- **Monitor Industry Trends:** Track broad industry terms (e.g., `"artificial intelligence in marketing"`) to see if the volume of content is growing or shrinking over time.
- **Measure Content Campaign Impact:** If you launch a big content campaign around a topic, track related queries to see if you are contributing to an increase in overall search results.
- **Identify SERP Volatility:** A sudden, massive jump or drop in results for a keyword can indicate a Google algorithm update or a major industry event.

Use Cases for Deeper Insights

1. **Competitive Analysis:** Track the number of indexed pages for a competitor's specific product line using a query like `site:competitor.com "product line name"`. A growing number indicates they are actively building out content.
2. **Digital PR Measurement:** Track the number of results for your company's name alongside a recent press release topic (`"YourCompany" "New Feature X"`). An increase after a campaign shows it's gaining traction online.
3. **Content Gap Analysis:** If a keyword has very few results (`0` or a low number), it might represent a content gap and an opportunity for you to create content and rank easily.

How to Customize the Automation: You can easily tweak the script's behavior without editing the core code.

- **Change the Number of Rows to Process:**
  - Open `pdp_checker.py` in a text editor.
  - At the top, find the line `PDP_END_ROW = 32`.
  - Change `32` to the last row number you want to process.
- **Adjust the Delays:**
  - In `pdp_checker.py`, find the `PDP_DELAY_CONFIG` section.
  - You can increase the `min` and `max` values for `"between_queries"` if you are getting blocked. For example, change it to `{"min": 15, "max": 30}` to make the script slower and safer.

Chapter 5: How the Magic Happens (A Simple Look at the Script)

What are Python and Selenium?

- **Python:** A popular programming language known for its readability and power. It's the language the entire script is written in.
- **Selenium:** A tool that allows Python to control a web browser, just like a person would. It can click buttons, type text, and read information from web pages.

The Journey of a Query: The Script's Logic

When you run `pdp_checker.py`, it follows a very logical, step-by-step process:

1. **Wake Up & Connect:** The script starts and uses your `gcp_credentials.json` file to connect to the Google Sheets API.
2. **Prepare the Battlefield:** It opens your specified Google Sheet, reads all the date headers in row 2 to find the last column, and inserts a new, empty column with today's date.
3. **Read the Mission List:** It reads all the queries from Column D into its memory.
4. **Launch the Browser:** It opens a new Chrome window using the dedicated `Chrome-Master-Profile`, which already has your login cookies.
5. **Begin the Loop (For each query):**
    - **Navigate:** It goes to `google.com`.
    - **Search:** It finds the search box, types in the query, and presses Enter.
    - **Wait & Analyze:** It waits for the page to load. It checks for any CAPTCHA warnings. If one is found, it pauses and waits for you.
    - **Click & Scrape:** It finds and clicks the "Tools" button. It then finds the element containing the result stats (e.g., "About 71,60,000 results") and extracts the number.
    - **Report Back:** It sends the extracted number back to the Google Sheet, placing it in the correct row within the new date-stamped column.
    - **Take a Break:** It waits for a random amount of time (e.g., 7-15 seconds) before starting the next query to appear more human.
6. **Mission Complete:** After the last query is processed, it closes the browser and shuts down.

Important Timings to Know

These timings are set within the `pdp_checker.py` script to ensure reliable performance.

- **Typing Delay:** A tiny, random delay between each letter typed into the search box.

- **After Page Load Delay:** Waits 3-5 seconds after a page loads before trying to scrape.
- **Between Queries Delay:** Waits 7-15 seconds after finishing one query before starting the next. This is the most important delay for avoiding blocks.

Appendix: Troubleshooting & Frequently Asked Questions (FAQ)

Q: I ran the script, but it crashed immediately with an error like
A: This usually means the required libraries are not installed correctly or your virtual environment is not active.

1. Open your command prompt and navigate to the project folder.
2. Make sure you activate the environment: `venv\Scripts\activate`.
3. Run `pip install -r requirements.txt` again to be sure.
4. Try running the script again: `python pdp_checker.py`.

Q: I'm getting an authentication or
A: This is almost always a Google Cloud or sharing issue. Check these three things:

1. Is the `gcp_credentials.json` file in your project folder and named correctly?
2. Did you enable both the **Google Drive API** and **Google Sheets API** in your GCP project?
3. Did you share your Google Sheet with the `client_email` from the JSON file, giving it **Editor** permissions?

Q: The script runs, but no data appears in my sheet.
A: Look at the output in the command prompt window. It will often tell you the problem.

- Check that the `PDP_SPREADSHEET_ID` in `pdp_checker.py` matches the ID in your sheet's URL.
- Ensure the sheet tab name is exactly `PDPs` (as defined in `PDP_SHEET_NAME`).
- Make sure your queries are in Column D.

Q: How do I schedule this script to run automatically?
A: This is an advanced step. You can use your operating system's built-in task scheduler.

- **On Windows:** Use the "Task Scheduler" application.
- **On Mac/Linux:** Use `cron`.
  You will need to create a task that runs a command pointing to your project's Python executable and the `pdp_checker.py` script. For example:
  `C:\Users\YourName\Documents\PDPs_Analysis_Project\venv\Scripts\python.exe C:\Users\YourName\Documents\PDPs_Analysis_Project\pdp_checker.py`.

Q: Can I change the follow-up days from 3 and 7 days to something else?

A: This question seems to be from the previous email outreach document. For this PDP Analysis tool, there are no follow-up days. The script runs and records data for the day it is executed.