5 CODING Exercises

Google Apps Script Exercises

Explore what you can do with Google Apps Script in Workspace.

1. Custom Email Campaign Manager

Objective: Create a script that fetches contacts from a Google Sheets document, creates personalized emails based on a template, and sends them at scheduled intervals. Include functionality to track which contacts have already been emailed and prevent duplicate emails.

Skills Practiced:

- Reading and writing data with Google Sheets API
- Sending emails using GmailApp

- Using HTML templates for email content
- Implementing triggers for automation

2. Automated Document Generator

Objective: Develop a script that generates Google Docs based on data from a Google Sheet. Each row in the sheet should correspond to a new document containing personalized information from the row. The script should also organize these documents into specified folders in Google Drive.

Skills Practiced:

- Manipulating Google Docs and Sheets
- Interacting with Google Drive to create and organize files
- Using data from Sheets to dynamically create content

3. Internal Help Desk System

Objective: Build an internal ticketing system using Google Forms for ticket submission, Google Sheets to track and manage tickets, and Gmail for notifications. Implement a system to assign, update, and mark tickets as resolved, with email notifications for ticket updates.

Skills Practiced:

- Integrating Google Forms, Sheets, and Gmail
- Implementing logic to manage ticket states
- Automating email notifications based on ticket actions

4. Resource Booking System

Objective: Create a script that interfaces with Google Calendar to allow users to book resources (like meeting rooms or equipment) through a Google Sheet. Include checks to prevent double-booking and automate confirmation emails to the user.

Skills Practiced:

- Google Calendar API for managing events
- Google Sheets as a booking interface
- Advanced data validation and error handling

5. Project Dashboard Creator

Objective: Develop a script that creates a comprehensive project management dashboard in Google Data Studio by fetching and consolidating data from multiple Google Sheets. This script should automate the data preparation process, including cleaning, merging, and summarizing data to be ready for visualization.

Skills Practiced:

- Data manipulation and preparation with Google Sheets API
- Interacting with Google Data Studio for reporting
- Automating regular data updates and refreshes

Tips for Success:

- Start by outlining the functionality and flow of your script before coding.
- Break down each exercise into smaller tasks and tackle them one at a time.
- Use the Google Apps Script documentation extensively to understand the capabilities and limitations of each service you're interacting with.
- Test your scripts regularly to catch and fix errors early in the development process.

These exercises are designed to be challenging and will help you build a strong foundation in Google Apps Script, covering a wide range of its capabilities and services. Good luck!

These will serve as a foundation, from which you can expand and customize according to your needs.

1. Custom Email Campaign Manager

Key Steps:

1. Read Contacts from Google Sheets: Use SpreadsheetApp to access your sheet and read the contacts.
2. Send Emails: Loop through each contact, generate a personalized email using a template, and send it using GmailApp.
3. Track Sent Emails: Mark contacts as emailed in your sheet to avoid duplicates.

Snippet:

```
function sendCustomEmails() {
 var sheet =
SpreadsheetApp.getActiveSpreadsheet().getSheetByName("C
ontacts");
 var startRow = 2; // Assuming first row is headers
 var numRows = sheet.getLastRow() - 1;
 var dataRange = sheet.getRange(startRow, 1, numRows,
2); // Adjust column numbers based on your data
 var data = dataRange.getValues();

 for (var i = 0; i < data.length; ++i) {
 var row = data[i];
 var emailAddress = row[0]; // First column
 var message = "Dear " + row[1] + ",\n\nYour
personalized message here."; // Customize this
 var subject = "Your Custom Email Subject";
 GmailApp.sendEmail(emailAddress, subject, message);
 sheet.getRange(startRow + i, 3).setValue("Email
Sent"); // Mark as sent in the third column
 SpreadsheetApp.flush(); // Apply changes to the
spreadsheet
 }
}
```

2. Automated Document Generator

Key Steps:

1. Fetch Data from Google Sheets: Use SpreadsheetApp to get the data for document creation.
2. Generate Documents: For each row, create a new Google Doc and populate it with the data.
3. Organize Documents: Move each document to a specific folder on Google Drive.

Snippet:

```
function generateDocuments() {
 var folder =
DriveApp.getFolderById('your-folder-id-here');
 var sheet =
SpreadsheetApp.getActiveSpreadsheet().getSheetByName("D
ata");
 var rows = sheet.getDataRange().getValues();

 rows.forEach(function(row, index) {
 if (index === 0) return; // Skip header row
 var doc = DocumentApp.create(row[0] + " Document"); //
Assume first column is the title
 var body = doc.getBody();
 body.appendParagraph("This is a document for " +
row[0]); // Customize your content
 // More content based on row data
 folder.addFile(DriveApp.getFileById(doc.getId()));
 });
}
```

3. Internal Help Desk System

Key Steps:

1. Setup Google Form for Ticket Submission: Create a form manually.
2. Use Form Responses to Populate Google Sheet: Configure form to record responses in a sheet.
3. Implement Ticket Management Logic: Use Apps Script to manage tickets, update statuses, and send emails for notifications.

Snippet:

```
function onFormSubmit(e) {
 var ticketInfo = e.values; // Contains form responses
 var emailAddress = ticketInfo[1]; // Assuming the
second question asks for the email
 var subject = "Ticket Received: " + ticketInfo[2]; //
Customize based on your form structure
 var message = "We have received your ticket. Our team
will get back to you soon.";
 GmailApp.sendEmail(emailAddress, subject, message);

 // Additional logic to manage ticket state
}
```

4. Resource Booking System

Key Steps:

1. Setup Google Sheet for Booking Requests: Structure a sheet for users to enter booking details.
2. Check Availability and Book in Google Calendar: Before booking, check for conflicts in a specific calendar.
3. Send Confirmation Email: Notify the user via email about the booking status.

Snippet:

```
function bookResource() {
 var sheet =
SpreadsheetApp.getActiveSpreadsheet().getSheetByName("B
ookings");
 var rows = sheet.getDataRange().getValues();
 rows.forEach(function(row, index) {
 if (index === 0 || row[5] === "Booked") return; //
Skip header row and already booked entries
 var calendar =
CalendarApp.getCalendarById('your-calendar-id');
 var startTime = new Date(row[2]);
 var endTime = new Date(row[3]);
 var events = calendar.getEvents(startTime, endTime);
 if (events.length === 0) { // No conflict
 calendar.createEvent(row[1], startTime, endTime);
 sheet.getRange(index + 1, 6).setValue("Booked"); //
Mark as booked
 GmailApp.sendEmail(row[4], "Booking Confirmation",
"Your booking is confirmed for " + row[1]);
 }
 });
}
```

5. Project Dashboard Creator

Key Steps:

1. Prepare Data in Google Sheets: Use multiple sheets to track different aspects of projects.
2. Automate Data Preparation: Write scripts to clean, merge, and summarize data across sheets.
3. Export Prepared Data for Google Data Studio: Ensure data is in a suitable format for visualization.

Snippet:

```
function prepareDataForDashboard() {
 var projectSheet = SpreadsheetApp.getActiveSpreadsheet().getSheetByName("Projects");
 var tasksSheet = SpreadsheetApp.getActiveSpreadsheet().getSheetByName("Tasks");
 // Example: Combine data from these sheets, summarize tasks by project, etc.

 // Data preparation logic here
 // This could involve creating a new sheet with summarized data ready for Data Studio
}
```

Next Steps:

For each exercise, expand upon the provided snippets to build the full functionality. You'll need to tailor the logic to fit your specific data structures and requirements. Also, remember to test your scripts incrementally to ensure each part works as expected before moving on to the next.