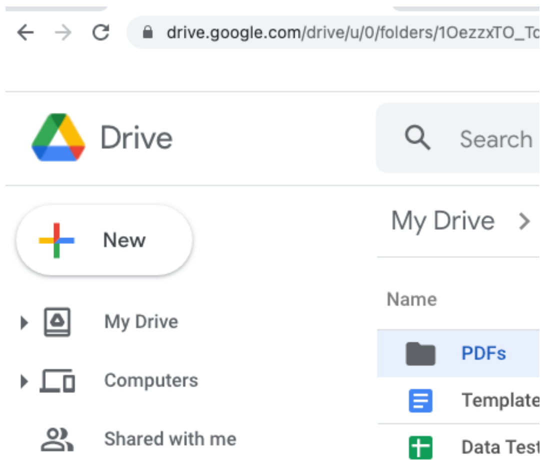
Google Apps Script PDF maker from Google Doc template. Send PDFs as Emails.

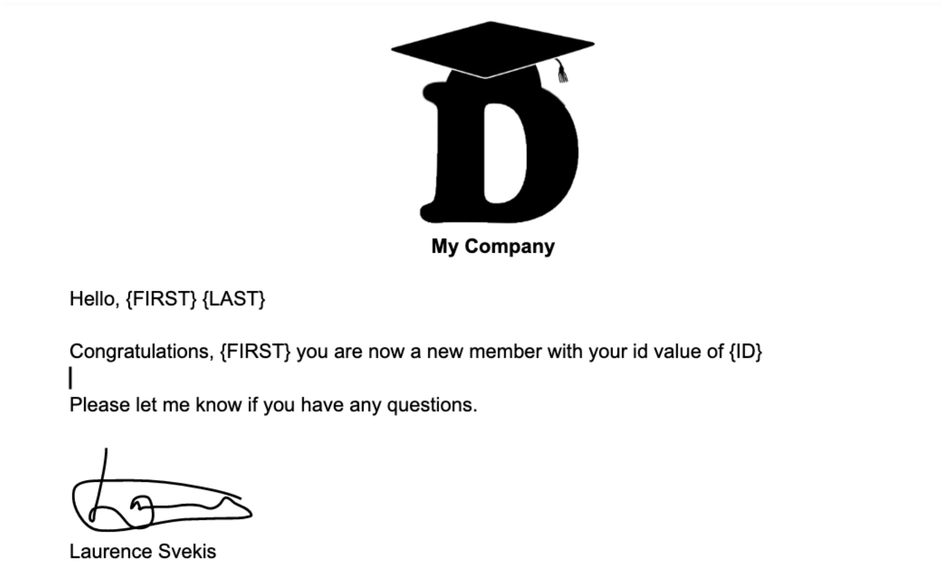
Setup files and folders

Log into your Google Account - Go to your Google Drive
Select the New button, and create a spreadsheet and a Google Doc. Give them names.



Add data and set up the template Doc

The template is going to be used as a base for forming the PDF documents. The template can also have values that will be automatically populated using Google Apps Script. I'm using the sheet heading values, as uppercase values and nested within curly brackets. This provides a way to uniquely identify the values to be replaced. Hello, {FIRST} {LAST}



My Company

Hello, {FIRST} {LAST}

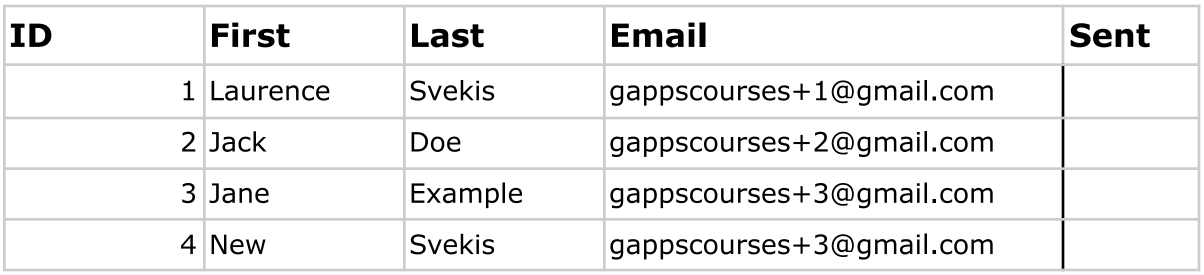
Congratulations, {FIRST} you are now a new member with your id value of {ID}

Please let me know if you have any questions.

[illegible]

Laurence Svekis

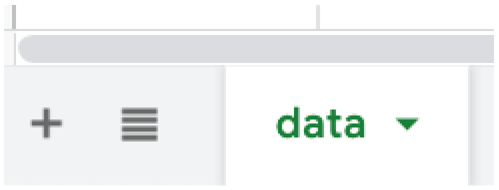
Add Data to the spreadsheet with column headings to match the fields to be updated in the template.



ID	First	Last	Email	Sent
1	Laurence	Svekis	gappscourses+1@gmail.com	
2	Jack	Doe	gappscourses+2@gmail.com	
3	Jane	Example	gappscourses+3@gmail.com	
4	New	Svekis	gappscourses+3@gmail.com	

Create the Script - Connect to Spreadsheet data

Get the unique ID for the spreadsheet, you can also select the activeSheet if you are using a bound script. Using the SpreadsheetApp Service, open the spreadsheet and select the sheet. Give the sheet a name so it can be selected.



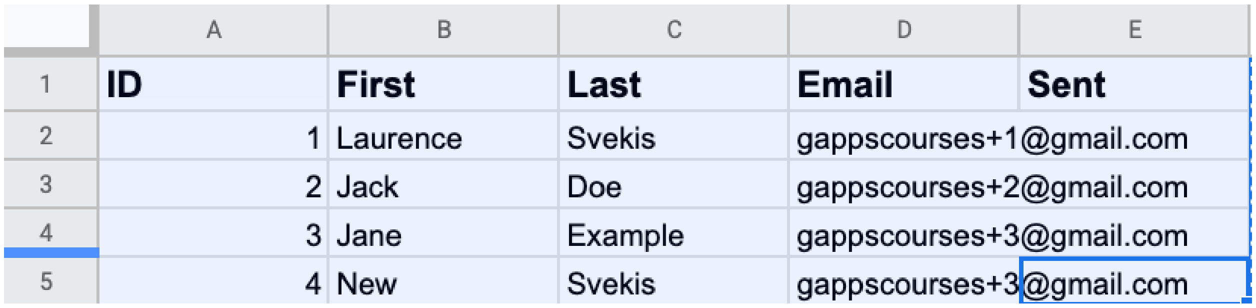
```
const SHEETID =
'1wZLJEsUkuuU0e4ACdvwtwRQTxuYOj9m9Nih8DV65fm4';



function sender() {
 const sheet =
SpreadsheetApp.openById(SHEETID).getSheetByName('d
ata');
}
```

Get the data from each row within the sheet data

The getDataRange() will create a range of rows and columns that have contents. This will select all the contents for the first 5 rows and up to column E from our example.

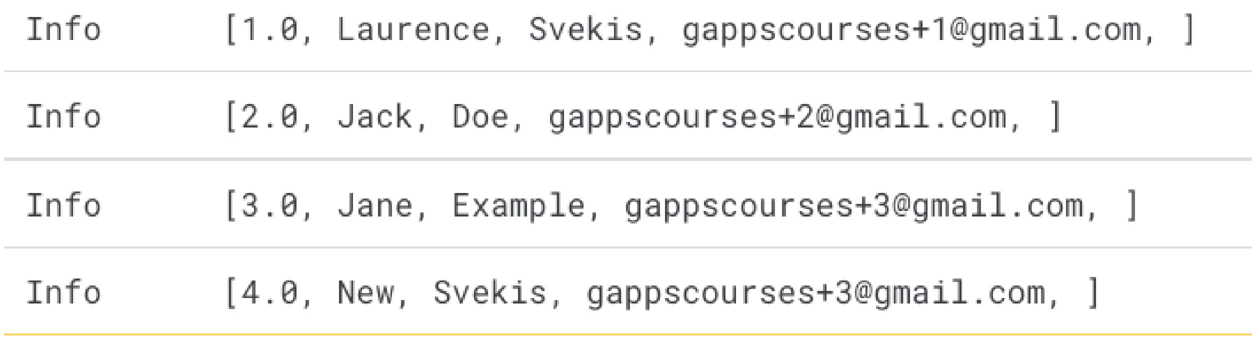


	A	B	C	D	E
1	ID	First	Last	Email	Sent
2	1	Laurence	Svekis	gappscourses+1@gmail.com	
3	2	Jack	Doe	gappscourses+2@gmail.com	
4	3	Jane	Example	gappscourses+3@gmail.com	
5	4	New	Svekis	gappscourses+3@gmail.com	

To get the data into a readable array use the getValues() method. Will return the sheet data into an array with each row as a nested array.

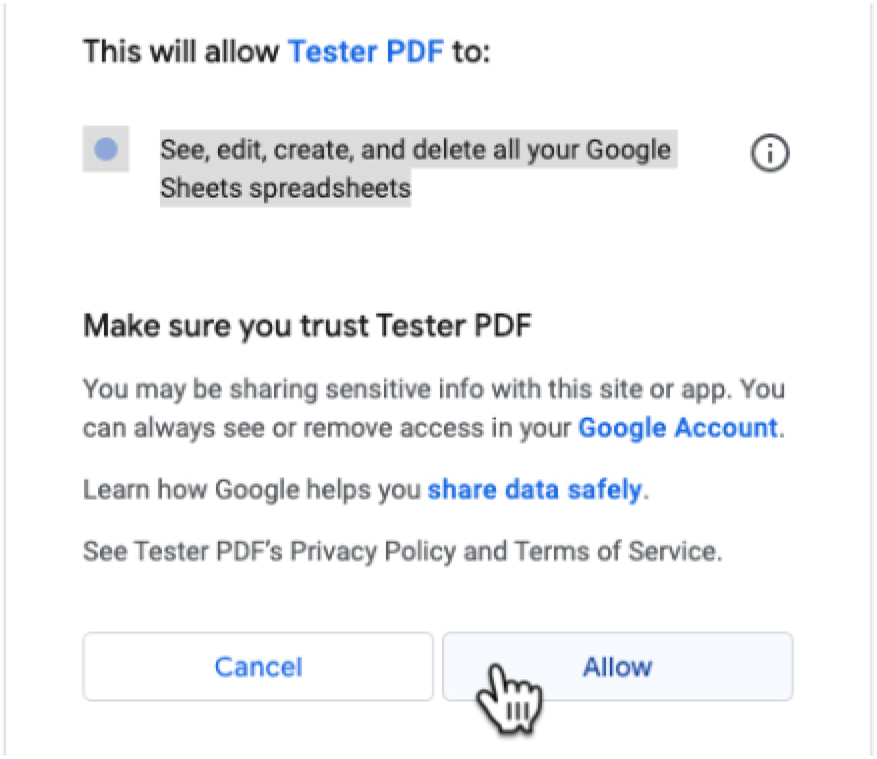
Using the slice() method remove the first row since this contains headings and not the data we want to populate with.

Loop through each row of data and output it into the logger.



```
Info        [1.0, Laurence, Svekis, gappscourses+1@gmail.com, ]
Info        [2.0, Jack, Doe, gappscourses+2@gmail.com, ]
Info        [3.0, Jane, Example, gappscourses+3@gmail.com, ]
Info        [4.0, New, Svekis, gappscourses+3@gmail.com, ]
```

You will need to accept permissions to access files and run the script.


This will allow Tester PDF to:

See, edit, create, and delete all your Google Sheets spreadsheets

Make sure you trust Tester PDF

You may be sharing sensitive info with this site or app. You can always see or remove access in your Google Account.

Learn how Google helps you share data safely.

See Tester PDF's Privacy Policy and Terms of Service.

```
function sender() {
 const sheet =
SpreadsheetApp.openById(SHEETID).getSheetByName('d
ata');
 const data = sheet.getDataRange().getValues();
 const rows = data.slice(1);
 rows.forEach((row,index)=>{
   Logger.log(row);
 })
```

Select the Google Doc Template

Using the DriveApp service, select the template Doc by its id value as a file object. To place the files into a folder on drive you will need to select the folder object also using the DriveApp.

```
const temp = DriveApp.getFileById(DOCID);
const folder = DriveApp.getFolderById(FOLDERID);
```

On each row of data, create a new document copy from the template, then select the text in the document using the getBody(). To make a replacement of the text use the replaceText() method on the body. The first argument is the value that will be replaced and the second is the replacement text. To use the second column data from the spreadsheet as the replacement text use the row data and index value of 1.

```
rows.forEach((row,index)=>{
  const file = temp.makeCopy(folder);
  const doc = DocumentApp.openById(file.getId());
  const body = doc.getBody();
  body.replaceText('{FIRST}',row[1]);
  Logger.log(row);
})
```

Populate the Doc with Sheet Data

The first row contains the heading values, to select the array use the data with index value of 0. This will return an array of the headings from the sheet. Using the forEach() method, loop through all the headings, set the headings as the value to be replaced and using the index value from the headings, update the values accordingly.

```
data[0].forEach((heading,i)=>{
  const header1 = heading.toUpperCase();
  body.replaceText(`{${header1}}`,row[i]);
})
```

Set the name of the document

The name of the document can be set using the setName() with a string value for the name. To use the sheet values as the name they can be used within the argument.

```
doc.setName(row[0]+row[1]);
```

Create and send the PDF

Get the document file as a PDF blob using   const blob = doc.getAs(MimeType.PDF);

```
const blob = doc.getAs(MimeType.PDF);
```

Close the Doc and save the changes. This will finalize the replace text to ensure its updated from the replace method.

```
doc.saveAndClose();
```

To Create a PDF version of the file you can use the createFile() method to create a PDF file from the blob object.

```
folder.createFile(blob).setName(row[0]+row[1]+'.pdf');
```

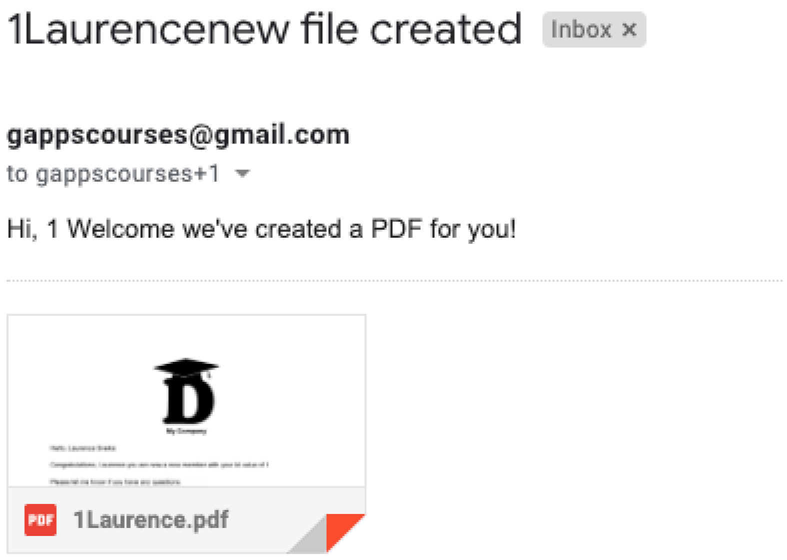
The Doc file can be moved into the trash if it's not needed within the folder.

```
file.setTrashed(true);
```

To send an email with the file as the attachment you can use the MailApp Service and sendEmail() method. Add the attachments property and include the blob as a PDF mime type within the attachments array. Create the subject and html message body to add into the sendEmail properties.

```
   const email = row[3];
   const subject = row[0]+row[1]+'new file created';
   const messageBody = `Hi, ${row[0]} Welcome we've created a PDF for you!`;
   MailApp.sendEmail({
     to:email,
     subject:subject,
     htmlBody: messageBody,
     attachments: [blob.getAs(MimeType.PDF)]
   });
```

This will send the PDF from the updated Doc file to the email address listed in the 3rd column from the spreadsheet.



1Laurencenew file created Inbox ×

gappscourses@gmail.com
to gappscourses+1

Hi, 1 Welcome we've created a PDF for you!

1Laurence.pdf

Update Row Column with Sent value

To track already sent emails, use the column with the heading sent. We can add a condition that if the cell for sent in the row has a value then we skip sending a new email. If it's blank then we can send and create the PDF. After this is done, update the column value with the current date. Select the range that should be updated and then set the value of the cell to the current date with new Date().

```
  const tempo = sheet.getRange(index+2,5,1,1);
  tempo.setValue(new Date());
```

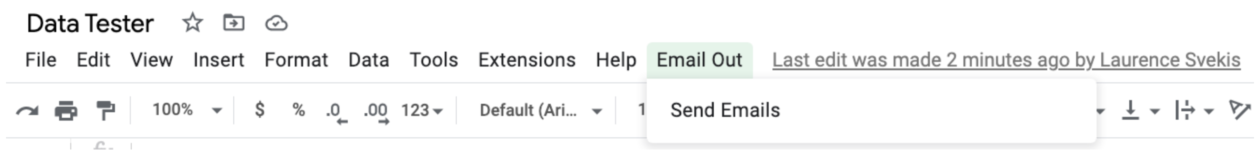
Skip over already sent emails, add a condition to the row to skip the email if the column 5 cell has a value in the current row.

```
rows.forEach((row,index)=>{
  if(row[4] ==''){
```

Add to the UI

This is only available for bound scripts and will not work on standalone scripts. Add an item that can run a Google Apps Script function.

```
function onOpen(){
 SpreadsheetApp.getUi()
 .createMenu('Email Out')
 .addItem('Send Emails','sender')
 .addToUi();
}
```


Data Tester
File Edit View Insert Format Data Tools Extensions Help Email Out
Last edit was made 2 minutes ago by Laurence Svekis
100%
Default (Ari...
Send Emails

Complete Project Script

```
const SHEETID = '1wZL****fm4';
const DOCID = '1RwP1****5T5U';
const FOLDERID = '1-****V4';


function onOpen() {
 SpreadsheetApp.getUi()
   .createMenu('Email Out')
   .addItem('Send Emails', 'sender')
   .addToUi();
}

function sender() {
 const sheet =
SpreadsheetApp.openById(SHEETID).getSheetByName('d
ata');
 const temp = DriveApp.getFileById(DOCID);
 const folder = DriveApp.getFolderById(FOLDERID);

 const data = sheet.getDataRange().getValues();
 const rows = data.slice(1);
 rows.forEach((row, index) => {
   if (row[4] == '') {
     const file = temp.makeCopy(folder);
     const doc =
DocumentApp.openById(file.getId());
```

```
    const body = doc.getBody();
    data[0].forEach((heading, i) => {
      const header1 = heading.toUpperCase();
      body.replaceText(`{${header1}}`, row[i]);
    })
    doc.setName(row[0] + row[1]);
    const blob = doc.getAs(MimeType.PDF);
    doc.saveAndClose();

    const pdf =
folder.createFile(blob).setName(row[0] + row[1] +
'.pdf');

    const email = row[3];
    const subject = row[0] + row[1] + 'new file
created';
    const messageBody = `Hi, ${row[0]} Welcome
we've created a PDF for you!`;
    MailApp.sendEmail({
      to: email,
        subject: subject,
        htmlBody: messageBody,
        attachments: [blob.getAs(MimeType.PDF)]
      });

      const tempo = sheet.getRange(index + 2, 5, 1,
1);
      tempo.setValue(new Date());
      Logger.log(row);
      file.setTrashed(true);
    }
  })
}
```