Google Apps Script coding examples

Add a new column to a Google Sheet	3
Sort the rows in a Google Sheet	5
Filter the rows in a Google Sheet	6
Create a pivot table in a Google Sheet	7
Create a new Google Slides presentation	9
Send an email notification to the user	12
Create a Google Drive file	14
Get the current time	15
Create a new Google Form	16
Send an email notification	18
Create a Google Sheet	19
Get the current time	21
Get the user's location	22
Add a new column to a Google Sheet	23
Sort the rows in a Google Sheet	25
Filter the rows in a Google Sheet	26
Create a pivot table in a Google Sheet	28
Sending Emails with Attachments	30
Merge Multiple sheets into one	31
Create a new Google Slides presentation	33
Send an email notification to the user	36
Create a Google Drive file	37
Get the current time	39
Create Google Forms from Sheets	40
Track Changes to Google Sheet	42
Generate Slides from Sheets data	44
Create a new Google Form	46
Send an email notification to the user	47
Create a Google Sheet	49

Get the current time 50
Get the user's location 51
Update Google Sheets from Forms 52
Generate Docs from Sheets Data 54
Send email notifications from Sheets 58
Generate Slides from Sheets 59
Create a Google Form 61
Send an email notification 62
Create a Google Sheet 64
Get the current time 65
This code will get the current time. 65
Get the user's location 66
This code will get the user's location. 66
Learn more about Google Apps Script 67
How to create a new spreadsheet? 68
How to add a row to a spreadsheet? 69
How to get the value of a cell in a spreadsheet? 70
How to set the value of a cell in a spreadsheet? 71
How to send an email? 71
How to create a new form? 72
How to send a notification? 73
How to create a new calendar event? 74
How to get the current date and time? 75
How to log an error? 75
How to get the value of a property? 76
How to set the value of a property? 76
How to loop through an array? 77
How to call a function asynchronously? 77
How to get the current user? 78
How to create a new folder? 78
How to upload a file? 79
How to download a file? 79
How to create a new link? 80
How to get the size of a file? 80

How to get the type of a file? 81
How to get the last modified date of a file? 81
How to make a copy of a file? 82
How to move a file to a new folder? 83
How to delete a file? 83
How to create a new slide in a presentation? 84
How to add a text box to a slide? 84
How to add an image to a slide? 85
How to add a shape to a slide? 85
How to change the slide layout? 85
How to add a transition to a slide? 86
How to preview a presentation? 87
How to publish a presentation? 87
How to protect a presentation? 88
How to create a new document? 88
How to add a heading to a document? 89
How to add a list to a document? 89
How to add an image to a document? 90
How to add a table to a document? 90
How to add a link to a document? 90
How to add a footnote to a document? 91
How to add a comment to a document? 91
How to protect a document? 92
Send email 10 most recent files exercise 92
Populate from Google Form Exercise 94
Create slides from Google Sheet exercise 97
Create calendar event email user exercise 100
Create File upload to Drive location exercise 103

Add a new column to a Google Sheet

This code will add a new column to a Google Sheet.

```
function addColumn() {
 // Get the sheet object.
 var sheet = SpreadsheetApp.getActiveSpreadsheet();
 // Get the current sheet name.
 var sheetName = sheet.getName();
 // Get the column index of the last column.
 var lastColumnIndex = sheet.getMaxColumns();
 // Add a new column at the specified index.
 sheet.insertColumn(lastColumnIndex + 1);
 // Set the column name.
 sheet.getRange(lastColumnIndex + 1, 1).setValue('New Column');
}
```

This code works by first getting the sheet object. Then, it gets the current sheet name and the column index of the last column. Finally, it adds a new column at the specified index and sets the column name.

This code could be used to add a new column to a Google Sheet for any purpose. For example, you could use this code to add a new column to a Google Sheet that stores the names and contact information of your customers.

Sort the rows in a Google Sheet

This code will sort the rows in a Google Sheet.

```
function sortRows() {

 // Get the sheet object.

 var sheet = SpreadsheetApp.getActiveSpreadsheet();

 // Get the column index of the column to sort by.

 var columnIndex = 2;

 // Sort the rows in ascending order.

 sheet.sortRange(1, 1, sheet.getMaxRows(), columnIndex, true);

}
```

This code works by first getting the sheet object. Then, it gets the column index of the column to sort by and sorts the rows in ascending order.

This code could be used to sort the rows in a Google Sheet for any purpose. For example, you could use this code to sort the rows in a Google Sheet that stores the names and contact information of your customers by their last name.

Filter the rows in a Google Sheet

This code will filter the rows in a Google Sheet.

```
function filterRows() {

 // Get the sheet object.

 var sheet = SpreadsheetApp.getActiveSpreadsheet();

 // Get the column index of the column to filter by.

 var columnIndex = 2;

 // Get the value to filter by.

 var value = 'John Doe';

 // Filter the rows that match the specified value.

 sheet.filterRange(1, 1, sheet.getMaxRows(), columnIndex, value);

}
```

This code works by first getting the sheet object. Then, it gets the column index of the column to filter by and the value to filter by and filters the rows that match the specified value.

This code could be used to filter the rows in a Google Sheet for any purpose. For example, you could use this code to filter the rows in a Google Sheet that stores the names and contact information of your customers by their last name.

Create a pivot table in a Google Sheet

This code will create a pivot table in a Google Sheet.

```
function createPivotTable() {

 // Get the sheet object.

 var sheet = SpreadsheetApp.getActiveSpreadsheet();

 // Get the range of data to include in the pivot table.

 var range = sheet.getRange('A1:C10');

 // Create a new pivot table.

 var pivotTable = sheet.newPivotTable(range);
```

```
// Set the column headers for the pivot table.

pivotTable.addColumnHeader('Name');

pivotTable.addColumnHeader('Email');

pivotTable.addColumnHeader('Phone');

// Set the row headers for the pivot table.

pivotTable.addRowHeader('Country');

// Set the values for the pivot table.

pivotTable.setValue('Count', 'Count');

// Display the pivot table.

pivotTable.show();

}
```

This code works by first getting the sheet object. Then, it gets the range of data to include in the pivot table and creates a new pivot table. Finally, it sets the column headers, the row headers, the values, and displays the pivot table

Create a new Google Slides presentation

This code will create a new Google Slides presentation.

```
function createPresentation() {

 // Create a new Google Slides presentation.

 var presentation = SlidesApp.create('My Presentation');

 // Add a new slide to the presentation.

 var slide = presentation.insertSlide();

 // Add a title slide to the presentation.

 slide.setTitle('My Presentation');

 // Add a subtitle slide to the presentation.

 slide = presentation.insertSlide();

 slide.setSubtitle('This is a subtitle slide.');

 // Add a new slide for each row of data in the sheet.

 for (var i = 0; i < data.length; i++) {

 slide = presentation.insertSlide();
```

```
// Add a title to the slide.

slide.setTitle(data[i][0]);

// Add a paragraph to the slide for each column of data in the row.

for (var j = 1; j < data[i].length; j++) {

slide.appendParagraph(data[i][j]);

}

}

// Save the presentation.

presentation.save();

}
```

This code works by first creating a new Google Slides presentation object. Then, it adds a new slide to the presentation, adds a title slide to the presentation, adds a subtitle slide to the presentation, adds a new slide for each row of data in the sheet, adds a title to each slide, and adds a paragraph to each slide for each column of data in the row. Finally, it saves the presentation.

This code could be used to create a presentation from data that is stored in a Google Sheet. For example, you could use this code to create a presentation from data that stores the names and contact information of your customers.

Send an email notification to the user

This code will send an email notification to the user.

```
function sendEmail() {

 // Get the user's email address.

 var userEmail = Utilities.getUserEmail();

 // Create an email message.

 var message = GmailApp.createEmail();

 message.setSubject('Your email notification');

 message.setBody('This is the body of your email notification.');

 // Send the email notification.

 message.sendTo(userEmail);

}
```

This code works by first getting the user's email address. Then, it creates an email message and sets the subject line and body of the message. Finally, it sends the email notification.

This code could be used to send email notifications to users about important events or updates. For example, you could use this code to send email notifications to users when their account is created or when their password is changed.

Create a Google Drive file

This code will create a new Google Drive file.

```
function createFile() {

 // Get the file name.

 var fileName = 'My File.txt';

 // Get the content of the file.

 var content = 'This is the content of my file.';

 // Create a new Google Drive file.

 var file = DriveApp.createFile(fileName);
```

```
 // Set the content of the file.

 file.setContent(content);

 // Save the file.

 file.save();

}
```

This code works by first getting the file name and then getting the content of the file. Once it has these two pieces of information, it creates a new Google Drive file, sets the content of the file, and then saves the file.

This code could be used to create a file that users can download. For example, you could use this code to create a file that users can download that contains a list of your products or services.

Get the current time

This code will get the current time.

```
function getCurrentTime() {

 // Get the current time.

 var currentTime = Utilities.now();
```

```
 // Display the current time.

 Logger.log('The current time is: ' + currentTime);

}
```

This code works by first getting the current time using the Utilities.now() method. Then, it displays the current time using

Create a new Google Form

This code will create a new Google Form.

```
function createForm() {

 // Create a new Google Form.

 var form = FormApp.create('My Form');

 // Add a question to the form.

 var question = form.addQuestion('What is your name?');

 question.setRequired(true);

 // Add another question to the form.

 var question = form.addQuestion('What is your email address?');
```

```
 question.setRequired(true);

 // Save the form.

 form.save();

}
```

This code works by first creating a new Google Form object. Then, it adds two questions to the form, one for the user's name and one for the user's email address. Finally, it saves the form.

This code could be used to create a form that users can fill out to provide their contact information. For example, you could use this code to create a form that users can fill out to sign up for your newsletter or to request a consultation.

Send an email notification

This code will send an email notification to the user.

```
function sendEmail() {

 // Get the user's email address.

 var userEmail = Utilities.getUserEmail();
```

```
 // Create an email message.

 var message = GmailApp.createEmail();

 message.setSubject('Your email notification');

 message.setBody('This is the body of your email notification.');

 // Send the email notification.

 message.sendTo(userEmail);

}
```

This code works by first getting the user's email address. Then, it creates an email message and sets the subject line and body of the message. Finally, it sends the email notification.

This code could be used to send email notifications to users about important events or updates. For example, you could use this code to send email notifications to users when their account is created or when their password is changed.

Create a Google Sheet

This code will create a new Google Sheet.

```
function createSheet() {

 // Create a new Google Sheet.

 var sheet = SpreadsheetApp.create('My Sheet');

 // Add a row to the sheet.

 var row = sheet.appendRow([1, 2, 3]);

 // Add another row to the sheet.

 var row = sheet.appendRow([4, 5, 6]);

 // Save the sheet.

 sheet.save();

}
```

This code works by first creating a new Google Sheet object. Then, it adds two rows to the sheet, one with the values 1, 2, and 3 and one with the values 4, 5, and 6. Finally, it saves the sheet.

This code could be used to create a Google Sheet that stores data. For example, you could use this code to create a Google Sheet that stores the names and contact information of your customers.

Get the current time

This code will get the current time.

```
function getCurrentTime() {

 // Get the current time.

 var currentTime = Utilities.now();

 // Display the current time.

 Logger.log('The current time is: ' + currentTime);

}
```

This code works by first getting the current time using the Utilities.now() method. Then, it displays the current time using the Logger.log() method.

This code could be used to get the current time for any purpose. For example, you could use this code to get the current time to display on a website or to log the current time in a database.

Get the user's location

This code will get the user's location.

```
function getUserLocation() {

 // Get the user's location.

 var userLocation = Geolocation.getGeolocation();

 // Display the user's location.

 Logger.log('The user's location is: ' + userLocation);

}
```

This code works by first getting the user's location using the Geolocation.getGeolocation() method. Then, it displays the user's location using the Logger.log() method.

Add a new column to a Google Sheet

This code will add a new column to a Google Sheet.

```
function addColumn() {

 // Get the sheet object.

 var sheet = SpreadsheetApp.getActiveSpreadsheet();

 // Get the current sheet name.

 var sheetName = sheet.getName();
```

```
 // Get the column index of the last column.

 var lastColumnIndex = sheet.getMaxColumns();

 // Add a new column at the specified index.

 sheet.insertColumn(lastColumnIndex + 1);

 // Set the column name.

 sheet.getRange(lastColumnIndex + 1, 1).setValue('New Column');

}
```

This code works by first getting the sheet object. Then, it gets the current sheet name and the column index of the last column. Finally, it adds a new column at the specified index and sets the column name.

This code could be used to add a new column to a Google Sheet for any purpose. For example, you could use this code to add a new column to a Google Sheet that stores the names and contact information of your customers.

Sort the rows in a Google Sheet

This code will sort the rows in a Google Sheet.

```
function sortRows() {

 // Get the sheet object.

 var sheet = SpreadsheetApp.getActiveSpreadsheet();

 // Get the column index of the column to sort by.

 var columnIndex = 2;

 // Sort the rows in ascending order.

 sheet.sortRange(1, 1, sheet.getMaxRows(), columnIndex, true);

}
```

This code works by first getting the sheet object. Then, it gets the column index of the column to sort by and sorts the rows in ascending order.

This code could be used to sort the rows in a Google Sheet for any purpose. For example, you could use this code to sort the rows in a Google Sheet that stores the names and contact information of your customers by their last name.

Filter the rows in a Google Sheet

This code will filter the rows in a Google Sheet.

```
function filterRows() {

 // Get the sheet object.

 var sheet = SpreadsheetApp.getActiveSpreadsheet();

 // Get the column index of the column to filter by.

 var columnIndex = 2;

 // Get the value to filter by.

 var value = 'John Doe';

 // Filter the rows that match the specified value.

 sheet.filterRange(1, 1, sheet.getMaxRows(), columnIndex, value);

}
```

This code works by first getting the sheet object. Then, it gets the column index of the column to filter by and the value to filter by and filters the rows that match the specified value.

This code could be used to filter the rows in a Google Sheet for any purpose. For example, you could use this code to filter the rows in a Google Sheet that stores the names and contact information of your customers by their last name.

Create a pivot table in a Google Sheet

This code will create a pivot table in a Google Sheet.

```
function createPivotTable() {

 // Get the sheet object.

 var sheet = SpreadsheetApp.getActiveSpreadsheet();

 // Get the range of data to include in the pivot table.

 var range = sheet.getRange('A1:C10');

 // Create a new pivot table.

 var pivotTable = sheet.newPivotTable(range);

 // Set the column headers for the pivot table.

 pivotTable.addColumnHeader('Name');

 pivotTable.addColumnHeader('Email');
```

```
 pivotTable.addColumnHeader('Phone');

 // Set the row headers for the pivot table.

 pivotTable.addRowHeader('Country');

 // Set the values for the pivot table.

 pivotTable.setValue('Count', 'Count');

 // Display the pivot table.

 pivotTable.show();

}
```

This code works by first getting the sheet object. Then, it gets the range of data to include in the pivot table and creates a new pivot table. Finally, it sets the column headers, the row headers, the values, and displays the pivot table.

Sending Emails with Attachments

Automatically Send Emails with Attachments: This code automatically sends emails with attachments to a list of recipients. It first retrieves the files to be attached from a specified folder in Google Drive. It then loops through the list of recipients and sends them an email with the attachment(s).

```
function sendEmailWithAttachments() {

 var folder = DriveApp.getFolderById("folder_id"); // specify folder ID

 var files = folder.getFiles();

  while (files.hasNext()) {

 var file = files.next();

 var blob = file.getBlob();

  var recipients = ['email1@example.com', 'email2@example.com']; //
specify recipient email addresses

 var subject = 'Email Subject';

 var body = 'Email Body';

  for (var i = 0; i < recipients.length; i++) {

 GmailApp.sendEmail(recipients[i], subject, body, {attachments: [blob]});

 }

 }

}
```

The DriveApp service is used to access the specified folder and retrieve the files within it. The while loop iterates through each file and retrieves its Blob object. The for loop then sends an email to each recipient, attaching the Blob object as an attachment.

Merge Multiple sheets into one

Merge Multiple Sheets into One: This code merges multiple sheets into one sheet. It first retrieves all the sheets in the specified spreadsheet and loops through them. For each sheet, it copies its data and pastes it into a new sheet.

```
function mergeSheets() {

 var ss = SpreadsheetApp.openById('spreadsheet_id'); // specify spreadsheet ID

 var sheets = ss.getSheets();

 var masterSheet = ss.insertSheet('Master Sheet');

  for (var i = 0; i < sheets.length; i++) {

 var sheet = sheets[i];

 var range = sheet.getDataRange();

 var values = range.getValues();
```

```
 if (i == 0) {

 masterSheet.getRange(range.getA1Notation()).setValues(values);

 } else {

 var lastRow = masterSheet.getLastRow();

 masterSheet.getRange(lastRow + 1, 1, values.length,
values[0].length).setValues(values);

 }

 }

}
```

The SpreadsheetApp service is used to access the specified spreadsheet and retrieve all its sheets. The for loop iterates through each sheet and retrieves its data. If it is the first sheet, its data is directly copied to the new sheet. If it is not the first sheet, its data is appended to the last row of the new sheet.

Create a new Google Slides presentation

This code will create a new Google Slides presentation.

```
function createPresentation() {
```

```
// Create a new Google Slides presentation.

var presentation = SlidesApp.create('My Presentation');



// Add a new slide to the presentation.

var slide = presentation.insertSlide();



// Add a title slide to the presentation.

slide.setTitle('My Presentation');



// Add a subtitle slide to the presentation.

slide = presentation.insertSlide();

slide.setSubtitle('This is a subtitle slide.');



// Add a new slide for each row of data in the sheet.

for (var i = 0; i < data.length; i++) {
```

```
slide = presentation.insertSlide();



// Add a title to the slide.

slide.setTitle(data[i][0]);



// Add a paragraph to the slide for each column of data in the row.

for (var j = 1; j < data[i].length; j++) {

slide.appendParagraph(data[i][j]);

}

}



// Save the presentation.

presentation.save();

}
```

This code works by first creating a new Google Slides presentation object. Then, it adds a new slide to the presentation, adds a title slide to the

presentation, adds a subtitle slide to the presentation, adds a new slide for each row of data in the sheet, adds a title to each slide, and adds a paragraph to each slide for each column of data in the row. Finally, it saves the presentation.

This code could be used to create a presentation from data that is stored in a Google Sheet. For example, you could use this code to create a presentation from data that stores the names and contact information of your customers.

Send an email notification to the user

This code will send an email notification to the user.

```
function sendEmail() {

 // Get the user's email address.

 var userEmail = Utilities.getUserEmail();
```

```
 // Create an email message.

 var message = GmailApp.createEmail();

 message.setSubject('Your email notification');

 message.setBody('This is the body of your email notification.');



 // Send the email notification.

 message.sendTo(userEmail);

}
```

This code works by first getting the user's email address. Then, it creates an email message and sets the subject line and body of the message. Finally, it sends the email notification.

This code could be used to send email notifications to users about important events or updates. For example, you could use this code to send email notifications to users when their account is created or when their password is changed.

Create a Google Drive file

This code will create a new Google Drive file.

```
function createFile() {
 // Get the file name.
 var fileName = 'My File.txt';

 // Get the content of the file.
 var content = 'This is the content of my file.';

 // Create a new Google Drive file.
 var file = DriveApp.createFile(fileName);

 // Set the content of the file.
```

```
 file.setContent(content);



 // Save the file.

 file.save();

}
```

This code works by first getting the file name and then getting the content of the file. Once it has these two pieces of information, it creates a new Google Drive file, sets the content of the file, and then saves the file.

This code could be used to create a file that users can download. For example, you could use this code to create a file that users can download that contains a list of your products or services.

Get the current time

This code will get the current time.

```
function getCurrentTime() {

 // Get the current time.

 var currentTime = Utilities.now();



 // Display the current time.

 Logger.log('The current time is: ' + currentTime);

}
```

This code works by first getting the current time using the Utilities.now() method. Then, it displays the current time using the `

Create Google Forms from Sheets

Automatically Create Google Forms from Spreadsheet Data: This code automatically creates Google Forms from data in a specified spreadsheet. It first retrieves the data from the spreadsheet and loops through it. For each row of data, it creates a new form with the specified title and questions.

```
function createFormsFromSpreadsheet() {
```

```
 var ss = SpreadsheetApp.openById('spreadsheet_id'); // specify
spreadsheet ID

 var sheet = ss.getSheetByName('Sheet1'); // specify sheet name

 var data = sheet.getDataRange().getValues();



 for (var i = 1; i < data.length; i++) {

 var row = data[i];

 var form = FormApp.create(row[0]); // specify form title

 form.addTextItem().setTitle(row[1]); // specify first question

 form.addMultipleChoiceItem().setTitle(row[2]).setChoices([row[3], row[4],
row[5]]); // specify second question and choices

 }

}
```

The SpreadsheetApp service is used to access the specified spreadsheet and retrieve the data from the specified sheet. The for loop iterates through each row of data (starting from the second row since the first row is assumed to contain column headers). For each row, a new form is created with the specified title. Two questions are added to the form: a text question with the

title specified in the second column of the row, and a multiple choice question with the title specified in the third column of the row and the choices specified in columns 4-6.

Track Changes to Google Sheet

Track Changes to Google Sheet Data: This code tracks changes to data in a specified sheet and sends an email notification when a change occurs. It first creates a copy of the specified sheet and sets up a trigger to run the trackChanges function whenever the sheet is edited. When the function is run, it retrieves the previous and current values of the edited cell(s) and sends an email with the old and new values.

```
function trackChanges() {

 var ss = SpreadsheetApp.getActiveSpreadsheet();

 var sheet = ss.getSheetByName('Sheet1'); // specify sheet name

 var range = sheet.getActiveRange();

 var oldValue = range.getDisplayValue();
```

```
 var newValue = e.value;


 if (oldValue != newValue) {
 var recipient = 'email@example.com'; // specify recipient email address
 var subject = 'Change to Sheet1';
 var body = 'Old Value: ' + oldValue + '\nNew Value: ' + newValue;


 MailApp.sendEmail(recipient, subject, body);
 }
}


function createTrigger() {
 var ss = SpreadsheetApp.getActiveSpreadsheet();
 var sheet = ss.getSheetByName('Sheet1'); // specify sheet name
 var copy = sheet.copyTo(ss).setName('Sheet1 Copy');
```

```
 ScriptApp.newTrigger('trackChanges').forSpreadsheet(ss).onEdit().create();

}
```

The SpreadsheetApp service is used to access the specified spreadsheet and sheet. The trackChanges function is set up to run whenever the sheet is edited. It retrieves the previous and current values of the edited cell(s) using the getActiveRange() method and compares them. If they are different, an email is sent with the old and new values. The createTrigger function creates a copy of the sheet and sets up a trigger to run the trackChanges function whenever the sheet is edited.

Generate Slides from Sheets data

Automatically Generate Google Slides from Google Sheets Data: This code automatically generates Google Slides from data in a specified sheet. It first retrieves the data from the sheet and loops through it. For each row of data, a new slide is created with the specified title and content.

```
function generateSlidesFromSheet() {

 var ss = SpreadsheetApp.openById('spreadsheet_id'); // specify
spreadsheet ID
```

```
var sheet = ss.getSheetByName('Sheet1'); // specify sheet name

var data = sheet.getDataRange().getValues();

var templateId = 'template_id'; // specify slide template ID

var template = SlidesApp.openById(templateId);

var destinationId = 'destination_id'; // specify destination folder ID



for (var i = 1; i < data.length; i++) {

var row = data[i];

var title = row[0];

var content = row[1];

var slide = template.duplicate().setName(title);

var shapes = slide.getShapes();

var text = shapes[0].getText();

text.setText(content);

var slideBlob = slide.getBlob();
```

```
 var folder = DriveApp.getFolderById(destinationId);

 folder.createFile(slideBlob);

 }

}
```

Create a new Google Form

This code will create a new Google Form.

```
function createForm() {

 // Create a new Google Form.

 var form = FormApp.create('My Form');



 // Add a question to the form.

 var question = form.addQuestion('What is your name?');

 question.setRequired(true);
```

```
 // Add another question to the form.

 var question = form.addQuestion('What is your email address?');

 question.setRequired(true);



 // Save the form.

 form.save();

}
```

This code works by first creating a new Google Form object. Then, it adds two questions to the form, one for the user's name and one for the user's email address. Finally, it saves the form.

This code could be used to create a form that users can fill out to provide their contact information. For example, you could use this code to create a form that users can fill out to sign up for your newsletter or to request a consultation.

Send an email notification to the user

This code will send an email notification to the user.

```
function sendEmail() {
 // Get the user's email address.
 var userEmail = Utilities.getUserEmail();

 // Create an email message.
 var message = GmailApp.createEmail();
 message.setSubject('Your email notification');
 message.setBody('This is the body of your email notification.');

 // Send the email notification.
```

```
 message.sendTo(userEmail);

}
```

This code works by first getting the user's email address. Then, it creates an email message and sets the subject line and body of the message. Finally, it sends the email notification.

This code could be used to send email notifications to users about important events or updates. For example, you could use this code to send email notifications to users when their account is created or when their password is changed.

Create a Google Sheet

This code will create a new Google Sheet.

```
function createSheet() {

 // Create a new Google Sheet.
```

```
var sheet = SpreadsheetApp.create('My Sheet');



// Add a row to the sheet.

var row = sheet.appendRow([1, 2, 3]);



// Add another row to the sheet.

var row = sheet.appendRow([4, 5, 6]);



// Save the sheet.

sheet.save();

}
```

This code works by first creating a new Google Sheet object. Then, it adds two rows to the sheet, one with the values 1, 2, and 3 and one with the values 4, 5, and 6. Finally, it saves the sheet.

This code could be used to create a Google Sheet that stores data. For example, you could use this code to create a Google Sheet that stores the names and contact information of your customers.

Get the current time

This code will get the current time.

```
function getCurrentTime() {
 // Get the current time.
 var currentTime = Utilities.now();

 // Display the current time.
 Logger.log('The current time is: ' + currentTime);
}
```

This code works by first getting the current time using the Utilities.now() method. Then, it displays the current time using the Logger.log() method.

This code could be used to get the current time for any purpose. For example, you could use this code to get the current time to display on a website or to log the current time in a database.

Get the user's location

This code will get the user's location.

```
function getUserLocation() {

 // Get the user's location.

 var userLocation = Geolocation.getGeolocation();



 // Display the user's location.

 Logger.log('The user's location is: ' + userLocation);
```

```
}
```

This code works by first getting the user's location using the Geolocation.getGeolocation() method. Then, it displays the user's location using the Logger.log() method.

Update Google Sheets from Forms

Automatically Update Google Sheets from Google Forms Responses: This code automatically updates a Google Sheet with responses from a specified Google Form. It first sets up a trigger to run the updateSheet function whenever a new response is submitted to the form. When the function is run, it retrieves the response data and adds it to the specified sheet.

```
function updateSheet(e) {

 var ss = SpreadsheetApp.openById('spreadsheet_id'); // specify spreadsheet ID

 var sheet = ss.getSheetByName('Sheet1'); // specify sheet name

 var data = e.response.getItemResponses();

 var rowData = [];
```

```
 for (var i = 0; i < data.length; i++) {

 rowData.push(data[i].getResponse());

 }



 sheet.appendRow(rowData);

}



function createTrigger() {

 var form = FormApp.openById('form_id'); // specify form ID


ScriptApp.newTrigger('updateSheet').forForm(form).onFormSubmit().create(
);

}
```

The FormApp service is used to access the specified form, and the SpreadsheetApp service is used to access the specified spreadsheet and sheet. The updateSheet function is set up to run whenever a new response is submitted to the form. It retrieves the response data using the

getItemResponses() method and adds it to the specified sheet using the appendRow() method. The createTrigger function sets up a trigger to run the updateSheet function whenever a new response is submitted to the form.

Generate Docs from Sheets Data

Automatically Generate Google Docs from Google Sheets Data: This code automatically generates Google Docs from data in a specified sheet. It first retrieves the data from the sheet and loops through it. For each row of data, a new document is created with the specified title and content.

```
function generateDocsFromSheet() {

 var ss = SpreadsheetApp.openById('spreadsheet_id'); // specify spreadsheet ID

 var sheet = ss.getSheetByName('Sheet1'); // specify sheet name

 var data = sheet.getDataRange().getValues();

 var templateId = 'template_id'; // specify document template ID

 var template = DocumentApp.openById(templateId);

 var destinationId = 'destination_id'; // specify destination folder ID
```

```
for (var i = 1; i < data.length; i++) {

var row = data[i];

var title = row[0];

var content = row[1];

var doc = template.makeCopy(title);

var body = doc.getBody();

body.replaceText('{{content}}', content);

var docBlob = doc.getAs('application/pdf');

var folder = DriveApp.getFolderById(destinationId);

folder.createFile(docBlob);

}

}
```

The SpreadsheetApp service is used to access the specified spreadsheet and sheet. The DocumentApp service is used to access the specified document template and create new documents. The generateDocsFromSheet function retrieves the data from the sheet and loops through it. For each row of data,

a new document is created with the specified title and content. The makeCopy() method is used to create a copy of the template, and the replaceText() method is used to replace the placeholder text with the actual content. The getAs() method is used to convert the document to PDF format, and the createFile() method is used to save the PDF file to the specified folder.

Generate Calendar Events from Sheets

Automatically Generate Google Calendar Events from Google Sheets Data: This code automatically generates Google Calendar events from data in a specified sheet. It first retrieves the data from the sheet and

```
function generateEventsFromSheet() {

 var ss = SpreadsheetApp.openById('spreadsheet_id'); // specify spreadsheet ID

 var sheet = ss.getSheetByName('Sheet1'); // specify sheet name

 var data = sheet.getDataRange().getValues();

 var calendarId = 'calendar_id'; // specify calendar ID

 var calendar = CalendarApp.getCalendarById(calendarId);
```

```
 for (var i = 1; i < data.length; i++) {

 var row = data[i];

 var title = row[0];

 var startTime = new Date(row[1]);

 var endTime = new Date(row[2]);

 var description = row[3];

 calendar.createEvent(title, startTime, endTime, {description: description});

 }

}
```

The SpreadsheetApp service is used to access the specified spreadsheet and sheet. The CalendarApp service is used to access the specified calendar and create new events. The generateEventsFromSheet function retrieves the data from the sheet and loops through it. For each row of data, a new event is created with the specified title, start time, end time, and description.

Send email notifications from Sheets

Automatically Send Email Notifications from Google Sheets Data: This code automatically sends email notifications to specified recipients from data in a specified sheet. It first retrieves the data from the sheet and loops through

it. For each row of data, an email is sent to the specified recipients with the specified subject and message.

```
function sendEmailsFromSheet() {

 var ss = SpreadsheetApp.openById('spreadsheet_id'); // specify spreadsheet ID

 var sheet = ss.getSheetByName('Sheet1'); // specify sheet name

 var data = sheet.getDataRange().getValues();



 for (var i = 1; i < data.length; i++) {

 var row = data[i];

 var recipients = row[0];

 var subject = row[1];

 var message = row[2];

 MailApp.sendEmail(recipients, subject, message);

 }

}
```

The SpreadsheetApp service is used to access the specified spreadsheet and sheet. The MailApp service is used to send emails. The sendEmailsFromSheet function retrieves the data from the sheet and loops through it. For each row of data, an email is sent to the specified recipients with the specified subject and message.

Generate Slides from Sheets

Automatically Generate Google Slides from Google Sheets Data: This code automatically generates Google Slides from data in a specified sheet. It first retrieves the data from the sheet and loops through it. For each row of data, a new slide is created with the specified title and content.

```
function generateSlidesFromSheet() {

 var ss = SpreadsheetApp.openById('spreadsheet_id'); // specify
spreadsheet ID

 var sheet = ss.getSheetByName('Sheet1'); // specify sheet name

 var data = sheet.getDataRange().getValues();

 var templateId = 'template_id'; // specify slide template ID

 var template = SlidesApp.openById(templateId);

 var destinationId = 'destination_id'; // specify destination folder ID
```

```
 for (var i = 1; i < data.length; i++) {

 var row = data[i];

 var title = row[0];

 var content = row[1];

 var slide = template.duplicate();

 slide.rename(title);

 var slideBlob = slide.getBlob();

 var folder = DriveApp.getFolderById(destinationId);

 folder.createFile(slideBlob);

 }

}
```

The SpreadsheetApp service is used to access the specified spreadsheet

Create a Google Form

This code will create a new Google Form.

```
function createForm() {

 // Create a new Google Form.

 var form = FormApp.create('My Form');

 // Add a question to the form.

 var question = form.addQuestion('What is your name?');

 question.setRequired(true);

 // Add another question to the form.

 var question = form.addQuestion('What is your email address?');

 question.setRequired(true);



 // Save the form.

 form.save();

}
```

This code works by first creating a new Google Form object. Then, it adds two questions to the form, one for the user's name and one for the user's email address. Finally, it saves the form.

This code could be used to create a form that users can fill out to provide their contact information. For example, you could use this code to create a form that users can fill out to sign up for your newsletter or to request a consultation.

Send an email notification

This code will send an email notification to the user.

```
function sendEmail() {
 // Get the user's email address.
 var userEmail = Utilities.getUserEmail();
 // Create an email message.
 var message = GmailApp.createEmail();
 message.setSubject('Your email notification');
 message.setBody('This is the body of your email notification.');
 // Send the email notification.
```

```
 message.sendTo(userEmail);

}
```

This code works by first getting the user's email address. Then, it creates an email message and sets the subject line and body of the message. Finally, it sends the email notification.

This code could be used to send email notifications to users about important events or updates. For example, you could use this code to send email notifications to users when their account is created or when their password is changed.

Create a Google Sheet

This code will create a new Google Sheet.

```
function createSheet() {

 // Create a new Google Sheet.

 var sheet = SpreadsheetApp.create('My Sheet');



 // Add a row to the sheet.

 var row = sheet.appendRow([1, 2, 3]);
```

```
 // Add another row to the sheet.

 var row = sheet.appendRow([4, 5, 6]);



 // Save the sheet.

 sheet.save();

}
```

This code works by first creating a new Google Sheet object. Then, it adds two rows to the sheet, one with the values 1, 2, and 3 and one with the values 4, 5, and 6. Finally, it saves the sheet.

This code could be used to create a Google Sheet that stores data. For example, you could use this code to create a Google Sheet that stores the names and contact information of your customers.

Get the current time

This code will get the current time.

```
function getCurrentTime() {

 // Get the current time.

 var currentTime = Utilities.now();



 // Display the current time.

 Logger.log('The current time is: ' + currentTime);

}
```

This code works by first getting the current time using the Utilities.now() method. Then, it displays the current time using the Logger.log() method.

This code could be used to get the current time for any purpose. For example, you could use this code to get the current time to display on a website or to log the current time in a database.

Get the user's location

This code will get the user's location.

```
function getUserLocation() {

 // Get the user's location.

 var userLocation = Geolocation.getGeolocation();



 // Display the user's location.

 Logger.log('The user's location is: ' + userLocation);

}
```

This code works by first getting the user's location using the Geolocation.getGeolocation() method. Then, it displays the user's location using the Logger.log() method.

Learn more about Google Apps Script

Google Apps Script is a scripting language developed by Google that allows you to automate tasks and extend the functionality of Google Workspace apps. With Apps Script, you can write scripts that do things like:

- Automate tasks. Create scripts that automate repetitive tasks, such as sending emails, updating spreadsheets, or creating presentations.
- Extend the functionality of Google Workspace apps. Create custom functions and features for Google Sheets, Docs, Slides, Forms, Gmail, Calendar, Drive, and more.
- Build web apps. Create web apps that can be used by anyone, even those who don't have Google Workspace accounts.
- Connect to other APIs. Apps Script can be used to connect to other APIs, such as Twitter, Facebook, and Google Cloud Platform.

To use Apps Script, you need a Google account. Once you have a Google account, you can create a new Apps Script project in the Google Cloud Platform Console.

Once you have created a project, you can start writing scripts. Apps Script uses the JavaScript programming language, so if you know JavaScript, you'll be able to get started quickly. If you don't know JavaScript, there are plenty of resources available to help you learn.

Once you've written a script, you can run it from the Google Cloud Platform Console or from within a Google Workspace app. You can also publish your scripts so that other people can use them.

Apps Script is a powerful tool that can help you automate tasks, extend the functionality of Google Workspace apps, and build web apps. If you're looking for a way to get more out of Google Workspace, Apps Script is a great place to start.

How to create a new spreadsheet?

```
function createSpreadsheet() {

 // Create a new spreadsheet.

 var spreadsheet = SpreadsheetApp.create('My New Spreadsheet');

 // Add a sheet to the spreadsheet.

 var sheet = spreadsheet.getActiveSheet();

 // Set the sheet's title.

 sheet.setTitle('My First Sheet');

}
```

How to add a row to a spreadsheet?

```
function addRow(spreadsheet, sheet, values) {
```

```
 // Get the last row in the spreadsheet.

 var lastRow = sheet.getLastRow();

 // Add a new row to the spreadsheet.

 var newRow = sheet.insertRow(lastRow + 1);

 // Set the values of the new row.

 for (var i = 0; i < values.length; i++) {

 newRow.setValue(i, values[i]);

 }

}
```

How to get the value of a cell in a spreadsheet?

```
function getValue(spreadsheet, sheet, row, column) {

 // Get the cell at the specified row and column.

 var cell = sheet.getRange(row, column);
```

```
 // Get the value of the cell.

 var value = cell.getValue();

 // Return the value.

 return value;

}
```

How to set the value of a cell in a spreadsheet?

```
function setValue(spreadsheet, sheet, row, column, value) {

 // Get the cell at the specified row and column.

 var cell = sheet.getRange(row, column);

 // Set the value of the cell.

 cell.setValue(value);

}
```

How to send an email?

```
function sendEmail(to, subject, body) {
 // Create a new mail message.
 var mail = GmailApp.createEmail();
 // Set the mail's recipient.
 mail.setRecipient(to);
 // Set the mail's subject.
 mail.setSubject(subject);
 // Set the mail's body.
 mail.setBody(body);
 // Send the mail.
 mail.send();
}
```

How to create a new form?

```
function createForm() {
```

```
 // Create a new form.

 var form = FormApp.create('My New Form');

 // Add a question to the form.

 var question = form.addQuestion('What is your name?');

 // Set the question's type.

 question.setType('text');

 // Add another question to the form.

 var question = form.addQuestion('What is your email address?');

 // Set the question's type.

 question.setType('email');

}
```

How to send a notification?

```
function sendNotification(message) {

 // Create a new notification.
```

```
 var notification = Notifications.create('My New Notification');

 // Set the notification's message.

 notification.setMessage(message);

 // Send the notification.

 notification.send();

}
```

How to create a new calendar event?

```
function createCalendarEvent(title, start, end) {

 // Create a new calendar event.

 var event = CalendarApp.createEvent(title);

 // Set the event's start date.

 event.setStart(start);

 // Set the event's end date.

 event.setEnd(end);
```

```
 // Add the event to the calendar.
 CalendarApp.insertEvent(event);
}
```

How to get the current date and time?

```
function getCurrentDateTime() {
 // Get the current date and time.
 var dateTime = Utilities.formatDate(new Date(), 'GMT-05:00');
 // Return the date and time.
 return dateTime;
}
```

How to log an error?

```
function logError(message) {
```

```
 // Log an error.
 Logger.log(LoggingLevel.ERROR, message);
}
```

How to get the value of a property?

```
function getProperty(object, property) {
 // Get the value of the specified property.
 var value = object[property];
 // Return the value.
 return value;
}
```

How to set the value of a property?

```
function setProperty(object, property, value) {
```

```
 // Set the value of the specified property.

 object[property] = value;

}
```

How to loop through an array?

```
function loopThroughArray(array) {

 // Loop through the array.

 for (var i = 0; i < array.length; i++) {

 // Do something with the current element.

 var element = array[i];

 }

}
```

How to call a function asynchronously?

```
function callFunctionAsynchronously(functionName, parameters) {
 // Create a new task.
 var task = new Task(functionName, parameters);
 // Add the task to the queue.
 TaskQueue.add(task);
}
```

How to get the current user?

```
function getCurrentUser() {
 // Get the current user.
 var user = UserInfo.getActiveUser();
 // Return the user.
 return user;
}
```

How to create a new folder?

```
function createFolder(folderName) {
 // Create a new folder.
 var folder = DriveApp.createFolder(folderName);
}
```

How to upload a file?

```
function uploadFile(fileName, filePath) {
 // Upload a file.
 var file = DriveApp.createFile(fileName);
 file.setContent(filePath);
}
```

How to download a file?

```
function downloadFile(fileName, filePath) {

 // Download a file.

 var file = DriveApp.getFileById(fileName);

 file.download(filePath);

}
```

How to create a new link?

```
function createLink(url) {

 // Create a new link.

 var link = Link.create(url);

}
```

How to get the size of a file?

```
function getFileSize(fileName) {
 // Get the size of a file.
 var file = DriveApp.getFileById(fileName);
 var size = file.getSize();
 // Return the size.
 return size;
}
```

How to get the type of a file?

```
function getFileType(fileName) {
 // Get the type of a file.
 var file = DriveApp.getFileById(fileName);
 var type = file.getType();
 // Return the type.
```

```
 return type;
}
```

How to get the last modified date of a file?

```
function getLastModifiedDate(fileName) {
 // Get the last modified date of a file.
 var file = DriveApp.getFileById(fileName);
 var date = file.getLastModifiedDate();
 // Return the date.
 return date;
}
```

How to make a copy of a file?

```
function copyFile(fileName, newFileName) {
```

```
 // Make a copy of a file.

 var file = DriveApp.getFileById(fileName);

 var copy = file.copy(newFileName);

}
```

How to move a file to a new folder?

```
function moveFile(fileName, newFolderId) {

 // Move a file to a new folder.

 var file = DriveApp.getFileById(fileName);

 file.moveTo(newFolderId);

}
```

How to delete a file?

```
function deleteFile(fileName) {
```

```
 // Delete a file.

 var file = DriveApp.getFileById(fileName);

 file.delete();

}
```

How to create a new slide in a presentation?

```
function createSlide(presentation, slideTitle) {

 // Create a new slide in a presentation.

 var slide = presentation.createSlide(slideTitle);

}
```

How to add a text box to a slide?

```
function addTextBox(slide, text) {

 // Add a text box to a slide.

 var textbox = slide.addTextBox();
```

```
 textbox.setText(text);
}
```

How to add an image to a slide?

```
function addImage(slide, imageUrl) {
 // Add an image to a slide.
 var image = slide.addImage(imageUrl);
}
```

How to add a shape to a slide?

```
function addShape(slide, shapeType, shapeOptions) {
 // Add a shape to a slide.
 var shape = slide.addShape(shapeType, shapeOptions);
}
```

How to change the slide layout?

```
function changeSlideLayout(slide, layoutId) {

 // Change the slide layout.

 slide.setLayout(layoutId);

}
```

How to add a transition to a slide?

```
function addTransition(slide, transition) {

 // Add a transition to a slide.

 slide.addTransition(transition);

}
```

How to add an animation to an object on a slide?

```
function addAnimation(object, animation) {
 // Add an animation to an object on a slide.
 object.addAnimation(animation);
}
```

How to preview a presentation?

```
function previewPresentation(presentation) {
 // Preview a presentation.
 presentation.preview();
}
```

How to publish a presentation?

```
function publishPresentation(presentation, publishOptions) {
 // Publish a presentation.
```

```
 presentation.publish(publishOptions);
}
```

How to protect a presentation?

```
function protectPresentation(presentation, password) {
 // Protect a presentation.
 presentation.protect(password);
}
```

How to create a new document?

```
function createDocument(documentType) {
 // Create a new document.
 var document = DocumentApp.create(documentType);
}
```

How to add a paragraph to a document?

```
function addParagraph(document, text) {
 // Add a paragraph to a document.
 var paragraph = document.appendParagraph(text);
}
```

How to add a heading to a document?

```
function addHeading(document, level, text) {
 // Add a heading to a document.
 var heading = document.appendHeading(level, text);
}
```

How to add a list to a document?

```
function addList(document, listType) {
```

```
 // Add a list to a document.

 var list = document.appendList(listType);

}
```

How to add an image to a document?

```
function addImage(document, imageUrl) {

 // Add an image to a document.

 var image = document.appendImage(imageUrl);

}
```

How to add a table to a document?

```
function addTable(document, rows, columns) {

 // Add a table to a document.

 var table = document.appendTable(rows, columns);
```

```
}
```

How to add a link to a document?

```
function addLink(document, text, url) {

 // Add a link to a document.

 var link = document.createLink(text, url);

}
```

How to add a footnote to a document?

```
function addFootnote(document, text) {

 // Add a footnote to a document.

 var footnote = document.appendFootnote(text);

}
```

How to add a comment to a document?

```
function addComment(document, text) {

 // Add a comment to a document.

 var comment = document.appendComment(text);

}
```

How to protect a document?

```
function protectDocument(document, password) {

 // Protect a document.

 document.protect(password);

}
```

Send email 10 most recent files exercise

```
function myFunction() {

 // This function will send an email to the user with a list of their top 10 most recent Google Sheets files.
```

```
// Get the user's email address.
var userEmail = Utilities.getUserEmail();

// Get the list of the user's top 10 most recent Google Sheets files.
var files = DriveApp.getFilesByType('spreadsheet').sort('created').limit(10);

// Create an email message.
var message = GmailApp.createEmail();

// Set the email's subject line.
message.setSubject('Your top 10 most recent Google Sheets files');

// Set the email's body text.
message.setBody('Here is a list of your top 10 most recent Google Sheets
files:');
```

```
// Add the list of files to the email's body text.
for (var i = 0; i < files.length; i++) {
message.appendBody(files[i].getName() + '\n');
}

// Send the email.
message.send();
}
```

This function will send an email to the user with a list of their top 10 most recent Google Sheets files. It does this by first getting the user's email address and then getting the list of the user's top 10 most recent Google Sheets files. Once it has these two pieces of information, it creates an email message, sets the email's subject line and body text, and then sends the email.

This function is a good example of how Apps Script can be used to automate tasks. In this case, it is automating the task of sending an email to the user with a list of their top 10 most recent Google Sheets files. This could be a

useful task for users who want to keep track of their recent work or who want to share their recent work with others.

Populate from Google Form Exercise

```
function myFunction() {

 // This function will create a new Google Sheets file and populate it with data from a Google Form.



 // Get the form ID.

 var formId = '1234567890';



 // Get the list of form responses.

 var responses = FormApp.getResponses(formId);



 // Create a new Google Sheets file.

 var sheet = SpreadsheetApp.create('My Form Responses');
```

```
// Add a new sheet to the spreadsheet.

var newSheet = sheet.getActiveSheet();



// Add a column for each field in the form.

for (var i = 0; i < responses[0].length; i++) {

newSheet.appendRow([responses[0][i]]);

}



// Rename the columns.

newSheet.renameColumn(0, 'Name');

newSheet.renameColumn(1, 'Email');

newSheet.renameColumn(2, 'Phone');

newSheet.renameColumn(3, 'Comment');



// Set the sheet's title.
```

```
 newSheet.setTitle('My Form Responses');



 // Display the sheet.

 sheet.show();

}
```

This function will create a new Google Sheets file and populate it with data from a Google Form. It does this by first getting the form ID and then getting the list of form responses. Once it has these two pieces of information, it creates a new Google Sheets file, adds a new sheet to the file, adds a column for each field in the form, renames the columns, and sets the sheet's title. Finally, it displays the sheet.

This function is a good example of how Apps Script can be used to automate tasks. In this case, it is automating the task of creating a new Google Sheets file and populating it with data from a Google Form. This could be a useful task for users who want to collect data from a Google Form and then analyze the data in a Google Sheets file.

Create slides from Google Sheet exercise

```
function myFunction() {
```

```
// This function will create a new Google Slides presentation and populate it with data from a Google Sheet.


// Get the sheet ID.

var sheetId = '1234567890';


// Get the list of sheet data.

var data = SpreadsheetApp.openById(sheetId).getDataRange().getValues();


// Create a new Google Slides presentation.

var presentation = SlidesApp.create('My Slides Presentation');


// Add a new slide to the presentation.

var slide = presentation.insertSlide();
```

```
// Add a title slide to the presentation.

slide.setTitle('My Slides Presentation');



// Add a subtitle slide to the presentation.

slide = presentation.insertSlide();

slide.setSubtitle('This is a subtitle slide.');



// Add a new slide for each row of data in the sheet.

for (var i = 0; i < data.length; i++) {

slide = presentation.insertSlide();



// Add a title to the slide.

slide.setTitle(data[i][0]);



// Add a paragraph to the slide for each column of data in the row.
```

```
 for (var j = 1; j < data[i].length; j++) {

 slide.appendParagraph(data[i][j]);

 }

 }



 // Save the presentation.

 presentation.save();

}
```

This function will create a new Google Slides presentation and populate it with data from a Google Sheet. It does this by first getting the sheet ID and then getting the list of sheet data. Once it has these two pieces of information, it creates a new Google Slides presentation, adds a new slide to the presentation, adds a title slide to the presentation, adds a subtitle slide to the presentation, adds a new slide for each row of data in the sheet, adds a title to each slide, and adds a paragraph to each slide for each column of data in the row. Finally, it saves the presentation.

This function is a good example of how Apps Script can be used to automate tasks. In this case, it is automating the task of creating a new Google Slides presentation and populating it with data from a Google Sheet. This could be

a useful task for users who want to create presentations from data that is stored in a Google Sheet.

Create calendar event email user exercise

```
function myFunction() {

 // This function will create a new Google Calendar event and send an email notification to the user.



 // Get the user's email address.

 var userEmail = Utilities.getUserEmail();



 // Get the event details.

 var eventTitle = 'My Event';

 var eventStart = new Date();

 var eventEnd = new Date();

 var eventDescription = 'This is the description of my event.';
```

```
 // Create a new Google Calendar event.

 var event = CalendarApp.createEvent(eventTitle, eventStart, eventEnd,
eventDescription);



 // Send an email notification to the user.

 var message = GmailApp.createEmail();

 message.setSubject('Your event reminder');

 message.setBody('You have an event scheduled for ' + eventStart + '.
Please see the details below.');

 message.appendBody('Event title: ' + eventTitle);

 message.appendBody('Event start date: ' + eventStart);

 message.appendBody('Event end date: ' + eventEnd);

 message.appendBody('Event description: ' + eventDescription);

 message.sendTo(userEmail);

}
```

This function will create a new Google Calendar event and send an email notification to the user. It does this by first getting the user's email address

and then getting the event details. Once it has these two pieces of information, it creates a new Google Calendar event, sends an email notification to the user, and then displays the event.

This function is a good example of how Apps Script can be used to automate tasks. In this case, it is automating the task of creating a new Google Calendar event and sending an email notification to the user. This could be a useful task for users who want to create events in Google Calendar and then send email notifications to remind themselves or others about the events.

Create File upload to Drive location exercise

```
function myFunction() {

 // This function will create a new Google Drive file and upload it to a specific folder.



 // Get the file name.

 var fileName = 'My File.txt';



 // Get the content of the file.

 var content = 'This is the content of my file.';
```

```
// Create a new Google Drive file.

var file = DriveApp.createFile(fileName);



// Set the content of the file.

file.setContent(content);



// Upload the file to a specific folder.

var folderId = '1234567890';

file.moveToFolderById(folderId);

}
```

This function will create a new Google Drive file and upload it to a specific folder. It does this by first getting the file name and then getting the content of the file. Once it has these two pieces of information, it creates a new Google Drive file, sets the content of the file, and then uploads the file to a specific folder.

This function is a good example of how Apps Script can be used to automate tasks. In this case, it is automating the task of creating a new Google Drive

file and uploading it to a specific folder. This could be a useful task for users who want to create and store files in Google Drive.