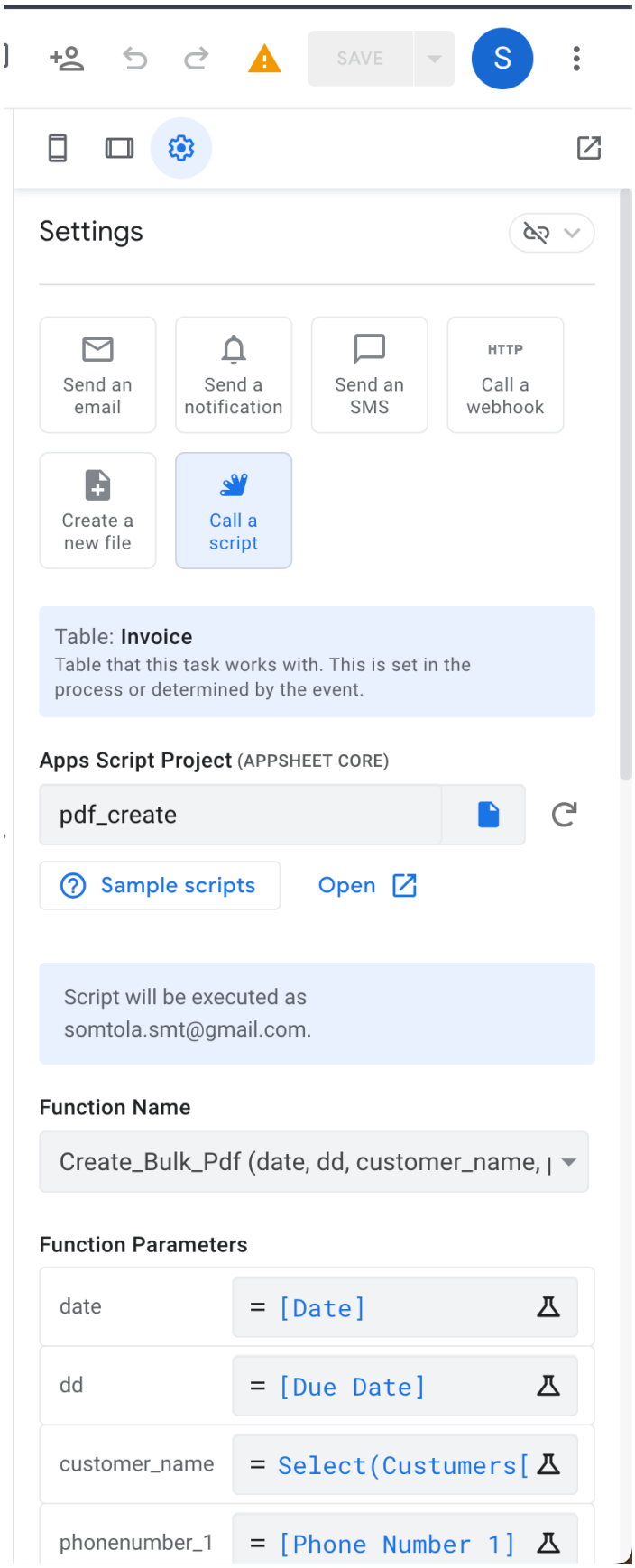
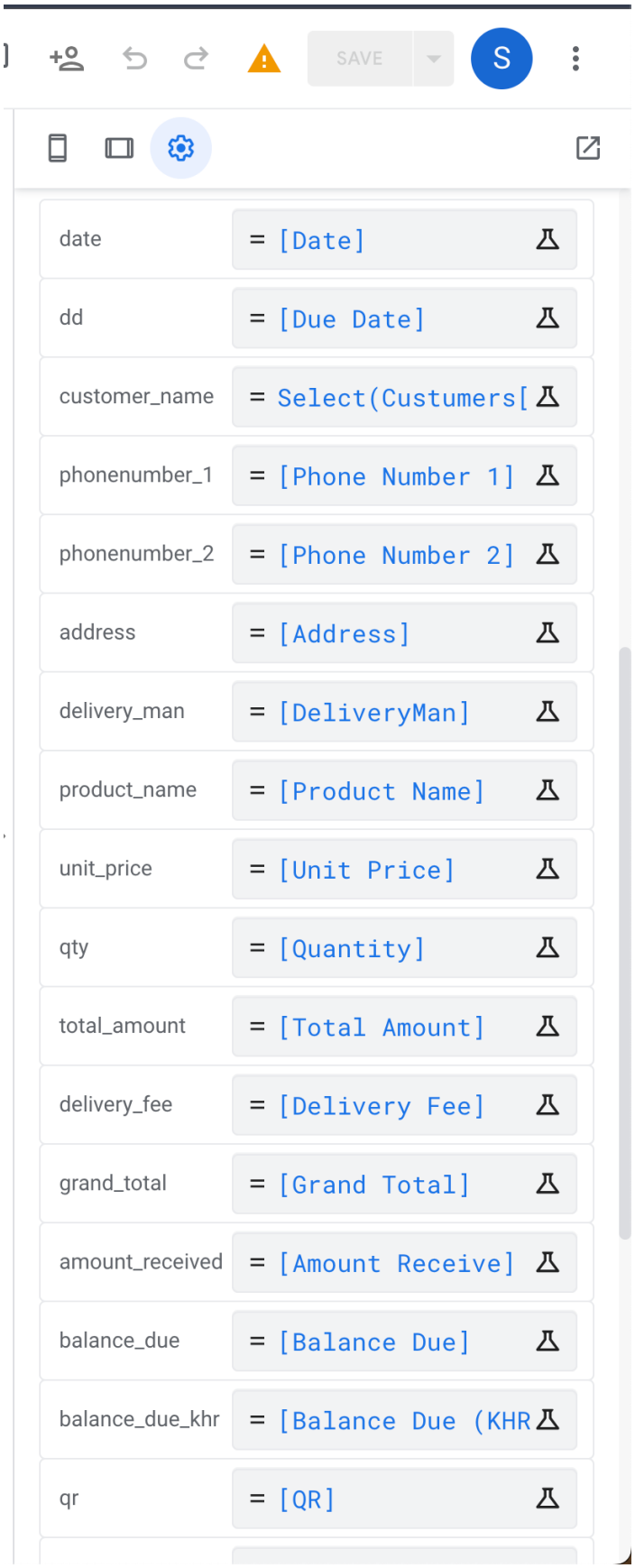
Appsheet (Paramater) – Post data to App Script





App Script use for execute on generate pdf and printnode

- Script

- Main function

```
function Create_Bulk_Pdf(date, dd, customer_name, phonenumber_1,
phonenumber_2, address, delivery_man, product_name, unit_price, qty,
total_amount, delivery_fee, grand_total, amount_received, balance_due,
balance_due_khr, qr, order, cashier, note, email, invID) {
  const fileId = "1yBUWsB0bnhqVhwhS_lsVykEqD1RRYH4fX8piGaf8YUk";
  const folderId = "1vIVcfwOJ5IG3YswgW9da14a04Yh11Sil";
  const folder = DriveApp.getFolderById(folderId);
  const tempFile = DriveApp.getFileById(fileId).makeCopy(folder);
  const docFile = DocumentApp.openById(tempFile.getId());

  const replacements = {
    "{date}": date,
    "{dd}": dd,
    "{customer_name}": customer_name,
    "{phonenumber_1}": phonenumber_1,
    "{phonenumber_2}": phonenumber_2,
    "{address}": address,
    "{delivery_man}": delivery_man,
    "{delivery_fee}": parseFloat(delivery_fee).toFixed(2),
    "{grand_total}": parseFloat(grand_total).toFixed(2),
    "{amount_received}": parseFloat(amount_received).toFixed(2),
    "{balance_due}": parseFloat(balance_due).toFixed(2),
    "{balance_due_khr}": parseFloat(balance_due_khr).toLocaleString('en-US'),
    "{order}": order,
    "{cashier}": cashier,
    "{note}": note,
  };

  // Perform all replacements in one go
  const body = docFile.getBody();
  Object.keys(replacements).forEach(key => {
    body.replaceText(key, replacements[key]);
  });

  // Batch replace placeholders for product details
  replacePlaceholders(docFile, product_name, "{product_name_", 10, false);
  replacePlaceholders(docFile, unit_price, "{unit_price_", 10, true);
  replacePlaceholders(docFile, qty, "{qty_", 10, false);
  replacePlaceholders(docFile, total_amount, "{total_amount_", 10, true);
```

```
  // Insert QR code image if present
  if (qr) {
    try {
      const imageBlob = UrlFetchApp.fetch(qr).getBlob();
      const searchResult = body.findText('{image}');
      if (searchResult) {
        const rangeElement = searchResult.getElement();
        const paragraph = rangeElement.getParent();
        paragraph.clear(); // Clear the placeholder
        paragraph.insertInlineImage(0, imageBlob);
      }
    } catch (error) {
      console.log("Failed to fetch QR image: " + error.message);
    }
  }

  docFile.saveAndClose();

  // Create PDF and remove temporary file in one operation
  const pdfFile =
folder.createFile(tempFile.getAs(MimeType.PDF)).setName(invID);
  folder.removeFile(tempFile);

  // Perform email and PrintNode operations concurrently
  Promise.all([
    sendEmailAsync(email, customer_name, "Invoice was generated.", pdfFile),
    sendPrintNodeAsync(pdfFile, "73757406",
"WbT2Sm6TrSZubcDXYTtcLvAU8Y0S67Y1cd0taSRzo2A")
  ]).then(() => {
    console.log("Email sent and print job submitted.");
  });
}
```

- Helper function to replace placeholders in the document

```
function replacePlaceholders(doc, values, placeholderPrefix,
maxPlaceholders, isCurrency) {
  const body = doc.getBody();
  for (let i = 0; i < values.length; i++) {
    let placeholder = placeholderPrefix + (i + 1) + "}";
    let value = values[i];
    let formattedValue = isCurrency ? "$" +
parseFloat(value).toFixed(2) : value;
    body.replaceText(placeholder, formattedValue);
  }
  for (let i = values.length + 1; i <= maxPlaceholders; i++) {
    let unusedPlaceholder = placeholderPrefix + i + "}";
    body.replaceText(unusedPlaceholder, "");
  }
}
```

- Async email sending function

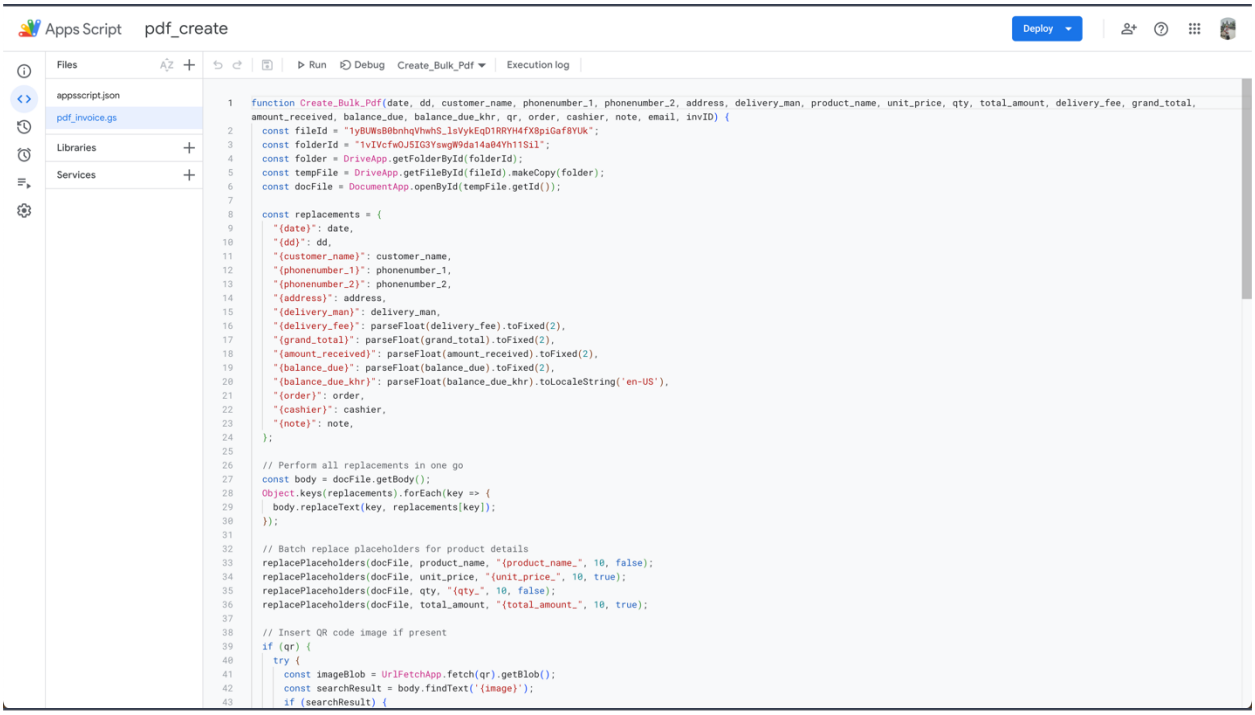
```
function sendEmailAsync(email, subject, body, pdfFile) {
  return new Promise((resolve) => {
    MailApp.sendEmail(email, subject, body, {
      attachments: [pdfFile.getAs(MimeType.PDF)],
    });
    resolve();
  });
}
```

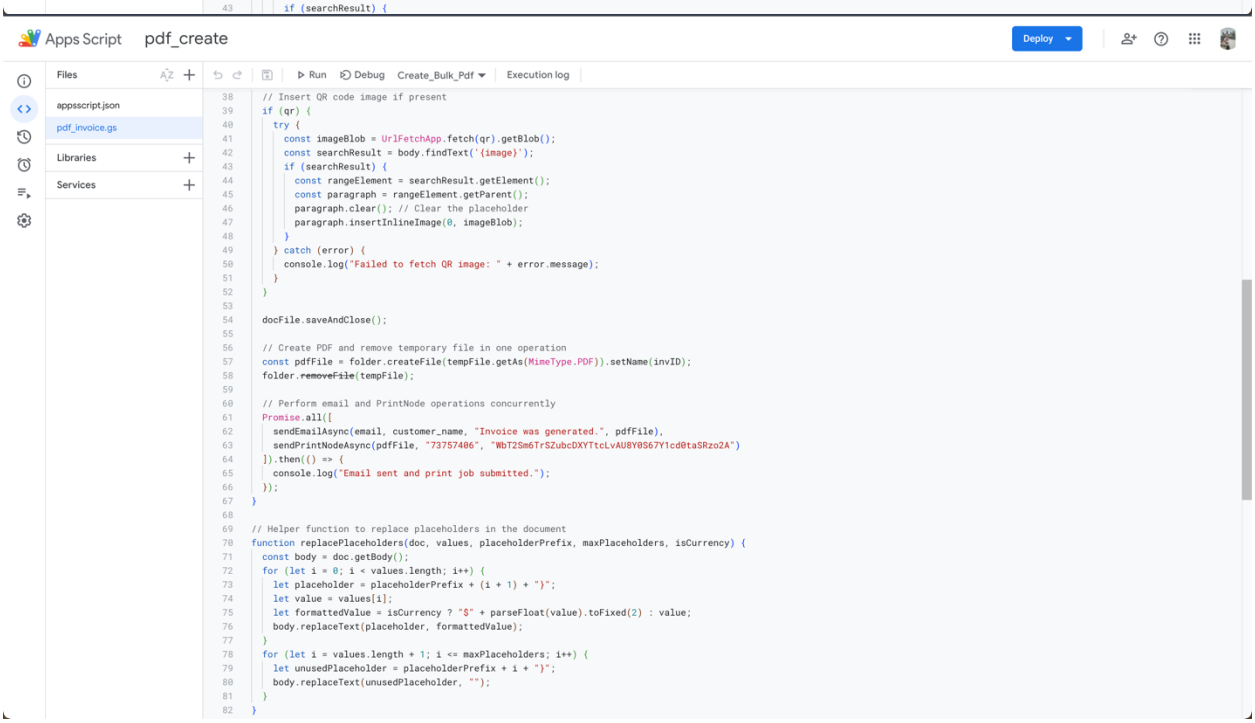
- Async function for sending print job

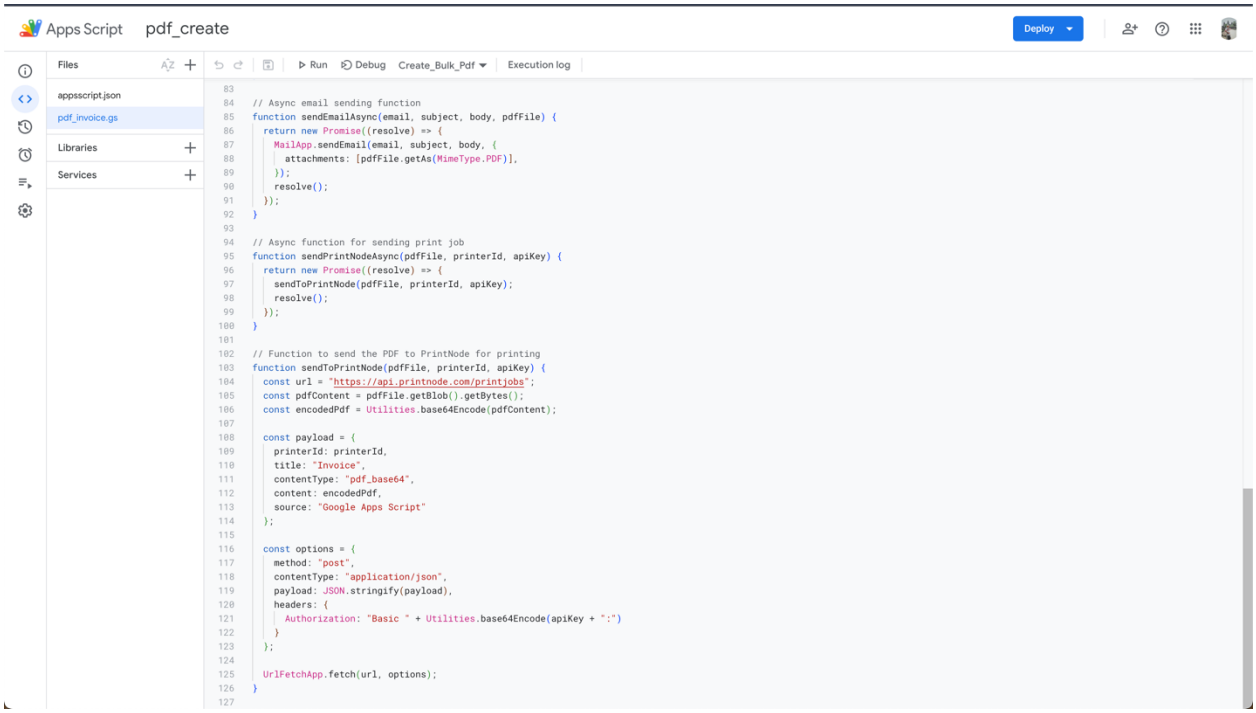
```
function sendPrintNodeAsync(pdfFile, printerId, apiKey) {
  return new Promise((resolve) => {
    sendToPrintNode(pdfFile, printerId, apiKey);
    resolve();
  });
}
```

- Function to send the pdf to PrintNode for printing

```
function sendToPrintNode(pdfFile, printerId, apiKey) {
  const url = "https://api.printnode.com/printjobs";
  const pdfContent = pdfFile.getBlob().getBytes();
  const encodedPdf = Utilities.base64Encode(pdfContent);

  const payload = {
    printerId: printerId,
    title: "Invoice",
    contentType: "pdf_base64",
    content: encodedPdf,
    source: "Google Apps Script"
  };

  const options = {
    method: "post",
    contentType: "application/json",
    payload: JSON.stringify(payload),
    headers: {
      Authorization: "Basic " + Utilities.base64Encode(apiKey + ":")
    }
  };

  UrlFetchApp.fetch(url, options);
}
```


Apps Script
pdf_create
Deploy
Files
appsscript.json
pdf_invoice.gs
Libraries
Services
Run
Debug
Create_Bulk_Pdf
Execution log
function Create_Bulk_Pdf(date, dd, customer_name, phonenumber_1, phonenumber_2, address, delivery_man, product_name, unit_price, qty, total_amount, delivery_fee, grand_total, amount_received, balance_due, balance_due_khr, qr, order, cashier, note, email, invID) {
const fileId = "1yBUNsB0bnhqVhwhS_lsVykEqD1RRYH4fX8piGaf8YUk";
const folderId = "1vIVcfwOJSIQ3YswgW9da14a04Yh11Sil";
const folder = DriveApp.getFolderById(folderId);
const tempFile = DriveApp.getFileById(fileId).makeCopy(folder);
const docFile = DocumentApp.openById(tempFile.getId());
const replacements = {
"{date}": date,
"{dd}": dd,
"{customer_name}": customer_name,
"{phonenumber_1}": phonenumber_1,
"{phonenumber_2}": phonenumber_2,
"{address}": address,
"{delivery_man}": delivery_man,
"{delivery_fee}": parseFloat(delivery_fee).toFixed(2),
"{grand_total}": parseFloat(grand_total).toFixed(2),
"{amount_received}": parseFloat(amount_received).toFixed(2),
"{balance_due}": parseFloat(balance_due).toFixed(2),
"{balance_due_khr}": parseFloat(balance_due_khr).toLocaleString('en-US'),
"{order}": order,
"{cashier}": cashier,
"{note}": note,
};
// Perform all replacements in one go
const body = docFile.getBody();
Object.keys(replacements).forEach(key => {
body.replaceText(key, replacements[key]);
});
// Batch replace placeholders for product details
replacePlaceholders(docFile, product_name, "{product_name_", 10, false);
replacePlaceholders(docFile, unit_price, "{unit_price_", 10, true);
replacePlaceholders(docFile, qty, "{qty_", 10, false);
replacePlaceholders(docFile, total_amount, "{total_amount_", 10, true);
// Insert QR code image if present
if (qr) {
try {
const imageBlob = UrlFetchApp.fetch(qr).getBlob();
const searchResult = body.findText('{image}');
if (searchResult) {


Apps Script
pdf_create
Deploy
Files
appsscript.json
pdf_invoice.gs
Libraries
Services
Run
Debug
Create_Bulk_Pdf
Execution log
// Insert QR code image if present
if (qr) {
try {
const imageBlob = UrlFetchApp.fetch(qr).getBlob();
const searchResult = body.findText('{image}');
if (searchResult) {
const rangeElement = searchResult.getElement();
const paragraph = rangeElement.getParent();
paragraph.clear(); // Clear the placeholder
paragraph.insertInlineImage(0, imageBlob);
}
} catch (error) {
console.log("Failed to fetch QR image: " + error.message);
}
}
docFile.saveAndClose();
// Create PDF and remove temporary file in one operation
const pdfFile = folder.createFile(tempFile.getAs(MimeType.PDF)).setName(invID);
folder.removeFile(tempFile);
// Perform email and PrintNode operations concurrently
Promise.all([
sendEmailAsync(email, customer_name, "Invoice was generated.", pdfFile),
sendPrintNodeAsync(pdfFile, "73757406", "WbT2Sm6TrSZubcDXYTtcLvAU8Y0S67Y1cd0taSRzo2A")
]).then(() => {
console.log("Email sent and print job submitted.");
});
}
// Helper function to replace placeholders in the document
function replacePlaceholders(doc, values, placeholderPrefix, maxPlaceholders, isCurrency) {
const body = doc.getBody();
for (let i = 0; i < values.length; i++) {
let placeholder = placeholderPrefix + (i + 1) + "}";
let value = values[i];
let formattedValue = isCurrency ? "$" + parseFloat(value).toFixed(2) : value;
body.replaceText(placeholder, formattedValue);
}
for (let i = values.length + 1; i <= maxPlaceholders; i++) {
let unusedPlaceholder = placeholderPrefix + i + "}";
body.replaceText(unusedPlaceholder, "");
}
}



- Template Google Docs
  Link:
  https://docs.google.com/document/d/1yBUWsB0bnhqVhwhS_lsVykEqD1RRYH4fX8piGaf8YUk/edit?usp=sharing