https://github.com/lsvekis/100-Advanced-Google-Apps-Script-Examples

100 Advanced Google Apps Script Examples

A) Sheets: Performance, Data Ops, Automation (1–25)

1) Batch update with `RangeList` (fast formatting)

Does: Applies formatting to many ranges in one go.

```
function ex01_rangeListFormat() {
  const sh = SpreadsheetApp.getActiveSheet();

sh.getRangeList(['A1:D1','A3:D3','A5:D5']).setFontWeight('bold').setBa
ckground('#f3f4f6');
}
```

2) Write values + formulas in one pass

Does: Sets values and formulas efficiently.

```
function ex02_valuesAndFormulas() {
  const sh = SpreadsheetApp.getActiveSheet();
  sh.getRange('A1:B1').setValues([['Item','Total']]);
  sh.getRange('B2:B').setFormulaR1C1('=IF(RC[-1]="","",LEN(RC[-1]))');
}
```

3) Build an index of unique keys → row numbers

Does: Creates a lookup map for fast searches.

```
function ex03_buildIndexMap() {
  const sh = SpreadsheetApp.getActiveSheet();
  const vals =
sh.getRange(2,1,sh.getLastRow()-1,1).getValues().flat();
  const map = {};
  vals.forEach((v,i)=> { const k=String(v).trim(); if(k) map[k]=i+2;
});
  Logger.log(map);
}
```

4) Upsert rows by key (update or append)

Does: Updates existing row if key exists, else appends.

```
function ex04_upsertByKey() {
  const sh = SpreadsheetApp.getActiveSheet();
  const keyCol = 1;
  const key = 'ABC-123';
  const payload = ['ABC-123','New Name', new Date()];
  const data = sh.getDataRange().getValues();
  const idx = data.findIndex((r,i)=> i>0 &&
String(r[keyCol-1]).trim()===key);
  if (idx > -1)
sh.getRange(idx+1,1,1,payload.length).setValues([payload]);
  else sh.appendRow(payload);
}
```

5) “Soft delete” rows (flag instead of deleting)

Does: Marks row as deleted for auditability.

```
function ex05_softDelete() {
  const sh = SpreadsheetApp.getActiveSheet();
  const row = 5;
  sh.getRange(row,
sh.getLastColumn()+1).setValue('DELETED').setNote('Soft deleted ' +
new Date());
```

```
}
```

6) Atomic log writer (prevents collisions)

Does: Uses `LockService` to avoid simultaneous writes.

```
function ex06_atomicLog() {
  const lock = LockService.getScriptLock();
  lock.waitLock(20000);
  try {
    const sh = SpreadsheetApp.getActive().getSheetByName('Logs') ||
SpreadsheetApp.getActive().insertSheet('Logs');
    sh.appendRow([new Date(), Session.getActiveUser().getEmail(),
'Event']);
  } finally {
    lock.releaseLock();
  }
}
```

7) Incremental processing with cursor (resume later)

Does: Processes 200 rows per run, saves cursor to properties.

```
function ex07_incrementalProcess() {
  const props = PropertiesService.getScriptProperties();
  const startRow = Number(props.getProperty('cursor') || 2);
  const sh = SpreadsheetApp.getActiveSheet();
  const last = sh.getLastRow();
  const batchSize = 200;
  const endRow = Math.min(last, startRow + batchSize - 1);
  if (startRow > last) { props.deleteProperty('cursor'); return; }

  const data =
sh.getRange(startRow,1,endRow-startRow+1,sh.getLastColumn()).getValues
();
  // do work...
  Logger.log(`Processed rows ${startRow}-${endRow}`);
```

```
  props.setProperty('cursor', String(endRow + 1));
}
```

8) De-duplicate sheet by composite key (keep first)

Does: Removes duplicates using key from multiple columns.

```
function ex08_dedupeComposite() {
  const sh = SpreadsheetApp.getActiveSheet();
  const data = sh.getDataRange().getValues();
  const header = data.shift();
  const seen = new Set();
  const out = [header];

  data.forEach(r=>{
    const key = [r[0],r[1],r[2]].map(v=>String(v).trim()).join('|');
    if (!key.replace(/\|/g,'')) return;
    if (!seen.has(key)) { seen.add(key); out.push(r); }
  });

  sh.clear();
  sh.getRange(1,1,out.length,out[0].length).setValues(out);
}
```

9) Add a “change log” note on edit (installable recommended)

Does: Appends notes with edit history.

```
function ex09_onEditNotes(e) {
  const r = e.range;
  if (r.getRow() === 1) return;
  const prev = r.getNote() || '';
  const msg = `${new Date().toISOString()} | ${e.oldValue ?? ''} →
${e.value ?? ''}`;
  r.setNote((prev ? prev + '\n' : '') + msg);
}
```

10) Create a protected “admin” column

Does: Protects a column while leaving others editable.

```
function ex10_protectAdminColumn() {
  const sh = SpreadsheetApp.getActiveSheet();
  const p = sh.getRange('H:H').protect().setDescription('Admin only');
  p.removeEditors(p.getEditors());
}
```

11) Generate a data dictionary tab

Does: Lists headers + column index + sample values.

```
function ex11_dataDictionary() {
  const ss = SpreadsheetApp.getActive();
  const sh = ss.getActiveSheet();
  const dict = ss.getSheetByName('Data Dictionary') ||
ss.insertSheet('Data Dictionary');
  const headers =
sh.getRange(1,1,1,sh.getLastColumn()).getValues()[0];
  const samples = sh.getRange(2,1,Math.min(50, sh.getLastRow()-1),
sh.getLastColumn()).getValues();

  const out = [['Column','Index','Sample Values']];
  headers.forEach((h,i)=>{
    const sample = samples.map(r=>r[i]).filter(v=>v!=='' &&
v!=null).slice(0,5).join(', ');
    out.push([h, i+1, sample]);
  });

  dict.clear();
  dict.getRange(1,1,out.length,out[0].length).setValues(out);
}
```

12) Create a “validated import” pipeline

Does: Validates required columns before accepting import.

```
function ex12_validateImport() {
  const sh = SpreadsheetApp.getActiveSheet();
```

```
  const headers =
sh.getRange(1,1,1,sh.getLastColumn()).getValues()[0].map(String);
  const required = ['Email','Name','Status'];
  const missing = required.filter(r=>!headers.includes(r));
  if (missing.length) throw new Error('Missing columns: ' +
missing.join(', '));
  Logger.log('Import schema OK');
}
```

13) Apply smart filters + freeze + styling (setup table)

```
function ex13_tableSetup() {
  const sh = SpreadsheetApp.getActiveSheet();
  sh.setFrozenRows(1);
  const r = sh.getDataRange();
  r.createFilter();
  sh.getRange(1,1,1,sh.getLastColumn()).setFontWeight('bold');
}
```

14) Split sheet into multiple sheets by category

```
function ex14_splitByCategory() {
  const ss = SpreadsheetApp.getActive();
  const sh = ss.getActiveSheet();
  const data = sh.getDataRange().getValues();
  const header = data.shift();
  const catCol = 3; // column C

  const buckets = {};
  data.forEach(r=>{
    const k = String(r[catCol-1]).trim() || 'Uncategorized';
    (buckets[k] ||= []).push(r);
  });

  Object.entries(buckets).forEach(([name,rows])=>{
    const tab = ss.getSheetByName(name) || ss.insertSheet(name);
    tab.clear();
    tab.getRange(1,1,1,header.length).setValues([header]);
    tab.getRange(2,1,rows.length,header.length).setValues(rows);
```

```
  });
}
```

15) Create an audit snapshot (copy values only)

```
function ex15_snapshotValuesOnly() {
  const ss = SpreadsheetApp.getActive();
  const src = ss.getActiveSheet();
  const snap = ss.insertSheet(`Snapshot ${Utilities.formatDate(new
Date(), ss.getSpreadsheetTimeZone(), 'yyyy-MM-dd HH:mm')}`);
  const values = src.getDataRange().getValues();
  snap.getRange(1,1,values.length,values[0].length).setValues(values);
}
```

16) Export a range to CSV and save to Drive

```
function ex16_exportRangeToCsv() {
  const sh = SpreadsheetApp.getActiveSheet();
  const values = sh.getRange('A1:D20').getValues();
  const csv =
values.map(r=>r.map(v=>`"${String(v).replace(/"/g,'""')}"`).join(','))
.join('\n');
  const file = DriveApp.createFile('export.csv', csv, MimeType.CSV);
  Logger.log(file.getUrl());
}
```

17) Custom function with caching

```
function EX17_CACHED_LEN(input) {
  const cache = CacheService.getScriptCache();
  const key = 'len:' + input;
  const hit = cache.get(key);
  if (hit) return Number(hit);
  const val = String(input ?? '').length;
  cache.put(key, String(val), 300);
  return val;
}
```

18) Named range “schema” enforcer

```
function ex18_enforceNamedRanges() {
  const ss = SpreadsheetApp.getActive();
  const required = ['INPUTS','OUTPUTS'];
  required.forEach(n=>{
    if (!ss.getRangeByName(n)) throw new Error(`Missing named range:
${n}`);
  });
  Logger.log('All named ranges present');
}
```

19) Auto-create weekly tabs

```
function ex19_createWeeklyTabs() {
  const ss = SpreadsheetApp.getActive();
  const tz = ss.getSpreadsheetTimeZone();
  for (let i=0;i<4;i++){
    const d = new Date(Date.now() + i*7*24*3600*1000);
    const name = 'Week ' + Utilities.formatDate(d, tz, 'YYYY-ww');
    if (!ss.getSheetByName(name)) ss.insertSheet(name);
  }
}
```

20) Conditional formatting rule builder (dynamic range)

```
function ex20_conditionalFormattingDynamic() {
  const sh = SpreadsheetApp.getActiveSheet();
  const rng = sh.getRange(2,4,Math.max(1,sh.getLastRow()-1),1); //
column D from row 2
  const rule = SpreadsheetApp.newConditionalFormatRule()
    .whenTextContains('ERROR')
    .setBackground('#ffe4e6')
    .setRanges([rng]).build();
  sh.setConditionalFormatRules([...sh.getConditionalFormatRules(),
rule]);
}
```

21) Import JSON into sheet (flatten basic objects)

```
function ex21_importJsonToSheet() {
  const url = 'https://httpbin.org/json';
  const json = JSON.parse(UrlFetchApp.fetch(url).getContentText());
  const sh = SpreadsheetApp.getActiveSheet();
  const rows = Object.entries(json).map(([k,v])=>[k, typeof v ===
'object' ? JSON.stringify(v) : v]);
  sh.getRange(1,1,rows.length,2).setValues(rows);
}
```

22) Sheet “safe write” (verify headers match)

```
function ex22_safeWriteByHeaders() {
  const sh = SpreadsheetApp.getActiveSheet();
  const headers =
sh.getRange(1,1,1,sh.getLastColumn()).getValues()[0];
  const expected = ['ID','Name','Status'];
  if (expected.join('|') !== headers.slice(0,3).join('|')) throw new
Error('Header mismatch.');
  sh.getRange(2,1,1,3).setValues([[1,'Test','OK']]);
}
```

23) Batch delete rows by predicate (bottom-up)

```
function ex23_deleteRowsIfBlankColB() {
  const sh = SpreadsheetApp.getActiveSheet();
  const last = sh.getLastRow();
  const vals = sh.getRange(2,2,last-1,1).getValues().flat();
  for (let i=vals.length-1;i>=0;i--) if (!String(vals[i]).trim())
sh.deleteRow(i+2);
}
```

24) Build a mini dashboard sheet (KPIs)

```
function ex24_dashboardKpis() {
  const ss = SpreadsheetApp.getActive();
  const dataSh = ss.getActiveSheet();
  const dash = ss.getSheetByName('Dashboard') ||
ss.insertSheet('Dashboard');
  const last = dataSh.getLastRow();
```

```
  const total = Math.max(0, last-1);
  const unique = new
Set(dataSh.getRange(2,1,total,1).getValues().flat().map(v=>String(v).t
rim()).filter(Boolean)).size;
  const out = [['Metric','Value'],['Total rows', total],['Unique IDs
(col A)', unique],['Updated', new Date()]];
  dash.clear();
  dash.getRange(1,1,out.length,2).setValues(out);
  dash.getRange('A1:B1').setFontWeight('bold');
}
```

25) Create and refresh an Apps Script “config” sheet

```
function ex25_configSheet() {
  const ss = SpreadsheetApp.getActive();
  const sh = ss.getSheetByName('Config') || ss.insertSheet('Config');
  const out =
[['Key','Value'],['ENV','dev'],['MAX_BATCH','200'],['LAST_RUN', new
Date().toISOString()]];
  sh.clear(); sh.getRange(1,1,out.length,2).setValues(out);
}
```

B) Drive: File Ops, Search, Structure (26–45)

26) Find files by query and list URLs

```
function ex26_driveSearch() {
  const it =
DriveApp.searchFiles('mimeType="application/vnd.google-apps.document"
and trashed=false');
  let n=0;
  while (it.hasNext() && n<10) { const f=it.next();
Logger.log(f.getName() + ' ' + f.getUrl()); n++; }
}
```

27) Deep clone folder (copies files + subfolders)

```
function ex27_cloneFolder() {
  const sourceId = 'PASTE_SOURCE_FOLDER_ID';
  const targetParentId = 'PASTE_TARGET_PARENT_FOLDER_ID';
  const src = DriveApp.getFolderById(sourceId);
  const parent = DriveApp.getFolderById(targetParentId);
  const dst = parent.createFolder(src.getName() + ' (Copy)');
  cloneFolder_(src, dst);
}
function cloneFolder_(src, dst) {
  const files = src.getFiles();
  while (files.hasNext()) dst.addFile(files.next().makeCopy());
  const folders = src.getFolders();
  while (folders.hasNext()) {
    const sub = folders.next();
    const newSub = dst.createFolder(sub.getName());
    cloneFolder_(sub, newSub);
  }
}
```

28) Create a “year/month” folder path if missing

```
function ex28_yearMonthFolders() {
  const rootId = 'PASTE_ROOT_FOLDER_ID';
  const root = DriveApp.getFolderById(rootId);
  const now = new Date();
  const year = String(now.getFullYear());
  const month = Utilities.formatDate(now, Session.getScriptTimeZone(),
'MM');
  const y = getOrCreateSub_(root, year);
  const m = getOrCreateSub_(y, month);
  Logger.log(m.getUrl());
}
function getOrCreateSub_(parent, name) {
  const it = parent.getFoldersByName(name);
  return it.hasNext() ? it.next() : parent.createFolder(name);
}
```

29) Convert a PDF/image to Google Doc (Drive OCR requires API—not built-in)

Does: Notes limitation; Apps Script alone can't OCR without external service.

```
function ex29_noteOcrLimit() {
  Logger.log('Apps Script DriveApp cannot OCR by itself. Use Drive API
+ "ocr" parameter or external OCR service.');
}
```

30) Move all files older than X days to "Archive"

```
function ex30_archiveOldFiles() {
  const folderId = 'PASTE_FOLDER_ID';
  const archiveId = 'PASTE_ARCHIVE_FOLDER_ID';
  const days = 30;
  const cutoff = Date.now() - days*24*3600*1000;

  const src = DriveApp.getFolderById(folderId);
  const archive = DriveApp.getFolderById(archiveId);
  const files = src.getFiles();
  while (files.hasNext()) {
    const f = files.next();
    if (f.getLastUpdated().getTime() < cutoff) archive.addFile(f),
src.removeFile(f);
  }
}
```

31) Rename files with a consistent prefix

```
function ex31_prefixRename() {
  const folderId='PASTE_FOLDER_ID';
  const prefix='2026-';
  const folder=DriveApp.getFolderById(folderId);
  const files=folder.getFiles();
  while(files.hasNext()){
    const f=files.next();
    if(!f.getName().startsWith(prefix)) f.setName(prefix +
f.getName());
```

```
  }
}
```

32) Create share links (view-only) for a list of files

```
function ex32_shareLinks() {
  const ids = ['PASTE_FILE_ID_1','PASTE_FILE_ID_2'];
  ids.forEach(id=>{
    const f = DriveApp.getFileById(id);
    f.setSharing(DriveApp.Access.ANYONE_WITH_LINK,
DriveApp.Permission.VIEW);
    Logger.log(f.getUrl());
  });
}
```

33) Export Google Doc as PDF into a folder

```
function ex33_exportDocPdfToFolder() {
  const docId='PASTE_DOC_ID', folderId='PASTE_FOLDER_ID';
  const file=DriveApp.getFileById(docId);
  const
pdf=file.getBlob().getAs(MimeType.PDF).setName(file.getName()+'.pdf');
  DriveApp.getFolderById(folderId).createFile(pdf);
}
```

34) Build a Drive inventory sheet (name, type, size, url)

```
function ex34_driveInventoryToSheet() {
  const folderId='PASTE_FOLDER_ID';
  const sh=SpreadsheetApp.getActiveSheet();
  const out=[['Name','MimeType','Size','URL','Last Updated']];
  const files=DriveApp.getFolderById(folderId).getFiles();
  while(files.hasNext()){
    const f=files.next();
    out.push([f.getName(), f.getMimeType(), f.getSize(), f.getUrl(),
f.getLastUpdated()]);
  }
  sh.clear();
  sh.getRange(1,1,out.length,out[0].length).setValues(out);
```

```
}
```

35) Find duplicate files by name in a folder

```
function ex35_duplicateNamesInFolder() {
  const folderId='PASTE_FOLDER_ID';
  const files=DriveApp.getFolderById(folderId).getFiles();
  const map = {};
  while(files.hasNext()){
    const f=files.next();
    (map[f.getName()] ||= []).push(f.getId());
  }
  const dupes = Object.entries(map).filter(([,ids])=>ids.length>1);
  Logger.log(dupes);
}
```

36) Trash files matching a pattern (careful!)

```
function ex36_trashMatching() {
  const folderId='PASTE_FOLDER_ID';
  const pattern=/^tmp_/i;
  const files=DriveApp.getFolderById(folderId).getFiles();
  while(files.hasNext()){
    const f=files.next();
    if(pattern.test(f.getName())) f.setTrashed(true);
  }
}
```

37) Copy a file and keep it in the same folder

```
function ex37_copyInPlace() {
  const fileId='PASTE_FILE_ID';
  const f=DriveApp.getFileById(fileId);
  const parents=f.getParents();
  if(!parents.hasNext()) throw new Error('No parent folder.');
  const folder=parents.next();
  const copy=f.makeCopy(f.getName() + ' (Copy)', folder);
  Logger.log(copy.getUrl());
}
```

38) Create “safe” unique filenames

```
function ex38_uniqueName(baseName, folderId) {
  const folder = DriveApp.getFolderById(folderId);
  let name = baseName, i=1;
  while (folder.getFilesByName(name).hasNext()) name = `${baseName}
(${i++})`;
  return name;
}
```

39) Bulk convert XLSX to Google Sheets (Drive API recommended)

Needs: Advanced Drive Service (`Drive`) enabled.

```
function ex39_convertXlsxToSheet() {
  const fileId='PASTE_XLSX_FILE_ID';
  const resource = { title: 'Converted Sheet', mimeType:
MimeType.GOOGLE_SHEETS };
  const newFile = Drive.Files.copy(resource, fileId);
  Logger.log(newFile.alternateLink);
}
```

40) Set file description + star it

```
function ex40_fileMeta() {
  const fileId='PASTE_FILE_ID';
  const f=DriveApp.getFileById(fileId);
  f.setDescription('Updated by Apps Script on ' + new Date());
  f.setStarred(true);
}
```

41) Find large files (>50MB) in Drive root

```
function ex41_largeFiles() {
  const it = DriveApp.searchFiles('trashed=false and "root" in parents
and quotaBytesUsed > 50000000');
  while(it.hasNext()) {
```

```
    const f=it.next();
    Logger.log(`${f.getName()} ${f.getSize()} ${f.getUrl()}`);
  }
}
```

42) Build a folder tree report (recursive)

```
function ex42_folderTree() {
  const rootId='PASTE_FOLDER_ID';
  const root=DriveApp.getFolderById(rootId);
  printTree_(root, 0);
}
function printTree_(folder, depth) {
  Logger.log('  '.repeat(depth) + folder.getName());
  const sub=folder.getFolders();
  while(sub.hasNext()) printTree_(sub.next(), depth+1);
}
```

43) Replace a file in a folder (version-style)

```
function ex43_replaceFileInFolder() {
  const folderId='PASTE_FOLDER_ID';
  const folder=DriveApp.getFolderById(folderId);
  const name='current-report.pdf';
  const existing=folder.getFilesByName(name);
  while(existing.hasNext()) existing.next().setTrashed(true);
  folder.createFile(Utilities.newBlob('new
content','text/plain',name));
}
```

44) Create a public “download” link (view only)

```
function ex44_publicLink() {
  const fileId='PASTE_FILE_ID';
  const f=DriveApp.getFileById(fileId);
  f.setSharing(DriveApp.Access.ANYONE_WITH_LINK,
DriveApp.Permission.VIEW);
  Logger.log(f.getUrl());
}
```

45) Drive permissions audit (requires Drive API)

Needs: Advanced Drive Service (`Drive`)

```
function ex45_permissionsAudit() {
  const fileId='PASTE_FILE_ID';
  const perms = Drive.Permissions.list(fileId).items || [];
  perms.forEach(p=>Logger.log(`${p.role} ${p.type} ${p.emailAddress ||
''}`));
}
```

C) Docs: Template Systems, Cleanup, Generation (46–65)

46) Generate a report doc from sheet rows

```
function ex46_sheetToDocReport() {
  const sh=SpreadsheetApp.getActiveSheet();
  const data=sh.getDataRange().getValues();
  const doc=DocumentApp.create('Sheet Report');
  const body=doc.getBody();

body.appendParagraph('Report').setHeading(DocumentApp.ParagraphHeading
.HEADING1);
  data.slice(1).forEach(r=> body.appendParagraph(`• ${r[0]} —
${r[1]}`));
  Logger.log(doc.getUrl());
}
```

47) Replace placeholders in a Doc (multiple fields)

```
function ex47_replacePlaceholders() {
  const doc=DocumentApp.getActiveDocument();
  const body=doc.getBody();
  const map = { '{{NAME}}':'Lars', '{{DATE}}': new
Date().toDateString() };
```

```
  Object.entries(map).forEach(([k,v])=>body.replaceText(k,
String(v)));
}
```

48) Insert table from array

```
function ex48_insertTable() {
  const doc=DocumentApp.getActiveDocument();
  const body=doc.getBody();
  const rows=[['Name','Score'],['Ava','95'],['Noah','88']];
  body.appendTable(rows);
}
```

49) Convert selected text to heading styles

```
function ex49_styleHeadings() {
  const doc=DocumentApp.getActiveDocument();
  const pars=doc.getBody().getParagraphs();
  pars.forEach(p=>{
    const t=p.getText();
    if(t.startsWith('## ')) {
p.setHeading(DocumentApp.ParagraphHeading.HEADING2);
p.setText(t.replace(/^##\s+/,''));
    } else if(t.startsWith('# ')) {
p.setHeading(DocumentApp.ParagraphHeading.HEADING1);
p.setText(t.replace(/^#\s+/,''));
    }
  });
}
```

50) Remove double blank lines

```
function ex50_removeDoubleBlankLines() {
  const body=DocumentApp.getActiveDocument().getBody();
  const paras=body.getParagraphs();
  for(let i=paras.length-2;i>=0;i--){
    if(!paras[i].getText().trim() && !paras[i+1].getText().trim())
paras[i+1].removeFromParent();
  }
```

```
}
```

51) Normalize bullet indentation (flatten)

```
function ex51_flattenBullets() {
  const body=DocumentApp.getActiveDocument().getBody();
  const paras=body.getParagraphs();
  paras.forEach(p=>{
    const li=p.editAsText(); // safe call; indentation set on
paragraph
    if (p.getListId()) p.setIndentStart(0).setIndentFirstLine(0);
  });
}
```

52) Insert page breaks before headings

```
function ex52_pageBreakBeforeH1() {
  const body=DocumentApp.getActiveDocument().getBody();
  const pars=body.getParagraphs();
  for(let i=1;i<pars.length;i++){
    if(pars[i].getHeading()===DocumentApp.ParagraphHeading.HEADING1){
      body.insertParagraph(body.getChildIndex(pars[i]),
'').appendPageBreak();
    }
  }
}
```

53) Build a Table of Contents automatically

```
function ex53_addTOC() {
  const body=DocumentApp.getActiveDocument().getBody();
  body.insertTableOfContents(0);
}
```

54) Convert markdown-style bullets to real bullets

```
function ex54_markdownBulletsToList() {
  const doc=DocumentApp.getActiveDocument();
  const body=doc.getBody();
```

```
  const pars=body.getParagraphs();
  pars.forEach(p=>{
    const t=p.getText();
    if(/^-\s+/.test(t)) { p.setText(t.replace(/^-+\s+/,''));
p.setBullet(true); }
  });
}
```

55) Add footer with date + page number

```
function ex55_footer() {
  const doc=DocumentApp.getActiveDocument();
  const footer=doc.addFooter();
  footer.appendParagraph('Generated: ' + new Date().toDateString());
}
```

56) Merge multiple docs into one

```
function ex56_mergeDocs() {
  const target=DocumentApp.create('Merged Doc');
  const body=target.getBody();
  const ids=['PASTE_DOC_ID_1','PASTE_DOC_ID_2'];
  ids.forEach(id=>{
    const src=DocumentApp.openById(id).getBody();
    src.getParagraphs().forEach(p=>
body.appendParagraph(p.getText()));
    body.appendPageBreak();
  });
  Logger.log(target.getUrl());
}
```

57) Extract all links from a doc

```
function ex57_extractLinks() {
  const body=DocumentApp.getActiveDocument().getBody();
  const found=[];
  body.getParagraphs().forEach(p=>{
    const text=p.editAsText();
    for(let i=0;i<text.getText().length;i++){
```

```
      const url=text.getLinkUrl(i);
      if(url) found.push(url);
    }
  });
  Logger.log([...new Set(found)]);
}
```

58) Replace smart quotes with straight quotes

```
function ex58_normalizeQuotes() {
  const body=DocumentApp.getActiveDocument().getBody();
  const t=body.editAsText();
  t.replaceText('[“”]', '"');
  t.replaceText('[‘’]', "'");
}
```

59) Highlight placeholder tokens like {{...}}

```
function ex59_highlightPlaceholders() {
  const body=DocumentApp.getActiveDocument().getBody().editAsText();
  const text=body.getText();
  const re=/\{\{[^}]+\}\}/g;
  let m;
  while((m=re.exec(text))){
    body.setBackgroundColor(m.index, m.index+m[0].length-1,
'#fff2cc');
  }
}
```

60) Create a “chapter” doc per heading

```
function ex60_splitDocByH1() {
  const doc=DocumentApp.getActiveDocument();
  const pars=doc.getBody().getParagraphs();
  let current=null;

  pars.forEach(p=>{
    if(p.getHeading()===DocumentApp.ParagraphHeading.HEADING1){
      current=DocumentApp.create(p.getText());
```

```
current.getBody().appendParagraph(p.getText()).setHeading(DocumentApp.
ParagraphHeading.HEADING1);
    } else if(current) {
      current.getBody().appendParagraph(p.getText());
    }
  });
}
```

61) Add code block styling (monospace + background)

```
function ex61_styleCodeBlocks() {
  const body=DocumentApp.getActiveDocument().getBody();
  body.getParagraphs().forEach(p=>{
    const t=p.getText();
    if(t.startsWith('```') || t.startsWith('    ')){
      p.setBackgroundColor('#f3f4f6');
      p.setFontFamily('Courier New');
    }
  });
}
```

62) Replace multiple whitespace with single spaces

```
function ex62_collapseSpaces() {
  const t=DocumentApp.getActiveDocument().getBody().editAsText();
  t.replaceText('\\s{2,}', ' ');
}
```

63) Generate a doc from JSON data

```
function ex63_docFromJson() {
  const data = { title:'Status',
items:[{name:'Ava',score:95},{name:'Noah',score:88}] };
  const doc = DocumentApp.create(data.title);
  const body = doc.getBody();

body.appendParagraph(data.title).setHeading(DocumentApp.ParagraphHeadi
ng.HEADING1);
```

```
  data.items.forEach(i=> body.appendParagraph(`${i.name}:
${i.score}`));
  Logger.log(doc.getUrl());
}
```

64) Mail-merge to PDFs (Doc template + sheet rows)

```
function ex64_mailMergeToPdfs() {
  const templateId='PASTE_TEMPLATE_DOC_ID';
  const folderId='PASTE_OUTPUT_FOLDER_ID';
  const sh=SpreadsheetApp.getActiveSheet();
  const rows=sh.getDataRange().getValues();
  const header=rows.shift();

  rows.forEach(r=>{
    const map={}; header.forEach((h,i)=>map[`{{${h}}}`]=String(r[i] ??
''));
    const copy=DriveApp.getFileById(templateId).makeCopy('Letter - ' +
(r[0]||'Recipient'));
    const doc=DocumentApp.openById(copy.getId());
    const body=doc.getBody();
    Object.entries(map).forEach(([k,v])=>body.replaceText(k,v));
    doc.saveAndClose();

    const
pdf=DriveApp.getFileById(copy.getId()).getBlob().getAs(MimeType.PDF).s
etName(copy.getName()+'.pdf');
    DriveApp.getFolderById(folderId).createFile(pdf);
    copy.setTrashed(true);
  });
}
```

65) Convert a doc to plain text and save to Drive

```
function ex65_docToTxt() {
  const doc=DocumentApp.getActiveDocument();
  const text=doc.getBody().getText();
  const file=DriveApp.createFile(doc.getName()+'.txt', text,
MimeType.PLAIN_TEXT);
```

```
  Logger.log(file.getUrl());
}
```

D) Gmail: Search, Threads, Automation (66–80)

66) Advanced Gmail search + summarize subjects

```
function ex66_gmailSearch() {
  const threads = GmailApp.search('newer_than:7d in:inbox', 0, 20);
  threads.forEach(t=>Logger.log(t.getFirstMessageSubject()));
}
```

67) Label messages by rule (e.g., invoices)

```
function ex67_labelInvoices() {
  const label = GmailApp.getUserLabelByName('Invoices') ||
GmailApp.createLabel('Invoices');
  const threads = GmailApp.search('subject:(invoice OR receipt)
newer_than:30d');
  threads.forEach(t=>label.addToThread(t));
}
```

68) Auto-archive newsletters (by sender)

```
function ex68_archiveSender() {
  const threads = GmailApp.search('from:newsletter@example.com
in:inbox');
  threads.forEach(t=>t.moveToArchive());
}
```

69) Send an email with HTML + inline image

```
function ex69_emailInlineImage() {
  const blob =
UrlFetchApp.fetch('https://www.google.com/images/branding/googlelogo/1
x/googlelogo_color_272x92dp.png').getBlob();
```

```
  GmailApp.sendEmail('someone@example.com','HTML Email','Your client
does not support HTML.',{
    htmlBody:'<h2>Hello</h2><p>Inline image:</p><img
src="cid:logo"/>',
    inlineImages:{ logo: blob }
  });
}
```

70) Daily digest: send yourself a summary of matching emails

```
function ex70_dailyDigest() {
  const threads = GmailApp.search('newer_than:1d in:inbox', 0, 50);
  const lines = threads.map(t=>'- ' +
t.getFirstMessageSubject()).join('\n');
  GmailApp.sendEmail(Session.getActiveUser().getEmail(), 'Daily
Digest', lines || 'No messages found.');
}
```

71) Save attachments to Drive folder

```
function ex71_saveAttachments() {
  const folderId='PASTE_FOLDER_ID';
  const folder=DriveApp.getFolderById(folderId);
  const threads=GmailApp.search('has:attachment newer_than:7d',0,20);
  threads.forEach(t=>{
    t.getMessages().forEach(m=>{
      m.getAttachments().forEach(a=>folder.createFile(a));
    });
  });
}
```

72) Convert CSV attachments to Sheets (Drive API helpful)

```
function ex72_csvAttachmentToSheet() {
  const threads=GmailApp.search('filename:csv newer_than:7d',0,5);
  if(!threads.length) return;
  const
att=threads[0].getMessages()[0].getAttachments().find(a=>a.getContentT
ype().includes('csv'));
```

```
  const csv=att.getDataAsString();
  const rows=Utilities.parseCsv(csv);
  const ss=SpreadsheetApp.create('Imported CSV ' + new
Date().toISOString());

ss.getSheets()[0].getRange(1,1,rows.length,rows[0].length).setValues(r
ows);
  Logger.log(ss.getUrl());
}
```

73) Auto-reply to specific threads (basic)

```
function ex73_autoReply() {
  const threads=GmailApp.search('subject:"Request Info"
newer_than:1d',0,10);
  threads.forEach(t=>{
    const msg=t.getMessages().pop();
    msg.reply('Thanks! We received your request and will respond
soon.');
  });
}
```

74) Unsubscribe helper: list emails with “unsubscribe” in body

```
function ex74_findUnsubscribe() {
  const threads=GmailApp.search('newer_than:30d "unsubscribe"',0,50);
  threads.forEach(t=>Logger.log(t.getFirstMessageSubject()));
}
```

75) Extract sender domains frequency (top 20)

```
function ex75_senderDomains() {
  const threads=GmailApp.search('newer_than:14d',0,100);
  const freq={};
  threads.forEach(t=>{
    const from=t.getMessages()[0].getFrom();
    const email=(from.match(/<(.+?)>/)?.[1]||from).trim();
    const domain=email.split('@')[1]||'';
    if(domain) freq[domain]=(freq[domain]||0)+1;
```

```
  });
  const top=Object.entries(freq).sort((a,b)=>b[1]-a[1]).slice(0,20);
  Logger.log(top);
}
```

76) Create a “Support” label + apply to unread from specific alias

```
function ex76_labelSupportAlias() {
  const
label=GmailApp.getUserLabelByName('Support')||GmailApp.createLabel('Su
pport');
  GmailApp.search('to:support@yourdomain.com
is:unread',0,50).forEach(t=>label.addToThread(t));
}
```

77) Send email from a draft template (replace tokens)

```
function ex77_sendFromDraft() {
  const subject='TEMPLATE: Welcome';
  const
drafts=GmailApp.getDrafts().filter(d=>d.getMessage().getSubject()===su
bject);
  if(!drafts.length) throw new Error('Draft not found: ' + subject);
  const msg=drafts[0].getMessage();
  const html=msg.getBody().replace('{{NAME}}','Friend');
  GmailApp.sendEmail('someone@example.com','Welcome!','', { htmlBody:
html });
}
```

78) Clean up old labels (danger: removes labels!)

```
function ex78_deleteEmptyLabels() {
  const labels=GmailApp.getUserLabels();
  labels.forEach(l=>{
    if(l.getThreads(0,1).length===0 && l.getUnreadCount()===0) {
      // l.deleteLabel(); // uncomment only if you really want it
      Logger.log('Empty label: ' + l.getName());
    }
  });
```

```
}
```

79) Thread summary to Google Doc

```
function ex79_threadToDoc() {
  const thread=GmailApp.search('newer_than:7d',0,1)[0];
  if(!thread) return;
  const doc=DocumentApp.create('Email Thread Summary');
  const body=doc.getBody();
  thread.getMessages().forEach(m=>{

body.appendParagraph(m.getFrom()).setHeading(DocumentApp.ParagraphHead
ing.HEADING2);
    body.appendParagraph(m.getDate().toString());
    body.appendParagraph(m.getPlainBody().slice(0,1000));
    body.appendHorizontalRule();
  });
  Logger.log(doc.getUrl());
}
```

80) Schedule a daily Gmail cleanup trigger

```
function ex80_createDailyTrigger() {

ScriptApp.newTrigger('ex68_archiveSender').timeBased().everyDays(1).at
Hour(7).create();
}
```

E) Calendar, Forms, Web Apps, Advanced Services (81–100)

81) Create events from sheet rows

```
function ex81_sheetToCalendar() {
  const cal=CalendarApp.getDefaultCalendar();
  const sh=SpreadsheetApp.getActiveSheet();
```

```
  const rows=sh.getRange(2,1,sh.getLastRow()-1,3).getValues(); //
Title, Start, End
  rows.forEach(r=>{
    const [title,start,end]=r;
    if(title && start && end) cal.createEvent(String(title), new
Date(start), new Date(end));
  });
}
```

82) Avoid duplicates: only create event if not already there

```
function ex82_calendarNoDuplicates() {
  const cal=CalendarApp.getDefaultCalendar();
  const title='Standup';
  const start=new Date(); const end=new
Date(start.getTime()+30*60000);
  const existing=cal.getEvents(start,end,{search:title});
  if(existing.length===0) cal.createEvent(title,start,end);
}
```

83) Add guests + conference (Meet link)

```
function ex83_eventWithGuests() {
  const cal=CalendarApp.getDefaultCalendar();
  const start=new Date(Date.now()+3600*1000), end=new
Date(start.getTime()+3600*1000);
  const
ev=cal.createEvent('Planning',start,end,{guests:'a@example.com,b@examp
le.com'});
  ev.addEmailReminder(30);
}
```

84) Form submit trigger → write to “processed” sheet

```
function ex84_onFormSubmit(e) {
  const ss=SpreadsheetApp.getActive();
  const
sh=ss.getSheetByName('Processed')||ss.insertSheet('Processed');
  sh.appendRow([new Date(), JSON.stringify(e.namedValues)]);
```

```
}
```

85) Web app: JSON API with API key check

```
function doGet(e) {
  const key = e.parameter.key;
  const expected =
PropertiesService.getScriptProperties().getProperty('API_KEY');
  if (!expected || key !== expected) {
    return
ContentService.createTextOutput(JSON.stringify({ok:false,error:'unauth
orized'}))
      .setMimeType(ContentService.MimeType.JSON);
  }
  return
ContentService.createTextOutput(JSON.stringify({ok:true,time:new
Date().toISOString()}))
    .setMimeType(ContentService.MimeType.JSON);
}
```

86) Web app: write POST body to sheet

```
function doPost(e) {
  const ss=SpreadsheetApp.openById('PASTE_SHEET_ID');
  const
sh=ss.getSheetByName('WebAppLog')||ss.insertSheet('WebAppLog');
  sh.appendRow([new Date(), e.postData.type, e.postData.contents]);
  return ContentService.createTextOutput('OK');
}
```

87) OAuth2 to external APIs (pattern)

Does: Apps Script needs OAuth2 library; this is the setup pattern.

```
function ex87_oauth2Pattern_note() {
  Logger.log('Use the OAuth2 library:
https://github.com/googleworkspace/apps-script-oauth2');
}
```

88) Use Advanced Drive API to create file in shared drive folder

Needs: Advanced Drive Service (`Drive`)

```
function ex88_driveApiCreateInFolder() {
  const folderId='PASTE_FOLDER_ID';
  const resource={ title:'hello.txt', mimeType:'text/plain',
parents:[{id:folderId}] };
  const file=Drive.Files.insert(resource, Utilities.newBlob('Hello'));
  Logger.log(file.alternateLink);
}
```

89) Admin SDK: list users (domain only)

Needs: Advanced Admin Directory Service (`AdminDirectory`) + admin rights.

```
function ex89_listUsers() {
  const users = AdminDirectory.Users.list({customer:'my_customer',
maxResults: 10}).users || [];
  users.forEach(u=>Logger.log(u.primaryEmail));
}
```

90) BigQuery job from Apps Script (advanced)

Needs: Advanced BigQuery service enabled.

```
function ex90_bigqueryQuery() {
  const projectId='PASTE_PROJECT_ID';
  const request={ query:'SELECT CURRENT_TIMESTAMP() AS now',
useLegacySql:false };
  const res=BigQuery.Jobs.query(request, projectId);
  Logger.log(JSON.stringify(res.rows));
}
```

91) Pub/Sub publish (advanced)

Needs: Advanced Pub/Sub service (or UrlFetch with token).

```
function ex91_pubsubNote() {
```

```
  Logger.log('Pub/Sub from Apps Script typically uses UrlFetch + OAuth
token; enable API in GCP.');
}
```

92) Create Slides deck from sheet data

```
function ex92_slidesFromSheet() {
  const sh=SpreadsheetApp.getActiveSheet();
  const
rows=sh.getRange(2,1,Math.max(1,sh.getLastRow()-1),2).getValues(); //
title, bullet
  const pres=SlidesApp.create('Auto Deck');
  rows.forEach(r=>{
    const [title,bullet]=r;
    const
slide=pres.appendSlide(SlidesApp.PredefinedLayout.TITLE_AND_BODY);

slide.getPlaceholder(SlidesApp.PlaceholderType.TITLE).asShape().getTex
t().setText(String(title||''));

slide.getPlaceholder(SlidesApp.PlaceholderType.BODY).asShape().getText
().setText('• ' + String(bullet||''));
  });
  pres.getSlides()[0].remove(); // remove default title slide
  Logger.log(pres.getUrl());
}
```

93) Generate PDF from Slides and email it

```
function ex93_emailSlidesPdf() {
  const presId='PASTE_SLIDES_ID';
  const file=DriveApp.getFileById(presId);
  const
pdf=file.getBlob().getAs(MimeType.PDF).setName(file.getName()+'.pdf');
  GmailApp.sendEmail('someone@example.com','Slides
PDF','Attached.',{attachments:[pdf]});
}
```

94) HTMLService UI + server call (`google.script.run`)

```
function ex94_showUi() {
  const html = HtmlService.createHtmlOutput(`
    <button onclick="go()">Run</button>
    <pre id="out"></pre>
    <script>
      function go(){

google.script.run.withSuccessHandler(r=>out.textContent=r).serverHello
();
      }
      const out=document.getElementById('out');
    </script>
  `).setWidth(300).setHeight(200);
  SpreadsheetApp.getUi().showModalDialog(html,'UI Demo');
}
function serverHello(){ return 'Hello from server at ' + new
Date().toISOString(); }
```

95) HTML template file pattern

```
function ex95_templateHtml() {
  const t = HtmlService.createTemplate('<h2>Hello <?= name ?></h2>');
  t.name = 'Lars';
  SpreadsheetApp.getUi().showSidebar(t.evaluate().setTitle('Template
Demo'));
}
```

96) Robust error wrapper with logging + email alert

```
function ex96_safeRun() {
  try {
    risky_();
  } catch (err) {
    Logger.log(err.stack || err);
    GmailApp.sendEmail(Session.getActiveUser().getEmail(),'Script
error', String(err.stack || err));
    throw err;
```

```
  }
}
function risky_(){ throw new Error('Boom'); }
```

97) Feature flags via PropertiesService

```
function ex97_featureFlag() {
  const props=PropertiesService.getScriptProperties();
  const enabled = props.getProperty('FEATURE_X') === 'true';
  Logger.log('FEATURE_X enabled? ' + enabled);
}
```

98) Rate-limited worker (sleep between API calls)

```
function ex98_rateLimitedCalls() {
  const urls=['https://example.com','https://example.com'];
  urls.forEach((u,i)=>{
    const res=UrlFetchApp.fetch(u,{muteHttpExceptions:true});
    Logger.log(i + ' ' + res.getResponseCode());
    Utilities.sleep(500); // throttle
  });
}
```

99) Centralized config loader (sheet → object)

```
function ex99_loadConfigFromSheet() {
  const ss=SpreadsheetApp.getActive();
  const sh=ss.getSheetByName('Config');
  if(!sh) throw new Error('Missing Config sheet');
  const rows=sh.getRange(2,1,sh.getLastRow()-1,2).getValues();
  const cfg={};
  rows.forEach(([k,v])=>{ if(k) cfg[String(k).trim()] = String(v ??
'').trim(); });
  Logger.log(cfg);
  return cfg;
}
```

100) End-to-end: fetch → store → report → email

Does: Fetches JSON, writes to sheet, generates summary, emails results.

```
function ex100_pipeline() {
  const ss=SpreadsheetApp.getActive();
  const sh=ss.getSheetByName('API Data')||ss.insertSheet('API Data');
  const
json=JSON.parse(UrlFetchApp.fetch('https://httpbin.org/json').getConte
ntText());

  const rows=Object.entries(json).map(([k,v])=>[k, typeof
v==='object'?JSON.stringify(v):v]);
  sh.clear();
  sh.getRange(1,1,rows.length,2).setValues(rows);

  const summary = `Loaded ${rows.length} keys into "${sh.getName()}"
at ${new Date().toISOString()}`;
  GmailApp.sendEmail(Session.getActiveUser().getEmail(),'Pipeline
Complete', summary);
}
```
