100 high-impact, copy-pasteable Google Sheets Apps Script snippets

GitHub Source code
https://github.com/lsvekis/100-high-impact-copy-pasteable-Google-Sheets-Apps-Script-snippets

How to use

1. In your Sheet: **Extensions ▸ Apps Script**
2. Paste a script into `Code.gs` (ONE at a time to learn or as many as you want).
3. Save ▶️ **Run** (grant permissions when prompted) or use the custom menu where provided.

A. Setup & Quality-of-Life

1) Add a Custom Menu

Creates a “Sheet Tools” menu with quick actions.

```
function onOpen() {
  SpreadsheetApp.getUi()
    .createMenu('Sheet Tools')
    .addItem('Clear Selected Range', 'clearSelectedRange')
    .addItem('Timestamp Selected Cells', 'timestampSelection')
    .addToUi();
}

function clearSelectedRange() {
  const range = SpreadsheetApp.getActiveRange();
  if (range) range.clearContent();
}

function timestampSelection() {
```

```
  const range = SpreadsheetApp.getActiveRange();
  if (range) range.setValue(new Date());
}
```

What it does & why: Adds handy actions to the UI so common tasks are one click away—no more hunting for functions.

---

2) Create a New “Log” Sheet (no blank first row)

```
function ensureLogSheet() {
  const ss = SpreadsheetApp.getActive();
  let log = ss.getSheetByName('log');
  if (!log) {
    log = ss.insertSheet('log');
    log.getRange(1,1).setValue('Timestamp');
    log.getRange(1,2).setValue('Message');
  }
  return log;
}
```

What & why: Ensures a log exists and seeds headers immediately so there’s no empty first row.

---

3) Quick Logger

```
function logMessage(msg) {
  const log = ensureLogSheet();
  log.appendRow([new Date(), msg]);
}
```

What & why: Dead-simple logging for troubleshooting or lightweight audit trails.

---

4) Sheet Snapshot (duplicate active sheet with timestamp)

```
function snapshotActiveSheet() {
  const sheet = SpreadsheetApp.getActiveSheet();
  const name = sheet.getName() + ' Snapshot ' +
Utilities.formatDate(new Date(), Session.getScriptTimeZone(),
'yyyy-MM-dd HH:mm');
  sheet.copyTo(SpreadsheetApp.getActive()).setName(name);
}
```

What & why: One-click “backup” copies before big edits.

---

5) Protect a Sheet (except header row editable by all)

```
function protectSheetExceptHeader() {
  const sheet = SpreadsheetApp.getActiveSheet();
  const protection = sheet.protect().setDescription('Protected except
header');
  protection.removeEditors(protection.getEditors());
  const unprotectedRange =
sheet.getRange(1,1,1,sheet.getMaxColumns()); // header row
  protection.setUnprotectedRanges([unprotectedRange]);
}
```

What & why: Prevents accidental edits while keeping headers tweakable.

B. Data Entry & Cleanup

6) Add Timestamp on Edit (specific column)

Adds a timestamp in column B when column A is edited.

```
function onEdit(e) {
  const sheet = e.range.getSheet();
  if (sheet.getName() !== 'Sheet1') return;
  if (e.range.getColumn() === 1 && e.value !== undefined) {
    sheet.getRange(e.range.getRow(), 2).setValue(new Date());
  }
}
```

What & why: Auto-audit for entries—when something in A changes, B logs when.

---

7) Fill Down Last Row (copy formulas)

```
function fillDownLastRow() {
  const sh = SpreadsheetApp.getActiveSheet();
  const lastRow = sh.getLastRow();
  if (lastRow <= 1) return;
  const source = sh.getRange(lastRow - 1, 1, 1, sh.getLastColumn());
  const target = sh.getRange(lastRow, 1, 1, sh.getLastColumn());
  source.copyTo(target, {contentsOnly:false});
}
```

What & why: Keeps formulas alive for new rows without manual dragging.

---

8) Trim Spaces & Normalize Case in Selection

```
function cleanSelection() {
  const r = SpreadsheetApp.getActiveRange();
  const values = r.getValues().map(row =>
    row.map(v => (typeof v === 'string' ? v.trim().replace(/\s+/g,' ')
: v))
  );
  r.setValues(values);
```

```
}
```

What & why: Removes double spaces/stray whitespace that break lookups & matching.

---

9) Remove Duplicates (entire sheet, by first column)

```
function removeDuplicatesByFirstColumn() {
  const sh = SpreadsheetApp.getActiveSheet();
  const data = sh.getDataRange().getValues();
  const header = data.shift();
  const seen = new Set();
  const filtered = data.filter(r => {
    const key = r[0];
    if (seen.has(key)) return false;
    seen.add(key);
    return true;
  });
  sh.clearContents();
  sh.getRange(1,1,1,header.length).setValues([header]);
  if (filtered.length)
sh.getRange(2,1,filtered.length,header.length).setValues(filtered);
}
```

What & why: Dedupes dataset using column A as the unique key.

---

10) Find & Replace (in selection)

```
function findReplaceInSelection() {
  const ui = SpreadsheetApp.getUi();
  const resFind = ui.prompt('Find text', ui.ButtonSet.OK_CANCEL);
  if (resFind.getSelectedButton() !== ui.Button.OK) return;
  const resReplace = ui.prompt('Replace with',
ui.ButtonSet.OK_CANCEL);
```

```
  if (resReplace.getSelectedButton() !== ui.Button.OK) return;

  const find = resFind.getResponseText();
  const replace = resReplace.getResponseText();

  const rng = SpreadsheetApp.getActiveRange();
  const vals = rng.getValues().map(row => row.map(v =>
    (typeof v === 'string') ? v.split(find).join(replace) : v
  ));
  rng.setValues(vals);
}
```

What & why: Targeted, safe in-range replacements—no global disasters.

---

11) Split Full Name to First/Last (into adjacent columns)

```
function splitNames() {
  const r = SpreadsheetApp.getActiveRange();
  const vals = r.getValues().map(row => {
    const name = row[0] ? String(row[0]).trim() : '';
    const parts = name.split(/\s+/);
    const first = parts.shift() || '';
    const last = parts.join(' ');
    return [first, last];
  });
  r.offset(0,1,vals.length,2).setValues(vals);
}
```

What & why: Quick parsing of names into first/last for CRM imports.

C. Sorting, Filtering, Grouping

12) Sort by Column (A asc, then B desc)

```
function sortByAThenB() {
  const sh = SpreadsheetApp.getActiveSheet();
  const lr = sh.getLastRow();
  const lc = sh.getLastColumn();
  if (lr <= 1) return;
  const range = sh.getRange(2,1,lr-1,lc);
  range.sort([{column:1, ascending:true}, {column:2,
ascending:false}]);
}
```

What & why: Common multi-key sort you can tweak as needed.

---

13) Filter Rows (copy matches to a new sheet)

```
function filterToNewSheet() {
  const sh = SpreadsheetApp.getActiveSheet();
  const ui = SpreadsheetApp.getUi();
  const res = ui.prompt('Filter: keep rows where Column A equals…',
ui.ButtonSet.OK_CANCEL);
  if (res.getSelectedButton() !== ui.Button.OK) return;
  const match = res.getResponseText();

  const data = sh.getDataRange().getValues();
  const header = data.shift();
  const filtered = data.filter(r => String(r[0]) === match);

  const out = SpreadsheetApp.getActive().insertSheet('Filtered ' +
match);
  out.getRange(1,1,1,header.length).setValues([header]);
  if (filtered.length)
out.getRange(2,1,filtered.length,header.length).setValues(filtered);
}
```

What & why: Extracts a subset to its own sheet for sharing/analysis.

---

14) Group by Category & Sum (simple pivot in script)

```
function groupByFirstColumnSumSecond() {
  const sh = SpreadsheetApp.getActiveSheet();
  const rows = sh.getDataRange().getValues();
  const header = rows.shift();
  const map = new Map();
  rows.forEach(r => {
    const key = r[0];
    const val = Number(r[1]) || 0;
    map.set(key, (map.get(key) || 0) + val);
  });

  const out = SpreadsheetApp.getActive().insertSheet('Grouped
Summary');
  out.getRange(1,1,1,2).setValues([['Category','Sum']]);
  const arr = [...map.entries()];
  if (arr.length) out.getRange(2,1,arr.length,2).setValues(arr);
}
```

What & why: A quick script-pivot when you don’t want a full PivotTable.

D. Formatting & Validation

15) Apply Header Style (bold, frozen, background)

```
function styleHeader() {
  const sh = SpreadsheetApp.getActiveSheet();
  const lc = sh.getLastColumn();
  if (lc === 0) return;
  const header = sh.getRange(1,1,1,lc);
```

```
  header.setFontWeight('bold').setBackground('#f2f2f2');
  sh.setFrozenRows(1);
}
```

What & why: Professional, readable tables in one click.

---

16) Number Formats (currency & date for selection)

```
function formatSelectionCurrencyDate() {
  const r = SpreadsheetApp.getActiveRange();
  r.setNumberFormat('[$$-409]#,##0.00;[Red]-[$$-409]#,##0.00');
}
```

What & why: Standard currency formatting for clean reporting.

---

17) Conditional Formatting: Highlight Duplicates in Column A

```
function highlightDuplicatesColA() {
  const sh = SpreadsheetApp.getActiveSheet();
  const rules = sh.getConditionalFormatRules();
  const range = sh.getRange(2,1,sh.getMaxRows()-1,1);
  const rule = SpreadsheetApp.newConditionalFormatRule()
    .whenFormulaSatisfied('=COUNTIF($A:$A,$A2)>1')
    .setBackground('#ffcccc')
    .setRanges([range])
    .build();
  rules.push(rule);
  sh.setConditionalFormatRules(rules);
}
```

What & why: Visual duplicate finder that updates as you type.

---

18) Data Validation: Dropdown from List

```
function addDropdownValidation() {
  const sh = SpreadsheetApp.getActiveSheet();
  const r = SpreadsheetApp.getActiveRange();
  const rule = SpreadsheetApp.newDataValidation()
    .requireValueInList(['Open','In Progress','Done'], true)
    .setAllowInvalid(false)
    .build();
  r.setDataValidation(rule);
}
```

What & why: Enforces controlled inputs for status/tags.

E. Import/Export & Connectivity

19) Import JSON from a URL to a Sheet

```
function importJsonToSheet() {
  const url = 'https://example.com/api/items'; // change me
  const resp = UrlFetchApp.fetch(url, {muteHttpExceptions:true});
  const json = JSON.parse(resp.getContentText());
  const ss = SpreadsheetApp.getActive();
  const sh = ss.insertSheet('Import JSON');
  if (!Array.isArray(json) || json.length === 0) {
    sh.getRange(1,1).setValue('No data');
    return;
  }
  const headers = Object.keys(json[0]);
  sh.getRange(1,1,1,headers.length).setValues([headers]);
  const rows = json.map(o => headers.map(h => o[h]));
  sh.getRange(2,1,rows.length,headers.length).setValues(rows);
```

```
}
```

What & why: Pulls an API response into a nice tabular sheet.

---

20) Export Active Sheet as CSV (Drive file link)

```
function exportActiveSheetCSV() {
  const sh = SpreadsheetApp.getActiveSheet();
  const csv = sh.getDataRange().getValues()
    .map(row => row.map(v => {
      v = (v===null || v===undefined) ? '' : String(v);
      return /[",\n]/.test(v) ? `"${v.replace(/"/g,'""')}"` : v;
    }).join(','))
    .join('\n');

  const file = DriveApp.createFile(sh.getName() + '.csv', csv,
MimeType.CSV);
  SpreadsheetApp.getUi().alert('CSV created: ' + file.getUrl());
}
```

What & why: One-click CSV exporter without downloading manually.

---

21) Send Sheet as PDF via Email

```
function emailActiveSheetAsPdf() {
  const sh = SpreadsheetApp.getActiveSheet();
  const ss = sh.getParent();
  const url = Drive.Files.export(ss.getId(), 'application/pdf'); //
Advanced Drive API not required
  const blob = UrlFetchApp.fetch(

`https://docs.google.com/feeds/download/spreadsheets/Export?key=${ss.g
etId()}&exportFormat=pdf&gid=${sh.getSheetId()}`
```

```
  , {headers:{'Authorization':'Bearer ' +
ScriptApp.getOAuthToken()}}).getBlob()
   .setName(sh.getName() + '.pdf');

  MailApp.sendEmail({
    to: Session.getActiveUser().getEmail(),
    subject: 'Sheet PDF: ' + sh.getName(),
    body: 'Attached PDF of the active sheet.',
    attachments: [blob]
  });
}
```

What & why: Emails the current sheet as a PDF—good for approvals/reports.
(Note: Uses export endpoint; works without enabling Advanced services.)

F. Lookups, Ranges, and Names

22) Create Named Range for Selection

```
function nameSelection() {
  const r = SpreadsheetApp.getActiveRange();
  const ui = SpreadsheetApp.getUi();
  const res = ui.prompt('Enter a name for this range:');
  if (res.getSelectedButton() !== ui.Button.OK) return;
  SpreadsheetApp.getActive().setNamedRange(res.getResponseText(), r);
}
```

What & why: Friendly naming for ranges used in formulas/scripts.

---

23) VLOOKUP via Script (write results)

```
function vlookupScript() {
```

```
  const sh = SpreadsheetApp.getActiveSheet();
  const keyCol = 1; // lookup from A
  const lookupTable = sh.getRange(1,5,sh.getLastRow(),2).getValues();
// E:F
  const map = new Map(lookupTable.map(r => [String(r[0]), r[1]]));

  const rng = sh.getRange(2, keyCol, sh.getLastRow()-1, 1);
  const vals = rng.getValues().map(r => [map.get(String(r[0])) ??
'']);
  rng.offset(0,1,vals.length,1).setValues(vals); // output to column B
}
```

What & why: Imperative vlookup—useful for snapshots or removing volatile formulas.

G. Sheet & Range Utilities

24) Delete Empty Rows (below last content)

```
function deleteEmptyRowsBelow() {
  const sh = SpreadsheetApp.getActiveSheet();
  const lastRow = sh.getLastRow();
  const max = sh.getMaxRows();
  if (max > lastRow) sh.deleteRows(lastRow + 1, max - lastRow);
}
```

What & why: Cleans up scrollbars/blank rows for tidy sheets.

---

25) Resize Columns to Fit

```
function autoResizeAllColumns() {
  const sh = SpreadsheetApp.getActiveSheet();
  const lc = sh.getLastColumn();
```

```
  for (let c = 1; c <= lc; c++) sh.autoResizeColumn(c);
}
```

What & why: Presentation-ready columns after import/cleanup.

H. Forms & Responses

26) Append Google Form Responses to a “Master” Sheet

```
function onFormSubmit(e) {
  // Installable trigger: From form-linked sheet ▸ Triggers ▸
onFormSubmit
  const ss = SpreadsheetApp.getActive();
  const master = ss.getSheetByName('Master') ||
ss.insertSheet('Master');
  const row = e.values; // includes timestamp
  master.appendRow(row);
}
```

What & why: Keeps a consolidated “Master” even if you add multiple form tabs later.

---

27) Auto-Assign Incrementing Ticket IDs on New Response

```
function onFormSubmitTicketId(e) {
  const sheet = e.range.getSheet(); // the responses sheet
  const row = e.range.getRow();
  const idCol = 2; // e.g., put Ticket ID in column B
  const lastId = Number(sheet.getRange(2, idCol, sheet.getLastRow()-1,
1).getValues()
    .map(r => r[0]).filter(Boolean).sort((a,b)=>b-a)[0]) || 1000;
  sheet.getRange(row, idCol).setValue(lastId + 1);
}
```

What & why: Generates unique IDs without formulas.

---

28) Auto-Email on Form Response (simple)

```
function onFormSubmitEmail(e) {
  const values = e.namedValues; // { "Email": ["x@y.com"], "Message":
["..."] }
  const to = (values['Email'] ||
[Session.getActiveUser().getEmail()])[0];
  const msg = (values['Message'] || ['(no message)'])[0];
  MailApp.sendEmail(to, 'Thanks for your submission', 'We received: '
+ msg);
}
```

What & why: Sends an acknowledgement—great UX and instant feedback.

---

29) Validate Required Columns After Response

```
function onFormSubmitValidate(e) {
  const sh = e.range.getSheet();
  const row = sh.getRange(e.range.getRow(), 1, 1,
sh.getLastColumn()).getValues()[0];
  const requiredCols = [2,3]; // B and C must not be blank
  const missing = requiredCols.filter(c => !row[c-1]);
  if (missing.length) {
    // flag the row in column A
    sh.getRange(e.range.getRow(), 1).setNote('Missing required fields
in columns: ' + missing.join(','));
  }
}
```

What & why: Flags incomplete responses for follow-up.

---

30) Copy Important Fields from Form Sheet to a Clean “Intake” Sheet

```
function onFormSubmitIntake(e) {
  const src = e.range.getSheet();
  const intake = src.getParent().getSheetByName('Intake') ||
src.getParent().insertSheet('Intake');
  const row = e.range.getRow();
  const vals =
src.getRange(row,1,1,src.getLastColumn()).getValues()[0];
  // Map columns: [Timestamp(A), Name(C), Email(D), Priority(F)]
  const out = [vals[0], vals[2], vals[3], vals[5]];
  intake.appendRow(out);
}
```

What & why: Keeps a human-friendly intake list without clutter.

I. Triggers & Automation

31) Create a Time-Based Trigger (Daily 8am)

```
function createDaily8amTrigger() {
  ScriptApp.newTrigger('dailyJob')
    .timeBased().atHour(8).everyDays(1).create();
}

function dailyJob() {
  logMessage('Daily job ran at ' + new Date());
}
```

What & why: Schedules a daily routine—reports, cleanups, etc.

---

32) Clear a Range Weekly (Mondays)

```
function clearWeekly() {
  const sh = SpreadsheetApp.getActive().getSheetByName('Weekly');
  sh.getRange('B2:F1000').clearContent();
}
```

What & why: Resets a working area for the new week; pair with Monday trigger.

---

33) Send Weekly Digest Email (summary of counts)

```
function weeklyDigest() {
  const sh = SpreadsheetApp.getActive().getSheetByName('Tasks');
  const data = sh.getDataRange().getValues();
  const header = data.shift();
  const statusIdx = header.indexOf('Status');
  const counts = data.reduce((m, r) => (m[r[statusIdx]] =
(m[r[statusIdx]]||0)+1, m), {});
  MailApp.sendEmail(Session.getActiveUser().getEmail(), 'Weekly Task
Digest', JSON.stringify(counts, null, 2));
}
```

What & why: Lightweight snapshot to your inbox.

---

34) Archive Rows Older Than N Days

```
function archiveOldRows() {
  const DAYS = 30;
  const ss = SpreadsheetApp.getActive();
  const src = ss.getSheetByName('Live');
```

```
  const arc = ss.getSheetByName('Archive') ||
ss.insertSheet('Archive');

  const rows = src.getDataRange().getValues();
  const head = rows.shift();
  const now = Date.now();
  const keep = [head];
  const move = [];

  rows.forEach(r => {
    const ts = r[0] instanceof Date ? r[0].getTime() : null;
    if (ts && (now - ts) > DAYS*86400000) move.push(r);
    else keep.push(r);
  });

  if (move.length) {
    if (arc.getLastRow() === 0) arc.appendRow(head);

arc.getRange(arc.getLastRow()+1,1,move.length,head.length).setValues(m
ove);
    src.clearContents();
    src.getRange(1,1,keep.length,head.length).setValues(keep);
  }
}
```

What & why: Keeps the main sheet fast and tidy.

---

35) Rotate Log Sheet (start a fresh one monthly)

```
function rotateLogMonthly() {
  const ss = SpreadsheetApp.getActive();
  const old = ss.getSheetByName('log');
  if (!old) return;
  old.setName('log ' + Utilities.formatDate(new Date(),
Session.getScriptTimeZone(), 'yyyy-MM'));
  ss.insertSheet('log');
```

```
}
```

What & why: Prevents massive logs from getting unwieldy.

---

36) Auto-Format New Rows on Edit

```
function onEditAutoFormat(e) {
  const r = e.range;
  const sh = r.getSheet();
  const row = r.getRow();
  if (row === 1) return;
  sh.getRange(row,1,1,sh.getLastColumn())
    .setVerticalAlignment('middle')
    .setWrap(true);
}
```

What & why: Consistent look as data grows.

---

37) Auto-Insert Today's Date if Column A is Edited

```
function onEditDateStamp(e) {
  if (e.range.getColumn() === 1 && e.value !== undefined) {
    const sh = e.range.getSheet();
    sh.getRange(e.range.getRow(), 3).setValue(new Date()); // put in
column C
  }
}
```

What & why: Separate from timestamps for tracking "last touched" date.

---

38) Delete Triggers by Function Name

```
function deleteTriggersFor(fnName) {
  ScriptApp.getProjectTriggers()
    .filter(t => t.getHandlerFunction() === fnName)
    .forEach(ScriptApp.deleteTrigger);
}
```

What & why: Clean up old triggers to avoid duplicates.

J. Analytics & Charts

39) Build a Summary Sheet with KPIs

```
function buildKPIs() {
  const sh = SpreadsheetApp.getActive().getSheetByName('Data');
  const data = sh.getDataRange().getValues();
  const head = data.shift();
  const amountIdx = head.indexOf('Amount');
  const cnt = data.length;
  const sum = data.reduce((a,r)=>a+(Number(r[amountIdx])||0),0);
  const avg = cnt ? sum/cnt : 0;

  const out = SpreadsheetApp.getActive().getSheetByName('KPIs') ||
SpreadsheetApp.getActive().insertSheet('KPIs');
  out.clear();

out.getRange('A1:B3').setValues([['Metric','Value'],['Count',cnt],['To
tal',sum]]);
  out.getRange('A4:B4').setValues([['Average',avg]]);
}
```

What & why: One-click KPI snapshot.

---

40) Create a Chart Programmatically (Column Chart)

```
function makeColumnChart() {
  const sh = SpreadsheetApp.getActive().getSheetByName('KPIs');
  const chart = sh.newChart()
    .setChartType(Charts.ChartType.COLUMN)
    .addRange(sh.getRange('A2:B4'))
    .setPosition(1,4,0,0)
    .build();
  sh.insertChart(chart);
}
```

What & why: Script-built charts for dashboards.

---

41) Conditional Alert if KPI Threshold Breached

```
function kpiAlert() {
  const kpi = SpreadsheetApp.getActive().getSheetByName('KPIs');
  const total = Number(kpi.getRange('B3').getValue()) || 0;
  if (total < 1000) {
    MailApp.sendEmail(Session.getActiveUser().getEmail(),'KPI
Alert','Total fell below 1000: ' + total);
  }
}
```

What & why: Early warning system via email.

---

42) Rolling 7-Day Summary

```
function rolling7Day() {
  const dataSh = SpreadsheetApp.getActive().getSheetByName('Data');
  const rows = dataSh.getDataRange().getValues();
  const head = rows.shift();
```

```
  const dateIdx = head.indexOf('Date');
  const amtIdx = head.indexOf('Amount');
  const cutoff = Date.now() - 7*86400000;

  const sum = rows.reduce((a,r)=>{
    const d = r[dateIdx] instanceof Date ? r[dateIdx].getTime() : 0;
    if (d >= cutoff) a += Number(r[amtIdx])||0;
    return a;
  }, 0);

  const out = SpreadsheetApp.getActive().getSheetByName('Rolling') ||
SpreadsheetApp.getActive().insertSheet('Rolling');
  out.getRange('A1:B1').setValues([['Metric','Value']]);
  out.getRange('A2:B2').setValues([['Last 7 days total', sum]]);
}
```

What & why: Quick time-window analytics.

---

43) Top-N by Amount (write to new sheet)

```
function topN() {
  const N = 10;
  const sh = SpreadsheetApp.getActive().getSheetByName('Data');
  const data = sh.getDataRange().getValues();
  const head = data.shift();
  const amtIdx = head.indexOf('Amount');
  data.sort((a,b)=>(Number(b[amtIdx])||0)-(Number(a[amtIdx])||0));
  const out = SpreadsheetApp.getActive().getSheetByName('TopN') ||
SpreadsheetApp.getActive().insertSheet('TopN');
  out.clear();
  out.getRange(1,1,1,head.length).setValues([head]);

out.getRange(2,1,Math.min(N,data.length),head.length).setValues(data.s
lice(0,N));
}
```

What & why: Leaderboard report—sales, time, whatever metric.

---

44) Add Sparklines to Show Trends

```
function addSparklines() {
  const sh = SpreadsheetApp.getActiveSheet();
  const lr = sh.getLastRow();
  if (lr < 3) return;
  // Assume values in B2:B, put sparkline in C
  for (let r=2; r<=lr; r++) {
    sh.getRange(r,3).setFormula(`=SPARKLINE(B$2:B${r})`);
  }
}
```

What & why: Tiny trendlines for quick visual scanning.

---

45) Bucketize Values (e.g., “Low/Med/High”)

```
function bucketize() {
  const sh = SpreadsheetApp.getActiveSheet();
  const lr = sh.getLastRow();
  const amtCol = 2;
  const outCol = 3;
  const vals = sh.getRange(2,amtCol,lr-1,1).getValues().map(r=>{
    const v = Number(r[0])||0;
    return [v<50 ? 'Low' : v<200 ? 'Medium' : 'High'];
  });
  sh.getRange(2,outCol,vals.length,1).setValues(vals);
}
```

What & why: Categorize numeric data for easier filtering.

K. Collaboration & Notifications

46) Share Sheet with Editors by Email List

```
function shareWithEditors() {
  const emails = ['user1@example.com','user2@example.com']; //
customize
  const file =
DriveApp.getFileById(SpreadsheetApp.getActive().getId());
  emails.forEach(e => file.addEditor(e));
}
```

What & why: Bulk share without opening Drive UI.

---

47) Revoke Commenter Access for a Domain

```
function revokeCommentersByDomain() {
  const file =
DriveApp.getFileById(SpreadsheetApp.getActive().getId());
  file.getCommenters().forEach(u => {
    if (/@example\.com$/i.test(u.getEmail())) file.removeCommenter(u);
  });
}
```

What & why: Tighten permissions, especially after a project ends.

---

48) Email Me Changes to a Specific Range

```
function onEditNotifyRange(e) {
  const watch = e.range.getSheet().getRange('B2:D50');
  if (e.range.getRow() >= watch.getRow() && e.range.getRow() <=
watch.getLastRow()
```

```
      && e.range.getColumn() >= watch.getColumn() &&
e.range.getColumn() <= watch.getLastColumn()) {
    MailApp.sendEmail(Session.getActiveUser().getEmail(),
      'Change in watched range',
      `Cell ${e.range.getA1Notation()} changed to: ${e.value}`);
  }
}
```

What & why: Get alerts when critical cells change.

---

49) Add a Comment to a Cell Programmatically

```
function addCommentToCell() {
  const r = SpreadsheetApp.getActiveRange();
  r.setNote('Please verify this value. Updated on ' + new Date());
}
```

What & why: Lightweight “comment” using Notes from script.

---

50) Make a Review Checklist Sheet

```
function makeReviewChecklist() {
  const sh = SpreadsheetApp.getActive().insertSheet('Review
Checklist');

sh.getRange('A1:C1').setValues([['Task','Owner','Done?']]).setFontWeig
ht('bold');
  sh.getRange('A2:A10').setValues([['Data validated'],['Headers
styled'],['No duplicates'],['Formulas checked'],['Permissions
set'],['KPIs built'],['Chart updated'],['Archive done'],['Backup
taken'],['Notes added']]);
  const rule =
SpreadsheetApp.newDataValidation().requireCheckbox().build();
```

```
  sh.getRange('C2:C10').setDataValidation(rule);
}
```

What & why: Standardizes QA before sharing.

---

51) Notify Slack via Incoming Webhook

```
function notifySlack(text) {
  const url = 'https://hooks.slack.com/services/XXX/YYY/ZZZ'; //
replace
  const payload = JSON.stringify({ text });
  UrlFetchApp.fetch(url, {method: 'post',
contentType:'application/json', payload});
}
```

What & why: Push sheet events into your team's Slack.

---

52) Notify on New Top-5 Change

```
function notifyOnTop5Change() {
  const sh = SpreadsheetApp.getActive().getSheetByName('TopN');
  const curr = sh.getRange(2,1,5,1).getValues().flat().join('|');
  const props = PropertiesService.getScriptProperties();
  const last = props.getProperty('top5') || '';
  if (curr !== last) {
    props.setProperty('top5', curr);
    MailApp.sendEmail(Session.getActiveUser().getEmail(),'Top 5
changed', curr);
  }
}
```

What & why: Alerts when leaderboard meaningfully shifts.

L. Advanced Imports / Exports / Drive

53) Import All CSVs from a Drive Folder

```
function importCSVsFromFolder() {
  const folderId = 'YOUR_FOLDER_ID';
  const folder = DriveApp.getFolderById(folderId);
  const ss = SpreadsheetApp.getActive();
  const files = folder.getFilesByType(MimeType.CSV);
  while (files.hasNext()) {
    const f = files.next();
    const sh = ss.insertSheet(f.getName().replace(/\.csv$/i,''));
    const csv = Utilities.parseCsv(f.getBlob().getDataAsString());
    sh.getRange(1,1,csv.length,csv[0].length).setValues(csv);
  }
}
```

What & why: Bulk-import multiple CSVs into individual tabs.

---

54) Merge All Sheets into One “All Data”

```
function mergeAllSheets() {
  const ss = SpreadsheetApp.getActive();
  const out = ss.getSheetByName('All Data') || ss.insertSheet('All
Data');
  out.clear();
  let writeRow = 1;
  ss.getSheets().forEach(sh => {
    if (sh.getName() === 'All Data') return;
    const values = sh.getDataRange().getValues();
```

```
out.getRange(writeRow,1,values.length,values[0].length).setValues(valu
es);
    writeRow += values.length;
  });
}
```

What & why: One long table (useful for pivoting across tabs).

---

55) Export Every Sheet as its Own CSV File

```
function exportEachSheetAsCSV() {
  const ss = SpreadsheetApp.getActive();
  ss.getSheets().forEach(sh => {
    const data = sh.getDataRange().getValues()
      .map(row => row.map(v => {
        v = v==null?'':String(v);
        return /[",\n]/.test(v) ? `"${v.replace(/"/g,'""')}"` : v;
      }).join(',')).join('\n');
    DriveApp.createFile(`${sh.getName()}.csv`, data, MimeType.CSV);
  });
}
```

What & why: Batch export for downstream tools.

---

56) List Files in a Drive Folder into a Sheet

```
function listFolderFiles() {
  const folderId = 'YOUR_FOLDER_ID';
  const folder = DriveApp.getFolderById(folderId);
  const files = folder.getFiles();
  const sh = SpreadsheetApp.getActive().insertSheet('Folder List');
```

```
  sh.getRange('A1:C1').setValues([['Name','URL','Last
Updated']]).setFontWeight('bold');
  let r = 2;
  while (files.hasNext()) {
    const f = files.next();
    sh.getRange(r,1,1,3).setValues([[f.getName(), f.getUrl(),
f.getLastUpdated()]]);
    r++;
  }
}
```

What & why: Inventory your assets for audits.

---

57) Import Google Sheet Range from Another Spreadsheet

```
function importFromAnotherSpreadsheet() {
  const sourceId = 'SOURCE_SPREADSHEET_ID';
  const sourceRangeA1 = 'Sheet1!A1:D100';
  const values = SpreadsheetApp.openById(sourceId).getRangeByName
    ?
SpreadsheetApp.openById(sourceId).getRange(sourceRangeA1).getValues()
    :
SpreadsheetApp.openById(sourceId).getSheetByName('Sheet1').getRange('A
1:D100').getValues();
  const dest = SpreadsheetApp.getActive().getSheetByName('Imported')
|| SpreadsheetApp.getActive().insertSheet('Imported');
  dest.clear();
  dest.getRange(1,1,values.length,values[0].length).setValues(values);
}
```

What & why: Pull data from a different file on demand.

---

58) Export Active Sheet to New Spreadsheet

```
function exportActiveToNewSpreadsheet() {
  const sh = SpreadsheetApp.getActiveSheet();
  const temp = SpreadsheetApp.create(sh.getName() + ' Export');
  const dst = temp.getSheets()[0];
  dst.clear();
  const vals = sh.getDataRange().getValues();
  dst.getRange(1,1,vals.length,vals[0].length).setValues(vals);
  SpreadsheetApp.getUi().alert('Created: ' + temp.getUrl());
}
```

What & why: Share a single tab as its own file.

---

59) Snapshot to PDF in Drive (no email)

```
function saveActiveSheetAsPdf() {
  const ss = SpreadsheetApp.getActive();
  const sh = ss.getActiveSheet();
  const url =
`https://docs.google.com/spreadsheets/d/${ss.getId()}/export?format=pd
f&gid=${sh.getSheetId()}`;
  const blob = UrlFetchApp.fetch(url, {headers:
{'Authorization':'Bearer ' + ScriptApp.getOAuthToken()}}).getBlob()
    .setName(sh.getName() + '.pdf');
  DriveApp.createFile(blob);
}
```

What & why: Generate a PDF artifact for records.

---

60) Move Current Spreadsheet to a Specific Drive Folder

```
function moveSpreadsheetToFolder() {
```

```
  const folder = DriveApp.getFolderById('YOUR_FOLDER_ID');
  const file =
DriveApp.getFileById(SpreadsheetApp.getActive().getId());
  const parents = file.getParents();
  while (parents.hasNext()) file.removeFromFolder(parents.next());
  folder.addFile(file);
}
```

What & why: Organize files programmatically.

M. Errors & Audit

61) Try/Catch Wrapper with Logging

```
function safeRun() {
  try {
    // do risky things
    fillDownLastRow();
    logMessage('safeRun: success');
  } catch (err) {
    logMessage('safeRun ERROR: ' + err.stack);
    throw err;
  }
}
```

What & why: Captures stack traces in your `log` sheet.

---

62) Capture onEdit Errors Gracefully

```
function onEditSafe(e) {
  try {
    onEditDateStamp(e);
```

```
    onEditAutoFormat(e);
  } catch (err) {
    const sh = e.range.getSheet();
    sh.getRange(e.range.getRow(), e.range.getColumn()).setNote('Error:
' + err.message);
  }
}
```

What & why: Avoids breaking edits with unhandled exceptions.

---

63) Change Log: Who Edited What Cell

```
function onEditChangeLog(e) {
  const sh = e.range.getSheet();
  const user = Session.getActiveUser().getEmail() || 'Unknown';
  const msg = `${user} edited
${sh.getName()}!${e.range.getA1Notation()} -> ${e.value}`;
  logMessage(msg);
}
```

What & why: Lightweight audit trail.

---

64) Validate Data Types in a Range (numbers only)

```
function validateNumbersOnly() {
  const r = SpreadsheetApp.getActiveRange();
  const values = r.getValues();
  const bad = [];
  values.forEach((row,i) => row.forEach((v,j)=>{ if (v!=='' &&
isNaN(Number(v))) bad.push(r.getCell(i+1,j+1).getA1Notation()); }));
  if (bad.length) SpreadsheetApp.getUi().alert('Non-numeric cells: ' +
bad.join(', '));
}
```

What & why: Quick sanity check before analysis/export.

---

65) Flag Outliers (Z-score > 3)

```
function flagOutliers() {
  const r = SpreadsheetApp.getActiveRange();
  const vals = r.getValues().flat().map(Number).filter(v=>!isNaN(v));
  const mean = vals.reduce((a,b)=>a+b,0)/vals.length || 0;
  const std =
Math.sqrt(vals.reduce((a,b)=>a+Math.pow(b-mean,2),0)/Math.max(1,vals.l
ength-1));
  for (let i=1; i<=r.getNumRows(); i++) {
    for (let j=1; j<=r.getNumColumns(); j++) {
      const v = Number(r.getCell(i,j).getValue());
      if (!isNaN(v) && std>0 && Math.abs((v-mean)/std) > 3) {
        r.getCell(i,j).setBackground('#ffe6e6');
      }
    }
  }
}
```

What & why: Visually tag extreme values.

---

66) Create a “Data Dictionary” Sheet from Headers

```
function buildDataDictionary() {
  const sh = SpreadsheetApp.getActiveSheet();
  const headers =
sh.getRange(1,1,1,sh.getLastColumn()).getValues()[0];
  const dict = SpreadsheetApp.getActive().getSheetByName('Data
Dictionary') || SpreadsheetApp.getActive().insertSheet('Data
Dictionary');
```

```
  dict.clear();

dict.getRange(1,1,1,3).setValues([['Field','Type','Description']]).set
FontWeight('bold');
  const rows = headers.map(h => [h, 'string|number|date', 'Describe
this field...']);
  dict.getRange(2,1,rows.length,3).setValues(rows);
}
```

What & why: Documentation for collaborators and future you.

N. Performance, Properties, Cache

67) Batch Write Example (fast updates)

```
function batchUpdateExample() {
  const sh = SpreadsheetApp.getActiveSheet();
  const lr = sh.getLastRow(), lc = sh.getLastColumn();
  const values = sh.getRange(2,1,lr-1,lc).getValues();
  const out = values.map(row => row.map(v => typeof v==='string' ?
v.trim() : v));
  sh.getRange(2,1,out.length,lc).setValues(out);
}
```

What & why: Read once, write once—significantly faster.

---

68) Use Script Properties to Store Config

```
function setConfig() {
  const props = PropertiesService.getScriptProperties();
  props.setProperty('ALERT_THRESHOLD', '1000');
}
```

```
function getConfigThreshold() {
  return
Number(PropertiesService.getScriptProperties().getProperty('ALERT_THRE
SHOLD')) || 1000;
}
```

What & why: Keep magic numbers out of the code.

---

69) Cache Expensive Computation for 10 Minutes

```
function getExpensiveResult() {
  const cache = CacheService.getScriptCache();
  const cached = cache.get('expensive');
  if (cached) return JSON.parse(cached);

  const result = { time: new Date(), value: Math.random() }; //
simulate
  cache.put('expensive', JSON.stringify(result), 600);
  return result;
}
```

What & why: Reduces quota usage and speeds repeated runs.

---

70) Range Chunk Writer (avoid 50k cell limits)

```
function writeInChunks(sh, startRow, startCol, data, chunkRows) {
  for (let i = 0; i < data.length; i += chunkRows) {
    const slice = data.slice(i, i + chunkRows);
    sh.getRange(startRow + i, startCol, slice.length,
slice[0].length).setValues(slice);
  }
}
```

What & why: Helper for huge writes that risk timeouts.

---

71) Detect and Skip Unchanged Rows

```
function updateOnlyChanged() {
  const sh = SpreadsheetApp.getActiveSheet();
  const data =
sh.getRange(2,1,sh.getLastRow()-1,sh.getLastColumn()).getValues();
  const out = [];
  data.forEach((r,i) => {
    if (String(r[1]).trim() !== String(r[2]).trim()) { // example
compare B vs C
      r[3] = 'Updated'; // write to D
      out.push({row:i+2, rowData:r});
    }
  });
  out.forEach(o =>
sh.getRange(o.row,1,1,data[0].length).setValues([o.rowData]));
}
```

What & why: Minimizes write operations for speed.

---

72) Store & Use Per-User Preferences

```
function setMyPreference(key, value) {
  PropertiesService.getUserProperties().setProperty(key,
String(value));
}
function getMyPreference(key, def='') {
  return PropertiesService.getUserProperties().getProperty(key) ??
def;
}
```

What & why: Personalize behavior per collaborator.

O. Power Tools

73) Multi-Criteria Filter into New Sheet

```
function filterMulti() {
  const sh = SpreadsheetApp.getActive().getSheetByName('Data');
  const rows = sh.getDataRange().getValues();
  const head = rows.shift();
  const idxStatus = head.indexOf('Status');
  const idxOwner = head.indexOf('Owner');

  const matches = rows.filter(r => (r[idxStatus]==='Open' ||
r[idxStatus]==='In Progress') && r[idxOwner]==='Lars');
  const out = SpreadsheetApp.getActive().getSheetByName('Filtered
Multi') || SpreadsheetApp.getActive().insertSheet('Filtered Multi');
  out.clear(); out.getRange(1,1,1,head.length).setValues([head]);
  if (matches.length)
out.getRange(2,1,matches.length,head.length).setValues(matches);
}
```

What & why: Reusable pattern for complex filters.

---

74) Pivot-Like Summary by Two Keys

```
function pivotTwoKeys() {
  const sh = SpreadsheetApp.getActive().getSheetByName('Data');
  const data = sh.getDataRange().getValues();
  const head = data.shift();
  const colA = head.indexOf('Region');
  const colB = head.indexOf('Product');
  const colV = head.indexOf('Amount');
```

```
  const keysetA = new Set(), keysetB = new Set();
  const map = new Map();
  data.forEach(r=>{
    const a = r[colA], b = r[colB], v = Number(r[colV])||0;
    keysetA.add(a); keysetB.add(b);
    const key = a+'||'+b;
    map.set(key, (map.get(key)||0)+v);
  });

  const A = [...keysetA], B = [...keysetB];
  const out = SpreadsheetApp.getActive().getSheetByName('Pivot2') ||
SpreadsheetApp.getActive().insertSheet('Pivot2');
  out.clear();
  out.getRange(1,1,1,B.length+1).setValues([['Region/Prod', ...B]]);
  const rows = A.map(a => [a, ...B.map(b => map.get(a+'||'+b)||0)]);
  out.getRange(2,1,rows.length, rows[0].length).setValues(rows);
}
```

What & why: Quick “pivot” without the UI.

---

75) Merge Duplicate Keys and Keep the Latest

```
function mergeKeepLatest() {
  const sh = SpreadsheetApp.getActiveSheet();
  const data = sh.getDataRange().getValues();
  const head = data.shift();
  const keyIdx = 1; // key in column B
  const dateIdx = 3; // date in column D
  const map = new Map();

  data.forEach(r => {
    const key = String(r[keyIdx]);
    const d = r[dateIdx] instanceof Date ? r[dateIdx].getTime() : 0;
    const prev = map.get(key);
```

```
    if (!prev || d > prev._ts) map.set(key, Object.assign({_ts:d},
r));
  });

  const merged = [...map.values()].map(o => {
    const {_ts, ...rest} = o;
    return Object.values(rest);
  });

  sh.clearContents();
  sh.getRange(1,1,1,head.length).setValues([head]);
  if (merged.length)
sh.getRange(2,1,merged.length,head.length).setValues(merged);
}
```

What & why: For CRMs/exports where duplicates arrive—keeps the freshest row.

P. Custom Functions (usable directly in cells)

76) UNIQUECOUNT(range)

Counts distinct, non-empty values.

```
/**
 * =UNIQUECOUNT(A2:A)
 */
function UNIQUECOUNT(range) {
  range = range.filter(r => r[0] !== '' && r[0] !== null && r[0] !==
undefined);
  const set = new Set(range.map(r => String(r[0])));
  return set.size;
}
```

What & why: Quick unique count without building a pivot.

---

77) TEXTJOINIF(delim, conditionRange, criterion, joinRange)

Conditional text join (like Excel's TEXTJOIN + IF).

```
/**
 * =TEXTJOINIF(", ", A2:A, "Open", B2:B)
 */
function TEXTJOINIF(delim, conditionRange, criterion, joinRange) {
  const out = [];
  for (let i = 0; i < conditionRange.length; i++) {
    if (String(conditionRange[i][0]) === String(criterion)) {
      const v = joinRange[i] && joinRange[i][0] != null ?
String(joinRange[i][0]) : '';
      if (v) out.push(v);
    }
  }
  return out.join(delim);
}
```

What & why: Build comma-separated lists based on a condition.

---

78) SUMVISIBLE(range)

Sums only the **visible** (unfiltered/unhidden) cells.

```
/**
 * =SUMVISIBLE(B2:B100)
 */
function SUMVISIBLE(range) {
  const sheet = SpreadsheetApp.getActiveRange().getSheet();
  const v =  range.reduce((sum, row, i) => {
    const r = SpreadsheetApp.getActiveRange().getRow() + i;
```

```
    if (!sheet.isRowHiddenByFilter(r) && !sheet.isRowHiddenByUser(r))
{
      const x = Number(row[0]);
      if (!isNaN(x)) sum += x;
    }
    return sum;
  }, 0);
  return v;
}
```

What & why: Totals that respect your filters/hiding.

---

79) REGEXEXTRACTALL(text, pattern)

Returns **all** matches (one per row).

```
/**
 * =REGEXEXTRACTALL(A2, "[A-Za-z]+")
 */
function REGEXEXTRACTALL(text, pattern) {
  const re = new RegExp(pattern, 'g');
  const out = [];
  let m;
  while ((m = re.exec(String(text))) !== null) out.push([m[0]]);
  return out.length ? out : [['']];
}
```

What & why: Google Sheets' REGEXEXTRACT returns only the first match—this gets them all.

---

80) PARSEJSONKEY(json, key)

Gets a top-level key's value from a JSON string.

```
/**
 * =PARSEJSONKEY(A2, "name")
 */
function PARSEJSONKEY(json, key) {
  try {
    const obj = JSON.parse(String(json));
    const val = obj[key];
    if (val === undefined || val === null) return '';
    if (Array.isArray(val)) return val.join(', ');
    if (typeof val === 'object') return JSON.stringify(val);
    return String(val);
  } catch (_) {
    return '';
  }
}
```

What & why: Extract quick fields without full scripting.

Q. UI (Sidebar/Dialog) Utilities

81) Sidebar “Quick Tools”

Adds a sidebar with buttons calling functions.

```
function showQuickToolsSidebar() {
  const html = HtmlService.createHtmlOutput(`
    <div style="font-family:sans-serif">
      <h3>Quick Tools</h3>
      <button
onclick="google.script.run.autoResizeAllColumns()">Auto-resize
columns</button>
      <button onclick="google.script.run.styleHeader()">Style
header</button>
```

```
      <button
onclick="google.script.run.snapshotActiveSheet()">Snapshot
sheet</button>
      <p id="status"></p>
      <script>
        google.script.run.withSuccessHandler(() => {
          document.getElementById('status').textContent = 'Ready';
        }).logMessage('Sidebar opened');
      </script>
    </div>
  `);
  html.setTitle('Quick Tools');
  SpreadsheetApp.getUi().showSidebar(html);
}
```

What & why: A friendly panel for non-technical users to run stuff.

---

82) Simple Input Dialog → Append to Sheet

Prompt for Name/Email; append a row.

```
function promptAndAppend() {
  const ui = SpreadsheetApp.getUi();
  const name = ui.prompt('Enter Name').getResponseText();
  const email = ui.prompt('Enter Email').getResponseText();
  const sh = SpreadsheetApp.getActive().getSheetByName('Intake') ||
SpreadsheetApp.getActive().insertSheet('Intake');
  if (sh.getLastRow() === 0)
sh.getRange(1,1,1,3).setValues([['Timestamp','Name','Email']]);
  sh.appendRow([new Date(), name, email]);
}
```

What & why: Zero-setup intake without Google Forms.

---

83) Global Find/Replace (all sheets)

Searches every tab and replaces text.

```
function findReplaceAllSheets() {
  const ui = SpreadsheetApp.getUi();
  const f = ui.prompt('Find what?').getResponseText();
  const r = ui.prompt('Replace with?').getResponseText();
  const ss = SpreadsheetApp.getActive();
  ss.getSheets().forEach(sh => {
    const range = sh.getDataRange();
    const vals = range.getValues().map(row => row.map(v => typeof v
=== 'string' ? v.split(f).join(r) : v));
    range.setValues(vals);
  });
}
```

What & why: Safer than manual: one run, consistent.

R. Formatting & Layout

84) Freeze Top N Rows (all sheets)

```
function freezeTopNAllSheets() {
  const ui = SpreadsheetApp.getUi();
  const n = Number(ui.prompt('Freeze how many header
rows?').getResponseText()) || 1;
  SpreadsheetApp.getActive().getSheets().forEach(sh =>
sh.setFrozenRows(n));
}
```

What & why: Uniform headers across the workbook.

---

85) Hide Columns by Header Name

```
function hideColumnsByHeaderNames() {
  const namesToHide = ['Internal Notes','Cost']; // edit me
  const sh = SpreadsheetApp.getActiveSheet();
  const head = sh.getRange(1,1,1,sh.getLastColumn()).getValues()[0];
  head.forEach((h, i) => {
    if (namesToHide.includes(String(h))) sh.hideColumns(i+1);
  });
}
```

What & why: Quickly hide sensitive columns.

---

86) Unhide All Rows & Columns (current sheet)

```
function unhideAll() {
  const sh = SpreadsheetApp.getActiveSheet();
  sh.showColumns(1, sh.getMaxColumns());
  sh.showRows(1, sh.getMaxRows());
}
```

What & why: Reset visibility when things get messy.

---

87) Zebra Striping (banded rows)

```
function zebraStripe() {
  const sh = SpreadsheetApp.getActiveSheet();
  const lr = sh.getLastRow(), lc = sh.getLastColumn();
  const rng = sh.getRange(2,1,Math.max(0,lr-1),lc);
  rng.setBackground(null);
  for (let r = 2; r <= lr; r+=2) {
    sh.getRange(r,1,1,lc).setBackground('#f7f7f7');
  }
```

```
}
```

What & why: Improves readability at a glance.

---

88) Build an “Index” Sheet with Links to Tabs

```
function buildSheetIndex() {
  const ss = SpreadsheetApp.getActive();
  const idx = ss.getSheetByName('Index') || ss.insertSheet('Index');
  idx.clear();
  idx.getRange('A1:B1').setValues([['Sheet
Name','Link']]).setFontWeight('bold');
  const rows = ss.getSheets().filter(s=>s.getName()!=='Index').map(s
=> [s.getName(), `=HYPERLINK("#gid=${s.getSheetId()}", "Open")`]);
  if (rows.length) idx.getRange(2,1,rows.length,2).setValues(rows);
}
```

What & why: Quick navigation hub.

---

89) Protect Ranges by Header Name (lock everything except allowed)

```
function protectExceptAllowedHeaders() {
  const allowed = ['Notes','Status']; // editable columns
  const sh = SpreadsheetApp.getActiveSheet();
  const head = sh.getRange(1,1,1,sh.getLastColumn()).getValues()[0];
  const protection = sh.protect().setDescription('Protected except
some columns');
  protection.removeEditors(protection.getEditors());
  const unprotected = head.map((h,i)=> allowed.includes(String(h)) ?
sh.getRange(1,i+1,sh.getMaxRows(),1) : null).filter(Boolean);
  protection.setUnprotectedRanges(unprotected);
}
```

What & why: Control edits by column names.

---

90) Data Validation from Named Range

```
function validateFromNamedRange() {
  const ss = SpreadsheetApp.getActive();
  const r = ss.getRangeByName('StatusList'); // e.g., a named list
  if (!r) throw new Error('Named range "StatusList" not found.');
  const rule =
SpreadsheetApp.newDataValidation().requireValueInRange(r,
true).setAllowInvalid(false).build();
  SpreadsheetApp.getActiveRange().setDataValidation(rule);
}
```

What & why: Centralized list control, easy to update.

S. Sheet Generation & Mass Actions

91) Duplicate Template for Each Value in a List

```
function duplicateTemplateForList() {
  const ss = SpreadsheetApp.getActive();
  const template = ss.getSheetByName('Template');
  const listing = ss.getSheetByName('Names'); // column A list
  const names =
listing.getRange(2,1,listing.getLastRow()-1,1).getValues().map(r=>Stri
ng(r[0])).filter(Boolean);
  names.forEach(n => {
    const copy = template.copyTo(ss);
    copy.setName(n.substring(0, 99));
    copy.getRange('B1').setValue(n); // example fill
```

```
  });
}
```

What & why: Batch-generate personalized tabs.

---

92) Compact Data: Remove Blank Rows Within Data Range

```
function compactRemoveBlankRows() {
  const sh = SpreadsheetApp.getActiveSheet();
  const range = sh.getDataRange();
  const data = range.getValues();
  const header = data.shift();
  const filtered = data.filter(r => r.some(c => c !== '' && c !==
null));
  sh.clearContents();
  sh.getRange(1,1,1,header.length).setValues([header]);
  if (filtered.length)
sh.getRange(2,1,filtered.length,header.length).setValues(filtered);
}
```

What & why: Eliminates pesky empty rows inside your table.

---

93) Append Selection to “log” (values only)

```
function appendSelectionToLog() {
  const ss = SpreadsheetApp.getActive();
  const log = ss.getSheetByName('log') || ss.insertSheet('log');
  const sel = SpreadsheetApp.getActiveRange();
  const vals = sel.getValues();
  vals.forEach(row => log.appendRow([new Date(), ...row]));
}
```

What & why: Quick paste-as-values history.

---

94) Toggle “Done” Checkbox → Set Completed Timestamp

```
function onEditToggleDoneTimestamp(e) {
  const sh = e.range.getSheet();
  const head = sh.getRange(1,1,1,sh.getLastColumn()).getValues()[0];
  const doneCol = head.indexOf('Done?') + 1;
  const tsCol = head.indexOf('Completed At') + 1;
  if (!doneCol || !tsCol) return;
  if (e.range.getColumn() === doneCol && e.range.getRow() > 1) {
    if (String(e.value) === 'TRUE') sh.getRange(e.range.getRow(),
tsCol).setValue(new Date());
    else sh.getRange(e.range.getRow(), tsCol).clearContent();
  }
}
```

What & why: Lightweight task tracking.

---

95) Fill Missing Dates in a Daily Series (A column)

```
function fillMissingDatesDaily() {
  const sh = SpreadsheetApp.getActiveSheet();
  const dates =
sh.getRange(2,1,sh.getLastRow()-1,1).getValues().map(r=>r[0]).filter(B
oolean).sort((a,b)=>a-b);
  if (!dates.length) return;
  const start = new Date(dates[0]), end = new
Date(dates[dates.length-1]);
  const set = new Set(dates.map(d=>new Date(d).toDateString()));
  const fill = [];
  for (let d = new Date(start); d <= end; d.setDate(d.getDate()+1)) {
    if (!set.has(d.toDateString())) fill.push([new Date(d)]);
```

```
  }
  if (fill.length)
sh.getRange(sh.getLastRow()+1,1,fill.length,1).setValues(fill);
}
```

What & why: Ensures continuity for charts/time-series.

---

96) Generate Business Days Between Two Cells

Takes start in B1, end in B2, writes to column D.

```
function listBusinessDays() {
  const sh = SpreadsheetApp.getActiveSheet();
  const start = new Date(sh.getRange('B1').getValue());
  const end = new Date(sh.getRange('B2').getValue());
  const out = [];
  for (let d = new Date(start); d <= end; d.setDate(d.getDate()+1)) {
    const day = d.getDay();
    if (day !== 0 && day !== 6) out.push([new Date(d)]);
  }
  sh.getRange(2,4,out.length,1).setValues(out); // column D
}
```

What & why: Helps plan workloads & SLAs.

---

97) Create a Calendar Grid for a Month

Year in B1, month (1–12) in B2. Outputs a 6x7 grid.

```
function buildMonthCalendar() {
  const sh = SpreadsheetApp.getActiveSheet();
  const year = Number(sh.getRange('B1').getValue());
  const month = Number(sh.getRange('B2').getValue()) - 1;
```

```
  const first = new Date(year, month, 1);
  const start = new Date(first);
  start.setDate(first.getDate() - ((first.getDay()+6)%7)); // start
Monday
  const grid = [];
  for (let r=0; r<6; r++) {
    const row = [];
    for (let c=0; c<7; c++) {
      row.push(new Date(start));
      start.setDate(start.getDate()+1);
    }
    grid.push(row);
  }
  const out = sh.getRange('E2:K7');
  out.setNumberFormat('d'); out.setValues(grid);

sh.getRange('E1:K1').setValues([['Mon','Tue','Wed','Thu','Fri','Sat','
Sun']]).setFontWeight('bold');
}
```

What & why: Roll-your-own calendar inside Sheets.

---

98) Email Reminders for Tasks Due in 3 Days

Assumes columns: `Task | Due Date | Assignee Email`.

```
function emailDueSoon() {
  const sh = SpreadsheetApp.getActive().getSheetByName('Tasks');
  const data = sh.getDataRange().getValues();
  const head = data.shift();
  const idxTask = head.indexOf('Task');
  const idxDue = head.indexOf('Due Date');
  const idxEmail = head.indexOf('Assignee Email');
  const now = new Date();
  const cutoff = new Date(now.getFullYear(), now.getMonth(),
now.getDate()+3);
```

```
  data.forEach(r => {
    const due = r[idxDue];
    const to = r[idxEmail];
    if (due instanceof Date && due <= cutoff && due >= now && to) {
      MailApp.sendEmail(String(to), 'Task due soon: ' + r[idxTask],
`Hi,\n\nReminder: "${r[idxTask]}" is due on ${due}.\n\nThanks`);
    }
  });
}
```

What & why: Nudge people before deadlines slip.

---

99) Remove All Conditional Formatting Rules (sheet)

```
function clearConditionalFormatting() {
  const sh = SpreadsheetApp.getActiveSheet();
  sh.setConditionalFormatRules([]);
}
```

What & why: Start fresh when rules pile up.

---

100) Safe Delete Sheets Except a Whitelist

```
function deleteSheetsExcept() {
  const keep = new Set(['Index','Template','Data','KPIs']); // edit me
  const ss = SpreadsheetApp.getActive();
  ss.getSheets().forEach(sh => {
    if (!keep.has(sh.getName())) ss.deleteSheet(sh);
  });
}
```

What & why: Clean a workbook without nuking critical tabs.