10 hands-on Google Apps Script exercises focused on Google Sheets

At the end, there's one **single .gs file** you can paste into your Apps Script project. It creates the sheets, seeds data, and lets you run each exercise from a custom menu.

How to get set up (once)

1. Open (or create) a Google Sheet.
2. Extensions → **Apps Script**.
3. Delete any placeholder code and paste in the **ONE full code file** below.
4. Click **Save** (give the project any name).
5. Run `ex1_setup_sales_sheet()` once to seed the sample data (or use **Exercises** menu added by the script).
6. Then run each exercise function in order, or use **Exercises → Run All Exercises**.

Exercise 1 — Create & Seed a “Sales” sheet

Objective: Learn to create a sheet, write headers/rows, and format columns.
You’ll accomplish: A new `Sales` sheet with sample rows: Date, Region, Salesperson, Product, Units, UnitPrice, Total.

Steps

1. Run `ex1_setup_sales_sheet()`.
2. Open the **Sales** sheet—data will be ready to use in the next exercises.

How it works (high-level)

- Creates or clears a sheet named **Sales**.
- Writes headers and 15+ sample rows.
- Calculates **Total** = Units × UnitPrice in the script before writing.
- Applies date and currency formats; freezes header row.

Exercise 2 — Totals by Product (write a summary)

Objective: Read ranges, aggregate data in JavaScript, and write a summary table.
You’ll accomplish: A `Totals` sheet with Product → Total Sales, sorted desc.

Steps

1. Run `ex2_totals_by_product()`.
2. Check the **Totals** sheet to see the summary.

How it works

- Reads all data from **Sales**.
- Builds a product → sum map in JS.
- Writes a two-column table and formats as currency.
- Sorts totals descending.

Exercise 3 — Append a New Sale Row

Objective: Append a new row with computed values and a timestamp.
You'll accomplish: Adds a new sale record to **Sales** (timestamped today).

Steps

1. Run `ex3_append_sale_row()`.
2. See the new row at the bottom of **Sales**.

How it works

- Picks a sample Region/Salesperson/Product, Units, and UnitPrice.
- Computes Total and appends the row.

Exercise 4 — Highlight the Top 3 Sales

Rows

Objective: Work with arrays from a sheet and set background colors programmatically.
You'll accomplish: The three highest-Total rows in **Sales** are highlighted light green.

Steps

1. Run `ex4_highlight_top3_sales()`.
2. Notice top 3 by **Total** shaded.

How it works

- Reads all rows, finds top 3 **Total** values, and sets background colors only on those rows.
- Resets others to white.

Exercise 5 — Standardize Salesperson Names (Find & Replace)

Objective: Use TextFinder and dictionary-based standardization.
You'll accomplish: Fixes inconsistent names (e.g., "Alyce" → "Alice", "bob" → "Bob").

Steps

1. Run `ex5_standardize_salesperson_names()`.
2. Check **Salesperson** column in **Sales**—inconsistencies are fixed.

How it works

- Uses a mapping dictionary of incorrect → correct spellings.
- For each mapping, uses `createTextFinder` over the Salesperson column to replace all.

Exercise 6 — Remove Duplicate Sales

Objective: Identify duplicates by a composite key and delete them.
You'll accomplish: Deduped **Sales** (keeps first occurrence).

Steps

1. Run `ex6_remove_duplicate_sales()`.
2. Rows that are exact repeats by Date+Salesperson+Product are removed.

How it works

- Builds a key for each row (date string + salesperson + product).
- Tracks seen keys and deletes later duplicates (bottom-up to keep row indexes stable).

Exercise 7 — Create a Filtered "HighValue" Sheet

Objective: Filter in code and write results to a new sheet.
You'll accomplish: A `HighValue` sheet containing only rows with **Total ≥ 500**.

Steps

1. Run `ex7_create_high_value_sheet()`.
2. Open **HighValue** to see the filtered subset.

How it works

- Filters the **Sales** array for totals ≥ threshold.

- Writes a new table with headers; keeps formats consistent.

Exercise 8 — Insert a Chart Programmatically

Objective: Create a chart with the Apps Script Charts API.
You'll accomplish: A column chart of **Totals by Product** embedded on the `Totals` sheet.

Steps

1. Run `ex8_insert_totals_chart()`.
2. Open **Totals**—a column chart appears under the summary table.

How it works

- Uses `newChart()` builder on the `Totals` sheet.
- Binds the chart to the product/total range.
- Positions it below the data.

Exercise 9 — Custom Menu & Custom Functions

Objective: Add a menu and define functions usable directly in cells.
You'll accomplish:

- A menu **Exercises** → run any exercise or **Run All**.
- Custom functions: `TAX_AMOUNT(amount, rate)` and

`AVERAGE_IGNORE_ZERO(range)`.

Steps

1. Reload your spreadsheet so `onOpen()` runs and menu appears.
2. In any cell, try: `=TAX_AMOUNT(100, 0.13)` → **13**
3. Try: `=AVERAGE_IGNORE_ZERO(A2:A100)` to ignore zeros.

How it works

- `onOpen()` builds the **Exercises** menu.
- Custom functions are top-level and return simple values/arrays to the cell.

Exercise 10 — Data Validation + onEdit Auto-Calc

Objective: Create a dropdown and calculate row totals automatically on edit.
You'll accomplish:

- An `Input` sheet with a **Product** dropdown and **Units/UnitPrice** fields.
- When you type Units or UnitPrice, **Total** updates instantly via `onEdit`.

Steps

1. Run `ex10_data_validation_with_onedit()`.
2. On **Input**, choose a product, type **Units** and **UnitPrice**—**Total** fills in.

How it works

- Creates list validation on **Product** from known products.

- `onEdit(e)` watches edits on **Input**; when Units/UnitPrice changes, it sets **Total** = Units × UnitPrice for that row.

One complete .gs file (paste this whole thing)

```
/***** Utilities *****/
function get_or_create_sheet_(name) {
  const ss = SpreadsheetApp.getActive();
  let sh = ss.getSheetByName(name);
  if (!sh) sh = ss.insertSheet(name);
  return sh;
}

function set_headers_(sh, headers) {
  sh.clear();
  sh.getRange(1, 1, 1, headers.length).setValues([headers]);
  sh.setFrozenRows(1);
}

function seed_sales_data_if_needed_() {
  const sh = get_or_create_sheet_("Sales");
  const headers = ["Date", "Region", "Salesperson", "Product",
"Units", "UnitPrice", "Total"];
  if (sh.getLastRow() < 2) {
    set_headers_(sh, headers);
    const rows = [
      // Date (YYYY, M-1, D),   Region,  Salesperson (some
intentionally messy), Product, Units, UnitPrice
      [new Date(2025,0,5),     "East",   "Alice",  "Widget", 5, 20],
      [new Date(2025,0,6),     "West",   "bob",    "Gizmo",  12, 35],
```

```
      [new Date(2025,0,7),     "South",  "Carmen", "Doodad", 3, 120],
      [new Date(2025,0,8),     "North",  "Deepa",  "Widget", 10,
22.5],
      [new Date(2025,0,9),     "Central","Evan",   "Gizmo",  9,  40],
      [new Date(2025,0,10),    "East",   "Alyce",  "Widget", 8,  20],
      [new Date(2025,0,11),    "West",   "Bob",    "Doodad", 2,  150],
      [new Date(2025,0,12),    "South",  "Carmen", "Gizmo",  7,  38],
      [new Date(2025,0,12),    "South",  "Carmen", "Gizmo",  7,  38],
// duplicate on purpose
      [new Date(2025,0,13),    "North",  "Deepa",  "Widget", 14, 21],
      [new Date(2025,0,14),    "Central","Evan",   "Doodad", 4,  130],
      [new Date(2025,0,15),    "East",   "Alice",  "Gizmo",  11,
36.5],
      [new Date(2025,0,16),    "West",   "Bob",    "Widget", 6,  23],
      [new Date(2025,0,17),    "South",  "Carmen", "Doodad", 5,  125],
      [new Date(2025,0,18),    "North",  "Deepa",  "Gizmo",  13, 34],
      [new Date(2025,0,19),    "Central","Evan",   "Widget", 9,  22]
    ];
    const out = rows.map(r => {
      const total = Number(r[4]) * Number(r[5]);
      return [r[0], r[1], r[2], r[3], r[4], r[5], total];
    });
    sh.getRange(2, 1, out.length, out[0].length).setValues(out);
    // Formats
    sh.getRange("A2:A").setNumberFormat("yyyy-mm-dd");
    sh.getRange("F2:G").setNumberFormat("$#,##0.00");
    sh.autoResizeColumns(1, 7);
  }
  return sh;
}
```

```
/***** Exercise 1 *****/
function ex1_setup_sales_sheet() {
  const sh = get_or_create_sheet_("Sales");
  sh.clear();
  seed_sales_data_if_needed_();
}

/***** Exercise 2 *****/
function ex2_totals_by_product() {
  const sales = seed_sales_data_if_needed_();
  const values = sales.getDataRange().getValues();
  const headers = values.shift(); // remove headers

  const idxProduct = headers.indexOf("Product");
  const idxTotal = headers.indexOf("Total");

  const totalsMap = {};
  values.forEach(r => {
    const p = r[idxProduct];
    const t = Number(r[idxTotal]) || 0;
    totalsMap[p] = (totalsMap[p] || 0) + t;
  });

  const totals = Object.entries(totalsMap).map(([p, t]) => [p, t]);
  totals.sort((a, b) => b[1] - a[1]);

  const sh = get_or_create_sheet_("Totals");
  set_headers_(sh, ["Product", "TotalSales"]);
  if (totals.length) {
    sh.getRange(2, 1, totals.length, 2).setValues(totals);
```

```
    sh.getRange(2, 2, totals.length, 1).setNumberFormat("$#,##0.00");
    sh.autoResizeColumns(1, 2);
  }
}

/***** Exercise 3 *****/
function ex3_append_sale_row() {
  const sh = seed_sales_data_if_needed_();
  const regions = ["East", "West", "North", "South", "Central"];
  const salespeople = ["Alice", "Bob", "Carmen", "Deepa", "Evan"];
  const products = ["Widget", "Gizmo", "Doodad"];

  // Simple demo data
  const region = regions[Math.floor(Math.random() * regions.length)];
  const person = salespeople[Math.floor(Math.random() *
salespeople.length)];
  const product = products[Math.floor(Math.random() *
products.length)];
  const units = Math.ceil(5 + Math.random() * 10); // 5..15
  const unitPrice = product === "Doodad" ? 120 + Math.random() * 40 :
                    product === "Gizmo"  ? 30 + Math.random() * 10  :
                                           20 + Math.random() * 5;
  const total = units * unitPrice;

  const row = [new Date(), region, person, product, units, unitPrice,
total];
  sh.appendRow(row);
  const last = sh.getLastRow();
  sh.getRange(last, 1, 1, 1).setNumberFormat("yyyy-mm-dd");
  sh.getRange(last, 6, 1, 2).setNumberFormat("$#,##0.00");
}
```

```
/***** Exercise 4 *****/
function ex4_highlight_top3_sales() {
  const sh = seed_sales_data_if_needed_();
  const lastRow = sh.getLastRow();
  if (lastRow < 3) return;

  const range = sh.getRange(2, 1, lastRow - 1, 7);
  const values = range.getValues();

  const headers = sh.getRange(1, 1, 1, 7).getValues()[0];
  const idxTotal = headers.indexOf("Total");

  // Collect totals with row offsets
  const totals = values.map((r, i) => ({ idx: i, total:
Number(r[idxTotal]) || 0 }));
  totals.sort((a, b) => b.total - a.total);

  const topCount = Math.min(3, totals.length);
  const topIdx = new Set(totals.slice(0, topCount).map(o => o.idx));

  // Build background matrix
  const bg = values.map((_, i) => {
    const color = topIdx.has(i) ? "#d8f7d4" : "#ffffff"; // light
green or white
    return new Array(7).fill(color);
  });
  range.setBackgrounds(bg);
}

/***** Exercise 5 *****/
```

```
function ex5_standardize_salesperson_names() {
  const sh = seed_sales_data_if_needed_();
  const headers = sh.getRange(1, 1, 1,
sh.getLastColumn()).getValues()[0];
  const idxSalesperson = headers.indexOf("Salesperson") + 1; //
1-based


  const fixes = {
    "Alyce": "Alice",
    "alyce": "Alice",
    "bob": "Bob",
    "Bob ": "Bob",
    "  Bob": "Bob"
  };


  const colRange = sh.getRange(2, idxSalesperson, sh.getLastRow() - 1,
1);
  Object.keys(fixes).forEach(bad => {

colRange.createTextFinder(bad).matchCase(false).matchEntireCell(false)
.replaceAllWith(fixes[bad]);
  });
}


/***** Exercise 6 *****/
function ex6_remove_duplicate_sales() {
  const sh = seed_sales_data_if_needed_();
  const range = sh.getDataRange();
  const values = range.getValues();
  const headers = values.shift();
  const idxDate = headers.indexOf("Date");
  const idxSalesperson = headers.indexOf("Salesperson");
```

```
  const idxProduct = headers.indexOf("Product");

  const seen = new Set();
  const toDelete = [];
  values.forEach((r, i) => {
    const d = r[idxDate] instanceof Date ? r[idxDate] : new
Date(r[idxDate]);
    const dateKey = Utilities.formatDate(d,
Session.getScriptTimeZone(), "yyyy-MM-dd");
    const key = [dateKey, r[idxSalesperson], r[idxProduct]].join("|");
    if (seen.has(key)) {
      toDelete.push(i + 2); // sheet row index
    } else {
      seen.add(key);
    }
  });

  // Delete from bottom up
  toDelete.sort((a, b) => b - a).forEach(row => sh.deleteRow(row));
}

/***** Exercise 7 *****/
function ex7_create_high_value_sheet() {
  const src = seed_sales_data_if_needed_();
  const dst = get_or_create_sheet_("HighValue");
  const headers = src.getRange(1, 1, 1, 7).getValues()[0];
  const idxTotal = headers.indexOf("Total");

  const values = src.getDataRange().getValues();
  values.shift(); // drop headers
```

```
  const threshold = 500;
  const filtered = values.filter(r => Number(r[idxTotal]) >=
threshold);

  set_headers_(dst, headers);
  if (filtered.length) {
    dst.getRange(2, 1, filtered.length,
headers.length).setValues(filtered);
    dst.getRange("A2:A").setNumberFormat("yyyy-mm-dd");
    dst.getRange("F2:G").setNumberFormat("$#,##0.00");
    dst.autoResizeColumns(1, headers.length);
  }
}

/***** Exercise 8 *****/
function ex8_insert_totals_chart() {
  ex2_totals_by_product(); // ensure Totals exists
  const sh = get_or_create_sheet_("Totals");
  const last = sh.getLastRow();
  if (last < 2) return;

  // Remove existing charts for a clean demo (optional)
  sh.getCharts().forEach(c => sh.removeChart(c));

  // Data range: Product (A), TotalSales (B)
  const dataRange = sh.getRange(1, 1, last, 2);

  const chart = sh.newChart()
    .setChartType(Charts.ChartType.COLUMN)
    .addRange(dataRange)
    .setOption("title", "Total Sales by Product")
```

```
    .setPosition( last + 2, 1, 0, 0)
    .build();

  sh.insertChart(chart);
}

/***** Exercise 9 *****/
function onOpen() {
  // Build a menu to run exercises quickly
  SpreadsheetApp.getUi()
    .createMenu("Exercises")
    .addItem("1) Setup Sales Sheet", "ex1_setup_sales_sheet")
    .addItem("2) Totals by Product", "ex2_totals_by_product")
    .addItem("3) Append Sale Row", "ex3_append_sale_row")
    .addItem("4) Highlight Top 3", "ex4_highlight_top3_sales")
    .addItem("5) Standardize Names",
"ex5_standardize_salesperson_names")
    .addItem("6) Remove Duplicates", "ex6_remove_duplicate_sales")
    .addItem("7) HighValue Sheet", "ex7_create_high_value_sheet")
    .addItem("8) Insert Totals Chart", "ex8_insert_totals_chart")
    .addItem("10) Setup Input & onEdit",
"ex10_data_validation_with_onedit")
    .addSeparator()
    .addItem("Run All Exercises", "run_all_exercises")
    .addToUi();
}

// Custom functions (can be used in cells)
function TAX_AMOUNT(amount, rate) {
  amount = Number(amount); rate = Number(rate);
  if (isNaN(amount) || isNaN(rate)) return "";
```

```
  return amount * rate;
}

function AVERAGE_IGNORE_ZERO(range) {
  const flat = (Array.isArray(range) ? range.flat() :
[range]).map(Number);
  const nonZero = flat.filter(n => !isNaN(n) && n !== 0);
  if (!nonZero.length) return 0;
  return nonZero.reduce((a, b) => a + b, 0) / nonZero.length;
}

/***** Exercise 10 *****/
function ex10_data_validation_with_onedit() {
  const input = get_or_create_sheet_("Input");
  const headers = ["Product", "Units", "UnitPrice", "Total", "Region",
"Salesperson"];
  set_headers_(input, headers);
  input.setColumnWidths(1, headers.length, 140);

  // Build product list from Sales (or fallback)
  const sales = seed_sales_data_if_needed_();
  const vals = sales.getRange(2, 4, sales.getLastRow() - 1,
1).getValues().flat(); // Product col
  const products = Array.from(new Set(vals.length ? vals :
["Widget","Gizmo","Doodad"]));

  // Apply dropdown to Product column
  const rule = SpreadsheetApp.newDataValidation()
    .requireValueInList(products, true)
    .setAllowInvalid(false)
    .build();
  input.getRange(2, 1, 100, 1).setDataValidation(rule);
```

```
  // Formats
  input.getRange("C2:C").setNumberFormat("$#,##0.00");
  input.getRange("D2:D").setNumberFormat("$#,##0.00");
}

function onEdit(e) {
  try {
    const sh = e.range.getSheet();
    if (sh.getName() !== "Input") return;

    const row = e.range.getRow();
    const col = e.range.getColumn();
    if (row < 2) return;

    // Columns: A=Product, B=Units, C=UnitPrice, D=Total
    const units = Number(sh.getRange(row, 2).getValue() || 0);
    const price = Number(sh.getRange(row, 3).getValue() || 0);
    if (col === 2 || col === 3) {
      const total = units * price;
      sh.getRange(row,
4).setValue(total).setNumberFormat("$#,##0.00");
    }
  } catch (err) {
    // Swallow errors in simple trigger
  }
}

/***** Convenience: run all in order *****/
function run_all_exercises() {
  ex1_setup_sales_sheet();
```

```
  ex2_totals_by_product();
  ex3_append_sale_row();
  ex4_highlight_top3_sales();
  ex5_standardize_salesperson_names();
  ex6_remove_duplicate_sales();
  ex7_create_high_value_sheet();
  ex8_insert_totals_chart();
  ex10_data_validation_with_onedit();
}
```

Tips & Notes

- If you see an authorization prompt the first time you run a function, grant permissions.
- **onOpen** and **onEdit** are simple triggers and run automatically (no manual install needed). If the menu doesn't appear, refresh the sheet.
- You can safely re-run `ex1_setup_sales_sheet()` to reset the **Sales** sheet to a known state.
- All function names avoid special characters—use them directly in the editor or via the menu.

If you want, I can adapt these to your exact teaching style (e.g., rename columns, add more products, or switch to a different dataset like "Students & Grades").