MCC script: Google Ads Demand Gen + Display Placement Excluder.

Follow the instructions on https://cobyagency.com/2025/12/29/placement-excluder-script/

```
/**************************************************
 * MCC-Level: Demand Gen + Display Placement Excluder
 * Author: Coby Agency
 *
 * What this script does
 * 1) Reads your approvals from the existing 'queue' tab.
 *    The ONLY manual action is to type 'yes' in column P (exclude).
 * 2) Applies website placement exclusions via a shared placement exclusion list
 *    in each labelled account (EXCLUDED_LIST_NAME).
 * 3) Pulls fresh flagged placements (Demand Gen + Display) and rewrites the queue.
 *    It preserves your existing approvals by matching row_key.
 * 4) Keeps an audit log in 'history' without creating duplicates (de-duped in script storage).
 *
 * Important limitation. Mobile app placements
 * Google Ads Scripts cannot exclude mobile app placements via shared exclusion lists.
 * If you want to block apps, do it manually in Google Ads:
 * Insights and reports -> Where and When your ads showed -> Where ads showed
 * Then exclude the relevant apps or categories there.
```


```
/**************************************************
 * MCC-Level: Demand Gen + Display Placement Excluder
 * Author: Coby Agency
 *
 * What this script does
 * 1) Reads your approvals from the existing 'queue' tab.
 *    The ONLY manual action is to type 'yes' in column P (exclude).
 * 2) Applies website placement exclusions via a shared placement exclusion list
 *    in each labelled account (EXCLUDED_LIST_NAME).
 * 3) Pulls fresh flagged placements (Demand Gen + Display) and rewrites the queue.
 *    It preserves your existing approvals by matching row_key.
 * 4) Keeps an audit log in 'history' without creating duplicates (de-duped in script storage).
 *
 * Important limitation. Mobile app placements
 * Google Ads Scripts cannot exclude mobile app placements via shared exclusion lists.
 * If you want to block apps, do it manually in Google Ads:
 * Insights and reports -> Where and When your ads showed -> Where ads showed
 * Then exclude the relevant apps or categories there.
```

SCRIPT:

```
 ************************************************/

// ====================== CONFIG ======================
var SPREADSHEET_URL =
"https://docs.google.com/spreadsheets/d/1kJk5-XRRl_fbEldfIpwwu4F7sJ8Eqfzn7SQ7g-6GPvQ/edit?usp=sharing"; // paste your empty Google Sheets URL
var ACCOUNT_LABEL = "MCC_placement_excluder"; // Give your accounts this label to include them

// Thresholds
var IMPRESSIONS_THRESHOLD = 200; // Your minimum amount of impressions
var CLICKS_THRESHOLD = 10; // Clicks threshold
var CTR_THRESHOLD_PCT = 1; // CTR in %, 1 = 1%
var LOOKBACK_DAYS = 30; // lookback period

// Suspicious TLDs
var BAD_TLDS = [".fun", ".biz", ".top", ".xyz", ".icu", ".site", ".win", ".work", ".click", ".rocks", ".games", ".game"];

// Auto-exclude hard rule
var ENABLE_AUTO_EXCLUDE_BAD_TLDS = false;

// Exclusion list name (per account)
var EXCLUDED_LIST_NAME = "Auto Excluded . Display + Demand Gen";

// Performance lever
var ROW_LIMIT = 1000;

// Email notifications
var ENABLE_EMAIL_ALERTS = true; // Set to true when you want to receive email alerts
var ALERT_EMAILS = "hello@cobyagency.com"; // Your email address

// PropertiesService keys
var PROP_PREFIX = "coby_v2_";
var PROP_PROCESSED_KEYSET = PROP_PREFIX + "processed_keys";
var PROP_EXCLUDED_KEYSET  = PROP_PREFIX + "excluded_keys";
var PROP_SETUP_DONE       = PROP_PREFIX + "setup_done";
// ====================================================

function main() {
  Logger.log("=== START v2 ===");

  var ss = SpreadsheetApp.openByUrl(SPREADSHEET_URL);

  var queueSheet = ensureSheet_(ss, "queue");
```

```
  var historySheet = ensureSheet_(ss, "history");
  var allowlistSheet = ensureSheet_(ss, "allowlist");
  var instructionsSheet = ensureSheet_(ss, "instructions");

  ensureOneTimeSetup_(queueSheet, instructionsSheet);

  var allowlist = loadAllowlist_(allowlistSheet);

  // Read approvals BEFORE queue rewrite
  var approvalsBefore = readQueueApprovals_(queueSheet);
  Logger.log("Approvals read from existing queue: " + Object.keys(approvalsBefore).length);

  // Apply exclusions
  var applyResult = applyApprovedExclusions_(approvalsBefore, allowlist);
  Logger.log(
    "Exclusions applied . Newly excluded: " + applyResult.excluded_now +
    " . Allowlist skipped: " + applyResult.allowlist_skipped +
    " . App skipped: " + applyResult.app_skipped +
    " . Invalid skipped: " + applyResult.invalid_skipped +
    " . Errors: " + applyResult.errors
  );

  // Collect fresh flagged placements
  var accounts = getLabeledAccounts_(ACCOUNT_LABEL);
  if (!accounts.hasNext()) {
    Logger.log("No accounts found with label: " + ACCOUNT_LABEL);
    return;
  }

  var runTs = Utilities.formatDate(new Date(), AdsApp.currentAccount().getTimeZone(),
"yyyy-MM-dd HH:mm:ss");

  var flaggedRows = collectFlaggedPlacements_(accounts, runTs);
  Logger.log("Total flagged rows collected: " + flaggedRows.length);

  // Rewrite queue, preserving approvals
  writeQueueFast_(queueSheet, flaggedRows, approvalsBefore);
  Logger.log("Queue written: " + flaggedRows.length + " rows");

  // Append new unique items to history
  var historyResult = appendHistory_(historySheet, flaggedRows, runTs,
applyResult.excludedKeysMap);
  Logger.log("History appended: " + historyResult.appended);

  // Email summary
  if (ENABLE_EMAIL_ALERTS) {
    try {
```

```
      sendSummaryEmail_(ss.getUrl(), runTs, flaggedRows.length, approvalsBefore,
applyResult, historyResult);
      Logger.log("Summary email sent to: " + ALERT_EMAILS);
    } catch (e) {
      Logger.log("Email failed: " + e);
    }
  }

  Logger.log("=== END v2 ===");
}

/** ==================== ONE-TIME SETUP ==================== */

function ensureOneTimeSetup_(queueSheet, instructionsSheet) {
  var props = PropertiesService.getScriptProperties();
  var done = props.getProperty(PROP_SETUP_DONE);

  // Always keep instructions updated (safe overwrite)
  writeInstructions_(instructionsSheet);

  if (!done) {
    ensureQueueHeaders_(queueSheet);
    applyQueueVisuals_(queueSheet);
    props.setProperty(PROP_SETUP_DONE, "1");
  } else {
    // Ensure header exists even if someone deleted it
    ensureQueueHeaders_(queueSheet);
  }
}

function writeInstructions_(sheet) {
  sheet.clear();

  var content = [
    ["How to use the Placement Excluder (Coby Agency)"],
    [""],
    ["✅ Only manual action"],
    ["Go to the 'queue' tab. In column P (exclude), type: yes"],
    [""],
    ["How the workflow runs"],
    ["1) You mark exclude = yes in the queue"],
    ["2) Next script run adds those WEBSITE placements to the shared exclusion list"],
    ["3) The script refreshes the queue with new flagged placements and keeps your previous
yes-es"],
    [""],
    ["What you should NOT edit"],
    ["• Don't edit other columns in the queue (only column P)"],
    ["• Don't edit the history tab (audit log)"],
```

```
    [""],
    ["Mobile apps (important)"],
    ["Google Ads Scripts cannot exclude app placements via shared exclusion lists."],
    ["If you want to block apps, do it manually in Google Ads:"],
    ["Insights and reports -> Where and When your ads showed -> Where ads showed"],
    ["Then exclude the relevant apps or categories there."],
    [""],
    ["If you need help, reach out to: hello@cobyagency.com"]
  ];

  sheet.getRange(1, 1, content.length, 1).setValues(content);
  sheet.setColumnWidth(1, 900);
  sheet.setFrozenRows(1);

  // Simple visual cues
  sheet.getRange(3, 1).setFontWeight("bold"); // ✅ Only manual action
  sheet.getRange(4, 1).setBackground("#FFF2CC");
}

/** ==================== COLLECT + FLAG ==================== */

function collectFlaggedPlacements_(accountsIter, runTs) {
  var all = [];

  while (accountsIter.hasNext()) {
    var account = accountsIter.next();
    AdsManagerApp.select(account);

    var accountId = account.getCustomerId();
    var accountName = account.getName ? account.getName() : "";

    Logger.log(">> Collecting account: " + accountId + " . " + accountName);

    var flagged = getFlaggedPlacementsForAccount_(accountId, accountName, runTs);
    Logger.log("   Flagged rows: " + flagged.length);

    for (var i = 0; i < flagged.length; i++) all.push(flagged[i]);
  }

  return all;
}

function getFlaggedPlacementsForAccount_(accountId, accountName, runTs) {
  var bad = [];

  var query =
    "SELECT " +
    "  detail_placement_view.placement, " +
```

```
    "  detail_placement_view.display_name, " +
    "  campaign.id, " +
    "  campaign.name, " +
    "  campaign.advertising_channel_type, " +
    "  metrics.impressions, " +
    "  metrics.clicks, " +
    "  metrics.ctr, " +
    "  metrics.cost_micros " +
    "FROM detail_placement_view " +
    "WHERE " +
    "  campaign.advertising_channel_type IN ('DISPLAY','DEMAND_GEN') " +
    "  AND metrics.impressions >= " + IMPRESSIONS_THRESHOLD + " " +
    "  AND segments.date DURING LAST_" + LOOKBACK_DAYS + "_DAYS " +
    "LIMIT " + ROW_LIMIT;

  Logger.log("   Note: LIMIT " + ROW_LIMIT + " applied. Increase only if needed.");

  var rows = AdsApp.search(query);
  var scanned = 0;

  while (rows.hasNext()) {
    var row = rows.next();
    scanned++;

    var placement = (row.detailPlacementView && row.detailPlacementView.placement)
      ? String(row.detailPlacementView.placement).trim()
      : "";

    var displayName = (row.detailPlacementView && row.detailPlacementView.displayName)
      ? String(row.detailPlacementView.displayName).trim()
      : "";

    var impr = Number(row.metrics.impressions) || 0;
    var clicks = Number(row.metrics.clicks) || 0;
    var ctrPct = (Number(row.metrics.ctr) || 0) * 100;
    var cost = (Number(row.metrics.costMicros) || 0) / 1e6;

    var placementNorm = normalizePlacement_(placement);

    var reasons = [];
    if (ctrPct < CTR_THRESHOLD_PCT) reasons.push("low_ctr");
    if (clicks < CLICKS_THRESHOLD) reasons.push("low_clicks");
    if (endsWithAny_(placementNorm, BAD_TLDS)) reasons.push("bad_tld");

    if (looksLikeMobileAppPlacement_(placement, displayName))
reasons.push("app_placement");

    if (reasons.length === 0) continue;
```

```
    var channel = row.campaign.advertisingChannelType ?
String(row.campaign.advertisingChannelType) : "";
    var campaignId = row.campaign.id ? String(row.campaign.id) : "";
    var campaignName = row.campaign.name ? String(row.campaign.name) : "";

    var rowKey = [accountId, campaignId, placementNorm].join("||");

    bad.push({
      run_ts: runTs,
      account_id: accountId,
      account_name: accountName,
      campaign_id: campaignId,
      campaign_name: campaignName,
      channel: channel,
      placement: placement,
      placement_normalized: placementNorm,
      display_name: displayName,
      impr: impr,
      clicks: clicks,
      ctr_pct: ctrPct.toFixed(2),
      cost: cost.toFixed(2),
      flag_reasons: reasons.join(", "),
      row_key: rowKey
    });
  }

  Logger.log("   Scanned rows: " + scanned);
  return bad;
}

/** ==================== QUEUE ==================== */

function ensureQueueHeaders_(sheet) {
  var headers = getQueueHeaders_();

  if (sheet.getLastRow() === 0) {
    sheet.getRange(1, 1, 1, headers.length).setValues([headers]);
    sheet.setFrozenRows(1);
    return;
  }

  // If headers are wrong/missing, reset header row only
  var existing = sheet.getRange(1, 1, 1, headers.length).getValues()[0];
  if (existing.join("|") !== headers.join("|")) {
    sheet.getRange(1, 1, 1, headers.length).setValues([headers]);
    sheet.setFrozenRows(1);
  }
```

```
}

function writeQueueFast_(sheet, rows, approvalsMap) {
  ensureQueueHeaders_(sheet);
  var headers = getQueueHeaders_();

  var lastRow = sheet.getLastRow();
  if (lastRow > 1) {
    sheet.getRange(2, 1, lastRow - 1, headers.length).clearContent();
  }

  if (!rows.length) return;

  var out = new Array(rows.length);
  for (var i = 0; i < rows.length; i++) {
    var r = rows[i];
    var carryYes = approvalsMap && approvalsMap[r.row_key] ? "yes" : "";

    out[i] = [
      r.run_ts,
      r.account_id,
      r.account_name,
      r.campaign_id,
      r.campaign_name,
      r.channel,
      r.placement,
      r.placement_normalized,
      r.display_name,
      r.impr,
      r.clicks,
      r.ctr_pct,
      r.cost,
      r.flag_reasons,
      r.row_key,
      carryYes
    ];
  }

  sheet.getRange(2, 1, out.length, headers.length).setValues(out);
}

function getQueueHeaders_() {
  return [
    "run_ts",
    "account_id",
    "account_name",
    "campaign_id",
    "campaign_name",
```

```
    "channel",
    "placement",
    "placement_normalized",
    "display_name",
    "impr",
    "clicks",
    "ctr_pct",
    "cost",
    "flag_reasons",
    "row_key",
    "exclude" // column P
  ];
}

function applyQueueVisuals_(sheet) {
  var excludeCol = 16; // P
  sheet.setFrozenRows(1);

  sheet.getRange(1, excludeCol)
    .setFontWeight("bold")
    .setBackground("#FFD966")
    .setNote("ONLY ACTION REQUIRED. Type 'yes' here to exclude on the next run.");

  sheet.getRange("P:P").setBackground("#FFF2CC");
  sheet.getRange(1, 1, 1, sheet.getLastColumn()).setFontWeight("bold");
  sheet.getRange(1, 1).setNote("Review placements. ONLY edit column P (exclude).");
}

/** ==================== APPROVALS ==================== */

function readQueueApprovals_(queueSheet) {
  var approvals = {}; // row_key -> {account_id, placement_raw, display_name, flag_reasons}
  if (!queueSheet || queueSheet.getLastRow() < 2) return approvals;

  var data = queueSheet.getDataRange().getValues();
  var headers = data[0];

  var idx = indexMap_(headers, [
    "account_id",
    "placement",
    "display_name",
    "exclude",
    "row_key",
    "flag_reasons"
  ]);

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

```
    var accountId = String(row[idx.account_id] || "").trim();
    var placementRaw = String(row[idx.placement] || "").trim();
    var displayName = String(row[idx.display_name] || "").trim();
    var reasons = String(row[idx.flag_reasons] || "");
    var rowKey = String(row[idx.row_key] || "").trim();
    if (!accountId || !rowKey) continue;

    var ex = String(row[idx.exclude] || "").trim().toLowerCase();
    if (ex === "yes") {
      approvals[rowKey] = {
        account_id: accountId,
        placement_raw: placementRaw,
        display_name: displayName,
        flag_reasons: reasons
      };
      continue;
    }

    if (ENABLE_AUTO_EXCLUDE_BAD_TLDS) {
      if (reasons.indexOf("bad_tld") !== -1) {
        approvals[rowKey] = {
          account_id: accountId,
          placement_raw: placementRaw,
          display_name: displayName,
          flag_reasons: reasons
        };
      }
    }
  }

  return approvals;
}

/** ==================== EXCLUSIONS ==================== */

function applyApprovedExclusions_(approvals, allowlist) {
  var excludedKeys = loadJsonMapProp_(PROP_EXCLUDED_KEYSET);

  // Group by account
  var byAccount = {};
  var approvalKeys = Object.keys(approvals);

  for (var i = 0; i < approvalKeys.length; i++) {
    var key = approvalKeys[i];
    if (excludedKeys[key]) continue;

    var item = approvals[key];
```

```
    if (!byAccount[item.account_id]) byAccount[item.account_id] = [];
    byAccount[item.account_id].push({ key: key, item: item });
  }

  // Map labelled accounts for selection
  var accIter = getLabeledAccounts_(ACCOUNT_LABEL);
  var accountMap = {};
  while (accIter.hasNext()) {
    var a = accIter.next();
    accountMap[a.getCustomerId()] = a;
  }

  var excludedNow = 0;
  var skippedAllowlist = 0;
  var invalidSkipped = 0;
  var appSkipped = 0;
  var errors = 0;

  for (var accId in byAccount) {
    if (!byAccount.hasOwnProperty(accId)) continue;

    var accObj = accountMap[accId];
    if (!accObj) continue;

    AdsManagerApp.select(accObj);
    var list = getOrCreateExcludedPlacementList_();

    var items = byAccount[accId];
    for (var j = 0; j < items.length; j++) {
      var key = items[j].key;
      var it = items[j].item;

      // Apps cannot be excluded via this script
      if (String(it.flag_reasons || "").indexOf("app_placement") !== -1 ||
          looksLikeMobileAppPlacement_(it.placement_raw, it.display_name)) {
        appSkipped++;
        Logger.log("Skipping app placement (not supported in Scripts): " + it.placement_raw);
        continue;
      }

      var formatted = formatWebsiteDomainForExclusion_(it.placement_raw);
      if (!formatted.ok) {
        invalidSkipped++;
        Logger.log("Skipping invalid placement: " + it.placement_raw + " . Reason: " +
formatted.reason);
        continue;
      }
```

```
      var placementToExclude = String(formatted.value || "").trim().toLowerCase();
      if (!placementToExclude) {
        invalidSkipped++;
        continue;
      }

      if (allowlist[placementToExclude]) {
        skippedAllowlist++;
        continue;
      }

      try {
        list.addExcludedPlacement(placementToExclude);
        excludedKeys[key] = true;
        excludedNow++;
      } catch (e) {
        errors++;
        Logger.log("Failed to exclude " + placementToExclude + " in " + accId + ": " + e);
      }
    }
  }

  saveJsonMapProp_(PROP_EXCLUDED_KEYSET, excludedKeys);

  return {
    approvals_total: approvalKeys.length,
    excluded_now: excludedNow,
    allowlist_skipped: skippedAllowlist,
    app_skipped: appSkipped,
    invalid_skipped: invalidSkipped,
    errors: errors,
    excludedKeysMap: excludedKeys
  };
}

function formatWebsiteDomainForExclusion_(placementRaw) {
  var raw = String(placementRaw || "").trim();
  if (!raw) return { ok: false, value: "", reason: "empty" };

  var v = raw.toLowerCase().replace(/^https?:\/\//i, "").replace(/^www\./i, "");
  v = v.split(/[?#]/)[0];
  v = v.split("/")[0];

  if (v.indexOf(" ") !== -1 || v.indexOf(".") === -1) {
    return { ok: false, value: v, reason: "not_domain_like" };
  }
  return { ok: true, value: v, reason: "domain" };
}
```

```
function getOrCreateExcludedPlacementList_() {
  var safeName = EXCLUDED_LIST_NAME.replace(/'/g, "\\'");
  var iter = AdsApp.excludedPlacementLists()
    .withCondition("Name = '" + safeName + "'")
    .get();

  if (iter.hasNext()) return iter.next();

  return AdsApp.newExcludedPlacementListBuilder()
    .withName(EXCLUDED_LIST_NAME)
    .build()
    .getResult();
}

/** ==================== HISTORY (DE-DUPED) ==================== */

function appendHistory_(historySheet, flaggedRows, runTs, excludedKeysMap) {
  var headers = [
    "first_seen_ts",
    "run_ts",
    "status",
    "account_id",
    "account_name",
    "campaign_id",
    "campaign_name",
    "channel",
    "placement",
    "placement_normalized",
    "display_name",
    "impr",
    "clicks",
    "ctr_pct",
    "cost",
    "flag_reasons",
    "row_key"
  ];

  ensureHeaders_(historySheet, headers);

  var processedKeys = loadJsonMapProp_(PROP_PROCESSED_KEYSET);

  var out = [];
  var appended = 0;

  for (var i = 0; i < flaggedRows.length; i++) {
    var r = flaggedRows[i];
    var key = r.row_key;
```

```
    if (!key) continue;
    if (processedKeys[key]) continue;

    var status = excludedKeysMap[key] ? "excluded" : "flagged";

    out.push([
      runTs,
      r.run_ts,
      status,
      r.account_id,
      r.account_name,
      r.campaign_id,
      r.campaign_name,
      r.channel,
      r.placement,
      r.placement_normalized,
      r.display_name,
      r.impr,
      r.clicks,
      r.ctr_pct,
      r.cost,
      r.flag_reasons,
      r.row_key
    ]);

    processedKeys[key] = true;
    appended++;
  }

  if (out.length) {
    var startRow = historySheet.getLastRow() + 1;
    historySheet.getRange(startRow, 1, out.length, headers.length).setValues(out);
  }

  saveJsonMapProp_(PROP_PROCESSED_KEYSET, processedKeys);

  return { appended: appended };
}

function ensureHeaders_(sheet, headers) {
  if (sheet.getLastRow() === 0) {
    sheet.getRange(1, 1, 1, headers.length).setValues([headers]);
    sheet.setFrozenRows(1);
    return;
  }
  var current = sheet.getRange(1, 1, 1, headers.length).getValues()[0];
  if (current.join("|") !== headers.join("|")) {
    sheet.clear();
```

```
    sheet.getRange(1, 1, 1, headers.length).setValues([headers]);
    sheet.setFrozenRows(1);
  }
}

/** ==================== ALLOWLIST ==================== */

function loadAllowlist_(sheet) {
  var allow = {};
  if (!sheet || sheet.getLastRow() === 0) return allow;

  var values = sheet.getDataRange().getValues();
  for (var r = 0; r < values.length; r++) {
    var v = String(values[r][0] || "").trim().toLowerCase();
    if (!v) continue;
    if (v === "placement_normalized" || v === "allowlist") continue;
    allow[v] = true;
  }
  return allow;
}

/** ==================== EMAIL ==================== */

function sendSummaryEmail_(sheetUrl, runTs, flaggedCount, approvalsMap, applyResult,
historyResult) {
  var recipients = String(ALERT_EMAILS || "")
    .split(",")
    .map(function(s) { return s.trim(); })
    .filter(function(s) { return s; });

  if (!recipients.length) throw new Error("ALERT_EMAILS is empty.");

  var subject = "Placement review update (" + runTs + ")";
  var body =
    "Hi,\n\n" +
    "Your MCC placement script ran.\n\n" +
    "This run:\n" +
    "- Flagged placements (queue): " + flaggedCount + "\n" +
    "- Approvals in queue (exclude=yes): " + Object.keys(approvalsMap).length + "\n" +
    "- Newly excluded now: " + applyResult.excluded_now + "\n" +
    "- Skipped due to allowlist: " + applyResult.allowlist_skipped + "\n" +
    "- Skipped because placement is a mobile app (Scripts limitation): " +
applyResult.app_skipped + "\n" +
    "- Skipped due to invalid format: " + applyResult.invalid_skipped + "\n" +
    "- Exclusion errors: " + applyResult.errors + "\n" +
    "- New history rows appended: " + historyResult.appended + "\n\n" +
    "Sheet:\n" + sheetUrl + "\n\n" +
    "— Coby Agency Placement Script v2";
```

```
  MailApp.sendEmail(recipients.join(","), subject, body);
}

/** ====================== HELPERS ====================== */

function ensureSheet_(ss, name) {
  var sheet = ss.getSheetByName(name);
  if (!sheet) sheet = ss.insertSheet(name);
  return sheet;
}

function getLabeledAccounts_(labelName) {
  return AdsManagerApp.accounts()
    .withCondition("LabelNames CONTAINS '" + labelName + "'")
    .get();
}

function normalizePlacement_(placement) {
  if (!placement) return "";
  var p = String(placement).trim().toLowerCase();
  p = p.replace(/^https?:\/\//, "");
  p = p.replace(/^www\./, "");
  p = p.split("#")[0];
  p = p.split("?")[0];
  p = p.split("/")[0];
  p = p.replace(/:\d+$/, "");
  return p;
}

function endsWithAny_(str, suffixes) {
  if (!str) return false;
  for (var i = 0; i < suffixes.length; i++) {
    if (str.endsWith(suffixes[i])) return true;
  }
  return false;
}

function looksLikeMobileAppPlacement_(placementRaw, displayName) {
  var raw = String(placementRaw || "").trim().toLowerCase();
  var dn = String(displayName || "").trim().toLowerCase();

  if (dn.indexOf("google play") !== -1 || dn.indexOf("mobile app:") === 0 || dn.indexOf("app
store") !== -1) {
    return true;
  }

  if (raw && raw.indexOf(".") !== -1 && raw.indexOf("/") === -1 && raw.indexOf(" ") === -1) {
```

```
    var webish = raw.endsWith(".com") || raw.endsWith(".net") || raw.endsWith(".org") ||
raw.endsWith(".co") || raw.endsWith(".io");
    if (!webish && (raw.indexOf("com.") === 0 || raw.indexOf("net.") === 0 ||
raw.indexOf("org.") === 0)) {
      return true;
    }
  }

  if (raw.indexOf("2-") === 0 && raw.indexOf("/") === -1) return true;

  return false;
}

function indexMap_(headers, names) {
  var map = {};
  for (var i = 0; i < names.length; i++) {
    var name = names[i];
    var idx = -1;
    for (var c = 0; c < headers.length; c++) {
      if (String(headers[c]).trim() === name) {
        idx = c;
        break;
      }
    }
    if (idx === -1) throw new Error("Missing required column: " + name);
    map[name] = idx;
  }
  return map;
}

function loadJsonMapProp_(key) {
  var props = PropertiesService.getScriptProperties();
  var raw = props.getProperty(key);
  if (!raw) return {};
  try {
    var obj = JSON.parse(raw);
    if (obj && typeof obj === "object") return obj;
  } catch (e) {}
  return {};
}

function saveJsonMapProp_(key, obj) {
  var props = PropertiesService.getScriptProperties();
  var json = JSON.stringify(obj);

  if (json.length > 800000) {
    Logger.log("Warning: property store too large. Resetting key store: " + key);
    props.deleteProperty(key);
```

```
    return;
  }

  props.setProperty(key, json);
}
```