```
/**
 * @title  PMax Search Terms Exporter (Single Account + MCC wrapper)
 * @version v1.4.2 · 2025-10-01
 * @author Yana Lyashenko · ADW Service
 * @contact · site: https://adwservice.com.ua/uk/ · YouTube: @AdwserviceCompany
 * @copyright © 2025 ADW Service
 * @license  ADW Service License — non-exclusive, non-transferable. Redistribution,
publication or resale is
 *          prohibited without prior written consent from ADW Service. Attribution required.
 * @support  Paid maintenance & feature requests available. Write to the contact above.
 * безкоштовно для некомерційного використання / з посиланням на автора”
 * Changelog:
 *  - clearSheet(): no residual data across runs
 *  - GAQL: segments.ad_network_type in SELECT (when used in WHERE)
 *  - CategoryInsights: added formatters + alias fmtImpRow→fmtImprRow



/* ===== CONFIG ===== */
const SHEET_URL = ''; // твій шит або залиш пустим - скрипт сам створить - шукай лог
const DAYS = 7;                    // період аналізу (без сьогодні)
const USE_ALL_CONV = false;            // true -> all_conversions/all_conversions_value
const BRAND_SENSITIVITY = 'normal';     // 'normal' | 'strict'

const LIMITS = {
  minImpr: 1,
  minClicksForCPC: 1,
  minClicksForCVR: 1,
  // NegativeIdeas:
  neg_minImpr: 50,
  neg_minCost: 90,
  neg_maxConv: 0,
  neg_lowCTR: 0.01,
  neg_lowROAS: 0.5
};
const NEGATIVE_COST_GATE = true;

const TOP_N = 10;
const HOT_ROAS_THRESHOLD = 0.6;
const HOT_MIN_COST = 20;

const ENFORCE_BOOK_RENAME = true;
const BOOK_NAME_PREFIX    = 'ADW Service — ';
const WATERMARK_TEXT      = 'ADW Service · adwservice.com.ua';
const WATERMARK_FONT_SIZE = 9;
const WATERMARK_FONT_COLOR= '#AAAAAA';
const WATERMARK_OFFSET_COL= 0;
```

```
const FORCE_BRAND_SUMMARY_REBUILD = true;
const MOVE_SUMMARY_TO_END         = true;
const HIDE_BRAND_SUMMARY          = false;
const PROTECT_BRAND_SUMMARY       = true;
const META_SHEET_NAME             = '__adw_meta';
const BRAND_SIGNATURE             = 'ADW_SERVICE_BRAND_v1';

const SKIP_WATERMARK_TABS = new Set([META_SHEET_NAME]);

const BRAND_BOOK_TITLE   = 'AI-аналіз пошукових запитів PMax — ADW Service';
const BRAND_LOGO_URL     =
'https://drive.google.com/uc?export=download&id=1SysNQV6WQ9mS-BbpHf7MOEMu-2esE
Ui2';
const BRAND_LOGO_MARK    = 'adw_logo';
const BRAND_LOGO_W       = 350;   // px
const BRAND_LOGO_H       = 58;    // px
const BRAND_LOGO_COL_OFF = 2;

/* Sheet tabs */
const TAB_SEARCH_TERMS    = 'PMax_SearchTerms';
const TAB_BRAND_SPLIT     = 'BrandSplit';
const TAB_CATEGORY_SPLIT  = 'CategorySplit';
const TAB_NEG_IDEAS       = 'NegativeIdeas';
const TAB_CONFIG          = 'Config';
const TAB_CATEGORIES      = 'Categories';
const TAB_HOT_CATEGORIES  = 'HotCategories';
const TAB_CATEGORY_INS    = 'CategoryInsights';
const TAB_BRAND_SUMMARY   = 'ADWService Summary';

const LOOKBACK_DAYS = (typeof DAYS !== 'undefined') ? DAYS : 7;

/* ===== MAIN ===== */
function main () {
  const tz = AdsApp.currentAccount().getTimeZone();
  const { start, end } = dateRangeNDays(LOOKBACK_DAYS, tz);

  const sheet = getOrCreateSheet();

  // aliases so it works whether you had ensureConfigTab or ensureConfigTabs
  if (typeof ensureConfigTabs === 'function') ensureConfigTabs(sheet);
  else if (typeof ensureConfigTab  === 'function') ensureConfigTab(sheet);
  else ensureConfigTab(sheet);

  if (typeof ensureCategoriesTab === 'function')  ensureCategoriesTab(sheet);

  let rows = [];
  try {
    rows = fetchPMaxTerms_UsingCampaignSTV(start, end);
```

```
    Logger.log('Fetched ' + rows.length + ' via campaign_search_term_view');
  } catch (e) {
    Logger.log('campaign_search_term_view failed; fallback to search_term_view: ' + e);
    rows = fetchPMaxTerms_UsingSTV(start, end);
    Logger.log('Fetched ' + rows.length + ' via search_term_view');
  }

  const brandList = getBrandList(sheet);
  const catDict   = getCategoryDict(sheet);

  const computed = rows.map(enrichMetrics).map(r => ({
    ...r,
    brand: isBrand(r.search_term, brandList, BRAND_SENSITIVITY, r.campaign_name),
    category: categorize(r.search_term, catDict)
  }));

  // ==== OUTPUT (з очищенням) ====
  writeSearchTermsTable(sheet, computed, { start, end });
  writeBrandSplit(sheet,   computed, { start, end });
  safeBuild(() => buildBrandCharts(sheet));

  const catAgg = writeCategorySplit(sheet, computed, { start, end });
  safeBuild(() => buildCategoryCharts(sheet));

  writeNegativeIdeas(sheet, computed);
  writeHotCategories(sheet, catAgg, computed);
  writeCategoryInsights(sheet, catAgg);
  safeBuild(() => buildCategoryInsightsCharts(sheet));

  if (typeof applyBranding === 'function') applyBranding(sheet);
  ensureBrandSummarySheet(sheet);

  Logger.log('Done.');
}


const BRAND_INFO = {
  name: 'ADW Service — AI-аналітика PMax',
  site: 'https://adwservice.com.ua/uk/',
  youtube: 'https://www.youtube.com/@AdwserviceCompany',
  logoUrl:
'https://drive.google.com/uc?export=download&id=1SysNQV6WQ9mS-BbpHf7MOEMu-2esE
Ui2',
  logoWidth: 350,
  logoHeight: 58
};

function ensureBrandSummarySheet(ss) {
```

```
  // технічний маячок
  ensureMetaSheet_(ss);

  let sh = ss.getSheetByName(TAB_BRAND_SUMMARY);
  if (!sh) sh = ss.insertSheet(TAB_BRAND_SUMMARY);

  // тримати праворуч
  if (MOVE_SUMMARY_TO_END) { ss.setActiveSheet(sh);
ss.moveActiveSheet(ss.getSheets().length); }

  // якщо вимкнули форс-ребілд, все одно гарантуємо базову структуру
  if (FORCE_BRAND_SUMMARY_REBUILD) {
    sh.setHiddenGridlines(true);
    sh.clearContents(); sh.clearFormats();
    sh.setRowHeights(1, 30, 22);
    for (let c = 1; c <= 8; c++) sh.setColumnWidth(c, c === 1 ? 260 : 180);

    const TOP=1, LEFT=1, ROWS=18, COLS=8;
    const rng = sh.getRange(TOP, LEFT, ROWS, COLS);
    rng.clearContent().clearFormat();

    // Заголовок без «автостворено скриптом»
    const title = sh.getRange(TOP, LEFT, 1, COLS);
    title.merge();
    title.setValue(BRAND_INFO.name)
         .setFontSize(14).setFontWeight('bold').setHorizontalAlignment('center');

    const body = [
      ['Про цей файл',''],
      ['• Джерело даних','Google Ads Scripts (PMax Search Terms)'],
      ['• Призначення','Огляд запитів/категорій, брендовий/небрендовий спліт,
негатив-ідеї'],
      ['',''],
      ['Контакти ADW Service',''],
      ['Сайт',   BRAND_INFO.site   ? `=HYPERLINK("${BRAND_INFO.site}","Перейти")` :
'—'],
      ['YouTube',BRAND_INFO.youtube?
`=HYPERLINK("${BRAND_INFO.youtube}","Канал")`  : '—'],
      ['E-mail', BRAND_INFO.email || '—'],
      ['','']
    ];
    sh.getRange(3,1,body.length,2).setValues(body);
    sh.getRange(3,1,body.length,1).setFontWeight('bold');


    try { (sh.getImages && sh.getImages() || []).forEach(img => img.remove()); } catch(e){}
    try {
      const blob = UrlFetchApp.fetch(BRAND_INFO.logoUrl).getBlob();
```

```
      const img = sh.insertImage(blob, 5, 5);
      try { img.setWidth(BRAND_INFO.logoWidth).setHeight(BRAND_INFO.logoHeight); }
catch(_){}
      try { img.setAltTextTitle(BRAND_LOGO_MARK); } catch(_){}
    } catch(_){}

    const foot = sh.getRange(TOP+ROWS, LEFT, 1, COLS);
    foot.merge();
    foot.setValue('© ADW Service')
        .setHorizontalAlignment('center').setFontSize(10).setFontColor('#666');
  }


  if (PROTECT_BRAND_SUMMARY) {
    try {
      const protections = sh.getProtections(SpreadsheetApp.ProtectionType.SHEET) || [];
      protections.forEach(p => p.remove());
      const p = sh.protect().setDescription('ADW Service branding — do not edit');
      p.setWarningOnly(true);
    } catch(e) {}
  }


  if (HIDE_BRAND_SUMMARY) { try { sh.hideSheet(); } catch(e) {} }
}


function fetchPMaxTerms_UsingCampaignSTV(start, end) {
  const gaql = `
    SELECT
      campaign.id, campaign.name,
      metrics.impressions, metrics.clicks, metrics.cost_micros,
      metrics.conversions, metrics.conversions_value,
      metrics.all_conversions, metrics.all_conversions_value,
      campaign_search_term_view.search_term,
      segments.search_term_match_type,
      segments.ad_network_type
    FROM campaign_search_term_view
    WHERE campaign.advertising_channel_type = PERFORMANCE_MAX
      AND segments.date BETWEEN '${start}' AND '${end}'
  `;
  return iterateRows(gaql);
}

function fetchPMaxTerms_UsingSTV(start, end) {
  const gaql = `
    SELECT
      campaign.id, campaign.name,
```

```
      metrics.impressions, metrics.clicks, metrics.cost_micros,
      metrics.conversions, metrics.conversions_value,
      metrics.all_conversions, metrics.all_conversions_value,
      search_term_view.search_term,
      segments.search_term_match_type,
      segments.ad_network_type
    FROM search_term_view
    WHERE campaign.advertising_channel_type = PERFORMANCE_MAX
      AND segments.date BETWEEN '${start}' AND '${end}'
  `;
  return iterateRows(gaql);
}

function iterateRows(gaql) {
  const it = AdsApp.search(gaql);
  const out = [];
  while (it.hasNext()) {
    const r = it.next();
    const conv = USE_ALL_CONV ? (Number(r.metrics.allConversions)||0) :
(Number(r.metrics.conversions)||0);
    const convValue = USE_ALL_CONV ? (Number(r.metrics.allConversionsValue)||0) :
(Number(r.metrics.conversionsValue)||0);
    let st = '';
    if (r.campaignSearchTermView && r.campaignSearchTermView.searchTerm) st =
r.campaignSearchTermView.searchTerm;
    else if (r.searchTermView && r.searchTermView.searchTerm)          st =
r.searchTermView.searchTerm;
    else                                                               st = (r.searchTerm || '');

    out.push({
      campaign_id: r.campaign.id,
      campaign_name: r.campaign.name,
      impressions: Number(r.metrics.impressions)||0,
      clicks: Number(r.metrics.clicks)||0,
      cost: microsToCurrency(r.metrics.costMicros),
      conversions: conv,
      conv_value: convValue,
      search_term: safeText(st),
      match_type: (r.segments && r.segments.searchTermMatchType) ?
r.segments.searchTermMatchType : '',
      network: (r.segments && r.segments.adNetworkType) ? r.segments.adNetworkType : ''
    });
  }
  return out.filter(x => x.impressions >= LIMITS.minImpr);
}

/* ===== PROCESSING ===== */
function enrichMetrics(row){
```

```
  const impr=row.impressions||0, clicks=row.clicks||0, cost=row.cost||0,
conv=row.conversions||0, val=row.conv_value||0;
  const ctr  = impr>0 ? clicks/impr : 0;
  const cpc  = clicks>=LIMITS.minClicksForCPC ? cost/clicks : 0;
  const cvr  = clicks>=LIMITS.minClicksForCVR ? conv/clicks : 0;
  const cpa  = conv>0 ? cost/conv : 0;
  const aov  = conv>0 ? val/conv : 0;
  const roas = cost>0 ? val/cost : 0;
  return {...row, ctr, cpc, cvr, cpa, aov, roas};
}

function isBrand(term, brandList, mode, campaignName){
  if (!term) return false;
  const t = normalize(term);
  const base = brandList.map(normalize).filter(Boolean);
  if (base.some(b => t.includes(b))) return true;
  if (mode==='strict' && campaignName){
    const cn = normalize(campaignName);
    return base.some(b => cn.includes(b));
  }
  return false;
}

function categorize(term, dict){
  const t = normalize(term);
  for (const d of dict){
    for (const p of d.patterns){
      if (t.includes(p)) return d.name;
    }
  }
  return 'Uncategorized';
}


function getOrCreateSheet(){
  if (SHEET_URL && SHEET_URL.trim()) return SpreadsheetApp.openByUrl(SHEET_URL);
  const ss = SpreadsheetApp.create(`PMax Search Terms —
${AdsApp.currentAccount().getName()}`);
  Logger.log(`Created new sheet: ${ss.getUrl()}`);
  return ss;
}

function ensureConfigTab(ss){
  const sh = ss.getSheetByName(TAB_CONFIG) || ss.insertSheet(TAB_CONFIG);
  if (sh.getLastRow() === 0){
    sh.getRange(1,1,1,1).setValues([['brand_terms']]);
  }
}
```

```
// alias (на випадок, якщо десь викликалось ensureConfigTabs)
function ensureConfigTabs(ss){ ensureConfigTab(ss); }

function ensureCategoriesTab(ss){
  const sh = ss.getSheetByName(TAB_CATEGORIES) ||
ss.insertSheet(TAB_CATEGORIES);
  if (sh.getLastRow() === 0){
    sh.getRange(1,1,1,2).setValues([[ 'category', 'patterns (comma-separated)' ]]);
    const defaults = [
      ['Brand Own',           ''],
      ['Конкурент',           ''],
      ['Gift / For Her',      ''],
      ['Transactional (Buy)', 'купи, купит, купить, купити, замовит, заказать'],
      ['Price-seeking',       'недорого, дешево, дешев, ціна, цена'],
      ['Uncategorized',       '(не чіпай; це фолбек від скрипта)']
    ];
    sh.getRange(2,1,defaults.length,2).setValues(defaults);
    sh.setFrozenRows(1);
    sh.autoResizeColumns(1, 2);
  }
}



function getBrandList(ss){
  const sh = ss.getSheetByName(TAB_CONFIG);
  if (!sh) return [];
  const n = Math.max(sh.getLastRow()-1,0);
  if (n===0) return [];
  return sh.getRange(2,1,n,1).getValues()
           .map(r => (r[0]||'').toString().trim())
           .filter(Boolean);
}


function getCategoryDict(ss){
  const sh = ss.getSheetByName(TAB_CATEGORIES);
  if (!sh) return [];
  const n = Math.max(sh.getLastRow()-1,0);
  if (n===0) return [];
  const rng = sh.getRange(2,1,n,2).getValues();
  const dict = [];
  for (const [nameRaw, patsRaw] of rng){
    const name = (nameRaw||'').toString().trim();
    const pats = (patsRaw||'').toString().split(',')
      .map(s => normalize(s)).filter(Boolean);
    if (name && pats.length) dict.push({ name, patterns: pats });
```

```
  }
  return dict;
}


function removeFilterIfAny(sheet) {
  try {
    const f = sheet.getFilter && sheet.getFilter();
    if (f) f.remove();
  } catch (e) {
    Logger.log('removeFilterIfAny: ' + e);
  }
}

function clearOrCreate(ss, name){
  const sh = ss.getSheetByName(name) || ss.insertSheet(name);
  removeFilterIfAny(sh);
  sh.clearContents();
  return sh;
}


function clearSheet_(sh) {
  try { const f = sh.getFilter && sh.getFilter(); if (f) f.remove(); } catch(e) {}
  try { (sh.getCharts && sh.getCharts() || []).forEach(c => sh.removeChart(c)); } catch(e) {}
  sh.clearContents();
  sh.clearFormats();
}
function ensureMetaSheet_(ss){
  let meta = ss.getSheetByName(META_SHEET_NAME);
  if (!meta) {
    meta = ss.insertSheet(META_SHEET_NAME);
    try { meta.hideSheet(); } catch(e) {}
  }
  if (meta.getLastRow() === 0) {
    meta.getRange(1,1,1,2).setValues([['signature','timestamp']]);
  }
  // оновлюємо «сигнатуру» щоразу
  const row = Math.max(2, meta.getLastRow()+1);
  meta.getRange(row,1,1,2).setValues([[BRAND_SIGNATURE, new Date()]]);
  try { meta.hideSheet(); } catch(e) {}
}
function safeRenameBook_(ss, prefix){
  try {
    const acc = AdsApp.currentAccount().getName();
    const want = (prefix || '') + acc;
    if (ss.getName() !== want) ss.rename(want);
  } catch(e){ Logger.log('safeRenameBook_: '+e); }
```

```
}


function addWatermarkToSheet_(sh){
  try {
    if (SKIP_WATERMARK_TABS.has(sh.getName())) return;
    const lastCol = Math.max(1, sh.getMaxColumns() - WATERMARK_OFFSET_COL);
    const cell = sh.getRange(1, lastCol, 1, 1);
    cell.setValue(WATERMARK_TEXT)
        .setFontSize(WATERMARK_FONT_SIZE)
        .setFontColor(WATERMARK_FONT_COLOR)
        .setHorizontalAlignment('right')
        .setVerticalAlignment('top')
        .setWrap(false);

    try {

  (sh.getProtections(SpreadsheetApp.ProtectionType.RANGE) || [])
    .filter(p => p.getRange && p.getRange().getA1Notation() === cell.getA1Notation())
    .forEach(p => p.remove());


  const rp = cell.protect().setDescription('ADW watermark cell');
  rp.setWarningOnly(true);
} catch(e){}
  } catch(e){ Logger.log('addWatermarkToSheet_: '+e); }
}


// Повертає існуючий аркуш по імені (ігнорує пробіли/реєстр) або безпечно створює
новий
function ensureSheetByName_(ss, name) {
  // 1) прямий пошук
  var sh = ss.getSheetByName(name);
  if (sh) return sh;

  // 2) толерантний пошук (trim + toLowerCase)
  var target = name.trim().toLowerCase();
  var sheets = ss.getSheets();
  for (var i = 0; i < sheets.length; i++) {
    var nm = (sheets[i].getName() || '').toString().trim().toLowerCase();
    if (nm === target) return sheets[i];
  }

  // 3) створюємо без імені та пробуємо перейменувати
  sh = ss.insertSheet();
  try { sh.setName(name); }
  catch (e) {
```

```
    // якщо за цей час уже створився аркуш з таким ім'ям — знайдемо його і
використаємо
    var again = ss.getSheetByName(name);
    if (again) { ss.deleteSheet(sh); return again; }
    throw e; // інша помилка — нехай летить
  }
  return sh;
}


function writeSearchTermsTable(ss, rows, {start,end}){
  var sh = ensureSheetByName_(ss, TAB_SEARCH_TERMS);
  clearSheet_(sh); // <-- додано

  const headers=['Campaign ID','Campaign Name','Search Term','Match
Type','Brand?','Category','Impr','Clicks','CTR','CPC','Cost','Conv','Conv
Value','CVR','CPA','AOV','ROAS'];
  const data=rows.map(r=>[
    r.campaign_id, r.campaign_name, r.search_term, r.match_type,
    r.brand?'Brand':'Non-Brand', r.category||'Uncategorized',
    r.impressions, r.clicks, pct(r.ctr), money(r.cpc), money(r.cost),
    r.conversions, money(r.conv_value), pct(r.cvr), money(r.cpa), money(r.aov), dec1(r.roas)
  ]);
  if (data.length){
    sh.getRange(1,1,1,headers.length).setValues([headers]);
    sh.getRange(2,1,data.length,headers.length).setValues(data);
    sh.getRange(1,1,Math.max(2,data.length+1),headers.length).createFilter();
  } else {
    sh.getRange(1,1).setValue(`No PMax search terms for ${start} to ${end}.`);
  }
}

function writeBrandSplit(ss, rows, {start,end}){
  var sh = ensureSheetByName_(ss, TAB_BRAND_SPLIT);

  clearSheet_(sh); // <-- додано

  const agg = aggregateByBrand(rows);
  const headers=['Type','Impr','Clicks','CTR','CPC','Cost','Conv','Conv
Value','CVR','CPA','AOV','ROAS'];
  const body=['Brand','Non-Brand'].map(type=>{
    const m=agg[type]||emptyAgg();
    return
[type,m.impr,m.clicks,pct(m.ctr),money(m.cpc),money(m.cost),m.conv,money(m.val),pct(m.c
vr),money(m.cpa),money(m.aov),dec1(m.roas)];
  });
  sh.getRange(1,1,1,headers.length).setValues([headers]);
  sh.getRange(2,1,body.length,headers.length).setValues(body);
```

```
  sh.getRange(1,1,body.length+1,headers.length).createFilter();
  sh.getRange(1, headers.length+2).setValue('Date range:');
  sh.getRange(1, headers.length+3).setValue(`${start} — ${end}`);
}

function buildBrandCharts(ss){
  const sh=ss.getSheetByName(TAB_BRAND_SPLIT);
  const lastRow=Math.max(2,sh.getLastRow());
  (sh.getCharts()||[]).forEach(c=>sh.removeChart(c));
  const startCol=headersEndCol(sh)+1;
  const c1=sh.newChart().setChartType(Charts.ChartType.COLUMN)

.addRange(sh.getRange(1,1,lastRow,1)).addRange(sh.getRange(1,6,lastRow,1)).addRange(
sh.getRange(1,8,lastRow,1))
    .setOption('title','Brand vs Non-Brand: Cost vs Conv
Value').setPosition(2,startCol,0,0).build();
  sh.insertChart(c1);
  const c2=sh.newChart().setChartType(Charts.ChartType.COLUMN)

.addRange(sh.getRange(1,1,lastRow,1)).addRange(sh.getRange(1,4,lastRow,1)).addRange(
sh.getRange(1,9,lastRow,1))
    .setOption('title','Brand vs Non-Brand: CTR & CVR').setPosition(20,startCol,0,0).build();
  sh.insertChart(c2);
  const c3=sh.newChart().setChartType(Charts.ChartType.COLUMN)
    .addRange(sh.getRange(1,1,lastRow,1)).addRange(sh.getRange(1,12,lastRow,1))
    .setOption('title','Brand vs Non-Brand: ROAS').setPosition(38,startCol,0,0).build();
  sh.insertChart(c3);
}

function writeCategorySplit(ss, rows, {start,end}){
  var sh = ensureSheetByName_(ss, TAB_CATEGORY_SPLIT);
  clearSheet_(sh); // <-- додано

  const map=new Map();
  for (const r of rows){
    const k=r.category||'Uncategorized';
    if (!map.has(k)) map.set(k, emptyAgg());
    const a=map.get(k);
    a.impr+=r.impressions||0;
    a.clicks+=r.clicks||0;
    a.cost+=r.cost||0;
    a.conv+=r.conversions||0;
    a.val+=r.conv_value||0;
  }
  const cats=Array.from(map.keys()).sort();
  cats.forEach(c=>recalcAgg(map.get(c)));
```

```
  const headers=['Category','Impr','Clicks','CTR','CPC','Cost','Conv','Conv
Value','CVR','CPA','AOV','ROAS'];
  const body=cats.map(cat=>{
    const m=map.get(cat)||emptyAgg();
    return
[cat,m.impr,m.clicks,pct(m.ctr),money(m.cpc),money(m.cost),m.conv,money(m.val),pct(m.cvr
),money(m.cpa),money(m.aov),dec1(m.roas)];
  });

  sh.getRange(1,1,1,headers.length).setValues([headers]);
  if (body.length){
    sh.getRange(2,1,body.length,headers.length).setValues(body);
    sh.getRange(1,1,Math.max(2,body.length+1),headers.length).createFilter();
  } else {
    sh.getRange(2,1).setValue('No data.');
  }
  sh.getRange(1, headers.length+2).setValue('Date range:');
  sh.getRange(1, headers.length+3).setValue(`${start} — ${end}`);
  return map;
}

function buildCategoryCharts(ss){
  const sh=ss.getSheetByName(TAB_CATEGORY_SPLIT);
  const lastRow=Math.max(2, sh.getLastRow());
  (sh.getCharts()||[]).forEach(c=>sh.removeChart(c));
  const startCol=headersEndCol(sh)+1;
  const cA=sh.newChart().setChartType(Charts.ChartType.COLUMN)
    .addRange(sh.getRange(1,1,lastRow,1)).addRange(sh.getRange(1,6,lastRow,1))
    .setOption('title','Categories: Cost').setPosition(2,startCol,0,0).build();
  sh.insertChart(cA);
  const cB=sh.newChart().setChartType(Charts.ChartType.COLUMN)
    .addRange(sh.getRange(1,1,lastRow,1)).addRange(sh.getRange(1,12,lastRow,1))
    .setOption('title','Categories: ROAS').setPosition(24,startCol,0,0).build();
  sh.insertChart(cB);
}

function writeNegativeIdeas(ss, rows){
  var sh = ensureSheetByName_(ss, TAB_NEG_IDEAS);
  clearSheet_(sh); // <-- додано

  const headers=['Search Term','Reason','Suggested
Match','Brand?','Category','Impr','Clicks','Cost','Conv','CTR','ROAS'];
  const ideas=[];
  for (const r of rows){
    const lowCtr = r.ctr>0 ? r.ctr < LIMITS.neg_lowCTR : false;
    const costly = r.cost >= LIMITS.neg_minCost;
    const noConv = (r.conversions||0) <= LIMITS.neg_maxConv;
    const lowRoas = r.cost>0 ? r.roas < LIMITS.neg_lowROAS : false;
```

```
    const passCostGate = NEGATIVE_COST_GATE ? costly : (costly || lowCtr || lowRoas);
    if (r.impressions >= LIMITS.neg_minImpr && noConv && passCostGate){
      const sugMatch = r.search_term.trim().split(/\s+/).length>1 ? 'PHRASE' : 'BROAD';
      const reason = [
        noConv ? 'Zero conv' : '',
        costly ? `Cost≥${LIMITS.neg_minCost}` : '',
        (!costly && NEGATIVE_COST_GATE) ? '' : (lowCtr ?
`CTR<${(LIMITS.neg_lowCTR*100).toFixed(1)}%` : ''),
        (!costly && NEGATIVE_COST_GATE) ? '' : (lowRoas ?
`ROAS<${LIMITS.neg_lowROAS}` : '')
      ].filter(Boolean).join('; ');
      ideas.push([ r.search_term, reason||'Cost gate', r.brand?'Exact? (brand!)':sugMatch,
                   r.brand?'Brand':'Non-Brand', r.category||'Uncategorized',
                   r.impressions, r.clicks, money(r.cost), r.conversions, pct(r.ctr), dec1(r.roas) ]);
    }
  }
  if (ideas.length){
    sh.getRange(1,1,1,headers.length).setValues([headers]);
    sh.getRange(2,1,ideas.length,headers.length).setValues(ideas);
    sh.getRange(1,1,ideas.length+1,headers.length).createFilter();
  } else {
    sh.getRange(1,1).setValue('No negative keyword ideas matched the thresholds/gate.');
  }
}


function writeHotCategories(ss, catAgg, rows){
  var sh = ensureSheetByName_(ss, TAB_HOT_CATEGORIES);
  clearSheet_(sh); // <-- додано

  const headers=['Category','Cost','Conv','Conv Value','ROAS','Impr','Clicks','Sample Terms
(up to 3)'];
  const arr=Array.from(catAgg.entries()).map(([cat,m])=>({cat,m}));
  let hot=arr.filter(x=> (x.m.cost>=HOT_MIN_COST) && (x.m.roas <
HOT_ROAS_THRESHOLD))
             .sort((a,b)=> b.m.cost - a.m.cost)
             .slice(0,TOP_N);

  if (!hot.length){
    hot = arr.filter(x => x.m.cost >= 5)
             .sort((a,b)=> (a.m.roas||0) - (b.m.roas||0))
             .slice(0, TOP_N);
  }

  const byCat = rows.reduce((acc,r)=>{ const k=r.category||'Uncategorized';
(acc[k]=acc[k]||[]).push(r); return acc; },{});
  const data = hot.map(x=>{
```

```
    const
terms=(byCat[x.cat]||[]).sort((a,b)=>b.cost-a.cost).slice(0,3).map(t=>t.search_term).join(' | ');
    return [ x.cat, money(x.m.cost), x.m.conv, money(x.m.val), dec1(x.m.roas), x.m.impr,
x.m.clicks, terms ];
  });

  sh.getRange(1,1,1,headers.length).setValues([headers]);
  if (data.length){
    sh.getRange(2,1,data.length,headers.length).setValues(data);
    sh.getRange(1,1,Math.max(2,data.length+1),headers.length).createFilter();
  } else {
    sh.getRange(2,1).setValue('No category data.');
  }
}


function fmtImprRow(mode, x) {
  if (mode === 'headers') return ['Category','Impr','Clicks','CTR'];
  return [x.cat, x.impr, x.clicks, x.ctr];
}
function fmtCostRow(mode, x) {
  // ці заголовки важливі для buildCategoryInsightsCharts (master-table детект)
  if (mode === 'headers') return ['Category','Cost','Conv','Conv Value','ROAS'];
  return [x.cat, x.cost, x.conv, x.val, x.roas];
}
function fmtConvRow(mode, x) {
  if (mode === 'headers') return ['Category','Conv','Cost','CPA'];
  return [x.cat, x.conv, x.cost, x.cpa];
}
function fmtValRow(mode, x) {
  if (mode === 'headers') return ['Category','Conv Value','Cost','ROAS'];
  return [x.cat, x.val, x.cost, x.roas];
}
function fmtRoasRow(mode, x) {
  if (mode === 'headers') return ['Category','ROAS','Cost','Conv Value'];
  return [x.cat, x.roas, x.cost, x.val];
}
// alias to cover any old calls
function fmtImpRow(mode, x) { return fmtImprRow(mode, x); }


function writeCategoryInsights(ss, catAgg){
 var sh = ensureSheetByName_(ss, TAB_CATEGORY_INS);
  clearSheet_(sh); // <-- додано

  // перетворення map->масив з готовими метриками
  const arr=Array.from(catAgg.entries()).map(([cat,m])=>{
```

```
    return {cat, impr:m.impr, clicks:m.clicks, ctr:m.ctr, cost:m.cost, conv:m.conv, val:m.val,
cpa:m.cpa, roas:m.roas};
  });

  const sections=[
    {title:`Top ${TOP_N} by Impressions`, key:'impr', fmt: fmtImprRow},
    {title:`Top ${TOP_N} by Cost`,        key:'cost', fmt: fmtCostRow},
    {title:`Top ${TOP_N} by Conversions`, key:'conv', fmt: fmtConvRow},
    {title:`Top ${TOP_N} by Conv Value`,  key:'val',  fmt: fmtValRow},
    {title:`Top ${TOP_N} by ROAS`,        key:'roas', fmt: fmtRoasRow}
  ];

  let row=1;
  sections.forEach(sec=>{
    const top=arr.slice().sort((a,b)=>(b[sec.key]||0)-(a[sec.key]||0)).slice(0,TOP_N);
    const headers=sec.fmt('headers');
    const body=top.map(x=>sec.fmt('row',x));
    sh.getRange(row,1,1,headers.length).setValues([headers]).setFontWeight('bold');
    if (body.length) sh.getRange(row+1,1,body.length,headers.length).setValues(body);
    row += Math.max(2, body.length + 3);
  });
}


function buildCategoryInsightsCharts(ss) {
  const sh = ss.getSheetByName(TAB_CATEGORY_INS);
  if (!sh) return;

  (sh.getCharts() || []).forEach(c => sh.removeChart(c));

  const t = findCategoryMasterTable_(sh);
  if (!t) {
    Logger.log('CategoryInsights: не знайдено таблицю з колонками Cost/Conv
Value/ROAS.');
    return;
  }

  const values  = sh.getRange(t.startRow, t.startCol, t.numRows, t.numCols).getValues();
  const headers = values[0];
  const idx = {
    category: headerIndex_(headers, 'Category'),
    cost:     headerIndex_(headers, 'Cost'),
    conv:     headerIndex_(headers, 'Conv'),
    convVal:  headerIndex_(headers, 'Conv Value'),
    roas:     headerIndex_(headers, 'ROAS'),
  };

  const rows = values.slice(1).filter(r => String(r[idx.category] || '').trim() !== '');
```

```
const byCat = {};
for (const r of rows) {
  const cat  = String(r[idx.category] || '').trim() || 'Uncategorized';
  const cost = num_(r[idx.cost]);
  const conv = num_(r[idx.conv]);
  const val  = num_(r[idx.convVal]);
  const roas = num_(r[idx.roas]);

  if (!byCat[cat]) byCat[cat] = { cost:0, conv:0, val:0, roasW:0, w:0 };
  byCat[cat].cost += cost;
  byCat[cat].conv += conv;
  byCat[cat].val  += val;
  byCat[cat].roasW += cost * (isFinite(roas) ? roas : 0);
  byCat[cat].w    += cost;
}

let data = Object.keys(byCat).map(cat => {
  const d = byCat[cat];
  const roas = d.w > 0 ? d.roasW / d.w : 0;
  return { cat, cost: d.cost, conv: d.conv, val: d.val, roas };
});

const TOP = 10;
data.sort((a,b) => b.cost - a.cost);
data = data.slice(0, TOP);

const outStartRow = t.startRow;
const outStartCol = Math.max(t.startCol + t.numCols + 2, 12);
const table = [['Category','Cost','Conversions','Conv Value','ROAS']]
  .concat(data.map(d => [d.cat, d.cost, d.conv, d.val, d.roas]));
sh.getRange(outStartRow, outStartCol, table.length, table[0].length).setValues(table);

const catRange  = sh.getRange(outStartRow+1, outStartCol+0, data.length, 1);
const costRange = sh.getRange(outStartRow+1, outStartCol+1, data.length, 1);
const convRange = sh.getRange(outStartRow+1, outStartCol+2, data.length, 1);
const valRange  = sh.getRange(outStartRow+1, outStartCol+3, data.length, 1);
const roasRange = sh.getRange(outStartRow+1, outStartCol+4, data.length, 1);

const WIDTH  = 520, HEIGHT = 280;
const base = {
  width: WIDTH, height: HEIGHT,
  chartArea: { left:'10%', width:'80%', height:'70%' },
  legend: { position:'top' },
  annotations: { alwaysOutside:true, textStyle:{ fontSize:10 } },
  hAxis: { slantedText:true, slantedTextAngle:20 }
};
const POS = [
```

```
    { row: outStartRow, col: outStartCol + table[0].length + 2 },
    { row: outStartRow, col: outStartCol + table[0].length + 2 + 8 },
    { row: outStartRow + 18, col: outStartCol + table[0].length + 2 }
  ];

  sh.insertChart(
    sh.newChart()
      .setChartType(Charts.ChartType.COMBO)
      .addRange(catRange).addRange(costRange).addRange(convRange)
      .setPosition(POS[0].row, POS[0].col, 0, 0)
      .setOption('title','TOP Categories — Spend vs Conversions')
      .setOption('series', { 0:{targetAxisIndex:0, type:'bars'}, 1:{targetAxisIndex:1, type:'line'} })
      .setOption('vAxes', { 0:{ title:'Cost' }, 1:{ title:'Conversions' } })
      .setOption('width', base.width).setOption('height', base.height)
      .setOption('chartArea', base.chartArea).setOption('legend', base.legend)
      .setOption('annotations', base.annotations).setOption('hAxis', base.hAxis)
      .build()
  );

  sh.insertChart(
    sh.newChart()
      .setChartType(Charts.ChartType.COMBO)
      .addRange(catRange).addRange(costRange).addRange(valRange)
      .setPosition(POS[1].row, POS[1].col, 0, 0)
      .setOption('title','TOP Categories — Spend vs Conversion Value')
      .setOption('series', { 0:{targetAxisIndex:0, type:'bars'}, 1:{targetAxisIndex:1, type:'line'} })
      .setOption('vAxes', { 0:{ title:'Cost' }, 1:{ title:'Conv. Value' } })
      .setOption('width', base.width).setOption('height', base.height)
      .setOption('chartArea', base.chartArea).setOption('legend', base.legend)
      .setOption('annotations', base.annotations).setOption('hAxis', base.hAxis)
      .build()
  );

  sh.insertChart(
    sh.newChart()
      .setChartType(Charts.ChartType.COMBO)
      .addRange(catRange).addRange(costRange).addRange(roasRange)
      .setPosition(POS[2].row, POS[2].col, 0, 0)
      .setOption('title','TOP Categories — Spend vs ROAS')
      .setOption('series', { 0:{targetAxisIndex:0, type:'bars'}, 1:{targetAxisIndex:1, type:'line'} })
      .setOption('vAxes', { 0:{ title:'Cost' }, 1:{ title:'ROAS' } })
      .setOption('width', base.width).setOption('height', base.height)
      .setOption('chartArea', base.chartArea).setOption('legend', base.legend)
      .setOption('annotations', base.annotations).setOption('hAxis', base.hAxis)
      .build()
  );
}
```

```
/* ===== helpers for CategoryInsights charts ===== */
function headerIndex_(headers, name){
  const q = String(name).toLowerCase();
  for (let i = 0; i < headers.length; i++){
    const h = String(headers[i] || '').toLowerCase();
    if (h === q) return i;
  }
  if (q === 'conv') {
    for (let i = 0; i < headers.length; i++){
      const h = String(headers[i] || '').toLowerCase();
      if (h.includes('conv') && !h.includes('value')) return i;
    }
  }
  if (q === 'conv value') {
    for (let i = 0; i < headers.length; i++){
      const h = String(headers[i] || '').toLowerCase();
      if (h.includes('conv') && h.includes('value')) return i;
    }
  }
  return -1;
}
function num_(v){ const n = Number(v); return isFinite(n) ? n : 0; }
function findCategoryMasterTable_(sh){
  const lastRow = sh.getLastRow();
  const lastCol = sh.getLastColumn();
  for (let r = 1; r <= lastRow; r++){
    const firstCell = String(sh.getRange(r, 1).getValue() || '').trim().toLowerCase();
    if (firstCell === 'category'){
      const headers = sh.getRange(r, 1, 1, lastCol).getValues()[0];
      const hasCost = headers.some(v => String(v).toLowerCase() === 'cost');
      const hasVal  = headers.some(v => String(v).toLowerCase().includes('conv value'));
      const hasRoas = headers.some(v => String(v).toLowerCase() === 'roas');
      if (hasCost && hasVal && hasRoas){
        let rr = r + 1;
        while (rr <= lastRow && String(sh.getRange(rr, 1).getValue() || '').trim() !== '') rr++;
        const numRows = rr - r;
        let cc = 1;
        while (cc <= lastCol && String(headers[cc - 1] || '').trim() !== '') cc++;
        const numCols = cc - 1;
        return { startRow: r, startCol: 1, numRows, numCols };
      }
    }
  }
  return null;
}


function applyBranding(ss){
```

```
  try {
    if (ENFORCE_BOOK_RENAME) safeRenameBook_(ss, BOOK_NAME_PREFIX);
  } catch (e){ Logger.log('rename book: '+e); }

  try {
    if (!BRAND_LOGO_URL || !BRAND_LOGO_URL.trim()) {
      // навіть без лого все одно ставимо водяні знаки
      ss.getSheets().forEach(sh => { try { addWatermarkToSheet_(sh); } catch(_){} });
      return;
    }
    const blob = UrlFetchApp.fetch(BRAND_LOGO_URL).getBlob().setName('logo');

    ss.getSheets().forEach(sh=>{
      try {
        // логотип
        (sh.getImages && sh.getImages() || []).forEach(img=>{
          try { if ((img.getAltTextTitle && img.getAltTextTitle()) === BRAND_LOGO_MARK)
img.remove(); } catch(_){}
        });
        const lastCol = sh.getMaxColumns();
        const col     = Math.max(1, lastCol - BRAND_LOGO_COL_OFF);
        const img     = sh.insertImage(blob, col, 1);
        try { img.setWidth(BRAND_LOGO_W).setHeight(BRAND_LOGO_H); } catch(_){}
        try {
          img.setAltTextTitle(BRAND_LOGO_MARK);
          img.setAnchorCell(sh.getRange(1, col));
          img.setAnchorCellXOffset(8);
          img.setAnchorCellYOffset(6);
        } catch(_){}
      } catch(e){ Logger.log('logo on '+sh.getName()+': '+e); }

      try { addWatermarkToSheet_(sh); } catch(_){}
    });
  } catch (e){ Logger.log('applyBranding: '+e); }
}


function dateRangeNDays(n, tz){
  const now=new Date();
  const today=new Date(Utilities.formatDate(now,tz,'yyyy-MM-dd'));
  const end=new Date(today); end.setDate(end.getDate()-1);
  const start=new Date(end); start.setDate(start.getDate()-n+1);
  return {
    start: Utilities.formatDate(start,tz,'yyyy-MM-dd'),
    end:   Utilities.formatDate(end,tz,'yyyy-MM-dd')
  };
}
```

```
function aggregateByBrand(rows){
  const agg={'Brand':emptyAgg(),'Non-Brand':emptyAgg()};
  for (const r of rows){
    const k=r.brand?'Brand':'Non-Brand';
    const a=agg[k];
    a.impr+=r.impressions||0; a.clicks+=r.clicks||0; a.cost+=r.cost||0; a.conv+=r.conversions||0;
a.val+=r.conv_value||0;
  }
  for (const k in agg) recalcAgg(agg[k]);
  return agg;
}

function emptyAgg(){ return
{impr:0,clicks:0,cost:0,conv:0,val:0,ctr:0,cpc:0,cvr:0,cpa:0,aov:0,roas:0}; }
function recalcAgg(a){ a.ctr=a.impr>0?a.clicks/a.impr:0; a.cpc=a.clicks>0?a.cost/a.clicks:0;
a.cvr=a.clicks>0?a.conv/a.clicks:0; a.cpa=a.conv>0?a.cost/a.conv:0;
a.aov=a.conv>0?a.val/a.conv:0; a.roas=a.cost>0?a.val/a.cost:0; }

function headersEndCol(sh){ return sh.getLastColumn(); }
function microsToCurrency(m){ return (Number(m)||0)/1e6; }
function safeText(v){ return (v||'').toString(); }
function normalize(s){ return (s||'').toString().trim().toLowerCase(); }
function pct(x){ return Number.isFinite(x)? Math.round(x*1000)/10 + '%': ''; }
function dec1(x){ return Number.isFinite(x)? Math.round(x*10)/10 : ''; }
function money(x){ return Number.isFinite(x)? Math.round(x*100)/100 : ''; }
function safeBuild(fn){ try{ fn(); } catch(e){ Logger.log('Chart build skipped: '+e); } }
```