```
/**
 * ╔═══════════════════════════════════════════════════════════════════════╗
 * ║                  FONCTION 1 : CRÉER LES FEUILLES                      ║
 * ║                                                                       ║
 * ║ Crée "Suivi Commandes" et "Paramètres"                                ║
 * ║ Régénère "Paramètres" à chaque fois pour garder la liste à jour       ║
 * ╚═══════════════════════════════════════════════════════════════════════╝
 */
function creerFeuilles() {
  const ss = SpreadsheetApp.getActiveSpreadsheet();

  let sheetSuivi = ss.getSheetByName('Suivi Commandes');
  if (!sheetSuivi) {
    sheetSuivi = ss.insertSheet('Suivi Commandes');
    Logger.log('✓ Onglet "Suivi Commandes" créé');
  }

  // Toujours régénérer Paramètres (pour avoir la liste à jour)
  let sheetParams = ss.getSheetByName('Paramètres');
  if (sheetParams) {
    // Supprimer l'ancien onglet
    ss.deleteSheet(sheetParams);
    Logger.log('✓ Ancien onglet "Paramètres" supprimé');
  }

  // Créer un nouveau Paramètres avec la liste à jour
  sheetParams = ss.insertSheet('Paramètres');
  creerParametres(sheetParams);
  Logger.log('✓ Onglet "Paramètres" régénéré avec la liste à jour');

  creerHeaderSiNecessaire();
}

/**
 * ╔═══════════════════════════════════════════════════════════════════════╗
 * ║                  FONCTION 2 : CRÉER LES PARAMÈTRES                    ║
 * ║                                                                       ║
 * ║ Pré-remplissage avec les adresses officielles des marchands français  ║
 * ║ + colonne "Mot-clé" optionnelle pour les cas spéciaux                 ║
 * ║                                                                       ║
 * ║ Vide et régénère l'onglet à chaque fois pour garder la liste à jour   ║
 * ╚═══════════════════════════════════════════════════════════════════════╝
 */
function creerParametres(sheet) {
  // Vider complètement la feuille (au cas où elle existe déjà)
  const range = sheet.getRange(1, 1, sheet.getMaxRows(), sheet.getMaxColumns());
  range.clearContent();

  // Header
  sheet.appendRow(['Marchand', 'Email du marchand', 'Mot-clé (optionnel)']);
  sheet.getRange(1, 1, 1, 3).setFontWeight('bold');
  sheet.getRange(1, 1, 1, 3).setBackground('#FFF9C4');

  // Liste des marchands avec adresses OFFICIELLES
  // Cette liste est mise à jour régulièrement
  const marchands = [
    ['Amazon', 'confirmation-commande@amazon.fr', 'votre commande amazon|confirmation
de commande|commande n°'],
    ['Google Store', 'googlestore-noreply@google.com', 'est prête à être expédiée|en
attente de retrait|GS.[0-9]'],
    ['Zalando', 'noreply@zalando.fr', 'commande|order'],
    ['Cdiscount', 'ne_pas_repondre.cdiscount@servicenotification.net', 'expédition de
votre commande|votre colis est en route|information de suivi'],
    ['Fnac', 'noreply@fnac.com', 'commande|merci'],
```

```
    ['Darty', 'no-reply@darty.com', 'confirmation de votre commande|enregistré votre
commande|n° de transaction'],
    ['La Redoute', 'laredoute@serviceclient.laredoute.fr', 'merci pour votre
commande|votre commande est confirmée|suivez votre commande'],
    ['eBay', 'noreply@ebay.fr', 'merci|thank|confirmation'],
    ['AliExpress', 'transaction@notice.aliexpress.com', 'bien été
envoyée|commande.*envoyée|suivez votre commande'],
    ['Vinted', 'noreply@vinted.com', 'confirmation|shipped'],
    ['ASOS', 'no-reply@asos.com', 'order confirmation|commande'],
    ['Shein', 'orders@mail.shein.com', 'order confirmation|merci'],
    ['Booking.com', 'confirmation@booking.com', 'booking confirmation|réservation'],
    ['Airbnb', 'reservations@airbnb.fr', 'réservation|reservation confirmed'],
    ['Uber Eats', 'noreply@uber.com', 'order confirmed|commande confirmée'],
    ['ManoMano', 'noreply@manomano.fr', 'commande confirmée|confirmation'],
    ['Rue du Commerce', 'noreply@rueducommerce.fr', 'commande|confirmation'],
  ];

  marchands.forEach(marchand => {
    sheet.appendRow(marchand);
  });

  sheet.autoResizeColumns(1, 3);

  // Instructions
  sheet.getRange(sheet.getLastRow() + 2, 1).setValue('💡 Tu peux ajouter tes propres
marchands ci-dessous');
  sheet.getRange(sheet.getLastRow(), 1).setFontSize(10);

  sheet.getRange(sheet.getLastRow() + 1, 1).setValue('📌 Mot-clé = optionnel, pour les
mails transférés ou perso');
  sheet.getRange(sheet.getLastRow(), 1).setFontSize(9);
  sheet.getRange(sheet.getLastRow(), 1).setFontStyle('italic');
}

/**
 * ╔═══════════════════════════════════════════════════════════════════════╗
 * ║                    FONCTION 3 : CRÉER LE MENU                         ║
 * ╚═══════════════════════════════════════════════════════════════════════╝
 */
function onOpen() {
  creerFeuilles();

  const ui = SpreadsheetApp.getUi();

  ui.createMenu('📦 Suivi Commandes')
    .addItem('Analyser les emails et créer suivi', 'analyserEmailsEtCreerSuivi')
    .addItem('Mettre à jour les statuts', 'mettreAJourStatuts')
    .addSeparator()
    .addItem('Voir le résumé', 'afficherResume')
    .addToUi();
}

/**
 * ╔═══════════════════════════════════════════════════════════════════════╗
 * ║              FONCTION 4 : CRÉER LE HEADER SI NÉCESSAIRE               ║
 * ╚═══════════════════════════════════════════════════════════════════════╝
 */
function creerHeaderSiNecessaire() {
  const ss = SpreadsheetApp.getActiveSpreadsheet();
  const sheet = ss.getSheetByName('Suivi Commandes');

  if (sheet.getLastRow() === 0) {
    sheet.appendRow([
      'Date de commande',
      'Marchand',
```

```
      'N° de commande',
      'N° de suivi',
      'Date estimée',
      'Statut',
      'Lien suivi',
      'Notes',
      'Email ID'
    ]);

    sheet.getRange(1, 1, 1, 9).setFontWeight('bold');
    sheet.getRange(1, 1, 1, 9).setBackground('#F3E5F5');
    sheet.autoResizeColumns(1, 9);

    Logger.log('✓ Header créé');
  }
}

/**
 * ╔═══════════════════════════════════════════════════════════════════════╗
 * ║            FONCTION 5 : CRÉER LES LABELS SI NÉCESSAIRE                ║
 * ╚═══════════════════════════════════════════════════════════════════════╝
 */
function creerLabelsAutomat() {
  try {
    const labelNames = [
      'Commandes_En_cours',
      'Commandes_Livrées',
      'Commandes_Probleme'
    ];

    labelNames.forEach(labelName => {
      try {
        GmailApp.getUserLabelByName(labelName);
        Logger.log(`✓ Label "${labelName}" existe déjà`);
      } catch (e) {
        GmailApp.createLabel(labelName);
        Logger.log(`✓ Label "${labelName}" créé`);
      }
    });
  } catch (e) {
    Logger.log('Erreur labels : ' + e.toString());
  }
}

/**
 * ╔═══════════════════════════════════════════════════════════════════════╗
 * ║        FONCTION 6 : CHARGER LES MARCHANDS + MOTS-CLÉS DEPUIS "PARAMÈTRES"
 * ║                                                                       ║
 * ║ Lit les 3 colonnes : Marchand | Email | Mot-clé                       ║
 * ╚═══════════════════════════════════════════════════════════════════════╝
 */
function chargerMarchands() {
  const ss = SpreadsheetApp.getActiveSpreadsheet();
  const sheetParams = ss.getSheetByName('Paramètres');

  if (!sheetParams) {
    Logger.log('❌ Feuille "Paramètres" non trouvée');
    return [];
  }

  const donnees = sheetParams.getDataRange().getValues();
  const marchands = [];

  // Boucler sur les lignes (ignorer le header)
```

```
  for (let i = 1; i < donnees.length; i++) {
    const nom = donnees[i][0];
    const email = donnees[i][1];
    const motCle = donnees[i][2];

    // Ignorer les lignes vides
    if (nom && nom.trim() !== '') {
      marchands.push({
        nom: nom.trim(),
        email: email ? email.trim() : '',
        motCle: motCle ? motCle.trim() : ''
      });
    }
  }

  Logger.log(`📋 ${marchands.length} marchand(s) chargé(s) depuis "Paramètres"`);

  return marchands;
}

/**
 * ╔═══════════════════════════════════════════════════════════════════╗
 * ║     FONCTION 7 : ANALYSER LES EMAILS ET CRÉER SUIVI (PRINCIPALE)  ║
 * ╚═══════════════════════════════════════════════════════════════════╝
 */
function analyserEmailsEtCreerSuivi() {
  try {
    const monEmail = Session.getActiveUser().getEmail();
    Logger.log(`👤 Email détecté : ${monEmail}`);

    const ss = SpreadsheetApp.getActiveSpreadsheet();
    const sheet = ss.getSheetByName('Suivi Commandes');

    creerHeaderSiNecessaire();
    creerLabelsAutomat();

    const marchands = chargerMarchands();

    if (marchands.length === 0) {
      afficherAlerte('❌ Aucun marchand configuré dans "Paramètres".\n\n' +
        'Va dans l\'onglet "Paramètres" et ajoute au moins un marchand.');
      return;
    }

    const toutes_les_lignes = sheet.getDataRange().getValues();
    const idsExistants = toutes_les_lignes.map(row => row[8]);
    const numsCommandesExistants = toutes_les_lignes.map(row => row[2]);

    const threads = GmailApp.search('to:' + monEmail + ' newer_than:7d');

    if (threads.length === 0) {
      afficherAlerte('Aucun email trouvé pour ' + monEmail + '\n\n' +
        'Vérifie que tu as reçu des emails récemment.');
      return;
    }

    Logger.log(`🔍 ${threads.length} email(s) trouvé(s)`);

    let compteur = 0;
    let doublons = 0;
    let erreurs = 0;
    let emailsCommandes = 0;

    threads.forEach((thread, index) => {
      try {
```

```
        const messages = thread.getMessages();
        const dernierMessage = messages[messages.length - 1];
        const emailId = dernierMessage.getId();

        if (idsExistants.includes(emailId)) {
          doublons++;
          return;
        }

        const date = dernierMessage.getDate();
        const from = dernierMessage.getFrom();
        const subject = dernierMessage.getSubject();
        const plainBody = dernierMessage.getPlainBody();

        // === DÉTECTION : Email + Mot-clé ===
        if (!estUnEmailDeCommande(from, subject, plainBody, marchands)) {
          return;
        }

        emailsCommandes++;
        Logger.log(`✓ Commande détectée : ${subject}`);

        // Extraire le marchand
        let marchand = extraireMarchand(from, marchands);

        const numCommande = extraireNumCommande(plainBody);

        if (numsCommandesExistants.includes(numCommande)) {
          doublons++;
          return;
        }

        const numSuivi = extraireNumSuivi(plainBody);
        const dateEstimee = extraireDateEstimee(plainBody);
        const lienSuivi = creerLienSuivi(marchand, numSuivi);

        const dateStr = Utilities.formatDate(date, Session.getScriptTimeZone(),
'dd/MM/yyyy');

        try {
          const labelObj = GmailApp.getUserLabelByName('Commandes_En_cours');
          thread.addLabel(labelObj);
          Logger.log(`✓ Email labellisé`);
        } catch (e) {
          Logger.log(`⚠️ Impossible de labelliser : ${e.toString()}`);
        }

        sheet.appendRow([
          dateStr,
          marchand,
          numCommande,
          numSuivi,
          dateEstimee,
          'En cours',
          lienSuivi,
          '',
          emailId
        ]);

        compteur++;

      } catch (e) {
        erreurs++;
        Logger.log('❌ Erreur sur email : ' + e.toString());
      }
```

```
    });

    let message = `✓ Analyse terminée !\n\n`;
    message += `🔍 Emails analysés : ${threads.length}\n`;
    message += `📦 Commandes détectées : ${emailsCommandes}\n`;
    message += `📝 Commandes ajoutées : ${compteur}\n`;
    if (doublons > 0) {
      message += `⚠️ Doublons évités : ${doublons}\n`;
    }
    if (erreurs > 0) {
      message += `❌ Erreurs : ${erreurs}\n`;
    }

    afficherAlerte(message);

  } catch (e) {
    afficherAlerte('❌ Erreur globale : ' + e.toString());
    Logger.log('Erreur : ' + e.toString());
  }
}

/**
 * ╔═══════════════════════════════════════════════════════════════════╗
 * ║          FONCTION 8 : VÉRIFIER SI C'EST UN EMAIL DE COMMANDE      ║
 * ║                                                                   ║
 * ║ LOGIQUE AMÉLIORÉE :                                               ║
 * ║ 1. Vérifier l'email du marchand                                   ║
 * ║ 2. ET vérifier que le mot-clé est présent                         ║
 * ║ 3. Si TOUT correspond → c'est une commande                        ║
 * ║                                                                   ║
 * ║ Résultat : ZÉRO faux positif !                                    ║
 * ╚═══════════════════════════════════════════════════════════════════╝
 */
function estUnEmailDeCommande(from, subject, body, marchands) {
  const fromLower = from.toLowerCase();
  const subjectLower = subject.toLowerCase();
  const bodyLower = body.toLowerCase();

  // Boucler sur les marchands
  for (const m of marchands) {
    // === VÉRIFIER L'EMAIL ===
    if (!m.email || !fromLower.includes(m.email.toLowerCase())) {
      continue; // Cet email ne correspond pas, passer au suivant
    }

    // L'email correspond ! Maintenant vérifier le mot-clé

    // === VÉRIFIER LE MOT-CLÉ ===
    if (m.motCle) {
      // Le marchand a un mot-clé, il FAUT le trouver
      const motsCles = m.motCle.split('|').map(m => m.trim());

      let motCleFound = false;
      for (const motCle of motsCles) {
        if (subjectLower.includes(motCle.toLowerCase()) ||
            bodyLower.includes(motCle.toLowerCase())) {
          motCleFound = true;
          break;
        }
      }

      if (!motCleFound) {
        // Email match mais pas de mot-clé → pas une commande
        Logger.log(`⚠️ Email match (${m.nom}) mais pas de mot-clé détecté`);
        continue;
```

```
      }

      // Email ET mot-clé correspondent
      Logger.log(`✓ Marchand ${m.nom} détecté (email + mot-clé)`);
      return true;
    } else {
      // Le marchand n'a pas de mot-clé (liste vide)
      // Juste l'email suffit
      Logger.log(`✓ Marchand ${m.nom} détecté (email uniquement)`);
      return true;
    }
  }

  // Aucun match trouvé
  Logger.log(`✗ Pas de marchand détecté`);
  return false;
}

/**
 * ╔═══════════════════════════════════════════════════════════════════╗
 * ║           FONCTION 9 : EXTRAIRE LE MARCHAND                       ║
 * ╚═══════════════════════════════════════════════════════════════════╝
 */
function extraireMarchand(from, marchands) {
  const fromLower = from.toLowerCase();

  // Chercher dans la liste des marchands
  for (const m of marchands) {
    if (m.email && fromLower.includes(m.email.toLowerCase())) {
      return m.nom;
    }
  }

  // Fallback : extraire du domaine
  const match = from.match(/@([\w-]+)\./);
  if (match) {
    return match[1].charAt(0).toUpperCase() + match[1].slice(1);
  }

  return 'Inconnu';
}

/**
 * ╔═══════════════════════════════════════════════════════════════════╗
 * ║           FONCTION 10 : EXTRAIRE LE NUMÉRO DE COMMANDE            ║
 * ╚═══════════════════════════════════════════════════════════════════╝
 */
function extraireNumCommande(body) {
  let match = body.match(/commande[#\s:]*(\d{3}-\d{7}-\d{7})/i);
  if (match) return match[1];

  match = body.match(/n°\s*de\s*commande[:\s]*([A-Z0-9]+)/i);
  if (match) return match[1];

  match = body.match(/ref\s*commande[:\s]*([0-9]+)/i);
  if (match) return match[1];

  match = body.match(/(?:commande|order|#)\s*[:\s]*([A-Z0-9\-]{6,20})/i);
  if (match) return match[1];

  return 'Non trouvé';
}

/**
```

```
 *
 * ╔═══════════════════════════════════════════════════════════════════╗
 * ║          FONCTION 11 : EXTRAIRE LE NUMÉRO DE SUIVI                ║
 * ╚═══════════════════════════════════════════════════════════════════╝
 */
function extraireNumSuivi(body) {
  let match = body.match(/(?:colissimo|poste)[:\s]*([0-9A-Z]{13,25})/i);
  if (match) return match[1];

  match = body.match(/(?:dpd)[:\s]*([0-9]{11,14})/i);
  if (match) return match[1];

  match = body.match(/(?:ups)[:\s]*([0-9A-Z]{18})/i);
  if (match) return match[1];

  match = body.match(/(?:gls)[:\s]*([0-9]{10,15})/i);
  if (match) return match[1];

  match = body.match(/(?:fedex)[:\s]*([0-9]{12,14})/i);
  if (match) return match[1];

  match = body.match(/(?:chronopost)[:\s]*([0-9]{10,20})/i);
  if (match) return match[1];

  match = body.match(/(?:suivi|tracking|track)[:\s]*([A-Z0-9]{8,30})/i);
  if (match) return match[1];

  return 'Non trouvé';
}

/**
 *
 * ╔═══════════════════════════════════════════════════════════════════╗
 * ║          FONCTION 12 : EXTRAIRE LA DATE ESTIMÉE                   ║
 * ╚═══════════════════════════════════════════════════════════════════╝
 */
function extraireDateEstimee(body) {
  let match = body.match(/livraison[^a-z]*(?:prévue|estimée|attendue).*?
(\d{1,2}\s+\w+)/i);
  if (match) return match[1];

  match = body.match(/(?:d\+|d \+|livraison\s+en\s+)([0-9]{1,2})\s+(?:jour|jours)/i);
  if (match) {
    const jours = parseInt(match[1]);
    const date = new Date();
    date.setDate(date.getDate() + jours);
    return Utilities.formatDate(date, Session.getScriptTimeZone(), 'dd/MM/yyyy');
  }

  match = body.match(/(\d{1,2}[\/\.]\d{1,2})/);
  if (match) return match[1];

  return 'Non spécifiée';
}

/**
 *
 * ╔═══════════════════════════════════════════════════════════════════╗
 * ║           FONCTION 13 : CRÉER LE LIEN DE SUIVI                    ║
 * ╚═══════════════════════════════════════════════════════════════════╝
 */
function creerLienSuivi(marchand, numSuivi) {
  if (numSuivi === 'Non trouvé' || numSuivi === '') {
    return '';
  }

  let url = '';
```

```
  if (marchand === 'Amazon') {
    url = 'https://www.amazon.fr/gp/your-account/order-details';
  } else if (marchand === 'Zalando') {
    url = 'https://www.zalando.fr/myaccount/orders';
  } else if (marchand === 'Cdiscount') {
    url = 'https://www.cdiscount.com/';
  } else {
    url = 'https://17track.net/en/track?nums=' + numSuivi;
  }

  return url;
}

/**
 * ╔═══════════════════════════════════════════════════════════════════╗
 * ║        FONCTION 14 : METTRE À JOUR LES STATUTS                    ║
 * ╚═══════════════════════════════════════════════════════════════════╝
 */
function mettreAJourStatuts() {
  try {
    const monEmail = Session.getActiveUser().getEmail();
    const ss = SpreadsheetApp.getActiveSpreadsheet();
    const sheet = ss.getSheetByName('Suivi Commandes');

    const threads = GmailApp.search('to:' + monEmail + ' label:Commandes_Livrées');

    if (threads.length === 0) {
      afficherAlerte('Aucune livraison nouvellement confirmée.');
      return;
    }

    let compteur = 0;

    threads.forEach(thread => {
      try {
        const message = thread.getMessages()[thread.getMessages().length - 1];
        const plainBody = message.getPlainBody();

        const numCommande = extraireNumCommande(plainBody);

        const toutes_les_lignes = sheet.getDataRange().getValues();

        for (let i = 1; i < toutes_les_lignes.length; i++) {
          if (toutes_les_lignes[i][2] === numCommande) {
            sheet.getRange(i + 1, 6).setValue('Livrée');
            compteur++;
            break;
          }
        }

      } catch (e) {
        Logger.log('Erreur : ' + e.toString());
      }
    });

    afficherAlerte(`✓ ${compteur} commande(s) marquée(s) comme livrée(s)`);

  } catch (e) {
    afficherAlerte('❌ Erreur : ' + e.toString());
  }
}

/**
 * ╔═══════════════════════════════════════════════════════════════════╗
 * ║           FONCTION 15 : AFFICHER RÉSUMÉ                           ║
```

```
 * ╚═══════════════════════════════════════════════════════════════════════╝
 */
function afficherResume() {
  try {
    const ss = SpreadsheetApp.getActiveSpreadsheet();
    const sheet = ss.getSheetByName('Suivi Commandes');
    const toutes_les_lignes = sheet.getDataRange().getValues();

    let en_cours = 0;
    let livrees = 0;
    let probleme = 0;

    for (let i = 1; i < toutes_les_lignes.length; i++) {
      const statut = toutes_les_lignes[i][5];

      if (statut === 'En cours') en_cours++;
      else if (statut === 'Livrée') livrees++;
      else if (statut === 'Problème') probleme++;
    }

    let resume = '📦 RÉSUMÉ DES COMMANDES\n\n';
    resume += `🚚 En cours : ${en_cours}\n`;
    resume += `✓ Livrées : ${livrees}\n`;
    resume += `⚠️ Problèmes : ${probleme}\n\n`;
    resume += `Total : ${toutes_les_lignes.length - 1}`;

    afficherAlerte(resume);

  } catch (e) {
    afficherAlerte('Erreur : ' + e.toString());
  }
}

/**
 * ╔═══════════════════════════════════════════════════════════════════════╗
 * ║          FONCTION 16 : AFFICHER ALERTE (UTILITAIRE)                   ║
 * ╚═══════════════════════════════════════════════════════════════════════╝
 */
function afficherAlerte(message) {
  SpreadsheetApp.getUi().alert(message);
}
```