RESOURCE · WORKBOOK

조직 자동화 9선

기획·HR·마케팅 — 누구나 30분

9개 자동화 통합본 · 완성 코드 · 시트 템플릿 · 트러블슈팅
Google Apps Script · Gemini · Slack · 월 0원 운영
브라우저 도구 9선 — 입력만으로 결과 즉시 생성

강사 김창환 · 네다바웨이
자료 허브: nedabah.org/auto
도구 모음: nedabah.org/auto/tools

목차

자료 허브: nedabah.org/auto · 도구 모음: nedabah.org/auto/tools · 강의 의뢰: nedabah.way@gmail.com

PLAN-01 · 기획 자동화

회의록 → 액션아이템 자동 정리

Google Doc 한 장을 던지면 AI가 결정사항·할일·담당자·기한을 추출해 시트·Slack에 적재합니다.

난이도	★
예상 소요	**30분**
전제 조건	**구글 계정 + Gemini API 키**
월 비용	**0원**

⚡ **셋업 전에 결과부터:** 같은 로직을 브라우저에서 입력 한 번으로 실행하는 미니 도구가 있습니다 → `nedabah.org/auto/tools/meeting-actions/`

한 줄 핵심: 회의록 Google Doc 한 장을 던지면, AI가 의사결정·할일·담당자·기한을 추출해 Sheet에 적재하고 Slack에 발송한다.

왜 이 자동화인가

항목	수동(Before)	자동(After)
회의록 정리 시간	30~60분/회	0분(트리거만)
액션아이템 누락률	30~50%	<5%
담당자 통보 지연	평균 2일	즉시
주간 회의 기준 연 절감	—	**약 130시간/팀**

적용 시나리오: 주간 팀회의 / 임원회의 / 프로젝트 스탠드업 / 1on1 정리.

구성요소 (모두 무료 또는 매우 저렴)

- Google Docs (회의록 원본)
- Google Sheets (액션아이템 적재)
- Google Apps Script (실행 엔진)
- Gemini API — `gemini-2.5-flash` 무료 티어 (15 RPM)
- Slack Incoming Webhook (선택)

총 비용: **월 0원** (소규모 팀 기준).

셋업 30분 가이드

Step 1. 시트 만들기

1. 새 Google Sheet 생성 → 이름: `회의록-액션아이템`
2. 1행에 헤더 추가: `회의일 | 회의명 | 결정사항 | 액션아이템 | 담당자 | 기한 | 상태 | 원본링크`
3. URL에서 시트 ID 복사 (`/d/{여기}/edit`).

Step 2. Gemini API 키 발급

1. https://aistudio.google.com/apikey 접속
2. **Create API key** → 키 복사 (한번만 노출됨)

Step 3. Slack Webhook (선택)

1. https://api.slack.com/apps → Create New App → From scratch
2. Incoming Webhooks 활성화 → 채널 선택 → Webhook URL 복사

Step 4. Apps Script 붙여넣기

1. Google Sheet에서 **확장 프로그램 → Apps Script**
2. `script.gs` 내용 붙여넣기 (아래)
3. **프로젝트 설정 → 스크립트 속성**에 다음 추가: - `GEMINI_API_KEY` = (발급받은 키) - `SHEET_ID` = (시트 ID) - `SLACK_WEBHOOK` = (선택)
4. **트리거 추가**: `onMeetingDocOpen` → 시간 기반 → 매일 오전 9시 또는 **Docs 메뉴에서 직접 실행**(아래 `installDocMenu` 사용)

Step 5. 회의록 작성 규칙 (사용자 약속)

- 문서 첫 줄: `회의명: 주간 마케팅 회의 / 2026-05-04`
- 본문은 자연어 그대로 작성 (별도 양식 불필요)
- 작성 완료 후 메뉴 🤖 **자동화 → 액션아이템 추출** 클릭

완성 코드 (`script.gs`)

```
/**
 * 회의록 → 액션아이템 자동 정리
 * 사용: Google Docs에서 메뉴 [자동화 → 액션아이템 추출] 클릭
 */

const PROPS = PropertiesService.getScriptProperties();
const GEMINI_KEY = PROPS.getProperty('GEMINI_API_KEY');
const SHEET_ID   = PROPS.getProperty('SHEET_ID');
const SLACK_HOOK = PROPS.getProperty('SLACK_WEBHOOK') || '';

function onOpen() {
  DocumentApp.getUi()
    .createMenu('🤖 자동화')
    .addItem('액션아이템 추출', 'extractActions')
    .addToUi();
}
```

```
function extractActions() {
  const doc  = DocumentApp.getActiveDocument();
  const text = doc.getBody().getText();
  const url  = doc.getUrl();

  const ui = DocumentApp.getUi();
  ui.alert('처리 중', 'AI가 회의록을 분석합니다. 10~20초 소요됩니다.', ui.ButtonSet.OK);

  const result = callGemini(text);
  appendToSheet(result, url);
  if (SLACK_HOOK) postToSlack(result, url);

  ui.alert('완료', `${result.actions.length}개 액션아이템을 시트에 적재했습니다.`, ui.ButtonSet.OK);
}

function callGemini(meetingText) {
  const prompt = `
다음 회의록에서 정보를 추출하세요. JSON만 반환하세요(설명 없이).

회의록:
"""
${meetingText.slice(0, 30000)}
"""

JSON 스키마:
{
  "meetingName": "회의명",
  "meetingDate": "YYYY-MM-DD",
  "decisions": ["결정사항1", "결정사항2"],
  "actions": [
    {"task":"액션내용","owner":"담당자","due":"YYYY-MM-DD 또는 빈문자열"}
  ]
}

규칙:
- 담당자는 회의록에 등장한 사람만 사용. 추정 금지.
- 기한이 명시되지 않으면 빈문자열.
- 결정사항과 액션아이템은 다르다(결정=합의, 액션=실행).
`;

  const url = `https://generativelanguage.googleapis.com/v1beta/models/gemini-2.5-flash:generateContent?key=$
{GEMINI_KEY}`;
  const res = UrlFetchApp.fetch(url, {
    method: 'post',
    contentType: 'application/json',
    payload: JSON.stringify({
      contents: [{parts: [{text: prompt}]}],
      generationConfig: {responseMimeType: 'application/json'}
    })
  });
  const json = JSON.parse(res.getContentText());
  const content = json.candidates[0].content.parts[0].text;
  return JSON.parse(content);
}

function appendToSheet(result, docUrl) {
  const sheet = SpreadsheetApp.openById(SHEET_ID).getSheets()[0];
  const decisionsStr = result.decisions.join(' | ');
  result.actions.forEach(a => {
    sheet.appendRow([
      result.meetingDate || new Date().toISOString().slice(0,10),
      result.meetingName || '',
      decisionsStr,
      a.task,
      a.owner || '',
      a.due || '',
```

```
      '대기',
      docUrl
    ]);
  });
}

function postToSlack(result, docUrl) {
  const lines = result.actions.map((a,i) =>
    `${i+1}. *${a.task}* — ${a.owner || '미지정'}${a.due ? ` (~${a.due})` : ''}`
  ).join('\n');
  const text = `📝 *${result.meetingName}* 회의 요약\n\n*결정사항*\n• ${result.decisions.join('\n• ')}\n\n*액션아이템*\n${lines}\n\n🔗 ${docUrl}`;
  UrlFetchApp.fetch(SLACK_HOOK, {
    method: 'post',
    contentType: 'application/json',
    payload: JSON.stringify({text})
  });
}
```

강의 시연 시나리오 (10분)

1. 강사가 미리 준비한 가짜 회의록 Doc 오픈
2. 메뉴 [🤖 자동화 → 액션아이템 추출] 클릭
3. 15초 대기 → Sheet에 5~7행 자동 추가되는 모습 시연
4. Slack 채널에 자동 발송된 카드 시연
5. 잘못된 행을 직접 수정하며 "AI는 보조, 사람은 검토" 메시지 강조

트러블슈팅

증상	원인	해결
`Cannot read properties of undefined`	Gemini 응답이 JSON이 아님	프롬프트 끝에 "JSON만 반환" 강조 + `responseMimeType` 확인
한국어 담당자 이름 깨짐	UTF-8 처리 실패	Apps Script v8 런타임 사용 (기본값)
시트에 빈 행만 추가	`actions` 배열 0개	회의록이 짧거나 액션이 없는 경우, 정상 동작
Gemini 429 에러	무료 티어 분당 한도 초과	`Utilities.sleep(4500)` 호출 사이 추가

응용 아이디어 (강의 후반 토론용)

- **임원 회의 보안 강화**: Sheet ID를 권한 그룹으로 잠그고 Slack은 비공개 채널만
- **다국어 회의**: 프롬프트에 `"답변은 회의 원문 언어로"` 추가
- **CRM 연동**: 액션의 `owner`가 영업일 경우 Salesforce/HubSpot Task로 자동 생성
- **음성 회의록**: Google Meet 녹화 → Whisper API 전사 → 본 스크립트로 연결

PLAN-02 · 기획 자동화

경쟁사·뉴스 일일 다이제스트

매일 새벽 RSS·키워드를 수집·요약해 메일/Slack 한 장 다이제스트로 발송합니다.

난이도	★★
예상 소요	**45분**
전제 조건	**구글 계정 + Gemini API 키 + (선택) Slack**
월 비용	**0원**

⚡ **셋업 전에 결과부터:** 같은 로직을 브라우저에서 입력 한 번으로 실행하는 미니 도구가 있습니다 → `nedabah.org/auto/tools/news-digest/`

> **한 줄 핵심**: 경쟁사 키워드와 뉴스 RSS를 매일 새벽 자동 수집·요약해, 출근 전 한 장의 다이제스트를 메일/Slack으로 받는다.

왜 이 자동화인가

항목	수동(Before)	자동(After)
매일 검색·정리	40~90분	0분
정보 누락	자주 발생	매우 드묾
보고용 리포트 작성	주 2시간	클릭 1회
1년 시간 절감	—	**약 200시간/기획자 1인**

적용 시나리오: 마케팅·전략기획·PR·투자분석·영업기획.

구성요소

- Google Sheet (키워드·결과 저장)
- 네이버 뉴스 RSS / Google News RSS / 산업 RSS (무료)
- Gemini API (요약·중복 제거)
- Gmail (자동 발송) 또는 Slack Webhook
- Apps Script 시간 트리거 (매일 06:30)

월 비용: **0원**.

셋업 가이드

Step 1. 시트 만들기 — 두 개의 탭

탭1 `keywords`: | 카테고리 | 키워드 | RSS_URL | |---|---|---| | 경쟁사 | 토스 | `https://news.google.com/rss/search?q=토스&hl=ko` | | 산업 | 핀테크 | `https://news.google.com/rss/search?q=핀테크&hl=ko` | | 자사 멘션 | 우리 회사명 | `https://news.google.com/rss/search?q=우리회사명&hl=ko` |

탭2 `digest_log` (자동 채워짐): | 발송일 | 제목 | URL | 카테고리 | 요약 |

Step 2. 키 발급 (이전 가이드 참고)

- `GEMINI_API_KEY`
- `SHEET_ID`
- `RECIPIENT_EMAIL` (또는 `SLACK_WEBHOOK`)

Step 3. Apps Script 붙여넣기 → 시간 트리거 매일 06:30 설정.

완성 코드 (`script.gs`)

```
const PROPS = PropertiesService.getScriptProperties();
const GEMINI_KEY = PROPS.getProperty('GEMINI_API_KEY');
const SHEET_ID   = PROPS.getProperty('SHEET_ID');
const EMAIL_TO   = PROPS.getProperty('RECIPIENT_EMAIL') || '';
const SLACK_HOOK = PROPS.getProperty('SLACK_WEBHOOK') || '';

const LOOKBACK_HOURS = 24;
const MAX_PER_KEYWORD = 8;

function dailyDigest() {
  const ss = SpreadsheetApp.openById(SHEET_ID);
  const kw = ss.getSheetByName('keywords').getDataRange().getValues();
  const log = ss.getSheetByName('digest_log');

  const since = new Date(Date.now() - LOOKBACK_HOURS * 3600 * 1000);
  const collected = [];

  for (let i = 1; i < kw.length; i++) {
    const [category, keyword, rssUrl] = kw[i];
    if (!rssUrl) continue;
    const items = fetchRSS(rssUrl, since).slice(0, MAX_PER_KEYWORD);
    items.forEach(it => collected.push({...it, category, keyword}));
    Utilities.sleep(800);
  }

  if (collected.length === 0) {
    Logger.log('수집된 뉴스 0건. 다이제스트 생략.');
    return;
  }

  const dedup = deduplicate(collected);
  // Gemini 입력 토큰 한도를 넘지 않도록 50건 단위로 청크 처리
  const summarized = [];
  for (let i = 0; i < dedup.length; i += 50) {
    const chunk = dedup.slice(i, i + 50);
    summarized.push(...summarizeBatch(chunk));
    if (i + 50 < dedup.length) Utilities.sleep(2000);
  }
```

```
  const html = buildDigestHTML(summarized);

  // 로그 적재
  const today = new Date().toISOString().slice(0,10);
  summarized.forEach(s => log.appendRow([today, s.title, s.url, s.category, s.summary]));

  if (EMAIL_TO) {
    GmailApp.sendEmail(EMAIL_TO, `[데일리] ${today} 경쟁사·산업 다이제스트`, '', {htmlBody: html});
  }
  if (SLACK_HOOK) {
    UrlFetchApp.fetch(SLACK_HOOK, {
      method: 'post', contentType: 'application/json',
      payload: JSON.stringify({text: htmlToSlackText(summarized, today)})
    });
  }
}

function fetchRSS(url, since) {
  try {
    const xml = UrlFetchApp.fetch(url, {muteHttpExceptions: true}).getContentText();
    const doc = XmlService.parse(xml);
    const items = doc.getRootElement().getChild('channel').getChildren('item');
    return items.map(it => ({
      title: it.getChildText('title') || '',
      url:   it.getChildText('link')  || '',
      pub:   new Date(it.getChildText('pubDate') || Date.now()),
      desc:  (it.getChildText('description') || '').replace(/<[^>]+>/g, '').slice(0, 500)
    })).filter(it => it.pub >= since);
  } catch(e) {
    Logger.log(`RSS 실패 ${url}: ${e}`);
    return [];
  }
}

function deduplicate(items) {
  const seen = new Set();
  return items.filter(it => {
    const key = it.title.replace(/\s+/g, '').slice(0, 30);
    if (seen.has(key)) return false;
    seen.add(key);
    return true;
  });
}

function summarizeBatch(items) {
  const prompt = `
다음 뉴스 ${items.length}건을 각각 한국어 1~2문장으로 요약하세요.
- 핵심 사실만, 추측 금지
- "~다"체로 통일
- JSON 배열만 반환: [{"summary":"..."},...]
순서 유지.

뉴스:
${items.map((it,i)=>`[${i+1}] ${it.title}\n${it.desc}`).join('\n\n')}
`;
  const url = `https://generativelanguage.googleapis.com/v1beta/models/gemini-2.5-flash:generateContent?key=$
{GEMINI_KEY}`;
  const res = UrlFetchApp.fetch(url, {
    method: 'post', contentType: 'application/json',
    payload: JSON.stringify({
      contents:[{parts:[{text: prompt}]}],
      generationConfig:{responseMimeType: 'application/json'}
    })
  });
  const arr = JSON.parse(JSON.parse(res.getContentText()).candidates[0].content.parts[0].text);
  return items.map((it, i) => ({...it, summary: (arr[i]||{}).summary || ''}));
}
```

```
function buildDigestHTML(items) {
  const grouped = {};
  items.forEach(it => {
    grouped[it.category] = grouped[it.category] || [];
    grouped[it.category].push(it);
  });
  let html = `<div style="font-family:'Noto Sans KR',sans-serif;max-width:680px;line-height:1.6;">
    <h2 style="border-bottom:2px solid #b45309;padding-bottom:.4rem;">📊 데일리 다이제스트</h2>
    <p style="color:#6a604f;">${new Date().toLocaleDateString('ko-KR')} · 총 ${items.length}건</p>`;
  for (const cat in grouped) {
    html += `<h3 style="margin-top:1.5rem;color:#3a322a;">${cat}</h3><ul>`;
    grouped[cat].forEach(it => {
      html += `<li style="margin-bottom:.6rem;"><a href="${it.url}" style="color:#b45309;text-decoration:none;font-
weight:600;">${it.title}</a><br><span style="color:#555;font-size:.95em;">${it.summary}</span></li>`;
    });
    html += `</ul>`;
  }
  html += `<p style="color:#999;font-size:.85em;margin-top:2rem;">자동 생성 · 출처는 각 링크 참조</p></div>`;
  return html;
}

function htmlToSlackText(items, date) {
  const grouped = {};
  items.forEach(it => { grouped[it.category] = grouped[it.category] || []; grouped[it.category].push(it); });
  let txt = `*📊 ${date} 데일리 다이제스트* (${items.length}건)\n`;
  for (const cat in grouped) {
    txt += `\n*${cat}*\n`;
    grouped[cat].forEach(it => txt += `• <${it.url}|${it.title}> — ${it.summary}\n`);
  }
  return txt;
}
```

강의 시연 포인트

1. `keywords` 시트의 한 행을 추가/수정하며 "코드 한 줄도 안 고치고 모니터링 대상 변경" 시연
2. `dailyDigest()` 수동 실행 → 1분 내 메일/Slack에 도착하는 모습 보여주기
3. 키워드 잘못 잡혔을 때 → AI 요약 한 줄로 빠르게 무관함 판별 가능 강조

트러블슈팅

증상	원인	해결
RSS 0건	키워드 너무 좁음 / RSS URL 만료	Google News RSS는 한국어 검색에 `&hl=ko&gl=KR&ceid=KR:ko` 추가
Gemini 응답 잘림	items가 50건 초과	`MAX_PER_KEYWORD` 줄이거나 `summarizeBatch`를 청크로 분할
Gmail 송신 실패	일일 한도(개인계정 100건)	Workspace 계정 사용 또는 Slack만 사용
트리거 시간 어긋남	Apps Script는 ±1시간	업무 시작 1시간 전으로 설정

응용 아이디어

- **카테고리별 가중치**: `keywords` 시트에 `priority` 컬럼 추가 → 높은 우선순위는 별도 강조

- **PDF 보고서**: HTML을 `DocumentApp`으로 변환해 주간 PDF로 자동 보관
- **Slack 인터랙션**: 다이제스트 카드에 "관심 / 무관심" 버튼 추가하여 학습 데이터로 활용
- **자사 멘션 알림**: 자사 키워드 카테고리만 즉시 알림(트리거 분리)으로 위기 대응 단축

PLAN-03 · 기획 자동화

주간 KPI 자동 리포트

AI가 변동·이상·다음 주 권고를 함께 작성한 한 장 KPI 리포트를 매주 임원진에 발송합니다.

난이도	★★
예상 소요	**60분**
전제 조건	**KPI 적재된 Google Sheet + Gemini API 키**
월 비용	**0원**

⚡ **셋업 전에 결과부터:** 같은 로직을 브라우저에서 입력 한 번으로 실행하는 미니 도구가 있습니다 → `nedabah.org/auto/tools/kpi-comment/`

> **한 줄 핵심**: 흩어진 KPI 시트를 매주 월요일 새벽에 모아, AI가 변화 포인트와 다음 주 권고를 작성한 한 장 리포트로 임원진에 발송한다.

왜 이 자동화인가

항목	수동(Before)	자동(After)
주간 리포트 작성	3~5시간/주	0분
데이터 출처 일관성	사람마다 다름	단일 소스
인사이트 깊이	시간 압박으로 얕음	AI가 변동·이상 자동 검출
1년 시간 절감	—	**약 200시간/팀**

적용 시나리오: 경영지원·전략기획·사업부장 보고·이사회 사전자료.

구성요소

- Google Sheet (KPI 원천 — 일별 적재 가정)
- Google Slides 또는 Gmail (리포트 출력)
- Apps Script + Gemini API
- Slack (선택)

셋업 가이드

Step 1. KPI 시트 형식 (예시)

탭 `kpi_daily`: | 날짜 | 지표 | 값 | 단위 | 주관 | |---|---|---|---|---| | 2026-04-28 | 신규가입 | 412 | 명 | 마케팅 | | 2026-04-28 | DAU | 18900 | 명 | 프로덕트 | | 2026-04-28 | 결제전환율 | 3.4 | % | 영업 |

> **약속**: 한 행 = 한 지표의 하루 값. 시트는 "단일 truth source".

Step 2. KPI 정의 시트

탭 `kpi_meta`: | 지표 | 좋은방향 | 목표값 | 단위 | |---|---|---|---| | 신규가입 | up | 500 | 명 | | DAU | up | 20000 | 명 | | 결제전환율 | up | 4.0 | % | | 이탈율 | down | 5.0 | % |

Step 3. Apps Script 붙여넣기 → 트리거 매주 월요일 06:30.

완성 코드 (`script.gs`)

```
const PROPS = PropertiesService.getScriptProperties();
const GEMINI_KEY = PROPS.getProperty('GEMINI_API_KEY');
const SHEET_ID   = PROPS.getProperty('SHEET_ID');
const EMAIL_TO   = PROPS.getProperty('RECIPIENT_EMAIL');
const SLACK_HOOK = PROPS.getProperty('SLACK_WEBHOOK') || '';

function weeklyReport() {
  const ss = SpreadsheetApp.openById(SHEET_ID);
  const daily = ss.getSheetByName('kpi_daily').getDataRange().getValues();
  const meta  = ss.getSheetByName('kpi_meta').getDataRange().getValues();

  const metaMap = {};
  for (let i = 1; i < meta.length; i++) {
    metaMap[meta[i][0]] = {direction: meta[i][1], target: meta[i][2], unit: meta[i][3]};
  }

  const now = new Date();
  const thisStart = startOfWeek(now);
  const lastStart = new Date(thisStart.getTime() - 7*86400000);
  const lastEnd   = new Date(thisStart.getTime() - 1);

  const stats = {};
  for (let i = 1; i < daily.length; i++) {
    const [date, indicator, value] = daily[i];
    if (!indicator) continue;
    const d = new Date(date);
    stats[indicator] = stats[indicator] || {thisWeek:[], lastWeek:[]};
    if (d >= thisStart && d < now)        stats[indicator].thisWeek.push(Number(value));
    else if (d >= lastStart && d <= lastEnd) stats[indicator].lastWeek.push(Number(value));
  }

  const summary = Object.entries(stats).map(([ind, v]) => {
    const tAvg = avg(v.thisWeek);
    const lAvg = avg(v.lastWeek);
    const wow  = lAvg ? ((tAvg - lAvg) / lAvg * 100) : 0;
    const m = metaMap[ind] || {};
    const status = judgeStatus(tAvg, m);
    return {indicator: ind, thisAvg: tAvg, lastAvg: lAvg, wow, target: m.target, direction: m.direction, unit: m.unit,
status};
```

```
  });

  const insight = aiCommentary(summary);
  const html = buildReportHTML(summary, insight, thisStart);

  if (EMAIL_TO) GmailApp.sendEmail(EMAIL_TO, `[주간] ${fmt(thisStart)} KPI 리포트`, '', {htmlBody: html});
  if (SLACK_HOOK) UrlFetchApp.fetch(SLACK_HOOK, {
    method: 'post', contentType: 'application/json',
    payload: JSON.stringify({text: buildSlack(summary, insight)})
  });
}

function startOfWeek(d) {
  const x = new Date(d);
  x.setHours(0,0,0,0);
  x.setDate(x.getDate() - x.getDay() + (x.getDay() === 0 ? -6 : 1)); // 월요일 시작
  return x;
}
function avg(arr) { return arr.length ? arr.reduce((a,b)=>a+b,0)/arr.length : 0; }
function fmt(d) { return d.toISOString().slice(0,10); }

function judgeStatus(val, meta) {
  if (!meta.target) return '—';
  const reached = meta.direction === 'up' ? val >= meta.target : val <= meta.target;
  return reached ? '✅ 달성' : '⚠️ 미달';
}

function aiCommentary(summary) {
  const prompt = `
당신은 경영진 보고를 작성하는 시니어 분석가입니다.
다음 주간 KPI를 보고 한국어로 간결하게 작성하세요.

규칙:
- 사실만, 추측 금지
- 5문장 이내
- 1) 가장 좋은 변화 1건  2) 가장 우려되는 변화 1건  3) 다음 주 권고 1건
- "~합니다"체

데이터:
${JSON.stringify(summary, null, 2)}

JSON으로 반환:
{"highlight":"...","concern":"...","recommendation":"..."}
`;
  const url = `https://generativelanguage.googleapis.com/v1beta/models/gemini-2.5-flash:generateContent?key=$
{GEMINI_KEY}`;
  const res = UrlFetchApp.fetch(url, {
    method:'post', contentType:'application/json',
    payload: JSON.stringify({
      contents:[{parts:[{text:prompt}]}],
      generationConfig:{responseMimeType:'application/json'}
    })
  });
  return JSON.parse(JSON.parse(res.getContentText()).candidates[0].content.parts[0].text);
}

function buildReportHTML(summary, insight, weekStart) {
  let rows = summary.map(s => {
    const arrow = s.wow > 1 ? '🔺' : s.wow < -1 ? '🔻' : '➖';
    const wowStr = s.wow ? `${s.wow.toFixed(1)}%` : '—';
    return `<tr>
      <td style="padding:.5rem;border-bottom:1px solid #eee;">${s.indicator}</td>
      <td style="padding:.5rem;border-bottom:1px solid #eee;text-align:right;">${s.thisAvg.toFixed(1)} ${s.unit||''}</
td>
      <td style="padding:.5rem;border-bottom:1px solid #eee;text-align:right;">${s.lastAvg.toFixed(1)}</td>
      <td style="padding:.5rem;border-bottom:1px solid #eee;text-align:right;">${arrow} ${wowStr}</td>
      <td style="padding:.5rem;border-bottom:1px solid #eee;text-align:right;">${s.target||'—'}</td>
```

```
      <td style="padding:.5rem;border-bottom:1px solid #eee;">${s.status}</td>
    </tr>`;
  }).join('');

  return `<div style="font-family:'Noto Sans KR',sans-serif;max-width:760px;line-height:1.6;color:#222;">
    <h2 style="border-bottom:2px solid #b45309;padding-bottom:.5rem;">주간 KPI 리포트</h2>
    <p style="color:#6a604f;">기준 주: ${fmt(weekStart)} ~ ${fmt(new Date())}</p>

    <h3>📌 핵심 코멘트</h3>
    <p><b>가장 좋은 변화 — </b>${insight.highlight}</p>
    <p><b>우려 포인트 — </b>${insight.concern}</p>
    <p><b>다음 주 권고 — </b>${insight.recommendation}</p>

    <h3 style="margin-top:1.5rem;">📊 지표별 표</h3>
    <table style="width:100%;border-collapse:collapse;font-size:.95em;">
      <thead style="background:#fbf6ec;">
        <tr><th style="padding:.5rem;text-align:left;">지표</th>
            <th style="padding:.5rem;text-align:right;">이번주 평균</th>
            <th style="padding:.5rem;text-align:right;">지난주 평균</th>
            <th style="padding:.5rem;text-align:right;">WoW</th>
            <th style="padding:.5rem;text-align:right;">목표</th>
            <th style="padding:.5rem;text-align:left;">상태</th></tr>
      </thead>
      <tbody>${rows}</tbody>
    </table>
    <p style="color:#999;font-size:.85em;margin-top:2rem;">자동 생성 · 원천 데이터: 회사 KPI 시트</p>
  </div>`;
}

function buildSlack(summary, insight) {
  const top = summary.map(s => `• ${s.indicator}: ${s.thisAvg.toFixed(1)}${s.unit||''} (${s.wow>0?'+':''}$
{s.wow.toFixed(1)}% WoW) ${s.status}`).join('\n');
  return `*📊 주간 KPI 리포트*\n\n${top}\n\n*하이라이트*: ${insight.highlight}\n*우려*: ${insight.concern}\n*권고*: $
{insight.recommendation}`;
}
```

강의 시연 포인트

1. KPI 시트의 한 셀을 일부러 큰 값으로 수정 → `weeklyReport()` 즉시 실행 → AI 코멘트가 "특정 지표 급등에 주목" 등으로 바뀌는 모습
2. `kpi_meta`의 `direction`을 `down`으로 바꿔 "이탈율 같은 부정 지표도 같은 코드로 처리됨" 시연
3. 임원에게 보낼 메일 미리보기로 "Excel 떡칠 보고서 vs 한 장 리포트" 비교

트러블슈팅

증상	원인	해결
WoW가 모두 0	일자가 문자열로 들어옴	`kpi_daily`의 날짜 컬럼 형식을 "날짜"로 강제
AI 코멘트가 일반론	데이터 너무 적음	`kpi_daily`에 최소 14일치 데이터 누적 후 사용
트리거가 한 번도 안 돌아감	인증 미완료	Apps Script에서 한 번 수동 실행하여 권한 부여
일부 지표 누락	meta에 등록 안 됨	`kpi_meta`에 추가하지 않아도 표시되지만, 목표/상태가 비어 보임

응용 아이디어

- **부서별 분기**: 시트에 `부서` 컬럼 추가 → 부서별 메일을 다른 수신자에 발송
- **Looker Studio 연동**: Apps Script가 Sheet에 적재한 결과를 Looker가 시각화
- **이상 감지 강화**: 이번주 값이 표준편차 ±2σ 벗어나면 즉시 임원 알림 분리
- **OKR 연결**: `kpi_meta`에 `okr_id` 컬럼 추가하여 OKR 트래킹과 통합

HR-01 · HR 자동화

신규 입사자 온보딩 키트

합격자 한 줄 입력 → 환영 메일·90일 체크리스트·1on1 캘린더·Slack 공지 자동 생성.

난이도	★★
예상 소요	**60분**
전제 조건	**구글 Workspace + Docs 템플릿 2개 + (선택) Slack**
월 비용	**0원**

⚡ **셋업 전에 결과부터:** 같은 로직을 브라우저에서 입력 한 번으로 실행하는 미니 도구가 있습니다 → `nedabah.org/auto/tools/onboarding-kit/`

한 줄 핵심: 인사 담당자가 Sheet에 합격자 정보 한 줄을 넣으면, 환영 메일·계정 신청서·90일 체크리스트·1on1 일정·슬랙 초대 안내가 자동 생성·발송된다.

왜 이 자동화인가

항목	수동(Before)	자동(After)
입사자 1인 준비 시간	3~5시간	10분(데이터 입력)
누락 발생률	높음 (장비·계정·소개)	0에 수렴
입사 첫날 만족도	부서마다 격차	표준화
연 50명 채용 시 절감	—	**약 200시간 + 휴먼에러 사실상 제거**

적용 시나리오: 정규직 채용·인턴·계약직·파트너 입사.

구성요소

- Google Sheet (입사자 마스터)
- Google Docs 템플릿 (환영 레터·체크리스트)
- Gmail (자동 발송)
- Google Calendar (1on1·교육 일정)
- Slack Webhook (HR 채널 공지)
- Apps Script (편집 트리거)

셋업 가이드

Step 1. 입사자 마스터 시트

탭 `hires`: | 이름 | 이메일 | 입사일 | 부서 | 직책 | 멘토 | 멘토이메일 | 상태 | |---|---|---|---|---|---|---|---| | 김신입 | sin@example.com | 2026-05-12 | 마케팅 | 매니저 | 박선배 | park@example.com | (자동) |

> 마지막 컬럼은 비워두세요. 스크립트가 `완료` 표시.

Step 2. Docs 템플릿 두 개 만들기

1. **환영 레터** (`{{이름}}` , `{{입사일}}` , `{{부서}}` , `{{멘토}}` 변수 포함)
2. **90일 체크리스트** (Day 1 · Day 7 · Day 30 · Day 60 · Day 90 항목)

각 Doc URL의 ID 복사 → 스크립트 속성에 등록.

Step 3. Apps Script 속성

- `MASTER_SHEET_ID`
- `WELCOME_TEMPLATE_ID`
- `CHECKLIST_TEMPLATE_ID`
- `OUTPUT_FOLDER_ID` (생성된 Doc 저장 폴더)
- `SLACK_HOOK` (HR 공지 채널)
- `HR_FROM_NAME` (메일 발신 표시명)

Step 4. 설치형 트리거 등록 (단순 트리거 아님, 중요)

> **왜 설치형인가**: 단순 `onEdit` 은 권한 한도가 낮아 `GmailApp.sendEmail` , `DriveApp.getFolderById` , `CalendarApp.createEvent` 같은 외부 서비스를 호출할 수 없습니다. 반드시 설치형 트리거로 등록해야 합니다.

1. 스크립트 상단의 **`installTrigger`** 함수를 한 번 실행 → 권한 동의 화면이 뜨면 모두 허용
2. Apps Script 좌측 **트리거(시계 아이콘)** 메뉴에서 `onEditInstallable` 이 `from spreadsheet → on edit` 으로 등록되어 있는지 확인

완성 코드 (`script.gs`)

```
const PROPS = PropertiesService.getScriptProperties();
const SHEET_ID  = PROPS.getProperty('MASTER_SHEET_ID');
const TPL_WEL   = PROPS.getProperty('WELCOME_TEMPLATE_ID');
const TPL_CHK   = PROPS.getProperty('CHECKLIST_TEMPLATE_ID');
const FOLDER    = PROPS.getProperty('OUTPUT_FOLDER_ID');
const SLACK     = PROPS.getProperty('SLACK_HOOK') || '';
const FROM_NAME = PROPS.getProperty('HR_FROM_NAME') || 'HR팀';

/**
 * 셋업 시 1회 실행: 설치형 onEdit 트리거를 등록한다.
```

```
 * 단순 트리거(onEdit)는 외부 서비스 권한이 없어 메일/캘린더/Drive 호출이 실패한다.
 */
function installTrigger() {
  const ss = SpreadsheetApp.openById(SHEET_ID);
  // 기존 동일 트리거가 있으면 제거 (중복 방지)
  ScriptApp.getProjectTriggers()
    .filter(t => t.getHandlerFunction() === 'onEditInstallable')
    .forEach(t => ScriptApp.deleteTrigger(t));
  ScriptApp.newTrigger('onEditInstallable').forSpreadsheet(ss).onEdit().create();
  Logger.log('설치형 onEdit 트리거 등록 완료');
}

function onEditInstallable(e) {
  const sheet = e.source.getActiveSheet();
  if (sheet.getName() !== 'hires') return;
  const row = e.range.getRow();
  if (row === 1) return;

  const data = sheet.getRange(row, 1, 1, 8).getValues()[0];
  const [name, email, joinDate, dept, role, mentor, mentorEmail, status] = data;
  if (status === '완료' || !name || !email || !joinDate) return;

  // 1) 환영 레터 + 체크리스트 Doc 생성
  const welcomeDoc = copyAndFill(TPL_WEL, `[환영] ${name} 님 - ${fmt(joinDate)}`, {
    이름: name, 입사일: fmt(joinDate), 부서: dept, 직책: role, 멘토: mentor || '추후 안내'
  });
  const checklistDoc = copyAndFill(TPL_CHK, `[체크리스트] ${name} 90일 플랜`, {
    이름: name, 입사일: fmt(joinDate), 부서: dept, 멘토: mentor || ''
  });

  // 2) 환영 메일 (입사자 + 멘토 cc)
  const body = buildWelcomeBody(name, joinDate, dept, role, mentor, welcomeDoc.url, checklistDoc.url);
  GmailApp.sendEmail(email, `${name} 님, ${fmt(joinDate)} 입사를 환영합니다`, body, {
    name: FROM_NAME,
    cc: mentorEmail || '',
    htmlBody: body
  });

  // 3) 캘린더 — Day 1 환영미팅 + Day 7·30·90 체크인 자동 생성
  const cal = CalendarApp.getDefaultCalendar();
  const join = new Date(joinDate);
  cal.createEvent(`${name} 환영 미팅`, atHour(join, 10), atHour(join, 11), {guests: `${email},${mentorEmail||''}`});
  [7, 30, 90].forEach(d => {
    const day = addDays(join, d);
    cal.createEvent(`${name} ${d}일 체크인`, atHour(day, 14), atHour(day, 14, 30), {guests: `${email},${mentorEmail||''}
`});
  });

  // 4) 슬랙 공지
  if (SLACK) {
    UrlFetchApp.fetch(SLACK, {
      method: 'post', contentType: 'application/json',
      payload: JSON.stringify({text:
        `🎉 *신규 입사 안내* \n*${name}* 님이 ${fmt(joinDate)} *${dept}/${role}* 으로 합류합니다.\n멘토: ${mentor||'미정'}
\n환영 한 마디 남겨주세요.`
      })
    });
  }

  // 5) 상태 업데이트
  sheet.getRange(row, 8).setValue('완료');
  sheet.getRange(row, 9).setValue(welcomeDoc.url);
  sheet.getRange(row, 10).setValue(checklistDoc.url);
}

function copyAndFill(templateId, title, vars) {
  const folder = DriveApp.getFolderById(FOLDER);
```

```
  const file = DriveApp.getFileById(templateId).makeCopy(title, folder);
  const doc = DocumentApp.openById(file.getId());
  const body = doc.getBody();
  for (const k in vars) {
    body.replaceText(`{{${k}}}`, String(vars[k]));
  }
  doc.saveAndClose();
  return {id: file.getId(), url: file.getUrl()};
}

function buildWelcomeBody(name, joinDate, dept, role, mentor, welcomeUrl, checklistUrl) {
  return `<div style="font-family:'Noto Sans KR',sans-serif;line-height:1.7;max-width:620px;">
    <h2 style="color:#b45309;">${name} 님, 환영합니다 🎉</h2>
    <p>${fmt(joinDate)} ${dept}/${role}로 입사하시는 ${name} 님께 첫날을 위한 안내를 드립니다.</p>
    <ul>
      <li>📄 <a href="${welcomeUrl}">환영 레터(개인 맞춤)</a></li>
      <li>✅ <a href="${checklistUrl}">90일 온보딩 체크리스트</a></li>
      <li>👥 멘토: ${mentor || '추후 배정'}</li>
      <li>🗓️ 첫날 10:00 환영 미팅이 캘린더에 등록되었습니다.</li>
    </ul>
    <p>입사 전 궁금한 점은 언제든 회신주세요. 좋은 인연이 되길 기대합니다.</p>
    <p style="color:#6a604f;">— ${FROM_NAME}</p>
  </div>`;
}

function fmt(d) { return new Date(d).toISOString().slice(0,10); }
function atHour(d, h, m=0) { const x = new Date(d); x.setHours(h,m,0,0); return x; }
function addDays(d, n) { const x = new Date(d); x.setDate(x.getDate()+n); return x; }
```

강의 시연 포인트

1. 시트에 입사자 한 줄을 시연용으로 입력 → 캘린더·메일·슬랙·드라이브가 동시에 반응
2. Docs 템플릿의 `{{이름}}` 같은 변수가 어떻게 치환되는지 강조
3. "왜 이 작업은 사람이 해야 하는가" 토론: 멘토 매칭·문화 안내 영상은 자동화 대상에서 의도적으로 제외했음을 설명

트러블슈팅

증상	원인	해결
`onEdit` 미발동 / 권한 오류	단순 트리거는 외부 서비스 호출 권한이 없음	셋업 Step 4에 따라 `installTrigger()` 를 1회 실행해 설치형 트리거로 등록
캘린더 이벤트 중복 생성	같은 행을 다시 편집	`status` 컬럼이 `완료` 면 조기 return — 이미 처리됨
환영 메일 스팸 분류	발신 도메인/SPF 미설정	Workspace 도메인 발신 + SPF/DKIM 설정
멘토 미정 시 캘린더 오류	guests 빈값	`mentorEmail` 이 빈문자열일 때 분기 처리(이미 코드에 반영)

응용 아이디어

- **계정 신청 연동**: 입사일 −7일에 IT팀 채널로 계정 신청 카드 발송

- **온보딩 진행률**: 체크리스트 Doc의 체크 항목을 주 1회 스캔해 시트에 진행률 적재
- **퇴사자 오프보딩 미러**: 같은 패턴으로 퇴사 절차(자료 인계·계정 정리) 자동화
- **다국어**: 외국인 직원의 경우 환영 레터를 영어 템플릿으로 분기

HR-02 · HR 자동화

휴가 신청 슬랙 승인 워크플로우

Form → 팀장 Slack 승인 카드 → 캘린더·연차 잔여 시트 자동 갱신.

난이도	★★★
예상 소요	**90분**
전제 조건	**Slack App(Bot Token) + Form + 팀장 Slack ID 매핑**
월 비용	**0원**

⚡ **셋업 전에 결과부터:** 같은 로직을 브라우저에서 입력 한 번으로 실행하는 미니 도구가 있습니다 → `nedabah.org/auto/tools/leave-summary/`

> **한 줄 핵심**: 직원이 Google Form에 휴가를 신청하면, 팀장 슬랙에 "승인/반려" 버튼 카드가 가고, 승인 시 캘린더·연차 잔여 시트가 자동으로 갱신된다.

왜 이 자동화인가

항목	수동(Before)	자동(After)
신청→승인 평균 시간	2~3일	평균 30분
연차 잔여 계산 오류	종종 발생	즉시 갱신·검증
팀 캘린더 누락	흔함	자동 등록
HR 인보크 횟수/주	많음	거의 0

적용 시나리오: 연차·반차·재택근무 신청·외근 등록.

구성요소

- Google Form (휴가 신청)
- Google Sheet (응답 + 잔여 잔고 + 마스터)
- Google Calendar (전사 휴가 캘린더)
- Slack App (Bot Token + Block Kit) — `chat.postMessage`로 승인 카드 발송, 인터랙션 응답은 doPost 응답 본문의 `replace_original` 으로 처리(별도 `chat.update` 호출 불필요)
- Apps Script Web App (Slack 인터랙션 콜백 수신) + 설치형 onEdit 트리거(비동기 후속 작업)

> **운영 보안 권고**: 본 가이드는 강의 시연·소규모 팀 사용을 전제로 합니다. 외부에서 임의로 `doPost` URL을 호출해 위조 페이로드를 보낼 가능성이 있으므로, 프로덕션에서는 Slack의 `X-Slack-Signature` 헤더와 `Slack Signing Secret`으로 HMAC 검증을 추가하세요. 검증 패턴은 가이드 끝 "응용 아이디어"에 요약되어 있습니다.

셋업 가이드

Step 1. Google Form

필수 질문: 1. 이름 (단답) 2. 직원 이메일 (단답, "응답을 기록할 수 있도록 이메일 수집"으로 자동 채움 권장) 3. 종류 (객관식: 연차 / 반차 오전 / 반차 오후 / 재택) 4. 시작일 (날짜) 5. 종료일 (날짜) 6. 사유 (장문) 7. 팀장 이메일 (단답)

응답 → 시트로 연결.

Step 2. 시트 추가 탭 만들기

탭 `balances` : `이메일 | 연초잔고 | 사용 | 잔여` **탭** `team_lead_slack` : `팀장이메일 | slack_user_id` (Slack 멤버 ID; @팀장 → 옵션 → 멤버 ID 복사)

Step 3. Slack App 설정

1. https://api.slack.com/apps → 새 앱 → Bot Token Scopes: `chat:write`, `chat:write.public`, `users:read`, `users:read.email`
2. **Interactivity & Shortcuts** ON → Request URL = (Apps Script Web App URL — Step 5에서 받음)
3. Bot User OAuth Token 복사 → Apps Script 속성 `SLACK_BOT_TOKEN`

Step 4. Apps Script 속성 등록

- `RESPONSE_SHEET_ID`
- `CALENDAR_ID` (전사 캘린더 ID — 캘린더 설정에서 복사)
- `SLACK_BOT_TOKEN`

Step 5. 트리거 + Web App 배포

1. 트리거 1: `onFormSubmit` → 폼 제출 시 실행 (Apps Script 트리거 메뉴에서 추가)
2. 트리거 2: 스크립트 편집기에서 `installApprovalTrigger` 함수를 1회 실행 → 권한 동의 후 설치형 onEdit 트리거가 등록됨 (단순 트리거가 아니어야 GmailApp/CalendarApp 호출 가능)
3. **배포 → 새 배포 → 웹 앱** → 실행: 본인 / 액세스: 누구나 → URL 복사 → Slack App의 Interactivity Request URL에 붙여넣기

> **왜 트리거가 두 개인가**: Slack은 인터랙션 응답을 3초 안에 받지 못하면 사용자에게 오류를 노출합니다. 따라서 `doPost`는 sheet 상태만 갱신하고 즉시 응답하며, 캘린더 등록·연차 차감·신청자 메일은 sheet 변화를 감지하는 별도 설치형 트리거가 비동기로 처리합니다.

완성 코드 (`script.gs`)

```
const PROPS = PropertiesService.getScriptProperties();
const SHEET_ID  = PROPS.getProperty('RESPONSE_SHEET_ID');
const CAL_ID    = PROPS.getProperty('CALENDAR_ID');
const SLACK_TOK = PROPS.getProperty('SLACK_BOT_TOKEN');

// ─────────────────────────────────────────────────────────────
// 1) 폼 제출 시: 팀장 슬랙으로 승인 요청 카드 발송
// ─────────────────────────────────────────────────────────────
function onFormSubmit(e) {
  const ss = SpreadsheetApp.openById(SHEET_ID);
  const responses = ss.getSheets()[0]; // 폼 응답 시트
  const lastRow = responses.getLastRow();
  const row = responses.getRange(lastRow, 1, 1, responses.getLastColumn()).getValues()[0];

  const [ts, email, name, type, start, end, reason, leadEmail] = row;
  const days = calcDays(type, start, end);

  // 잔여 확인
  const bal = getBalance(email);
  if (type === '연차' && bal && bal.remaining < days) {
    notifyApplicant(email, `잔여(${bal.remaining}일)보다 신청(${days}일)이 큽니다. HR에 문의하세요.`);
    return;
  }

  const slackUid = lookupTeamLeadSlack(leadEmail);
  if (!slackUid) {
    notifyApplicant(email, `팀장 Slack ID 미등록. HR에 알려주세요.`);
    return;
  }

  const payload = {
    channel: slackUid,
    text: `${name}의 ${type} 신청이 도착했습니다`,
    blocks: [
      {type:'header', text:{type:'plain_text', text:`📋 ${name} - ${type} 신청`}},
      {type:'section', fields: [
        {type:'mrkdwn', text:`*기간*\n${fmt(start)} ~ ${fmt(end)} (${days}일)`},
        {type:'mrkdwn', text:`*잔여(연차기준)*\n${bal? bal.remaining: '—'}일`},
      ]},
      {type:'section', text:{type:'mrkdwn', text:`*사유*\n${reason || '—'}`}},
      {type:'actions', block_id: `req_${lastRow}`, elements: [
        {type:'button', style:'primary', text:{type:'plain_text', text:'승인'},
         value: JSON.stringify({row: lastRow, action: 'approve'}),
         action_id: 'approve'},
        {type:'button', style:'danger',  text:{type:'plain_text', text:'반려'},
         value: JSON.stringify({row: lastRow, action: 'reject'}),
         action_id: 'reject'}
      ]}
    ]
  };
  const r = slackPost('chat.postMessage', payload);
  // channel·ts·상태 저장 → doPost에서 meta 구성용(chat.update API는 호출하지 않으며,
  // 메시지 갱신은 doPost 응답 본문의 replace_original 으로 처리한다)
  responses.getRange(lastRow, responses.getLastColumn()+1).setValue(`${r.channel}|${r.ts}|대기`);
}

// ─────────────────────────────────────────────────────────────
// 2) Slack 인터랙션 콜백: 버튼 눌림 처리
//
// Slack은 Interactive payload에 대한 HTTP 응답을 3초 안에 받지 못하면
// "This app took too long to respond" 오류를 노출한다. 따라서 doPost는
// (a) 빠른 sheet 상태 갱신만 수행하고 (b) 즉시 Slack에 message 교체 응답을
// 반환한다. 캘린더 등록·연차 잔여 갱신·신청자 메일은 sheet 상태 컬럼을
// 감시하는 설치형 onEdit 트리거(`onApprovalStatusChange`)에서 비동기로
```

```
// 처리한다. 이 분리가 라이브 시연 안정성의 핵심이다.
// ──────────────────────────────────────────────────────────────
function doPost(e) {
  const payload = JSON.parse(e.parameter.payload);
  const action = payload.actions[0];
  const value = JSON.parse(action.value);
  const userName = payload.user.name;

  const ss = SpreadsheetApp.openById(SHEET_ID);
  const sheet = ss.getSheets()[0];
  const lastCol = sheet.getLastColumn();
  const row = sheet.getRange(value.row, 1, 1, lastCol).getValues()[0];
  const [ts, email, name, type] = row;

  // 멱등성 가드: 이미 처리된 행이면 무시 (사용자가 메시지를 새로고쳐 두 번 누르는 경우 방지)
  const meta = String(row[lastCol - 1] || '').split('|');
  if (meta[2] && meta[2] !== '대기') {
    return jsonResponse({text: `이미 ${meta[2]} 처리된 신청입니다.`, response_type: 'ephemeral'});
  }

  // sheet의 마지막 컬럼(`channel|ts|상태`)을 갱신 — onApprovalStatusChange 트리거가 이 변화를 감지해 무거운 작업을 수행
  const decided = value.action === 'approve' ? '승인' : '반려';
  sheet.getRange(value.row, lastCol).setValue(`${meta[0]||payload.channel.id}|${meta[1]||payload.message.ts}|${decided}|
${userName}`);

  // 즉시 Slack에 message 교체 응답 — chat.update API 호출 불필요 (왕복 시간 절약)
  const resultText = decided === '승인'
    ? `✅ ${userName} 님이 ${name}의 ${type} *승인* 했습니다.`
    : `❌ ${userName} 님이 ${name}의 ${type} *반려* 했습니다.`;
  return jsonResponse({
    replace_original: true,
    text: resultText,
    blocks: [
      ...payload.message.blocks.filter(b => b.type !== 'actions'),
      {type:'context', elements:[{type:'mrkdwn', text: resultText + ' (후속 처리 진행 중)'}]}
    ]
  });
}

function jsonResponse(obj) {
  return ContentService.createTextOutput(JSON.stringify(obj))
    .setMimeType(ContentService.MimeType.JSON);
}

/**
 * 설치형 onEdit 트리거. 폼 응답 시트의 마지막 컬럼이 `...|승인` 또는 `...|반려`로
 * 바뀌면 캘린더·연차잔여·신청자 메일을 처리한다. 셋업 시 `installApprovalTrigger()`를 1회 실행.
 */
function onApprovalStatusChange(e) {
  if (!e || !e.range) return;
  const sheet = e.range.getSheet();
  // 트리거는 폼 응답 시트(첫 시트)의 메타 컬럼 변화만 처리한다.
  // 시트 인덱스 비교(getIndex === 1)가 이름 비교보다 가벼워 추가 openById 호출이 불필요하다.
  if (sheet.getIndex() !== 1) return;
  const lastCol = sheet.getLastColumn();
  if (e.range.getColumn() !== lastCol) return; // 메타 컬럼만 감시

  const r = e.range.getRow();
  if (r === 1) return;
  const meta = String(e.value || '').split('|');
  const decided = meta[2];
  if (decided !== '승인' && decided !== '반려') return;
  // 처리 멱등성: 4번째 토큰("done")이 이미 있으면 skip
  if (meta[4] === 'done') return;

  const row = sheet.getRange(r, 1, 1, lastCol).getValues()[0];
  const [ts, email, name, type, start, end, reason, leadEmail] = row;
```

```
  const days = calcDays(type, start, end);

  if (decided === '승인') {
    CalendarApp.getCalendarById(CAL_ID).createAllDayEvent(
      `${name} ${type}`, new Date(start), new Date(new Date(end).getTime()+86400000)
    );
    if (type === '연차') updateBalance(email, days);
    notifyApplicant(email, `✅ ${type} 승인되었습니다. (${fmt(start)} ~ ${fmt(end)})`);
  } else {
    notifyApplicant(email, `❌ ${type} 신청이 반려되었습니다. 팀장과 협의해주세요.`);
  }
  // done 마킹으로 멱등성 보장
  sheet.getRange(r, lastCol).setValue(`${meta[0]}|${meta[1]}|${decided}|${meta[3]||''}|done`);
}

/**
 * 셋업 시 1회 실행: onApprovalStatusChange를 설치형 onEdit 트리거로 등록.
 * 단순 트리거는 GmailApp/CalendarApp 호출 권한이 없으므로 반드시 설치형이어야 한다.
 */
function installApprovalTrigger() {
  const ss = SpreadsheetApp.openById(SHEET_ID);
  ScriptApp.getProjectTriggers()
    .filter(t => t.getHandlerFunction() === 'onApprovalStatusChange')
    .forEach(t => ScriptApp.deleteTrigger(t));
  ScriptApp.newTrigger('onApprovalStatusChange').forSpreadsheet(ss).onEdit().create();
  Logger.log('승인 상태 변경 트리거 등록 완료');
}

// ─────────────────────────────────────────────────────────────
// 헬퍼
// ─────────────────────────────────────────────────────────────
function calcDays(type, start, end) {
  if (type.indexOf('반차') >= 0) return 0.5;
  if (!end) return 1;
  const ms = new Date(end) - new Date(start);
  return Math.max(1, Math.round(ms / 86400000) + 1);
}
function getBalance(email) {
  const sh = SpreadsheetApp.openById(SHEET_ID).getSheetByName('balances');
  const data = sh.getDataRange().getValues();
  for (let i = 1; i < data.length; i++) {
    if (data[i][0] === email) {
      return {row: i+1, initial: data[i][1], used: data[i][2], remaining: data[i][3]};
    }
  }
  return null;
}
function updateBalance(email, days) {
  const bal = getBalance(email); if (!bal) return;
  const sh = SpreadsheetApp.openById(SHEET_ID).getSheetByName('balances');
  sh.getRange(bal.row, 3).setValue((bal.used||0) + days);
  sh.getRange(bal.row, 4).setValue((bal.initial||0) - ((bal.used||0)+days));
}
function lookupTeamLeadSlack(leadEmail) {
  const sh = SpreadsheetApp.openById(SHEET_ID).getSheetByName('team_lead_slack');
  const data = sh.getDataRange().getValues();
  for (let i = 1; i < data.length; i++) if (data[i][0] === leadEmail) return data[i][1];
  return null;
}
function notifyApplicant(email, text) {
  GmailApp.sendEmail(email, '[휴가 신청 알림]', text);
}
function slackPost(method, body) {
  const r = UrlFetchApp.fetch(`https://slack.com/api/${method}`, {
    method: 'post', contentType: 'application/json; charset=utf-8',
    headers: {Authorization: `Bearer ${SLACK_TOK}`},
    payload: JSON.stringify(body)
```

```
  });
  return JSON.parse(r.getContentText());
}
function fmt(d) { return new Date(d).toISOString().slice(0,10); }
```

강의 시연 포인트

1. 강사가 폼 제출 → 30초 내 팀장(강사 본인) Slack에 카드 도착
2. "승인" 버튼 클릭 → 캘린더·시트가 동시 갱신, Slack 카드 본문도 즉시 업데이트
3. 잔여 부족 시나리오: 연차 잔여를 0으로 미리 세팅한 직원의 신청이 자동 차단되는 모습 시연

트러블슈팅

증상	원인	해결
슬랙 카드 미수신	Bot이 채널/DM에 미초대	팀장 DM은 `users:read` 후 `slackUid` 를 정확히 (U... 형식)
`doPost` 가 동작 안 함	Web App 배포가 "본인만"으로 됨	"누구나"로 재배포, 새 URL을 Slack App에 갱신
캘린더 권한 오류	`CALENDAR_ID` 가 비공개	캘린더 공유 → 스크립트 실행 계정에 변경 권한 부여
Slack에 "took too long to respond" 표시	`doPost` 가 3초 안에 응답 못함	본 가이드 코드는 무거운 작업을 설치형 onEdit 트리거로 분리해 `doPost` 를 0.5~1초 수준으로 유지함. `installApprovalTrigger` 가 등록되어 있는지 확인
같은 신청을 두 번 처리	메시지 새로고침 후 재클릭	sheet 메타 컬럼의 멱등성 가드(`'대기' 외 상태면 무시`)가 자동으로 차단함

응용 아이디어

- **2단계 승인**: 팀장 → 임원 두 단계가 필요한 경우 `state` 컬럼으로 단계 관리
- **연차 자동 적립**: 매월 1일 트리거로 `balances` 에 비례 적립 자동 추가
- **공휴일 보정**: `calcDays` 에서 공공API(공휴일 RSS)로 주말·공휴일 제외
- **이상 신청 감지**: 같은 직원이 단기간 반복 신청할 때 HR에 부드러운 알림

프로덕션 보강: Slack 서명 검증 (HMAC)

운영 환경에서는 외부 위조 페이로드를 차단하기 위해 `doPost` 진입 직후 Slack 서명 검증을 수행하세요. Slack App 관리 화면 → Basic Information → **Signing Secret**을 발급받아 스크립트 속성 `SLACK_SIGNING_SECRET` 에 저장한 뒤, 다음 가드를 코드 최상단에 추가합니다.

```
function verifySlackSignature(e) {
  const secret = PROPS.getProperty('SLACK_SIGNING_SECRET');
  if (!secret) return true; // 셋업 안 했으면 검증 생략 (개발용)
  const ts  = e.parameter['X-Slack-Request-Timestamp'] || e.headers && e.headers['X-Slack-Request-Timestamp'];
  const sig = e.parameter['X-Slack-Signature']         || e.headers && e.headers['X-Slack-Signature'];
```

```
  if (!ts || !sig) return false;
  // 5분 이상 지난 요청은 재전송 공격 가능성으로 거부
  if (Math.abs(Date.now()/1000 - Number(ts)) > 300) return false;
  const base = `v0:${ts}:${e.postData && e.postData.contents || ''}`;
  const mac  = Utilities.computeHmacSha256Signature(base, secret)
    .map(b => ('0' + (b & 0xff).toString(16)).slice(-2)).join('');
  return `v0=${mac}` === sig;
}
```

> Apps Script Web App은 표준 헤더 접근에 제한이 있어 Slack이 보내는 서명 정보를 캡처하려면 별도 프록시(Cloud Functions, Cloudflare Worker 등)를 두는 패턴이 가장 안전합니다. 강의에서는 "왜 검증이 필요한지·어디서 막을지"를 토론 주제로 다루고, 실제 코드 추가는 후속 워크숍에서 다룹니다.

HR-03 · HR 자동화

분기 펄스서베이 + AI 감성 분석

응답 1,000건도 30분 안에. 식별 정보는 AI에 전달되지 않도록 분리 설계.

난이도	★★
예상 소요	**45분**
전제 조건	**Google Form + Gemini API 키**
월 비용	**0원**

⚡ **셋업 전에 결과부터:** 같은 로직을 브라우저에서 입력 한 번으로 실행하는 미니 도구가 있습니다 → `nedabah.org/auto/tools/pulse-analysis/`

> **한 줄 핵심**: 익명 설문을 자동 발송·수집한 뒤, AI가 감성·핵심 주제·위험 신호를 분류해 경영진 1페이지 리포트로 전달한다.

왜 이 자동화인가

항목	수동(Before)	자동(After)
응답 1,000건 코딩	외부 발주 시 200~500만 원	0원 (Gemini 무료 티어로 충분)
분석 소요	1~2주	30분
무기명성	종이/구글 폼만으로는 의심	응답 분리·집계 자동화로 신뢰성↑
액션 연결	보고서로 끝남	주제별 권고가 자동 작성됨

적용 시나리오: 분기 ENPS·번아웃 점검·회식 만족도·리더십 360°.

구성요소

- Google Form (익명 설문 — "응답자 이메일 수집 안 함"으로 설정)
- Google Sheet (응답 + 분석 결과)
- Gemini API (감성·주제 분류)
- Apps Script (배치 분석 + 리포트 생성)
- Gmail (경영진 리포트)

> **개인정보 안전 원칙**: 정성 응답을 AI에 보낼 때 이름·부서 등 식별 정보는 절대 함께 전송하지 않는다. 본 스크립트는 정성 텍스트만 추출하여 전송한다.

셋업 가이드

Step 1. 폼 질문 설계 (예시)

1. 직군 (객관식 — 식별 위험 낮은 큰 그룹만: 본부 단위)
2. 일에 몰입한다 (1~5)
3. 회사 추천의향(ENPS) (0~10)
4. 가장 잘된 점 (장문) ← AI 분석 대상
5. 가장 답답한 점 (장문) ← AI 분석 대상
6. 자유 의견 (장문) ← AI 분석 대상

> **중요**: Google Forms는 응답 시트 첫 컬럼에 **Timestamp**를 자동으로 추가합니다. 따라서 시트 구조는 `A=Timestamp, B=직군, C=몰입, D=ENPS, E=잘된 점, F=답답한 점, G=자유 의견`이 됩니다. 코드의 `enpsCol`, `TEXT_COLS` 상수는 이 1-base 인덱스에 맞춰져 있습니다(폼 질문 순서를 바꾸면 두 상수도 함께 조정).

Step 2. 응답 시트

폼이 자동 생성한 응답 시트를 그대로 사용. 추가 탭 두 개 만들기: - `analyzed`: `행 | 직군 | 감성 | 주제 | 요지` (AI가 채움) - `report`: HR/경영진용 1페이지 리포트 결과 저장

Step 3. 스크립트 속성

- `RESPONSE_SHEET_ID`
- `GEMINI_API_KEY`
- `RECIPIENT_EMAIL` (CHRO/CEO)

Step 4. 트리거

- `analyzeAll` — 매주 월요일 새벽 (또는 폼 마감일 다음 날)
- `buildReport` — `analyzeAll` 직후 (5분 차이로 트리거 등록)

완성 코드 (`script.gs`)

```
const PROPS = PropertiesService.getScriptProperties();
const SHEET_ID   = PROPS.getProperty('RESPONSE_SHEET_ID');
const GEMINI_KEY = PROPS.getProperty('GEMINI_API_KEY');
const EMAIL_TO   = PROPS.getProperty('RECIPIENT_EMAIL');

// 응답 시트 1-base 컬럼 인덱스
// A(1)=Timestamp, B(2)=직군, C(3)=몰입(1~5), D(4)=ENPS(0~10),
// E(5)=잘된 점, F(6)=답답한 점, G(7)=자유 의견
const GROUP_COL = 2;          // 직군 (B열)
const TEXT_COLS = [5, 6, 7];  // 정성 응답 컬럼 인덱스 — AI에 전송되는 유일한 데이터
```

```
// ──────────────────────────────────────────────────────────────
// 1) 응답 일괄 분석
// ──────────────────────────────────────────────────────────────
function analyzeAll() {
  const ss  = SpreadsheetApp.openById(SHEET_ID);
  const src = ss.getSheets()[0];
  const dst = ss.getSheetByName('analyzed') || ss.insertSheet('analyzed');
  if (dst.getLastRow() === 0) dst.appendRow(['행', '직군', '감성', '주제', '요지']);

  const data = src.getDataRange().getValues();
  const processed = new Set(dst.getRange('A2:A').getValues().flat().map(String));

  const batch = []; // [{row, group, text}]
  for (let i = 1; i < data.length; i++) {
    if (processed.has(String(i+1))) continue;
    const group = data[i][GROUP_COL - 1]; // 직군 (B열, 1-base index 2)
    const merged = TEXT_COLS.map(c => String(data[i][c-1] || '').trim()).filter(Boolean).join(' / ');
    if (!merged) continue;
    batch.push({row: i+1, group, text: merged});
  }
  if (batch.length === 0) return;

  // 50건씩 배치 처리
  for (let i = 0; i < batch.length; i += 50) {
    const slice = batch.slice(i, i+50);
    const result = classifyBatch(slice);
    slice.forEach((b, idx) => {
      const r = result[idx] || {};
      dst.appendRow([b.row, b.group, r.sentiment || '중립', (r.topics||[]).join(','), r.gist || '']);
    });
    Utilities.sleep(2000);
  }
}

function classifyBatch(items) {
  // 식별정보 분리: text만 전송, group은 후처리
  const prompt = `
각 응답에 대해 한국어로 분석하세요. JSON 배열만 반환.
- sentiment: positive/negative/neutral
- topics: 1~3개 주제(짧은 한국어 명사구) — 회사문화/보상/성장/리더십/업무량/복지/관계/제도/도구/번아웃 등에서 자유롭게
- gist: 한 문장 요약(15자 이내)

응답:
${items.map((it,i)=>`[${i+1}] ${it.text}`).join('\n')}

반환 형식: [{"sentiment":"...","topics":["..."],"gist":"..."}]
`;
  const url = `https://generativelanguage.googleapis.com/v1beta/models/gemini-2.5-flash:generateContent?key=$
{GEMINI_KEY}`;
  const res = UrlFetchApp.fetch(url, {
    method:'post', contentType:'application/json',
    payload: JSON.stringify({
      contents:[{parts:[{text: prompt}]}],
      generationConfig:{responseMimeType:'application/json'}
    })
  });
  return JSON.parse(JSON.parse(res.getContentText()).candidates[0].content.parts[0].text);
}

// ──────────────────────────────────────────────────────────────
// 2) 1페이지 경영진 리포트
// ──────────────────────────────────────────────────────────────
function buildReport() {
  const ss = SpreadsheetApp.openById(SHEET_ID);
  const src = ss.getSheets()[0];
  const ana = ss.getSheetByName('analyzed').getDataRange().getValues();
```

```
  // 정량
  const all = src.getDataRange().getValues();
  // Forms는 첫 컬럼에 Timestamp를 자동 추가하므로 ENPS는 D열(1-base 4)에 위치한다.
  const enpsCol = 4;
  const enpsScores = all.slice(1).map(r => Number(r[enpsCol-1])).filter(n => !isNaN(n));
  const enps = enpsScore(enpsScores);

  // 정성: 주제 카운트 + 감성 비율
  const topicCount = {}, byTopicSamples = {};
  let pos = 0, neg = 0, neu = 0;
  for (let i = 1; i < ana.length; i++) {
    const sentiment = ana[i][2];
    const topics = String(ana[i][3]).split(',').map(s => s.trim()).filter(Boolean);
    const gist = ana[i][4];
    topics.forEach(t => {
      topicCount[t] = (topicCount[t]||0) + 1;
      byTopicSamples[t] = byTopicSamples[t] || [];
      if (byTopicSamples[t].length < 3) byTopicSamples[t].push(gist);
    });
    if (sentiment === 'positive') pos++;
    else if (sentiment === 'negative') neg++;
    else neu++;
  }

  const topTopics = Object.entries(topicCount).sort((a,b)=>b[1]-a[1]).slice(0, 6);
  const insight = aiExecutiveSummary(enps, {pos, neg, neu}, topTopics, byTopicSamples);

  const html = renderReport(enps, {pos, neg, neu}, topTopics, byTopicSamples, insight);
  if (EMAIL_TO) GmailApp.sendEmail(EMAIL_TO, `[펄스서베이] ${new Date().toISOString().slice(0,10)} 분석 리포트`, '',
{htmlBody: html});

  const rep = ss.getSheetByName('report') || ss.insertSheet('report');
  rep.appendRow([new Date(), enps, JSON.stringify(insight)]);
}

function enpsScore(scores) {
  if (!scores.length) return 0;
  const promoters = scores.filter(s => s >= 9).length;
  const detractors = scores.filter(s => s <= 6).length;
  return Math.round((promoters - detractors) / scores.length * 100);
}

function aiExecutiveSummary(enps, sent, topTopics, samples) {
  const ctx = topTopics.map(([t,c]) => `- ${t}(${c}건): ${(samples[t]||[]).join(' | ')}`).join('\n');
  const prompt = `
당신은 CHRO 코치입니다. 분기 펄스서베이 결과로 경영진용 한 페이지 코멘트를 작성하세요.

데이터:
- ENPS: ${enps}
- 감성 분포: 긍정 ${sent.pos} / 부정 ${sent.neg} / 중립 ${sent.neu}
- 상위 주제:
${ctx}

규칙:
- 추측 금지, 데이터 기반
- 250자 이내
- 1) 가장 시급한 주제 1건과 권고 행동 2개  2) 칭찬할 강점 1건  3) 다음 분기 측정 제안 1건
JSON: {"urgent":"...","actions":["...","..."],"strength":"...","next":"..."}
`;
  const url = `https://generativelanguage.googleapis.com/v1beta/models/gemini-2.5-flash:generateContent?key=$
{GEMINI_KEY}`;
  const res = UrlFetchApp.fetch(url, {method:'post', contentType:'application/json',
    payload: JSON.stringify({contents:[{parts:[{text:prompt}]}], generationConfig:{responseMimeType:'application/
json'}})
  });
  return JSON.parse(JSON.parse(res.getContentText()).candidates[0].content.parts[0].text);
```

```
}

function renderReport(enps, sent, topTopics, samples, ins) {
  const total = sent.pos + sent.neg + sent.neu || 1;
  const topicHtml = topTopics.map(([t,c]) => `<li><b>${t}</b> · ${c}건<br><span style="color:#666;font-size:.92em;">$
{(samples[t]||[]).join(' · ')}</span></li>`).join('');
  return `<div style="font-family:'Noto Sans KR',sans-serif;line-height:1.7;max-width:720px;color:#222;">
    <h2 style="border-bottom:2px solid #b45309;padding-bottom:.4rem;">분기 펄스서베이 — 경영진 요약</h2>
    <p style="color:#6a604f;">${new Date().toLocaleDateString('ko-KR')} 기준</p>

    <h3>정량 지표</h3>
    <ul>
      <li><b>ENPS</b>: ${enps}</li>
      <li><b>감성 분포</b>: 긍정 ${(sent.pos/total*100).toFixed(0)}% / 부정 ${(sent.neg/total*100).toFixed(0)}% / 중립 $
{(sent.neu/total*100).toFixed(0)}%</li>
    </ul>

    <h3>상위 주제 (응답 빈도순)</h3>
    <ul>${topicHtml}</ul>

    <h3>📌 핵심 인사이트</h3>
    <p><b>가장 시급한 이슈 — </b>${ins.urgent}</p>
    <p><b>권고 액션</b></p>
    <ol>${ins.actions.map(a=>`<li>${a}</li>`).join('')}</ol>
    <p><b>강점 — </b>${ins.strength}</p>
    <p><b>다음 분기 측정 제안 — </b>${ins.next}</p>

    <p style="color:#999;font-size:.85em;margin-top:2rem;">자동 생성 · 응답은 익명이며 식별 정보는 AI에 전달되지 않았습니
다.</p>
  </div>`;
}
```

강의 시연 포인트

1. 폼 응답 5건을 미리 준비, `analyzeAll()` 1회 실행 → `analyzed` 탭에 자동 채워지는 모습
2. `buildReport()` 실행 → 메일에 1페이지 리포트 도착
3. **개인정보 안전 강조**: 코드의 `classifyBatch`에서 직군은 별도 보관, AI에는 텍스트만 전송됨을 보여주기

트러블슈팅

증상	원인	해결
`analyzed` 시트가 일부만 채워짐	50건 배치에서 일부 응답 비어 있음	빈 응답 사전 필터링(코드에 반영됨)
Gemini가 라벨을 영어로 반환	프롬프트 강제 부족	시스템 메시지에 "한국어 명사구"를 한 번 더 강조
ENPS가 NaN	객관식 응답이 텍스트로 들어옴	폼 질문을 0~10 척도형으로 정확히 설정
ENPS가 항상 -100	폼 질문 순서를 바꿔 컬럼 인덱스가 틀어짐	본 문서 Step 1의 6개 질문 순서를 그대로 유지하거나, `GROUP_COL` / `enpsCol` / `TEXT_COLS` 상수를 새 순서에 맞게 갱신
직군 식별 위험	본부가 5명 미만	직군 라벨링을 본부→사업부 단위로 묶어 익명성 보장

응용 아이디어

- **부정 응답 즉시 핫라인**: 감성이 negative이면서 키워드에 "괴롭힘/안전" 포함 시 윤리경영 채널로 즉시 알림
- **분기 비교 트렌드**: `report` 탭에 분기별 누적, 다음 회차에 변화 자동 코멘트
- **부서별 미니 리포트**: 본부장 권한 메일 자동 발송(개인 응답이 아닌 본부 단위 집계만)
- **번역**: 외국인 직원 응답이 영어/일어인 경우 분류 전 한국어 번역 단계 추가

MKT-01 · 마케팅 자동화

30일 콘텐츠 캘린더 자동 생성

월 테마 한 줄 → 30일치 채널별 헤드라인·후크·CTA·해시태그 자동 채움.

난이도	★
예상 소요	**30분**
전제 조건	**구글 계정 + Gemini API 키**
월 비용	**0원**

⚡ **셋업 전에 결과부터:** 같은 로직을 브라우저에서 입력 한 번으로 실행하는 미니 도구가 있습니다 → `nedabah.org/auto/tools/content-calendar/`

한 줄 핵심: 한 달 테마 1줄과 채널 목록만 입력하면, AI가 30일치 콘텐츠 캘린더(블로그·SNS·뉴스레터)를 헤드라인·후크·CTA·해시태그까지 자동 생성해 시트에 적재한다.

왜 이 자동화인가

항목	수동(Before)	자동(After)
콘텐츠 캘린더 작성	1~2일	5분
아이디어 고갈	자주	드묾 (AI가 30개 변주)
채널별 톤 분기	누락 빈번	자동
월 절감	—	**약 12시간/마케터 1인**

적용 시나리오: 1인 운영·SMB 마케팅·에이전시·신규 캠페인 런칭.

구성요소

- Google Sheet (입력 + 결과)
- Gemini API
- Apps Script (커스텀 메뉴 + 단일 함수)
- (선택) Notion API — 캘린더 자동 sync

셋업 가이드

Step 1. 시트 4탭 구성

탭 `input`: | 키 | 값 | |---|---| | 월 테마 | 5월 — '실전형 사이드프로젝트' | | 비즈니스 | 1인 코치/컨설턴트 | | 타깃 | 30대 직장인, 부업 시작 단계 | | 채널 | 블로그, 인스타그램, 링크드인, 뉴스레터 | | 톤 | 따뜻하고 단정한, 거품 없는 | | 시작일 | 2026-05-01 |

탭 `calendar`: `날짜 | 채널 | 형식 | 헤드라인 | 후크 | 본문 가이드 | CTA | 해시태그 | 상태`

탭 `pillars` (선택, 입력하면 AI가 균형 있게 분배): | 기둥 | |---| | 마인드셋 | | 실전 케이스 | | 도구 사용법 | | 커뮤니티/Q&A | | 자기 사례 |

탭 `prompts` (수정하면 AI 결과가 바뀜): 시스템 프롬프트 텍스트 1셀.

Step 2. 스크립트 속성

- `SHEET_ID`
- `GEMINI_API_KEY`

Step 3. Apps Script — 메뉴 추가 후 시트에서 클릭으로 실행.

완성 코드 (`script.gs`)

```
const PROPS = PropertiesService.getScriptProperties();
const SHEET_ID  = PROPS.getProperty('SHEET_ID');
const GEMINI    = PROPS.getProperty('GEMINI_API_KEY');

function onOpen() {
  SpreadsheetApp.getUi()
    .createMenu('🤖 콘텐츠 자동화')
    .addItem('30일 캘린더 생성', 'generateCalendar')
    .addItem('선택 행 다시 쓰기', 'regenerateRow')
    .addToUi();
}

function readInputs() {
  const ss = SpreadsheetApp.openById(SHEET_ID);
  const inp = Object.fromEntries(ss.getSheetByName('input').getDataRange().getValues().slice(1));
  const pillars = ss.getSheetByName('pillars').getRange('A2:A').getValues().flat().filter(Boolean);
  const channels = String(inp['채널']||'').split(',').map(s=>s.trim()).filter(Boolean);
  return {
    theme: inp['월 테마'], business: inp['비즈니스'], target: inp['타깃'],
    channels, tone: inp['톤'], startDate: new Date(inp['시작일']),
    pillars
  };
}

function generateCalendar() {
  const ui = SpreadsheetApp.getUi();
  const i = readInputs();
  if (!i.theme || !i.channels.length) {
    ui.alert('input 탭의 월 테마/채널을 먼저 입력하세요.'); return;
  }

  const prompt = buildMonthPrompt(i);
  const items = callGemini(prompt); // [{date, channel, format, headline, hook, body, cta, hashtags}, ...]
```

```
  const cal = SpreadsheetApp.openById(SHEET_ID).getSheetByName('calendar');
  cal.clearContents();
  cal.appendRow(['날짜','채널','형식','헤드라인','후크','본문 가이드','CTA','해시태그','상태']);
  items.forEach(it => cal.appendRow([
    it.date, it.channel, it.format, it.headline, it.hook, it.body, it.cta, (it.hashtags||[]).join(' '), '예정'
  ]));
  ui.alert(`✅ ${items.length}건 생성 완료`);
}

function buildMonthPrompt(i) {
  const pillarLine = i.pillars.length ? `다음 콘텐츠 기둥을 균형 있게 분배: ${i.pillars.join(', ')}` : '주제는 자율적으로 다
양화';
  return `
당신은 ${i.business} 도메인의 시니어 콘텐츠 디렉터입니다.
다음 정보로 30일 콘텐츠 캘린더를 작성하세요.

월 테마: ${i.theme}
타깃: ${i.target}
채널: ${i.channels.join(', ')}
톤: ${i.tone}
시작일: ${i.startDate.toISOString().slice(0,10)}
${pillarLine}

규칙:
- 각 채널 특성에 맞춰 톤·형식 분기 (예: 블로그=long, 인스타=짧은 후크+이미지 가이드, 링크드인=인사이트, 뉴스레터=주제 묶음)
- 같은 주제도 채널별 변주
- 날짜는 시작일부터 30일, 하루에 1~2건 배치 (총 약 30~40건)
- 헤드라인은 12~22자, 후크는 1문장, 본문 가이드는 50자 이내, CTA는 명확한 행동 1개, 해시태그 3~6개
- 자기과시·과장 금지

JSON 배열만 반환:
[{"date":"YYYY-MM-DD","channel":"...","format":"long|short|image|reel|
email","headline":"...","hook":"...","body":"...","cta":"...","hashtags":["..."]}]
`;
}

function callGemini(prompt) {
  const url = `https://generativelanguage.googleapis.com/v1beta/models/gemini-2.5-flash:generateContent?key=${GEMINI}`;
  const res = UrlFetchApp.fetch(url, {
    method:'post', contentType:'application/json',
    payload: JSON.stringify({
      contents:[{parts:[{text: prompt}]}],
      generationConfig:{responseMimeType:'application/json', maxOutputTokens: 8192}
    })
  });
  return JSON.parse(JSON.parse(res.getContentText()).candidates[0].content.parts[0].text);
}

function regenerateRow() {
  const ss = SpreadsheetApp.openById(SHEET_ID);
  const sh = ss.getSheetByName('calendar');
  const r  = sh.getActiveCell().getRow();
  if (r === 1) return;
  const row = sh.getRange(r, 1, 1, 9).getValues()[0];
  const [date, channel, format, headline] = row;

  const i = readInputs();
  const prompt = `
다음 한 건만 새로운 변형으로 다시 작성하세요(같은 날짜·채널·형식 유지).
원본 헤드라인: ${headline}
월 테마: ${i.theme}, 타깃: ${i.target}, 톤: ${i.tone}

JSON 단일 객체:
{"headline":"...","hook":"...","body":"...","cta":"...","hashtags":["..."]}
`;
  const o = JSON.parse(JSON.parse(UrlFetchApp.fetch(
```

```
    `https://generativelanguage.googleapis.com/v1beta/models/gemini-2.5-flash:generateContent?key=${GEMINI}`,
    {method:'post', contentType:'application/json',
     payload: JSON.stringify({contents:[{parts:[{text:prompt}]}], generationConfig:{responseMimeType:'application/
json'}})}
  ).getContentText()).candidates[0].content.parts[0].text);

  sh.getRange(r, 4).setValue(o.headline);
  sh.getRange(r, 5).setValue(o.hook);
  sh.getRange(r, 6).setValue(o.body);
  sh.getRange(r, 7).setValue(o.cta);
  sh.getRange(r, 8).setValue((o.hashtags||[]).join(' '));
}
```

강의 시연 포인트

1. 빈 시트에서 `input` 입력 → 메뉴 한 번 클릭 → 30일치 헤드라인이 30초 내 채워지는 모습
2. 한 행 선택 후 "다시 쓰기" → 같은 주제의 다른 변주 시연
3. 강의 후 워크숍: 학습자가 자기 비즈니스로 직접 시도 — 결과 비교 토론

트러블슈팅

증상	원인	해결
30건 미만 생성	토큰 부족	`maxOutputTokens` 8192 유지 + 프롬프트 단순화
날짜가 시작일 이전	AI가 가끔 거꾸로	후처리에서 정렬 + 시작일 검증 추가
채널별 톤 비슷함	프롬프트 강조 부족	채널별 미니 톤 가이드를 `pillars`에 추가
JSON 깨짐	따옴표 이스케이프	`responseMimeType: 'application/json'` 필수

응용 아이디어

- **이미지 자동 시안**: 헤드라인을 받아 DALL·E API로 인스타 카드 이미지 자동 생성
- **Notion sync**: Notion DB로 양방향 동기화 → 팀 협업
- **A/B 헤드라인**: 한 행에 헤드라인 2개 생성 → 게시 후 클릭률 시트로 회수
- **계절·트렌드 보강**: 발행일 기준 검색 트렌드(네이버 트렌드) 반영하도록 입력 자동화

MKT-02 · 마케팅 자동화

인입 리드 자동 스코어링·배정

룰 60% + AI 40%로 0~100점 스코어링, Hot/Warm/Cold 등급별 담당자 Slack 카드 발송.

난이도	★★
예상 소요	**60분**
전제 조건	**Google Form + Gemini API 키 + Slack Bot**
월 비용	**0원**

⚡ **셋업 전에 결과부터:** 같은 로직을 브라우저에서 입력 한 번으로 실행하는 미니 도구가 있습니다 → `nedabah.org/auto/tools/lead-scoring/`

한 줄 핵심: 문의 폼이 들어오면 AI가 회사·문의 내용을 보고 점수(Hot/Warm/Cold)를 매기고, 룰에 따라 담당자에게 자동 배정·알림한다.

왜 이 자동화인가

항목	수동(Before)	자동(After)
첫 응답까지 시간	평균 6~24시간	2~10분
영업 우선순위 판단	주관적	일관된 룰 + AI
Hot 리드 누락	종종 발생	거의 0
전환율 영향	—	응답시간 5분 이내일 때 21배 ↑ (HBR)

적용 시나리오: B2B 컨설팅·SaaS·교육·강연 의뢰·고가 코칭.

구성요소

- Google Form (문의)
- Google Sheet (응답 + 룰 + 담당자)
- Gemini API (스코어링 보강)
- Slack Webhook (담당자별 채널)
- Apps Script (Form 제출 트리거)

셋업 가이드

Step 1. 폼 질문 (필수만)

1. 회사/소속
2. 이름
3. 이메일
4. 직무 (객관식: 대표/임원/부서장/실무자/기타)
5. 직원 수 (객관식: 1~10/11~50/51~200/200+)
6. 예상 예산 범위 (객관식: 1천만 미만/1천~5천/5천~1억/1억+)
7. 의뢰 내용 (장문)
8. 희망 시작 시기 (객관식: 즉시/1개월/3개월/미정)

Step 2. 시트 구조

탭 `responses`: 폼 자동 생성 **탭 `rules`**: | 조건 | 점수 | |---|---| | 직무=대표 | 30 | | 직무=임원 | 20 | | 직무=실무자 | 5 | | 예산>=1억 | 35 | | 예산 5천~1억 | 25 | | 예산 1천~5천 | 12 | | 직원 200+ | 15 | | 시작=즉시 | 20 | | 시작=1개월 | 10 |

탭 `assignees`: `등급 | 담당자명 | 이메일 | slack_channel | slack_user_id` - HOT 70+ → 대표 본인 - WARM 40~69 → 시니어 매니저 - COLD <40 → 자동 자료만 발송 + 주1회 다이제스트

탭 `log`: `타임스탬프 | 회사 | 점수 | 등급 | 담당자 | AI코멘트`

Step 3. 스크립트 속성

- `RESPONSE_SHEET_ID`
- `GEMINI_API_KEY`
- `SLACK_BOT_TOKEN` (또는 채널별 webhook URL을 시트에 직접 둬도 됨)
- `FROM_NAME`

Step 4. 트리거: `onFormSubmit` 폼 제출 시.

완성 코드 (`script.gs`)

```
const PROPS = PropertiesService.getScriptProperties();
const SHEET_ID = PROPS.getProperty('RESPONSE_SHEET_ID');
const GEMINI   = PROPS.getProperty('GEMINI_API_KEY');
const SLACK_TOK = PROPS.getProperty('SLACK_BOT_TOKEN') || '';
const FROM = PROPS.getProperty('FROM_NAME') || '영업팀';

function onFormSubmit(e) {
  const ss = SpreadsheetApp.openById(SHEET_ID);
  const responses = ss.getSheets()[0];
  const rules = ss.getSheetByName('rules').getDataRange().getValues().slice(1);
  const assignees = ss.getSheetByName('assignees').getDataRange().getValues().slice(1);

  const lastRow = responses.getLastRow();
  const row = responses.getRange(lastRow, 1, 1, responses.getLastColumn()).getValues()[0];
  const [ts, company, name, email, jobTitle, headcount, budget, content, timing] = row;
```

```
  // 1) 룰 기반 점수
  const facts = `직무=${jobTitle}|직원=${headcount}|예산=${budget}|시작=${timing}`;
  let ruleScore = 0;
  rules.forEach(([cond, score]) => {
    if (matchesCond(cond, jobTitle, headcount, budget, timing)) ruleScore += Number(score);
  });

  // 2) AI 보강 점수 + 코멘트
  const ai = aiAssess({company, name, jobTitle, headcount, budget, content, timing});
  const score = Math.min(100, Math.round(ruleScore * 0.6 + ai.score * 0.4));
  const grade = score >= 70 ? 'HOT' : score >= 40 ? 'WARM' : 'COLD';

  // 3) 담당자 결정
  const a = assignees.find(r => r[0] === grade);
  const assignee = a ? {name:a[1], email:a[2], channel:a[3], userId:a[4]} : null;

  // 4) 즉시 알림
  if (assignee && grade !== 'COLD') {
    sendSlackCard(assignee, {company, name, email, jobTitle, score, grade, content, ai});
    autoReplyToProspect(email, name, FROM);
  } else {
    autoReplyToProspect(email, name, FROM);
  }

  // 5) 로그 적재
  const log = ss.getSheetByName('log');
  log.appendRow([new Date(), company, score, grade, assignee ? assignee.name : '자동응대', ai.comment]);
}

function matchesCond(cond, job, hc, budget, timing) {
  cond = String(cond).trim();
  // 정확한 옵션 문자열만 매칭한다. 부분문자열 매칭은 절대 사용하지 않는다
  // (예: budget이 '5천~1억'이면 '예산>=1억' 룰이 부분일치로 잘못 발동되어 점수가 이중 가산되는 버그)
  if (cond.startsWith('직무=')) return job === cond.slice(3);
  if (cond.startsWith('직원=')) return hc === cond.slice(3);
  if (cond === '예산>=1억') return budget === '1억+';
  if (cond === '예산 5천~1억') return budget === '5천~1억';
  if (cond === '예산 1천~5천') return budget === '1천~5천';
  if (cond === '직원 200+') return hc === '200+';
  if (cond.startsWith('시작=')) return timing === cond.slice(3);
  return false;
}

function aiAssess(lead) {
  const prompt = `
당신은 B2B 세일즈 코치입니다. 다음 리드의 실제 구매 의향과 적합성을 평가하세요.

리드 정보:
- 회사: ${lead.company}
- 이름: ${lead.name}
- 직무: ${lead.jobTitle} / 직원: ${lead.headcount} / 예산: ${lead.budget} / 시작: ${lead.timing}
- 의뢰 내용: """${(lead.content||'').slice(0, 4000)}"""

규칙:
- score: 0~100 정수 (구매 의향 + 우리 솔루션 적합성)
- comment: 한 문장, 영업 담당자에게 도움될 핵심 한 마디
- urgent: true/false (24시간 내 응답 권고 여부)

JSON: {"score":int,"comment":"...","urgent":bool}
`;
  const url = `https://generativelanguage.googleapis.com/v1beta/models/gemini-2.5-flash:generateContent?key=${GEMINI}`;
  const res = UrlFetchApp.fetch(url, {method:'post', contentType:'application/json',
    payload: JSON.stringify({contents:[{parts:[{text:prompt}]}], generationConfig:{responseMimeType:'application/
json'}})
  });
  return JSON.parse(JSON.parse(res.getContentText()).candidates[0].content.parts[0].text);
```

```
}

function sendSlackCard(assignee, lead) {
  const channel = assignee.userId || assignee.channel;
  const emoji = lead.grade === 'HOT' ? '🔥' : '⚠️';
  const body = {
    channel,
    text: `${emoji} ${lead.grade} 리드 인입 — ${lead.company}`,
    blocks: [
      {type:'header', text:{type:'plain_text', text:`${emoji} ${lead.grade} 리드 — ${lead.score}점`}},
      {type:'section', fields:[
        {type:'mrkdwn', text:`*회사*\n${lead.company}`},
        {type:'mrkdwn', text:`*이름/직무*\n${lead.name} / ${lead.jobTitle}`},
        {type:'mrkdwn', text:`*이메일*\n${lead.email}`},
        {type:'mrkdwn', text:`*점수*\n${lead.score}/100`},
      ]},
      {type:'section', text:{type:'mrkdwn', text:`*AI 코멘트*\n${lead.ai.comment}\n${lead.ai.urgent ? ':alarm_clock: 24시
간 내 응답 권고' : ''}`}},
      {type:'section', text:{type:'mrkdwn', text:`*의뢰 요약*\n${(lead.content||'').slice(0, 600)}`}},
      {type:'context', elements:[{type:'mrkdwn', text:`담당자: *${assignee.name}*`}]}
    ]
  };
  if (SLACK_TOK) {
    UrlFetchApp.fetch('https://slack.com/api/chat.postMessage', {
      method:'post', contentType:'application/json; charset=utf-8',
      headers:{Authorization:`Bearer ${SLACK_TOK}`}, payload: JSON.stringify(body)
    });
  }
}

function autoReplyToProspect(email, name, fromName) {
  const html = `<div style="font-family:'Noto Sans KR',sans-serif;line-height:1.7;max-width:600px;">
    <p>${name||'고객'} 님, 문의 감사드립니다.</p>
    <p>담당자가 영업일 기준 24시간 안에 직접 연락드리겠습니다. 그 사이 아래 자료를 먼저 보시면 도움이 됩니다.</p>
    <ul>
      <li><a href="https://www.nedabah.org/cases.html">사례 모음</a></li>
      <li><a href="https://www.nedabah.org/lectures/">강의/워크숍 안내</a></li>
    </ul>
    <p style="color:#6a604f;">— ${fromName}</p>
  </div>`;
  GmailApp.sendEmail(email, `문의를 잘 받았습니다 — ${fromName}`, '', {htmlBody: html, name: fromName});
}
```

강의 시연 포인트

1. 폼 제출 → 1분 내 점수 계산 + 담당자 Slack 카드 + 자동 응답 메일
2. 시트 `rules` 한 줄을 수정해 점수 즉시 변경되는 모습 (코드 0줄 수정)
3. 같은 의뢰가 직무·예산만 다를 때 등급이 어떻게 갈리는지 비교

트러블슈팅

증상	원인	해결
점수가 0	룰 조건 문자열 불일치	`rules` 시트의 라벨을 폼 보기 그대로 복사
Slack 카드 미수신	채널에 Bot 미초대	`/invite @봇이름`
AI가 너무 후함	프롬프트의 "적합성" 강조 부족	"우리 솔루션과 맞지 않으면 50 미만" 한 줄 추가
자동 응답이 누락	Gmail 송신 실패	`MailApp.sendEmail`로 대체 가능

안전·윤리 메모

- 본 자동화의 `aiAssess`는 회사명·이름·의뢰 내용을 Gemini에 전송합니다. **B2B 영업 적합성 평가**라는 명확한 비즈니스 목적이 있으므로 PII 위반에 해당하지 않습니다 — 개인 응답 분석과는 다른 맥락이라는 점이 중요합니다.
- 비교: HR 펄스서베이(가이드 HR-03)는 **개인 익명 응답**이므로 식별 정보를 AI에 절대 전송하지 않도록 별도 설계되어 있습니다. 강의에서는 이 두 케이스를 함께 보여 "어떤 데이터를 AI에 보낼지의 판단 기준"을 토론합니다.

응용 아이디어

- **CRM 연동**: HubSpot/세일즈포스 API로 리드 자동 생성 (Apps Script가 webhook으로 push)
- **이메일 인박스 분석**: Gmail 라벨로 들어온 새 메일을 같은 로직으로 스코어링
- **다국어 자동 응답**: 폼 응답이 영어이면 영어 템플릿으로 분기
- **No-Response 리텐션**: 14일간 연락 없는 리드에 자동 follow-up 메일

MKT-03 · 마케팅 자동화

리뷰·멘션 주간 다이제스트

네이버 블로그·카페·뉴스 멘션 자동 수집. 부정 멘션은 24시간 안에 즉시 알림.

난이도	★★
예상 소요	**45분**
전제 조건	**네이버 검색 API 키 + Gemini API 키**
월 비용	**0원**

⚡ **셋업 전에 결과부터:** 같은 로직을 브라우저에서 입력 한 번으로 실행하는 미니 도구가 있습니다 → `nedabah.org/auto/tools/mention-classifier/`

한 줄 핵심: 네이버 블로그·카페·뉴스의 자사/제품 멘션을 매일 수집하고, 매주 월요일 AI가 감성·핵심 메시지를 분류한 1페이지 리포트를 발송한다.

왜 이 자동화인가

항목	수동(Before)	자동(After)
모니터링 시간	주 4~6시간	0분
부정 멘션 인지 지연	평균 1주	24시간 이내
VOC → 제품 반영 속도	분기 1회 정리	주 1회
위기관리 골든타임	놓침	부정 멘션 즉시 알림 분기

적용 시나리오: 브랜드 매니저·PR·CS·제품팀 VOC.

구성요소

- 네이버 검색 API (블로그·카페 무료 일 25,000건) — `client_id`, `client_secret` 필요
- Google News RSS (보강)
- Google Sheet (수집 + 분석)
- Gemini API
- Apps Script (일별 수집 트리거 + 주간 리포트 트리거)

셋업 가이드

Step 1. 네이버 개발자 센터 키 발급

1. https://developers.naver.com/main/ → 애플리케이션 등록
2. "검색" API 사용 신청 → `Client ID`, `Client Secret` 복사

Step 2. 시트 구조

탭 **`keywords`** : 키워드 | 종류 (자사/경쟁사/카테고리) **탭** **`mentions`** : 날짜 | 키워드 | 출처 | 제목 | URL | 본문스니펫 | 감성 | 주제 **탭** **`weekly_report`** : 누적 리포트 적재

Step 3. 스크립트 속성

- `SHEET_ID`
- `NAVER_ID`, `NAVER_SECRET`
- `GEMINI_API_KEY`
- `RECIPIENT_EMAIL`
- `URGENT_SLACK_HOOK` (부정 멘션 즉시 알림)

Step 4. 트리거

- `dailyCollect` — 매일 07:00
- `weeklyReport` — 매주 월요일 08:00

완성 코드 (`script.gs`)

```
const PROPS = PropertiesService.getScriptProperties();
const SHEET_ID = PROPS.getProperty('SHEET_ID');
const N_ID = PROPS.getProperty('NAVER_ID');
const N_SEC = PROPS.getProperty('NAVER_SECRET');
const GEMINI = PROPS.getProperty('GEMINI_API_KEY');
const EMAIL_TO = PROPS.getProperty('RECIPIENT_EMAIL');
const URGENT = PROPS.getProperty('URGENT_SLACK_HOOK') || '';

// ─────────────────────────────────────────────────────────────
// 1) 매일 수집
// ─────────────────────────────────────────────────────────────
function dailyCollect() {
  const ss = SpreadsheetApp.openById(SHEET_ID);
  const kws = ss.getSheetByName('keywords').getDataRange().getValues().slice(1);
  const dst = ss.getSheetByName('mentions');
  const today = new Date().toISOString().slice(0,10);

  // 중복 방지
  const seen = new Set(dst.getRange('E2:E').getValues().flat().map(String));

  const newItems = [];
  kws.forEach(([keyword, kind]) => {
    if (!keyword) return;
    ['blog','cafearticle','news'].forEach(api => {
      const items = naverSearch(api, keyword);
      items.forEach(it => {
        if (seen.has(it.link)) return;
        seen.add(it.link);
```

```
        newItems.push({date: today, keyword, kind, source: api, title: stripHtml(it.title), url: it.link, snippet:
stripHtml(it.description||'')});
      });
      Utilities.sleep(300);
    });
  });

  if (newItems.length === 0) return;

  // AI 감성·주제 분류 (배치)
  for (let i = 0; i < newItems.length; i += 30) {
    const slice = newItems.slice(i, i+30);
    const cls = classify(slice);
    slice.forEach((it, idx) => {
      const c = cls[idx] || {};
      dst.appendRow([it.date, it.keyword, it.source, it.title, it.url, it.snippet, c.sentiment || '중립', (c.topics||
[]).join(',')]);
      // 부정 즉시 알림
      if (c.sentiment === 'negative' && URGENT) {
        UrlFetchApp.fetch(URGENT, {
          method:'post', contentType:'application/json',
          payload: JSON.stringify({text: `:warning: 부정 멘션 — *${it.keyword}*\n• <${it.url}|${it.title}>\n• 주제: $
{(c.topics||[]).join(', ')}`})
        });
      }
    });
    Utilities.sleep(2000);
  }
}

function naverSearch(api, query) {
  const url = `https://openapi.naver.com/v1/search/${api}.json?query=${encodeURIComponent(query)}&display=20&sort=date`;
  try {
    const res = UrlFetchApp.fetch(url, {
      headers: {'X-Naver-Client-Id': N_ID, 'X-Naver-Client-Secret': N_SEC},
      muteHttpExceptions: true
    });
    if (res.getResponseCode() !== 200) return [];
    return JSON.parse(res.getContentText()).items || [];
  } catch(e) { return []; }
}

function stripHtml(s) { return String(s||'').replace(/<[^>]+>/g, '').replace(/&[a-z]+;/gi, ' '); }

function classify(items) {
  const prompt = `
각 멘션에 대해 한국어로 분류하세요. JSON 배열만 반환.
- sentiment: positive/negative/neutral
- topics: 1~3개 짧은 명사구 (가격/품질/배송/디자인/UX/CS/마케팅/이벤트/리더십/위기/기타)

멘션:
${items.map((it,i)=>`[${i+1}] ${it.title} — ${it.snippet.slice(0,300)}`).join('\n')}

[{"sentiment":"...","topics":["..."]}]
`;
  const url = `https://generativelanguage.googleapis.com/v1beta/models/gemini-2.5-flash:generateContent?key=${GEMINI}`;
  const res = UrlFetchApp.fetch(url, {method:'post', contentType:'application/json',
    payload: JSON.stringify({contents:[{parts:[{text:prompt}]}], generationConfig:{responseMimeType:'application/
json'}})
  });
  return JSON.parse(JSON.parse(res.getContentText()).candidates[0].content.parts[0].text);
}

// ─────────────────────────────────────────────────────────────
// 2) 주간 리포트
// ─────────────────────────────────────────────────────────────
function weeklyReport() {
```

```
  const ss = SpreadsheetApp.openById(SHEET_ID);
  const all = ss.getSheetByName('mentions').getDataRange().getValues();
  const head = all[0];
  const since = new Date(Date.now() - 7*86400000);
  const week = all.slice(1).filter(r => new Date(r[0]) >= since);
  if (week.length === 0) return;

  const grouped = {};
  let pos=0, neg=0, neu=0;
  const topicCount = {};
  week.forEach(r => {
    const [date, keyword, source, title, url, snippet, sentiment, topics] = r;
    grouped[keyword] = grouped[keyword] || {pos:0, neg:0, neu:0, items:[]};
    grouped[keyword].items.push({title, url, sentiment, topics, snippet});
    if (sentiment === 'positive') {pos++; grouped[keyword].pos++;}
    else if (sentiment === 'negative') {neg++; grouped[keyword].neg++;}
    else {neu++; grouped[keyword].neu++;}
    String(topics||'').split(',').filter(Boolean).forEach(t => topicCount[t.trim()] = (topicCount[t.trim()]||0)+1);
  });

  const topTopics = Object.entries(topicCount).sort((a,b)=>b[1]-a[1]).slice(0, 8);

  const insight = aiInsight({total: week.length, pos, neg, neu, topTopics, grouped});
  const html = render(week.length, {pos,neg,neu}, topTopics, grouped, insight);

  if (EMAIL_TO) GmailApp.sendEmail(EMAIL_TO, `[주간] 브랜드 멘션 다이제스트 — ${new Date().toISOString().slice(0,10)}`,
'', {htmlBody: html});
  ss.getSheetByName('weekly_report').appendRow([new Date(), week.length, pos, neg, neu, JSON.stringify(insight)]);
}

function aiInsight(d) {
  const ctx = d.topTopics.map(([t,c])=>`${t}(${c})`).join(', ');
  const prompt = `
브랜드 매니저용 주간 코멘트 작성. 한국어 250자 이내.
데이터:
- 총 멘션: ${d.total}
- 감성: 긍정 ${d.pos} / 부정 ${d.neg} / 중립 ${d.neu}
- 상위 주제: ${ctx}

JSON: {"summary":"...","watch":"...","action":"..."}
- summary: 한 문장 종합
- watch: 다음 주 주의해서 지켜볼 한 가지
- action: 마케팅·CS·제품팀 중 한 곳에 보낼 권고 한 줄
`;
  const url = `https://generativelanguage.googleapis.com/v1beta/models/gemini-2.5-flash:generateContent?key=${GEMINI}`;
  const r = UrlFetchApp.fetch(url, {method:'post', contentType:'application/json',
    payload: JSON.stringify({contents:[{parts:[{text:prompt}]}], generationConfig:{responseMimeType:'application/
json'}})
  });
  return JSON.parse(JSON.parse(r.getContentText()).candidates[0].content.parts[0].text);
}

function render(total, sent, topTopics, grouped, ins) {
  const sumTotal = sent.pos + sent.neg + sent.neu || 1;
  const topicHtml = topTopics.map(([t,c])=>`<li><b>${t}</b> · ${c}건</li>`).join('');
  let kwHtml = '';
  for (const k in grouped) {
    const g = grouped[k];
    const negs = g.items.filter(it=>it.sentiment==='negative').slice(0,3);
    const negHtml = negs.length ? `<ul>${negs.map(n=>`<li><a href="${n.url}">${n.title}</a></li>`).join('')}</ul>` : '<p
style="color:#888;">부정 멘션 없음</p>';
    kwHtml += `<h4>${k}</h4>
      <p>긍정 ${g.pos} / 부정 ${g.neg} / 중립 ${g.neu}</p>
      <p style="color:#b45309;"><b>주의 멘션</b></p>${negHtml}`;
  }
  return `<div style="font-family:'Noto Sans KR',sans-serif;line-height:1.7;max-width:740px;color:#222;">
    <h2 style="border-bottom:2px solid #b45309;padding-bottom:.5rem;">브랜드 멘션 주간 다이제스트</h2>
```

```
    <p style="color:#6a604f;">${new Date().toLocaleDateString('ko-KR')} · 총 ${total}건</p>
    <h3>전체 감성</h3>
    <p>긍정 ${(sent.pos/sumTotal*100).toFixed(0)}% / 부정 ${(sent.neg/sumTotal*100).toFixed(0)}% / 중립 ${(sent.neu/
sumTotal*100).toFixed(0)}%</p>
    <h3>상위 주제</h3>
    <ul>${topicHtml}</ul>
    <h3>📌 핵심 인사이트</h3>
    <p><b>요약 — </b>${ins.summary}</p>
    <p><b>다음 주 관찰 — </b>${ins.watch}</p>
    <p><b>권고 액션 — </b>${ins.action}</p>
    <hr>
    <h3>키워드별 상세</h3>
    ${kwHtml}
    <p style="color:#999;font-size:.85em;margin-top:2rem;">자동 생성 · 출처: 네이버 블로그/카페/뉴스</p>
  </div>`;
}
```

강의 시연 포인트

1. `keywords` 시트에 자사 키워드 한 줄 추가 → `dailyCollect()` 즉시 실행 → 시트 채워지는 모습
2. `weeklyReport()` 즉시 실행 → 메일 도착
3. 일부러 부정 키워드("환불"·"후기 안 좋다") 추가 시 부정 즉시 Slack 알림 시연 → 위기관리 흐름 토론

트러블슈팅

증상	원인	해결
네이버 API 401	Key 누락/오타	콘솔에서 새 Application 등록 확인
일 25,000 한도 초과	키워드 너무 많음	키워드 5개 이내, 검색 sort=date 유지
AI가 모두 "중립"	본문 스니펫이 너무 짧음	가능하면 검색결과 `description` 외 본문 일부 추가 (별도 fetch)
부정 알림 폭주	키워드가 흔한 단어	키워드를 정확한 브랜드/제품명으로 좁힘

응용 아이디어

- **자동 응답 초안**: 네이버 블로그 부정 후기에 대해 CS 톤의 답변 초안 자동 작성 → 사람이 검토 후 발송
- **경쟁사 비교 트렌드**: 자사·경쟁사 멘션 추이를 차트로 시각화
- **제품팀 VOC 자동 라우팅**: 주제가 "품질/UX"이면 제품팀, "가격"이면 영업팀으로 자동 분기
- **인플루언서 스코어링**: 자주 언급하는 블로거를 자동 식별해 협업 후보 리스트 작성