AI 문서 자동화 실무 과정

타이핑하지 않는 보고서

VS Code와 클로드 코드로, 자료 판독에서 한글(HWPX) 보고서 자동 작성까지 —
보고서를 쓰는 실무자를 위한 14장 실습 교재

대상	선수 지식	준비물	실습 사례
보고서·분석 실무자 연구원 · 기획 · 정책 담당	프로그래밍 지식 불필요 한글·엑셀 사용 경험이면 충분	유료 AI 구독 평소 쓰는 보고서 양식 1건	세율 인상 시나리오 추계 → 한글 보고서 자동 작성
도착점			
보고서 본문의 90% 이상을 손으로 타이핑하지 않기			

목차

PART 1

무엇을, 왜 하는가

도구를 켜기 전에 도착점을 먼저 정한다. 그리고 AI에게 무엇을 먹일 수 있고 무엇을 먹일 수 없는지를 판별하는 눈을 갖춘다.

제 1 장

오늘의 도착점

이 교재의 목표는 “AI에게 보고서를 시키는 법”이 아니다. **연구자가 설계·판단하고, 조판과 타이핑은 기계가 하는 작업 구조**를 손에 붙이는 것이다.

1.1 왜 챗봇이 아니라 편집기인가

보고서를 쓰는 실무자들이 AI를 쓰는 방식은 대체로 세 갈래다.

- Perplexity — 사실관계 조사와 사례 검색 위주
  - “반대 의견·검색 위주로 자료 수집”, “사례 조사”
- 챗봇(ChatGPT·Claude 웹) — 대화창에 붙여넣고 묻는 방식
- Codex 시도 — 채팅칸에서 시도했으나 엑셀 연결 단계에서 플러그인 요구로 막힘

진단은 분명하다. 간단한 질의응답까지는 챗봇으로 충분하지만, **재정분석처럼 원자료·중간산출물·최종 보고서가 함께 굴러가는 작업은 챗봇에서 통제가 안 된다.**

> 챗봇에 넣으면 그 친구는 우리가 못 보는 가상 환경에서 데이터 셋을 처리합니다. 우리가 컨트롤할 수 없습니다. 그래서 가능하면 VS Code를 쓰시라고 말씀드리겠습니다.

웹 챗봇의 파일 처리는 눈에 보이지 않는 임시 환경에서 일어난다. 어떤 파일이 어떻게 읽혔는지, 중간 결과가 무엇이었는지 확인할 수 없다. 반면 편집기(VS Code) 위에서 작업하면 **폴더·파일·중간 산출물이 전부 내 디스크에 남는다.** 검증과 재현이 가능해진다.

1.2 최종 목표 — 한글 보고서 자동화

이 작업 방식의 도착점은 **HWPX 파일**이다. 분석 결과를 기관 고유의 보고서 서식에 맞춰 한글 파일로 떨어뜨리는 것까지가 한 사이클이다.

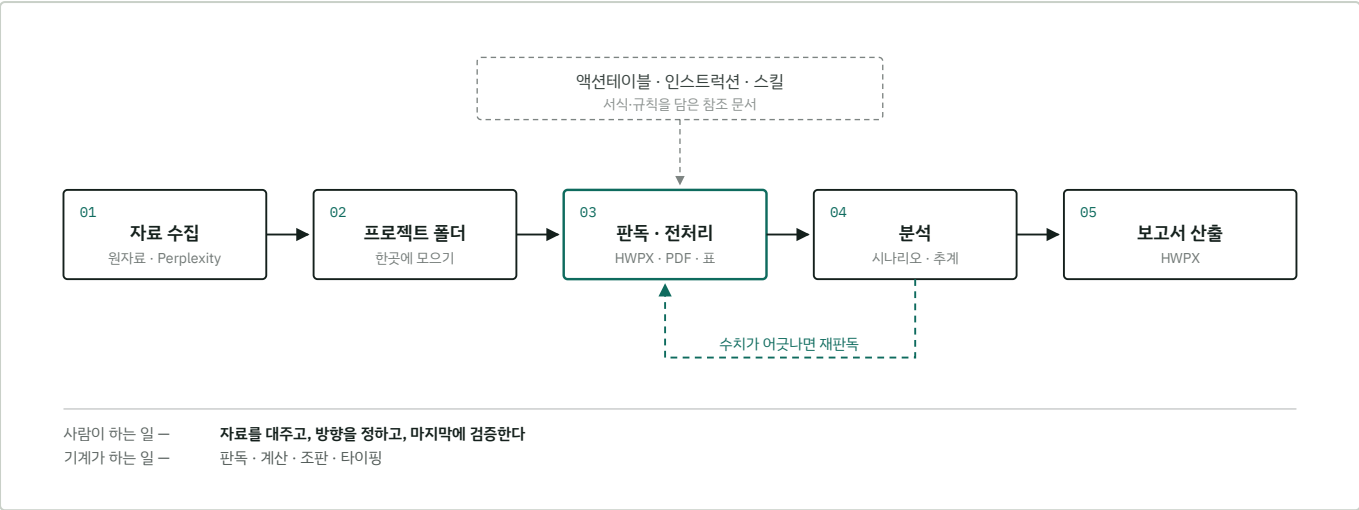


그림 **1-1** 이 교재 전체가 이 다섯 단계를 한 바퀴 도는 구성이다. 03과 04 사이의 점선 루프 — 결과가 원자료와 어긋나면 판독 단계로 되돌아가는 것 — 이 실무에서 가장 자주 돌게 되는 구간이다.

> 저희도 보고서를 많이 쓰거든요. 수시로 쓰기 때문에, **보고서에 글자를 최대한 내 손으로 타이핑하지 않는다**가 목표입니다.

이 방식이 정착한 조직에서는 "양식의 90%, 실제로는 99%를 손대지 않고 만든다"는 수준까지 간다. 처음부터 그 수준을 노릴 필요는 없다. 1장의 목표는 **도착점이 어디인지 아는 것**이다.

1.3 액션 테이블 — 한글 자동화의 열쇠

AI가 한글 문서를 직접 조판하려면 한글이 제공하는 조작 명령(메소드)을 알아야 한다. 그 명세가 **액션 테이블(Action Table)**이다. 한글과컴퓨터가 배포하는 문서로, HWPX에 대응하는 메소드와 파라미터가 정리되어 있다.

- 실습 폴더에 `ActionTable_2504.pdf` 형태로 함께 넣어 둔다
  - 내용을 사람이 읽을 필요는 **전혀 없다**. AI가 참조한다
  - AI가 한글 처리에서 막히면 "액션 테이블을 봐라"라고 지시하면 스스로 해결한다
  - 표 삽입은 물론 **그림 삽입**까지 이 명세로 처리된다

> 준비물 — 액션 테이블
>
> 액션 테이블 PDF를 미리 내려받아 프로젝트 폴더에 두는 것. 이것 하나가 "AI가 한글을 못 읽겠다"는 상황의 대부분을 해결한다.

※ 액션 테이블은 한글과컴퓨터가 공개 배포하는 문서다. 파일명의 숫자는 판 번호이므로, 쓰는 한글 버전에 맞는 최신판을 내려받아 두면 된다.

1.4 실습 소재

이 교재는 실제 업무 문서 두 건을 소재로 실습을 진행한다.

- **세율 인상 관련 추계 보고서** — 분석·시나리오 추계 실습용
  - 부가가치세 대비 지방소비세율 25.3% → 개정안 35.3% → (실습) 40% 시나리오
- **비용추계서 샘플** — 보고서 서식 분석 실습용
  - 「간병보험료에 대한 세액공제 신설」 추계서(샘플)

※ 실습을 준비할 때는 **평소 쓰는 보고서 양식 한 건**과 **분석 대상 문서 한 건**을 미리 확보해 두면 된다.
그 두 건이 있으면 이 교재의 모든 실습을 자기 업무로 바꿔 따라할 수 있다.

제 2 장

AI가 읽는 파일, 못 읽는 파일

이 장을 건너뛰면 나머지 열세 장이 전부 헛돈다. **AI 분석의 정확도는 모델이 아니라 입력 파일의 형식이 8할을 결정한다.**

2.1 HWP와 HWPX는 다른 물건이다

확장자 뒤에 `X` 하나가 붙고 안 붙고의 차이가 결정적이다.

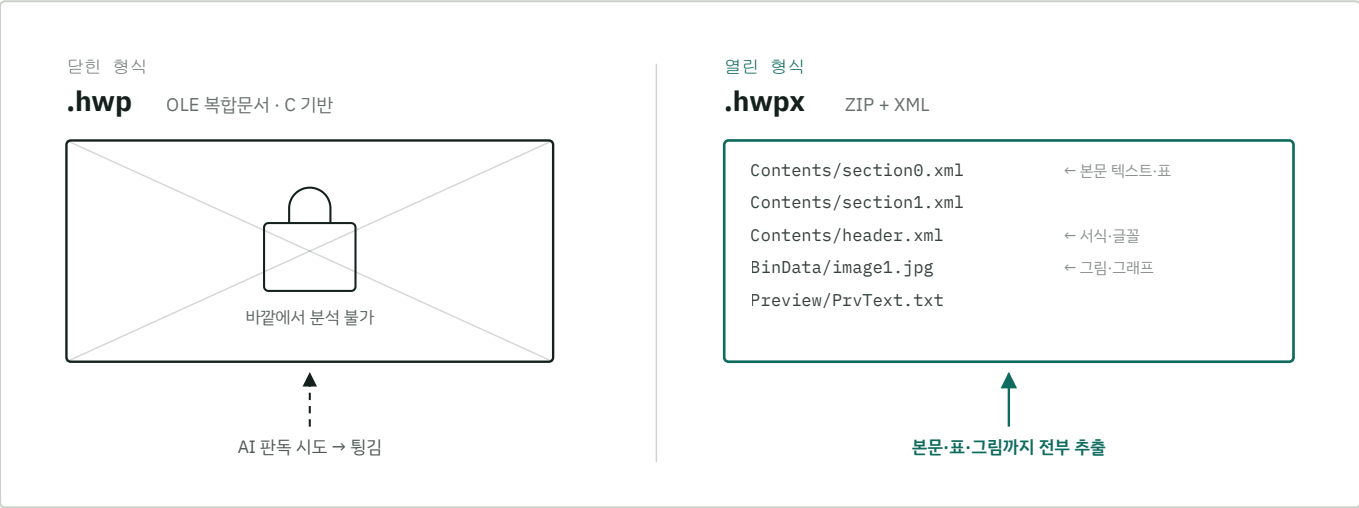


그림 2-1 HWPX 파일의 확장자를 `.zip`으로 바꿔 압축을 풀면 실제로 나오는 구조. 왼쪽의 옛 HWP는 OLE 객체 지향 방식이라 외부 프로그램이 안을 들여다볼 수 없다.

> 용어 — XML
>
> 웹페이지를 만드는 언어가 HTML(HyperText **Markup** Language)이다. 여기서 갈라져 나와 "확장 가능한(eXtensible)" 표시 언어가 된 것이 XML이다. 태그로 구조를 표시해 두기 때문에 **바깥 프로그램이 안의 구조를 그대로 읽을 수 있다.** HWPX·DOCX·XLSX처럼 끝에 X가 붙은 형식이 모두 이 계열이다.

확인 실습 — HWPX의 정체 확인하기

1. HWPX 파일을 복사해 사본을 하나 만든다 (원본 보호).
2. 사본의 확장자를 `.hwpx` → `.zip`으로 바꾼다.
3. 압축을 푼다. `Contents/`, `BinData/`, `META-INF/` 폴더가 나온다.

4 안이 전부 XML 텍스트이고, 문서에 들어간 그림은 `BinData/`에 개별 이미지 파일로 분해되어 있는 것을 확인한다.

이 구조 때문에 **보고서 안의 그래프 이미지까지 AI가 꺼내서 읽을 수 있다.** 옛 HWP에서는 불가능한 일이다.

2.2 변환 전략

- 옛 `.hwp` 파일이 있으면 한글에서 **다른 이름으로 저장**
  - 1순위: **HWPX**로 저장
  - 2순위: **PDF**로 저장
  - 둘 중 하나로는 바꿔야 그나마 읽힌다. 그래도 100%는 아니다
- 최신 Gemini 등은 옛 HWP도 "읽기는 읽는다"
  - 글자만 있는 문서는 읽어낼 수도 있다
  - 그러나 **금액·숫자가 많은 재정 문서는 HWPX를 써야 한다**
- 바꿀 파일이 수십·수백 건이면 손으로 하지 않는다
  - 일괄 변환 프로그램(파서)을 만들면 된다. "생각보다 금방 짤 수 있다"

2.3 형식별 판독 적합도

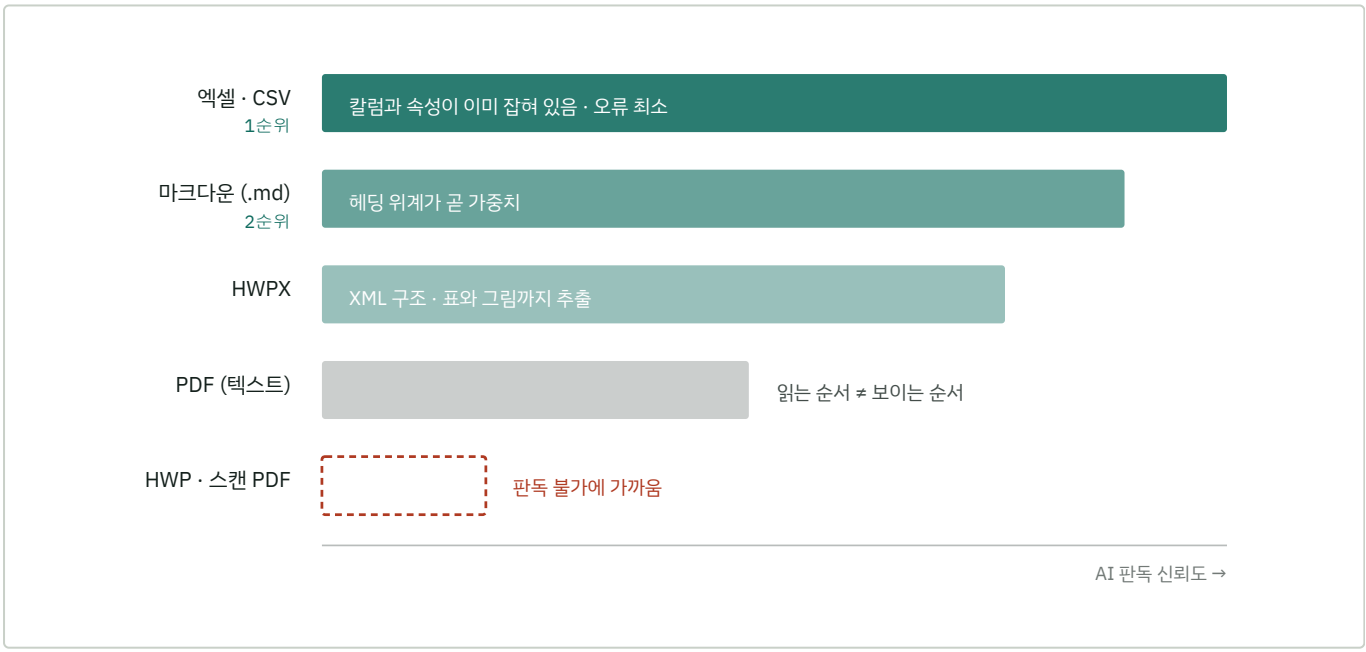


그림 2-2 입력 형식을 한 단계 올리는 것이 모델을 한 등급 올리는 것보다 정확도에 크게 기여한다. 원자료를 엑셀로 확보할 수 있으면 다른 어떤 조치보다 먼저 그것을 한다.

표 2-1 · 형식별 특성과 취급 요령

형식	특성	실무 요령
엑셀 / CSV	표 구조·칼럼 속성이 명시됨. 판독 오류가 가장 적다	가능하면 원자료를 이 형태로 확보한다. 엑셀을 CSV로 한 번 더 내리면 가장 정확하다
마크다운 (.md)	AI가 가장 선호하는 텍스트 형식. 헤딩 위계가 그대로 중요도 신호가 된다	조사 자료는 전부 .md로 모은다 (3장)
HWPX	ZIP+XML. 본문·표·그림·서식까지 분해 가능	기관 문서의 기본 저장 형식으로 삼는다
PDF	내부는 마크업이지만 **배치 순서가 시각 순서와 다르다**	보조·교차확인용. 단독 의존은 피한다
스캔 PDF	글자가 이미지. OCR을 거쳐야 하고 인식률이 떨어진다	OCR 수행 후에도 결과를 반드시 눈으로 확인
HWP (구형)	OLE 복합문서. 외부 분석이 막혀 있다	쓰지 않는다. HWPX 또는 PDF로 변환 후 사용

2.4 표에서 사고가 난다

재정 문서에서 가장 자주 깨지는 것은 문장이 아니라 **표**다.

주의 — 판독 사고가 나는 지점

- **병합된 셀** — "병합된 것은 오류가 생길 수 있다고 생각하셔야 됩니다"
- **페이지를 넘어가며 걸친 표** — 헤더가 날아가면 뒷장의 칼럼이 무엇인지 알 수 없게 된다
- **천 단위 콤마** — 인식에 실패하면 자릿수가 밀려 **금액이 달라진다**. 가장 위험한 오류
- **보기 좋게 편집된 표** — 사람 눈에 예쁜 조판일수록 기계 판독에는 불리하다

대응은 **다중 판독**이다. 같은 문서를 HWPX로 한 번, PDF로 한 번 읽히고, 그래도 미심쩍으면 이미지로 변환해 비전 모델로 한 번 더 읽힌 뒤 **차이나는 지점만 보고하라**고 시킨다. 자세한 절차는 8장에서 다룬다.

※ **원자료(엑셀)를 보유한 조직**은 이 문제에서 훨씬 유리하다. 최종 편집본만 받아 작업하는 쪽이 오히려 검증 부담이 크다. 원자료를 항상 함께 넣어 두는 것이 가장 확실한 방어다.

제 3 장

마크다운과 자료 수집

자료를 모을 때의 원칙은 하나다. **무엇을 모으든 마크다운으로 모은다.**

3.1 마크다운이란

확장자 `.md`. HTML이 복잡한 표시 언어라면 마크다운은 그 아주 단순한 사촌이다. 제목 앞에 `#`을 붙이는 것이 전부라고 해도 과장이 아니다.

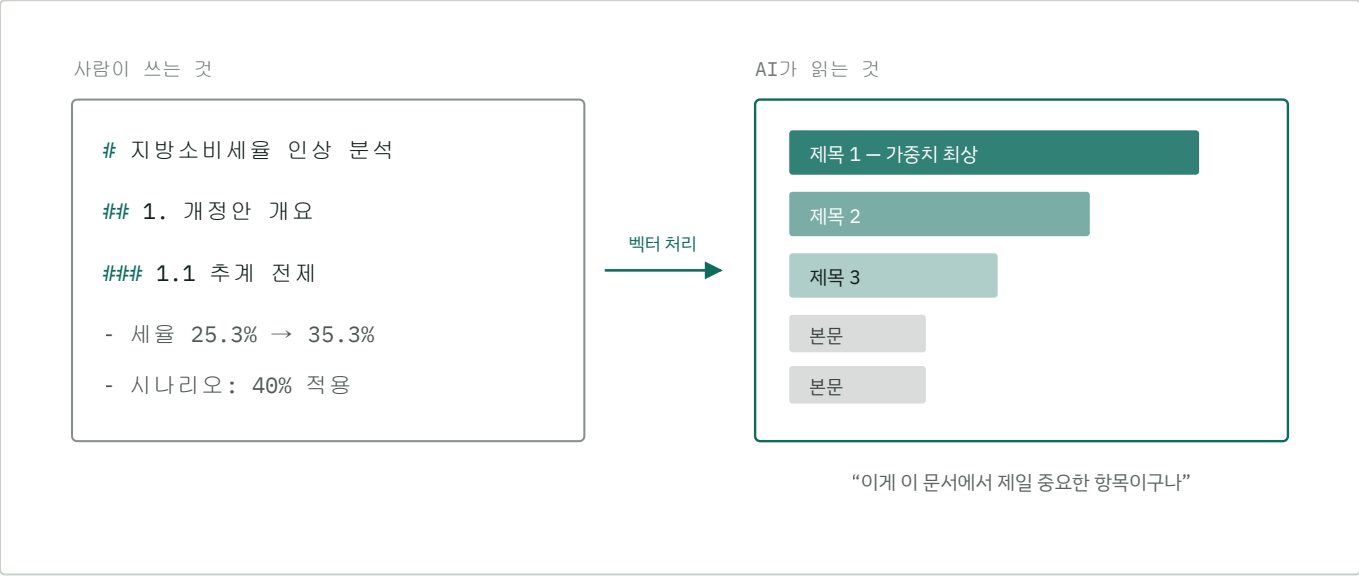


그림 3-1 `#`의 개수가 제목 단계(최대 6단계)를 만든다. 사람에게는 글자 크기 차이로 보이지만, AI는 그 위계를 문서 내 중요도로 인식한다. 자료를 마크다운으로 모으는 이유가 여기에 있다.

※ 굵게·기울임·리스트 같은 나머지 문법은 실무상 크게 중요하지 않다. **제목 위계와 링크**만 살아 있으면 충분하다.

챗봇에서 답변을 받아 내려받으면 기본이 마크다운인 것도 같은 이유다. AI가 만들고 AI가 다시 읽기 좋은 형식이기 때문이다.

3.2 Perplexity로 근거 모으기

자료 조사의 **1번**으로 삼을 도구다.

- 무료로도 검색은 충분하다
  - 유료 전환 시 차이는 **심층 리서치** — 3~4배 더 자세히 검색
  - 오른쪽의 에이전트형 리서치는 보고서 형태로 나오지만 비용이 크다

- 결과는 반드시 **마크다운으로 내려받는다**
  - PDF도 가능하나, 프로젝트 폴더에 쌓을 형식은 .md가 우선

예시 — 법령 충돌 사례 조사

```
› 소방기본법과 건축법이 정면으로 충돌한 사례를 찾아줘. 10가지 이상.
```

결과 12건 → 다운로드 → .md 파일로 저장 → 프로젝트 폴더에 투입

3.3 법령·시점 검증 — 실제 사고 사례

주의 — 폐지된 법령을 인용한 사고

> 법령을 하나 검색했는데 법령이 나오는 거예요. 그걸 가지고 보고서에 각주 달아서 썼거든요. 썼는데 알고 보니까 **3개월 전에 폐기가 됐습니다.** 없는 법령을 있다고 하고 올린 겁니다.

대응 요령:

- 법령 조사에는 **시점(시행일·개정일·폐지 여부)을 반드시 함께 확인**한다
- 검색 조건에 **"블로그는 제외"**, **"현재 유효한 법령만"**을 넣는다
- 조건을 좁히면 건수는 줄지만(12건 → "10건 이상은 어렵다") 신뢰도는 올라간다

> 우리는 보고서 100가지 중에 한 가지만 이상해도 신뢰성이 떨어지기 때문에, 연구자 자체가 여기에 좀 집약해서 들어가야 될 필요가 있습니다.

이 원칙은 4부(14장)의 "검증은 집필자의 책임"으로 다시 이어진다.

PART 2

환경 구축

이 과정에서 유일한 진짜 허들이 이 부(部)에 있다. 여기만 넘으면 나머지는 익숙해지는 문제일 뿐이다.

제 4 장

도구와 구독 선택

돈 이야기부터 정직하게 한다. 이 작업 방식은 **유료 구독을 전제로 한다.** 무료 요금제로는 시작조차 되지 않는다.

4.1 챗봇과 에이전트는 다르다

> 용어 — 챗봇 / 에이전트
>
> - **챗봇** — 내가 누르면 대답하고, 또 누르면 또 대답한다. 한 번의 질문에 한 번의 답
> - **에이전트** — 방향이 반대다. 목표만 제시하면 **그 일을 끝낼 때까지 스스로 수행하고**, 필요할 때 **먼저 나에게 묻는다**

웹의 클로드 화면에서 왼쪽의 대화창은 챗봇이다. 오늘 쓸 것은 오른쪽의 **코드(Code)** 쪽이고, 이것이 에이전트로 동작한다.

※ 웹 챗봇에 재정 자료를 넣고 물어보는 것 자체는 가능하고 답도 나온다. 다만 처리 과정이 완벽하게 통제되지 않으므로 개인 학습용으로만 권한다.

4.2 모델 계열

클로드 안에는 성능과 비용이 다른 모델이 여러 층으로 들어 있다. 이름과 세대는 계속 바뀌므로, 층위의 성격만 파악해 두면 된다.

표 4-1 · 모델 선택 기준

모델	성격	재정분석 실무 권고
Fable	가장 비싸고 가장 잘한다	프로그램을 짤 때. 일반 보고서 작업에는 **굳이 쓰지 않아도 된다**
Opus	범용 최상위 · 기본값	**기본으로 둔다.** 보고서 집필·해석 모두 무난
Sonnet	실용적 · 한도에 여유	한도가 빠듯할 때. "재정분석 정도까지는 소넷이 가장 쓸 만하다"
Haiku	가장 싸고 빠르지만 실수 가능성	**쓰지 않는다.** "괜히 맡겼다가 두 번 일하기 싫어서"

4.3 요금제 — 정직한 이야기

- 무료 요금제 — **불가**. 편집기 연동 자체가 되지 않는다
- 월 20달러대 요금제 — **권하지 않는다**. 한도가 금방 바닥난다
- 월 100달러 / 200달러(Max) — 실무 사용 구간
  - 클로드와 코덱스를 함께 상위 요금제로 쓰면 월 35만 원대
  - "맥스 중에 20배 된 것도 저희는 모자랍니다"
- 팀 공용 계정도 써 봤으나 **결국 개인 계정으로 간다**
  - 각자의 인스트럭션·스킬이 계정에 쌓이기 때문

> 처음에 22달러짜리 쓰다가 답답하니까 100달러로 가겠다고 하셨어요. 한 달만 쓰고 다음 달에 다시 돌아오겠다 했는데, **1년째 못 돌아가고 있습니다.** 못 돌아옵니다. 편하기 때문에.

※ 사용량은 **5시간 단위 세션 한도**와 **주간 한도**로 나뉘어 관리된다. 확인 방법은 6.4절에서 다룬다.

제 5 장

VS Code 화면 읽는 법

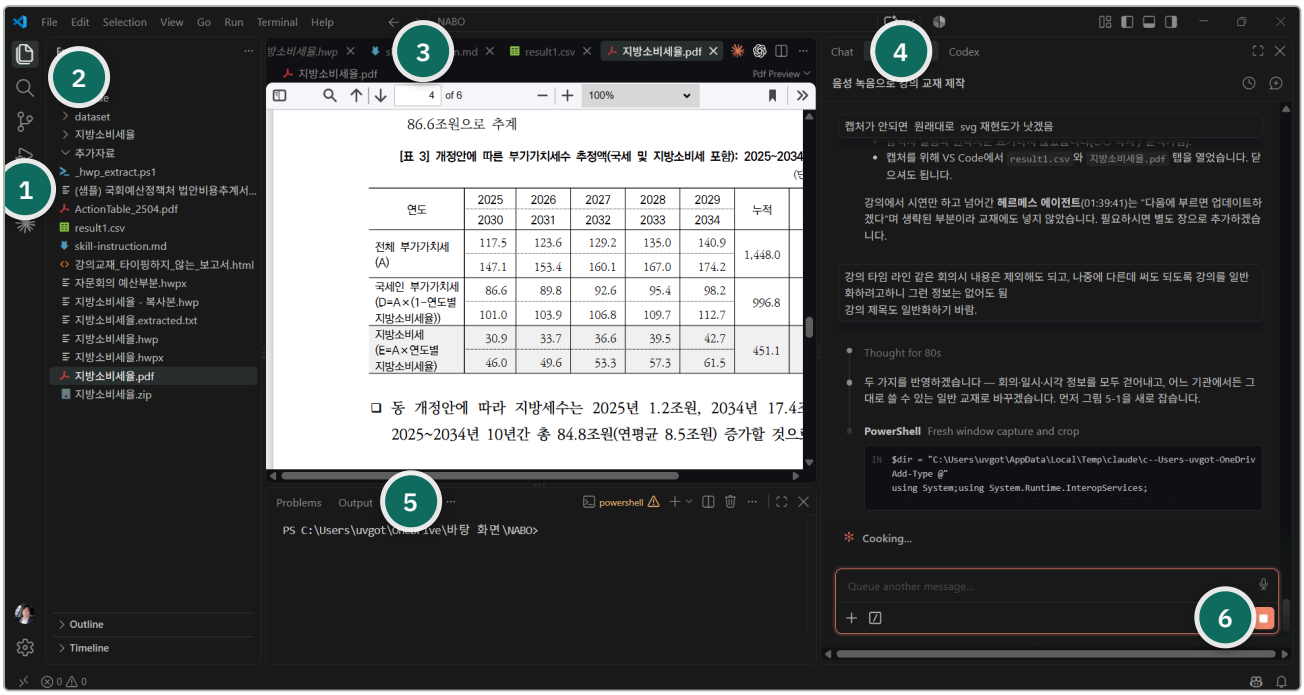
까만 화면이 처음 뜨는 순간이 최대 고비다. 화면을 **네 구역**으로만 나눠서 보면 된다.

5.1 설치

1. “Visual Studio Code”를 검색해 자신의 운영체제(Windows / macOS)용을 내려받는다.
2. 설치 과정은 기본값 그대로 따라간다.
3. 실행하면 **까만 화면**이 나온다. 여기서 겁먹지 않는 것이 전부다.

> 이제 여기서부터 이게 허들입니다. 여기만 남으시면 엄청 좋은 환경이 보일 겁니다.

5.2 화면 네 구역



1. **활동 표시줄** — 여기서 쓰는 것은 **확장(Extensions)** 하나뿐이다
2. **탐색기** — 프로젝트 폴더의 파일 목록. 토글로 열고 닫는다
3. **편집기** — 탐색기에서 파일을 누르면 내용이 여기 뜬다
4. **AI 패널** — Chat / **Claude Code** / Codex 탭. 기본은 안 보이며 오른쪽 사이드바를 켜야 나온다
5. **터미널** — 명령을 직접 치는 곳. 열고 닫을 수 있다
6. **실행 모드** — Auto / Plan 등 자율성 수준 표시 (6.5절)

그림 5-1 실제 작업 중인 화면. 왼쪽 탐색기에 열려 있는 것이 실습용 프로젝트 폴더이고, 가운데 편집기에는 원문 PDF가, 오른쪽에는 지시를 내리는 AI 패널이 떠 있다. 이 배치 하나가 작업의 전부다.

> 기억할 것은 이 두 가지뿐

- **파일 탐색기 토글** — 열었다 닫았다 하는 버튼
- **확장(익스텐션)** — 확장팩을 설치하는 곳

※ 검색·소스제어·실행 등 나머지 아이콘은 전부 신경 쓰지 않아도 된다.

글자가 작아 안 보일 때

`Ctrl` + `+` / `Ctrl` + `-` 로 화면 전체를 확대·축소한다. 여럿이 함께 보는 자리라면 미리 키워 두는 편이 좋다.

5.3 터미널 — 몰라도 된다

아래쪽 터미널은 예전 도스(DOS) 명령창과 같은 자리다. `dir`, `copy`, `xcopy`, `mkdir` 같은 명령이 그대로 동작한다.

```
PS C:\Users\...\NABO> dir
Directory: C:\Users\...\바탕 화면\NABO
ActionTable_2504.pdf   result1.csv   지방소비세율.hwpx   ...
```

> 지금부터 다 걱정되시죠? 명령어, 나도 기억 안 나는데 어떡하지? — **하나도 모르셔도 됩니다.** 우리의 에이전트가 오른쪽에 다 붙어 있습니다.

명령을 외울 필요가 없다. 실제 작업에서는 오른쪽 AI 패널에 한국어로 말하면 AI가 터미널 명령을 대신 만들어 실행한다.

제 6 장

클로드 코드 설치와 조작

넘어야 할 관문은 정확히 세 개다. **유료 계정 · 확장팩 설치 · 로그인 승인.** 이 셋을 통과하면 배울 것이 사실상 없다.

6.1 확장팩 설치

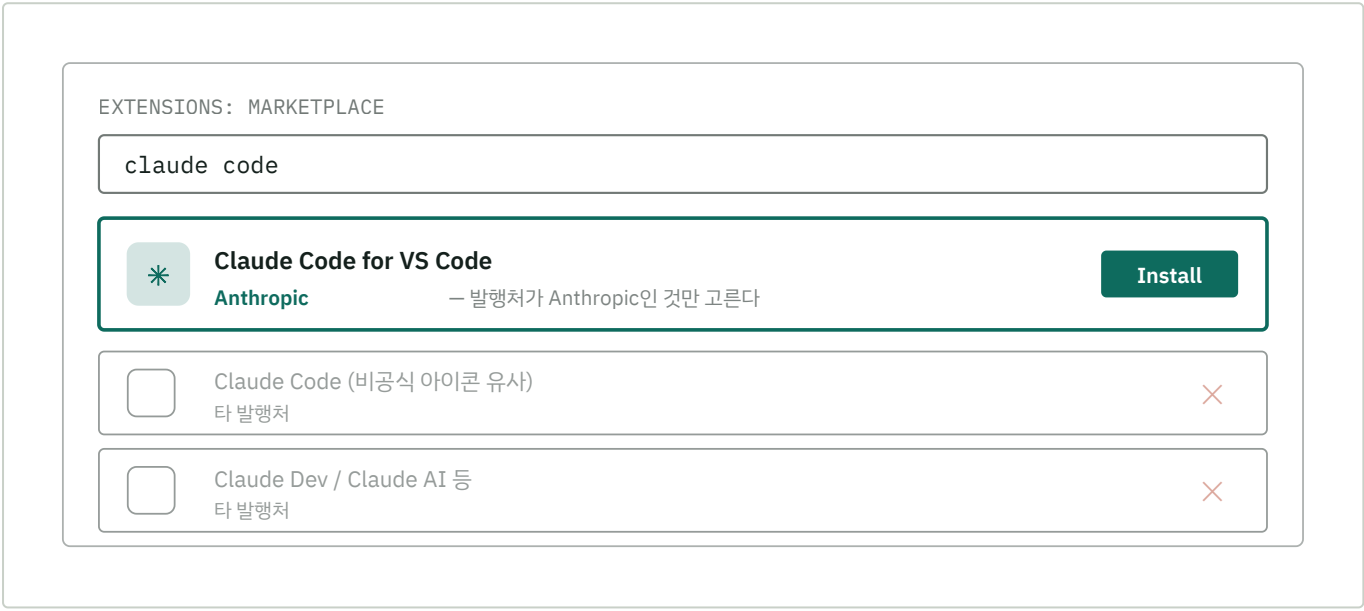


그림 6-1 확장 검색 화면 재현도. 아이콘이 비슷한 유사 확장이 여럿 나오므로, 발행처가 Anthropic인 것만 선택해야 한다. 설치 단계에서 가장 흔히 나오는 실수다.

1. 활동 표시줄에서 **확장(Extensions)** 아이콘을 클릭한다.
2. 검색창에 `claude code`를 영문으로 입력한다.
3. 결과 중 **Claude Code for VS Code (Anthropic)**를 고른다. 아이콘이 비슷한 다른 확장을 고르면 안 된다.
4. **Install**을 누른다. 설치가 끝나면 `Uninstall`로 바뀐다.

※ `Disable`은 누르면 확장이 동작하지 않게 된다. 설치 직후 `Enable` 상태이면 문제 없다.

6.2 로그인과 승인

1. 왼쪽(또는 오른쪽) 패널에 나타난 **클로드 코드**를 연다.
2. 로그인 안내가 뜨면 **Open**을 누른다. 브라우저의 클로드 로그인 화면으로 넘어간다.

3. “Claude Code가 세션에 접근합니다”라는 **권한 승인** 화면에서 승인한다.
4. 편집기로 돌아와 입력창에 커서가 깜빡이면 성공이다.

> 주의 — 여기가 유일한 관문
>
> 클로드 코드는 **웹사이트에서 유료 구독 중인 계정**만 연결된다. 무료 계정은 이 단계에서 막힌다.
>
> 막히면 인터넷에서 “VS Code 클로드 코드 로그인”을 검색해 그대로 따라 하면 된다. 강의 중에도 “요세 가지라서 간단합니다 — 유료, 로그인, 확장팩”이라고 정리했다.

6.3 세 개의 탭

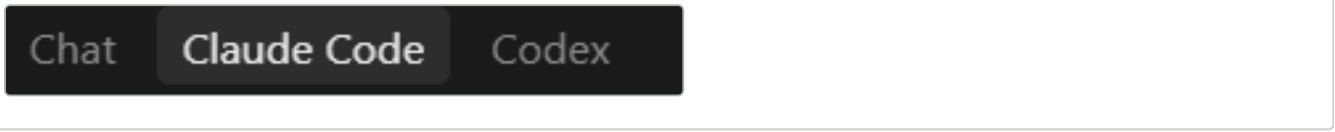


그림 6-2 AI 패널 상단의 탭. 오늘 쓰는 것은 가운데 `Claude Code` 다. 왼쪽 `Chat` 은 GitHub Copilot 계열로 신경 쓰지 않아도 되고, 오른쪽 `Codex` 는 14장의 교차검증에서 다시 등장한다.

6.4 슬래시 명령

입력창에서 `/`를 누르면 명령 목록이 뜬다. 실무에서 쓰는 것은 두 개뿐이다.

- `/usage`
  남은 사용량 확인 — 5시간 단위 세션 한도와 주간 한도가 각각 표시된다. 상위 모델(Fable)에는 별도 한도가 걸려 있다.

- `/model`
  모델 전환 — Fable / Opus / Sonnet / Haiku 중 선택. 함께 표시되는 effort(노력 수준)와 thinking(사고) 옵션도 여기서 조정한다.

6.5 노력·사고·실행 모드

effort — 얼마나 애쓰게 할 것인가

슬라이더를 오른쪽으로 밀수록 더 오래 생각하고 더 잘한다. 대신 토큰을 많이 쓴다.

> 대충 이렇게 하면 대충 하고 빨리 하고요. 오른쪽으로 하면 돈 좀 많이 줘야 됩니다. 인센티브 줘야 되고 토큰 많이 씁니다. — 직원하고 똑같습니다.

주의 — 옵션을 잘못 잡으면 업무가 늘어진다

사고(thinking)를 켜고 노력을 최대로 올려 두면 단순 작업까지 지나치게 오래 끈다. "한 페이지 쓰라 그랬는데 30분 뒤에 나오는 경우도 있다". **글만 쓰면 되는 단순 작업에서는 사고를 끄거나 중간에 둔다.** 반대로 검증을 제대로 시키고 싶을 때만 최대로 올린다.

실행 모드 — 어디까지 물어보게 할 것인가

표 6-1 · 실행 모드 네 가지

모드	동작	쓰는 상황
Manual	조금만 바뀌어도 매번 물어본다	내가 한 단계씩 확인하며 갈 때
Edit automatically	파일 편집은 자율로 진행하되 중요한 지점에서 묻는다	기본 권장. 자율성과 통제의 절충
Plan	바로 실행하지 않고 **계획부터 세운다.** 계획이 명확해질 때까지 다듬는다	작업 범위가 큰 분석을 시작할 때
Auto	한 번 지시하면 **끝까지 간다.** 터미널 명령도 스스로 실행	추가로 물을 것이 없는 명확한 작업

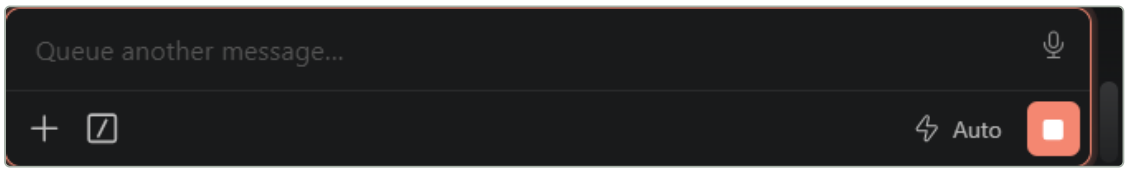


그림 6-3 입력창 하단. 오른쪽의 `Auto` 가 현재 실행 모드다. 오른쪽 끝 마이크 아이콘이 이른바 바이브 코딩(말로 지시하기)의 입구이며, 왼쪽에는 현재 지정된 파일이 표시된다.

※ Auto 모드에서는 파워셸 명령이 승인 없이 바로 실행된다. Auto가 아니면 "이 명령을 수행할까요?"를 매번 묻는다.

PART 3

실습 — 지방소비세율 보고서

여기서부터는 실제 업무 문서로 한 바퀴를 돈다. 판독 → 분석 → 보고서까지, 실무에서 손이 움직이는 순서 그대로다.

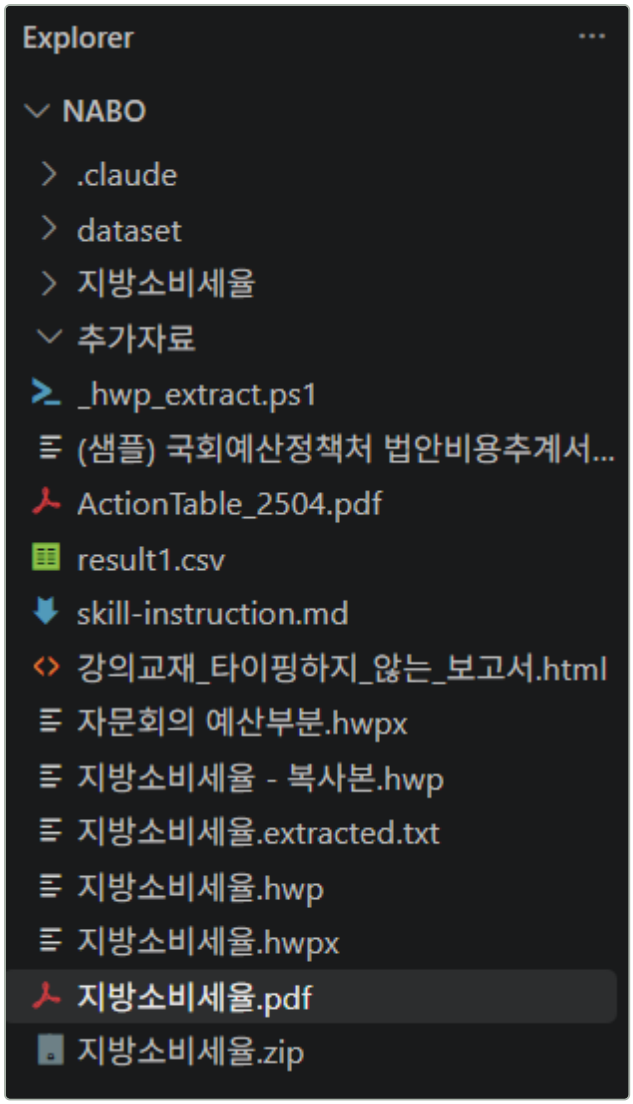
제 7 장

프로젝트 폴더 구성

VS Code는 **폴더 단위로 프로젝트를 운영한다.** 사안 하나에 폴더 하나를 만들고, 관련된 모든 것을 그 안에 던져 넣는 것이 시작이다.

7.1 폴더 열기

1. 바탕화면 등에 사안별 폴더를 하나 만든다 (예: `NABO`).
2. 받은 자료를 **정리하지 않고 그냥 전부** 그 폴더에 넣는다.
3. VS Code에서 **File > Open Folder** → 그 폴더를 선택한다.
4. 왼쪽 탐색기에 파일 목록이 쭉 들어온다. 이제 이 폴더가 프로젝트의 **루트**다.




```
NABO/ ← 루트 폴더 ├ .claude/ ├ dataset/ ← 배경 데이터셋 ├ 추가자료/ ← 조사 자료(.md) 투입구 ├ 지방소비세율/ ← hwpx 압축 해제 결과 ├ ActionTable_2504.pdf ← 한글 자동화 명세 ├ (샘플) 법안비용추계서.hwpx ├ 지방소비세율.hwp ├ 지방소비세율.hwpx ← 실제로 쓰는 파일 ├ 지방소비세율.pdf ← 교차 판독용 ├ result1.csv ← 분석 산출물 └ skill-instruction.md ← 보고서 작성 지침
```

그림 7-1 실습용 프로젝트 폴더의 실제 구성. 같은 문서의 HWP·HWPX·PDF를 일부러 함께 넣어 둔 점, 액션 테이블이 동봉된 점, 산출물(`result1.csv`)과 지침(`skill-instruction.md`)이 같은 폴더에 있는 점이 이 구성의 핵심이다.

7.2 무엇을 넣는가

- **원자료** — 엑셀·CSV가 있으면 반드시 함께 넣는다
  - "원 데이터가 있는 것은 걱정이 하나도 없고, 최종 파일만 받으면 고민이 됩니다"
- **대상 문서** — HWPX를 기본으로, 교차 판독용 PDF도 함께
- **액션 테이블** — `ActionTable_2504.pdf`
- **조사 자료** — Perplexity 등에서 받은 `.md` 파일
- **지침·스킬** — 보고서 작성 규칙(12장)

하위 폴더로 나눠도 좋다. 탐색기에서 마우스 오른쪽 → 새 폴더로 `dataset`, `추가자료` 같은 구획을 만들어 두면 나중에 지시하기 편하다. **하위 폴더까지 전부 인식된다.**

7.3 인덱싱과 컨텍스트 한계

폴더 안의 모든 파일을 한 번에 다 읽는 것은 아니다.

- 전체는 **인덱싱**한다 — "어디에 무슨 파일이 있다"는 지도를 만든다
- 실제로 한 번에 머리에 올릴 수 있는 양(컨텍스트)은 제한적이다
  - 1GB를 넣었다고 1GB를 다 읽고 있는 것이 아니다
- 그래서 **주목할 파일을 직접 지목**해 줘야 한다 → 다음 절의 `@`

7.4 `@` — 파일 지목하기

입력창에서 `@`를 누르면 폴더 안의 파일·폴더 목록이 뜬다. 여기서 대상을 정확히 찍는다.

> `@지방소비세율.hwpx 이 파일을 읽어 봐`
>
> @ 없이 "지방소비세율 자료 읽어 봐"라고 해도 대개는 알아듣는다. 오타가 있어도 대충 안다. 다만 같은 이름의 hwp·hwpx·pdf가 섞여 있을 때는 **@로 정확히 지정**해야 엉뚱한 파일을 읽지 않는다.

※ 폴더에 파일을 새로 넣으면 `@` 목록에 바로 나타난다. 반대로 **비어 있는 폴더는 목록에 뜨지 않는다.**
폴더를 만들어 뒀는데 `@`에 안 보인다면 대개 아직 파일을 넣지 않은 것이다.

7.5 PDF 확장팩

폴더 안의 PDF를 처음 클릭하면 "PDF를 읽으려면 확장을 설치하라"는 안내가 뜬다. **Install**과 **신뢰**를 누르면 된다. 이후 PDF가 편집기 안에서 바로 미리보기된다.

> 이게 원래는 처음에 물어보셨던 클로드 웹사이트에서는 이 조작이 힘듭니다.
>
> 그래서 여기서 쓰는 겁니다. 여러 익스텐션 때문에 그렇고.

제 8 장

실습 1 · 자료 판독과 전처리

분석의 정확도는 여기서 결정된다. **한 번 읽히고 끝내지 않는 것**이 이 장의 전부다.

8.1 첫 지시

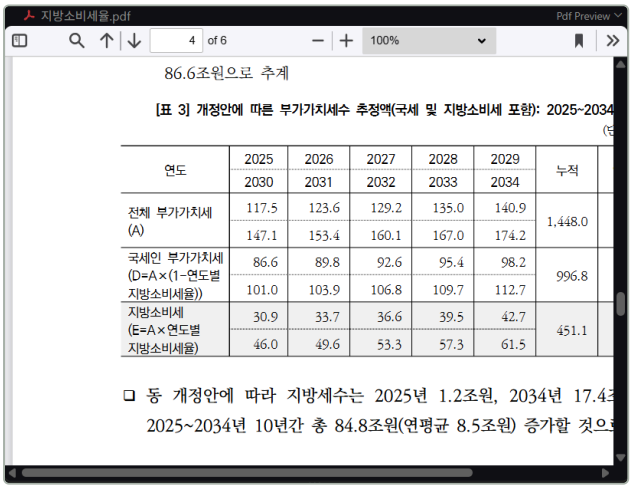
> `@지방소비세율.hwpx 관련한 자료를 읽어 봐. 안에 분석해 봐`
>
> 타이핑 대신 마이크를 켜고 말로 지시해도 된다. 혼자 쓰는 공간이라면 이 방식이 훨씬 빠르다.

AI는 곧바로 “HWPX는 XML 포맷이므로 압축을 풀어서 전체 본문 구조를 살려 추출하겠다”고 답하고 작업을 시작한다. 2장에서 본 내부 구조를 스스로 이용하는 것이다.

판독 결과 확인

읽기가 끝나면 개요가 나온다. 예시 문서에서는 안건, 개정안 내용(세율 25.3% → 35.3%), 핵심 추계 결과, 부대 의견, 추계 전제가 차례로 정리되고, 마지막에 “어떤 부분을 더 파고들까요?”라고 되묻는다.

8.2 교차 판독 — 같은 문서를 두 번, 세 번 읽힌다



같은 문서, 다른 경로

- 1차 — **HWPX**로 읽힌다 (구조가 살아 있음)
- 2차 — **PDF**로 다시 읽힌다 (그림 8-1)
- 3차 — 그래도 이상하면 **이미지로 변환**해 비전 모델로 읽힌다

> `둘 다 읽어. 차이점이 있는 건 보고해`
>
> 클로드는 비전(이미지 판독)도 되므로 그림으로 된 표까지 읽을 수 있다.

그림 8-1 PDF 확장팩을 설치하면 편집기 안에서 원문 표를 바로 확인할 수 있다. 화면의 「표 3」이 판독 대상이자 대조 기준이다 — 전체 부가가치세(A) 누적 1,448.0조원, 국세분(D) 996.8조원, 지방소비세(E) 451.1조원. 9장에서 AI가 재현해 낸 수치를 이 표와 맞춰 보게 된다.

8.3 깨지기 쉬운 곳을 먼저 의심한다

주의 — 실습 문서에서 실제로 발견된 문제

- **병합된 셀** — 예시 문서에도 병합 셀이 있었다. 병합된 표는 오류가 생길 수 있다고 전제하고 봐야 한다
- **페이지를 걸친 표** — 앞 페이지의 헤더가 날아가면 뒷장 칼럼의 정체를 알 수 없게 된다. PDF에서 특히 잦다
- **사람 눈에 예쁜 조판** — 편집기로 다듬은 표일수록 기계 판독에 불리하다

> 한글이 보통 보면 NABO 자료를 제가 받으면 엑셀인 경우 너무 좋아요. 그 엑셀을 나중에 CSV 파일 같은 콤마가 있는 텍스트 파일로 변환시키면 **제일 정확합니다.**

8.4 전처리 체크리스트

1. 원자료(엑셀·CSV)가 있으면 같은 폴더에 넣고, **"원자료와 대조하라"**고 명시한다.
2. HWPX로 1차 판독시킨다.
3. PDF로 2차 판독시키고 **차이나는 지점만** 보고받는다.
4. 금액·비율 등 핵심 수치는 원문을 직접 열어 **눈으로 대조**한다. 특히 천 단위 콤마.
5. 표가 계속 깨지면 해당 표만 이미지로 변환해 비전 판독을 추가한다.

※ 원자료를 보유한 조직은 4번 단계의 부담이 크게 줄어든다. 최종 편집본만 받아 작업하는 경우가 오히려 검증이 어렵다 — 대조할 기준이 없기 때문이다.

제 9 장

실습 2 · 시나리오 분석과 결과 저장

읽힌 자료 위에 **새로운 질문**을 얹는다. 개정안이 35.3%인데, 40%까지 가면 어떻게 되는가.

9.1 목표를 말로 정한다

판독된 문서 위에 새 질문을 얹는다. 예시에서는 중앙·지방 간 배분율을 40%까지 올리는 경우를 물었다.

```
› 지방·중앙 배분율이 40%로 늘면 어떻게 되는지 분석해 봐. 예측해 봐
```

실행 모드는 **Auto**로 둔다. 추가로 물을 것이 없는 명확한 지시이기 때문이다.

지시하지 않은 것까지 나눠 온다

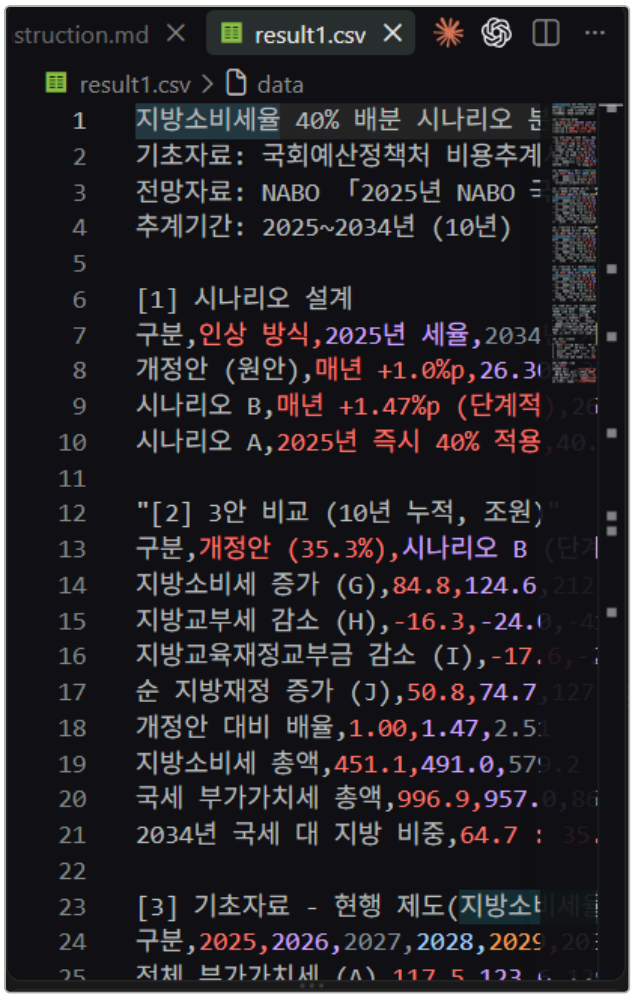
AI는 "도달 방식에 따라 결과가 크게 갈리므로 **시나리오를 두 개로 나눠** 계산하겠다"고 응답했다. 즉시 40%를 적용하는 안과, 단계적으로 40%에 도달하는 안이다.

> 똑똑하네요. 제가 그걸 얘기를 안 했는데.

동시에 **원 추계서와 동일한 기초자료**(NABO 부가가치세수 전망)를 그대로 쓰겠다고 선언했다. 임의의 가정을 새로 만들지 않고 원문의 전제를 승계한 것이다.

※ 작업 중 아래 터미널에서 파워셸이 저절로 돌아가는 것이 보인다. AI가 파이썬 모듈을 불러 계산·변환을 수행하는 과정이다. Auto 모드라 승인을 묻지 않고 진행된다. 우리가 파이썬을 배울 필요는 없다.

9.2 산출물



분석 결과를 파일로 떨어뜨린다

> 결과를 CSV 파일로 만들어 줘
>
> 엑셀(.xlsx)로도 되지만 변환에 시간이 걸린다. **빠르게 갈 때는 CSV**가 낫다. 표를 텍스트로 저장한 형식이라 재판독 정확도도 높다.

※ 파일명 끝의 `x`(`.xlsx`)가 붙는 이유는 2장의 HWPX와 같다 — XML 기반의 편집 가능한 형식이라는 뜻이다.

그림 9-1 실제 산출물 `result1.csv`. 시나리오 설계, 3안 비교, 연도별 추계표, 추계 전제, 분석 포인트, 추계의 한계까지 아홉 개 블록으로 구성되어 저장되었다.

표 9-1 · 산출물에서 발췌 — 3안 비교 (10년 누적, 조원)

구분	개정안 (35.3%)	시나리오 B 단계적 40%	시나리오 A 즉시 40%
지방소비세 증가 (G)	84.8	124.6	212.9
지방교부세 감소 (H)	−16.3	−24.0	−41.0
지방교육재정교부금 감소 (I)	−17.6	−25.9	−44.3
순 지방재정 증가 (J)	50.8	74.7	127.6
개정안 대비 배율	1.00	1.47	2.51
2034년 국세 : 지방 비중	64.7 : 35.3	60.0 : 40.0	60.0 : 40.0

9.3 스스로 검증까지 붙여 온다

산출물에는 **모델 검증** 항목이 포함되어 있었다. 새로 만든 계산 모델에 개정안 조건(35.3%)을 그대로 넣었을 때 원 추계서의 수치가 재현되는지를 스스로 확인한 것이다.

> 산출물 원문 — 모델 검증
>
> "개정안(35.3%) 조건 적용 시 원 추계서 수치(지방세 84.8 / 순증 50.8 / 교부세 −16.3 / 교육교부금 −17.6)와 **연도별까지 정확히 일치**"
>
> ※ 반올림 처리까지 명시했다 — 수준값은 원 추계서와 연도에 따라 ±0.1 차이가 있을 수 있으나, 차분값은 원문과 완전히 일치한다는 단서를 스스로 달았다.

이것이 8장에서 원자료와 원문을 함께 넣어 둔 효과다. **대조할 기준이 폴더 안에 있으면 검증이 작업의 일부가 된다.**

산출물에 담긴 해석 예시

- **도달 경로가 결과를 지배한다**
  - 시나리오 A와 B의 2034년 단년 효과는 15.4조원으로 동일하나(종착 세율 동일), 10년 누적은 212.9조원 대 124.6조원으로 1.7배 차이
- **순증 비율은 약 60%로 고정된다**
  - 늘어난 지방세의 40.03%(지방교부세 19.24% + 지방교육재정교부금 20.79%)가 교부금 감소로 자동 상계
- **손실 주체와 수혜 주체의 불일치**
  - 지방소비세는 시·도 및 시·군·구로 안분되나 교육교부금은 시·도교육청 회계로 귀속

> 주의 — 이 해석들은 검증 대상이다
>
> 위 항목들은 AI가 생성한 **초안**이다. 전제·계수·귀속 구조에 대한 판단은 연구자가 다시 확인해야 한다. 14장의 "검증은 집필자의 책임"이 여기에 걸린다.

9.4 실무 메모

- **중지** — 글자가 나오는 동안에는 입력창 오른쪽의 정지 버튼으로 멈춘다. 이미 뒤에서 프로그램이 돌기 시작하면 중간에 끊기 어렵다

- **하드웨어** — GPU는 필요 없다
  - 연산은 대부분 서버에서 일어나고, 로컬은 텍스트·엑셀 처리 수준이다
  - 시연에 쓴 노트북도 내장 그래픽이었고 “충분히 괜찮다, 문제 없다”
  - 머신러닝으로 대량 학습을 돌릴 때에만 별도 그래픽카드가 의미가 있다
- **아카이빙** — 결과가 마음에 들면 그때그때 파일로 저장해 둔다. 이것이 쌓이면 아카이브가 되고, 다음 단계인 보고서 집필의 재료가 된다

제 10 장

실습 3 · 한글 보고서 자동 작성

1장에서 말한 도착점이다. 분석 결과를 **기관 고유의 서식**에 맞춰 한글 파일로 떨어뜨린다.

10.1 서식은 위계 구조다

기존 보고서를 열어 놓고 그 안의 위계를 하나씩 짚어 보자. 보고서 자동화란 결국 이 위계를 기계가 재현하게 만드는 일이다.

표 10-1 · 보고서 위계 구조 (예시 서식 분석 결과)

단계	기호	역할
1단계	아라비아 숫자 **1.**	가장 큰 구조. 장·절에 해당
2단계	네모 ■	대항목. 주요 주제
3단계	동그라미 ❍	중항목. 세부 설명을 다소 긴 개조식으로
4단계	줄표 –	소항목. 부연 설명
보조	당구장 ※	주석·참고 정보

※ 서술 형식도 함께 읽어야 한다. 예시 서식은 **개조식**으로, "~함"처럼 명사형으로 끝나거나 단어로 끝나는 형태였다. 이 교재의 본문에서도 같은 기호 체계를 그대로 쓰고 있다.

10.2 스타일 분석 지시

> `@지방소비세율.hwpx 이 양식의 보고서 스타일을 분석해 줘`
>
> PDF도 함께 있으면 같이 지목한다. "두 개를 같이 참고해서 분석해 줘"

AI는 문단 속성, 글머리 문단의 수준, 글꼴까지 추출해 스타일 정의로 정리해 준다.

한글 원본이 없고 PDF만 있을 때

외부에서 PDF로만 받은 자료라도 **70~80%는 구현해 낸다.** "글꼴이 뭐였다"는 것까지 알아낸다. 한글 원본이 있으면 당연히 더 쉽다.

10.3 가장 확실한 방법 — 원본을 복제한다

스타일을 처음부터 재구성하는 것은 시간이 걸린다. 한글 원본이 있다면 더 확실한 길이 있다.

> 지방소비세율.hwpx의 내용을 그대로 복제해서, 안의 내용만 새 분석 결과로 바꿔 넣어 줘

서식·글꼴·글머리표가 이미 완성된 파일을 껍데기로 쓰고 내용만 교체하는 방식이다. 실습을 준비할 때 **한글 원본**을 확보해 두라고 한 이유가 이것이다.

10.4 막히면 액션 테이블

AI가 한글 조판에서 막히는 경우가 있다. 한글 관련 명세를 늘 기억하고 있는 것이 아니기 때문이다.

> 액션 테이블을 봐
>
> 1장에서 폴더에 넣어 둔 `ActionTable_2504.pdf`를 가리킨다. 우리가 읽어 줄 필요는 없다. "거기 다 있습니다. 알아서 찾고 알아서 문제 해결할 겁니다."

이 명세로 처리되는 범위에는 **보고서에 그림을 삽입하는 것**까지 포함된다.

10.5 실전 결과

실제로 운영 중인 정책보고서 자동 작성 사례는 다음과 같이 굴러간다.

- 제목·목차 구조를 프로그램으로 잡아 두고, 본문은 AI가 채운다
- **스스로 결정할 수 없는 것만 사람에게 묻는다**
  - "여기는 표가 들어가야 되겠다" → 먼저 자리를 그려 놓고 데이터를 되묻는다
  - 1장·2장·3장을 쓰다가 지시받은 부분을 고치는 중이었다
- 양식 재현도는 거의 완벽하다. 다만 해당 글꼴이 설치되지 않은 PC에서는 배치가 약간 틀어져 보인다

> 저희 연구원에서 쓰는 양식은 90%는 타이핑하지 않고 보고서를 만듭니다. —
> 아니, 50%는 되게 겸손합니다. **99%. 절대 손대지 않고 만들어 냅니다.**

※ 다만 자동화 수준은 문서의 성격에 따라 조절할 일이다. 수치의 무게가 큰 재정 보고서는 "하나하나 손을 좀 대야 한다"는 유보가 따라붙는다. 비전·정책 제안형 보고서일수록 자동화 여지가 크다.

10.6 한글 → PDF 저장이 깨지는 경우

주의 – 자주 발생하는 실패

한글에서 PDF로 저장했는데, 마우스로 긁어 보니 **글자가 낱자로 동글동글 흩어져** 있는 경우가 있다.
이런 PDF는 AI가 연관 분석을 하지 못한다.

> 이건 못 씁니다. 이렇게 저장하시면 안 되고, 한글에서 PDF로 저장하시더라도
> **잘 저장됐는지 대충이라도 한번 체크를 하셔야 돼요.**

확인법 – 저장한 PDF를 열어 본문을 마우스로 드래그해 본다. 한 번에 죽 긁히면 정상, 낱자로 끊기면 실패다.

복구법 – PDF 도구의 **OCR**을 수행해 흩어진 글자를 정상화한다. 다만 PDF의 OCR이 100% 되는 경우는 드물다.

※ 이 실패는 대개 **정품 한글이 아닌 대체 워드 프로그램**에서 저장했을 때 난다. 서드파티 한글 호환 프로그램은 PDF 변환 안정성이 떨어질 수 있다.

아티팩트 옵션은 지나간다

작업 중 “아티팩트로 만들까요?”라는 제안이 뜰 수 있다. 웹페이지 형태의 미리보기를 만드는 기능으로, 목표가 마크다운과 HWPX인 지금 단계에서는 신경 쓰지 않아도 된다.

PART 4

한 번의 작업을 자산으로

같은 일을 두 번 설명하지 않는 구조를 만든다. 지시 방식, 축적, 교차검증, 그리고 마지막까지 사람이 지는 책임에 관하여.

제 11 장

컨텍스트 엔지니어링

"너는 재정 전문가야"로 시작하는 프롬프트 기법은 이미 지나갔다. 지금은 **상황을 설명하고 방향을 주는** 단계다.

11.1 프롬프트 엔지니어링은 끝났다

> 3년 전, 챗GPT 처음 나왔을 때 "잘 물어야 대답을 잘한다"가 프롬프트 엔지니어링이었죠. 너는 재정 전문가야, 이렇게 시작하는 롤을 설정하고. —
> **이제 지나갔죠. 이렇게 안 씁니다.**

역할을 부여할 필요가 없다. 그냥 묻고 싶은 것을 물으면, 이미 대화의 맥락에서 상황을 파악한다.

11.2 네 단계

그림 11-1 지시 방식의 네 단계. 오른쪽으로 갈수록 문장으로 일일이 지정하는 대신 방향과 판단을 위임한다. 3단계 하네스부터 실질적인 자동화가 시작된다.

컨텍스트 — 상황을 말한다

> 우리 원장님한테 지금 보고를 해야 돼. 보고서가 대충 나오면 내 입장이 어떻게 되겠냐. 자세해야 돼.
>
> 다급한 상황을 설명해 주면 "아, 심각하구나. 대충 얘기하지 말아야 되겠다" 하고 맥락을 잡는다.

하네스 — 방향을 잡아 준다

말을 타고 방향을 계속 제시하는 것에 비유할 수 있다. 목적지를 북쪽으로 정해 두면, 중간에 장애물을 만나 돌아가더라도 계속 북쪽으로 간다.

> 내 의도 알지? 나 밥 먹고 올 테니까 네가 다 만들어라. 그런데 중간에 진짜 중요한 의사결정은 나한테 물어봐.

메타 — 질문 자체를 묻는다

> 내가 지금 이 상황에서 너한테 뭘 물어야 잘 물었다고 할 수 있을까? 내가 물어야 할 질문을 알려 주면 내가 그걸 물을게.

11.3 웹의 개인화 설정은 여기 따라오지 않는다

자주 나오는 질문

Q. 웹 채팅에서 페르소나나 답변 형식을 설정해 두는데, 그게 VS Code에서도 영향을 주나요, 아니면 독립인가요?

A. 독립입니다. 웹의 개인화·메모리는 그 채팅 계정에 귀속된다. 편집기 쪽에서는 **폴더 안의 지침 파일**이 그 역할을 대신한다 — 다음 장의 주제다.

제 12 장

인스트럭션과 스킬

보고서 스타일을 매번 다시 설명하지 않는 방법. **한 번 잘 된 방식을 파일로 굳혀 두는 것**이다.

12.1 지침 파일 만들기

기관에는 보고서 작성 매뉴얼이나 지침이 이미 있다. 그것을 프로젝트 폴더 안에 마크다운 파일로 넣어 둔다.

1. 탐색기에서 마우스 오른쪽 → **New File**.
2. 이름은 무엇이든 좋다. `instruction.md`, `지시사항.md` 등.
3. 기존 매뉴얼·지침 내용을 붙여 넣는다.
4. AI에게 "이게 우리 보고서 스타일이야"라고 알려 준다.

예시 폴더에 들어 있는 `skill-instruction.md`는 한 연구기관의 보고서 표준을 지침으로 옮긴 것이다. 다음과 같은 항목이 규정되어 있다.

> 지침 파일에 들어가는 것
>
> - **문서 구조** — 겉표지 → 내지표지 → 연구진 → 요약 → 목차 → 본문 → 참고문헌
> - **페이지 설정** — A4, 여백(상25·하20·좌25·우20mm), 장은 홀수 페이지 시작, 페이지 번호 하단 중앙
> - **문체 규칙** — 명사형 종결("~임", "~함"), 수치 표기("26.3만 명", "9.9%")
> - **위계 구조** — ■ 대항목 / ❍ 중항목 / – 소항목 / ※ 참고 (10장의 표 10-1)
> - **표·그림 형식** — `[표 X-X] 제목 (단위: 명, 원)`, 헤더 음영, 숫자 우측 정렬, 출처 필수

※ 이 교재도 같은 규칙을 따라 조판했다. 그림 번호를 `[그림 X-X]` 형식으로, 위계 기호를 ■ ❍ – ※로 쓴 것이 그 결과다.

12.2 지침을 손으로 쓰지 않는 방법 — 스킬

지침을 처음부터 손으로 다 쓰는 것은 노가다다. 반대로 간다. **잘 나온 결과물에서 규칙을 역산하게 한다.**

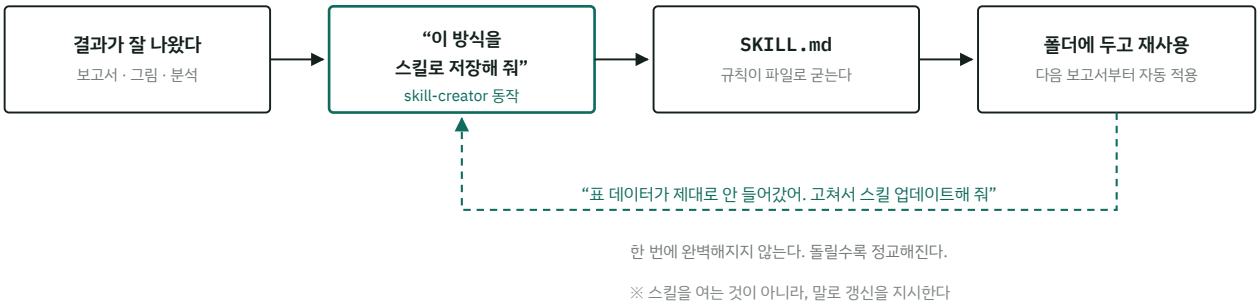


그림 12-1 스킬 축적 순환. 파인 튜닝하듯이 내 노하우가 쌓여 가는 파일이라고 보면 된다. 파일을 직접 편집하지 않고 대화로 갱신한다는 점이 핵심이다.

> 이 보고서들의 공통 부분을 찾아서, 내가 이 스타일대로 보고서를 만들 수 있게 스킬로 저장해 줘
>
> 기존 보고서 두세 개를 함께 지목한다. 전부 줄 필요는 없다 — "위계 구조와 스타일만 파악할 수 있도록" 앞부분만 끊어서 줘도 된다.

스킬이란

용어 — 스킬(SKILL)

클로드가 붙인 정식 용어이지만, 실체는 **마크다운으로 쓰인 지시서**다. 로컬 폴더에 파일로 존재한다.

이름에 얽매일 필요는 없다. "나는 스킬 말고 인스트럭션이라고 부를래"라고 선언하고 시작해도 된다.
AI가 인식만 하면 된다.

※ .md와 스킬을 혼동하지 않아도 된다. **.md는 형식**이고 **스킬은 그 파일의 역할**이다.

12.3 스킬 사용하기

1. 생성된 `SKILL.md`를 프로젝트 폴더에 둔다.
2. 작업을 시작할 때 "**지침부터 읽어 봐**"라고 한다.
3. 이후 데이터만 주면 그 스타일대로 결과가 나온다.
4. 결과가 어긋나면 어긋난 지점을 말해 스킬을 갱신시킨다.

> 문서 지침 딱 주고, 새 직원이 왔는데 "지침부터 읽어 봐" — 똑같습니다.

12.4 문서에만 쓰는 것이 아니다

- 그림 생성 — 이미지 생성 API 키를 넣어 두고, 몇 번 그려 본 뒤 “이 스타일대로 스킬을 만들어 줘”
- 영상 · 분석 방법론 · 검색 절차 등 **반복되는 작업 방식이면 무엇이든** 스킬이 된다

본질은 하나다. **정교하게 다듬은 지시를 버리지 않고 재사용하려고 만드는 것**이다.

※ 인터넷에는 스킬을 판매하는 시장도 생겼다. “한 개 50만 원, 100만 원에 팔고” — 파일 자체는 마크다운 몇 장에 불과하지만, 그 안에 축적된 조율의 값이다.

제 13 장

시각화와 통계 도구 연계

보고서 안의 그림 문제. 개념도·다이어그램을 엑셀 도형으로 그리던 일을 대체한다.

13.1 다이어그램 — SVG와 draw.io

용어 — SVG

Scalable Vector Graphics. 그림을 점의 집합이 아니라 **좌표와 도형의 명세**로 기술하는 형식이다.
확대해도 깨지지 않고, 내용이 텍스트라 나중에 고치기 쉽다. 이 교재의 모든 도해도 SVG로 그려져 있다.

1. 웹에서 **draw.io**를 연다. 무료이고 설치가 필요 없다.
2. AI에게 그리고 싶은 것을 설명한다. 손으로 그린 조직도를 사진 찍어 붙여 넣고 "이대로 그려 줘"라고 해도 된다.
3. 결과를 draw.io에 붙여 넣으면 편집 가능한 다이어그램이 된다.
4. 색·배치를 조정한 뒤 보고서에 넣는다.

주의 — 반드시 "XML로 써 줘"라고 한다

그냥 요청하면 **SVG 다운로드 버튼** 형태로 내주는 경우가 있다. 그러면 draw.io에 붙여 넣기 위해 한 번 더 변환해야 한다. 처음부터 XML로 달라고 하는 편이 빠르다.

```
› 같은 내용을 draw.io에서 편집 가능한 XML로 써 줘
```

13.2 머메이드(Mermaid)

같은 다이어그램을 텍스트 문법으로 쓰는 또 다른 방식이다. 마크다운 문서 안에 코드로 들어가므로 문서와 함께 버전 관리된다.

```
› 머메이드 형태로 그려 줘
```

※ SVG와 머메이드 중 어느 쪽이 편한지, 또 기존처럼 엑셀 도형으로 그리는 것과 비교해 어느 쪽이 나은지는 직접 한 번씩 해 보고 정하는 편이 좋다.

13.3 도구별 강약 — 시연 중 관찰된 것

같은 지시를 두 도구에 각각 넣어 보면 강약이 드러난다.

표 13-1 · 작업 성격에 따른 도구 선택

작업	유리한 쪽	비고
보고서 집필 · 해석	**클로드**	"클로드로 글을 쓴다고 할 때는 클로드를 쓰시는 게 좋다"
프로그램 작성	**코덱스**	"프로그램을 너무 잘 짭니다"
다이어그램 · 도식	상황에 따라	도식 생성은 GPT 쪽이 빠른 경우가 많다
R · 통계 스크립트	**GPT**	"클로드보다는 GPT가 좀 더 낫습니다"

※ 도구 간 우열은 버전에 따라 계속 바뀐다. 위 표는 고정된 결론이 아니라 **"작업 성격에 따라 갈아탄다"**는 원칙으로 받아들이는 편이 좋다.

13.4 R · Stata 연계

- R Studio를 쓰고 있다면 그대로 이어 쓰면 된다
  - 데이터셋을 분석할 **스크립트를 AI에게 받아** R Studio에 붙여 실행
  - 상관관계 분석, 회귀 등 필요한 스크립트가 바로 나온다
- Stata도 마찬가지다. "같은 통계 툴이라서 최대한 그대로 쓰시면 된다"
  - 초기에는 오류가 많았으나 최근에는 훨씬 나아졌다

13.5 막히면 화면을 찍어서 물어본다

> 화면 쓰다가 뭐 이런 툴 같은 게 안 되면, 저는 제일 원초적으로 **다 화면 캡처해 가지고** — 그게 제일 좋을 수도 있어요. 오류 찾아 달라고.

도구 이름도 모르겠고 오류 메시지도 낯설 때 가장 확실한 방법이다. 화면을 캡처해 붙여 넣고 "무엇이 잘못됐는지 찾아 줘"라고 하면 된다.

제 14 장

교차검증과 검증 책임

두 개의 AI를 서로 감사시킨다. 그러나 마지막 판단은 언제나 사람의 것이다.

14.1 코덱스 붙이기

1. 확장(Extensions)에서 `codex`를 검색한다.
2. 발행처가 **OpenAI**인 것을 설치한다.
3. 설치하면 클로드 코드 옆에 **Codex** 탭이 붙는다 (그림 6-2).
4. 자기 계정으로 로그인하면 챗봇 형태로 사용할 수 있다.

14.2 감사(Audit) 시키기

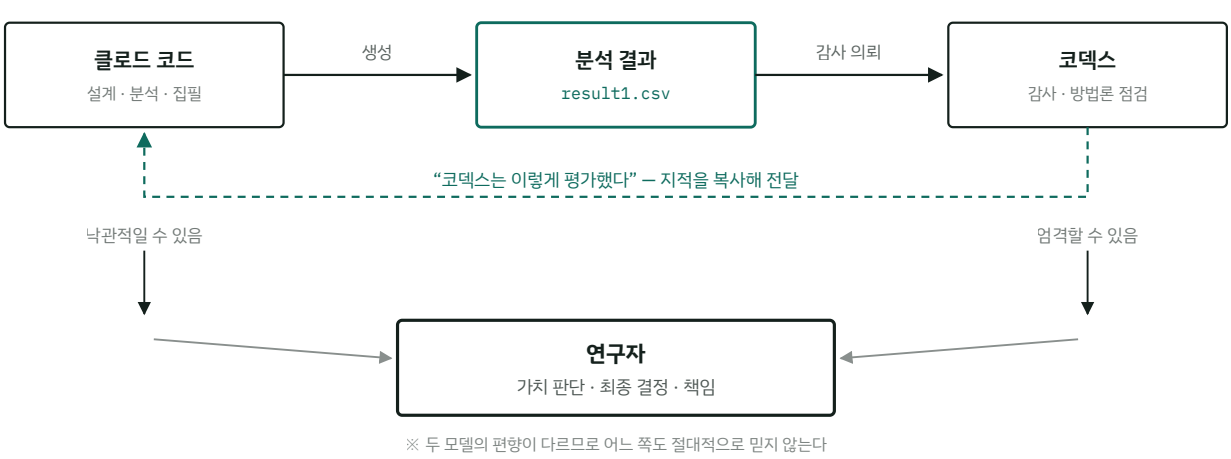


그림 14-1 교차검증 구조. 분석을 수행한 쪽에 그 분석의 검증을 맡기지 않는 것이 요지다. 두 모델은 학습된 편향이 다르므로 서로 다른 지적을 내놓는다.

> 앞에서 분석한 파일은 ○○이고, 기존 데이터는 ○○였어. 제대로 분석했는지 네가 감사를 해줘
>
> 코덱스 쪽 입력창에 넣는다. 프로그램 관점의 검증과 데이터 분석 방법론 관점의 지적이 함께 나온다.

코덱스의 지적을 복사해 클로드에 되돌리면 "내가 잘못했다"고 수정하기도 하고, 반대로 "클로드가 너무 낙관적으로 평가했다"는 지적이 나오기도 한다.

주의 — 상호 검증도 만능이 아니다

두 모델이 서로 다른 편향으로 학습되었기 때문에 서로 다른 답을 낸다. **어느 쪽도 절대적으로 믿으면 안 되고**, 가치 판단이 개입하는 사안은 연구자가 스스로 결정해야 한다. 둘 다 해석이 가능한 경우도 생긴다.

14.3 수정 — 한글로 넘기기 전에 끝낸다

결과가 마음에 들지 않을 때 어디서 어떻게 고치는가.

- **한글 변환은 마지막에 한 번**
  - 그 전까지는 `.md` 나 텍스트 파일 상태로 두고 수정한다
- **고칠 곳을 정확히 지목한다**
  - "그냥 바꿔 줘" 하면 엉뚱한 데를 고친다
  - 해당 부분을 **복사해서 붙여 넣고** "이 부분을 이렇게 바꿔 줘"라고 한다
  - 파일명과 위치(칼럼·행)를 함께 주면 바로 찾아간다
- **내 의견을 개진한다**
  - "40%로 올릴 경우의 결과에 대해 다른 자료를 찾아보니 의견이 다르다" → 근거를 제시하고 판단이 서면 고치게 한다

14.4 버전 관리와 롤백

- 중간중간 버전을 매겨 저장한다 — 버전 1, 2, 3...
- 다른 방식으로 해석해 보다가 되돌아와야 할 때
  - **"롤백해 줘. 다시 뒤로 돌려"**라고 하면 그 전으로 돌아간다
- "관리하기 나름입니다" — 도구가 대신해 주지 않는 부분이다

※ 웹 클로드 코드나 터미널 환경에서는 중간 파일을 찾아가기 어렵다. 편집기 위에서 작업하는 이유가 여기서도 드러난다. 파일이 눈에 보이고, 언제든 열어서 확인할 수 있다.

14.5 검증은 집필자의 책임

> 검증은 **편집자(집필자)의 책임**이에요. 자료가 없어서 얘가 해석 못 하는 것은 이 친구 책임이 아니잖아요. 그래서 Perplexity든 내가 갖고 있는 원자료든 **잘 갖다 대 줘야 됩니다.**

- PDF를 줬는데 이미지로 되어 있어 OCR이 안 되고 인식률이 떨어지는데도 그냥 믿고 준 경우
  - 이것은 **사람의 책임**이다. 어떻게든 엑셀로 바꿔 주는 것이 가장 좋다
- 원자료를 제대로 주지 않고 쓰라고 하는 것은 곤란하다
- 목표는 하나다 — **신뢰도를 높이는 것**

> 자기가 AI가 쓰는 게 아니고 **자기가 컨트롤하면서 쓰기 때문에** 보고서가 더 풍성해집니다. 더 좋아지고, 나중에 편집하는 노가다가 없기 때문에. — 그런데도 한 번쯤은 검수는 해야 되죠.

14.6 학습 곡선 — 마무리

- 손에 붙는 데 걸리는 시간
  - “일주일만 하시면 손에 딱 붙습니다. 일주일이 뭡니까? 한 3일만 하시면”
- 앞단계(아이디어 착안·구성)도 여기서 한다
  - “저는 여기서 다 씁니다. 이게 제일 낫습니다”
- 진짜 허들은 기능이 아니라 **첫 화면을 여는 것**이다
  - “여기를 못 넘으신 분들은 다 끝나거든요. 넘는 사람은 그다음부터 새로운 세상이 열리는 거지”

> 이 동네까지 오면 더 이상 못 돌아갑니다.

대신 방향은 사람이 잡는다. 자료를 잘 갖다 대고, 설계를 하고, 알고리즘을 짜고, “이렇게 갔으면 좋겠다”를 계속 제시하는 것. 사람이 서는 자리는 **슈퍼바이저**다.

APPENDIX

부록

용어 정리와 착수 체크리스트.

부록 A

용어집

용어	뜻
HWPX	ZIP으로 압축된 XML 묶음 형태의 한글 문서. 외부 프로그램이 내부 구조를 읽을 수 있다 (2장)
OLE	구형 HWP가 쓰는 객체 지향 복합문서 방식. 외부 분석이 막혀 있다
XML	확장 가능한 표시 언어. 태그로 구조를 명시해 기계가 읽을 수 있게 한다
마크다운(.md)	`#`으로 제목 위계를 표시하는 단순 텍스트 형식. AI가 가장 잘 인식한다 (3장)
액션 테이블	한글과컴퓨터가 배포하는 HWPX 조작 메소드 명세. AI가 참조해 한글 문서를 직접 조판한다 (1장)
확장 / 익스텐션	VS Code에 기능을 얹는 추가 프로그램. 클로드 코드·코덱스·PDF 리더 모두 확장이다
터미널	명령을 직접 입력하는 창. 예전 도스 명령창과 같은 자리 (5장)
에이전트	목표를 주면 끝날 때까지 스스로 수행하고, 필요할 때 먼저 되묻는 방식 (4장)
바이브 코딩	편집기에서 말이나 자연어 문장으로 작업을 지시하는 방식
컨텍스트	모델이 한 번에 다룰 수 있는 정보의 범위. 또한 지시할 때 함께 주는 상황 설명 (7장·11장)
하네스	세부를 지정하는 대신 방향만 제시하고 중요 결정만 확인받는 지시 방식 (11장)
스킬 / 인스트럭션	반복되는 작업 규칙을 굳혀 둔 마크다운 지시서 (12장)
SVG	좌표와 도형 명세로 기술되는 벡터 그림 형식. draw.io에서 편집 (13장)
OCR	이미지 속 글자를 문자로 인식하는 처리. 깨진 PDF 복구에 쓴다 (10장)
롤백	이전 상태로 되돌리기 (14장)

부록 B

착수 체크리스트

강의 다음 날 자리에 앉아 그대로 따라 할 수 있는 순서.

1단계 · 계정과 도구 (한 번만)

- 클로드 **유료 구독**을 확보한다 (무료·저가 요금제로는 진행 불가)
- **VS Code**를 설치한다
- 확장에서 **Claude Code for VS Code (Anthropic)** 설치 → 로그인 → 승인
- 액션 테이블 PDF를 내려받아 둔다

2단계 · 사안별 폴더 (매 과제마다)

- 사안 이름으로 폴더를 하나 만든다
- 넣을 것
  - 원자료(엑셀·CSV) — 있으면 반드시
  - 대상 문서 HWPX (구형 HWP는 미리 변환)
  - 교차 판독용 PDF
  - 액션 테이블 PDF
  - 조사 자료 `.md`
  - 보고서 지침 / 스킬 파일
- VS Code에서 **Open Folder**

3단계 · 작업

- `@`로 파일을 지목해 판독시킨다 → HWPX·PDF 교차 확인
- 핵심 수치는 원문과 눈으로 대조한다 (특히 천 단위 콤마)
- 분석을 지시하고 결과를 **CSV**로 떨어뜨린다
- 수정은 `.md`/텍스트 단계에서 위치를 지목해 끝낸다
- 보고서 서식을 분석시키거나 원본을 복제해 내용만 교체한다
- 한글(HWPX) 변환은 **마지막에 한 번**

4단계 · 검증과 축적

- 코덱스에 감사를 의뢰해 교차검증한다
- 법령·통계는 **시점**을 반드시 확인한다
- 결과가 좋으면 “이 방식을 스킬로 저장해 줘”
- 버전을 매겨 저장하고, 되돌릴 때는 롤백을 지시한다

막 혔을 때

- AI가 한글을 못 읽는다 → **“액션 테이블을 봐”**
- 표가 깨진다 → PDF·이미지로 **교차 판독** 지시
- 한글→PDF가 낱자로 흩어진다 → **OCR 수행**
- 도구 자체가 안 된다 → **화면을 캡처해서 물어본다**

타이핑하지 않는 보고서 — 실무 강의를 교재로 재구성한 것으로, 준비 → 환경 구축 → 실습 → 축적의 순서를 그대로 따릅니다. 4시간 안팎의 실습형 교육 한 회차 분량이며, 각 부를 독립된 세션으로 나눠 진행해도 됩니다.

그림에 관하여 — 그림 5-1·6-2·6-3·7-1·8-1·9-1은 실제 작업 화면을 캡처한 것입니다. 그림 1-1·2-1·2-2·3-1·6-1·11-1·12-1·14-1은 개념과 화면을 도해로 재구성했습니다. 화면 구성은 도구 버전에 따라 달라질 수 있습니다.

수치에 관하여 — 9장의 표와 발췌는 실습 과정에서 실제로 생성된 산출물에서 그대로 옮긴 것으로, **교육용 예시**입니다. AI가 생성한 초안이므로 어떤 형태로든 인용·활용하기 전에 원 추계서와의 대조가 필요합니다.