Chapter 4-12: 최종 진화, 개인 맞춤형 AI 분석 비서

Prologue: 당신의 언어를 이해하는 조수의 탄생

지금까지 우리는 AI에게 일반적인 질문을 던져왔습니다. 하지만 이제 우리는 AI에게 '나'를 가르치고, '나의 관점'에서 채용 공고를 분석하도록 명령할 것입니다. "나의 강점과 일치하는가?", "내가 새로 배워야 할 것은 무엇인가?" 와 같은, 오직 신호용 님만이 던질 수 있는 질문에 답하는 개인 비서를 만드는 것입니다.

이번 최종 챕터에서는, 신호용 님이 직접 설계하신 고도로 개인화된 프롬프트를 시스템에 이식하고, 그 결과를 Notion에 가장 가독성 높은 형태로 저장하여, 이 프로젝트의 '버전 1.0'을 완벽하게 완성할 것입니다.

1. 학습 목표 (Objectives)

- 자신의 기술 스택과 같은 '컨텍스트(Context)'를 프롬프트에 포함시켜, 고도로 개인화된 AI 응답을 생성할 수 있다.
- AI가 생성한 결과물의 출력 형식(마크다운)을 지정하여, 가독성을 극대화하는 방법을 이해한다.
- 프로젝트의 최종 목표에 맞춰, Notion 데이터베이스 스키마와 저장 로직을 최종적으로 완성한다.

[Mission 30] AI 분석 비서 최종 완성

Step 1: `gemini_analyzer.py`의 프롬프트 최종 교체

`analyze_job_posting` 함수 안의 `prompt`를, 신호용 님이 작성해주신 완벽한 명세서로 교체합니다.

```
# analysis/gemini_analyzer.py (프롬프트 최종 교체)

# ...
def analyze_job_posting(job_description: str):
    # ...
    # ★★★★★ 신호용 님의 개인화된 최종 프롬프트 ★★★★★
    prompt = f"""
    너는 지금부터 IT 채용 전문가의 어시스턴트 역할을 수행한다.
    아래에 입력되는 채용 공고의 전체 텍스트를 분석하여, 다음 형식에 맞춰 핵심 정보만을 추출하고 분석하여 결과를 출력해야 한다.
    각 항목에 정보가 없는 경우, '정보 없음'이라고 명시한다.
    반드시 마크다운 형식으로 응답해줘.

    ---
    ### 1. 핵심 자격 요건
    - **경력사항**: 공고에 명시된 요구 경력을 정확히 추출한다. (예: "신입", "신입 ~ 3년차", "5년 이상")
    - **필수 언어/프레임워크**: '자격 요건', '필수 사항' 섹션에 명시된 프로그래밍 언어와 프레임워크를 모두 나열한다.

    ### 2. 주요 담당 업무
    - 공고의 '주요 업무' 섹션을 바탕으로, 이 포지션이 수행할 핵심 역할 2~3가지를 불렛 포인트로 요약한다.
```

```
    ### 3. 기술 적합도 분석
    - **나의 강점과 일치하는 기술**: 공고에 언급된 기술 중, 나의 핵심 기술 스택
(Python, Django, PostgreSQL, AWS, Git, Nginx, Gunicorn)과 일치하는 것을 모두
찾아 나열한다.
    - **새롭게 배워야 할 기술**: 공고에 언급된 기술 중, 나의 핵심 기술 스택에 포
함되지 않은 새로운 기술을 모두 찾아 나열한다.

    ### 4. 우대사항 요약
    - '우대 사항' 섹션에서 가장 중요해 보이는 순서대로 최대 3가지만 요약하여 나열
한다.
    ---

    [채용 공고 내용]
    {job_description}
    """
    #
★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★

    try:
        response = model.generate_content(prompt)
        # Gemini는 이제 마크다운 형식의 '텍스트'를 반환할 것임
        return response.text
    except Exception as e:
        print(f"  🚨 [Gemini 오류] API 호출 중 문제 발생: {e}")
        return None
```

(참고: `json.loads`는 더 이상 필요 없으므로 `gemini_analyzer.py`에서 삭제했습니다. 이제 순수한 텍스트를 다룹니다.)

Step 2: Notion 데이터베이스 스키마 최종 수정

신호용 님의 제안대로, 데이터베이스를 더 깔끔하고 효율적으로 만듭니다.

1. Notion DB에서 기존의 `AI 요약`, `핵심 기술` 속성을 **삭제**합니다.
2. `+` 버튼을 눌러, **`AI 분석 결과`** 라는 이름의 **`텍스트`** 유형 속성을 새로 추가합니다.

Step 3: `main.py`의 Notion 저장 로직 최종 완성

`main.py`의 Notion 저장 루프에서, `properties_to_save`를 구성하는 부분을 아래의 최종 코드로 교체합니다.

```
# main.py (Notion 저장 루프 최종 수정)

# ...
for i, job in enumerate(all_jobs):
    # ... (중복 확인 및 필터링 로직은 동일) ...

    # --- 3단계: Gemini AI 분석 ---
    analysis_result_text = None # 텍스트 결과를 담을 변수
    if description:
        analysis_result_text = analyze_job_posting(description)
```

```
    # --- 4단계: Notion에 저장 ---
    properties_to_save = {
        '직무': {'title': [{'text': {'content': title}}]},
        '회사명': {'rich_text': [{'text': {'content': company}}]},
        '링크': {'url': link},
        '출처': {'rich_text': [{'text': {'content': source}}]},
        '수집일': {'date': {'start': collection_date}},
        '마감일': {'rich_text': [{'text': {'content': deadline}}]}
    }

    # ★★★★★ AI 분석 결과가 있을 경우, 'AI 분석 결과' 속성에 통째로 추가
★★★★★
    if analysis_result_text:
        properties_to_save['AI 분석 결과'] = {'rich_text': [{'text':
{'content': analysis_result_text}}]}
    #
★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★
★

    try:
        notion.pages.create(
            parent={"database_id": NOTION_DATABASE_ID},
            properties=properties_to_save
        )
        success_count += 1
        print(f"    -> ✅ '{title}' Notion 저장 성공!")
    except Exception as e:
        print(f"    -> 🚨 [오류] '{title}' 저장 실패! 원인: {e}")
# ... (최종 결과 요약은 동일)
```

마지막 최종 행동 계획 (Next Action)

1. [ ] `gemini_analyzer.py`의 `prompt`를 위 최종 프롬프트로 교체하고, `json.loads` 관련 코드를 삭제합니다.
2. [ ] Notion 데이터베이스의 속성을 정리하여, `AI 분석 결과`(텍스트) 속성을 추가합니다.
3. [ ] `main.py`의 Notion 저장 로직을 위 최종 코드로 업데이트합니다.
4. [ ] **마지막 최종 시운전**을 실행하여, Notion DB의 'AI 분석 결과' 칸에, 신호용 님이 완벽하게 설계한 형식의 분석 보고서가 아름다운 마크다운으로 저장되는 것을 확인합니다.