**지금까지 확정된 운영 원칙**

* **소스**: Yahoo Finance
* **대상**: ETF (지수/S&P500·Nasdaq100·Semiconductor부터 시작)
* **주기**: monthly, weekly, daily, 30m
* **저장 경로/규칙**: Listen2FM\_Viewer/data/usStocks/ETF/{TICKER}\_{frequency}.json
* **파일 포맷**: OHLCV 배열(JSON) — [{ time, open, high, low, close, volume }, ...]
* **리스트 관리**: etf\_list.json에서 티커/분류/레버리지/인버스 관리
* **실행**: 당분간 수동, 이후 자동 스케줄 전환
* **30분봉**: 야후가 제공하는 **현재 가용 구간만** 전량 수집(무제한 보존 X)

**스크립트 제작 직전, “딱 이것만” 더 확정해요**

1. **조정(Adjusted) 여부**

* 일/주/월봉: **조정 OHLC**(split/배당 반영) 사용 권장
  + 레버리지 ETF는 (정/역)분할이 잦아 **비조정 일봉은 단절/점프**가 생기기 쉬움
  + 이동평균/이격도 등 지표에 안정적
* 30분봉: \*\*원시(비조정)\*\*로 수집(인트라데이에는 실질 차이 없음)

1. **타임스탬프 & 정렬**

* time = **UTC epoch(초)** 로 통일 (오름차순 정렬)
* 중복 타임 제거(merge 시 동일 time은 덮어쓰기)

1. **주/월 집계 방식**

* 가능하면 **야후의 weekly/monthly**를 그대로 사용  
  (초기엔 간단·안전 / 추후 필요 시 daily→재샘플 전환 가능)
* 파일명과 내용 주기 일치(예: \*\_monthly.json이 정말 월봉)

1. **요청/안정성**

* 호출 묶음 단위(batch)로 순차 처리 + **짧은 대기**(rate-limit 회피)
* 실패 시 **2~3회 재시도** 후 스킵, 오류 로깅

1. **로그·마니페스트(선택 사항, 데이터 파일은 순수 배열 유지)**

* Listen2FM\_Viewer/data/usStocks/ETF/\_manifest.json에 최근 업데이트 시간을 기록해두면 상태 파악이 쉽습니다.  
  (데이터 파일에는 meta를 넣지 않기로 했으니, 별도 관리 파일로 분리)

**✅ 확정 규칙(최종)**

**0) 공통**

* 저장 위치: Listen2FM\_Viewer/data/usStocks/ETF/{TICKER}\_{frequency}.json
* 저장 포맷: **순수 배열** [ {time, open, high, low, close, volume}, ... ]
* time: **UTC epoch(초)**, **오름차순** 정렬 유지
* OHLC: **조정(Adjusted) 값 사용** *(일/주/월)*
* 30분봉: 원시 intraday(조정 개념 무의미), **야후가 제공하는 “가용 구간 전체”를 매번 수집**

**1) 월봉 (monthly)**

* **항상 전체 구간 풀 리프레시(처음부터 끝까지 다시 받기)**
* 이유: 데이터량 작고, 분할/수정 반영 재처리 안전
* 저장: 기존 파일 **완전 대체(atomic write)**

**2) 주봉 (weekly)**

* **전체 구간 풀 리프레시 권장** (월봉과 동일한 이유: 데이터량 작고 중요도 높음)
* 저장: 기존 파일 **완전 대체(atomic write)**

**3) 일봉 (daily)**

* **증분 병합 방식**(빠르고 안전)
  1. 기존 파일이 있으면 **마지막 캔들의 time = t\_last** 읽기
  2. **완충(overlap) 5~7거래일** 앞당겨서 t\_start = t\_last - 7일부터 오늘까지 요청
     + 이유: 분할/정정/휴일 등으로 최근 구간이 바뀌는 경우를 자동 보정
  3. 기존 배열과 새 배열을 **time 키로 병합**
     + 같은 time은 **신규 데이터로 덮어쓰기**
     + 병합 후 **오름차순 정렬** 및 **중복 제거**
  4. 저장: 기존 파일 완전 대체(atomic write)

**4) 30분봉 (30m)**

* **항상 “가용 구간 전체”를 통째로 가져와서 완전 대체**
  + 의도: “일봉 5/20이평 감시”가 목적 → 장기 과거 불필요
  + 야후 제공 범위(수주~수개월)가 곧 “필요 충분” 범위
* 저장: 기존 파일 완전 대체(atomic write)

*atomic write*: {파일명}.tmp에 먼저 저장 → 저장 성공 시 원본으로 **rename**.  
쓰다 끊겨서 빈 파일이 남는 사고를 방지합니다. 이전 파일은 옵션으로 .bak 보관 가능.

**🔁 “10일 전 수집 → 오늘 재수집” 시 동작 예**

* **월봉**: 오늘 전체 다시 받아서 \*\_monthly.json **통째로 교체**
* **주봉**: 오늘 전체 다시 받아서 \*\_weekly.json **통째로 교체**
* **일봉**: 기존 \*\_daily.json 읽어 t\_last 확인 → t\_last-7일부터 오늘까지 받기 → **중복 time은 덮어쓰기** → 병합/정렬 후 저장
* **30분봉**: 오늘 “가용 범위 전체” 다시 받아 **통째로 교체**

→ 결과: **모든 주기가 최신**, **조정/정정도 자동 반영**, **파일 손상/중복 없음**, **일봉만 네트워크 절약**

**🔒 안정성 체크리스트(실행 시 적용)**

1. **타임스탬프 변환**
   * 야후가 ms(밀리초)로 줄 수 있으니 **반드시 1000으로 나눠 초 단위**로 저장
   * 예: Math.floor(ms/1000)
2. **정렬 & 중복 제거**
   * 정렬: time 오름차순
   * 중복: **같은 time은 1건만**(최신으로 유지)
3. **결측·휴일**
   * 야후 원천을 그대로 쓰므로 휴일/반일장 데이터는 그대로 반영
   * 병합 시 **연속성만 보장**(누락일 있어도 오류 아님)
4. **레이트 리밋/리트라이**
   * 티커를 32초)\*\*
   * 실패 시 **2~3회 재시도** → 그래도 실패하면 스킵·로그
5. **파일 쓰기**
   * \*.tmp → fsync → rename(원자적 교체)
   * 옵션: 기존 파일 .bak 1개만 보관(최근 상태 롤백용)
6. **검증**
   * 저장 직후 **JSON 파싱 재확인**(형식 손상 여부)
   * 첫/끝 캔들 time 로그로 출력(모니터링용)

**📑 etf\_list.json 운용(요약)**

* 위치: data/usStocks/ETF/etf\_list.json
* 필드: symbol, name, category, leverage(1x/3x), direction(long/inverse), enabled
* 실행 시: enabled=true만 대상 처리
* 신규 ETF 추가/제외를 **파일만** 수정해서 반영

**🧠 자주 묻는 부분(이번 설계 기준)**

* **샘플 데이터와 “타임스탬프 & 정렬”이 다르냐?**  
  → 아닙니다. **지금 쓰시는 샘플 포맷 그대로** 유지합니다. (epoch 초, 오름차순, OHLCV 키 이름 동일)
* **주/월은 왜 풀 리프레시?**  
  → 데이터량이 작고 중요도 높아 **항상 최신 조정/정정 반영**이 안전합니다. 코드는 단순, 리스크 최소.
* **일봉만 왜 증분?**  
  → 데이터량이 크므로 네트워크/시간 절약 목적. 대신 1주일 **완충 병합**으로 정확성과 최신성 모두 확보.
* **30분봉은 왜 항상 전부?**  
  → 야후가 제공하는 구간 자체가 “최신 단기 감시”에 충분하며, 파일 크기도 감내 가능. 매번 전체 갱신이 더 간단하고 안전.

 data\_scripts/ 폴더에서 실행하든

 프로젝트 루트에서 python data\_scripts/collect\_us\_etf.py 로 실행하든  
모두 동일하게 data/usStocks/ETF/에 저장됩니다.