DM_timesplit_60_180_final 설계노트 **(Version v1.1 scope)**

1, 목적**(Why)**

동기간(0–180) 내에서 행동(Consistency)과 성과(LTV/Retention)를 같이 측정하는 한계를 줄이기 위해, 관측(0–60d)과 성과(60–180d)를 time-split한다.

즉, "초기 Activation(14d) + 초기 Consistency(0–60d)가 이후 성과(60–180d 구매/매출/리텐션)를 예측/설명하는가"를 검증하는 v1.1 핵심 검증 DM이다.

2, Grain

1 user_id = 1 row

기준일은 signup_date이며, Activation(0–14), Observation(0–60), Performance(60–180)로 window를 분리한다.

3, Input tables & Join key

- DM_user_window - 유저 속성 + 14d funnel reach(+ purchase) 입력, key - user_id
- sessions - 0–60d 관측 consistency 계산 + 180d 마지막주 리텐션 판정, key - user_id *(session_id*는 *volume*용*)*
- orders - 60–180d 성과(주문) 집계, key - user_id *(order_id*는 주문 단위*)*
- order_items - 60–180d 성과 매출(revenue = line_amount 합) 계산용, key - order_id
- 참고(스캔 최적화): bounds(min_signup, max_signup)로 sessions/orders를 min_signup ≤ date < max_signup + 180일로 먼저 필터한 뒤, 최종 집계에서 유저별 window 조건을 다시 적용한다.

4, Partition / Clustering

- **PARTITION BY :** signup_date 유저 코호트/기간 필터가 기본(가입월/기간 slice)
- **CLUSTER BY :** user_id downstream 조인/필터가 user_id 중심

5, Window 정의

- 기준일**: signup_date**

- **Activation window (0–14d)**
    - DM_user_window의 14d reach flags로 activation_stage_14d 생성
- **Observation window (0–60d)**
    - **signup_date ≤ session_date < signup_date + 60**일 (0~59)
    - 방문 리듬(consistency) 계산 윈도우
- **Performance window (60–180d)**
    - **signup_date + 60**일 **≤ order_date < signup_date + 180**일
    - 장기 성과(outcome) 계산 윈도우
- **Retention** 판정**(180d last week)**
    - **signup_date + 174**일 **≤ session_date < signup_date + 181**일
    - day_index 174~180 중 세션 1회 이상이면 TRUE (DM_retention_cohort의 day 180 포함 방식과 정합)
- 경계 규칙**:** 시작일 포함**,** 종료일 미포함 (단, retention은 +181로 day 180 포함)

6, Main Features and 계산 로직

A. Activation stage (14d)

- **activation_stage_14d**: A0~A5
    - Logic: purchase → checkout → add_to_cart → click → view → none 순서로 stage 부여
    - Source: DM_user_window의 has_*_14d flags

B. Observation Consistency (0–60d, 세션 기반)

- Volume controls
    - **session_cnt_obs_60d**: distinct session_id count
    - **active_days_obs_60d**: distinct session_date count

- Inter-visit stats (방문일 시퀀스 기반)
    - distinct (user_id, session_date) → LAG로 gap(days) 계산
    - **intervisit_mean/median/std/cv_obs_60d**
    - CV = std/mean (SAFE_DIVIDE + NULLIF)
- Weekly regularity
    - **active_weeks_obs_60d**: 활동 주 수
    - **weekly_active_ratio_obs_60d**: 활동 주 비율(활동 주 / 총 주)
- Score & Segment
    - **consistency_score_obs_60d** = z(active_days_obs_60d) - z(intervisit_cv_obs_60d)
    - **consistency_segment_obs_60d**: NTILE(5) → C1~C5
    - score가 NULL(방문일 2회 미만 등)은 보수적으로 **C1** 처리

C. Performance Outcomes (60–180d, 주문/매출)

- **orders_60_180**: distinct order_id count
- **revenue_60_180**: order_items(line_amount) → order_id 단위 revenue 합산 후 60–180 window에서 유저별 합
- **aov_60_180** = revenue_60_180 / orders_60_180
- **has_purchase_60_180** = (orders_60_180 > 0)

D. Retention (180d last week)

- **retention_last_week_180d**: day_index 174~180 중 세션 1회 이상이면 TRUE

7, Sanity checks

- PK 유일성: row 수 = distinct user_id 수

- Window 분리 정합성:

    - obs(0–60)와 perf(60–180)는 겹치지 않아야 함(60일 경계 포함/미포함 확인)
    - retention_last_week_180d는 174~180만 사용(181 end로 day 180 포함)

- 값 범위:

    - counts ≥ 0, revenue ≥ 0, aov ≥ 0(NULL 가능)
    - weekly_active_ratio_obs_60d는 0~1(NULL 가능)

- 세그먼트 분포: consistency_segment_obs_60d가 C1~C5로 과도하게 한쪽에 몰리지 않는지(특히 NULL→C1 비중)

- 매출 조인 중복 점검: order_rev(order_id) 1행 보장 여부 + 샘플 유저 검증

8. DM이 꼭 필요한지

필요 **-** 프로젝트의 **v1.1** 핵심**(**관측 **0–60 vs** 성과 **60–180** 분리**)**을 구현한 **DM**으로**,** 동기간 상관 한계를 완화한 설명**/**예측 구조를 제공한다**.**