DAX/Logic Notes

Active User Counts

```
-- Count of unique active users who did NOT adopt any new features
UniqueActive_NonAdopters = CALCULATE(
    DISTINCTCOUNT(user_weekly[user_id]),
    user_weekly[active] = 1,
    user_weekly[adopter_group] = "non_adopter"
)

-- Count of unique active users who adopted at least one new feature
UniqueActiveUsers_Adopters = CALCULATE(
    DISTINCTCOUNT(user_weekly[user_id]),
    user_weekly[active] = 1,
    user_weekly[adopter_group] = "adopter"
)

-- Total count of all unique active users
UniqueActiveUsers = CALCULATE(
    DISTINCTCOUNT(user_weekly[user_id]),
    user_weekly[active] = 1
)
```

Retention Metrics

```
-- Retention rate = retained users / all users
retention_rate = DIVIDE([retained_users], [all_users])
```

Feature-Level Adoption Logic

```
-- Create a user-feature mapping to track who used which feature (calculated table)
FeatureUserMap =
SUMMARIZE (
    activity_log,
    activity_log[user_id],
    activity_log[activity_type]
)

-- Count of distinct users who adopted any feature
FeatureAdopterCount =
CALCULATE (
   DISTINCTCOUNT ( FeatureUserMap[user_id] )
)

-- Retention rate for users who adopted a specific feature (post-launch)
FeatureRetentionRate =
CALCULATE (
   DISTINCTCOUNT ( user_weekly[user_id] ),
   TREATAS (
      VALUES ( FeatureUserMap[user_id] ),
      user_weekly[user_id]
   ),
   user_weekly[active] = 1,
   user_weekly[wk] >= 0
)

-- % of feature adopters retained in a specific week (e.g., from slicer)
FeatureRetentionRatePct =
VAR selectedWeek =
   MAX ( user_weekly[wk] )
RETURN
DIVIDE (
   CALCULATE (
      DISTINCTCOUNT ( user_weekly[user_id] ),
      TREATAS (
         VALUES ( FeatureUserMap[user_id] ),
         user_weekly[user_id]
      ),
      user_weekly[active] = 1,
      user_weekly[wk] = selectedWeek
   ),
   CALCULATE (
      DISTINCTCOUNT ( FeatureUserMap[user_id] )
   ),
   0
)
```

Support Tickets & Ratings

```
-- Filter all support tickets submitted after the launch date (calculated table)
TicketsPostLaunch =
FILTER(support_tickets, support_tickets[submitted_at] >= DATE(2025,2,20))

-- Calculate the average feature rating after launch
AvgRating_PostLaunch =
CALCULATE(
   AVERAGE(feedback[rating]),
   feedback[submission_timestamp] >= DATE(2025, 2, 20)
)

-- Calculate the average feature rating before launch
AvgRating_PreLaunch =
CALCULATE(
   AVERAGE(feedback[rating]),
   feedback[submission_timestamp] < DATE(2025, 2, 20)
)

-- Calculate the percentage change in ratings from pre-launch to post-launch
Rating_Change_Percent =
DIVIDE(
   [AvgRating_PostLaunch] - [AvgRating_PreLaunch],
   [AvgRating_PreLaunch]
) * 100

-- % of feedback that was negative (post-launch)
NegativeFeedbackPct =
DIVIDE(
  CALCULATE(COUNTROWS(feedback), feedback[comment_type] = "Negative",
feedback[submission_timestamp] >= DATE(2025, 2, 20)),
  CALCULATE(COUNTROWS(feedback), feedback[submission_timestamp] >= DATE(2025, 2,
20))
)

-- % of feedback that was positive (post-launch)
PositiveFeedbackPct =
DIVIDE(
  CALCULATE(COUNTROWS(feedback), feedback[comment_type] = "Positive",
feedback[submission_timestamp] >= DATE(2025, 2, 20)),
  CALCULATE(COUNTROWS(feedback), feedback[submission_timestamp] >= DATE(2025, 2,
20))
)
-- Overall user sentiment after launch (positive / negative / neutral)
UserSentiment_PostLaunch =
VAR Pos =
    CALCULATE(
        COUNTROWS(feedback),
        feedback[comment_type] = "Positive",
        feedback[submission_timestamp] >= DATE(2025, 2, 20)
    )
VAR Neg =
    CALCULATE(
        COUNTROWS(feedback),
        feedback[comment_type] = "Negative",
        feedback[submission_timestamp] >= DATE(2025, 2, 20)
    )
RETURN
    SWITCH(
        TRUE(),
        Pos > Neg, "Positive",
        Pos < Neg, "Negative",
        "Neutral"
    )
```