Table of Contents

# 📹 Video Tracking Scenarios - Complete Guide

## For Business Analysts, Product Owners, Developers & Testers

## 🎯 Purpose & Goal

### What Are We Trying to Achieve?

We want to **understand how users interact with our videos** by tracking their viewing behavior and calculating meaningful engagement metrics.

**Business Goal:** Answer questions like: - How much of each video do users actually watch? - Which videos keep users engaged? - Where do users drop off? - Are users rewatching content?

**Technical Goal:** Transform raw video events (play, pause, resume, end) into **one aggregated row per User+Video combination** with metrics like: - Total watch time - Completion percentage - Engagement score - Unique seconds watched

## 📊 The Big Picture

### Input vs Output

┌────────────────────────────────────────────────────────────────────────────────────────────┐  
│ INPUT: Raw Events (Many Rows) │  
├────────────────────────────────────────────────────────────────────────────────────────────┤  
│ timestamp | userId | videoId | eventName | position | Action │  
│ 2024-01-15 10:00:00| peter | video\_001 | video\_play | 0 | Started playing │  
│ 2024-01-15 10:00:30| peter | video\_001 | video\_pause | 30 | Watched 30s │  
│ 2024-01-15 10:01:00| peter | video\_001 | video\_resume | 30 | Resumed after 30s │  
│ 2024-01-15 10:02:30| peter | video\_001 | video\_pause | 120 | Watched 90s more │  
│ ... │  
└────────────────────────────────────────────────────────────────────────────────────────────┘  
 ↓  
 [ AGGREGATION PROCESS ]  
 ↓  
┌─────────────────────────────────────────────────────────────────┐  
│ OUTPUT: One Row Per User+Video │  
├─────────────────────────────────────────────────────────────────┤  
│ userId: peter │  
│ videoId: video\_001 │  
│ totalWatchTime: 130 seconds │  
│ watchPercentage: 43.3% │  
│ completionPercentage: 40% │  
│ engagementScore: 58.5 │  
│ sessionCount: 1 │  
└─────────────────────────────────────────────────────────────────┘

## 🎬 Understanding Video Events

### The Four Event Types

Our system tracks **four types of video events**:

| **Event** | **Symbol** | **Description** | **When It Fires** |
| --- | --- | --- | --- |
| **video\_play** | ▶️ | User starts video | User clicks play button from stopped state |
| **video\_pause** | ⏸️ | User pauses video | User clicks pause or video auto-pauses |
| **video\_resume** | ▶️ | User resumes video | User clicks play after pausing |
| **video\_ended** | ✅ | Video completed | Video reaches the end naturally |

### Event Data Structure

Each event contains:

{  
 timestamp: "2024-01-15 10:00:00", // Real-world time  
 userId: "peter", // Who is watching  
 sessionId: "session\_001", // Browser session ID  
 videoId: "video\_001", // Which video  
 eventName: "video\_play", // Event type  
 currentTime: 0.0 // Position in video (seconds)  
}

## 🔑 The Golden Rule: Event Pairs

### Why We Need BOTH Start and End Events

╔══════════════════════════════════════════════════════════════════╗  
║ THE MOST IMPORTANT CONCEPT ║  
╠══════════════════════════════════════════════════════════════════╣  
║ ║  
║ Watch time is ONLY counted between valid event PAIRS: ║  
║ ║  
║ ✅ VALID PAIRS (We count these): ║  
║ • video\_play → video\_pause ║  
║ • video\_play → video\_ended ║  
║ • video\_resume → video\_pause ║  
║ • video\_resume → video\_ended ║  
║ ║  
║ ❌ INVALID (We DON'T count these): ║  
║ • video\_play → [nothing] (browser closed) ║  
║ • video\_resume → [nothing] (browser closed) ║  
║ • video\_pause → video\_resume (not watching) ║  
║ ║  
║ WHY? Without both events, we cannot know how long the user ║  
║ actually watched. We use a CONSERVATIVE approach: only count ║  
║ what we KNOW for certain. ║  
║ ║  
╚══════════════════════════════════════════════════════════════════╝

### Visual Example

User watches video:  
  
▶️ play(0s) ━━━━━━━━━━━━━━━━━━━► ⏸️ pause(30s)  
└──────────────────────────────┘  
 ✅ COUNTED: 30s  
 (We have both events!)  
  
⏸️ pause(30s) ────────────────► ▶️ resume(30s)  
└──────────────────────────────┘  
 ❌ NOT COUNTED: 0s  
 (User not watching while paused)  
  
▶️ resume(30s) ━━━━━━━━━━━━━━━► ❌ [browser closed]  
└──────────────────────────────┘  
 ❌ NOT COUNTED: 0s  
 (No closing event = can't calculate)

## 📋 All Tracking Scenarios

Let’s walk through **every possible scenario** with examples showing raw input data and transformed output.

### **Scenario 1: Perfect Viewing - Start to Finish** ✅

**Description:** User plays video and watches until the end without interruption.

#### Raw Input Events:

timestamp | userId | videoId | eventName | currentTime | Action  
--------------------|--------|------------|--------------|-------------|----------------------------------  
2024-01-15 10:00:00 | anna | video\_001 | video\_play | 0 | Started playing from beginning  
2024-01-15 10:05:00 | anna | video\_001 | video\_ended | 300 | Watched 300s, video completed

#### Visual Timeline:

Video (300s duration):  
0s ══════════════════════════════════════════════════► 300s  
├────────────────────────────────────────────────────┤  
│ ✅ Watched continuously: 300s │  
└────────────────────────────────────────────────────┘  
  
▶️ play(0s) ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━► ✅ ended(300s)

#### Calculated Output:

userId: anna  
videoId: video\_001  
videoDuration: 300s  
  
# Watch Time Metrics  
totalWatchTime: 300s # Full video watched  
uniqueSecondsWatched: 300s # All seconds unique (no replay)  
watchPercentage: 100% # 300/300 \* 100  
completionPercentage: 100% # Reached the end  
  
# Session Metrics  
sessionCount: 1 # Single session  
maxPositionReached: 300s # Got to the end  
completed: true # Video finished  
completionCount: 1  
  
# Interaction Metrics  
pauseCount: 0 # Never paused  
forwardSkipCount: 0 # No skipping  
backwardSkipCount: 0 # No rewinds  
  
# Engagement  
engagementScore: 60.0 # High score  
engagementTier: "High" # Tier assignment

**Business Insight:** Perfect engagement - user watched entire video without interruption.

### **Scenario 2: Simple Pause & Resume** ⏸️▶️

**Description:** User pauses video, takes a break, then resumes and finishes.

#### Raw Input Events:

timestamp | userId | videoId | eventName | currentTime | Action  
--------------------|--------|------------|----------------|-------------|----------------------------------  
2024-01-15 10:00:00 | bob | video\_001 | video\_play | 0 | Started playing from beginning  
2024-01-15 10:01:00 | bob | video\_001 | video\_pause | 60 | Watched 60s, then paused  
2024-01-15 10:06:00 | bob | video\_001 | video\_resume | 60 | Resumed after 5min break  
2024-01-15 10:10:00 | bob | video\_001 | video\_ended | 300 | Watched 240s more, completed

#### Visual Timeline:

Video (300s duration):  
0s ──────────► 60s [PAUSED] 60s ────────────────────► 300s  
  
Segments:  
▶️ play(0s) ━━━━━━━━━━━━━━━━━━━► ⏸️ pause(60s)  
└─────────── ✅ 60s watched ──────┘  
  
⏸️ pause(60s) ─ [5 min break] ─► ▶️ resume(60s)  
└────────── ❌ 0s (not watching) ──┘  
  
▶️ resume(60s) ━━━━━━━━━━━━━━━━► ✅ ended(300s)  
└─────────── ✅ 240s watched ─────┘  
  
Total: 60s + 240s = 300s

#### Calculated Output:

userId: bob  
videoId: video\_001  
videoDuration: 300s  
  
totalWatchTime: 300s # 60 + 240 = 300  
uniqueSecondsWatched: 300s  
watchPercentage: 100%  
completionPercentage: 100%  
  
sessionCount: 1  
pauseCount: 1 # Paused once  
completed: true  
  
engagementScore: 55.0 # Slightly lower due to pause  
engagementTier: "High"

**Business Insight:** User completed video with one pause (bathroom break?). Still high engagement.

### **Scenario 3: Browser Close (Lost Session)** ❌

**Description:** User starts watching but closes browser without pausing. This is the **most common data loss scenario**.

#### Raw Input Events:

timestamp | userId | videoId | eventName | currentTime | Action  
--------------------|--------|------------|----------------|-------------|----------------------------------  
2024-01-15 10:00:00 | lisa | video\_001 | video\_play | 0 | Started playing from beginning  
2024-01-15 10:00:45 | lisa | video\_001 | video\_pause | 45 | Watched 45s, then paused  
2024-01-15 10:00:50 | lisa | video\_001 | video\_resume | 45 | Resumed after 5s  
[Browser closed - no more events - lost tracking of remaining watch time]

#### Visual Timeline:

Video (300s duration):  
0s ────► 45s [PAUSE] 45s ━━━━━━━━━━━━► [~unknown position, browser closed]  
  
Segments:  
▶️ play(0s) ━━━━━━━━━━━━━━━━━━━► ⏸️ pause(45s)  
└─────────── ✅ 45s watched ──────┘  
  
▶️ resume(45s) ━━━━━━━━━━━━━━━━► ❌ [browser close at ~75s?]  
└─────────── ❌ 0s counted ───────┘  
 (No closing event!)  
  
Total Counted: 45s  
Actually Watched: ~75s (estimated)  
Lost: ~30s

#### Calculated Output:

userId: lisa  
videoId: video\_001  
videoDuration: 300s  
  
totalWatchTime: 45s # Only first segment counted!  
uniqueSecondsWatched: 45s  
watchPercentage: 15% # Appears low  
completionPercentage: 15% # Based on last known position (45s)  
  
sessionCount: 1  
pauseCount: 1  
completed: false # Did not finish  
completionCount: 0  
  
engagementScore: 5.25 # Very low (lost data)  
engagementTier: "Minimal"

**Business Insight:** This looks like low engagement, but in reality, we **lost tracking data** when the user closed their browser. This is why closing events matter!

**What Actually Happened:** User probably watched for ~75 seconds but we can only count the first 45 seconds.

**How to Fix:** Implement heartbeat events (send position every 30s) or browser close detection.

### **Scenario 4: Skip Forward** ⏩

**Description:** User skips ahead in the video to find interesting content.

#### Raw Input Events:

timestamp | userId | videoId | eventName | currentTime | Action  
--------------------|--------|------------|----------------|-------------|----------------------------------  
2024-01-15 10:00:00 | tom | video\_002 | video\_play | 0 | Started playing from beginning  
2024-01-15 10:00:30 | tom | video\_002 | video\_pause | 30 | Watched 30s, then paused  
2024-01-15 10:00:32 | tom | video\_002 | video\_resume | 300 | Skipped forward 270s (4.5min)  
2024-01-15 10:01:32 | tom | video\_002 | video\_ended | 360 | Watched 60s more, completed

#### Visual Timeline:

Video (600s duration):  
0s ──► 30s [SKIP ⏩] 300s ──────────────► 360s  
  
Segments:  
▶️ play(0s) ━━━━━━━━━━━━━━━━━━━► ⏸️ pause(30s)  
└─────────── ✅ 30s watched ──────┘  
  
⏸️ pause(30s) ─ [instant] ─► ▶️ resume(300s)  
└── ⚠️ Forward jump detected: 270s ──┘  
 (User skipped 4.5 minutes)  
  
▶️ resume(300s) ━━━━━━━━━━━━━━━► ✅ ended(360s)  
└─────────── ✅ 60s watched ──────┘  
  
Total Watched: 30 + 60 = 90s  
Skipped Content: 270s not watched

#### Calculated Output:

userId: tom  
videoId: video\_002  
videoDuration: 600s  
  
totalWatchTime: 90s # Only watched segments: 30 + 60  
uniqueSecondsWatched: 90s # 0-30 and 300-360  
watchPercentage: 15% # 90/600 \* 100  
completionPercentage: 60% # Reached 360s / 600s = 60%  
  
sessionCount: 1  
maxPositionReached: 360s # Got to 360s mark  
completed: false # Didn't reach end (600s)  
  
forwardSkipCount: 1 # One skip detected  
jumpType: "forward"  
  
engagementScore: 51.5 # Medium (completed but skipped)  
engagementTier: "Medium"

**Business Insight:** User searched for specific content by skipping. Common behavior for tutorials or long-form content. High completion % but low watch %.

### **Scenario 5: Rewind / Skip Backward** ⏪

**Description:** User goes back to rewatch a section they missed or found interesting.

#### Raw Input Events:

timestamp | userId | videoId | eventName | currentTime | Action  
--------------------|--------|------------|----------------|-------------|----------------------------------  
2024-01-15 10:00:00 | peter | video\_001 | video\_play | 0 | Started playing from beginning  
2024-01-15 10:00:30 | peter | video\_001 | video\_pause | 30 | Watched 30s, then paused  
2024-01-15 10:00:35 | peter | video\_001 | video\_resume | 30 | Resumed after 5s  
2024-01-15 10:02:05 | peter | video\_001 | video\_pause | 120 | Watched 90s more (30→120)  
2024-01-15 10:02:10 | peter | video\_001 | video\_resume | 110 | Rewound 10s back to rewatch  
2024-01-15 10:02:20 | peter | video\_001 | video\_pause | 120 | Watched 10s again (110→120)

#### Visual Timeline:

Video (300s duration):  
0s ──────────► 30s [PAUSE] 30s ─────────────► 120s [REWIND ⏪] 110s ──► 120s  
  
Segments Watched:  
1️⃣ ▶️ play(0s) ━━━━━━━━━━━━━━━► ⏸️ pause(30s)  
 └────── ✅ 30s watched ──────┘  
  
2️⃣ ▶️ resume(30s) ━━━━━━━━━━━━► ⏸️ pause(120s)  
 └────── ✅ 90s watched ──────┘  
  
3️⃣ ▶️ resume(110s) ━━━━━━━━━━━► ⏸️ pause(120s)  
 └────── ✅ 10s watched ──────┘  
  
Total Watch Time: 30 + 90 + 10 = 130s  
Unique Seconds: 0-120s = 120s (without counting 110-120 twice)

#### Calculated Output:

userId: peter  
videoId: video\_001  
videoDuration: 300s  
  
totalWatchTime: 130s # Sum of all segments (includes replay)  
uniqueSecondsWatched: 120s # Without counting duplicates  
watchPercentage: 43.3% # 130/300 \* 100  
completionPercentage: 40% # 120/300 \* 100  
uniqueWatchPercentage: 40% # 120/300 \* 100  
  
sessionCount: 1  
maxPositionReached: 120s # Furthest point reached  
pauseCount: 3  
  
backwardSkipCount: 1 # One rewind detected  
replayBehavior: true  
  
engagementScore: 58.5 # High (rewatching = engaged)  
engagementTier: "High"

**Business Insight:** User rewound to rewatch content - indicates high engagement and interest in understanding the material. Common in educational videos.

**Important Note:** - totalWatchTime = 130s (counts the replay) - uniqueSecondsWatched = 120s (doesn’t count seconds twice)

### **Scenario 6: Multiple Sessions (Replay Video)** 🔁

**Description:** User watches video across multiple sessions on different days.

#### Raw Input Events:

**Session 1 (Day 1):**

timestamp | userId | sessionId | videoId | eventName | currentTime | Action  
--------------------|--------|------------|------------|--------------|-------------|----------------------------------  
2024-01-15 10:00:00 | sarah | session\_1 | video\_003 | video\_play | 0 | Started playing from beginning  
2024-01-15 10:01:00 | sarah | session\_1 | video\_003 | video\_pause | 60 | Watched 60s, stopped for the day

**Session 2 (Day 2):**

timestamp | userId | sessionId | videoId | eventName | currentTime | Action  
--------------------|--------|------------|------------|--------------|-------------|----------------------------------  
2024-01-16 14:00:00 | sarah | session\_2 | video\_003 | video\_play | 0 | Came back, started from beginning  
2024-01-16 14:03:00 | sarah | session\_2 | video\_003 | video\_ended | 180 | Watched full 180s, completed

#### Visual Timeline:

Video (180s duration = 3 minutes):  
  
Day 1 - Session 1:  
0s ──────────────────────────────────────────────► 60s [STOPPED]  
├──────────────────────────────────────────────┤  
│ ✅ Watched 60s (33%) │  
└──────────────────────────────────────────────┘  
  
Day 2 - Session 2:  
0s ══════════════════════════════════════════════════════════════► 180s ✅  
├────────────────────────────────────────────────────────────────┤  
│ ✅ Watched 180s (100%) │  
└────────────────────────────────────────────────────────────────┘  
  
Combined:  
Session 1: 60s watched  
Session 2: 180s watched  
Total: 240s  
Unique: 180s (0-180s, counting each second only once)

#### Calculated Output:

userId: sarah  
videoId: video\_003  
videoDuration: 180s  
  
# Aggregated across BOTH sessions  
totalWatchTime: 240s # 60 + 180 = 240s total  
uniqueSecondsWatched: 180s # Full video coverage  
watchPercentage: 133% # 240/180 \* 100 (over 100%!)  
completionPercentage: 100% # Reached end  
uniqueWatchPercentage: 100% # Covered all seconds  
  
sessionCount: 2 # Two separate sessions  
completionCount: 1 # Completed once (in session 2)  
isReplay: true # Watched multiple times  
isCompletedAtLeastOnce: true  
  
# Temporal tracking  
firstWatchDate: 2024-01-15 # First interaction  
lastWatchDate: 2024-01-16 # Most recent  
avgWatchTimePerSession: 120s # 240/2 = 120s per session  
  
engagementScore: 114.0 # Very high (multiple sessions + completion)  
engagementTier: "High"

**Business Insight:** User came back the next day to finish the video. Shows strong interest. Common in: - Training materials (watch partially, apply, come back) - Complex topics (need multiple viewings) - Reference videos (watch when needed)

### **Scenario 7: Multi-Video Session (Binge Watching)** 📺

**Description:** User watches multiple videos in one session.

#### Raw Input Events:

timestamp | userId | sessionId | videoId | eventName | currentTime | Action  
--------------------|--------|------------|------------|--------------|-------------|----------------------------------  
2024-01-15 10:00:00 | max | session\_1 | video\_001 | video\_play | 0 | Started video\_001  
2024-01-15 10:05:00 | max | session\_1 | video\_001 | video\_ended | 300 | Watched 300s, completed video\_001  
2024-01-15 10:05:10 | max | session\_1 | video\_002 | video\_play | 0 | Started video\_002 (10s later)  
2024-01-15 10:10:10 | max | session\_1 | video\_002 | video\_ended | 300 | Watched 300s, completed video\_002  
2024-01-15 10:10:20 | max | session\_1 | video\_003 | video\_play | 0 | Started video\_003 (10s later)  
2024-01-15 10:13:20 | max | session\_1 | video\_003 | video\_ended | 180 | Watched 180s, completed video\_003

#### How Data is Aggregated:

System creates THREE separate output rows (one per User+Video):  
  
Row 1: max + video\_001  
 totalWatchTime: 300s  
 watchPercentage: 100%  
 sessionCount: 1  
  
Row 2: max + video\_002  
 totalWatchTime: 300s  
 watchPercentage: 100%  
 sessionCount: 1  
  
Row 3: max + video\_003  
 totalWatchTime: 180s  
 watchPercentage: 100%  
 sessionCount: 1

#### Calculated Output (Example for Video 001):

userId: max  
videoId: video\_001  
videoDuration: 300s  
  
totalWatchTime: 300s  
watchPercentage: 100%  
completionPercentage: 100%  
sessionCount: 1  
completed: true  
  
engagementScore: 60.0  
engagementTier: "High"

**Business Insight:** User is highly engaged and consuming multiple pieces of content. Perfect for: - Analyzing user journey (which videos watched in sequence) - Course progression tracking - Content recommendations (video\_002 follows video\_001)

**Note:** Each user+video combination gets its own row. To analyze the full session, query all videos with sessionId = "session\_1".

### **Scenario 8: Abandoned Early (Low Engagement)** 😞

**Description:** User starts video but loses interest quickly.

#### Raw Input Events:

timestamp | userId | videoId | eventName | currentTime | Action  
--------------------|--------|------------|----------------|-------------|----------------------------------  
2024-01-15 10:00:00 | john | video\_001 | video\_play | 0 | Started playing from beginning  
2024-01-15 10:00:08 | john | video\_001 | video\_pause | 8 | Watched only 8s, abandoned

#### Visual Timeline:

Video (300s duration):  
0s ──────► 8s [ABANDONED]  
├────────┤ (292s unwatched)  
│ ✅ 8s │  
└────────┘  
  
User watched only 8 seconds (2.7%) then left.

#### Calculated Output:

userId: john  
videoId: video\_001  
videoDuration: 300s  
  
totalWatchTime: 8s # Minimal engagement  
uniqueSecondsWatched: 8s  
watchPercentage: 2.7% # Very low  
completionPercentage: 2.7%  
  
sessionCount: 1  
maxPositionReached: 8s  
completed: false  
completionCount: 0  
  
engagementScore: 0.13 # Very low  
engagementTier: "Minimal"  
dataQualityFlag: "very\_short\_watch"  
  
dropoffPoint: 8s # Early abandonment

**Business Insight:** User abandoned video within seconds. Possible reasons: - Video content doesn’t match expectation - Poor video quality/audio - Wrong video clicked - Distraction

**Action Items:** Analyze drop-off patterns at the beginning to improve: - Video titles/thumbnails (set correct expectations) - Video intro (hook users faster) - Technical quality

### **Scenario 9: Multiple Pauses & Complex Navigation** 🎛️

**Description:** User has complicated viewing pattern with many pauses, skips, and rewinds.

#### Raw Input Events:

timestamp | userId | videoId | eventName | currentTime | Action  
--------------------|--------|------------|----------------|-------------|----------------------------------  
2024-01-15 10:00:00 | alex | video\_002 | video\_play | 0 | Started playing from beginning  
2024-01-15 10:01:00 | alex | video\_002 | video\_pause | 60 | Watched 60s, paused  
2024-01-15 10:01:05 | alex | video\_002 | video\_resume | 60 | Resumed after 5s  
2024-01-15 10:03:05 | alex | video\_002 | video\_pause | 180 | Watched 120s more (60→180)  
2024-01-15 10:03:10 | alex | video\_002 | video\_resume | 300 | Skipped forward 120s (180→300)  
2024-01-15 10:04:10 | alex | video\_002 | video\_pause | 360 | Watched 60s (300→360)  
2024-01-15 10:04:15 | alex | video\_002 | video\_resume | 200 | Rewound 160s back (360→200)  
2024-01-15 10:11:55 | alex | video\_002 | video\_ended | 600 | Watched 400s (200→600), completed

#### Visual Timeline:

Video (600s = 10 min duration):  
  
0s ──► 60s [P] 60s ──────► 180s [P] 300s ──► 360s [P] 200s ───────────────► 600s  
 60s 120s [skip] 60s [rewind] 400s  
  
Segments:  
1️⃣ 0→60: 60s ✅  
2️⃣ 60→180: 120s ✅  
3️⃣ 300→360: 60s ✅  
4️⃣ 200→600: 400s ✅  
  
Total: 640s watched  
Unique: 0-180, 200-600 = 580s unique (0-180=180s, 200-600=400s)

#### Calculated Output:

userId: alex  
videoId: video\_002  
videoDuration: 600s  
  
totalWatchTime: 640s # Includes replays  
uniqueSecondsWatched: 580s # Without duplicates  
watchPercentage: 106.7% # Over 100% due to replays  
completionPercentage: 100% # Reached the end  
uniqueWatchPercentage: 96.7% # Almost all unique seconds  
  
sessionCount: 1  
maxPositionReached: 600s  
completed: true  
  
pauseCount: 4 # Many pauses  
forwardSkipCount: 1 # Skip forward (180→300)  
backwardSkipCount: 1 # Skip back (360→200)  
  
avgPausesPerSession: 4.0  
navigationComplexity: "high" # Custom flag  
  
engagementScore: 65.7  
engagementTier: "High" # High despite complexity

**Business Insight:** User is highly engaged but navigating actively. Possible reasons: - Technical/educational content (pausing to practice) - Note-taking behavior - Looking for specific information - Following along with hands-on tutorial

**Action Items:** This is actually **positive engagement** - don’t penalize it!

### **Scenario 10: Skip to End (Completion Gaming)** 🎮

**Description:** User skips directly to end to mark video as “watched” without actually watching.

#### Raw Input Events:

timestamp | userId | videoId | eventName | currentTime | Action  
--------------------|--------|------------|----------------|-------------|----------------------------------  
2024-01-15 10:00:00 | mike | video\_001 | video\_play | 0 | Started playing from beginning  
2024-01-15 10:00:05 | mike | video\_001 | video\_pause | 5 | Watched only 5s, paused  
2024-01-15 10:00:06 | mike | video\_001 | video\_resume | 295 | Skipped forward 290s to near end  
2024-01-15 10:00:11 | mike | video\_001 | video\_ended | 300 | Watched last 5s, gaming system

#### Visual Timeline:

Video (300s duration):  
0s ─► 5s [SKIP ⏩ 290s] 295s ───► 300s  
 5s 5s  
  
Watched: Only 10 seconds out of 300  
Skipped: 290 seconds (97%)  
Completed: YES (technically)

#### Calculated Output:

userId: mike  
videoId: video\_001  
videoDuration: 300s  
  
totalWatchTime: 10s # Very low  
uniqueSecondsWatched: 10s  
watchPercentage: 3.3% # Almost nothing  
completionPercentage: 100% # But marked complete!  
uniqueWatchPercentage: 3.3%  
  
sessionCount: 1  
completed: true # Video ended  
completionCount: 1  
  
forwardSkipCount: 1  
skipAmount: 290s  
  
engagementScore: 50.2 # Medium due to completion bonus  
engagementTier: "Medium"  
dataQualityFlag: "completed\_without\_sufficient\_watch" # ⚠️ RED FLAG

**Business Insight:** User “gamed” the completion metric by skipping to the end. This is **fake engagement**.

**Detection:** Flag raised because: - completed = true BUT watchPercentage < 75%

**Action Items:** - Filter out these records from “completion rate” KPIs - Require minimum watch percentage for completion credit - Analyze if certification/credit is being gamed

## 📈 Key Metrics Explained

### Metric Definitions & Formulas

| **Metric** | **Formula** | **What It Measures** | **Good Value** |
| --- | --- | --- | --- |
| **totalWatchTime** | Sum of all valid watch segments | Total time spent watching (includes replays) | Higher = more engaged |
| **uniqueSecondsWatched** | Count of unique seconds covered | Actual video coverage without duplicates | Higher = more content seen |
| **watchPercentage** | (totalWatchTime / videoDuration) × 100 | How much time invested (can exceed 100%) | >75% = engaged |
| **completionPercentage** | (maxPositionReached / videoDuration) × 100 | How far into video user got | 100% = finished |
| **maxPositionReached** | MAX(currentTime) across all events | Furthest point in video | = videoDuration is best |
| **sessionCount** | COUNT(DISTINCT sessionId) | Number of viewing sessions | >1 = replay behavior |
| **engagementScore** | (watchTime/60) + (completions × 50) + (sessions × 5) - (skips × 2) | Overall engagement quality | >50 = good |

### Example Comparison

┌──────────────────┬──────────┬──────────┬──────────┐  
│ Metric │ User A │ User B │ User C │  
│ │ (Best) │ (Good) │ (Gaming) │  
├──────────────────┼──────────┼──────────┼──────────┤  
│ totalWatchTime │ 300s │ 280s │ 10s │  
│ uniqueSeconds │ 300s │ 280s │ 10s │  
│ watchPct │ 100% │ 93% │ 3% │  
│ completionPct │ 100% │ 100% │ 100% │  
│ completed │ ✅ │ ✅ │ ✅ │  
│ engagementScore │ 60 │ 59 │ 50 │  
│ dataQualityFlag │ ok │ ok │ ⚠️ FLAG │  
└──────────────────┴──────────┴──────────┴──────────┘  
  
User A: Perfect viewing  
User B: Watched most, then skipped to end (acceptable)  
User C: Gaming the system (skipped almost everything)

## 🔍 Data Quality & Edge Cases

### Quality Flags

The system automatically detects problematic data:

| **Flag** | **Condition** | **What It Means** |
| --- | --- | --- |
| ok | Normal viewing pattern | Data looks good ✅ |
| excessive\_watch\_time | watchPercentage > 120% | Possible data quality issue or heavy replay |
| very\_short\_watch | totalWatchTime < 5s | User abandoned immediately |
| completed\_without\_sufficient\_watch | completed = true AND watchPercentage < 75% | Gaming/skipping to end |
| negative\_watch\_time | Calculated time < 0 | Data corruption |

### Edge Cases Handled

#### 1. Session Timeout

User leaves video paused for hours:  
  
▶️ play(0s) ━━━━━► ⏸️ pause(30s)  
⏸️ pause(30s) ─[3 hours later]─► ▶️ resume(30s)  
  
Solution: Only count actual watch time (30-0 = 30s for first segment)  
Time paused doesn't count as engagement.

#### 2. Out-of-Order Events

Events arrive out of sequence:  
  
Received: pause(30s), play(0s), ended(300s), resume(30s)  
Sorted: play(0s), pause(30s), resume(30s), ended(300s)  
  
Solution: Events sorted by timestamp before processing.

#### 3. Duplicate Events

User double-clicks pause button:  
  
▶️ play(0s)  
⏸️ pause(30s)  
⏸️ pause(30s) [DUPLICATE]  
  
Solution: Deduplication logic filters repeated events.

#### 4. Invalid Jumps

User shows impossible behavior:  
  
▶️ resume(100s) ━━━━━► ⏸️ pause(5000s)  
  
currentTime jumped 4900 seconds but only 10 seconds real time passed.  
  
Solution: Segment rejected as invalid. Only count if:  
- timeDelta < 7200s (max 2 hours per segment)  
- timeDelta ≤ timestampDelta + 5s (can't watch faster than real-time)

#### 5. Negative Watch Time

Events out of logical order:  
  
⏸️ pause(100s) appears before ▶️ resume(100s)  
  
Would result in: 100 - 100 = 0s or negative  
  
Solution: Validate prevEvent type. Only count if:  
- prevEvent = play/resume  
- currentEvent = pause/ended

## 🎯 Business Use Cases

### 1. Content Performance Dashboard

**Question:** Which videos perform best?

**Query Approach:**

SELECT  
 videoId,  
 videoTitle,  
 COUNT(DISTINCT userId) as uniqueViewers,  
 AVG(watchPercentage) as avgWatchPct,  
 AVG(completionPercentage) as avgCompletionPct,  
 SUM(completionCount) as totalCompletions,  
 AVG(engagementScore) as avgEngagement  
FROM aggregated\_user\_video\_engagement  
WHERE dataQualityFlag = 'ok' -- Exclude problematic data  
GROUP BY videoId, videoTitle  
ORDER BY avgEngagement DESC

**Insights:** - High avgEngagement + High avgWatchPct = Great video ⭐ - High completionPct + Low watchPct = Users skip to end (might need better content) - Low both = Poor content or wrong audience

### 2. User Segmentation

**Question:** Who are my power users vs casual viewers?

**Segments:**

Power Users:  
 - engagementTier = "High"  
 - sessionCount > 3  
 - avgWatchPercentage > 75%  
 Action: Target for advanced content, beta features  
  
Engaged Learners:  
 - backwardSkipCount > 0 (rewatching)  
 - pauseCount > 2 (taking notes)  
 - completionCount > 0  
 Action: Offer certifications, downloadable resources  
  
Casual Browsers:  
 - sessionCount = 1  
 - watchPercentage < 50%  
 - No completions  
 Action: Better recommendations, shorter content  
  
At Risk:  
 - avgWatchPercentage < 15%  
 - Multiple videos with "very\_short\_watch"  
 Action: Survey for feedback, improve onboarding

### 3. Drop-off Analysis

**Question:** Where do users stop watching?

**Approach:**

-- Find common drop-off points  
SELECT  
 videoId,  
 FLOOR(maxPositionReached / 30) \* 30 as dropOffBucket,  
 COUNT(\*) as userCount  
FROM aggregated\_user\_video\_engagement  
WHERE completionCount = 0 -- Users who didn't finish  
GROUP BY videoId, dropOffBucket  
ORDER BY videoId, dropOffBucket

**Visualization:**

Video\_001 Drop-off Points:  
  
0-30s: ████████████ (120 users) - Intro too long?  
30-60s: ████████ (80 users)  
60-90s: ████ (40 users)  
90-120s: ██ (20 users)  
...  
240-270s: █████████████████ (170 users) - Problem section!

**Action:** Improve the 240-270s section (confusing content, technical issue, etc.)

### 4. Course Completion Tracking

**Question:** Are users finishing our training series?

**Multi-Video Analysis:**

-- Users who completed all videos in a course  
WITH course\_videos AS (  
 SELECT videoId FROM videos WHERE courseId = 'COURSE\_101'  
),  
user\_completions AS (  
 SELECT  
 userId,  
 COUNT(DISTINCT videoId) as videosCompleted  
 FROM aggregated\_user\_video\_engagement  
 WHERE  
 videoId IN (SELECT videoId FROM course\_videos)  
 AND isCompletedAtLeastOnce = true  
 AND dataQualityFlag = 'ok'  
 GROUP BY userId  
)  
SELECT  
 videosCompleted,  
 COUNT(userId) as userCount,  
 ROUND(COUNT(userId) \* 100.0 / SUM(COUNT(userId)) OVER(), 2) as percentage  
FROM user\_completions  
GROUP BY videosCompleted  
ORDER BY videosCompleted DESC

### 5. Engagement Trends Over Time

**Question:** Is engagement improving?

**Temporal Analysis:**

SELECT  
 DATE\_TRUNC('week', firstWatchDate) as week,  
 AVG(watchPercentage) as avgWatchPct,  
 AVG(engagementScore) as avgEngagement,  
 COUNT(DISTINCT userId) as activeUsers  
FROM aggregated\_user\_video\_engagement  
WHERE firstWatchDate >= '2024-01-01'  
GROUP BY week  
ORDER BY week

**Trend Chart:**

Week | Avg Watch % | Engagement | Active Users  
----------|-------------|------------|-------------  
2024-W01 | 45% | 32.5 | 1,250  
2024-W02 | 48% | 35.2 | 1,420  
2024-W03 | 52% | 38.7 | 1,680 ← Improving!  
2024-W04 | 55% | 42.1 | 1,890

## 🛠️ Implementation Details

### How the Aggregation Works

#### Step-by-Step Process:

┌─────────────────────────────────────────────────────────────────┐  
│ Step 1: Load Raw Events │  
│ Filter: Valid events only (play/pause/resume/ended) │  
│ Non-null userId, videoId, currentTime │  
└─────────────────────────────────────────────────────────────────┘  
 ↓  
┌─────────────────────────────────────────────────────────────────┐  
│ Step 2: Calculate Watch Segments │  
│ - Sort events by timestamp │  
│ - Use LAG to get previous event │  
│ - Calculate timeDelta = currentTime - prevTime │  
│ - Validate segment (start event + end event) │  
└─────────────────────────────────────────────────────────────────┘  
 ↓  
┌─────────────────────────────────────────────────────────────────┐  
│ Step 3: Calculate Unique Seconds (Optional) │  
│ - Merge overlapping intervals │  
│ - Count unique seconds watched │  
└─────────────────────────────────────────────────────────────────┘  
 ↓  
┌─────────────────────────────────────────────────────────────────┐  
│ Step 4: Aggregate by Session │  
│ - Group by userId + videoId + sessionId │  
│ - SUM(watchedSeconds) as watchTime │  
│ - MAX(currentTime) as maxPosition │  
│ - COUNT pauses, skips, etc. │  
└─────────────────────────────────────────────────────────────────┘  
 ↓  
┌─────────────────────────────────────────────────────────────────┐  
│ Step 5: Aggregate by User+Video │  
│ - Group by userId + videoId (across all sessions) │  
│ - SUM watchTime from all sessions │  
│ - COUNT sessions │  
│ - Calculate averages │  
└─────────────────────────────────────────────────────────────────┘  
 ↓  
┌─────────────────────────────────────────────────────────────────┐  
│ Step 6: Enrich with Metadata │  
│ - Join video metadata (duration, title) │  
│ - Calculate percentages │  
│ - Calculate engagement score │  
│ - Apply data quality flags │  
└─────────────────────────────────────────────────────────────────┘  
 ↓  
┌─────────────────────────────────────────────────────────────────┐  
│ Output: One row per User+Video │  
└─────────────────────────────────────────────────────────────────┘

### Code Reference

Main processing logic is in <databricks_video_aggregation.py>:

* **Line 78-139:** calculate\_watch\_segments() - Identifies valid watch segments
* **Line 141-179:** calculate\_unique\_seconds\_watched() - Counts unique seconds
* **Line 250-282:** aggregate\_sessions() - Session-level aggregation
* **Line 284-346:** aggregate\_user\_video() - Final user+video aggregation
* **Line 348-418:** enrich\_with\_video\_metadata() - Adds metadata and calculates scores

## 🧪 Testing Scenarios

### Test Data Examples

Use <databricks_example_notebook.py> to generate test data and validate results.

**Key Test Cases:**

| **Test Case** | **Expected Result** | **Validates** |
| --- | --- | --- |
| Perfect viewing (play → end) | 100% watch, 100% completion | Basic happy path |
| Pause + resume | Correct time excluding pause | Segment calculation |
| Browser close (no end event) | Only counts completed segments | Conservative approach |
| Skip forward | Lower watch %, higher completion % | Jump detection |
| Rewind | Higher watch %, correct unique seconds | Replay handling |
| Multiple sessions | Aggregation across sessions | Session grouping |
| Skip to end | Completion flag, data quality alert | Gaming detection |

### Validation Queries

-- Test 1: Verify no negative watch time  
SELECT \* FROM aggregated\_user\_video\_engagement  
WHERE totalWatchTime < 0;  
-- Expected: 0 rows  
  
-- Test 2: Watch percentage should be reasonable  
SELECT \* FROM aggregated\_user\_video\_engagement  
WHERE watchPercentage > 200; -- Over 200% is suspicious  
-- Expected: Few or no rows  
  
-- Test 3: Completion requires sufficient watch time  
SELECT \* FROM aggregated\_user\_video\_engagement  
WHERE completed = true  
 AND watchPercentage < 50  
 AND dataQualityFlag = 'ok';  
-- Expected: 0 rows (should all be flagged)  
  
-- Test 4: Max position can't exceed video duration  
SELECT \* FROM aggregated\_user\_video\_engagement  
WHERE maxPositionReached > videoDuration;  
-- Expected: 0 rows

## 📊 Sample Output Schema

### Complete Field Reference

# Identifiers  
userId: string # User identifier  
videoId: string # Video identifier  
videoTitle: string # Video name (from metadata)  
videoDuration: double # Video length in seconds  
  
# Watch Time Metrics  
totalWatchTime: double # Total seconds watched (includes replays)  
totalUniqueSecondsWatched: double # Unique seconds (no double-counting)  
watchPercentage: double # (totalWatchTime / duration) \* 100  
completionPercentage: double # (maxPosition / duration) \* 100  
uniqueWatchPercentage: double # (uniqueSeconds / duration) \* 100  
  
# Position Tracking  
maxPositionReached: double # Furthest point in video (seconds)  
  
# Session Metrics  
sessionCount: long # Number of viewing sessions  
avgWatchTimePerSession: double # Average watch time per session  
avgSessionDuration: double # Average session length (real time)  
firstWatchDate: timestamp # First interaction  
lastWatchDate: timestamp # Most recent interaction  
  
# Completion Tracking  
isCompletedAtLeastOnce: boolean # Ever reached the end  
completionCount: long # How many times completed  
completed: boolean # Completed in any session  
  
# Interaction Metrics  
pauseCount: long # Total pauses across all sessions  
avgPausesPerSession: double # Average pauses per session  
forwardSkipCount: long # Number of forward skips  
backwardSkipCount: long # Number of rewinds/replays  
  
# Engagement Scoring  
engagementScore: double # Calculated engagement score  
engagementTier: string # High / Medium / Low / Minimal  
isReplay: boolean # Watched in multiple sessions  
  
# Data Quality  
dataQualityFlag: string # ok / excessive\_watch\_time / very\_short\_watch / completed\_without\_sufficient\_watch  
processedAt: timestamp # When this row was calculated

## ⚠️ Known Limitations & Solutions

### Limitation 1: Browser Close Detection

**Problem:** When users close browser without pausing, we lose tracking.

**Example:** User watches for 60s, closes browser. - We only counted segments up to last event - Lost tracking of final 60s

**Solutions:** 1. **Heartbeat Events** (Recommended) - Send position update every 30 seconds automatically - Code: setInterval(() => trackEvent('heartbeat', currentTime), 30000)

1. **beforeunload Handler**

* window.addEventListener('beforeunload', () => {  
   trackEvent('video\_pause', video.currentTime);  
  });

1. **Accept Conservative Estimates**
   * Understand ~10-20% of watch time may be lost
   * Focus on trends rather than absolute values

### Limitation 2: Multi-Device Sessions

**Problem:** Same user on multiple devices shows as different sessions.

**Example:** User starts video on phone, continues on laptop. - Shows as 2 incomplete sessions - Can’t track cross-device journey

**Solution:** Implement user authentication and device fingerprinting.

### Limitation 3: Unique Seconds Performance

**Problem:** Calculating unique seconds is memory-intensive for very long videos (>2 hours).

**Impact:** May slow down processing or cause memory issues.

**Solutions:** 1. Use efficient interval merging (already implemented in calculate\_unique\_seconds\_efficient()) 2. Disable unique seconds calculation for videos >2 hours 3. Calculate unique seconds offline as a secondary job

### Limitation 4: Livestream vs VOD

**Problem:** Livestreams have dynamic duration.

**Current Behavior:** System assumes fixed video duration.

**Solution:** For livestreams, use different logic: - Track “time spent watching” instead of “percentage watched” - Don’t calculate completion percentage - Add isLivestream flag to video metadata

## 🚀 Next Steps & Improvements

### Phase 1: Quick Wins (Week 1-2)

* ✅ Implement heartbeat events (30s intervals)
* ✅ Add browser close handler
* ✅ Setup data quality monitoring dashboard

### Phase 2: Enhanced Tracking (Week 3-4)

* ⬜ Add playback speed tracking (2x speed viewing)
* ⬜ Track fullscreen vs embedded viewing
* ⬜ Add video quality/buffering events
* ⬜ Track mobile vs desktop viewing

### Phase 3: Advanced Analytics (Week 5-8)

* ⬜ Heatmap visualization (which sections rewatched most)
* ⬜ A/B testing framework
* ⬜ Predictive analytics (likelihood to complete)
* ⬜ Cohort analysis (Day 1 vs Day 7 vs Day 30)

### Phase 4: ML/AI Integration (Week 9-12)

* ⬜ Content recommendation engine
* ⬜ Automatic video tagging based on engagement
* ⬜ Anomaly detection for data quality
* ⬜ Churn prediction

## 📚 Additional Resources

### Documentation Files

* [**README.md**](README.md) - Project overview and quick start
* [**QUICK\_REFERENCE\_CARD.md**](QUICK_REFERENCE_CARD.md) - Quick reference for developers
* [**VISUAL\_GUIDE\_CLOSING\_EVENTS.md**](VISUAL_GUIDE_CLOSING_EVENTS.md) - Deep dive on why closing events matter
* [**databricks\_video\_aggregation.py**](databricks_video_aggregation.py) - Main implementation code
* [**databricks\_example\_notebook.py**](databricks_example_notebook.py) - Example notebook with test data

### Support & Feedback

For questions or issues: 1. Check this guide first 2. Review the code comments in the Python scripts 3. Test with sample data from the example notebook 4. Consult with the data engineering team

## ✅ Summary Checklist

### For Business Analysts:

* Understand what each metric means (totalWatchTime vs uniqueSeconds vs completionPercentage)
* Know which metrics to use for different business questions
* Recognize data quality flags and filter them appropriately
* Understand scenario patterns (rewatching = good, skipping to end = gaming)

### For Product Owners:

* Understand the limitation of browser close tracking
* Know the difference between actual engagement vs tracked engagement
* Plan for implementing heartbeat events
* Define acceptance criteria for “video completion”

### For Developers:

* Understand the event pair concept
* Know how watch segments are calculated
* Implement frontend tracking correctly (play/pause/resume/end events)
* Add error handling and validation

### For Testers:

* Test all 10 scenarios in this guide
* Validate output matches expected results
* Check data quality flags are triggered correctly
* Test edge cases (browser close, invalid jumps, etc.)

## 🎓 Glossary

| **Term** | **Definition** |
| --- | --- |
| **Event Pair** | A start event (play/resume) + end event (pause/ended) that forms a valid watch segment |
| **Watch Segment** | Period of time where user actively watched video, bounded by event pair |
| **Total Watch Time** | Sum of all watch segments, including replays |
| **Unique Seconds** | Count of video seconds watched at least once, without double-counting replays |
| **Max Position** | Furthest point reached in video (highest currentTime value) |
| **Completion %** | How far into video user got (maxPosition / duration) |
| **Watch %** | How much time invested (totalWatchTime / duration), can exceed 100% |
| **Session** | Single viewing instance (from browser open to close) |
| **Replay** | Watching same video across multiple sessions |
| **Forward Skip** | Jumping ahead in video timeline |
| **Backward Skip** | Rewinding to earlier point |
| **Engagement Score** | Weighted metric combining watch time, completions, and interactions |
| **Data Quality Flag** | Automated indicator of potentially problematic data |

**Document Version:** 1.0 **Last Updated:** 2024-01-15 **Maintained By:** Data Analytics Team

*Made with 🎬 for better video analytics*