Table of Contents

# 🚀 Getting Started with Video Analytics Aggregation

## Welcome!

This guide will help you get up and running with the video analytics aggregation solution in **under 30 minutes**.

## 📋 Prerequisites

Before you start, make sure you have:

1. ✅ Access to Databricks workspace
2. ✅ Raw video events table with these columns:
   * timestamp (TimestampType)
   * userId (StringType)
   * sessionId (StringType)
   * videoId (StringType)
   * eventName (StringType): values like “video\_play”, “video\_pause”, “video\_resume”, “video\_ended”
   * currentTime (DoubleType): position in video in seconds
3. ✅ (Optional) Video metadata table with:
   * videoId (StringType)
   * duration (DoubleType)
   * title (StringType)

## 🎯 What You’ll Get

After following this guide, you’ll have:

* ✅ Aggregated table with **one row per user+video**
* ✅ Metrics: watch time, completion %, engagement score
* ✅ Working example with test data
* ✅ Foundation for dashboards

## 📝 Step-by-Step Guide

### Step 1: Upload the Script (2 minutes)

1. Download databricks\_video\_aggregation.py from this package
2. In Databricks:
   * Go to **Workspace** → **Users** → **Your Email**
   * Create new folder: video\_analytics
   * Click **Upload** → Select databricks\_video\_aggregation.py

**Result:** Script is now available at /Workspace/Users/your-email@company.com/video\_analytics/databricks\_video\_aggregation

### Step 2: Create Test Notebook (5 minutes)

1. In Databricks, create a new **Python Notebook**
2. Name it: video\_analytics\_test
3. Copy-paste this code:

# Cell 1: Load the script  
%run /Workspace/Users/your-email@company.com/video\_analytics/databricks\_video\_aggregation  
  
# Cell 2: Import and setup  
from databricks\_video\_aggregation import VideoEngagementAggregator  
from pyspark.sql.functions import col  
  
# Configure with YOUR table names  
INPUT\_TABLE = "your\_raw\_events\_table" # ← CHANGE THIS  
OUTPUT\_TABLE = "aggregated\_user\_video\_engagement"  
VIDEO\_METADATA = None # Or "your\_video\_metadata\_table"  
  
# Initialize aggregator  
aggregator = VideoEngagementAggregator(  
 spark=spark,  
 input\_table=INPUT\_TABLE,  
 output\_table=OUTPUT\_TABLE,  
 video\_metadata\_table=VIDEO\_METADATA  
)  
  
# Cell 3: Test with sample data first (optional but recommended)  
# See databricks\_example\_notebook.py for complete sample data generation  
  
# Cell 4: Run aggregation  
result = aggregator.run\_aggregation(  
 calculate\_unique\_seconds=True,  
 use\_efficient\_method=True  
)  
  
# Cell 5: Preview results  
result.show(10)  
  
# Cell 6: Save results  
aggregator.save\_results(result, mode="overwrite")  
  
print("✅ Done! Check table:", OUTPUT\_TABLE)

1. Update INPUT\_TABLE with your actual table name
2. Run all cells

**Result:** You now have an aggregated table!

### Step 3: Validate Results (5 minutes)

Check if the aggregation worked correctly:

# Read the output table  
df = spark.table("aggregated\_user\_video\_engagement")  
  
# Show sample  
df.show(5, truncate=False)  
  
# Check row count  
print(f"Total user-video combinations: {df.count()}")  
  
# Check a specific user (replace with real user ID)  
df.filter(col("userId") == "your\_test\_user\_id").show(vertical=True)

**Validation Checklist:** - ✅ One row per user+video combination? - ✅ totalWatchTime looks reasonable? - ✅ watchPercentage between 0-100? - ✅ sessionCount >= 1?

### Step 4: Quick Analysis (5 minutes)

Run some quick queries to validate the data makes sense:

# Top 10 videos by watch time  
spark.sql("""  
 SELECT   
 videoId,  
 COUNT(DISTINCT userId) as uniqueViewers,  
 SUM(totalWatchTime) / 3600 as totalWatchHours,  
 AVG(watchPercentage) as avgWatchPercentage  
 FROM aggregated\_user\_video\_engagement  
 GROUP BY videoId  
 ORDER BY totalWatchHours DESC  
 LIMIT 10  
""").show()  
  
# Engagement distribution  
spark.sql("""  
 SELECT   
 engagementTier,  
 COUNT(\*) as count,  
 AVG(watchPercentage) as avgWatchPercentage  
 FROM aggregated\_user\_video\_engagement  
 GROUP BY engagementTier  
 ORDER BY   
 CASE engagementTier  
 WHEN 'High' THEN 1  
 WHEN 'Medium' THEN 2  
 WHEN 'Low' THEN 3  
 ELSE 4  
 END  
""").show()  
  
# Data quality check  
spark.sql("""  
 SELECT   
 dataQualityFlag,  
 COUNT(\*) as count  
 FROM aggregated\_user\_video\_engagement  
 GROUP BY dataQualityFlag  
""").show()

**What to look for:** - ✅ Multiple videos showing up in top 10? - ✅ Engagement distribution looks reasonable (not all in one tier)? - ✅ Most data quality flags are “ok”?

### Step 5: Schedule Daily Job (10 minutes)

Once validation looks good, schedule it to run daily:

1. In Databricks, go to **Workflows** → **Create Job**
2. Configure:
   * **Name**: Video Analytics Daily Aggregation
   * **Type**: Notebook
   * **Notebook path**: Your test notebook
   * **Cluster**: Choose existing cluster or create new
3. Set **Schedule**:
   * **Cron**: 0 2 \* \* \* (runs at 2 AM daily)
   * **Timezone**: Your timezone
4. Add **Parameters** (optional):

* {  
   "mode": "append",  
   "start\_date": "{{job.start\_date}}"  
  }

1. Click **Create**

**Result:** Script runs automatically every day!

## 🎨 Connect to BI Tool (Bonus - 10 minutes)

### For Tableau:

1. In Tableau, click **Connect** → **More…** → **Databricks**
2. Enter connection details:
   * Server: Your Databricks workspace URL
   * HTTP Path: Your cluster’s HTTP path
3. Select database and table: aggregated\_user\_video\_engagement
4. Start building visualizations!

### For Power BI:

1. Get Data → More → Databricks
2. Enter server and HTTP path
3. Select table: aggregated\_user\_video\_engagement
4. Load data

### Recommended First Dashboard:

**Top Metrics:** - Total unique viewers - Total watch hours - Average completion rate

**Charts:** 1. **Bar Chart**: Top 10 videos by unique viewers 2. **Line Chart**: Active users over time (by firstWatchDate) 3. **Pie Chart**: Engagement tier distribution 4. **Scatter Plot**: Watch % vs Completion % per video

## 🐛 Troubleshooting

### Issue: “Table not found”

**Solution:** Check table name spelling, ensure you have access

### Issue: “Column not found: currentTime”

**Solution:** Your raw events table might use different column names. Update the script’s column mappings:

# In the load\_raw\_events method, update column names  
df = df.withColumnRenamed("your\_time\_column", "currentTime")

### Issue: “Out of memory”

**Solution:** Process smaller date ranges:

# Process only last 7 days  
from datetime import datetime, timedelta  
start\_date = datetime.now() - timedelta(days=7)  
result = aggregator.run\_aggregation(start\_date=start\_date)

### Issue: Results look strange

**Solution:** Run data quality checks:

# Check for common issues  
spark.sql("""  
 SELECT   
 'Excessive watch time' as issue,  
 COUNT(\*) as count  
 FROM aggregated\_user\_video\_engagement  
 WHERE totalWatchTime > videoDuration \* 1.2  
   
 UNION ALL  
   
 SELECT   
 'Negative watch time' as issue,  
 COUNT(\*) as count  
 FROM aggregated\_user\_video\_engagement  
 WHERE totalWatchTime < 0  
""").show()

## 📚 Next Steps

Now that you have the basics running:

1. **Week 1**: Monitor data quality, fix any issues
2. **Week 2**: Build comprehensive dashboards
3. **Week 3**: Share insights with stakeholders
4. **Week 4**: Iterate based on feedback

### Advanced Topics (Later):

* **Incremental Processing**: Only process new data each day
* **Real-time Dashboards**: Use Structured Streaming
* **Predictive Analytics**: Predict which users will complete videos
* **A/B Testing**: Compare different video strategies

## 🆘 Need Help?

1. **Check documentation**:
   * README.md - Overview
   * quick\_reference\_guide.md - Detailed reference
   * executive\_summary.md - Business context
2. **Review example**:
   * databricks\_example\_notebook.py - Complete working example
3. **Common questions**:
   * How to handle missing events? → See session timeout logic
   * How to calculate unique seconds? → Set calculate\_unique\_seconds=True
   * How to optimize performance? → Use use\_efficient\_method=True

## ✅ Success Checklist

Before considering this “done”:

* Script runs without errors
* Output table has expected number of rows
* Sample user’s metrics look correct
* Top videos query shows reasonable results
* Data quality flags are mostly “ok”
* Scheduled job runs successfully
* Dashboard connected and showing data
* Team can access and understand results

## 🎉 Congratulations!

You’ve successfully set up video analytics aggregation! Your team can now:

* ✅ Understand which videos are most engaging
* ✅ See how users interact with content
* ✅ Make data-driven decisions about video strategy
* ✅ Track engagement over time

**Happy analyzing! 🚀**

*For questions or issues, refer to the other documentation files in this package or reach out to your data engineering team.*