# Automated Intelligence Architecture for Digital Education: A Technical Framework for Ovi English School

## 1. The Strategic Imperative of Automated Analytics in EdTech

The digital education landscape has evolved from a repository of static lectures into a dynamic, algorithmic ecosystem where content visibility is governed by engagement velocity and retention mechanics. For an educational entity such as "Ovi English School," the transition from manual, intuitive channel management to a robust, automated analytics infrastructure is not merely an operational upgrade—it is a fundamental maturity milestone. In an environment where the YouTube recommendation engine processes billions of signals daily, the ability to programmatically harvest, structure, and interrogate performance data provides the competitive advantage necessary to convert casual viewers into dedicated students.

The objective of this comprehensive research report is to delineate a precise, architectural blueprint for automating the collection and analysis of YouTube analytics data. This framework moves beyond the superficial "vanity metrics" of total views and likes, focusing instead on the deep-stream data availability provided by the Google Cloud Platform (GCP). By leveraging the YouTube Data API v3 and the YouTube Analytics API, Ovi English School can construct a bespoke intelligence pipeline capable of distinguishing between high-intent search traffic (critical for "evergreen" educational content) and passive algorithmic discovery.

This report addresses the specific technical challenges inherent in the 2024-2025 YouTube ecosystem, including the complex bifurcation of content into Long-form and Shorts (now extending up to three minutes), the strict management of API usage quotas, and the implementation of statistical anomaly detection to identify viral curriculum topics. The proposed solution architects a Python-based pipeline that authenticates via OAuth 2.0, minimizes resource consumption through strategic inventory fetching, and visualizes insights via a custom Streamlit dashboard, thereby transforming raw JSON responses into actionable pedagogical intelligence.

## 2. The YouTube API Topography: Selecting the Right Tools

The foundation of any robust automation strategy lies in the correct selection of Application Programming Interfaces (APIs). Google offers a constellation of APIs for YouTube, each architected for distinct operational scopes. A common failure mode in early-stage automation is the conflation of these services, leading to inefficient data retrieval and rapid quota exhaustion. For Ovi English School, the architecture relies on a symbiotic relationship between two primary interfaces: the **YouTube Data API v3** and the **YouTube Analytics API**.

### 2.1 The Operational Backbone: YouTube Data API v3

The YouTube Data API v3 serves as the administrative interface for the channel. It provides access to the public-facing metadata that defines the "inventory" of the school—the videos themselves.1 Its primary role in this architecture is transactional and inventory-based rather than analytical.

The capabilities of the Data API v3 are centered on "resources," which are JSON representations of YouTube entities such as Videos, Playlists, and Channels.2 For example, a video resource contains snippet data (title, description, tags, published date) and content details (duration, definition). While the Data API v3 does return some statistical information via the statistics part (view count, like count, comment count), these metrics are **cumulative snapshots**. They provide the total number of views a video has received since its upload but cannot answer temporal questions such as, "How many views did this video generate yesterday?" or "What was the view velocity in the first 24 hours?".3

Therefore, the Data API v3 is utilized in this framework strictly for **Inventory Management**: discovering new uploads, retrieving static metadata (titles, thumbnails), and classifying content duration. It is the mechanism by which the system understands *what* content exists, before asking *how* it performed.

### 2.2 The Intelligence Engine: YouTube Analytics API

The core of the automated reporting system is the YouTube Analytics API. This service allows for targeted, real-time queries against the channel's private performance database, mirroring the capabilities of the YouTube Studio dashboard but offering programmatic flexibility.4

Unlike the Data API, the Analytics API functions as a query engine. It does not return "resources" but rather "reports." The primary method, reports.query, accepts a complex set of parameters including start dates, end dates, metrics, dimensions, and filters.4 This API enables the extraction of granular data that is critical for educational assessment, such as:

* **Audience Retention:** Minute-by-minute viewing data to identify where students lose interest in a lesson.
* **Demographics:** Age and gender distributions to tailor curriculum complexity.6
* **Traffic Sources:** Distinguishing between students searching for specific grammar rules (Search) versus those browsing the homepage (Browse Features).

It is crucial to distinguish the Analytics API from the **YouTube Reporting API**. The Reporting API is designed for bulk data retrieval, generating large, asynchronous CSV files intended for enterprise-level warehousing and Multi-Channel Networks (MCNs).1 For a single channel operator like Ovi English School, the Reporting API introduces unnecessary architectural overhead (handling bulk file downloads and parsing). The Analytics API, with its synchronous JSON responses, provides the agility required for a custom Python dashboard.

### 2.3 Comparative Feature Matrix

To clarify the distinct roles of these APIs within the Ovi English School architecture, the following comparison highlights their capabilities regarding key data points.

| **Feature / Data Point** | **YouTube Data API v3** | **YouTube Analytics API** | **Architecture Role** |
| --- | --- | --- | --- |
| **Video Metadata** | Titles, Descriptions, Thumbnails, Duration | Not Available | **Inventory Source** |
| **View Counts** | Lifetime Total (Cumulative) | Daily / Periodic (Time-Series) | **Data API for Snapshot; Analytics for Trend** |
| **Demographics** | Not Available | Age Group, Gender, Geography | **Analytics Source** |
| **Watch Time** | Not Available | Estimated Minutes Watched, AVD | **Analytics Source** |
| **Traffic Sources** | Not Available | Search, Suggested, Browse, Shorts Feed | **Analytics Source** |
| **Quota Cost** | High (for Search), Low (for IDs) | Negligible / Separate Quota Pool | **Optimization Focus** |
| **Latency** | Near Real-Time | 2-4 Day Lag | **Scheduling Constraint** |

This bifurcation of duties ensures that the system uses the most efficient tool for each task: the Data API v3 builds the catalog of "lessons," and the Analytics API grades their performance.

## 3. Security and Access Control: The OAuth 2.0 Protocol

Automating the collection of private analytics data requires traversing Google's secure authentication layer. Unlike public data scraping, which might rely on a simple API Key, accessing a channel's financial and performance data mandates **OAuth 2.0** authentication.5 This protocol ensures that the Python automation script is authorized to act on behalf of the channel owner without exposing the owner's password.

### 3.1 The Mechanics of Offline Access

The primary challenge for an automated script—whether running on a cloud server or a local cron job—is the absence of a user to interact with the login prompt. Standard OAuth flows require a user to click "Allow" in a browser. To bypass this for subsequent runs, the system must utilize a **Refresh Token**.8

The authentication lifecycle proceeds in two distinct phases:

1. **The Initial Handshake (User Present):** The script initiates a flow using the google\_auth\_oauthlib library. It launches a local web server (or provides a URL) where the Ovi English School administrator logs in. Crucially, the script must request access\_type='offline' and include\_granted\_scopes='true'.10 This signals to Google's authorization server that the application requires a Refresh Token—a long-lived credential that persists after the user closes the browser.
2. **The Automated Exchange (User Absent):** For all future executions, the script loads the stored Refresh Token. When the short-lived Access Token (typically valid for 1 hour) expires, the google-api-python-client library automatically presents the Refresh Token to the token endpoint (https://oauth2.googleapis.com/token) to silently acquire a new Access Token.9 This mechanism allows the analytics pipeline to run indefinitely in the background.

### 3.2 Scope Configuration and Least Privilege

In cybersecurity, the principle of least privilege dictates that an application should only possess the permissions necessary for its function. For this analytics pipeline, specific scopes must be defined in the OAuth request to balance functionality with security.

* **https://www.googleapis.com/auth/yt-analytics.readonly**: This is the primary scope required to retrieve reports via the Analytics API. It grants read-only access to viewing data, traffic sources, and demographics.5
* **https://www.googleapis.com/auth/youtube.readonly**: This scope allows the Data API v3 to read public channel information, such as video titles and upload dates. It explicitly prevents the script from editing videos, deleting content, or managing comments, thereby mitigating the risk of accidental data loss.5
* **https://www.googleapis.com/auth/yt-analytics-monetary.readonly**: If Ovi English School is part of the YouTube Partner Program (YPP), this scope is required to access revenue data (AdSense metrics). If revenue tracking is not a priority, this scope should be omitted to reduce the security footprint.5

### 3.3 Implementation Best Practices

The credentials file (often named client\_secret.json) obtained from the Google Cloud Console contains the Client ID and Client Secret. These are the keys to the castle.

* **Environment Variables:** Ideally, these secrets should be loaded from environment variables rather than hardcoded in the script.
* **Token Persistence:** The resulting Access and Refresh tokens are typically serialized (pickled) into a file (e.g., token.pickle or token.json). This file must be treated with the same secrecy as a password. If the project code is stored in a version control system like Git, token.json must be added to .gitignore to prevent it from being pushed to a public repository.11

## 4. Resource Economy: Quota Management and Cost Optimization

One of the most critical, yet frequently overlooked, aspects of YouTube API automation is Quota Management. Google implements strict usage limits to ensure service stability, and exceeding these limits results in an immediate 403 Forbidden error, halting the analytics pipeline.13

### 4.1 The Daily Quota Budget

As of the current API posture, the default quota allocation for a Google Cloud Project with the YouTube Data API enabled is **10,000 units per day**.13 It is vital to understand that this limit is tied to the *project*, not the API key. If Ovi English School runs multiple scripts (e.g., one for comments and one for analytics) under the same project, they share this single pool of 10,000 units.15 The quota resets at midnight Pacific Time (PT).16

While quota extensions can be requested, the process involves a compliance audit of the application.13 Therefore, the most prudent engineering path is to architect the system to operate comfortably within the default limit.

### 4.2 The Cost of Operations

Quota consumption is not uniform; different API methods incur drastically different costs. Understanding this pricing model is the key to scalable automation.

* **Read Operation (Low Cost):** A standard list request, such as videos.list or playlistItems.list, costs **1 unit**.16 This is highly efficient.
* **Search Operation (High Cost):** A search.list request costs **100 units**.16 This is the "quota trap." Many developers instinctively use the search endpoint to find the latest videos on a channel. Doing so is functionally equivalent to paying 100 times the market rate for data.
* **Write Operation (Extreme Cost):** Uploading a video costs **1,600 units**. While not part of the analytics scope, it illustrates the heavy weight of write actions.16
* **Analytics Queries:** Crucially, the YouTube Analytics API reports.query method generally does not consume the Data API quota in the same manner. It operates on a separate, often higher, limit based on queries per user per day, making it relatively "cheap" for reporting purposes.14

### 4.3 The "Uploads Playlist" Optimization Strategy

To automate the inventory collection (finding all videos on the Ovi English School channel) without depleting the quota via the expensive search endpoint, the architecture must utilize the **Uploads Playlist** strategy.

Every YouTube channel possesses a hidden system playlist that contains every video uploaded to that channel. This playlist is accessible via the Data API, and accessing it costs only 1 unit per page of results (up to 50 videos).

**The Algorithm for Optimization:**

1. **Derive the Playlist ID:** The ID of the Uploads playlist is algorithmically related to the Channel ID. If the Channel ID is UCbqEsRhXQI5KWYVr8mSWWWw, the Uploads playlist ID is typically derived by replacing the second character 'C' with 'U', resulting in UUbqEsRhXQI5KWYVr8mSWWWw.3 Alternatively, this ID can be retrieved once via a channels.list call (cost: 1 unit) by inspecting the contentDetails.relatedPlaylists.uploads field.
2. **Fetch Inventory:** The script calls playlistItems.list using this derived ID. This returns the 50 most recent videos for a cost of 1 unit.
3. **Comparison:**
   * *Naive Approach:* search.list for top 50 videos = 100 units.
   * *Optimized Approach:* playlistItems.list for top 50 videos = 1 unit.
   * *Efficiency Gain:* **99% reduction in quota usage.**

By implementing this strategy, the Ovi English School automation can run every hour of every day without ever risking a quota limit exception, leaving ample buffer for other operations.

## 5. Inventory Acquisition and Content Classification

Before analytics can be gathered, the system must establish an accurate inventory of the channel's content. This involves not only listing the videos but classifying them correctly. In the modern YouTube ecosystem (2024-2025), the distinction between "Long-form" videos and "Shorts" is the most significant variable in performance analysis.

### 5.1 The Taxonomy of Content: Shorts vs. Long-form

YouTube Shorts—vertical videos under three minutes—operate on a fundamentally different algorithmic rail than traditional videos. They are discovered primarily through a dedicated feed rather than search or suggested referrals.20 Merging the analytics of Shorts (which typically have high view counts but low watch time) with Long-form lessons (lower views, higher watch time) creates "noisy" data that obscures the performance of both formats.

**The Classification Challenge:** The YouTube Data API v3 does not currently provide a boolean field like isShorts in the video resource snippet. The system must inference this classification based on available metadata.22

### 5.2 Algorithmic Classification Logic

To automate this tagging, the Python script must implement a heuristic classification engine based on the latest platform rules.

1. **Duration Filtering:** The primary determinant is duration. Historically, Shorts were limited to 60 seconds. However, as of late 2024/early 2025, YouTube has expanded the eligibility for Shorts to videos up to **3 minutes (180 seconds)**.21
   * *Implementation:* The script retrieves the contentDetails.duration field (e.g., PT2M15S). It parses this ISO 8601 string into total seconds. If duration\_seconds <= 180, the video is a *candidate* for Shorts classification.
2. **Aspect Ratio Check:** While the API does not explicitly state "Vertical," the classification can be reinforced by checking the fileDetails (if available to the content owner) or generally assuming that educational content under 3 minutes uploaded in the modern era is intended as Shorts.
3. **Traffic Source Validation:** The most definitive confirmation comes post-hoc from the Analytics API. If a video receives traffic from the insightTrafficSourceType dimension value SHORTS, it is definitively a Short.24
   * *Refinement Logic:* The system can initially flag videos ≤ 180s as POTENTIAL\_SHORT. On the next analytics run, if SHORTS traffic > 0, the status is updated to CONFIRMED\_SHORT.

This binary classification allows Ovi English School to generate segmented reports: "How is our Short-form vocabulary series performing compared to our Long-form grammar tutorials?" This insight is impossible if the data remains aggregated.

## 6. The Analytics Pipeline: Metrics, Dimensions, and Latency

With the inventory established and classified, the automation pipeline moves to the extraction of performance data. This phase is governed by the specific pedagogical goals of Ovi English School—moving beyond views to understanding *learning behavior*.

### 6.1 The "Learning" Metrics

For an education channel, specific metrics hold higher value than they might for an entertainment channel. The pipeline should be configured to request the following metrics via the reports.query method:

* **averageViewDuration & estimatedMinutesWatched:** These are the proxies for student engagement. A high Average View Duration (AVD) on a grammar lesson indicates the explanation was clear and the pacing was appropriate.26
* **subscribersGained:** Tracking this at the video level reveals which specific topics convert viewers into long-term students.26
* **relativeRetentionPerformance:** This metric compares the video's retention to all other YouTube videos of similar length. It is a benchmark of production quality.27

### 6.2 The Dimension of Intent: Traffic Sources

Understanding *how* a student arrived at a video is as important as *if* they watched it. The Analytics API dimension insightTrafficSourceType breaks down views into categories that reveal user intent 6:

* **YT\_SEARCH:** High search traffic indicates the video is solving a specific problem (e.g., "Present Perfect vs. Past Simple"). This content is "Evergreen"—it will continue to generate views for years without promotion. For Ovi English School, maximizing this source is a key stability strategy.
* **RELATED\_VIDEO (Suggested):** High suggested traffic means the algorithm is finding an audience for the content. This is often the driver of "viral" spikes.
* **SHORTS:** Traffic originating from the Shorts feed. This confirms the reach of micro-learning content.25
* **BROWSE:** Views from the homepage. This indicates a strong subscriber connection or high click-through rate (CTR) on thumbnails.

### 6.3 Operational Constraint: Data Latency

A critical engineering constraint of the YouTube Analytics API is **Data Latency**. Unlike the real-time counters visible in the YouTube Studio interface, the data available via the API is batch-processed and typically trails by **2 to 4 days**.29

* **The "Zero Data" Trap:** If the automation script attempts to fetch data for "Today" () or "Yesterday" (), the API will often return empty datasets or incomplete numbers.
* **Pipeline Configuration:** The script must be programmed with a "safe window." A robust configuration is to fetch data for the date  (four days ago). For example, if the script runs on Friday, it should request analytics for Monday. This ensures the data has settled and is accurate.
* **Implication for Real-Time:** The system cannot alert Ovi English School to a viral trend *as it happens* (within the hour). It is a retrospective analysis tool. For real-time monitoring, the manual Studio app remains necessary; the API is for deep, historical, and trend analysis.31

## 7. Data Engineering and Storage Architecture

Collecting data is futile without a structured storage mechanism that enables historical comparison. While simple scripts might append to a CSV file, a relational database is required to handle the complexity of time-series data, video metadata, and demographic segmentation. For a single-channel application, **SQLite** offers the perfect balance of power and portability.32

### 7.1 Schema Design for Educational Analytics

The database schema must be normalized to handle the one-to-many relationship between a video and its daily performance metrics. The following SQLite schema is recommended:

#### 7.1.1 Table: video\_inventory (Dimension Table)

This table acts as the central registry of all content.

* video\_id (TEXT PRIMARY KEY): The unique YouTube ID (e.g., dQw4w9WgXcQ).
* title (TEXT): The title of the lesson.
* published\_at (DATETIME): Normalized UTC timestamp.
* duration\_seconds (INTEGER): For filtering Short vs. Long.
* video\_type (TEXT): 'SHORT' or 'LONG\_FORM'.
* thumbnail\_url (TEXT): For display in the dashboard.

#### 7.1.2 Table: daily\_performance (Fact Table)

This table stores the time-series data. It grows linearly with time (Videos × Days).

* id (INTEGER PRIMARY KEY AUTOINCREMENT)
* video\_id (TEXT, Foreign Key): Links to video\_inventory.
* date (DATE): The specific day the metrics represent.
* views (INTEGER): Views generated on that specific date.
* watch\_time\_minutes (REAL).
* avg\_view\_duration (REAL).
* subscribers\_gained (INTEGER).
* **Constraint:** A unique constraint on (video\_id, date) prevents duplicate entries if the script runs multiple times.34

#### 7.1.3 Table: retention\_curves (Complex Data)

Storing the audience retention curve—which is essentially an array of 100 floating-point numbers (0% to 100% of video duration)—requires a specific strategy in SQLite, which lacks a native array type.

* **Strategy:** Serialize the array into a JSON string or a binary blob before storage.35
* *Columns:* video\_id, retention\_data (TEXT/BLOB).
* *Process:* Python's json.dumps(data\_list) converts the API response into a string for storage. json.loads(db\_string) retrieves it for visualization.

### 7.2 Idempotency and Data Integrity

The Python script must be designed to be **idempotent**—meaning multiple executions produce the same result as a single execution.

* **Upsert Logic:** When writing to the daily\_performance table, the script should use the INSERT OR REPLACE or INSERT... ON CONFLICT UPDATE SQL syntax.
* **Scenario:** If the script crashes halfway through processing, re-running it should not create double view counts for the processed videos. This ensures the integrity of the long-term data set.37

## 8. Algorithmic Intelligence: From Data to Insight

The true power of automation is not just storage, but analysis. With the data in SQLite, Ovi English School can apply statistical algorithms to detect trends that are invisible to the naked eye.

### 8.1 Viral Detection via Z-Score Analysis

To identify videos that are performing significantly above expectations, the **Z-Score (Standard Score)** algorithm is the industry standard for anomaly detection.38 This statistical method measures how many standard deviations a data point is from the mean.

**The Educational Use Case:** If a specific grammar video suddenly spikes in views, a Z-Score analysis can flag it instantly.

* **Formula:** 
  + : Views for Video A today.
  + : Average daily views for Video A over the last 28 days.
  + : Standard deviation of views for Video A over the last 28 days.
* **Thresholds:**
  + ****: Moderate increase (Worth monitoring).
  + : Statistical Anomaly (Viral Event).40

**Modified Z-Score:** Educational data can be noisy. A single day of external promotion can skew the mean. A **Modified Z-Score**, using the Median and Median Absolute Deviation (MAD), is often more robust for YouTube data, preventing a single viral day from "breaking" the baseline for subsequent days.42

### 8.2 Retention Slope Analysis: identifying "Rewatch Peaks"

For an English school, a dip in retention is bad, but a *peak* (where the graph goes up) is gold. It means students are rewinding to watch a section again.

* **Automated Insight:** The script can iterate through the retention array stored in the database. It calculates the slope between points  and .
* **Logic:** If slope > 0 for more than 3 consecutive seconds, tag this timestamp.
* **Action:** The dashboard highlights these timestamps. The creator can then check the video: "Ah, students are rewinding the explanation of 'Third Conditional'. This is a difficult topic; I should make a dedicated Short explaining just this concept." This closes the loop between analytics and content strategy.43

## 9. The Visualization Layer: Building the Streamlit Dashboard

The final component of the architecture is the user interface. **Streamlit** is a Python framework that allows for the rapid development of interactive data applications, making it the ideal choice for an internal analytics tool for Ovi English School.45

### 9.1 Dashboard Design Principles

The dashboard should not merely replicate YouTube Studio. It should provide the specific views relevant to the school's strategy.

**1. The "Pulse" Overview:**

* **Metric Cards:** Display aggregated "Last 28 Days" metrics for Total Views, Subscribers Gained, and—crucially—**Search Traffic %**.
* **Trend Chart:** A line chart plotting Long-form Views vs. Shorts Views on dual axes. This visualizes the channel's reliance on each format.46

**2. The Educational Deep Dive:**

* **Cohort Analysis:** A visual comparison of retention curves. The user can select "Grammar Videos" from a dropdown, and the dashboard plots the retention curves of the top 5 grammar videos on a single chart. This allows for instant comparison: "Why did the 'Verbs' video hold attention 20% better than the 'Nouns' video?".45
* **Heatmap:** A visualization of Age Group vs. Time of Day (if available) or Age Group vs. Topic, helping to visualize the target demographic's preferences.

**3. The "Viral Radar" (Z-Score View):**

* A sorted table displaying videos with the highest Z-Score for the current data batch. This effectively functions as a "News Feed" of the channel's rising stars, alerting the educator to "Sleeper Hits"—older videos that the algorithm has suddenly resurfaced.48

## 10. Operational Roadmap & Future-Proofing

Implementing this architecture requires a phased approach to ensure stability and accuracy.

### 10.1 Implementation Phases

1. **Phase 1: Foundation (Days 1-3):**
   * Set up Google Cloud Project and enable APIs.
   * Generate OAuth credentials and perform the "offline" handshake to acquire the Refresh Token.
   * Write the "Hello World" script: Fetch the channel profile to verify authentication.
2. **Phase 2: Inventory & Storage (Days 4-7):**
   * Implement the uploads\_playlist fetcher.
   * Design the SQLite database and run the migration script.
   * Populate the video\_inventory table with historical data (e.g., last 365 days).
3. **Phase 3: Analytics Integration (Days 8-14):**
   * Build the loop to query reports.query for the active inventory.
   * Implement the T-4 day delay logic to handle latency.
   * Store metric data in daily\_performance.
4. **Phase 4: Visualization (Days 15+):**
   * Install Streamlit and build the dashboard.
   * Implement the Z-Score logic within the view layer.

### 10.2 Future-Proofing the System

The YouTube platform is volatile. The definition of a "Short" recently changed from 60 seconds to 3 minutes.23 The API fields change; metrics are deprecated.26

* **Defensive Coding:** The Python script must use try-except blocks around every API call. If a specific metric (e.g., cardClicks) is deprecated, the script should log a warning and continue processing the rest of the data, rather than crashing the entire pipeline.
* **Modular Design:** Separating the "Fetcher" logic (getting data) from the "Storage" logic (saving data) and the "View" logic (Streamlit) allows Ovi English School to swap out components (e.g., upgrading from SQLite to PostgreSQL) without rewriting the entire system.

By adhering to this technical framework, Ovi English School can transcend the limitations of manual analytics. This automated architecture transforms data from a passive report into an active asset, directly informing curriculum development and ensuring that every educational video reaches its maximum potential student base.

#### Works cited

1. Chapter 24 Youtube API | APIs for social scientists: A collaborative review, accessed February 1, 2026, <https://paulcbauer.github.io/apis_for_social_scientists_a_review/youtube-api.html>
2. API Reference | YouTube Data API - Google for Developers, accessed February 1, 2026, <https://developers.google.com/youtube/v3/docs>
3. Using the youtube analytics api or youtube data api, is there a way to get the video from a channel with the most comments? - Stack Overflow, accessed February 1, 2026, <https://stackoverflow.com/questions/73721202/using-the-youtube-analytics-api-or-youtube-data-api-is-there-a-way-to-get-the-v>
4. YouTube Analytics and Reporting APIs - Google for Developers, accessed February 1, 2026, <https://developers.google.com/youtube/analytics>
5. API Reference | YouTube Analytics and Reporting APIs - Google for Developers, accessed February 1, 2026, <https://developers.google.com/youtube/analytics/reference>
6. YouTube Analytics API: Content Owner Reports - Google for Developers, accessed February 1, 2026, <https://developers.google.com/youtube/analytics/content_owner_reports>
7. Implementing OAuth 2.0 Authorization | YouTube Data API | Google for Developers, accessed February 1, 2026, <https://developers.google.com/youtube/v3/guides/authentication>
8. Using OAuth 2.0 for Web Server Applications | YouTube Live Streaming API, accessed February 1, 2026, <https://developers.google.com/youtube/v3/live/guides/auth/server-side-web-apps>
9. How to get a Refresh Token for Google APIs in a Python Script | by Colin Hobday - Medium, accessed February 1, 2026, <https://medium.com/@developer_29635/how-to-get-a-refresh-token-for-google-apis-in-a-python-script-b68f1b6aa668>
10. Google API - OAuth 2.0 - Get AccessToken and Refresh Token Part-1 - YouTube, accessed February 1, 2026, <https://www.youtube.com/watch?v=tGDn3V-mIOM>
11. Using OAuth 2.0 for Web Server Applications | Authorization - Google for Developers, accessed February 1, 2026, <https://developers.google.com/identity/protocols/oauth2/web-server>
12. How do I use the oauth2 access token with the youtube data api - Stack Overflow, accessed February 1, 2026, <https://stackoverflow.com/questions/48243959/how-do-i-use-the-oauth2-access-token-with-the-youtube-data-api>
13. Quota and Compliance Audits | YouTube Data API - Google for Developers, accessed February 1, 2026, <https://developers.google.com/youtube/v3/guides/quota_and_compliance_audits>
14. Please explain the YouTube Data API's Quota / Limits - Stack Overflow, accessed February 1, 2026, <https://stackoverflow.com/questions/77551759/please-explain-the-youtube-data-apis-quota-limits>
15. Your Complete Guide to YouTube Data API v3 – Quotas, Methods, and More - Elfsight, accessed February 1, 2026, <https://elfsight.com/blog/youtube-data-api-v3-limits-operations-resources-methods-etc/>
16. Quota Calculator | YouTube Data API - Google for Developers, accessed February 1, 2026, <https://developers.google.com/youtube/v3/determine_quota_cost>
17. Youtube API limits : How to calculate API usage cost and fix exceeded API quota | Phyllo, accessed February 1, 2026, <https://www.getphyllo.com/post/youtube-api-limits-how-to-calculate-api-usage-cost-and-fix-exceeded-api-quota>
18. I Have 22 Subscribers and the YouTube Data API is Already Trying to End My Career : r/googlecloud - Reddit, accessed February 1, 2026, <https://www.reddit.com/r/googlecloud/comments/1kngbb6/i_have_22_subscribers_and_the_youtube_data_api_is/>
19. Understanding YouTube quota limits: total per day vs. per 100 seconds? - Stack Overflow, accessed February 1, 2026, <https://stackoverflow.com/questions/41835710/understanding-youtube-quota-limits-total-per-day-vs-per-100-seconds>
20. An In-depth YouTube Shorts Analytics Guide - Socialinsider, accessed February 1, 2026, <https://www.socialinsider.io/blog/youtube-shorts-analytics-guide/>
21. YouTube's new search filters make clearer distinctions between long-form videos and Shorts, accessed February 1, 2026, <https://www.tubefilter.com/2026/01/09/youtube-search-filters-shorts-vs-long-form/>
22. Is there a way using YouTube's v3 API to determine if a video (by ID) is a "short" or a regular video? - Stack Overflow, accessed February 1, 2026, <https://stackoverflow.com/questions/78597268/is-there-a-way-using-youtubes-v3-api-to-determine-if-a-video-by-id-is-a-shor>
23. Understand three-minute YouTube Shorts - Google Help, accessed February 1, 2026, <https://support.google.com/youtube/answer/15424877?hl=en>
24. Revision History | YouTube Analytics and Reporting APIs - Google for Developers, accessed February 1, 2026, <https://developers.google.com/youtube/reporting/revision_history>
25. Dimensions | YouTube Analytics and Reporting APIs - Google for Developers, accessed February 1, 2026, <https://developers.google.com/youtube/analytics/dimensions>
26. Metrics | YouTube Analytics and Reporting APIs | Google for Developers, accessed February 1, 2026, <https://developers.google.com/youtube/analytics/metrics>
27. YouTube Analytics API: Channel Reports - Google for Developers, accessed February 1, 2026, <https://developers.google.com/youtube/analytics/channel_reports>
28. Audience Retention Data of Youtube Video via API - Stack Overflow, accessed February 1, 2026, <https://stackoverflow.com/questions/72610500/audience-retention-data-of-youtube-video-via-api>
29. Issues with YouTube Analytics data syncing - Google Help, accessed February 1, 2026, <https://support.google.com/youtube/thread/400074099?hl=en&msgid=400315918>
30. The YouTube Reporting API - Kevin Urban, accessed February 1, 2026, <https://krbnite.github.io/The-YouTube-Reporting-API/>
31. Latency with Youtube Analytics API - Stack Overflow, accessed February 1, 2026, <https://stackoverflow.com/questions/13018142/latency-with-youtube-analytics-api>
32. Use R & SQLite for Data Analysis - YouTube, accessed February 1, 2026, <https://www.youtube.com/watch?v=ZfliyOawJFk>
33. Tutorials - Simon Willison: Data analysis with SQLite and Python - YouTube, accessed February 1, 2026, <https://www.youtube.com/watch?v=5TdIxxBPUSI>
34. SQLite Schema Information Metadata - Stack Overflow, accessed February 1, 2026, <https://stackoverflow.com/questions/6460671/sqlite-schema-information-metadata>
35. Storing Data with SQLite | Python For The Lab, accessed February 1, 2026, <https://pythonforthelab.com/blog/storing-data-with-sqlite/>
36. Inserting an array into sqlite3 python - Stack Overflow, accessed February 1, 2026, <https://stackoverflow.com/questions/11542930/inserting-an-array-into-sqlite3-python>
37. SQLite Large Time Series Data - Reddit, accessed February 1, 2026, <https://www.reddit.com/r/sqlite/comments/18oxinr/sqlite_large_time_series_data/>
38. How to Calculate Z-Scores in Python ( scipy.stats as stats) - Kaggle, accessed February 1, 2026, <https://www.kaggle.com/general/210726>
39. How to Calculate Z-Scores in Python - Statology, accessed February 1, 2026, <https://www.statology.org/z-score-python/>
40. Python Outlier Detection: IQR Method and Z-score Implementation | by Gen. Devin DL., accessed February 1, 2026, <https://medium.com/@tubelwj/python-outlier-detection-iqr-method-and-z-score-implementation-8e825edf4b32>
41. How to detect outliers with z-score - Machine Learning Plus, accessed February 1, 2026, <https://machinelearningplus.com/machine-learning/how-to-detect-outliers-with-z-score/>
42. Modified Z-Score Explained (Python Outlier Detection) - YouTube, accessed February 1, 2026, <https://www.youtube.com/watch?v=wCXQo6OYbhg>
43. YouTube analytics guide 2025: How to read, track, and grow your channel - Sociality.io, accessed February 1, 2026, <https://sociality.io/blog/youtube-analytics/>
44. YouTube Retention Graphs Explained: Boost Your Video Performance - OpusClip Blog, accessed February 1, 2026, <https://www.opus.pro/blog/youtube-retention-graphs-explained>
45. Interactive Data Visualization: Streamlit, Dash, and Bokeh - DEV Community, accessed February 1, 2026, <https://dev.to/diegocastillo12/interactive-data-visualization-streamlit-dash-and-bokeh-13ba>
46. Python Interactive Dashboard Development using Streamlit and Plotly - YouTube, accessed February 1, 2026, <https://www.youtube.com/watch?v=7yAw1nPareM>
47. How to Analyze Your Audience Retention Graph Like a Pro (Using Free Tools), accessed February 1, 2026, <https://www.retentionrabbit.com/blog/analyze-youtube-audience-retention-graph-free-tools>
48. How to build a real-time live dashboard with Streamlit - Show the Community!, accessed February 1, 2026, <https://discuss.streamlit.io/t/how-to-build-a-real-time-live-dashboard-with-streamlit/24437>