Data Engineering 2: Big Data Architechtures

YouTube Data Pipeline

Project Report

Team Members

Mr. Dharshan Dhanashekar(11038649)
Ms. Harshita Jamadade(11038574)
Mr. Navneeth Krishna Aravind (11038618)

Dr. Frank Schulz & Eunnuri Cho
Applied Data Science and Analytics

ABSTRACT

Many organizations rely on data insights to make better decisions and stay competitive. The challenge is not just storing and processing data but turning it into useful information. Analytics helps organizations figure out how to handle data and what to focus on. A well-designed data pipeline ensures data flows smoothly from where it is collected to where it can be used effectively.

To connect with Gen Z on YouTube, it’s important to understand what keeps them interested. The Sidemen are a great example—they’ve built a huge following by looking at how people watch and interact with their videos, like tracking views, likes, and comments. This helps them see what content works best and make even better videos. For marketers, the Sidemen’s engagement (how much people like and interact with their content) helps them plan better ads and sponsorships. The challenge is building an easy system to gather and show all this data so creators and businesses can learn from it and grow their own audiences.

Table Of Contents

1. Introduction

This project report provides a clear explanation of how we built a system to analyse and improve the performance of the Sidemen's YouTube channels. It explains where the data comes from, the main problems we are solving, the steps taken to create the system, challenges we faced, and future possibilities.

The Sidemen, a well-known YouTube group, run multiple channels with different types of content, such as gaming videos, challenges, and collaborations. These channels produce a lot of data, including views, likes, comments, and how engaged viewers are. Collecting, organizing, and analysing this data is important for understanding which videos work best, increasing audience engagement, and creating better promotional strategies.

The main challenge is gathering data from multiple Sidemen channels, handling large amounts of information, and turning it into useful insights for content creators. This project focuses on building an efficient system that can automatically pull data from YouTube, process it, store it in a cloud platform, and display the results in an easy-to-understand dashboard.

2. Related Work

This section highlights existing tools, research, and systems related to YouTube data analytics and identifies how this project adds value.

Several platforms currently offer YouTube performance tracking and analytics. Third-party tools like **Social Blade**, **TubeBuddy**, and **VidIQ** offer additional insights, such as subscriber growth, keyword optimization, and surface-level trends. While useful, these tools lack the flexibility and scalability needed to process large datasets for deeper analysis and real-time insights.

Research in the field of YouTube analytics has focused on audience engagement. Studies have analysed metrics like views, likes, and comments to determine video popularity trends. Sentiment analysis techniques, using Natural Language Processing (NLP), have also been applied to YouTube comments to gauge audience reactions.

In the field of data engineering, cloud-based analytics frameworks like **Google Cloud Platform (GCP)**, **AWS**, and **Azure** have proven effective for building real-time pipelines. Tools like BigQuery and cloud storage solutions allow for large-scale data ingestion, processing, and visualization. While these frameworks are widely used, implementations specific to YouTube analytics remain limited.

This project addresses these gaps by combining real-time data extraction, processing, and visualization into a **custom, scalable pipeline**. By integrating YouTube video performance metrics this system provides actionable insights tailored for content creators, marketing teams.

3. Data Source

This project collects data using the **YouTube Data API v3** to analyse the performance of the Sidemen's YouTube channels. The API provides real-time access to key information, such as channel details and video performance metrics, which are crucial for understanding trends and engagement.

a. APIs Used
The following YouTube API endpoints are used to gather the data:

1. **Channel Statistics**
   - **Endpoint**: youtube/v3/channels
   - **Purpose**: Fetch basic channel information like subscriber count, total views, video count, and description.
2. **Uploads Playlist**
   - **Endpoint**: youtube/v3/playlistItems
   - **Purpose**: Retrieve all video IDs from the channel's uploads playlist.
3. **Video Details**
   - **Endpoint**: youtube/v3/videos
   - **Purpose**: Collect video-specific metrics such as views, likes, comments, tags, and upload date.

b. Data Features
The collected data is divided into **channel-level data** and **video-level data**.

1. **Channel-Level Data**:
   - **Channel Name**: The name of the YouTube channel.
   - **Subscriber Count**: Number of people subscribed to the channel.
   - **Total Views**: Total views across all videos.
   - **Total Videos**: Total number of uploaded videos.
   - **Description**: Short description of the channel.
   - **Country**: Country associated with the channel.
   - **Thumbnail URL**: Link to the channel's logo or thumbnail image.

2. **Video-Level Data**:
    - **Video ID**: Unique identifier for each video.
    - **Title**: Title of the video.
    - **Views**: Total number of views for the video.
    - **Likes**: Number of likes.
    - **Comments**: Number of comments.
    - **Tags**: Keywords or hashtags associated with the video.
    - **Description**: Video description.
    - **Published Date**: Date and time the video was uploaded.
    - **Duration**: The length of the video.

c. Why This Data?
This data helps us:

- Track how each Sidemen channel is performing over time.
- Understand which videos are getting the most views, likes, and comments.
- Identify trends based on tags, video durations, and upload schedules.
- Compare the performance of Sidemen's channels to find what works best.

The data is saved in **Google Cloud Storage (GCS)** in JSON format, making it easy to access and analyse for reporting and visualization.

4. Methodology

The automated pipeline for analysing Sidemen's YouTube data uses Google Cloud services to manage everything smoothly.

a. Automated Scheduling with Cloud Scheduler

Every Monday, a tool called Cloud Scheduler runs automatically. It triggers a program that fetches fresh data from YouTube for all Sidemen channels.

b. Fetching Data Using Cloud Functions

A Python program hosted in Cloud Functions connects to YouTube's API to gather data about the channels (like subscribers, views, and video details). Once collected, this data is saved as files in Google Cloud Storage.

c. Storing Data in Google Cloud Storage

The data files are stored in a Google Cloud Storage bucket. There are separate files for overall channel data and individual video details to keep things organized.

d. Starting Data Processing Automatically

When new files arrive in the storage, another program is triggered. This program triggers a pipeline that processes the raw data into a clean data.

e. Pipeline for Cleaning and Organizing

Pipelines takes the raw data and improves it:

- It calculates average views per video for each channel.
- For individual videos, it checks for trends, tags, and even figures out if a video is "viral" based on how many views it gets. The cleaned and processed data is then sent to a database for further use.

f. Storing Data in BigQuery

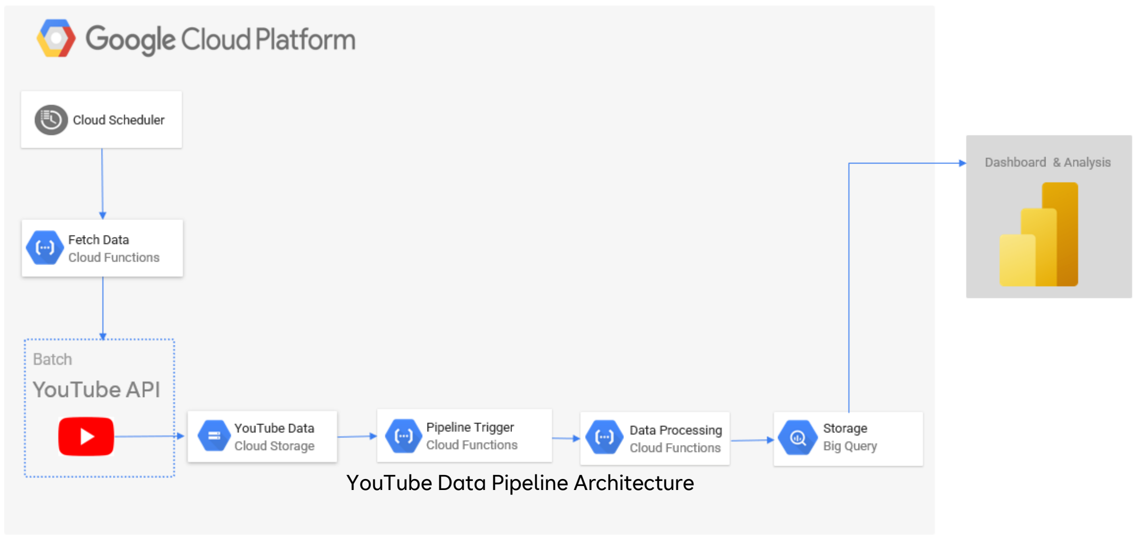
The processed data is stored in BigQuery. It organizes the information in a way that makes it easy to run calculations.

g. Creating Dashboards with PowerBi

Finally, the data in BigQuery is connected to PowerBi, a tool for making graphs and dashboards. This step creates visual reports to track Sidemen's subscriber growth, video performance, and engagement with their fans.

5. Data Pipeline

The architecture for the Sidemen YouTube data pipeline is designed to collect, process, store, and visualize batch data from YouTube's API. The pipeline involves several components and steps, ensuring an automated flow of data from YouTube to end-user dashboards for insights.



YouTube Data Pipeline Architecture

- **Cloud Scheduler** is configured to trigger the fetching of the data every Monday. It ensures that the data-fetching process starts consistently without manual intervention.

- **Google Cloud Functions(Api-Fetch)** executes a Python script when triggered by Cloud Scheduler. This Python script fetches channel-level and video-level data from the **YouTube Data API**.
    - The fetched data includes details like subscriber count, total views, video metrics (likes, comments, tags), and video durations.
    - Once fetched, the data is stored as JSON files in **Google Cloud Storage** (GCS).

- **Google Cloud Storage (GCS)** acts as the storage layer for raw data. Two types of JSON files are saved:
    - channel_stats.json – containing channel-level statistics.
    - video_analytics.json – containing detailed video analytics.

- **Cloud Functions(Trigger)** automatically detect when new files arrive in the GCS bucket. Upon detecting these files, **Cloud Function(preprocessing)** starts executing data processing pipeline.

- The data is processed and transformed in two ways
    - **Channel Pipeline**: Processes channel statistics, calculates average views per video, and organizes the channel data.
    - **Video Pipeline**: Processes video-level metrics, calculates engagement rates, identifies viral videos, standardizes video durations, and cleans the tags and descriptions.

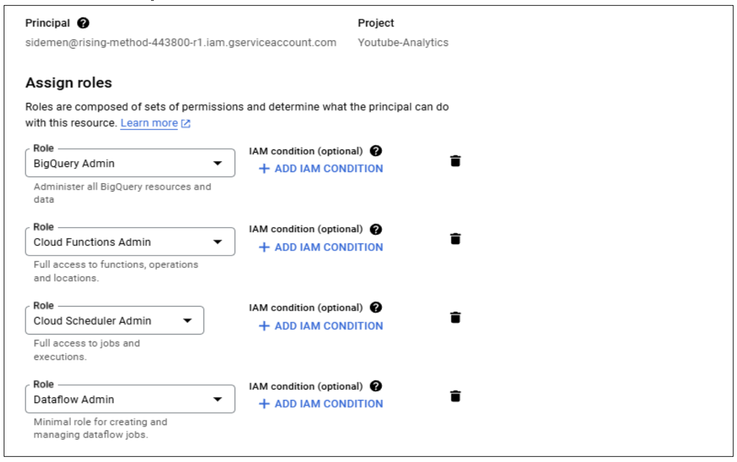
- The **processed data** is stored in **Google BigQuery**, a fully managed data warehouse.
    - BigQuery organizes the data into two tables: channel_stats and video_stats.

- **PowerBi** is used to create interactive dashboards that visualize the processed data.
    - Power BI's DirectQuery mode helps automate dashboard updates, ensuring that the latest data is reflected without manual intervention.
    - The dashboards provide insights into the Sidemen YouTube channels.

6. Implementation

a. Create a Service Account with Required Permissions

1. Go to the **Google Cloud Console**.
2. Click on the **hamburger icon** on the left side.
3. Hover over **IAM & Admin**.
4. Click on **Service Accounts**.
5. Click on **CREATE SERVICE ACCOUNT**.
6. Add the following roles:
    - BigQuery Admin
    - Cloud Functions Admin
    - Cloud Scheduler Admin
    - Storage Admin
7. Click on **Done**.
8. Follow the steps in this link to create **service account keys**.



Principal
sidemen@rising-method-443800-r1.iam.gserviceaccount.com

Project
Youtube-Analytics

Assign roles

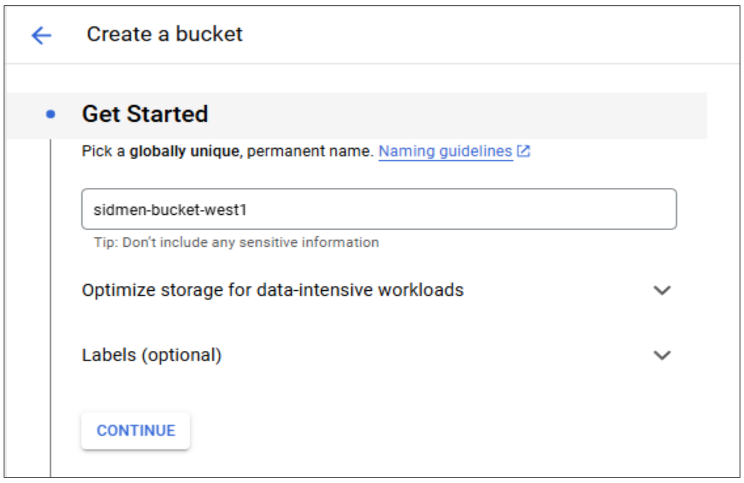
Roles are composed of sets of permissions and determine what the principal can do with this resource. Learn more

Role	Description	IAM condition (optional)
BigQuery Admin	Administer all BigQuery resources and data	ADD IAM CONDITION
Cloud Functions Admin	Full access to functions, operations and locations.	ADD IAM CONDITION
Cloud Scheduler Admin	Full access to jobs and executions.	ADD IAM CONDITION
Dataflow Admin	Minimal role for creating and managing dataflow jobs.	ADD IAM CONDITION

Service Account creation page

b. Create a Cloud Storage Bucket

1. Search for **Cloud Storage** in the Google Cloud Console.
2. Click on **CREATE**.
3. Provide a name for the bucket (e.g., sidemen-bucket-west1).
4. Choose a region and create the bucket.


Create a bucket

Get Started

Pick a **globally unique**, permanent name. Naming guidelines

sidmen-bucket-west1

Tip: Don't include any sensitive information

Optimize storage for data-intensive workloads

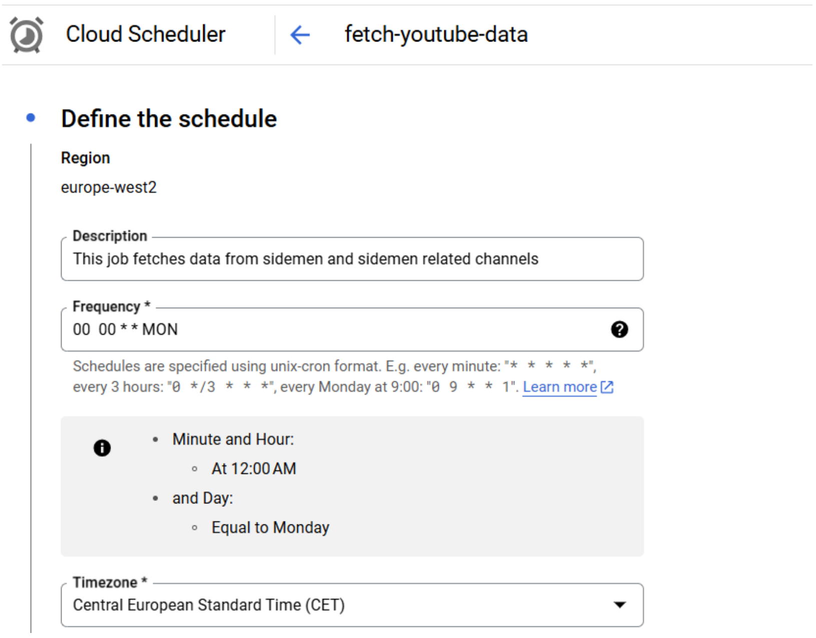
Labels (optional)

CONTINUE

Cloud Storage bucket creation page.

c. Create Cloud Scheduler

1. Search for **Cloud Scheduler**.
2. Click on **CREATE JOB**.
3. Provide a name, select a region, and set the frequency to 0 0 * * 1 (every Monday).
4. Select time zone and click **Continue**.
5. Set target type as **HTTP**.
6. Provide the **HTTP URL** of the Cloud Function (generated later in step e).
7. Use HTTP Method **GET** or **POST** (depending on your function).
8. Save the job.


Cloud Scheduler | fetch-youtube-data

Define the schedule

Region
europe-west2

Description: This job fetches data from sidemen and sidemen related channels

Frequency *: 00 00 * * MON

Schedules are specified using unix-cron format. E.g. every minute: "* * * * *", every 3 hours: "0 */3 * * *", every Monday at 9:00: "0 9 * * 1". Learn more

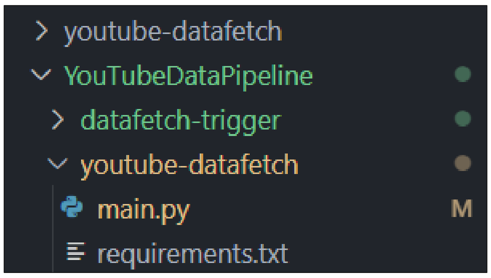
- Minute and Hour:
  - At 12:00 AM
- and Day:
  - Equal to Monday

Timezone *: Central European Standard Time (CET)

Cloud Scheduler setup with frequency

d. Prepare Python Code

1. Replace the environment variable with the path of the generated service account key.
2. Create a requirements.txt file containing the dependencies
3. Zip the following files:
    - main.py (your Python script for fetching YouTube data).
    - requirements.txt.

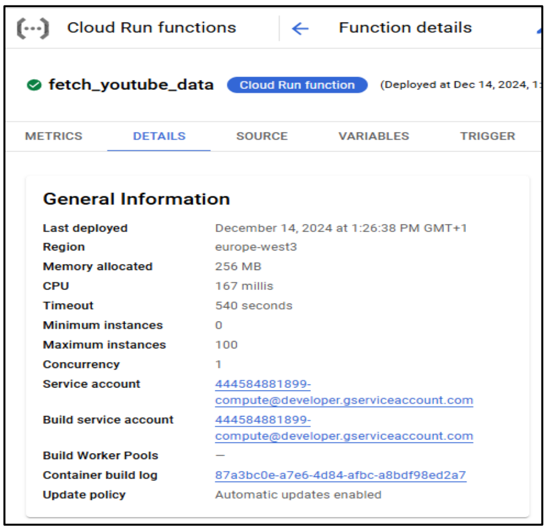


```
> youtube-datafetch
v YouTubeDataPipeline
  > datafetch-trigger
  v youtube-datafetch
      main.py              M
      requirements.txt
```

Folder structure with main.py, requirements.txt

e. Create Cloud Functions

1. Search for **Cloud Functions**.
2. Click on **CREATE FUNCTION**.
3. Provide a name (e.g., fetch-youtube-data) and select a region.
4. Select trigger type as **HTTP** .
5. Click on **MORE OPTIONS** and select the created **service account**.
6. Click **NEXT**.
7. Select runtime: **Python 3.11**.
8. Select source code: **ZIP upload**.
9. Provide the **entry point** as fetch_youtube_data.
10. Upload the ZIP file and deploy.



fetch_youtube_data Cloud Run function (Deployed at Dec 14, 2024, 1

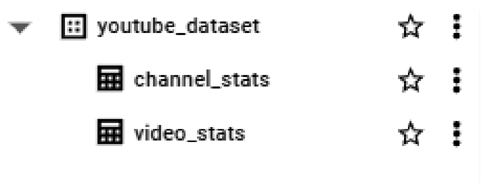
General Information

Last deployed	December 14, 2024 at 1:26:38 PM GMT+1
Region	europe-west3
Memory allocated	256 MB
CPU	167 millis
Timeout	540 seconds
Minimum instances	0
Maximum instances	100
Concurrency	1
Service account	444584881899-compute@developer.gserviceaccount.com
Build service account	444584881899-compute@developer.gserviceaccount.com
Build Worker Pools	—
Container build log	87a3bc0e-a7e6-4d84-afbc-a8bdf98ed2a7
Update policy	Automatic updates enabled

Cloud Function details

f. Create a BigQuery Dataset

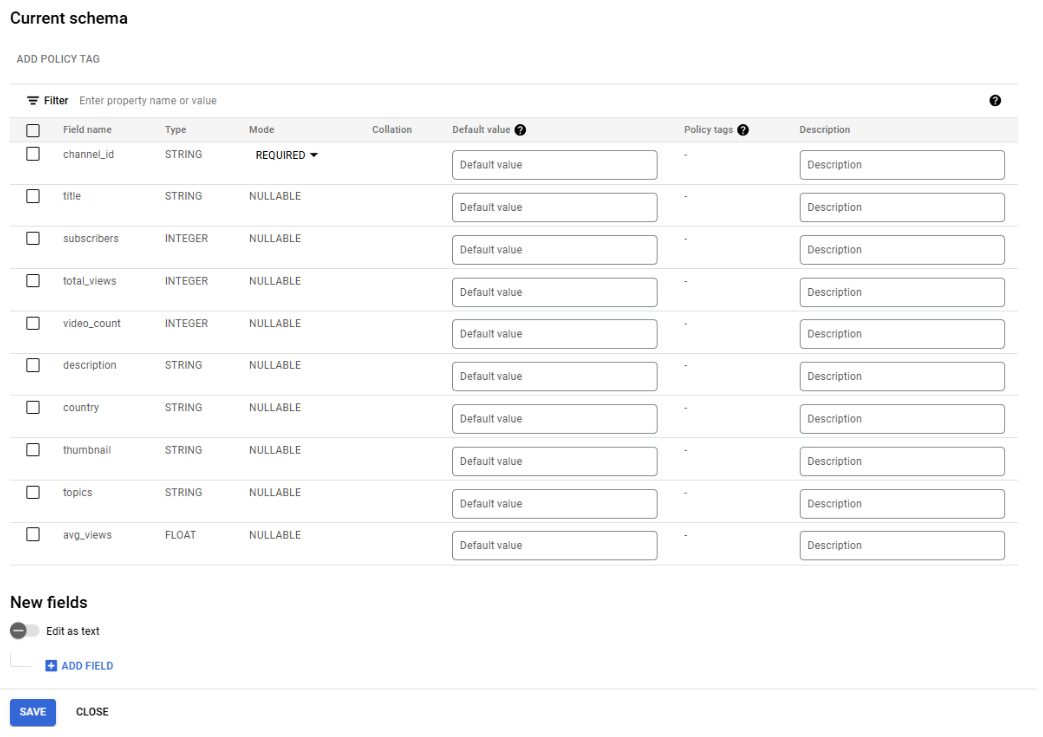
1. Search for **BigQuery** in the Google Cloud Console.
2. Click on the **three dots** next to the project name.
3. Select **Create Dataset**.
4. Provide a name (e.g., youtube_dataset), select a region, and click **Create**.



BigQuery dataset

g. Create BigQuery Tables

1. In the **BigQuery console**, select the created dataset.
2. Click on the **three dots** next to the dataset and select **Create Table**.
3. Provide a name for the table:
   - channel_stats for channel-level data.
   - video_stats for video-level analytics.
4. Use the **schema** based on your pipeline:
   - Channel Table: channel_id, title, subscribers, total_views, etc.
   - Video Table: video_id, title, views, likes, comments, isviral, etc.
5. Save the tables.



Current schema

ADD POLICY TAG

Filter Enter property name or value

Field name	Type	Mode	Collation	Default value	Policy tags	Description
channel_id	STRING	REQUIRED		Default value	-	Description
title	STRING	NULLABLE		Default value	-	Description
subscribers	INTEGER	NULLABLE		Default value	-	Description
total_views	INTEGER	NULLABLE		Default value	-	Description
video_count	INTEGER	NULLABLE		Default value	-	Description
description	STRING	NULLABLE		Default value	-	Description
country	STRING	NULLABLE		Default value	-	Description
thumbnail	STRING	NULLABLE		Default value	-	Description
topics	STRING	NULLABLE		Default value	-	Description
avg_views	FLOAT	NULLABLE		Default value	-	Description

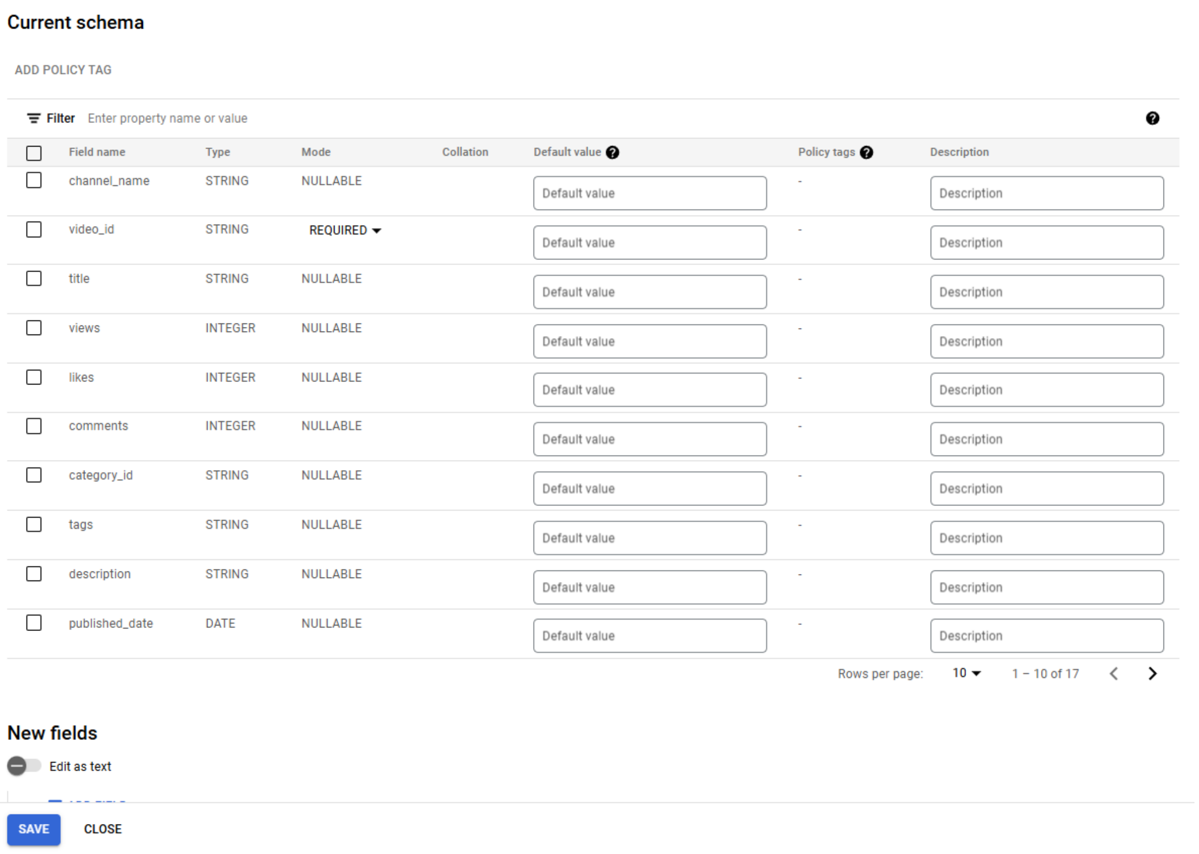
New fields

Edit as text

ADD FIELD

SAVE CLOSE

Schema of Channel information.


Current schema

ADD POLICY TAG

Filter Enter property name or value

	Field name	Type	Mode	Collation	Default value	Policy tags	Description
☐	channel_name	STRING	NULLABLE		Default value	-	Description
☐	video_id	STRING	REQUIRED		Default value	-	Description
☐	title	STRING	NULLABLE		Default value	-	Description
☐	views	INTEGER	NULLABLE		Default value	-	Description
☐	likes	INTEGER	NULLABLE		Default value	-	Description
☐	comments	INTEGER	NULLABLE		Default value	-	Description
☐	category_id	STRING	NULLABLE		Default value	-	Description
☐	tags	STRING	NULLABLE		Default value	-	Description
☐	description	STRING	NULLABLE		Default value	-	Description
☐	published_date	DATE	NULLABLE		Default value	-	Description

Rows per page: 10 1 – 10 of 17

New fields

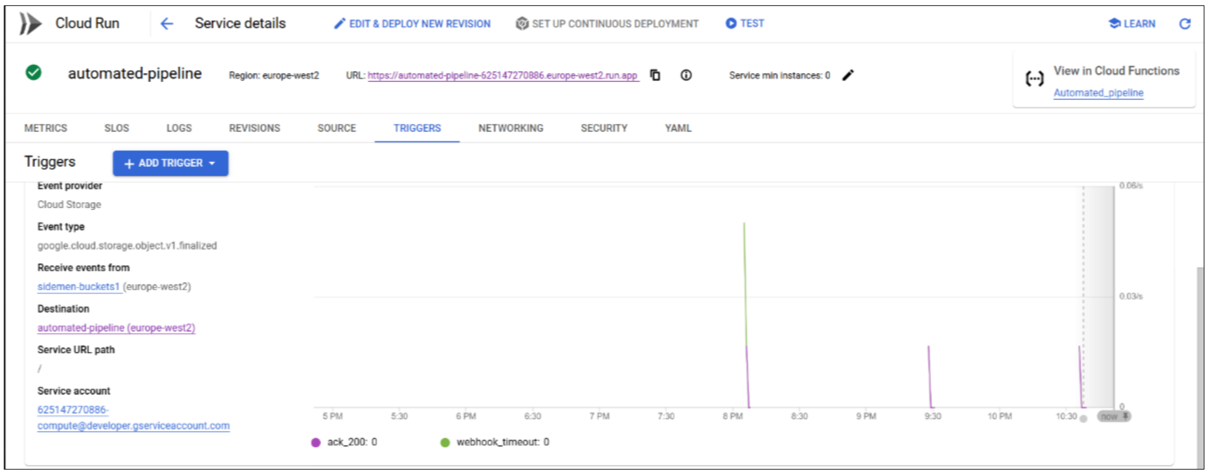
Edit as text

SAVE CLOSE

Example Schema of Video information

h. Create Cloud Functions to Trigger Pipelines

- Ensure the Cloud Function executes the logic to trigger two pipelines:
  - **Channel Pipeline**: Processes channel_stats.json.
  - **Video Pipeline**: Processes video_analytics.json.



automated-pipeline Region: europe-west2 URL: https://automated-pipeline-625147270886.europe-west2.run.app Service min instances: 0

View in Cloud Functions Automated_pipeline

Triggers

Event provider
Cloud Storage

Event type
google.cloud.storage.object.v1.finalized

Receive events from
sidemen-buckets1 (europe-west2)

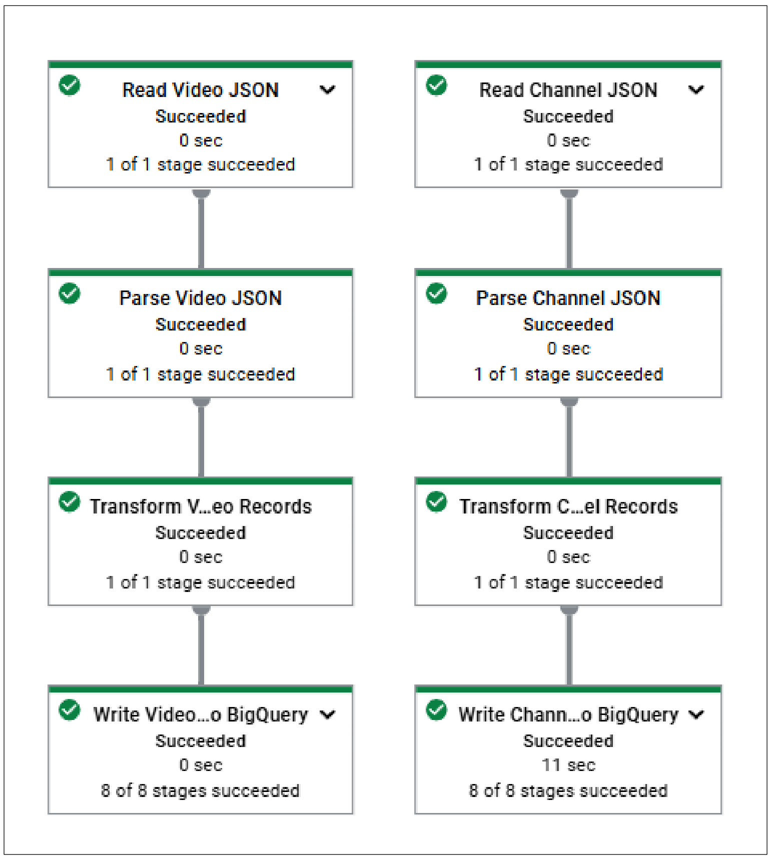
Destination
automated-pipeline (europe-west2)

Service URL path
/

Service account
625147270886-compute@developer.gserviceaccount.com

● ack_200: 0 ● webhook_timeout: 0

Graph of triggered functions


Read Video JSON
Succeeded
0 sec
1 of 1 stage succeeded

Parse Video JSON
Succeeded
0 sec
1 of 1 stage succeeded

Transform V...eo Records
Succeeded
0 sec
1 of 1 stage succeeded

Write Video...o BigQuery
Succeeded
0 sec
8 of 8 stages succeeded

Read Channel JSON
Succeeded
0 sec
1 of 1 stage succeeded

Parse Channel JSON
Succeeded
0 sec
1 of 1 stage succeeded

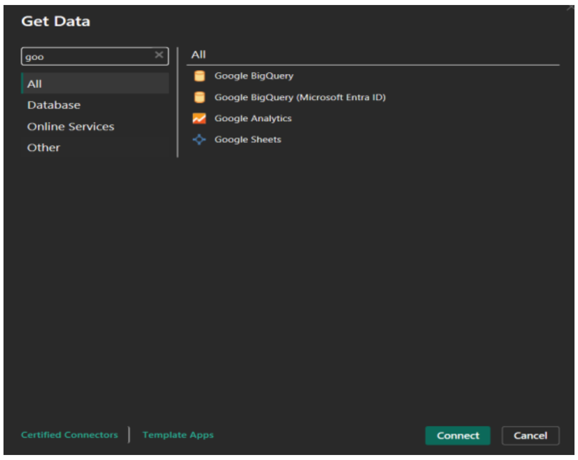
Transform C...el Records
Succeeded
0 sec
1 of 1 stage succeeded

Write Chann...o BigQuery
Succeeded
11 sec
8 of 8 stages succeeded

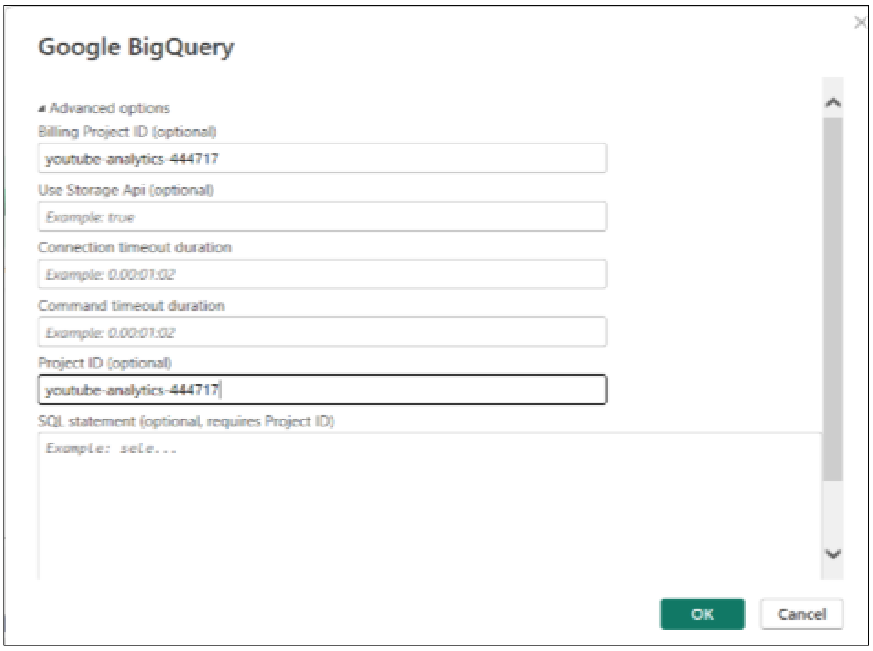
Triggered pipelines when ran individually

i. Connect Power BI to BigQuery in DirectQuery Mode

1. In Power BI Desktop, click **Get Data**.
2. Search for and select **Google BigQuery**.
3. Click **Connect**.
4. After authenticating, browse your BigQuery project
5. Select the desired dataset and table(s).
6. When prompted to choose the connection mode, select **DirectQuery**.
7. Click **Load** to establish the connection.



Connection to Google BigQuery



Google BigQuery

Advanced options

Billing Project ID (optional): youtube-analytics-444717

Use Storage Api (optional): Example: true

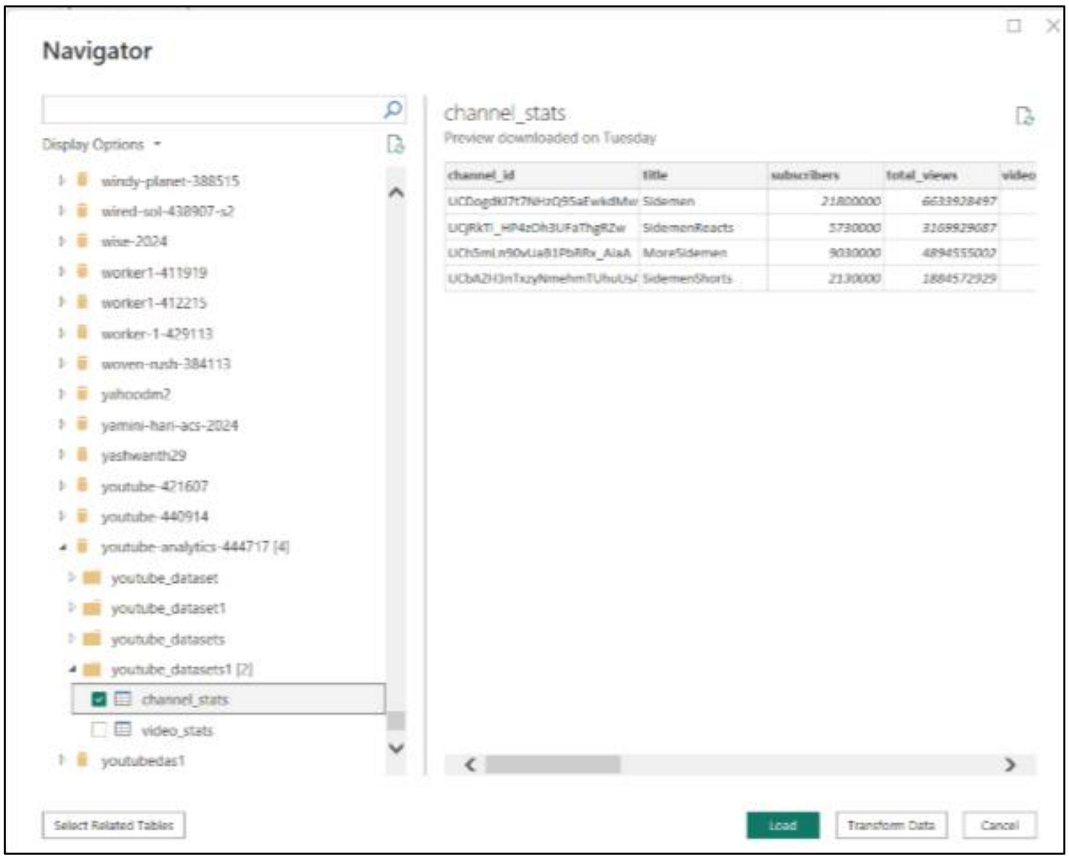
Connection timeout duration: Example: 0.00:01:02

Command timeout duration: Example: 0.00:01:02

Project ID (optional): youtube-analytics-444717

SQL statement (optional, requires Project ID): Example: sele...

Setting up same project id as your GCP project id



Navigator

- windy-planet-388515
- wired-sol-438907-s2
- wise-2024
- worker1-411919
- worker1-412215
- worker-1-429113
- woven-rush-384113
- yahoodm2
- yamini-hari-acs-2024
- yashwanth29
- youtube-421607
- youtube-440914
- youtube-analytics-444717 [4]
  - youtube_dataset
  - youtube_dataset1
  - youtube_datasets
  - youtube_datasets1 [2]
    - [✓] channel_stats
    - [ ] video_stats
- youtubedas1

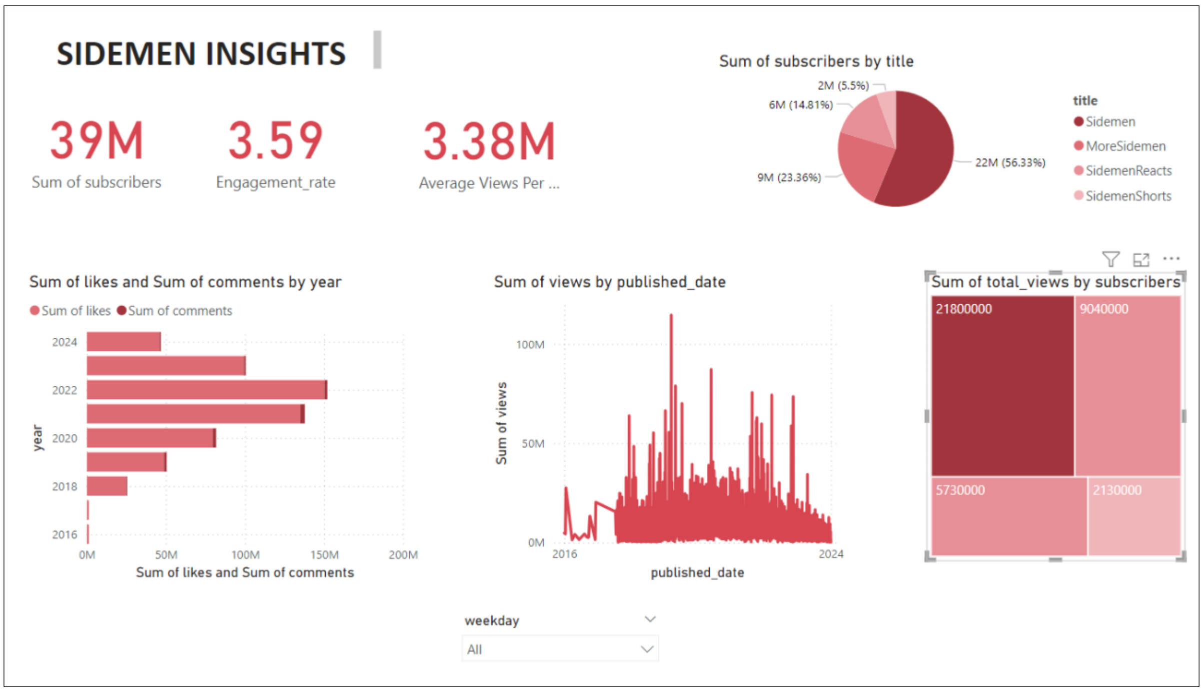
channel_stats

Preview downloaded on Tuesday

channel_id	title	subscribers	total_views	video
UCDogdKl7t7NHzQ95aFwkdMw	Sidemen	21800000	6633928497	
UCjRkTl_HP4zOh3UFaThgRZw	SidemenReacts	5730000	3169929687	
UChSmLn90vLia81PbRRx_AlaA	MoreSidemen	9030000	4894555002	
UCbAZH3nTxzyNmehmTUhuUs/	SidemenShorts	2130000	1884572929	

Selecting table to be visualized

Visualisation



7. Problems Faced

1. Data Fetching and API Usage

While fetching data from the YouTube API, "Too Many Requests" errors were encountered due to API rate limits, especially when handling multiple channels and videos.
Solution: Introduced delays between API requests to comply with rate-limiting restrictions. Optimized API usage by limiting data fetches to only required fields.

2. Runtime and Timeout Issues

The Cloud Function encountered timeout errors because processing large datasets (e.g., multiple videos and analytics) exceeded the default 60-second limit.
Solution: Extended the **Cloud Function timeout limit** to the maximum duration of **540 seconds (9 minutes)**.

3. Handling Large JSON Files in Apache Beam

Parsing and processing large JSON files in the pipeline posed challenges in maintaining performance.
Solution: Implemented custom pipeline options to efficiently handle file paths and JSON input formats.

4. Connecting PowerBi to BigQuery

Errors occurred during the final stages of connecting PowerBi to BigQuery, especially related to schema mismatches and data compatibility.
Solution: Ensured proper datatype compatibility between PowerBi fields and BigQuery schemas.

5. Designing Dashboards

Creating visually appealing and user-friendly dashboards was challenging due to resizing and layout issues in PowerBi.
Solution:

- Avoided overloading dashboards with unnecessary visuals

Adjusting to GCP services like Cloud Functions required additional time for debugging and learning. Certain services experienced provisioning delays in specific regions, requiring deployment in alternate regions (e.g., europe-west1)

8. Future Scope

Although we successfully achieved our project objectives, there are several opportunities for further enhancement and expansion. The potential areas for future development include:

- **Incorporate Real-Time Data Processing**: Currently, the pipeline operates in batch mode, using BigQuery for aggregations.
- **Integrating Data from Other Platforms**: Expand the project to include data from other social media platforms (e.g., Twitter, Instagram) to perform cross-platform analysis and gain a holistic view of the Sidemen's online presence.
- **Optimize API Usage and Performance**: Improve API request management further by implementing parallel data fetching and advanced error-handling strategies. This will allow the system to fetch large amounts of data faster while maintaining compliance with API rate limits.

9. Results and Conclusions

The project successfully automated the process of collecting and analyzing data from Sidemen's YouTube channels. Using tools like **Cloud Scheduler**, **Cloud Functions**, and **BigQuery**, the pipeline fetched data about subscribers, video views, likes, and other key metrics. This data was then cleaned, organized, and visualized in **Power BI dashboards** to provide clear insights into video performance, subscriber growth, and engagement trends.

In simple terms, this project made it easy to track how well Sidemen's videos and channels are doing. The results allow stakeholders to quickly see what works best, helping them make better decisions about future content and strategies.

Here you can view the entire pipeline : Video link

GitHub Link: https://github.com/JamadadeHarshita/YouTubeDataPipeline

References

1. Apache Beam, "Pipeline Options," [Online]. Available: https://beam.apache.org/documentation/patterns/pipeline-options/. [Accessed: 16-Dec-2024].
2. Google Cloud, "Setting Pipeline Options," [Online]. Available: https://cloud.google.com/dataflow/docs/guides/setting-pipeline-options#python. [Accessed: 18-Dec-2024].
3. Stack Overflow, "TypeError: Worker takes 0 positional arguments but 1 was given," [Online]. Available: https://stackoverflow.com/questions/18884782/typeerror-worker-takes-0-positional-arguments-but-1-was-given. [Accessed: 15-Dec-2024].
4. Apache Beam, "Apache Beam 2.33.0 Pydoc Documentation," [Online]. Available: https://beam.apache.org/releases/pydoc/2.33.0/apache_beam.options.pipeline_options.html. [Accessed: 18-Dec-2024].
5. Apache Beam, "Apache Beam BigQuery I/O," [Online]. Available: https://beam.apache.org/releases/pydoc/current/apache_beam.io.gcp.bigquery.html. [Accessed: 18-Dec-2024].
6. Pulsar Platform, "How the Sidemen Built a Huge Gen Z Audience and Turned Them into Customers," [Online]. Available: https://www.pulsarplatform.com/blog/2023/how-the-sidemen-built-a-huge-gen-z-audience-and-turned-them-into-customers. [Accessed: 4-Dec-2024].
7. Viralyft, "YouTube Engagement," [Online]. Available: https://viralyft.com/blog/youtube-engagement. [Accessed: 5-Dec-2024].
8. VidIQ, "YouTube Stats for Sidemen," [Online]. Available: https://vidiq.com/youtube-stats/channel/UCDogdKl7t7NHzQ95aEwkdMw/. [Accessed: 4-Dec-2024].

Tasks Distribution

Harshita

1) Setting up Cloud schedulers
2) Fetching YouTube Data Using cloud functions and Uploading in Cloud Storage Bucket
3) Designing The Architecture
4) Setting up PPT
5) Forming Detailed Report [Chapter 1,2,3,4,5,6]

Navneeth

1) Creation of 2 pipelines: Channel and Video pipeline consisting of Preprocessing the Data
2) Setting up a trigger function for 2 pipelines.
3) Updating the table in BigQuery whenever a change occurs in bucket.
4) Formatting Detailed Report [Abstract, Chapter 4,5,6,7]

Dharshan

1) Making BigQuery Schemas
2) Connecting to PowerBi for Visualisation
3) Writing Queries
4) Formatting detailed report[References, Chapter 4,6,8]