

Rapport de travail

Introduction à la big data

Groupe : Brian Madapathage, Daniel Monteiro, Yacine Boucenna, David Chau

Introduction

Contexte

Pour mener à bien notre étude, nous avons choisi d'effectuer une analyse approfondie des données en utilisant trois sources distinctes. Ces sources serviront à collecter et à croiser les informations nécessaires pour examiner les tendances sur YouTube et comprendre les mécanismes de son algorithme. En nous appuyant sur cette méthodologie, nous chercherons à mettre en lumière les principaux facteurs qui influencent la visibilité des vidéos, notamment dans les recommandations et les classements des tendances. Ce choix de plateforme s'explique par l'importance croissante de YouTube en tant que principal acteur dans la diffusion de contenu vidéo et son rôle déterminant dans la manière dont les utilisateurs découvrent et consomment les médias numériques.

Données

Pour réaliser ce projet, nous avons entrepris une démarche structurée pour sélectionner et exploiter des datasets pertinents afin d'analyser les données et de mieux comprendre le fonctionnement de l'algorithme de YouTube. L'objectif est de s'appuyer sur des sources fiables et récentes pour explorer les dynamiques derrière les recommandations et les classements des vidéos sur la plateforme.

Nous avons choisi trois datasets clés, pour garantir l'actualité et la pertinence des analyses :

- Statistiques mondiales de YouTube 2023 ([kaggle.com](https://www.kaggle.com/datasets/youtubedata/worldwide-youtube-statistics-2023))

Ce dataset contient des données globales sur l'activité de YouTube, y compris le volume de vues, les interactions, et d'autres métriques agrégées. Il permet d'identifier les tendances générales et de comprendre les comportements des utilisateurs à grande échelle.

- Classement des chaînes, par rapport au nombre d'abonnés (top 1000) ([github.com](https://www.github.com/youtubedata/top-1000-youtube-channels))

Ce dataset liste les 1000 chaînes les plus suivies sur YouTube, en précisant des informations telles que le nombre d'abonnés, les catégories de contenu, et l'activité globale des chaînes. Ces données sont essentielles pour analyser les facteurs qui contribuent à la popularité des créateurs et leur impact sur les recommandations et classements.

- Recommandations de YouTube des vidéos aux usa ([kaggle.com](https://www.kaggle.com/datasets/youtubedata/youtube-recommendations-usa))

Ce dataset fournit des informations spécifiques sur les vidéos recommandées aux utilisateurs américains, y compris les titres, les mots-clés, et les métriques associées. Il constitue une base de travail idéale pour explorer les mécanismes des recommandations personnalisées et leur alignement avec les tendances générales.

Grâce à cette sélection, nous pourrions croiser les informations, identifier des corrélations pertinentes et tirer des conclusions solides sur les éléments qui influencent les recommandations et les tendances sur YouTube.

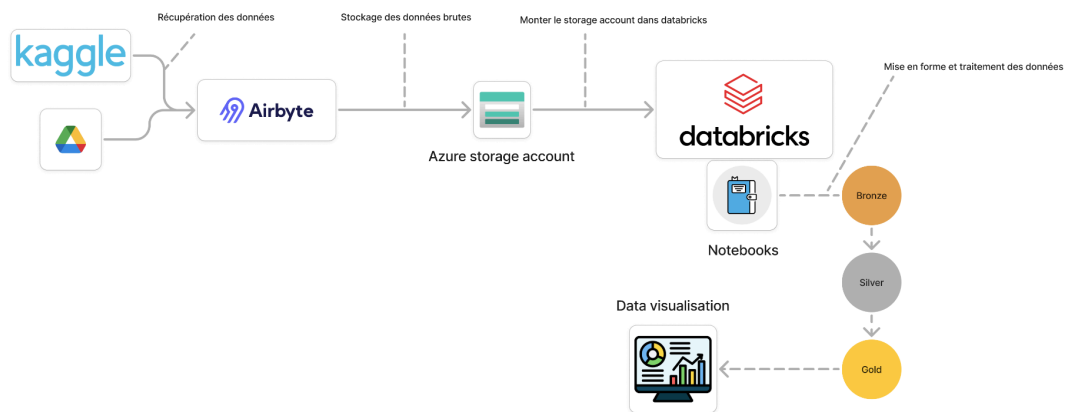
Objectif

L'objectif principal de ce devoir est de comprendre les mécanismes et critères utilisés par l'algorithme de YouTube pour prioriser les vidéos dans les recommandations et les tendances. À travers une analyse approfondie de données issues de plusieurs sources, il s'agit de :

1. Identifier les facteurs clés influençant la visibilité des vidéos, tels que, les likes, les tags, les descriptions et les mots-clés et abonnés.
2. Explorer l'impact de ces paramètres sur le classement des vidéos, dans les tendances générales.
3. Mettre en lumière les relations entre les performances des vidéos (par exemple, vues, abonnés, interactions) et leur positionnement dans les algorithmes de recommandation de YouTube.

Ce projet vise également à développer des compétences pratiques en analyse de données, en croisant différentes sources pour produire des insights, et à mettre en œuvre des outils d'exploration et de visualisation adaptés, tout en tenant compte des enjeux technologiques et éthiques liés à l'utilisation d'algorithmes.

Architecture du projet :



Traitements effectués

Traitements Airbyte :

Durant ce projet, nous avons choisi d'utiliser Airbyte pour synchroniser notre compte de stockage Azure avec les différentes sources de données décrites précédemment.

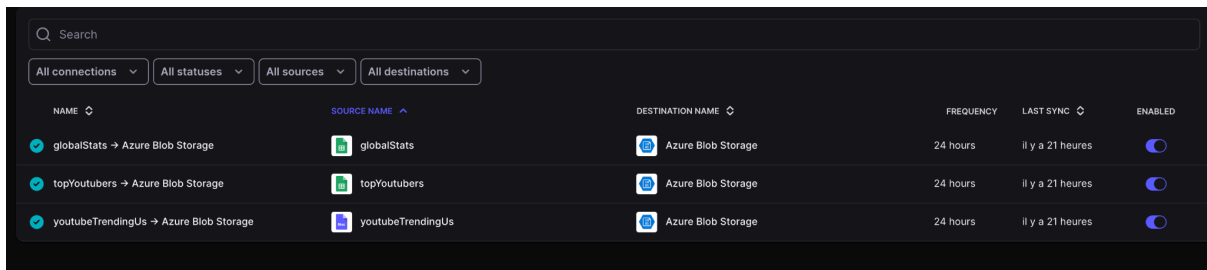
Airbyte s'est avéré être un outil essentiel dans le cadre de cette tâche, grâce à ses fonctionnalités de connecteurs prêts à l'emploi et à sa flexibilité pour intégrer des données provenant de multiples origines. Voici les étapes principales suivies pour mener à bien cette synchronisation :

- a. Configuration des connecteurs :

Nous avons configuré des connecteurs Airbyte pour accéder aux sources de données suivantes :

- Les statistiques mondiales de YouTube 2023. (Depuis un site web hébergé)
- Le classement des chaînes YouTube (Top 1000). (Depuis un Google Sheet partagé)

- Les recommandations de vidéos YouTube aux États-Unis. (Depuis un Google Sheet partagé)

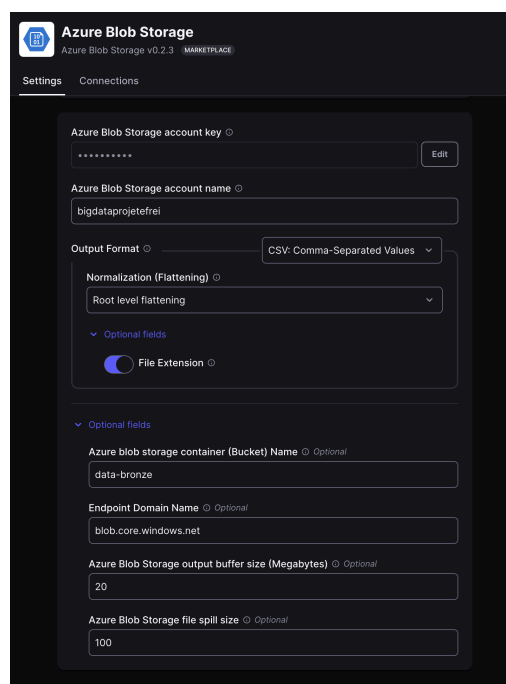


NAME	SOURCE NAME	DESTINATION NAME	FREQUENCY	LAST SYNC	ENABLED
globalStats → Azure Blob Storage	globalStats	Azure Blob Storage	24 hours	il y a 21 heures	<input checked="" type="checkbox"/>
topYoutubers → Azure Blob Storage	topYoutubers	Azure Blob Storage	24 hours	il y a 21 heures	<input checked="" type="checkbox"/>
youtubeTrendingUs → Azure Blob Storage	youtubeTrendingUs	Azure Blob Storage	24 hours	il y a 21 heures	<input checked="" type="checkbox"/>

Ces connecteurs ont permis de se connecter directement aux plateformes ou fichiers contenant les données nécessaires.

- Définition de la destination :

J'ai défini notre compte de stockage Azure comme destination pour toutes les données synchronisées (comme décrit ci-dessous). Grâce à la compatibilité d'Airbyte avec Azure Blob Storage, nous avons pu facilement centraliser les datasets sous forme de fichiers CSV, prêts à être exploités.



Azure Blob Storage
Azure Blob Storage v0.2.3 MARKETPLACE

Settings Connections

Azure Blob Storage account key Edit

Azure Blob Storage account name

Output Format

Normalization (Flattening)

Optional fields

☒ File Extension

Optional fields

Azure blob storage container (Bucket) Name

Endpoint Domain Name

Azure Blob Storage output buffer size (Megabytes)

Azure Blob Storage file spill size

- **Mappage des schémas :**

Une attention particulière a été portée au mappage des schémas pour garantir une correspondance cohérente entre les colonnes des données sources et celles utilisées dans notre entrepôt de données. Cela a permis de simplifier les étapes ultérieures de transformation et d'analyse.

En effet, nous avons rencontré un problème à ce niveau, car les données créées étaient au format JSON et ont été ensuite factorisées. Il nous a fallu donc reconfigurer la connexion.

- **Automatisation des synchronisations :**

Afin d'assurer une mise à jour régulière des données, j'ai configuré des synchronisations planifiées dans Airbyte. Cela garantit que nos analyses reposent toujours sur des données fraîches.

- **Validation des données synchronisées :**

Une fois les données transférées dans le compte de stockage Azure, j'ai effectué des contrôles de qualité pour m'assurer de leur intégrité et de leur conformité avec les attentes initiales.

En conclusion, l'utilisation d'Airbyte a permis d'accélérer et de simplifier le processus d'intégration des données. Cela nous a offert une base solide pour passer à l'étape suivante : la transformation des données en tables **bronze**, **silver**, et **gold**, afin de répondre aux différents cas d'utilisation et analyses prévus.

Transformation des données

Nous avons utilisé comme approche L'architecture en médaille, un schéma de conception des données utilisé pour organiser logiquement les données dans un lakehouse, avec pour objectif d'améliorer progressivement la qualité des données à mesure qu'elles traversent différentes couches.

- **Création des données Bronze :** Cette couche contient les données brutes récupérées directement des sources externes, sans transformation ou traitement significatif. Dans notre cas, les données sont sous la forme de fichiers csv issus de kaggle.com. L'objectif est de constituer nos sources de données pour une utilisation future.

Les étapes du traitement de la couche bronze :

- Déclaration des variables de montage du storage.

```
▶ Just now (<1s) 1 Python
storage_name = "bigdataprojetefrei"
container_name_bronze = "data-bronze"
access_key = "qexMiGy313KNlsvukIhDAcTvcpgEkswIhjZvPswI15D8V/Ag/FaOsE2A/hkkCM704wPkXKAeTQXF+ASteVzNxw=="
mount_point_name_bronze = "/mnt/data-bronze"
```

- Montage du storage Azure :

```
▶ 3 days ago (24s) 2 Python
if any(m.mountPoint == mount_point_name_bronze for m in dbutils.fs.mounts()):
    dbutils.fs.unmount(mount_point_name_bronze)
dbutils.fs.mount(
    source = f"wasbs://{container_name_bronze}@{storage_name}.blob.core.windows.net/",
    mount_point = mount_point_name_bronze,
    extra_configs = {
        f"fs.azure.account.key.{storage_name}.blob.core.windows.net": access_key
    }
)
```

- Inclusion et exécution du code de montage du storage Azure

```

▶ 01:45 PM (22s) 1 Python
%run ./include/include-bronze

/mnt/data-bronze has been unmounted.

True

```

- Création des schemas et chargement des données dans des dataframes

```

StructField("comments_disabled", StringType(), True),
StructField("description", StringType(), True),
StructField("dislikes", StringType(), True),
StructField("likes", StringType(), True),
StructField("publishedAt", StringType(), True),
StructField("ratings_disabled", StringType(), True),
StructField("tags", StringType(), True),
StructField("thumbnail_link", StringType(), True),
StructField("title", StringType(), True),
StructField("trending_date", StringType(), True),
StructField("video_id", StringType(), True),
StructField("view_count", StringType(), True),
])

spark.conf.set(
    "fs.azure.account.key.bigdataprojetefrei.dfs.core.windows.net",
    "qexMiGy313KNlsvukIhDacTvcpgEkswIhjZvPsWI15D8V/Ag/FaOsE2A/hkkCM704wPkXKAeTQXF+ASteVzNxxw==")
df_globalStats = spark.read.format("csv").option("header", "true").schema(schemaGlobalStats).load("abfss://
data-bronze@bigdataprojetefrei.dfs.core.windows.net/globalStatsglobalStats_0/2025_01_05_1736095483934_0.
csv")
df_topYoutuber = spark.read.format("csv").option("header", "true").schema(schematopYoutuber).load("abfss://
data-bronze@bigdataprojetefrei.dfs.core.windows.net/topYoutubers_0/2025_01_05_1736095487527_0.csv")
df_youtubeTrendingUS = spark.read.format("csv").option("header", "true").schema(schemayoutubeTrendingUS).
load("abfss://data-bronze@bigdataprojetefrei.dfs.core.windows.net/youtubeTrendingUs_0/
2025_01_05_1736095496297_0.csv")

```

- Utilisation des dataframes pour créer des fichiers parquet

```

▶ 01:46 PM (32s) 4 Python
df_globalStats.write.mode("overwrite").parquet("/bronze/parquet/globalStats")
df_topYoutuber.write.mode("overwrite").parquet("/bronze/parquet/topYoutuber")
df_youtubeTrendingUS.write.mode("overwrite").parquet("/bronze/parquet/youtubeTrendingUS")

```

- Créer la database bronze

```

▶ 01:47 PM (<1s) 8 Python
spark.sql("CREATE DATABASE IF NOT EXISTS bronze")

DataFrame[]

```

- Créer les tables avec les données correspondantes

```

df_parquet_globalStats.withColumnRenamed("video views", "video_views").withColumnRenamed("Gross tertiary
education enrollment (%)", "gross_tertiary_education_enrollment").withColumnRenamed("Unemployment rate",
"Unemployment_rate").writeTo("bigdataefreidavid.bronze.globalStats").createOrReplace()

df_parquet_topYoutuber.withColumnRenamed("video views", "video_views").withColumnRenamed("Video Count",
"Video_Count").writeTo("bigdataefreidavid.bronze.topYoutuber").createOrReplace()

df_parquet_youtubeTrendingUS.writeTo("bigdataefreidavid.bronze.youtubeTrendingUS").createOrReplace()

```

- Création des données Silver : Dans cette couche, les données brutes de la couche Bronze sont nettoyées, dédoublées et transformées pour créer une vue cohérente et conforme des données. Les transformations appliquées ici sont généralement légères et visent à rendre les données prêtes pour une analyse plus approfondie.

Les étapes du traitement de la couche silver:

globalStats :

```

global_stats_df_clean = (df_globalStats
    .withColumn("gross_tertiary_education_enrollment",
        F.when(F.col("gross_tertiary_education_enrollment") == "nan", None)
        .otherwise(F.col("gross_tertiary_education_enrollment").cast("float")))
    .withColumn("Latitude",
        F.when(F.col("Latitude") == "nan", None)
        .otherwise(F.col("Latitude").cast("float")))
    .withColumn("Longitude",
        F.when(F.col("Longitude") == "nan", None)
        .otherwise(F.col("Longitude").cast("float")))
    .withColumn("Population",
        F.when(F.col("Population") == "nan", None)
        .otherwise(F.col("Population").cast("int")))
    .withColumn("Unemployment_rate",
        F.when(F.col("Unemployment_rate") == "nan", None)
        .otherwise(F.col("Unemployment_rate").cast("float")))
    .withColumn("Urban_population",
        F.when(F.col("Urban_population") == "nan", None)
        .otherwise(F.col("Urban_population").cast("float")))
    .withColumn("channel_type_rank",
        F.when(F.col("channel_type_rank") == "nan", None)
        .otherwise(F.col("channel_type_rank").cast("int")))
    .withColumn("country_rank",
        F.when(F.col("country_rank") == "nan", None)
        .otherwise(F.col("country_rank").cast("int")))
    .withColumn("highest_monthly_earnings",
        F.when(F.col("highest_monthly_earnings") == "nan", None)
        .otherwise(F.col("highest_monthly_earnings").cast("float")))
    .filter(F.col("Population").isNull() & F.col("Youtuber").isNull()))

global_stats_df_clean.createOrReplaceTempView("global_stats")

```

Pour le dataset globalStats, plusieurs transformations ont été appliquées afin de préparer les données pour l'analyse.

Nous avons d'abord utilisé la fonction when pour vérifier les valeurs "nan" dans chaque colonne. Lorsque nous avons rencontré des valeurs "nan", elles ont été remplacées par None, ce qui permet de gérer les données manquantes de manière efficace. Par exemple, pour la colonne gross_tertiary_education_enrollment, si une valeur était "nan", elle a été remplacée par None, sinon elle a été convertie en type float. Cette opération a été réalisée sur d'autres colonnes comme Latitude, Longitude, Urban_population, et highest_monthly_earnings.

Ensuite, nous avons converti certaines colonnes en types de données appropriés : les colonnes gross_tertiary_education_enrollment, Latitude, Longitude, Urban_population, et highest_monthly_earnings ont été converties en float, tandis que les colonnes Population, Unemployment_rate, channel_type_rank, et country_rank ont été converties en int.

Enfin, nous avons appliqué un filtre pour éliminer les lignes où les colonnes Population et Youtuber contenaient des valeurs nulles. Cela nous a permis de ne conserver que les données complètes et valides pour l'analyse.

Une fois ces étapes terminées, une vue temporaire appelée global_stats a été créée à partir du DataFrame nettoyé, permettant son utilisation dans d'autres traitements ou analyses.

En résumé, ces transformations ont permis de préparer les données en éliminant les valeurs manquantes, en effectuant les conversions nécessaires et en filtrant les enregistrements incomplets.

topYoutubers



```
topyoutuber_df_clean = (df_topYoutuber
    .withColumn("Rank",
        F.when(F.col("Rank") == "nan", None)
        .otherwise(F.col("Rank").cast("int")))
    .withColumn("Started", F.col("Started").cast("int"))
    .withColumn("Subscribers",
        F.when(F.col("Subscribers") == "nan", None)
        .otherwise(F.col("Subscribers").cast("long")))
    .withColumn("Video_Count",
        F.when(F.col("Video_Count") == "nan", None)
        .otherwise(F.col("Video_Count").cast("int")))
    .withColumn("video_views",
        F.when(F.col("video_views") == "nan", None)
        .otherwise(F.col("video_views").cast("long")))
    .filter(F.col("Youtuber").isNotNull() & F.col("Rank").isNotNull())
)

topyoutuber_df_clean.createOrReplaceTempView("topyoutuber")
```

Générer (⌘ +)

▶ topyoutuber_df: pyspark.sql.dataframe.DataFrame = [_airbyte_ab_id: string, _airbyte_emitted_at: string ... 7 autres champs]
▶ topyoutuber_df_clean: pyspark.sql.dataframe.DataFrame = [_airbyte_ab_id: string, _airbyte_emitted_at: string ... 7 autres champs]

Pour le dataset topYoutubers, plusieurs transformations ont été appliquées afin de nettoyer les données et les préparer pour l'analyse.

Tout d'abord, nous avons traité la colonne Rank en vérifiant si la valeur était égale à "nan". Si c'était le cas, la valeur a été remplacée par None. Sinon, la valeur a été convertie en type int. Cette opération a été réalisée de manière similaire pour les colonnes Subscribers, Video_Count, et video_views, où les valeurs "nan" ont été remplacées par None, puis chaque colonne a été convertie dans le type de données approprié : long pour Subscribers et video_views, et int pour Rank et Video_Count.

Ensuite, la colonne Started a été convertie en type int pour uniformiser les données.

Enfin, un filtrage a été appliqué pour exclure les lignes où les colonnes Youtuber ou Rank contiennent des valeurs nulles. Cela permet de s'assurer que seules les données complètes et valides soient conservées.

Une fois toutes ces transformations effectuées, une vue temporaire appelée topyoutuber a été créée à partir du DataFrame nettoyé, ce qui permet son utilisation dans des analyses ou traitements futurs.

En résumé, les étapes ont permis de nettoyer les colonnes contenant des valeurs "nan", de convertir les types de données aux formats appropriés, puis de filtrer les enregistrements incomplets.

Youtube Trending Us :

```
youtubetrendingus_df_clean = (df_youtubeTrendingUS
    .withColumn("comment_count",
        F.when(F.col("comment_count") == "nan", None)
        .otherwise(F.col("comment_count").cast("long")))
    .withColumn("likes",
        F.when(F.col("likes") == "nan", None)
        .otherwise(F.col("likes").cast("long")))
    .withColumn("dislikes",
        F.when(F.col("dislikes") == "nan", None)
        .otherwise(F.col("dislikes").cast("long")))
    .withColumn("view_count",
        F.when(F.col("view_count") == "nan", None)
        .otherwise(F.col("view_count").cast("long")))
    .withColumn("comments_disabled",
        F.when(F.col("comments_disabled") == "True", True)
        .otherwise(False))
    .withColumn("ratings_disabled",
        F.when(F.col("ratings_disabled") == "True", True)
        .otherwise(False))
    .filter(F.col("video_id").isNotNull() &
        F.col("title").isNotNull() &
        F.col("view_count").isNotNull())
)
```

Pour le dataset youtubetrendingus, plusieurs transformations ont été appliquées afin de nettoyer et préparer les données pour une analyse approfondie.

Tout d'abord, les colonnes comment_count, likes, dislikes, et view_count ont été nettoyées en vérifiant si les valeurs étaient égales à "nan". Si c'était le cas, elles ont été remplacées par None (valeur nulle). Sinon, les valeurs ont été converties au type long pour correspondre aux valeurs numériques attendues.

Ensuite, les colonnes `comments_disabled` et `ratings_disabled`, qui contiennent des valeurs de type chaîne, ont été transformées en valeurs booléennes (True ou False). Si la colonne contient la valeur "True", elle est remplacée par True; dans le cas contraire, la valeur est définie sur False.

Enfin, un filtre a été appliqué pour éliminer les lignes où les colonnes `video_id`, `title`, ou `view_count` contiennent des valeurs nulles (None). Cela permet de ne conserver que les enregistrements complets et valides pour l'analyse.

Une fois ces transformations terminées, une vue temporaire nommée `youtubetrendingus` a été créée à partir du DataFrame nettoyé, rendant les données prêtes à être utilisées dans des traitements ou des analyses ultérieures.

En résumé, ces étapes ont consisté à nettoyer les données en remplaçant les valeurs "nan" par None, en convertissant les colonnes aux types appropriés, en transformant les colonnes booléennes et en filtrant les enregistrements incomplets.

- Création des données Gold : La couche Gold contient des données enrichies et agrégées, prêtes à être utilisées par les utilisateurs finaux. Pour notre projet, la couche gold sera celle que nous utiliserons pour établir notre analyse.

Pour cela nous avons créé un notebook, qui permet de créer cette data, et de l'enregistrer dans un folder défini

```
spark.sql("""
CREATE DATABASE IF NOT EXISTS gold
LOCATION '{gold_path}'
""")

# Création de la vue temporaire gold_video_data
gold_video_data_df = spark.sql("""
SELECT
    video.video_id,
    video.channelId,
    video.channelTitle,
    video.title,
    video.description,
    video.tags,
    SIZE(SPLIT(video.tags, '\\\\|')) AS tag_count,
    video.categoryId,
    CAST(video.view_count AS BIGINT) AS view_count,
    CAST(video.likes AS BIGINT) AS likes,
    CAST(video.dislikes AS BIGINT) AS dislikes,
    CAST(video.comment_count AS BIGINT) AS comment_count,
    LENGTH(video.description) AS description_length,
    CASE
        WHEN video.view_count > 0 THEN video.likes / video.view_count
        ELSE 0
    END AS like_to_view_ratio,
    CASE
        WHEN video.view_count > 0 THEN video.dislikes / video.view_count
        ELSE 0
    END AS dislike_to_view_ratio,
    CASE
        WHEN video.comment_count > 0 THEN video.comment_count / video.view_count
        ELSE 0
    END AS comment_to_view_ratio,
    video.comments_disabled,
    video.ratings_disabled,
    youtube.Subscribers AS channel_subscribers,
    youtube.Video_Count AS channel_video_count,
    youtube.video_views AS channel_total_views
FROM silver.youtubetrendingus video
JOIN silver.topyoutube youtube
    ON video.channelTitle = youtube.Youtuber
WHERE video.video_id IS NOT NULL
    AND video.view_count IS NOT NULL
    AND EXISTS(SELECT 1 FROM silver.topyoutube youtube WHERE video.channelTitle = youtube.Youtuber);
""")
```

Ce script crée ou remplace selon le cas une vue temporaire appelée `gold_video_data`. Cette vue contient des données transformées, extraites de la table `silver.youtubetrendings` et jointes avec la table `silver.topyoutube` basée sur la correspondance entre le champ `channelTitle` de la table `video` et le champ `Youtuber` de la table `youtube`.

Dans la requête `SELECT`, plusieurs colonnes sont sélectionnées et des transformations ou calculs sont effectués sur les données :

- `tags` : Compte du nombre de tags associés à chaque vidéo.

- view_count, likes, dislikes, comment_count : Conversion des valeurs des colonnes en entiers (CAST(... AS BIGINT)).
- Ratios : plusieurs ratios sont calculés pour pouvoir les analyser
 - Ratio like_to_view_ratio : Nombre de likes divisé par le nombre de vues.
 - Ratio dislike_to_view_ratio : Nombre de dislikes divisé par le nombre de vues.
 - Ratio comment_to_view_ratio : Nombre de commentaires divisé par le nombre de vues.
- Autres informations : La longueur de la description de la vidéo est calculée (LENGTH(video.description) AS description_length).

La table silver.topyoutubeur est jointe avec la table des vidéos, permettant d'ajouter des informations supplémentaires sur chaque YouTubeur, telles que :

- channel_subscribers : Nombre d'abonnés du YouTubeur.
- channel_video_count : Nombre de vidéos publiées par le YouTubeur.
- channel_total_views : Nombre total de vues du YouTubeur.

4. Filtrage des données :

La clause WHERE filtre les enregistrements de la table video où :

- video_id IS NOT NULL : Seules les vidéos avec un identifiant valide sont prises en compte.
- view_count IS NOT NULL : Seules les vidéos ayant un nombre de vues valide sont conservées.
- EXISTS : La condition EXISTS vérifie si le YouTubeur lié à la vidéo existe dans la table silver.topyoutubeur.

Cette vue temporaire est ensuite enregistrée dans “/mnt/data-gold”

la deuxième donnée gold :

```
14:08 (9s) 1 Python

gold_path = "/mnt/data-gold"

spark.sql("""
CREATE DATABASE IF NOT EXISTS gold
LOCATION '{gold_path}'
""")

global_recommendation_analysis_df = spark.sql("""
SELECT
    r.video_id,
    r.title AS video_title,
    r.channelTitle AS channel_name,
    r.categoryId,
    r.tags,
    r.view_count AS recommendation_views,
    r.likes AS recommendation_likes,
    r.comment_count AS recommendation_comments,
    g.Country AS country,
    g.Population AS country_population,
    g.Unemployment_rate AS country_unemployment_rate,
    g.gross_tertiary_education_enrollment AS education_enrollment_rate,
    CASE
        WHEN r.view_count > 0 THEN r.likes / r.view_count
        ELSE 0
    END AS like_to_view_ratio,
    CASE
        WHEN r.view_count > 0 THEN r.comment_count / r.view_count
        ELSE 0
    END AS comment_to_view_ratio
FROM
    silver.youtubetrendingus r
JOIN
    silver.globalstats g
ON r.categoryId = g.category
WHERE
    r.video_id IS NOT NULL;
""")

global_recommendation_analysis_df.write.format("delta").mode("overwrite").saveAsTable("gold.gold_recommendation_analysis")

6 jobs Spark
```

Modèle Conceptuels des Données

Cette requête sélectionne plusieurs colonnes issues de deux tables : silver.youtubetrendingus (alias r) et silver.globalstats (alias g).

Les colonnes extraites de la table youtubetrendingus incluent :

- video_id : L'identifiant unique de la vidéo.
- title : Le titre de la vidéo (alias video_title).
- channelTitle : Le nom de la chaîne (alias channel_name).
- categoryId : L'identifiant de la catégorie de la vidéo.
- tags : Les tags associés à la vidéo.
- view_count : Le nombre de vues de la vidéo (alias recommendation_views).
- likes : Le nombre de likes de la vidéo (alias recommendation_likes).
- comment_count : Le nombre de commentaires de la vidéo (alias recommendation_comments).

Les colonnes extraites de la table globalstats incluent :

- Country : Le nom du pays associé à la catégorie de la vidéo.

- Population : La population du pays (alias country_population).
- Unemployment_rate : Le taux de chômage dans le pays (alias country_unemployment_rate).
- gross_tertiary_education_enrollment : Le taux d'inscription dans l'enseignement supérieur du pays (alias education_enrollment_rate).

2. Calcul des ratios :

Deux nouveaux ratios sont calculés dans cette requête :

- like_to_view_ratio : Le ratio des likes par rapport aux vues de la vidéo. Si le nombre de vues est supérieur à zéro, il calcule le ratio $\text{likes} / \text{view_count}$. Sinon, il retourne zéro.
- comment_to_view_ratio : Le ratio des commentaires par rapport aux vues de la vidéo. Si le nombre de vues est supérieur à zéro, il calcule le ratio $\text{comment_count} / \text{view_count}$. Sinon, il retourne zéro.

Ces ratios permettent de mesurer l'engagement des spectateurs vis-à-vis de la vidéo en termes de likes et de commentaires par rapport au nombre de vues.