

TP — Pipeline PySpark ETL (IMDb Datasets)

Cours Data Warehousing — MSc Data Engineer

Université / École

8 janvier 2026

Plan

- 1 Dataset et objectif
- 2 Extraction et Staging
- 3 DW Core : Dimensions et faits
- 4 Marts BI et export
- 5 Consignes et évaluation

Dataset utilisé : IMDb Datasets (officiel)

Source officielle :

<https://datasets.imdbws.com/>

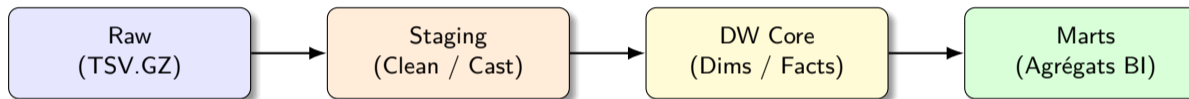
Tables utiles (TSV compressés) :

- `title.basics.tsv.gz` : titres (type, année, durée, genres)
- `title.ratings.tsv.gz` : note moyenne + nombre de votes
- (optionnel) `name.basics.tsv.gz`, `title.principals.tsv.gz`

Objectif du TP :

- Construire un pipeline PySpark complet : Raw → Staging → DW → Marts
- Produire des agrégats BI : top films par année / genre, volumes de votes, distributions de notes

Vue globale : Raw \rightarrow Staging \rightarrow DW \rightarrow Marts

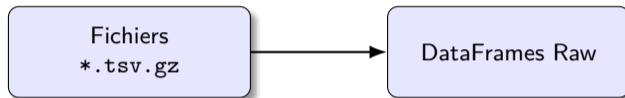


Livrable final : des dossiers Parquet partitionnés (DW + marts).

Plan

- 1 Dataset et objectif
- 2 Extraction et Staging**
- 3 DW Core : Dimensions et faits
- 4 Marts BI et export
- 5 Consignes et évaluation

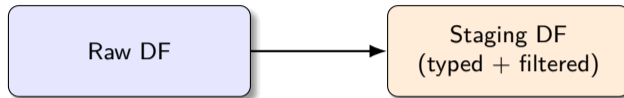
Étape 1 : Extraction (Raw Zone)



```
titlesraw = (spark.read.option("sep", "\t").option("header", "true").csv("raw/title.basics.tsv.gz"))  
ratingsraw = (spark.read.option("sep", "\t").option("header", "true").csv("raw/title.ratings.tsv.gz"))
```

Note : les valeurs manquantes IMDb sont souvent codées \N.

Étape 2 : Staging (Nettoyage / Cast / Filtrage)

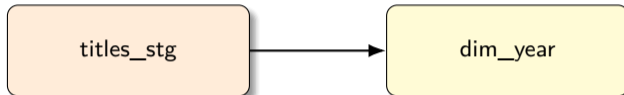


```
from pyspark.sql.functions import col, when
titlesstg = (titlesraw.withColumn("startYear", when(col("startYear") == "
N", None).otherwise(col("startYear")).cast("int")).withColumn("runtimeMinutes", when(col("runtimeM
N", None).otherwise(col("runtimeMinutes")).cast("int")).withColumn("isAdult", col("isAdult").cast("i
"movie").dropDuplicates(["tconst"])))
ratingsstg = (ratingsraw.withColumn("averageRating", col("averageRating").cast("double")).withColumn
```

Plan

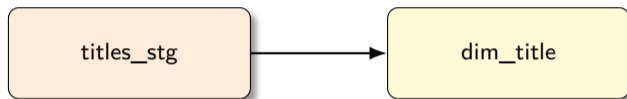
- 1 Dataset et objectif
- 2 Extraction et Staging
- 3 DW Core : Dimensions et faits**
- 4 Marts BI et export
- 5 Consignes et évaluation

Étape 3 : Dimension dim_year



```
from pyspark.sql.functions import lit
dim_year = (titles_stg.select(col("startYear").alias("year")).where(col("year").isNotNull()).dropDuplicates())
```

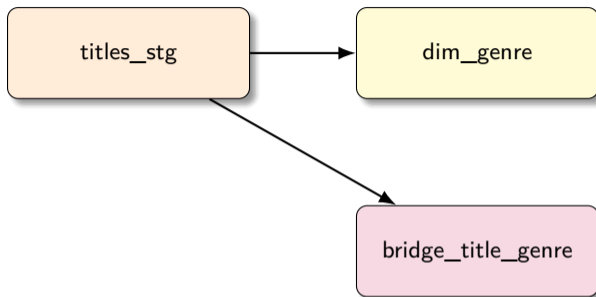
Étape 4 : Dimension `dim_title`



```
dim_title = (titles_stg.select(col("tconst").alias("title_key"), "primaryTitle", "originalTitle", "titleType", col("titleKey").alias("titleKey")))
```

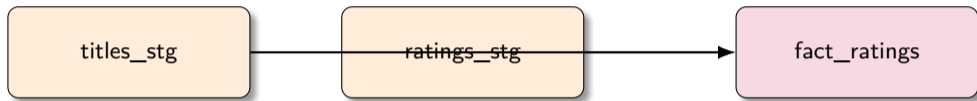
Remarque : ici, `title_key = tconst` (clé stable, pratique en TP).

Étape 5 : dim_genre + table de pont (N :N)



```
from pyspark.sql.functions import split, explode, trim, lower
title_genres = (titles_stg.select(col("tconst").alias("title_key"), when(col("genres") == "
N", None).otherwise(col("genres")).alias("genres")).where(col("genres").isNotNull()).withColumn("ge
dim_genre = (title_genres.select("genre").dropDuplicates().withColumn("genre_key", col("genre")).select(
bridge_title_genre = (title_genres.join(dim_genre, "genre", "inner").select("title_key", "genre_key").dropDuplic
```

Étape 6 : Table de faits fact_ratings



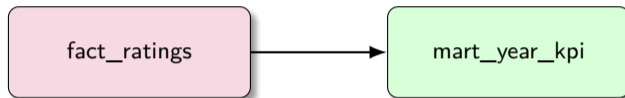
```
from pyspark.sql.functions import coalesce
fact_ratings = (titles_stg.join(ratings_stg, "tconst", "inner").select(col("tconst").alias("title_key"), col("sta
```

Mesures : avg_rating, num_votes, runtime_min.

Plan

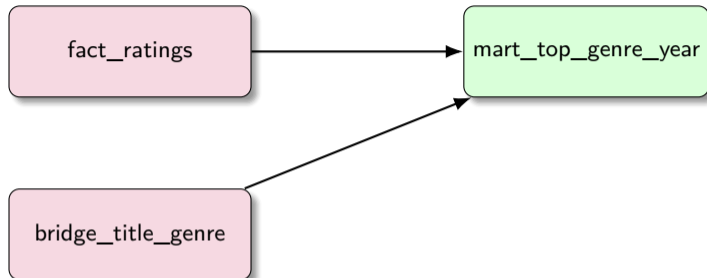
- 1 Dataset et objectif
- 2 Extraction et Staging
- 3 DW Core : Dimensions et faits
- 4 Marts BI et export**
- 5 Consignes et évaluation

Étape 7 : Mart — KPI par année



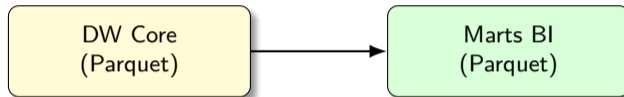
```
from pyspark.sql.functions import avg, sum as fsum, count
mart_year_kpi = (fact_ratings.groupBy("year_key").agg(count("*").alias("n_movies"), avg("avg_rating").alias("mean_rating"), fsum("num_votes").alias("total_votes")))
```

Étape 7 (bis) : Mart — Top films par genre et année



```
from pyspark.sql.window import Window from pyspark.sql.functions import desc, row_number
joined = (fact_ratings.join(bridge_title_genre, "title_key", "inner"))
w = Window.partitionBy("year_key", "genre_key").orderBy(desc("num_votes"))
mart_top_genre_year = (joined.withColumn("rk", row_number().over(w)).where(col("rk") <=
10).select("year_key", "genre_key", "title_key", "avg_rating", "num_votes"))
```

Étape 8 : Export (Parquet + partitionnement)



```
dimyear.write.mode("overwrite").parquet("dw/dimyear")dimtitle.write.mode("overwrite").parquet("dw/dimtitle")  
(factratings.write.mode("overwrite").partitionBy("yearkey").parquet("dw/factratings"))  
martyearkey.write.mode("overwrite").parquet("marts/martyearkey")marttopgenre.write.mode("overwrite").parquet("marts/marttopgenre")
```

Plan

- 1 Dataset et objectif
- 2 Extraction et Staging
- 3 DW Core : Dimensions et faits
- 4 Marts BI et export
- 5 Consignes et évaluation

Travail demandé et critères d'évaluation

Livrables :

- Code PySpark (notebook ou scripts) + arborescence raw/, dw/, marts/
- Exports Parquet : dim_year, dim_title, dim_genre, fact_ratings, marts
- Mini-rapport PDF : choix de nettoyage, clés, schéma, tests qualité
- Bonus : visualisation (PowerBI / Superset / matplotlib) sur mart_year_kpi

Barème (exemple) :

- 40% pipeline complet + robustesse (nulls, \N, déduplication)
- 40% modélisation DW (dims/facts cohérents, clés, jointures)
- 20% marts + analyse + présentation