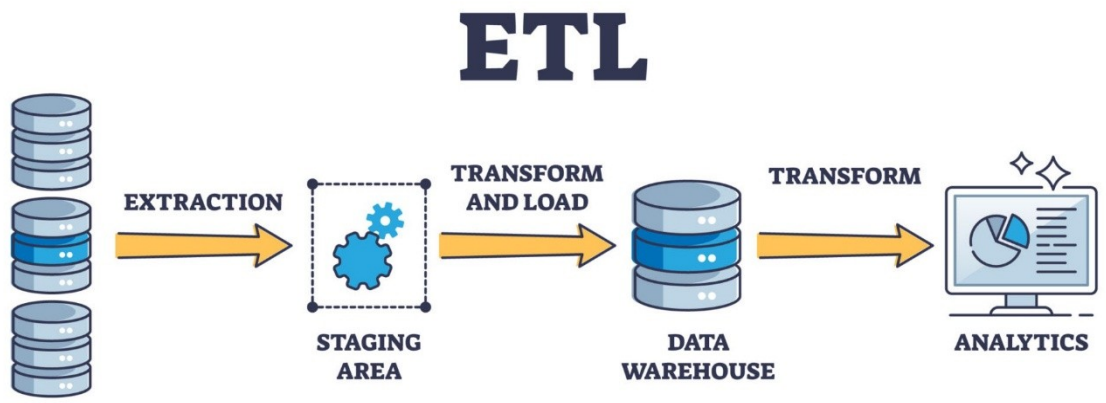
TD Spark

🎬 **TD PySpark : ETL et Analyse de Données IMDb pour un Festival de Films Historiques**
Ce TD utilise les datasets publics d'IMDb (https://developer.imdb.com/non-commercial-datasets/) pour simuler un processus ETL (Extract, Transform, Load) avec PySpark. L'objectif final est de préparer un jeu de données propre et analysé pour la sélection de films historiques pour un festival.



Getty Images

1. Initialisation et ETL Préliminaire (Extract & Transform)
Le processus ETL commence par l'extraction des données brutes et leur transformation initiale pour les rendre utilisables. Nous allons nous concentrer sur les fichiers title.basics.tsv.gz, title.ratings.tsv.gz, et name.basics.tsv.gz / title.principals.tsv.gz.

Prérequis :

- Installation de Python, Java, et PySpark.
- Téléchargement des fichiers .tsv.gz des datasets IMDb.

Exercice 1.1 : Chargement des Données

1. Initialisez une session Spark.
2. Chargez les datasets suivants dans des DataFrames Spark en gérant le format TSV (séparateur \t) et les valeurs manquantes (représentées par \N).
    - title.basics.tsv.gz $\rightarrow$ df_basics
    - title.ratings.tsv.gz $\rightarrow$ df_ratings
    - name.basics.tsv.gz $\rightarrow$ df_names (pour les noms d'acteurs)
    - title.principals.tsv.gz $\rightarrow$ df_principals (pour les relations film-acteur)
3. Affichez le schéma (printSchema()) et les 5 premières lignes (show(5)) pour chaque DataFrame.

Exercice 1.2 : Nettoyage et Préparation (Focus "Historique")
Pour notre festival de films historiques, nous avons besoin de données propres et pertinentes.

1. **Nettoyage des Dates :** Dans df_basics, remplacez les valeurs \N dans la colonne startYear par NULL et convertissez cette colonne en type **Integer**.
2. **Filtrage du Type :** Filtrez df_basics pour ne conserver que les entrées de type 'movie' et 'tvMovie' dans une nouvelle DataFrame df_movies.
3. **Filtrage du Genre (Focus Historique) :** Ajoutez un filtre sur df_movies pour ne conserver que les films dont la colonne genres contient le terme 'History' ou 'Documentary'. Nommez ce nouveau DataFrame df_historical_selection.
4. **Conversion en Vue SQL :** Créez une vue temporaire SQL pour df_historical_selection nommée historical_films.

2. Analyse des Titres et Transformation (Exploration & Load)

Cette section utilise les DataFrames pour répondre aux questions d'analyse.

Exercice 2.1 : Statistiques Générales sur les Titres

Utilisez la vue SQL historical_films ou le DataFrame df_historical_selection :

1. **Nombre Total :** Trouvez le nombre total de titres dans la sélection historique.
2. **Titres par Type :** Trouvez le nombre de titres par titleType (devrait être majoritairement 'movie' et 'tvMovie' après le filtrage).

Exercice 2.2 : Exploration des Genres

1. **Popularité des Genres :** En utilisant la colonne genres dans df_historical_selection (qui est une chaîne de caractères séparée par des virgules) :
    - **Transformation :** Séparez les genres de chaque ligne pour les mettre sur des lignes séparées (utilisez la fonction Spark SQL explode avec split).
    - **Agrégation :** Trouvez les **10 genres les plus populaires** (en comptant l'occurrence de chaque genre) dans notre sélection historique.

Exercice 2.3 : Longueur des Titres

1. **Longueurs :** Calculez la longueur de la colonne primaryTitle pour chaque film.
2. **Top/Flop Longueur :**
    - Trouver les **5 titres les plus longs**.
    - Trouver les **5 titres les plus courts**.

3. Jointures et Requêtes Spécifiques (Préparation pour la Sélection)

Pour affiner la sélection du festival, nous allons joindre les données de base avec les notes (ratings) et les informations sur les acteurs.

Exercice 3.1 : Jointure Basique (Films et Ratings)

1. **Jointure :** Joignez df_historical_selection avec df_ratings en utilisant la colonne commune tconst avec une jointure interne (inner). Nommez le résultat df_joined_ratings.
2. **Conversion en Vue SQL :** Créez une vue temporaire SQL nommée historical_rated_films.

Exercice 3.2 : Requêtes Basées sur les Notes

Utilisez la vue SQL historical_rated_films :

1. **Les Parfaits (Note 10) :**
    - Rechercher les **5 films historiques** ayant un averageRating de **10.0** et qui ont au moins **100 votes** (numVotes).
2. **Les Sélections Intermédiaires (Années Spécifiques) :**
    - Rechercher tous les films historiques ayant un score (averageRating) **entre 5 et 6 (exclus)** et dont l'année de début (startYear) est **entre 1980 et 1999 (inclus)**.

Exercice 3.3 : Jointure Triple (Films, Acteurs et Ratings)

Pour identifier les acteurs clés dans notre sélection :

1. **Triple Jointure :** Réalisez une jointure enchaînée :
    - df_joined_ratings (votre jointure historique film-rating)
    - Jointure avec df_principals sur tconst.
    - Jointure avec df_names sur la colonne d'identifiant de la personne (nconst dans df_principals et nconst dans df_names).
    - Filtrez df_principals pour ne retenir que les lignes où category est 'actor' ou 'actress'.
    - Nommez le résultat df_actor_ratings.
2. **Conversion en Vue SQL :** Créez une vue temporaire SQL nommée historical_actors_ratings.

Exercice 3.4 : Requêtes Basées sur les Acteurs

Utilisez la vue SQL historical_actors_ratings :

1. **Acteurs à Note 5 :**
    - Rechercher les **5 noms d'acteurs/actrices** (primaryName de df_names) qui ont joué dans le plus grand nombre de films historiques ayant exactement un rating de **5.0**.
2. **Recherche par Acteur :**
    - Rechercher tous les films (titre et année) avec l'acteur **'Ben Affleck'** (utilisez son nom exact) qui ont obtenu un averageRating **entre 5 et 10 (inclus)**. Triez par année décroissante.

Voulez-vous que je développe le code PySpark pour la première partie de ces exercices (Chargement des Données et Nettoyage de Base) ou préférez-vous commencer par les requêtes SQL/DataFrames ?