Flow and Structure

hdfs dfs -mkdir -p /user/tejashree/project/data/raw

hdfs dfs -mkdir -p /user/tejashree/project/data/processed

hdfs dfs -mkdir -p /user/tejashree/project/data/mappings

hdfs dfs -mkdir -p /user/tejashree/project/models/recommendation

hdfs dfs -mkdir -p /user/tejashree/project/models/sentiment

hdfs dfs -mkdir -p /user/tejashree/project/outputs

project/

├── data/

│ ├── raw/

│ │ └── steam\_reviews.csv

│ ├── processed/

│ │ ├── cleaned\_steam\_reviews.parquet

│ │ ├── steam\_review\_english.parquet

│ └── mappings/

│ ├── author\_mapping.parquet

│ └── games\_mapping.parquet

│

├── models/

│ ├── recommendation/

│ │ ├── als\_game\_recommendation\_model/

│ │ └── test\_als\_game\_recommendation\_model/

│ └── sentiment/

│ ├── sentiment\_model/

│ └── sentiment\_model\_with\_bigrams/

│

└── outputs/

└── user\_recommendations.parquet

A screenshot of a computer

Description automatically generated

Sample flow diagram:

A diagram of a software development

Description automatically generatedA diagram of data cleaning

Description automatically generated

**1. Data Loading and Cleaning**

**1. Column Name Standardization**

* Replaced dots (.) in column names with underscores (\_) to make them valid identifiers.

**2. Missing Value Handling**

* **Dropped rows** where any of the following *critical fields* were missing:  
  app\_id, app\_name, review\_id, review, recommended, author\_steamid
* **Filled NULLs** in numeric columns (votes\_helpful, votes\_funny, comment\_count, weighted\_vote\_score) with 0.
* **Filled NULLs** in categorical fields like steam\_purchase and received\_for\_free with 'unknown'.

**3. Text Normalization (Review Cleaning)**

* Converted all review text to lowercase.
* Removed:
  + HTML tags
  + Emojis and non-text symbols
  + Anything that isn't a Unicode letter or whitespace
* Trimmed whitespace from the cleaned reviews.

**4. Type Conversion**

* Explicitly casted columns to appropriate data types:
  + app\_id → Integer
  + review\_id, author\_steamid → Long
  + recommended, steam\_purchase → Boolean
  + author\_num\_games\_owned, author\_num\_reviews → Integer
  + author\_playtime\_forever, author\_playtime\_at\_review → Float
* Converted timestamp columns (timestamp\_created, author\_last\_played, timestamp\_updated) to proper **timestamp** data type using from\_unixtime.

**5. Content-Based Filtering**

* Filtered out rows where:
  + app\_name, review, or cleaned\_review consisted **only of numbers** (non-informative or noise).

**6. Saved the Cleaned Dataset**

* Saved the cleaned DataFrame as a **Parquet file** to HDFS:  
  /user/tejashree/project/data/processed/cleaned\_steam\_reviews.parquet

**7. Filtered for English-Language Reviews**

* Loaded cleaned data and filtered rows where language == "english".
* Saved the English-only subset as a Parquet file:  
  /user/tejashree/project/data/processed/steam\_review\_english.parquet

**2. Sentiment Analysis Model**

Predict sentiment (Positive or Negative) of each review.

**Steps:**

* **Load English-Only Data**:

english\_df = spark.read.parquet("/user/tejashree/project/english\_only\_reviews.parquet")

* **Prepare the Pipeline**:
  + **Tokenizer**: Split review into words.
  + **TF-IDF Vectorizer**: Convert text to numeric features.
  + **Logistic Regression**: Classify sentiment.

**3. Recommendation Engine (Collaborative Filtering)**

Recommend games based on user's previous activity.

1. **Load Cleaned Steam Reviews**  
   Import relevant columns (game ID, game name, user ID, playtime, recommendation flag).
2. **Filter Out Unpopular Games**  
   Remove games with fewer than 200 reviews to focus on commonly played titles.
3. **Remove Outliers in Playtime**  
   Exclude users with extremely high or low playtime to reduce noise.
4. **Identify Serious Players**  
   Select users who have played games at least 5× the average playtime and have reviewed more than one game.
5. **Convert Game Names and User IDs to Numeric Indices**  
   Use StringIndexer to map game names and user IDs to integers for model compatibility.
6. **Assign Ratings Based on Recommendation**  
   Convert the recommended boolean into numeric ratings: 5 for recommended, 1 otherwise.
7. **Train ALS Model (Collaborative Filtering)**  
   Use Spark’s ALS algorithm with:
   * userCol = app\_index (games)
   * itemCol = author\_index (users)
   * ratingCol = Rating
8. **Generate Top 5 Game Recommendations per Game**  
   ALS predicts the top 5 users for each game (or vice versa), effectively recommending similar games based on shared user preferences.
9. **Save Model and Recommendation Outputs**  
   Save the trained model, recommendations, and ID mappings for future use.
10. **Evaluate Model Performance**  
    Calculate RMSE using test data to evaluate how well predicted ratings match actual data.

**4. Prepare Data for Frontend**

Save processed data so Streamlit app can read.

* **Save sentiment predictions**
* **Save game recommendations**:

**5. Streamlit Frontend App**

**Gamesphere tab**

**- Steam ID input (author\_steamid)**

**- Map to author\_index**

**- Show Top-N game recommendation**

**- Show User Profile Summary**

**- Enable CSV download**

**EDA Dashboard (Interactive Tabs)**

**- Game/month filters**

**- Playtime and recommendation**

**- Trends and pie charts**

\

**Introduction to Steam Review Analysis Project**

**Objective:** The Steam Review Analysis project aims to explore and derive insights from a large dataset containing reviews of games on the Steam platform. Leveraging PySpark for data processing, Pandas for exploratory data analysis (EDA), and visualization libraries like Matplotlib and Seaborn, the project focuses on several key categories to uncover patterns, trends, and valuable information from the vast amount of Steam review data.

**Data Source:** The dataset, sourced from Kaggle ([Steam Reviews 2021](https://www.kaggle.com/datasets/najzeko/steam-reviews-2021)), comprises approximately 8 GB of data with 17 million rows and 23 columns. The dataset provides detailed information about individual game reviews, including attributes such as the review text, author details, game information, and more.

**Project Components:**

1. **Games Analysis:**
   * Explore the distribution of reviews among different games.
   * Identify the most reviewed and highest-rated games.
   * Analyze the trends in game reviews over time.
2. **Language Analysis:**
   * Examine the distribution of reviews across different languages.
   * Explore the sentiment of reviews in different languages.
   * Identify popular languages among Steam users.
3. **Time Analysis:**
   * Investigate how the volume of reviews changes over time.
   * Analyze temporal patterns in sentiment.
   * Identify peak times for reviews.
4. **Author Analysis:**
   * Explore characteristics of authors, such as the number of games owned and reviewed.
   * Identify prolific reviewers and their reviewing behavior.
   * Analyze playtime patterns of reviewers.
5. **Game Recommendation using ALS model:**
   * Alternating Least Squares matrix factorization model is implemented to recommend games to players who plays a game.
   * It is a popular collaborative filtering algorithm.
   * The model is trained using the author indices, app indices, and ratings.
   * We use the trained model to generate recommendations for all gamers.

**Technology Stack:**

* **Spark SQL:** Used for extraction, processing, and analysis of the data set.
* **Spark MLib:** Uses the ALS collaborative filtering algorithm for recommending games.
* **Pandas:** Employed for exploratory data analysis and manipulation.
* **Matplotlib and Seaborn:** Utilized for data visualization to derive meaningful insights.
* **Hadoop Distributed File System (HDFS):** Chosen for storing the extensive dataset.

**References:** The project drew inspiration and guidance from existing Kaggle kernels, particularly [Steam Reviews](https://www.kaggle.com/code/gonzafrancoandres/steam-reviews) and [Review Analysis by PySpark](https://www.kaggle.com/code/iplori/review-analysis-by-pyspark), for building upon best practices in working with Steam review data.

Through comprehensive analysis across these categories, the Steam Review Analysis project seeks to provide valuable insights for game developers, platform administrators, and researchers interested in understanding user sentiments and behaviors within the Steam gaming community.