Spotify Data Transformation and Analysis

Task Description: This task entails the ingestion, transformation, and analysis of Spotify sample datasets from Kaggle. The task involves using a Node.js script to clean up and organise these datasets and SQL for data analysis.

Requirements:

1. **Data Source:**
    - Take these Spotify sample datasets:
        - https://www.kaggle.com/datasets/yamaerenay/spotify-dataset-19212020-600k-tracks?select=artists.csv;
        - https://www.kaggle.com/datasets/yamaerenay/spotify-dataset-19212020-600k-tracks?select=tracks.csv.
2. **Data Transformation:**
    - Implement a Node.js script (use Typescript) to ingest the data from the source and transform it as follows:
        - Filter out records that meet specific criteria:
            1. Ignore the tracks that have no name;
            2. Ignore the tracks shorter than 1 minute;
            3. Load only these artists that have tracks after the filtering above.
        - Format and structure the data as needed for analysis:
            1. Explode track release date into separate columns: year, month, day.
            2. Transform track danceability into string values based on these intervals:
                a. [0; 0.5) assign ‘Low’;
                b. [0,5; 0.6] assign ‘Medium’;
                c. (0.6; 1] assign ‘High’.
3. **Data Storage:**
    - Store the cleaned and transformed data into AWS S3 as your storage solution.
    - Load data from S3 into locally hosted PostgreSQL.

4. **Data Processing:**
    - Create 3 SQL views that perform the following tasks on the data stored:
        - Return track id, name, popularity, energy, danceability (Low, Medium, High) and number of artist followers;
        - Return artist id, name, track id, name. Take only these tracks, which artists has followers;
        - Pick the most energising track of each release year. Return release year, track id, name and its energy.

Deliverables:

1. A Node.js script(s) for data ingestion and transformation, use Typescript..
2. Cover your Node.js based data transformation solution with unit tests. You can use Jest for this (or other testing framework/library).
3. SQL script(s) for data storage, processing.
4. Instructions on how to run the solution, including any configuration settings.
5. All source code is available in the GitHub repository.

Additional Information:

- You can choose specific technologies and services from the AWS ecosystem (e.g., AWS Lambda.) to complete the task, but be ready to justify your choices.
- Consider best practices for error handling, data validation in your implementation.
- Your solution should be well-structured, self-explanatory, and easily maintainable.
- If for some reason you won't be able to deliver some of the requirements or optional things in the form of code then please share your thoughts on how you would do that using pseudo code, comments etc.