MUSIC DATABASE ETL PROJECT

Advanced SQL Project

[Building a Music Media Database]

An End-to-End ETL Pipeline for Data Management in SQLite

Presented by Kasifa Bano

Overview & Agenda

Goal

"To design and build a relational database for a music media, and to implement a robust ETL pipeline to extract data from raw files, clean it using temporary tables, and load it into a final, structured database."

Agenda

- Problem Statement & Objectives
- Technology Stack
- Database Schema & Design
- The ETL Pipeline: From Raw Data to Clean Tables
- Data Analysis: Querying the Music Media
- Challenges & Solutions
- Conclusion & Next Steps

Projects Statement & Objectives

The Problem ❗

“Raw music data was disorganized and contained inconsistencies (null values, duplicates, incorrect formats), making it unreliable for analysis and application use."

Key Objectives 🎯

- Design a normalized schema for artists, albums, tracks, and genres.
- Automate data ingestion from source files into temporary SQLite tables using a shell script.
- Perform data cleaning and transformation directly within the database using SQL.
- Transfer the clean, validated data into permanent, structured tables.
- Incorporate and link a new data source (Media table).

Technology Stack

Database: SQLite

"Perfect for this project due to its serverless, file-based nature, making it easy to set up and manage."

Scripting: Shell Scripting

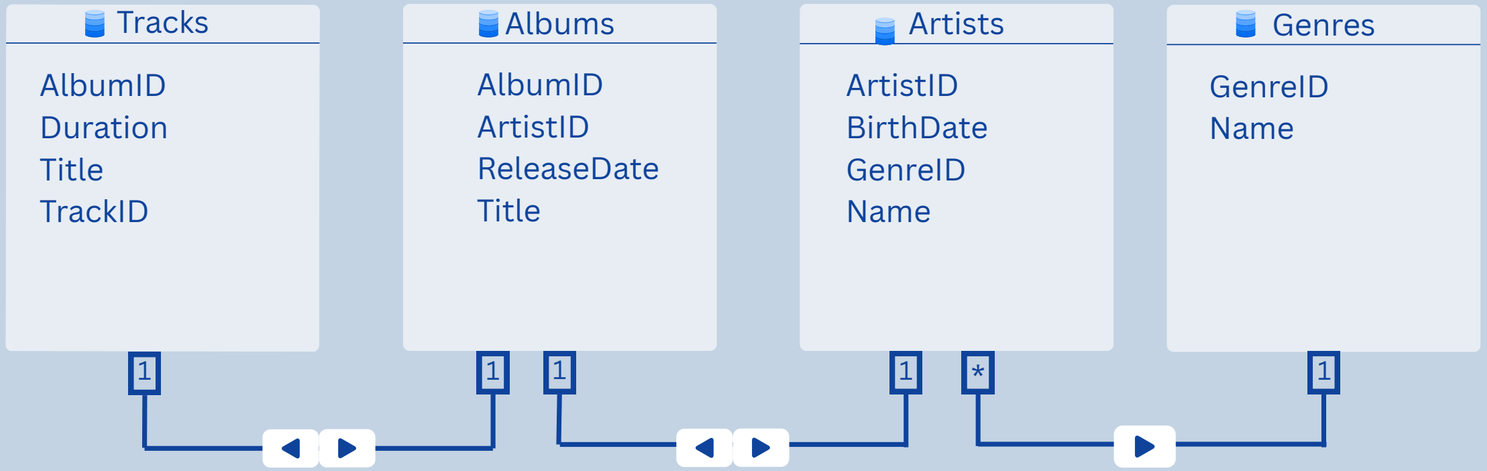
"Shell scripting is excellent for automating command-line tasks, and the sqlite3 CLI provides a powerful way to execute SQL commands and import data directly from the script."

Key Tools

- GitBash: To execute the etl_script.sh.
- sqlite3 Command-Line Interface (CLI): The core engine for interacting with the database from the script.

Database Design & Schema

Entity-Relationship Diagram (ERD)



Data Flow Overview

This schema serves as the final destination for our ETL pipeline. Raw data is first extracted from source files, transformed by cleaning inconsistencies and structuring the information, and then loaded into these normalized tables. This clean, relational structure allows for efficient and powerful SQL queries to analyze the music library.

Genres

Stores unique music genres.
GenreID: The unique identifier for the genre.
Name: The name of the genre (e.g., Rock, Jazz).

Artists

Stores individual song details and is linked to an album.
TrackID: The unique identifier for the track.
Title: The title of the track.
Duration: The length of the track.
AlbumID: Links the track to a specific album in the Albums table.

Albums

Stores unique artist information.
ArtistID: The unique identifier for the artist.
Name: The artist's name.
BirthDate: The artist's date of birth.
GenreID: Links the artist to a specific genre in the Genres table.

Tracks

Stores album details and is linked to an artist.
AlbumID: The unique identifier for the album.
Title: The title of the album.
ReleaseDate: The date the album was released.
ArtistID: Links the album to a specific artist in the Artists table.

The ETL Pipeline

Workflow

Raw Data Files

↓

etl_script.sh (run via GitBash)

↓

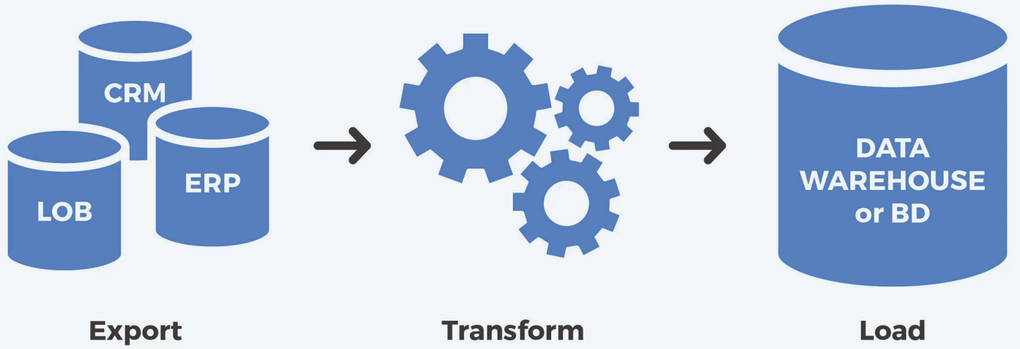
Load into Temporary Tables in SQLite

↓

Clean & Transform Data (SQL)

↓

Transfer to Final, Cleaned Tables



Key Features

This process uses temporary tables as a staging area, allowing us to clean and validate data without affecting the final, production-ready tables.

Step 1 & 2: Extract & Load to Staging

Automation

An etl_script.sh shell script was created to automate the entire process.

Initial Load

The script calls the sqlite3 CLI tool to import the raw data from files directly into a set of temporary tables(Artist_Temp, Album_Temp, etc.).

Process

Show a small snippet from your etl_script.sh file.
Example: sqlite3 music.db ".import --csv raw_data/artists.csv temp_Artists”
Mention that this entire step is initiated with a single command in the terminal: ./etl_script.sh

Step 3: Transformation (The Cleaning Process)

Data Cleaning Operations (in SQL)

- **Handling Duplicate**
- **Removing Nullss**
- **Correcting Data Types**

Used DELETE with subqueries or GROUP BY clauses to remove duplicate entries from temporary tables.
Identified and deleted rows with critical missing information (e.g., a track without a name).
Ensured columns like dates and track lengths were in a consistent, correct format.

Step 4: Loading the Final Tables

Process

Once the data in the temporary tables was cleaned, the script executed SQL commands to transfer the clean data into the final, permanent tables using INSERT INTO ... SELECT FROM.

Integrating New Data

The Media table was created and populated separately, then linked to the Tracks table, demonstrating the extensibility of the database design.

Data Analysis & Insights with SQL

Questions

What is the average track length for the“Rock” genre

List all tracks from a specific album“Thrill Seeker”

Find all albums released by a specific artist “Angela Green”

Queries

```sql
SELECT g.Name AS genre, AVG(t.Duration) AS average_duration
FROM Tracks AS t
JOIN Albums AS al ON t.AlbumID = al.AlbumID
JOIN Artists AS ar ON al.ArtistID = ar.ArtistID
JOIN Genres AS g ON ar.GenreID = g.GenreID
WHERE g.Name = 'Rock'
GROUP BY g.Name;
```

```sql
SELECT t.TrackID, t.Title AS track_title, t.Duration
FROM Tracks AS t
JOIN Albums AS al ON t.AlbumID = al.AlbumID
WHERE al.Title = 'Thrill Seeker';
```

```sql
SELECT al.Title AS album_title, al.ReleaseDate
FROM Albums AS al
JOIN Artists AS ar ON al.ArtistID = ar.ArtistID
WHERE ar.Name = 'Angela Green';
```

Output

	genre	average_duration
1	Rock	284.5

	TrackID	track_title	Duration
1	3	Thrill Seeker Theme	272

	album_title	ReleaseDate
1	Soul Keys	2019-07-02

Challenges & Solutions

Challenge 1

The initial process required running many SQL commands manually, which was error-prone.

Solution

Consolidated all steps into a single etl_script.sh. This makes the entire ETL process repeatable, consistent, and easy to execute.

Challenge 2

Ensuring the correct order of operations (e.g., you can't load a track for an album that doesn't exist yet).

Solution

Structured the commands within the shell script to load parent tables (Artists) before child tables (Albums), respecting foreign key constraints.

Conclusion

Summary of Achievements

- Successfully built a clean, normalized music database from messy raw files.
- Created a repeatable and robust ETL process using a shell script and a staging area for data cleaning.
- Demonstrated the database's value with practical SQL queries.

Thank You