



## Problem Statement: “Amazon Sonic – Launching a Unified Music & Creator Economy Platform”

Amazon is planning to launch **Amazon Sonic**, a new **music and creator-economy platform** that aims to compete with Spotify, YouTube Music, and SoundCloud in emerging markets such as India, Brazil, and Indonesia. The platform will not only allow users to **stream music** but also to **upload, monetize, and promote** their own tracks. In addition, it will integrate **Amazon Pay, Prime Video Music, and Alexa-based recommendations** to create a seamless multi-device listening experience.

The senior leadership has mandated the data engineering team to **design a scalable relational database system** that can support millions of daily active users, artists, and content transactions. As a student data analyst, you have been brought in to design the **complete end-to-end data model**, create **PostgreSQL DDL scripts**, and test the logic with **sample synthetic data**.

The new product, Amazon Sonic, will start with three subscription tiers—**Free (Ad-Supported), Prime Music (Standard), and Sonic Pro (Premium)**—each offering different streaming quality and ad-free limits. The system must capture **user demographics, devices used for playback, and listening patterns** such as daily active minutes, songs played, skips, and likes. Every user can follow artists, create playlists, and share songs with friends using in-app social features.

The platform also introduces a **creator marketplace**, allowing verified artists to upload original tracks, specify genre and mood tags, and link those tracks to albums, EPs, or podcasts. Artists will receive **royalty payments** based on stream counts, geographic reach, and ad impressions tied to their content. Labels and independent distributors will have dashboards to track earnings, pending payouts, and listener demographics.

To encourage engagement, Amazon Sonic will host **weekly challenges and top-charts**, ranking artists and songs based on algorithmic popularity. The database should therefore store **real-time stream logs, ranking metadata, and historical snapshots** to support trend analysis. Users can comment on songs, report inappropriate content, or rate them on a 1–5 scale, creating a continuous feedback loop for the recommendation engine.

Each user’s activity should be connected to **Amazon Pay** for transactions such as subscription renewals, tip-based donations to artists, and concert-ticket purchases. Payment failures, refunds, and disputes need to be recorded with audit trails. The system must also store **advertiser details**, since the Free tier is ad-supported and advertisers pay per thousand impressions. An advertiser can run multiple campaigns, each linked to a set of target demographics, regions, and time periods.

Additionally, the company plans to integrate **Alexa devices** as input sources. When a user plays a song through Alexa, the stream event must capture the **device ID, location, and voice**

**command** used. This data will be valuable for measuring cross-device engagement and identifying top devices driving subscriptions.

The customer support team needs structured data for all **tickets and complaints**, such as buffering issues, incorrect billing, or content removal requests. Each ticket should link back to the concerned user, subscription ID, and (if applicable) transaction ID. Resolution time and support-agent performance will later be part of operational KPIs.

The marketing analytics division has requested a schema to log **promotional campaigns, coupons, influencer collaborations, and referral programs**. They need to track how coupon codes convert into new subscriptions and measure ROI for each campaign channel (social media, email, push notifications, etc.). The analytics team also wants to run **A/B tests** for UI changes, so each experiment should be captured with test groups, control groups, metrics, and timestamps.

The database must be flexible enough to handle **multi-language content** (English, Hindi, Tamil, Portuguese, Bahasa Indonesia) and store localized metadata such as song title, lyrics, and translations. Every song can have multiple genre tags (e.g., “Pop”, “Electronic Dance”, “Indie”) and mood labels (“Happy”, “Workout”, “Chill”), requiring many-to-many relationships.

To maintain compliance with global data regulations (GDPR / CCPA), the system must log **consent versions, data-retention preferences**, and anonymization status for each user. Deletion requests must cascade across all linked tables, including playlists, comments, and transactions.

Performance monitoring is another priority. Amazon Sonic’s tech team will require database tables to record **API response times, server health checks, and batch-job execution logs**. These internal operations will help track uptime and troubleshoot issues affecting end-user experience.

In future releases, Amazon plans to launch **Sonic Live**, an event-streaming feature for live concerts. Each event will include ticketing, virtual attendance, live chat, and post-event recordings. Therefore, your database must be designed so that the same core entities—users, artists, transactions—can extend naturally into the live-events domain.

The system should support integration with Tableau / Power BI dashboards to analyze metrics such as **Average Revenue per User (ARPU), Monthly Active Users (MAU), Churn Rate, Artist Earnings, and Top Genres by Region**. Queries must be optimized with proper indexing and foreign-key relationships to handle large-scale joins efficiently.

The leadership also expects future use cases for machine-learning models, such as **user recommendation systems, fraud detection, and dynamic pricing**. To prepare for that, the schema must clearly separate **fact tables** (e.g., streams, transactions) and **dimension tables** (e.g., users, artists, songs, devices) to simplify ETL pipelines later.

Your deliverables include a detailed **ER diagram** with at least **20 entities**, a **PostgreSQL schema** with appropriate data types, constraints, and indexes, **sample data inserts** to

validate relationships, and **complex analytical queries** (such as top-streamed artists by month, revenue per subscription tier, churn prediction features, and advertiser performance).

Finally, you must provide a **data-verification script** that checks referential integrity (e.g., no stream exists without a valid user ID or song ID) and ensures there are no orphan records.

In summary, Amazon Sonic's database is intended to serve as a **central nervous system** for the entire creator ecosystem — storing everything from subscriptions and transactions to streams and royalties. Your design and implementation will directly influence how efficiently Amazon can understand its users, compensate its artists, and expand its digital-entertainment presence across emerging markets.