Steam Video Game Recommendation System — Case Study

Abstract: This case study walks through how implicit user behavior data (purchases and play hours) from Steam was transformed into a recommendation system. Using SQL for data engineering and Python for modeling, the project demonstrates how collaborative filtering can personalize game recommendations and improve engagement.

Problem Understanding

Players on Steam interact in multiple ways — buying, browsing, or playing games. Steam ’ s massive library makes discovery challenging. The goal was to recommend games based on what users actually play or purchase.

Dataset & Business Context

The dataset includes 200K user – game interactions, where each event is either a purchase (1) or playtime (hours). These implicit signals allow us to measure player interest without explicit ratings, mirroring real-world personalization systems like Xbox or Netflix.

Data Journey: Bronze → Silver → Gold

• Bronze: Raw event logs loaded into DuckDB. • Silver: Aggregated user – game behaviors (purchase flag and total playtime). • Gold: Derived a unified preference score using (purchased × 2.0) + log(1 + hours_played). This captures both intent and engagement.

Modeling & Recommendation Logic

We built an ALS (Alternating Least Squares) collaborative filtering model using the Implicit library. ALS factorizes the user – game matrix into latent dimensions that represent player preferences and game attributes. These hidden patterns drive personalized recommendations.

Evaluation & Results

The ALS model achieved a Recall@10 of 0.398 — a 77% improvement over the popularity baseline (0.225). This means the model successfully ranked relevant games in the top 10 suggestions for most users.

Key Insights

• Purchases = strong intent, Playtime = sustained engagement. • Latent features captured meaningful relationships (genre, gameplay style). • Recommendations were explainable — users received logical, similar titles.

Conclusion

By combining SQL and Python, we built a complete end-to-end recommender pipeline — from raw event logs to a personalized ranking engine. The system shows how data-driven personalization improves user experience, similar to production-grade recommendation systems.

Tech Stack: SQL ¦ Python ¦ DuckDB ¦ Implicit (ALS) ¦ Matplotlib