Chat Conversation Export

Generated on February 08, 2026 at 10:35 PM

You:

I want to change the structure of this table to the following: Movies: it should contain in each row: Suicide Squad, Bad Moms, Ghostbusters, Sausage Party, Star Trek Beyond, Me Before You and one row for each user such as User: 1 Movie: Suicide Squad, Rating: 2. In Postgres.app

Claude:

I helped you restructure the movie ratings table from wide format to long format in PostgreSQL. I created two SQL scripts: 1. restructure_movie_ratings_direct.sql - Contains INSERT statements with the data already transformed 2. restructure_movie_ratings.sql - Loads from CSV then transforms The restructured format has columns: user_id, movie, rating with one row per user-movie combination.

You:

Let's redo this table and ensure that if user does not have a rating the insert NULL

Claude:

I updated both SQL scripts to include NULL values for missing ratings. Now each user has exactly 6 rows (one for each movie), and missing ratings are represented as NULL instead of being omitted. This gives you a complete dataset with 36 total rows (6 users × 6 movies).

You:

Currently i have a data set as the following: movie_summary <- movies_clean %>% group_by(original_title) %>% summarise(avg_rating = mean(rating,1), vote_avg = first(vote_avg)) Reshape this dataset so it have the following columns 'movie' 'rating_type' and 'rating_value'

Claude:

I created an R script showing how to reshape your movie_summary dataset using pivot_longer(). This transforms the wide format (with avg_rating and vote_avg as separate columns) into long format with rating_type and rating_value columns. Each movie gets 2 rows instead of 1.

You:

explain how does pivot_longer works

Claude:

I created a comprehensive explanation document that covers: • What pivot_longer() does (transforms wide to long format) • Step-by-step breakdown with your movie example • Visual representations • Multiple examples including survey data and time series • Common use cases • Different column selectors you can use The key concept: pivot_longer() takes column NAMES and makes them VALUES in a new column, while stacking the original VALUES into another column.

This conversation covered data restructuring in SQL and R, including pivot_longer() operations.