Querying Data with SQL

**Dataset Used:** <https://www.kaggle.com/datasets/ashirwadsangwan/imdb-dataset>

**Blog Link:** <https://medium.com/@ghazal3501/how-i-mastered-sql-in-7-days-a-data-engineers-journey-2153382d025e>

**GitHub Repo:** <https://github.com/ghazal-ashar/7-day-sql-recap/>

## Foundation

**-- Query 1:** Simple SELECT - Retrieve all movies

**-- Purpose:** Get a quick look at movie records in the dataset

**-- Concepts Used:** SELECT, WHERE, LIMIT

SELECT \*

FROM title\_basics

WHERE titleType = 'movie'

LIMIT 10;

**-- Expected Output:** 10 sample movies with all columns

**-- Business Insight:**

-- Provides an initial sense of the dataset’s breadth and structure.

-- Useful for checking if attributes like runtime, year, and genres are consistently populated.

-- Serves as a launchpad for deeper filtering and aggregation work.

**-- Query 2:** Column Selection - Movie title, year, runtime

**-- Purpose:** Focus only on essential descriptive columns

**-- Concepts Used:** SELECT specific columns, WHERE

SELECT primaryTitle, startYear, runtimeMinutes

FROM title\_basics

WHERE titleType = 'movie'

LIMIT 10;

**-- Expected Output:** Movie name, release year, runtime

**-- Business Insight:**

-- Allows analysts to zero in on core attributes without noise from ancillary fields.

-- Helps streamline reporting and dashboard design.

-- Immediately highlights missing runtimes or release years, which can distort trend analysis.

**-- Query 3:** WHERE Filtering - Movies released after 2015

**-- Purpose: Identify** recent movies

**-- Concepts Used:** SELECT, WHERE, ORDER BY

SELECT primaryTitle, startYear

FROM title\_basics

WHERE titleType = 'movie' AND startYear > 2015

ORDER BY startYear ASC;

**-- Expected Output:** Movies from 2016 onwards

**-- Business Insight:**

-- Establishes a cut of “modern” content for contemporary trend analysis.

-- Useful in studying audience shifts, streaming behaviors, or franchise reboots post-2015.

-- Helps business teams identify fresh IPs or competitive market entries.

**-- Query 4:** Multiple Conditions - Long action movies after 2010

**-- Purpose:** Identify action films with long runtimes

**-- Concepts Used:** SELECT, WHERE with multiple AND conditions, LIKE

SELECT primaryTitle, startYear, runtimeMinutes, genres

FROM title\_basics

WHERE titleType = 'movie'

AND startYear > 2010

AND runtimeMinutes > 120

AND genres LIKE '%Action%';

**-- Expected Output:** Action movies post-2010 with runtime > 120 min

**-- Business Insight:**

-- Surfaces big-budget “blockbuster-style” films for performance tracking.

-- Highlights where studios invest heavily in runtime and spectacle.

-- Critical for analyzing ROI of high-investment productions.

**-- Query 5:** ORDER BY - Sort movies by release year

**-- Purpose:** Show the newest movies first

**-- Concepts Used:** SELECT, WHERE, ORDER BY DESC

SELECT primaryTitle, startYear

FROM title\_basics

WHERE titleType = 'movie'

ORDER BY startYear DESC

LIMIT 10;

**-- Expected Output:** Top 10 most recent movies

**-- Business Insight:**

-- Quick validation of dataset freshness.

-- Helps content teams see which latest titles are tracked in IMDb.

-- Useful for monitoring competitive pipelines and launch cycles.

Aggregation

**-- Query 6:** COUNT with GROUP BY - Movies per genre

**-- Purpose:** Count how many movies exist per genre

**-- Concepts Used:** COUNT, GROUP BY, ORDER BY

SELECT genres, COUNT(\*) AS movie\_count

FROM title\_basics

WHERE titleType = 'movie'

GROUP BY genres

ORDER BY movie\_count DESC;

**-- Expected Output:** Genre vs total movies

**-- Business Insight:**

-- Reveals dominant genres within the catalog.

-- Provides strategic signals on saturation vs opportunity for niche genres.

-- Foundation metric for content portfolio benchmarking.

**-- Query 7:** SUM with GROUP BY - Total votes per genre

**-- Purpose:** Aggregate audience engagement by genre

**-- Concepts Used: JOIN, SUM, GROUP BY**

SELECT b.genres, SUM(r.numVotes) AS total\_votes

FROM title\_basics b

JOIN title\_ratings r ON b.tconst = r.tconst

WHERE b.titleType = 'movie'

GROUP BY b.genres

ORDER BY total\_votes DESC;

**-- Expected Output:** Genre vs total votes

**-- Business Insight:**

-- Captures audience demand and participation intensity per genre.

-- Signals where audience loyalty and fan communities concentrate.

-- Guides promotional spend and content acquisition decisions.

**-- Query 8:** AVG with GROUP BY - Average rating per genre

**-- Purpose:** Identify genre quality based on audience ratings

**-- Concepts Used: JOIN**, AVG, GROUP BY

SELECT b.genres, AVG(r.averageRating) AS avg\_rating

FROM title\_basics b

JOIN title\_ratings r ON b.tconst = r.tconst

WHERE b.titleType = 'movie'

GROUP BY b.genres

ORDER BY avg\_rating DESC;

**-- Expected Output:** Genre vs average rating

**-- Business Insight:**

-- Identifies genres that deliver quality consistently, not just quantity.

-- Enables differentiation between commercially popular vs critically acclaimed segments.

-- Helps prioritize prestige genres for awards campaigns or branding.

**-- Query 9:** HAVING - Genres with >500 movies

**-- Purpose:** Filter popular genres

**-- Concepts Used:** GROUP BY, HAVING, COUNT

SELECT genres, COUNT(\*) AS total\_movies

FROM title\_basics

WHERE titleType = 'movie'

GROUP BY genres

HAVING COUNT(\*) > 500

ORDER BY total\_movies DESC;

**-- Expected Output:** Only genres with 500+ movies

**-- Business Insight:**

-- Narrows focus to statistically significant genres.

-- Prevents strategic bias from outlier categories with thin data.

-- Ensures that market analysis is based on genres with real commercial weight.

**-- Query 10:** MAX/MIN with GROUP BY - Yearly extremes

**-- Purpose:** Find highest and lowest rated movies per year

**-- Concepts Used:** JOIN, GROUP BY, MAX, MIN

SELECT b.startYear,

MAX(r.averageRating) AS highest\_rating,

MIN(r.averageRating) AS lowest\_rating

FROM title\_basics b

JOIN title\_ratings r ON b.tconst = r.tconst

WHERE b.titleType = 'movie'

GROUP BY b.startYear

ORDER BY b.startYear DESC;

**-- Expected Output:** Year vs top & bottom movie ratings

**-- Business Insight:**

-- Surfaces volatility in audience perception year over year.

-- Can flag years with strong hits but also notable flops.

-- Provides a balanced view of both upside and risk in content pipelines.

Multi-Table JOINs

**-- Query 11:** INNER JOIN - Movies with ratings

**-- Purpose:** Combine movie details with ratings

**-- Concepts Used: INNER JOIN, ORDER BY**

SELECT b.primaryTitle, r.averageRating, r.numVotes

FROM title\_basics b

INNER JOIN title\_ratings r ON b.tconst = r.tconst

WHERE b.titleType = 'movie'

ORDER BY r.averageRating DESC

LIMIT 10;

**-- Expected Output:** Top-rated movies with votes

**-- Business Insight:**

-- Links production data with performance feedback.

-- Useful for quickly validating dataset joins and rating coverage.

-- Powers “best of all time” reports for critical acclaim.

**-- Query 12:** LEFT JOIN - Movies including unrated

**-- Purpose:** Ensure all movies are shown, even unrated ones

**-- Concepts Used:** LEFT JOIN, ORDER BY NULLS LAST

SELECT b.primaryTitle, r.averageRating, r.numVotes

FROM title\_basics b

LEFT JOIN title\_ratings r ON b.tconst = r.tconst

WHERE b.titleType = 'movie'

ORDER BY r.averageRating NULLS LAST

LIMIT 10;

**-- Expected Output:** Movies with or without ratings

**-- Business Insight:**

-- Exposes content gaps where user engagement has not yet been captured.

-- Critical for understanding early lifecycle of new titles.

-- Prevents skewed reporting that omits unrated films.

**-- Query 13:** RIGHT JOIN - Ratings and corresponding movies

**-- Purpose:** Include all rating records, even if unmatched

**-- Concepts Used:** RIGHT JOIN, ORDER BY

SELECT b.primaryTitle, r.averageRating

FROM title\_basics b

RIGHT JOIN title\_ratings r ON b.tconst = r.tconst

ORDER BY r.averageRating DESC

LIMIT 10;

**-- Expected Output:** Ratings aligned with movie titles

**-- Business Insight:**

-- Validates referential integrity between ratings and titles.

-- Ensures no orphan ratings exist without a movie reference.

-- Strengthens trust in the dataset for downstream reporting.

**-- Query 14:** Multiple JOINs - Movie, rating, actor

**-- Purpose:** Link movies to actors alongside ratings

**-- Concepts Used:** Multiple JOINs (INNER), ORDER BY

SELECT b.primaryTitle, r.averageRating, n.primaryName AS actor

FROM title\_basics b

JOIN title\_ratings r ON b.tconst = r.tconst

JOIN title\_principals p ON b.tconst = p.tconst

JOIN name\_basics n ON p.nconst = n.nconst

WHERE b.titleType = 'movie'

AND p.category = 'actor'

ORDER BY r.averageRating DESC

LIMIT 10;

**-- Expected Output:** Top-rated movies with actor names

**-- Business Insight:**

-- Connects content quality with on-screen talent.

-- Enables star-power analysis—actors correlated with high ratings.

-- Valuable for casting, talent acquisition, and negotiation insights.

Nested Queries & CTEs

**-- Query 15a:** Subquery - Above average movies

**-- Purpose:** Find movies rated higher than global average

**-- Concepts Used:** Subquery, JOIN, AVG

SELECT primaryTitle, averageRating

FROM title\_basics b

JOIN title\_ratings r ON b.tconst = r.tconst

WHERE r.averageRating > (

SELECT AVG(averageRating)

FROM title\_ratings

)

ORDER BY r.averageRating DESC

LIMIT 10;

**-- Expected Output:** List of standout movies above mean rating

**-- Business Insight:**

-- Identifies “winners” that outperform the market baseline.

-- Focuses attention on exceptional quality rather than volume.

-- Supports awards targeting and critical brand positioning.

**-- Query 15b:** CTE - Top 5 movies per year

**-- Purpose:** Rank best movies yearly

**-- Concepts Used:** CTE, ROW\_NUMBER, PARTITION BY

WITH yearly\_ranked AS (

SELECT b.startYear, b.primaryTitle, r.averageRating,

ROW\_NUMBER() OVER (PARTITION BY b.startYear ORDER BY r.averageRating DESC) AS rank\_in\_year

FROM title\_basics b

JOIN title\_ratings r ON b.tconst = r.tconst

WHERE b.titleType = 'movie'

)

SELECT \* FROM yearly\_ranked

WHERE rank\_in\_year <= 5

ORDER BY startYear DESC, averageRating DESC;

**-- Expected Output:** Top 5 movies each year

**-- Business Insight:**

-- Structures data into yearly competitive landscapes.

-- Useful for understanding how quality distribution evolves annually.

-- Supports investor decks and strategic planning by spotlighting top content leaders.

Integration

**-- Query 16:** Complex yearly report

**-- Purpose:** Generate annual stats with top genres

**-- Concepts Used:** CTEs, Aggregation, RANK, JOIN

WITH yearly\_stats AS (

SELECT b.startYear,

COUNT(\*) AS total\_movies,

AVG(r.averageRating) AS avg\_rating

FROM title\_basics b

JOIN title\_ratings r ON b.tconst = r.tconst

WHERE b.titleType = 'movie'

GROUP BY b.startYear

),

genre\_stats AS (

SELECT b.startYear, b.genres, AVG(r.averageRating) AS genre\_avg\_rating,

RANK() OVER (PARTITION BY b.startYear ORDER BY AVG(r.averageRating) DESC) AS genre\_rank

FROM title\_basics b

JOIN title\_ratings r ON b.tconst = r.tconst

WHERE b.titleType = 'movie'

GROUP BY b.startYear, b.genres

)

SELECT y.startYear, y.total\_movies, y.avg\_rating, g.genres AS top\_genre, g.genre\_avg\_rating

FROM yearly\_stats y

JOIN genre\_stats g ON y.startYear = g.startYear

WHERE g.genre\_rank = 1

ORDER BY y.startYear DESC;

**-- Expected Output:** Yearly movie counts, avg rating, and best genre

**-- Business Insight:**

-- Consolidates multiple KPIs (volume, quality, genre leadership) into one annual report.

-- Supports long-term strategic trend analysis across genres and years.

-- Valuable for greenlighting, investment roadmaps, and forecasting content direction.