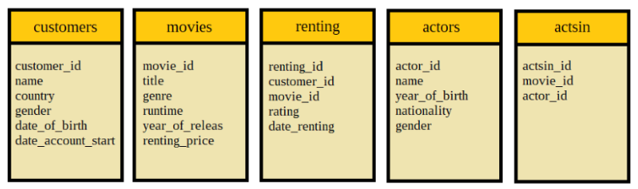
**Data-Driven Decision Making in SQL**

**1. Introduction to business intelligence for an online movie rental database**

**MovieNow data structure:**

****

**Exploring the database**

First, we explore the tables and its columns.

Which of the following quantities can't be computed?

* The number of customers from each country.
* The number of movies with an international award.
* The average rating of a movie.
* The number of movies with the actor Daniel Radcliffe.

The number of customers from each country can be computed as follows:

SELECT country, count(\*)

FROM customers

GROUP BY 1

ORDER BY 2 DESC;

The average rating of a movie can be computed as follows:

SELECT round(avg(rating), 2)

FROM renting;

The number of movies with the actor Daniel Radcliffe can be computed as follows:

SELECT count(\*)

FROM actsin

WHERE actor\_id = (SELECT actor\_id

    FROM actors

    WHERE name LIKE 'Daniel Radcliffe');

Answer: The number of movies with an international award cannot be computed because there is no information about movie awards in the data.

**Exploring the table renting**

The table renting includes all records of movie rentals.

* Each record has a unique ID renting\_id.
* It also contains information about customers (customer\_id) and which movies they watched (movie\_id).
* Customers can give a rating after watching the movie, and the day the movie was rented is recorded.

To select all columns from renting:

SELECT \*

FROM renting;

To select the columns from renting which are required to calculate the average rating per movie:

SELECT movie\_id, rating

FROM renting;

We would like to check if there are any missing values (null). In which column of renting null values?

The column rating in the renting table has a lot of null values since not every customer gives a rating after watching a movie.

**Working with dates**

For the analysis of monthly or annual changes, it is important to select data from specific time periods. The format of dates is 'YYYY-MM-DD'.

To select all movies rented on October 9th, 2018:

SELECT \*

FROM renting

WHERE date\_renting = '2018-10-09';

To select all records of movie rentals between beginning of April 2018 till end of August 2018.

SELECT \*

FROM renting

WHERE date\_renting BETWEEN '2018-04-01' AND '2018-08-31';

To select the most recent records of movie rentals on top of the resulting table and order them in decreasing order:

SELECT \*

FROM renting

WHERE date\_renting BETWEEN '2018-04-01' AND '2018-08-31'

ORDER BY date\_renting DESC;

**Selecting movies**

The table movies contain all movies available on the online platform. Here are some examples for selecting movies.

To select all movies which are not dramas:

SELECT \*

FROM movies

WHERE genre <> 'Drama';

To select the movies 'Showtime', 'Love Actually' and 'The Fighter'.

SELECT \*

FROM movies

WHERE title IN ('Showtime', 'Love Actually', 'The Fighter');

To order the movies by increasing renting price:

SELECT \*

FROM movies

ORDER BY renting\_price;

**Select from renting**

Only some users give a rating after watching a movie and we will explore only those movie rentals where a rating was provided.

To select from table renting all movie rentals from 2018 and filter only those records which have a movie rating:

SELECT \*

FROM renting

WHERE date\_renting BETWEEN '2018-01-01' AND '2018-12-31'

AND rating IS NOT NULL;

**Summarizing customer information**

Customers are analyzed in groups, such as customers per country or customers per age group when making business decisions.

To count the number of customers born in the 80s:

SELECT COUNT(\*)

FROM customers

WHERE date\_of\_birth BETWEEN '1980-01-01' AND '1989-12-31';

To count the number of customers from Germany:

SELECT COUNT(\*)

FROM customers

WHERE country = 'Germany';

To count the number of countries where MovieNow has customers:

SELECT COUNT(DISTINCT country)

FROM customers;

We’ve aggregated data from the customers table and figured out that MovieNow has customers in 11 countries and that there are no customers from Germany.

**Ratings of movie 25**

The movie ratings give us insight into the preferences of the customers.

Example: Report summary statistics, such as the minimum, maximum, average, and count, of ratings for the movie with ID 25.

To select all movie rentals of the movie with movie\_id 25 from the table renting, and for those records, calculate the minimum, maximum and average rating and count the number of ratings for this movie.

SELECT MIN(rating) min\_rating,

       MAX(rating) max\_rating,

       AVG(rating) avg\_rating,

       COUNT(rating) number\_ratings

FROM renting

WHERE movie\_id = 25;

We’ve summarized the ratings for the movie 25 and used meaningful column names. This movie has 8 ratings with minimum 5, maximum 10 and average 7.5 rating.

**Examining annual rentals**

Case scenario: You are asked to provide a report about the development of the company. Specifically, your manager is interested in the total number of movie rentals, the total number of ratings and the average rating of all movies since the beginning of 2019.

First, to select all records of movie rentals since January 1st 2019:

SELECT \*

FROM renting

WHERE date\_renting >= '2019-01-01';

To count the number of movie rentals and calculate the average rating since the beginning of 2019:

SELECT

    COUNT(\*),

    AVG(rating)

FROM renting

WHERE date\_renting >= '2019-01-01';

To use as alias column names number\_renting and average\_rating respectively:

SELECT

    COUNT(\*) AS number\_renting,

    AVG(rating) AS average\_rating

FROM renting

WHERE date\_renting >= '2019-01-01';

Finally to count how many ratings exist since 2019-01-01:

SELECT

    COUNT(\*) AS number\_renting,

    AVG(rating) AS average\_rating,

    COUNT(rating) AS number\_ratings

FROM renting

WHERE date\_renting >= '2019-01-01';

We’ve managed to extract all KPIs from the renting records by working with aggregations, NULL values and dates.

**2. Decision Making with simple SQL queries**

**First account for each country**

Case scenario: You’re asked to conduct an analysis to see when the first customer accounts were created for each country.

We create a table with a row for each country and columns for the country name and the date when the first customer account was created. We order by date in ascending order.

SELECT country,

    MIN(date\_account\_start) AS first\_account

FROM customers

GROUP BY country

ORDER BY first\_account;

We learned that the first customer account was created in France.

**Average movie ratings**

Case scenario: For each movie the average rating, the number of ratings and the number of views has to be reported. Generate a table with meaningful column names. Group the data in the table renting by movie\_id and report the ID and the average rating. Order the rows of the table by the average rating such that it is in decreasing order. Observe what happens to NULL values.

SELECT movie\_id,

       AVG(rating) AS avg\_rating,

       COUNT(rating) AS number\_ratings,

       COUNT(\*) AS number\_renting

FROM renting

GROUP BY movie\_id

ORDER BY avg\_rating DESC;

The average is null because all of the ratings of the movie are null.

**Average rating per customer**

Case scenario: you will now look at the average movie ratings, this time for customers. So you will obtain a table with the average rating given by each customer. Further, you will include the number of ratings and the number of movie rentals per customer. You will report these summary statistics only for customers with more than 7 movie rentals and order them in ascending order by the average rating.

SELECT customer\_id,

      AVG(rating),

      COUNT(rating),

      COUNT(\*)

FROM renting

GROUP BY customer\_id

HAVING COUNT(\*) > 7

ORDER BY AVG(rating);

We found out that customer number 104 gave the lowest average ratings for 4 movies.

**Join renting and customers**

For many analyses it is necessary to add customer information to the data in the table renting.

To augment the table renting with all columns from the table customers with a LEFT JOIN:

SELECT \*

FROM renting AS r

LEFT JOIN customers AS c

ON r.customer\_id =  c.customer\_id;

To select only records from customers coming from Belgium:

SELECT \*

FROM renting AS r

LEFT JOIN customers AS c

ON r.customer\_id = c.customer\_id

WHERE country = 'Belgium';

To calculate average ratings of customers from Belgium:

SELECT AVG(rating)

FROM renting AS r

LEFT JOIN customers AS c

ON r.customer\_id = c.customer\_id

WHERE c.country='Belgium';

We’ve calculated the average rating for customers from Belgium, which is 8.9.

**Aggregating revenue, rentals and active customers**

Case scenario: The management of MovieNow wants to report key performance indicators (KPIs) for the performance of the company in 2018. They are interested in measuring the financial successes as well as user engagement. Important KPIs are, therefore, the revenue coming from movie rentals, the number of movie rentals and the number of active customers.

First, we need to join movies on renting to include the renting\_price from the movies table for each renting record.

SELECT \*

FROM renting AS r

LEFT JOIN movies AS m

ON r.movie\_id = m.movie\_id;

We calculate the revenue coming from movie rentals, the number of movie rentals and the number of customers who rented a movie.

SELECT

    SUM(m.renting\_price),

    COUNT(\*),

    COUNT(DISTINCT r.customer\_id)

FROM renting AS r

LEFT JOIN movies AS m

ON r.movie\_id = m.movie\_id;

Now, we can report these values for the year 2018. We calculate the revenue in 2018, the number of movie rentals and the number of active customers in 2018. An active customer is a customer who rented at least one movie in 2018.

SELECT

    SUM(m.renting\_price),

    COUNT(\*),

    COUNT(DISTINCT r.customer\_id)

FROM renting AS r

LEFT JOIN movies AS m

ON r.movie\_id = m.movie\_id

WHERE date\_renting BETWEEN '2018-01-01' AND '2018-12-31';

We've calculated a turnover of 658.02 and found the number of rentals to be 298 and the number of active users to be 93 in 2018.

**Movies and actors**

Case scenario: You are asked to give an overview of which actors play in which movie.

We create a list of actor names and movie titles in which they act. We make sure that each combination of actor and movie appears only once.

SELECT a.name,

       m.title

FROM actsin AS ai

LEFT JOIN movies AS m

ON m.movie\_id = ai.movie\_id

LEFT JOIN actors AS a

ON a.actor\_id = ai.actor\_id;

We’ve joined three tables and created a list of movie titles and actors in these movies.

**Income from movies**

Case scenario: How much income did each movie generate?

We use a join to get the movie title and price for each movie rental.

SELECT m.title,

       m.renting\_price

FROM renting AS r

LEFT JOIN movies AS m

ON m.movie\_id = r.movie\_id;

We report the total income for each movie and order the result by decreasing income.

SELECT title,

       SUM(renting\_price) AS income\_movie

FROM

       (SELECT m.title,

               m.renting\_price

       FROM renting AS r

       LEFT JOIN movies AS m

       ON r.movie\_id=m.movie\_id) AS rm

GROUP BY title

ORDER BY income\_movie DESC;

**Age of actors from the USA**

Case scenario: Now you will explore the age of American actors and actresses. Report the date of birth of the oldest and youngest US actor and actress.

We create a subsequent SELECT statements in the FROM clause to get all information about actors from the USA.

We report for actors from the USA the year of birth of the oldest and the year of birth of the youngest actor and actress.

SELECT gender,

       MAX(year\_of\_birth),

       MIN(year\_of\_birth)

FROM

   (SELECT \*

   FROM actors

   WHERE nationality = 'USA') AS a

GROUP BY gender;

We learned that in the MovieNow actors' record the oldest actor was born in 1930 and the oldest actress in 1945.

**Identify favorite movies for a group of customers**

Case scenario: Which is the favorite movie on MovieNow? Answer this question for a specific group of customers: for all customers born in the 70s.

We augment the table renting with customer information and information about the movies.

SELECT \*

FROM renting AS r

LEFT JOIN customers AS c

ON r.customer\_id = c.customer\_id

LEFT JOIN movies AS m

ON r.movie\_id = m.movie\_id;

We select only those records of customers born in the 70s.

SELECT \*

FROM renting AS r

LEFT JOIN customers AS c

ON c.customer\_id = r.customer\_id

LEFT JOIN movies AS m

ON m.movie\_id = r.movie\_id

WHERE c.date\_of\_birth BETWEEN '1970-01-01' AND '1979-12-31';

For each movie, we report the number of times it was rented, as well as the average rating. We limit our results to customers born in the 1970s.

SELECT m.title,

COUNT(\*),

AVG(r.rating)

FROM renting AS r

LEFT JOIN customers AS c

ON c.customer\_id = r.customer\_id

LEFT JOIN movies AS m

ON m.movie\_id = r.movie\_id

WHERE c.date\_of\_birth BETWEEN '1970-01-01' AND '1979-12-31'

GROUP BY m.title;

We remove those movies from the table with only one rental.

We order the result table such that movies with highest rating come first.

SELECT m.title,

COUNT(\*),

AVG(r.rating)

FROM renting AS r

LEFT JOIN customers AS c

ON c.customer\_id = r.customer\_id

LEFT JOIN movies AS m

ON m.movie\_id = r.movie\_id

WHERE c.date\_of\_birth BETWEEN '1970-01-01' AND '1979-12-31'

GROUP BY m.title

HAVING COUNT(\*) > 1

ORDER BY avg DESC;

We found out that ‘The Fellowship of the Ring’ is a movie with one of the best ratings among people born in the 70s.

**Identify favorite actors for Spain**

Case scenario: You're now going to explore actor popularity in Spain.

We augment the table renting with information about customers and actors.

SELECT \*

FROM renting AS r

LEFT JOIN customers AS c

ON r.customer\_id = c.customer\_id

LEFT JOIN actsin AS ai

ON r.movie\_id = ai.movie\_id

LEFT JOIN actors AS a

ON a.actor\_id = ai.actor\_id ;

We report the number of movie rentals and the average rating for each actor, separately for male and female customers.

We report only actors with more than 5 movie rentals.

SELECT a.name,  c.gender,

       COUNT(\*) AS number\_views,

       AVG(r.rating) AS avg\_rating

FROM renting as r

LEFT JOIN customers AS c

ON r.customer\_id = c.customer\_id

LEFT JOIN actsin as ai

ON r.movie\_id = ai.movie\_id

LEFT JOIN actors as a

ON ai.actor\_id = a.actor\_id

\_\_\_ -- Select only customers from Spain

GROUP BY a.name, c.gender

HAVING AVG(r.rating) IS NOT NULL

  AND COUNT(\*) > 5

ORDER BY avg\_rating DESC, number\_views DESC;

We report the favorite actors only for customers from Spain.

SELECT a.name,  c.gender,

       COUNT(\*) AS number\_views,

       AVG(r.rating) AS avg\_rating

FROM renting as r

LEFT JOIN customers AS c

ON r.customer\_id = c.customer\_id

LEFT JOIN actsin as ai

ON r.movie\_id = ai.movie\_id

LEFT JOIN actors as a

ON ai.actor\_id = a.actor\_id

WHERE c.country = 'Spain'

GROUP BY a.name, c.gender

HAVING AVG(r.rating) IS NOT NULL

  AND COUNT(\*) > 5

ORDER BY avg\_rating DESC, number\_views DESC;

We found out that in this table Catherine Keener is the favorite actress among female Spain customers and that male customers from Spain like the actors from Harry Potter best: Emma Watson, Daniel Radcliffe and Rupert Grint.

**KPIs per country**

Case scenario: You were asked to provide a report about the development of the company. This time you have to prepare a similar report with KPIs for each country separately. Your manager is interested in the total number of movie rentals, the average rating of all movies and the total revenue for each country since the beginning of 2019.

We augment the table renting with information about customers and movies.

We select only records about rentals since beginning of 2019.

SELECT \*

FROM renting AS r

LEFT JOIN customers AS c

ON r.customer\_id = c.customer\_id

LEFT JOIN movies AS m

ON r.movie\_id = m.movie\_id

WHERE date\_renting >= '2019-01-01';

We calculate the number of movie rentals, the average rating and the revenue from movie rentals. We then report these KPIs for each country.

SELECT

    c.country,

    COUNT(r.renting\_id) AS number\_renting,

    AVG(r.rating) AS average\_rating,

    SUM(m.renting\_price) AS revenue

FROM renting AS r

LEFT JOIN customers AS c

ON c.customer\_id = r.customer\_id

LEFT JOIN movies AS m

ON m.movie\_id = r.movie\_id

WHERE date\_renting >= '2019-01-01'

GROUP BY c.country;

We’ve calculated a total revenue of 57.94 for Spain, with 26 movie rentals and an average rating of 8.1.