

# Grouping movies

DATA-DRIVEN DECISION MAKING IN SQL



**Bart Baesens**

Professor Data Science and Analytics

# GROUP BY Applications

- Preferences of customers by country or gender.
- The popularity of movies by genre or year of release.
- The average price of movies by genre.

# Table: movies\_selected

title	genre	renting_price
Fight Club	Drama	2.49
The Lord of the Rings: The Fellowship of the Ring	Fantasy	2.59
The Lord of the Rings: The Two Towers	Fantasy	2.69
The Lord of the Rings: The Return of the King	Fantasy	2.79
The Prestige	Sci-Fiction	2.85
Ratatouille	Animation	2.89
WALL.E	Animation	2.89
Inception	Sci-Fiction	2.89
The Big Shot	Drama	2.99
The Imitation Game	Drama	2.99
Moonlight	Drama	2.99
LaLaLand	Romance	2.99
Zootopia	Animation	2.99

# GROUP BY

```
SELECT genre
FROM movies_selected
GROUP BY genre;
```

```
| genre          |
|-----|
| Drama         |
| Fantasy       |
| Sci-Fiction    |
| Animation     |
| Romance       |
```

# Average renting price

```
SELECT genre,  
       AVG(renting_price) AS avg_price  
FROM movies_selected  
GROUP BY genre;
```

genre	avg_price
Drama	2.865
Fantasy	2.69
Sci-Fiction	2.87
Animation	2.923333333
Romance	2.99

# movies\_selected table

title	genre	renting_price
Fight Club	Drama	2.49
The Lord of the Rings: The Fellowship of the Ring	Fantasy	2.59
The Lord of the Rings: The Two Towers	Fantasy	2.69
The Lord of the Rings: The Return of the King	Fantasy	2.79
The Prestige	Sci-Fiction	2.85
Ratatouille	Animation	2.89
WALL.E	Animation	2.89
Inception	Sci-Fiction	2.89
The Big Shot	Drama	2.99
The Imitation Game	Drama	2.99
Moonlight	Drama	2.99
LaLaLand	Romance	2.99
Zootopia	Animation	2.99

# movies\_selected table split

title	genre	renting_price
The Lord of the Rings: The Fellowship of the Ring	Fantasy	2.59
The Lord of the Rings: The Two Towers	Fantasy	2.69
The Lord of the Rings: The Return of the King	Fantasy	2.79

title	genre	renting_price
Fight Club	Drama	2.49
The Big Shot	Drama	2.99
The Imitation Game	Drama	2.99
Moonlight	Drama	2.99

title	genre	renting_price
LaLaLand	Romance	2.99

title	genre	renting_price
The Prestige	Sci-Fiction	2.85
Inception	Sci-Fiction	2.89

title	genre	renting_price
Ratatouille	Animation	2.89
WALL.E	Animation	2.89
Zootopia	Animation	2.99

# Average rental price and number of movies

```
SELECT genre,  
       AVG(renting_price) AS avg_price,  
       COUNT(*) AS number_movies  
FROM movies_selected  
GROUP BY genre
```

genre	avg_price	number_movies
Drama	2.865	4
Fantasy	2.69	3
Sci-Fiction	2.87	2
Animation	2.923333333	3
Romance	2.99	1



# HAVING

```
SELECT genre,  
       AVG(renting_price) avg_price,  
       COUNT(*) number_movies  
FROM movies  
GROUP BY genre  
HAVING COUNT(*) > 2;
```

genre	avg_price	number_movies
Drama	2.865	4
Fantasy	2.69	3
Animation	2.923333333	3

# Let's practice!

DATA-DRIVEN DECISION MAKING IN SQL

# Joining movie ratings with customer data

DATA-DRIVEN DECISION MAKING IN SQL

SQL

Irene Ortner

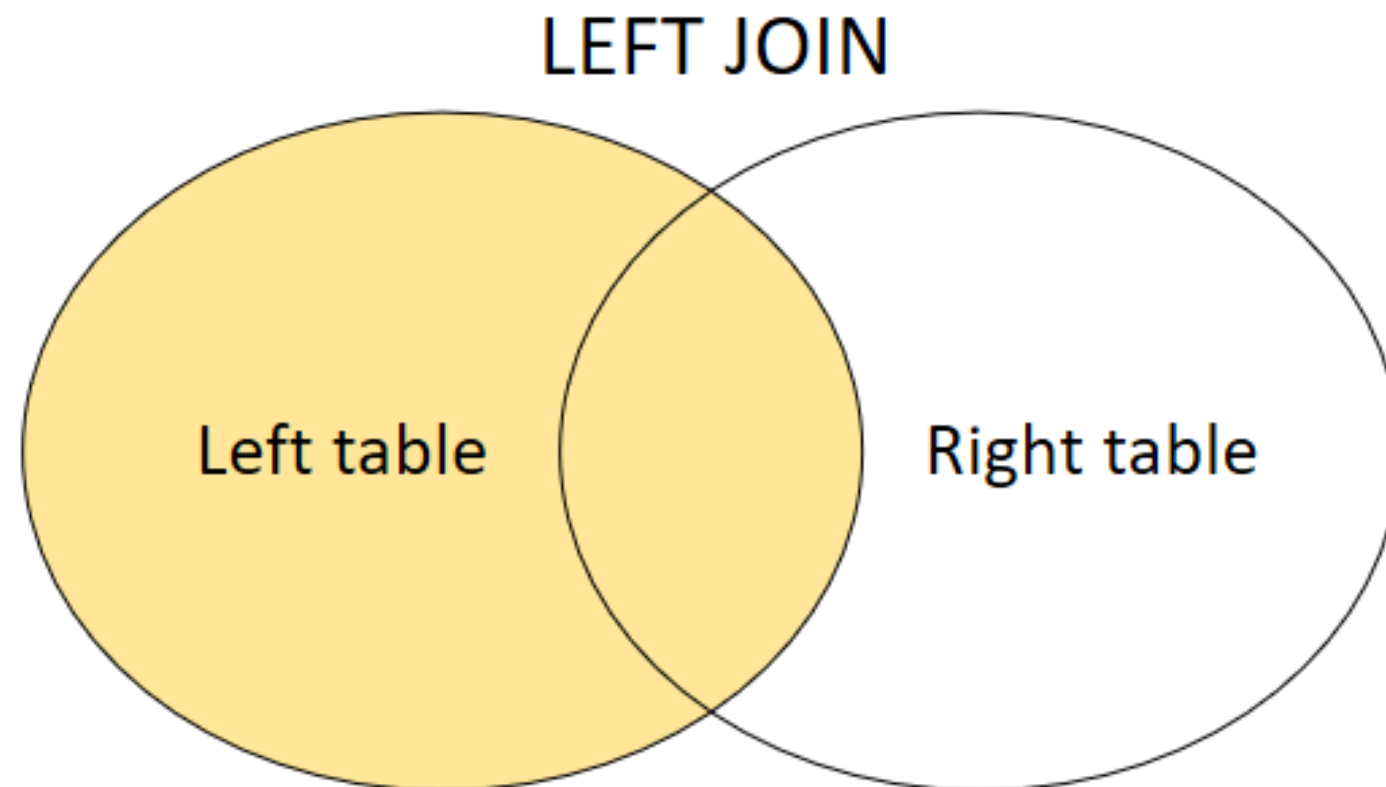
Data Scientist at Applied Statistics

# JOIN

- Prerequisite course about joining data in SQL

# LEFT JOIN

- `LEFT JOIN` is an outer join.
- Keep all rows of the left table, match with rows in the right table.
- Use identifier to define which rows of two tables can be matched.



# Giving a table a name

```
SELECT *  
FROM customers AS c  
WHERE c.customer_id = 1;
```

# Tables for LEFT JOIN

Left table: `renting_selected`

<code>renting_id</code>	<code>customer_id</code>	<code>rating</code>
518	1	<code>`null`</code>
203	2	6
478	4	7
292	4	8
477	5	<code>`null`</code>
400	6	<code>`null`</code>

Right table: `customers_selected`

<code>customer_id</code>	<code>name</code>	<code>gender</code>
1	Robert Bohm	male
2	Wolfgang Ackermann	male
3	Daniela Herzog	female
4	Julia Jung	female

# LEFT JOIN example

```
SELECT *  
FROM renting_selected AS r  
LEFT JOIN customers_selected AS c  
ON r.customer_id = c.customer_id;
```

renting_id	customer_id	rating	customer_id	name	gender
518	1	`null`	1	Robert Bohm	male
203	2	6	2	Wolfgang Ackermann	male
478	4	`null`	4	Julia Jung	female
292	4	8	4	Julia Jung	female
477	5	7	`null`	`null`	`null`



# More than one JOIN

```
SELECT m.title,  
       c.name  
FROM renting AS r  
LEFT JOIN movies AS m  
ON r.movie_id = m.movie_id  
LEFT JOIN customers AS c  
ON r.customer_id = c.customer_id;
```

# Let's practice!

DATA-DRIVEN DECISION MAKING IN SQL

# Money spent per customer with sub-queries

DATA-DRIVEN DECISION MAKING IN SQL

SQL

Tim Verdonck

Professor Statistics and Data Science

# Subsequent SELECT statements - actresses

Query 1:

```
SELECT *  
FROM actors  
WHERE gender = 'female';
```

actor_id	name	year_of_birth	nationality	gender
1	Abbie Cornish	1982	Australia	female
4	Amy Adams	1974	USA	female

# Subsequent SELECT statements - actresses

```
SELECT * -- Query 1
FROM actors
WHERE gender = 'female';
```

- Group result table of query 1 by nationality.
- Report year of birth for the oldest and youngest actress in each country.

```
SELECT af.nationality,
       MIN(af.year_of_birth),
       MAX(af.year_of_birth)
FROM
  (SELECT *
   FROM actors
   WHERE gender = 'female') AS af
GROUP BY af.nationality;
```

# Result subsequent SELECT statement - actresses

```
SELECT af.nationality,  
       MIN(af.year_of_birth),  
       MAX(af.year_of_birth)  
FROM  
  (SELECT *  
   FROM actors  
   WHERE gender = 'female') AS af  
GROUP BY af.nationality;
```

nationality	min	max
Italy	1976	1976
Iran	1952	1952
USA	1945	1993

# How much money did each customer spend?

- First step: Add `renting_price` from `movies` to table `renting`.

```
SELECT r.customer_id,  
       m.renting_price  
FROM renting AS r  
LEFT JOIN movies AS m  
ON r.movie_id=m.movie_id;
```

customer_id	renting_price
41	2.59
10	2.79
108	2.39
39	1.59
104	1.69

# How much money did each customer spend?

- Second step:
  - group result table from first step by `customer_id`
  - take the sum of `renting_price`

```
SELECT rm.customer_id,  
       SUM(rm.renting_price)  
FROM  
  (SELECT r.customer_id,  
         m.renting_price  
   FROM renting AS r  
   LEFT JOIN movies AS m  
   ON r.movie_id=m.movie_id) AS rm  
GROUP BY rm.customer_id;
```



# How much money did each customer spend?

customer_id	sum
-----	-----
116	7.47
87	17.53
71	6.87
68	1.59
51	4.87

# Let's practice!

DATA-DRIVEN DECISION MAKING IN SQL

# Identify favorite actors of customer groups

DATA-DRIVEN DECISION MAKING IN SQL

SQL

Irene Ortner

Data Scientist at Applied Statistics

# Combining SQL statements in one query

- LEFT JOIN
- WHERE
- GROUP BY
- HAVING
- ORDER BY

# From renting records to customer and actor information

Our question: Who is the favorite actor for a certain customer group?

Join table `renting` with tables

- `customers`
- `actsin`
- `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 ai.actor_id = a.actor_id;
```

# Male customers

- Actors which play most often in movies watched by male customers.

```
SELECT a.name,  
       COUNT(*)  
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.gender = 'male'  
GROUP BY a.name;
```

# Who is the favorite actor?

- Actor being watched most often.
- Best average rating when being watched.

```
SELECT a.name,  
       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.gender = 'male'
```

# Add HAVING and ORDER BY

```
SELECT a.name,  
       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.gender = 'male'  
GROUP BY a.name  
HAVING AVG(r.rating) IS NOT NULL  
ORDER BY avg_rating DESC, number_views DESC;
```



# Add HAVING and ORDER BY

name	number_views	avg_rating
-----	-----	-----
Ray Romano	3	10.00
Sean Bean	2	10.00
Leonardo DiCaprio	3	9.33
Christoph Waltz	3	9.33

# Let's practice!

DATA-DRIVEN DECISION MAKING IN SQL