

Nested query

DATA-DRIVEN DECISION MAKING IN SQL



Irene Ortner

Data Scientist at Applied Statistics

Nested query

- SELECT block in WHERE or HAVING clauses
- Inner query returns single or multiple values
- Use result from the inner query to select specific rows in another query

The inner query

Step 1: The inner query

```
SELECT DISTINCT customer_id  
FROM renting  
WHERE rating <= 3
```

```
| customer_id |  
|-----|  
| 28          |  
| 41          |  
| 86          |  
| 120         |
```

Result in the WHERE clause

```
SELECT name  
FROM customers  
WHERE customer_id IN (28, 41, 86, 120);
```

The outer query

Step 2: The outer query

```
SELECT name
FROM customers
WHERE customer_id IN
  (SELECT DISTINCT customer_id
   FROM renting
   WHERE rating <= 3);
```

```
| name          |
|-----|
| Sidney G n reux |
| Zara Mitchell  |
```

Nested query in the HAVING clause

Step 1: The inner query

```
SELECT MIN(date_account_start)
FROM customers
WHERE country = 'Austria';
```

```
| min          |
|-----|
| 2017-11-22  |
```

Nested query in the HAVING clause

Step 2: The outer query

```
SELECT country, MIN(date_account_start)
FROM customers
GROUP BY country
HAVING MIN(date_account_start) <
    (SELECT MIN(date_account_start)
     FROM customers
     WHERE country = 'Austria');
```

country	min
Spain	2017-02-14
Great Britain	2017-03-31

Who are the actors in the movie Ray?

```
SELECT name
FROM actors
WHERE actor_id IN
  (SELECT actor_id
   FROM actsin
   WHERE movie_id =
     (SELECT movie_id
      FROM movies
      WHERE title='Ray'));
```

```
| name          |
|-----|
| Jamie Foxx   |
| Kerry Washington |
| Regina King  |
```


Let's practice!

DATA-DRIVEN DECISION MAKING IN SQL

Correlated nested queries

DATA-DRIVEN DECISION MAKING IN SQL



Bart Baesens

Professor Data Science and Analytics

Correlated queries

- Condition in the WHERE clause of the inner query.
- References some column of a table in the outer query.

Example correlated query

- Number of movie rentals more than 5

```
SELECT *  
FROM movies as m  
WHERE 5 <  
    (SELECT COUNT(*)  
     FROM renting as r  
     WHERE r.movie_id=m.movie_id);
```

Evaluate inner query

```
SELECT COUNT(*)  
FROM renting as r  
WHERE r.movie_id = 1;
```

```
| count |  
|-----|  
| 8     |
```

Evaluate outer query

Number of movie rentals larger than 5

```
SELECT *  
FROM movies as m  
WHERE 5 <  
    (SELECT COUNT(*)  
     FROM renting as r  
     WHERE r.movie_id = m.movie_id);
```

movie_id	title	genre	runtime	year_of_release	renting_price
1	One Night at McCool's	Comedy	93	2001	2.09
2	Swordfish	Drama	99	2001	2.19

Less than 5 movie rentals

Select movies with less than 5 movie rentals.

```
SELECT *
FROM movies as m
WHERE 5 >
    (SELECT COUNT(*)
     FROM renting as r
     WHERE r.movie_id = m.movie_id);
```

movie_id	title	genre	runtime	year_of_release	renting_price
17	The Human Stain	Mystery & Suspense	106	2003	1.99
20	Love Actually	Comedy	135	2003	2.29

Let's practice!

DATA-DRIVEN DECISION MAKING IN SQL

Queries with EXISTS

DATA-DRIVEN DECISION MAKING IN SQL



Irene Ortner

Data Scientist at Applied Statistics

EXISTS

- Special case of a correlated nested query.
- Used to check if result of a correlated nested query is empty.
- It returns: TRUE or FALSE
- TRUE = not empty -> row of the outer query is selected.
- FALSE = empty
- Columns specified in SELECT component not considered - use `SELECT *`

Movies with at least one rating

```
SELECT *  
FROM movies AS m  
WHERE EXISTS  
  (SELECT *  
   FROM renting AS r  
   WHERE rating IS NOT NULL  
   AND r.movie_id = m.movie_id);
```

Movies with at least one rating

```
SELECT *  
FROM renting AS r  
WHERE rating IS NOT NULL  
AND r.movie_id = 11;
```

```
| renting_id | customer_id | movie_id | rating | renting_price |  
|-----|-----|-----|-----|-----|
```

Movies with at least one rating

```
SELECT *  
FROM renting AS r  
WHERE rating IS NOT NULL  
AND r.movie_id = 1;
```

renting_id	customer_id	movie_id	rating	renting_price
71	111	1	5	2018-07-21
170	36	1	10	2018-10-18

EXISTS query with result

```
SELECT *  
FROM movies AS m  
WHERE EXISTS  
  (SELECT *  
   FROM renting AS r  
   WHERE rating IS NOT NULL  
   AND r.movie_id = m.movie_id);
```

movie_id	title	genre	runtime	year_of_release	renting_price
1	One Night at McCool's	Comedy	93	2001	2.09
2	Swordfish	Drama	99	2001	2.19

NOT EXISTS

- TRUE = table is empty -> row of the outer query is selected.

```
SELECT *  
FROM movies AS m  
WHERE NOT EXISTS  
  (SELECT *  
   FROM renting AS r  
   WHERE rating IS NOT NULL  
   AND r.movie_id = m.movie_id);
```

movie_id	title	genre	runtime	year_of_release	renting_price
11	Showtime	Comedy	95	2002	1.79

Let's practice!

DATA-DRIVEN DECISION MAKING IN SQL

Queries with UNION and INTERSECT

DATA-DRIVEN DECISION MAKING IN SQL

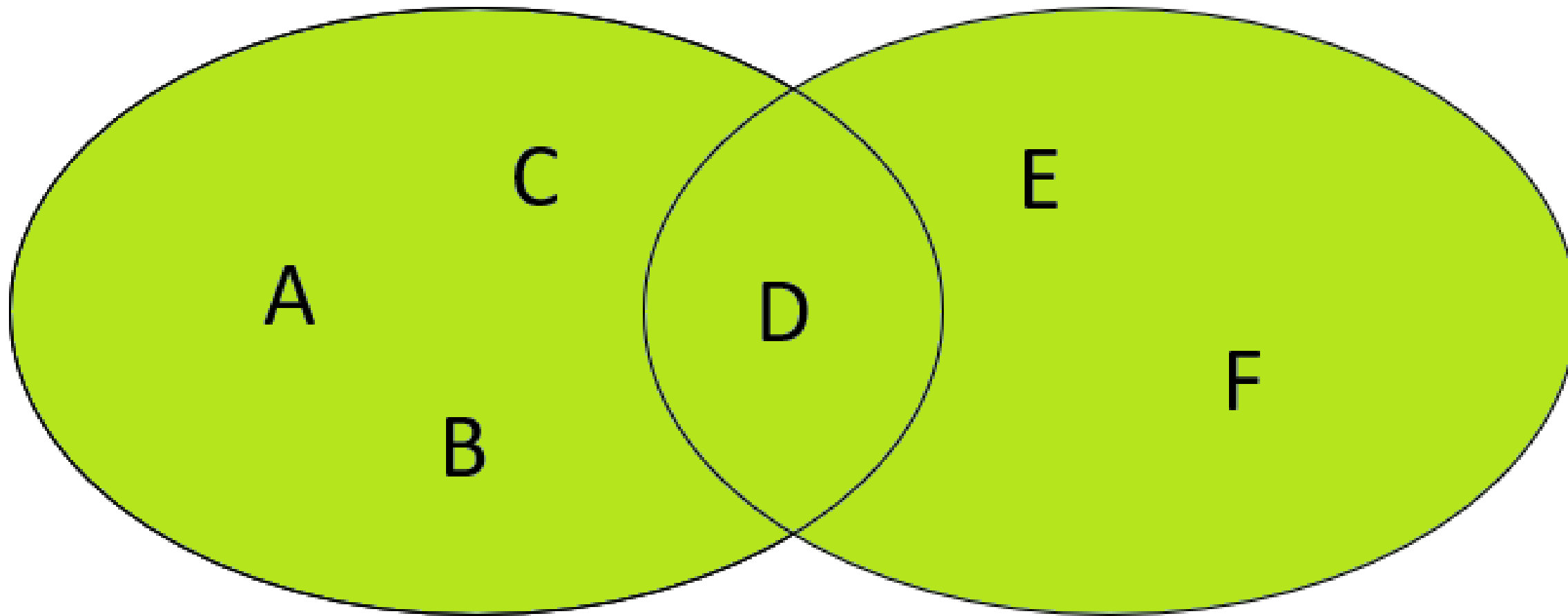


Tim Verdonck

Professor Statistics and Data Science

UNION

UNION



Example - UNION

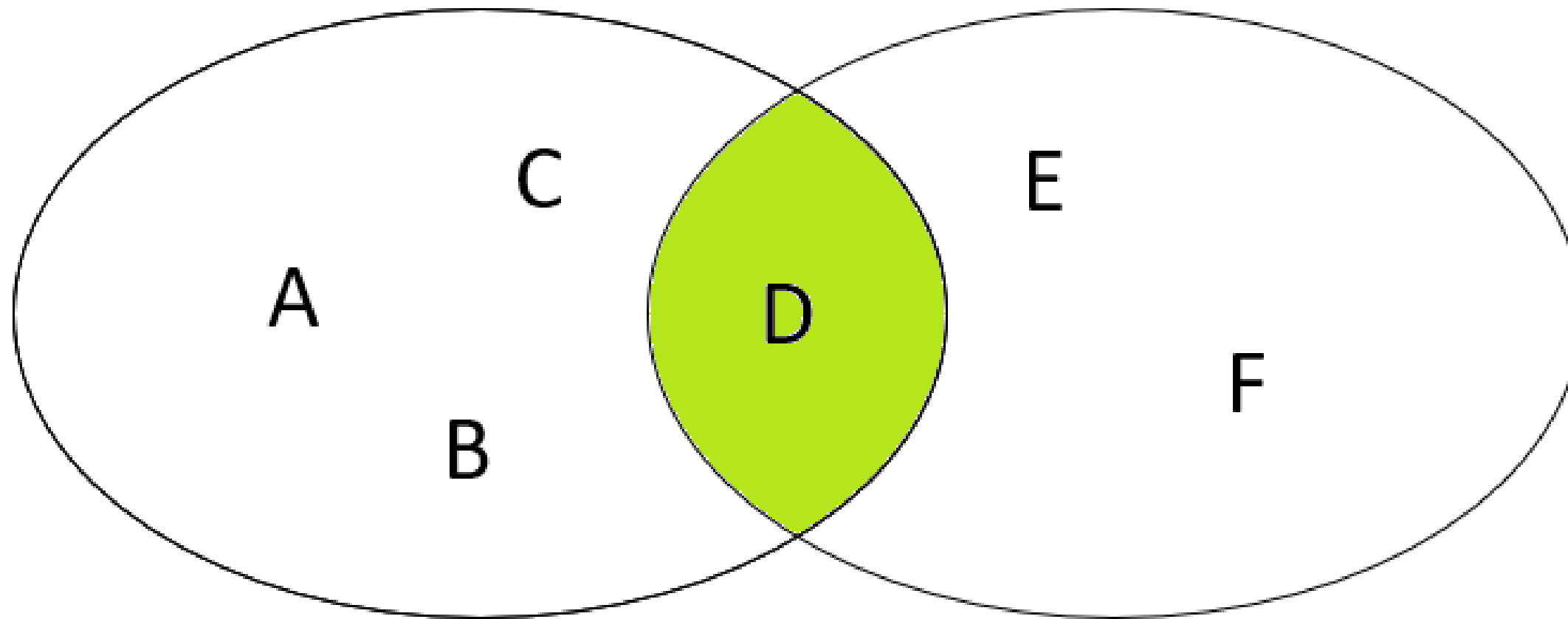
```
SELECT title,  
        genre,  
        renting_price  
FROM movies  
WHERE renting_price > 2.8  
UNION  
SELECT title,  
        genre,  
        renting_price  
FROM movies  
WHERE genre = 'Action & Adventure';
```

```
SELECT title,  
       genre,  
       renting_price  
FROM movies  
WHERE renting_price > 2.8  
UNION  
SELECT title,  
       genre,  
       renting_price  
FROM movies  
WHERE genre = 'Action & Adventure';
```

title	genre	renting_price
Fool's Gold	Action & Adventure	2.69
Astro Boy	Action & Adventure	2.89
Fair Game	Drama	2.89

INTERSECT

INTERSECT



Example - INTERSECT

```
SELECT title,  
       genre,  
       renting_price  
FROM movies  
WHERE renting_price > 2.8  
INTERSECT  
SELECT title,  
       genre,  
       renting_price  
FROM movies  
WHERE genre = 'Action & Adventure';
```

title	genre	renting_price
Astro Boy	Action & Adventure	2.89

Let's practice!

DATA-DRIVEN DECISION MAKING IN SQL