Subqueries

- Often an alternative to joins
- May be more intuitive or clearer

Set operations

Intersection

Let's use this to find the films with both Humphrey Bogart and Katharin Hepburn:

```
(

SELECT movieid
FROM actors
JOIN castings
ON actor.id = castings.actorid
WHERE actor.name = 'Humphrey Bogart'

)
INTERSECT
(

…
WHERE actor.name = 'Katharine Hepburn'

);
```

However, INTERSECTION is not supported by MySQL.

Union

UNION *is* supported by MySQL.

Let's list the ids of all actors whose name is 'Jack' or who have at least ten films to their credit:

```
( /*ids of actors named Jack*/
(
UNION
( /*ids of actors with at least ten films*/
);
```

For UNION, the subquery results must have the same number of columns and column types. The columns must also have the same names.

SQL normally allows duplicates, but UNION suppresses them by default. This makes sense with sets.

If you need duplicates, you can use UNION ALL.

What film has greatest score?

```
SELECT title, score
FROM movies
WHERE score =
( SELECT MAX(score)

FROM movies

);
```

The inner subquery will return the max score (wrapped in a 1x1 table), and the outer query uses this value in its WHERE clause.

Which actors appeared in "The Godfather"?

```
SELECT actorid
FROM castings
WHERE movieid =
( SELECT id

FROM movies
WHERE title = 'Godfather, The'

);
```

Note that if there's more than one movie with that name, we'll be comparing a value (movieid) with a list, which won't make sense with '='.

We can use IN instead, as described below.

Conditions Involving Relations

Tables with a single column are known as unary relations, essentially lists.

SQL provides some Boolean functions that operate on these:

- EXISTS R: true if R is not empty
- s IN R: true if s is one of the values in R
- s NOT IN R: opposite to above

- s > ALL R: true if s is greater than each and every value in R
- s > ANY R: true if s is greater than any one value in R
- These two above can use any comparative operator.

Subqueries returning multi-column tables

We can use subqueries instead of table names in FROM clauses.

In this case you must give the subquery an alias, so that you can refer to it in the SELECT and WHERE clauses.

What's the greatest number of films made by any actor?

```
SELECT MAX(num_films)
FROM
( SELECT actorid, COUNT(movieid) AS 'num_films'
FROM castings
GROUP BY actorid
) AS film_counts;
```

Note the subquery returns a two-column, multi-row table (actors and the number of their films there are).