Q4Me -1

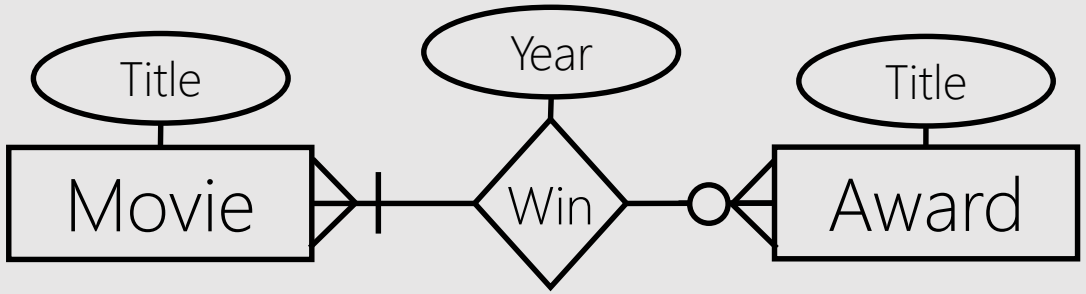
Book vs. Slides	W10: CH06 (1st & 2nd Ed.)
Lab	?
Last Weeks	?

Q4U

0

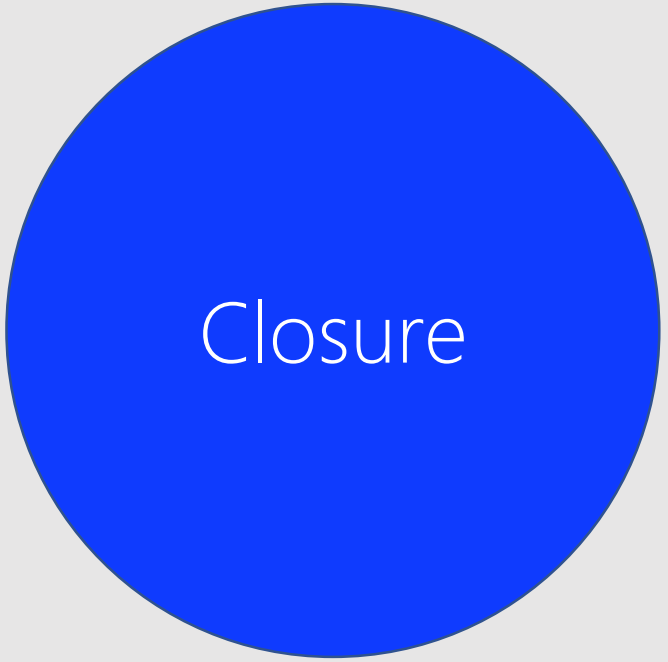
Given the following E/R, write SQL to answer the followings:

1- Which movie has won an Oscar award? [Movie Title, Award Title, Year]
2- Which movie has won nothing? [Movie Title]
3- Which movie has won all Oscar awards? [Movie Title, Award Title, Year]



Relational Algebra

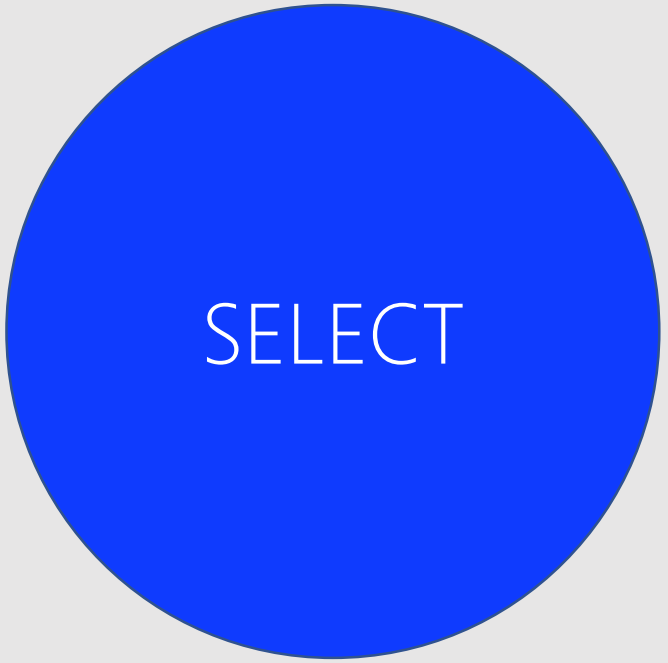
The result of relational operations on relations is also a relation.



Advanced SQL × Subquery

The result of SELECT on tables is also a table (temporary though) and can be used inside another query.

Subquery | Nested Query | Inner Query



SQL × DML × SELECT

1 (SELECT ...)
3 UNION
2 (SELECT ...)

1 (SELECT ...)
3 INTERSECT
2 (SELECT ...)

1 (SELECT ...)
3 EXCEPT
2 (SELECT ...)

Subquery × UNION

FirstName	LastName
Stanley	*Kubrick*
Clint	*Eastwood*
John	*Travolta*
Samuel	*Jackson*
Uma	*Thurman*

```
(SELECT FirstName, LastName FROM Director WHERE PlaceOfBirth='USA')
UNION
(SELECT FirstName, LastName FROM Actor WHERE PlaceOfBirth='USA')
```

~~SELECT FirstName, LastName FROM Director UNION Actor WHERE PlaceOfBirth='USA'~~ ✖

Subquery × UNION ALL

FirstName	LastName
Stanley	*Kubrick*
Clint	*Eastwood*
John	*Travolta*
Samuel	*Jackson*
Uma	*Thurman*
Clint	*Eastwood*

```
(SELECT FirstName, LastName FROM Director WHERE PlaceOfBirth='USA')
UNION ALL
(SELECT FirstName, LastName FROM Actor WHERE PlaceOfBirth='USA')
```

Subquery × INTERSECT

FirstName	LastName
Clint	*Eastwood*

```
(SELECT FirstName, LastName FROM Director)
INTERSECT
(SELECT FirstName, LastName FROM Actor)
```

~~SELECT FirstName, LastName FROM Director INTERSECT Actor~~ ❌

Subquery × EXCEPT

FirstName	LastName
Stanley	*Kubrick*
Alfred	*Hitchcock*

```
(SELECT FirstName, LastName FROM Director)
EXCEPT
(SELECT FirstName, LastName FROM Actor)
```

~~SELECT FirstName, LastName FROM Director EXCEPT Actor~~ ✗

Subquery × Bulk × INSERT

Subquery can be used to do bulk INSERT.

INSERT INTO *TableName*(*c*, *c'*, *c''*, ...) (SELECT ...);

Subquery × Bulk × INSERT

Subquery can be used to do bulk INSERT.

INSERT INTO *TableName*(*c*, *c'*, *c''*, ...) (SELECT ...);

- The number of columns must be same.
- The data type of columns must be compatible.

Subquery × Bulk × INSERT

Subquery can be used to do bulk INSERT.

INSERT INTO *TableName*(*c*, *c'*, *c''*, ...) (SELECT ...);

- The number of columns must be same.
- The data type of columns must be compatible.

INSERT INTO *TableName* (SELECT ...);

- If column list is omitted, the order in *TableName's definition is assumed*.

Subquery × Bulk × INSERT

Subquery can be used to do bulk INSERT.

INSERT INTO *TableName*(*c*, *c'*, *c''*, ...) (SELECT ...);

- The number of columns must be same.
- The data type of columns must be compatible.

INSERT INTO *TableName* (SELECT ...);

- If column list is omitted, the order in *TableName's definition is assumed*.

In SQLite, drop the parenthesis around subquery.

Subquery × Bulk × INSERT


```
INSERT INTO OldMovie
(SELECT * FROM Movie WHERE ReleaseDate < 1940)
```

Subquery × WHERE

Subquery can be used in WHERE clause of DML statements, i.e., SELECT, UPDATE, INSERT INTO, & DELETE FROM.

Herein, we use WHERE clause in SELECT statement as it is the most common DML.

Subquery × WHERE

4 SELECT Columns
1 FROM Tables
3 WHERE (c_1, c_2, …, c_n) OP (SELECT c'_1, c'_2, …, c'_n FROM …)
2

- OP ∈ {=, >, >=, <, <=, <>}
- Subquery cannot have more than one row, BUT can have multiple columns
- Subquery always on right side of OP.

Subquery × WHERE × ~~AGG~~

Movie				
Id	Title	Language	ReleaseDate	RunningTime
1	*2001: A Space Odyssey*	*English*	*1968*	*142*
2	*Rosemary's Baby*	*English*	*1968*	NULL
3	*The Birds*	*English*	*1963*	*119*
4	*Planet of the Apes*	*EN*	*1968*	*112*

What is the oldest movie?

```
SELECT *
FROM Movie
WHERE ~~ReleaseDate = MIN(ReleaseDate)~~
```

Subquery × WHERE × AGG

Movie				
Id	Title	Language	ReleaseDate	RunningTime
1	*2001: A Space Odyssey*	*English*	*1968*	*142*
2	*Rosemary's Baby*	*English*	*1968*	NULL
3	*The Birds*	*English*	*1963*	*119*
4	*Planet of the Apes*	*EN*	*1968*	*112*

What is the oldest movie?

```
SELECT *
FROM Movie
WHERE ReleaseDate = (SELECT MIN(ReleaseDate) FROM Movie)
```

Subquery × WHERE × ~~AGG~~

Movie				
Id	Title	Language	ReleaseDate	RunningTime
1	*2001: A Space Odyssey*	*English*	*1968*	*142*
2	*Rosemary's Baby*	*English*	*1968*	NULL
3	*The Birds*	*English*	*1963*	*119*
4	*Planet of the Apes*	*EN*	*1968*	*112*

What are the longest movies in 1963?

```
SELECT *
FROM Movie
WHERE ReleaseDate = 1963 AND
      ~~RunningTime = MAX(RunningTime)~~
```

Subquery × WHERE × AGG

Movie				
Id	Title	Language	ReleaseDate	RunningTime
1	*2001: A Space Odyssey*	*English*	*1968*	*142*
2	*Rosemary's Baby*	*English*	*1968*	NULL
3	*The Birds*	*English*	*1963*	*119*
4	*Planet of the Apes*	*EN*	*1968*	*112*

What are the longest movies in 1963?

```
SELECT *
FROM Movie
WHERE ReleaseDate = 1963 AND
        RunningTime = (SELECT MAX(RunningTime) FROM Movie)
```

Subquery × WHERE × AGG

Movie				
Id	Title	Language	ReleaseDate	RunningTime
1	*2001: A Space Odyssey*	*English*	*1968*	*142*
2	*Rosemary's Baby*	*English*	*1968*	NULL
3	*The Birds*	*English*	*1963*	*119*
4	*Planet of the Apes*	*EN*	*1968*	*112*

What are the longest movies in 1963?

```
SELECT *
FROM Movie
WHERE ReleaseDate = 1963
        RunningTime = (SELECT MAX(RunningTime)
                       FROM Movie
                       WHERE ReleaseDate = 1963)
```

Subquery × WHERE × AGG

Movie				
Id	Title	Language	ReleaseDate	RunningTime
1	*2001: A Space Odyssey*	*English*	*1968*	*142*
2	*Rosemary's Baby*	*English*	*1968*	NULL
3	*The Birds*	*English*	*1963*	*119*
4	*Planet of the Apes*	*EN*	*1968*	*112*

What is the oldest, yet longest movie?

```
SELECT *
FROM Movie
WHERE ReleaseDate = (SELECT MIN(ReleaseDate) FROM Movie) AND
        RunningTime = (SELECT MAX(RunningTime) FROM Movie)
```

Subquery × WHERE × AGG

Movie				
Id	Title	Language	ReleaseDate	RunningTime
1	*2001: A Space Odyssey*	*English*	*1968*	*142*
2	*Rosemary's Baby*	*English*	*1968*	NULL
3	*The Birds*	*English*	*1963*	*119*
4	*Planet of the Apes*	*EN*	*1968*	*112*

What is the oldest, yet longest movie?

```
SELECT *
FROM Movie
WHERE (ReleaseDate, RunningTime) = (SELECT MIN(ReleaseDate), MAX(RunningTime)
                                    FROM Movie)
```

Subquery × WHERE × AGG

Movie				
Id	Title	Language	ReleaseDate	RunningTime
1	*2001: A Space Odyssey*	*English*	*1968*	*142*
2	*Rosemary's Baby*	*English*	*1968*	NULL
3	*The Birds*	*English*	*1963*	*119*
4	*Planet of the Apes*	*EN*	*1968*	*112*

What movies are shorter than average length of movies in 1968?

```
SELECT *
FROM Movie
WHERE RunningTime < (SELECT AVG(RunningTime)
                     FROM Movie
                     WHERE ReleaseDate = 1968)
```

Subquery × WHERE × AGG

Movie				
Id	Title	Language	ReleaseDate	RunningTime
1	*2001: A Space Odyssey*	*English*	*1968*	*142*
2	*Rosemary's Baby*	*English*	*1968*	NULL
3	*The Birds*	*English*	*1963*	*119*
4	*Planet of the Apes*	*EN*	*1968*	*112*

What movies are shorter than average length of movies, in 1968?

Subquery × WHERE × AGG

Movie				
Id	Title	Language	ReleaseDate	RunningTime
1	*2001: A Space Odyssey*	*English*	*1968*	*142*
2	*Rosemary's Baby*	*English*	*1968*	NULL
3	*The Birds*	*English*	*1963*	*119*
4	*Planet of the Apes*	*EN*	*1968*	*112*

What movies are shorter than average length of movies, in 1968?

```
SELECT *
FROM Movie
WHERE ReleaseDate = 1968
        RunningTime < (SELECT AVG(RunningTime)
                       FROM Movie
                       WHERE ReleaseDate = 1968)
```

WHERE × IN

Movie				
Id	Title	Language	ReleaseDate	RunningTime
1	*2001: A Space Odyssey*	*English*	*1968*	*142*
2	*Rosemary's Baby*	*English*	*1968*	NULL
3	*The Birds*	*English*	*1963*	*119*
4	*Planet of the Apes*	*EN*	*1968*	*112*

Find the movies in English language?

```
SELECT *
FROM Movie
WHERE Language = 'English' OR
      Language = 'EN'
```

WHERE × IN

Movie				
Id	Title	Language	ReleaseDate	RunningTime
1	*2001: A Space Odyssey*	*English*	*1968*	*142*
2	*Rosemary's Baby*	*English*	*1968*	NULL
3	*The Birds*	*English*	*1963*	*119*
4	*Planet of the Apes*	*EN*	*1968*	*112*

Find the movies in English language?

```
SELECT *
FROM Movie
WHERE Language IN ('English', 'EN')
```

WHERE × IN

Movie				
Id	Title	Language	ReleaseDate	RunningTime
1	*2001: A Space Odyssey*	*English*	*1968*	*142*
2	*Rosemary's Baby*	*English*	*1968*	NULL
3	*The Birds*	*English*	*1963*	*119*
4	*Planet of the Apes*	*EN*	*1968*	*112*

Find the movies in non-English language?

```
SELECT *
FROM Movie
WHERE NOT (Language = 'English' OR
           Language = 'EN')
```

WHERE × IN

Movie				
Id	Title	Language	ReleaseDate	RunningTime
1	*2001: A Space Odyssey*	*English*	*1968*	*142*
2	*Rosemary's Baby*	*English*	*1968*	NULL
3	*The Birds*	*English*	*1963*	*119*
4	*Planet of the Apes*	*EN*	*1968*	*112*

Find the movies in non-English language?

```
SELECT *
FROM Movie
WHERE Language <> 'English' AND
      Language <> 'EN'
```

*De Morgan's law: NOT (A OR B) = NOT(A) AND NOT(B)

WHERE × IN

Movie				
Id	Title	Language	ReleaseDate	RunningTime
1	*2001: A Space Odyssey*	*English*	*1968*	*142*
2	*Rosemary's Baby*	*English*	*1968*	NULL
3	*The Birds*	*English*	*1963*	*119*
4	*Planet of the Apes*	*EN*	*1968*	*112*

Find the movies in non-English language?

```
SELECT *
FROM Movie
WHERE Language NOT IN ('English', 'EN')
```

Subquery × WHERE × IN

4 SELECT Columns
1 FROM Tables
3 WHERE (c_1, c_2, …, c_n) [NOT] IN (SELECT c'_1, c'_2, …, c'_n FROM …)
2

- Subquery can have multiple rows and columns.
- Subquery always on right side of IN.

Subquery × WHERE × IN

Director						
Id	FirstName	LastName	DateOfBirth	PlaceOfBirth	BestMovieId	MovieCount
1	Stanley	Kubrick	Jul. 26, 1928	USA	1	13
2	Alfred	Hitchcock	Aug. 13, 1899	England	NULL	47
3	Clint	Eastwood	May 31, 1930	USA	NULL	35

Movie			
Id	Title	Language	RunningTime
1	2001: A Space Odyssey	English	142
2	Rosemary's Baby	English	NULL

What movies are directors' best movies?

```
SELECT *
FROM Movie
WHERE Id IN (SELECT BestMovieId FROM Director)
```

Subquery × WHERE × IN

Director						
Id	FirstName	LastName	DateOfBirth	PlaceOfBirth	BestMovieId	MovieCount
1	Stanley	Kubrick	Jul. 26, 1928	USA	1	13
2	Alfred	Hitchcock	Aug. 13, 1899	England	NULL	47
3	Clint	Eastwood	May 31, 1930	USA	NULL	35

Movie			
Id	Title	Language	RunningTime
1	2001: A Space Odyssey	English	142
2	Rosemary's Baby	English	NULL

What movies are directors' best movies?

```
SELECT *
FROM Movie
WHERE Id = (SELECT BestMovieId FROM Director)
```

FROM × INNER JOIN

Director						
Id	FirstName	LastName	DateOfBirth	PlaceOfBirth	BestMovieId	MovieCount
1	*Stanley*	*Kubrick*	*Jul. 26, 1928*	*USA*	*1*	*13*
2	*Alfred*	*Hitchcock*	*Aug. 13, 1899*	*England*	NULL	*47*
3	*Clint*	*Eastwood*	*May 31, 1930*	*USA*	NULL	*35*

Movie			
Id	Title	Language	RunningTime
1	*2001: A Space Odyssey*	*English*	*142*
2	*Rosemary's Baby*	*English*	NULL

What movies are directors' best movies?

```
SELECT M.*
FROM Movie AS M
INNER JOIN Director AS D ON M.Id = D.BestMovieId
```

WHERE × IN vs. FROM × INNER JOIN

What movies are directors' best movies?

```
SELECT *
FROM Movie
WHERE Id IN (SELECT BestMovieId FROM Director)
```

Which one?

```
SELECT M.*
FROM Movie AS M
INNER JOIN Director AS D ON M.Id = D.BestMovieId
```

WHERE × IN vs. INTERSECT

FirstName	LastName
Clint	*Eastwood*

```
(SELECT FirstName, LastName FROM Director)
INTERSECT
(SELECT FirstName, LastName FROM Actor)
```

```
SELECT *
FROM Director
WHERE (FirstName, LastName) IN (SELECT FirstName, LastName FROM Actor)
```

MySQL does not support INTERSECT.

WHERE × IN vs. EXCEPT

FirstName	LastName
Stanley	*Kubrick*
Alfred	*Hitchcock*

```
(SELECT FirstName, LastName FROM Director)
EXCEPT
(SELECT FirstName, LastName FROM Actor)
```

```
SELECT *
FROM Director
WHERE (FirstName, LastName) NOT IN (SELECT FirstName, LastName FROM Actor)
```

MySQL does not support EXCEPT.

Subquery × WHERE × ANY | SOME

4 SELECT Columns
1 FROM Tables
3 WHERE c_1 OP ANY (SELECT c'_1 FROM ...)
2

The OP will be true if it is satisfied by one | more values in the subquery.

- OP ∈ {=, >, >=, <, <=, <>}
- Subquery can have multiple rows, BUT only one column.
- Subquery always on right side of ANY.
- ANY is same as SOME.

Subquery × WHERE × ANY | SOME

SELECT Columns
FROM Tables
WHERE c_1 = ANY (SELECT c'_1 FROM ...)

Same as:

SELECT Columns
FROM Tables
WHERE c_1 IN (SELECT c'_1 FROM ...)

Subquery × WHERE × ANY | SOME

Director						
Id	FirstName	LastName	DateOfBirth	PlaceOfBirth	BestMovieId	MovieCount
1	*Stanley*	*Kubrick*	*Jul. 26, 1928*	*USA*	*1*	*13*
2	*Alfred*	*Hitchcock*	*Aug. 13, 1899*	*England*	NULL	*47*
3	*Clint*	*Eastwood*	*May 31, 1930*	*USA*	NULL	*35*

What question this SQL statement is trying to answer?

```
(SELECT * FROM Director)
EXCEPT
(SELECT * FROM Director
 WHERE MovieCount < ANY (SELECT MovieCount FROM Director))
```

Subquery × WHERE × ANY | SOME

Director						
Id	FirstName	LastName	DateOfBirth	PlaceOfBirth	BestMovieId	MovieCount
1	*Stanley*	*Kubrick*	*Jul. 26, 1928*	*USA*	*1*	*13*
2	*Alfred*	*Hitchcock*	*Aug. 13, 1899*	*England*	NULL	*47*
3	*Clint*	*Eastwood*	*May 31, 1930*	*USA*	NULL	*35*

What question this SQL statement is trying to answer?

```
SELECT * FROM Director
WHERE MovieCount = (SELECT MAX(MovieCount) FROM Director)
```

Subquery × WHERE × ALL

4 SELECT Columns
1 FROM Tables
3 WHERE c_1 OP ALL (SELECT c'_1 FROM ...)
2

The OP will be true if it is satisfied by ALL values in the subquery.

- OP ∈ {=, >, >=, <, <=, <>}
- Subquery can have multiple rows, BUT only one column.
- Subquery always on right side of ALL.
- ANY is same as SOME.

Subquery × WHERE × ALL

What question this SQL statement is trying to answer?

```
SELECT * FROM Director
WHERE MovieCount > All (SELECT MovieCount
                        FROM Director
                        WHERE PlaceOfBirth = 'USA')
```

Subquery × WHERE × ALL

What question this SQL statement is trying to answer?

```
SELECT * FROM Director
WHERE MovieCount > All (SELECT MovieCount
                        FROM Director
                        WHERE PlaceOfBirth = 'USA')
```

```
SELECT * FROM Director
WHERE MovieCount > (SELECT MAX(MovieCount)
                    FROM Director
                    WHERE PlaceOfBirth = 'USA')
```

Subquery × WHERE × EXISTS

4 SELECT Columns
1 FROM Tables
3 WHERE [NOT] EXISTS (SELECT 1 | * | … FROM …)
2

The condition is true if |subquery|>0, i.e., at least one row in subquery.

- Subquery can have multiple rows and columns.
- Subquery always on right side of EXISTS.

The main use case is with *correlated* subquery, ending slides ☺

Subquery × FROM

3 SELECT Columns
1 FROM Tables, (SELECT …) AS S1, (SELECT …) AS S2, …
2 WHERE θ

Each subquery is assumed to be a Table.

- Subquery can have multiple rows and columns.
- Subquery always with alias, i.e., AS.

Subquery × FROM

5 SELECT Columns
1 FROM Table
2 INNER | LEFT | RIGHT OUTER JOIN (SELECT …) AS S ON θ'
3 WHERE θ

Each subquery is assumed to be a Table.

- Subquery can have multiple rows and columns.
- Subquery always with alias, i.e., AS.

Subquery × FROM

Movie				
Id	Title	Language	ReleaseDate	RunningTime
1	*2001: A Space Odyssey*	*English*	*1968*	*142*
2	*Rosemary's Baby*	*English*	*1968*	NULL
3	*The Birds*	*English*	*1963*	*119*
4	*Planet of the Apes*	*EN*	*1968*	*112*

Movies longer than the average movies in the same release year?

Subquery × FROM

Movies longer than the average movies in the same release year?

```
SELECT *
FROM Movie AS M
WHERE M.RunningTime > ?
```

Subquery × FROM

Movies longer than the average movies in the same release year?

```
SELECT ReleaseDate, AVG(RunningTime) AS AvgTime
FROM Movie
GROUP BY ReleaseDate
```

ReleaseDate	AvgTime
1968	*127*
1963	*119*

Subquery × FROM

Movie				
Id	Title	Language	ReleaseDate	RunningTime
1	*2001: A Space Odyssey*	*English*	*1968*	*142*
2	*Rosemary's Baby*	*English*	*1968*	NULL
3	*The Birds*	*English*	*1963*	*119*
4	*Planet of the Apes*	*EN*	*1968*	*112*

ReleaseDate	AvgTime
1968	*127*
1963	*119*

Movies longer than the average movies in the same release year?

Subquery × FROM

Movie						
Id	Title	Language	ReleaseDate	RunningTime	ReleaseDate	AvgTime
1	*2001: A Space Odyssey*	*English*	*1968*	*142*	*1968*	*127*
2	*Rosemary's Baby*	*English*	*1968*	NULL	*1968*	*127*
3	*The Birds*	*English*	*1963*	*119*	*1963*	*119*
4	*Planet of the Apes*	*EN*	*1968*	*112*	*1968*	*127*

Movies longer than the average movies in the same release year?

```
SELECT *
FROM Movie AS M, (SELECT ReleaseDate, AVG(RunningTime) AS AvgTime
                  FROM Movie
                  GROUP BY ReleaseDate) AS S
WHERE M.ReleaseDate = S. ReleaseDate
```

Subquery × FROM

Movie						
Id	Title	Language	ReleaseDate	RunningTime	ReleaseDate	AvgTime
1	*2001: A Space Odyssey*	*English*	*1968*	*142*	*1968*	*127*
2	*Rosemary's Baby*	*English*	*1968*	NULL	*1968*	*127*
3	*The Birds*	*English*	*1963*	*119*	*1963*	*119*
4	*Planet of the Apes*	*EN*	*1968*	*112*	*1968*	*127*

Movies longer than the average movies in the same release year?

```
SELECT *
FROM Movie AS M
INNER JOIN (SELECT ReleaseDate, AVG(RunningTime) AS AvgTime
            FROM Movie
            GROUP BY ReleaseDate) AS S ON M.ReleaseDate = S. ReleaseDate
```

Subquery × FROM

Movie						
Id	Title	Language	ReleaseDate	RunningTime	ReleaseDate	AvgTime
1	*2001: A Space Odyssey*	*English*	*1968*	*142*	*1968*	*127*
2	*Rosemary's Baby*	*English*	*1968*	NULL	*1968*	*127*
3	*The Birds*	*English*	*1963*	*119*	*1963*	*119*
4	*Planet of the Apes*	*EN*	*1968*	*112*	*1968*	*127*

Movies longer than the average movies in the same release year?

```
SELECT *
FROM Movie AS M
INNER JOIN (SELECT ReleaseDate, AVG(RunningTime) AS AvgTime
            FROM Movie
            GROUP BY ReleaseDate) AS S ON M.ReleaseDate = S. ReleaseDate
WHERE M.RunningTime > S.AvgTime
```

Correlated Subquery

A correlated subquery (aka synchronized subquery) is a subquery that uses columns of tables from the outer query.

Correlated Subquery

Movies longer than the average movies in the same release year?

```
SELECT *
FROM Movie AS M1
WHERE RunningTime > (SELECT AVG(RunningTime)
                     FROM Movie AS M2
                     WHERE M2.ReleaseDate = M1.ReleaseDate)
```

Correlated Subquery

Movies longer than the average movies in the same release year?

```
SELECT *
FROM Movie AS M1
WHERE RunningTime > (SELECT AVG(RunningTime)
                     FROM Movie AS M2
                     WHERE M2.ReleaseDate = M1.ReleaseDate)
```

Which one?

```
SELECT *
FROM Movie AS M
INNER JOIN (SELECT ReleaseDate, AVG(RunningTime) AS AvgTime
            FROM Movie
            GROUP BY ReleaseDate) AS S ON M.ReleaseDate = S. ReleaseDate
WHERE M.RunningTime > S.AvgTime
```

Correlated Subquery

The subquery is evaluated once for each row processed by the outer query, it is inefficient!

```
SELECT *
FROM Movie AS M1
WHERE RunningTime > (SELECT AVG(RunningTime)
                     FROM Movie AS M2
                     WHERE M2.ReleaseDate = M1.ReleaseDate)
```

```
SELECT *
FROM Movie AS M
INNER JOIN (SELECT ReleaseDate, AVG(RunningTime) AS AvgTime
            FROM Movie
            GROUP BY ReleaseDate) AS S ON M.ReleaseDate = S. ReleaseDate
WHERE M.RunningTime > S.AvgTime
```

Correlated Subquery

Update movie count for each director.

Correlated Subquery

Update movie count for each director.

```
UPDATE Director SET MovieCount = ? WHERE Id = ??
```

Correlated Subquery

Update movie count for each director.

```
SELECT D.Id, COUNT(*) FROM Director AS D
INNER JOIN MovieDirector AS MD ON D.Id = MD.DirectorId
GROUP BY D.Id
```

Correlated Subquery

Update movie count for each director.

```
UPDATE Director AS DD SET MovieCount = (
    SELECT COUNT(*) FROM Director AS D
    INNER JOIN MovieDirector AS MD ON D.Id = MD.DirectorId
    GROUP BY D.Id
    HAVING D.Id = DD.Id)
```

Correlated Subquery

Update movie count for each director.

```
UPDATE Director AS DD SET MovieCount = (
    SELECT COUNT(*) FROM Director AS D
    INNER JOIN MovieDirector AS MD ON D.Id = MD.DirectorId
    WHERE D.Id = DD.Id)
```

Correlated Subquery × WHERE × EXISTS 62

The main use case is with *correlated* subquery:

```
DELTE FROM Director AS D WHERE NOT EXISTS (
    SELECT 1 FROM MovieDirector AS MD ON D.Id = MD.DirectorId)
```

Subquery × Final Notes

- The ORDER BY clause may not be used in a subquery.
- The subquery cannot be used inside BETWEEN for outer query. However, the BETWEEN operator can be used within subquery.
- Likewise, subquery can be used in HAVING clause.
- There might be some SQL-92 support issues by different DBMSs.

Subquery × Division (/)

3- Which movie has won all Oscar awards?
[Movie Title, Award Title, Year]

Subquery × Division (/)

3- Which movie has won all Oscar awards?
[Movie Title, Award Title, Year]

A) Pair all movies with all awards by Cartesian product.
B) Find the actual awards of movies by INNER JOIN.
C) (A) EXCEPT (B): missing awards for movies.
D) (A) EXCEPT ((A) EXCEPT (B))