SQL × DML

Data Manipulation Language to

- INSERT ✗
- UPDATE ✗
- DELETE ✗
- [✓ SELECT]

from tables.

6

σ

SQL × DML × SELECT (Relational Algebra)

Operations in relational algebra , i.e.,

- Project(π)
- Select(σ)
- Rename(ρ)
- Union(∪)
- Set Difference(\)
- Cartesian Product(×)
- Intersection(∩), Division, Joins, ...

Only by one statement!

SQL × DML × SELECT × Project(π)

```
2 SELECT * | ColumnName1,
             ColumnName2,
             …
             ColumnNameN
1 FROM TableName
```

$$\pi_{ColumnName1,\ ColumnName2,\ \dots,\ ColumnNameN}(TableName)$$

SQL × DML × SELECT × Select(σ)

3 SELECT * | *ColumnName1*,
ColumnName2,
…
ColumnNameN

1 FROM *TableName*

2 WHERE θ

$$\pi_{ColumnName1,\ ColumnName2,\ \ldots,\ ColumnNameN}(\sigma_{\theta}(TableName))$$

SQL × DML × SELECT × Rename(ρ)

```
3 SELECT * | ColumnName1 AS ColumnAlias1,
             ColumnName2 AS ColumnAlias2,
             ...
             ColumnNameN AS ColumnAliasN
1 FROM TableName AS TableAlias
2 WHERE θ
```

$$\rho_{(ColumnAlias1/ColumnName1,\ \ldots)}(\pi_{ColumnName1,\ \ldots,\ ColumnNameN}(\sigma_{\theta}\ (\rho_{TableAlias}(TableName))))$$

SQL × DML × SELECT × Union(∪)

1 (SELECT ...) AS A1

3 UNION

2 (SELECT ...) AS A2

SQL × DML × SELECT × Intersection(∩) 81

1 (SELECT ...)

3 INTERSECT

2 (SELECT ...)

SQL × DML × SELECT × Set Diff(\ )

1 (SELECT ...) A

3 EXCEPT

2 (SELECT ...) B

A \ B

SQL × DML × SELECT × Product(×)

3 SELECT Column List

1 FROM *TableName1, TableName2, …, TableNameN*

2 WHERE θ

= AND >, <, <>

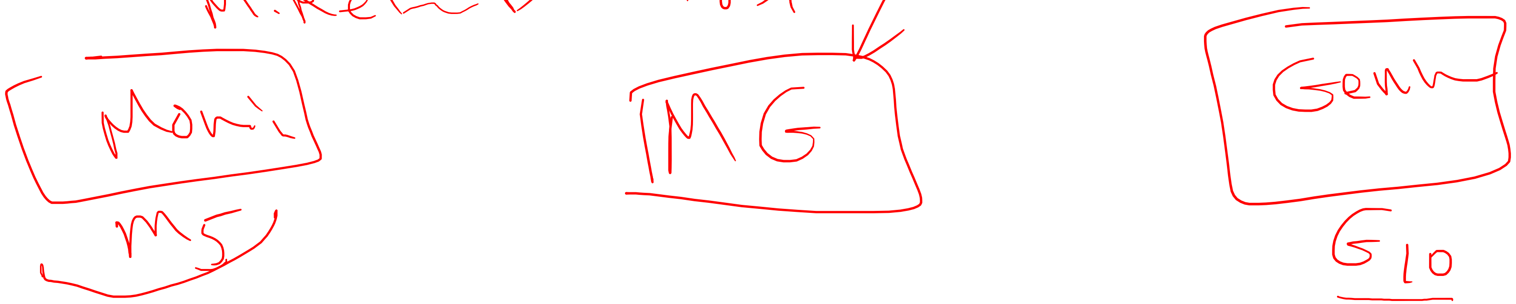
$$\pi_{<Column\ List>}(\sigma_{\theta}(TableName1 \times TableName2 \times \ldots \times TableNameN))$$

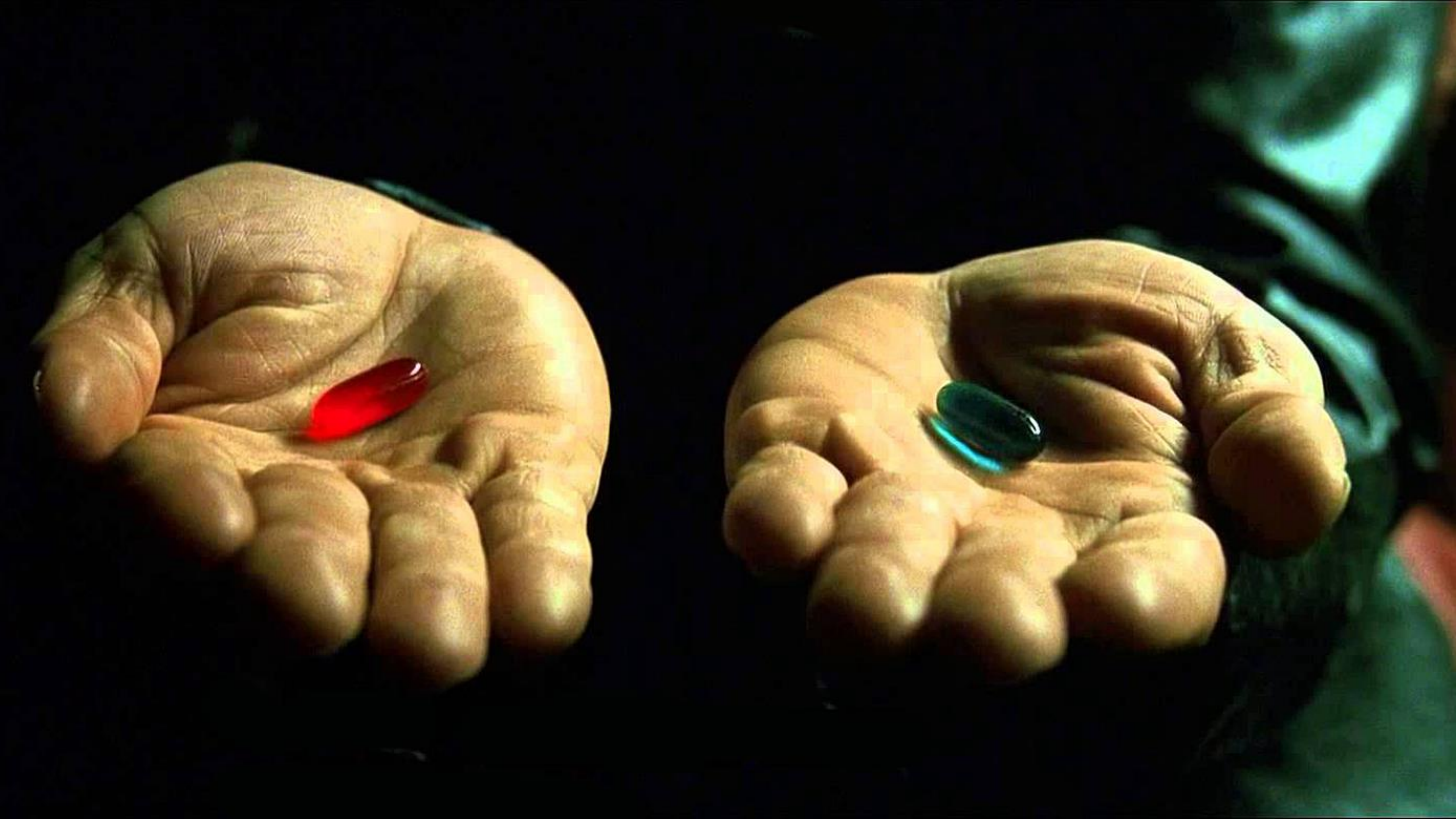
SQL × DML × SELECT × Product(×)

```
SELECT *
FROM Movie AS M, MovieGenre AS MG, Genre AS G
WHERE M.Id   =MG.MovieId AND
      G.Id   =MG.GenreId
```

~~M.Title~~

M.RelD = 1983





Advanced SQL × SELECT

5 SELECT DISTINCT Columns
1 FROM Tables
2 [WHERE θ]
3 [GROUP BY Columns]
4 [HAVING θ']
6 [ORDER BY Columns [ASC | DESC]]
7 [LIMIT # [OFFSET #]]

Advanced SQL × DISTINCT

3 SELECT DISTINCT Columns
1 FROM Table
2 [WHERE θ]

To eliminate duplicate tuples, considering all columns.

Advanced SQL × Math Operation

3 SELECT Column {+, -, *, /, %, ...} Column | Constant

1 FROM Tables

2 [WHERE θ]

To apply a function on each value of a column.

Advanced SQL × Built-in Function

3 SELECT FUNCTION(Column), ...
1 FROM Tables
2 [WHERE θ]

To apply a function on each value of a column.

Advanced SQL × AGG Function

```
3 SELECT COUNT | SUM | MAX | MIN | AVG(Column)
1 FROM Tables
2 [WHERE θ]
```

To apply AGGregation functions on non-NULL values of one column and return a single value.

Advanced SQL × AGG Function

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's the shortest, longest, and average length of movies have been made in 1968?

Title, Langu, ReleDate
Title

```
SELECT MIN(RunningTime)         AS Min,
       MAX(RunningTime)         AS Max,
       AVG(RunningTime)         AS Avg,
       SUM(RunningTime)         AS Sum,
       COUNT(RunningTime)       AS Count,
       COUNT(*)
1 FROM Movie
2 WHERE ReleaseDate = 1968
```

Min	Max	Avg	Sum	Count	Count(*)
112	*142*	~~*183*~~	*254*	*2*	3
		127	142 +112		

Advanced SQL × AGG × DISTINCT

```
3 SELECT COUNT | SUM | MAX | MIN | AVG(DISTINCT Column)
1 FROM Tables
2 [WHERE θ]
```

To apply AGG functions on non-NULL values of one column, after removing duplicates, and return a single value.

Advanced SQL × AGG × Math

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's the longest movie in hour which have been made in 1968?

```
SELECT MAX(RunningTime / 60) AS Max,
FROM Movie
WHERE ReleaseDate = 1968
```

Max
2

Advanced SQL × AGG × Built-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*	2022 *1963*	*119*
4	*Planet of the Apes*	*EN*	*1968*	*112*

How many years old is the oldest movie?

2022

```
SELECT MAX(STRFTIME('%Y', 'now') – ReleaseDate) AS Result
FROM Movie
WHERE ReleaseDate = 1968
```

2022, 11, 8

Result
56

59

Advanced SQL × WHERE

3 SELECT Columns

1 FROM Tables

2 [WHERE θ]

Advanced SQL × WHERE × Math
Advanced SQL × WHERE × Built-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*

List all movies which are older than 55 years?

2023

```
SELECT * FROM Movie
WHERE STRFTIME('%Y', 'now') – ReleaseDate > 55
```

Sel ST( )
From Mo

Advanced SQL × 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 which have been made in 1968?

SELECT * Max
FROM Movie
WHERE ReleaseDate = 1968 AND
~~RunningTime = MAX(RunningTime)~~

✗ X

SELE Max (RT) F

Mo

Advanced SQL × 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)~~ ✗

Advanced SQL × WHERE × NULL

NULL represents two facts about information

I) No Value, e.g., non-American directors do not have SSN

II) Missing | Unknow Value, e.g., PlaceOfBirth for a director might be missed, or unknown at the time of data entrance,

Advanced SQL × WHERE × NULL

Comparing with NULL value result in NULL, not FALSE, not TRUE!

NULL = NULL

```
SELECT NULL=1   AS Result;
SELECT NULL<>1 AS Result;
SELECT NULL>1   AS Result;
SELECT NULL>=1 AS Result;
...
```

Result
NULL

Advanced SQL × WHERE × IS NULL

To compare with NULL value explicitly, IS NULL | IS NOT NULL:

```
SELECT NULL IS NULL         AS Result1,
       NULL IS NOT NULL     AS Result2;
```

Result1	Result2
TRUE	FALSE

Advanced SQL × WHERE × IS NULL

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*

How many movies whose running time is missing?

```sql
SELECT COUNT(*) AS IncompleteMovieCount
FROM Movie
WHERE RunningTime IS NULL
```

IncompleteMovieCount
1

Advanced SQL × WHERE × LIKE

VARCHAR CHAR

To compare with CHAR-based values symbol, SQL has LIKE | NOT LIKE operator and pattern matching symbols:

I) %, Represents 0 or more of any CHAR, called wildcard

II) _ , Represents any single character

Title = 'Rosen'

Advanced SQL × WHERE × LIKE

e.g.,

Title LIKE 'H%' : matches values start with 'H' or 'h', the rest can be anything.

LIKE 'H_ _ _' : matches values with exactly four CHARs starting with 'H' or 'h'.

LIKE '%e' : matches values ends with 'e' or 'E'.

LIKE '%birds%': matches values containing 'birds', or 'BiRds'.

NOT LIKE 'H%': matches values whose first character is not 'H' or 'h'.

LIKE operator is <u>not case sensitive</u> for ASCII chars

Advanced SQL × WHERE × LIKE

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 movie 'space odyssey'?

```
SELECT *
FROM Movie
WHERE Title LIKE '%space odyssey%'
```

Advanced SQL × WHERE × LIKE

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 movie 'space odyssey'?

```
SELECT *
FROM Movie
WHERE Title = '%space odyssey%'
```

✗

Advanced SQL × FROM

3 SELECT Columns

1 FROM Tables

2 [WHERE θ]

Advanced SQL × FROM × Product(×)

3 SELECT Columns
1 FROM *Table1*, *Table2*, ..., *TableN*
2 WHERE θ

$$\pi_{<Columns>}(\sigma_{\theta}(Table1 \times Table2 \times \ldots \times TableNameN))$$

Advanced SQL × FROM × Product(×)

Movie				MovieGenre		Genre	
Id	Title	Language	RunningTime	MovieId	GenreId	Id	Title
1	*2001: A Space Odyssey*	*English*	*142*	*1*	*1*	*1*	*Sci-fi*
1	*2001: A Space Odyssey*	*English*	*142*	*1*	*3*	*3*	*Adventure*

Find genres of the movie ‘2001: A Space Odyssey'?

$$\sigma_{\text{Genre.Id=GenreId}}(\sigma_{\substack{\text{Movie.Id=MovieId AND}\\ \text{Title='2001: A Space Odyssey'}}}(\text{Movie}\times\text{MovieGenre}))\times\text{Genre})$$

Advanced SQL × FROM × Product(×)

Movie				MovieGenre		Genre	
Id	Title	Language	RunningTime	MovieId	GenreId	Id	Title
1	*2001: A Space Odyssey*	*English*	*142*	*1*	*1*	*1*	*Sci-fi*
1	*2001: A Space Odyssey*	*English*	*142*	*1*	*3*	*3*	*Adventure*

Find genres of the movie ‘2001: A Space Odyssey'?

$$\sigma_{\text{Genre.Id=GenreId}}(\sigma_{\substack{\text{Movie.Id=MovieId AND}\\ \text{Title='2001: A Space Odyssey'}}}(\text{Movie}\times\text{MovieGenre}))\times\text{Genre})$$

I) Corollary: $\sigma_{\theta}(\sigma_{\theta'}(\sigma_{\theta''}..._{}(\sigma_{\theta'''}(R)) = \sigma_{\theta\text{ AND }\theta'\text{AND }\theta''\text{... AND }\theta'''}(R)$

II) Product has commutative property, i.e., $(R1\times R2)\times R3= R1\times(R2\times R3)=R1\times R2\times R3$

Advanced SQL × FROM × Product(×)

Movie				MovieGenre		Genre	
Id	Title	Language	RunningTime	MovieId	GenreId	Id	Title
1	*2001: A Space Odyssey*	*English*	*142*	*1*	*1*	*1*	*Sci-fi*
1	*2001: A Space Odyssey*	*English*	*142*	*1*	*3*	*3*	*Adventure*

$$\sigma_{\substack{\text{Movie.Id=MovieId AND} \\ \text{Genre.Id=GenreId AND} \\ \text{Title='2001: A Space Odyssey'}}}(\text{Movie} \times \text{MovieGenre} \times \text{Genre})$$

Advanced SQL × FROM × Product(×)

Movie				MovieGenre		Genre	
Id	Title	Language	RunningTime	MovieId	GenreId	Id	Title
1	*2001: A Space Odyssey*	*English*	*142*	*1*	*1*	*1*	*Sci-fi*
1	*2001: A Space Odyssey*	*English*	*142*	*1*	*3*	*3*	*Adventure*

$$\sigma_{\substack{\text{Movie.Id=MovieId AND} \\ \text{Genre.Id=GenreId AND} \\ \text{Title='2001: A Space Odyssey'}}}(\text{Movie} \times \text{MovieGenre} \times \text{Genre})$$

```
SELECT *
FROM Movie, MovieGenre, Genre
WHERE Movie.Id      =MovieId AND
      Genre.Id      =GenreId AND
      Title         ='2001: A Space Odyssey'
```

Relational Algebra × θ-Join

$\bowtie_{\theta}$, θ-join, is product (×) of relations followed by selection (σ)

$$R1 \underset{\uparrow}{\bowtie}_{\theta} R2 = \underline{\sigma_{\theta}(R1 \times R2)}$$

Advanced SQL × FROM × INNER JOIN

3 SELECT Columns
1 FROM Table1, Table2
2 WHERE θ

$\sigma_{\theta}(R1 \times R2)$

------------------------------ SAME AS ------------------------------

3 SELECT Columns
1 FROM Table1
2 INNER JOIN Table2 ON θ

$R1 \bowtie_{\theta} R2$

Advanced SQL × FROM × INNER JOIN

Movie				MovieGenre		Genre	
Id	Title	Language	RunningTime	MovieId	GenreId	Id	Title
1	*2001: A Space Odyssey*	*English*	*142*	*1*	*1*	*1*	*Sci-fi*
1	*2001: A Space Odyssey*	*English*	*142*	*1*	*3*	*3*	*Adventure*

2

5

$$\sigma_{\text{Title='2001: A Space Odyssey'}}((\text{Movie} \bowtie_{\text{Movie.Id=MovieId}} \text{MovieGenre}) \bowtie_{\text{Genre.Id=GenreId}} \text{Genre})$$

SELECT *

FROM Movie

INNER JOIN MovieGenre ON Movie.Id = MovieId

INNER JOIN Genre ON Genre.Id = GenreId AND

~~WHERE~~ Title='2001: A Space Odyssey'

Advanced SQL × 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

3x6

1

What are directors' best movie name?

```
SELECT D.Id, FirstName, LastName, BestMovieId, M.Id, Title
FROM Director AS D, Movie AS M
WHERE M.Id = BestMovieId
```

Advanced SQL × 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 are directors' best movie name?

```
SELECT D.Id, FirstName, LastName, BestMovieId, M.Id, Title
FROM Director AS D
INNER JOIN Movie AS M ON M.Id = BestMovieId
```

Advanced SQL × FROM × INNER JOIN

Id	FirstName	LastName	BestMovieId	Id	Title
1	*Stanley*	*Kubrick*	*1*	*1*	*2001: A Space Odyssey*

What are directors' best movie name?

```
SELECT D.Id, FirstName, LastName, BestMovieId, M.Id, Title
FROM Director AS D
INNER JOIN Movie AS M ON M.Id = BestMovieId
```

Advanced SQL × FROM × LEFT JOIN

FirstName	LastName	BestMovieId	Id	Title
Stanley	*Kubrick*	*1*	*1*	*2001: A Space Odyssey*
Alfred	*Hitchcock*	NULL	NULL	NULL
Clint	*Eastwood*	NULL	NULL	NULL

What are directors' best movie name if any?

Director ⟕$_{\text{BestMovieId}=\text{Movie.Id}}$ Movie

```sql
SELECT D.Id, FirstName, LastName, BestMovieId, M.Id, Title
FROM Director AS D
LEFT [OUTER] JOIN Movie AS M ON M.Id = BestMovieId
```

Advanced SQL × FROM × RIGHT JOIN

FirstName	LastName	BestMovieId	Id	Title
Stanley	*Kubrick*	*1*	*1*	*2001: A Space Odyssey*
NULL	NULL	NULL	*2*	*Rosemary's Baby*

List all movies and identify whether each one is the best of its director?

Director ⟖$_{\text{BestMovieId=Movie.Id}}$ Movie

From Movie
Left Join D

```
SELECT D.Id, FirstName, LastName, BestMovieId, M.Id, Title
FROM Director AS D
RIGHT [OUTER] JOIN Movie AS M ON M.Id = BestMovieId
```

Advanced SQL × FROM × FULL JOIN

FirstName	LastName	BestMovieId	Id	Title
Stanley	*Kubrick*	*1*	*1*	*2001: A Space Odyssey*
Alfred	*Hitchcock*	NULL	NULL	NULL
Clint	*Eastwood*	NULL	NULL	NULL
NULL	NULL	NULL	*2*	*Rosemary's Baby*

NULL
NULL
NULL
N

$\text{Director} ⟗_{\text{BestMovieId=Movie.Id}} \text{Movie}$

```
SELECT D.Id, FirstName, LastName, BestMovieId, M.Id, Title , NULL
① FROM Director AS D
② FULL [OUTER] JOIN Movie AS M ON M.Id = BestMovieId
WHERE PlaceofBn = 'USA'
```

Advanced SQL × FROM × RIGHT JOIN

Advanced SQL × FROM × FULL JOIN

NOT currently supported in SQLite!
Think about a workaround & bring it with you next week.

m1
m2
m3
b1
b2
b3
m1 b1
m2 b1
m1 b1
m2 b
m2 NULL
NULL b2
NULL b3
NULL
U
U

Advanced SQL × SELECT

5 SELECT DISTINCT Columns

1 FROM Tables

2 [WHERE θ]

3 [GROUP BY Columns]

4 [HAVING θ']

AVG

πM Mov →

Advanced SQL × GROUP BY

To group tuples based on values on columns, usually followed by AGG functions.

Advanced SQL × GROUP BY

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*	*English*	*1968*	*112*

How many movies have been made in each year?

```
SELECT ReleaseDate, COUNT(*)
FROM Movie
GROUP BY ReleaseDate
```

AV(, RunniTime

ReleaseDate	COUNT(*)
1968	*3*
1963	*1*

142+~~NULL~~+112

AVG(Run

119

Advanced SQL × GROUP BY

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*

How many movies have been made in each year per language?

```
SELECT ReleaseDate, Language, COUNT(*)
FROM Movie
GROUP BY ReleaseDate, Language
```

ReleaseDate	Language	COUNT(*)
1968	*English*	*2*
1968	*EN*	*1*
1963	*English*	*1*

Advanced SQL × GROUP BY

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*

How many movies have been made in each year per language?

SELECT ~~Title~~, ReleaseDate, Language, COUNT(*)
FROM Movie
GROUP BY ReleaseDate, Language

ReleaseDate	Language	COUNT(*)
1968	*English*	*2*
1968	*EN*	*1*
1963	*English*	*1*

SELECT clause only accepts AGG or columns in GROUP BY list.

Advanced SQL × GROUP BY × HAVING

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*	*English*	*1968*	*112*

In which years more than 2 movies have been released?

```
SELECT ReleaseDate, COUNT(*)
FROM Movie
GROUP BY ReleaseDate
HAVING COUNT(*) > 2
```

ReleaseDate	COUNT(*)
1968	*3*

HAVING only accepts AGG or columns in GROUP BY list.

Advanced SQL × Sorting

5 SELECT DISTINCT Columns
1 FROM Tables
2 [WHERE θ]
3 [GROUP BY Columns]
4 [HAVING θ']
6 [ORDER BY Columns [ASC | DESC]]

Advanced SQL × Sorting

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

List movies sorted by release date & running time?

```
SELECT *
FROM Movie
ORDER BY ReleaseDate DESC, RunningTime
```

Advanced SQL × Paging

5 SELECT DISTINCT Columns

1 FROM Tables

2 [WHERE θ]

3 [GROUP BY Columns]

4 [HAVING θ']

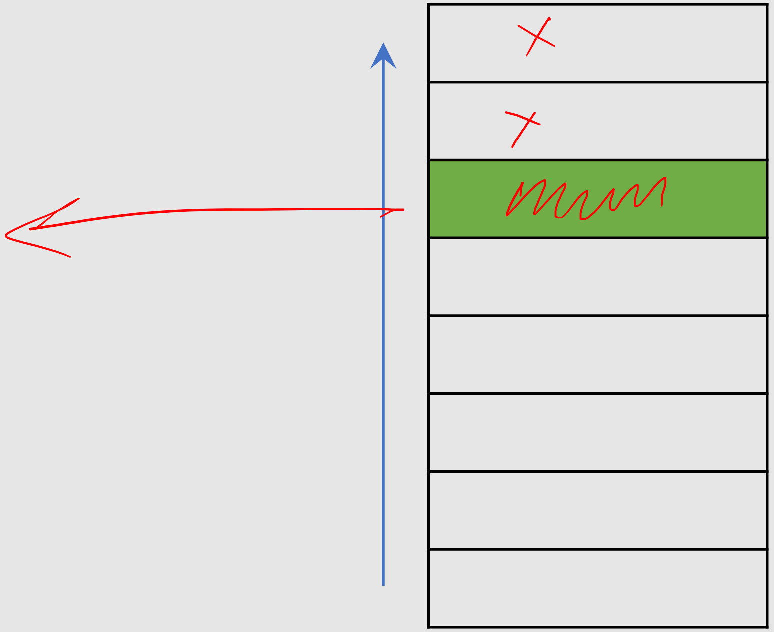
6 [ORDER BY Columns [ASC | DESC]]

7 [LIMIT # [OFFSET #]]

Advanced SQL × Paging

Page 3 of recent movies? (each page shows 10 movies)
Top-10 recent movies after skipping the top-2×10 most recent ones?

```
SELECT *
FROM Movie
ORDER BY ReleaseDate DESC
LIMIT 10 OFFSET 2×10
```



Advanced SQL × Paging

Page *n* of recent items? (each page shows N items)
Top-N recent items after skipping the top-(n-1)×N most recent ones?

```
SELECT *
FROM Items
ORDER BY Date DESC
LIMIT N OFFSET (n-1)×N
```

Advanced SQL × Paging

i = 1 → 100

Page *n* of expensive items? (each page shows N items)
Top-N expensive items after skipping the top-(n-1)×N most expensive?

```
SELECT *
FROM Items
ORDER BY Price DESC
LIMIT N OFFSET (n-1)×N
```