SQL Tutorial

CS 151: Privacy, Security, and Data

Slides Courtesy of Johes Bater and Nazli Goharian

September 28 2022

Agenda

Paper Presentations

SQL Primer

Agenda

Paper Presentations

SQL Primer

Structured Query Language

Brief History

- SQUARE (1975) - first relational query language
- First commercial implementation by Oracle (1979)
- SQL/DS (1981) from IBM

Uses

- Data Definition Language (e.g. CREATE, ALTER)
- Data Manipulation Language (e.g. SELECT, INSERT)

Basic SQL Query

SELECT *target-list*
FROM *relation-list*
WHERE *qualifiers*;

Table: connections

ID	IP	OS	Timestamp
rthomp06	192.168.10.1	Mac OS	1663926732
rthomp06	192.168.235.12	iOS	1661539932
jbater	192.168.80.5	Mac OS	1664376732

Basic SQL Query

SELECT *target-list*
FROM *relation-list*
WHERE *qualifiers*;

Targets – also called columns or attributes for the table

Table: connections

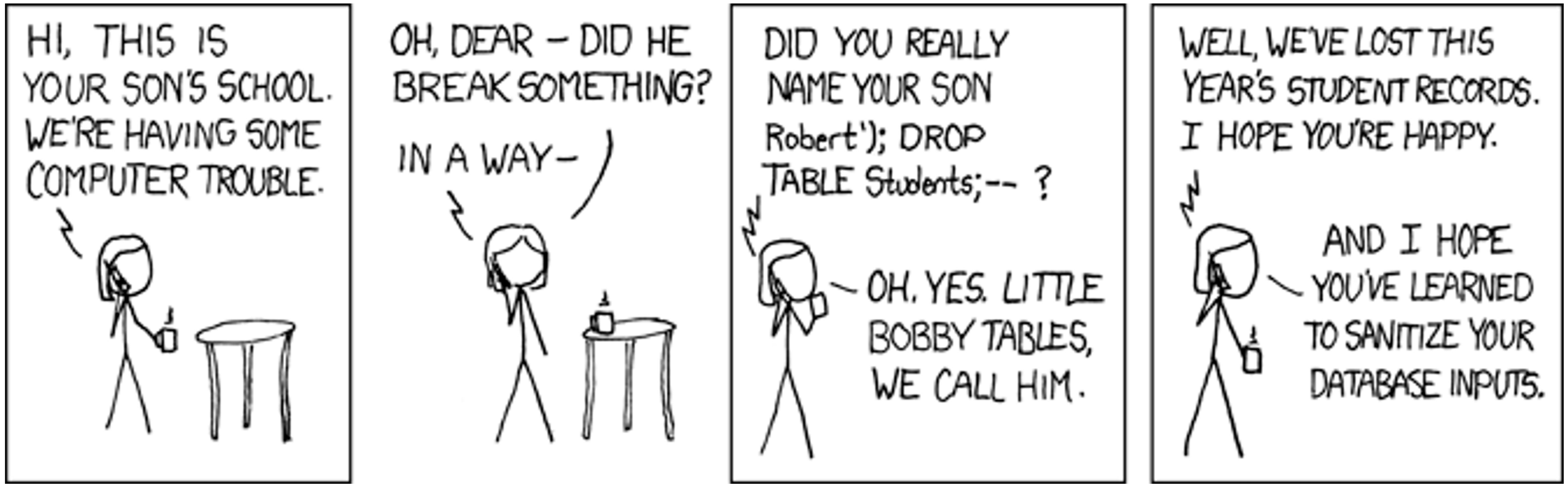
ID	IP	OS	Timestamp
rthomp06	192.168.10.1	Mac OS	1663926732
rthomp06	192.168.235.12	iOS	1661539932
jbater	192.168.80.5	Mac OS	1664376732

Basic SQL Query

SELECT *target-list*
FROM *relation-list*
WHERE *qualifiers*;

Semicolon signifies end of query

Be careful - SQL Injection



Basic SQL Query

```
SELECT *
FROM    connections;
```

* == *everything*

Table: connections

ID	IP	OS	Timestamp
rthomp06	192.168.10.1	Mac OS	1663926732
rthomp06	192.168.235.12	iOS	1661539932
jbater	192.168.80.5	Mac OS	1664376732

Results returned

ID	IP	OS	Timestamp
rthomp06	192.168.10.1	Mac OS	1663926732
rthomp06	192.168.235.12	iOS	1661539932
jbater	192.168.80.5	Mac OS	1664376732

Basic SQL Query

```
SELECT ID
FROM   connections
WHERE  Timestamp > 1663000000;
```

Table: connections

ID	IP	OS	Timestamp
rthomp06	192.168.10.1	Mac OS	1663926732
rthomp06	192.168.235.12	iOS	1661539932
jbater	192.168.80.5	Mac OS	1664376732

Results returned

ID
rthomp06
jbater

Basic SQL Query

```
SELECT ID
FROM    connections
WHERE  Timestamp > 1663000000
AND ID = ‘jbater’;
```

Table: connections

ID	IP	OS	Timestamp
rthomp06	192.168.10.1	Mac OS	1663926732
rthomp06	192.168.235.12	iOS	1661539932
jbater	192.168.80.5	Mac OS	1664376732

Results returned

ID
jbater

Basic SQL Query

```
SELECT DISTINCT IP
FROM   connections;
```

DISTINCT finds unique values

Table: connections

ID	IP	OS	Timestamp
rthomp06	192.168.10.1	Mac OS	1663926732
rthomp06	192.168.235.12	iOS	1661539932
jbater	192.168.80.5	Mac OS	1664376732

Results returned

IP
192.168.10.1
192.168.235.12
192.168.80.5

Basic SQL Query

```
SELECT DISTINCT ID
FROM    connections;
```

Table: connections

ID	IP	OS	Timestamp
rthomp06	192.168.10.1	Mac OS	1663926732
rthomp06	192.168.235.12	iOS	1661539932
jbater	192.168.80.5	Mac OS	1664376732

Results returned

ID
rthomp06
jbater

Basic SQL Query

```
SELECT DISTINCT ID
FROM    connections
WHERE ID LIKE ‘r_%;
```

LIKE is for string matching

`_` single character

`%` 0 or more characters

Table: connections

ID	IP	OS	Timestamp
rthomp06	192.168.10.1	Mac OS	1663926732
rthomp06	192.168.235.12	iOS	1661539932
jbater	192.168.80.5	Mac OS	1664376732

Results returned

ID
rthomp06

Aggregate Operators

MIN
MAX
AVG
SUM
COUNT

Aggregate Queries

Table: connections

ID	IP	OS	Timestamp	Duration
rthomp06	192.168.10.1	Mac OS	1663926732	43482
rthomp06	192.168.235.12	iOS	1661539932	30
jbater	192.168.80.5	Mac OS	1664376732	520

```
SELECT ID, AVG(Duration)
FROM    connections;
```

Results returned

ID	AVG(Duration)
rthomp06	21756
jbater	520

Aggregate Queries

Table: connections

ID	IP	OS	Timestamp	Duration
rthomp06	192.168.10.1	Mac OS	1663926732	43482
rthomp06	192.168.235.12	iOS	1661539932	30
jbater	192.168.80.5	Mac OS	1664376732	520

```
SELECT COUNT(DISTINCT ID)
FROM   connections;
```

Results returned

COUNT(DISTINCT ID)
2

Other Operators

GROUP BY - Group results by a column(s)

HAVING - Restrict based on a grouping

ORDER BY - Sort the results

UNION - Union of two sets (removes dupes)

INTERSECT - Intersection of two sets

Aggregation (GROUP BY/HAVING)

SELECT *target-list*
FROM *relation-list*
WHERE *qualifiers*
GROUP BY *grouping-list*
HAVING *group-qualification*;

Note: Columns selected (in target-list) that are not in GROUP BY must use an aggregation operations

Evaluating an Aggregation

- Steps
  - Remove ‘unnecessary’ fields for relations in the relation-list
  - Compute the cross-product of resulting projections
  - Discard tuples that fail qualification in WHERE clause
  - Partition the remaining tuples into groups by the value combination of attributes in grouping-list.
  - Apply the group-qualification is then applied to eliminate some groups. Expressions in group-qualification must have a single value per group!

- An attribute in group-qualification that is not an argument of an aggregate function must also appear in grouping-list.

- One answer tuple is generated per qualifying group.

GROUP BY Example

```
SELECT Status, COUNT(*)
FROM users u
GROUP BY Status;
```

Table: users

ID	FirstName	LastName	Status
rthomp06	Ron	Thompson	Student
jbater	Johes	Bater	Professor

Note: Using COUNT(*) is good practice in case there are NULLS in one column

HAVING Example

```
SELECT Status, COUNT(*)
FROM users u
GROUP BY Status
HAVING COUNT(*) > 1;
```

Table: users

ID	FirstName	LastName	Status
rthomp06	Ron	Thompson	Student
jbater	Johes	Bater	Professor

ORDER BY Example

```
SELECT Status, ID, AVG(Duration)
FROM connections c, users u
ON c.ID = u.ID
GROUP BY Status, ID
ORDER BY Status ASC, AVG(Duration) DESC
```

Note: You can specify direction using (ASC/DESC) or leave blank (defaults to ASC)

UNION Example

```
SELECT ID
FROM current_users
UNION
SELECT ID
FROM inactive_users
```

NOTE: Make sure that you have the same column names to ensure no weirdness

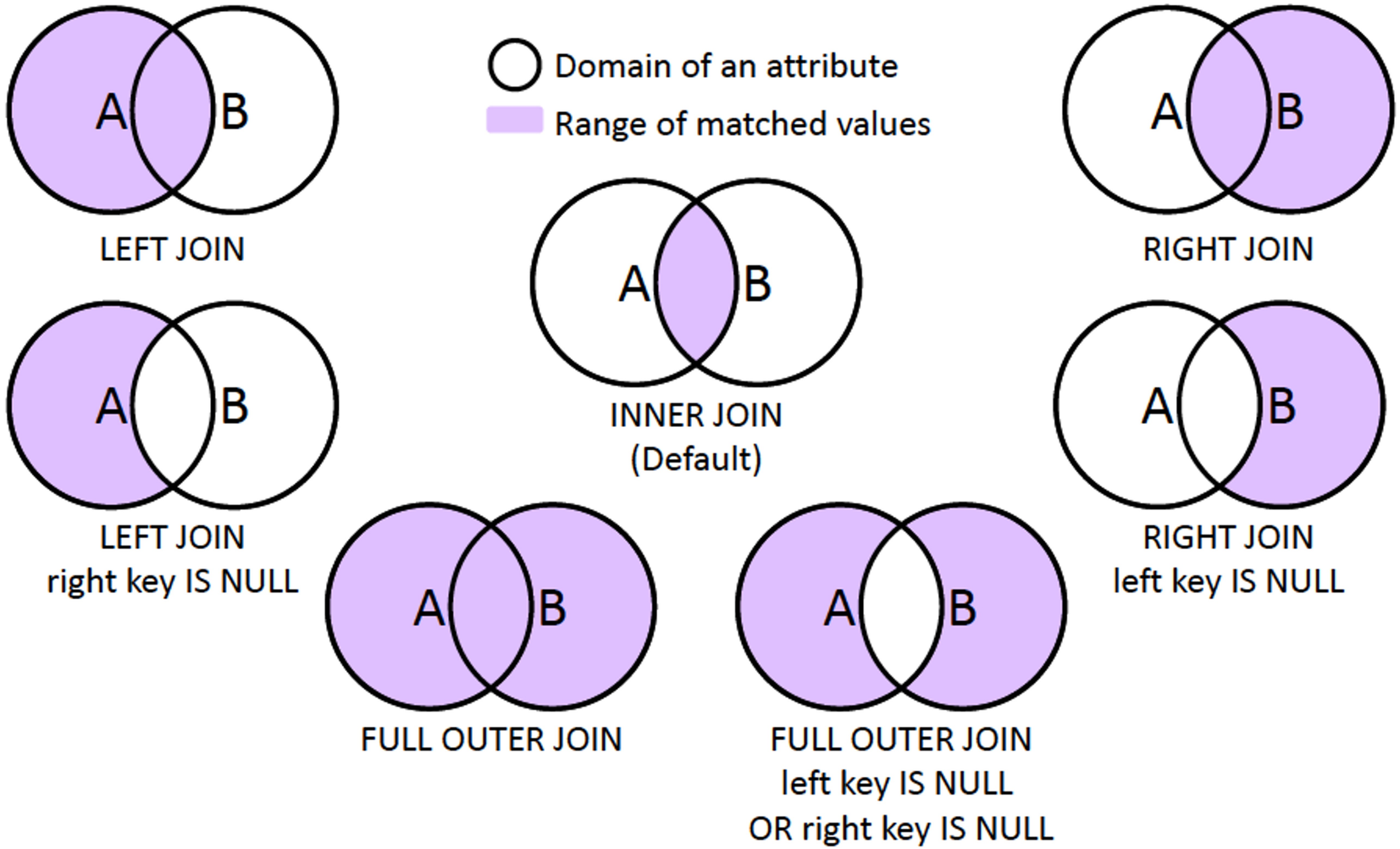
INTERSECT Example

SELECT ID
FROM current_users
INTERSECT
SELECT ID
FROM inactive_users

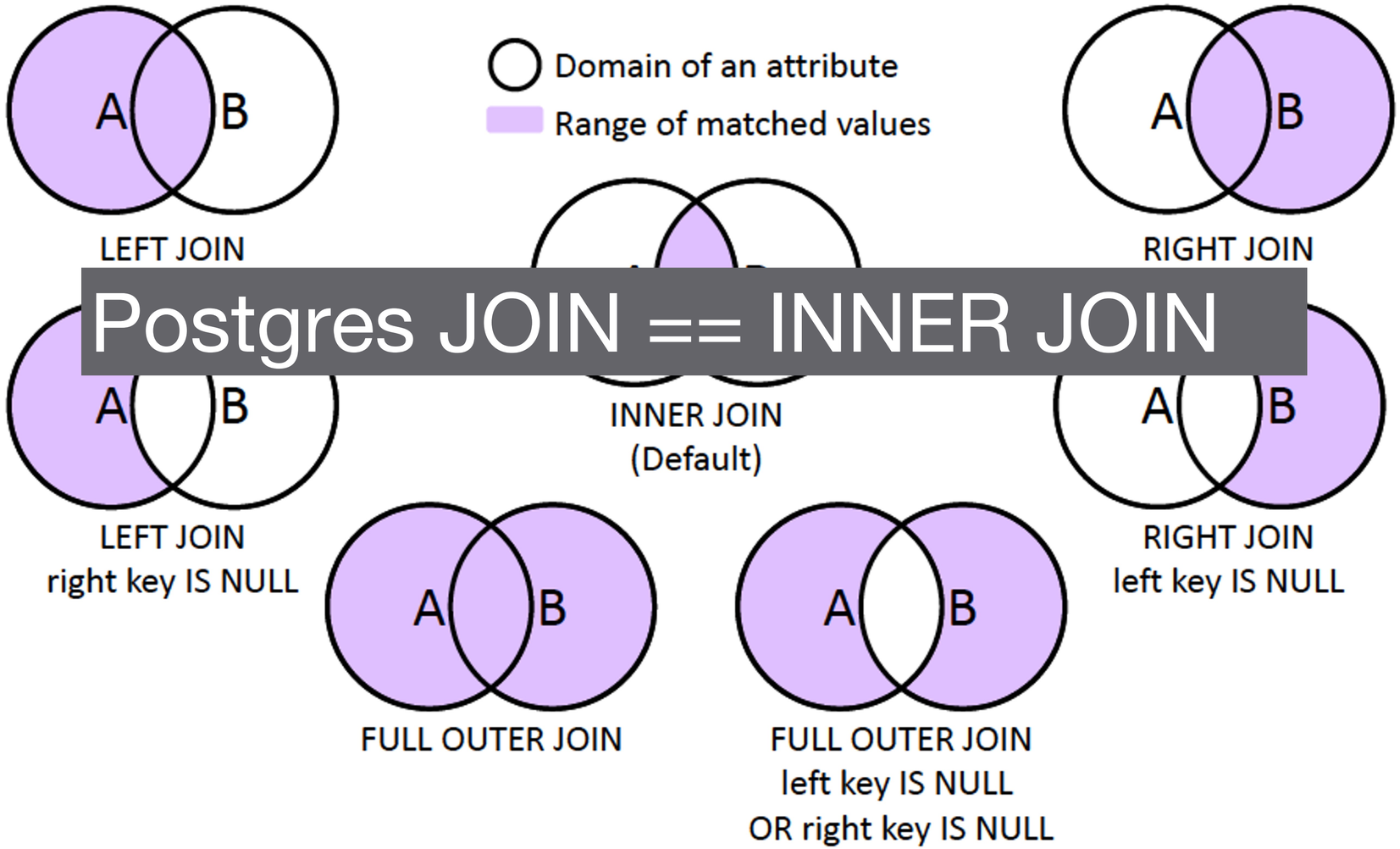
^ *WOULD RETURN NOTHING*

NOTE: Make sure that you have the same column names to ensure no weirdness
Also able to use IN/NOT IN

JOINS


Domain of an attribute
Range of matched values
A
B
LEFT JOIN
INNER JOIN
(Default)
RIGHT JOIN
LEFT JOIN
right key IS NULL
FULL OUTER JOIN
FULL OUTER JOIN
left key IS NULL
OR right key IS NULL
RIGHT JOIN
left key IS NULL

JOINS



Postgres JOIN == INNER JOIN

JOINS Example Query

Table: users

ID	FirstName	LastName	Status
rthomp06	Ron	Thompson	Student
jbater	Johes	Bater	Professor

Table: connections

ID	IP	OS	Timestamp
rthomp06	192.168.10.1	Mac OS	1663926732
rthomp06	192.168.235.12	iOS	1661539932
jbater	192.168.80.5	Mac OS	1664376732

```
SELECT u.FirstName, u.LastName, MAX(c.Timestamp) AS LastLogin
FROM    connections AS c
INNER JOIN users AS u
ON c.ID = u.ID
WHERE Status = ‘Student’
GROUP BY u.FirstName, u.LastName;
```

JOINS Example Query

Table: users

ID	FirstName	LastName	Status
rthomp06	Ron	Thompson	Student
jbater	Johes	Bater	Professor

Table: connections

ID	IP	OS	Timestamp
rthomp06	192.168.10.1	Mac OS	1663926732
rthomp06	192.168.235.12	iOS	1661539932
jbater	192.168.80.5	Mac OS	1664376732

```
SELECT u.FirstName, u.LastName, MAX(c.Timestamp) AS LastLogin
FROM    connections AS c
INNER JOIN users AS u
ON c.ID = u.ID
WHERE Status = ‘Student’
GROUP BY u.FirstName, u.LastName;
```

ALIAS

JOINS Example Query

Table: users

ID	FirstName	LastName	Status
rthomp06	Ron	Thompson	Student
jbater	Joh		

Table: connections

ID	IP	OS	Timestamp
rthomp06	192.168.10.1	Mac OS	1663926732
			1661539932
			1664376732

FirstName	LastName	LastLogin
Ron	Thompson	1661539932

```
SELECT u.FirstName, u.LastName, MAX(c.Timestamp) AS LastLogin
FROM    connections AS c
INNER JOIN users AS u
ON c.ID = u.ID
WHERE Status = 'Student'
GROUP BY u.FirstName, u.LastName;
```

ALIAS

JOINS Example Query

Table: users

ID	FirstName	LastName	Status
rthomp06	Ron	Thompson	Student
jbater	Johes	Bater	Professor

Table: connections

ID	IP	OS	Timestamp
rthomp06	192.168.10.1	Mac OS	1663926732
rthomp06	192.168.235.	iOS	1661539932
			1664376732

EQUIVALENT

```
SELECT FirstName, LastName, MAX(Timestamp)
FROM    connections
JOIN users
ON connections.ID = users.ID
GROUP BY u.FirstName, u.LastName;
```

JOINS Example Query

Table: users

ID	FirstName	LastName	Status
rthomp06	Ron	Thompson	Student
jbater	Johes	Bater	Professor

Table: connections

ID	IP	OS	Timestamp
rthomp06	192.168.10.1	Mac OS	1663926732
rthomp06	192.168.235.	iOS	1661539932
	.5	Mac OS	1664376732

ALSO EQUIVALENT

```
SELECT FirstName, LastName, MAX(Timestamp)
FROM    connections c, users u
WHERE c.ID = u.ID
GROUP BY u.FirstName, u.LastName;
```

(Unnecessarily) Complex Example

```
SELECT FirstName, LastName, Timestamp AS LastLogin
FROM connections AS c
INNER JOIN users AS u
ON c.ID = u.ID
WHERE Status = ‘Student’
AND c.ID = (SELECT ID
                    FROM (SELECT ID, MAX(Timestamp)
                                FROM connections
                                GROUP BY ID)
            );
```

(Unnecessarily) Complex Example

EQUIVALENT

```
SELECT u.FirstName, u.LastName, MAX(c.Timestamp) AS LastLogin
FROM    connections AS c
INNER JOIN users AS u
ON c.ID = u.ID
WHERE Status = ‘Professor’;
```

Debugging

- Use EXPLAIN ANALYZE to isolate your query if it hangs.
- Remember that NULLs change our results, e.g., COUNT(*)
- Good coding style (aliases, smaller blocks of logic) will make debugging SQL easier
  - Can leverage temporary tables

SQL Flavors

PostgreSQL (what we are using)

MySQL

Oracle SQL

MSSQL (use T-SQL)

Hive (Hadoop based)

NOTE: There are some slight syntax differences between different flavors

Some tools

pgAdmin

DBeaver

Navicat ($$$)

SparkSQL

pandas (Python package)