Joins, Unions and Subqueries

Pinal Dave
http://blog.sqlauthority.com
@pinaldave

pluralsight
hardcore developer training

Prerequisites

- MySQL Server
- MySQL Workbench or any other IDE
- Sample Database – sakila

MySQL Fundamentals

MySQL is a popular choice of database for use in web applications, and is a central component of the widely used LAMP open source web application software stack. This course covers the fundamentals of MySQL, including how to install MySQL as well as write basic data retrieval and data modification queries.

A SQL JOIN combines columns from two or more tables in a single result set.

Joins

- **Inner Join**
- **Outer Join**
  - Left Outer Join
  - Right Outer Join
  - Full Outer Join
- **Cross Join**

- **Self Join**
- **Equi Join**
- **Natural Join**
- **Straight Join***

* Discussed in Performance Tuning Course

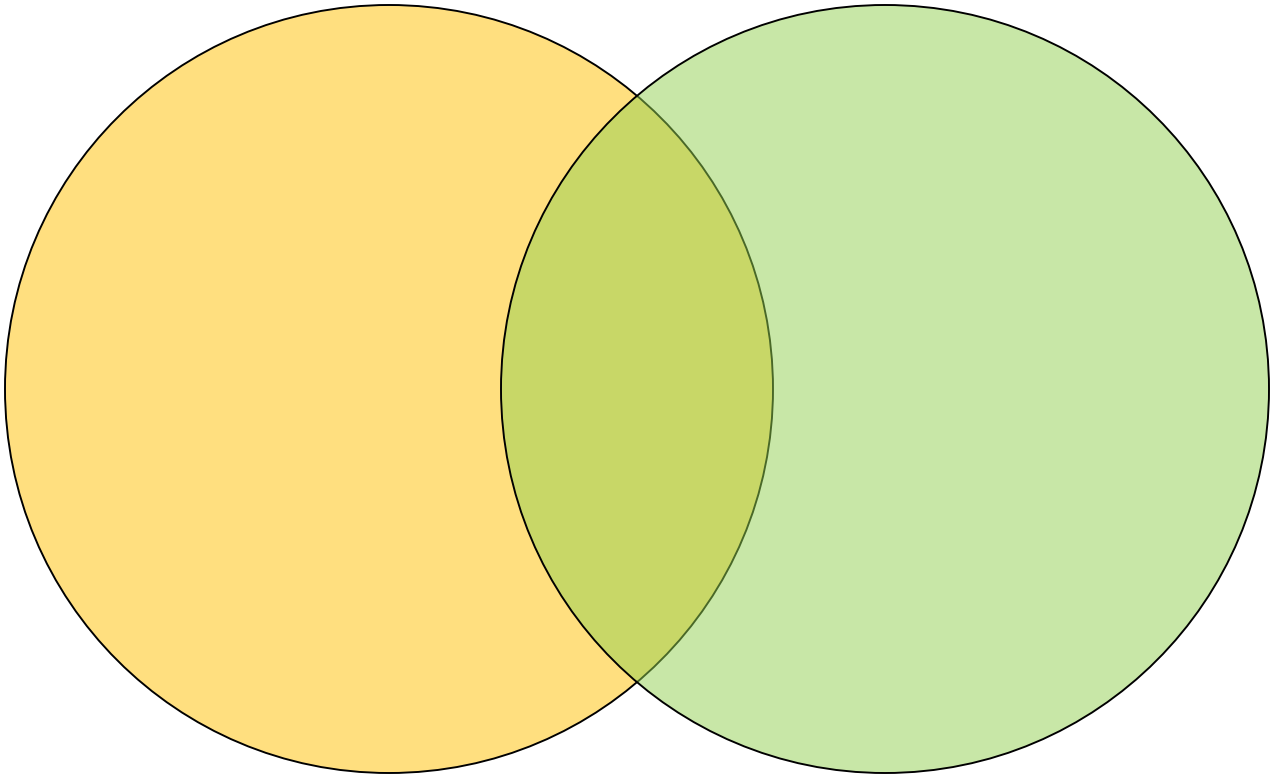


Table 1
Table 2

Inner Join

- INNER joins returns rows when there is at least one match in both the tables
- Avoid ambiguity by qualifying each column name with table name
- Join tables based on relationships as well ad-hoc
- Operators for Join
  - =
  - >
  - <
  - <=
  - >=

Inner Join

Inner Join

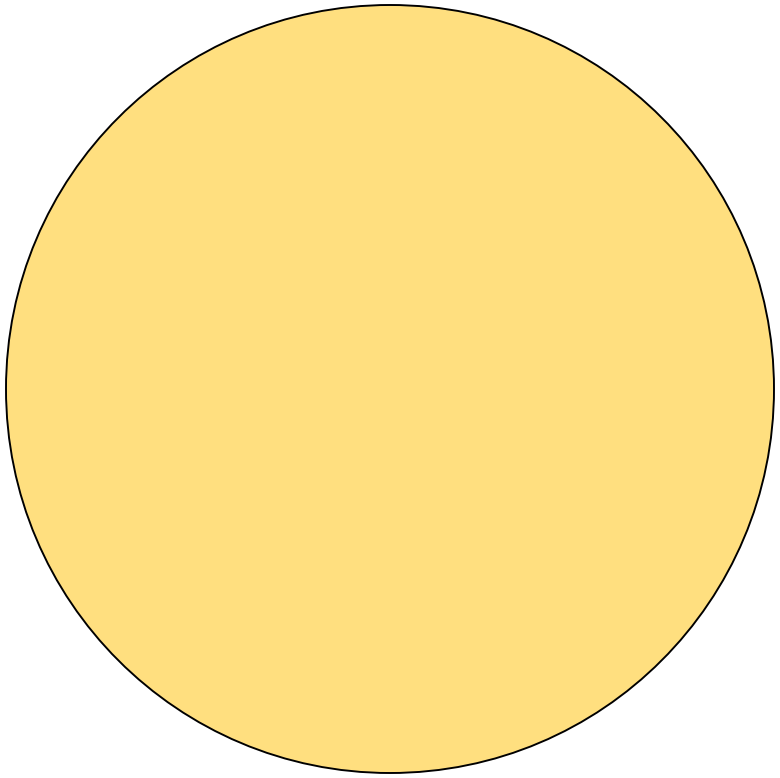


Table 1

Inner Join

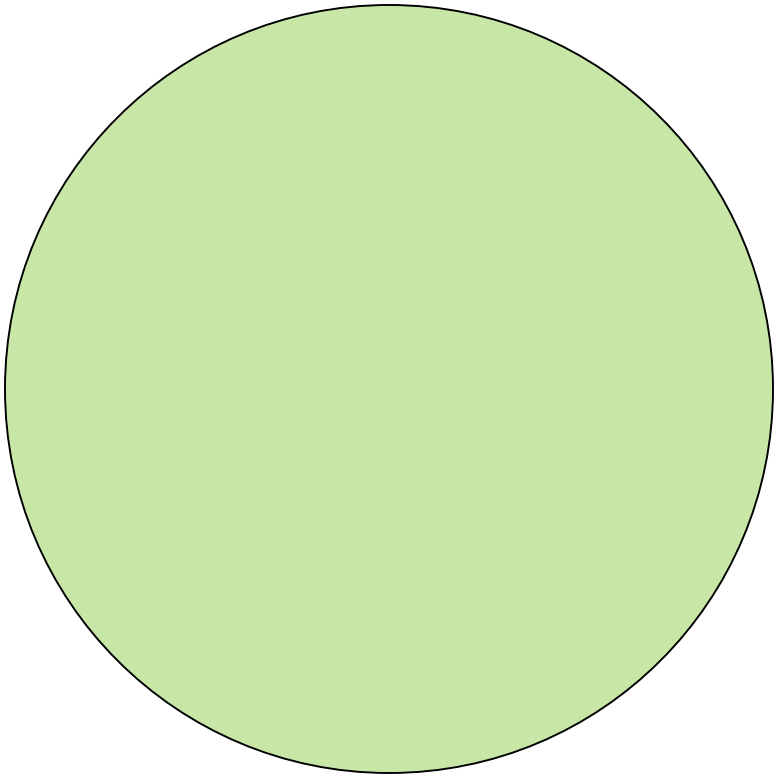


Table 2

Inner Join

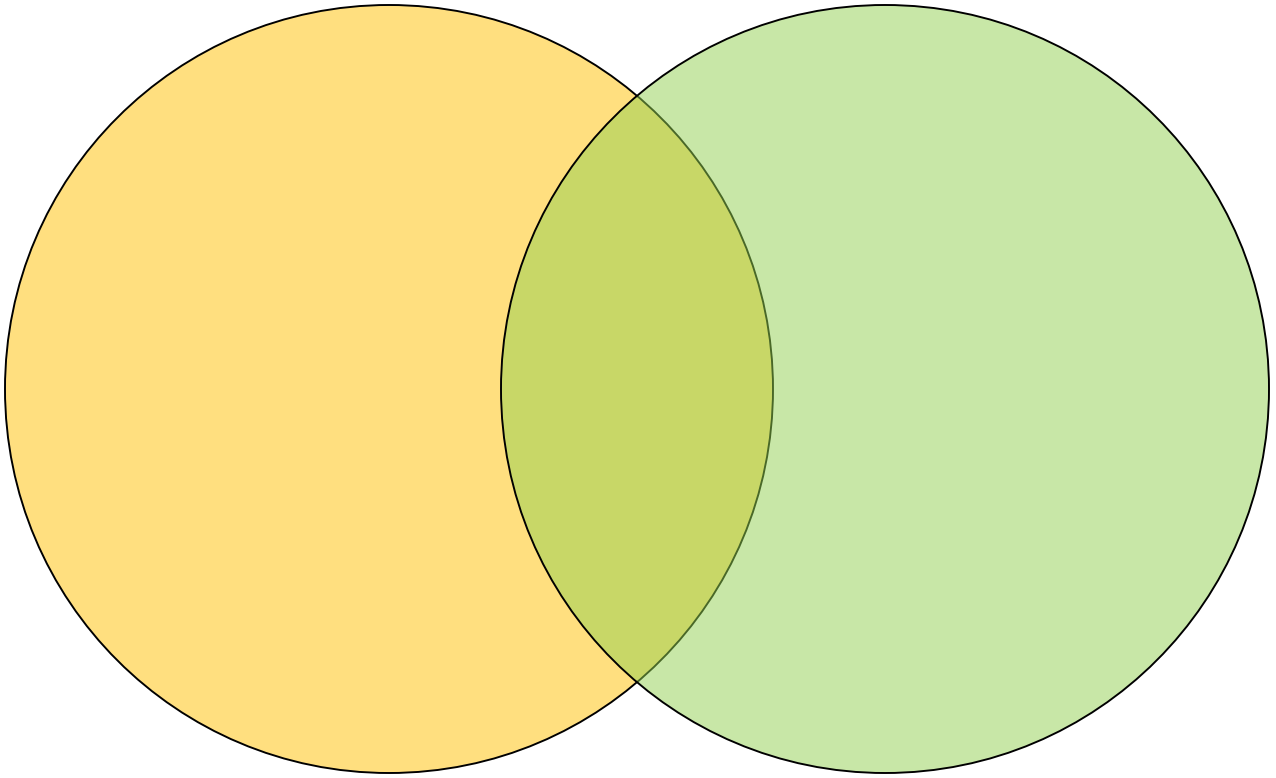


Table 1 Table 2

Inner Join

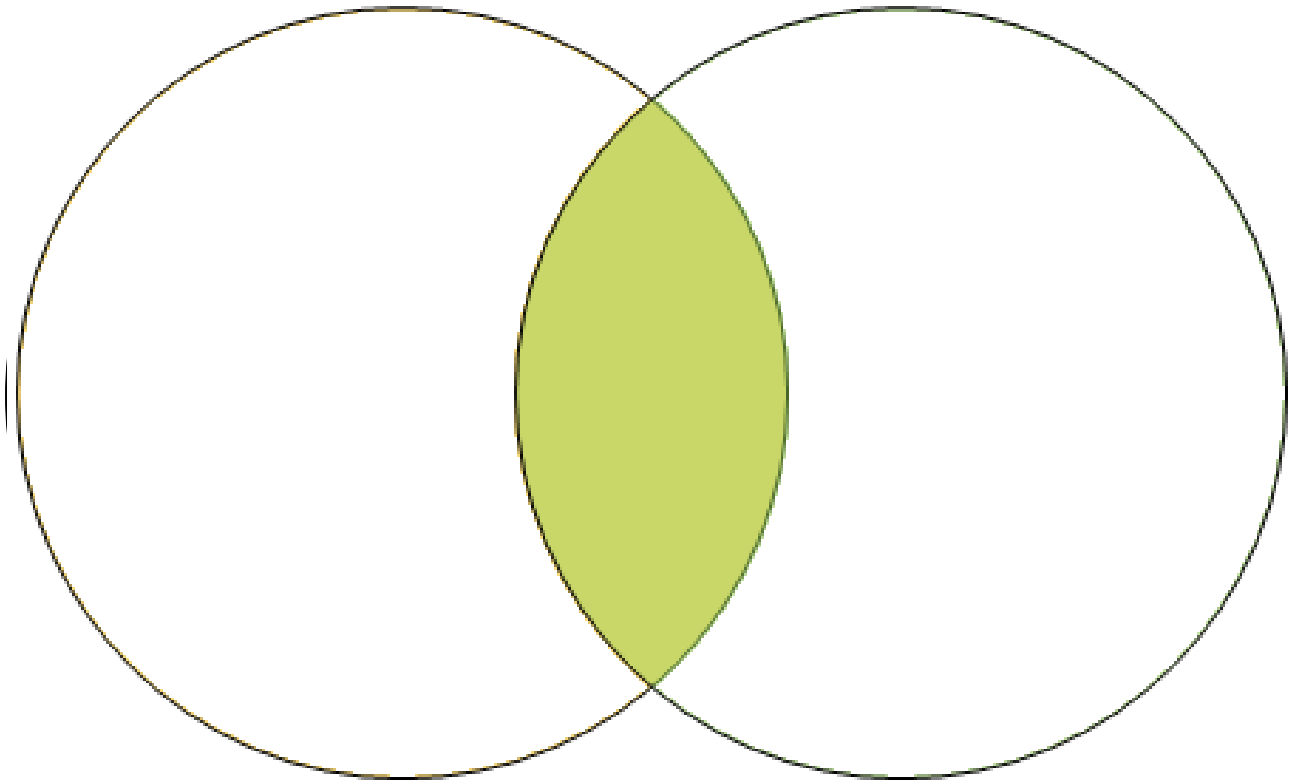


Table 1 Table 2

Inner Join

ID	Value
1	First
2	Second
3	Third
4	Fourth
5	Fifth

Table 1

ID	Value
1	First
2	Second
3	Third
6	Sixth
7	Seventh
8	Eighth

Table 2

Inner Join

ID	Value
1	First
2	Second
3	Third
4	Fourth
5	Fifth

Table 1

ID	Value
1	First
2	Second
3	Third
6	Sixth
7	Seventh
8	Eighth

Table 2

Implicit Syntax vs Explicit Syntax

- Implicit Syntax

```
SELECT t1.*, t2.*
FROM Table1 t1, Table2 t2
WHERE t1.ID = t2.ID
```

- Explicit Syntax (SQL-92)

```
SELECT t1.*, t2.*
FROM Table1 t1
INNER JOIN Table2 t2 ON t1.ID = t2.ID
```

Outer Join

- **Left Outer Join**
- **Right Outer Join**
- **Full Outer Join***

- **Outer keyword is optional**

* MySQL does not support full outer join syntax

Left Outer Join

- LEFT OUTER join returns all the rows from the left table with the matching rows from the right table
- If there are no columns matching in the right table, it returns NULL values

Left Outer Join

Left Outer Join

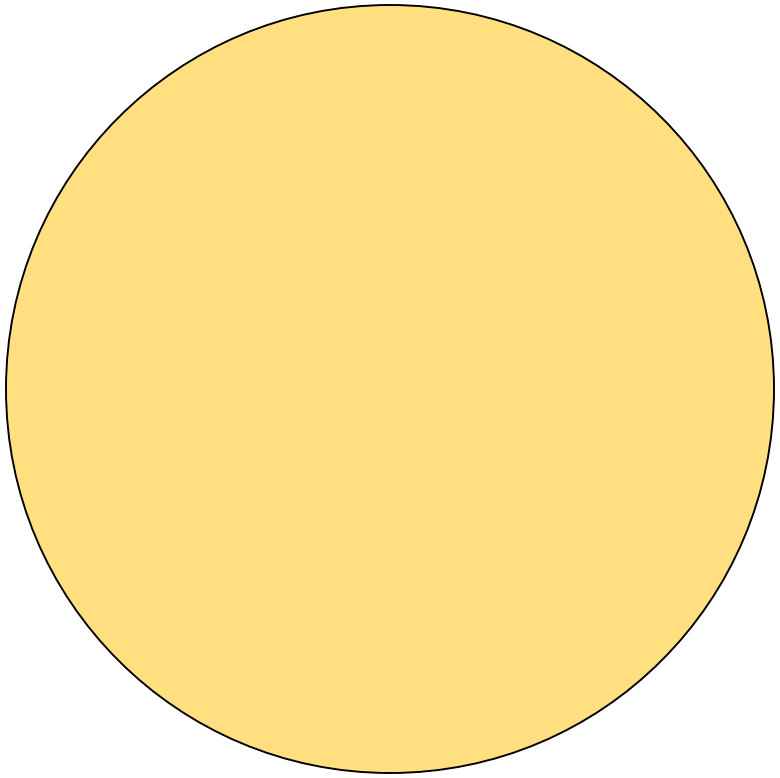


Table 1

Left Outer Join

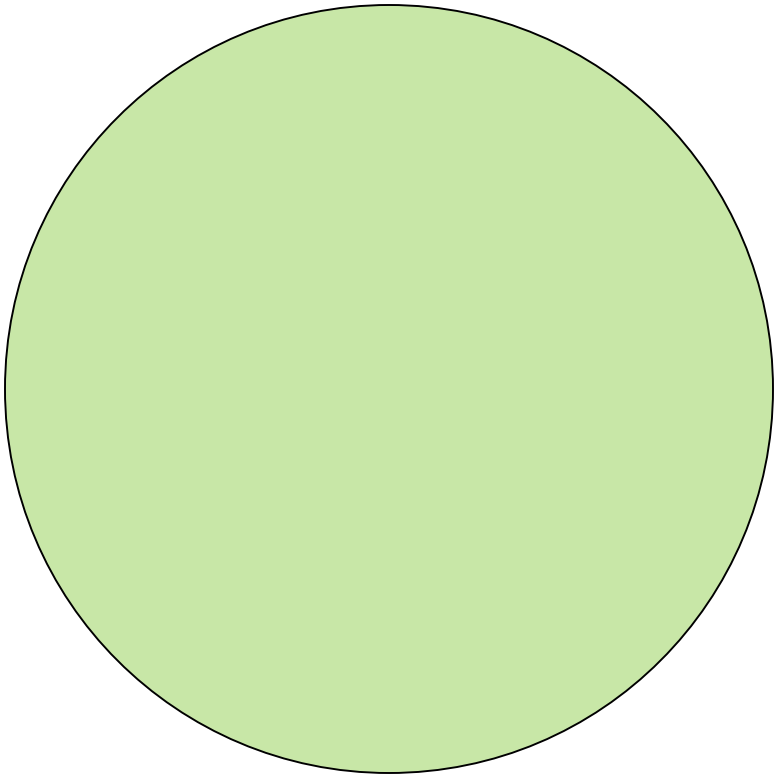


Table 2

Left Outer Join

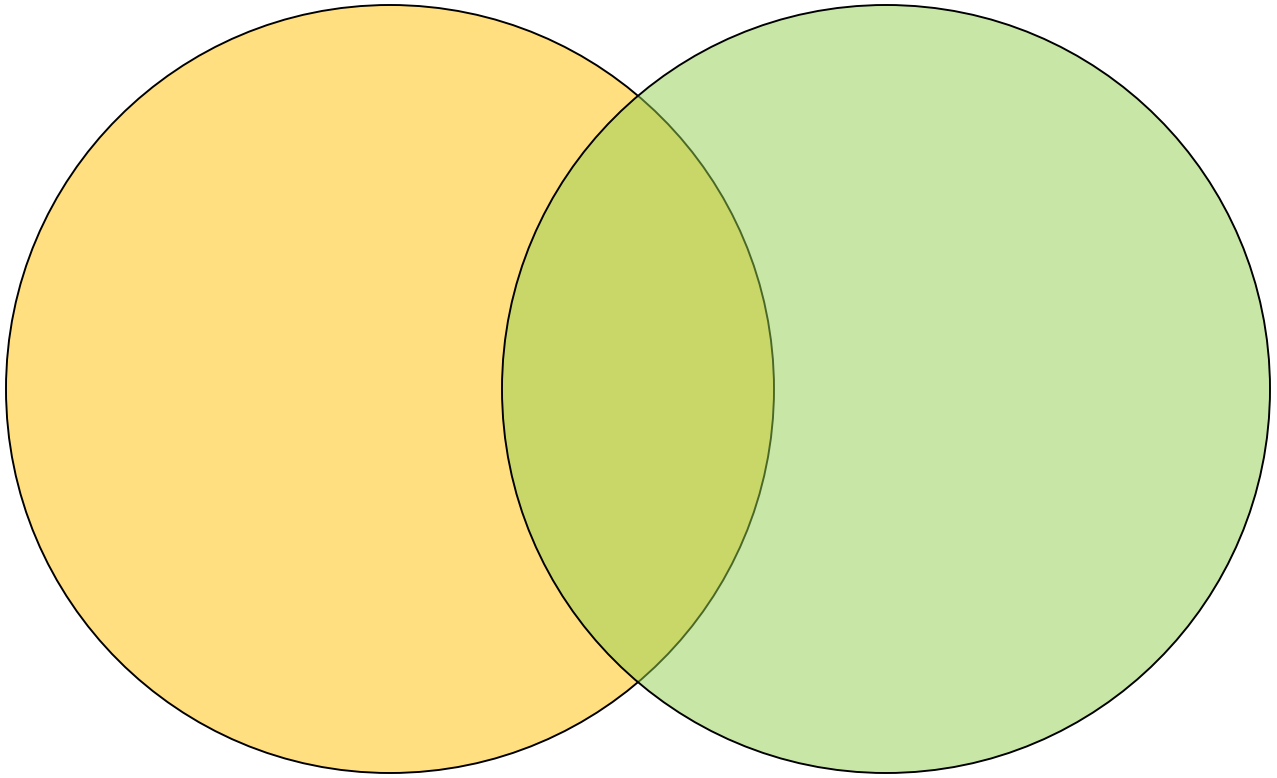
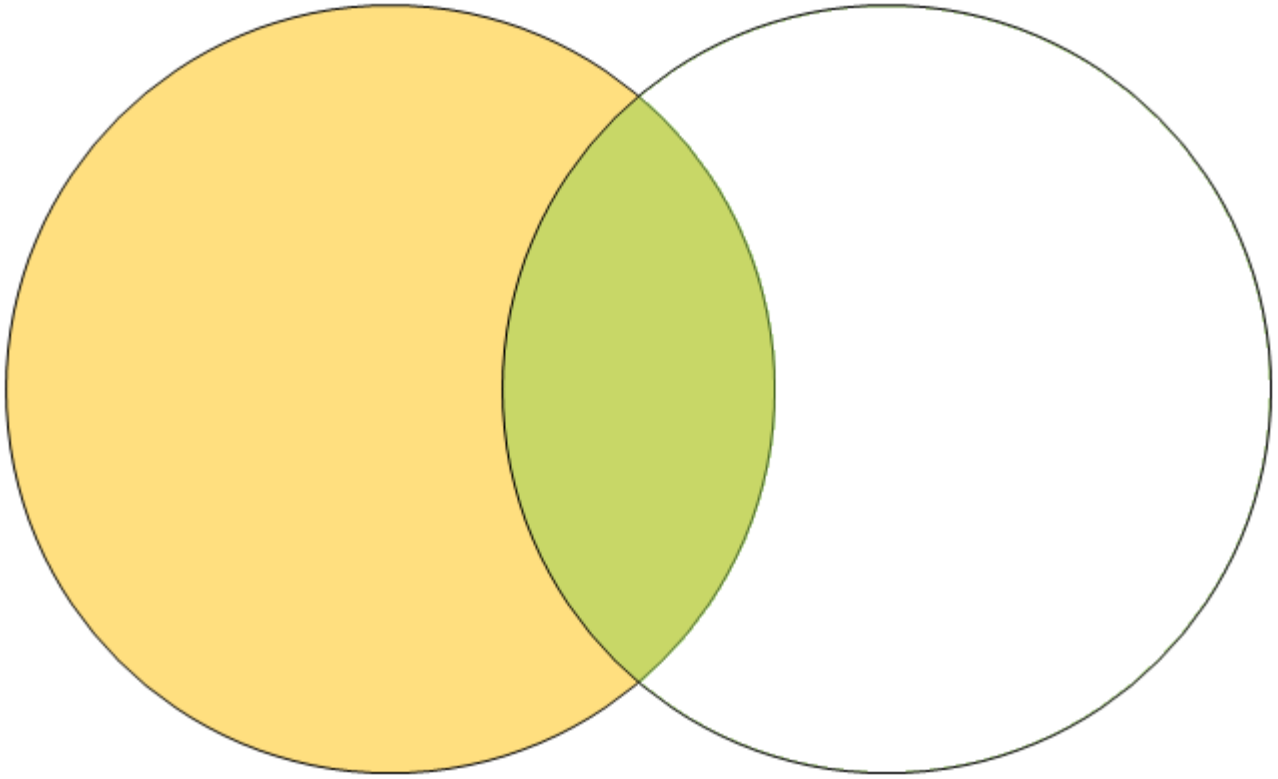


Table 1 Table 2

Left Outer Join



Left Outer Join

ID	Value
1	First
2	Second
3	Third
4	Fourth
5	Fifth

Table 1

ID	Value
1	First
2	Second
3	Third
6	Sixth
7	Seventh
8	Eighth

Table 2

Left Outer Join

ID	Value
1	First
2	Second
3	Third
4	Fourth
5	Fifth

Table 1

ID	Value
1	First
2	Second
3	Third
6	Sixth
7	Seventh
8	Eighth

Table 2

Left Outer Join

```
SELECT t1.*, t2.*
FROM Table1 t1
LEFT OUTER JOIN Table2 t2 ON t1.ID = t2.ID
```

Right Outer Join

- RIGHT OUTER join returns all the rows from the right table with the matching rows from the left table
- If there are no columns matching in the left table, it returns NULL values

Right Outer Join

Right Outer Join

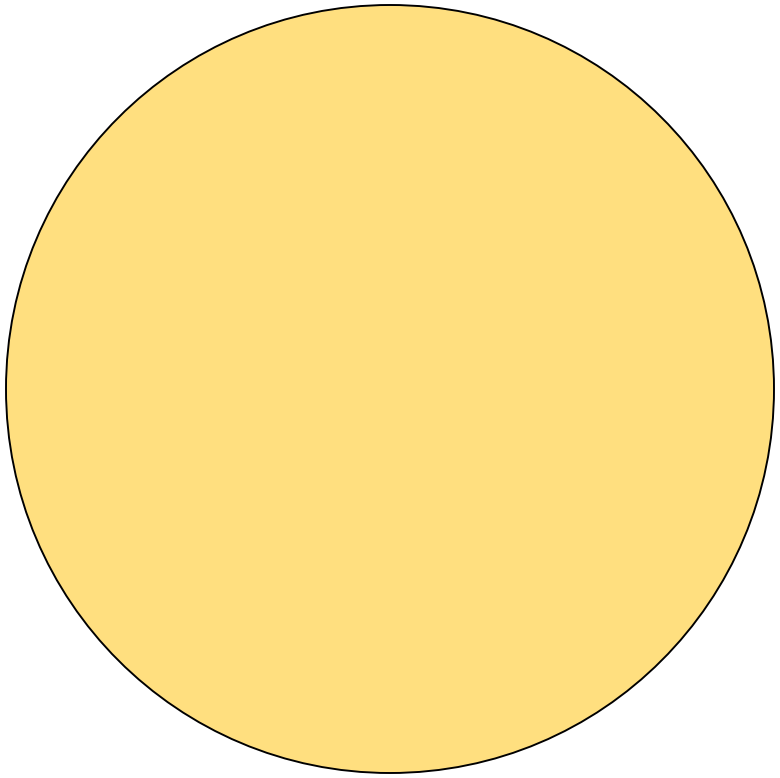


Table 1

Right Outer Join

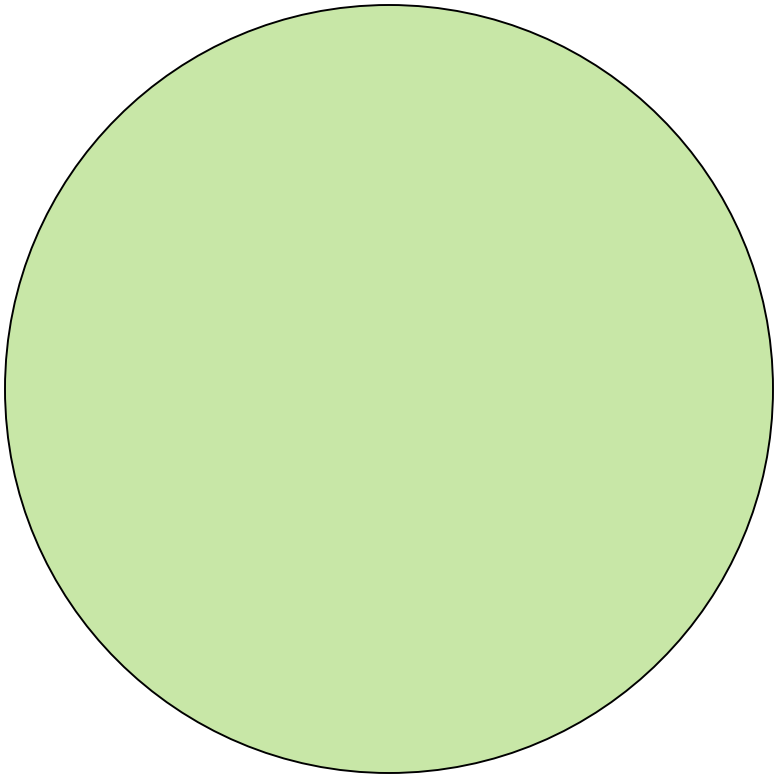


Table 2

Right Outer Join

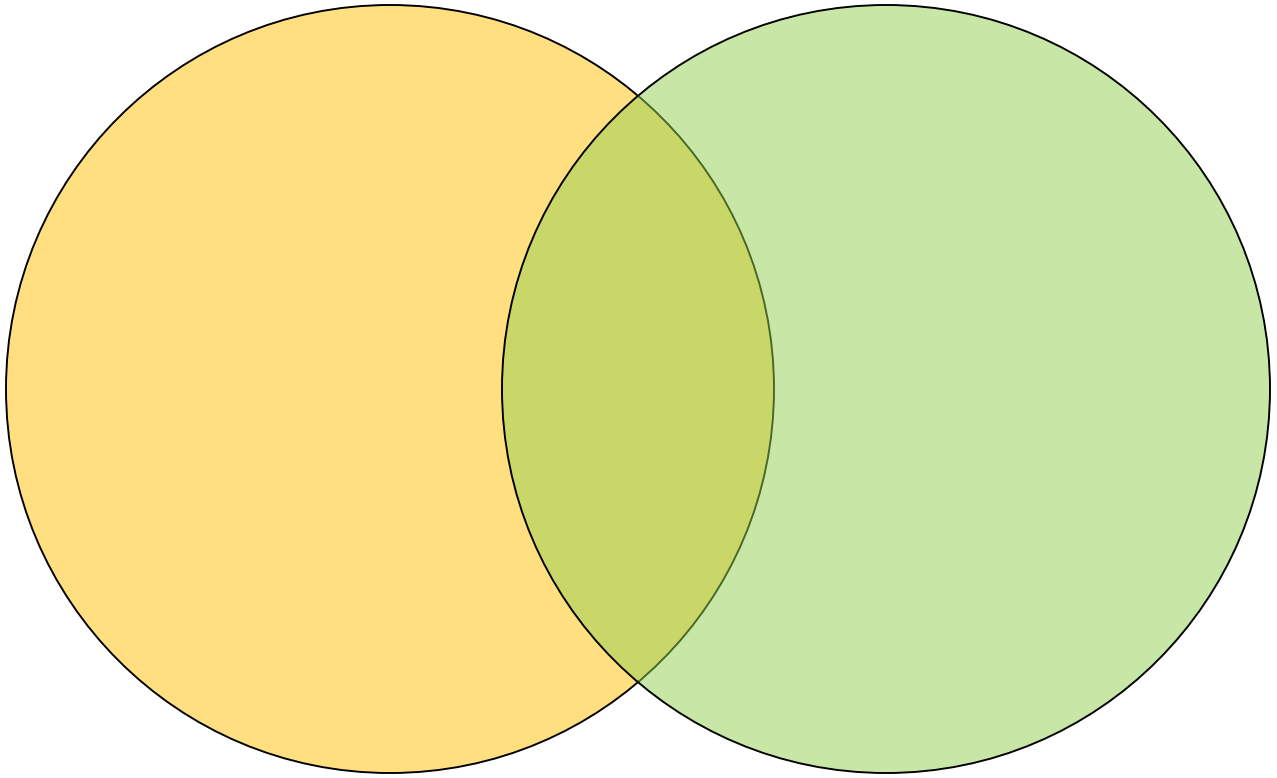


Table 1 Table 2

Right Outer Join

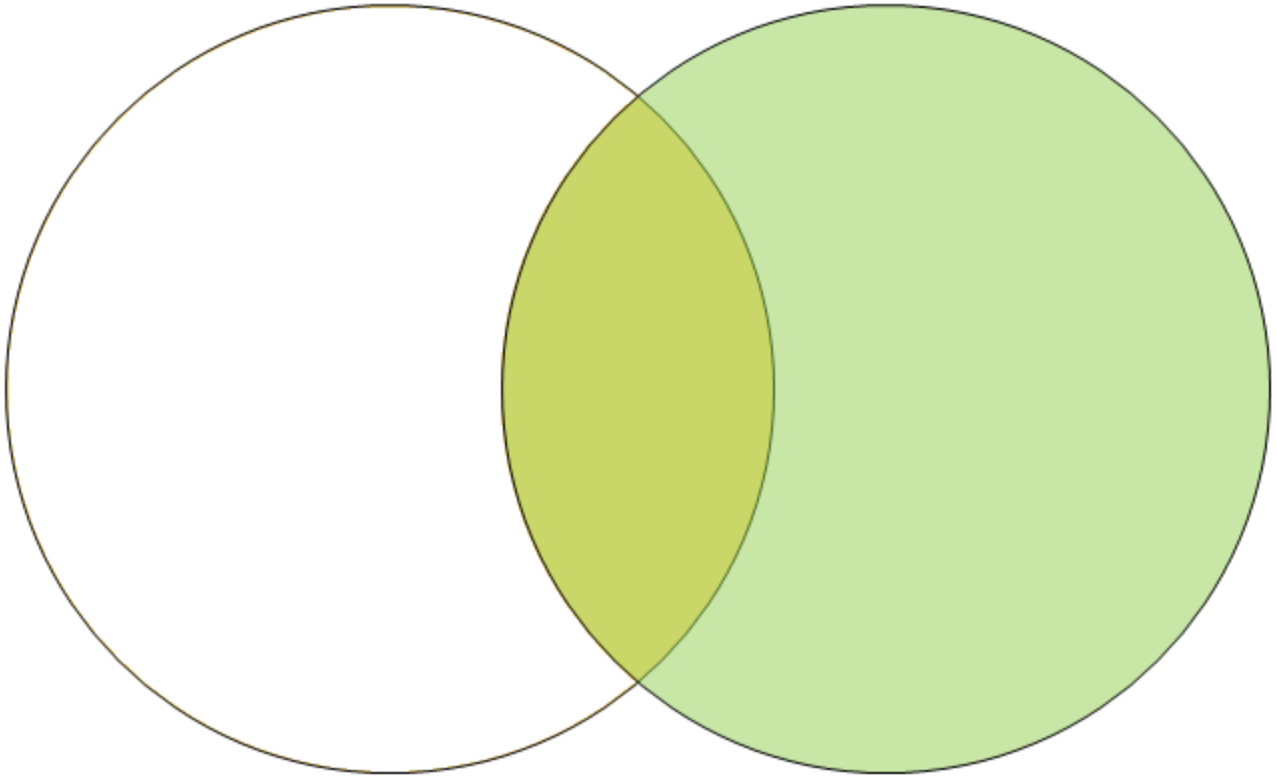


Table 1 Table 2

Right Outer Join

ID	Value
1	First
2	Second
3	Third
4	Fourth
5	Fifth

Table 1

ID	Value
1	First
2	Second
3	Third
6	Sixth
7	Seventh
8	Eighth

Table 2

Right Outer Join

ID	Value
1	First
2	Second
3	Third
4	Fourth
5	Fifth

Table 1

ID	Value
1	First
2	Second
3	Third
6	Sixth
7	Seventh
8	Eighth

Table 2

Right Outer Join

```
SELECT t1.*, t2.*
FROM Table1 t1
RIGHT OUTER JOIN Table2 t2 ON t1.ID = t2.ID
```

Full Outer Join

- **FULL OUTER join combines left outer join and right outer join**
- **This join returns rows from either table when the conditions are met and returns a null value when there is no match**
- **MySQL does not support FULL OUTER JOIN syntax**
  - Simulate FULL OUTER JOIN using LEFT and RIGHT join with UNION

Full Outer Join

Full Outer Join

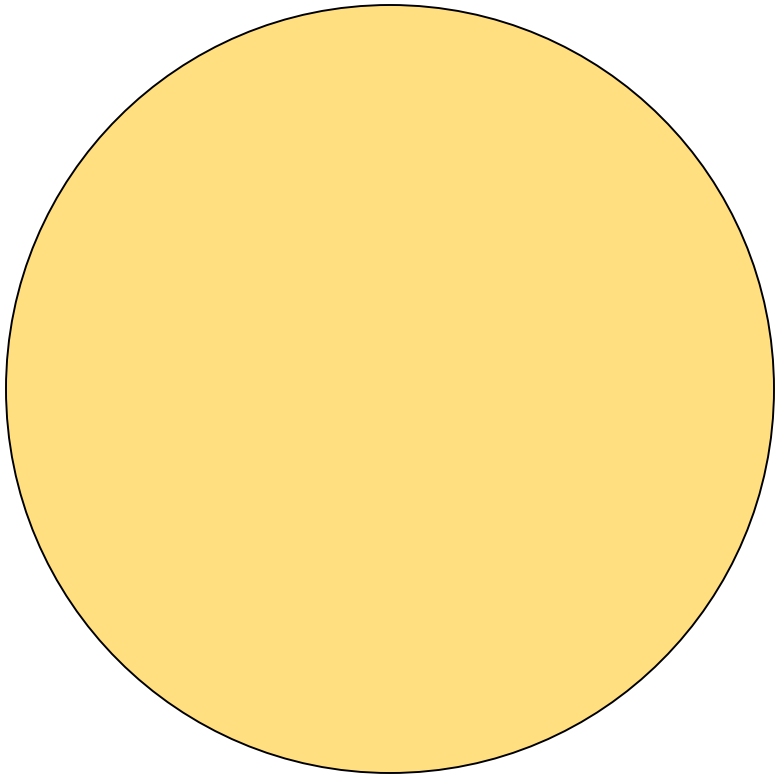


Table 1

Full Outer Join

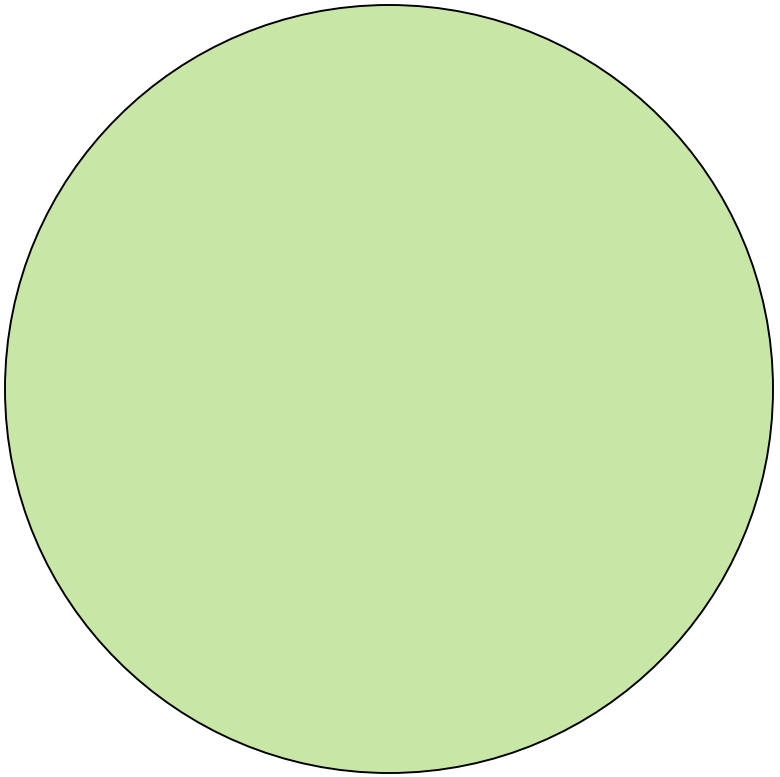


Table 2

Full Outer Join

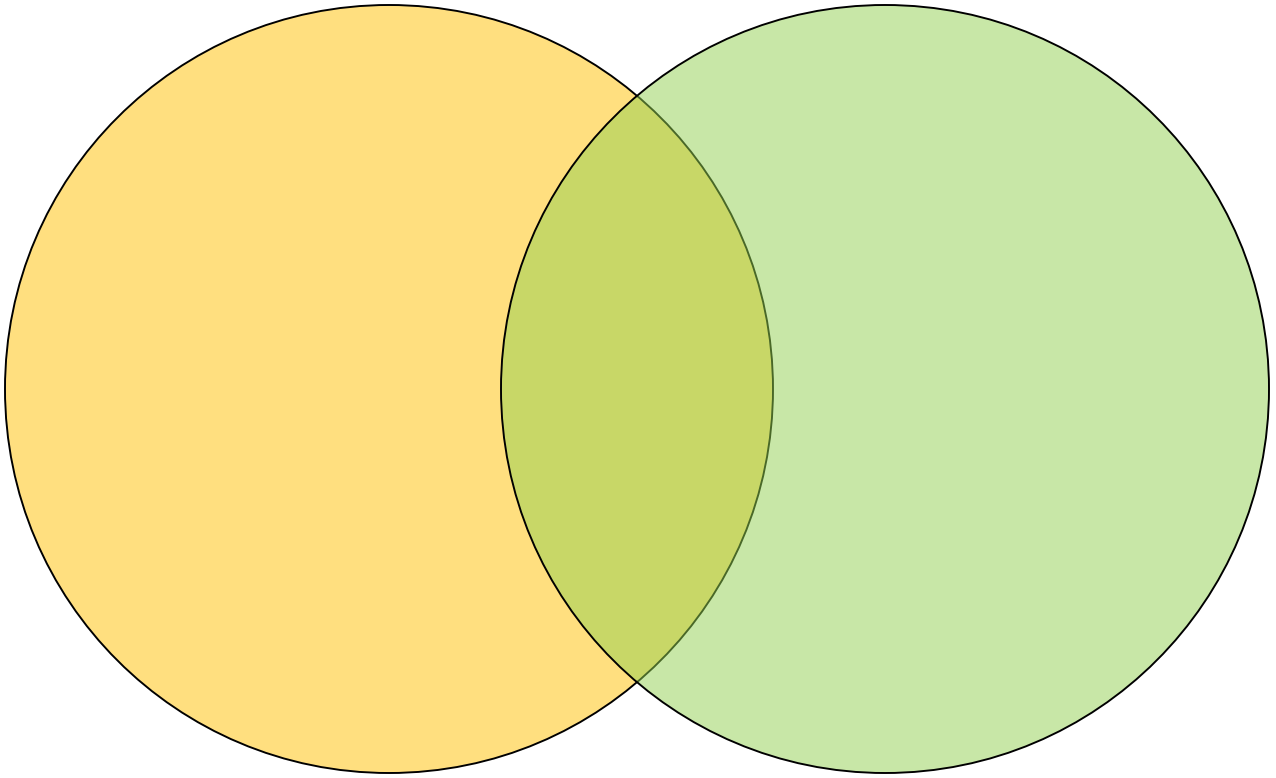
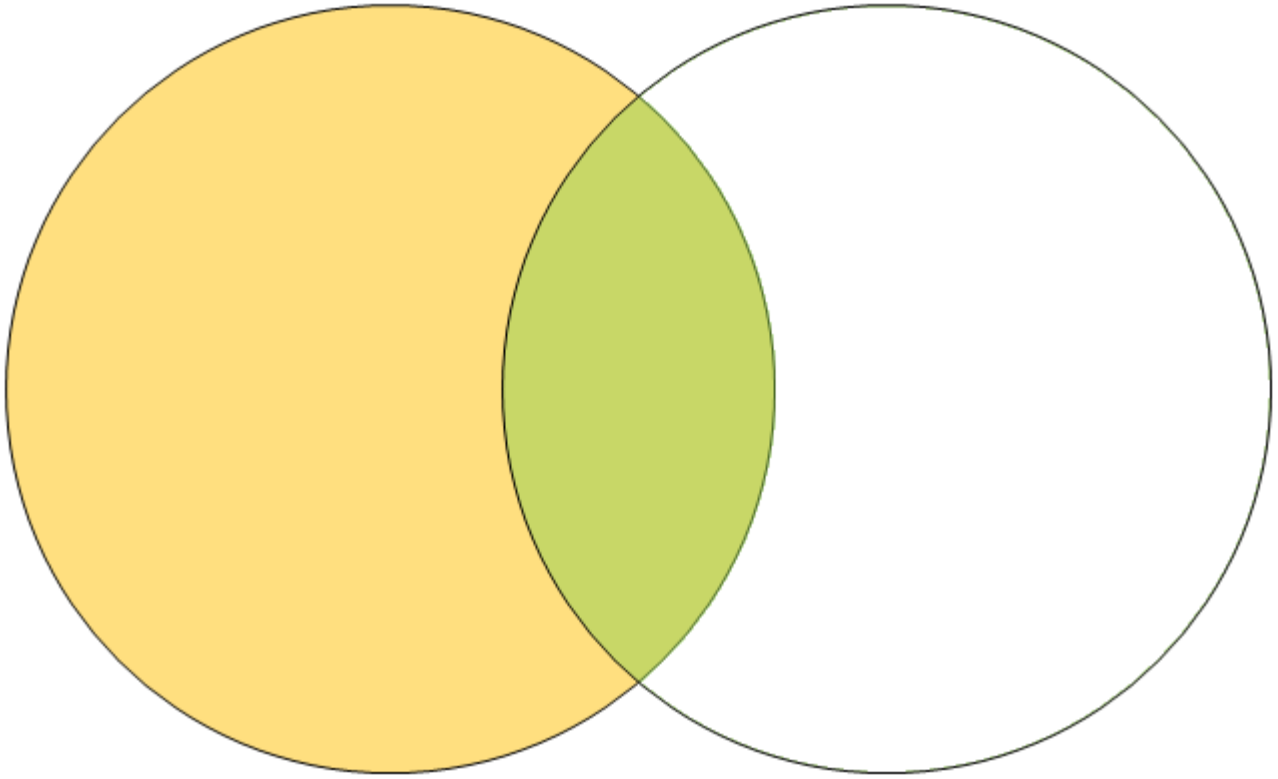


Table 1 Table 2

Left Outer Join



Right Outer Join

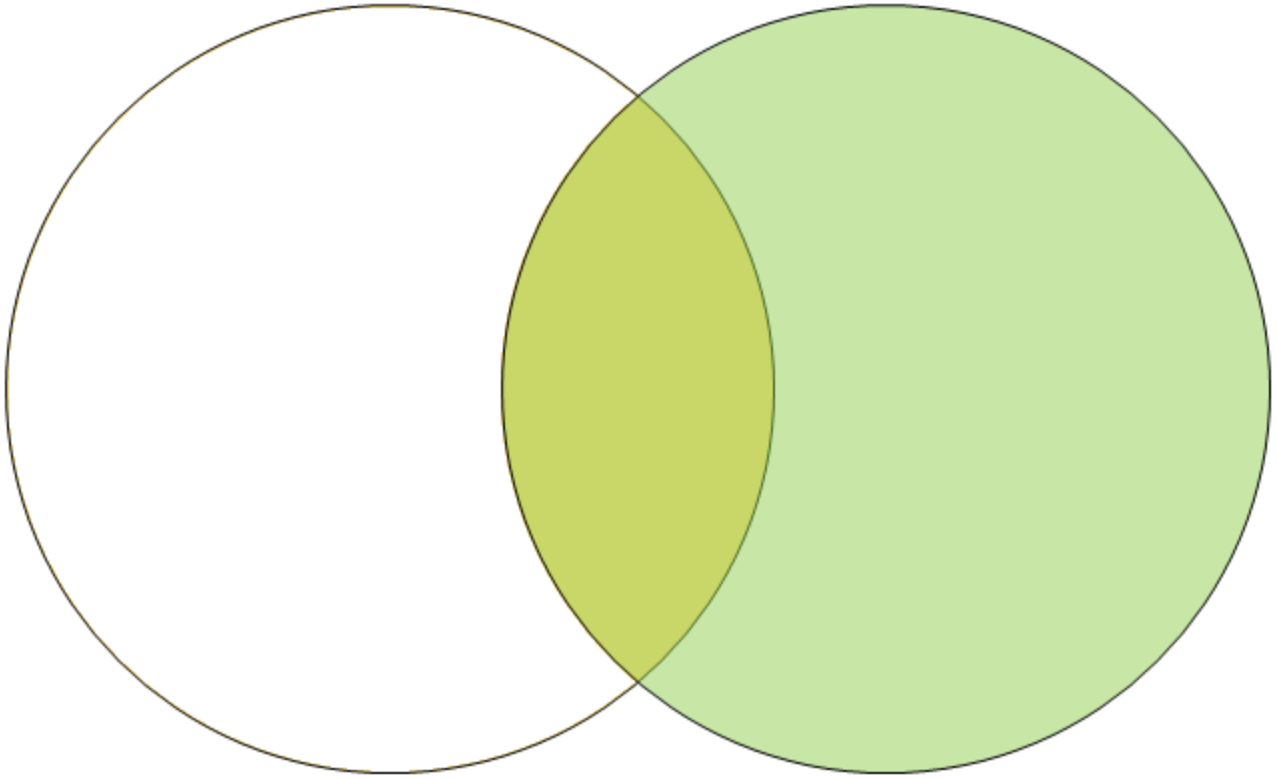


Table 1 Table 2

Full Outer Join

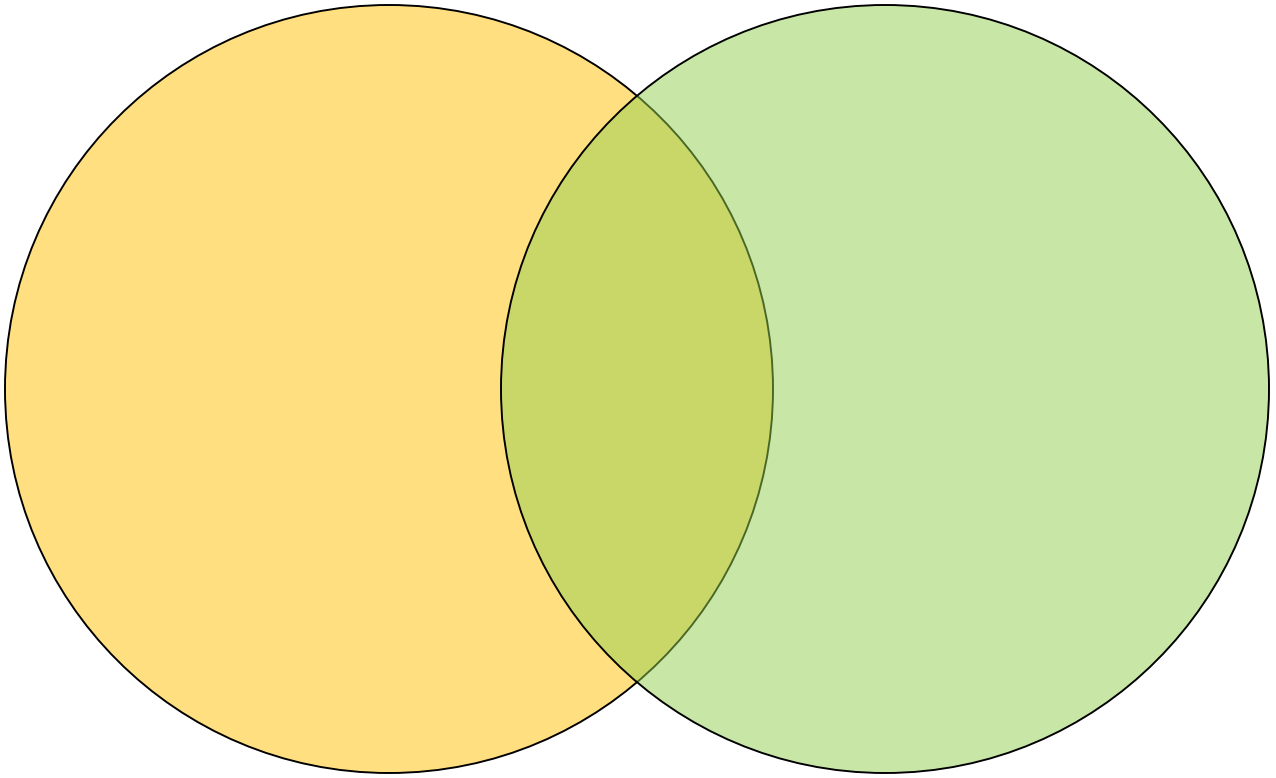


Table 1 Table 2

Cross Join

- CROSS join is a Cartesian join that does not necessitate any condition to join
- The result set contains records that are multiples of the record number of both the tables

Cross Join

ID	Value
1	First
2	Second
3	Third
4	Fourth
5	Fifth

Table 1

ID	Value
1	First
2	Second
3	Third
6	Sixth
7	Seventh
8	Eighth

Table 2

Cross Join

ID	Value
1	First
2	Second
3	Third
4	Fourth
5	Fifth

Table 1

ID	Value
1	First
2	Second
3	Third
6	Sixth
7	Seventh
8	Eighth

Table 2

Cross Join

ID	Value
1	First
2	Second
3	Third
4	Fourth
5	Fifth

Table 1

ID	Value
1	First
2	Second
3	Third
6	Sixth
7	Seventh
8	Eighth

Table 2

Cross Join

ID	Value
1	First
2	Second
3	Third
4	Fourth
5	Fifth

Table 1

ID	Value
1	First
2	Second
3	Third
6	Sixth
7	Seventh
8	Eighth

Table 2

Cross Join

ID	Value
1	First
2	Second
3	Third
4	Fourth
5	Fifth

Table 1

ID	Value
1	First
2	Second
3	Third
6	Sixth
7	Seventh
8	Eighth

Table 2

Cross Join

ID	Value
1	First
2	Second
3	Third
4	Fourth
5	Fifth

Table 1

ID	Value
1	First
2	Second
3	Third
6	Sixth
7	Seventh
8	Eighth

Table 2

Cross Join

ID	Value
1	First
2	Second
3	Third
4	Fourth
5	Fifth

Table 1

ID	Value
1	First
2	Second
3	Third
6	Sixth
7	Seventh
8	Eighth

Table 2

Cross Join

ID	Value
1	First
2	Second
3	Third
4	Fourth
5	Fifth

Table 1

ID	Value
1	First
2	Second
3	Third
6	Sixth
7	Seventh
8	Eighth

Table 2

Cross Join

ID	Value
1	First
2	Second
3	Third
4	Fourth
5	Fifth

Table 1

ID	Value
1	First
2	Second
3	Third
6	Sixth
7	Seventh
8	Eighth

Table 2

Cross Join

ID	Value
1	First
2	Second
3	Third
4	Fourth
5	Fifth

Table 1

ID	Value
1	First
2	Second
3	Third
6	Sixth
7	Seventh
8	Eighth

Table 2

Cross Join

ID	Value
1	First
2	Second
3	Third
4	Fourth
5	Fifth

Table 1

ID	Value
1	First
2	Second
3	Third
6	Sixth
7	Seventh
8	Eighth

Table 2

Cross Join

ID	Value
1	First
2	Second
3	Third
4	Fourth
5	Fifth

Table 1

ID	Value
1	First
2	Second
3	Third
6	Sixth
7	Seventh
8	Eighth

Table 2

Cross Join

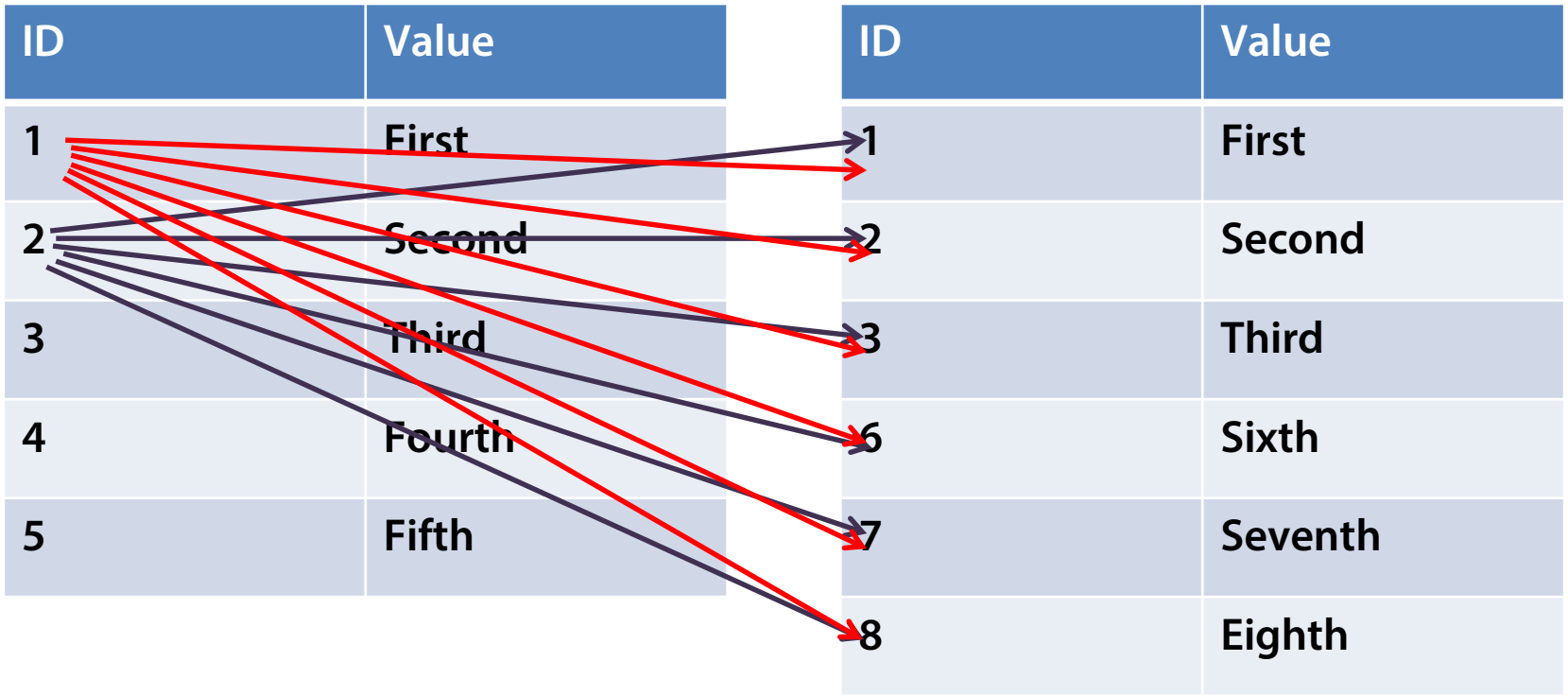
ID	Value
1	First
2	Second
3	Third
4	Fourth
5	Fifth

Table 1

ID	Value
1	First
2	Second
3	Third
6	Sixth
7	Seventh
8	Eighth

Table 2

Cross Join



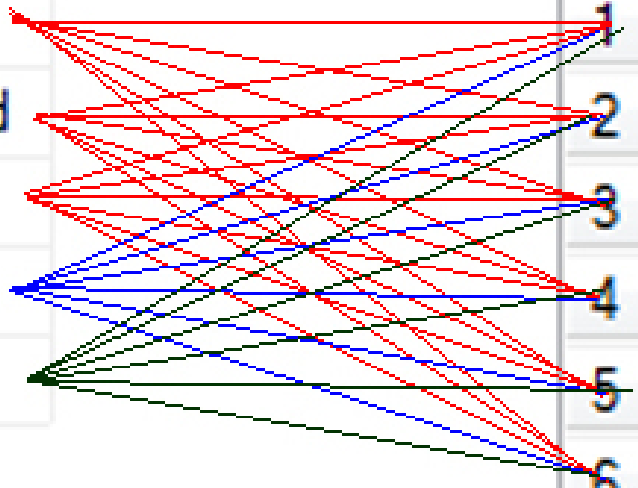
ID	Value
1	First
2	Second
3	Third
4	Fourth
5	Fifth

Table 1

ID	Value
1	First
2	Second
3	Third
6	Sixth
7	Seventh
8	Eighth

Table 2

	ID	Value
1	1	First
2	2	Second
3	3	Third
4	4	Fourth
5	5	Fifth



	ID	Value
1	1	First
2	2	Second
3	3	Third
4	6	Sixth
5	7	Seventh
6	8	Eighth

	ID	Value	ID	Value
1	1	First	1	First
2	1	First	2	Second
3	1	First	3	Third
4	1	First	6	Sixth
5	1	First	7	Seventh
6	1	First	8	Eighth
7	2	Second	1	First
8	2	Second	2	Second
9	2	Second	3	Third
10	2	Second	6	Sixth
11	2	Second	7	Seventh
12	2	Second	8	Eighth

Pop Quiz

- **Scenario:**
  - We have three tables – 1) Students 2) Classes and 3) StudentClass
  - The student can sign up maximum of three classes
  - In summer student can opt out and can sign up for no classes

- **Question 1: What will be the best possible join if we want to retrieve all the students who have signed up for the classes in the summer?**

- **Question 2: What will be the best possible join if we want to retrieve all the students who have signed up for no classes in summer?**

Students

StudentId	StudentName
1	John
2	Matt
3	James

Classes

ClassId	ClassName
1	Maths
2	Arts
3	History

StudentClass

ClassId	StudentId
1	1
1	2
3	1
3	2
3	3

Equi Join

- An EQUI join is a specific type of comparator-based join, that uses only equality comparisons in the join-predicate

```
SELECT t1.*, t2.*
FROM Table1 t1
INNER JOIN Table2 t2 ON t1.ID = t2.ID
```

Non-equi Join

- A NON-EQUI join is a specific type of comparator-based join, that does not use equality comparisons in the join-predicate

```
SELECT t1.*, t2.*
FROM Table1 t1
INNER JOIN Table2 t2 ON t1.ID > t2.ID
```

Self Join

- A SELF join is a join in which a table is joined with itself
- The user must alias tables used in self join
- The user must qualify each column name used in SELECT clauses with a table alias to avoid ambiguity

```
SELECT t1.*, t2.*
FROM Table1 t1
INNER JOIN Table1 t2 ON t1.ID > t2.ID
```

Natural Join

- A NATURAL join is kind of join which joins two (or more) tables based on all the columns in the two tables with the same name
- It can be either INNER or OUTER join

```
SELECT t1.*, t2.*
FROM Table1 t1
NATURAL JOIN Table2 t2
```

Joins with USING Keyword

- USING keyword simplifies syntax for joining tables when the columns have the same name in both the tables
- USING keyword can be used with INNER or OUTER joins
- You can use more than one column with USING keywords

```
SELECT t1.*, t2.*
FROM Table1 t1
INNER JOIN Table2 t2 USING (ID, VALUE)
```

UNION Operators

- UNION Combines two or more SELECT statements into a single result set
- Each SELECT statement of UNION operator must have the same number of Columns
- UNION removes duplicate rows
- UNION ALL does not remove duplicate rows
- Only one ORDER BY clause sorting entire result set
- Simulate FULL OUTER JOIN using LEFT and RIGHT join with UNION

Subqueries

- A subquery is a nested query where the results of one query can be used in another query via a relational operator or aggregation function
- A subquery must be enclosed with parentheses
- A subquery can have only one column in the SELECT clause if used in WHERE clause
- An ORDER BY clause is not allowed in a subquery
- Subqueries can be nested within other subqueries
- Subqueries are used in WHERE, HAVING, FROM and SELECT clause

```
SELECT t1.*
FROM Table1 t1
WHERE t1.ID NOT IN (SELECT t2.ID FROM Table2 t2)
```

Joins vs Subqueries

- **Joins**
  - Can include columns from joining tables in the SELECT clause
  - Easy to read and more intuitive

- **Subqueries**
  - Can pass the aggregate values to the main query
  - Simplifies long and complex queries

Correlated Subqueries

- A correlated subquery is a subquery that is executed once for each row
- A correlated subquery returns results based on the column of the main query

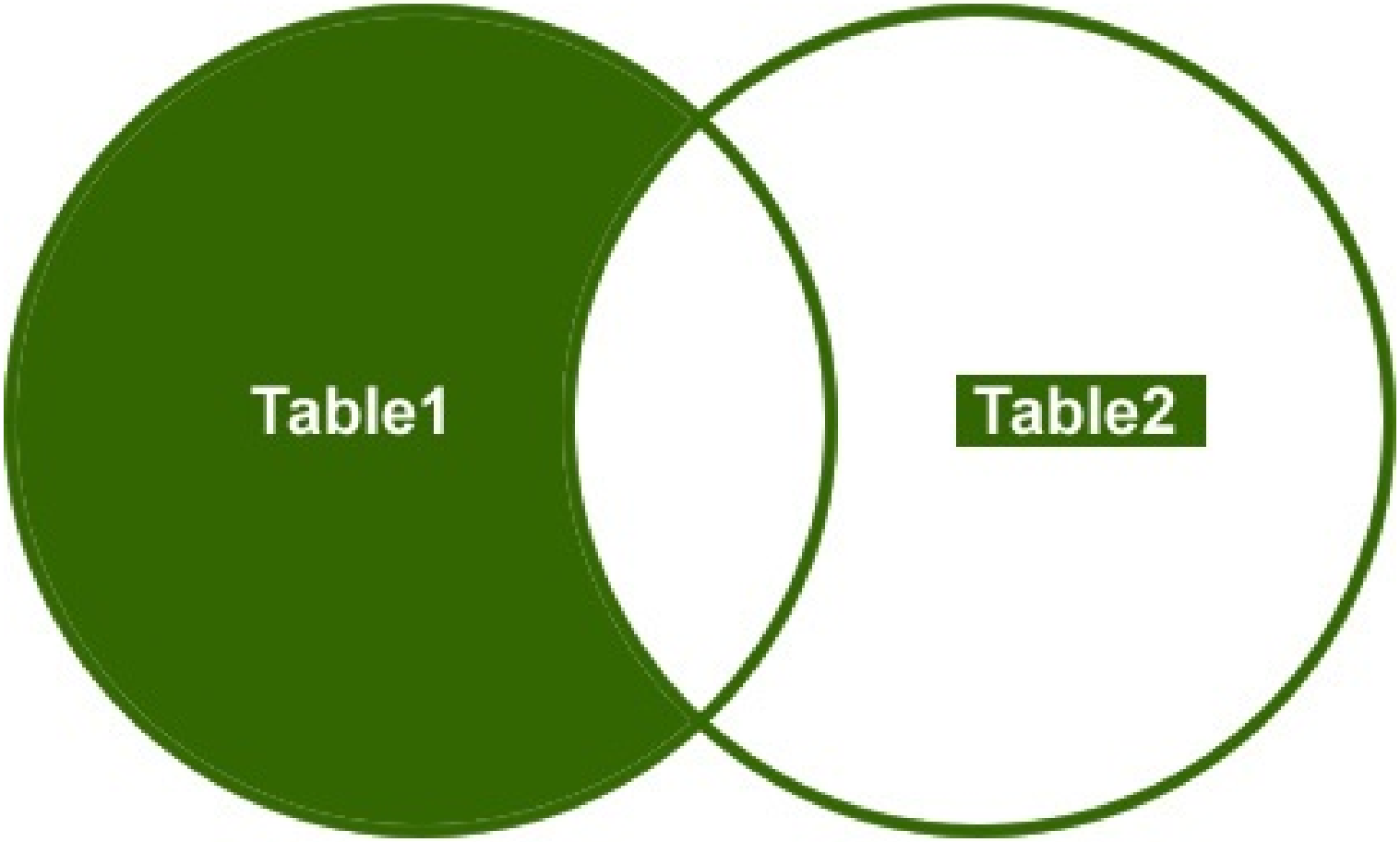
```
SELECT t1.*
FROM Table1 t1
WHERE t1.ID IN
(SELECT t2.ID
FROM Table2 t2
WHERE t2.value = t1.value)
```

Outer Join with NULL

Problem: Rewrite following SQL subquery using SQL Joins

```
SELECT t1.*
FROM Table1 t1
WHERE t1.ID NOT IN (SELECT t2.ID FROM Table2 t2)
```

LEFT OUTER JOIN - WHERE NULL




```
SELECT *
FROM Table1 t1
LEFT OUTER JOIN Table2 t2
        ON t1.Col1 = t2.Col1
WHERE t2.Col1 IS NULL
```

Summary in Sixty Seconds

#1 An SQL JOIN combines columns from two or more tables in a single result set

#2 Always alias your column with table to avoid ambiguity in the code

#3 UNION keyword removes duplicate from resultset but UNION ALL keyword does not

#4 A correlated subquery is executed once for each row in the main query