STRUCTURED QUERY LANGUAGE III

Lecturer: Jailani Abdul Rahman

This lecture

- In this lecture, we will further enhance our MySQL Select Statement.

Group By

- **MySQL Group By**, is used with the **SELECT** statement to group rows into subgroups by one or more columns or expressions.

- It is added after **FROM** or **WHERE** clause.

- To use **GROUP BY**:

SELECT column1, column2, … **FROM** table **WHERE** conditions **GROUP BY** column1, column2, …;	**SELECT** column1, column2, … **FROM** table **GROUP BY** column1, column2, …;

Example: Group By

- StudentGroup Table

studentID	groupID
101	CWEB4
102	CWEB4
103	CIST5
104	CWEB4
105	CIST6
106	CIST6
107	CIST5
108	CWEB4
109	CIST5
110	CIST5

```
SELECT groupID
FROM StudentGroup
GROUP BY groupID;
```

groupID
CWEB4
CIST5
CIST6

```
SELECT groupID, count(*)
FROM StudentGroup
GROUP BY groupID;
```

groupID	count(*)
CWEB4	4
CIST5	4
CIST6	2

Example: Group By (cont.)

- It is possible to organise the group.
- StudentGroup Table

studentID	groupID
101	CWEB4
102	CWEB4
103	CIST5
104	CWEB4
105	CIST6
106	CIST6
107	CIST5
108	CWEB4
109	CIST5
110	CIST5

```
SELECT groupID, count(*)
FROM StudentGroup
GROUP BY groupID ASC;
```

groupID	count(*)
CIST5	4
CIST6	2
CWEB4	4

```
SELECT groupID, count(*)
FROM StudentGroup
GROUP BY groupID DESC;
```

groupID	count(*)
CWEB4	4
CIST6	2
CIST5	4

Example: Group By

- It is possible as well to group multiple columns.

- GroupModule Table

groupID	moduleID	year
CWEB4	DBMS	2015
CWEB4	INMD	2015
CIST5	DBMS	2015
CIST6	DBMS	2015
CIST5	OOP	2015
CIST6	OPP	2015

```
SELECT year, count(*)
FROM GroupModule
GROUP BY year;
```

year	count(*)
2015	6

```
SELECT year, moduleID, count(*)
FROM GroupModule
GROUP BY year, moduleID;
```

year	moduleID	count(*)
2015	DBMS	3
2015	INMD	1
2015	OOP	2

Having

- **MySQL Having**, is used in the **SELECT** statement to specify filter conditions for group of rows.

- Often used with **GROUP BY** which it will filter the result the same way as WHERE clause.

- To use **HAVING**:

SELECT column1, column2, … **FROM** table **WHERE** conditions **GROUP BY** column1, column2, … **HAVING** conditions;	**SELECT** column1, column2, … **FROM** table **GROUP BY** column1, column2, … **HAVING** conditions;

Example: Having

- StudentGroup Table

studentID	groupID
101	CWEB4
102	CWEB4
103	CIST5
104	CWEB4
105	CIST6
106	CIST6
107	CIST5
108	CWEB4
109	CIST5
110	CIST5

```
SELECT groupID, count(*)
FROM StudentGroup
GROUP BY groupID
HAVING count(*) >= 4;
```

groupID	count(*)
CWEB4	4
CIST5	4

- **** For HAVING clause, it is possible to use the ALIAS of the column.**

```
SELECT groupID, count(*) AS TotalStudent
FROM StudentGroup
GROUP BY groupID
HAVING TotalStudent >= 4;
```

groupID	TotalStudent
CWEB4	4
CIST5	4

IN Operator

- MySQL IN operator allows you to specify multiple values in a WHERE clause.

- To use **IN Operator**:

```
SELECT column1, column2, …
FROM table
WHERE somecolumn IN (value1, value2, …);
```

- You can add **NOT operator** with IN operator.

Example: IN Operator

- StudentDetails Table

studentID	name
101	Steven
102	Frank
103	Didier
104	Peter

```
SELECT studentID, name
FROM StudentDetails
WHERE name IN ('Steven','Didier','Peter');
```

studentID	name
101	Steven
103	Didier
104	Peter

- It is the same as:

```
SELECT studentID, name FROM StudentDetails
WHERE name='Steven' OR name='Didier' OR name='Peter';
```

- The problem with this is you have to use **OR operator** if it is more than one value to be searched.

Example: IN Operator (cont.)

- StudentDetails

studentID	name
101	Steven
102	Frank
103	Didier
104	Peter

- Adding **NOT operator** with IN operator.

```
SELECT studentID, name
FROM StudentDetails
WHERE name NOT IN ('Steven','Didier','Peter');
```

studentID	name
102	Frank

Subquery

- **MySQL Subquery**, is a query that is **nested** inside another query.
- There are Three ways of subquery:

1. Subquery within a WHERE clause
   a) Using with comparison operators
   b) Using with IN and NOT IN operators
   c) Using with EXISTS and NOT EXISTS operators
2. Subquery within FROM clause
3. Correlated Subquery

Subquery within a WHERE clause

- **(a) Using with comparison operators**

```
SELECT column1, column2, …
FROM table
WHERE some_column[comparator](subquery);
```

- If you use comparison operators, the result of the subquery has to return one column and row only.

Subquery within a WHERE clause (cont.)

- **(b) Using with IN and NOT IN operators**

```
SELECT column1, column2, …
FROM table
WHERE some_column IN (subquery);
```

```
SELECT column1, column2, …
FROM table
WHERE some_column NOT IN (subquery);
```

Subquery within a WHERE clause (cont.)

- **(c) Using with EXISTS and NOT EXISTS operators**

```
SELECT column1, column2, …
FROM table
WHERE EXISTS (subquery);
```

```
SELECT column1, column2, …
FROM table
WHERE NOT EXISTS (subquery);
```

Example: Subquery within a WHERE clause

- **NOTE: Example is movierating database from Lecture 6.**
- **(a) Using with comparison operators – Example**

```
SELECT *
FROM Rating
WHERE mID=
(SELECT mID
FROM Movie
WHERE title="E.T.");
```

```
mysql> select mID from Movie where title='E.T.';
+------+
| mID  |
+------+
|  104 |
+------+
1 row in set (0.00 sec)
```

```
mysql> select * from Rating where mID=
    -> (select mID from Movie where title='E.T.');
+------+------+-------+------------+
| rID  | mID  | stars | ratingDate |
+------+------+-------+------------+
|  205 |  104 |     2 | 2011-01-22 |
|  208 |  104 |     3 | 2011-01-02 |
+------+------+-------+------------+
2 rows in set (0.00 sec)
```

Show the rating of Movie E.T.

Example: Subquery within a WHERE clause (cont.)

- **(b) Using with IN operators - Example**

```
SELECT *
FROM Rating
WHERE mID IN
(SELECT mID
FROM Movie
WHERE title
LIKE 'a%');
```

```
mysql> select mID from Movie
    -> where title LIKE 'a%';
+------+
| mID  |
+------+
|  107 |
|  109 |
+------+
2 rows in set (0.00 sec)
```

Show the rating of Movie that starts with an 'a'.

```
mysql> select * from Rating where mID IN
    -> (select mID from Movie where title LIKE 'a%');
+------+------+-------+------------+
| rID  | mID  | stars | ratingDate |
+------+------+-------+------------+
|  206 |  107 |     3 | 2011-01-15 |
|  207 |  107 |     5 | 2011-01-20 |
|  201 |  109 |     5 | 2015-09-12 |
|  204 |  109 |     3 | 2015-09-12 |
+------+------+-------+------------+
4 rows in set (0.00 sec)
```

Example: Subquery within a WHERE clause (cont.)

- **(b) Using with NOT IN operators - Example**

```
SELECT *
FROM Rating
WHERE mID NOT IN
(SELECT mID
FROM Movie
WHERE title
LIKE 'a%');
```

Show the rating of Movie that DOES NOT starts with an 'a'.

```
mysql> select * from Rating where mID NOT IN
    -> (select mID from Movie where title LIKE 'a%');
+------+------+-------+------------+
| rID  | mID  | stars | ratingDate |
+------+------+-------+------------+
|  201 |  101 |     2 | 2011-01-22 |
|  201 |  101 |     4 | 2011-01-27 |
|  202 |  106 |     4 | NULL       |
|  203 |  103 |     2 | 2011-01-20 |
|  203 |  108 |     4 | 2011-01-12 |
|  203 |  108 |     2 | 2011-01-30 |
|  204 |  101 |     3 | 2011-01-09 |
|  205 |  103 |     3 | 2011-01-27 |
|  205 |  104 |     2 | 2011-01-22 |
|  205 |  108 |     4 | NULL       |
|  206 |  106 |     5 | 2011-01-19 |
|  208 |  104 |     3 | 2011-01-02 |
|  201 |  110 |     4 | 2015-09-12 |
+------+------+-------+------------+
13 rows in set (0.00 sec)
```

Example: Subquery within a WHERE clause (cont.)

- **(c) Using with EXISTS and NOT EXISTS operators - Example**

```
SELECT *
FROM Rating
WHERE mID EXISTS
(SELECT mID
FROM Movie
WHERE title
LIKE 'aa%');
```

```
mysql> (select mID from Movie where title LIKE 'aa%');
Empty set (0.00 sec)
```

```
mysql> select * from Rating where exists
    -> (select mID from Movie where title LIKE 'aa%');
Empty set (0.00 sec)
```

This shows all the data in Rating if any data exists in the subquery result.

NOT EXISTS will show all the data in Rating if NO data exists in the subquery result

Example: Subquery within a FROM clause

- This subquery is within a FROM clause where the query will act as a table where the main query is selecting.

- For this it is required to give an ALIAS using AS statement to the subquery.

```
SELECT column1, column2, …
FROM (subquery) AS alias1
WHERE conditions;
```

Example: Subquery within a FROM clause

```
SELECT * FROM
(SELECT mID
FROM Movie
WHERE title
LIKE 'a%') AS MovieA
WHERE MovieA.year<2010;
```

```
mysql> select * from Movie where title like 'a%';
+------+--------+------+---------------+
| mID  | title  | year | director      |
+------+--------+------+---------------+
|  107 | Avatar | 2009 | James Cameron |
|  109 | Antah  | 2015 | Antah         |
+------+--------+------+---------------+
2 rows in set (0.00 sec)
```

```
mysql> select * from
    -> (select * from Movie
    -> where title like 'a%')
    -> AS MovieA
    -> where MovieA.year<2010;
+------+--------+------+---------------+
| mID  | title  | year | director      |
+------+--------+------+---------------+
|  107 | Avatar | 2009 | James Cameron |
+------+--------+------+---------------+
1 row in set (0.01 sec)
```

Correlated Subquery

- A correlated subquery is a subquery that uses the information from the outer query

- You can say that a correlated subquery depends on the outer query.

Example: Correlated Subquery

```
SELECT * FROM Rating rat
WHERE stars >
(SELECT AVG(stars)
FROM Rating
WHERE mID=rat.mID
AND rID=rat.rID);
```

```
mysql> select AVG(stars) from Rating;
+------------+
| AVG(stars) |
+------------+
|     3.4118 |
+------------+
1 row in set (0.00 sec)
```

```
mysql> select * from Rating rat
    -> where stars >
    -> (select AVG(stars) from Rating
    -> where mID=rat.mID and rID=rat.rID);
+------+------+-------+------------+
| rID  | mID  | stars | ratingDate |
+------+------+-------+------------+
|  201 |  101 |     4 | 2011-01-27 |
|  203 |  108 |     4 | 2011-01-12 |
+------+------+-------+------------+
2 rows in set (0.00 sec)
```

Join

- Join is use to combine two or more tables.

- There are four ways of joining tables.

1. Inner Join
2. Left Join
3. Right Join
4. Self Join

Example for Join

- **These tables will be used for Inner, Left and Right Join example.**

Student

stuID	stuName
201	Abu
202	Bakar
203	Siti

Grade

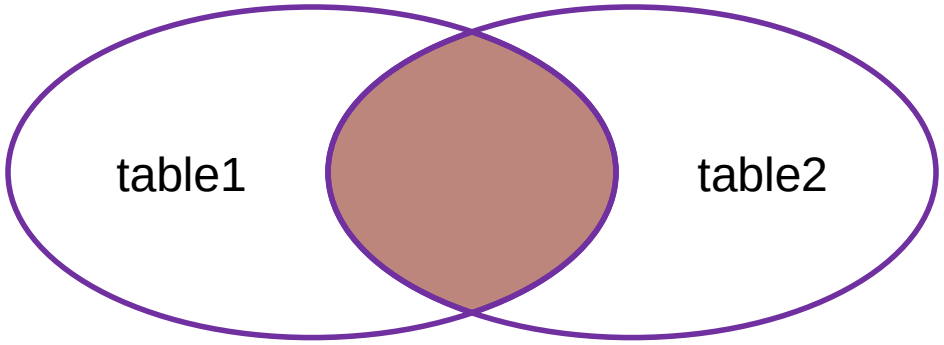
modID	stuID	Grade
DBMS	201	Distinction
DBMS	202	Merit
ISMS	202	Merit
ISMS	201	Distinction
OOP	204	Merit

Join – Inner Join

- The MySQL INNER JOIN clause matches rows in one table with rows in other tables and allows you to query rows that contain columns from both tables.

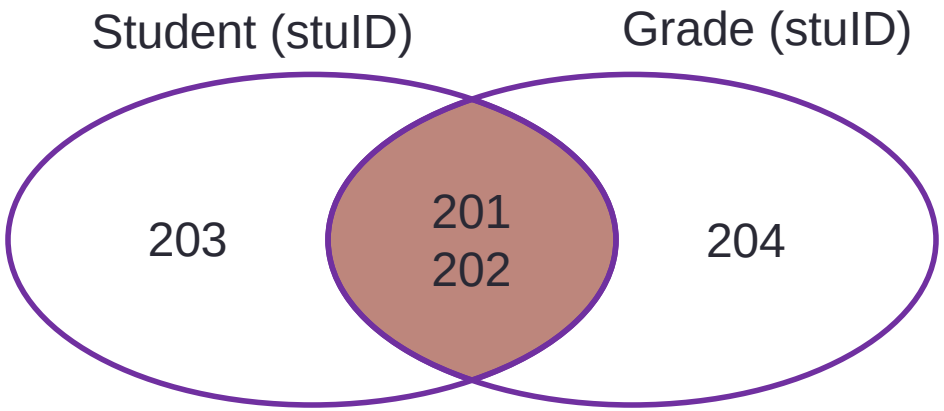
```
SELECT column1, column2, …
FROM table1
INNER JOIN table2 ON join_condition1
INNER JOIN table3 ON join_condition2
...
WHERE where_conditions;
```

This can be represented in venn diagram.



Example: Join – Inner Join

```
SELECT *
FROM Student s
INNER JOIN Grade g
ON s.stuID=g.stuID;
```

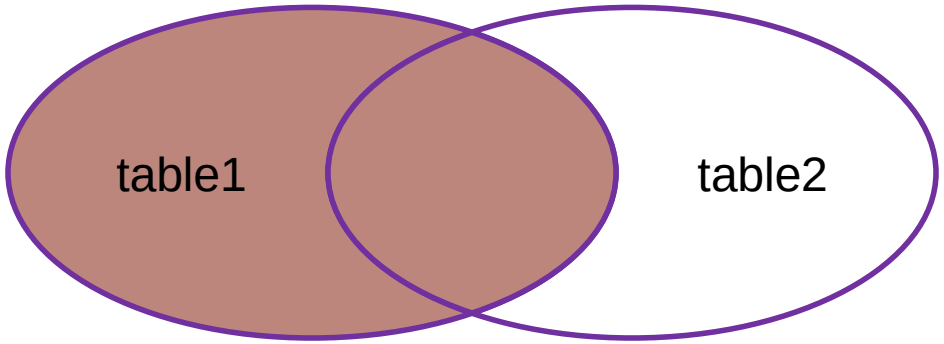


stuID	stuName	modID	stuID	Grade
201	Abu	DBMS	201	Distinction
202	Bakar	DBMS	202	Merit
202	Bakar	ISMS	202	Merit
201	Abu	ISMS	201	Distinction

Join – Left Join

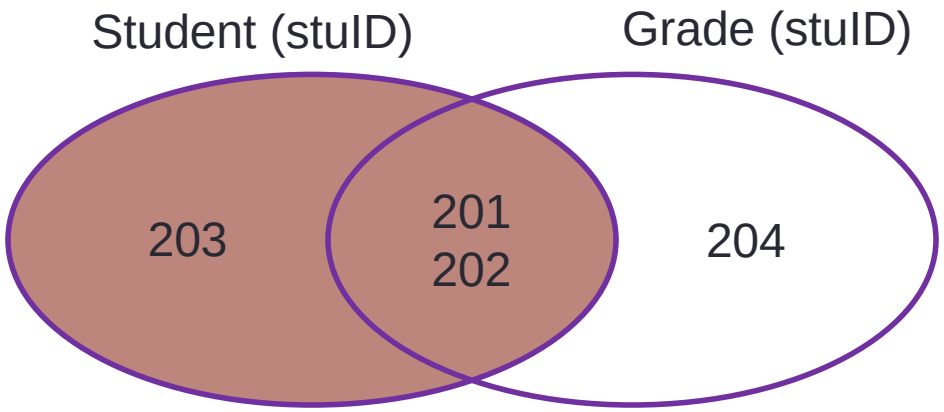
```
SELECT column1, column2, …
FROM table1
LEFT JOIN table2 ON join_condition1
LEFT JOIN table3 ON join_condition2
...
WHERE where_conditions;
```

This can be represented in venn diagram.



Example: Join – Left Join

```
SELECT *
FROM Student s
LEFT JOIN Grade g
ON s.stuID=g.stuID;
```

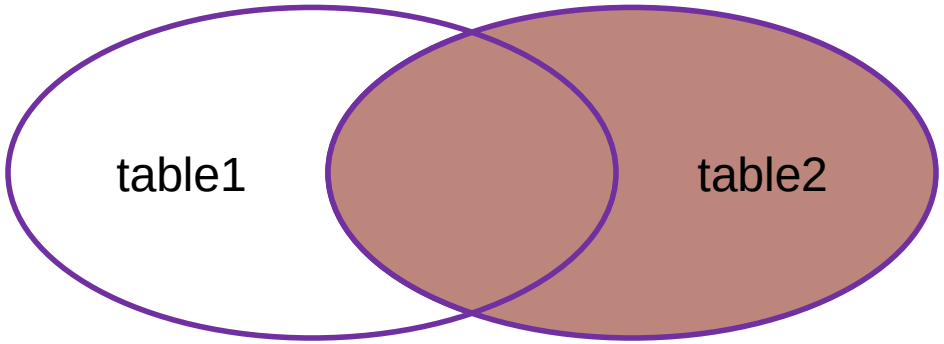


stuID	stuName	modID	stuID	Grade
201	Abu	DBMS	201	Distinction
202	Bakar	DBMS	202	Merit
202	Bakar	ISMS	202	Merit
201	Abu	ISMS	201	Distinction
203	Siti	NULL	NULL	NULL

Join – Right Join

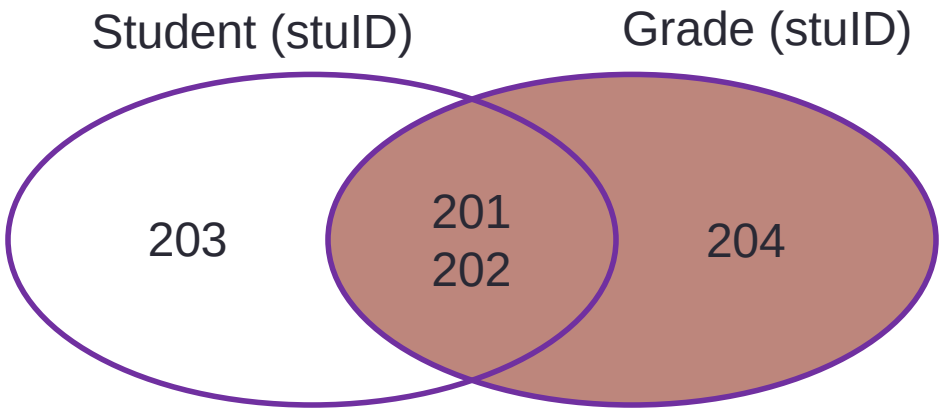
```
SELECT column1, column2, …
FROM table1
RIGHT JOIN table2 ON join_condition1
RIGHT JOIN table3 ON join_condition2
...
WHERE where_conditions;
```

This can be represented in venn diagram.



Example: Join – Right Join

```
SELECT *
FROM Student s
RIGHT JOIN Grade g
ON s.stuID=g.stuID;
```



stuID	stuName	modID	stuID	Grade
201	Abu	DBMS	201	Distinction
202	Bakar	DBMS	202	Merit
202	Bakar	ISMS	202	Merit
201	Abu	ISMS	201	Distinction
NULL	NULL	OOP	204	Merit

Join – Self Join

- Self Join is use to join a table with itself.
- There are **NO** MySQL Self Join clause.
  - So we use either INNER, LEFT or RIGHT Join

SELECT column1, column2, … **FROM** table1 **INNER JOIN** table1 **ON** join_condition1 **WHERE** where_conditions;	**SELECT** column1, column2, … **FROM** table1 **LEFT JOIN** table1 **ON** join_condition1 **WHERE** where_conditions;	**SELECT** column1, column2, … **FROM** table1 **RIGHT JOIN** table1 **ON** join_condition1 **WHERE** where_conditions;

Example: Join – Self Join

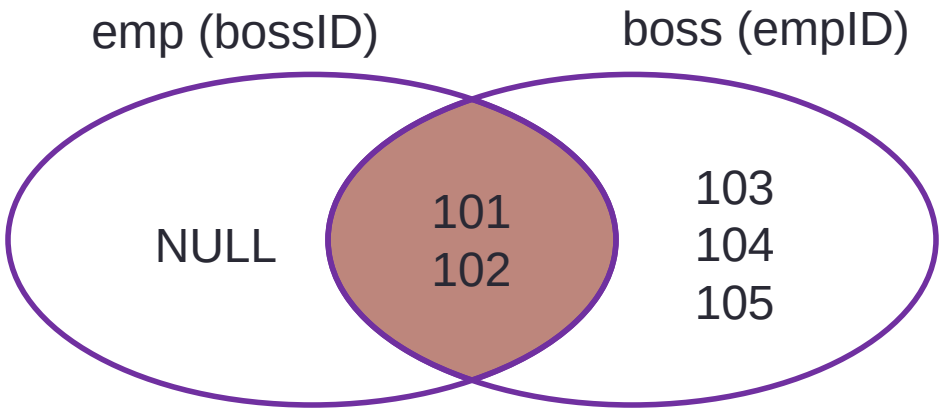
- For this example, we use **Employee** table:
  - Each employee has only one boss.
  - The boss is also the employee of the company.

empID	name	bossID
101	Abu	102
102	Siti	NULL
103	Rahman	101
104	Mohammad	101
105	Nurul	101

Example: Join – Self Join (cont.)

Self joining using INNER JOIN

```
SELECT *
FROM Employee emp
INNER JOIN Employee boss
ON emp.bossID=boss.empID;
```

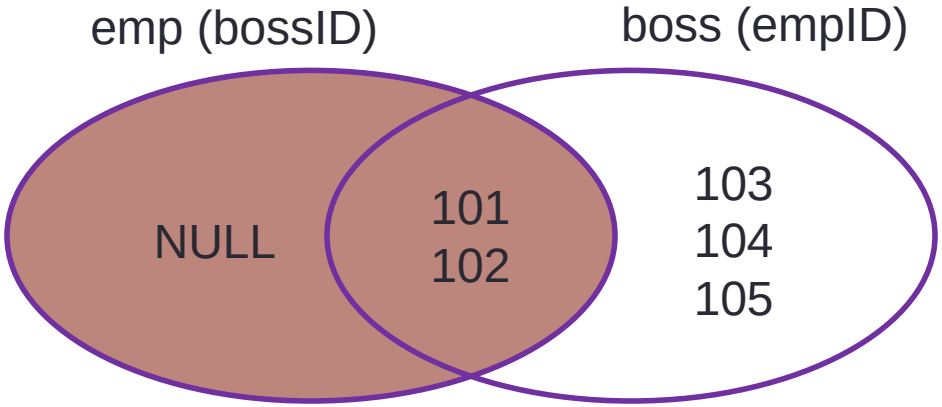


empID	name	bossID	empID	name	bossID
101	Abu	102	102	Siti	NULL
103	Rahman	101	101	Abu	102
104	Mohammad	101	101	Abu	102
105	Nurul	101	101	Abu	102

Example: Join – Self Join (cont.)

Self joining using LEFT JOIN

```
SELECT *
FROM Employee emp
LEFT JOIN Employee boss
ON emp.bossID=boss.empID;
```

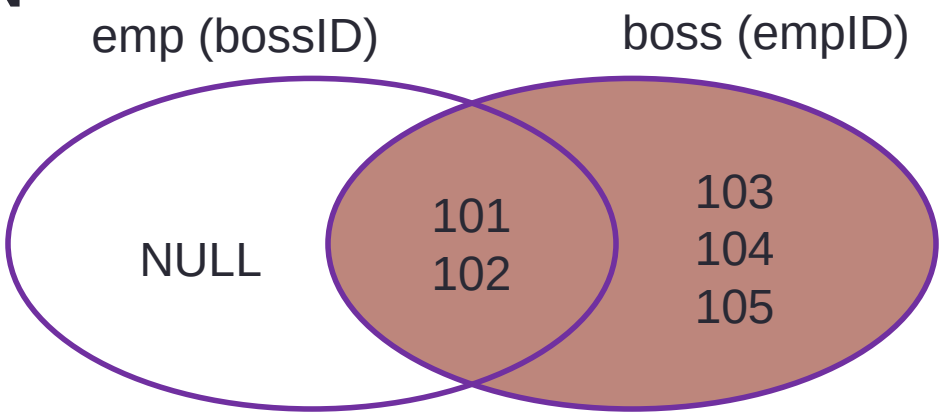


empID	name	bossID	empID	name	bossID
101	Abu	102	102	Siti	NULL
103	Rahman	101	101	Abu	102
104	Mohammad	101	101	Abu	102
105	Nurul	101	101	Abu	102
102	Siti	NULL	NULL	NULL	NULL

Example: Join – Self Join (cont.)

Self joining using RIGHT JOIN

```
SELECT *
FROM Employee emp
RIGHT JOIN Employee boss
ON emp.bossID=boss.empID;
```



empID	name	bossID	empID	name	bossID
101	Abu	102	102	Siti	NULL
103	Rahman	101	101	Abu	102
104	Mohammad	101	101	Abu	102
105	Nurul	101	101	Abu	102
NULL	NULL	NULL	103	Rahman	101
NULL	NULL	NULL	104	Mohammad	101
NULL	NULL	NULL	105	Nurul	101