CM1603 - Database Systems

Lecture 10| SQL JOINs & Sub Queries

Dileeka Alwis – Senior Lecturer Grade II / Level Coordinator,

Department of Computing, IIT

Learning Outcomes

- Covers LO3 for Module - Use SQL as a data definition and data manipulation language, and to query a relational database.

- Partially covers LO4 for Module – Implement and test a relational database using a query language with a suitable interface.

- On completion of this lecture, students are expected to be able to:
    - Create DDL and DML statements
    - Use mySql to create databases and manipulate data

Lesson Outline

- Introduction to SQL Join
- Types of SQL JOIN
  - INNER JOIN
  - OUTER JOIN : LEFT OUTER, RIGHT OUTER, FULL OUTER
  - SELF JOIN
- Sub query
  - Single row sub query
  - Multiple row sub query

SQL Join

- Combine records of two or more tables, **based on a common field** between the two tables.
- The two tables are linked (joined) together with the **primary key and foreign key relationship**.
- Performs a JOIN against equality or matching field values of the associated tables.
- An equal sign (=) is used as comparison operator in the WHERE clause to refer equality.

```
SELECT <Field_Names>
FROM <LEFT_Table_Name>
JOIN <RIGHT_Table_Name>
ON <Joining_Condition>
```

Example

PK **Student Table**

StudentID	FirstName	Surname	DOB	Gender
1001	Kate	West	12/10/1994	F
1002	Julie	McLain	3/7/1995	F
1003	Tom	Smith	24/12/1994	M
1004	Mark	Foster	5/11/1996	M
1005	Jane	Knight	17/6/1995	F
1006	Matt	Smith	24/12/1995	M
1007	Karen	Edwards	3/7/1995	M
1008	John	Williams	15/11/1996	M
1009	Allison	Cambell	10/10/1994	M
1010	Shirley	Thomas	15/4/1995	F

FK **Marks Table**

StudentID	ModuleID	Marks
1001	M2	54
1002	M3	67
1003	M1	84
1001	M1	94
1001	M3	38
1002	M1	54
1003	M3	67
1001	M7	82
1002	M4	55
1003	M2	25

Example

- Display student ID, name and total marks of each student.

> SELECT **S.**StudentID, FirstName, Surname, SUM(Marks) AS Total
> FROM Student **AS** S
> **JOIN** Marks **AS** M
> **ON S.**StudentID = **M.**StudentID;

Note:

- Field names of PK and FK are different.
- **JOIN** and **ON** keywords are used.
- **Alias** is used to refer tables easily.

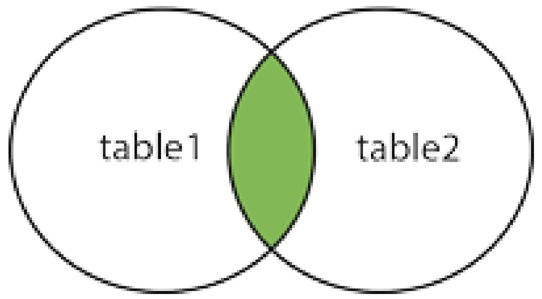
Types of SQL JOIN

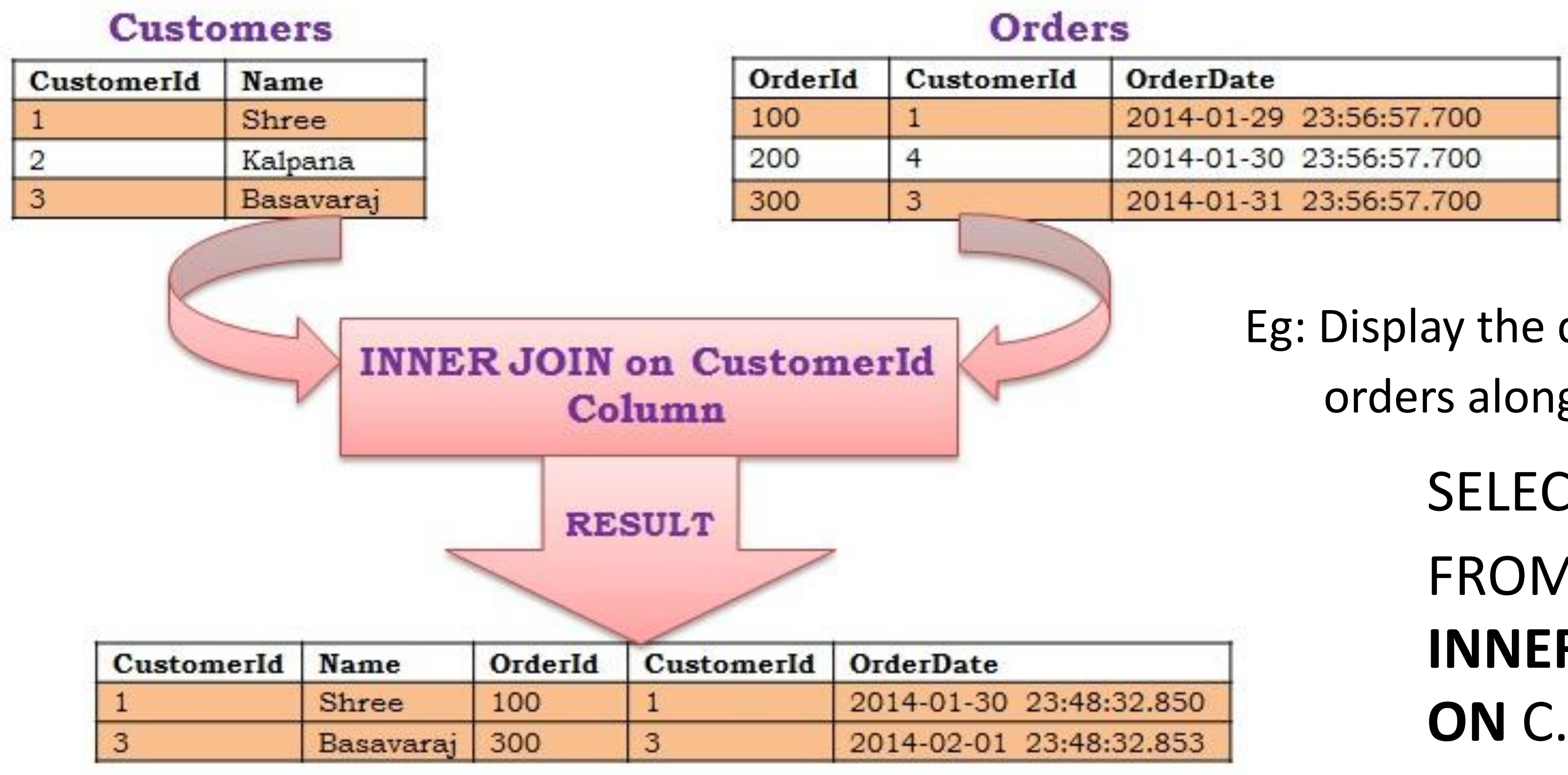
- INNER JOIN
- OUTER JOIN
  - LEFT OUTER JOIN
  - RIGHT OUTER JOIN
  - FULL OUTER JOIN
- SELF JOIN

ROBERT GORDON UNIVERSITY ABERDEEN | INFORMATICS INSTITUTE OF TECHNOLOGY

INNER JOIN

- Displays only the records that have matching values in both tables
- Default join in SQL.





Customers

CustomerId	Name
1	Shree
2	Kalpana
3	Basavaraj

Orders

OrderId	CustomerId	OrderDate
100	1	2014-01-29 23:56:57.700
200	4	2014-01-30 23:56:57.700
300	3	2014-01-31 23:56:57.700

INNER JOIN on CustomerId Column

RESULT

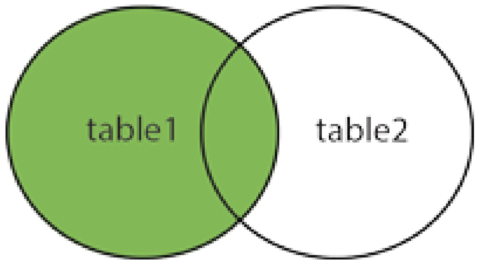
CustomerId	Name	OrderId	CustomerId	OrderDate
1	Shree	100	1	2014-01-30 23:48:32.850
3	Basavaraj	300	3	2014-02-01 23:48:32.853

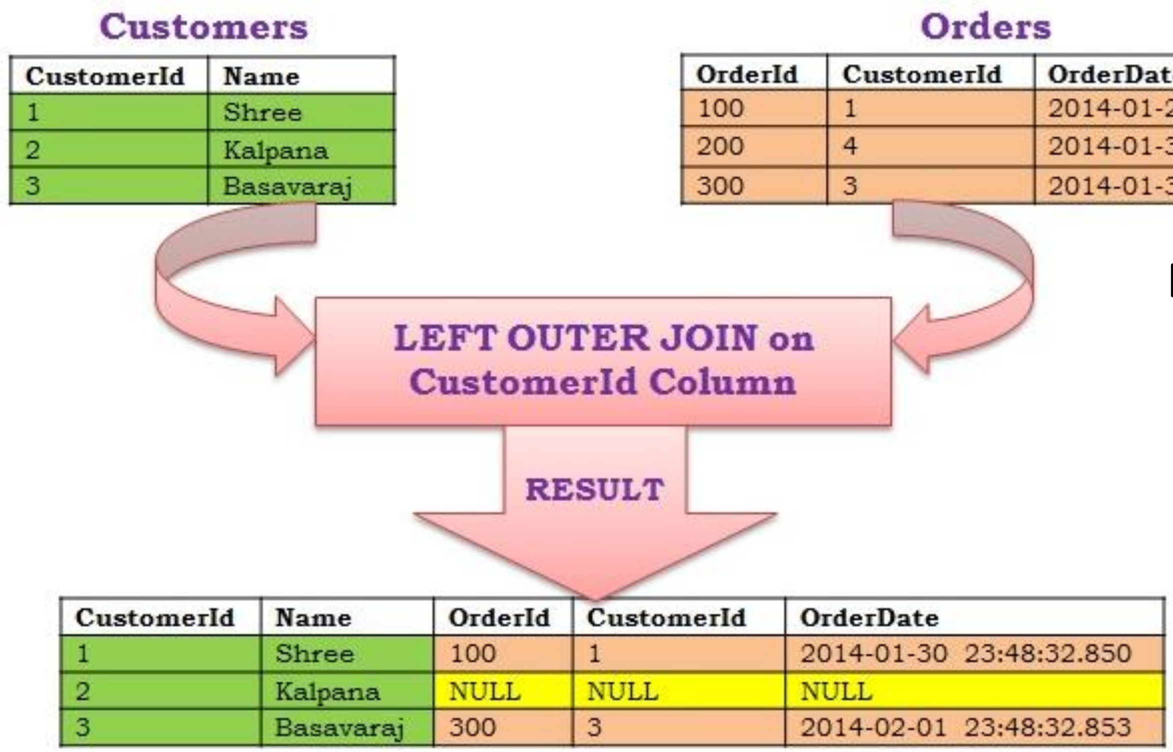
Eg: Display the details of customer who have placed orders along with their order details.

SELECT *
FROM Customers AS C
INNER JOIN Orders AS O
ON C.CustomerId = O.CustomerId

LEFT OUTER JOIN

- Return all rows from the left side table, and the matching rows from the right-side table.
- The result is NULL in the right side when there is no match.





Customers

CustomerId	Name
1	Shree
2	Kalpana
3	Basavaraj

Orders

OrderId	CustomerId	OrderDate
100	1	2014-01-29 23:56:57.700
200	4	2014-01-30 23:56:57.700
300	3	2014-01-31 23:56:57.700

CustomerId	Name	OrderId	CustomerId	OrderDate
1	Shree	100	1	2014-01-30 23:48:32.850
2	Kalpana	NULL	NULL	NULL
3	Basavaraj	300	3	2014-02-01 23:48:32.853

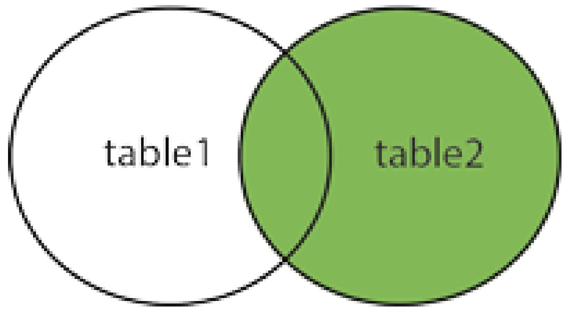
Eg: Display the details of all customers with the orders they have placed.

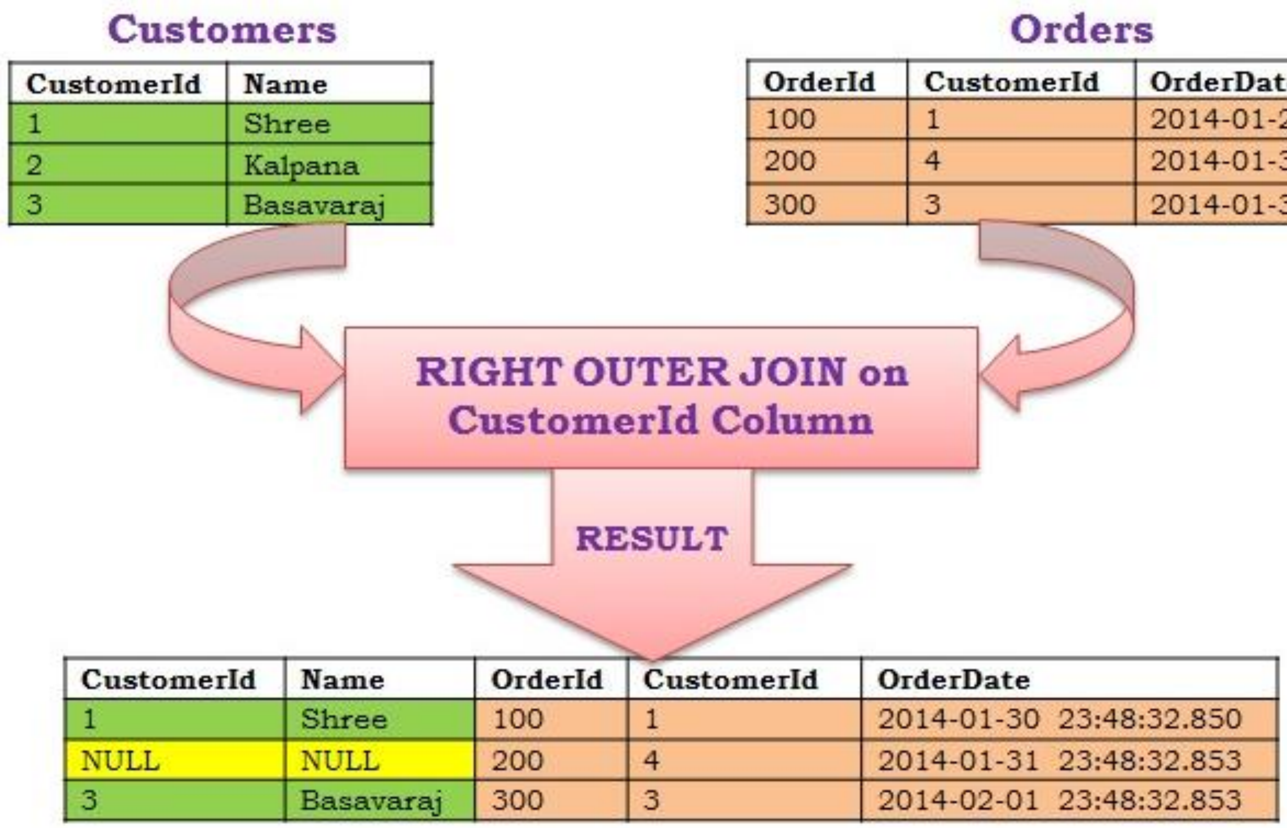
SELECT *

FROM Customers AS C
LEFT OUTER JOIN Orders AS O
ON C.CustomerId = O.CustomerId

RIGHT OUTER JOIN

- Return all rows from the right table, and the matched rows from the left table.
- The result is NULL in the left side when there is no match.





Customers

CustomerId	Name
1	Shree
2	Kalpana
3	Basavaraj

Orders

OrderId	CustomerId	OrderDate
100	1	2014-01-29 23:56:57.700
200	4	2014-01-30 23:56:57.700
300	3	2014-01-31 23:56:57.700

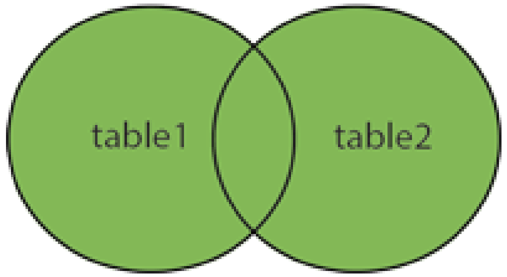
CustomerId	Name	OrderId	CustomerId	OrderDate
1	Shree	100	1	2014-01-30 23:48:32.850
NULL	NULL	200	4	2014-01-31 23:48:32.853
3	Basavaraj	300	3	2014-02-01 23:48:32.853

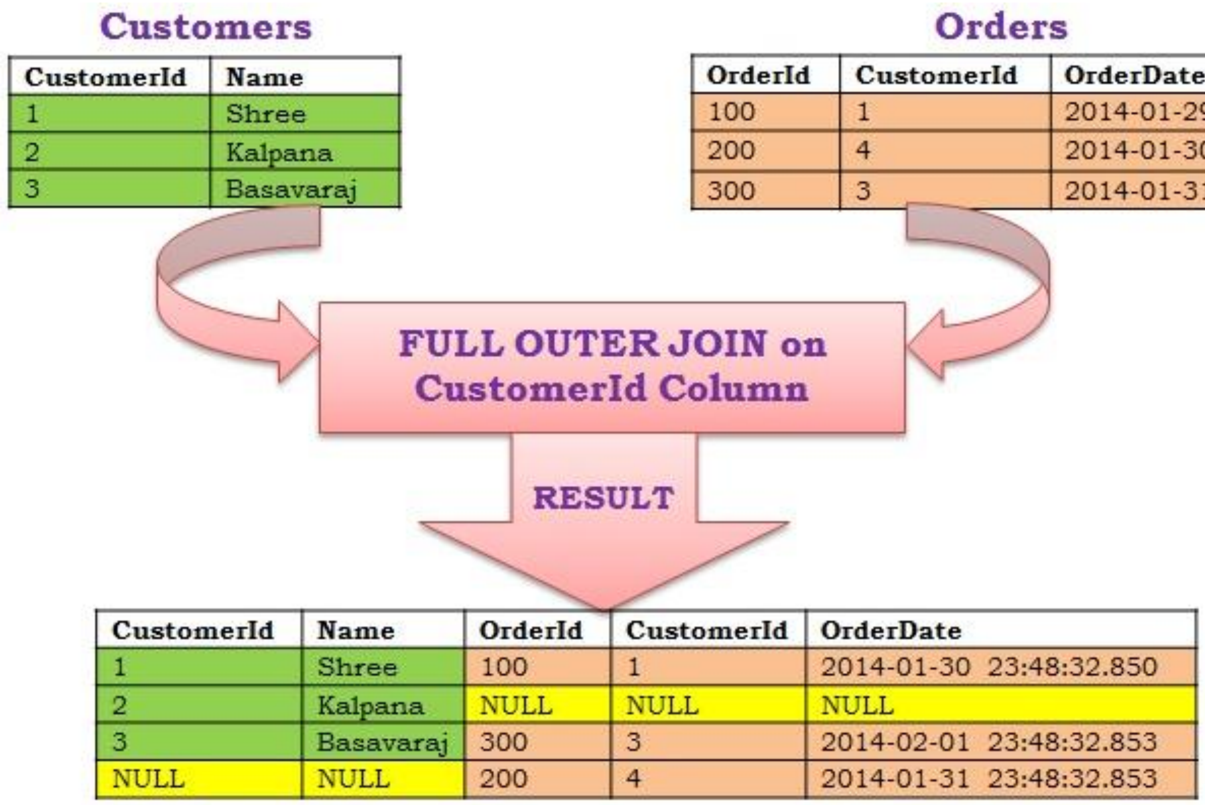
Eg: Display the details of all orders with the customers who placed them.

```
SELECT *
FROM  Customers AS C
RIGHT OUTER JOIN Orders AS O
ON C.CustomerId = O.CustomerId
```

FULL OUTER JOIN

- Return all rows when there is a match in ONE of the tables.
- Returns all the rows from both tables whether it has been matched or not.
- Combines the result of both LEFT and RIGHT joins.





Customers

CustomerId	Name
1	Shree
2	Kalpana
3	Basavaraj

Orders

OrderId	CustomerId	OrderDate
100	1	2014-01-29 23:56:57.700
200	4	2014-01-30 23:56:57.700
300	3	2014-01-31 23:56:57.700

CustomerId	Name	OrderId	CustomerId	OrderDate
1	Shree	100	1	2014-01-30 23:48:32.850
2	Kalpana	NULL	NULL	NULL
3	Basavaraj	300	3	2014-02-01 23:48:32.853
NULL	NULL	200	4	2014-01-31 23:48:32.853

Eg: Display the details of all the customers and all the orders.

SELECT *

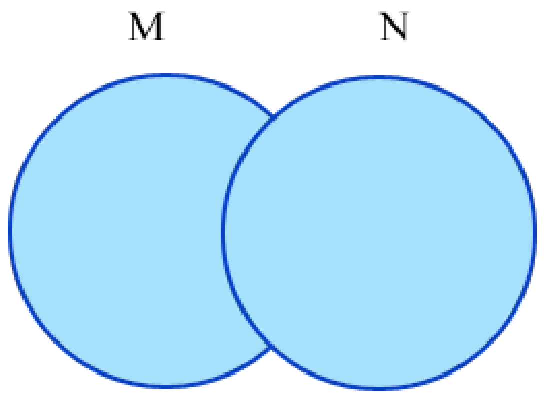
FROM Customers AS C

FULL OUTER JOIN Orders AS O

ON C.CustomerId = O.CustomerId

UNION Operator

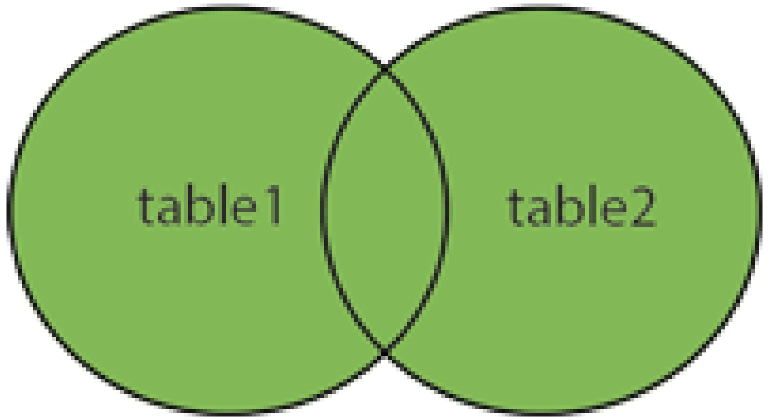
- MySQL **does not** support 'Full Outer Join'. The UNION of Left Join and Right Join gives the same result set instead.
- UNION combines the result sets of two or more SELECT statements.
- The default characteristic of UNION is, it removes the duplicate rows from the result.
- Each SELECT statement within the UNION must have
  - same number of columns
  - columns must have similar data types
  - columns in each SELECT statement must be in the same order



UNION Operator

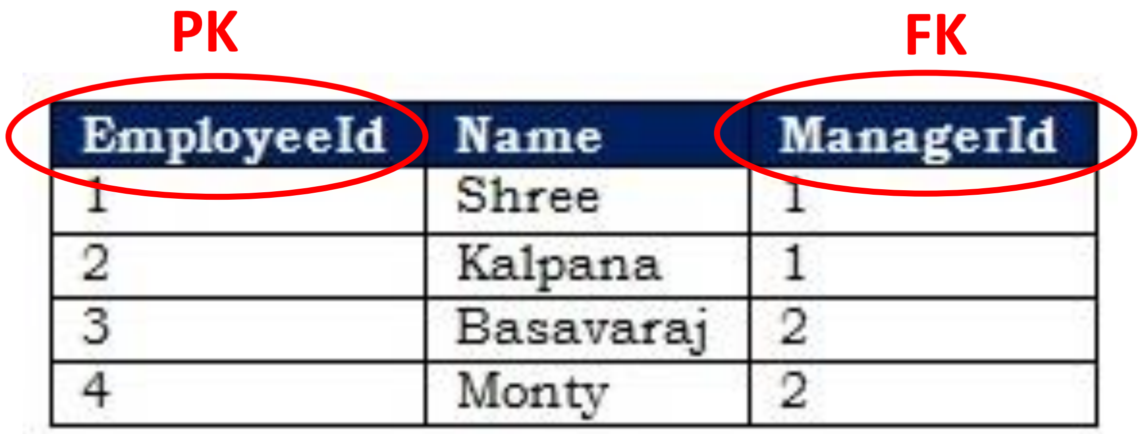
- Display the details of all the customers and all the orders.

```
SELECT *
FROM  Customers AS C
LEFT OUTER JOIN Orders AS O
ON C.CustomerId = O.CustomerId
UNION
SELECT *
FROM  Customers AS C
RIGHT OUTER JOIN Orders AS O
ON C.CustomerId = O.CustomerId
```



SELF JOIN

- A table is joined to itself using one an inner join or outer join.
- Self join is used to retrieve the records having some relationships or similarity with other records in the same table.
- Need to use aliases for the same table to set a self join and retrieve records satisfying by the condition in WHERE clause.



EmployeeId	Name	ManagerId
1	Shree	1
2	Kalpana	1
3	Basavaraj	2
4	Monty	2

Eg: Display the name of the employee and his manager's name.

```
SELECT E.Name AS 'Employee Name’,
M.Name AS ‘Manager Name’
FROM Employee AS E
INNER JOIN Employee AS M
ON E.ManagerId = M.EmployeeId
```

SQL Sub Queries

- A Sub query (Inner query/ Nested query) is a query within another SQL query that is embedded to the WHERE clause of the main query.

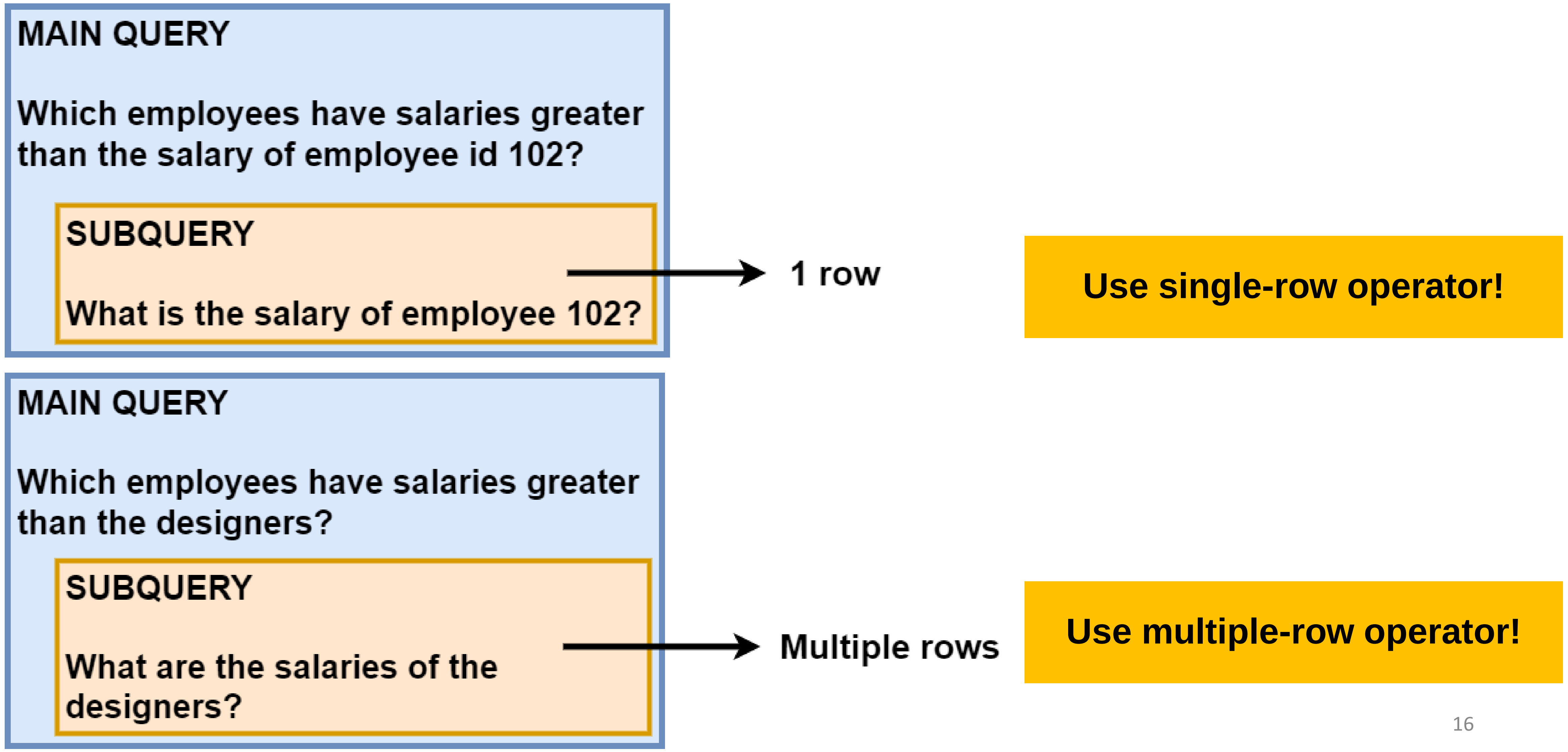
- Enclose sub query within parentheses and indent to the right of the main query.

- Can’t apply on columns containing text.

- The parameter required by the condition in the main query, must be the same that is returned by the sub query.

- Sub query may return a **single value** or **list of values**.
  - If the subquery returns one row, use single-row comparison operators.
  - If the subquery returns multiple rows, use multiple-row comparison operators.

Single-row vs. multiple-row subquery



MAIN QUERY

Which employees have salaries greater than the salary of employee id 102?

SUBQUERY

What is the salary of employee 102?

1 row

Use single-row operator!

MAIN QUERY

Which employees have salaries greater than the designers?

SUBQUERY

What are the salaries of the designers?

Multiple rows

Use multiple-row operator!

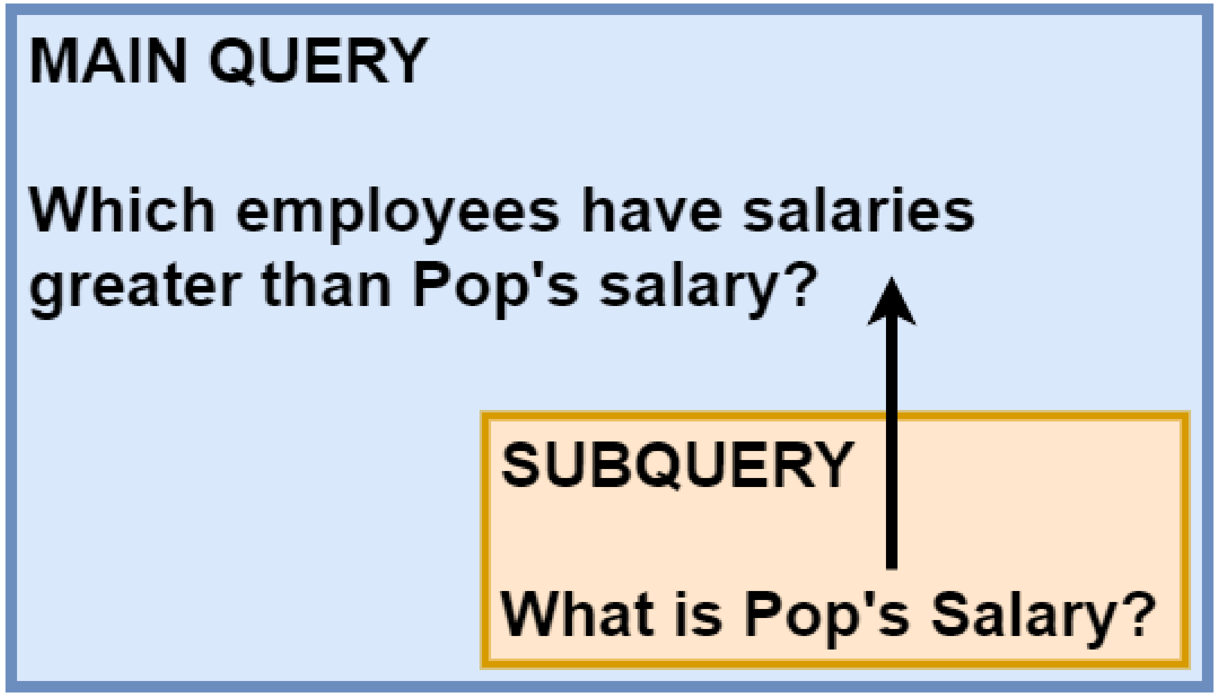
Single-row comparison operators

- If subquery returns **ONE ROW**, use these operators:

Operator	Description
=	Equal to
>	Strictly greater than
>=	Greater than or equal to
<	Strictly less than
<=	Less than or equal to
<>	Not equal to

Example

- Find out who earns more than Mr. Pop.



```
SELECT fName, lName, salary
FROM Employee
WHERE salary >
        (SELECT salary
         FROM Employee
         WHERE lName = 'Pop');
```

Multiple-row comparison operators

- If subquery returns **MULTIPLE ROWS**, use these operators:

Operator	Description
IN	Matches any member of the list.
ANY	Compare value to each value returned by subquery. Condition verified if it compares favourably with **AT LEAST ONE** of the returned values.
ALL	Compare value to each value returned by subquery. Condition verified if it compares favourably with **ALL** of the returned values.

Example

- Find out whose salaries are similar to database staff salary?

```
SELECT fName, lName, position, salary
FROM Emp
WHERE  salary IN
                    (SELECT salary
                     FROM Emp
                     WHERE position LIKE '%Database%’)
AND position NOT LIKE '%Database%';
```

Example

- Find out who earn less than any of the database staff?

```
SELECT fName, lName, position, salary
FROM Emp
WHERE  salary < ANY
                    (SELECT salary
                     FROM Emp
                     WHERE position LIKE '%Database%’)
AND position NOT LIKE '%Database%';
```

Example

- Find out who earn less than all of the database staff?

```
SELECT fName, lName, position, salary
FROM Emp
WHERE  salary < ALL
                    (SELECT salary
                     FROM Emp
                     WHERE position LIKE '%Database%’)
AND position NOT LIKE '%Database%';
```

Thank you

Contact: dileeka.a@iit.ac.lk