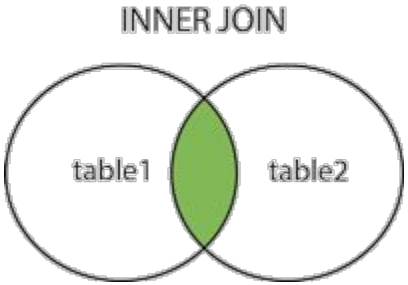
Tutorial 4

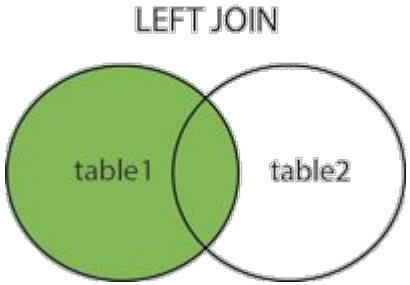
Venika Sruthi
Abhishek Sharma

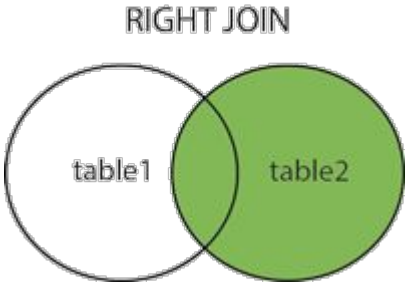
INNER JOIN




```
SELECT column_name(s)

FROM table1

INNER JOIN table2

ON table1.column_name = table2.column_name;
```

LEFT JOIN and RIGHT JOIN

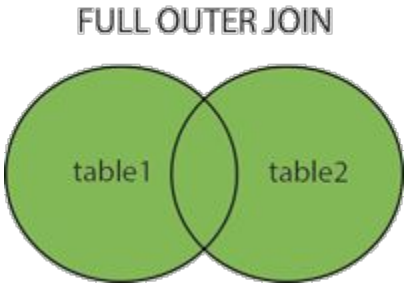





```
SELECT column_name(s)

FROM table1

LEFT JOIN table2

ON table1.column_name =
table2.column_name;
```

CustomerName	OrderID	OrderDate
Alfreds Futterkiste	*null*	*null*
Ana Trujillo Emparedados y helados	10308	1996-09-18
Antonio Moreno Taquería	10365	1996-11-27
Around the Horn	10355	1996-11-15

FULL OUTER JOIN




```
SELECT column_name(s)
FROM table1
FULL OUTER JOIN table2
ON table1.column_name = table2.column_name
WHERE condition;
```

CustomerName	OrderID
Null	10309
Null	10310
Alfreds Futterkiste	*Null*
Ana Trujillo Emparedados y helados	10308
Antonio Moreno Taquería	*Null*

VIEWS

Views in SQL are kind of virtual tables. A view also has rows and columns as they are in a real table in the database. We can create a view by selecting fields from one or more tables present in the database.

```
CREATE VIEW view_name AS

SELECT column1, column2, ...

FROM table_name

WHERE condition;
```

A view always shows up-to-date data! The database engine recreates the view, every time a user queries it.

UPDATING VIEWS

1. The SELECT statement which is used to create the view should not include GROUP BY clause or ORDER BY clause.
2. The SELECT statement should not have the DISTINCT keyword.
3. The View should have all NOT NULL values.
4. The view should not be created using nested queries or complex queries.
5. The view should be created from a single table. If the view is created using multiple tables then we will not be allowed to update the view.

UPDATING VIEWS

Add or remove fields for view :

```
CREATE OR REPLACE VIEW view_name AS
SELECT column1,column2,..
FROM table_name
  WHERE condition;
```

UPDATING VIEWS

Inserting a row in the view :

```
INSERT INTO view_name(column1, column2 , column3,..)
VALUES(value1, value2, value3..);
```

Deleting :

```
DELETE FROM view_name
WHERE condition;
```

WITH

The clause is used for defining a temporary relation such that the output of this temporary relation is available and is used by the query that is associated with the WITH clause. The name assigned to the sub-query is treated as though it was an inline view or table.

```
WITH temporaryTable (averageValue) as
    (SELECT avg(Attr1)
    FROM Table)
    SELECT Attr1
    FROM Table, temporaryTable
      WHERE Table.Attr1 > temporaryTable.averageValue;
```

SUB QUERIES

A subquery, or nested query, is a query placed within another SQL query. When requesting information from a database, you may find it necessary to include a subquery into the SELECT, FROM , JOIN, or WHERE clause.

Example :

In Class Activity Question 1:

Retrieve the names of all employees who work in the department that has the employee with the highest salary among all employees.

```
SELECT Employee_name FROM Employee WHERE Salary = (Select MAX(Salary) from Employee)
;

SELECT Employee_name FROM Employee WHERE Salary = (Select Salary FROM Employee ORDER
BY SALARY DESC LIMIT 1) ;
```

`galleries`

id	city
1	London
2	New York
3	Munich

`paintings`

id	name	gallery_id	price
1	Patterns	3	5000
2	Ringer	1	4500
3	Gift	1	3200
4	Violin Lessons	2	6700
5	Curiosity	2	9800

`sales_agents`

id	last_name	first_name	gallery_id	agency_fee
1	Brown	Denis	2	2250
2	White	Kate	3	3120
3	Black	Sarah	2	1640
4	Smith	Helen	1	4500
5	Stewart	Tom	3	2130

`managers`

id	gallery_id
1	2
2	3
4	1

Types of SubQueries

There are several types of SQL subqueries:

1. **Scalar subqueries** return a single value, or exactly one row and exactly one column.

   Use Single Row Comparison Operators : = , > , >= , < , <= , <> or =!

   Example :

   Show only those sales agents who received a higher-than-average agency fee last month:

   SELECT * FROM sales_agent WHERE agency_fee > (SELECT AVG(agency_fee)

   FROM sales_agents);

2. **Multirow subqueries** return either:

- One column with multiple rows (i.e. a list of values), or
- Multiple columns with multiple rows (i.e. tables).
- Use Multiple Comparison Operator like like IN, NOT IN, ANY, ALL, EXISTS, or NOT EXISTS

Example :

Calculate the average agency fee for those agents who are not managers.

```
SELECT AVG(agency_fee) FROM sales_agents
WHERE id NOT IN (SELECT id  FROM managers);
```

Show the first name of agents who does not have access to gallery 2 but have

```
SELECT first_name FROM sales_agent WHERE agency_fee < ANY(SELECT agency_fee
FROM sales_agent WHERE gallery_id = 2) AND gallery_id <> 2;
```

3. **Correlated Subqueries**

- inner query relies on information obtained from the outer query

Example :

Calculate the number of paintings found in each of our galleries

```
SELECT city,
 (SELECT count(*)
  FROM paintings p
  WHERE g.id = p.gallery_id) total_paintings
FROM galleries g;
```

The inner query depends on the outer query. We pull the gallery ID from the `galleries` table, which is in the outer query. In other words, you cannot run the inner query as an independent query – it will just throw an error.

Write Query using Join using previous SubQuery:

```
SELECT g.city, count(p.name) AS total_paintings
FROM galleries g
JOIN paintings p
ON g.id = p.gallery_id
GROUP BY g.city;
```

Practice Questions

JOINS:

1. https://www.hackerrank.com/challenges/full-score/problem?isFullScreen=true
2. https://www.hackerrank.com/challenges/placements/problem?isFullScreen=true

Subqueries:

1. https://www.interviewbit.com/problems/job-offers-2-0/
2. https://www.interviewbit.com/problems/actors-and-their-movies/

Doubts?