Window Functions

Agenda:

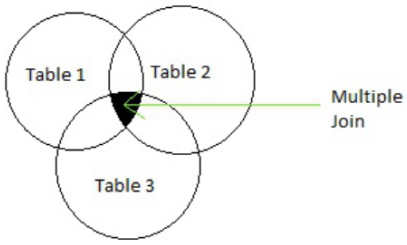
In today's session, we'll cover essential topics, including:-

- Problem Statement
- Over(), Partition By
- Aggregated Window Functions
- Row_number()
- Rank(), Dense_rank()

Summary of Previous Lecture:

MULTIPLE JOINS

- Multiple joins can be described as a query containing joins of the same or different types used multiple times, thus giving them the ability to combine more than two tables.



SELF JOIN

- In this type of join, we join a table to itself.
- **Example:** Given the *employee* table. For each employee, find out the name of their manager.

emp_id	emp_name	job_name	manager_id	hire_date	salary	commission	dep_id
68319	KAYLING	PRESIDENT		1991-11-18	6000.00		1001
66928	BLAZE	MANAGER	68319	1991-05-01	2750.00		3001
67832	CLARE	MANAGER	68319	1991-06-09	2550.00		1001
65646	JONAS	MANAGER	68319	1991-04-02	2957.00		2001
64989	ADELYN	SALESMAN	66928	1991-02-20	1700.00	400.00	3001
65271	WADE	SALESMAN	66928	1991-02-22	1350.00	600.00	3001

```
SELECT
          emp.emp_name AS employee,
          mgr.emp_name AS manager
```

```
FROM employees AS emp
JOIN employees AS mgr
ON mgr.emp_id = emp.manager_id
```

CROSS JOIN

- Combines every row from one table with every row from another table, resulting in a Cartesian product.
- Syntax:

```
SELECT columns
FROM table1
CROSS JOIN table2;
```

Equi & Non-equi Joins

- Equi Join: Regular join based on the equality of columns.
- Non-equi Join: Join based on inequality operators like <, >, <=, >=, !=, BETWEEN, etc.
- **Example: Recommend toys to each kid who is above the minimum required age to play with those toys.**
- Non-equi Join using >= operator.
- Query:

```
SELECT *
FROM dataset.Toys_info T
JOIN dataset.Kids_info K
ON K.Age >= T.min_age;
```