JOINS AND BASIC FUNCTIONS

BY VISHWA PATEL

SHIFTKEY LABS

JANUARY EVENTS

JAN 6	ACADEMY UP: SQL FOUNDATIONS	6:00PM-8:00PM	Goldberg Computer Science Building
JAN 10	INDUSTRY SHOWCASE: UBISOFT	4:00PM-5:30PM	Ubisoft Halifax
JAN 13	ACADEMY UP: SQL FOUNDATIONS	6:00PM-8:00PM	Goldberg Computer Science Building
JAN	INDUSTRY SHOWCASE: CIBC	Cancelled	
JAN 16	CYBERSECURITY IN NETWORKS	6:00PM-8:00PM	Goldberg Computer Science Building
JAN 17-19	GEN AI AND MENTAL WELLBEING HACKATHON		Goldberg Computer Science Building
JAN 20	ACADEMY UP: SQL FOUNDATIONS	6:00PM-8:00PM	Goldberg Computer Science Building
JAN 23	INDUSTRY SHOWCASE: XEROX CANADA	4:00PM-5:30PM	Goldberg Computer Science Building
JAN 24-26	GLOBAL GAME JAM		CHEB and Goldberg Computer Science Building
JAN 25	GAME JAM NEXT	10:00AM-3:00PM	The PIER
JAN 27	INDUSTRY SHOWCASE: IBM	4:00PM-5:30PM	IBM Client Innovation Centre Nova Scotia
JAN 27	ACADEMY UP: SQL FOUNDATIONS	6:00PM-8:00PM	Goldberg Computer Science Building
JAN 29	PROMPT ENGINEERING	6:00PM-7:00PM	Goldberg Computer Science Building
JAN 30	SCRUM AGILE WORKSHOP	5:30-7:30PM	Goldberg Computer Science Building

SQL KEYWORDS

LIKE, AGGREGATE FUNCTIONS, AS, GROUP BY, HAVING

LIKE

- LIKE keyword is for finding patterns using WHERE in a column
- 2 common wildcards used are:
  - _ (underscore) -> single character
  - % (percent) -> 0/1/multiple characters

```
SELECT column1,column2… FROM table_name WHERE column1 LIKE ‘abc_’;
```

OR

```
SELECT column1,column2… FROM table_name WHERE column1 LIKE ‘abc%’;
```

AS

- AS is used to give alias (temporary) name to any table or a column in table

SELECT column1 **AS oneWord** ,column2… FROM table_name;

OR

SELECT column1 **AS ‘two words or more’** ,column2… FROM table_name;

QUERY AS EXPRESSIONS

- Use expressions to write more complex logic on column values in a query
- Use it with AS for displaying the result in pretty way
- Use BODMAS rules for mathematical expressions

```
SELECT (column1/2)*5 AS
new_name ,column2… FROM
table_name;
```

AGGREGATE FUNCTIONS

- MIN and MAX - Finds the minimum/ maximum of the column selected
- SUM - sums the values of the column
- COUNT - counts the number of rows that satisfy the condition
- AVG - average value of the column or selected columns is returned
- Mostly used with GROUP BY

AGGREGATE FUCTIONS

SELECT **MIN**(column1),column2… FROM table_name;

SELECT **COUNT**(column1) ,column2… FROM table_name GROUP BY column2;

SELECT **SUM**(column1),column2… FROM table_name GROUP BY column2;

SELECT **AVG**(column1),column2… FROM table_name;

GROUP BY

- Can't use it without an aggregate function
- Groups the data with one or more columns

SELECT column1,column2… FROM table_name **GROUP BY** column1;

HAVING

- Used with GROUP BY keyword
- To add an additional condition to the query after grouping data

```
SELECT column1,column2… FROM
table_name GROUP BY column1
HAVING column2 > value;
```

ORDER OF EXECUTION

SELECT DISTINCT column, AGG_FUNC(*column_or_expression*), ….

FROM mytable

WHERE *constraint_expression*

GROUP BY column

HAVING *constraint_expression*

ORDER BY *column* ASC/DESC

LIMIT *count* OFFSET *COUNT;*

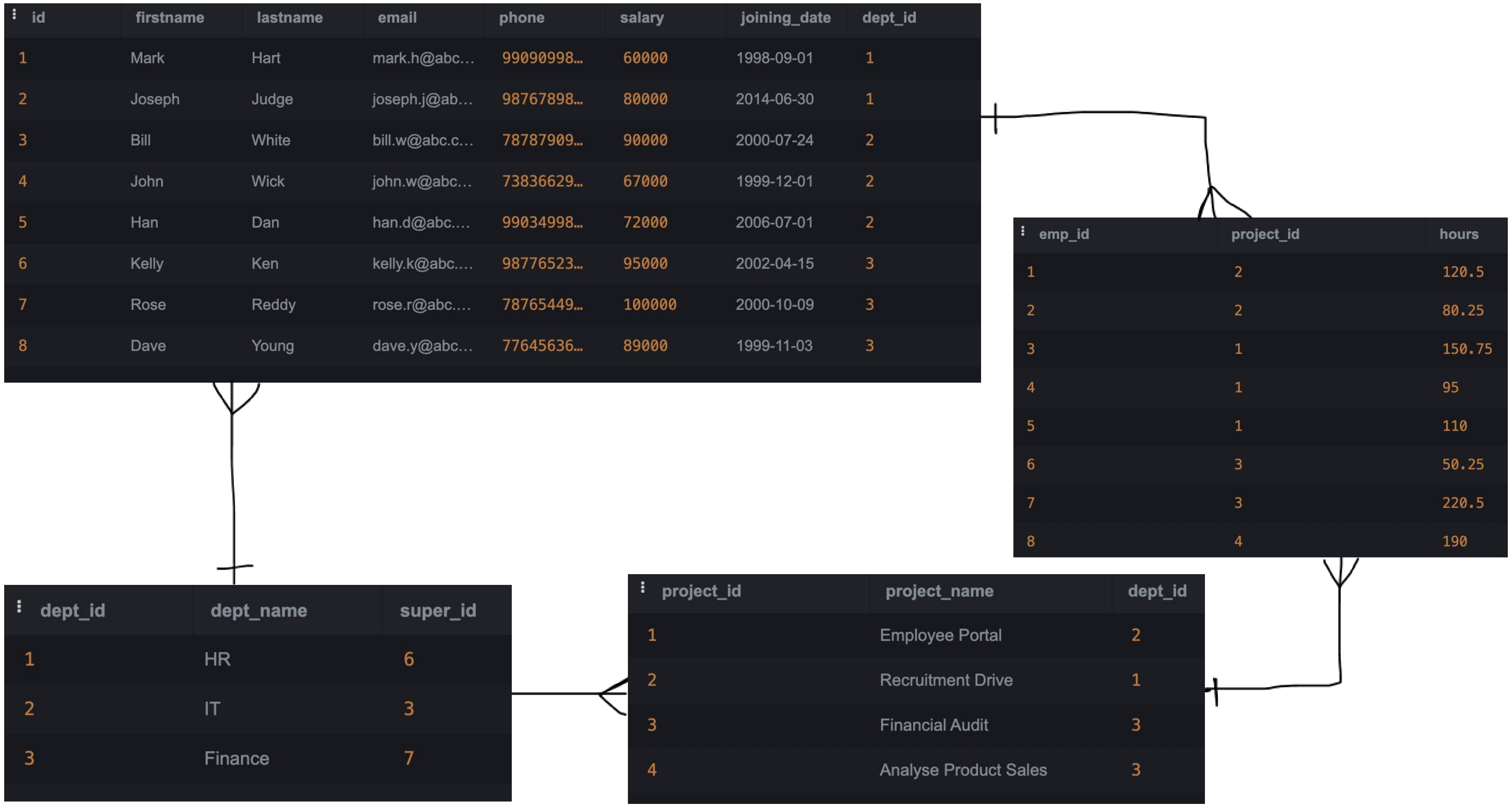
Note: Constraint expression would contain and, or, between and other conditional statements.

JOINS

INNER JOIN, LEFT JOIN, RIGHT JOIN AND OUTER JOIN (FULL JOIN)

DATABASE FOR JOINS

- For learning about joins we will need more than one table because we will be joining 2 tables or more
- We have an employees table to store information about employees
- We have departments table to store departments information and the manager id for that department
- Projects table to store what projects each department is working on
- Working_on table to store what employees are working on what project and how many hours have they invested in a project



id	firstname	lastname	email	phone	salary	joining_date	dept_id
1	Mark	Hart	mark.h@abc…	99090998…	60000	1998-09-01	1
2	Joseph	Judge	joseph.j@ab…	98767898…	80000	2014-06-30	1
3	Bill	White	bill.w@abc.c…	78787909…	90000	2000-07-24	2
4	John	Wick	john.w@abc…	73836629…	67000	1999-12-01	2
5	Han	Dan	han.d@abc.…	99034998…	72000	2006-07-01	2
6	Kelly	Ken	kelly.k@abc.…	98776523…	95000	2002-04-15	3
7	Rose	Reddy	rose.r@abc.…	78765449…	100000	2000-10-09	3
8	Dave	Young	dave.y@abc…	77645636…	89000	1999-11-03	3

emp_id	project_id	hours
1	2	120.5
2	2	80.25
3	1	150.75
4	1	95
5	1	110
6	3	50.25
7	3	220.5
8	4	190

dept_id	dept_name	super_id
1	HR	6
2	IT	3
3	Finance	7

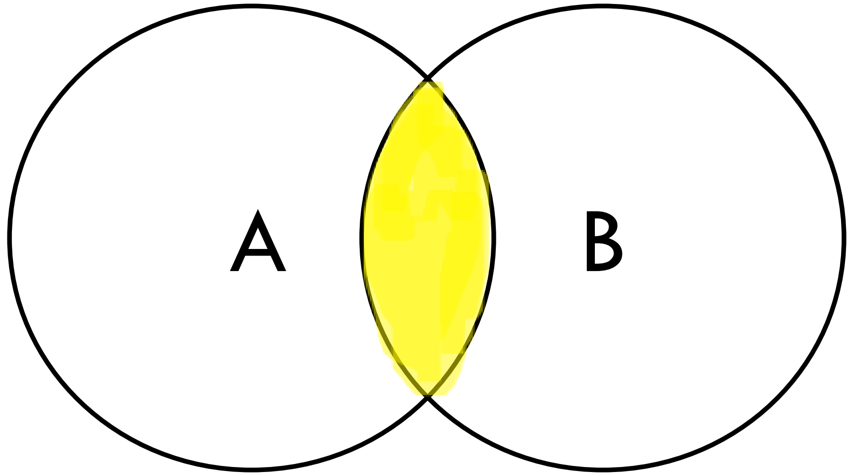
project_id	project_name	dept_id
1	Employee Portal	2
2	Recruitment Drive	1
3	Financial Audit	3
4	Analyse Product Sales	3

TYPES OF JOINS

- INNER JOIN – intersection
- LEFT JOIN – intersection and columns from first table
- RIGHT JOIN – intersection and columns from second table
- FULL/OUTER JOIN – first table, intersection and second table
- Combining information from different tables based on the related column

INNER JOIN

- Selects the records which are present in both the tables
- You can even write just JOIN



INNER JOIN

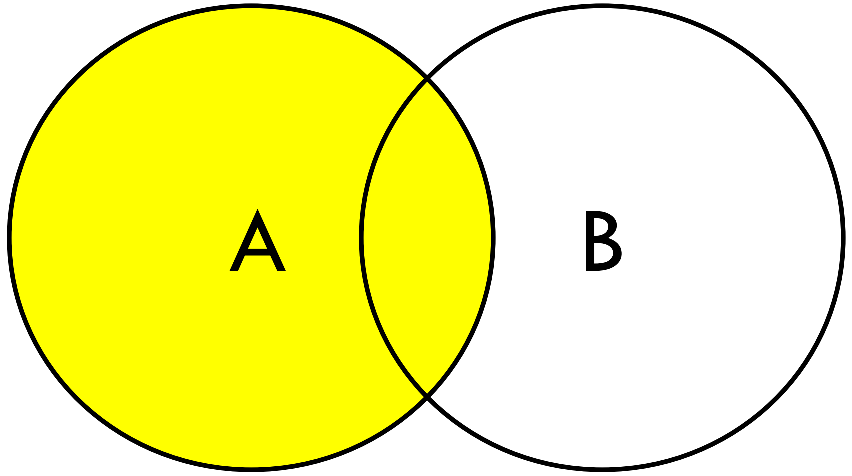
```
SELECT table1.column1,
table2.column2, … FROM table1
INNER JOIN table2 WHERE
table1.column = table2.column;
```

OR

```
SELECT table1.column1,
table2.column2, … FROM table1
INNER JOIN table2 ON
table1.column = table2.column;
```

LEFT JOIN

- Selects the records which are present in both the tables, and which are only on the first/left table



LEFT JOIN

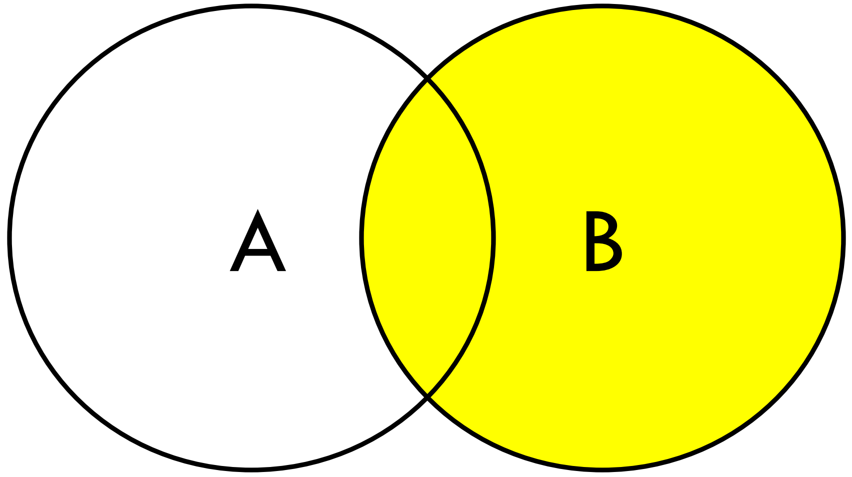
SELECT table1.column1, table2.column2, … FROM table1 **LEFT JOIN** table2 **WHERE** table1.column = table2.column;

OR

SELECT table1.column1, table2.column2, … FROM table1 **LEFT JOIN** table2 **ON** table1.column = table2.column;

RIGHT JOIN

- Selects the records which are present in both the tables, and which are only on the second/right table
- NULL for the columns it does not have a value for



RIGHT JOIN

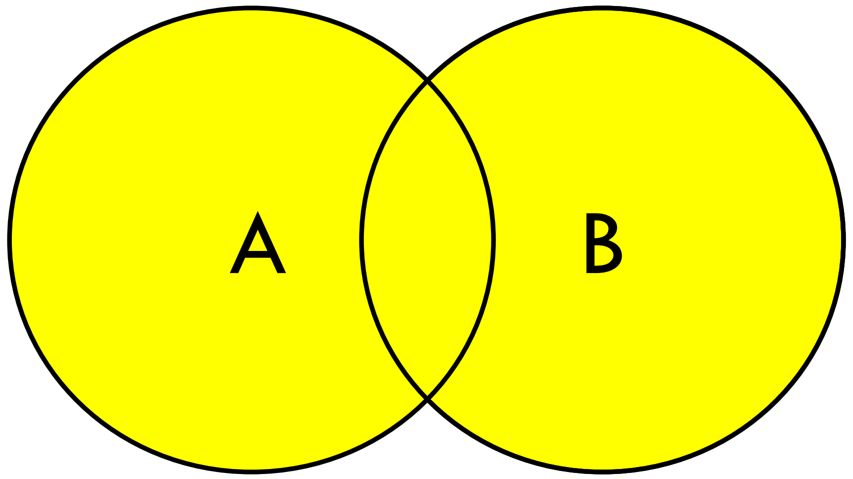
SELECT table1.column1, table2.column2, … FROM table1 **RIGHT JOIN** table2 **WHERE** table1.column = table2.column;

OR

SELECT table1.column1, table2.column2, … FROM table1 **RIGHT JOIN** table2 **ON** table1.column = table2.column;

FULL/OUTER JOIN

- Returns all records when there is a match in left or right table records and intersection records are only displayed once



FULL/OUTER JOIN

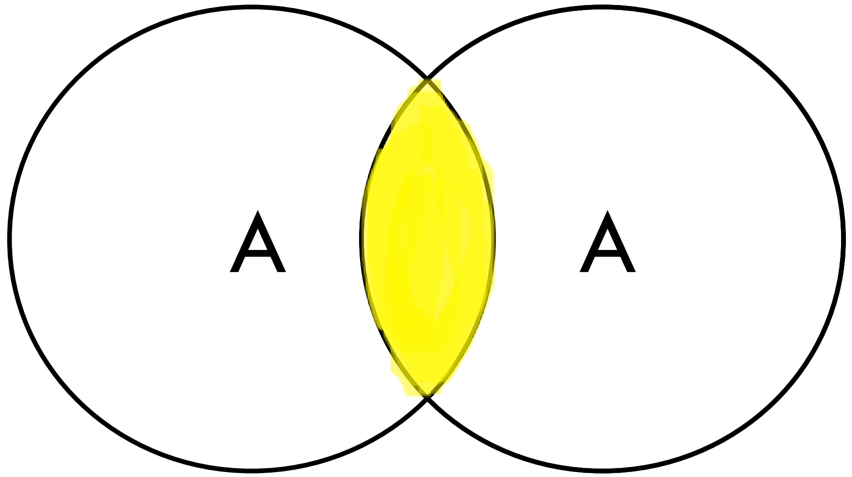
SELECT table1.column1, table2.column2, … FROM table1 **FULL JOIN** table2 **WHERE** table1.column = table2.column;

OR

SELECT table1.column1, table2.column2, … FROM table1 **FULL JOIN** table2 **ON** table1.column = table2.column;

SELF JOIN

- A regular inner join where it joins the table with itself



SELF JOIN

SELECT table1.column1, table2.column2, … FROM table1 **SELF JOIN** table2 **WHERE** table1.column = table2.column;

OR

SELECT table1.column1, table2.column2, … FROM table1 **SELF JOIN** table2 **ON** table1.column = table2.column;

ORDER OF EXECUTION

```
SELECT DISTINCT column, AGG_FUNC(column_or_expression), ….
FROM mytable
    JOIN another_table
      ON mytable.column = another_table.column
    WHERE constraint_expression
    GROUP BY column
    HAVING constraint_expression
    ORDER BY column ASC/DESC
    LIMIT count OFFSET COUNT;
```

Note: Constraint expression would contain and, or, between and other conditional statements.

SUMMARY

- Aggregate functions like SUM, COUNT, AVG, MIN and MAX can be used with queries to perform more analysis on the numerical values of the table
- Using of LIKE keyword can help you match a pattern in the string or numerical values of the table
- JOINS are of majorly 5 types, INNER, LEFT, RIGHT, OUTER/FULL and SELF JOIN to perform analysis on combined rows of 2 tables based on the related column
- There is a particular order of execution of keywords that needs to be followed for every query.