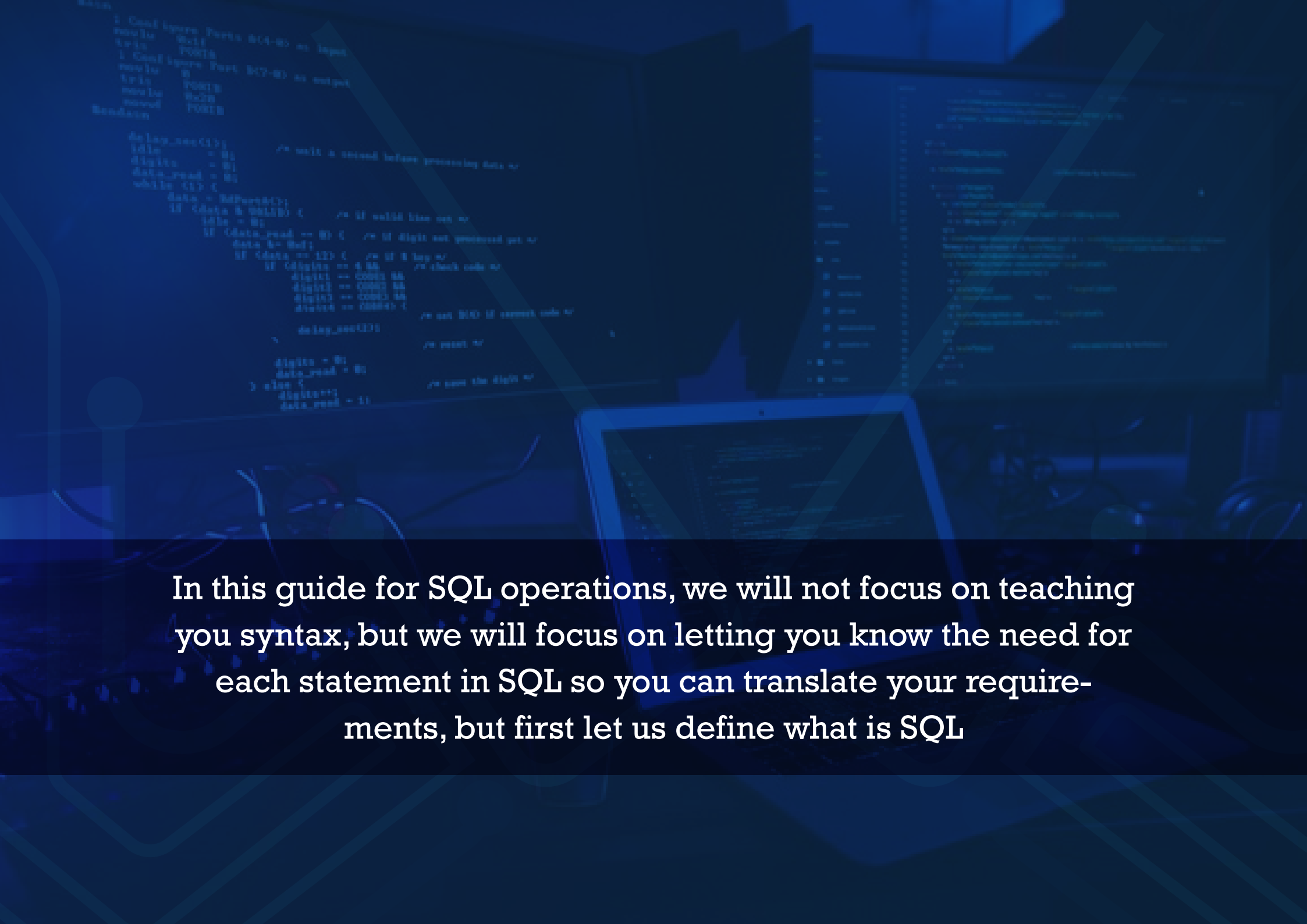




SQL Practical Guide

Learn and write SQL by understanding
SQL query components



In this guide for SQL operations, we will not focus on teaching you syntax, but we will focus on letting you know the need for each statement in SQL so you can translate your requirements, but first let us define what is SQL

What is SQL?

SQL stands for “Structured Query Language”, which is a language that was invented to enable any user from extracting information from structured datasets “Tables”.

Tables consist of two main components: Rows and Columns, each column has his own name, and data type which describes the type of information inside these columns such as string, number, decimals, and so on.

So now we know SQL is a language to help us extract information from Rows and Columns, lets now jump to the basic blocks of any SQL query

DWH_ID	FIRST_NAME	LAST_NAME	DEPARTMENT_ID	SALARY	HIRE_DATE	EMAIL	JOB_TITLE
922941124	Abraham	McKee	2	63833.56	2018-01-04	AbrahamAcevedo@company.com	Assistant Vice President
1410715718	Reinaldo	Eubanks	2	61359.99	2009-01-04	Jude_High169@company.com	Application Development Super...
1346404200	Elijah	Hand	7	59950.92	2016-07-28	Bauer@company.com	Chief Information Officer
1983882757	Tanner	Abney	4	33062.21	1999-01-04	Alarcon@company.com	Building Manager
1916920311	Jewel	Evans	9	55803.28	2009-01-08	gndd9978@company.com	Specialist
256308514	Alfonso	Bowens	9	NULL	2015-05-03	Palmer_Lofton@company.com	Trainer/Consultant
559600483	Jason	Stanley	7	110627.12	2019-08-16	Abernathy@company.com	Service Engineer
1171267201	Rico	Handley	2	68856.60	2011-08-05	Adaline_Abbott@company.com	Technical Writer
748410784	Yasmine	Everett	2	70511.75	2015-01-21	Shayne_Britton589@company.c...	Regional Manager
1714308015	Darcey	Lowry	8	52940.19	2012-07-26	mxwygtmf_ahjpwboxdy@comp...	Branch Manager

Column

Column

Record

Basic Blocks

SELECT ← Which columns you want to see
FROM ← From which tables you need to see the data

SELECT FIRST_NAME, LAST_NAME, HIRE_DATE
FROM EMPLOYEES

Query Translation

Get in the result from table “EMPLOYEES”,
the following columns:

- FIRST_NAME
- LAST_NAME
- HIRE_DATE

So, you can select certain columns
out from all table columns

DWH_ID	FIRST_NAME	LAST_NAME	DEPARTMENT_ID	SALARY	HIRE_DATE	EMAIL	JOB_TITLE
922941124	Abraham	McKee	2	63833.56	2018-01-04	AbrahamAcevedo@company.com	Assistant Vice President
1410715718	Reinaldo	Eubanks	2	61359.99	2009-01-04	Jude_High169@company.com	Application Development Super...
1346404200	Elijah	Hand	7	59950.92	2016-07-28	Bauer@company.com	Chief Information Officer
1983882757	Tanner	Abney	4	33062.21	1999-01-04	Alarcon@company.com	Building Manager
1916920311	Jewel	Evans	9	55803.28	2009-01-08	gndd9978@company.com	Specialist
256308514	Alfonso	Bowens	9	55803.28	2015-05-03	Palmer_Loifton@company.com	Trainer/Consultant
559600483	Jason	Stanley	7	110627.12	2019-08-16	Abernathy@company.com	Service Engineer
1171267201	Rico	Handley	2	68856.60	2011-08-05	Adaline_Abbott@company.com	Technical Writer
748410784	Yasmine	Everett	2	70511.75	2015-01-21	Shayne_Britton589@company.c...	Regional Manager
1714308015	Darcey	Lowry	8	52940.19	2012-07-26	mxxvygtmf_ahjpwboxdy@comp...	Branch Manager
4615735007	Rebecca	McKee	2	55803.28	2018-01-04	RebeccaMcKee@company.com	Technical Features

If you need to see all columns of a table, simply you can replace the column names with “*”

```
SELECT *
```

```
FROM EMPLOYEES
```

I do not like these column names!

What if you do not want this column names to be in your final result or you need to represent these data to someone else who will not be ware of these column names and you need to make It more clear for him.

In this case we can use an “column alias”, which change the column name in the output query

```
SELECT FIRST_NAME "First Name", LAST_NAME As "Last Name", HIRE_DATE "Hire Date", GENDER  
FROM EMPLOYEES
```

To do that we can simply mention the new column name directly in double quotes or use **AS** keyword

First Name	Last Name	Hiring Date	MALE/FEMALE
Saniya	NULL	1994-09-15	M
Sumant	NULL	1985-02-18	F
Duangkaew	NULL	1989-08-24	F
Mary	Sluis	1990-01-22	F
Patricio	Bridgland	1992-12-18	M
Eberhardt	Terkki	1985-10-20	M
Berni	Genin	1987-03-11	M
Guoxiang	Nooteboom	1987-07-02	M
Kazuhito	Cappelletti	1995-01-27	M
Cristinel	Bouloucos	1993-08-03	F

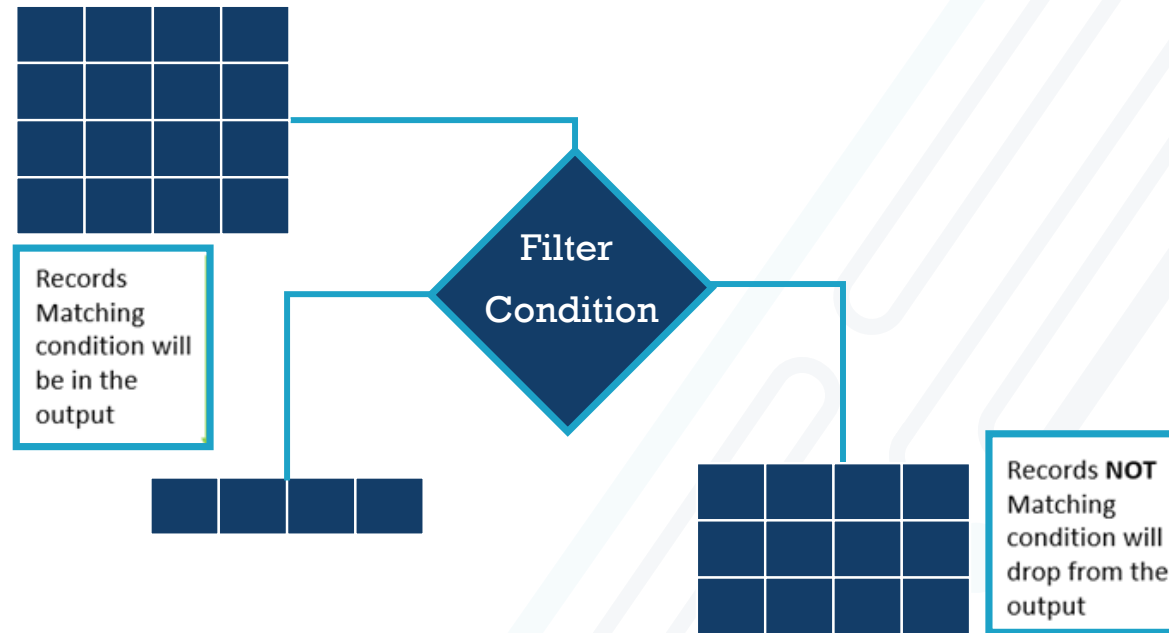
Query Translation

Get from “EMPLOYEES” table the following columns:

- FIRST_NAME → Rename it to “First Name”
- LAST_NAME → Rename it to “Last Name”
- HIRE_DATE → Rename it to “Hire Date”
- GENDER → Rename it to “MALE/FEMALE”

We need to filter this data

In this requirement we don't want to retrieve full data from the table, but we need to get data that matching certain conditions



To filter records based on one or more conditions we add the keyword **"WHERE"** then we add our conditions

```
SELECT *  
FROM EMPLOYEES  
WHERE ANNUAL_SALARY < 50000
```

Filter Condition
<Column Name> <Operator> <Value>

Query Translation

Get all columns from “EMPLOYEES” but only records the matching conditions that ANNUAL_SALARY is less than 50K

You can merge between conditions using logical operators such as **AND**, **OR**, and you can check the opposite of a condition using **NOT** operator, in the following table we summarize the operators

Operator	Description	Example
=	Equal to	FIRS_NAME = 'John'
>	Greater than	SALARY > 1000
<	Less than	AGE < 30
>=	Greater than or equal	AGE >=25
<=	Less than or equal	WORKING_HOURS <=17
<>	Not equal	DEPARTMENT <> 'HR'
BETWEEN ... AND ...	Check that value is within a range	SALARY BETWEEN 115000 AND 120000
IN	Check the column values exist or not in specific values	CITY IN ('NEW-YORK','CAIRO','DUBAI')
IS NULL	Check if column value is Null or not	LAST_NAME IS NULL
NOT	Check if the condition is not met	NOT DEPARTMENT IS NULL Check where DEPARTMENT column has a not NULL value

Examples

Query	Explanation
<pre>SELECT * FROM employees WHERE gender='M';</pre>	Get all columns from Employees table that has a gender equal to "M"
<pre>SELECT FIRST_NAME, DEPARTMENT FROM salaries WHERE SALARY>10000;</pre>	Get First name, and department name for employees who their salary is greater than 10K
<pre>SELECT EMPLOYEE_ID WHERE DEPARTMENT IN ('HR','FINANCE','IT')</pre>	Get Employee ID from employees table where employees department is one of the following (HR, Finance, IT)

We have another kind of operators to combine between multiple conditions such as [AND](#), and [OR](#)

Operator	Description	Example
AND	Merge more than one condition and all of the conditions must be true for example, Condition 1 AND Condition 2 Condition 1 and Condition 2 must be fulfilled	FIRS_NAME = 'John' AND DEPARTMENT_ID = 10
OR	One or both of the conditions can be true to the overall statement will be true For example, Condition 1 OR Condition 2 If Condition 1 or Condition 2 is fulfilled that means the overall condition is fulfilled	HIRE_DATE > '2020-10-1' OR SALARY > 1000

Examples

Query	Explanation
<pre>SELECT * FROM employees WHERE LAST_NAME IS NULL AND gender='F';</pre>	Get all columns from Employees where last name has null value and gender of the employee is Female
<pre>SELECT * FROM EMPLOYEES WHERE (ANNUAL_SALARY BETWEEN 40000 AND 50000) AND HIRE_DATE > '2019-10-01';</pre>	Get all columns from employees tables where annual salary is between 40K and 50K and employee hiring date is after 1st of October 2019

We need data in Specific Order

Now lets add another block to our SQL query to specify the order we need to be in our output results, let's say I need to show the most recent hired employees data or I need to see the orders placed in my store by the value from the bigger value to smaller value. We can achieve that by using **ORDER BY**

```
SELECT  <* or Column names>
FROM    <Table Name>
WHERE   <Condition(s)>

ORDER BY Column Name <ASC or DESC>, ....
```

As we can see from this template to order based on column or a set of columns you simply put the column name after ORDER BY statement and specify the order is ASC (Ascending) or DESC (Descending). You can order by more than one column by placing a comma “,” then add other column with order. What will happen is that the database engine will order based on first column then order by the next column and next column and so on.

Lets take an example of the below employees table

EMP_ID	FIRST_NAME	LAST_NAME	ANNUAL_SALARY	HIRE_DATE	DEPARTMENT
100	Xinglin	Eugenio	60000	2019-02-15	HR
10	JOHN	MARK	10000	2019-02-20	IT
20	JESSY	roberston	15000	2019-02-20	IT
30	JIM	STEVE	30000	2019-02-25	IT
120	Amabile	Gomatam	10000	2019-02-27	HR

We have a requirement to read data per department, then we need to see inside each department employee's data from highest salary to lowest salary, to achieve this output we will write the following query

```
SELECT *  
FROM EMPLOYEES_DB  
ORDER BY DEPARTMENT, ANNUAL_SALARY DESC
```

Output will be

EMP_ID	FIRST_NAME	LAST_NAME	ANNUAL_SALARY	HIRE_DATE	DEPARTMENT
100	Xinglin	Eugenio	60000	2019-02-15	HR
120	Amabile	Gomatam	10000	2019-02-27	HR
30	JIM	STEVE	30000	2019-02-25	IT
20	JESSY	roberston	15000	2019-02-20	IT
10	JOHN	MARK	10000	2019-02-20	IT

Note in case of character column order will be based on alphabetical order, in our case H letter is before I that is why we have HR department comes first, in case we will order Descending the IT department will come first.

By default, the order is Ascending. If you did not specify the order of the column

Summarize your data and get insights (Aggregation)

Now we have another scenario that we need to get information from aggregating or summarizing data from multiple records, in this case we are talking about aggregation functions.

a) Aggregation over all records

Let's say we need to get count of all records in our table, or get the total salaries across all employees, check the following examples

Query	Explanation
<code>SELECT COUNT(*) FROM employees;</code>	Count number of records from employees table
<code>SELECT COUNT(COMMISSION_PCT) FROM employees;</code>	Get count of all available values in COMMISSION_PCT Note: Records with null value in this column will not counted
<code>SELECT SUM(SALARY) FROM employees;</code>	Sum all values in SALARY column to get the total value of salaries for all employees
<code>SELECT AVG(SALARY) FROM employees;</code>	Get average of all values in SALARY column
<code>SELECT STD(SALARY) FROM employees;</code>	Get standard deviation of all values in SALARY column

You can add filters on above queries to select specific set of records based on your condition as we learnt before, and you can meagre between more than one aggregation function to be in the output as following

Number of Employees	Number of Employees with Commission
107	35

b) Aggregate data based on group of columns

We need to get aggregations for our employees table, but we need this to be per department or per any other column, in this case we need to use another block to inform engine which group column we need

```

SELECT <Group by Column names>,<Aggregation functions>
FROM <Table Name>
WHERE <Condition(s)>
GROUP BY Column Name,....

```

DEPARTMENT_ID	Number of Employees
90	3
60	5
100	6
30	6
50	45
80	34

```

SELECT DEPARTMENT_ID, COUNT(*) 'Number of Employees'
FROM employees
GROUP BY DEPARTMENT_ID;

```

In this query we are telling the engine to get number of records since we didn't specify column names and group the output per department ID so we can know number of employees per department

Note

Its not allowed to specify group by columns without GROUP BY statements because it is not logical to tell the engine we need get column values beside aggregated values without telling the engine that we need the columns mentioned to be used as group by for aggregations

```
SELECT DEPARTMENT_ID,  
COUNT(*) 'Number of Employees'  
FROM employees;
```

Error Code: 1140. In aggregated query without GROUP BY, expression #1 of SELECT list contains nonaggregated column 'employees.DEPARTMENT_ID';

c) Filtering Aggregated value

We learned before that we can filter our query output using **WHERE**, but what if we need to filter the aggregated value itself, in this case we will use different statement which is **HAVING**

```
SELECT <Group by Column names>,<Aggregation functions>
FROM <Table Name>
WHERE <Condition(s)>
GROUP BY Column Name,....
HAVING <Aggregation function> <Operator> <Value> , .....
```

Lets say we need to check departments where total number of employees is greater than 30, in this case we will write the same query but we will add our filter using **HAVING** as following

```
SELECT DEPARTMENT_ID, COUNT(*) 'Number of Employees'
FROM employees
GROUP BY DEPARTMENT_ID
HAVING COUNT(*) > 30;
```

DEPARTMENT_ID	Number of Employees
50	45
80	34

You can merge multiple conditions also in **HAVING** as we did before in **WHERE** clause

In our case let's say we need also to know departments which have over 30 employees and total salaries for this department are over 300K

```
SELECT DEPARTMENT_ID, COUNT(*) 'Number of Employees'  
FROM employees  
GROUP BY DEPARTMENT_ID  
HAVING COUNT(*) > 30 AND SUM(SALARY) > 300000;
```

DEPARTMENT_ID	Number of Employees
80	34

Note

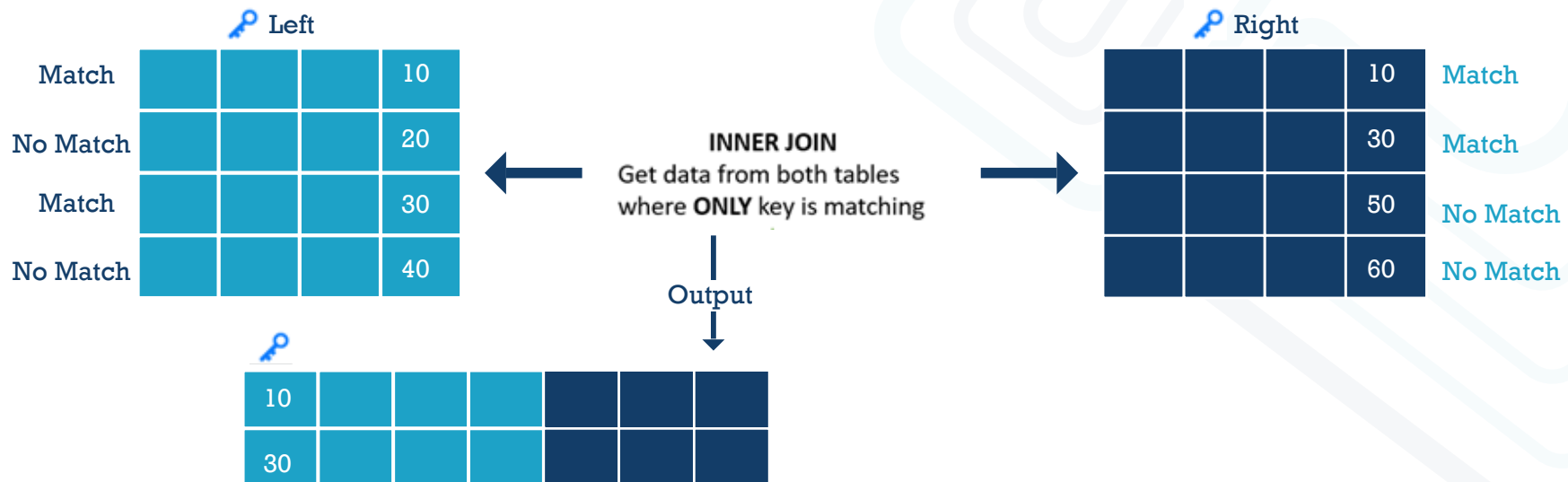
here we didn't specify SUM(SALARY) to be part of the output columns, we included it in the condition only and this is totally fine.

Get data from more than one table

In most of the cases, you do not have your data in one table, you have more than tables in some cases hundreds of tables, to do that we need to join data from multiple tables and to do that we will use JOIN and its different types. We have different types of join between two tables depends on your requirement you select the proper type. To join between two tables, you need to identify a Key, which is a column or a set of columns that links between the information in the two tables, for example, to join between EMPLOYEES table which has employees information and DEPARTMENT table which has department information you may join based on department ID.

Inner Join

In this type get all matching and only matching data records from both tables that has the same key values.



SELECT Table A columns, Table B columns

FROM Table A **INNER JOIN** Table B **ON** Table A key = Table B key

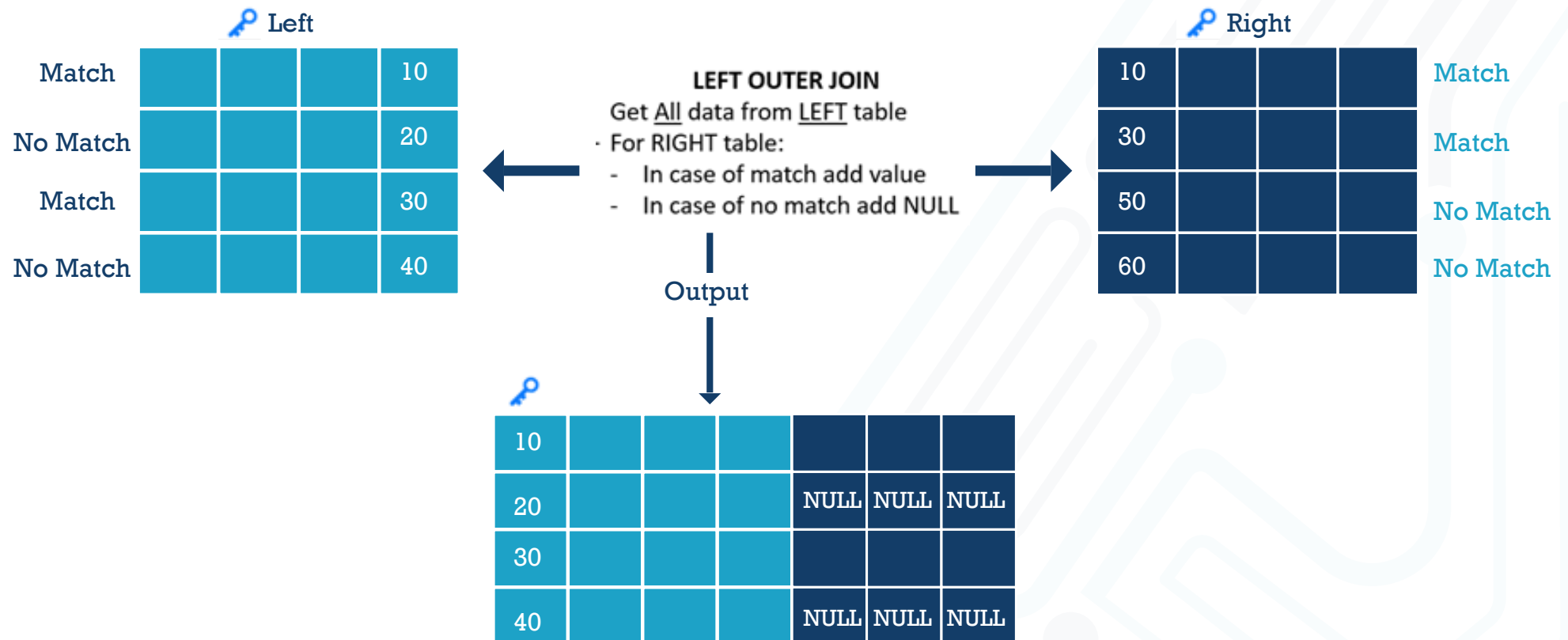
You can give table alias the same as we did for columns before, and use this alias to refer to any column from that table as we will see in the following example of the inner join

```
SELECT E.FIRST_NAME, E.LAST_NAME, D.DEPARTMENT_NAME  
FROM employees E INNER JOIN departments D ON E.DEPARTMENT_ID = D.DEPARTMENT_ID ;
```

FIRST_NAME	LAST_NAME	DEPARTMENT_NAME
Steven	King	Executive
Neena	Kochhar	Executive
Lex	De Haan	Executive
Alexander	Hunold	IT

Left Outer Join

Get all data from the Left side table regardless there is a matching key or not, for the Left side, if there are matching records bring its value, in case no matching values



SELECT Table A columns, Table B columns
FROM Table A **LEFT OUTER JOIN** Table B
ON Table A key = Table B key

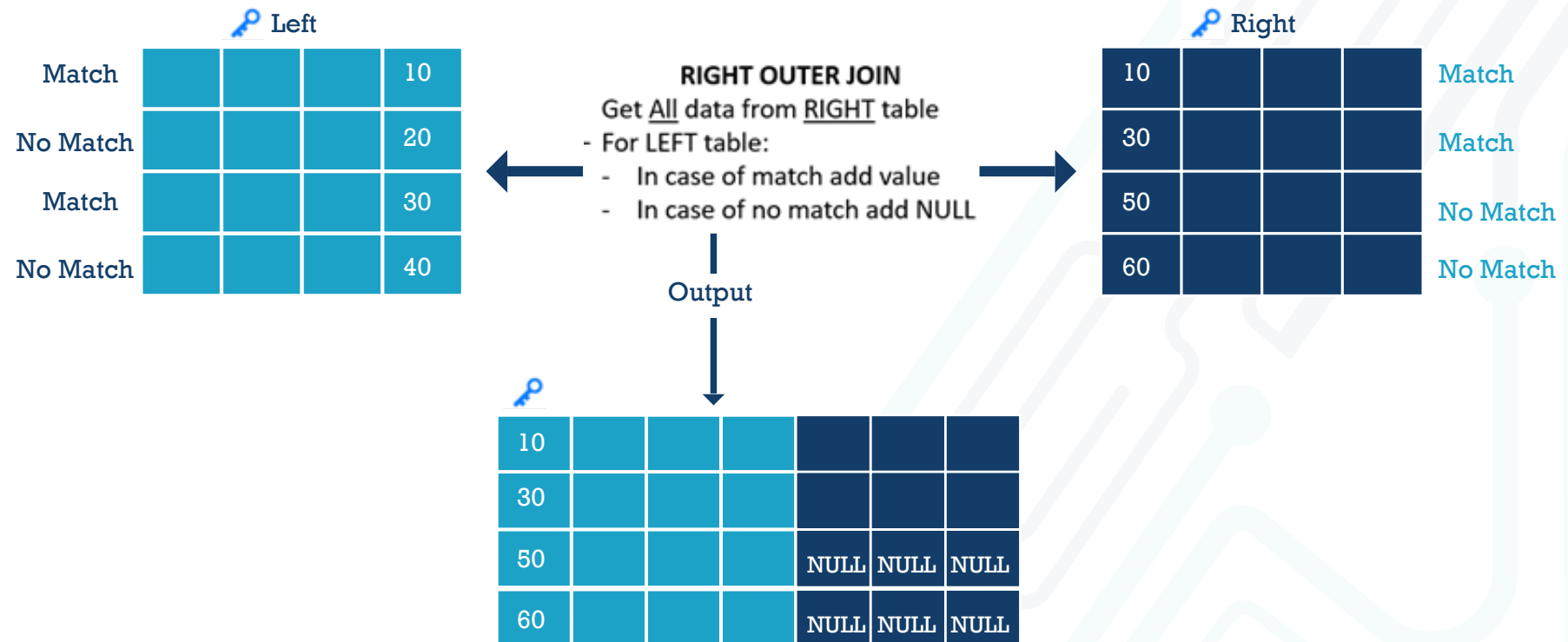
```
SELECT E.FIRST_NAME, E.LAST_NAME, J.JOB_TITLE  
FROM employees E LEFT OUTER JOIN JOBS_2 J ON E.JOB_ID = J.JOB_ID ;
```

Here we will get all records from employees table and try to match with records from jobs table using JOB_ID to get the job description, as we are using left outer join and, in this case, left table is Employees, so in case no match we will get NULL in job description as we described before.

FIRST_NAME	LAST_NAME	JOB_TITLE
Michael	Hartstein	Marketing Manager
Pat	Fay	Marketing Representative
Susan	Mavris	Human Resources Repres...
Hermann	Baer	Public Relations Repre...
Steven	King	NULL
Neena	Kochhar	NULL
Lex	De Haan	NULL

Right Outer Join

Get all data from the Right-side table regardless there is a matching key or not, for the Left side, if there is matching records bring its value, in case no matching values.



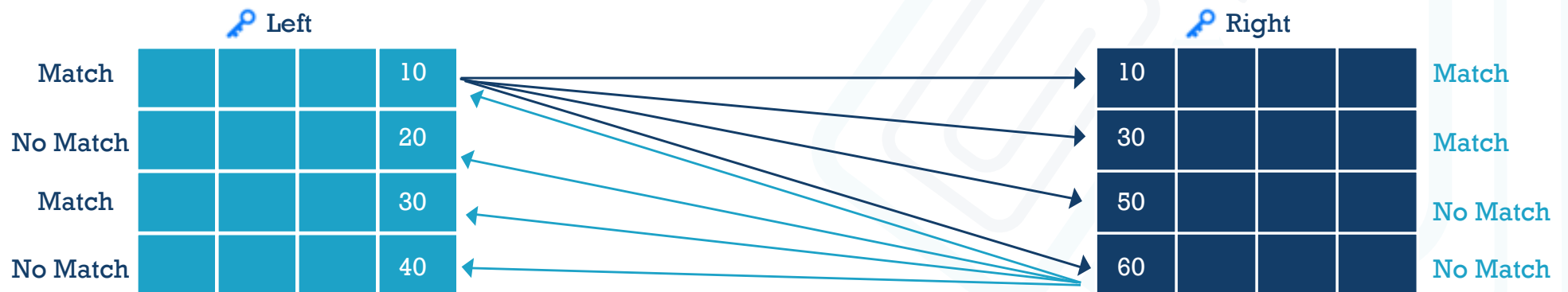
SELECT Table A columns, Table B columns

FROM Table A **RIGHT OUTER JOIN** Table B

ON Table A key = Table B key

CROSS Join

You will get all the possible combinations between the two tables for example if you have 10 records in both tables, you will get 100 records in the output, TAKE CARE and be very careful when using this type as it is very costly in terms of processing and resources usage on your engine especially if you have big number of records.



CROSS JOIN

In the output you will have all the possible combinations between the two table

Summary

In this quick guide we described the main building blocks of SQL query and when you will need each statement and how you will use it in your query, following is summary of what we had discussed in this quick guide:

- Select columns from table – SELECT
- Rename output columns – Column Alias
- Filter output results – WHERE
- Order the results by columns – ORDER BY
- Aggregate the data and group aggregated data by specific columns – Aggregation functions
- How to join between multiple tables – Join (INNER, LEFT, RIGHT, CROSS)

We would like to hear from you if you have any enhancement or recommendations to make this guide better or enrich it please email us on wecare@datavalley.technologies