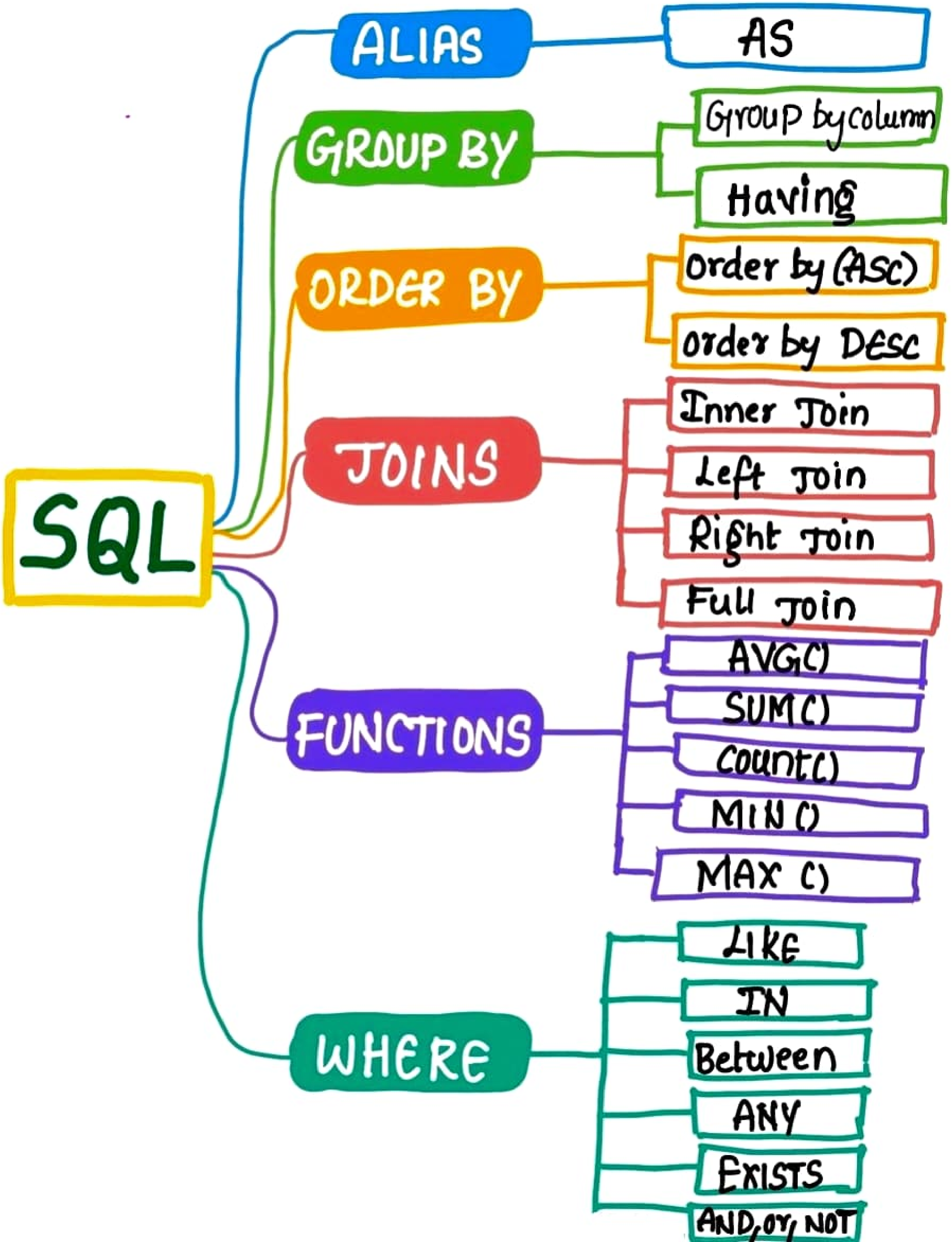
SQL ZERO TO HERO!



ALIAS — AS

Example:

An Alias is created with 'As' Keyword.

```
SELECT column_name AS alias_name
FROM   table_name;
```

GROUP BY

⇒ GROUP BY COLUMN

Example:

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
ORDER BY column_name(s);
```

⇒ Having

Example:

```
SELECT column name(s)
FROM table name
WHERE condition
GROUP BY column_name(s)
HAVING condition
ORDER BY column_name(s);
```

ORDER BY

⇒ ORDER BY (ASC) or ORDER BY DES

The order by keyword is used to sort the result-set in ascending or descending order

Example:

```
SELECT column1, Column2, .....
FROM table_name
ORDER BY column1, column2,..... ASC/DES;
```

JOINS

⇒ INNER JOIN

This keyword selects that have matching values in both tables

Example:
```
SELECT column_name(s)
FROM table1
INNERJOIN table2
ON table.column_name=table2.column_name;
```

⇒ LEFT JOIN

Example:
```
SELECT column_name (s)
FROM table1
LEFT JOIN table2
```

⇒ **RIGHT JOIN**

Example:
```
SELECT column_name(s)
FROM table1
RIGHT JOIN table2
ON table1.column_name = table2.column_name;
```

⇒ **FULL JOIN**

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

FUNCTIONS

⇒ **AVG ()**

Example:
```
SELECT AVG(column_name)
FROM table_name
WHERE condition;
```

⇒ **SUM ()**

Example:
```
SELECT SUM(column_name)
FROM table_name
WHERE condition;
```

⇒ **COUNT ()**

Example:
```
SELECT (Product ID)
FROM Products;
```

⇒ **MIN ()**

Example :

```
SELECT MIN (column_name)
FROM table_name
WHERE condition;
```

→ **MAX ()**

Example :

```
SELECT MAX(column_name)
FROM table_name
WHERE condition;
```

WHERE

⇒ **LIKE**

Example :

```
SELECT column1, coum2,....
FROM table_name
WHERE columnN LIKE Pattern
```

⇒ **IN**

Example :

```
SELECT column_name(s)
FROM table_name
WHERE column_name IN
    (value1, value2,....);
```

⇒ **BETWEEN**

Example :

```
SELECT column_name
FROM table_name
WHERE column_name BETWEEN
    value1 AND value2;
```

⇒ **ANY**

Example :

```
SELECT column_name (s)
FROM table_name
WHERE column_name
            operator ANY
(SELECT column_name
 FROM table_name
```

⇒ **Exists**

Example:

```
SELECT column_name(s)
FROM table_name
WHERE EXISTS
SELECT column_name FROM table_name
                       WHERE condition);
```

⇒ **ALL**

Example:

```
SELECT ALL column_name(s)
FROM table_name
WHERE condition;
```

⇒ **AND**

Example:

```
SELECT column1, column2,....
FROM table_name
WHERE condition1 AND condition2 AND condition3...;
```

⇒ **OR**

Example:

```
SELECT column1, column2,....
FROM table_name
WHERE condion1 or condition2 or condition3....;
```

⇒ **NOT**

Example:

```
SELECT column1, column2,....
FROM table_name
WHERE NOT condition;
```

Top 10 SQL Commands (must Know)

Case When:

It allows you to write complex conditional statements If you want to allocate a certain value or class depen--ding on other variables. Less commonly known, It also allows you to pivot data.

Select Distinct:

`SELECT DISTINCT` is something that you should always have at the back of your head. It extremely common to use `SELECT DISTINCT` statements with aggregate functions (which is #3)

Example:

```
SELECT
    COUNT (order_id) / COUNT (DISTINCT customer_id) as
    Orders_Per_cust
    FROM
    customer_orders
```

Aggregate functions:

Related to Point #2, you should have strong understanding of aggregate functions like min, max, sum, count, etc. This also means that you have a strong understanding of the GROUP BY and HAVING clause.

Example:

Id	Email
1	a@b.com
2	c@d.com
3	a@b.com

Answer

```
SELECT
    Email
FROM
    Person
GROUP By
    Email
HAVING
    COUNT(Email) > 1
```

Left joint Vs Inner Joint

For those who are relatively new to SQL or have not used it in a while, It can be easy to mix up left joints and inner joints. Make sure you clearly understand, How each joints derives different results.

Self Joins :

A SQL self-join joins a table with itself. you might think that serves no purpose, But you'd be surprised at how common this is. In many real life settings, Data is stored in one large table rather than many smaller tables. In such cases, self-joins may be required to solve unique problems.

Example :

Id	Name	Salary	Manager Id
1	Joe	70000	3
2	Henry	80000	4
3	Sam	60000	NULL
4	Max	90000	NULL

Answer

```
SELECT
    a.Name as Employee
FROM
    Employee as a
     JOIN Employee as b on a.Manager ID= b.Id
WHERE  a.Salary > b.Salary
```

Sub queries:

A Sub query, is also known as an inner query or a nested query, is a query with in a query and is embedded in the WHERE clause. This is a great way to solve unique problems that require multiple queries in sequence in order produce a given outcome. Sub queries and WITH AS Statements are both extremely using when querying so you should absolutely make sure that you know how to use them.

Example:

Table: customers.

Id	Name
1	Joe
2	Henry
3	Sam
4	max

Table: orders

Id	customerId
1	3
2	1

Answer

```
SELECT
  Name as customers
FROM
  customers
WHERE
  Id NOT IN C
  SELECT
  CustomerId
  FROM Orders
  )
```

String Formatting:

String functions are important especially when working with data that isn't clean. Thus, companies may test you on String formatting and manipulation to make sure that you know how to manipulate data.

String formatting includes things like:

- LEFT, RIGHT
- TRIM
- POSITION
- SUBSTR
- CONCAT
- UPPER, LOWER
- COALESCE

Date-Time Manipulation:

you should definitely expect some sort of SQL Questions that involves date-time data. For example you may be required to group data by months or convert a variable format from DD-MM-YYYY to simply the month.

Some functions you should know are

- EXTRACT
- DATEDIFF

Example :

Id (INT)	Record Date (DATE)	Temperature (INT)
1	2015-01-01	10
2	2015-01-02	25
3	2015-01-03	20
4	2015-01-04	30

Answer:

```
SELECT
    a.Id
 FROM
    Weather a,
    weather b
  WHERE
     a.Temperature > b.Temperature
   AND DATEDIFF(a.Record Date, b.Record Date) = 1
```

Window Functions:

Window function allow you to perform an aggregate value on all rows, instead of return only one row (which is what a GROUP BY statement does). Its extremely useful if you want to rank rows, calculate cumulative sums, and more.

Example

depname	empno	salary
develop	11	5200
develop	7	4200
develop	9	4500
develop	8	6000
develop	10	5200
Personnel	5	3500
Personnel	2	3900
Sales	3	4800
Sales	1	5000
Sales	4	4800

Answers

```
WITH Sal_rank AS
  (SELECT
   empno,
   RANK() OVER (ORDER BY salary DESC) rnk
```

↓ continues

```
    FROM
      Salaries)
    SELECT
      empno
    FROM
      sal_rank
    WHERE
       rnk = 1;
```

Union:

As a bonus, #10 is UNION! while it doesn't come up often, you'll be asked about this the odd time and its good to know in general. If you have two tables with the same columns and you want to combine them, this is when you'd use UNION.