Topics Covered

1) SQL
2) Nomenclature
3) Data Integrity
4) Data Normalization
5) Data types
6) Types of SQL commands
7) DDL (Data Definition Language)
8) DML (Data Manipulation Language)
9) DQL (Data Query language)
10) WHERE Clause
11) Operators
    - Comparision
    - Arithmetic
    - Bitwise
    - Compound
    - Logical
    - Like, IN, Between

Introduction to SQL:

SQL is a standard language for accessing and manipulating databases.

what is SQL?

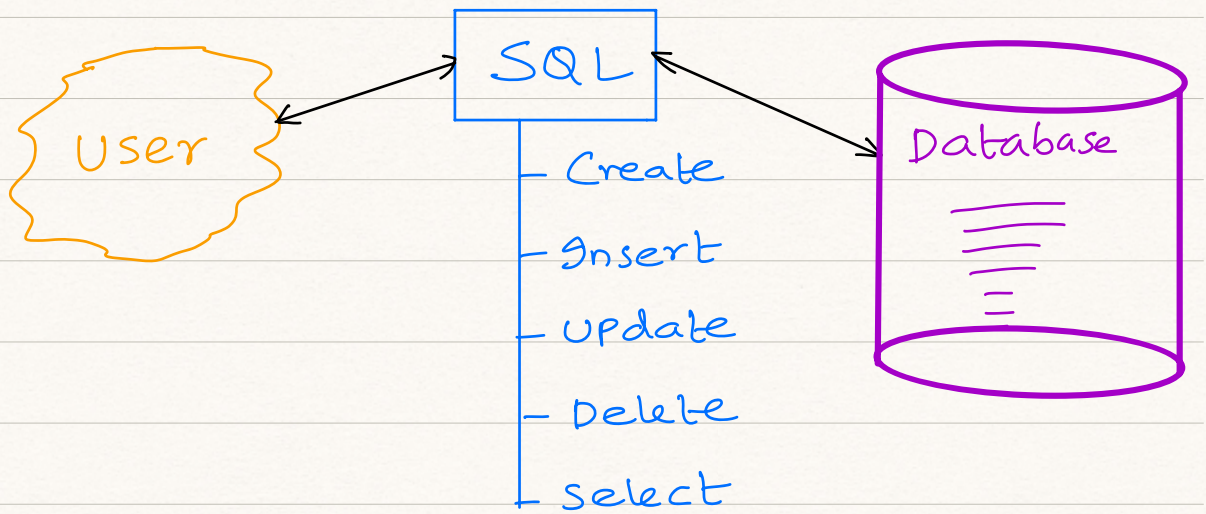
- SQL stands for structured Query language.
- SQL lets you access and manipulate databases.
- SQL is an ANSI (American National standards institute) standard.

what can SQL do?

- SQL can execute queries against a database
- SQL can retreive data from database
- SQL can insert records in a database
- SQL can update records in a database
- SQL can delete records from a database.
- SQL can create new databases
- SQL can create new tables in a database
- SQL can create stored procedures in a database
- SQL can create views in a Database

- SQL can set permissions on tables, procedures and views.

* SQL is standard, but there are different versions of SQL language. However, to be compliant they all support major commands.



* SQL is not case-sensitive language.

what is RDBMS?

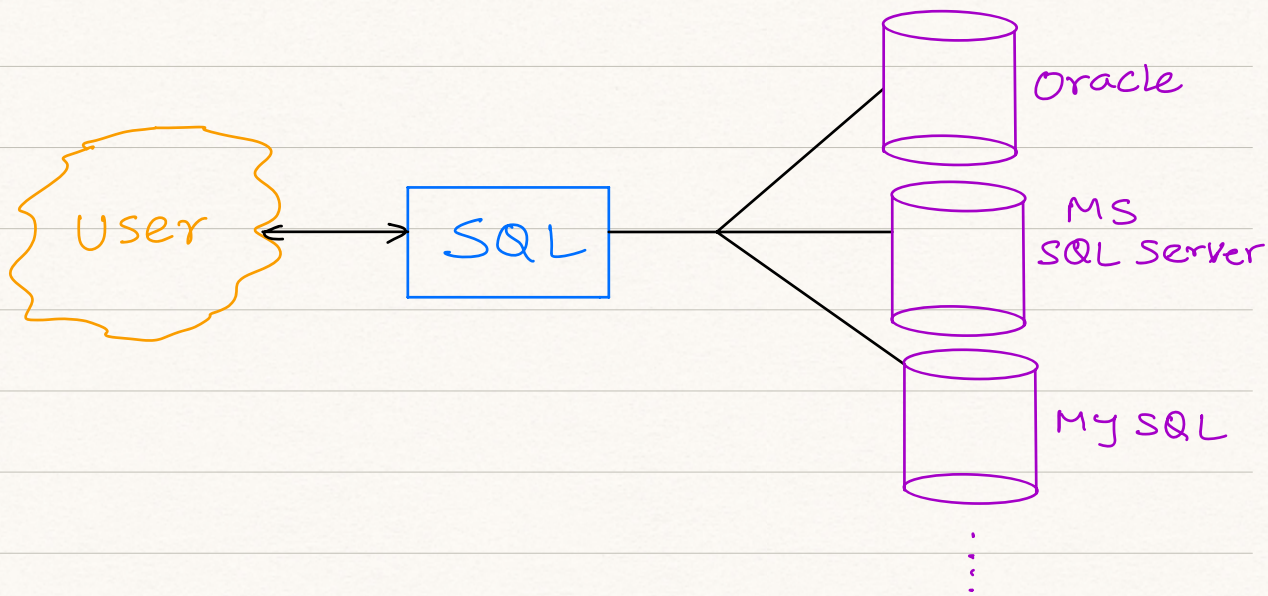
RDBMS - Relational Database Management System

- RDBMS is the basis for SQL, and for all modern database systems like MS SQL Server,

IBM DB2, Oracle, My SQL, and Microsoft Access.

- RDBMS is a database management system (DBMS) based on the relational model as introduced by E.F. Codd.

*SQL is the only language that can communicate with any RDBMS product.



Attributes of RDBMS:

what is a table?

The data in a RDBMS is stored in database objects which are called tables. This table is basically a collection of related data entries and it consists of numerous columns and rows.

- A table is the most common and simplest form of data storage in a relational database..

ex:-

Customers table

ID	NAME	AGE	ADDRESS	SALARY
-	-	-	-	-
-	-	-	-	-
-	-	-	-	-
-	-	-	-	-
-	-	-	-	-

what is a field?

Every table is broken into smaller entities called fields. The fields in the Customers table consist of ID, NAME, AGE, ADDRESS, SALARY. A **field** is a column in a table that is designed to maintain specific information about every record in the table.

what is a Record or Row?

A **Record** is also called as a row of data and is each individual entry that exists

in a table.

A record is a horizontal entity in a table.

ex:- From Customer's table.

1	Ramesh	32	Ahmedabad	2000.0

what is a column?

A column is a vertical entity in a table that contains all information associated with a specific field in a table

ex:- From Customers table

ADDRESS
Ahmedabad
Delhi
Kolkata
Hyderabad
Vizag
chennai

Data Integrity:

The following categories of data integrity exist with each RDBMS:

- **Entity Integrity:** There are no duplicate rows in a table.
- **Domain Integrity:** Enforces valid entries for a given column by restricting the type, the format, or the range of values.
- **Referential Integrity:** Rows cannot be deleted which are used by other records.
- **User defined Integrity:** Enforces some specific business rules that do not fall in to entity, domain or referential integrity.

Database Normalization:

- Database Normalization is the process of efficiently organizing data in a database.
- There are two reasons for this normalization process
    1) Eliminating redundant data, for example storing the same data in more than one table.
    2) Ensuring data dependencies makes sense.
- Normalization reduces the amount of space a database consumes.
- It also ensures the data is logically stored.
- Normalization consists a series of guidelines that help us in creating a good database structure.
- Normalization guidelines are divided in to Normal forms.
- form is the way a database is layed out.
- The aim of normal forms is to organize the database structure so that it compiles with the rules of first normal form, then second normal form and finally the third

normal form.

- It is our choice to take it further and go to fourth normal form, fifth normal form and so on, but in general third normal form is more than enough.

  * First Normal Form (1NF)
  * Second Normal Form (2NF)
  * Third Normal Form (3NF)

Data types in SQL: There are different types of databases in SQL that we would discuss briefly.

Numeric data types in SQL:

Data Type	From	TO
bigint	-9,223,372,036,854,775,808	9,223,372,036,854,775,807
int	-2,147,483,648	2,147,483,647
small int	-32,768	32,767
tiny int	0	255
bit	0	1
decimal	$-10^{38}+1$	$10^{38}-1$
money	-922,337,203,685,477.5808	922,337,203,685,477.5807
float	-1.79E+308	1.79E+308

Date and Time Data Types:

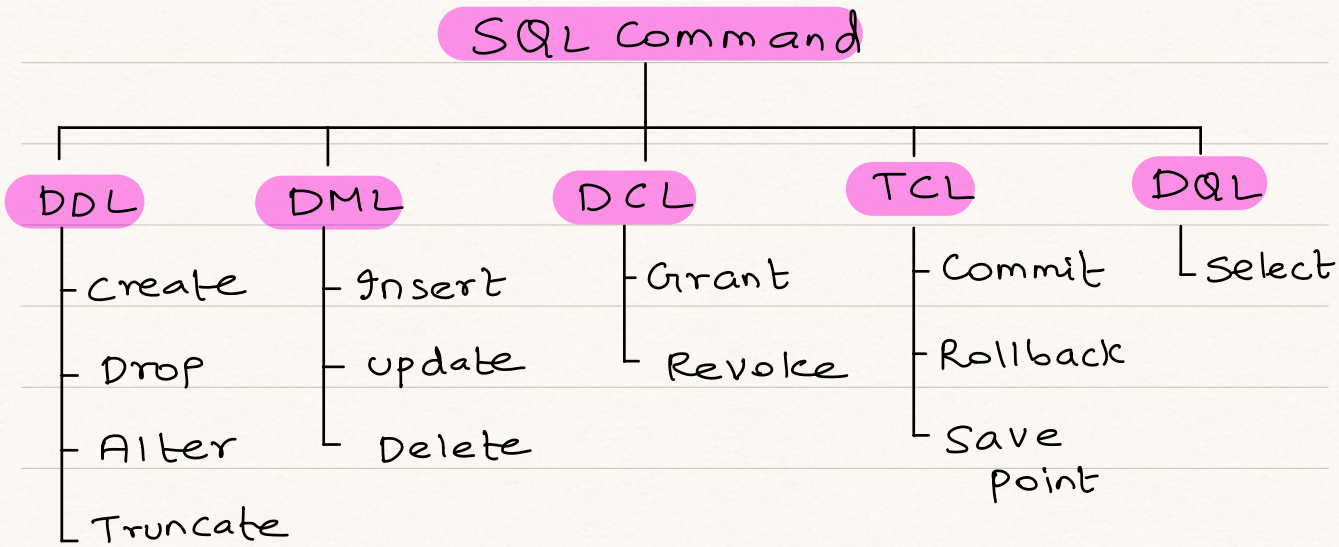
Data Type	From	TO
datetime	Jan 1, 1753 with time	Dec 31, 9999 with time
small datetime	Jan 1, 1990	June 6, 2079
Date	Jan 1, 1753	Dec 31, 9999

Character Strings Data Types:

- **char** : Maximum length of 8,000 characters (Fixed length non-unicode characters)
- **Varchar** : Maximum of 8,000 characters (variable length non-unicode characters)
- **Varchar(max)**: Maximum length of 2E+31 characters, variable length non-unicode characters (SQL server 2005 only)
- **text** : Maximum length of 2,147,483,647 characters, (Variable length non-unicode characters)

Commands in SQL

- SQL commands are instructions. It is used to communicate with the database. It is also used to perform specific tasks, functions, and queries of data.
- SQL can perform various tasks like create a table, add data to tables, drop the table, modify the table, set permissions for users.



I) Data Definition Language (DDL):

- This language commands are used to define, modify or drop an object or database from SQL Server.
- All commands of DDL are auto-commited,

which means it permanently saves all the changes in the database.

* **Create :** Creating a new database or new table in SQL server.

Step 1 : **Create a new database in SQL server.**

Syntax: Create database <DB_NAME>;

Example: Create database Innomatics;

Step 2: Select the required database from SQL server

Syntax: USE <DB_NAME>;

Example: USE Innomatics;

Step 3: **Create new table in database**

Syntax: Create table <Table_name> (
<column1_name> <datatype> (size),
<column2_name> <datatype> (size),
:
1000 columns);

Example: Create table Customers(id int,
Firstname varchar(50), Lastname varchar(50),
Address text, City varchar(50));

Step4: To view the structure of the table

Syntax: method1: Describe/Desc <Table_name>;

method2: SHOW COLUMNS From <Table_name>;

Example: Describe Customers;

(or)

show columns from customers;

* **ALTER:** TO change or modify the structure of a table or a database.

By using the Alter command we can perform the following three operations on existing table.

1) ALTER TABLE – ADD Column
2) ALTER TABLE – DROP Column
3) ALTER TABLE – MODIFY COLUMN

ALTER TABLE is also used to add or drop various constraints on an existing table.

1) ALTER TABLE – ADD Column

Syntax: ALTER TABLE <table_name>
ADD <Column_name> <datatype>;

Example: ALTER TABLE Customers

ADD Email varchar(255);

2) ALTER TABLE - DROP COLUMN

Syntax: ALTER TABLE <table_name>
DROP COLUMN <column_name>

Example: ALTER TABLE Customers
DROP COLUMN Email;

3) ALTER TABLE - MODIFY COLUMN

Syntax: ALTER TABLE <table_name>
MODIFY COLUMN <column_name>
<datatype>;

Example: ALTER TABLE Customers
MODIFY COLUMN Address varchar(255);

* **TRUNCATE**: Deleting rows from the table, but not the structure of the table.

By using Truncate we cannot delete a specific row from the table because it does not support 'where' clause condition.

Syntax: TRUNCATE table <table_name>

Example: TRUNCATE table Customers

* **DROP**: Dropping a table from a database permanently.

Dropping a table needs to proceeded with caution as it will result in deleting the table and all the information stored in the table.

Syntax: DROP table <table_name>;

Example: DROP table customers;

II) Data Manipulation Language (DML):

These language commands are used to change or manipulate data in the database table.

* **INSERT**: The INSERT INTO statement is used to insert new records in a table.

It is possible to write the INSERT INTO statement in two ways:

1) Explicit method:

specify both the column names and the values to be inserted:

Syntax: INSERT INTO <table_name>(column1, column2, ...)

VALUES (value1, value2, ...);

Example: INSERT INTO customers (ID, firstname, lastname, Address, city)
VALUES (G, Bhupathiraju, Sobhadra, Kukatpally, Hyderabad);

2) Implicit method:

If we are adding values for all the columns of the table, we do not need to specify the column names in the SQL query. However, we need to make sure the order of the values is same as the columns in the table.

Syntax: INSERT INTO <table_name>
VALUES (value1, value2, ...);

Example: INSERT INTO Customers
VALUES (G, Sobhadra, Bhupathiraju, Kukatpally, Hyderabad);

* **UPDATE**: Updating all records in a table at a time or a specific record in a table by using 'where' condition.

If we do not mention 'where' condition then all the records in the table will get updated.

Syntax: UPDATE <table-name>
SET Column1 = Value1, Column2 = Value2, ...
WHERE <condition>;

Example: UPDATE Customers
SET firstname = 'Shuba'
WHERE ID = 6;

* **DELETE**: Deleting all the rows from the table at a time or a specific record by using the 'where' condition.

If we do not mention 'where' condition then all the records in the table will get updated.

Syntax: DELETE FROM <table-name>
WHERE <condition>;

Example: DELETE FROM Customers
WHERE ID = 6;

Differences between DELETE & TRUNCATE

	DELETE	TRUNCATE
1)	It is a DML operation	It is a DDL operation
2)	It can delete a specific record from the table	It cannot delete one specific record from the table
3)	It supports the 'where' condition.	It does not support 'where' condition.
4)	It is temporary data deletion	It is a permanent data deletion.
5)	we can restore the deleted data using roll back.	we cannot restore the deleted data by using roll back.
6)	Execution speed is slow	Execution speed is Fast.

DCL & TCL commands we will not discuss in detail here.

III) Data Control Language (DCL):

These commands deal with the rights, permissions, and other controls of the database.

* **GRANT**: This command gives user's access privileges to the database.
* **REVOKE**: This command withdraws the user's access privileges given by using the GRANT command.

IV) Transaction control language (TCL)

Transactions group a set of tasks in to a single execution unit. Each transaction begins with a specific task and ends when all the tasks in the group successfully complete. If any of the tasks fail, the transaction fails. Therefore, a transaction has only two results: success or failure.

* **COMMIT**: Commits a Transaction

  Syntax: `COMMIT;`

* **ROLLBACK**: Rollsback a transaction in case of any errors

Syntax: ROLLBACK;

* **SAVEPOINT**: Sets a savepoint within a transaction.

Syntax: SAVEPOIN <savepoint_name>;

V) Data Query Language (DQL):

These language commands are used to get some data from the table based on the query passed to it, and imposing order on it.

* **SELECT**: The SELECT statement is used to select data from a database. The data returned is stored in a result table, called the result_set.

Syntax: SELECT column1, column2,.....
FROM <table_name>; → for selected columns

(or)

SELECT * FROM <table_name>;
→ for all columns

Example: SELECT Firstname
FROM Customers;

(or)

SELECT * FROM Customers;

* **SELECT DISTINCT**: The SELECT DISTINCT statement is used to return only distinct (different or unique) values.

Inside a table, a column often contains many duplicate values; But sometimes we only want the distinct values.

Syntax: SELECT DISTINCT column1, column2,....
FROM <table_name>;

Example: SELECT DISTINCT Firstname
FROM Customers;

SQL WHERE clause:

The WHERE clause is used to filter records. i.e it is used to extract only the records that fulfill a specific condition.

Syntax: SELECT column1, column2, ...
FROM <table_name>
WHERE <condition>;

* WHERE clause can be used in SELECT, UPDATE, DELETE statements etc.
* Please note comparision is case-sensitive.

Example: SELECT Firstname
FROM Customers
WHERE ID>3;

Text fields vs Numerical fields:

* SQL requires quotes around text values.
* Numerical values should not be enclosed in quotes

Example: ID>3 or Lastname = 'Bhupathiraju'

Operators in SQL:

My SQL comparision operators:

Operator	Description
=	Equal to
>	Greater than
<	Less than
<=	Less than equal to
>=	Greater than equal to
<>	Not equal to

Few Examples:

```
SELECT *
FROM Customers
WHERE ID = 1;
```

```
SELECT *
FROM Customers
WHERE ID < 6;
```

```
SELECT *
FROM Customers
WHERE ID > 1;
```

```
SELECT *
FROM Customers
WHERE ID <> 1;
```

MYSQL Arithmetic Operators :

Operator	Description
+	Add
-	Subtract
*	multiply
/	Divide
%	Modulo

Few Examples :

```
SELECT Customer_name, Opening_amt,
    receive_amt, (opening_amt + receive_amt)
FROM Customers
WHERE (opening_amt+ receive_amt) >15000;
```

```
SELECT Customer_name, Opening_amt,
    Payment_amt, outstanding_amt
FROM Customers
WHERE (outstanding_amt - payment_amt)
    = receive_amt;
```

MySQL Bitwise Operators: on Binary values

Operator	Description
&	Bitwise AND
|	Bitwise OR
^	Bitwise exclusive OR
~	Bitwise NOT
<<	Bitwise left shift
>>	Bitwise right shift

Few Examples :

SELECT 12 & 9 ; -- Result 8

-- Binary representation 1100 & 1001 = 1000

SELECT 12 | 9 ; -- Result 13

-- Binary representation 1100 | 1001 = 1101

SELECT 12 ^ 9 ; -- Result 5

-- Binary representation 1100 ^ 1001 = 0101

SELECT ~12 ; -- Result -13

-- Binary representation ~1100 = -1101

SELECT 12<<2; -- Result 48

-- Binary representation 1100 << 2 = 110000

SELECT 12>>2; -- Result 3

-- Binary representation 1100 >> 2 = 0011

My SQL Compound Operators:

Operator	Description
+=	Add equals
-=	Subtract equals
*=	multiply equals
/=	Divide equals
%=	Modulo equals
&=	Bitwise AND equals
^=	Bitwise exclusive equals
|=	Bitwise OR equals

Few Examples:

```
SET x = 5;
SET x += 3; -- Equivalent to: SET x = x+3;
-- Result: x = 8

SET x = 10
SET x -= 4; -- Equivalent to: SET x = x-4;
-- Result: x = 6

SET x = 3;
SET x *= 4; -- Equivalent to: SET x = x *4;
-- Result: x = 12

SET x = 20;
SET x /= 5; -- Equivalent to: SET x = x/5;
-- Result: x = 4

SET x = 15;
SET x %= 7; -- Equivalent to: SET x = x %7;
-- Result: x = 1
```

```
SET X=12
SET X&=9 -- Equivalent to: SET X=X&9
 -- Result: X=8

SET X=12
SET X|=9; -- Equivalent to: SET X=X|9
 -- Result: X= 13

SET X=12;
SET X^=9; -- Equivalent to: SET X=X^9
  -- Result: X= 5
```

My SQL Logical Operators:

Operator	Description
ALL	TRUE if all of Subquery values meet the condition
AND	TRUE if all conditions seperated by AND are TRUE
ANY	TRUE if any of the subquery values meet the condition
BETWEEN	TRUE if the operand is within the range of comparision
EXISTS	TRUE if the subquery returns one or more records
IN	TRUE if the operand is equal to one of the list of expressions
LIKE	TRUE if the operand matches a pattern
NOT	Displays a record if the condition(s) is NOT TRUE
OR	TRUE if any of the conditions seperated by OR is TRUE
SOME	TRUE if any of the subquery values meet the condition

we will see a little more of AND, OR and NOT operators which are commonly used.

AND Syntax

```
SELECT Column1, Column2,...
FROM <table_name>
WHERE Cond1 AND Cond2 And ....;
```

OR Syntax

```
SELECT Column1, Column2,...
FROM <table_name>
WHERE Cond1 OR Cond2 OR ....;
```

NOT Syntax

```
SELECT Column1, Column2,...
FROM <table_name>
WHERE NOT Condition;
```

* we can also combine AND, OR and NOT
* Always better to use parenthesis to form complex expressions.

For the purpose of Examples let us consider a table customers with below table structure:

```
CREATE TABLE Customers(
        Id INT,
        name VARCHAR(50),
        country VARCHAR(50),
        city VARCHAR(50)
        );
```

AND Example:

```
SELECT *
FROM Customers
WHERE Country = 'India' AND City='Hyderabad';
```

OR Example:

```
SELECT *
FROM Customers
WHERE City = 'Chennai' OR City='Hyderabad';
```

NOT Example:

```
SELECT *
FROM Customers
WHERE NOT Country = 'India';
```

Combination Example:

```
SELECT *
FROM Customers
WHERE Country = 'India' AND
      (City = 'Chennai' OR City = 'Hyderabad');
```

Explanation: This SQL statement selects all fields from Customers where Country is 'India' and City must be Chennai or Hyderabad.

```
SELECT *
FROM Customers
WHERE NOT Country = 'India' AND NOT
      Country = 'USA';
```

Explanation: This SQL statement selects all fields from Customers where Country is NOT 'India' and not 'USA'.

* let us consider additional columns (age INT, SALARY INT) in Customers table.

BETWEEN Syntax:

```
SELECT column1, column2, ...
FROM <table_name>
WHERE column1 BETWEEN cond1 AND cond2;
```

Example:

```
SELECT *
FROM customers
WHERE age BETWEEN 25 AND 35;
```

ANY syntax:

```
SELECT column1, column2, ...
FROM <table_name>
WHERE column1 = ANY (cond1, cond2);
```

Example:

```
SELECT *
FROM customers
WHERE country = ANY ('India', 'USA');
```

SQL LIKE Operator:

The LIKE operator is used in a WHERE clause to search for a specified pattern in a column.

SQL Wildcard Characters:

* A wildcard character is used to substitute any other character(s) in a string.
* Wildcard characters are used with LIKE operator.
* % Percent and _ underscore are wild characters.
* % - The percent sign represents zero, one, or multiple characters
* _ - The underscore represents a single character

Note: MS Access use (?) instead of (_).

The percent and underscore can also be used in combinations.

LIKE Syntax:

```
SELECT column1, column2,....
FROM <table_name>
where column_name LIKE pattern;
```

* We can also combine other conditions using

AND or OR operators.

Examples:

```
SELECT *
FROM Customers
WHERE Customer_name LIKE 'a%';
```

'a%' → starts with a

'%a' → ends with a

'%or%' → contains or

'_r%' → has r in second position so on...

SQL IN Operator:

The IN operator allows us to specify multiple values in a WHERE clause

* The IN operator is short hand for multiple OR conditions.

IN Syntax:

```
SELECT column1, column2,...
FROM <table_name>
WHERE column_name IN (value1, value2,....);
```

(or)

```
SELECT column1, column2,...
FROM <table_name>
WHERE column_name IN (SELECT STATEMENT);
```

Subqueries we will learn later

Example:

```
SELECT *
FROM customers
WHERE country IN ('India', 'USA', 'UK');
```

```
SELECT *
FROM customers
WHERE country NOT IN ('India', 'USA', 'UK');
```

```
SELECT *
FROM Customers
WHERE Country IN ( SELECT country
                   From Suppliers);
```

different table (sub query)

SQL BETWEEN Operator:

The BETWEEN operator selects values within a given range. The values can be numbers, text, or dates.

* The BETWEEN operator is inclusive : begin and end values are included.

BETWEEN Syntax:

```
SELECT Column1, Column2,...
FROM <table_name>
WHERE Column_name BETWEEN Value1 AND
                          value2;
```

Example:

```
SELECT *
FROM Customers
WHERE Salary BETWEEN 10000 AND 20000;
```

```
SELECT *
FROM Customers
WHERE Salary NOT BETWEEN 20000 AND
                         40000;
```

```
SELECT *
FROM Customers
WHERE (Salary BETWEEN 10000 AND 20000)
AND NOT ID IN (1,2,3);
```

```
SELECT *
FROM Customers
WHERE Customer_name BETWEEN
    'xxx' AND 'YYY';
```

Dates Examples using a random table

```
SELECT *
FROM ORDERS
WHERE OrderDate Between #07/04/2020#
                AND #09/29/2022#;
```