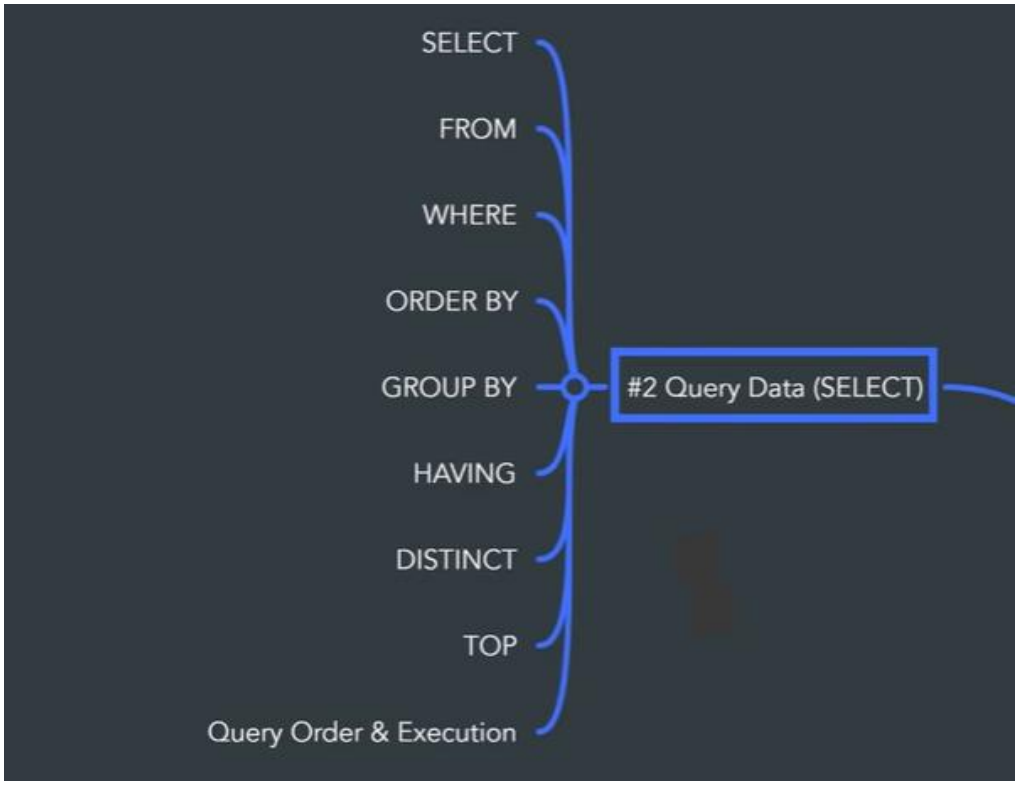
SQL notes

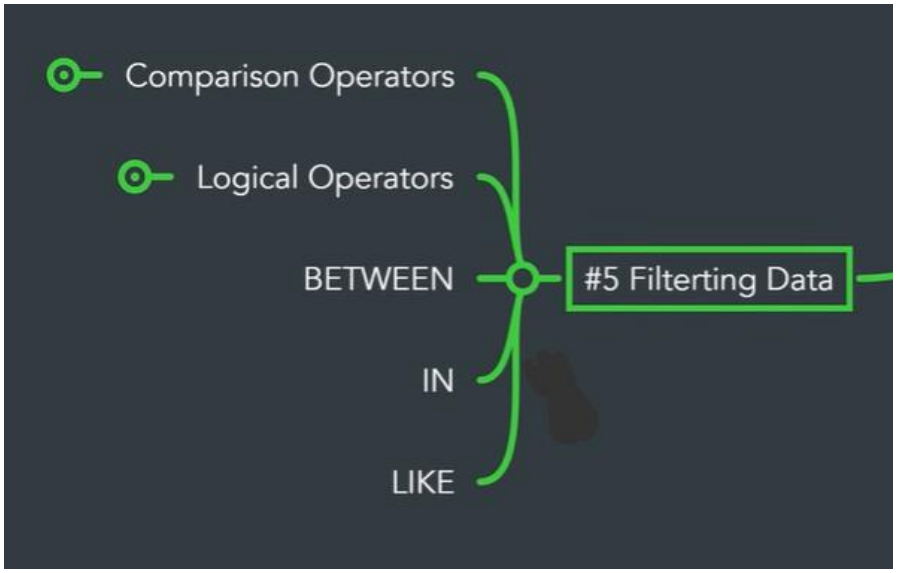
Agenda:

Detailed agenda:

Basic level-



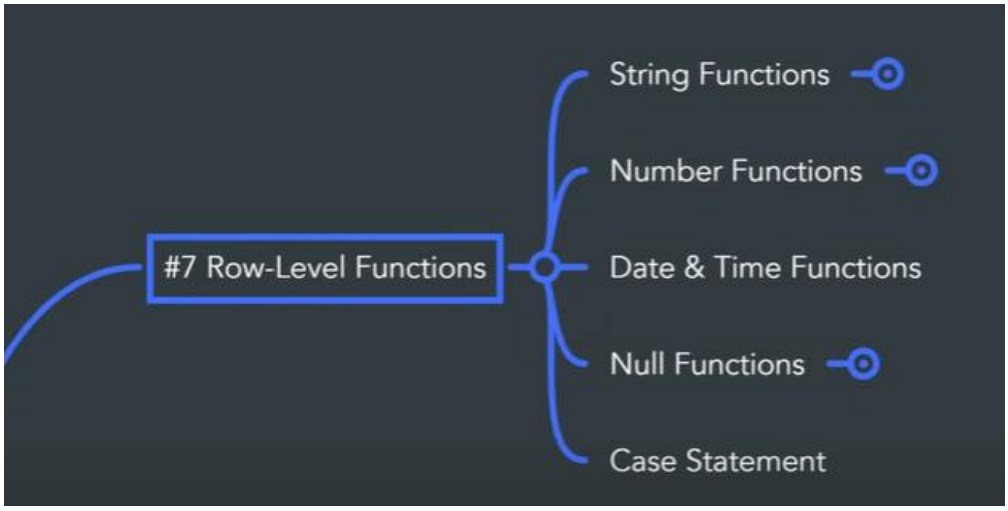
Intermediate level-

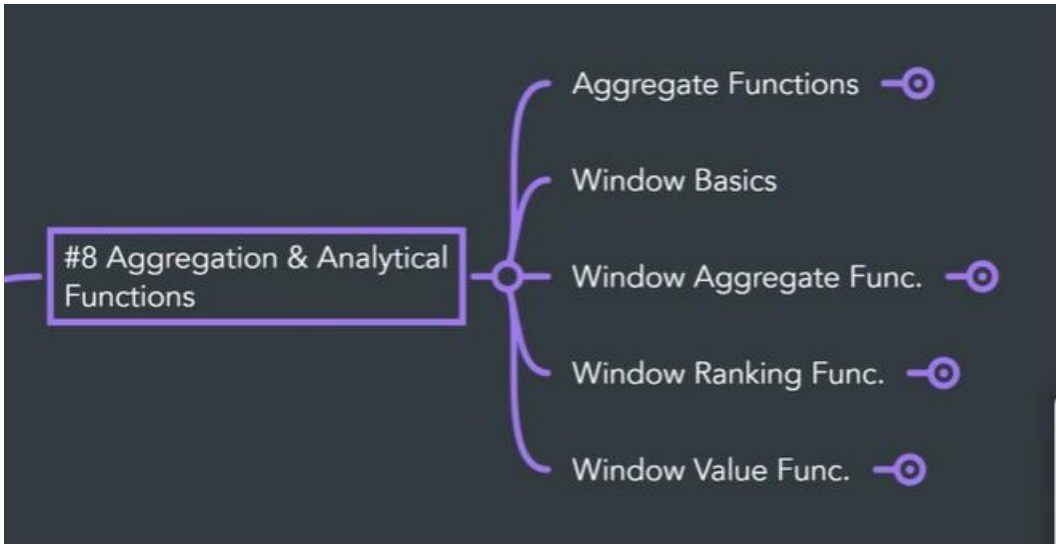



Joining Data
SET Operators
#6 Combining Data

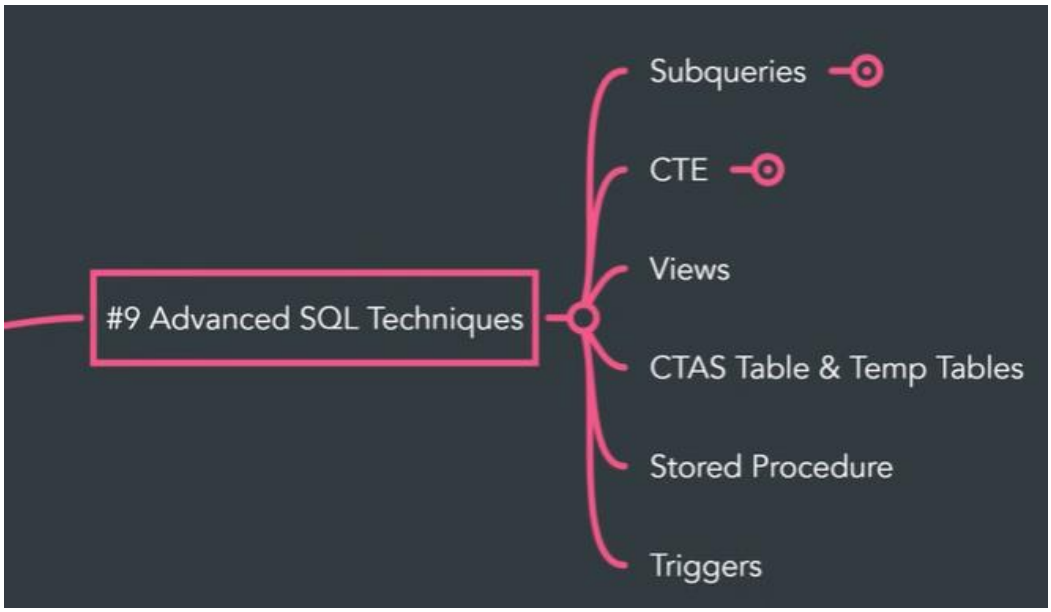

INNER Join
LEFT Join
RIGHT Join
FULL Join
Basic Joins
LEFT Anti Join
Right Anti Join
FULL Anti Join
Cross Join
Advanced Joins
Joining Data
How to Choose The Right Join
How To Join Multiple Tables

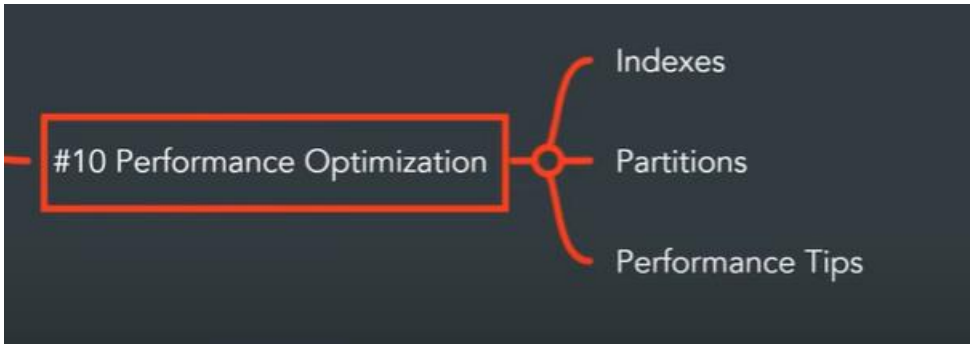

UNION
UNION ALL
EXCEPT
INTERSECT
SET Operators

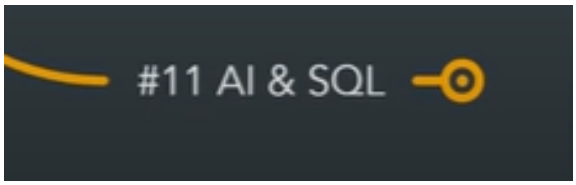




Advanced SQL-



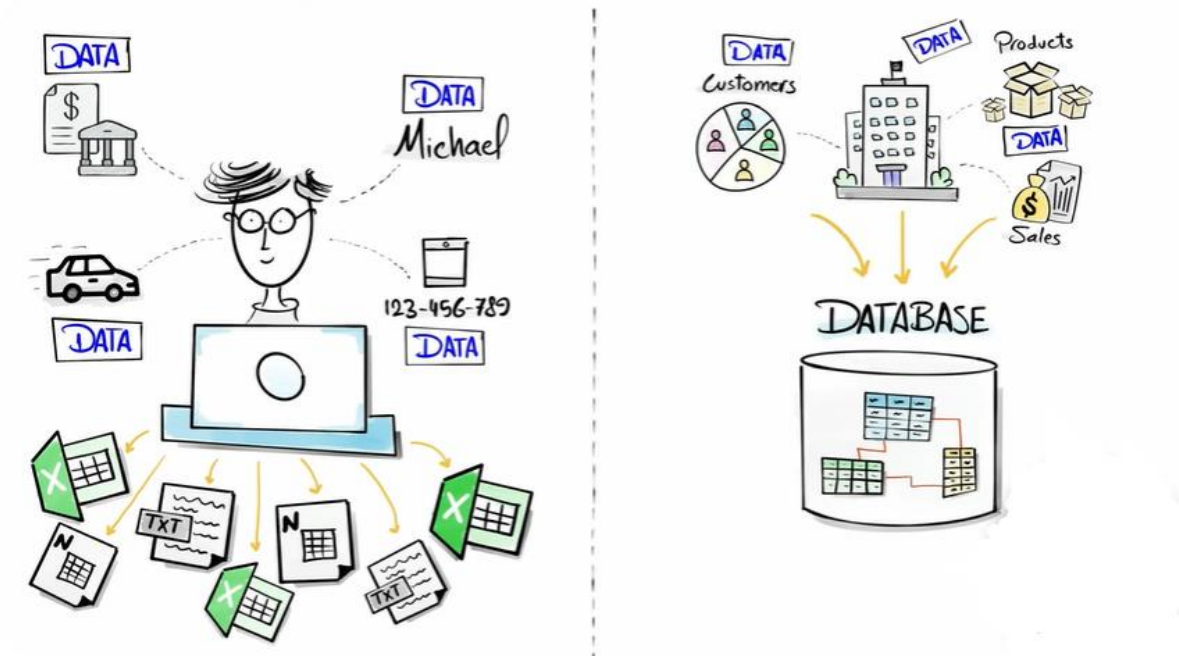

#10 Performance Optimization
Indexes
Partitions
Performance Tips


#11 AI & SQL


#12 SQL Projects
Data Warehousing Project
Exploratory Data Analysis (EDA) project
Advanced Data Analytics Project

Introduction

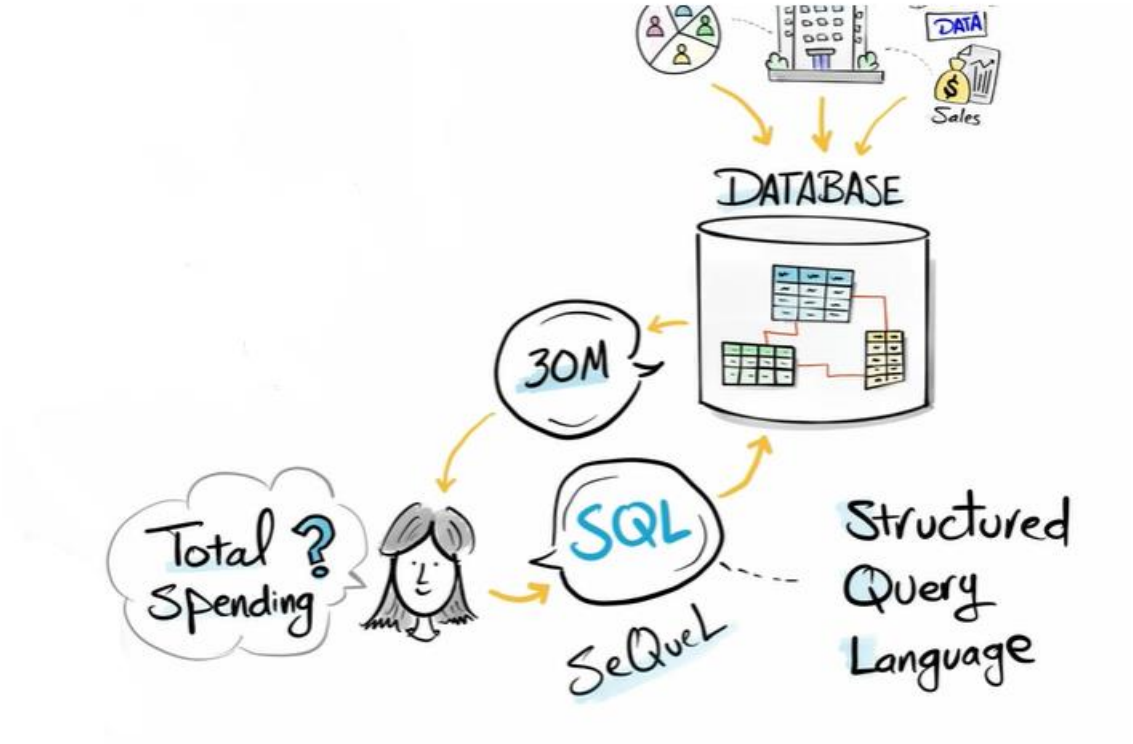
Everything generates data, everywhere we have data.



Where to store the data? - excel/text files are not a good option to store data.

Here is where Database comes into picture. Database is like a huge container which stores data in an organized manner for easy access and management of data.

With conventional excel and text files, it would be very difficult to understand data and get answers.



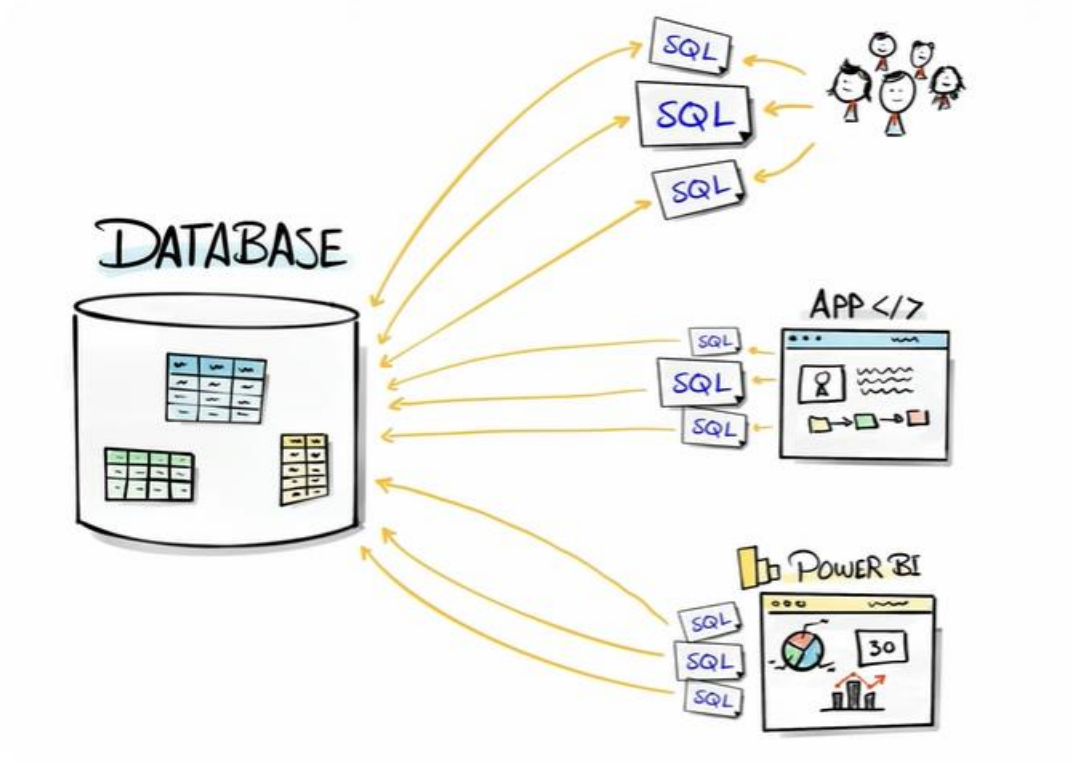
When data stores in a database, we can speak with the database about our query to get results.

How can we speak with database – using SQL – Structured Query language.

SQL **Structured Query language** is a programming language used to communicate with database.

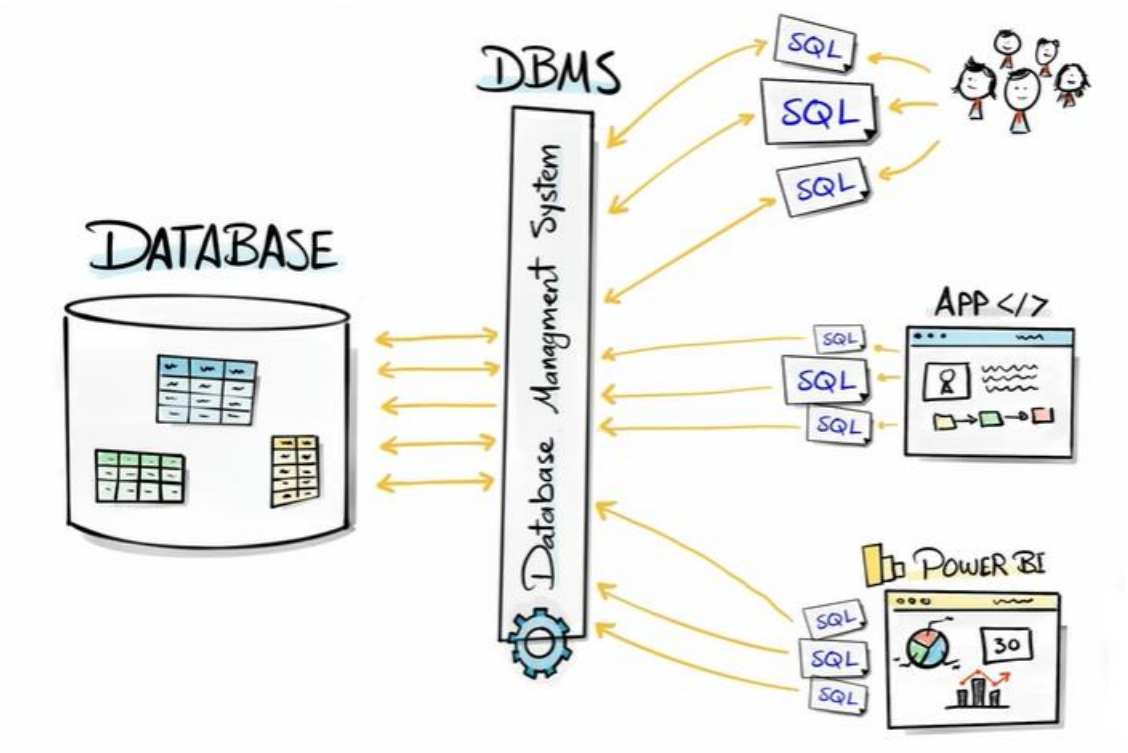
*Conventional excel/text files will not handle huge amounts of data, so we use Database which will handle huge amounts of data.

*Also, its safe and secure to store critical information in database than storing in Excel/text files and we can control who can have access.

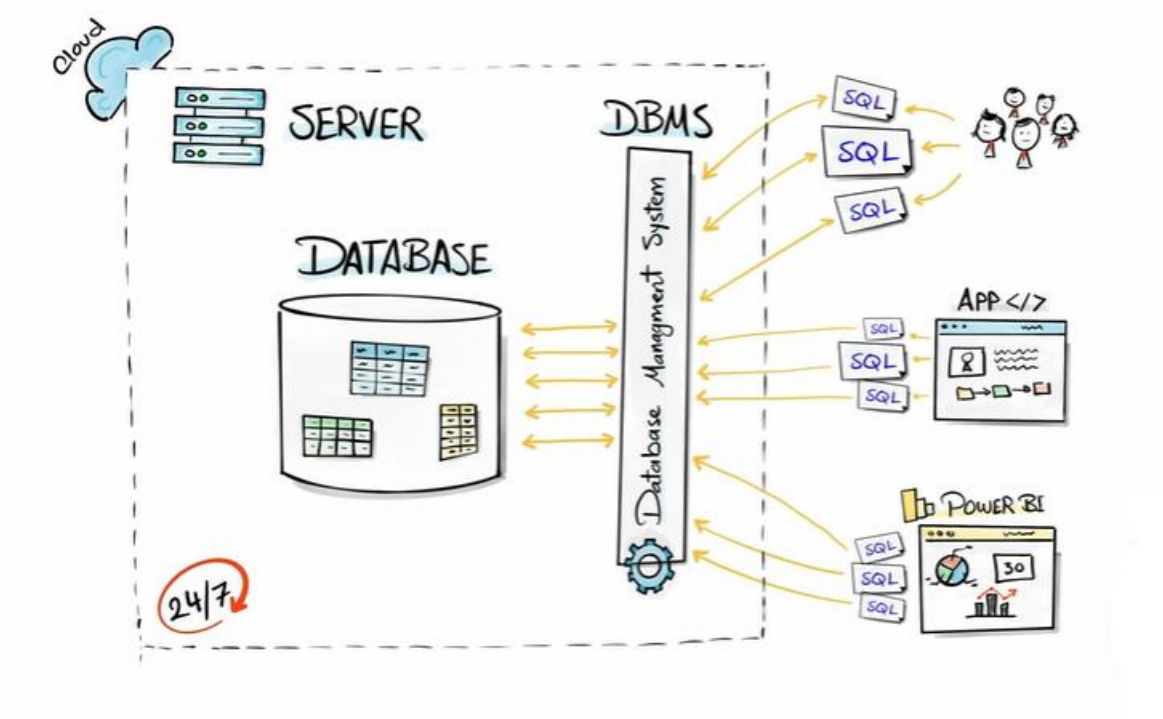


A database can have many requests from users, application and then Visualization tools.

In-order to manage all these requests we need a management system that will manage all the incoming requests. – A database Management system



Database Management system is a software that manages all the different requests to our database, and makes priority which query to be executed first and also checks the security.



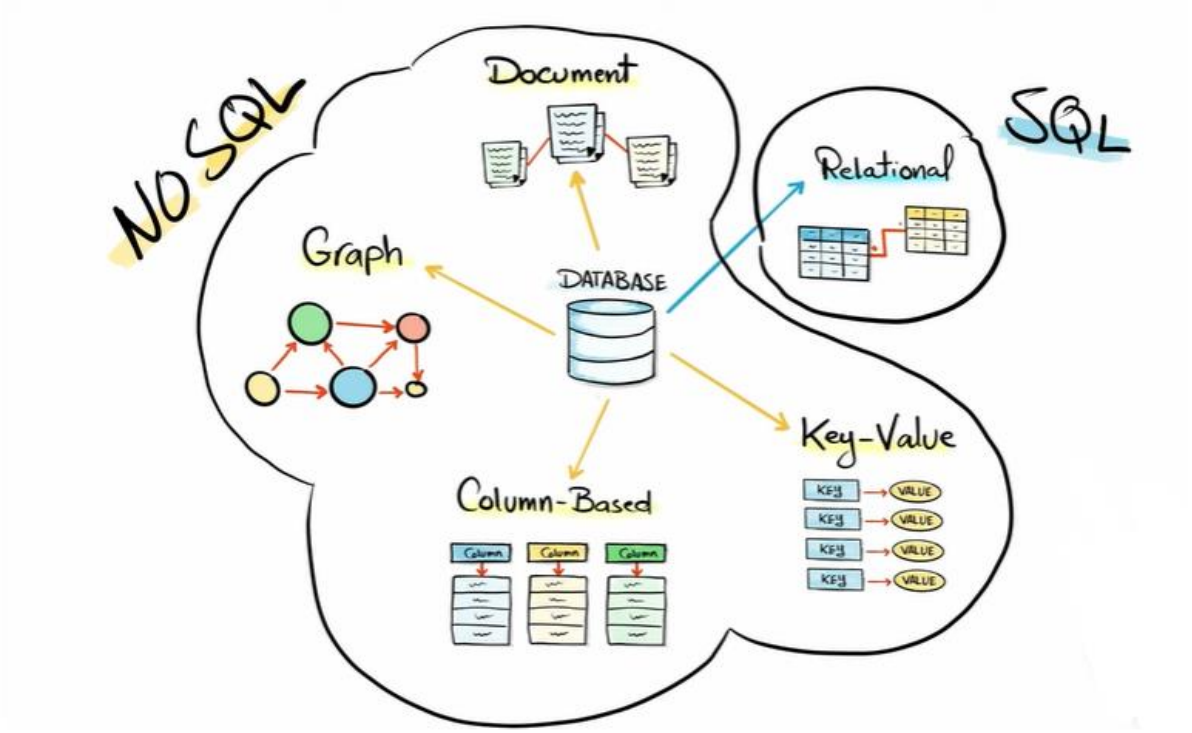
Database
Stores Data

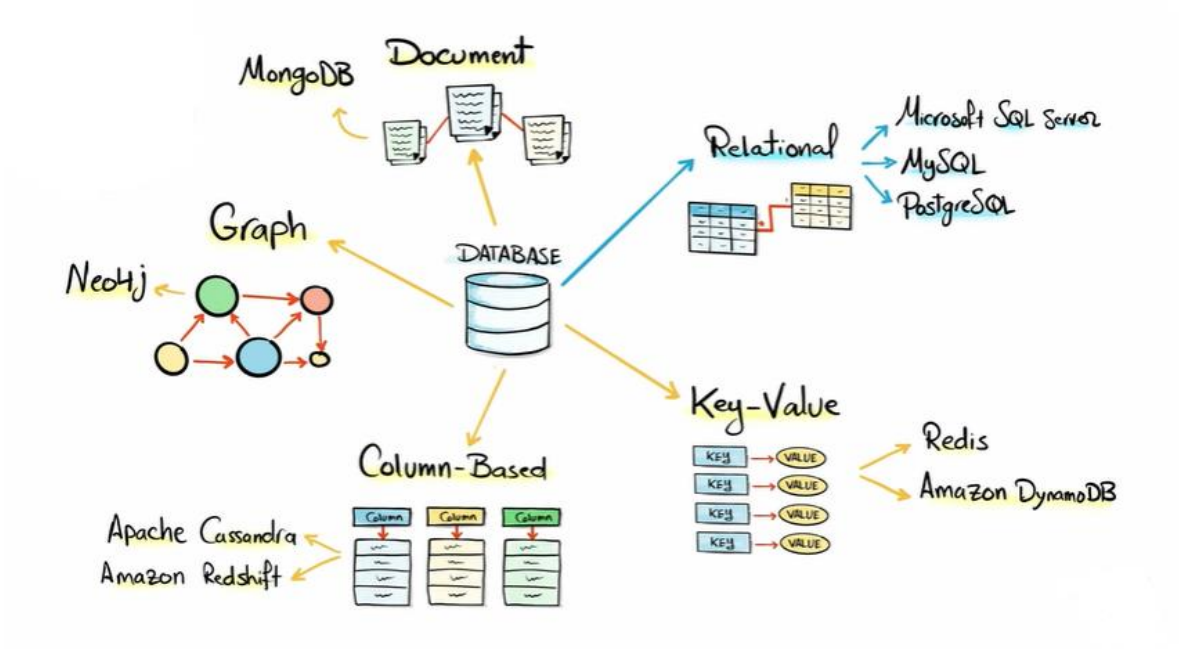
SQL
Speak to Database

DBMS
Manages Database

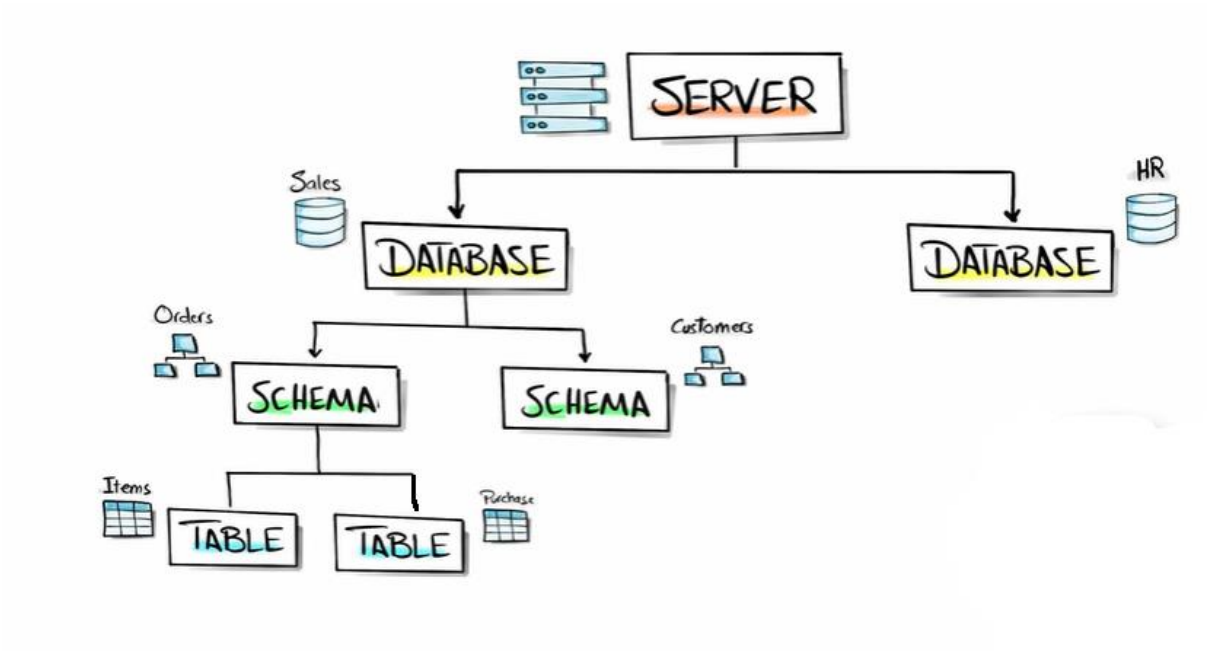
Server
Where Database lives

Types of Databases





Database Structure



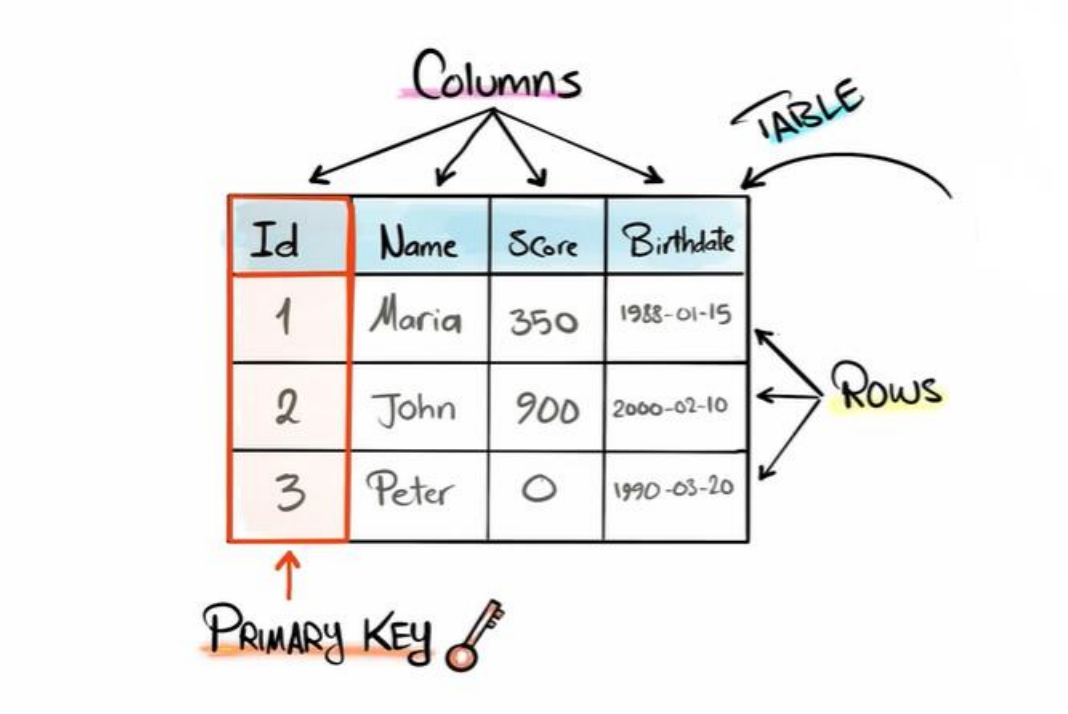
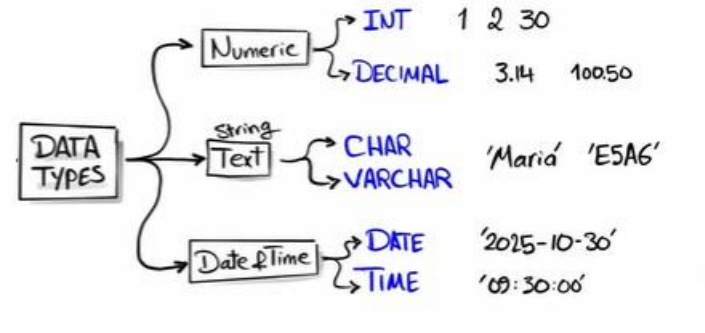


Table: A table is like a spreadsheet which organizes data into columns and rows.

Columns- are also called as fields. The column defines the data that you store.
Rows- are also called as records. The row is actually where data stores. In above example each row represents one customer info.

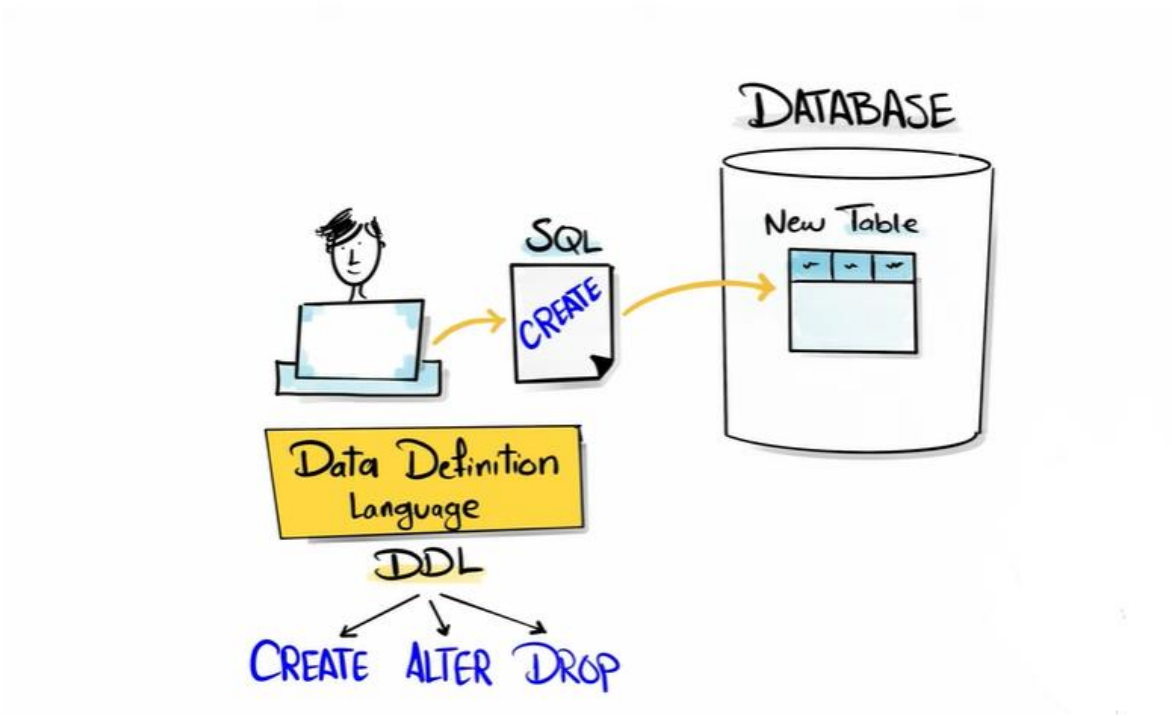
Primary key: A unique identifier for each row.



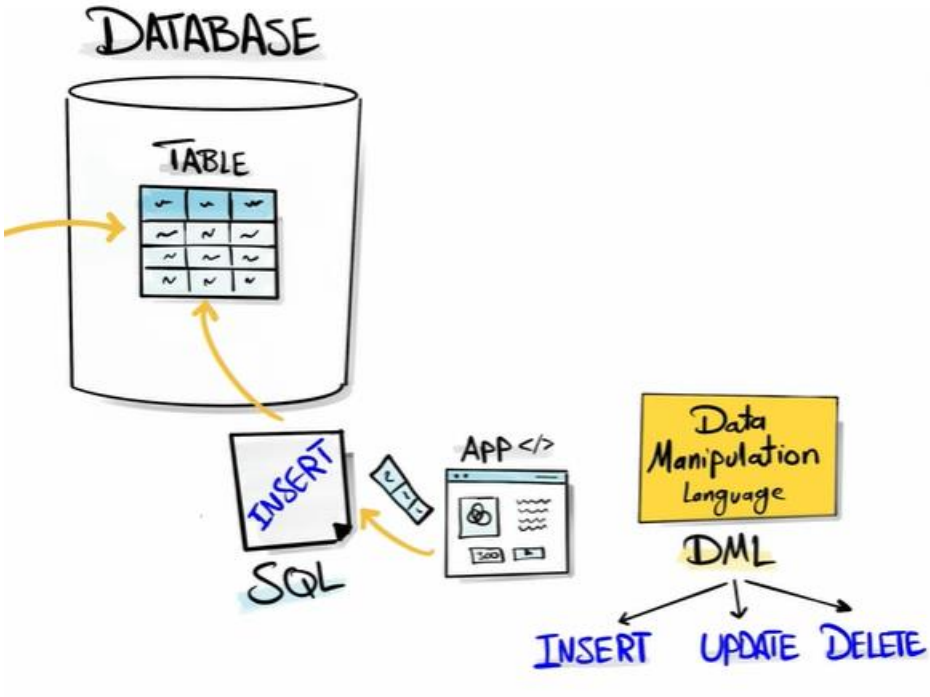
Data type: What kind of data we are storing.

Types of SQL Commands

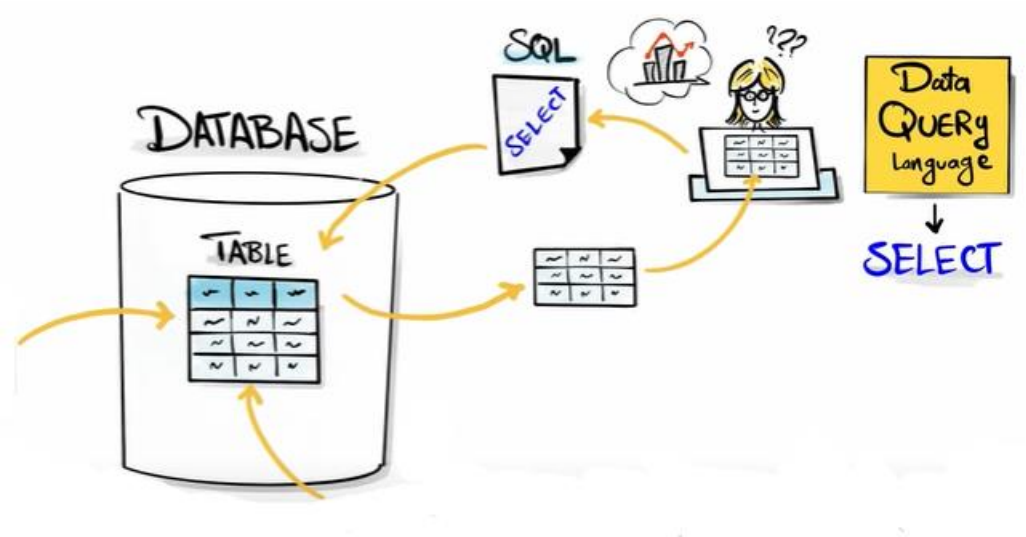
Data Definition language – used to perform actions related to table- create/modify/delete a table.



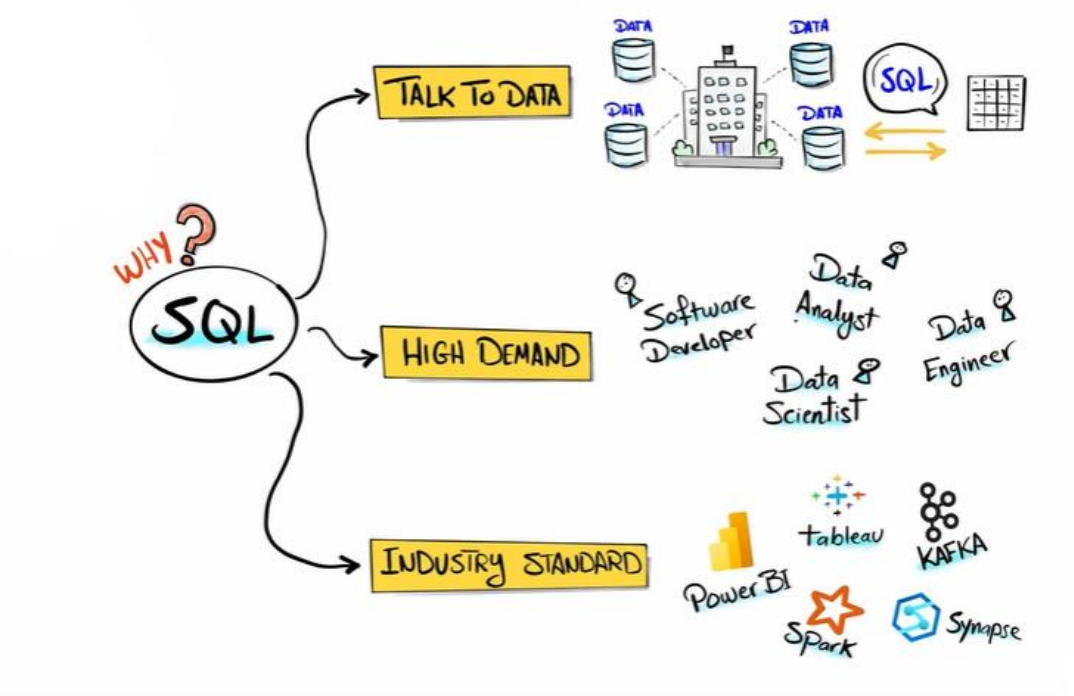
Data manipulation language – used to perform actions related to rows – Insert/Update/Delete a row.



Data query Language – query to get results.



Why SQL



Basic level

Create database

1) Right click Database>Click on New Database>Provide database name.
2) Using script to create database.

```
USE master;
```

```
GO

-- Drop and recreate the 'Database_name' database
IF EXISTS (SELECT 1 FROM sys.databases WHERE name = 'Database_name')
BEGIN
    ALTER DATABASE Database_name SET SINGLE_USER WITH ROLLBACK IMMEDIATE;
    DROP DATABASE Database_name;
END;
GO

-- Create the 'Database_name' database
CREATE DATABASE Database_name;
```

Restore database

Url: https://learn.microsoft.com/en-us/sql/samples/adventureworks-install-configure?view=sql-server-ver16&tabs=ssms

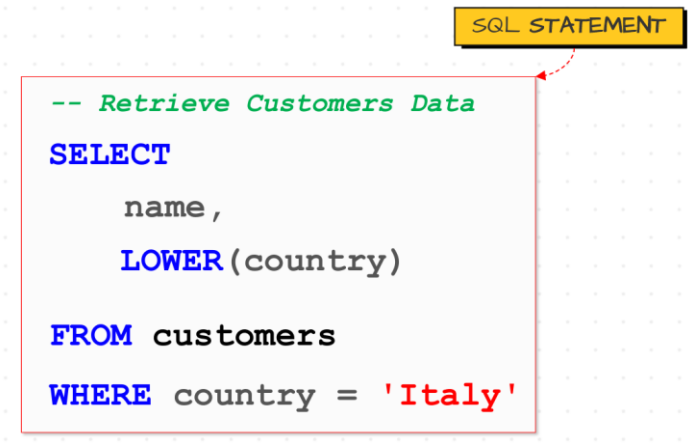
Download .bak file.

Place the file in the location: C:\Program Files\Microsoft SQL Server\MSSQL16.SQLEXPRESS\MSSQL\Backup

Open SSMS >Right click on Database>Click on Restore Database>Select device>Select database>Click Ok.

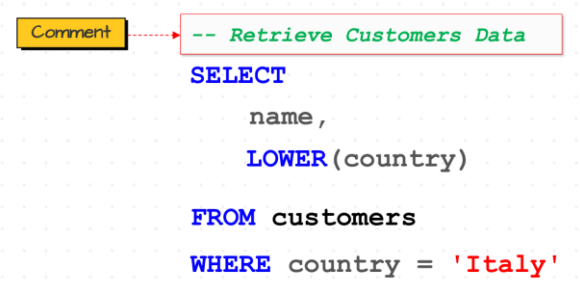
SQL Components

SQL Statement



A SQL statement is a single command or query written in the SQL (Structured Query Language) syntax to interact with a database. SQL statements are used to perform various tasks, such as retrieving, inserting, updating, deleting, or managing data within a database.

SQL Comments

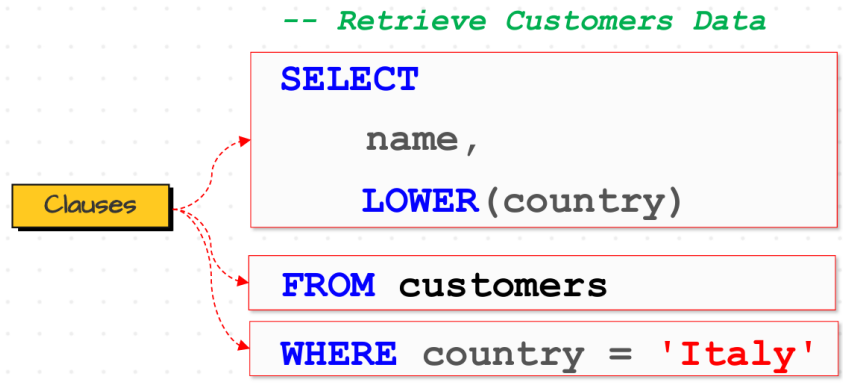


In SQL, comments are annotations or notes that you can add to your code to make it easier to understand. They are not executed by the database engine and are purely meant for human readability. Comments are particularly helpful when collaborating with others or when you revisit your code after some time.

There are two main types of comments in SQL:

1) **Single-Line Comments**: Start with -- (double hyphen). Anything after these symbols on the same line is considered a comment.
2) **Multi-Line Comments**: Enclosed between /* and */. These can span multiple lines.

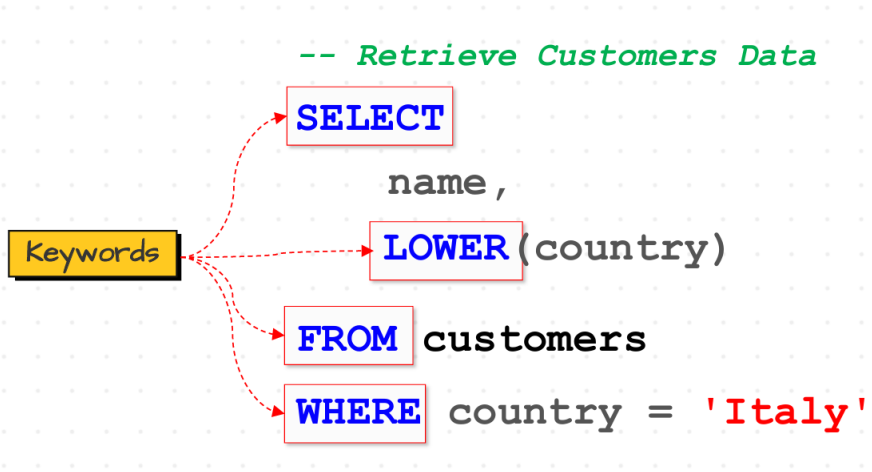
SQL Clause



A SQL clause is a component of a SQL statement used to define specific conditions or modify its behaviour. Clauses help refine queries, control data manipulation, and manage database interactions more effectively.

SELECT	**DISTINCT**	**CASE**
FROM	**UNION / UNION ALL**	**SET**
WHERE	**EXCEPT**	**ALTER**
ORDER BY	**INTERSECT**	**INSERT INTO**
GROUP BY	**IN**	**DELETE**
HAVING	**BETWEEN**	**TRUNCATE**
JOIN (INNER, LEFT, RIGHT, FULL OUTER)	**LIKE**	**WITH (Common Table Expressions, or CTEs)**
LIMIT / TOP / FETCH	**IS NULL / IS NOT NULL**	

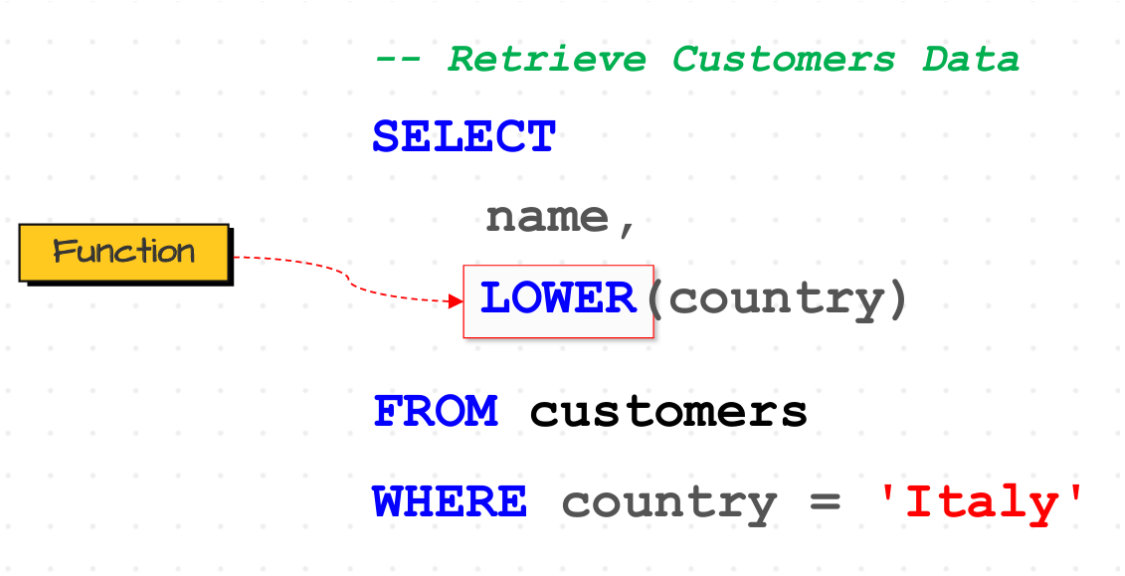
Keywords



Keywords in SQL are reserved words that have special meanings and are part of the language syntax. They cannot be used as identifiers (like table names, column names, etc.) unless enclosed in quotes.

ADD	**DELETE**	**INNER**	**PRIMARY**
ALL	**DESC**	**INSERT**	**REFERENCES**
ALTER	**DISTINCT**	**INTERSECT**	**REVOKE**
AND	**DROP**	**INTO**	**RIGHT**
ANY	**ELSE**	**IS**	**ROLLBACK**
AS	**EXISTS**	**JOIN**	**SELECT**
ASC	**FETCH**	**KEY**	**SET**
BETWEEN	**FOR**	**LEFT**	**TABLE**
BY	**FOREIGN**	**LIKE**	**TOP**
CASE	**FROM**	**LIMIT**	**TRUNCATE**
CHECK	**FULL**	**NOT**	**UNION**
COLUMN	**GRANT**	**NULL**	**UNIQUE**
COMMIT	**GROUP**	**ON**	**UPDATE**
CONSTRAINT	**HAVING**	**OR**	**VALUES**
CREATE	**IN**	**ORDER**	**VIEW**
DEFAULT	**INDEX**	**OUTER**	**WHERE**
WITH			

Functions



Functions in SQL are built-in operations that allow you to manipulate and process data effectively. They can perform calculations, modify values, and analyze data. SQL functions are broadly categorized into:

1. Aggregate Functions
COUNT()
SUM()
AVG()
MAX()
MIN()

2.Scalar Functions
UPPER()
LOWER()
LENGTH() / LEN()
ROUND()

3.String Functions
SUBSTRING() / MID()
TRIM()
REPLACE()
CONCAT()
CHARINDEX() / INSTR()

4. Date and Time Functions
GETDATE() / NOW()
DATEPART()
YEAR(), MONTH(), DAY()
DATENAME()
DATEDIFF()
DATEADD()

5. Mathematical Functions
ABS()
POWER()
SQRT()
ROUND()
CEIL() / CEILING()
FLOOR()

6.Conversion Functions
Cast()
Convert()
Format()

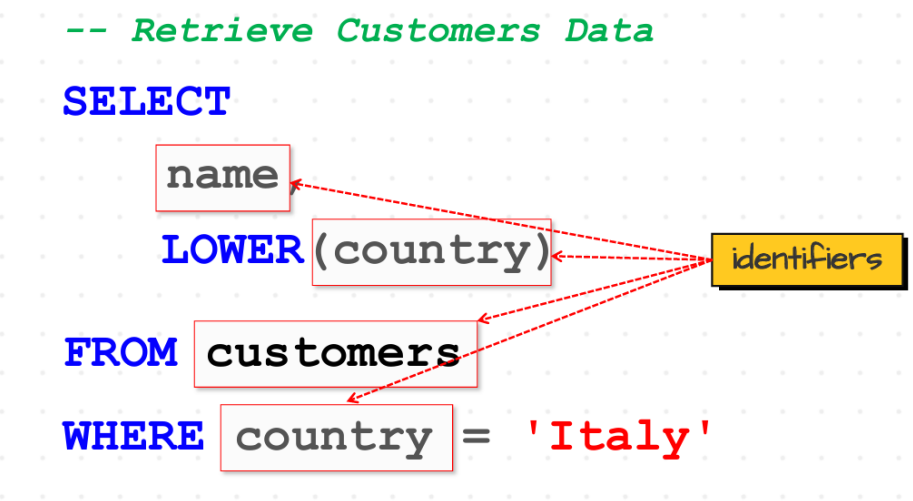
7. Others

COALESCE()

ISNULL() / IFNULL()

CASE()

Identifiers



Identifiers in SQL are the names used to represent database objects like tables, columns, indexes, views, procedures, and more. They serve as unique labels that help identify and refer to these objects in SQL queries and commands.

1. **Rules for Naming**:

- An identifier typically starts with a letter (A-Z or a-z) or an underscore (_), followed by letters, numbers (0-9), or underscores.
- Special characters (e.g., @, $, #) and spaces are generally not allowed unless enclosed in double quotes (") or square brackets ([]) in some systems.
- Reserved keywords in SQL (e.g., SELECT, WHERE) cannot be used as identifiers unless enclosed.

2. **Case Sensitivity**:

- SQL identifiers' case sensitivity depends on the database system. For instance, SQL Server treats identifiers as case-insensitive by default, whereas PostgreSQL treats them as case-sensitive when enclosed in double quotes.

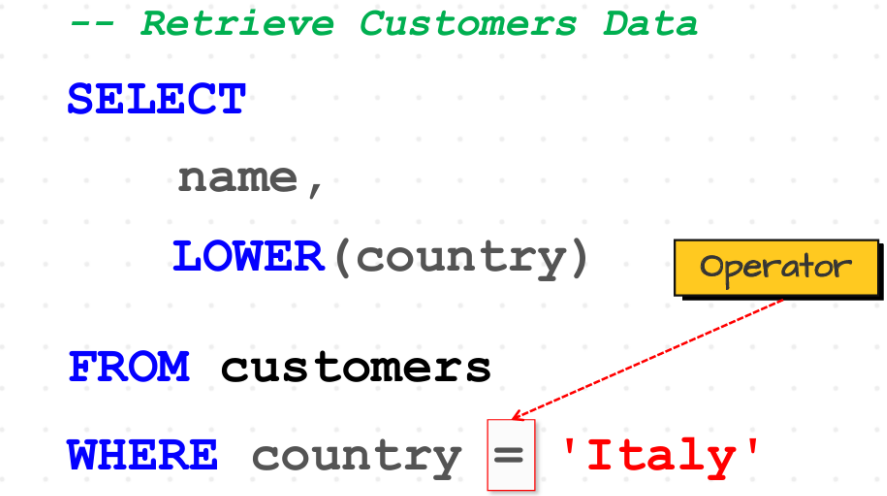
3. **Types of Identifiers**:

- **Regular Identifiers**: Follow the standard naming conventions and do not require quotes.
- **Delimited Identifiers**: Enclosed in double quotes (") or square brackets ([]), allowing the use of special characters and reserved keywords.

4. **Naming Best Practices**:

- Use meaningful names (e.g., user_id, order_date) to improve code readability.
- Stick to a consistent naming convention, like snake_case (order_details) or camelCase (orderDetails).

Operators



SQL operators are symbols or keywords used to perform various operations on data stored in a database. They help manipulate and query data efficiently. Here's an overview of the types of operators in SQL:

1. Arithmetic Operators

These are used for mathematical computations:

+ (Addition)
- (Subtraction)
* (Multiplication)
/ (Division)
% (Modulo)

2. Comparison Operators

These are used to compare two values:

= (Equal to)'
<> or != (Not equal to)
> (Greater than)
< (Less than)
>= (Greater than or equal to)
<= (Less than or equal to)

3. Logical Operators

These combine multiple conditions:

AND (Both conditions must be true)
OR (At least one condition must be true)
NOT (Negates a condition)

4. Bitwise Operators

Operate at the binary level:

& (Bitwise AND)
| (Bitwise OR)
^ (Bitwise XOR)

5. Set Operators

These are used to combine results of two queries:

6. Other Operators

LIKE: Used for pattern matching (e.g., WHERE name LIKE 'A%').

UNION (Combines results, removes duplicates)

UNION ALL (Combines results, keeps duplicates)

INTERSECT (Returns common results)

EXCEPT (Returns results from the first query not present in the second)

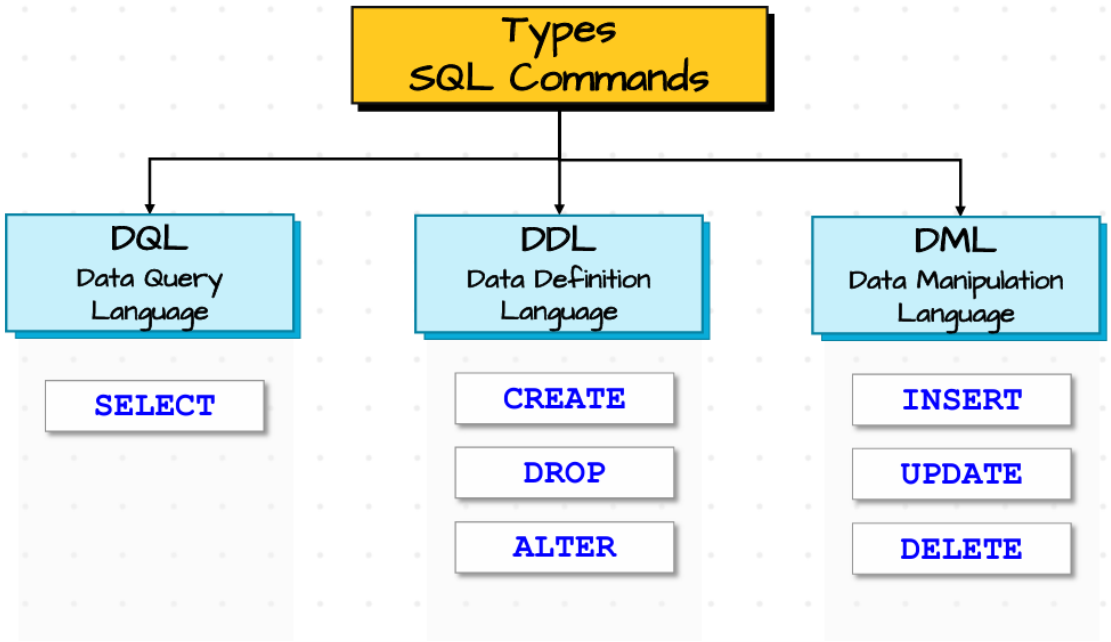
IN: Checks if a value exists in a list (e.g., WHERE id IN (1, 2, 3)).

BETWEEN: Checks if a value falls within a range (e.g., WHERE age BETWEEN 18 AND 25).

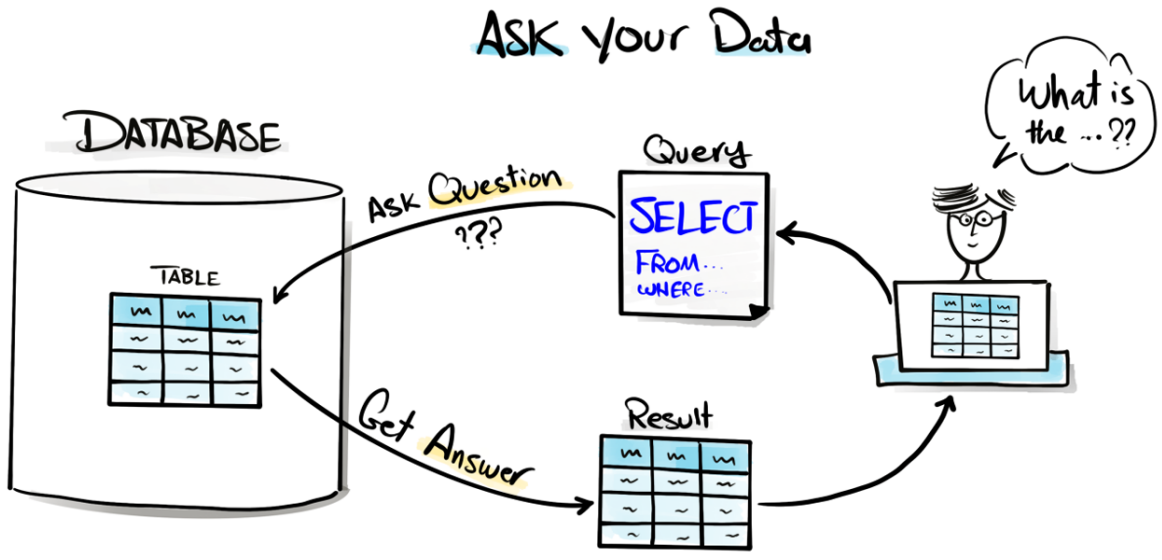
IS NULL: Tests for null values.

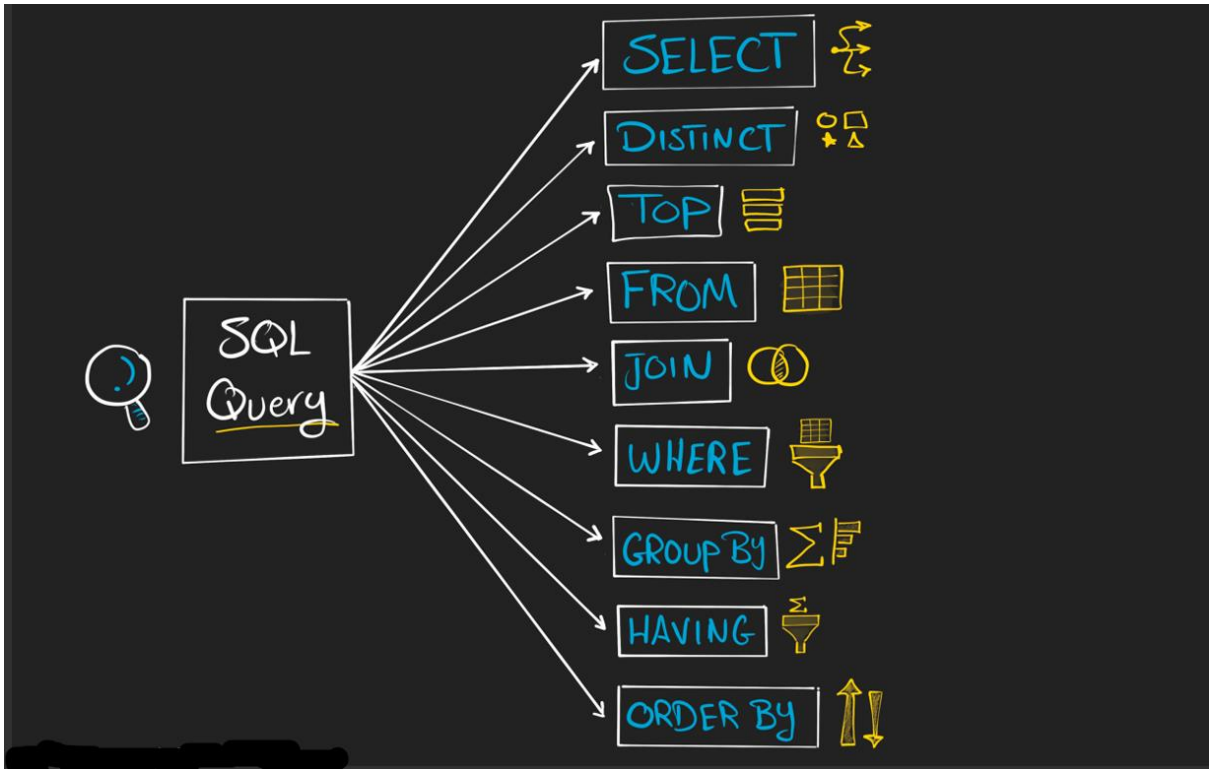
EXISTS: Checks if a subquery returns any rows.

Types of SQL Commands

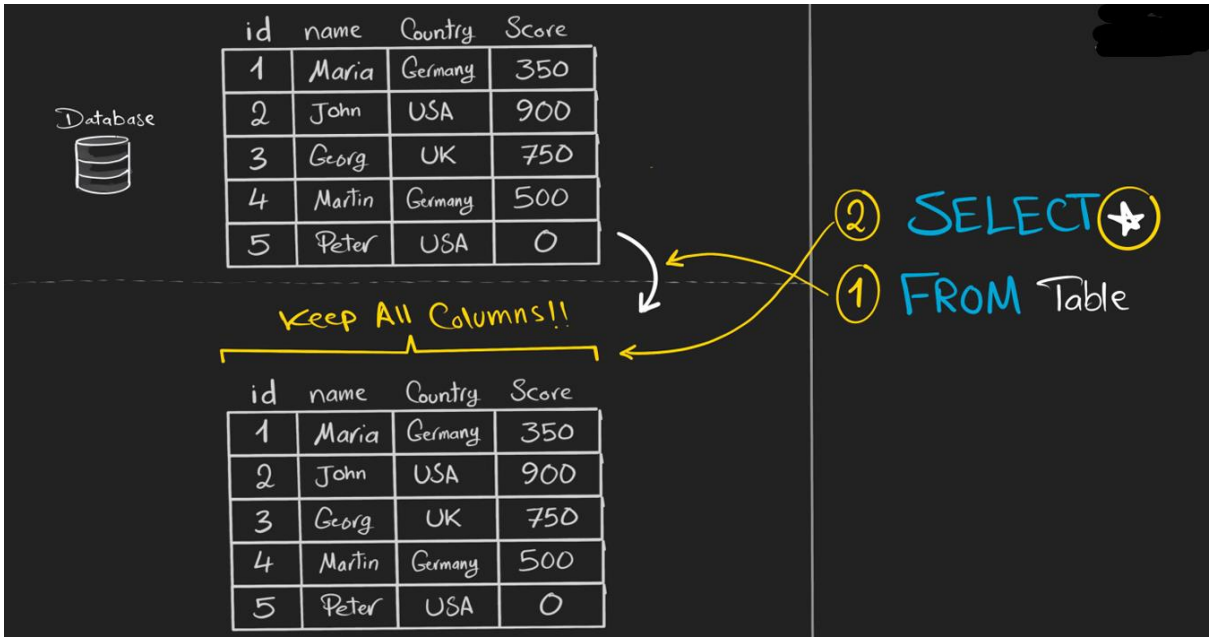


Data Query Language

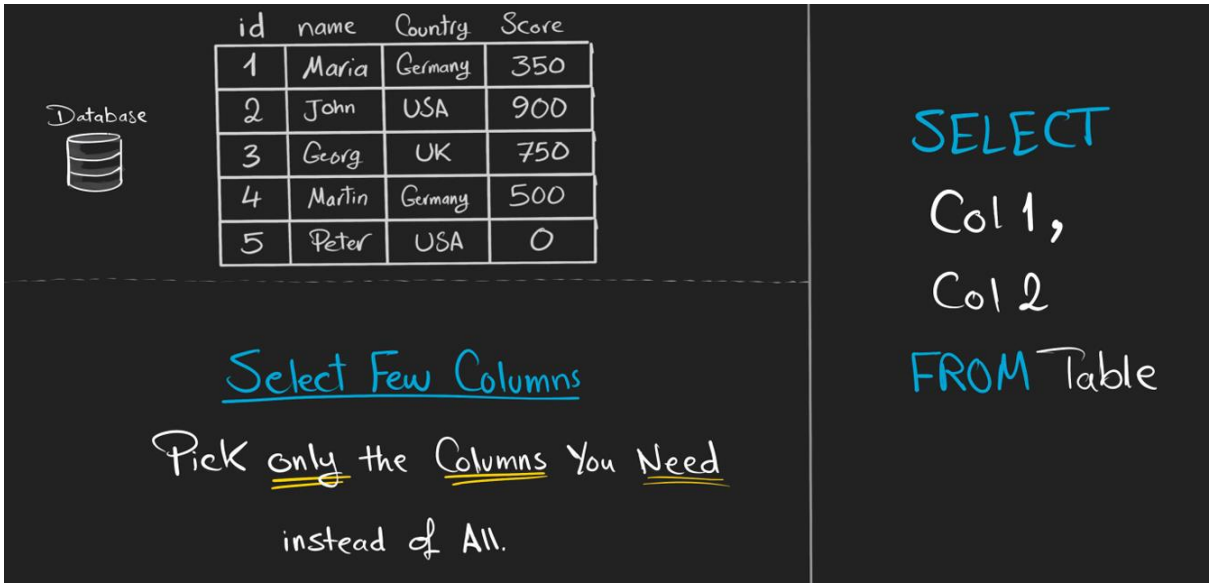


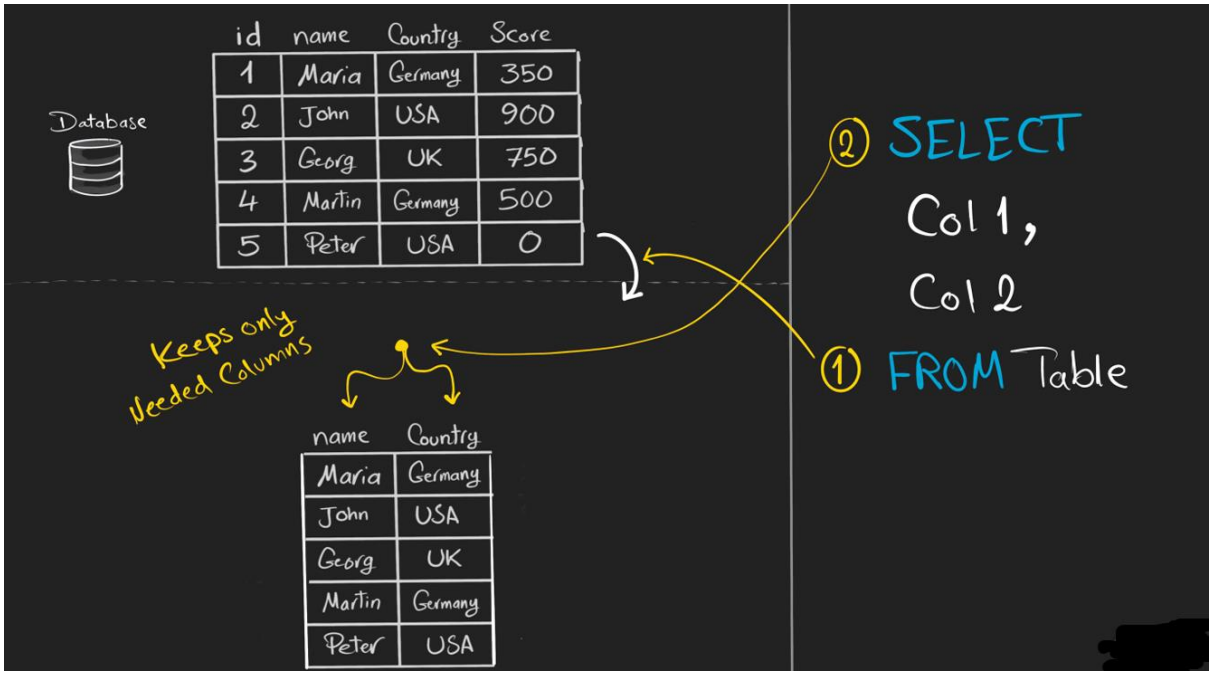


1)Select all the columns



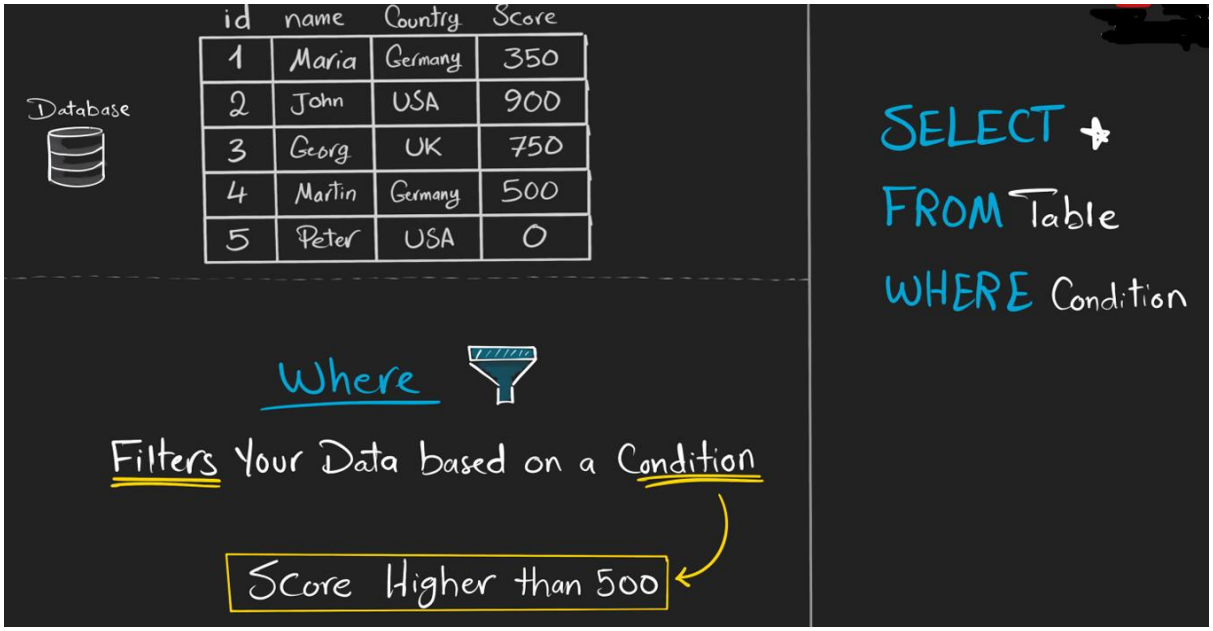
2)Select only mentioned columns in the SQL Statement

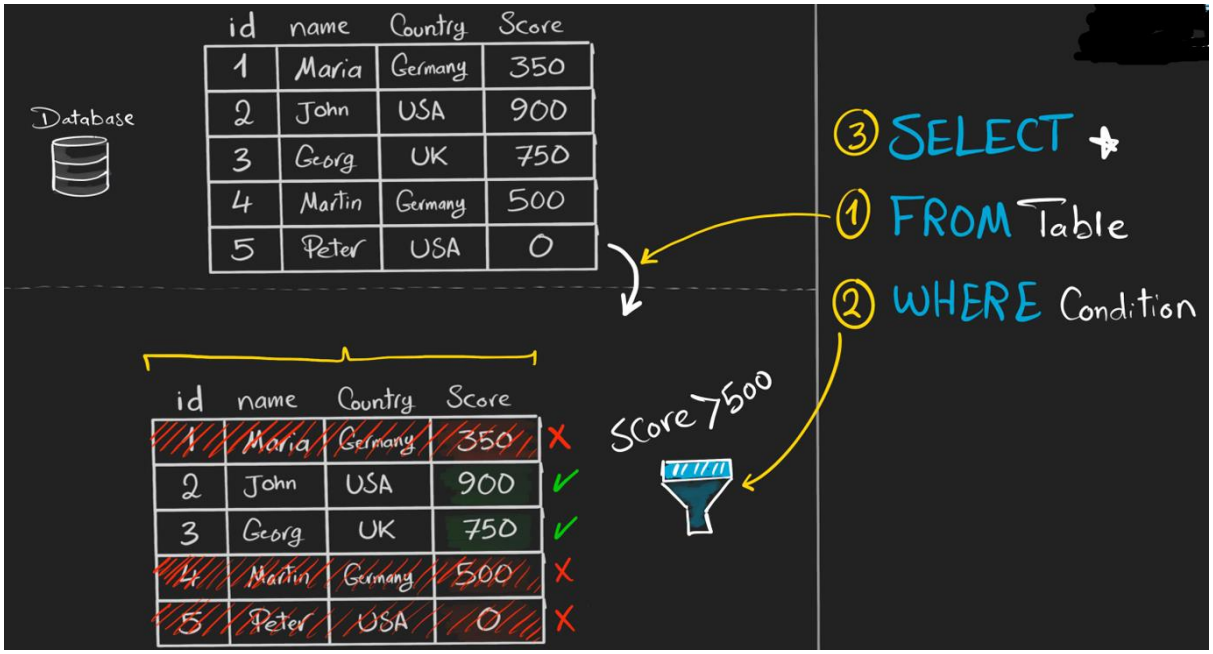




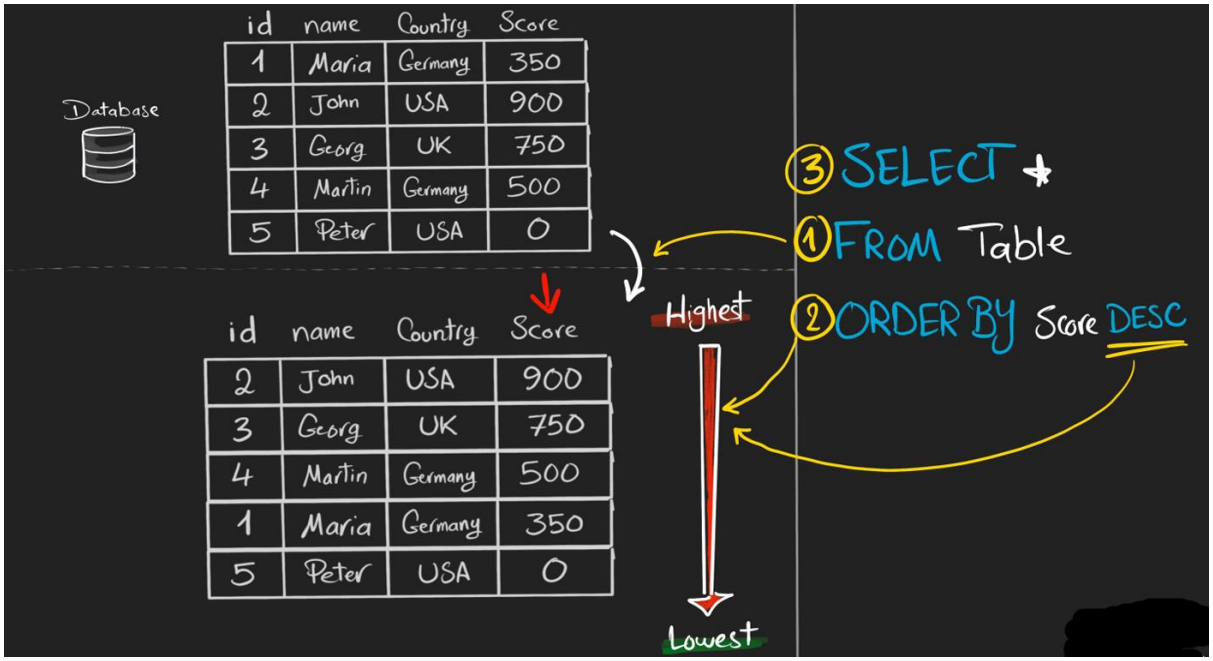
SQL Order of execution

Using where clause

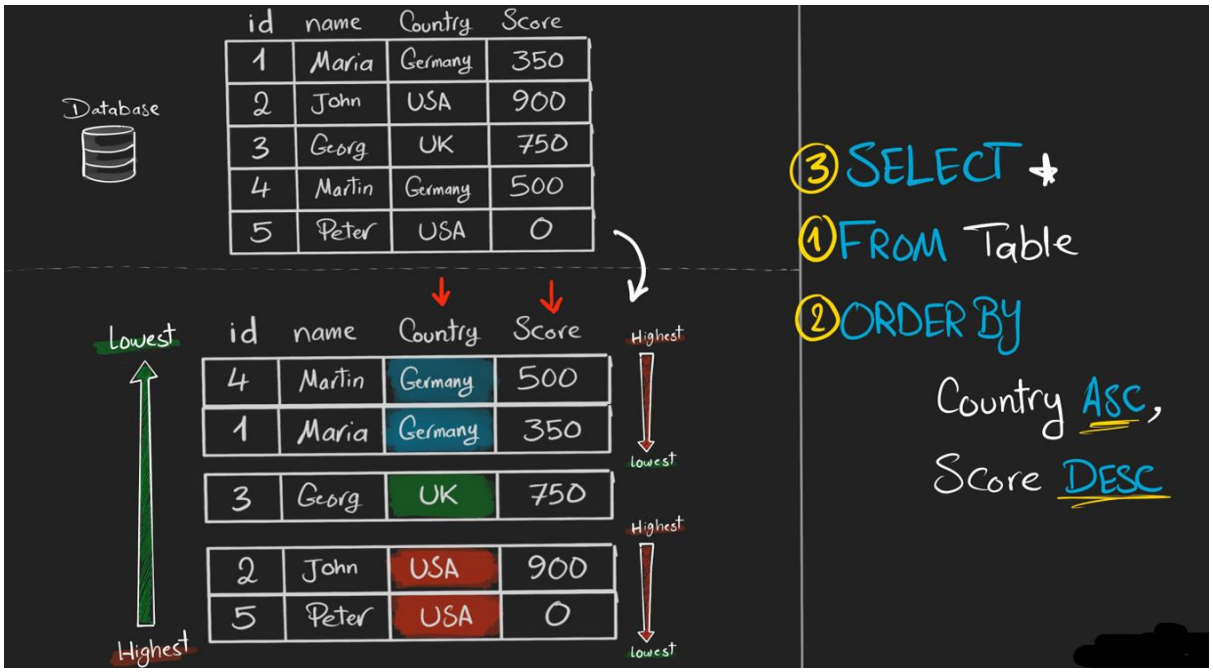




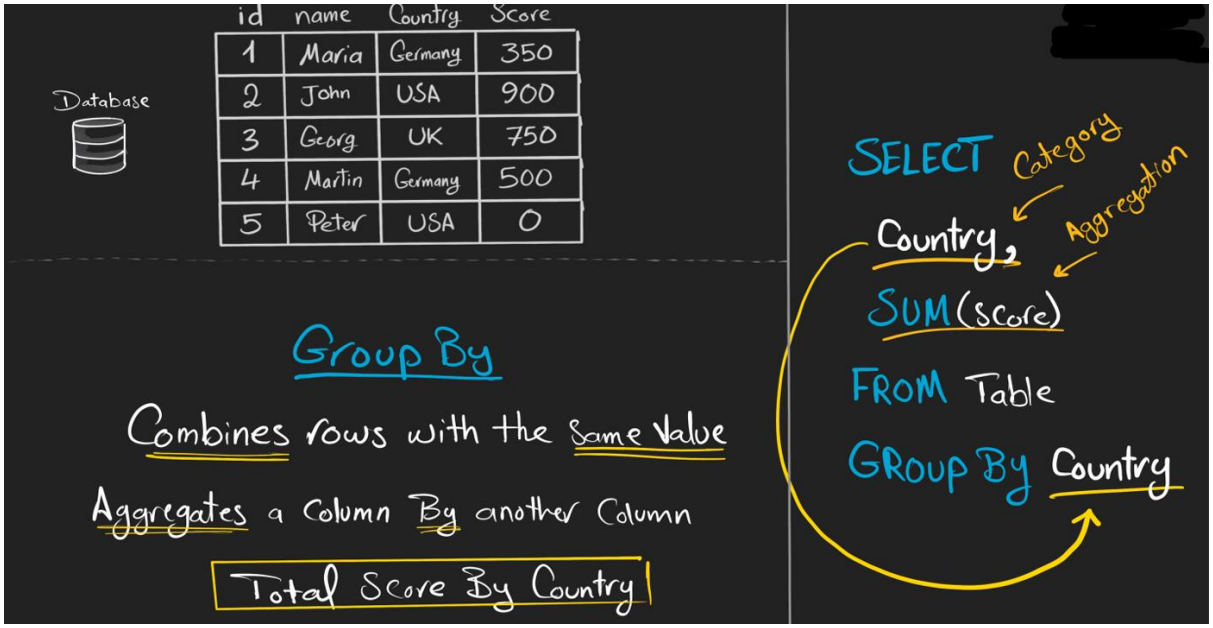
Using Order by clause

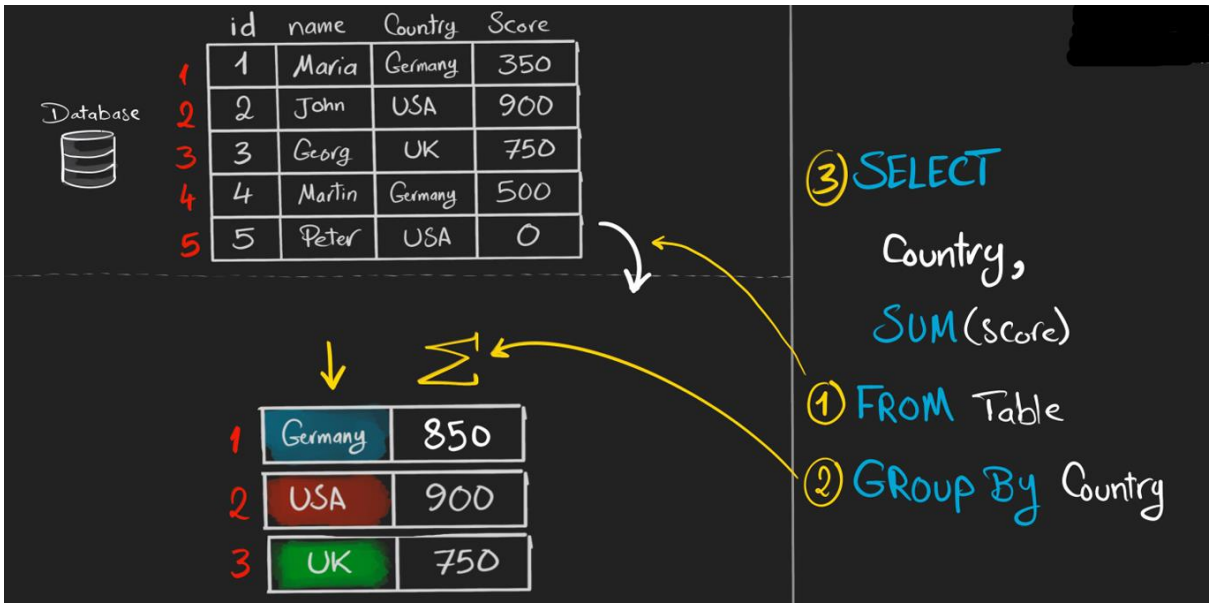


Order by 2 columns

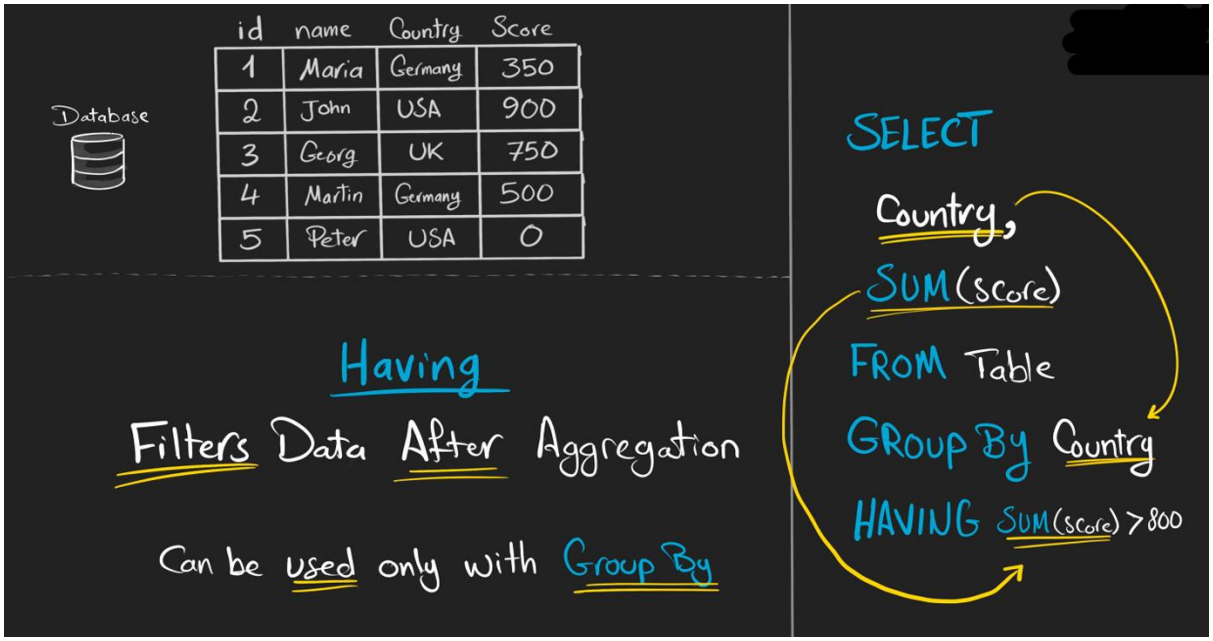


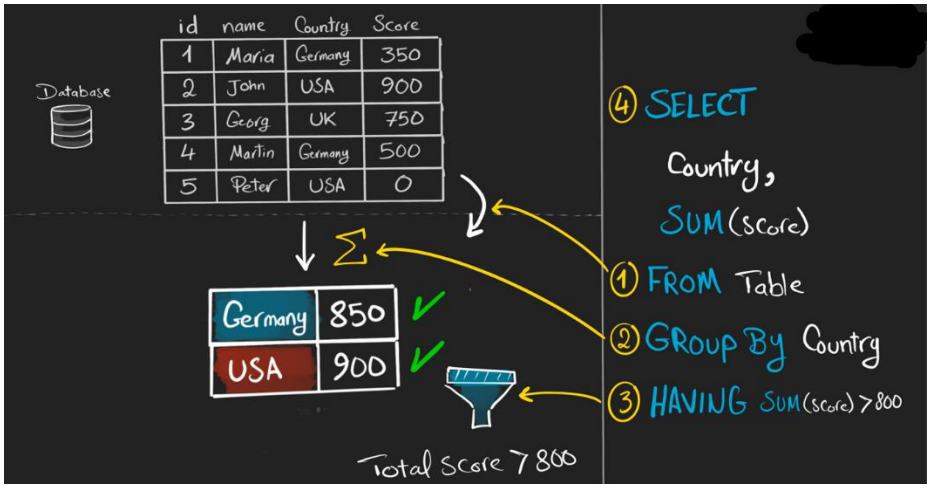
Using Group by clause

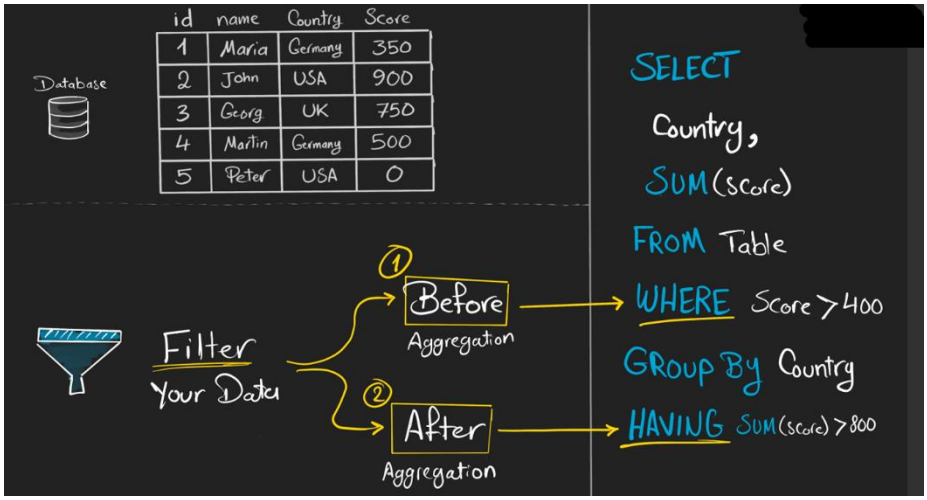




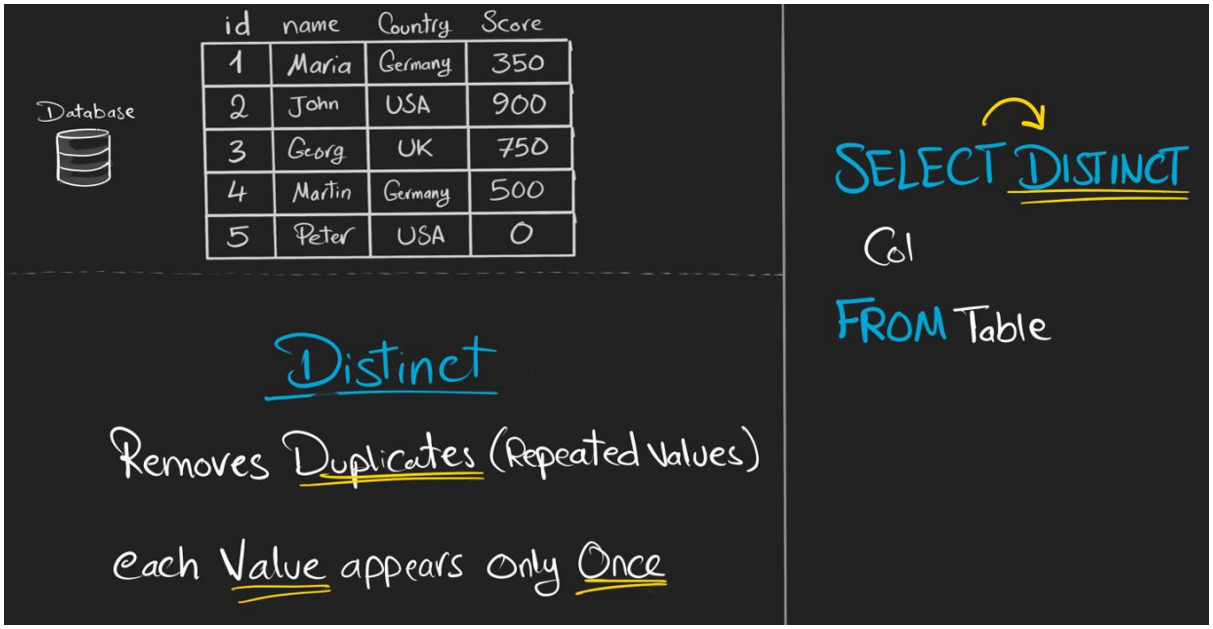
Using Having

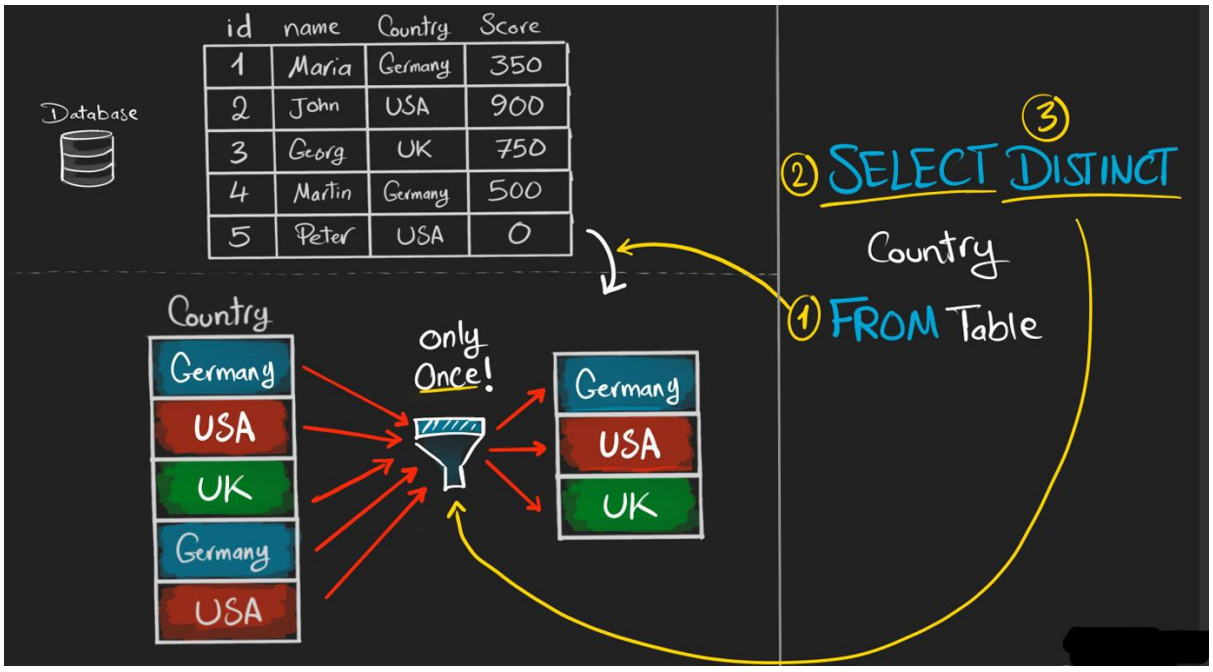




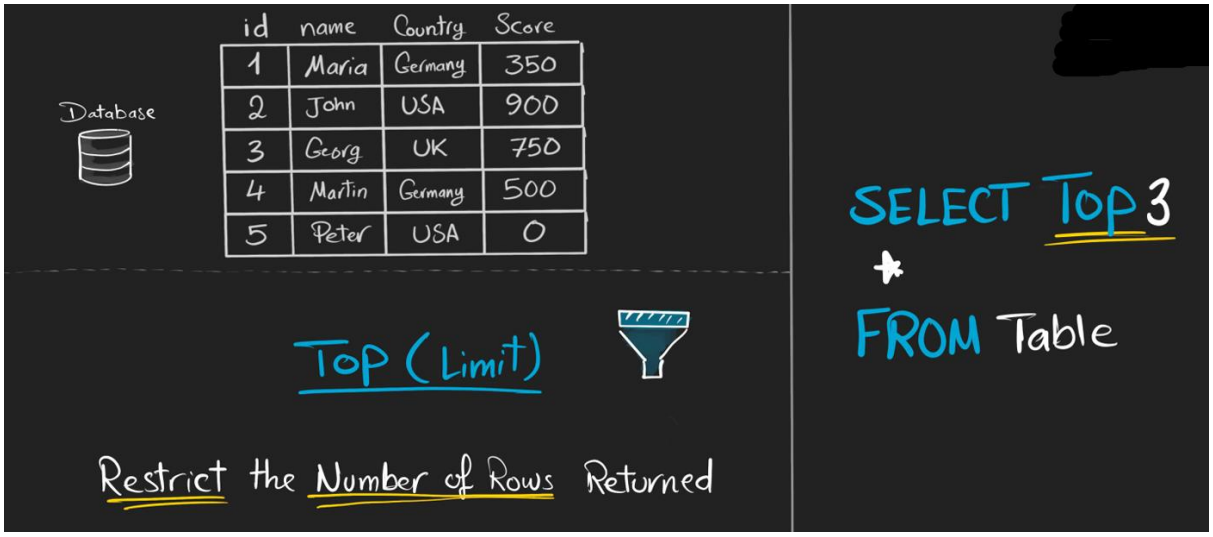


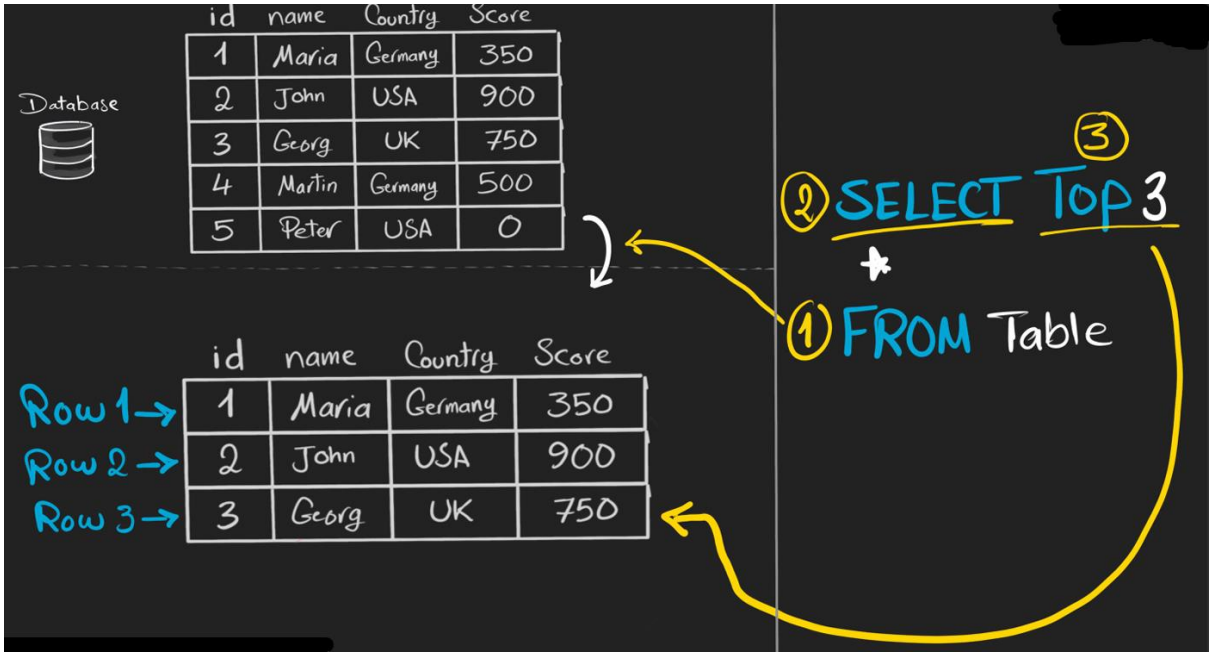
To select Distinct values



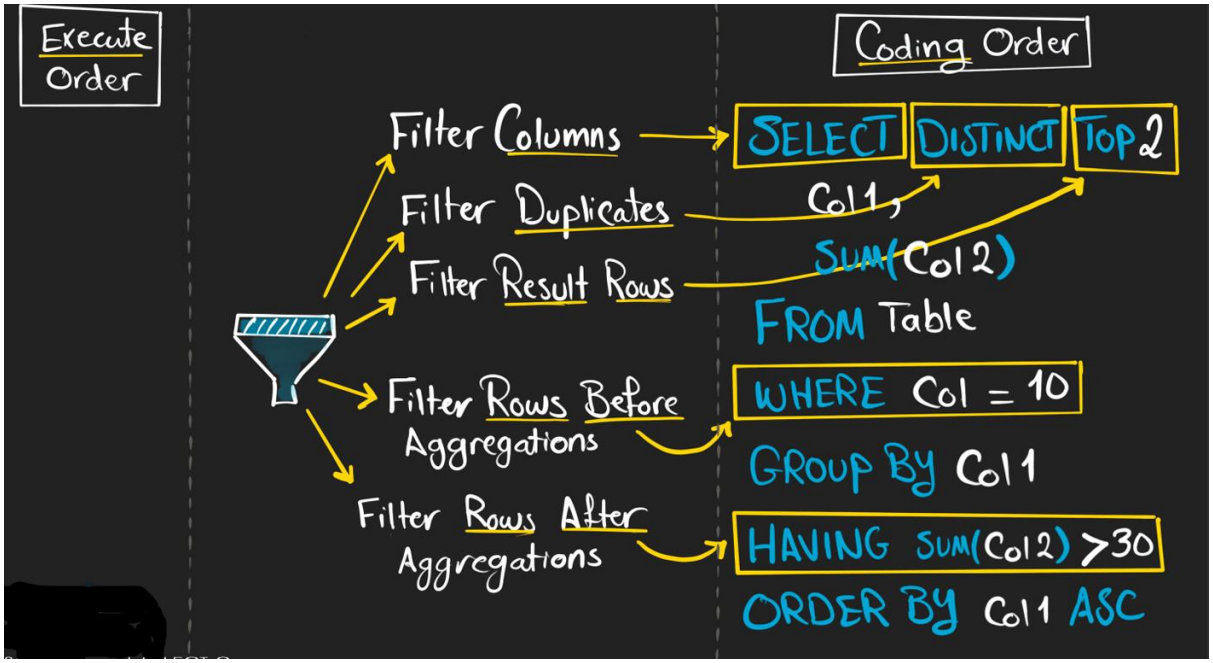


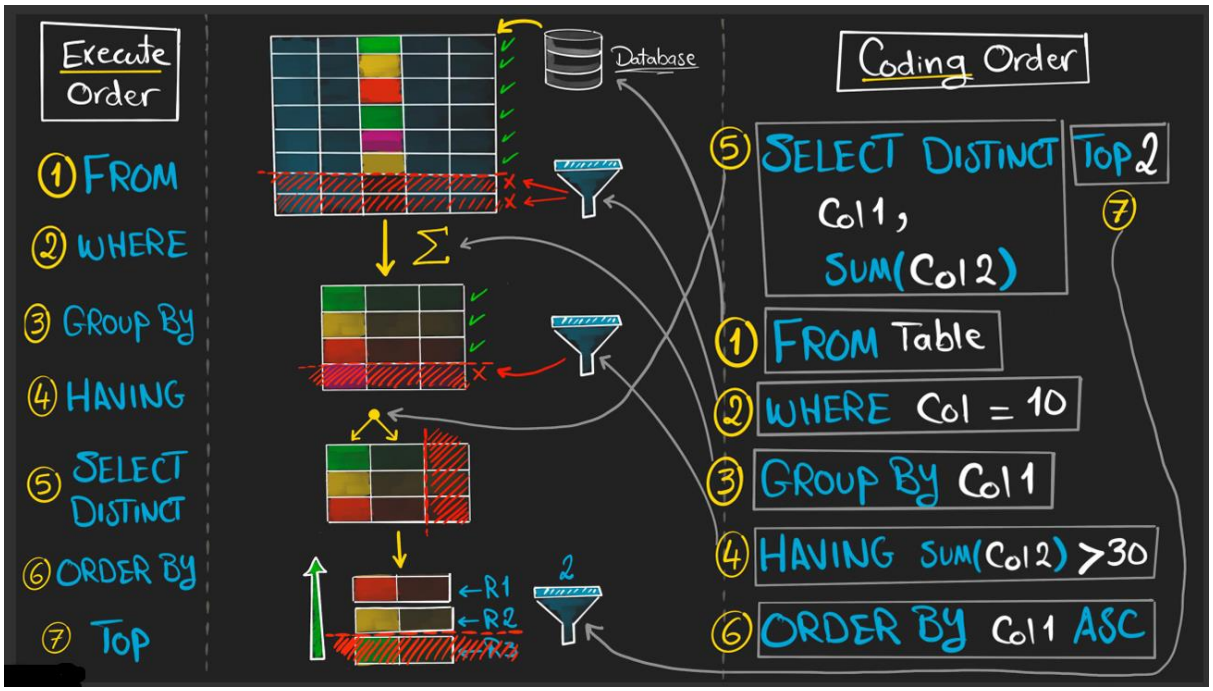
To select Top n rows



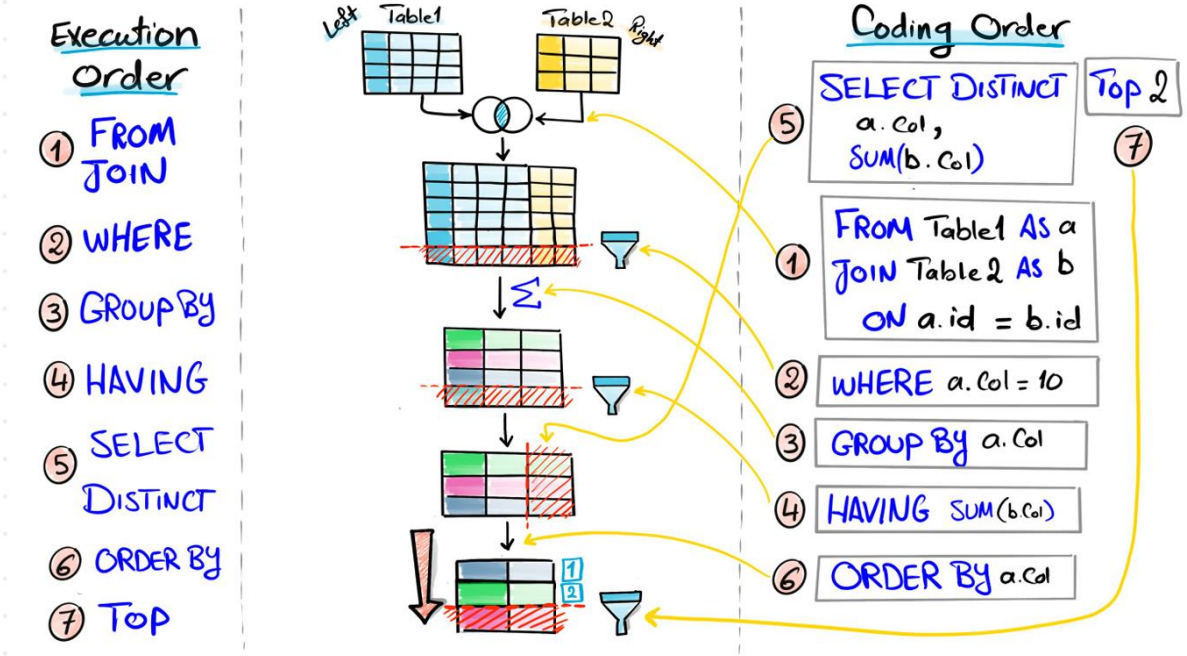


Execution order

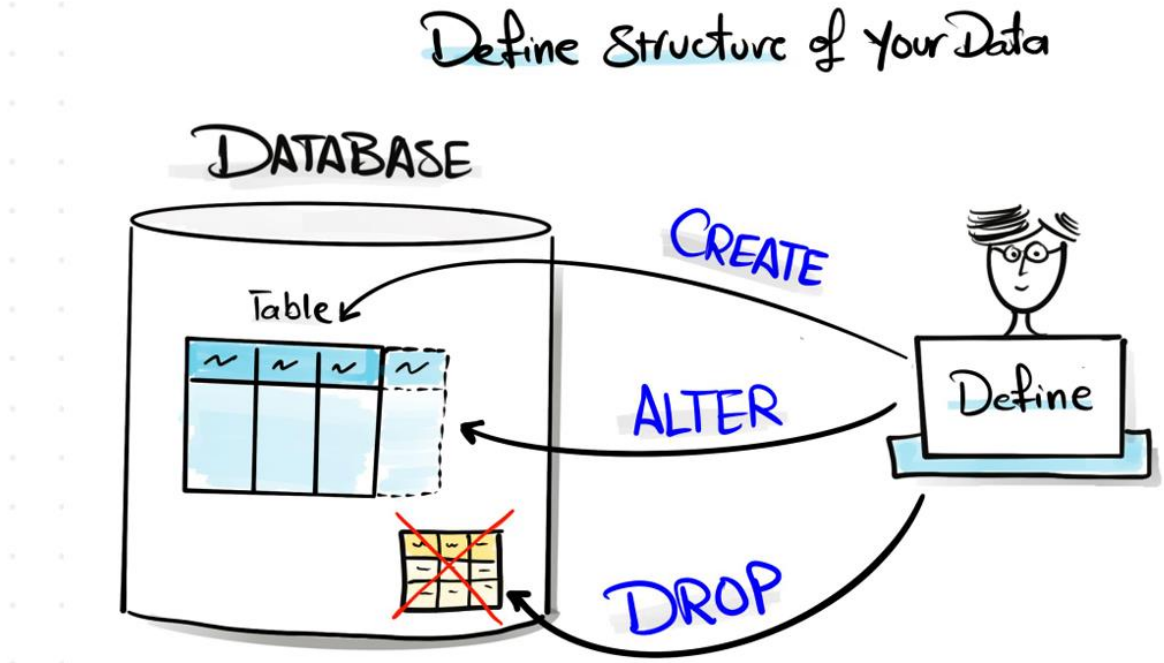




Complete execution order



Data Definition Language



Create a table

```
/* Create a new table called persons
with columns: id, person_name, birth_date, and phone */

CREATE TABLE persons (
    id INT NOT NULL,
    person_name VARCHAR(50) NOT NULL,
    birth_date DATE,
    phone VARCHAR(15) NOT NULL,
    CONSTRAINT pk_persons PRIMARY KEY (id)
)
```

Alter a table

1

```
-- Add a new column called email to the persons table

ALTER TABLE persons
ADD email VARCHAR(50) NOT NULL
```

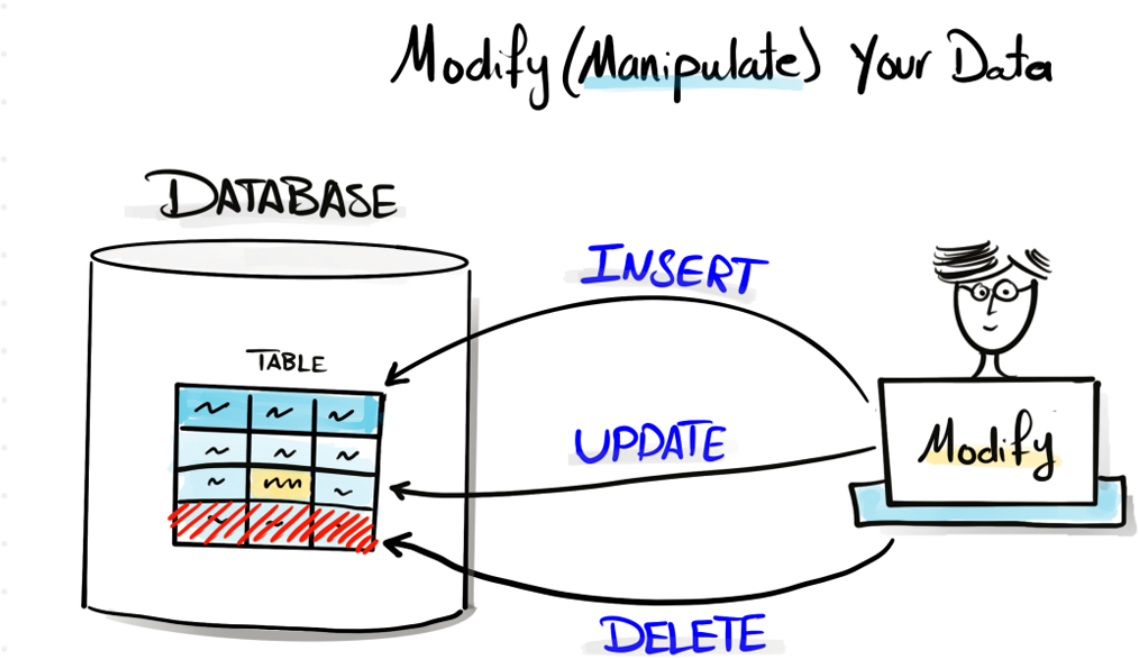
2

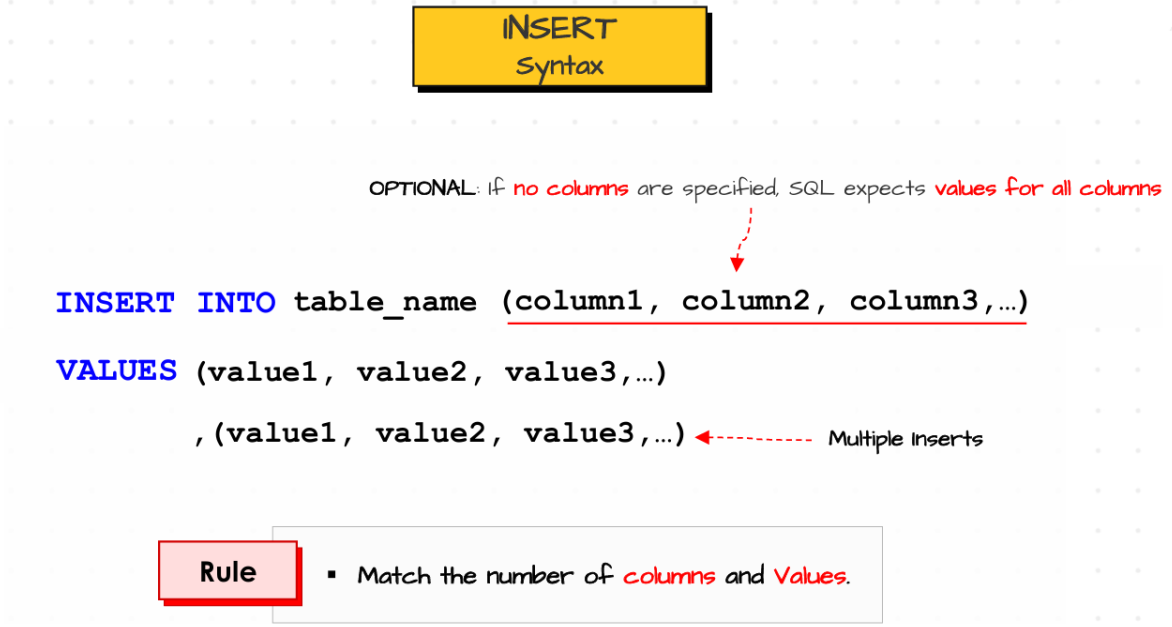
```
-- Remove the column phone from the persons table

ALTER TABLE persons
DROP COLUMN phone
```

Drop a table

```
-- Delete the table persons from the database

DROP TABLE persons
```

Data Manipulation Language





INSERT INTO employees (id, name, position, salary)

VALUES

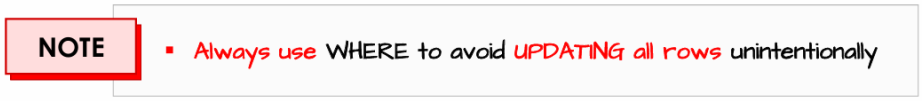
(1, 'Alice Johnson', 'Manager', 75000),

(2, 'Bob Smith', 'Developer', 60000),

(3, 'Charlie Brown', 'Analyst', 50000),

(4, 'Diana Green', 'Designer', 55000);


```
UPDATE table_name
   SET  column1 = value1,
        column2 = value2
 WHERE <condition>
```

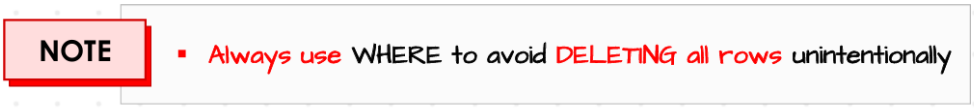


UPDATE employees

SET salary = 70000

WHERE name = 'Charlie Brown';


```
DELETE FROM table_name
WHERE <condition>
```



DELETE FROM employees

WHERE id = 3;