# SQL for Data Analysts


**By**

**TechHub**

# What’s inside

01 Introduction to sql

02 Importance and types of sql

03 Types of sql and Aggregate functions

04 Querying data commands and subqueries

# Introduction

SQL stands for Structured Query Language.
SQL is used to communicate with database in the form of queries.

- It is used to manage and interact with relational databases
- Think of it like the language you use to talk to a database to ask questions or update data.

Real-Life Example:
Imagine a library:
Books = Data
Library = Database
SQL = Librarian — who helps you find a book ,add a nem one, or remove an old one.

# What is SQL and MySQL??

# Importance Of SQL

1.**Manages data  in databases**

SQL helps you:

- Insert, update, delete, and retrieve data
- Keep data organized and accurate

2. **Quickly Finds Information**

- You can search for specific data using commands like SELECT, WHERE, etc.

  Example: Find all students who scored above 90 marks.

3. **Used in All Industries**

- Websites, mobile apps, banks, hospitals, e-commerce — all use SQL to handle data.

4.  **Required Skill for Jobs**

- SQL is a must-have skill for developers, data analysts, software engineers, and more.

# Types of SQL

SQL commands are grouped into 4 types:

- DDL - Data Definition Language
- DML - Data Manipulation Language
- DCL - Data Control Language
- TCL - Transaction Control Language

# Data Definition Language (DDL) Commands

| Command | Description | Syntax | Example |
|---|---|---|---|
| CREATE | The CREATE command creates a new database and objects, such as a table, index, view, or stored procedure. | `CREATE TABLE table_name (column1 datatype1, column2 datatype2, ...);` | `CREATE TABLE employees ( employee_id INT PRIMARY KEY, first_name VARCHAR(50), last_name VARCHAR(50), age INT );` |
| ALTER | The ALTER command adds, deletes, or modifies columns in an existing table. | `ALTER TABLE table_name ADD column_name datatype;` | `ALTER TABLE customers ADD email VARCHAR(100);` |
| DROP | The DROP command is used to drop an existing table in a database. | `DROP TABLE table_name;` | `DROP TABLE customers;` |
| TRUNCATE | The TRUNCATE command is used to delete the data inside a table, but not the table itself. | `TRUNCATE TABLE table_name;` | `TRUNCATE TABLE customers;` |

# Data Manipulation Language (DML) Commands

| Command | Description | Syntax | Example |
|---|---|---|---|
| SELECT | The SELECT command retrieves data from a database. | `SELECT column1, column2 FROM table_name;` | `SELECT first_name, last_name FROM customers;` |
| INSERT | The INSERT command adds new records to a table. | `INSERT INTO table_name (column1, column2) VALUES (value1, value2);` | `INSERT INTO customers (first_name, last_name) VALUES ('Mary', 'Doe');` |
| UPDATE | The UPDATE command is used to modify existing records in a table. | `UPDATE table_name SET column1 = value1, column2 = value2 WHERE condition;` | `UPDATE employees SET employee_name = 'John Doe', department = 'Marketing';` |
| DELETE | The DELETE command removes records from a table. | `DELETE FROM table_name WHERE condition;` | `DELETE FROM employees WHERE employee_name = 'John Doe';` |

## DCL

DCL is used for granting and revoking user access on a database

- Grrant - gives user's access Privileges to database
- Revoke -withdraw access privileges given with the Grant command

## TCL

Transactions control language has commands which are used to manage the transactions or the coduct of a database.

They manage the changes made by the DML statements and also group of statements into logical statement.

- COMMIT - It is used to save the transaction Permanently.
- SAVEPOINT- helps in identifying a point in the transaction, can be rolled back to the identified point
- ROllBACK- has feature of  restoring the database to genuine point, since from the last commit

# Aggregate Functions

There are five aggregate functions in SQL

**1.COUNT() -** Returns the number of rows

Ex: SELECT COUNT(*) FROM Students;

**2.SUM() -** Returns the total sum of a column.

Ex:  SELECT SUM(Marks) FROM Students;

**3.AVG() -** Returns the average value

Ex: SELECT AVG(Marks) FROM Students;

**4.MAX() -** Returns the highest value

Ex:  SELECT MAX(Marks) FROM Students;

**5.MIN() -** Returns the lowest value

Ex:  SELECT MIN(Marks) FROM Students;

# Querying Data Commands



| Command | Description | Syntax | Example |
| --- | --- | --- | --- |
| SELECT Statement | The SELECT statement is the primary command used to retrieve data from a database | `SELECT column1, column2 FROM table_name;` | `SELECT first_name, last_name FROM customers;` |
| WHERE Clause | The WHERE clause is used to filter rows based on a specified condition. | `SELECT * FROM table_name WHERE condition;` | `SELECT * FROM customers WHERE age > 30;` |
| ORDER BY Clause | The ORDER BY clause is used to sort the result set in ascending or descending order based on a specified column. | `SELECT * FROM table_name ORDER BY column_name ASC\|DESC;` | `SELECT * FROM products ORDER BY price DESC;` |
| GROUP BY Clause | The GROUP BY clause groups rows based on the values in a specified column. It is often used with aggregate functions like COUNT, SUM, AVG, etc. | `SELECT column_name, COUNT(*) FROM table_name GROUP BY column_name;` | `SELECT category, COUNT(*) FROM products GROUP BY category;` |
| HAVING Clause | The HAVING clause filters grouped results based on a specified condition. | `SELECT column_name, COUNT(*) FROM table_name GROUP BY column_name HAVING condition;` | `SELECT category, COUNT(*) FROM products GROUP BY category HAVING COUNT(*) > 5;` |

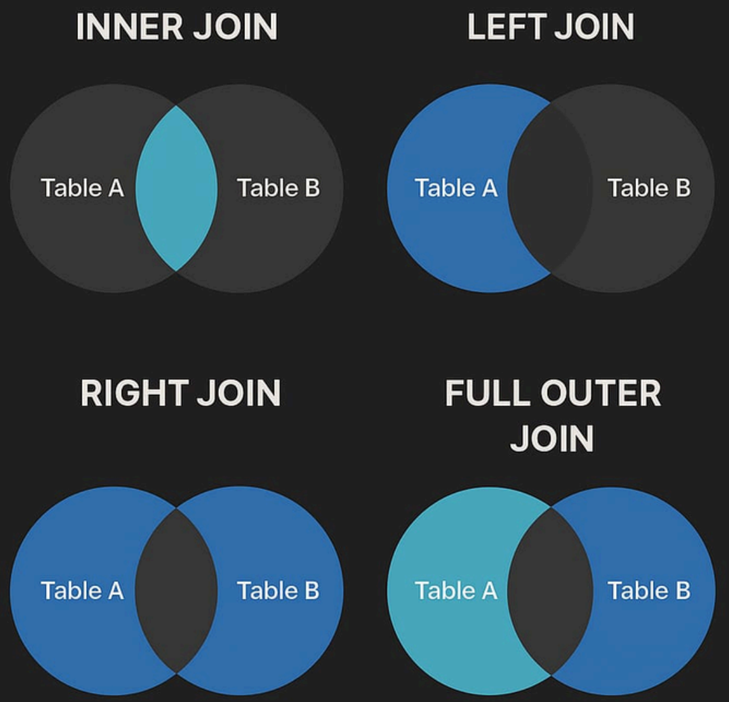
# JOINS in SQL

**What is joins in SQl?**

SQL joins combines rows from more than one table by using common columns in both the tables.

**There are 4 types of joins in SQL**:

| TYPE | USE CASES |
|---|---|
| 1. INNER JOIN | Matching rows in both tables |
| 2. LEFT JOIN | All from left + matches from right |
| 3. RIGHT JOIN | All from right + matches from left |
| 4. FULL JOIN | All records from both tables |



# Examples of SQL Quary



**INNER JOIN:**

```
SELECT c.name, o.order_id
FROM customers c
INNER JOIN orders o
ON c.id = o.customer_id;
```

**LEFT JOIN:**

```
SELECT c.name, o.order_id
FROM customers c
LEFT JOIN orders o
ON c.id = o.customer_id;
```

**RIGHT JOIN:**

```
SELECT c.name, o.order_id
FROM customers c
RIGHT JOIN orders o
ON c.id = o.customer_id;
```

**FULL JOIN:**

```
SELECT c.name, o.order_id
FROM customers c
FULL OUTER JOIN orders o
ON c.id = o.customer_id;
```

# Constraints in SQL

01 NOT NULL - Ensures that a column cannot have a NULL value

02 UNIQUE - Ensures that all values in a column are different

03 PRIMARY KEY - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table

04 FOREIGN KEY - Prevents actions that would destroy links between tablesFOREIGN KEY - Prevents actions that would destroy links between tables

05 CHECK - Ensures that the values in a column satisfies a specific condition

06 DEFAULT - Sets a default value for a column if no value is specified



01 **Primary key:**

```
CREATE TABLE employees (
    emp_id INT PRIMARY KEY,
    name VARCHAR(100),
    department VARCHAR(50)
);

✅ Works
INSERT INTO employees VALUES (1, 'Alice', 'IT');

 ✕ Error: Duplicate primary key
INSERT INTO employees VALUES (1, 'Bob', 'HR');

✕ Error: NULL not allowed
INSERT INTO employees VALUES (NULL, 'Eve',
'Finance');
```

02 **Foregin key:**

```
CREATE TABLE departments (
    dept_id INT PRIMARY KEY,
    dept_name VARCHAR(100)
);
CREATE TABLE employees (
    emp_id INT PRIMARY KEY,
    name VARCHAR(100),
    dept_id INT,
    FOREIGN KEY (dept_id) REFERENCES
departments(dept_id)
);
✅ Works (10 exists in departments)
INSERT INTO employees VALUES (1, 'Alice', 10);

✕ Error (50 does NOT exist in departments)
INSERT INTO employees VALUES (2, 'Bob', 50);
```

03

**NOT NULL:**

```
CREATE TABLE users (
    id INT PRIMARY KEY,
    username VARCHAR(50) NOT NULL
);
```

04

**UNIQUE:**

```
CREATE TABLE employees (
    emp_id INT PRIMARY KEY,
    email VARCHAR(100) UNIQUE
);
```

05

**CHECK:**

```
CREATE TABLE students (
    student_id INT PRIMARY KEY,
    age INT CHECK (age >= 18)
);
```

06

**DEFAULT:**

```
CREATE TABLE orders (
    order_id INT PRIMARY KEY,
    status VARCHAR(20) DEFAULT 'Pending’
);
```


TEHE.....
THANK YOU HONEY

# ANY QUERIES???