SQL Fundamentals: Data Types & Constraints

# Table of Contents

1. Data Types

1.1 Numeric Data Types

1.2 Character Data Types

1.3 Date/Time Data Types

1.4 Binary Data Types

1.5 NULL Values

1.6 Special Data Types

1.7 Choosing the Right Data Type

2. Database Constraints

3. Types of SQL Commands

3.1 Data Definition Language (DDL)

3.2 Data Manipulation Language (DML)

3.3 Data Control Language (DCL)

3.4 Transaction Control Language (TCL)

# 1. Data Types

Data types in SQL define the kind of values that can be stored in a column. They help maintain data integrity, ensure accuracy, and optimize performance. SQL data types are broadly categorized into Numeric, Character, Date/Time, Binary, and Special data types.

## 1.1 Numeric Data Types

Numeric types store numerical values, either as integers or with decimal points. They can be exact (like INT, DECIMAL) or approximate (like FLOAT, REAL).

• BIGINT – Large integers, e.g., 9781234567890 for ISBN numbers.

• INT – Standard integers, e.g., 5 for available copies.

• SMALLINT – Smaller integers, e.g., 4 for a rating out of 5.

• TINYINT – Very small integers (0–255), e.g., 30 for page count of a pamphlet.

• DECIMAL/NUMERIC – Fixed-point numbers, e.g., 19.99 for price.

**Approximate Numeric Types**

• FLOAT – Approximate floating-point numbers, e.g., 4.25 for average rating.

• REAL – Less precise floating-point numbers, e.g., 25.75 pages read per day.

## 1.2 Character Data Types

Character types store textual or alphanumeric data, including fixed and variable-length formats.

• CHAR – Fixed-length, non-Unicode, e.g., 'US' for country code.

• VARCHAR – Variable-length, non-Unicode characters, e.g., book titles.

• TEXT – Large variable-length non-Unicode character data for text, e.g., book synopsis.

• NCHAR – Fixed-length Unicode, e.g., titles in multiple languages.

• NVARCHAR – Variable-length Unicode, e.g., author names with special characters.

## 1.3 Date/Time Data Types

Allows working with dates, times, or both. These are crucial when working with temporal data, such as timestamp or historical records.

• DATE – Stores only date values, e.g., '1925-04-10'.

• TIME – Stores only time values, e.g., '15:00:00'.

• DATETIME – Stores both date and time, e.g., '2023-10-09 10:30:00'.

## 1.4 Binary Data Types

• BINARY – Fixed-length binary data, e.g., book cover images.

• VARBINARY – Variable-length binary data, e.g., audiobook samples.

## 1.5 NULL Values

NULL represents the absence of a value. It is distinct from zero or an empty string. Example: Author = NULL if the author is unknown.

## 1.6 Special Data Types

These special data types help accommodate unique data storage needs and improve data integrity.

• BLOB – Stores large binary data such as images, audio, or videos.

• ENUM – Restricts column values to a predefined single option list. E.g., Book Genre.

• SET – Allows storing multiple values from a predefined list. E.g., Multiple Authors.

## 1.7 Choosing the Right Data Type: A Balancing Act

Selecting the correct data type improves data integrity, storage efficiency, and performance. Additional tips for selecting data types includes:

* Considering the Nature of Data.
* Evaluating Data Size.
* Anticipating Future Needs.

# 2. Database Constraints

Constraints enforce rules to maintain accuracy and reliability in data.

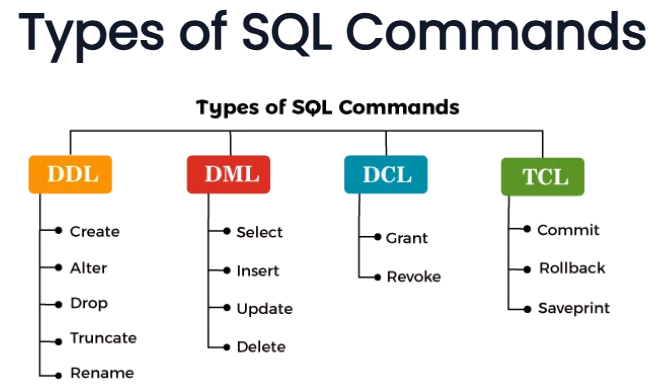
• PRIMARY KEY – Unique identifier for each records.

• AUTO\_INCREMENT – Automatically generates unique values for a column.

• NOT NULL – Ensures a column cannot have NULL values.

• UNIQUE – Prevents duplicate values in a column.

# 3. Types of SQL Commands



**3.1 Data Definition Language (DDL)**

**Definition**:  
DDL is a core subset of SQL used to define, create, alter, and remove database objects such as **tables, views, indexes, schemas, and users**. It deals with the *structure* of the database, not the actual data inside.

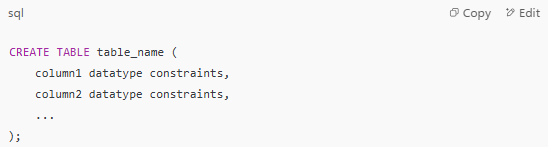
**Why it matters**:

* **Structure control** – DDL builds the foundation for how data is organized.
* **Data integrity** – Ensures the right data types and constraints are applied from the start.
* **Performance** – A well-structured schema makes queries faster and storage more efficient.

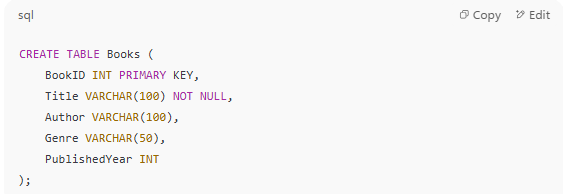
**Key DDL Commands**

**1. CREATE – Define new objects**

* **Purpose**: To create new tables, indexes, or other database objects.
* **Syntax**:



* **Example** (Library system):



* **Explanation**: This sets the blueprint for storing books. BookID is unique (PRIMARY KEY), Title cannot be NULL, and the rest are optional.

**2. ALTER – Modify an existing object**

* **Purpose**: To adjust a table's structure without dropping it.
* **Use cases**: Add/remove columns, change data types, rename columns.
* **Example**:



* **Note**: ALTER preserves existing data while changing structure.

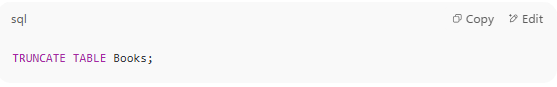
**3. DROP – Permanently delete an object**

* **Purpose**: Remove a table, view, or index permanently.
* **Caution**: Irreversible – all data is lost.



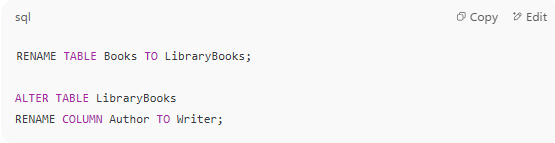
**4. TRUNCATE – Delete all rows but keep structure**

* **Purpose**: Quickly remove all data in a table without deleting the table.
* **Performance**: Faster than DELETE because it doesn’t log each row deletion.



**5. RENAME – Change object names**

* **Purpose**: Update names to reflect new use cases.



**3. 2 Data Manipulation Language (DML)**

**Definition**:  
DML is the part of SQL that deals with **managing the actual data** inside database tables.  
Where **DDL** defines the *structure* of the database, **DML** works *within* that structure to **add, modify, or remove data**.

**Key Points**:

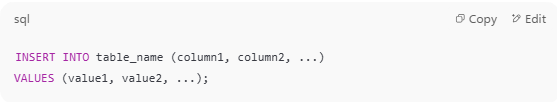
* DML statements usually affect **rows** inside tables.
* DML is **not automatically permanent** — changes only become final after a COMMIT (unless auto-commit is enabled).
* DML works only on **existing database objects** created by DDL.

**1. INSERT – Add new data**

**Purpose**: Adds new rows to a table.  
**Syntax**:



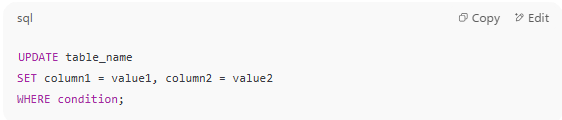
**Example**: Adding a new book record



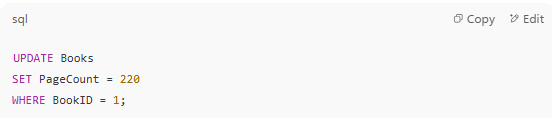
**Explanation**: A new row with all details of the book is inserted into the Books table.

**2. UPDATE – Modify existing data**

**Purpose**: Changes the values of one or more columns for specific rows.  
**Syntax**:



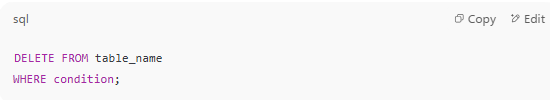
**Example**: Updating a book’s page count

****

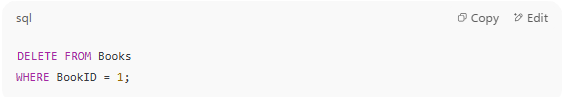
**Note**: Always use a **WHERE** clause; without it, all rows will be updated.

**3. DELETE – Remove existing data**

**Purpose**: Deletes rows from a table based on a condition.  
**Syntax**:



**Example**: Removing a discontinued book



**Difference from TRUNCATE**:

**DELETE TRUNCATE**

DELETE removes selected rows and can be rolled back;  
TRUNCATE removes **all** rows and cannot target specific rows.

**3.3 Data Control Language (DCL)**

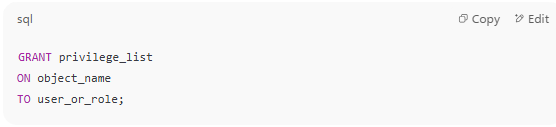
**Definition**:  
DCL is used to **control database access** by granting or removing permissions for users or roles.

**Key Points**:

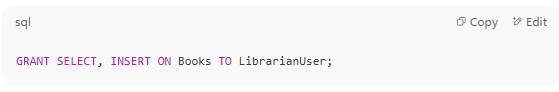
* Helps maintain **security** and **data privacy**.
* Works alongside **user accounts** created in the database.

**1. GRANT – Give permissions**

**Purpose**: Allows a user/role to perform certain actions on database objects.  
**Syntax**:



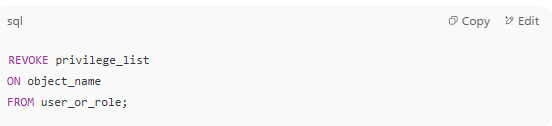
**Example**:



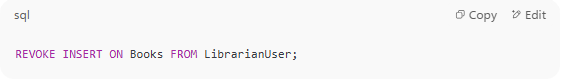
**Meaning**: LibrarianUser can read (SELECT) and add (INSERT) rows to the Books table.

**2. REVOKE – Remove permissions**

**Purpose**: Takes away previously granted privileges.  
**Syntax**:



**Example**:



**Meaning**: LibrarianUser can no longer insert new records into Books.

**3. 4 Transaction Control Language (TCL)**

**Definition**:  
TCL manages **transactions** — sequences of one or more SQL operations treated as a **single logical unit**.

**Why important**:  
Ensures **data consistency** — either all changes happen, or none do.

**1. COMMIT – Make changes permanent**

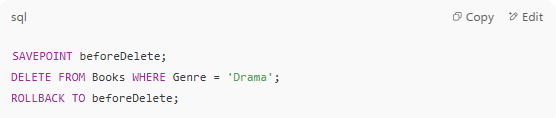
* Saves all changes since the last COMMIT or ROLLBACK.
* Cannot be undone after execution.

**2. ROLLBACK – Undo changes**

* Reverts the database to its state before the last COMMIT.
* Useful when an error occurs during a transaction.

**3. SAVEPOINT – Partial rollback marker**

**Example**:



* Allows undoing part of a transaction without rolling back everything.