Table of Contents

What is SQL? 2

What Can SQL Do? 2

RDBMS (Relational Database Management System) 2

Key Terms: 2

Types of SQL Commands 3

Command Details 3

Commands with syntax 4

Basic SQL Syntax Rules: 5

Common Clauses in SQL: 5

Joins in SQL: 5

Example – Creating a Table 5

What is SQL?

SQL (Structured Query Language) is a standard programming language used to manage and interact with relational databases.

It allows you to create, modify, retrieve, and manage data efficiently.

- Standardized by ANSI in 1986 and ISO in 1987.

- Works with all major RDBMS platforms like MySQL, Oracle, SQL Server, PostgreSQL, and MS Access.

What Can SQL Do?

- Execute queries against a database.

- Retrieve data from a database.

- Insert new records.

- Update existing records.

- Delete records.

- Create new databases.

- Create tables, views, and stored procedures.

- Set permissions on database objects.

RDBMS (Relational Database Management System)

- The foundation for SQL and modern databases.

- Stores data in tables (rows and columns).

Examples: MySQL, Oracle, SQL Server, IBM DB2, MS Access.

Key Terms:

- Table – A collection of related data entries (rows and columns).

- Field – A column in a table (e.g., CustomerName).

- Record (Row) – A single data entry in a table.

- Column – A vertical set of values for a specific field.

Types of SQL Commands

Category	Full Form	Purpose	Examples
DDL	Data Definition Language	Defines and manages database structure	CREATE, ALTER, DROP, TRUNCATE
DML	Data Manipulation Language	Manages data inside tables	INSERT, UPDATE, DELETE
DCL	Data Control Language	Controls access to data	GRANT, REVOKE
TCL	Transaction Control Language	Manages transactions	COMMIT, ROLLBACK, SAVEPOINT
DQL	Data Query Language	Retrieves data	SELECT

Command Details

1. DDL (Data Definition Language)

- CREATE – Creates a new table, view, or database object.

- ALTER – Modifies an existing database object.

- DROP – Deletes a table, view, or other object.

- TRUNCATE – Removes all data from a table but keeps the structure.

2. DML (Data Manipulation Language)

- INSERT – Adds new records.

- UPDATE – Changes existing records.

- DELETE – Removes records.

3. DCL (Data Control Language)

- GRANT – Gives user permissions.

- REVOKE – Removes user permissions.

4. TCL (Transaction Control Language)

- COMMIT – Saves all changes permanently.

- ROLLBACK – Undoes changes since the last commit.

- SAVEPOINT – Marks a point in a transaction to roll back to.

5. DQL (Data Query Language)

- SELECT – Retrieves specific data from one or more tables.

Commands with syntax

Category	Command	Purpose	Syntax
DDL (Data Definition Language)	**CREATE**	Creates a new table, view, or database object	`CREATE TABLE table_name ( column1 datatype, column2 datatype, ... );`
	ALTER	Modifies an existing database object	`ALTER TABLE table_name ADD column_name datatype; ALTER TABLE table_name MODIFY column_name new_datatype; ALTER TABLE table_name DROP COLUMN column_name;`
	DROP	Deletes a table, view, or other object	`DROP TABLE table_name;`
	TRUNCATE	Removes all data from a table but keeps the structure	`TRUNCATE TABLE table_name;`
DML (Data Manipulation Language)	**INSERT**	Adds new records	`INSERT INTO table_name (column1, column2, ...) VALUES (value1, value2, ...);`
	UPDATE	Changes existing records	`UPDATE table_name SET column1 = value1, column2 = value2 WHERE condition;`
	DELETE	Removes records	`DELETE FROM table_name WHERE condition;`
DCL (Data Control Language)	**GRANT**	Gives user permissions	`GRANT privilege_name ON object_name TO user_name;`
	REVOKE	Removes user permissions	`REVOKE privilege_name ON object_name FROM user_name;`
TCL (Transaction Control Language)	**COMMIT**	Saves all changes permanently	`COMMIT;`
	ROLLBACK	Undoes changes since the last commit	`ROLLBACK;`
	SAVEPOINT	Marks a point in a transaction to roll	`SAVEPOINT savepoint_name; ROLLBACK TO savepoint_name;`

Category	Command	Purpose	Syntax
		back to	
DQL (Data Query Language)	**SELECT**	Retrieves specific data from one or more tables	`SELECT column1, column2, ... FROM table_name WHERE condition ORDER BY column_name ASC

Basic SQL Syntax Rules:

- SQL keywords are not case-sensitive (SELECT is the same as select).
- Statements end with a semicolon (;).
- Strings are enclosed in single quotes (' ').
- Column and table names should not have spaces; use underscores if needed.

Common Clauses in SQL:

- WHERE – Filters records based on a condition.
- ORDER BY – Sorts results in ascending (ASC) or descending (DESC) order.
- GROUP BY – Groups rows that have the same values in specified columns.
- HAVING – Filters groups after grouping.
- DISTINCT – Returns only unique values.

Joins in SQL:

- INNER JOIN – Returns records with matching values in both tables.
- LEFT JOIN – Returns all records from the left table and matching records from the right table.
- RIGHT JOIN – Returns all records from the right table and matching records from the left table.
- FULL JOIN – Returns all records when there is a match in either table.

Example – Creating a Table

```
CREATE TABLE Customers (
    CustomerID INT PRIMARY KEY,
    CustomerName VARCHAR(100),
    ContactName VARCHAR(100),
    Address VARCHAR(255),
    City VARCHAR(50),
    PostalCode VARCHAR(20),
    Country VARCHAR(50)
);
This table will be created
```

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	John Doe Ltd.	John Doe	123 Main Street	New York	10001	USA
2	Sunrise Traders	A. Khan	45 Market Road	Mumbai	400001	India
3	Green Valley Farms	S. Smith	78 Country Lane	Sydney	2000	

```
Example – Inserting Data
INSERT INTO Customers (CustomerID, CustomerName, ContactName, Address, City,
PostalCode, Country)
VALUES (1, 'John Doe', 'John', '123 Main St', 'New York', '10001', 'USA');

Example – Selecting Data
SELECT CustomerName, City FROM Customers WHERE Country = 'USA';

Example – Updating Data
UPDATE Customers SET City = 'Los Angeles' WHERE CustomerID = 1;

Example – Deleting Data
DELETE FROM Customers WHERE CustomerID = 1;

Best Practices:
- Always back up your database before making major changes by commit or
savepoint.
- Use WHERE clauses with UPDATE and DELETE to avoid affecting all rows.
- Use proper indexing to speed up queries.
- Avoid SELECT * in production queries; specify only needed columns.
- Normalize data to reduce redundancy.
```