# BASICS CONCEPTS OF SQL

SQL (Structured Query Language) is the standard language used to manage and manipulate relational databases. Below is a complete overview of the key concepts in SQL:

---

**1. Basics of SQL**

- **Relational Databases**: Data is stored in tables (rows & columns).
- **Tables**: Collections of rows (records) and columns (fields).
- **Case-Insensitive**: SQL keywords (e.g., SELECT, INSERT) are not case-sensitive.

---

**2. SQL Commands (Types)**

SQL commands are divided into the following categories:

**a. Data Definition Language (DDL)**

Defines the structure of the database.

- **CREATE**: Create database objects (tables, indexes, etc.).

```
CREATE TABLE table_name (
    column1 datatype,
    column2 datatype,
    ...);
```

- **ALTER**: Modify an existing database structure.

```
ALTER TABLE table_name ADD column_name datatype;
```

- **DROP**: Delete tables or databases.

```
DROP TABLE table_name;
```

- **TRUNCATE**: Remove all rows from a table (resets identity columns).

```
TRUNCATE TABLE table_name;
```

**b. Data Manipulation Language (DML)**

Manipulates data in tables.

- **INSERT**: Add new records.

INSERT INTO table_name (column1, column2) VALUES (value1, value2);

- **UPDATE**: Modify existing records.

  UPDATE table_name SET column1 = value WHERE condition;

- **DELETE**: Remove records.

  DELETE FROM table_name WHERE condition;

**c. Data Query Language (DQL)**

Used to query data.

- **SELECT**: Retrieve data.

  SELECT column1, column2 FROM table_name WHERE condition;

**d. Data Control Language (DCL)**

Control permissions and access.

- **GRANT**: Provide access.

  GRANT SELECT ON table_name TO user;

- **REVOKE**: Remove access.

  REVOKE SELECT ON table_name FROM user;

**e. Transaction Control Language (TCL)**

Manage database transactions.

- **COMMIT**: Save changes permanently.

  COMMIT;

- **ROLLBACK**: Revert changes.

  ROLLBACK;

- **SAVEPOINT**: Set a savepoint within a transaction.

  SAVEPOINT savepoint_name;

---

**3. SQL Clauses**

Clauses are used to filter, group, and manipulate query results.

- **WHERE**: Filter rows based on a condition.

  SELECT * FROM table_name WHERE column = value;

- **ORDER BY**: Sort rows in ascending/descending order.

  SELECT * FROM table_name ORDER BY column ASC;

- **GROUP BY**: Group rows based on a column.

  SELECT column, COUNT(*) FROM table_name GROUP BY column;

- **HAVING**: Filter grouped rows.

  SELECT column, COUNT(*) FROM table_name GROUP BY column HAVING COUNT(*) > 5;

---

**4. SQL Operators**

Used to perform operations in queries.

- **Comparison**: =, <>, <, >, <=, >=
- **Logical**: AND, OR, NOT
- **Arithmetic**: +, -, *, /, %
- **IN**: Check if a value matches a list.
- **LIKE**: Pattern matching (e.g., % for any characters, _ for a single character).

  SELECT * FROM table_name WHERE column LIKE 'A%';

- **BETWEEN**: Select range of values.

  SELECT * FROM table_name WHERE column BETWEEN 10 AND 20;

---

**5. Joins in SQL**

Combining data from multiple tables:

- **INNER JOIN**: Matches rows in both tables.

  SELECT * FROM table1 INNER JOIN table2 ON table1.column = table2.column;

- **LEFT JOIN**: Includes all rows from the left table.
- **RIGHT JOIN**: Includes all rows from the right table.
- **FULL OUTER JOIN**: Combines LEFT and RIGHT joins.
- **SELF JOIN**: A table joined with itself.
- **CROSS JOIN**: Cartesian product of two tables.

---

**6. Aggregate Functions**

Perform calculations on data:

- **COUNT**: Count rows.
- **SUM**: Total of a column.
- **AVG**: Average of a column.
- **MAX**: Maximum value.
- **MIN**: Minimum value.

---

**7. Subqueries**

Nested queries:

- Used within SELECT, WHERE, or FROM.

SELECT * FROM table_name WHERE column = (SELECT MAX(column) FROM table_name);

---

**8. Indexes**

Used to speed up queries by creating a sorted structure:

CREATE INDEX index_name ON table_name (column_name);

---

**9. Views**

Virtual tables based on queries:

CREATE VIEW view_name AS SELECT column1, column2 FROM table_name;

---

**10. Stored Procedures and Functions**

Reusable SQL code:

- **Stored Procedure**:

```
CREATE PROCEDURE procedure_name()
BEGIN
   SQL statements;
END;
```

- **Function**:

```
CREATE FUNCTION function_name(parameters) RETURNS datatype AS
BEGIN
   RETURN value;
END;
```

---

**11. Triggers**

Automatic actions triggered by events:

```
CREATE TRIGGER trigger_name
AFTER INSERT ON table_name
FOR EACH ROW
BEGIN
   SQL statements;
END;
```

---

**12. SQL Best Practices**

- Use proper indexing for large datasets.
- Normalize tables to reduce redundancy.
- Use meaningful table and column names.
- Avoid using SELECT *; specify columns.
- Regularly back up databases.