1. **SQL:**

SQL is Structured Query Language, which is a computer language for storing, manipulating and retrieving data stored in a relational database.

SQL is the standard language for Relational Database System.

### **DDL - Data Definition Language**

|  |  |
| --- | --- |
| **Sr.No.** | **Command & Description** |
| 1 | **CREATE**  Creates a new table, a view of a table, or other object in the database. |
| 2 | **ALTER**  Modifies an existing database object, such as a table. |
| 3 | **DROP**  Deletes an entire table, a view of a table or other objects in the database. |

### **DML - Data Manipulation Language**

|  |  |
| --- | --- |
| **Sr.No.** | **Command & Description** |
| 1 | **SELECT**  Retrieves certain records from one or more tables. |
| 2 | **INSERT**  Creates a record. |
| 3 | **UPDATE**  Modifies records. |
| 4 | **DELETE**  Deletes records. |

### **DCL - Data Control Language**

|  |  |
| --- | --- |
| **Sr.No.** | **Command & Description** |
| 1 | **GRANT**  Gives a privilege to user. |
| 2 | **REVOKE**  Takes back privileges granted from user. |

* 1. **Constraints:**

Following are some of the most commonly used constraints available in SQL −

* NOT NULL Constraint − Ensures that a column cannot have a NULL value.
* DEFAULT Constraint − Provides a default value for a column when none is specified.
* UNIQUE Constraint − Ensures that all the values in a column are different.
* PRIMARY Key − Uniquely identifies each row/record in a database table.
* FOREIGN Key − Uniquely identifies a row/record in any another database table.
* CHECK Constraint − The CHECK constraint ensures that all values in a column satisfy certain conditions.
* INDEX − Used to create and retrieve data from the database very quickly.
  1. **SYNTAX:**

### **SQL SELECT Statement**

SELECT column1, column2....columnN

FROM table\_name;

### **SQL DISTINCT Clause**

SELECT DISTINCT column1, column2....columnN

FROM table\_name;

### **SQL WHERE Clause**

SELECT column1, column2....columnN

FROM table\_name

WHERE CONDITION;

### **SQL AND/OR Clause**

SELECT column1, column2....columnN

FROM table\_name

WHERE CONDITION-1 {AND|OR} CONDITION-2;

### **SQL IN Clause**

SELECT column1, column2....columnN

FROM table\_name

WHERE column\_name IN (val-1, val-2,...val-N);

### **SQL BETWEEN Clause**

SELECT column1, column2....columnN

FROM table\_name

WHERE column\_name BETWEEN val-1 AND val-2;

### **SQL LIKE Clause**

SELECT column1, column2....columnN

FROM table\_name

WHERE column\_name LIKE { PATTERN };

### **SQL ORDER BY Clause**

SELECT column1, column2....columnN

FROM table\_name

WHERE CONDITION

ORDER BY column\_name {ASC|DESC};

### **SQL GROUP BY Clause**

SELECT SUM(column\_name)

FROM table\_name

WHERE CONDITION

GROUP BY column\_name;

### **SQL COUNT Clause**

SELECT COUNT(column\_name)

FROM table\_name

WHERE CONDITION;

### **SQL HAVING Clause**

SELECT SUM(column\_name)

FROM table\_name

WHERE CONDITION

GROUP BY column\_name

HAVING (arithematic function condition);

### **SQL CREATE TABLE Statement**

CREATE TABLE table\_name(

column1 datatype,

column2 datatype,

column3 datatype,

.....

columnN datatype,

PRIMARY KEY( one or more columns )

);

### **SQL DROP TABLE Statement**

DROP TABLE table\_name;

### **SQL CREATE INDEX Statement**

CREATE UNIQUE INDEX index\_name

ON table\_name ( column1, column2,...columnN);

### **SQL DROP INDEX Statement**

ALTER TABLE table\_name

DROP INDEX index\_name;

### **SQL DESC Statement**

DESC table\_name;

### **SQL TRUNCATE TABLE Statement**

TRUNCATE TABLE table\_name;

### **SQL ALTER TABLE Statement**

ALTER TABLE table\_name {ADD|DROP|MODIFY} column\_name {data\_ype};

### **SQL ALTER TABLE Statement (Rename)**

ALTER TABLE table\_name RENAME TO new\_table\_name;

### **SQL INSERT INTO Statement**

INSERT INTO table\_name( column1, column2....columnN)

VALUES ( value1, value2....valueN);

### **SQL UPDATE Statement**

UPDATE table\_name

SET column1 = value1, column2 = value2....columnN=valueN

[ WHERE CONDITION ];

### **SQL DELETE Statement**

DELETE FROM table\_name

WHERE {CONDITION};

### **SQL CREATE DATABASE Statement**

CREATE DATABASE database\_name;

### **SQL DROP DATABASE Statement**

DROP DATABASE database\_name;

### **SQL USE Statement**

USE database\_name;

### **SQL COMMIT Statement**

COMMIT;

### **SQL ROLLBACK Statement**

ROLLBACK;

## SQL Logical Operators

Here is a list of all the logical operators available in SQL.

[Show Examples](https://www.tutorialspoint.com/sql/sql-logical-operators.htm)

|  |  |
| --- | --- |
| **Sr.No.** | **Operator & Description** |
| 1 | **ALL**  The ALL operator is used to compare a value to all values in another value set. |
| 2 | **AND**  The AND operator allows the existence of multiple conditions in an SQL statement's WHERE clause. |
| 3 | **ANY**  The ANY operator is used to compare a value to any applicable value in the list as per the condition. |
| 4 | **BETWEEN**  The BETWEEN operator is used to search for values that are within a set of values, given the minimum value and the maximum value. |
| 5 | **EXISTS**  The EXISTS operator is used to search for the presence of a row in a specified table that meets a certain criterion. |
| 6 | **IN**  The IN operator is used to compare a value to a list of literal values that have been specified. |
| 7 | **LIKE**  The LIKE operator is used to compare a value to similar values using wildcard operators. |
| 8 | **NOT**  The NOT operator reverses the meaning of the logical operator with which it is used. Eg: NOT EXISTS, NOT BETWEEN, NOT IN, etc. **This is a negate operator.** |
| 9 | **OR**  The OR operator is used to combine multiple conditions in an SQL statement's WHERE clause. |
| 10 | **IS NULL**  The NULL operator is used to compare a value with a NULL value. |
| 11 | **UNIQUE**  The UNIQUE operator searches every row of a specified table for uniqueness (no duplicates). |

* 1. Create Table syntax:

The basic syntax of the CREATE TABLE statement is as follows −

CREATE TABLE table\_name(

column1 datatype,

column2 datatype,

column3 datatype,

.....

columnN datatype,

PRIMARY KEY( one or more columns )

);

The **SQL INSERT INTO** syntax will be as follows −

INSERT INTO TABLE\_NAME VALUES (value1,value2,value3,...valueN);

The basic syntax of the UPDATE query with a WHERE clause is as follows −

UPDATE table\_name

SET column1 = value1, column2 = value2...., columnN = valueN

WHERE [condition];

DELETE FROM table\_name

WHERE [condition];

The basic syntax of % and \_ is as follows −

SELECT FROM table\_name

WHERE column LIKE 'XXXX%'

or

SELECT FROM table\_name

WHERE column LIKE '%XXXX%'

or

SELECT FROM table\_name

WHERE column LIKE 'XXXX\_'

or

SELECT FROM table\_name

WHERE column LIKE '\_XXXX'

or

SELECT FROM table\_name

WHERE column LIKE '\_XXXX\_'

The basic syntax of the TOP clause with a SELECT statement would be as follows.

SELECT TOP number|percent column\_name(s)

FROM table\_name

WHERE [condition]

GROUP BY clause must follow the conditions in the WHERE clause and must precede the ORDER BY clause if one is used.

SELECT column1, column2

FROM table\_name

WHERE [ conditions ]

GROUP BY column1, column2

ORDER BY column1, column2

The basic syntax of DISTINCT keyword to eliminate the duplicate records is as follows −

SELECT DISTINCT column1, column2,.....columnN

FROM table\_name

WHERE [condition]

**JOINS:**

There are different types of joins available in SQL −

* [INNER JOIN](https://www.tutorialspoint.com/sql/sql-inner-joins.htm) − returns rows when there is a match in both tables.
* [LEFT JOIN](https://www.tutorialspoint.com/sql/sql-left-joins.htm) − returns all rows from the left table, even if there are no matches in the right table.
* [RIGHT JOIN](https://www.tutorialspoint.com/sql/sql-right-joins.htm) − returns all rows from the right table, even if there are no matches in the left table.
* [FULL JOIN](https://www.tutorialspoint.com/sql/sql-full-joins.htm) − returns rows when there is a match in one of the tables.
* [SELF JOIN](https://www.tutorialspoint.com/sql/sql-self-joins.htm) − is used to join a table to itself as if the table were two tables, temporarily renaming at least one table in the SQL statement.
* [CARTESIAN JOIN](https://www.tutorialspoint.com/sql/sql-cartesian-joins.htm) − returns the Cartesian product of the sets of records from the two or more joined tables.

INDEX:

### **Single-Column Indexes**

A single-column index is created based on only one table column. The basic syntax is as follows.

CREATE INDEX index\_name

ON table\_name (column\_name);

### **Unique Indexes**

Unique indexes are used not only for performance, but also for data integrity. A unique index does not allow any duplicate values to be inserted into the table. The basic syntax is as follows.

CREATE UNIQUE INDEX index\_name

on table\_name (column\_name);

### **Composite Indexes**

A composite index is an index on two or more columns of a table. Its basic syntax is as follows.

CREATE INDEX index\_name

on table\_name (column1, column2);

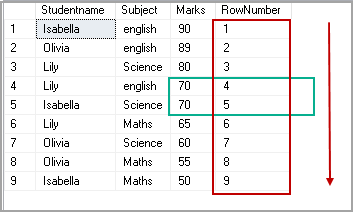
### **Implicit Indexes**

Implicit indexes are indexes that are automatically created by the database server when an object is created. Indexes are automatically created for primary key constraints and unique constraints.

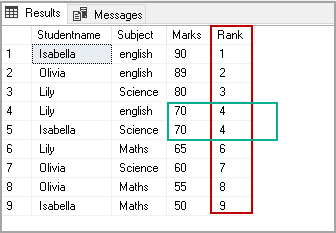
**RANK:**

We have the following rank functions.

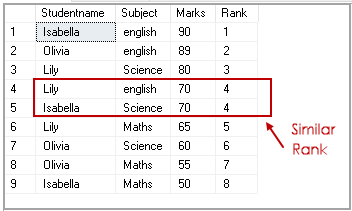
* ROW\_NUMBER() -ROW\_Number() SQL RANK function to get a unique sequential number for each row



* RANK()



* DENSE\_RANK() -  if we have duplicate values, SQL assigns different ranks to those rows as well. Ideally, we should get the same rank for duplicate or similar values.



* NTILE()

### **Using SQL Server RANK() function over a result set example**

SELECT

product\_id,

product\_name,

list\_price,

RANK () OVER (

ORDER BY list\_price DESC

) price\_rank

FROM

production.products;

Here is the result set:

