What Can SQL do?

- SQL can execute queries against a database
- SQL can retrieve data from a database
- SQL can insert records in a database
- SQL can update records in a database
- SQL can delete records from a database
- SQL can create new databases
- SQL can create new tables in a database
- SQL can create stored procedures in a database
- SQL can create views in a database
- SQL can set permissions on tables, procedures, and views

Some of The Most Important SQL Commands

- **SELECT** - extracts data from a database
- **UPDATE** - updates data in a database
- **DELETE** - deletes data from a database
- **INSERT INTO** - inserts new data into a database
- **CREATE DATABASE** - creates a new database
- **ALTER DATABASE** - modifies a database
- **CREATE TABLE** - creates a new table
- **ALTER TABLE** - modifies a table
- **DROP TABLE** - deletes a table
- **CREATE INDEX** - creates an index (search key)
- **DROP INDEX** - deletes an index

Operators in The WHERE Clause

The following operators can be used in the WHERE clause:

Operator	Description	Example
=	Equal	Try it
>	Greater than	Try it
<	Less than	Try it
>=	Greater than or equal	Try it
<=	Less than or equal	Try it
<>	Not equal. **Note:** In some versions of SQL this operator may be written as !=	Try it
BETWEEN	Between a certain range	Try it
LIKE	Search for a pattern	Try it
IN	To specify multiple possible values for a column	

```
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

```
DELETE FROM table_name  WHERE condition;
```

SQL Server / MS Access Syntax:

```
SELECT  TOP number|percent column_name(s)
FROM table_name
WHERE condition;
```

MySQL Syntax:

```
SELECT  column_name(s)
FROM table_name
WHERE condition
LIMIT number;
```

Oracle Syntax:

```
SELECT  column_name(s)
FROM table_name
WHERE ROWNUM <= number;
```

MIN() Syntax

```
SELECT  MIN(column_name)
FROM table_name
WHERE condition;
```

MAX() Syntax

```
SELECT  MAX(column_name)
FROM table_name
WHERE condition;
```

```
SELECT  COUNT(column_name)
FROM table_name
WHERE condition;
```

AVG() Syntax

```
SELECT  AVG(column_name)
FROM table_name
WHERE condition;
```

SUM() Syntax

```
SELECT  SUM(column_name)
FROM table_name
WHERE condition;
```

The SQL LIKE Operator

The LIKE operator is used in a WHERE clause to search for a specified pattern in a column.

There are two wildcards often used in conjunction with the LIKE operator:

- % - The percent sign represents zero, one, or multiple characters
- _ - The underscore represents a single character

Note: MS Access uses an asterisk (*) instead of the percent sign (%), and a question mark (?) instead of the underscore (_).

The percent sign and the underscore can also be used in combinations!

LIKE Syntax

```
SELECT column1, column2, ...
FROM table_name
WHERE columnN LIKE pattern;
```

The SQL IN Operator

The IN operator allows you to specify multiple values in a WHERE clause.

The IN operator is a shorthand for multiple OR conditions.

IN Syntax

```
SELECT column_name(s)
FROM table_name
WHERE column_name IN (value1, value2, ...);
```

or:

```
SELECT column_name(s)
FROM table_name
WHERE column_name IN (SELECT STATEMENT);
```

Alias Column Syntax

```
SELECT column_name AS alias_name
FROM table_name;
```

The SQL BETWEEN Operator

The BETWEEN operator selects values within a given range. The values can be numbers, text, or dates.

The BETWEEN operator is inclusive: begin and end values are included.

BETWEEN Syntax

```
SELECT  column_name(s)
FROM table_name
WHERE column_name BETWEEN value1 AND value2;
```

The following SQL statement creates an alias named "Address" that combine four columns (Address, PostalCode, City and Country):

Example

```
SELECT  CustomerName, Address + ', ' + PostalCode + ' ' + City + ', ' +
Country AS Address
FROM Customers;
```

Note: To get the SQL statement above to work in MySQL use the following:

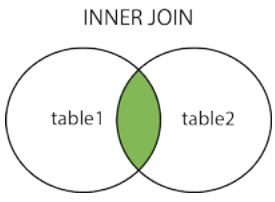
```
SELECT  CustomerName, CONCAT(Address,', ',PostalCode,', ',City,',
',Country) AS Address
FROM Customers;
```

SQL INNER JOIN Keyword

The INNER JOIN keyword selects records that have matching values in both tables.

INNER JOIN Syntax

```
SELECT  column_name(s)
FROM table1
INNER JOIN table2
ON table1.column_name = table2.column_name;
```



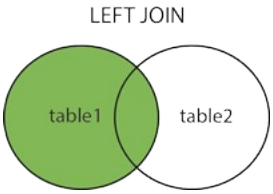
SQL LEFT JOIN Keyword

The LEFT JOIN keyword returns all records from the left table (table1), and the matched records from the right table (table2). The result is NULL from the right side, if there is no match.

LEFT JOIN Syntax

```
SELECT  column_name(s)
FROM table1
LEFT JOIN table2
ON table1.column_name = table2.column_name;
```

Note: In some databases LEFT JOIN is called LEFT OUTER JOIN.



The SQL EXISTS Operator

The EXISTS operator is used to test for the existence of any record in a subquery.

The EXISTS operator returns true if the subquery returns one or more records.

EXISTS Syntax

```
SELECT  column_name(s)
FROM table_name
WHERE EXISTS
(SELECT column_name FROM table_name WHERE condition);
```

The SQL INSERT INTO SELECT Statement

The INSERT INTO SELECT statement copies data from one table and inserts it into another table.

- INSERT INTO SELECT requires that data types in source and target tables match
- The existing records in the target table are unaffected

INSERT INTO SELECT Syntax

Copy all columns from one table to another table:

```
INSERT  INTO table2
SELECT * FROM table1
WHERE condition;
```

The SQL SELECT INTO Statement

The SELECT INTO statement copies data from one table into a new table.

SELECT INTO Syntax

Copy all columns into a new table:

```
SELECT  *
INTO newtable [IN externaldb]
FROM oldtable
WHERE condition;
```