SQL Queries

SQL SELECT Statement

```
SELECT column_name,column_name
FROM table_name
```

```
SELECT * FROM table_name;
```

SQL SELECT DISTINCT Statement

```
SELECT DISTINCT column_name,column_name
FROM table_name;
```

SQL WHERE Clause

```
SELECT column_name,column_name
FROM table_name
WHERE column_name operator value;
```

Operator	Description
=	Equal
<>	Not equal. **Note:** In some versions of SQL this operator may be writte
>	Greater than
<	Less than
>=	Greater than or equal

<=	Less than or equal
BETWEEN	Between an inclusive range
LIKE	Search for a pattern
IN	To specify multiple possible values for a column

SQL AND & OR Operators

```
SELECT * FROM Customers
WHERE Country='Germany'
AND (City='Berlin' OR City='München');
```

SQL ORDER BY Keyword

```
SELECT column_name, column_name
FROM table_name
ORDER BY column_name ASC|DESC, column_name ASC|DESC;
```

SQL INSERT INTO Statement

```
INSERT INTO table_name
VALUES (value1,value2,value3,...);
```

```
INSERT INTO table_name (column1,column2,column3,...)
VALUES (value1,value2,value3,...);
```

SQL UPDATE Statement

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

SQL DELETE Statement

```
DELETE FROM table_name
WHERE some_column=some_value;
```

SQL Injection

```
txtUserId = getRequestString("UserId");
txtSQL = "SELECT * FROM Users WHERE UserId = " + txtUserId
```

SQL SELECT TOP Clause

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

SQL LIKE Operator

```
SELECT column_name(s)
FROM table_name
WHERE column_name LIKE pattern;
```

SQL Wildcards

Wildcard	Description
%	A substitute for zero or more characters
_	A substitute for a single character
[*charlist*]	Sets and ranges of characters to match

[^*charlist*] or [!*charlist*]	Matches only a character NOT specified within the brackets

SQL IN Operator

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

SQL BETWEEN Operator

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

SQL Aliases

SQL Alias Syntax for Columns

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

SQL Alias Syntax for Tables

```
SELECT column_name(s)
FROM table_name AS alias_name;
```

SQL INNER JOIN Keyword

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

SQL LEFT JOIN Keyword

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

SQL RIGHT JOIN Keyword

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

SQL FULL OUTER JOIN Keyword

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

SQL UNION Operator

```
SELECT column_name(s) FROM table1
UNION
SELECT column_name(s) FROM table2;
```

SQL CREATE DATABASE Statement

```
CREATE DATABASE dbname;
```

SQL CREATE TABLE Statement

```
CREATE TABLE table_name
(
column_name1 data_type(size),
column_name2 data_type(size),
column_name3 data_type(size),
....
);
```

SQL Constraints

```
CREATE TABLE table_name
(
column_name1 data_type(size) constraint_name,
column_name2 data_type(size) constraint_name,
column_name3 data_type(size) constraint_name,
....
);
```

- **NOT NULL** - Indicates that a column cannot store NULL value
- **UNIQUE** - Ensures that each row for a column must have a unique value
- **PRIMARY KEY** - A combination of a NOT NULL and UNIQUE. Ensures that a column (or combination of two or more columns) have a unique identity which helps to find a particular record in a table more easily and quickly
- **FOREIGN KEY** - Ensure the referential integrity of the data in one table to match values in another table
- **CHECK** - Ensures that the value in a column meets a specific condition
- **DEFAULT** - Specifies a default value for a column

SQL CREATE INDEX Statement

```
CREATE INDEX index_name
ON table_name (column_name)
```

SQL ALTER TABLE Statement

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

SQL Aggregate Functions

SQL aggregate functions return a single value, calculated from values in a column.

Useful aggregate functions:

- AVG() - Returns the average value
- COUNT() - Returns the number of rows
- FIRST() - Returns the first value
- LAST() - Returns the last value
- MAX() - Returns the largest value

- MIN() - Returns the smallest value
- SUM() - Returns the sum

SQL Scalar functions

SQL scalar functions return a single value, based on the input value.

Useful scalar functions:

- UCASE() - Converts a field to upper case
- LCASE() - Converts a field to lower case
- MID() - Extract characters from a text field
- LEN() - Returns the length of a text field
- ROUND() - Rounds a numeric field to the number of decimals specified
- NOW() - Returns the current system date and time
- FORMAT() - Formats how a field is to be displayed