# A Brief Introduction to SQL Commands

Prof. N. Basit

```
SELECT [ DISTINCT ] * | LIST OF COLUMNS,
FUNCTIONS, CONSTANTS
FROM LIST OF TABLES OR VIEWS
[ WHERE CONDITION(S) [AND|OR|NOT] ]
[ ORDER BY ORDERING COLUMN(S) [ ASC | DESC ] ]
[ GROUP BY GROUPING COLUMN(S) ]
[ HAVING CONDITION(S) ]
```

***Note***: There are many configurations related to the basic SQL query structure (as can be seen above). Optional components are within square brackets [ ]. This brief tutorial will cover a subset of these. Good resources can be found online for additional information (such as on www.w3schools.com/sql)

**** Selection and Projection**

*"Select" will return the desired columns of the table*
*"Where" acts as a filter and returns rows that match ('true') the given condition(s)*

```
SELECT <list of columns> // SELECT = "Projection"
FROM <table(s), query that returns a (table) result>
WHERE <condition(s); can use AND / OR / NOT> // WHERE = "Selection"
```

**Examples:**

```
SELECT FirstName
FROM Persons

SELECT * FROM Movies // Selecting " * " means to select ALL columns

SELECT PartID FROM Parts
WHERE Price > 25.00 // The condition

SELECT CustomerName FROM Customer
WHERE City = 'London' OR 'New York'
```

**Operators for WHERE:** …WHERE <column><operator><value>
=, <> (not equal), >, >=, <, <=, LIKE (pattern search), IN (find value in a set; can be "NOT IN")
**Values:** string in 'single quotes', numerical value, another attribute, or IS NULL

**** Selection with "Distinct"**

*"Select" with the optional "Distinct" removes any duplicates from the solution*

```
SELECT DISTINCT <list of columns> ...
```

**Example:**

```
SELECT DISTINCT LoanAmount
FROM CustomerLoans ...
```

**** Order By (ASC | DESC) [optional component]**

*Sorts results by given columns either ascending or descending*

```
SELECT <list of columns>
FROM <table(s)>
ORDER BY <ordering columns> [ASC | DESC]
```

**Examples:**

```
SELECT * FROM Customers
ORDER BY Country DESC

SELECT * FROM Customers
ORDER BY Country ASC, CustomerName DESC
```

**** IN operator**

*Allows you to specify multiple values in a WHERE clause or can be used as shorthand for multiple OR conditions (e.g. value within some list)*

```
SELECT <list of columns>
FROM <table(s)>
WHERE <column name> IN (<list of values within parenthesis>)

-or-

SELECT <list of columns>
FROM <table(s)>
WHERE <column name> IN (SELECT <statement to create another table>)
```

Can also use "NOT IN"

**Examples:**

```
SELECT * FROM Customers
WHERE Country IN ('Germany', 'Scotland', 'France')

SELECT * FROM Customers
WHERE Country NOT IN ('Germany', 'Spain')

SELECT * FROM Customers
WHERE Country IN (SELECT Country FROM Suppliers)
```

**** CREATE TABLE Statement**

*Used to create a new table in a database*

```
CREATE TABLE <table name> (
      <column1> <datatype> [(size)],
      <column2> <datatype> [(size)],
      <column3> <datatype> [(size)]
      ...)
```

**Example:**

```
CREATE TABLE Persons (
      PersonID int,
      LastName varchar(255),
      FirstName varchar(255),
      Address varchar(255),
      City varchar(255)
)
```

**** INSERT INTO Statement**

*Used to insert new records in a table*

```
INSERT INTO <table name> (<comma separated list of column names>)
VALUES (<comma separated list of values that correspond to the columns>)
```

If you are adding values for all the columns of the table you can use a shortened version:

```
INSERT INTO <table name>
VALUES (<comma separated list of values that correspond to ALL the columns>)
```

**Example:**

```
INSERT INTO Customers (CustomerName, ContactName, Address, City, PostalCode,
Country)
VALUES ('Cardinal', 'Tom B. Erichsen', 'Skagen 21', 'Stavanger', '4006',
'Norway');
```

Example of inserting into only specific columns:

```
INSERT INTO Customers (CustomerName, City, Country)
VALUES ('Cardinal', 'Stavanger', 'Norway');
```

**** UPDATE Statement**

*Used to modify the existing records in a table*

```
UPDATE <table name>
SET <comma separated list of the following format: col1 = val1>
WHERE <condition>
```

**Note:** Be careful when updating records in a table! Notice the WHERE clause in the UPDATE statement. The WHERE clause specifies which record(s) that should be updated. If you omit the WHERE clause, all records in the table will be updated!

**Examples:**

```
UPDATE Customers
SET ContactName = 'Alfred Schmidt', City= 'Frankfurt'
WHERE CustomerID = 1
```

Updating multiple records:

```
UPDATE Customers
SET ContactName='Juan'
WHERE Country='Mexico';
```

**** ALTER TABLE Statement**

*Used to add, delete, or modify columns in an existing table. It can also be used to add and drop various constraints on an existing table*

```
ALTER TABLE <table name>
ADD <column name> <datatype>

ALTER TABLE <table name>
DROP <column name>

ALTER TABLE <table name>
MODIFY COLUMN <column name> <datatype>
```

**Examples:**

```
ALTER TABLE Persons
ADD DateOfBirth date

ALTER TABLE Persons
DROP COLUMN DateOfBirth
```

**** DELETE Statement**

*Used to delete existing records in a table*

```
DELETE FROM <table name>
WHERE <condition>
```

**Note:** Be careful when deleting records in a table! Notice the WHERE clause in the DELETE statement. The WHERE clause specifies which record(s) that should be deleted. If you omit the WHERE clause, all records in the table will be deleted!

**Examples:**

```
DELETE FROM Customers
WHERE CustomerName='Clayton Arnold'
```

If you wish to delete ALL records:

```
DELETE FROM <table name>
```

-or-

```
DELETE * FROM <table name>
```

**** DROP TABLE Statement**

*Used to drop an existing table in a database*

```
DROP TABLE <table name>
```

**Note:** Be careful before dropping a table. Deleting a table will result in loss of complete information stored in the table!

**Examples:**

```
DROP TABLE Shippers
```

If you wish to delete the data inside a table, but not the table itself you can use the TRUNCATE TABLE statement:

```
TRUNCATE TABLE <table name>
```