SQL Reference guide by Dennis Cassøe, v. 1.01

The following is mainly for access-databases, but most of it should also work with a MS-SQL server.
Send comments and expansions to suggest@cassoee.dk

General SQL

General:
SELECT table1.attribute1, table 2.attribute2
FROM table 1, table 2
WHERE table 1.attributeID = table2.attributeID
AND Somthing_that_has_to_be_evaluated_to_true

OR

SELECT attributes
FROM table1 t1 INNER JOIN table2 t2 ON t1.attributeID=t2.attributeID
WHERE Somthing_that_has_to_be_evaluated_to_true

Sorting the results:
SELECT * FROM TableA ORDER BY attribute1 asc, attributenavn2 desc
asc: Ascending, desc: Descending

Remove duplicates:
SELECT DISTINCT attribute1

Simple operations

Arithmetical operations (+,-,*,/):
SELECT attribute1, attribute2*100
SELECT attribute1-3, (attribute2+5)*100

Dates: WHERE Created = #dd-mm-yyyy#

Change the attributes name:
SELECT attribute1 AS NewName

Use abbreviations for table names in SQL:
SELECT t1.attributeA, t2.attributeC
FROM table1 t1, table2 t2

Comparisons in the where-clause
<> (not), =, >, >=, <, <=
Between … AND …
IN (list over values comma-separated)
LIKE (Text comparison)
IS NULL

Joins

Ordinary join:
TableA INNER JOIN TableB on TableA.ID = TableB.ID

LEFT/RIGHT, FULL OUTER JOIN:
TableA LEFT JOIN TableB on TableA.ID = TableB.ID
Includes the rows from TableA's which would not be included in the join

Join of three tables:
(TableA INNER JOIN TableB on TableA.ID = TableB.ID) INNER JOIN TableC on TableB.ID2 = TableC.ID2

Functions

Calculate Periods between dates:
Datediff("type_of_periods", start, end)
Type_of_periods: d (day), m (month), yyyy (year) ……
Now() is the systems current date

Rounding: Round()
Generally: Try VB's functions

Aggregate data

AVG, COUNT, MAX, MIN, STDDEV, SUM, VARIANCE

Can aggregate the total of rows: SELECT COUNT(AttributeA)

Can aggregate in groups
SELECT SUM(AttributeA), AttributeB
FROM TABLE A
GROUP BY AttributeB

HAVING can be used to exclude groups:
HAVING SUM(AttributeA) > 100

Insert, delete or update rows

Insert a row:
INSERT INTO TableA (Text1,Number2,Date3)
VALUES ("A",2,#07-02-1999#)

Delete rows:
DELETE FROM TABLEA WHERE ID=2

Update rows:
UPDATE TableA
SET Tekst1="DC", Tal2=1, Dato3=#07-07-2002#
WHERE Somthing_that_has_to_be_evaluated_to_true

UNION, INTERSECT, EXCEPT

UNION: Combines all rows from to tables
INTERSECT: Pick the rows which are in both tables
EXCEPT: Pick the rows which are in table1 but not in table2

IMPORTANT: The tables must have the same number of colums and of the same type.

Ex.
SELECT Text1,Number2 FROM TableA UNION
SELECT Text2,Number3 FROM TableB

Insert, delete or change tables

Create a table:
CREATE TABLE TableA (AttributeA Integer, AttributeB char(15), primary key (AttributeA))

Delete a table: DROP TABLE TableA
Append an attribute: ALTER TABLE TableA ADD AttributeC Integer
Remove an attribute: ALTER TABLE TableA DROP AttributeC

Sub queries, subselect, nested queries

A sub query is a complete select-statement embedded in an another SQL-statement.
Because there now are two SELECT's, the main on is called the outer and the one situated in the outer's where-clause is called for inner.

Ex.: Find the name and price of the most expensive wine
SELECT Name, price FROM wine
WHERE price = (Select MAX(price) FROM wine);
Note: The inner query finds the price and the outer specify the result

Possibilities to combine the two queries:
=, >, <, >=, <=
IN, NOT IN
EXISTS, NOT EXISTS
SOME, ALL (Here you can also use =, >, <, >=, <= in front of some/all)

Remeber:

- Do not use ORDER BY in a sub query
- Use explicit references, if there is a reference to a table in the outer query
- A sub query must always stand on the right side of an operator in the where-clause

Examples of Sub queries, subselect, nested queries

Make a list of all the red wines that are more expensive than the average price for red wines in the database
SELECT w1.Name, w1.price FROM wine w1
WHERE w1.price > (SELECT AVG(w2.price)
FROM wine w2
WHERE w2.type = 'Red')
AND w1.type ='Red';

Make a list of wines currently in one or more orders
SELECT w.wno, w.Name FROM wine w
WHERE w.wno IN (SELECT ol.wno FROM OrderLine ol)

Find the white wines, which is more expensive than any of the red wines:
SELECT w1.wno, w1.Name FROM wine w1
WHERE w1.price > ALL
(SELECT w2.price FROM wine w2 WHERE w2.type= 'Red')
AND w1.type = 'White