SQL (Structured Query Language) in one page

Table of contents:

<table>
<tr><td colspan="3">Database Manipulation</td></tr>
<tr><td>CREATE DATABASE database_name</td><td>Create a database</td><td>CREATE DATABASE My_First_Database</td></tr>
<tr><td>DROP DATABASE database_name</td><td>Delete a database</td><td>DROP DATABASE My_First_Database</td></tr>
<tr><td colspan="3">Table Manipulation</td></tr>
<tr><td>CREATE TABLE "table_name"
("column_1"<br>"data_type_for_column_1",
"column_2"<br>"data_type_for_column_2",
...)</td><td>Create a table in a database.
<table>
<caption>Data Types</caption>
<tr><th>Data Type</th><th>Description</th></tr>
<tr><td>integer(size)
int(size)
smallint(size)
tinyint(size)</td><td>Hold integers only. The maximum number of digits are specified in parenthesis.</td></tr>
<tr><td>decimal(size,d)
numeric(size,d)</td><td>Hold numbers with fractions. The maximum number of digits are specified in "size". The maximum number of digits to the right of the decimal is specified in "d".</td></tr>
<tr><td>char(size)</td><td>Holds a fixed length string (can contain letters, numbers, and special characters). The fixed size is specified in parenthesis.</td></tr>
<tr><td>varchar(size)</td><td>Holds a variable length string (can contain letters, numbers, and special characters). The maximum size is specified in parenthesis.</td></tr>
<tr><td>date(yyyymmdd)</td><td>Holds a date</td></tr>
</table>
</td><td>CREATE TABLE Person
(LastName varchar,
FirstName varchar,
Address varchar,
Age int)</td></tr>
<tr><td>ALTER TABLE table_name ADD column_name datatype</td><td>Add columns in an existing table.</td><td>ALTER TABLE Person ADD Sex char(6)</td></tr>
<tr><td>ALTER TABLE table_name DROP column_name datatype</td><td>Delete columns in an existing table.</td><td>ALTER TABLE Person DROP Sex char(6)</td></tr>
<tr><td>DROP TABLE table_name</td><td>Delete a table.</td><td>DROP TABLE Person</td></tr>
<tr><td colspan="3">Index Manipulation</td></tr>
<tr><td>CREATE INDEX index_name
ON table_name (column_name_1, column_name_2, ...)</td><td>Create a simple index.</td><td>CREATE INDEX PersonIndex
ON Person (LastName, FirstName)</td></tr>
<tr><td>CREATE UNIQUE INDEX index_name
ON table_name (column_name_1, column_name_2, ...)</td><td>Create a unique index.</td><td>CREATE UNIQUE INDEX PersonIndex
ON Person (LastName DESC)</td></tr>
<tr><td>DROP INDEX table_name.index_name</td><td>Delete a index.</td><td>DROP INDEX Person.PersonIndex</td></tr>
<tr><td colspan="3">Data Manipulation</td></tr>
<tr><td>INSERT INTO table_name
VALUES (value_1, value_2,....)</td><td rowspan="2">Insert new rows into a table.</td><td>INSERT INTO Persons
VALUES('Hussein', 'Saddam', 'White House')</td></tr>
<tr><td>INSERT INTO table_name (column1, column2,...)
VALUES (value_1, value_2,....)</td><td>INSERT INTO Persons (LastName, FirstName, Address)
VALUES('Hussein', 'Saddam', 'White House')</td></tr>
<tr><td>UPDATE table_name
SET column_name_1 = new_value_1, column_name_2 = new_value_2
WHERE column_name = some_value</td><td>Update one or several columns in rows.</td><td>UPDATE Person
SET Address = 'ups'
WHERE LastName = 'Hussein'</td></tr>
<tr><td>DELETE FROM table_name
WHERE column_name = some_value</td><td>Delete rows in a table.</td><td>DELETE FROM Person WHERE LastName = 'Hussein'</td></tr>
<tr><td>TRUNCATE TABLE table_name</td><td>Deletes the data inside the table.</td><td>TRUNCATE TABLE Person</td></tr>
<tr><td colspan="3">Select</td></tr>
<tr><td>SELECT column_name(s) FROM table_name</td><td>Select data from a table.</td><td>SELECT LastName, FirstName FROM Persons</td></tr>
<tr><td>SELECT * FROM table_name</td><td>Select all data from a table.</td><td>SELECT * FROM Persons</td></tr>
<tr><td>SELECT DISTINCT column_name(s) FROM table_name</td><td>Select only distinct (different) data from a table.</td><td>SELECT DISTINCT LastName, FirstName
FROM Persons</td></tr>
<tr><td rowspan="2">SELECT column_name(s) FROM table_name
WHERE column operator value
AND column operator value
OR column operator value
AND (... OR ...)
...</td><td>Select only certain data from a table.</td><td>SELECT * FROM Persons WHERE sex='female'
SELECT * FROM Persons WHERE Year>1970</td></tr>
<tr><td>
<table>
<caption>Operators</caption>
<tr><th>Operator</th><th>Description</th></tr>
<tr><td>=</td><td>Equal</td></tr>
<tr><td><></td><td>Not equal</td></tr>
<tr><td>></td><td>Greater than</td></tr>
<tr><td><</td><td>Less than</td></tr>
<tr><td>>=</td><td>Greater than or equal</td></tr>
<tr><td><=</td><td>Less than or equal</td></tr>
<tr><td>BETWEEN</td><td>Between an inclusive range</td></tr>
<tr><td>LIKE</td><td>Search for a pattern.
A "%" sign can be used to define wildcards (missing letters in the pattern) both before and after the pattern.</td></tr>
</table>
</td><td>SELECT * FROM Persons
WHERE FirstName='Saddam'
AND LastName='Hussein'
SELECT * FROM Persons
WHERE FirstName='Saddam'
OR LastName='Hussein'
SELECT * FROM Persons WHERE
(FirstName='Tove' OR FirstName='Stephen')
AND LastName='Svendson'
SELECT * FROM Persons WHERE FirstName LIKE 'O%'
SELECT * FROM Persons WHERE FirstName LIKE '%a'
SELECT * FROM Persons WHERE FirstName LIKE '%la%'</td></tr>
<tr><td>SELECT column_name(s) FROM table_name
WHERE column_name IN (value1, value2, ...)</td><td>The IN operator may be used if you know the exact value you want to return for at least one of the columns.</td><td>SELECT * FROM Persons
WHERE LastName IN ('Hansen','Pettersen')</td></tr>
<tr><td>SELECT column_name(s) FROM table_name
ORDER BY row_1, row_2 DESC, row_3 ASC, ...</td><td>Select data from a table with sort the rows.
Note:
• ASC (ascend) is a alphabetical and numerical order (optional)
• DESC (descend) is a reverse alphabetical and numerical order</td><td>SELECT * FROM Persons
ORDER BY LastName
SELECT FirstName, LastName FROM Persons
ORDER BY LastName DESC
SELECT Company, OrderNumber FROM Orders
ORDER BY Company DESC, OrderNumber ASC</td></tr>
<tr><td>SELECT column_1, ...,
SUM(group_column_name)
FROM table_name
GROUP BY group_column_name</td><td>GROUP BY... was added to SQL because aggregate functions (like SUM) return the aggregate of all column values every time they are called, and without the GROUP BY function it was impossible to find the sum for each individual group of column values.
<table>
<caption>Some aggregate functions</caption>
<tr><th>Function</th><th>Description</th></tr>
<tr><td>AVG(column)</td><td>Returns the average value of a column</td></tr>
<tr><td>COUNT(column)</td><td>Returns the number of rows (without a NULL value) of a column</td></tr>
<tr><td>MAX(column)</td><td>Returns the highest value of a column</td></tr>
<tr><td>MIN(column)</td><td>Returns the lowest value of a column</td></tr>
<tr><td>SUM(column)</td><td>Returns the total sum of a column</td></tr>
</table>
</td><td>SELECT Company, SUM(Amount)
FROM Sales
GROUP BY Company</td></tr>
<tr><td>SELECT column_1, ...,
SUM(group_column_name)
FROM table_name
GROUP BY group_column_name
HAVING
SUM(group_column_name)
condition value</td><td>HAVING... was added to SQL because the WHERE keyword could not be used against aggregate functions (like SUM), and without HAVING... it would be impossible to test for result conditions.</td><td>SELECT Company, SUM(Amount)
FROM Sales
GROUP BY Company
HAVING SUM(Amount)>10000</td></tr>
<tr><td colspan="3">Alias</td></tr>
<tr><td>SELECT column_name AS column_alias FROM table_name</td><td>Column name alias</td><td>SELECT LastName AS Family, FirstName AS Name
FROM Persons</td></tr>
<tr><td>SELECT table_alias.column_name
FROM table_name AS table_alias</td><td>Table name alias</td><td>SELECT LastName, FirstName
FROM Persons AS Employees</td></tr>
<tr><td colspan="3">Join</td></tr>
<tr><td>SELECT column_1_name, column_2_name, ...
FROM first_table_name
INNER JOIN second_table_name
ON first_table_name.keyfield = second_table_name.foreign_keyfield</td><td>The INNER JOIN returns all rows from both tables where there is a match. If there are rows in first table that do not have matches in second table, those rows will not be listed.</td><td>SELECT Employees.Name, Orders.Product
FROM Employees
INNER JOIN Orders
ON
Employees.Employee_ID=Orders.Employee_ID</td></tr>
<tr><td>SELECT column_1_name, column_2_name, ...
FROM first_table_name
LEFT JOIN second_table_name
ON first_table_name.keyfield = second_table_name.foreign_keyfield</td><td>The LEFT JOIN returns all the rows from the first table, even if there are no matches in the second table. If there are rows in first table that do not have matches in second table, those rows also will be listed.</td><td>SELECT Employees.Name, Orders.Product
FROM Employees
LEFT JOIN Orders
ON
Employees.Employee_ID=Orders.Employee_ID</td></tr>
<tr><td>SELECT column_1_name, column_2_name, ...
FROM first_table_name
RIGHT JOIN second_table_name
ON first_table_name.keyfield = second_table_name.foreign_keyfield</td><td>The RIGHT JOIN returns all the rows from the second table, even if there are no matches in the first table. If there had been any rows in second table that did not have matches in first table, those rows also would have been listed.</td><td>SELECT Employees.Name, Orders.Product
FROM Employees
RIGHT JOIN Orders
ON
Employees.Employee_ID=Orders.Employee_ID</td></tr>
<tr><td colspan="3">UNION</td></tr>
<tr><td>SQL_Statement_1
UNION
SQL_Statement_2</td><td>Select all different values from SQL_Statement_1 and SQL_Statement_2</td><td>SELECT E_Name FROM Employees_Norway
UNION
SELECT E_Name FROM Employees_USA</td></tr>
<tr><td>SQL_Statement_1
UNION ALL
SQL_Statement_2</td><td>Select all values from SQL_Statement_1 and SQL_Statement_2</td><td>SELECT E_Name FROM Employees_Norway
UNION
SELECT E_Name FROM Employees_USA</td></tr>
<tr><td colspan="3">SELECT INTO/IN</td></tr>
<tr><td>SELECT column_name(s)
INTO new_table_name
FROM source_table_name
WHERE query</td><td>Select data from table(S) and insert it into another table.</td><td>SELECT * INTO Persons_backup FROM Persons</td></tr>
<tr><td>SELECT column_name(s)
IN external_database_name
FROM source_table_name
WHERE query</td><td>Select data from table(S) and insert it in another database.</td><td>SELECT Persons.* INTO Persons IN 'Backup.db' FROM Persons WHERE City='Sandnes'</td></tr>
<tr><td colspan="3">CREATE VIEW</td></tr>
<tr><td>CREATE VIEW view_name AS
SELECT column_name(s)
FROM table_name
WHERE condition</td><td>Create a virtual table based on the result-set of a SELECT statement.</td><td>CREATE VIEW [Current Product List] AS
SELECT ProductID, ProductName
FROM Products
WHERE Discontinued=No</td></tr>
<tr><td colspan="3">OTHER</td></tr>
</table>

Valid Valid