SQL (Structured Query Language) in one page

Table of contents: Database Manipulation (CREATE, DROP DATABASE), Table Manipulation (CREATE, ALTER, DROP TABLE, Data Types), Index Manipulation (CREATE, DROP INDEX), Data Manipulation (INSERT, UPDATE, DELETE, TRUNCATE TABLE), Select (SELECT, FROM, WHERE, ORDER BY, GROUP BY, HAVING, Operators, Aggregate functions), Alias, Join, UNION, SELECT INTO/IN, CREATE VIEW.

Database Manipulation

CREATE DATABASE *database_name*	Create a database	CREATE DATABASE My_First_Database
DROP DATABASE *database_name*	Delete a database	DROP DATABASE My_First_Database

Table Manipulation

CREATE TABLE "*table_name***"** **("***column_1***" "***data_type_for_column_1***",** **"***column_2***" "***data_type_for_column_2***",** **...)**	Create a table in a database.	CREATE TABLE Person (LastName varchar, FirstName varchar, Address varchar, Age int)

Data Types

Data Type	Description

integer(size)
int(size)
smallint(size)
tinyint(size)
	Hold integers only. The maximum number of digits are specified in parenthesis.

decimal(size,d)
numeric(size,d)
	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".

char(size)	Holds a fixed length string (can contain letters, numbers, and special characters). The fixed size is specified in parenthesis.
varchar(size)	Holds a variable length string (can contain letters, numbers, and special characters). The maximum size is specified in parenthesis.
date(yyyymmdd)	Holds a date

ALTER TABLE *table_name* **ADD** *column_name datatype*	Add columns in an existing table.	ALTER TABLE Person ADD Sex char(6)
ALTER TABLE *table_name* **DROP** *column_name datatype*	Delete columns in an existing table.	ALTER TABLE Person DROP Sex char(6)
DROP TABLE *table_name*	Delete a table.	DROP TABLE Person

Index Manipulation

CREATE INDEX *index_name* **ON** *table_name* **(***column_name_1***,** *column_name_2***, ...)**	Create a simple index.	CREATE INDEX PersonIndex ON Person (LastName, FirstName)
CREATE UNIQUE INDEX *index_name* **ON** *table_name* **(***column_name_1***,** *column_name_2***, ...)**	Create a unique index.	CREATE UNIQUE INDEX PersonIndex ON Person (LastName DESC)
DROP INDEX *table_name.index_name*	Delete a index.	DROP INDEX Person.PersonIndex

Data Manipulation

INSERT INTO *table_name*
VALUES (*value_1***,** *value_2***,....)**
	Insert new rows into a table.	INSERT INTO Persons
		VALUES('Hussein', 'Saddam', 'White House')

INSERT INTO *table_name* **(***column1***,** *column2***,...)**
VALUES (*value_1***,** *value_2***,....)**
		INSERT INTO Persons (LastName, FirstName, Address)
		VALUES('Hussein', 'Saddam', 'White House')

UPDATE *table_name*
SET *column_name_1* = *new_value_1*, *column_name_2* = *new_value_2*
WHERE *column_name* = *some_value*
	Update one or several columns in rows.	UPDATE Person
		SET Address = 'ups'
		WHERE LastName = 'Hussein'

DELETE FROM *table_name*
WHERE *column_name* = *some_value*
	Delete rows in a table.	DELETE FROM Person WHERE LastName = 'Hussein'

TRUNCATE TABLE *table_name*	Deletes the data inside the table.	TRUNCATE TABLE Person

Select

SELECT *column_name(s)* **FROM** *table_name*	Select data from a table.	SELECT LastName, FirstName FROM Persons
SELECT * FROM *table_name*	Select all data from a table.	SELECT * FROM Persons
SELECT DISTINCT *column_name(s)* **FROM** *table_name*	Select only distinct (different) data from a table.	SELECT DISTINCT LastName, FirstName FROM Persons

SELECT *column_name(s)* **FROM** *table_name*
WHERE *column operator value*
AND *column operator value*
OR *column operator value*
AND (... OR ...)
...
	Select only certain data from a table.	SELECT * FROM Persons WHERE sex='female'
		SELECT * FROM Persons WHERE Year>1970
		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%'

Operators

Operator	Description
=	Equal
<>	Not equal
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
BETWEEN	Between an inclusive range
LIKE	Search for a pattern. A "%" sign can be used to define wildcards (missing letters in the pattern) both before and after the pattern.

SELECT *column_name(s)* **FROM** *table_name*
WHERE *column_name* **IN (***value1***,** *value2***, ...)**
	The IN operator may be used if you know the exact value you want to return for at least one of the columns.	SELECT * FROM Persons
		WHERE LastName IN ('Hansen','Pettersen')

SELECT *column_name(s)* **FROM** *table_name*
ORDER BY *row_1*, *row_2* **DESC,** *row_3* **ASC, ...**
	Select data from a table with sort the rows.
	
	Note:
		SELECT * FROM Persons
		ORDER BY LastName

	• **ASC** (ascend) is a alphabetical and numerical order (optional)
	• **DESC** (descend) is a reverse alphabetical and numerical order
		SELECT FirstName, LastName FROM Persons
		ORDER BY LastName DESC
		
		SELECT Company, OrderNumber FROM Orders
		ORDER BY Company DESC, OrderNumber ASC

SELECT *column_1*, **...**, **SUM(***group_column_name***)**
FROM *table_name*
GROUP BY *group_column_name*
	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.	SELECT Company, SUM(Amount)
		FROM Sales
		GROUP BY Company

Some aggregate functions

Function	Description
AVG(column)	Returns the average value of a column
COUNT(column)	Returns the number of rows (without a NULL value) of a column
MAX(column)	Returns the highest value of a column
MIN(column)	Returns the lowest value of a column
SUM(column)	Returns the total sum of a column

SELECT *column_1*, **...**, **SUM(***group_column_name***)**
FROM *table_name*
GROUP BY *group_column_name*
HAVING SUM(*group_column_name***)** *condition value*
	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.	SELECT Company, SUM(Amount)
		FROM Sales
		GROUP BY Company
		HAVING SUM(Amount)>10000

Alias

SELECT *column_name* **AS** *column_alias* **FROM** *table_name*	Column name alias	SELECT LastName AS Family, FirstName AS Name
		FROM Persons

SELECT *table_alias.column_name* **FROM** *table_name* **AS** *table_alias*	Table name alias	SELECT LastName, FirstName
		FROM Persons AS Employees

Join

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*
	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.	SELECT Employees.Name, Orders.Product
		FROM Employees
		INNER JOIN Orders
		ON Employees.Employee_ID=Orders.Employee_ID

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*
	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.	SELECT Employees.Name, Orders.Product
		FROM Employees
		LEFT JOIN Orders
		ON Employees.Employee_ID=Orders.Employee_ID

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*
	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.	SELECT Employees.Name, Orders.Product
		FROM Employees
		RIGHT JOIN Orders
		ON Employees.Employee_ID=Orders.Employee_ID

UNION

SQL_Statement_1
UNION
SQL_Statement_2
	Select all different values from *SQL_Statement_1* and *SQL_Statement_2*	SELECT E_Name FROM Employees_Norway
		UNION
		SELECT E_Name FROM Employees_USA

SQL_Statement_1
UNION ALL
SQL_Statement_2
	Select all values from *SQL_Statement_1* and *SQL_Statement_2*	SELECT E_Name FROM Employees_Norway
		UNION
		SELECT E_Name FROM Employees_USA

SELECT INTO/IN

SELECT *column_name(s)*
INTO *new_table_name*
FROM *source_table_name*
WHERE *query*
	Select data from table(S) and insert it into another table.	SELECT * INTO Persons_backup FROM Persons

SELECT *column_name(s)*
IN *external_database_name*
FROM *source_table_name*
WHERE *query*
	Select data from table(S) and insert it in another database.	SELECT Persons.* INTO Persons IN 'Backup.db' FROM Persons WHERE City='Sandnes'

CREATE VIEW

CREATE VIEW *view_name* **AS**
SELECT *column_name(s)*
FROM *table_name*
WHERE *condition*
	Create a virtual table based on the result-set of a SELECT statement.	CREATE VIEW [Current Product List] AS
		SELECT ProductID, ProductName
		FROM Products
		WHERE Discontinued=No

OTHER