

|  |  |  |
| --- | --- | --- |
| Database Manipulation | | |
|  | **CREATE DATABASE***database\_name* | Create a database | CREATE DATABASE TestDB |
|  | **DROP DATABASE***database\_name* | Delete a database | DROP DATABASE TestDB |
|  | 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(50),  FirstName varchar(50),  Address varchar(100),  Age int  ) |
|  | |  |  | | --- | --- | | Data Types | | | **Data Type** | **Description** | | integer(size) | Hold integers only. The maximum number of digits are specified in parenthesis. | | int(size) | | smallint(size) | | tinyint(size) | | decimal(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". | | numeric(size,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 Gender char(6) |
|  | **ALTER TABLE***table\_name***DROP***column\_name**datatype* | Delete columns in an existing table. | ALTER TABLE Person DROP Gender 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 all 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' |
|  | |  |  | | --- | --- | | 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 \* 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%' |
|  | **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:**   * **ASC** (ascend) is a alphabetical and numerical order (optional) * **DESC** (descend) is a reverse alphabetical and numerical order | 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 |
|  | **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 |

* What is difference between Inner join and Outer join?
* What is difference between Function and Stored procedure
* What are table variables? What is difference between temp (#) tables and table variables?
* What is difference Primary key and foreign key?
* What are indexes, what is it used for. What are the different types.
* What is normalization. What are the different forms.