Title

Landscape, Margin all 0.1 except bottom 2.5, font, 14

[H1 2](#_Toc310762800)

[H1 again 2](#_Toc310762801)

[H2 2](#_Toc310762802)

[H2 again 2](#_Toc310762803)

[H1 again 2](#_Toc310762804)

[H2 again 2](#_Toc310762805)

H1

* No Spacing

H1 again

H2

* No Spacing

H2 again

* No Spacing

H1 again

H2 again

* No Spacing

#### T-SQL

**SELECT** column **FROM** table

SELECT **DISTINCT** column FROM table

SELECT column FROM table **WHERE** column operator value

* To conditional select data from a table
* Using ‘’ around **text** value

SELECT column FROM table WHERE column **LIKE** pattern

* use ‘%’ to define wildcards

**INSERT INTO** table **VALUES** (value1, value2, …)

* Insert a new complete row

INSERT INTO table **(*column1*, *column2*, …)** VALUES (value1, value2, …)

* Insert data into **specified** column(s)

**UPDATE** table **SET** column1 = new\_value1, column2 = new\_value2 **WHERE** column = some\_value

**DELETEFROM** table **WHERE** column = value

**DELETE \*FROM** table

* Delete ALL rows

SELECT column1, column2, … FROM table **ORDER BY** column1, column2, …

* ORDER BY column **DESC/ASC**

SELECT column FROM table **WHERE** column1 operator value1 **AND/OR** column2 operator value2

SELECT column(s) FROM table WHERE column **IN** (value1, value2, …)

* If you know the **EXACT** value a column is going to return

SELECT column(s) FROM table WHERE column **BETWEEN** value1 **AND** value2

* Depends on database, the test values may not be included in the result

SELECT column **AS** column\_alias FROM table **AS** table\_alias

Select data from 2 tables

* + SELECT field1, field2, … FROM ***first\_table* , *second\_table*WHERE** first\_table.keyfield = second\_table.foreign\_keyfield
  + *e.g. SELECT Customers.Name, Orders.Product FROM Customers, Orders WHERE Customers.Customer\_ID = Orders.Customer\_ID*

**WHERE**

* + SELECT field1, field2, … FROM ***first\_table*INNER | LEFT/RIGHT/OUTER JOIN*second\_table*ON** first\_table.keyfield = second\_table.foreign\_keyfield
  + *e.g. SELECT Customers.Name, Orders.Product FROM Customers INNER | LEFT/RIGHT/FULL OUTER JOIN Orders ON Customers.Customer\_ID = Orders.Customer\_ID*
    - **SAME** as using **WHERE**
    - **Returns all the rows from both tables ONLY ON those which has a match**. e.g. If there are rows in Customers that do not have matches in Orders, those rows will NOT be listed. i.e. ID: 02, 04

INNER JOIN

* + - **Returns all the rows from the 1st table, even if there are no matches in the 2nd table**.
    - If there are rows in 1st table (Customers) that do not have matches in 2nd table (Orders), those **rows** in the **1st table** will **ALSO** be **listed**. **Empty field** will contain a **<null>** value.
    - Helps you quickly spot inactive customers!

LEFT OUTER JOIN

* + - **Returns all the rows from the 2nd table, even if there are no matches in the 1st table**.
    - If there had been any rows in 2nd table (Orders) that do not have matches in 1st table (Customers), those **rows** in the **2nd table** would **ALSO** be **listed**. **Empty field** will contain a **<null>** value.
    - Helps you quickly spot orphans in the Orders table, i.e. orders which doesn’t have a matching customers!

RIGHT OUTER JOIN

* + - **Returns all the rows from both tables even there are no matches on either table.Empty field** will contain a **<null>** value.
    - Helps you spot both inactive customers and orphan orders!

FULL OUTER JOIN

* + - Returns a Cartesian product. It combines every row from the left table with every row in the right table.
    - Returns all the rows from both table as if NO WHERE clause was specified.

CROSS JOIN

**JOIN**: note – replacing comma **“,”** with **INNER | LEFT/RIGHT/OUTER JOIN**and **WHERE** with **ON**

**Results:**

|  |
| --- |
| **Employees:**    **Orders:**    **WHERE Result:**    **INNER JOIN Result:**    **LEFT OUTER JOIN Result:**    **RIGHT OUTER JOIN Result:** |

SQL\_statement\_1 **UNION** SQL\_statement\_2

* Combine 2 result sets together
* **ALL** of the selected columns and their **data type** need to be the same
* **Onlydistinct** values are selected

SQL\_statement\_1 **UNION ALL** SQL\_statement\_2

* **NO distinct** values requirement

Functions: SELECT **function(column, some\_optional\_param)** FROM table

* + **AVG**(c), **COUNT**(\*), **COUNT**(DISTINCT c), **MAX**(c), **MIN**(c), **SUM**(c)

Aggregate: operate against a collection of values and return a single value

* + UCASE(c), LCASE(c), LEN(c), MID(c,start[,end]), LEFT(c,#\_of\_char), RIGHT(c,#\_of\_char), ROUND(c, decimals), MOD(x,y), NOW(), FORMAT(c,format), DATEDIFF(d, date1, date2)

Scalar: operate against a single value and return a single value

SELECT column, SUM(column) FROM table **GROUP BY** column

* GROUP BY is added to SQL ‘cos - aggregate functions (like SUM) return the aggregate of **ALL column values** and without the GROUP BY function it was impossible to find the sum for **EACH individual group of column values**

|  |
| --- |
| This "Sales" Table:    And This SQL:  Returns this result:    The above code is **invalid** ‘cos the column returned is not part of an aggregate. A GROUP BY clause will solve this problem:    Returns this result: |

SELECT column, SUM(column) FROM table GROUP BY column **HAVING** SUM(column) condition value

* HAVING is added to SQL ‘cos – the **WHERE keyword could not be used against aggregate functions** (like SUM), and without HAVING…it would be impossible to test for result conditions. **Using HAVING to combine aggregate function and a testing condition.**

|  |
| --- |
| This "Sales" Table:    This SQL:    Returns this result |

**SELECT** column(s) **INTOnew\_dest\_table** [IN external\_database] FROM source\_table

* Creating backup or temp table

Online Tutorial: <http://www.devx.com/dbzone/Article/17403/0/page/2>

1