SQL JOINS:

<http://www.sqlservertutorial.net/sql-server-basics/>

SQL INNER JOIN syntax

Scenario Explanation Link: <http://www.sqlservertutorial.net/sql-server-basics/sql-server-inner-join/>

The following illustrates INNER JOIN syntax for joining two tables:

SELECT column\_name(s)  
FROM table1  
INNER JOIN table2ON table1.column\_name = table2.column\_name;

Let’s examine the syntax above in greater detail:

The table\_1 and table\_2 are called joined-tables.

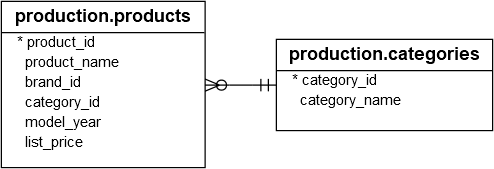
For each row in the table\_1, the query find the corresponding row in the table\_2 that meet the join condition.

If the corresponding row found, the query returns a row that contains data from both tables.

Otherwise, it examines next row in the table\_1, and this process continues until all the rows in the table\_1 are examined.



The inner join is one of the most commonly used joins in SQL Server. The inner join clause allows you to query data from two or more related tables.



SELECT

product\_name,

list\_price,

**category\_id**

FROM

production.products

ORDER BY

product\_name DESC;



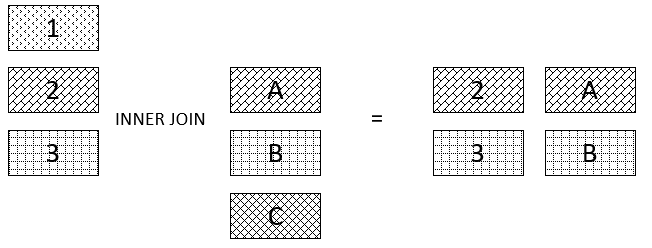
The query returned only a list of category identification numbers, not the category names. To include the **category names** in the result set, you use the INNER JOIN clause as follows:

SELECT product\_name, category\_name, list\_price FROM production.products p

INNER JOIN production.categories c ON c.category\_id = p.category\_id

ORDER BY product\_name DESC;





==================================

SQL OUTER JOIN – left outer join

Scenario Explanation Link: <http://www.sqlservertutorial.net/sql-server-basics/sql-server-left-join/>

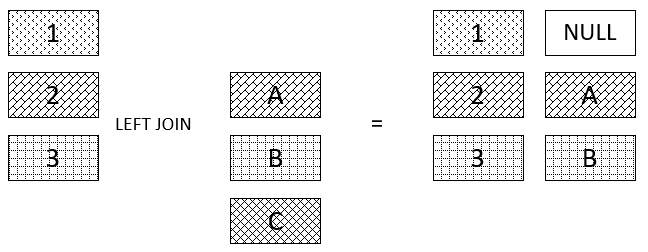
SELECT column\_name(s)  
FROM table1  
LEFT JOIN table2ON table1.column\_name = table2.column\_name;

My understanding-Things that we want to achieve using right join can also be achieved using left join if we place the table name in left hand side of “LEFT JOIN” keyword.

SQL left outer join is also known as SQL left join.

Suppose, we want to join two tables: A and B. SQL left outer join returns all rows in the left table (A)

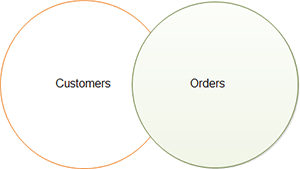
and all the matching rows(matching the join condition) found in the right table (B). It means the result of the SQL left join always contains the rows in the left table.





SELECT Customers.CustomerName, Orders.OrderID  
FROM Customers  
LEFT JOIN Orders ON Customers.CustomerID = Orders.CustomerID  
ORDER BY Customers.CustomerName;

Right outer join: opposite of left outer join.



=============

SQL full outer join returns:

SELECT column\_name(s)  
FROM table1  
FULL OUTER JOIN table2ON table1.column\_name = table2.column\_nameWHERE condition;

all rows in the left table table\_A.

all rows in the right table table\_B.

and all matching rows in both tables.



## **JOIN Three Tables**

SELECT Orders.OrderID, Customers.CustomerName, Shippers.ShipperName  
FROM ((Orders INNER JOIN Customers ON Orders.CustomerID Customers.CustomerID)INNER JOIN Shippers ON Orders.ShipperID = Shippers.ShipperID);

The following statement uses two INNER JOIN clauses to query data from the three tables

SELECT

    product\_name,

    category\_name,

    brand\_name,

    list\_price

FROM

    production.products p

INNER JOIN production.categories c ON c.category\_id = p.category\_id

INNER JOIN production.brands b ON b.brand\_id = p.brand\_id

ORDER BY

    product\_name DESC;

SQL - CARTESIAN or CROSS JOINS

The basic syntax of the **CARTESIAN JOIN** or the **CROSS JOIN** is as follows −

SELECT table1.column1, table2.column2...

FROM table1, table2 [, table3 ]

SELECT table1.column1, table2.column2...

FROM table1, table2 [, table3 ]

# SQL - SELF JOINS

SELECT a.column\_name, b.column\_name...

FROM table1 a, table1 b

WHERE a.common\_field = b.common\_field;

### **Using WHERE with Inner Join**

SELECT pz.CompanyId, pz.CompanyCity, pz.CompanyName,f.ItemName, f.UnitsSold

FROM PizzaCompany pz

INNER JOIN Foods f ON pz.CompanyId = f.CompanyId

WHERE f.UnitsSold > 6

ORDER BY pz.CompanyCity

### **Using Group By with Inner Join**

SELECT pz.CompanyCity, pz.CompanyName, SUM(f.UnitsSold) AS TotalQuantitySold

FROM PizzaCompany pz

INNER JOIN Foods f ON pz.CompanyId = f.CompanyId

GROUP BY pz.CompanyCity, pz.CompanyName

ORDER BY pz.CompanyCity

### **A brief note on Equi and Non-Equi(Theta) Join**

### **Equi Join**

As the name suggests, equi join contains an equality operator ‘=’ either in the Join clause or in the WHERE condition. SQL Inner, Left, Right are all equi joins when ‘=’ operator is being used as a comparison operator. Usually, when there is a mention of SQL Inner Join, it is considered as an Inner equi Join, in an unusual situation only, equality operator is not used.

### **Theta Join (Non-equi join)**

Non-equi join is basically opposite of equi-join and is used when we join on a condition other than ‘=’ operator. This type is rarely used in practice. Below is an example that makes use of theta join with an inequality operator (<) to evaluate profit by estimating cost and selling prices in two tables.

SELECT \* FROM Table1 T1, Table2 T2 WHERE T1.ProductCost < T2.SalesPrice

salesman

salesman\_id | name | city | commission

-------------+------------+----------+------------

5001 | James Hoog | New York | 0.15

5002 | Nail Knite | Paris | 0.13

5005 | Pit Alex | London | 0.11

5006 | Mc Lyon | Paris | 0.14

5007 | Paul Adam | Rome | 0.13

5003 | Lauson Hen | San Jose | 0.12

cutsomer

customer\_id | cust\_name | city | grade | salesman\_id

-------------+----------------+------------+-------+-------------

3002 | Nick Rimando | New York | 100 | 5001

3007 | Brad Davis | New York | 200 | 5001

3005 | Graham Zusi | California | 200 | 5002

3008 | Julian Green | London | 300 | 5002

3004 | Fabian Johnson | Paris | 300 | 5006

3009 | Geoff Cameron | Berlin | 100 | 5003

3003 | Jozy Altidor | Moscow | 200 | 5007

orders

ord\_no purch\_amt ord\_date customer\_id salesman\_id

---------- ---------- ---------- ----------- -----------

70001 150.5 2012-10-05 3005 5002

70009 270.65 2012-09-10 3001 5005

70002 65.26 2012-10-05 3002 5001

70004 110.5 2012-08-17 3009 5003

70007 948.5 2012-09-10 3005 5002

70005 2400.6 2012-07-27 3007 5001

70008 5760 2012-09-10 3002 5001

Write a SQL statement to prepare a list with salesman name, customer name and their cities for the salesmen and customer who belongs to the same city.

*select s.name, c.cust\_name, s.city from salesman as s, customer as c where s.city=c.city;*

Comments:

-This is supposed to test knowledge of joins, yet there's no join in the solution. The question is not worded well.

It should ask that we produce a list of any combination of customers/salesman as long as they live in the same city.

-To join two or more tables, it is not necessary to use JOIN keyword always.

FROM customer, salesman >> this is cartesian join

FROM customer c JOIN salesman s >> which you used, is inner join. so you missed the results derived from salesman to customer

Write a SQL statement to make a list with order no, purchase amount, customer name and their cities for those orders which order amount between 500 and 2000.

*select o.ord\_no, o.purch\_amt,c.cust\_name,c.city from orders as o,customer as c where o.customer\_id=c.customer\_id and o.purch\_amt between 500 and 2000;*