SQL Joins

What is SQL Joins?

An SQL JOIN clause combines rows from two or more tables. It creates a set of rows in a temporary table.

How to Join two tables in SQL?

A JOIN works on two or more tables if they have at least one common field and have a relationship between them.

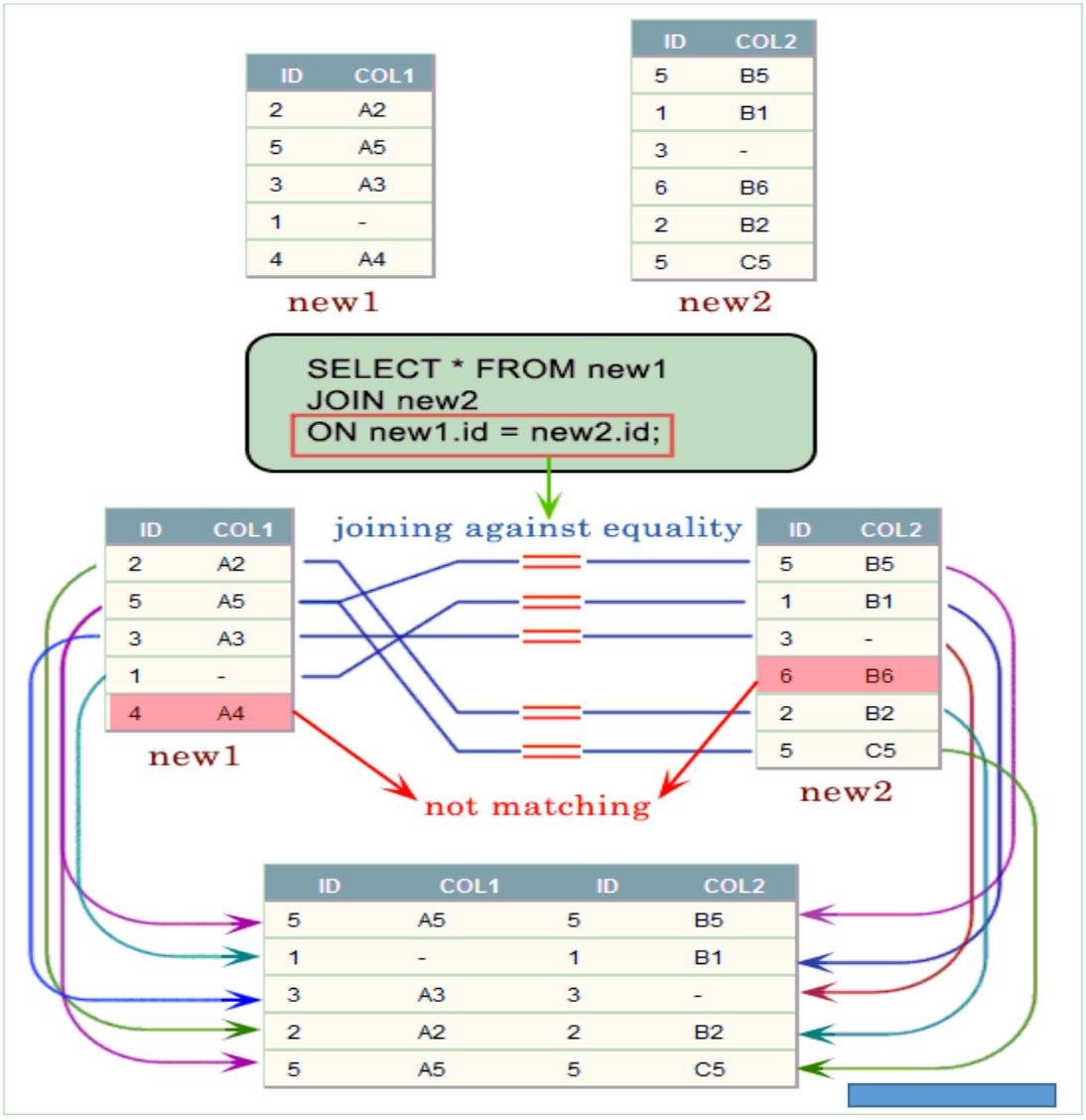
JOIN keeps the base tables (structure and data) unchanged.

What is Equi Join in SQL?

SQL EQUI JOIN performs a JOIN against equality or matching column(s) values of the associated tables. An equal sign (=) is used as comparison operator in the where clause to refer equality.

You may also perform EQUI JOIN by using JOIN keyword followed by ON keyword and then specifying names of the columns along with their associated tables to check equality.

Pictorial presentation of SQL Equi Join:



Syntax:

```
SELECT column_list
FROM table1, table2....
WHERE table1.column_name =
table2.column_name;
```

or

```
SELECT *
FROM table1
JOIN table2
[ON (join_condition)]
```

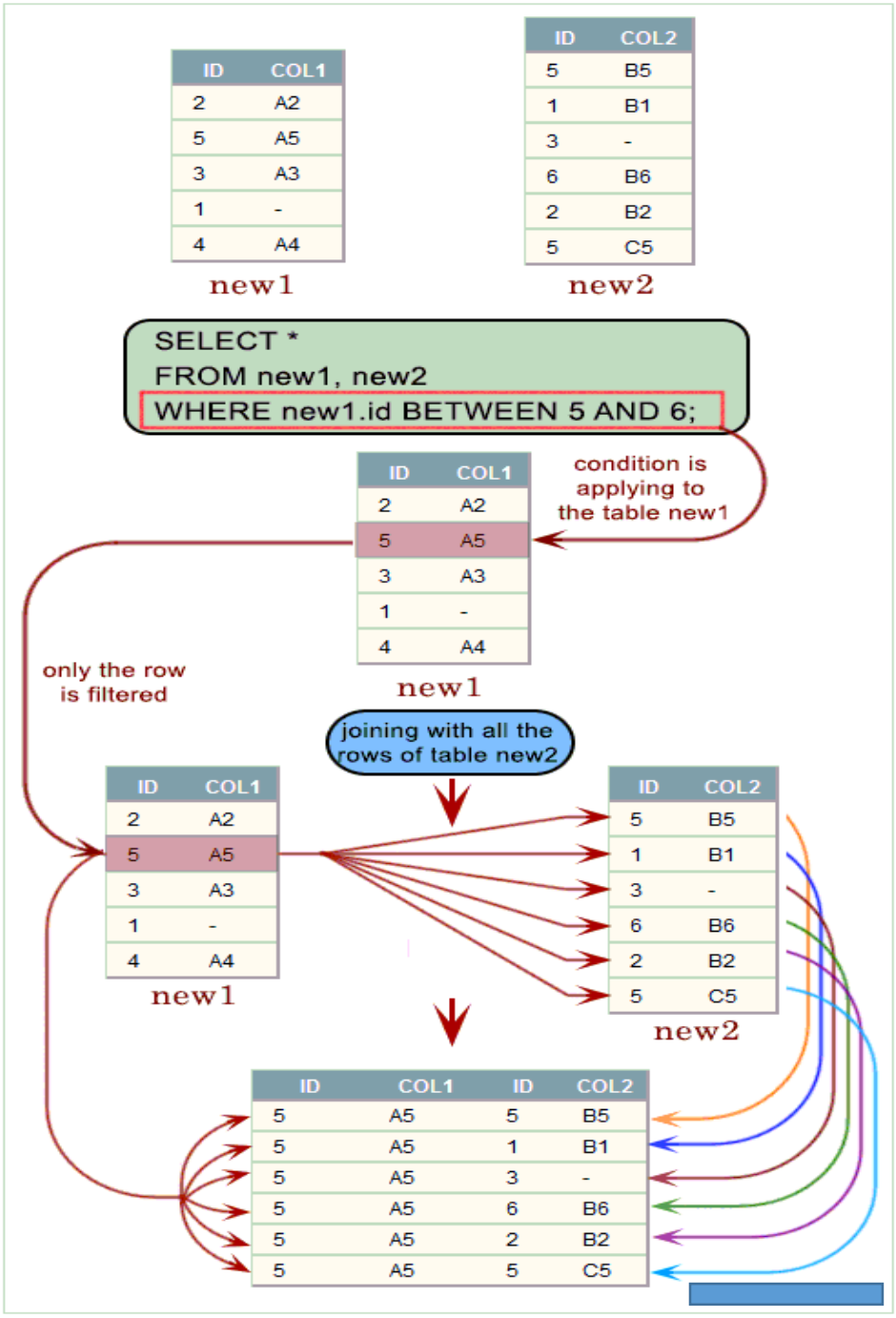
SQL Code:

```
1  SELECT agents.agent_name,customer.cust_name,
2  customer.cust_city
3  FROM agents,customer
4  WHERE agents.working_area=customer.cust_city;
```

NON EQUI JOIN

The SQL NON EQUI JOIN uses comparison operator instead of the equal sign like **>, <, >=, <=** along with conditions.

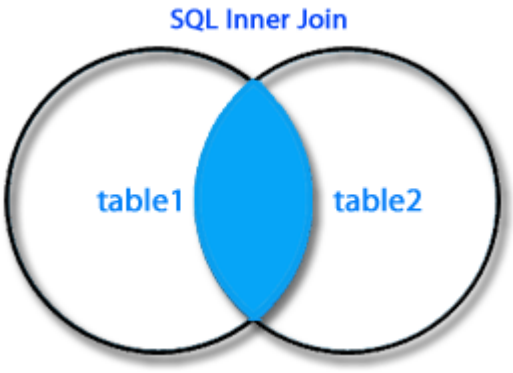
Pictorial presentation of SQL Non Equi Join:



What is Inner Join in SQL?

The INNER JOIN selects all rows from both participating tables as long as there is a match between the columns. An SQL INNER JOIN is same as JOIN clause, combining rows from two or more tables.

Pictorial presentation of SQL Inner Join:



Syntax:

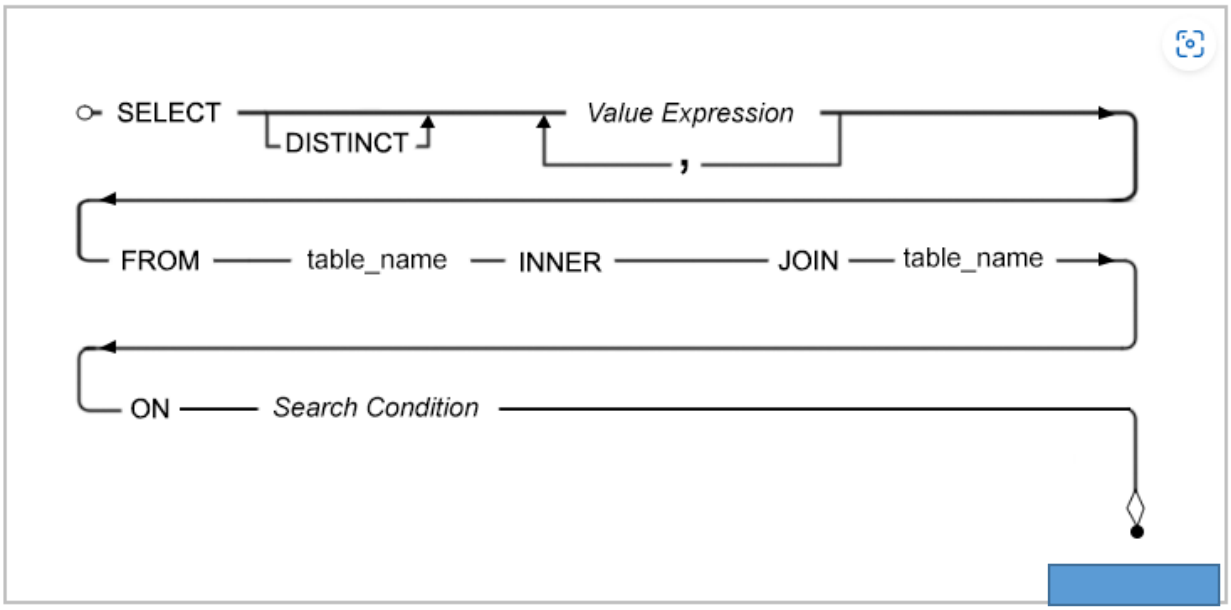
```
SELECT *
FROM table1 INNER JOIN table2
ON table1.column_name = table2.column_name;
```

OR

```
SELECT *
FROM table1
JOIN table2
ON table1.column_name = table2.column_name;
```

The INNER JOIN in SQL joins two tables according to the matching of a certain criteria using a comparison operator.

Syntax diagram - INNER JOIN



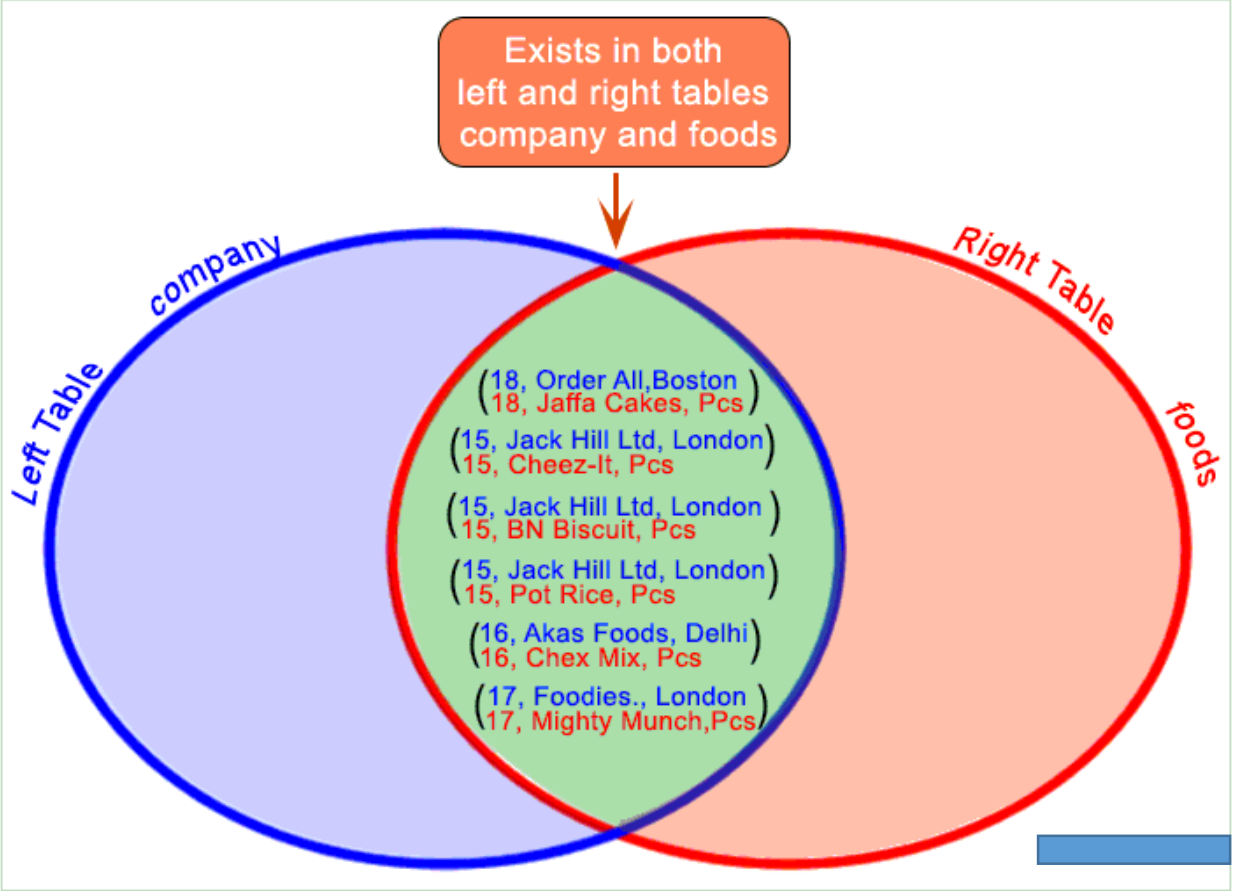
SQL Code:

```sql
SELECT foods.item_name,foods.item_unit,
company.company_name,company.company_city
FROM foods
INNER JOIN company
ON foods.company_id =company.company_id;
```

SQL Code:

```sql
SELECT foods.item_name,foods.item_unit,
company.company_name,company.company_city
FROM foods
JOIN company
ON foods.company_id =company.company_id;
```

Pictorial Presentation of SQL Inner Join of Company and Foods Tables:



To get all the columns from foods and company table after joining, with the following condition –

SQL Code:

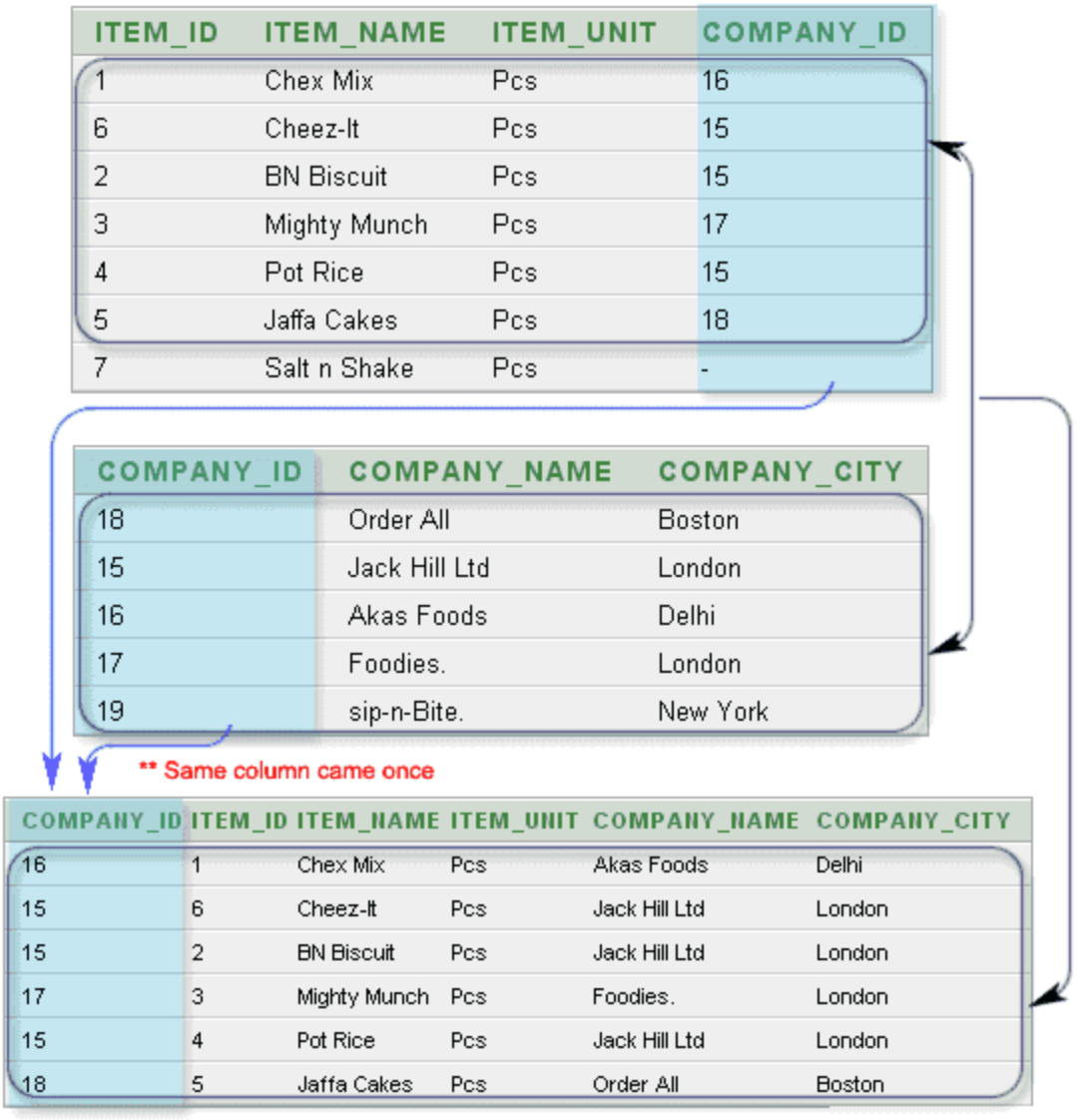
```
1  SELECT *
2  FROM foods
3  JOIN company
4  ON foods.company_id =company.company_id;
```

What is Natural Join in SQL?

We have already learned that an EQUI JOIN performs a JOIN against equality or matching column(s) values of the associated tables and an equal sign (=) is used as comparison operator in the where clause to refer equality.

The SQL NATURAL JOIN is a type of EQUI JOIN and is structured in such a way that, columns with the same name of associated tables will appear once only.

Pictorial presentation of the above SQL Natural Join:



ITEM_ID	ITEM_NAME	ITEM_UNIT	COMPANY_ID
1	Chex Mix	Pcs	16
6	Cheez-It	Pcs	15
2	BN Biscuit	Pcs	15
3	Mighty Munch	Pcs	17
4	Pot Rice	Pcs	15
5	Jaffa Cakes	Pcs	18
7	Salt n Shake	Pcs	-

COMPANY_ID	COMPANY_NAME	COMPANY_CITY
18	Order All	Boston
15	Jack Hill Ltd	London
16	Akas Foods	Delhi
17	Foodies.	London
19	sip-n-Bite.	New York

** Same column came once

COMPANY_ID	ITEM_ID	ITEM_NAME	ITEM_UNIT	COMPANY_NAME	COMPANY_CITY
16	1	Chex Mix	Pcs	Akas Foods	Delhi
15	6	Cheez-It	Pcs	Jack Hill Ltd	London
15	2	BN Biscuit	Pcs	Jack Hill Ltd	London
17	3	Mighty Munch	Pcs	Foodies.	London
15	4	Pot Rice	Pcs	Jack Hill Ltd	London
18	5	Jaffa Cakes	Pcs	Order All	Boston

Natural Join: Guidelines

- The associated tables have one or more pairs of identically named columns.
- The columns must be the same data type.
- Don’t use ON clause in a natural join.

Syntax:

```
SELECT *
FROM table1
NATURAL JOIN table2;
```

SQL Code:

```
1  SELECT *
2  FROM foods
3  NATURAL JOIN company;
```

Difference between natural join and inner join

There is one significant difference between INNER JOIN and NATURAL JOIN is the number of columns returned.

SQL Code:

```
1  SELECT *
2  FROM company
3  INNER JOIN foods
4  ON company.company_id = foods.company_id;
```

Output:

```
COMPANY_ID COMPANY_NAME    COMPANY_CITY    ITEM_ID    ITEM_NAME    ITEM_UNIT     COMPANY_ID
 ---   ----         -----                ---    ----           ----               ----
```

SQL Code:

```
SELECT *
FROM company
NATURAL JOIN foods;
```

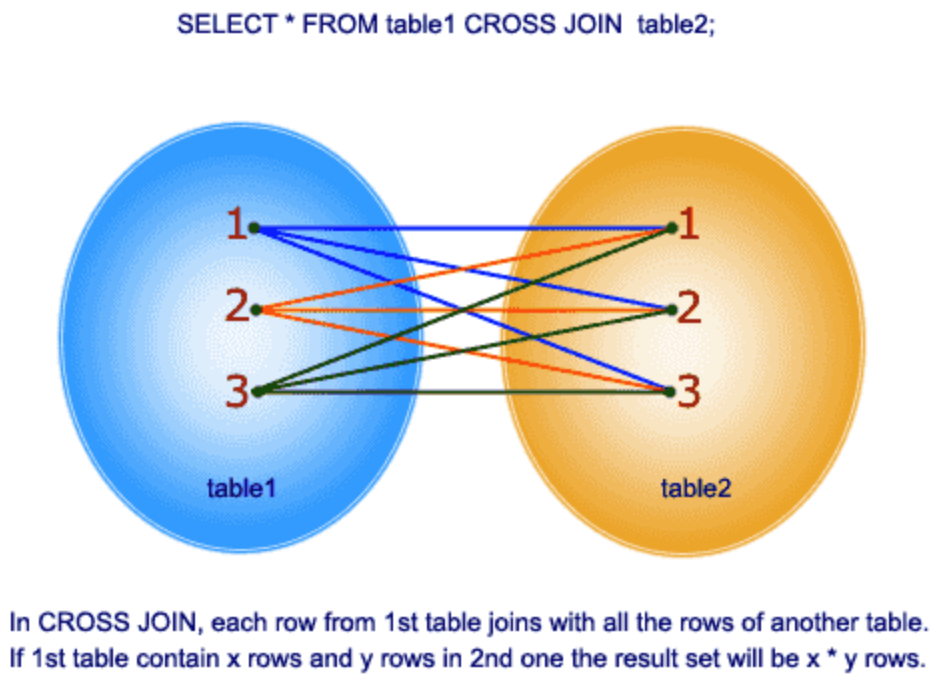
```
COMPANY ID COMPANY NAME    COMPANY CITY    ITEM ID    ITEM NAME       ITEM UNIT
---             -----         ---          ----       ---             ---
```

What is Cross Join in SQL?

The SQL CROSS JOIN produces a result set which is the number of rows in the first table multiplied by the number of rows in the second table if no WHERE clause is used along with CROSS JOIN.This kind of result is called as Cartesian Product.

If WHERE clause is used with CROSS JOIN, it functions like an INNER JOIN.

Pictorial Presentation of SQL Cross Join syntax:



An alternative way of achieving the same result is to use column names separated by commas after SELECT and mentioning the table names involved, after a FROM clause.

Syntax:

```
SELECT *
FROM table1
CROSS JOIN table2;
```

To get item name and item unit columns from foods table and company name, company city columns from company table, after a CROSS JOINING with these mentioned tables, the following SQL statement can be used:

SQL Code:

```
1  SELECT foods.item_name,foods.item_unit,
2  company.company_name,company.company_city
3  FROM foods
4  CROSS JOIN company;
```

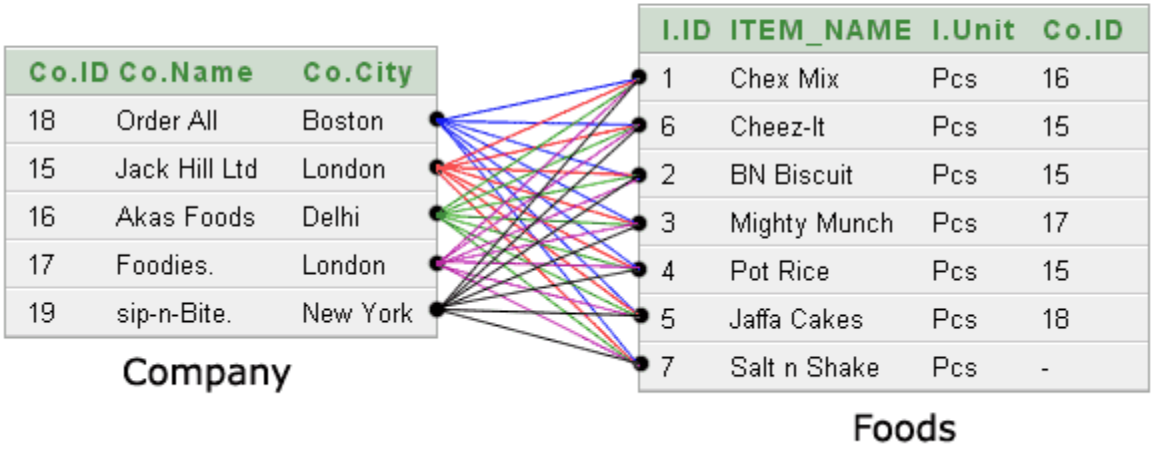
or

SQL Code:

```
1  SELECT foods.item_name,foods.item_unit,
2  company.company_name,company.company_city
3  FROM foods,company;
```

How cross joining happend into two tables

```
SELECT foods.item_name,foods.item_unit,
company.company_name,company.company_city
FROM foods
CROSS JOIN company;
```



Co.ID	Co.Name	Co.City
18	Order All	Boston
15	Jack Hill Ltd	London
16	Akas Foods	Delhi
17	Foodies.	London
19	sip-n-Bite.	New York

Company

I.ID	ITEM_NAME	I.Unit	Co.ID
1	Chex Mix	Pcs	16
6	Cheez-It	Pcs	15
2	BN Biscuit	Pcs	15
3	Mighty Munch	Pcs	17
4	Pot Rice	Pcs	15
5	Jaffa Cakes	Pcs	18
7	Salt n Shake	Pcs	-

Foods

More presentaion of the Sql Query output:

OUTER JOIN

The SQL OUTER JOIN returns all rows from both the participating tables which satisfy the join condition along with rows which do not satisfy the join condition. The SQL OUTER JOIN operator (+) is used only on one side of the join condition only.

Syntax:

```
Select *
FROM table1, table2
WHERE conditions [+];
```

To get company name and company id columns from company table and company id, item name, item unit columns from foods table, after an OUTER JOINING with these mentioned tables, the following SQL statement can be used:

SQL Code:

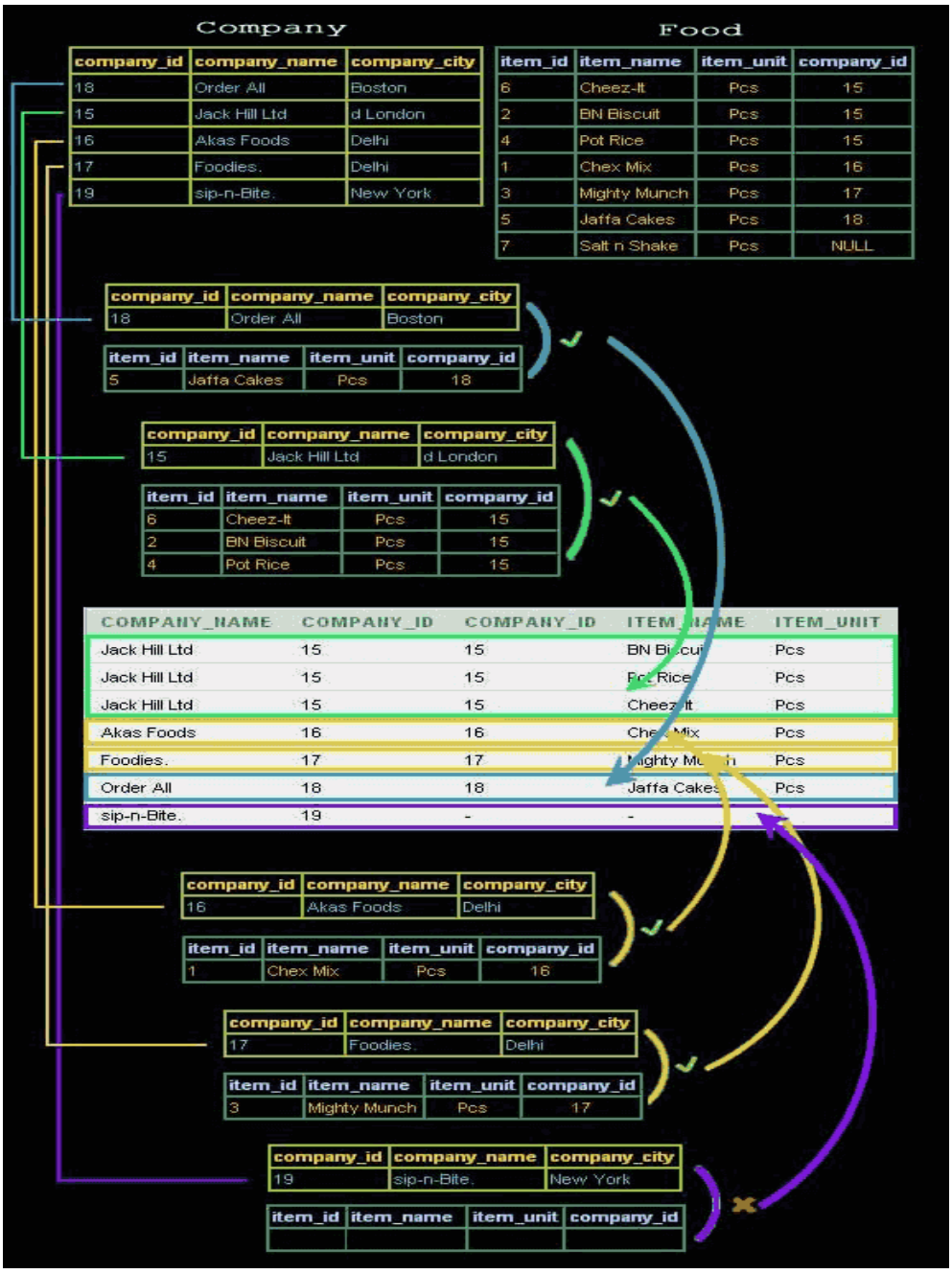
```
1  SELECT company.company_name,company.company_id,
2  foods.company_id,foods.item_name,foods.item_unit
3  FROM company, foods
4  WHERE company.company_id = foods.company_id(+);
```

Explanation:

This SQL statement would return all rows from the company table and only those rows from the foods table where the joined fields are equal.

The (+) after the foods.company_id field indicates that, if a company_id value in the company table does not exist in the foods table, all fields in the foods table will be displayed as NULL in the result set.

Pictorial Presentation of SQL Outer Join



Output:

```
COMPANY_NAME     COMPANY_ID COMPANY_ID ITEM_NAME        ITEM_UNIT
---------------- ---------- ---------- ---------------- ----------
Akas Foods       16         16         Chex Mix         Pcs
Jack Hill Ltd    15         15         Cheez-It         Pcs
Jack Hill Ltd    15         15         BN Biscuit       Pcs
Foodies.         17         17         Mighty Munch     Pcs
Jack Hill Ltd    15         15         Pot Rice         Pcs
Order All        18         18         Jaffa Cakes      Pcs
sip-n-Bite.      19
```

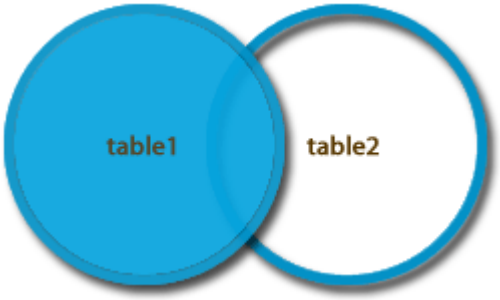
The subtypes of SQL OUTER JOIN

- LEFT OUTER JOIN or LEFT JOIN
- RIGHT OUTER JOIN or RIGHT JOIN
- FULL OUTER JOIN

LEFT JOIN

The SQL LEFT JOIN (specified with the keywords LEFT JOIN and ON) joins two tables and fetches all matching rows of two tables for which the SQL-expression is true, plus rows from the frist table that do not match any row in the second table.

Pictorial presentation of SQL Left Join:

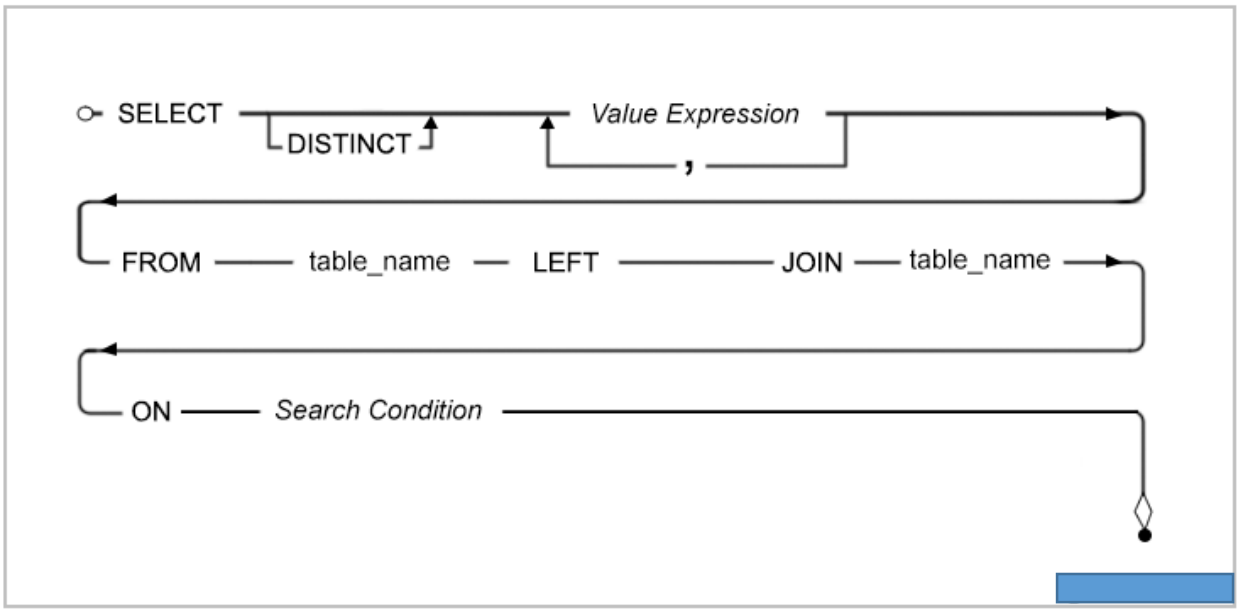


Left Join: Syntax

```
SELECT *
FROM table1
LEFT [ OUTER ] JOIN table2
ON table1.column_name=table2.column_name;
```

SQL LEFT join fetches a complete set of records from table1, with the matching records (depending on the availability) in table2. The result is NULL in the right side when no matching will take place.

Syntax diagram - LEFT JOIN



Example of SQL Left Join

To get company name and company id columns from company table and company id, item name, item unit columns from foods table, after an OUTER JOINING with these mentioned tables, the following SQL statement can be used.

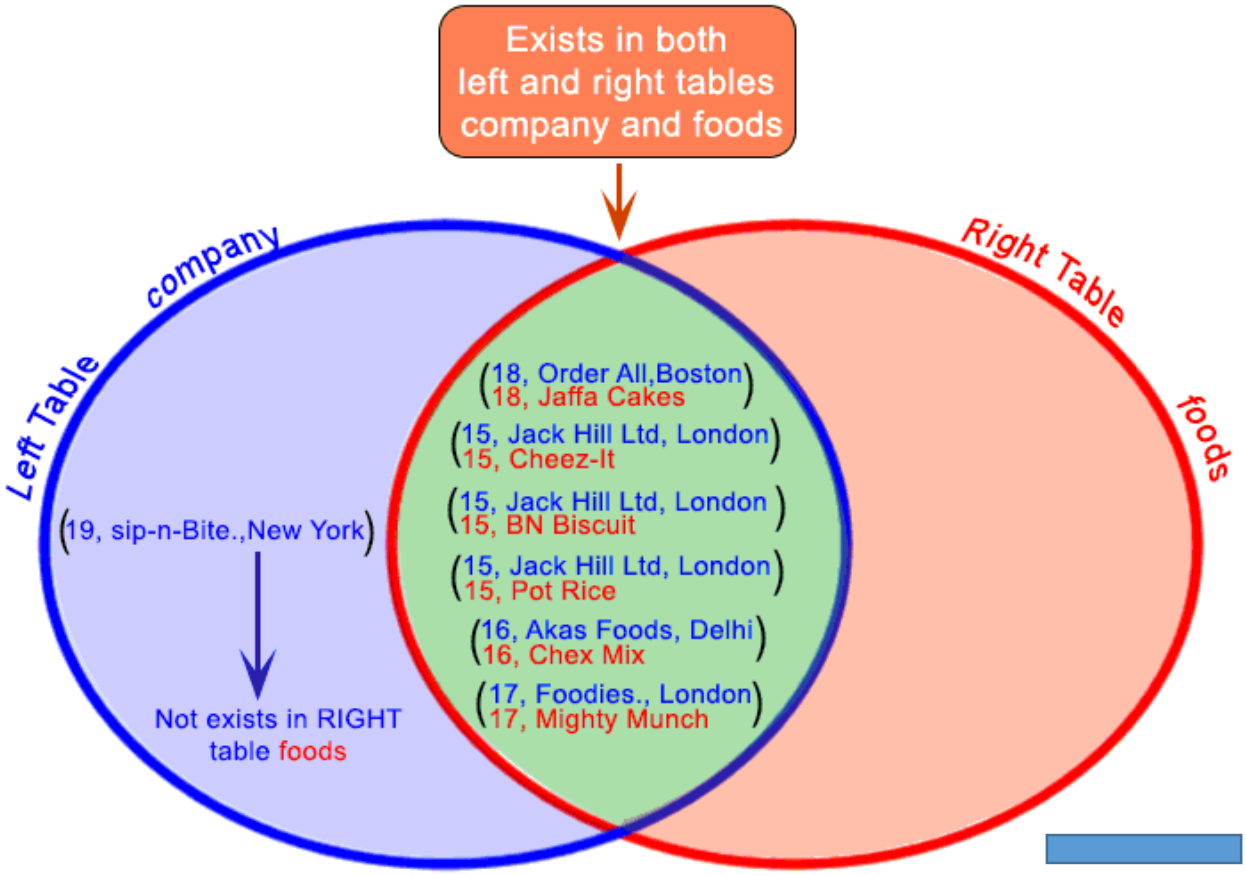
SQL Code:

```
1  SELECT company.company_id,company.company_name,
2  company.company_city,foods.company_id,foods.item_name
3  FROM   company
4  LEFT JOIN foods
5  ON company.company_id = foods.company_id;
```

Explanation:

This SQL statement would return all rows from the company table and only those rows from the foods table where the joined fields are equal and if the ON clause matches no records in the 'foods' table, the join will still return rows, but the NULL in each column of the right table.

Pictorial Presentation of the above example SQL Left Join:



Example of SQL Left Join using multiple columns

To filtered out those bill number, item name and the bill amount for each bill which bill amount exceeds the value 500 and must be available at the food stall, the following SQL statement can be used :

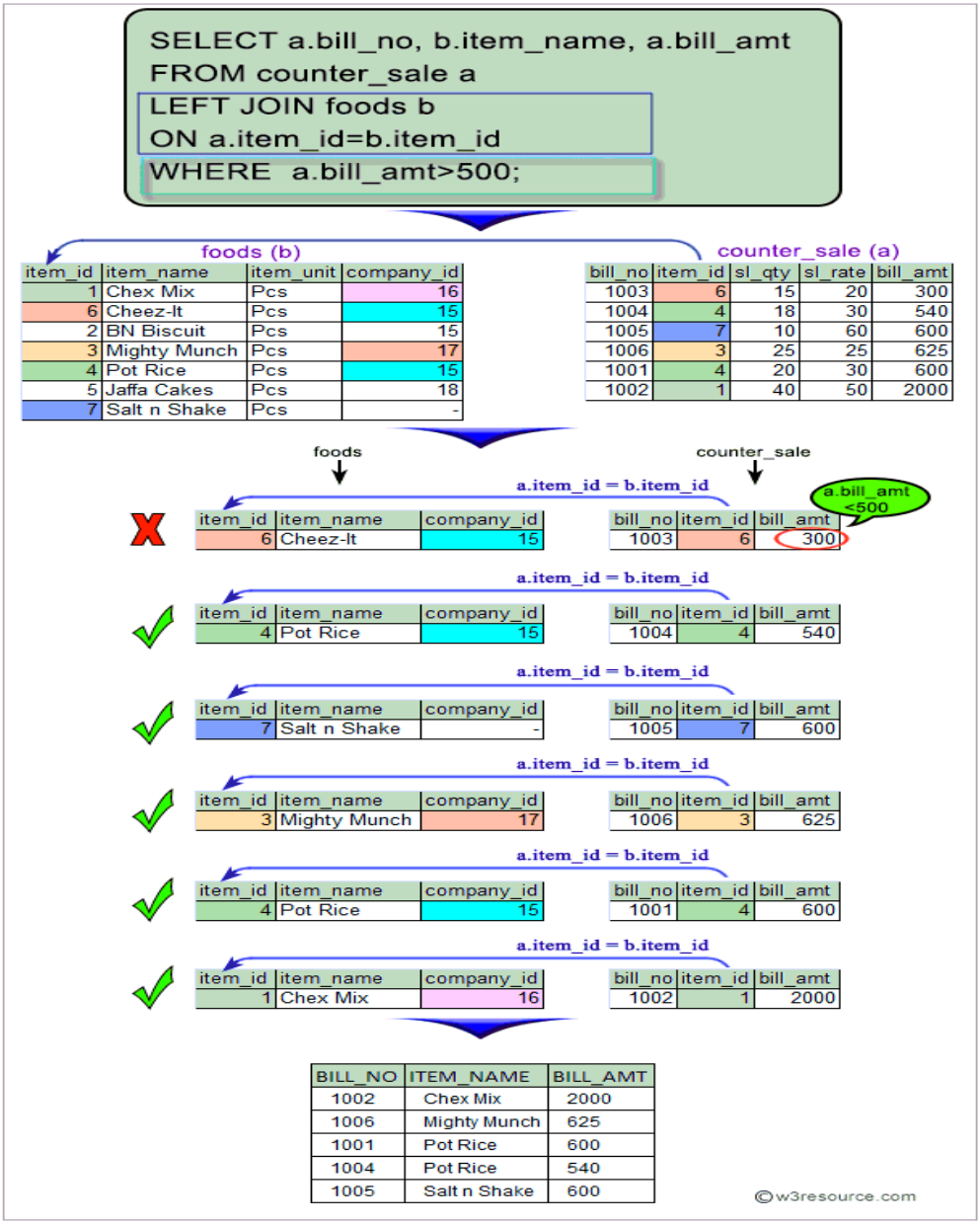
SQL Code:

```sql
SELECT a.bill_no, b.item_name, a.bill_amt
FROM counter_sale a
LEFT JOIN foods b
ON a.item_id=b.item_id
WHERE  a.bill_amt>500;
```

Explanation:

This SQL statement will first join all rows from the counter_sale table and only those rows from the foods table where the joined fields are equal and if the ON clause matches no records in the foods table, the join will still return rows, but the NULL in each column of right table, therefore eliminates those rows which bill amount is less than or equal to 500.

Pictorial Presentation of SQL Left Join using Multiple Columns:



Example of SQL Left Join using multiple tables

To filtered out those bill number, item name, company name and city and the bill amount for each bill, which items are available in foods table, and their manufacturer must have enlisted to supply that item, and no NULL value for manufacturer are not allowed, the following SQL statement can be used:

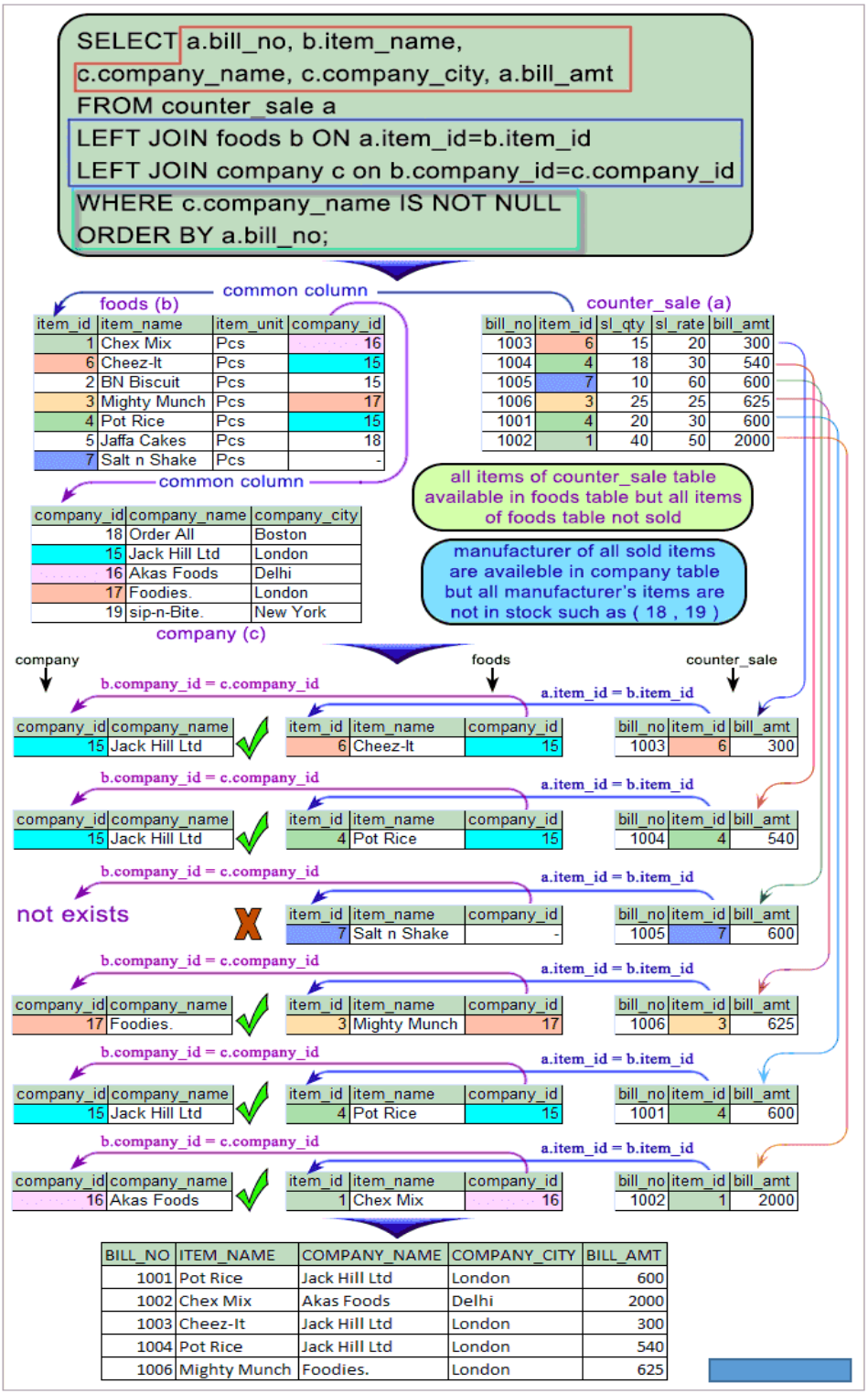
SQL Code:

```sql
SELECT a.bill_no, b.item_name,c.company_name,
c.company_city, a.bill_amt
FROM counter_sale a
LEFT JOIN foods b ON a.item_id=b.item_id
LEFT JOIN company c ON b.company_id=c.company_id
WHERE c.company_name IS NOT NULL
ORDER BY a.bill_no;
```

Explanation:

This SQL statement will first join all rows from the counter_sale table and only those rows from the foods table where the joined fields are matching and if the ON clause matches no records in the foods table, the join will still return rows, but the NULL in each column of the right table. Therefore this result will join with company table and all rows from result table and matched and unmatched rows from company table will also come, but for the unmatched rows of company table, the column value will be NULL. Therefore the WHERE clause will eliminate those rows which company name column value is NULL and after that, the ORDER BY clause will arrange the rows in ascending order according to the bill number.

Pictorial Presentation of SQL Left Join using Multiple Tables:



BILL_NO	ITEM_NAME	COMPANY_NAME	COMPANY_CITY	BILL_AMT
1001	Pot Rice	Jack Hill Ltd	London	600
1002	Chex Mix	Akas Foods	Delhi	2000
1003	Cheez-It	Jack Hill Ltd	London	300
1004	Pot Rice	Jack Hill Ltd	London	540
1006	Mighty Munch	Foodies.	London	625

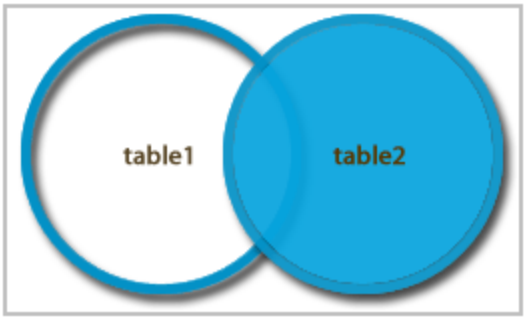
What is the difference between Left Join and Left Outer Join in SQL?

There is actually no difference between a left join and a left outer join – both of them refer to the similar operation in SQL.

RIGHT JOIN

The SQL RIGHT JOIN, joins two tables and fetches rows based on a condition, which is matching in both the tables (before and after the JOIN clause mentioned in the syntax below) , and the unmatched rows will also be available from the table written after the JOIN clause (mentioned in the syntax below).

Pictorial presentation of SQL Right Join:

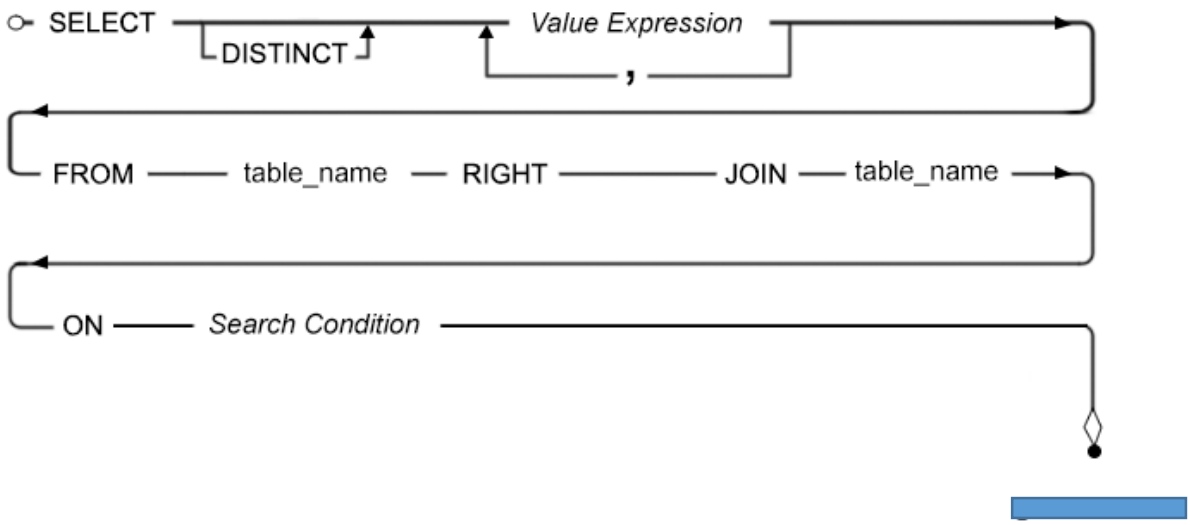


Syntax:

```
SELECT *
FROM table1
RIGHT [ OUTER ] JOIN table2
ON table1.column_name=table2.column_name;
```

SQL RIGHT join fetches a complete set of records from table2, i.e. the rightmost table after JOIN clause, with the matching records (depending on the availability) in table1. The result is NULL in the left side when no matching will take place.

Syntax diagram - SQL Right Join



To get company ID, company name and company city columns from company table and company ID, item name columns from foods table, after an OUTER JOINING with these mentioned tables, the following SQL statement can be used:

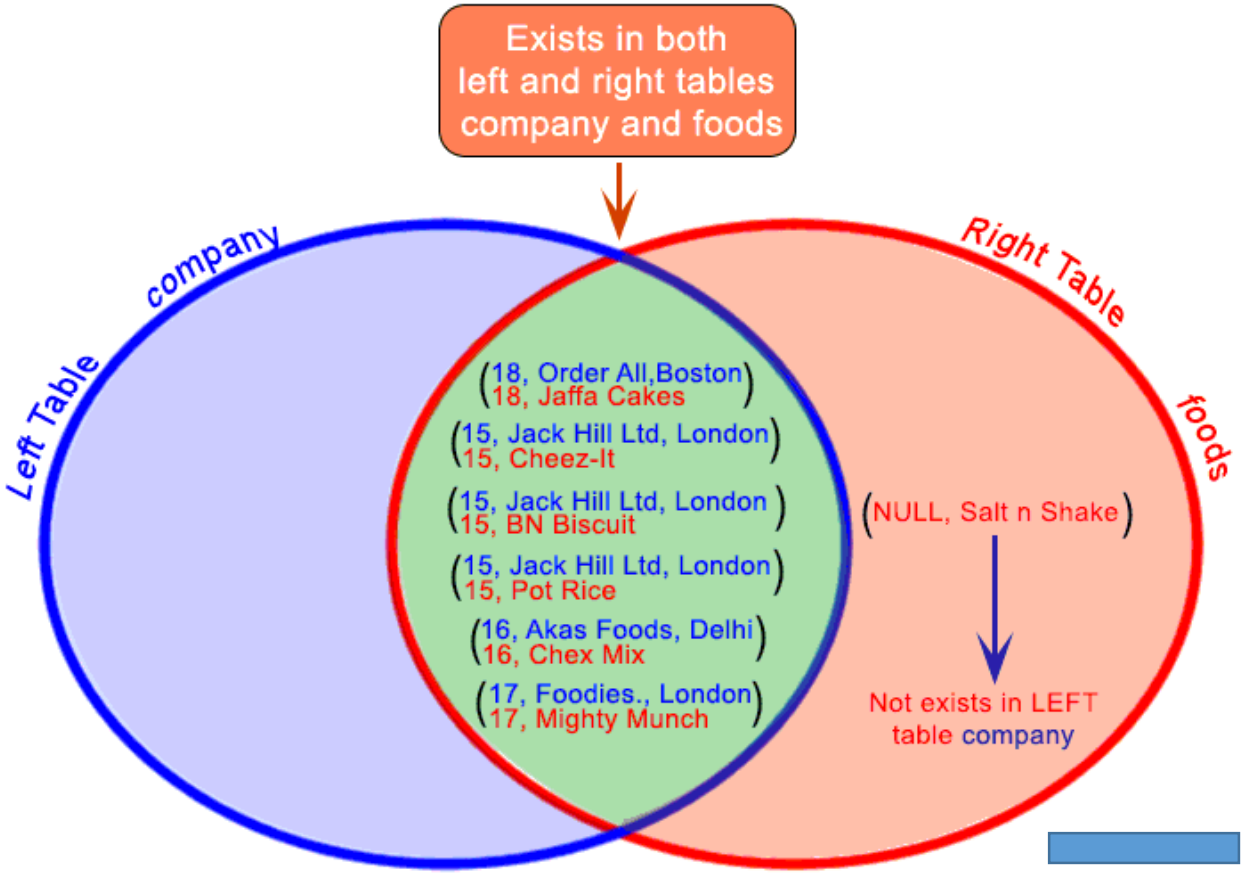
SQL Code:

```
1  SELECT company.company_id,company.company_name,
2  company.company_city,foods.company_id,foods.item_name
3  FROM   company
4  RIGHT JOIN foods
5  ON company.company_id = foods.company_id;
```

Explanation:

This SQL statement would return all rows from the foods table and only those rows from the company table where the joined fields are equal and if the ON clause matches no records in the company table, the join will still return rows, but the NULL in each column of company table.

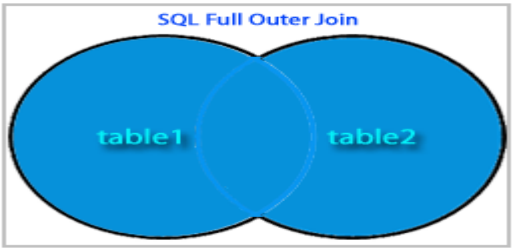
Pictorial Presentation of the above example:

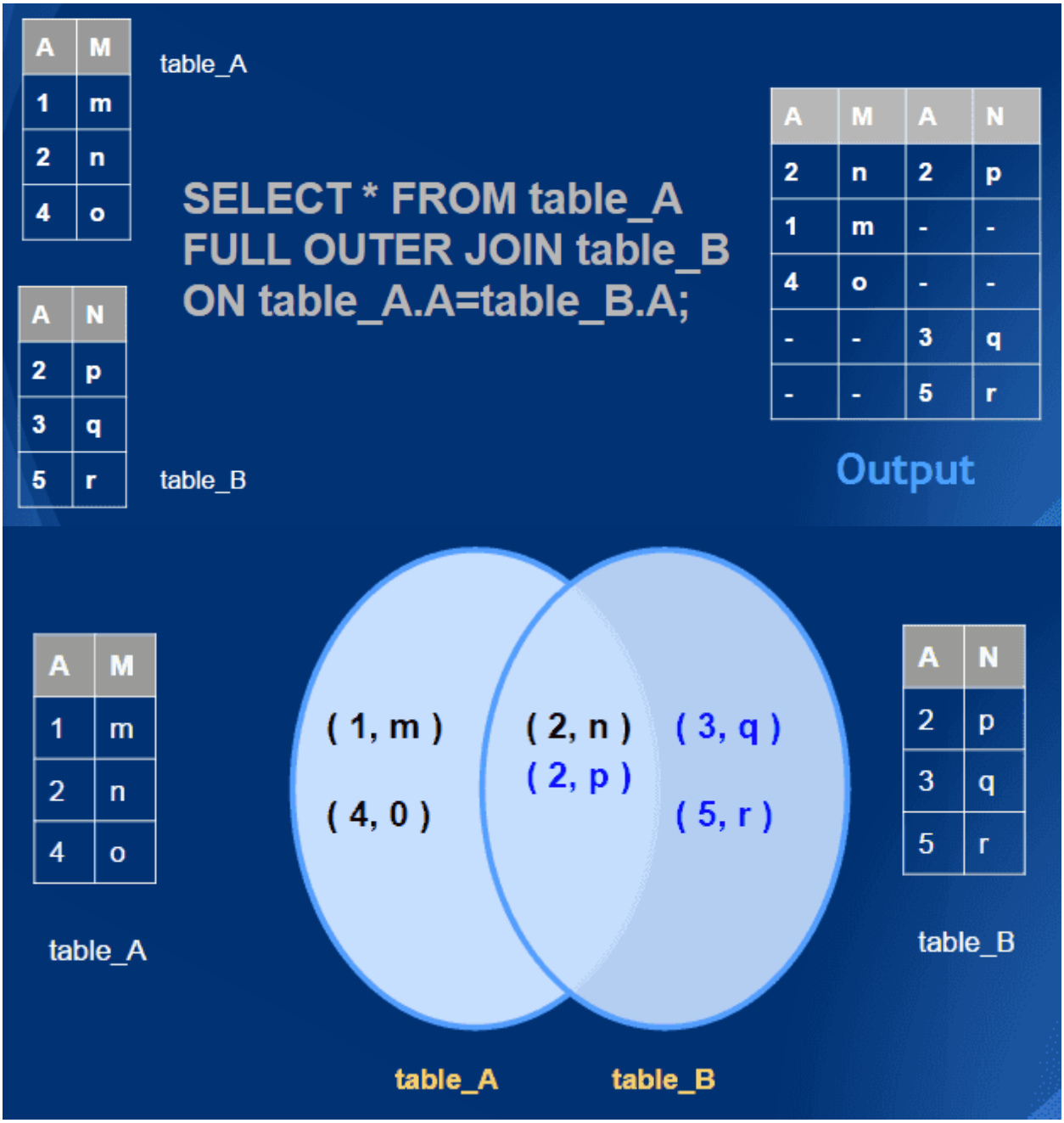


What is Full Outer Join in SQL?

In SQL the FULL OUTER JOIN combines the results of both left and right outer joins and returns all (matched or unmatched) rows from the tables on both sides of the join clause

Pictorial Presentation: SQL Full Outer Join

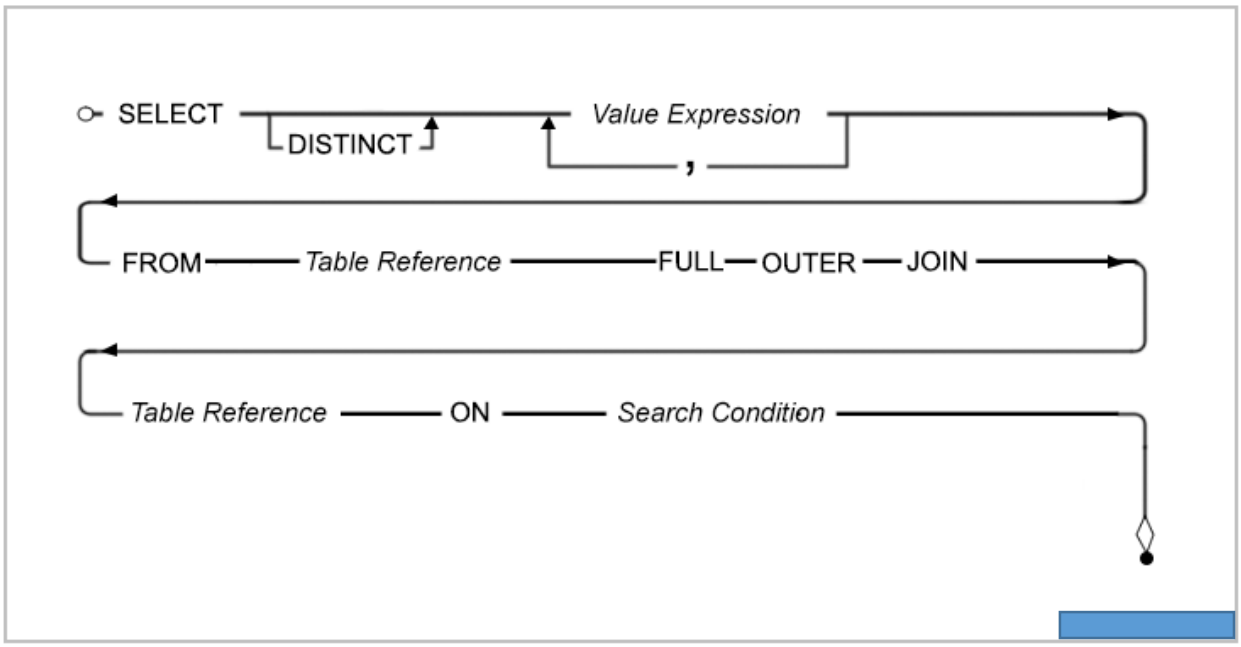




Syntax:

```
SELECT *
FROM table1
FULL OUTER JOIN table2
ON table1.column_name=table2.column_name;
```

Syntax diagram - FULL OUTER JOIN



Example: SQL Full Outer Join

Let’s combine the same two tables using a full join.

table_A

A	M
1	m
2	n
4	o

table_B

A	N
2	p
3	q
5	r

SQL Code:

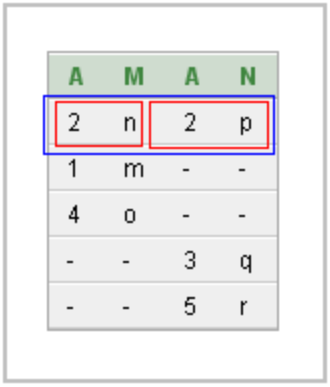
```
1  SELECT * FROM table_A
2  FULL OUTER JOIN table_B
3  ON table_A.A=table_B.A;
```

Output:

A	M	A	N
2	n	2	p
1	m	-	-
4	o	-	-
-	-	3	q
-	-	5	r

Because this is a full join, all rows (both matching and nonmatching) from both tables are included in the output. There is only one match between table table_A and table table_B, so **only one row of output displays values in all columns**. All remaining rows of output contain only values from table table_A or table table_B, with the remaining columns set to missing values

only one row of output displays values in all columns explain below –



A	M	A	N
2	n	2	p
1	m	-	-
4	o	-	-
-	-	3	q
-	-	5	r

FULL OUTER JOIN using WHERE clause

We can include a WHERE clause with a FULL OUTER JOIN to get return only those rows where no matching data between the joining tables are exist.

The following query returns only those company that have no matching food product in foods, as well as that food product in foods that are not matched to the listed company.

```
SELECT a.company_id AS "a.ComID",
a.company_name AS "C_Name",
b.company_id AS "b.ComID",
b.item_name AS "I_Name"
FROM   company a
FULL OUTER JOIN foods b
ON a.company_id = b.company_id
WHERE a.company_id IS NULL
OR b.company_id IS NULL
ORDER BY company_name;
```

Join a table to itself

A SELF JOIN is another type of join in SQL which is used to join a table to itself, especially when the table has a FOREIGN KEY which references its own PRIMARY KEY.

In this join, the participating table appears twice after the FROM clause and is followed by aliases for the tables that qualify column names in the join condition

In this join, those rows are returned from the table which are satisfying the conditions.

Example:

Sample table: company

```
+------------+---------------+--------------+
| COMPANY_ID | COMPANY_NAME  | COMPANY_CITY |
+------------+---------------+--------------+
| 18         | Order All     | Boston       |
| 15         | Jack Hill Ltd | London       |
| 16         | Akas Foods    | Delhi        |
| 17         | Foodies.      | London       |
| 19         | sip-n-Bite.   | New York     |
+------------+---------------+--------------+
```

To get 'company_name' and 'company_city' from 'company' table which is entitled as alias 'a' and 'company_name' form 'company' table which is entitled as alias 'b' after an SELF JOINING with a table itself, the following SQL statement can be used:

SQL Code:

```
1  SELECT a.company_name,b.company_name,a.company_city
2  FROM company a, company b
3  WHERE a.company_city=b.company_city;
```

Explanation:

This is a note for the example:
'a' and 'b' are aliases for the table 'company'.

the a.company_city=b.company_city excludes all pairs containing companies of different cities.

END