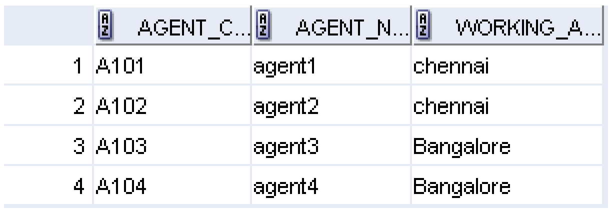
Ex: 1 (Equijoins)

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.

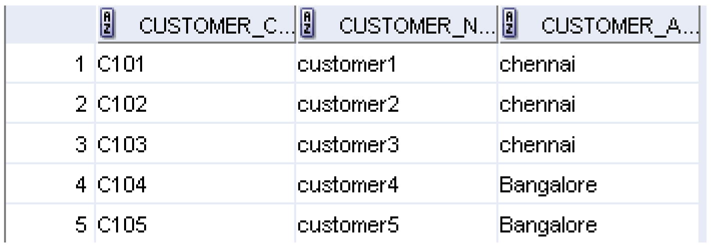
You may also perform EQUI JOIN by using JOIN keyword followed by ON keyword and then specifying names of the columns along with their associated tables to check equality.

SQL> select * form agents;



	AGENT_C...	AGENT_N...	WORKING_A...
1	A101	agent1	chennai
2	A102	agent2	chennai
3	A103	agent3	Bangalore
4	A104	agent4	Bangalore

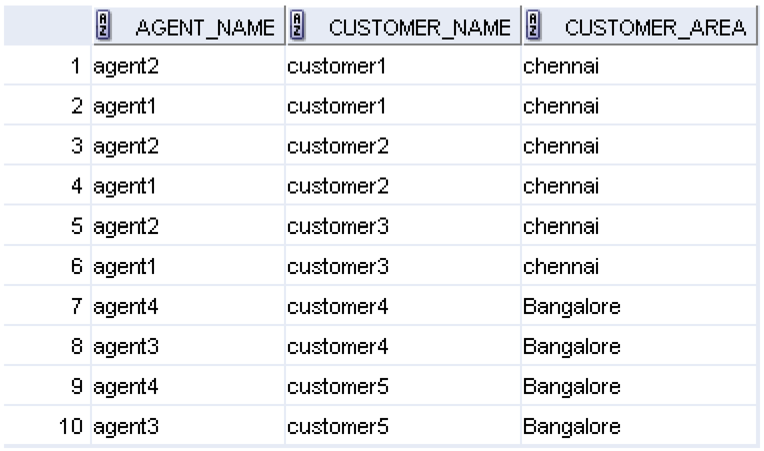
SQL> select * from customers;



	CUSTOMER_C...	CUSTOMER_N...	CUSTOMER_A...
1	C101	customer1	chennai
2	C102	customer2	chennai
3	C103	customer3	chennai
4	C104	customer4	Bangalore
5	C105	customer5	Bangalore

SQL> select agents.agent_name, customers.customer_name, customers.customer_area

from agents, customers where agents.working_area = customers.customer_area;



	AGENT_NAME	CUSTOMER_NAME	CUSTOMER_AREA
1	agent2	customer1	chennai
2	agent1	customer1	chennai
3	agent2	customer2	chennai
4	agent1	customer2	chennai
5	agent2	customer3	chennai
6	agent1	customer3	chennai
7	agent4	customer4	Bangalore
8	agent3	customer4	Bangalore
9	agent4	customer5	Bangalore
10	agent3	customer5	Bangalore

SQL> select a.agent_name, c.customer_name, c.customer_area

from agents a, customers c where a.working_area = c.customer_area;

SQL> select a.agent_name, c.customer_name, c.customer_area

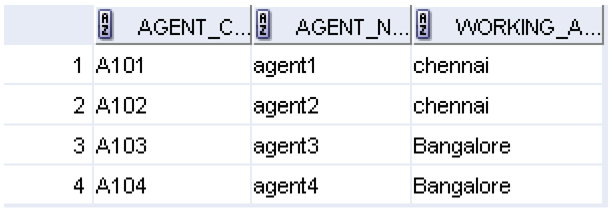
from agents a **join** customers c **on** a.working_area=c.customer_area;

SQL> select agents.agent_name, customers.customer_name, customers.customer_area

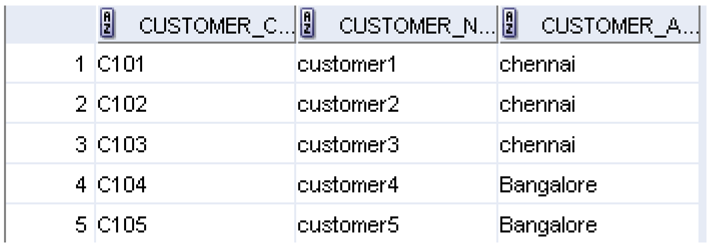
from agents **join** customers **on** agents.working_area = customers.customer_area;

Ex: 2 (Non Equijoins)

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



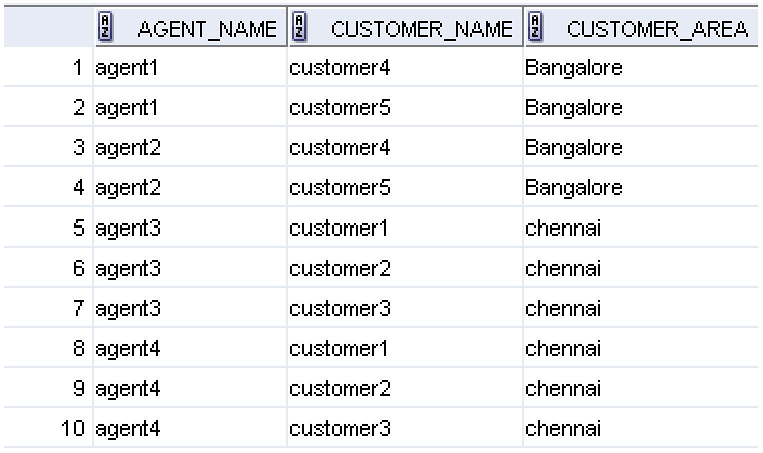
	AGENT_C...	AGENT_N...	WORKING_A...
1	A101	agent1	chennai
2	A102	agent2	chennai
3	A103	agent3	Bangalore
4	A104	agent4	Bangalore



	CUSTOMER_C...	CUSTOMER_N...	CUSTOMER_A...
1	C101	customer1	chennai
2	C102	customer2	chennai
3	C103	customer3	chennai
4	C104	customer4	Bangalore
5	C105	customer5	Bangalore

SQL> select agents.agent_name, customers.customer_name, customers.customer_area

from agents join customers on agents.working_area <> customers.customer_area;



	AGENT_NAME	CUSTOMER_NAME	CUSTOMER_AREA
1	agent1	customer4	Bangalore
2	agent1	customer5	Bangalore
3	agent2	customer4	Bangalore
4	agent2	customer5	Bangalore
5	agent3	customer1	chennai
6	agent3	customer2	chennai
7	agent3	customer3	chennai
8	agent4	customer1	chennai
9	agent4	customer2	chennai
10	agent4	customer3	chennai

Ex: 3 (Inner Join)

The INNER JOIN will select 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 tables.

SQL> select agents.agent_name, customers.customer_name, customers.customer_area

from agents **inner join** customers **on** agents.working_area = customers.customer_area;

(output: Same as Equijoin)

Ex: 5 (Outer Join)

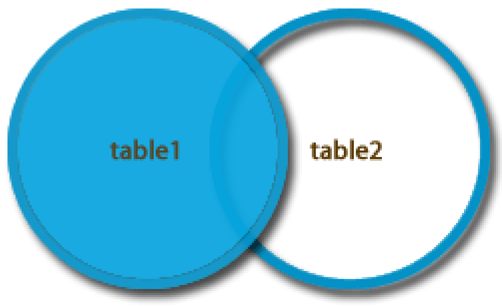
The SQL OUTER JOIN returns all rows from both the participating tables which satisfy the join condition along with rows which do not satisfy the join condition. The SQL OUTER JOIN operator (+) is used only on one side of the join condition only.

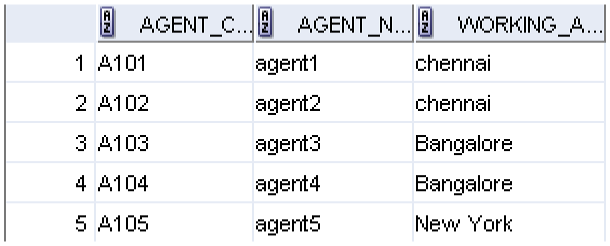
The subtypes of SQL OUTER JOIN

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

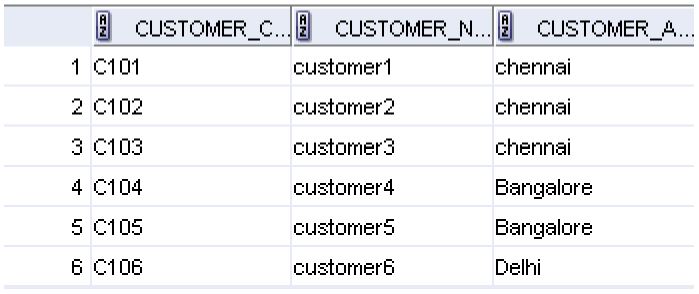
Ex: 6 (Left Outer Join or Left Join)

The SQL LEFT JOIN, joins two tables and fetches rows based on a condition, which are matching in both the tables, and the unmatched rows will also be available from the table before the JOIN clause.





	AGENT_C...	AGENT_N...	WORKING_A...
1	A101	agent1	chennai
2	A102	agent2	chennai
3	A103	agent3	Bangalore
4	A104	agent4	Bangalore
5	A105	agent5	New York

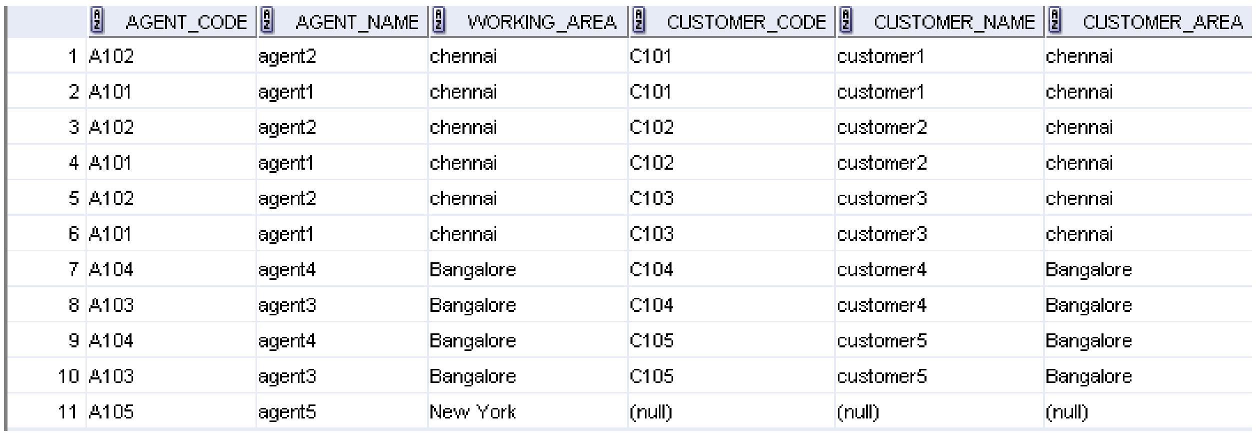


	CUSTOMER_C...	CUSTOMER_N...	CUSTOMER_A...
1	C101	customer1	chennai
2	C102	customer2	chennai
3	C103	customer3	chennai
4	C104	customer4	Bangalore
5	C105	customer5	Bangalore
6	C106	customer6	Delhi

SQL> select * from agents **left outer join** customers on agents.working_area = customers.customer_area;

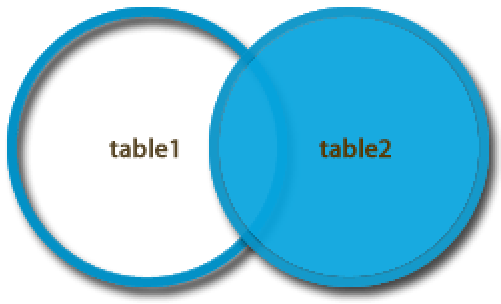
SQL> select * from agents **left join** customers on agents.working_area = customers.customer_area;

SQL> select * from agents , customers where agents.working_area = customers.customer_area(+);



	AGENT_CODE	AGENT_NAME	WORKING_AREA	CUSTOMER_CODE	CUSTOMER_NAME	CUSTOMER_AREA
1	A102	agent2	chennai	C101	customer1	chennai
2	A101	agent1	chennai	C101	customer1	chennai
3	A102	agent2	chennai	C102	customer2	chennai
4	A101	agent1	chennai	C102	customer2	chennai
5	A102	agent2	chennai	C103	customer3	chennai
6	A101	agent1	chennai	C103	customer3	chennai
7	A104	agent4	Bangalore	C104	customer4	Bangalore
8	A103	agent3	Bangalore	C104	customer4	Bangalore
9	A104	agent4	Bangalore	C105	customer5	Bangalore
10	A103	agent3	Bangalore	C105	customer5	Bangalore
11	A105	agent5	New York	(null)	(null)	(null)

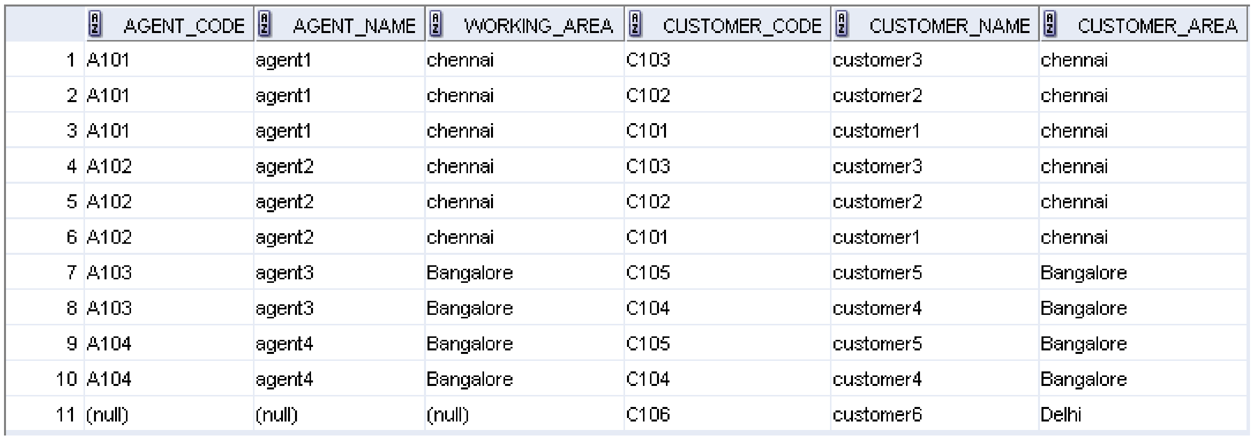
Ex: 7 (Right Outer Join or Right Join)



SQL> select * from agents right outer join customers on agents.working_area = customers.customer_area;

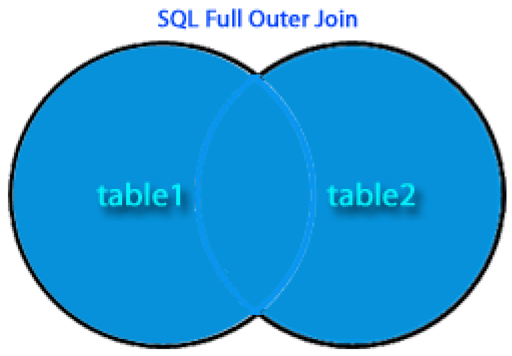
SQL> select * from agents right join customers on agents.working_area = customers.customer_area;

SQL> select * from agents, customers where agents.working_area(+) = customers.customer_area;



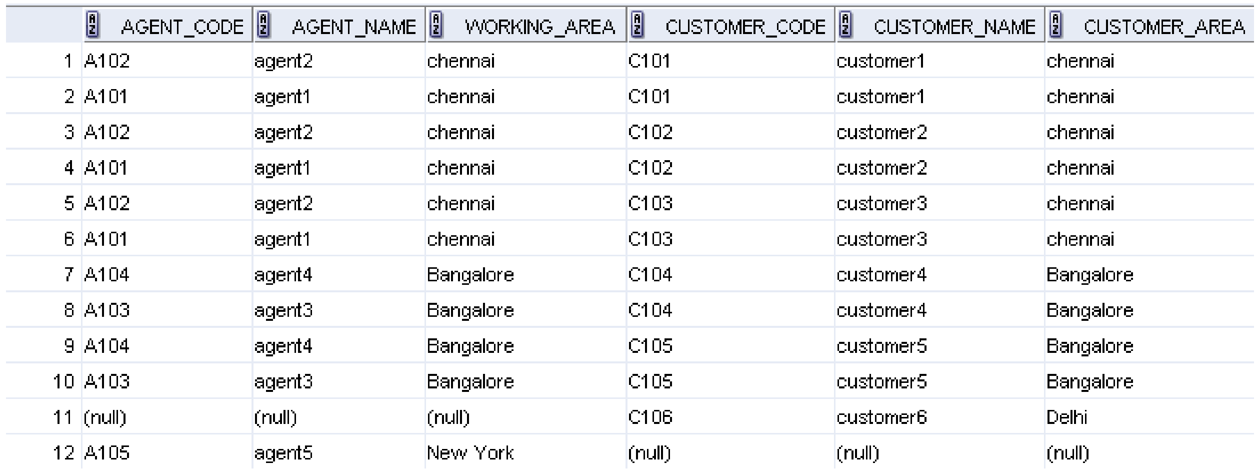
	AGENT_CODE	AGENT_NAME	WORKING_AREA	CUSTOMER_CODE	CUSTOMER_NAME	CUSTOMER_AREA
1	A101	agent1	chennai	C103	customer3	chennai
2	A101	agent1	chennai	C102	customer2	chennai
3	A101	agent1	chennai	C101	customer1	chennai
4	A102	agent2	chennai	C103	customer3	chennai
5	A102	agent2	chennai	C102	customer2	chennai
6	A102	agent2	chennai	C101	customer1	chennai
7	A103	agent3	Bangalore	C105	customer5	Bangalore
8	A103	agent3	Bangalore	C104	customer4	Bangalore
9	A104	agent4	Bangalore	C105	customer5	Bangalore
10	A104	agent4	Bangalore	C104	customer4	Bangalore
11	(null)	(null)	(null)	C106	customer6	Delhi

Ex: 8 (Full Outer Join)



SQL> select * from agents full outer join customers on agents.working_area = customers.customer_area;

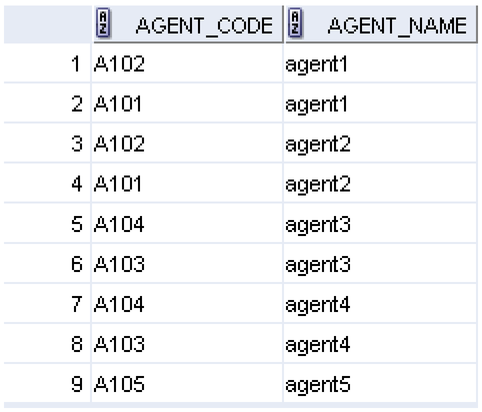
SQL> select * from agents full join customers on agents.working_area = customers.customer_area;



	AGENT_CODE	AGENT_NAME	WORKING_AREA	CUSTOMER_CODE	CUSTOMER_NAME	CUSTOMER_AREA
1	A102	agent2	chennai	C101	customer1	chennai
2	A101	agent1	chennai	C101	customer1	chennai
3	A102	agent2	chennai	C102	customer2	chennai
4	A101	agent1	chennai	C102	customer2	chennai
5	A102	agent2	chennai	C103	customer3	chennai
6	A101	agent1	chennai	C103	customer3	chennai
7	A104	agent4	Bangalore	C104	customer4	Bangalore
8	A103	agent3	Bangalore	C104	customer4	Bangalore
9	A104	agent4	Bangalore	C105	customer5	Bangalore
10	A103	agent3	Bangalore	C105	customer5	Bangalore
11	(null)	(null)	(null)	C106	customer6	Delhi
12	A105	agent5	New York	(null)	(null)	(null)

Ex: 9 (Self Join)

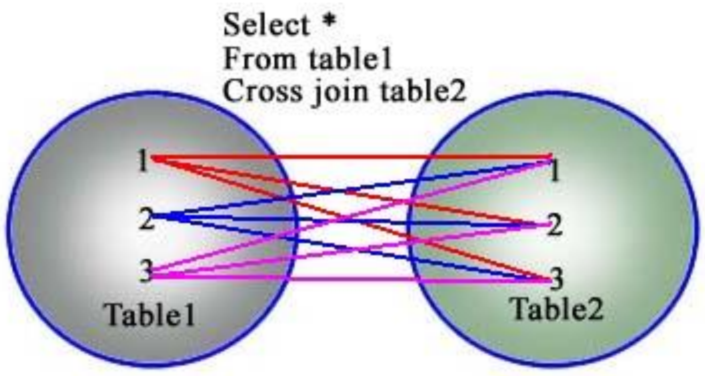
SQL> select a.agent_code, b.agent_name from agents a, agents b where a.working_area = b.working_area;



	AGENT_CODE	AGENT_NAME
1	A102	agent1
2	A101	agent1
3	A102	agent2
4	A101	agent2
5	A104	agent3
6	A103	agent3
7	A104	agent4
8	A103	agent4
9	A105	agent5

Ex: 4 (Cross Join)

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.



In cross joins, each row from first table joins with all the the rows of another table.
If 1st table contain x rows and y rows in 2nd one the result set will be x*y rows.

SQL> select agents.agent_name, customers.customer_name, customers.customer_area

from agents cross join customers;

Ex: 4 (Natural Join)

```
SQL> select agents.agent_name, customers.customer_name, customers.customer_area

from agents natural join customers;
```