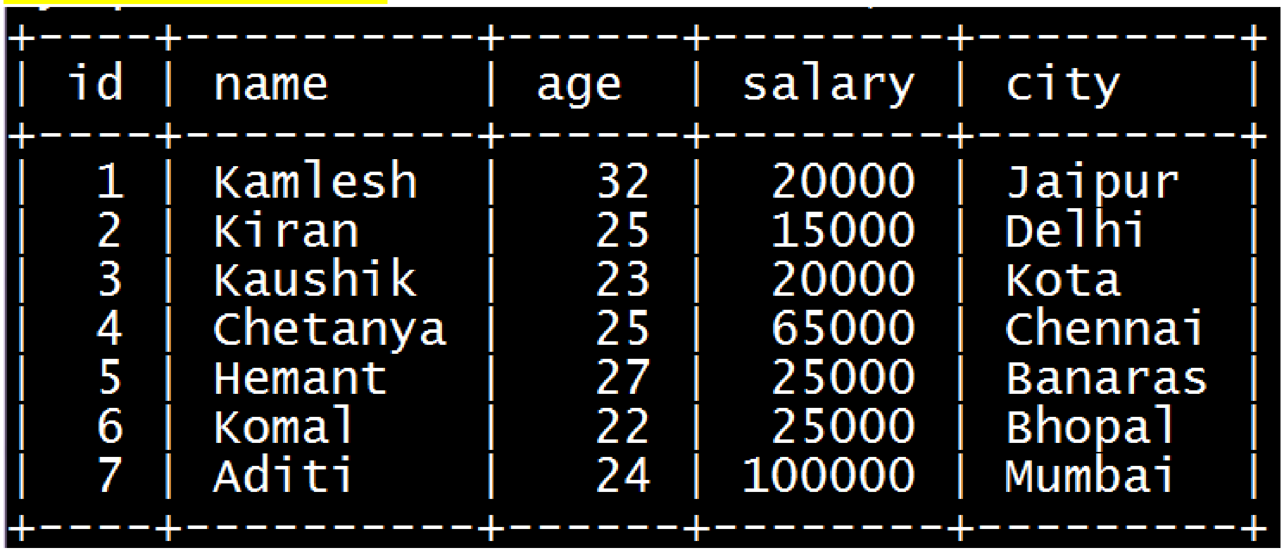
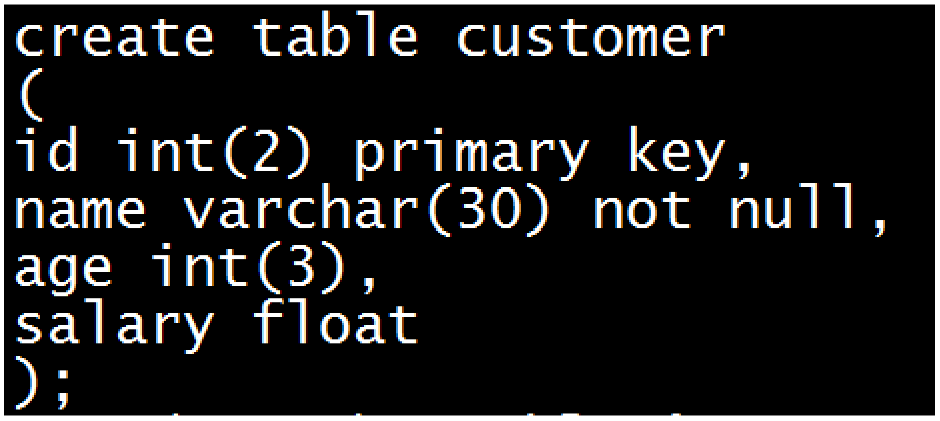
# SQL Join Commands

**The SQL Joins clause is used to combine records from two or more tables in a database. A JOIN is a means for combining fields from two tables by using values common to each**
**Consider the following two tables.**
**Here, it is noticeable that the join is performed in the WHERE clause. Several operators can be used to join tables, such as =, <, >, <>, <=, >=, !=, BETWEEN, LIKE, and NOT; they can all be used to join tables. However, the most common operator is the equal to symbol.**

**Table 1 – CUSTOMER**

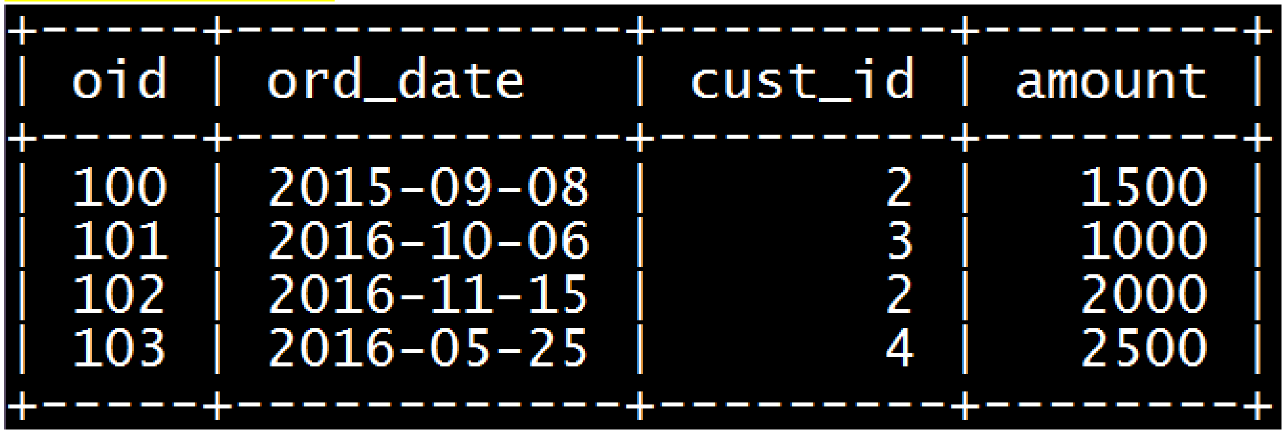


```
+----+-----------+-------+---------+----------+
| id | name      | age   | salary  | city     |
+----+-----------+-------+---------+----------+
|  1 | Kamlesh   |    32 |   20000 | Jaipur   |
|  2 | Kiran     |    25 |   15000 | Delhi    |
|  3 | Kaushik   |    23 |   20000 | Kota     |
|  4 | Chetanya  |    25 |   65000 | Chennai  |
|  5 | Hemant    |    27 |   25000 | Banaras  |
|  6 | Komal     |    22 |   25000 | Bhopal   |
|  7 | Aditi     |    24 |  100000 | Mumbai   |
+----+-----------+-------+---------+----------+
```

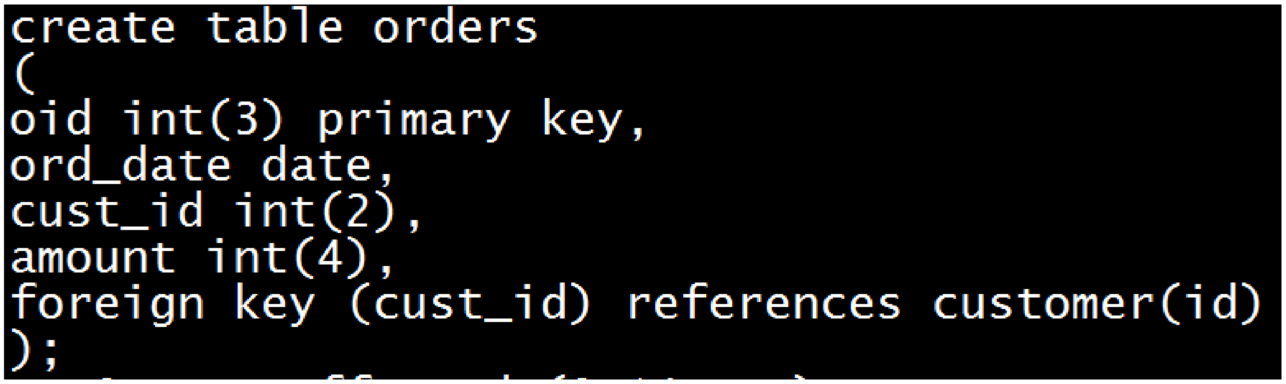


```
create table customer
(
id int(2) primary key,
name varchar(30) not null,
age int(3),
salary float
);
```

**Table 2 – ORDERS**



```
+-----+------------+---------+--------+
| oid | ord_date   | cust_id | amount |
+-----+------------+---------+--------+
| 100 | 2015-09-08 |       2 |   1500 |
| 101 | 2016-10-06 |       3 |   1000 |
| 102 | 2016-11-15 |       2 |   2000 |
| 103 | 2016-05-25 |       4 |   2500 |
+-----+------------+---------+--------+
```

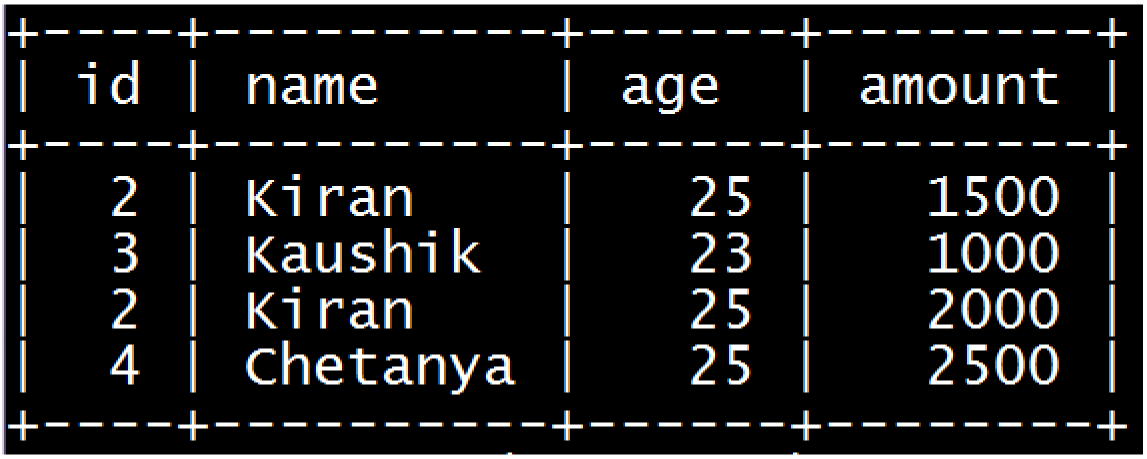


```
create table orders
(
oid int(3) primary key,
ord_date date,
cust_id int(2),
amount int(4),
foreign key (cust_id) references customer(id)
);
```

**Now, let us join these two tables in our SELECT statement as shown below.**

```
SQL> SELECT ID, NAME, AGE, AMOUNT
   FROM CUSTOMER, ORDERS
   WHERE  CUSTOMER.ID = ORDERS.CUST_ID;
```

**This would produce the following result.**



```
+----+----------+------+--------+
| id | name     | age  | amount |
+----+----------+------+--------+
|  2 | Kiran    |   25 |   1500 |
|  3 | Kaushik  |   23 |   1000 |
|  2 | Kiran    |   25 |   2000 |
|  4 | Chetanya |   25 |   2500 |
+----+----------+------+--------+
```

## Types of joins in SQL

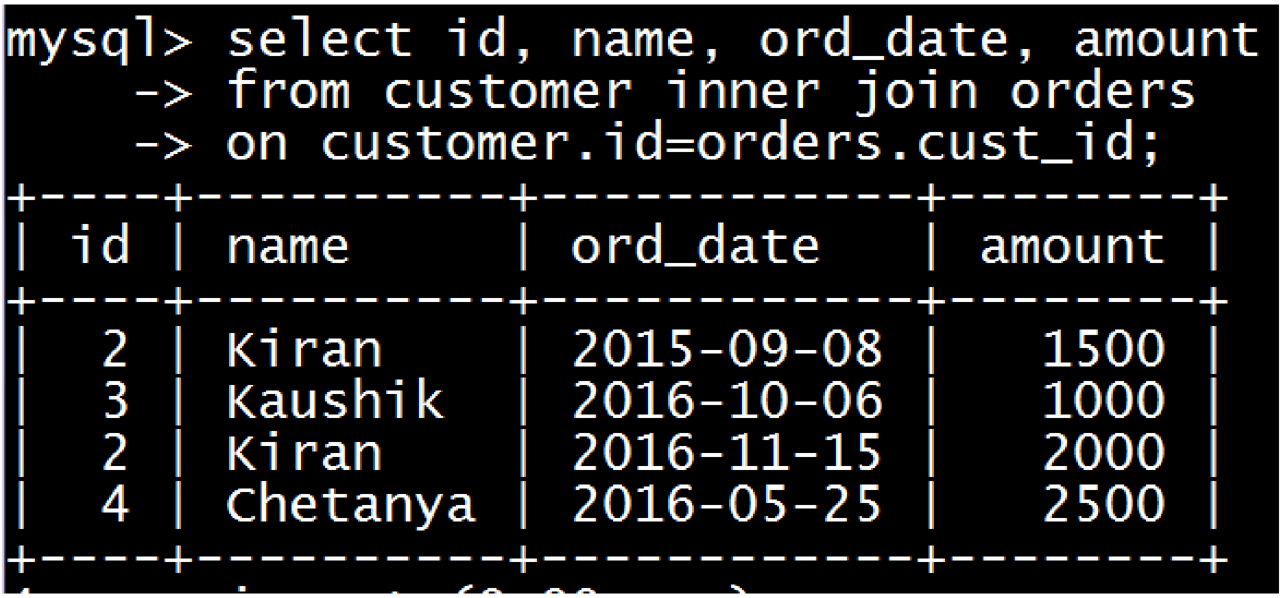
1. INNER JOIN (EQUI JOIN)
2. LEFT JOIN
3. RIGHT JOIN
4. NAURAL JOIN
5. SELF JOIN
6. CARTESIAN PRODUCT JOIN

## INNER JOIN (EQUI JOIN):

The INNER JOIN creates result table by combining column values of two tables (table1 and table2) based upon the join-condition. The query compares each row of table1 with each row of table2 to find all pairs of rows which satisfy the join-condition. When the join-condition is satisfied, column values for each matched pair of rows of table A and table B are combined into a result row.

Syntax

```
SELECT table1.column1, table2.column2...
FROM table1 INNER JOIN table2
ON table1.common_field = table2.common_field;
```



```
mysql> select id, name, ord_date, amount
    -> from customer inner join orders
    -> on customer.id=orders.cust_id;
+----+----------+------------+--------+
| id | name     | ord_date   | amount |
+----+----------+------------+--------+
|  2 | Kiran    | 2015-09-08 |   1500 |
|  3 | Kaushik  | 2016-10-06 |   1000 |
|  2 | Kiran    | 2016-11-15 |   2000 |
|  4 | Chetanya | 2016-05-25 |   2500 |
+----+----------+------------+--------+
```

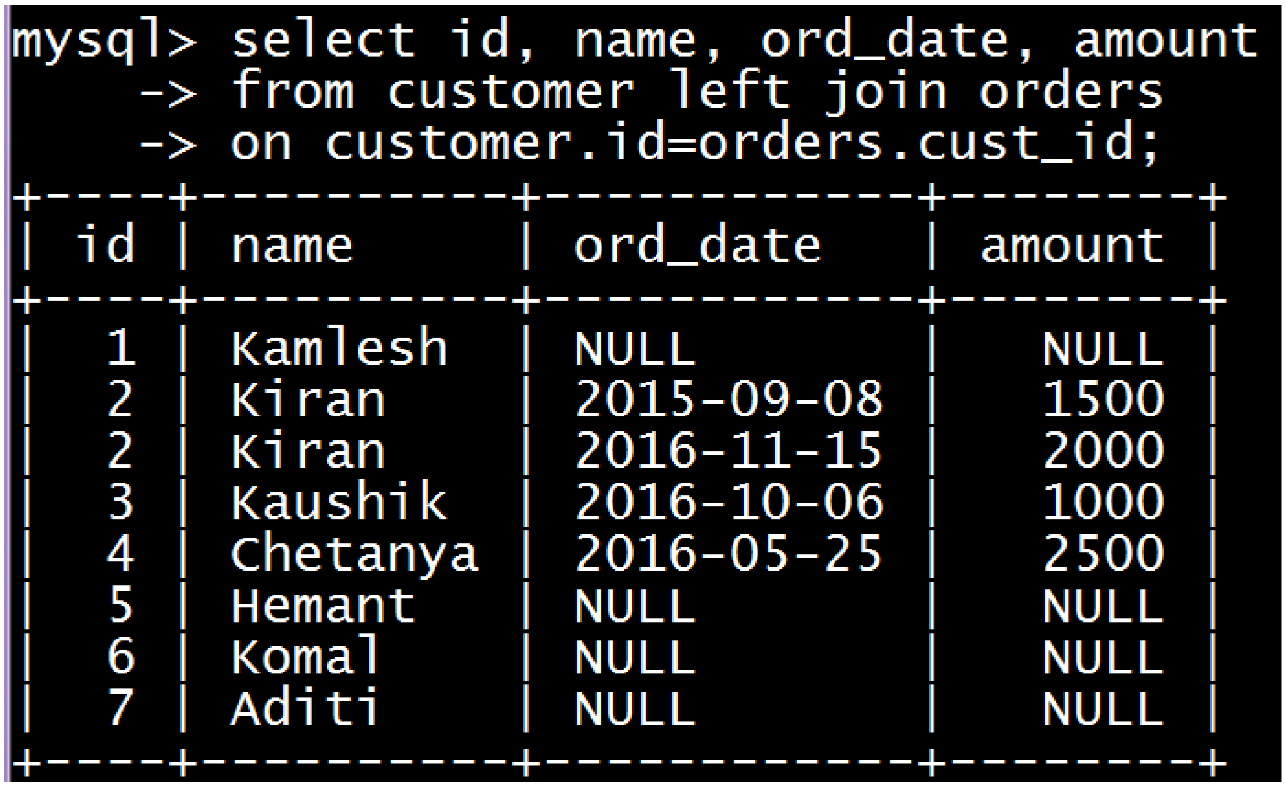
## LEFT JOIN

**The SQL LEFT JOIN returns all rows from the left table, even if there are no matches in the right table. This means that if the ON clause matches 0 (zero) records in the right table; the join will still return a row in the result, but with NULL in each column from the right table.**

**This means that a left join returns all the values from the left table, plus matched values from the right table or NULL in case of no matching join Condition.**

### Syntax

**SELECT table1.column1, table2.column2...**
**FROM table1**
**LEFT JOIN table2**
**ON table1.common_field = table2.common_field;**



```
mysql> select id, name, ord_date, amount
    -> from customer left join orders
    -> on customer.id=orders.cust_id;
+----+----------+------------+--------+
| id | name     | ord_date   | amount |
+----+----------+------------+--------+
|  1 | Kamlesh  | NULL       |   NULL |
|  2 | Kiran    | 2015-09-08 |   1500 |
|  2 | Kiran    | 2016-11-15 |   2000 |
|  3 | Kaushik  | 2016-10-06 |   1000 |
|  4 | Chetanya | 2016-05-25 |   2500 |
|  5 | Hemant   | NULL       |   NULL |
|  6 | Komal    | NULL       |   NULL |
|  7 | Aditi    | NULL       |   NULL |
+----+----------+------------+--------+
```

## RIGHT JOIN:

**The SQL RIGHT JOIN returns all rows from the right table, even if there are no matches in the left table. This means that if the ON clause matches 0 (zero) records in the left table; the join will still return a row in the result, but with NULL in each column from the left table.**

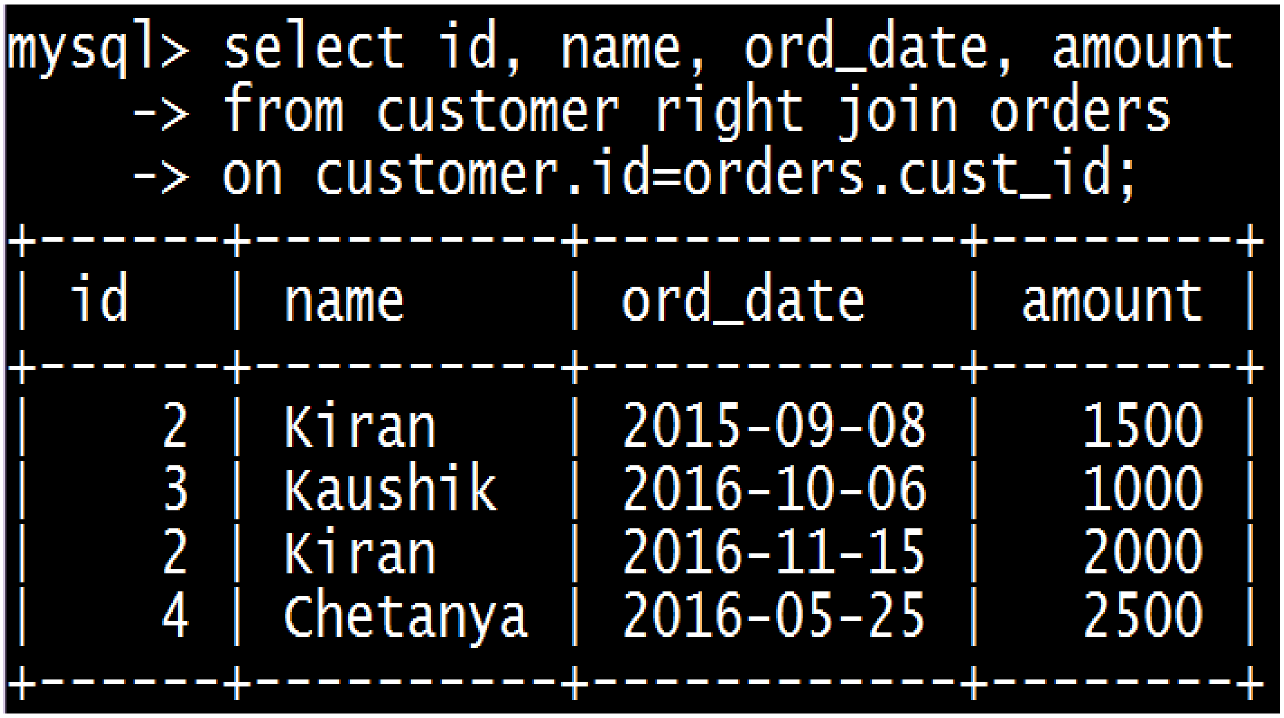
**This means that a right join returns all the values from the right table, plus matched values from the left table or NULL in case of no matching join Condition.**

## Syntax

**SELECT table1.column1, table2.column2...**

**FROM table1 RIGHT JOIN table2**

**ON table1.common_field = table2.common_field;**



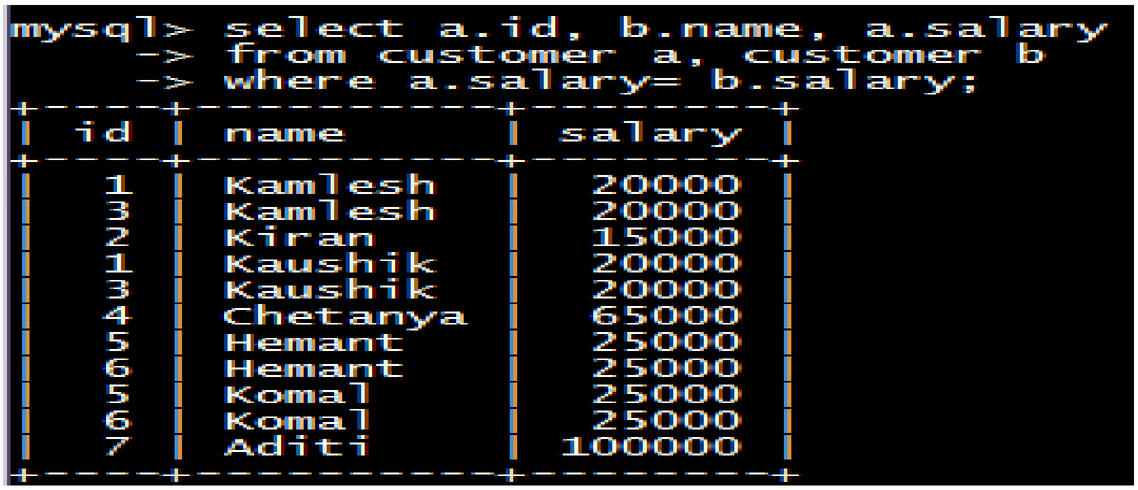
```
mysql> select id, name, ord_date, amount
    -> from customer right join orders
    -> on customer.id=orders.cust_id;
+------+----------+------------+--------+
| id   | name     | ord_date   | amount |
+------+----------+------------+--------+
|    2 | Kiran    | 2015-09-08 |   1500 |
|    3 | Kaushik  | 2016-10-06 |   1000 |
|    2 | Kiran    | 2016-11-15 |   2000 |
|    4 | Chetanya | 2016-05-25 |   2500 |
+------+----------+------------+--------+
```

## SELF JOIN:

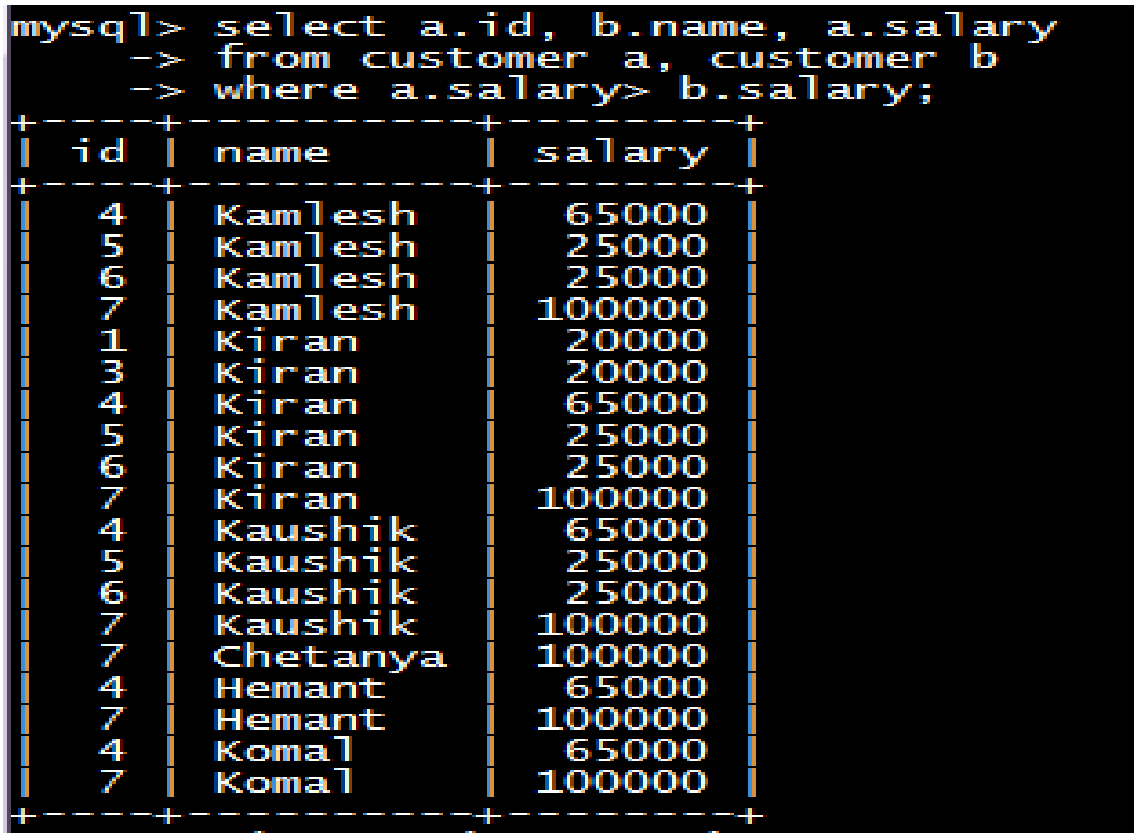
**The SQL SELF JOIN is used to join a table to itself as if the table were two tables; temporarily renaming at least one table in the SQL statement.**

**Syntax:**

```
SELECT a.column_name, b.column_name...
FROM table1 a, table1 b
WHERE a.common_field = b.common_field;
```



```
mysql> select a.id, b.name, a.salary
    -> from customer a, customer b
    -> where a.salary= b.salary;
+----+-----------+--------+
| id | name      | salary |
+----+-----------+--------+
|  1 | Kamlesh   |  20000 |
|  3 | Kamlesh   |  20000 |
|  2 | Kiran     |  15000 |
|  1 | Kaushik   |  20000 |
|  3 | Kaushik   |  20000 |
|  4 | Chetanya  |  65000 |
|  5 | Hemant    |  25000 |
|  6 | Hemant    |  25000 |
|  5 | Komal     |  25000 |
|  6 | Komal     |  25000 |
|  7 | Aditi     | 100000 |
+----+-----------+--------+
```



```
mysql> select a.id, b.name, a.salary
    -> from customer a, customer b
    -> where a.salary> b.salary;
+----+-----------+--------+
| id | name      | salary |
+----+-----------+--------+
|  4 | Kamlesh   |  65000 |
|  5 | Kamlesh   |  25000 |
|  6 | Kamlesh   |  25000 |
|  7 | Kamlesh   | 100000 |
|  1 | Kiran     |  20000 |
|  3 | Kiran     |  20000 |
|  4 | Kiran     |  65000 |
|  5 | Kiran     |  25000 |
|  6 | Kiran     |  25000 |
|  7 | Kiran     | 100000 |
|  4 | Kaushik   |  65000 |
|  5 | Kaushik   |  25000 |
|  6 | Kaushik   |  25000 |
|  7 | Kaushik   | 100000 |
|  7 | Chetanya  | 100000 |
|  4 | Hemant    |  65000 |
|  7 | Hemant    | 100000 |
|  4 | Komal     |  65000 |
|  7 | Komal     | 100000 |
+----+-----------+--------+
```

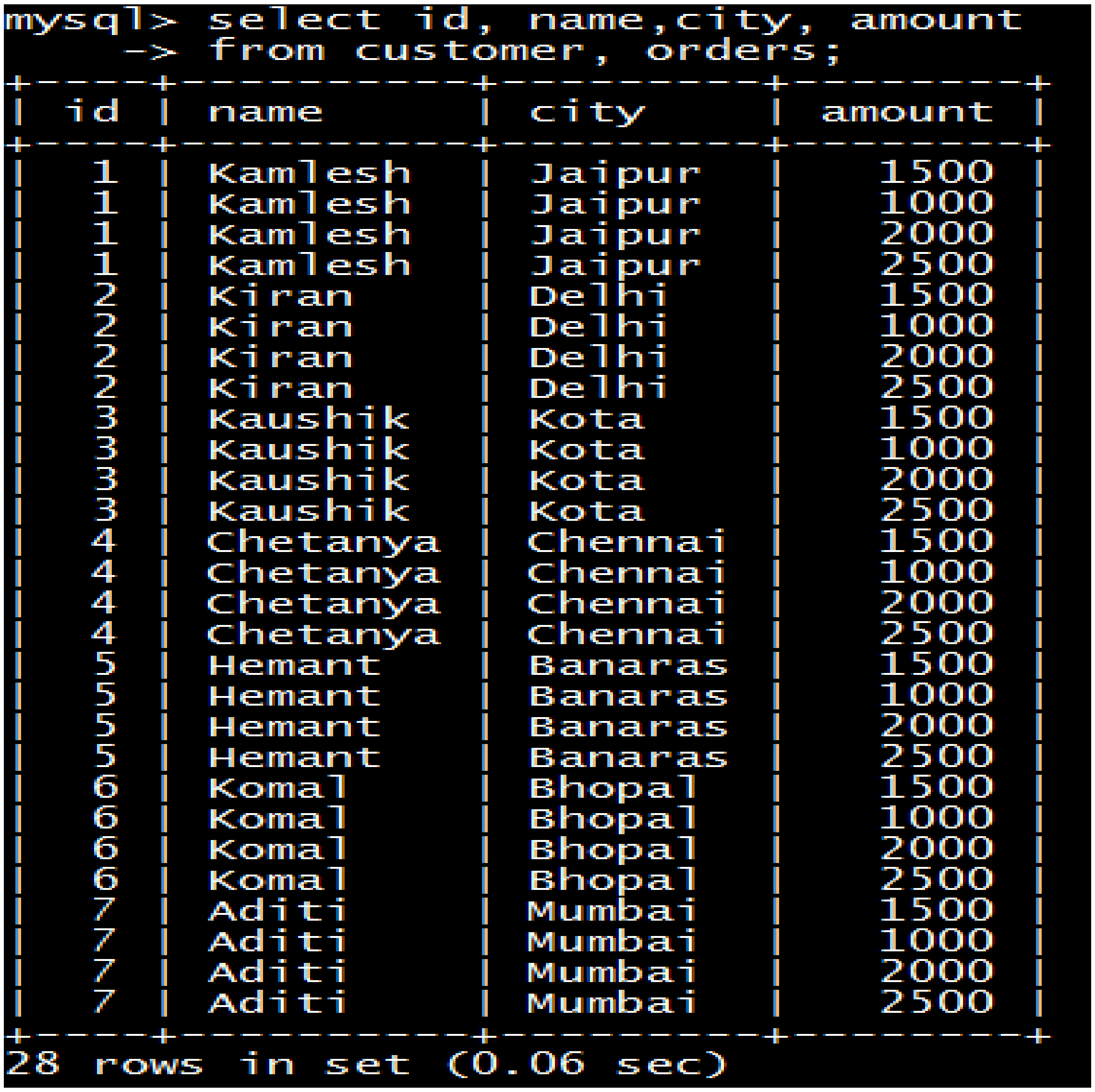
## CARTESIAN JOIN:

**The CARTESIAN JOIN or CROSS JOIN returns the Cartesian product of the sets of records from two or more joined tables. Thus, it equates to an inner join where the join-condition always evaluates to either True or where the join-condition is absent from the statement.**

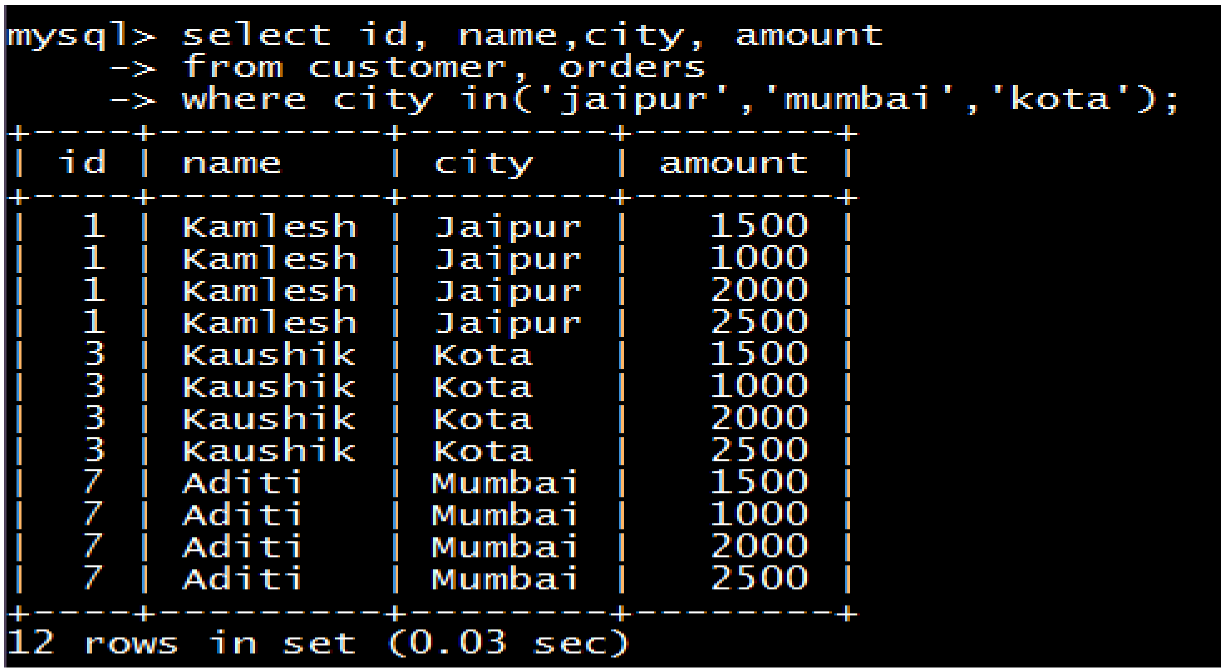
**Syntax:**

**SELECT table1.column1, table2.column2...**

**FROM table1, table2 [, table3 ]**



```
mysql> select id, name,city, amount
    -> from customer, orders;
+----+-----------+-----------+---------+
| id | name      | city      | amount  |
+----+-----------+-----------+---------+
|  1 | Kamlesh   | Jaipur    |    1500 |
|  1 | Kamlesh   | Jaipur    |    1000 |
|  1 | Kamlesh   | Jaipur    |    2000 |
|  1 | Kamlesh   | Jaipur    |    2500 |
|  2 | Kiran     | Delhi     |    1500 |
|  2 | Kiran     | Delhi     |    1000 |
|  2 | Kiran     | Delhi     |    2000 |
|  2 | Kiran     | Delhi     |    2500 |
|  3 | Kaushik   | Kota      |    1500 |
|  3 | Kaushik   | Kota      |    1000 |
|  3 | Kaushik   | Kota      |    2000 |
|  3 | Kaushik   | Kota      |    2500 |
|  4 | Chetanya  | Chennai   |    1500 |
|  4 | Chetanya  | Chennai   |    1000 |
|  4 | Chetanya  | Chennai   |    2000 |
|  4 | Chetanya  | Chennai   |    2500 |
|  5 | Hemant    | Banaras   |    1500 |
|  5 | Hemant    | Banaras   |    1000 |
|  5 | Hemant    | Banaras   |    2000 |
|  5 | Hemant    | Banaras   |    2500 |
|  6 | Komal     | Bhopal    |    1500 |
|  6 | Komal     | Bhopal    |    1000 |
|  6 | Komal     | Bhopal    |    2000 |
|  6 | Komal     | Bhopal    |    2500 |
|  7 | Aditi     | Mumbai    |    1500 |
|  7 | Aditi     | Mumbai    |    1000 |
|  7 | Aditi     | Mumbai    |    2000 |
|  7 | Aditi     | Mumbai    |    2500 |
+----+-----------+-----------+---------+
28 rows in set (0.06 sec)
```



```
mysql> select id, name,city, amount
    -> from customer, orders
    -> where city in('jaipur','mumbai','kota');
+----+---------+---------+--------+
| id | name    | city    | amount |
+----+---------+---------+--------+
|  1 | Kamlesh | Jaipur  |   1500 |
|  1 | Kamlesh | Jaipur  |   1000 |
|  1 | Kamlesh | Jaipur  |   2000 |
|  1 | Kamlesh | Jaipur  |   2500 |
|  3 | Kaushik | Kota    |   1500 |
|  3 | Kaushik | Kota    |   1000 |
|  3 | Kaushik | Kota    |   2000 |
|  3 | Kaushik | Kota    |   2500 |
|  7 | Aditi   | Mumbai  |   1500 |
|  7 | Aditi   | Mumbai  |   1000 |
|  7 | Aditi   | Mumbai  |   2000 |
|  7 | Aditi   | Mumbai  |   2500 |
+----+---------+---------+--------+
12 rows in set (0.03 sec)
```

## NATURAL JOIN:

**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.**

## Special conditions of Natural Join:

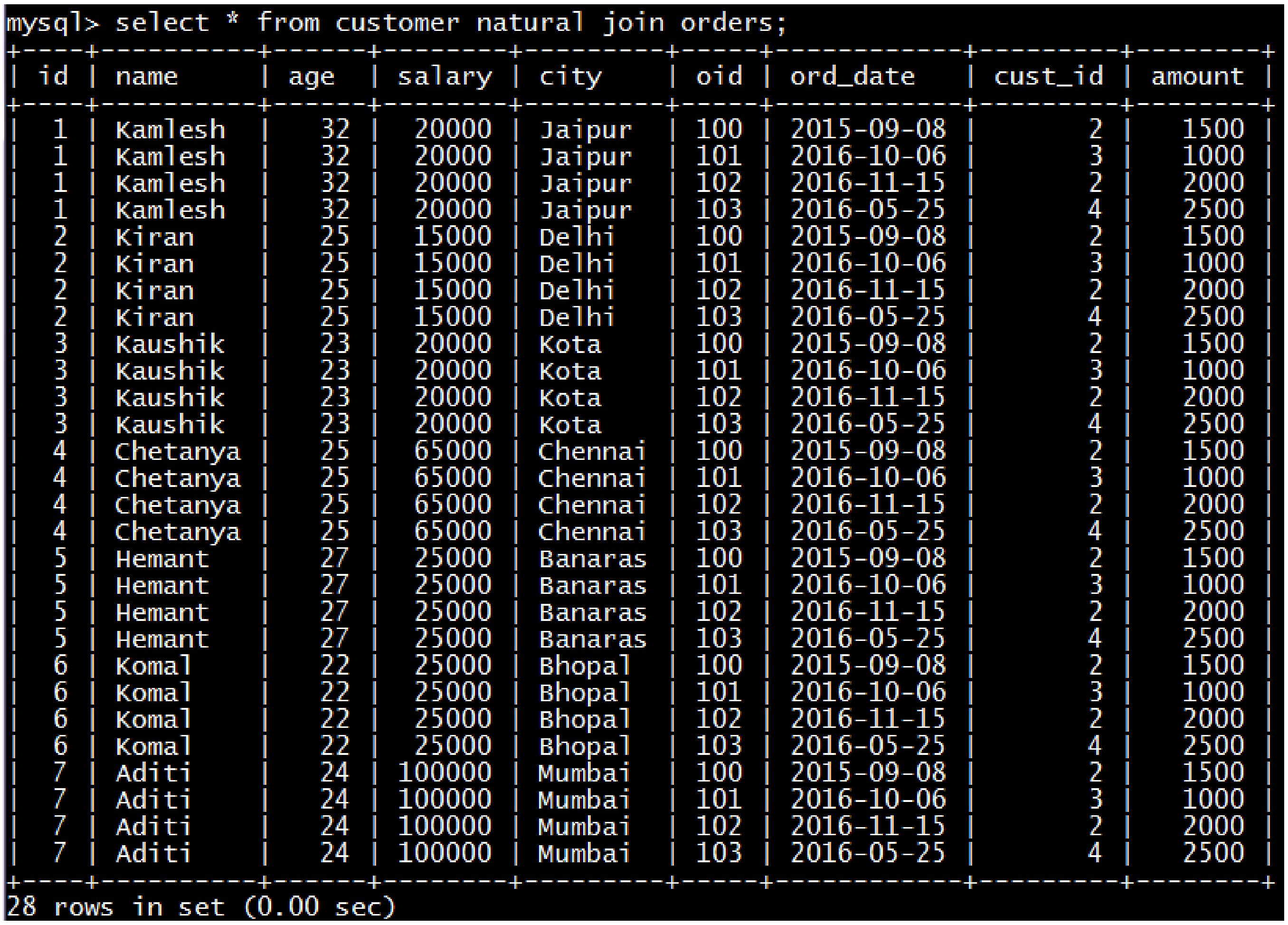
- **The associated tables have one or more pairs of identically named columns.**
- **The columns must be the same data type.**
- **Don’t use ON clause in a natural join.**

**Syntax:**

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

**Other Syntax:**

**SELECT * FROM table1, table2;**



```
mysql> select * from customer natural join orders;
+----+----------+------+--------+---------+-----+------------+---------+--------+
| id | name     | age  | salary | city    | oid | ord_date   | cust_id | amount |
+----+----------+------+--------+---------+-----+------------+---------+--------+
|  1 | Kamlesh  |   32 |  20000 | Jaipur  | 100 | 2015-09-08 |       2 |   1500 |
|  1 | Kamlesh  |   32 |  20000 | Jaipur  | 101 | 2016-10-06 |       3 |   1000 |
|  1 | Kamlesh  |   32 |  20000 | Jaipur  | 102 | 2016-11-15 |       2 |   2000 |
|  1 | Kamlesh  |   32 |  20000 | Jaipur  | 103 | 2016-05-25 |       4 |   2500 |
|  2 | Kiran    |   25 |  15000 | Delhi   | 100 | 2015-09-08 |       2 |   1500 |
|  2 | Kiran    |   25 |  15000 | Delhi   | 101 | 2016-10-06 |       3 |   1000 |
|  2 | Kiran    |   25 |  15000 | Delhi   | 102 | 2016-11-15 |       2 |   2000 |
|  2 | Kiran    |   25 |  15000 | Delhi   | 103 | 2016-05-25 |       4 |   2500 |
|  3 | Kaushik  |   23 |  20000 | Kota    | 100 | 2015-09-08 |       2 |   1500 |
|  3 | Kaushik  |   23 |  20000 | Kota    | 101 | 2016-10-06 |       3 |   1000 |
|  3 | Kaushik  |   23 |  20000 | Kota    | 102 | 2016-11-15 |       2 |   2000 |
|  3 | Kaushik  |   23 |  20000 | Kota    | 103 | 2016-05-25 |       4 |   2500 |
|  4 | Chetanya |   25 |  65000 | Chennai | 100 | 2015-09-08 |       2 |   1500 |
|  4 | Chetanya |   25 |  65000 | Chennai | 101 | 2016-10-06 |       3 |   1000 |
|  4 | Chetanya |   25 |  65000 | Chennai | 102 | 2016-11-15 |       2 |   2000 |
|  4 | Chetanya |   25 |  65000 | Chennai | 103 | 2016-05-25 |       4 |   2500 |
|  5 | Hemant   |   27 |  25000 | Banaras | 100 | 2015-09-08 |       2 |   1500 |
|  5 | Hemant   |   27 |  25000 | Banaras | 101 | 2016-10-06 |       3 |   1000 |
|  5 | Hemant   |   27 |  25000 | Banaras | 102 | 2016-11-15 |       2 |   2000 |
|  5 | Hemant   |   27 |  25000 | Banaras | 103 | 2016-05-25 |       4 |   2500 |
|  6 | Komal    |   22 |  25000 | Bhopal  | 100 | 2015-09-08 |       2 |   1500 |
|  6 | Komal    |   22 |  25000 | Bhopal  | 101 | 2016-10-06 |       3 |   1000 |
|  6 | Komal    |   22 |  25000 | Bhopal  | 102 | 2016-11-15 |       2 |   2000 |
|  6 | Komal    |   22 |  25000 | Bhopal  | 103 | 2016-05-25 |       4 |   2500 |
|  7 | Aditi    |   24 | 100000 | Mumbai  | 100 | 2015-09-08 |       2 |   1500 |
|  7 | Aditi    |   24 | 100000 | Mumbai  | 101 | 2016-10-06 |       3 |   1000 |
|  7 | Aditi    |   24 | 100000 | Mumbai  | 102 | 2016-11-15 |       2 |   2000 |
|  7 | Aditi    |   24 | 100000 | Mumbai  | 103 | 2016-05-25 |       4 |   2500 |
+----+----------+------+--------+---------+-----+------------+---------+--------+
28 rows in set (0.00 sec)
```

**The Above same result can also be produce by using the given below command.**

**SELECT * FROM CUSTOMER, ORDERS;**

**: : Finished : :**