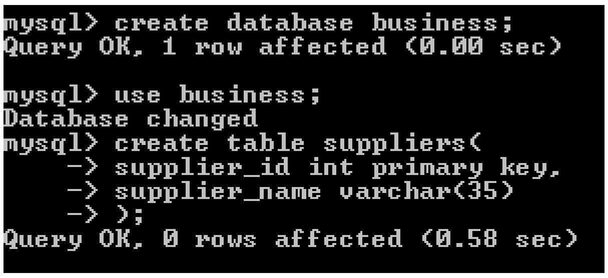
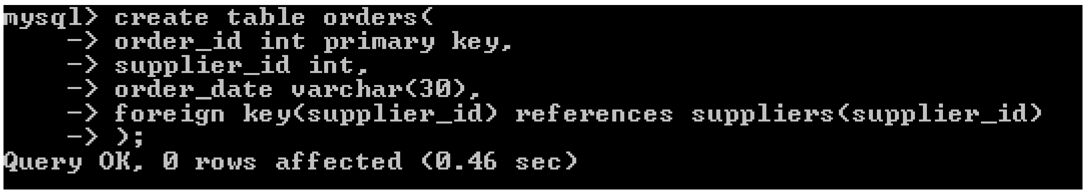
Somnath PaulChoudhury notes of Computer Science SQL Joins Part 1 for CBSE India

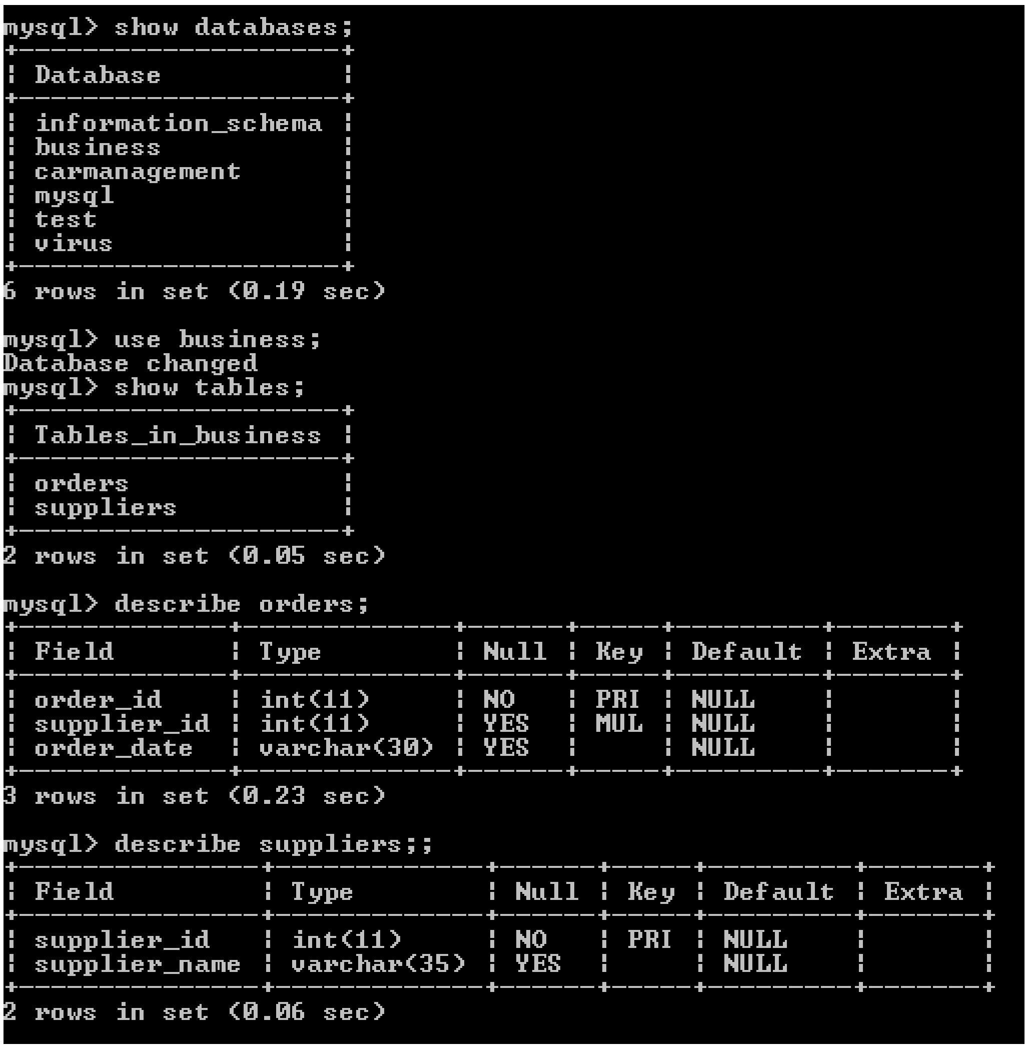
Lets create a database and two tables and set foreign key relationship. Study the details below.



```
mysql> create database business;
Query OK, 1 row affected (0.00 sec)

mysql> use business;
Database changed
mysql> create table suppliers(
    -> supplier_id int primary key,
    -> supplier_name varchar(35)
    -> );
Query OK, 0 rows affected (0.58 sec)
```



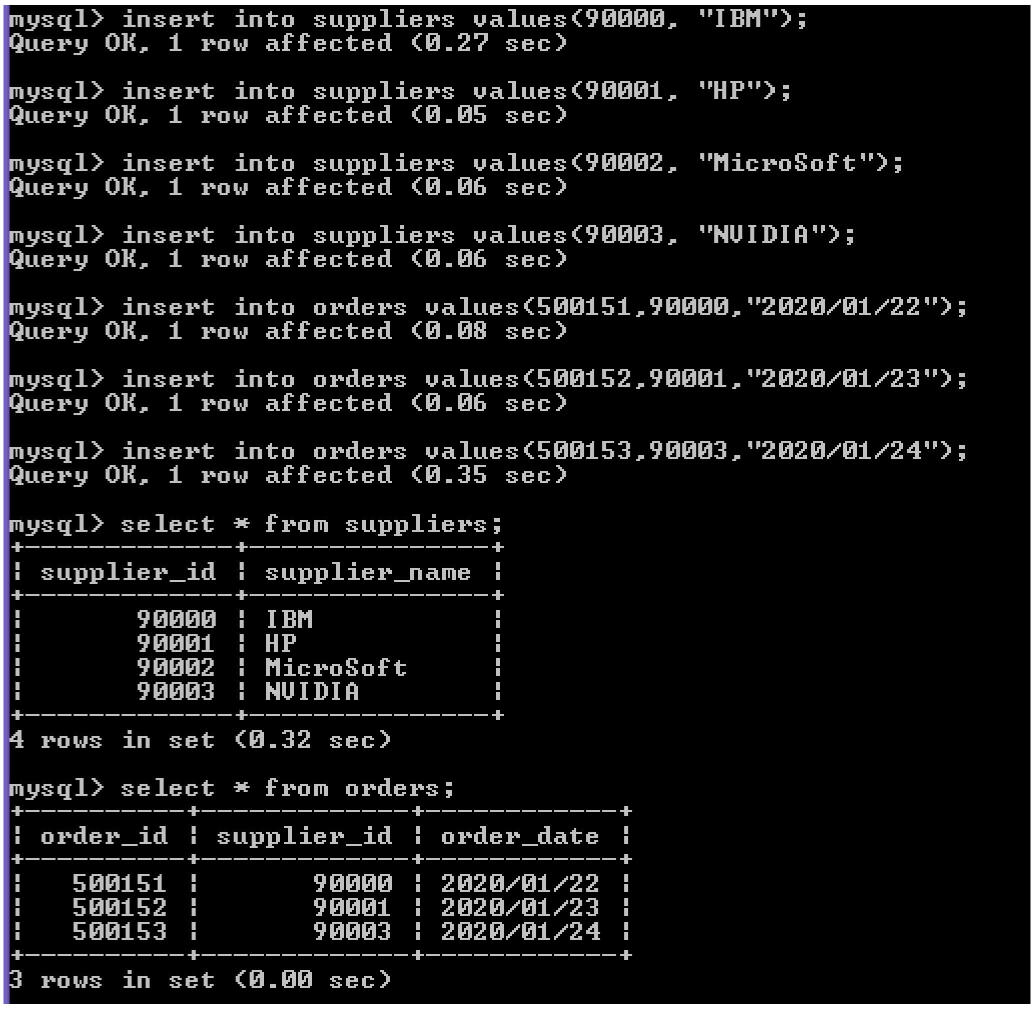
```
mysql> create table orders(
    -> order_id int primary key,
    -> supplier_id int,
    -> order_date varchar(30),
    -> foreign key(supplier_id) references suppliers(supplier_id)
    -> );
Query OK, 0 rows affected (0.46 sec)
```



```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| business           |
| carmanagement      |
| mysql              |
| test               |
| virus              |
+--------------------+
6 rows in set (0.19 sec)

mysql> use business;
Database changed
mysql> show tables;
+--------------------+
| Tables_in_business |
+--------------------+
| orders             |
| suppliers          |
+--------------------+
2 rows in set (0.05 sec)

mysql> describe orders;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| order_id    | int(11)     | NO   | PRI | NULL    |       |
| supplier_id | int(11)     | YES  | MUL | NULL    |       |
| order_date  | varchar(30) | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
3 rows in set (0.23 sec)

mysql> describe suppliers;;
+---------------+-------------+------+-----+---------+-------+
| Field         | Type        | Null | Key | Default | Extra |
+---------------+-------------+------+-----+---------+-------+
| supplier_id   | int(11)     | NO   | PRI | NULL    |       |
| supplier_name | varchar(35) | YES  |     | NULL    |       |
+---------------+-------------+------+-----+---------+-------+
2 rows in set (0.06 sec)
```

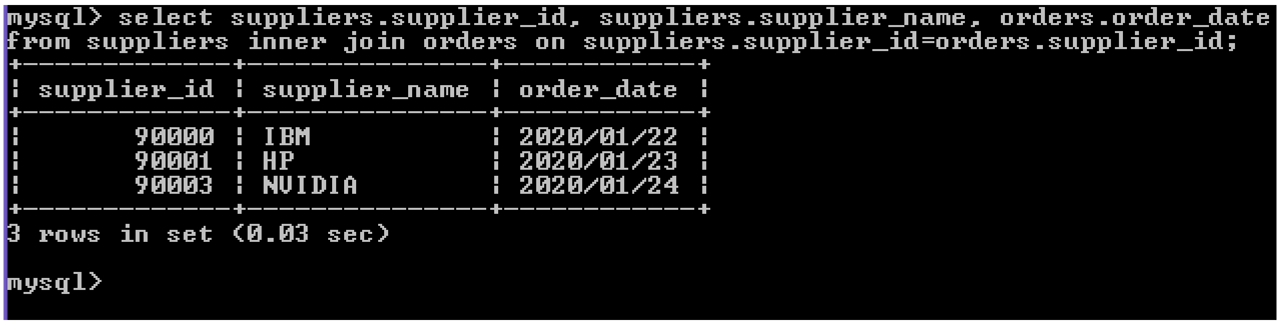
We will now upload some records in both the tables

Here are the records



```
mysql> insert into suppliers values(90000, "IBM");
Query OK, 1 row affected (0.27 sec)

mysql> insert into suppliers values(90001, "HP");
Query OK, 1 row affected (0.05 sec)

mysql> insert into suppliers values(90002, "MicroSoft");
Query OK, 1 row affected (0.06 sec)

mysql> insert into suppliers values(90003, "NVIDIA");
Query OK, 1 row affected (0.06 sec)

mysql> insert into orders values(500151,90000,"2020/01/22");
Query OK, 1 row affected (0.08 sec)

mysql> insert into orders values(500152,90001,"2020/01/23");
Query OK, 1 row affected (0.06 sec)

mysql> insert into orders values(500153,90003,"2020/01/24");
Query OK, 1 row affected (0.35 sec)

mysql> select * from suppliers;
+-------------+---------------+
| supplier_id | supplier_name |
+-------------+---------------+
|       90000 | IBM           |
|       90001 | HP            |
|       90002 | MicroSoft     |
|       90003 | NVIDIA        |
+-------------+---------------+
4 rows in set (0.32 sec)

mysql> select * from orders;
+----------+-------------+------------+
| order_id | supplier_id | order_date |
+----------+-------------+------------+
|   500151 |       90000 | 2020/01/22 |
|   500152 |       90001 | 2020/01/23 |
|   500153 |       90003 | 2020/01/24 |
+----------+-------------+------------+
3 rows in set (0.00 sec)
```

Now lets try the SQL inner join

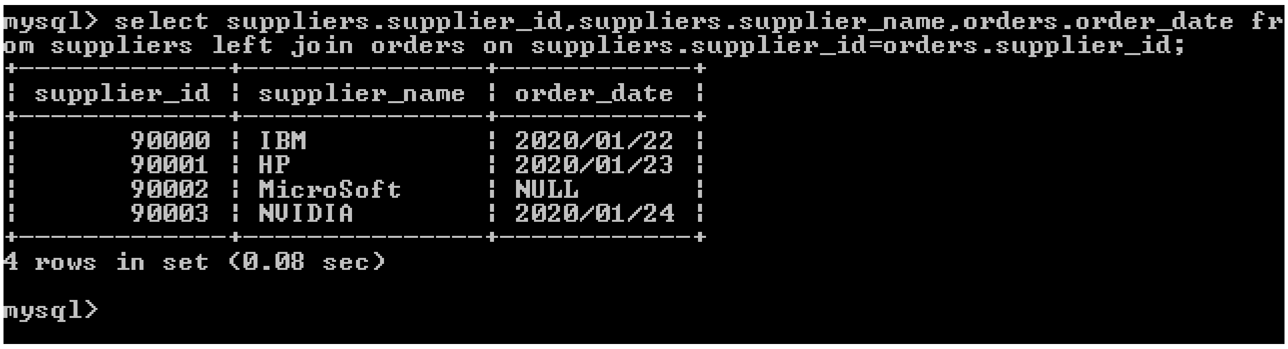


```
mysql> select suppliers.supplier_id, suppliers.supplier_name, orders.order_date
from suppliers inner join orders on suppliers.supplier_id=orders.supplier_id;
+-------------+---------------+------------+
| supplier_id | supplier_name | order_date |
+-------------+---------------+------------+
|       90000 | IBM           | 2020/01/22 |
|       90001 | HP            | 2020/01/23 |
|       90003 | NVIDIA        | 2020/01/24 |
+-------------+---------------+------------+
3 rows in set (0.03 sec)

mysql>
```

SQL inner join output explanation

If we consider the tables as over overlapping circles the area common to both of them is inner join output

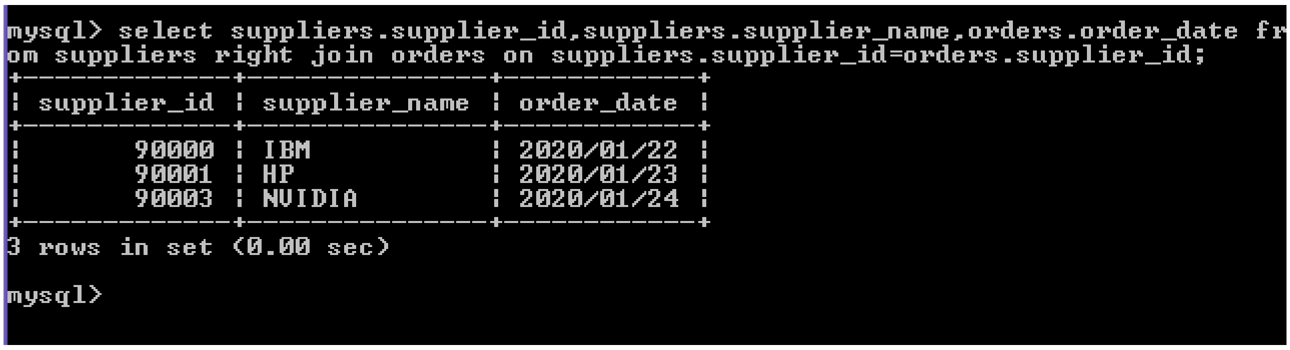
Now lets try the SQL left join



```
mysql> select suppliers.supplier_id,suppliers.supplier_name,orders.order_date fr
om suppliers left join orders on suppliers.supplier_id=orders.supplier_id;
+-------------+---------------+------------+
| supplier_id | supplier_name | order_date |
+-------------+---------------+------------+
|       90000 | IBM           | 2020/01/22 |
|       90001 | HP            | 2020/01/23 |
|       90002 | MicroSoft     | NULL       |
|       90003 | NVIDIA        | 2020/01/24 |
+-------------+---------------+------------+
4 rows in set (0.08 sec)

mysql>
```

If we consider the tables as over overlapping circles then left join is the entire left table and the corresponding common area of the right is included. If there are some missing values in table 2 we get NULL in that row as shown.

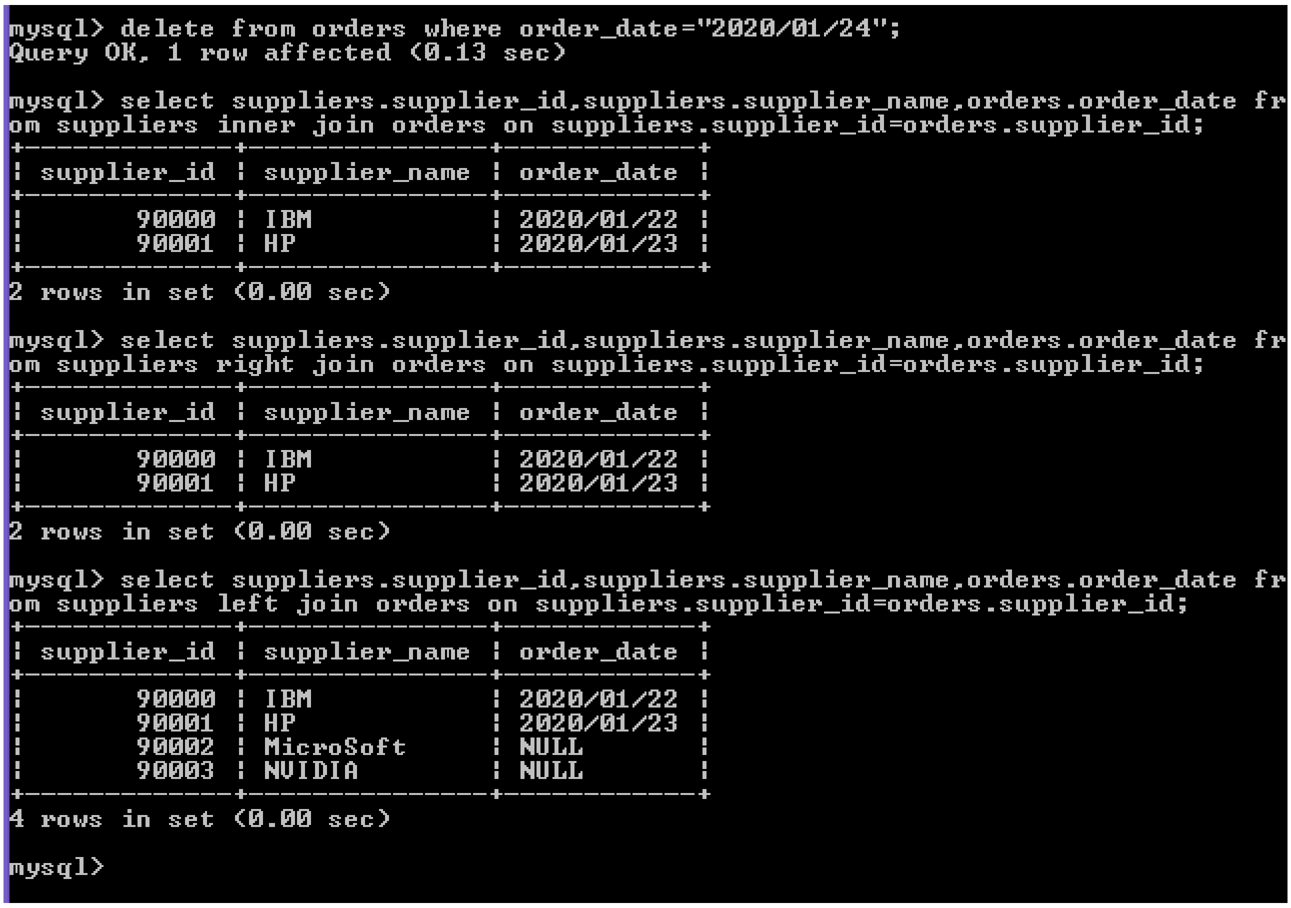
This is the SQL right join



```
mysql> select suppliers.supplier_id,suppliers.supplier_name,orders.order_date fr
om suppliers right join orders on suppliers.supplier_id=orders.supplier_id;
+-------------+---------------+------------+
| supplier_id | supplier_name | order_date |
+-------------+---------------+------------+
|       90000 | IBM           | 2020/01/22 |
|       90001 | HP            | 2020/01/23 |
|       90003 | NVIDIA        | 2020/01/24 |
+-------------+---------------+------------+
3 rows in set (0.00 sec)

mysql>
```

Now lets delete one record from orders and try all the joins above once more and see the output

Lets delete the last record from orders and try the commands again.



```
mysql> delete from orders where order_date="2020/01/24";
Query OK, 1 row affected (0.13 sec)

mysql> select suppliers.supplier_id,suppliers.supplier_name,orders.order_date fr
om suppliers inner join orders on suppliers.supplier_id=orders.supplier_id;
+-------------+---------------+------------+
| supplier_id | supplier_name | order_date |
+-------------+---------------+------------+
|       90000 | IBM           | 2020/01/22 |
|       90001 | HP            | 2020/01/23 |
+-------------+---------------+------------+
2 rows in set (0.00 sec)

mysql> select suppliers.supplier_id,suppliers.supplier_name,orders.order_date fr
om suppliers right join orders on suppliers.supplier_id=orders.supplier_id;
+-------------+---------------+------------+
| supplier_id | supplier_name | order_date |
+-------------+---------------+------------+
|       90000 | IBM           | 2020/01/22 |
|       90001 | HP            | 2020/01/23 |
+-------------+---------------+------------+
2 rows in set (0.00 sec)

mysql> select suppliers.supplier_id,suppliers.supplier_name,orders.order_date fr
om suppliers left join orders on suppliers.supplier_id=orders.supplier_id;
+-------------+---------------+------------+
| supplier_id | supplier_name | order_date |
+-------------+---------------+------------+
|       90000 | IBM           | 2020/01/22 |
|       90001 | HP            | 2020/01/23 |
|       90002 | MicroSoft     | NULL       |
|       90003 | NVIDIA        | NULL       |
+-------------+---------------+------------+
4 rows in set (0.00 sec)

mysql>
```

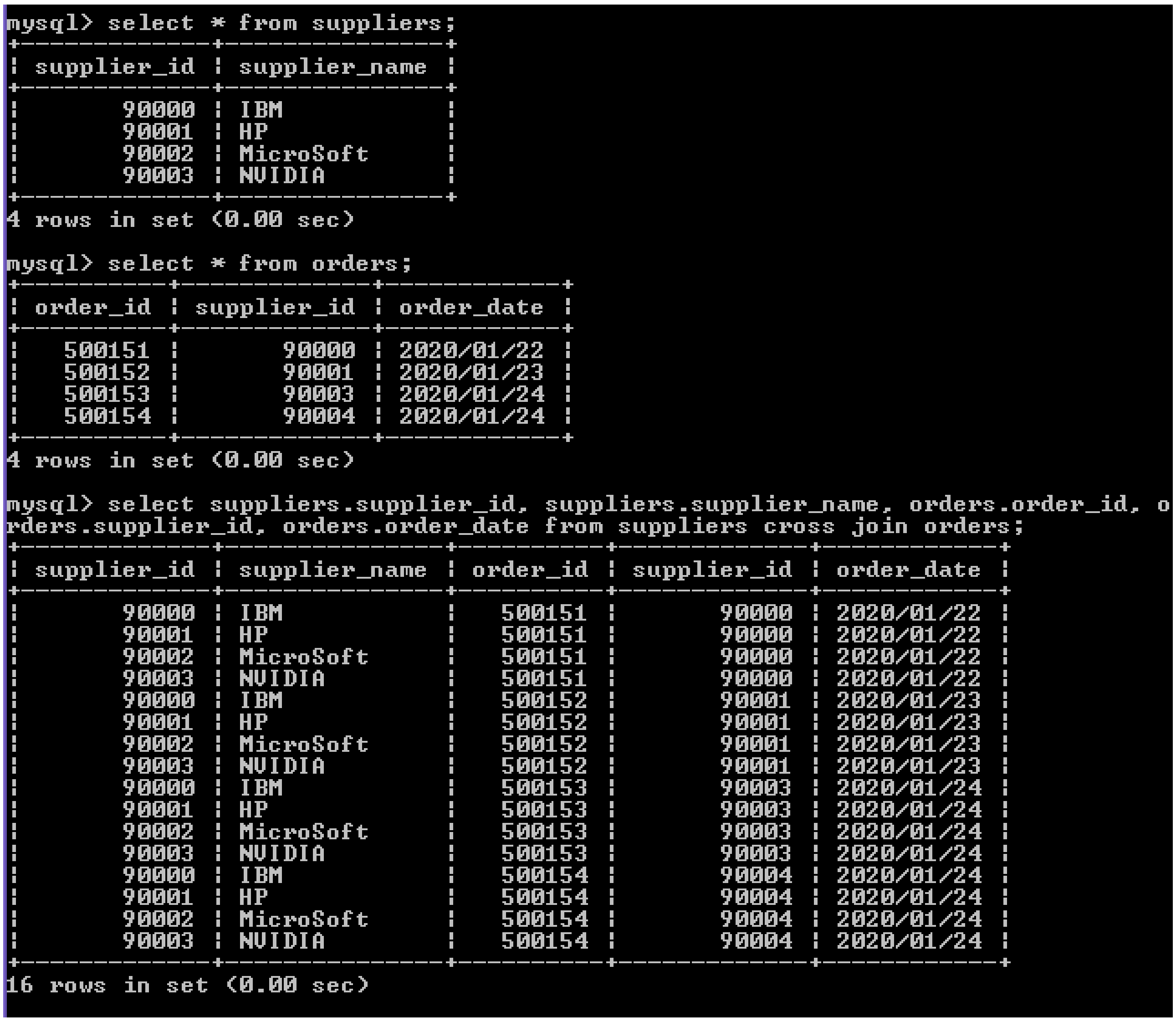
Now we are just trying to add some records in orders with a supplier_id that is not present in the suppliers so it will give errors for the constraint.

```
mysql> insert into orders values(500154, 90004, "2020/01/24");

ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint f

ails (`business`.`orders`, CONSTRAINT `orders_ibfk_1` FOREIGN KEY (`supplier_id`

) REFERENCES `suppliers` (`supplier_id`))

mysql> alter table orders drop foreign key;

ERROR 1005 (HY000): Can't create table 'business.#sql-7cc_1' (errno: 150)

mysql> alter table orders drop foreign key orders_ibfk_1;

Query OK, 3 rows affected (0.33 sec)

Records: 3 Duplicates: 0 Warnings: 0


mysql> describe orders;
```

```
+-------------+-------------+------+-----+---------+-------+

| Field | Type | Null | Key | Default | Extra |

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

| order_id | int(11) | NO | PRI | NULL | |

| supplier_id | int(11) | YES | MUL | NULL | |

| order_date | varchar(30) | YES | | NULL | |

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

3 rows in set (0.22 sec)


mysql> insert into orders values(500154, 90004, "2020/01/24");

Query OK, 1 row affected (0.03 sec)
```

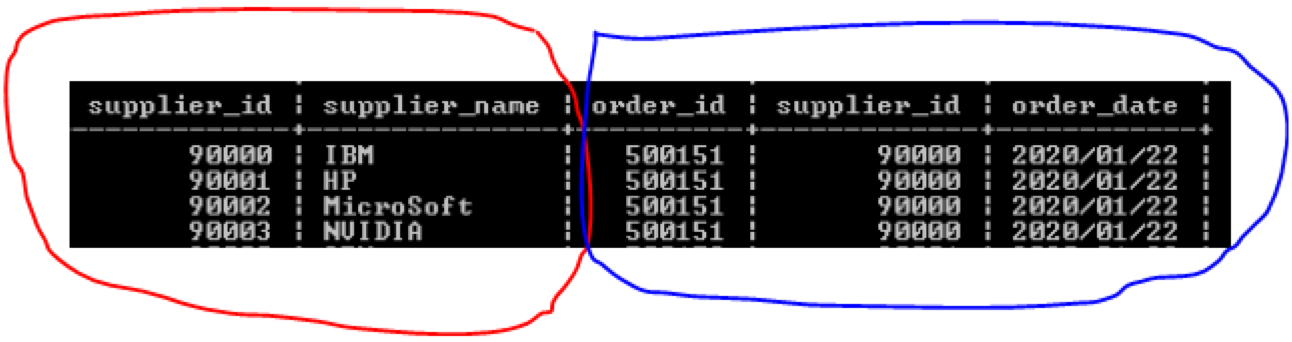
Now we can have it

Let us try the Cross join which is nothing but the cartesian product



```
mysql> select * from suppliers;
+-------------+---------------+
| supplier_id | supplier_name |
+-------------+---------------+
|       90000 | IBM           |
|       90001 | HP            |
|       90002 | MicroSoft     |
|       90003 | NVIDIA        |
+-------------+---------------+
4 rows in set (0.00 sec)

mysql> select * from orders;
+----------+-------------+------------+
| order_id | supplier_id | order_date |
+----------+-------------+------------+
|   500151 |       90000 | 2020/01/22 |
|   500152 |       90001 | 2020/01/23 |
|   500153 |       90003 | 2020/01/24 |
|   500154 |       90004 | 2020/01/24 |
+----------+-------------+------------+
4 rows in set (0.00 sec)

mysql> select suppliers.supplier_id, suppliers.supplier_name, orders.order_id, o
rders.supplier_id, orders.order_date from suppliers cross join orders;
+-------------+---------------+----------+-------------+------------+
| supplier_id | supplier_name | order_id | supplier_id | order_date |
+-------------+---------------+----------+-------------+------------+
|       90000 | IBM           |   500151 |       90000 | 2020/01/22 |
|       90001 | HP            |   500151 |       90000 | 2020/01/22 |
|       90002 | MicroSoft     |   500151 |       90000 | 2020/01/22 |
|       90003 | NVIDIA        |   500151 |       90000 | 2020/01/22 |
|       90000 | IBM           |   500152 |       90001 | 2020/01/23 |
|       90001 | HP            |   500152 |       90001 | 2020/01/23 |
|       90002 | MicroSoft     |   500152 |       90001 | 2020/01/23 |
|       90003 | NVIDIA        |   500152 |       90001 | 2020/01/23 |
|       90000 | IBM           |   500153 |       90003 | 2020/01/24 |
|       90001 | HP            |   500153 |       90003 | 2020/01/24 |
|       90002 | MicroSoft     |   500153 |       90003 | 2020/01/24 |
|       90003 | NVIDIA        |   500153 |       90003 | 2020/01/24 |
|       90000 | IBM           |   500154 |       90004 | 2020/01/24 |
|       90001 | HP            |   500154 |       90004 | 2020/01/24 |
|       90002 | MicroSoft     |   500154 |       90004 | 2020/01/24 |
|       90003 | NVIDIA        |   500154 |       90004 | 2020/01/24 |
+-------------+---------------+----------+-------------+------------+
16 rows in set (0.00 sec)
```

If we observe the output carefully we find a pattern, the entire supplier table is written four times and every time one row of orders is written



```
 supplier_id | supplier_name | order_id | supplier_id | order_date |
-------------+---------------+----------+-------------+------------+
       90000 | IBM           |   500151 |       90000 | 2020/01/22 |
       90001 | HP            |   500151 |       90000 | 2020/01/22 |
       90002 | MicroSoft     |   500151 |       90000 | 2020/01/22 |
       90003 | NVIDIA        |   500151 |       90000 | 2020/01/22 |
```

Inner Join:- This join returns all the rows from both tables where there is a match. In other words you can say that it gives all the records of the left table which have the matching records from the right table

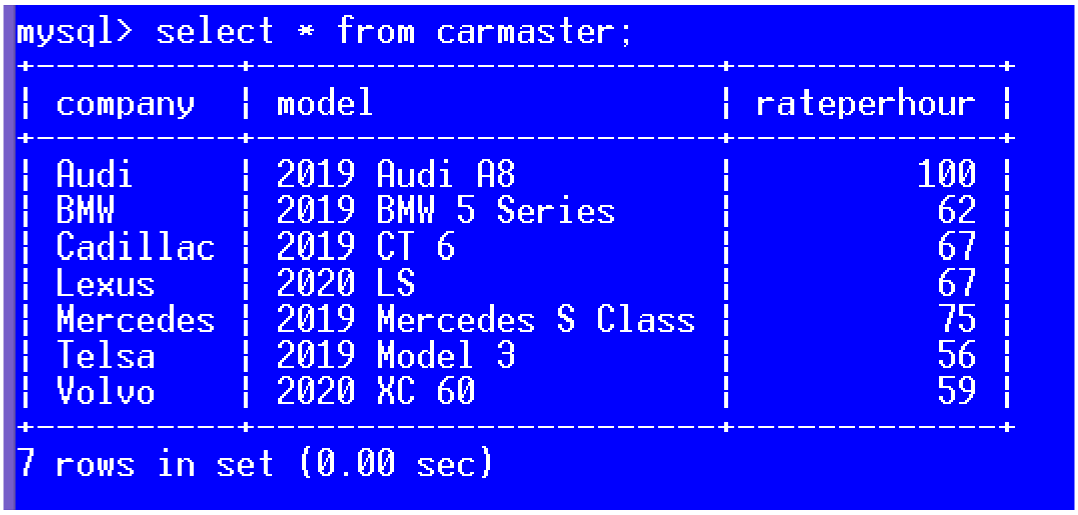
Left Outer Join:- This join returns all records from the left table irrespective of whether right table contains the matching records or not. For all matching records, it returns the matched records from the right table and for not matching records, it return with NULL value.

Right Outer Join:- This join returns all records from the Right table irrespective of whether Left table contains the matching records or not.

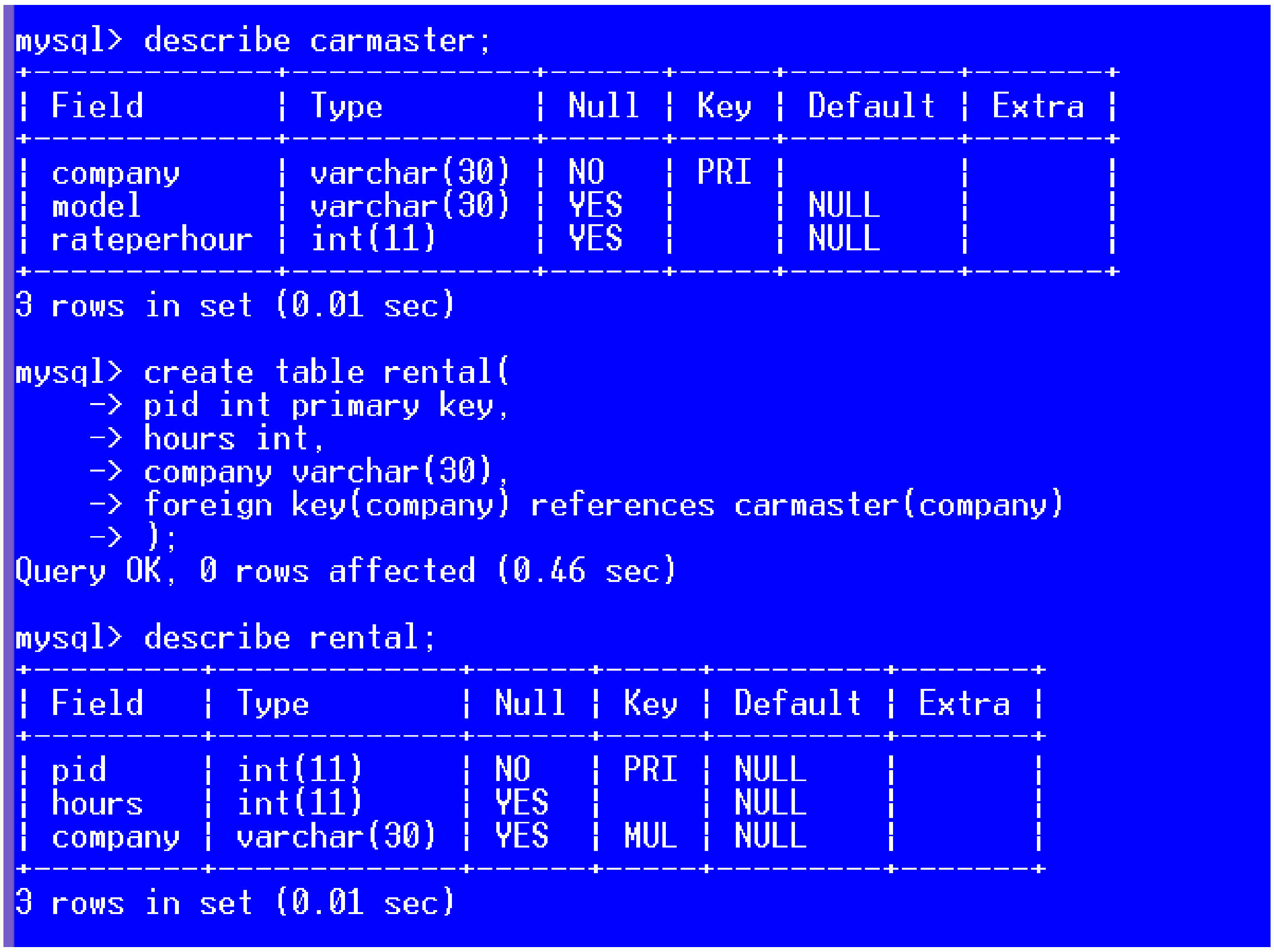
For all matching records from the right table, it returns the matched records from the Left table and for not matching records, it return with NULL value.

MySQL Equi Join

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.

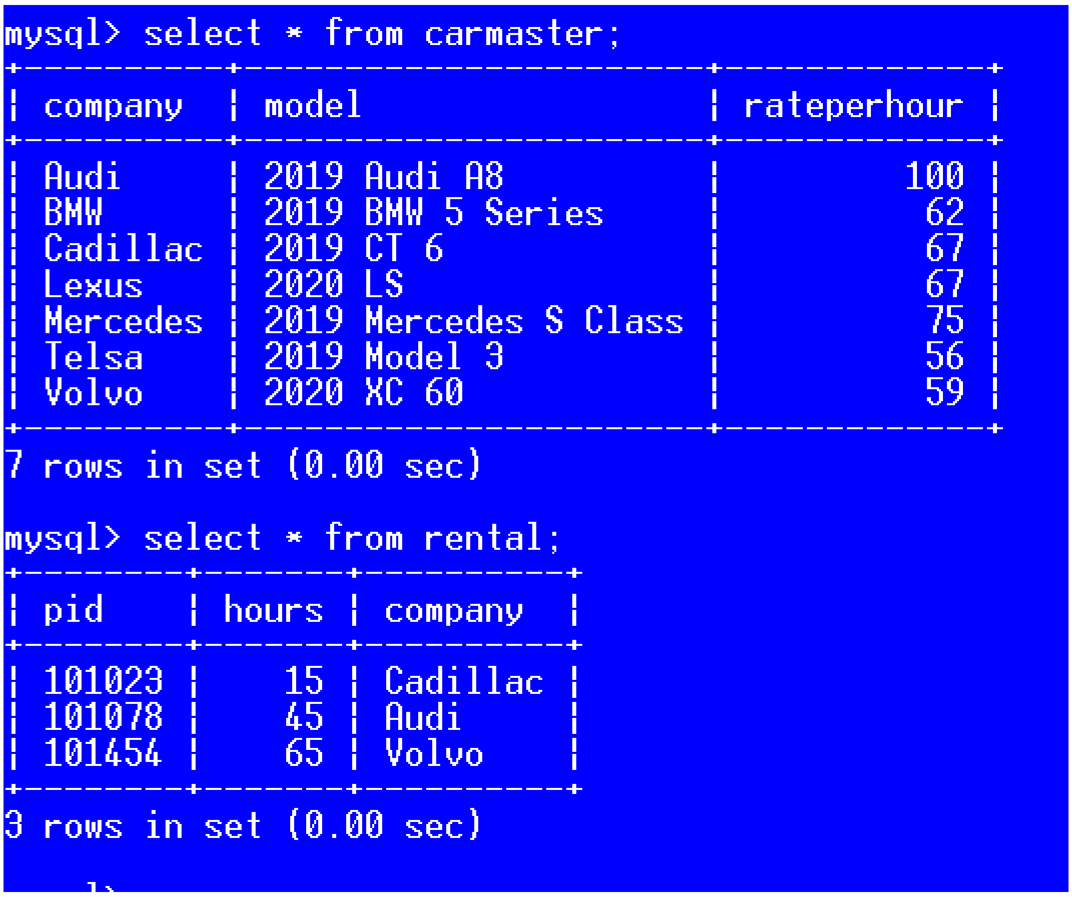


```
mysql> select * from carmaster;
+----------+------------------------------+--------------+
| company  | model                        | rateperhour |
+----------+------------------------------+--------------+
| Audi     | 2019 Audi A8                 |          100 |
| BMW      | 2019 BMW 5 Series            |           62 |
| Cadillac | 2019 CT 6                    |           67 |
| Lexus    | 2020 LS                      |           67 |
| Mercedes | 2019 Mercedes S Class        |           75 |
| Telsa    | 2019 Model 3                 |           56 |
| Volvo    | 2020 XC 60                   |           59 |
+----------+------------------------------+--------------+
7 rows in set (0.00 sec)
```



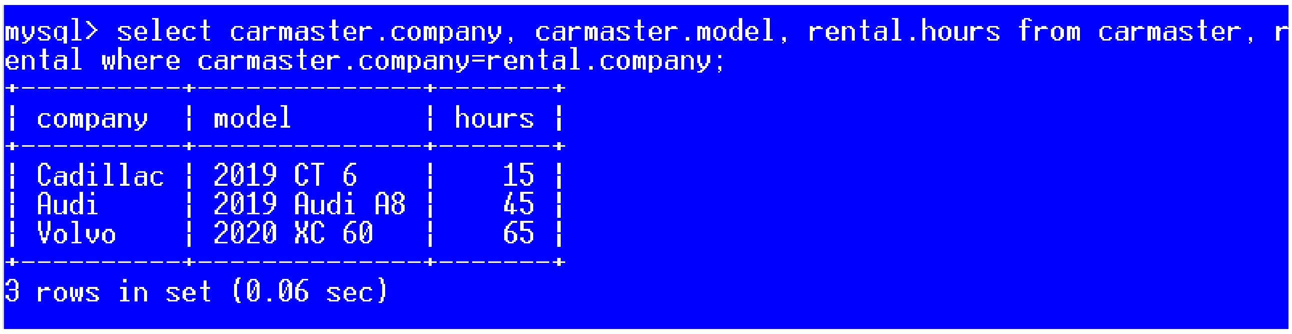
```
mysql> describe carmaster;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| company     | varchar(30) | NO   | PRI |         |       |
| model       | varchar(30) | YES  |     | NULL    |       |
| rateperhour | int(11)     | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
3 rows in set (0.01 sec)

mysql> create table rental(
    -> pid int primary key,
    -> hours int,
    -> company varchar(30),
    -> foreign key(company) references carmaster(company)
    -> );
Query OK, 0 rows affected (0.46 sec)

mysql> describe rental;
+---------+-------------+------+-----+---------+-------+
| Field   | Type        | Null | Key | Default | Extra |
+---------+-------------+------+-----+---------+-------+
| pid     | int(11)     | NO   | PRI | NULL    |       |
| hours   | int(11)     | YES  |     | NULL    |       |
| company | varchar(30) | YES  | MUL | NULL    |       |
+---------+-------------+------+-----+---------+-------+
3 rows in set (0.01 sec)
```

```
mysql> select * from carmaster;
+----------+----------------------+-------------+
| company  | model                | rateperhour |
+----------+----------------------+-------------+
| Audi     | 2019 Audi A8         |         100 |
| BMW      | 2019 BMW 5 Series    |          62 |
| Cadillac | 2019 CT 6            |          67 |
| Lexus    | 2020 LS              |          67 |
| Mercedes | 2019 Mercedes S Class |         75 |
| Telsa    | 2019 Model 3         |          56 |
| Volvo    | 2020 XC 60           |          59 |
+----------+----------------------+-------------+
7 rows in set (0.00 sec)

mysql> describe carmaster;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| company     | varchar(30) | NO   | PRI |         |       |
| model       | varchar(30) | YES  |     | NULL    |       |
| rateperhour | int(11)     | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
3 rows in set (0.01 sec)

mysql> create table rental(
    -> pid int primary key,
    -> hours int,
    -> company varchar(30),
    -> foreign key(company) references carmaster(company)
    -> );
Query OK, 0 rows affected (0.46 sec)



mysql> describe rental;
+---------+-------------+------+-----+---------+-------+
| Field   | Type        | Null | Key | Default | Extra |
+---------+-------------+------+-----+---------+-------+
| pid     | int(11)     | NO   | PRI | NULL    |       |
| hours   | int(11)     | YES  |     | NULL    |       |
| company | varchar(30) | YES  | MUL | NULL    |       |
+---------+-------------+------+-----+---------+-------+
3 rows in set (0.01 sec)
mysql>
```



```
mysql> select * from carmaster;
+----------+------------------------+-------------+
| company  | model                  | rateperhour |
+----------+------------------------+-------------+
| Audi     | 2019 Audi A8           |         100 |
| BMW      | 2019 BMW 5 Series      |          62 |
| Cadillac | 2019 CT 6              |          67 |
| Lexus    | 2020 LS                |          67 |
| Mercedes | 2019 Mercedes S Class  |          75 |
| Telsa    | 2019 Model 3           |          56 |
| Volvo    | 2020 XC 60             |          59 |
+----------+------------------------+-------------+
7 rows in set (0.00 sec)

mysql> select * from rental;
+--------+-------+----------+
| pid    | hours | company  |
+--------+-------+----------+
| 101023 |    15 | Cadillac |
| 101078 |    45 | Audi     |
| 101454 |    65 | Volvo    |
+--------+-------+----------+
3 rows in set (0.00 sec)
```

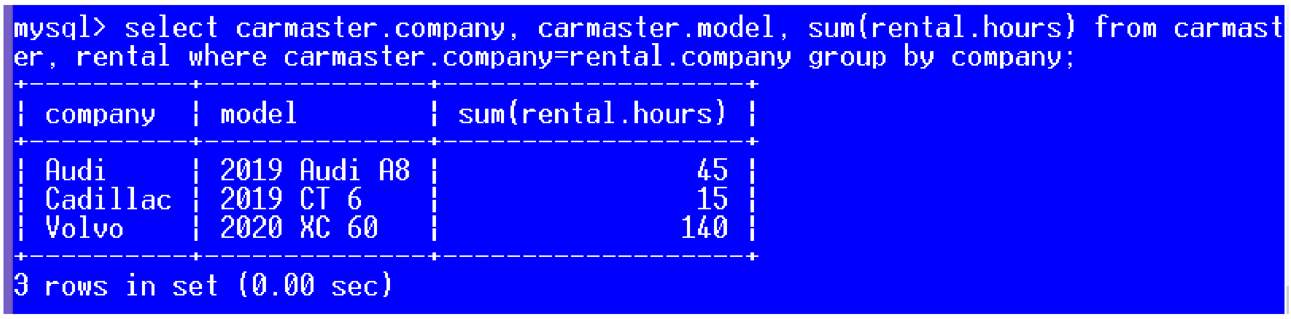
SQL EQUI JOIN performs a JOIN against equality or matching column(s) values of the associated tables. An equal sign (=) is used as comparison operator in the where clause to refer equality.



```
mysql> select carmaster.company, carmaster.model, rental.hours from carmaster, r
ental where carmaster.company=rental.company;
+----------+--------------+-------+
| company  | model        | hours |
+----------+--------------+-------+
| Cadillac | 2019 CT 6    |    15 |
| Audi     | 2019 Audi A8 |    45 |
| Volvo    | 2020 XC 60   |    65 |
+----------+--------------+-------+
3 rows in set (0.06 sec)
```

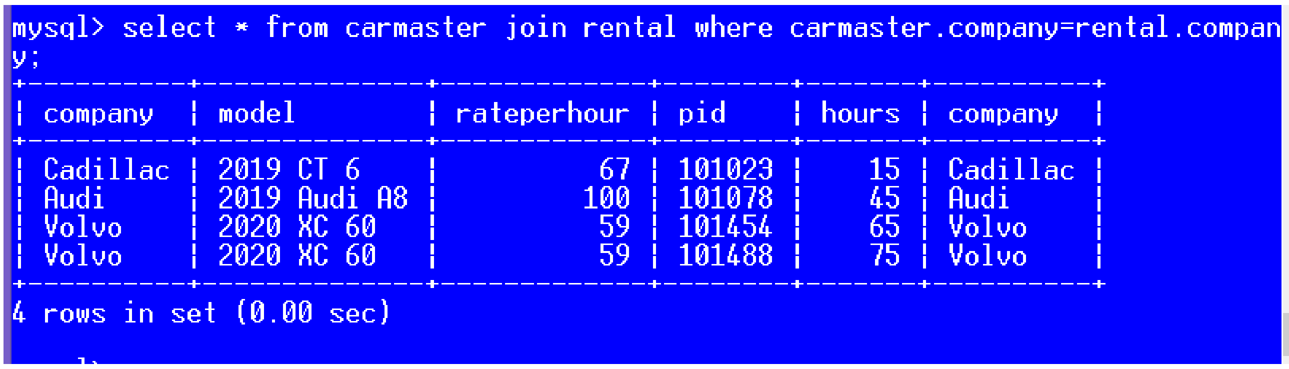
Now where is the application of the equi join?

Somnath PaulChoudhury notes of Computer Science SQL Joins Part 1 for CBSE India



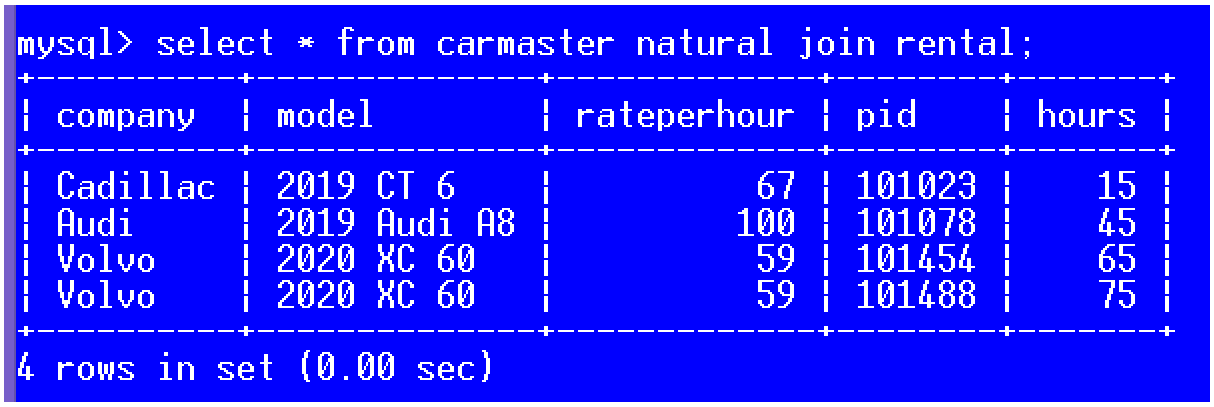
```
mysql> select carmaster.company, carmaster.model, sum(rental.hours) from carmast
er, rental where carmaster.company=rental.company group by company;
+----------+---------------+-------------------+
| company  | model         | sum(rental.hours) |
+----------+---------------+-------------------+
| Audi     | 2019 Audi A8  |                45 |
| Cadillac | 2019 CT 6     |                15 |
| Volvo    | 2020 XC 60    |               140 |
+----------+---------------+-------------------+
3 rows in set (0.00 sec)
```

Displaying all the columns



```
mysql> select * from carmaster join rental where carmaster.company=rental.compan
y;
+----------+---------------+--------------+--------+-------+----------+
| company  | model         | rateperhour  | pid    | hours | company  |
+----------+---------------+--------------+--------+-------+----------+
| Cadillac | 2019 CT 6     |           67 | 101023 |    15 | Cadillac |
| Audi     | 2019 Audi A8  |          100 | 101078 |    45 | Audi     |
| Volvo    | 2020 XC 60    |           59 | 101454 |    65 | Volvo    |
| Volvo    | 2020 XC 60    |           59 | 101488 |    75 | Volvo    |
+----------+---------------+--------------+--------+-------+----------+
4 rows in set (0.00 sec)
```

Natural join



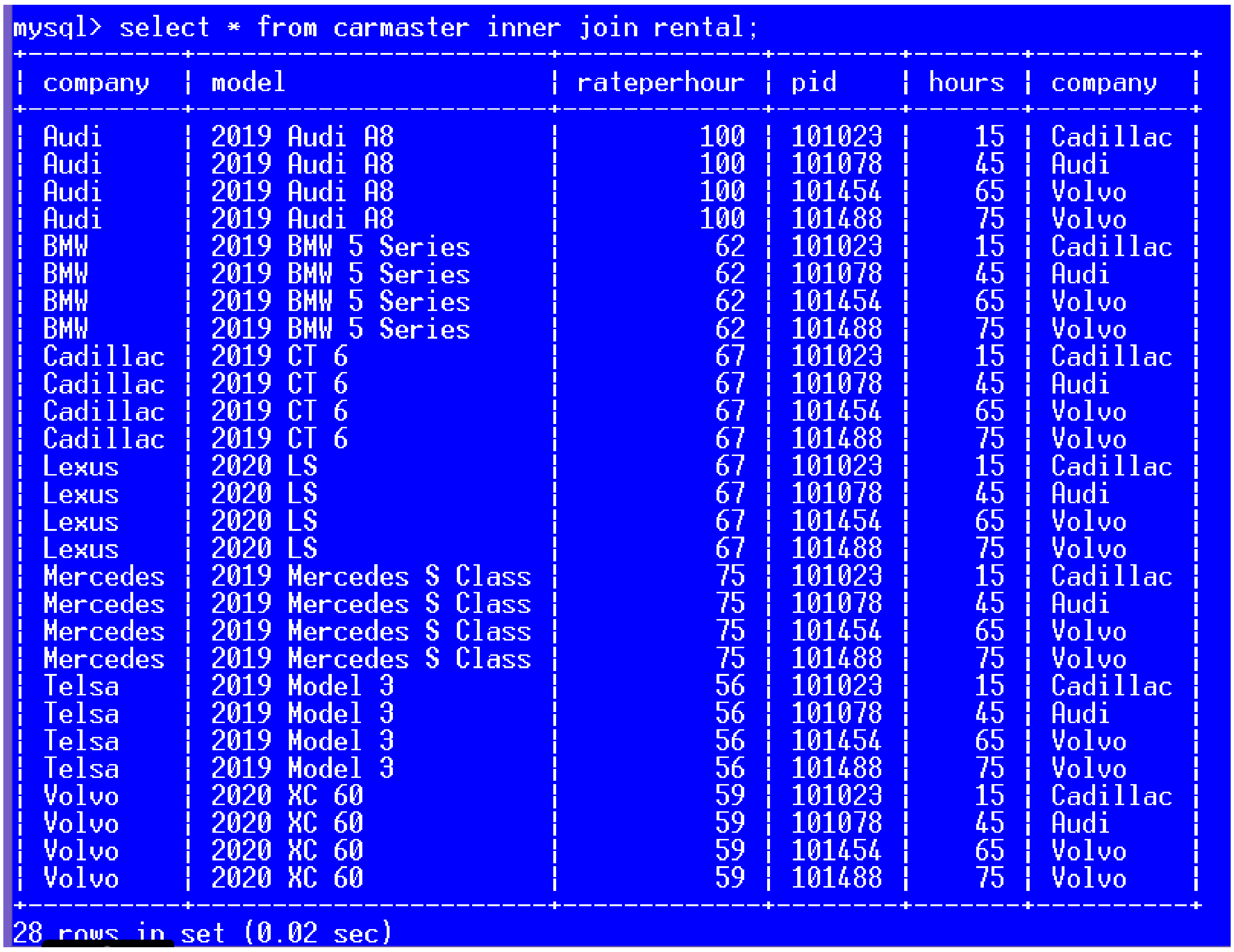
```
mysql> select * from carmaster natural join rental;
+----------+---------------+--------------+--------+-------+
| company  | model         | rateperhour  | pid    | hours |
+----------+---------------+--------------+--------+-------+
| Cadillac | 2019 CT 6     |           67 | 101023 |    15 |
| Audi     | 2019 Audi A8  |          100 | 101078 |    45 |
| Volvo    | 2020 XC 60    |           59 | 101454 |    65 |
| Volvo    | 2020 XC 60    |           59 | 101488 |    75 |
+----------+---------------+--------------+--------+-------+
4 rows in set (0.00 sec)
```

Same column name came once

Difference between natural join and inner join

Below is a simple Inner join

Somnath PaulChoudhury notes of Computer Science SQL Joins Part 1 for CBSE India



```
mysql> select * from carmaster inner join rental;
+----------+-----------------------+-------------+--------+-------+----------+
| company  | model                 | rateperhour | pid    | hours | company  |
+----------+-----------------------+-------------+--------+-------+----------+
| Audi     | 2019 Audi A8          |         100 | 101023 |    15 | Cadillac |
| Audi     | 2019 Audi A8          |         100 | 101078 |    45 | Audi     |
| Audi     | 2019 Audi A8          |         100 | 101454 |    65 | Volvo    |
| Audi     | 2019 Audi A8          |         100 | 101488 |    75 | Volvo    |
| BMW      | 2019 BMW 5 Series     |          62 | 101023 |    15 | Cadillac |
| BMW      | 2019 BMW 5 Series     |          62 | 101078 |    45 | Audi     |
| BMW      | 2019 BMW 5 Series     |          62 | 101454 |    65 | Volvo    |
| BMW      | 2019 BMW 5 Series     |          62 | 101488 |    75 | Volvo    |
| Cadillac | 2019 CT 6             |          67 | 101023 |    15 | Cadillac |
| Cadillac | 2019 CT 6             |          67 | 101078 |    45 | Audi     |
| Cadillac | 2019 CT 6             |          67 | 101454 |    65 | Volvo    |
| Cadillac | 2019 CT 6             |          67 | 101488 |    75 | Volvo    |
| Lexus    | 2020 LS               |          67 | 101023 |    15 | Cadillac |
| Lexus    | 2020 LS               |          67 | 101078 |    45 | Audi     |
| Lexus    | 2020 LS               |          67 | 101454 |    65 | Volvo    |
| Lexus    | 2020 LS               |          67 | 101488 |    75 | Volvo    |
| Mercedes | 2019 Mercedes S Class |          75 | 101023 |    15 | Cadillac |
| Mercedes | 2019 Mercedes S Class |          75 | 101078 |    45 | Audi     |
| Mercedes | 2019 Mercedes S Class |          75 | 101454 |    65 | Volvo    |
| Mercedes | 2019 Mercedes S Class |          75 | 101488 |    75 | Volvo    |
| Telsa    | 2019 Model 3          |          56 | 101023 |    15 | Cadillac |
| Telsa    | 2019 Model 3          |          56 | 101078 |    45 | Audi     |
| Telsa    | 2019 Model 3          |          56 | 101454 |    65 | Volvo    |
| Telsa    | 2019 Model 3          |          56 | 101488 |    75 | Volvo    |
| Volvo    | 2020 XC 60            |          59 | 101023 |    15 | Cadillac |
| Volvo    | 2020 XC 60            |          59 | 101078 |    45 | Audi     |
| Volvo    | 2020 XC 60            |          59 | 101454 |    65 | Volvo    |
| Volvo    | 2020 XC 60            |          59 | 101488 |    75 | Volvo    |
+----------+-----------------------+-------------+--------+-------+----------+
28 rows in set (0.02 sec)
```