CS 207: Applied Database Practicum Week 4

Varun Dutt

School of Computing and Electrical Engineering
School of Humanities and Social Sciences
Indian Institute of Technology Mandi, India

Scaling the Heights

GROUP BY STATEMENT

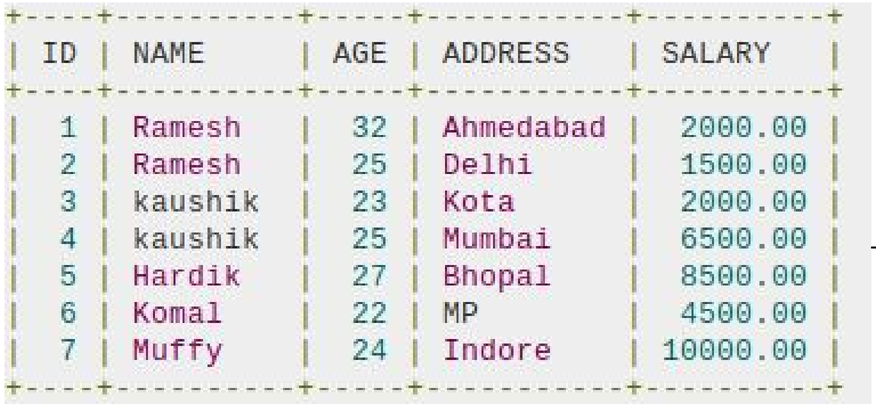
- The GROUP BY statement is used to group the result-set by one or more columns or group rows that have the same values.
- It is often used with aggregate functions (COUNT, MAX, MIN, SUM, AVG)
- GROUP BY Syntax :

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
ORDER BY column_name(s);
```

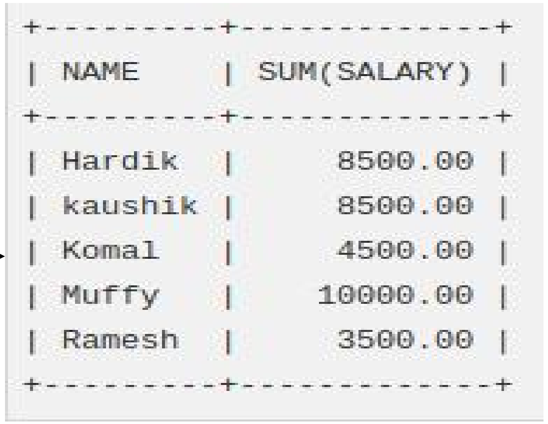
EXAMPLE – GROUP BY

- The following SQL statement lists he total amount of salary on each customer:

SELECT NAME, SUM(SALARY) FROM CUSTOMERS
GROUP BY NAME;



```
+----+----------+-----+-----------+----------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+----------+
|  1 | Ramesh   |  32 | Ahmedabad |  2000.00 |
|  2 | Ramesh   |  25 | Delhi     |  1500.00 |
|  3 | kaushik  |  23 | Kota      |  2000.00 |
|  4 | kaushik  |  25 | Mumbai    |  6500.00 |
|  5 | Hardik   |  27 | Bhopal    |  8500.00 |
|  6 | Komal    |  22 | MP        |  4500.00 |
|  7 | Muffy    |  24 | Indore    | 10000.00 |
+----+----------+-----+-----------+----------+
```



```
+---------+-------------+
| NAME    | SUM(SALARY) |
+---------+-------------+
| Hardik  |     8500.00 |
| kaushik |     8500.00 |
| Komal   |     4500.00 |
| Muffy   |    10000.00 |
| Ramesh  |     3500.00 |
+---------+-------------+
```

EXAMPLE – GROUP BY WITH COUNT FUNCTION

- The following SQL statement lists the number of customers in each country, sorted high to low:

```
SELECT Name, COUNT(*), FROM Customers
GROUP BY Name
ORDER BY COUNT(Name) DESC;
```

Name	**Count(*)**
Ramesh	2
Kaushik	2
Hardik	1
Komal	1
Muffy	1

ORDER BY CLAUSE

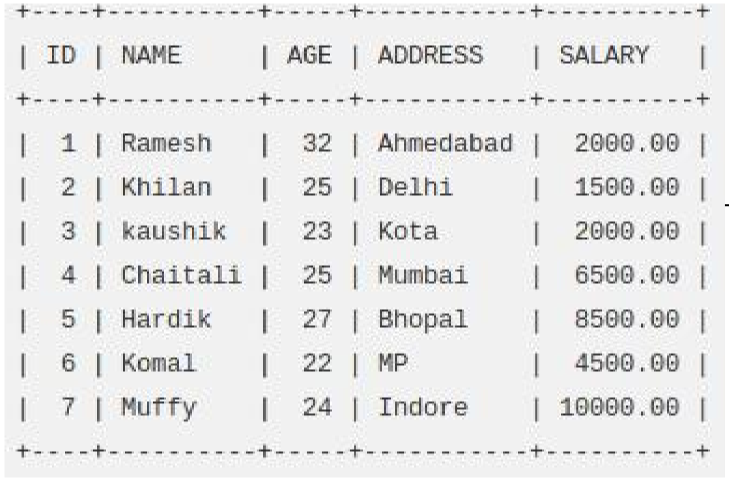
- The SQL ORDER BY clause is used to sort the data in ascending or descending order, based on one or more columns.
- Some databases sort the query results in an ascending order by default.
- Syntax :

```
SELECT column-list
FROM table_name
[WHERE condition]
[ORDER BY column1, column2, .. columnN]
[ASC | DESC];
```

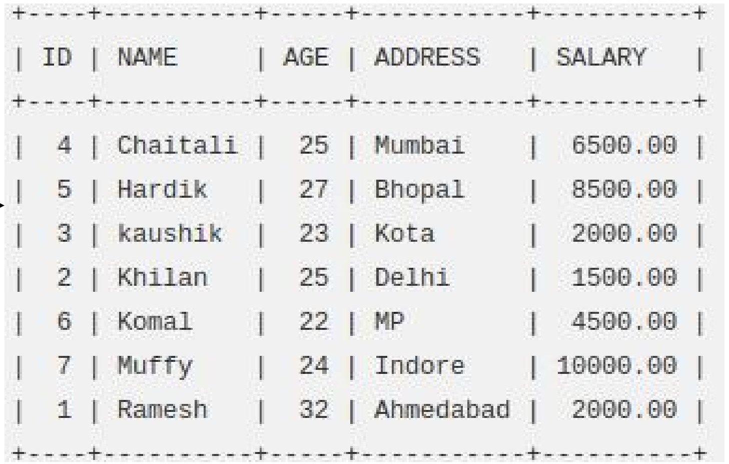
ORDER BY EXAMPLE

- The following SQL statement lists the customers information in an ascending order by the NAME and the SALARY

```
SELECT * FROM CUSTOMERS
ORDER BY NAME, SALARY;
```



```
+----+----------+-----+-----------+----------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+----------+
|  1 | Ramesh   |  32 | Ahmedabad |  2000.00 |
|  2 | Khilan   |  25 | Delhi     |  1500.00 |
|  3 | kaushik  |  23 | Kota      |  2000.00 |
|  4 | Chaitali |  25 | Mumbai    |  6500.00 |
|  5 | Hardik   |  27 | Bhopal    |  8500.00 |
|  6 | Komal    |  22 | MP        |  4500.00 |
|  7 | Muffy    |  24 | Indore    | 10000.00 |
+----+----------+-----+-----------+----------+
```



```
+----+----------+-----+-----------+----------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+----------+
|  4 | Chaitali |  25 | Mumbai    |  6500.00 |
|  5 | Hardik   |  27 | Bhopal    |  8500.00 |
|  3 | kaushik  |  23 | Kota      |  2000.00 |
|  2 | Khilan   |  25 | Delhi     |  1500.00 |
|  6 | Komal    |  22 | MP        |  4500.00 |
|  7 | Muffy    |  24 | Indore    | 10000.00 |
|  1 | Ramesh   |  32 | Ahmedabad |  2000.00 |
+----+----------+-----+-----------+----------+
```

UNION Operator

- The UNION operator is used to combine the result-set of two or more SELECT statements.
- Each SELECT statement within UNION must have the same number of columns
- The columns must also have similar data types
- The columns in each SELECT statement must also be in the same order
- The UNION operator selects only **Distinct** values by default.
- To allow duplicate values, use **UNION ALL**.

UNION SYNTAX

UNION syntax

SELECT *column_name(s)* FROM *table1*
UNION
SELECT *column_name(s)* FROM *table2*;

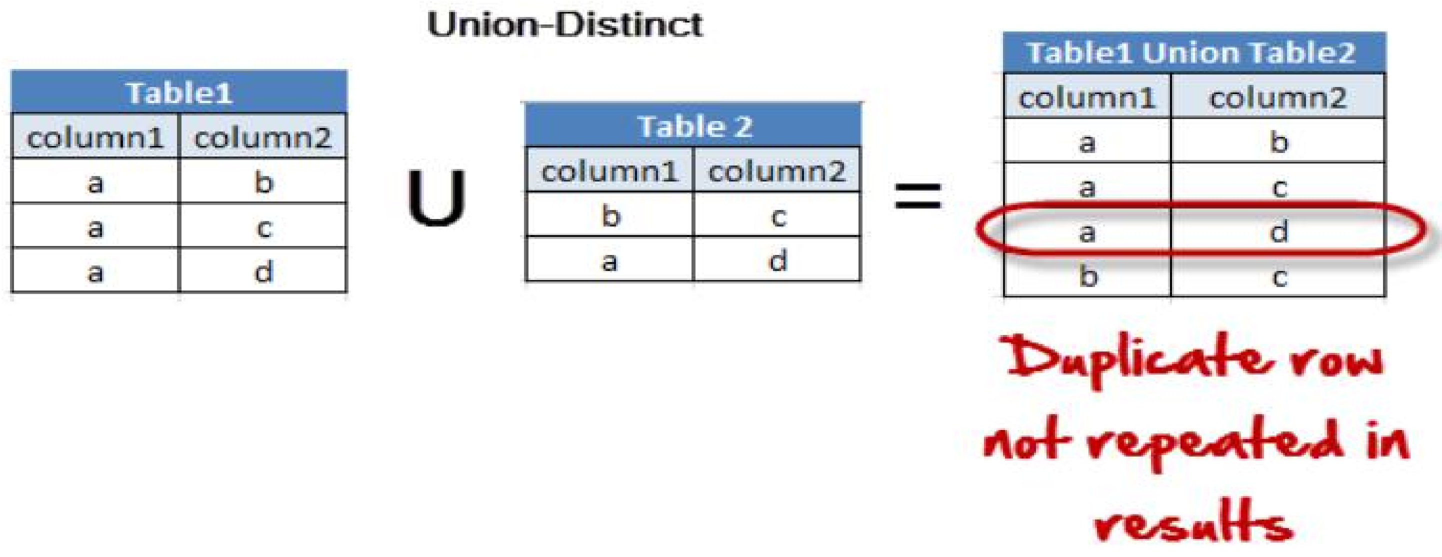

Union-Distinct

Table1	
column1	column2
a	b
a	c
a	d

U

Table 2	
column1	column2
b	c
a	d

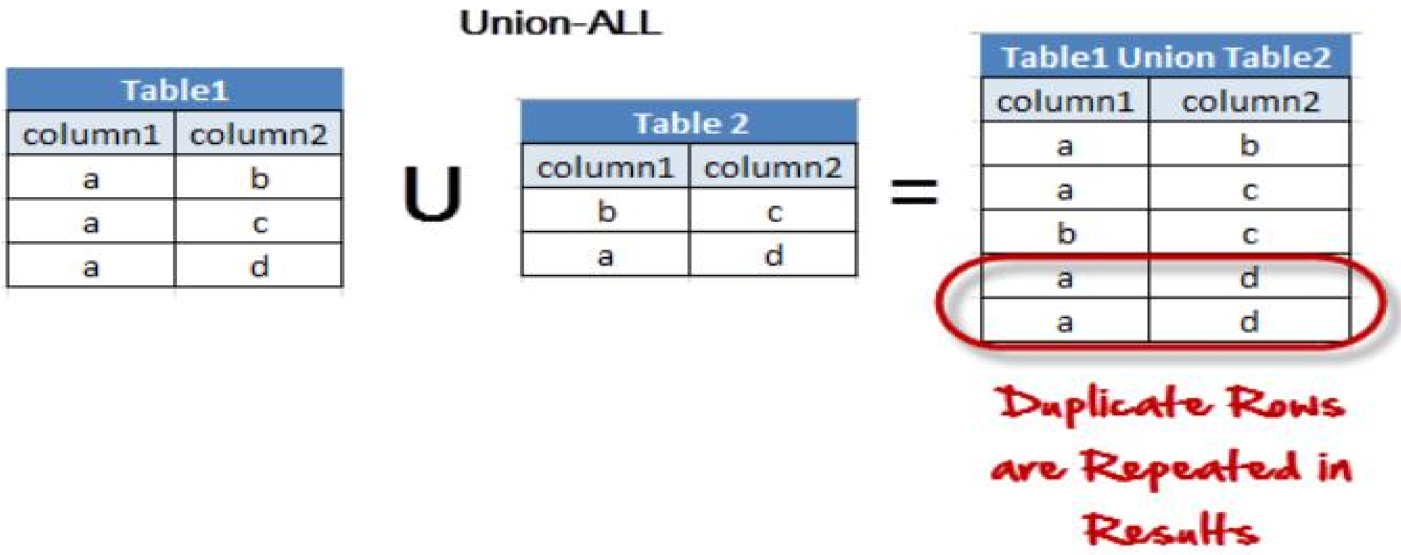
=

Table1 Union Table2	
column1	column2
a	b
a	c
a	d
b	c

UNION ALL SYNTAX

UNION ALL syntax

```
 SELECT column_name(s) FROM table1
UNION ALL
SELECT column_name(s) FROM table2;
```



Union-ALL

Table1	
column1	column2
a	b
a	c
a	d

U

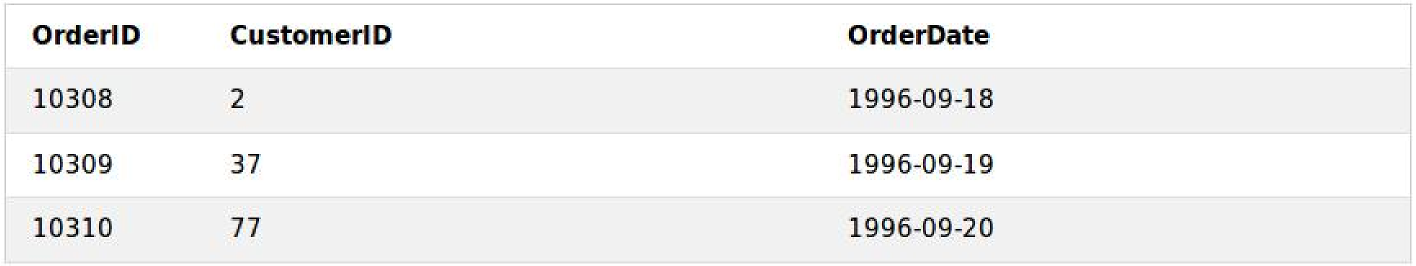
Table 2	
column1	column2
b	c
a	d

=

Table1 Union Table2	
column1	column2
a	b
a	c
b	c
a	d
a	d

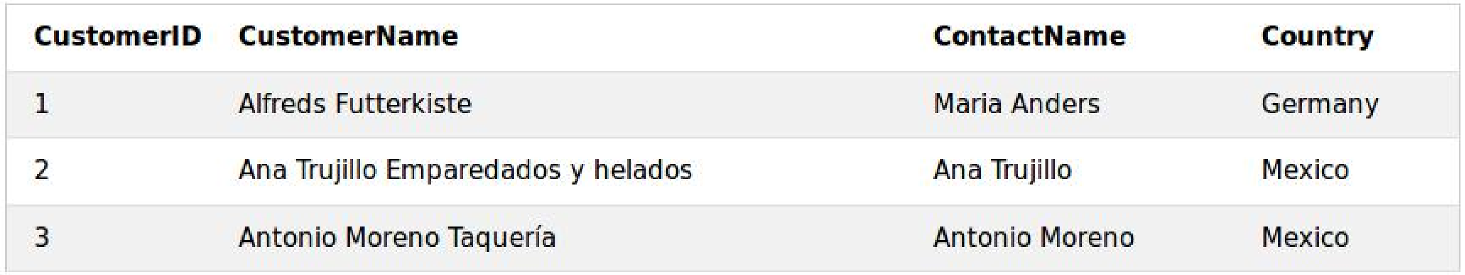
Joins in MySQL

- A JOIN clause is used to combine rows from two or more tables, based on a related column between them.
- For example, have a look at the tables:
- "Orders":



OrderID	CustomerID	OrderDate
10308	2	1996-09-18
10309	37	1996-09-19
10310	77	1996-09-20

- “Customers”:



CustomerID	CustomerName	ContactName	Country
1	Alfreds Futterkiste	Maria Anders	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mexico

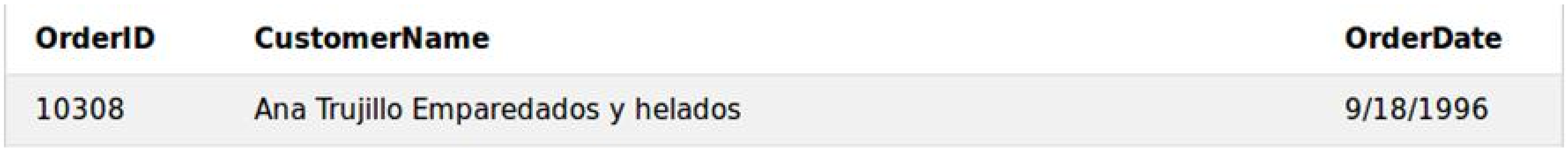
Joins in MySQL

- Notice that the "CustomerID" column in the "Orders" table refers to the "CustomerID" in the "Customers" table. The relationship between the two exemplar tables is the "CustomerID" column.
- Then, we can create the following SQL statement (that contains an INNER JOIN), that selects records that have matching values in both tables:

```
SELECT Orders.OrderID, Customers.CustomerName, Orders.OrderDate
FROM Orders
INNER JOIN Customers ON Orders.CustomerID=Customers.CustomerID;
```

Joins in MySQL

- The output of the previous statement would be:



OrderID	CustomerName	OrderDate
10308	Ana Trujillo Emparedados y helados	9/18/1996

Different Types of MySQL Joins

- **INNER JOIN** - Returns records that have matching values in both tables
- **LEFT (OUTER) JOIN**: Returns all records from the left table, and the matched records from the right table
- **RIGHT (OUTER) JOIN**: Returns all records from the right table, and the matched records from the left table
- **NATURAL JOIN**: It performs the same task as an INNER or LEFT JOIN, in which the ON or USING clause refers to all columns that the tables to be joined have in common.
- **CROSS JOIN:** Returns the complete cross product of the two tables

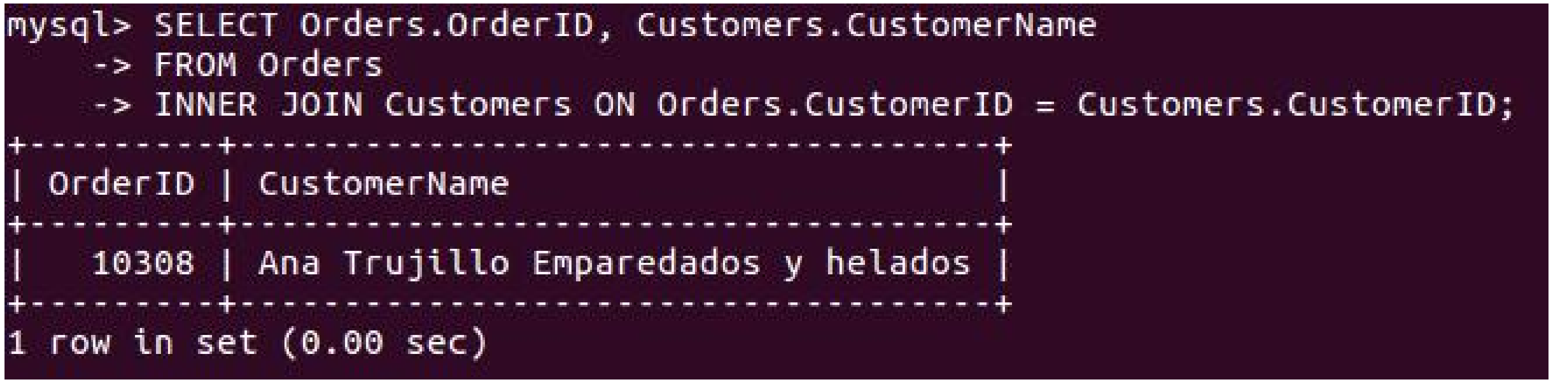
MySQL Inner Join

● The INNER JOIN keyword selects records that have matching values in both tables.

Syntax:

```
SELECT column_name(s)
FROM table1
INNER JOIN table2 ON table1.column_name = table2.column_name;
```

For example:



```
mysql> SELECT Orders.OrderID, Customers.CustomerName
    -> FROM Orders
    -> INNER JOIN Customers ON Orders.CustomerID = Customers.CustomerID;
+---------+------------------------------------+
| OrderID | CustomerName                       |
+---------+------------------------------------+
|   10308 | Ana Trujillo Emparedados y helados |
+---------+------------------------------------+
1 row in set (0.00 sec)
```

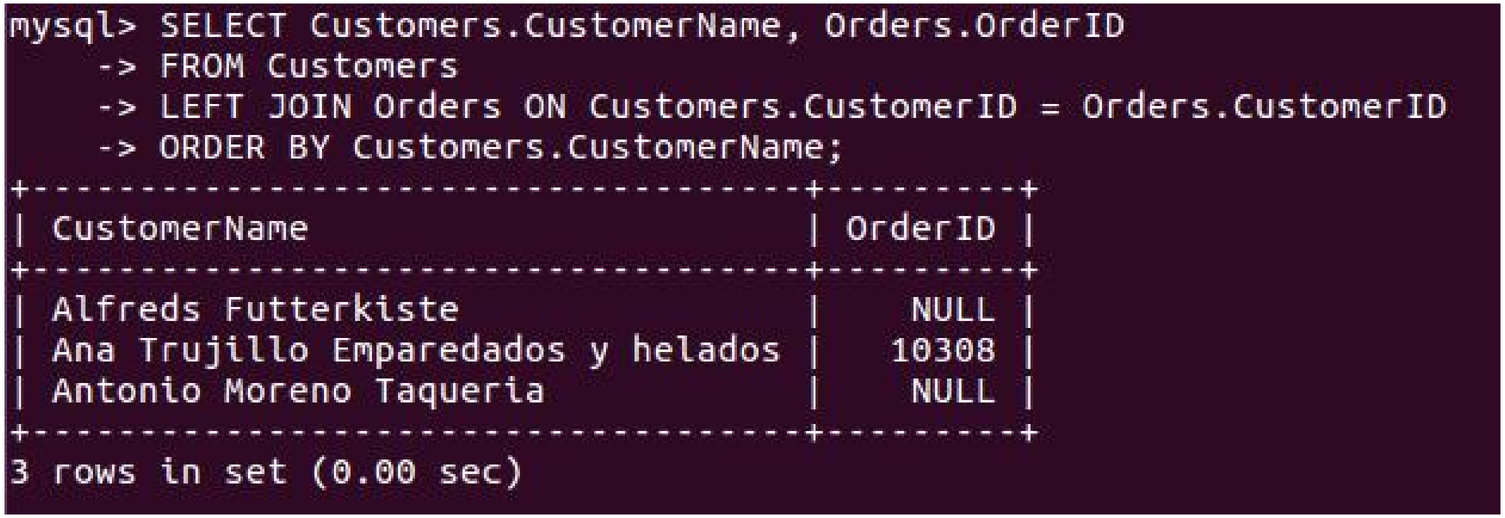
MySQL Left Join

● The LEFT JOIN keyword returns all records from the left table (table1), and the matched records from the right table (table2). The result is NULL from the right side, if there is no match.

Syntax:

```
SELECT column_name(s)
FROM table1
LEFT JOIN table2 ON table1.column_name = table2.column_name;
```

For example:

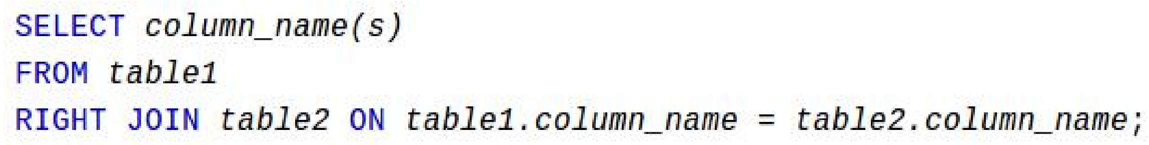


```
mysql> SELECT Customers.CustomerName, Orders.OrderID
    -> FROM Customers
    -> LEFT JOIN Orders ON Customers.CustomerID = Orders.CustomerID
    -> ORDER BY Customers.CustomerName;
+------------------------------------+---------+
| CustomerName                       | OrderID |
+------------------------------------+---------+
| Alfreds Futterkiste                |    NULL |
| Ana Trujillo Emparedados y helados |   10308 |
| Antonio Moreno Taqueria            |    NULL |
+------------------------------------+---------+
3 rows in set (0.00 sec)
```

MySQL Right Join

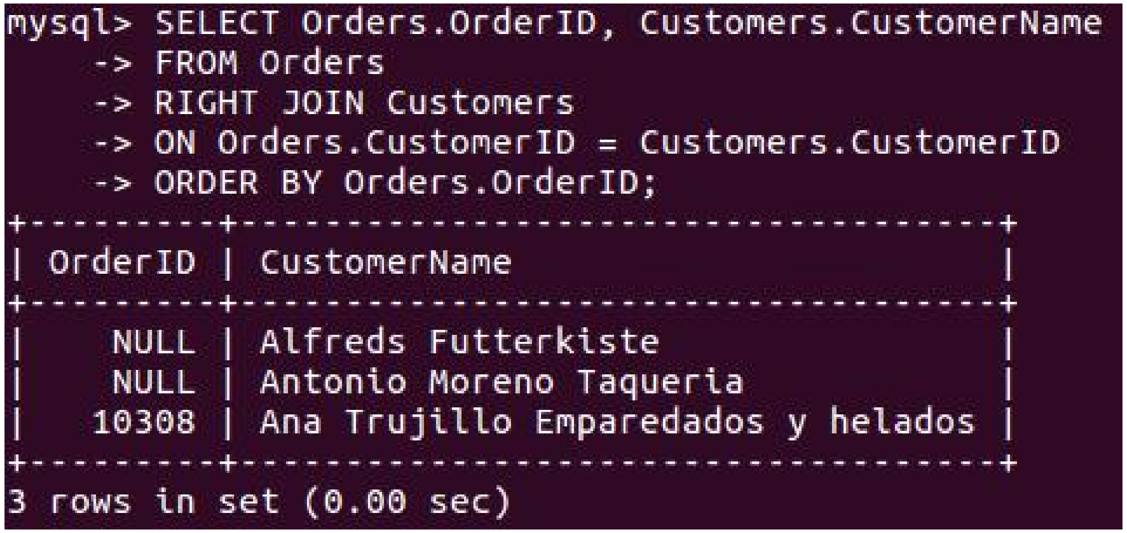
● The RIGHT JOIN keyword returns all records from the right table (table2), and the matched records from the left table (table1). The result is NULL from the left side, when there is no match.

Syntax:



```
SELECT column_name(s)
FROM table1
RIGHT JOIN table2 ON table1.column_name = table2.column_name;
```

For example:

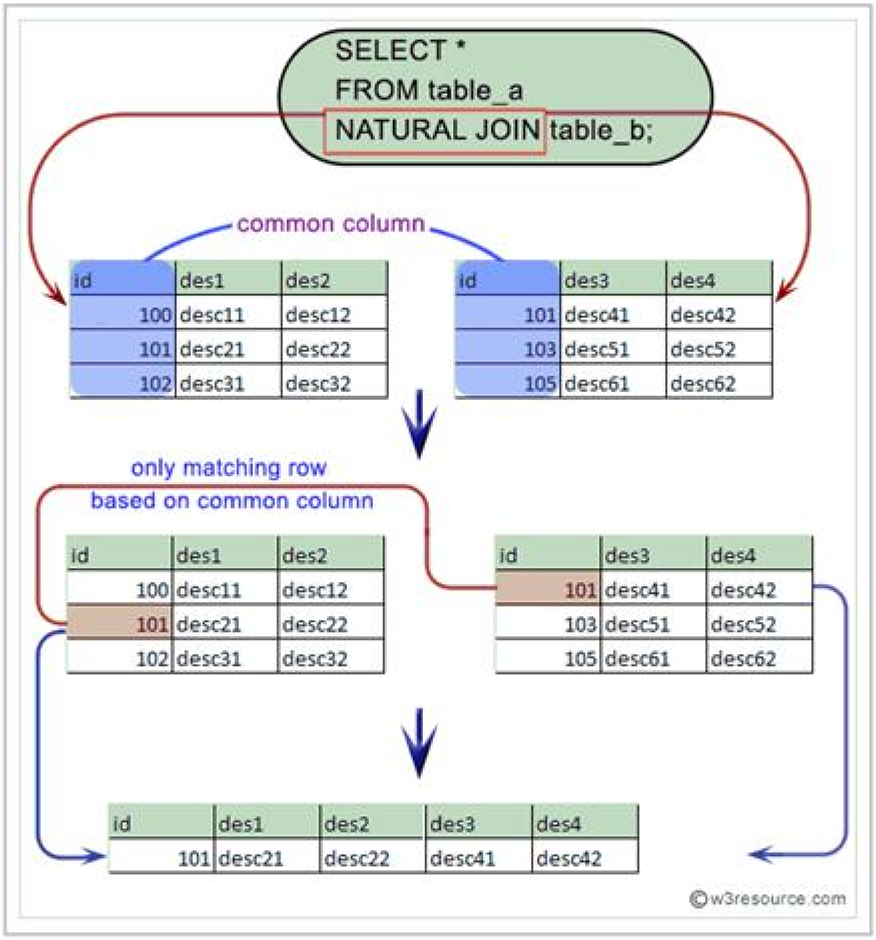


```
mysql> SELECT Orders.OrderID, Customers.CustomerName
    -> FROM Orders
    -> RIGHT JOIN Customers
    -> ON Orders.CustomerID = Customers.CustomerID
    -> ORDER BY Orders.OrderID;
+---------+----------------------------------+
| OrderID | CustomerName                     |
+---------+----------------------------------+
|    NULL | Alfreds Futterkiste              |
|    NULL | Antonio Moreno Taqueria          |
|   10308 | Ana Trujillo Emparedados y helados |
+---------+----------------------------------+
3 rows in set (0.00 sec)
```

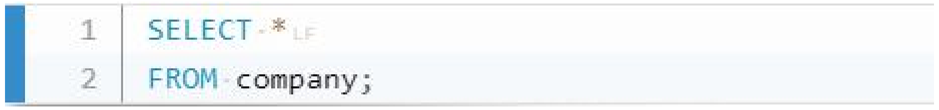
MySQL Natural join

Syntax:
SELECT *column_name(s)* FROM Table 1 NATURAL JOIN Table2;

In MySQL, the NATURAL JOIN is a join that performs the same task as an INNER JOIN except it does not return any redundant columns.

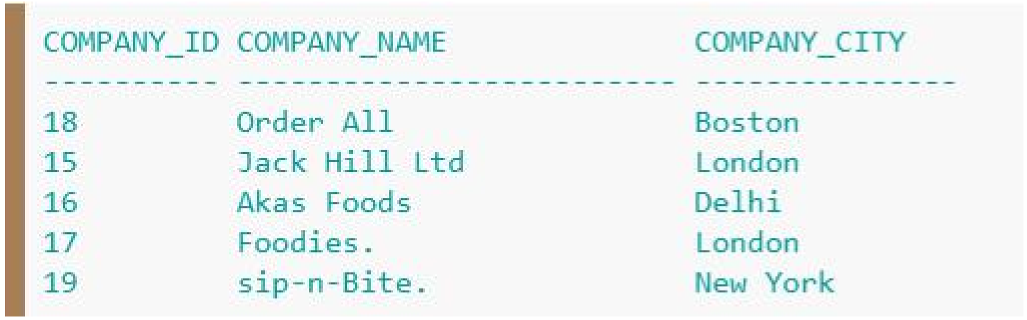


Difference between Natural join and Inner join

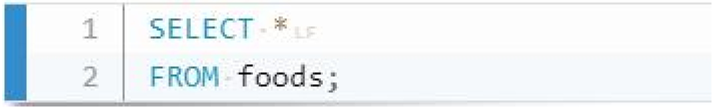


```sql
SELECT *
FROM company;
```

Table 1

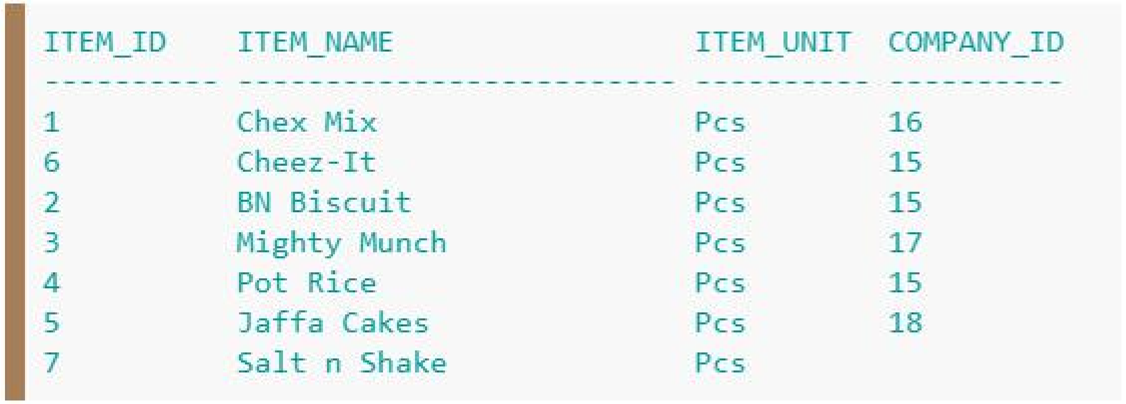


```
COMPANY_ID COMPANY_NAME              COMPANY_CITY
---------- ------------------------- ---------------
18         Order All                 Boston
15         Jack Hill Ltd             London
16         Akas Foods                Delhi
17         Foodies.                  London
19         sip-n-Bite.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



```sql
SELECT *
FROM foods;
```

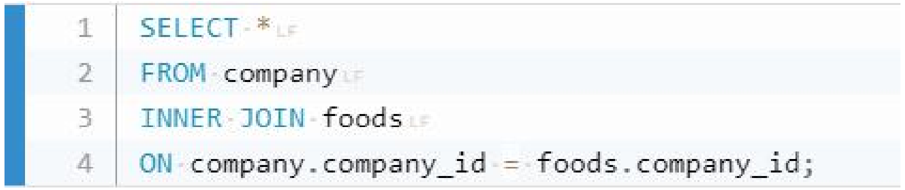
Table 2



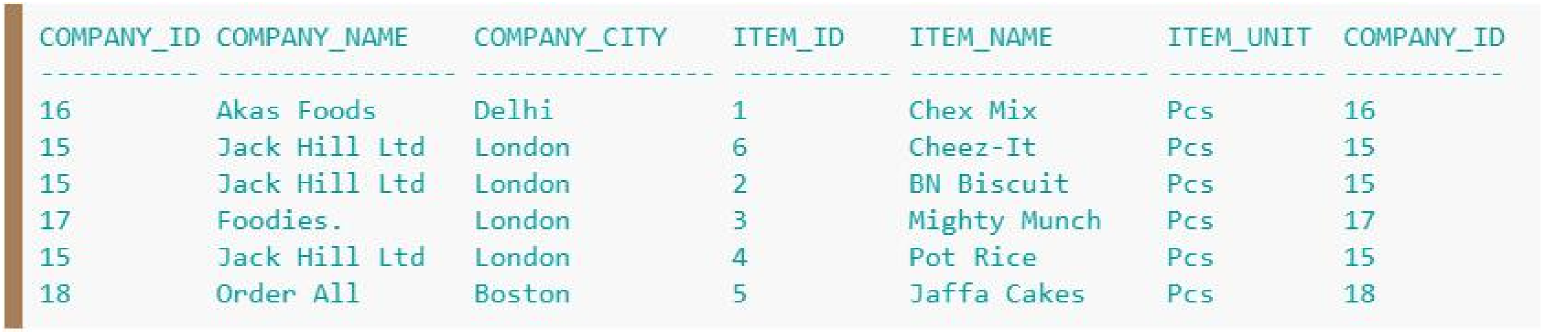
```
ITEM_ID    ITEM_NAME                 ITEM_UNIT  COMPANY_ID
---------- ------------------------- ---------- ----------
1          Chex Mix                  Pcs        16
6          Cheez-It                  Pcs        15
2          BN Biscuit                Pcs        15
3          Mighty Munch  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

Difference between Natural join and Inner join

Inner Join

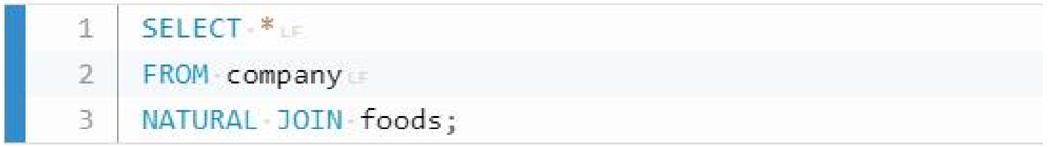


```sql
SELECT *
FROM company
INNER JOIN foods
ON company.company_id = foods.company_id;
```

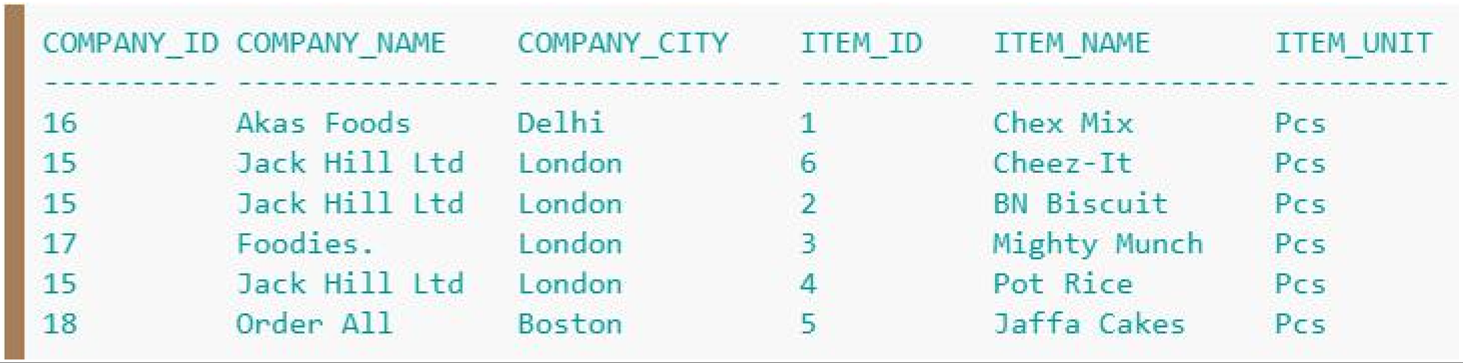


```
COMPANY_ID COMPANY_NAME    COMPANY_CITY    ITEM_ID    ITEM_NAME       ITEM_UNIT  COMPANY_ID
---------- --------------- --------------- ---------- --------------- ---------- ----------
16         Akas Foods      Delhi           1          Chex Mix        Pcs        16
15         Jack Hill Ltd   London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

Natural Join



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



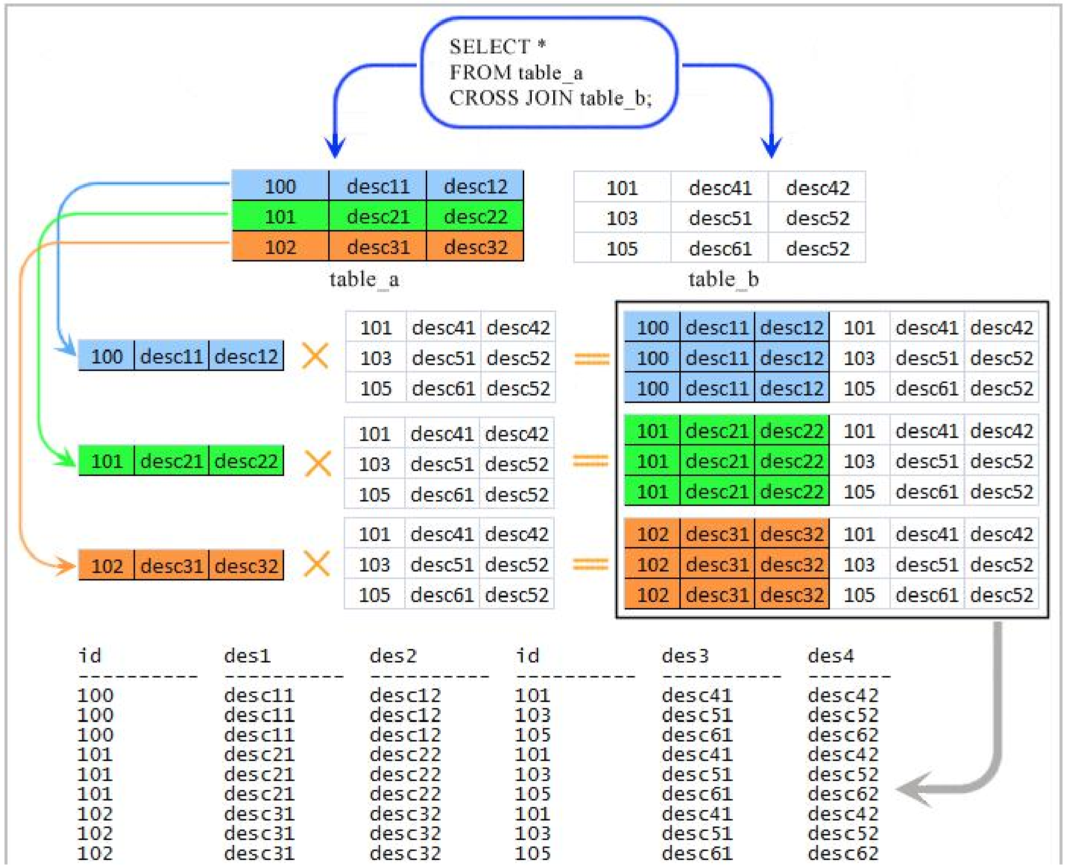
```
COMPANY_ID COMPANY_NAME    COMPANY_CITY    ITEM_ID    ITEM_NAME       ITEM_UNIT
---------- --------------- --------------- ---------- --------------- ----------
16         Akas Foods      Delhi           1          Chex Mix        Pcs
15         Jack Hill Ltd   London          6          Cheez-It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

MySQL cross join

Syntax:

SELECT *column_name(s)* FROM Table 1 CROSS JOIN Table2;

In this join, the result table is obtained by multiplying each row of the first table with all rows in the second table if no condition is introduced with CROSS JOIN.



```
SELECT *
FROM table_a
CROSS JOIN table_b;
```

table_a

100	desc11	desc12
101	desc21	desc22
102	desc31	desc32

table_b

101	desc41	desc42
103	desc51	desc52
105	desc61	desc52

100	desc11	desc12	101	desc41	desc42
100	desc11	desc12	103	desc51	desc52
100	desc11	desc12	105	desc61	desc52
101	desc21	desc22	101	desc41	desc42
101	desc21	desc22	103	desc51	desc52
101	desc21	desc22	105	desc61	desc52
102	desc31	desc32	101	desc41	desc42
102	desc31	desc32	103	desc51	desc52
102	desc31	desc32	105	desc61	desc52

```
id          des1        des2        id          des3        des4
----------  ----------  ----------  ----------  ----------  -------
100         desc11      desc12      101         desc41      desc42
100         desc11      desc12      103         desc51      desc52
100         desc11      desc12      105         desc61      desc62
101         desc21      desc22      101         desc41      desc42
101         desc21      desc22      103         desc51      desc52
101         desc21      desc22      105         desc61      desc62
102         desc31      desc32      101         desc41      desc42
102         desc31      desc32      103         desc51      desc52
102         desc31      desc32      105         desc61      desc62
```

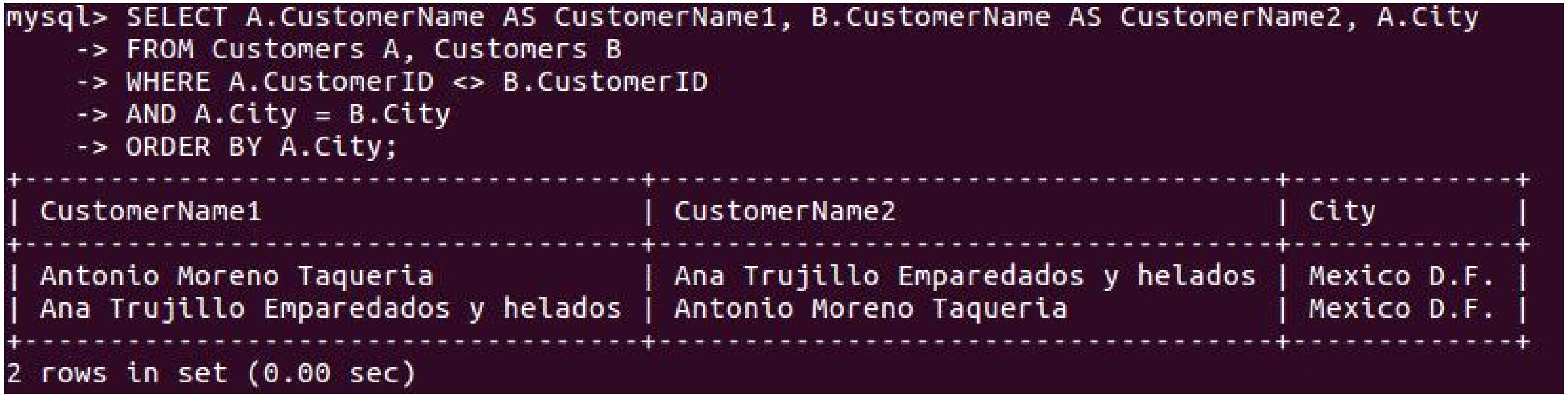
MySQL Self Join

- A self JOIN is a regular join, but the table is joined with itself.

  Syntax:

```
SELECT column_name(s)
FROM table1 T1, table1 T2
WHERE condition;
```

For example:



```
mysql> SELECT A.CustomerName AS CustomerName1, B.CustomerName AS CustomerName2, A.City
    -> FROM Customers A, Customers B
    -> WHERE A.CustomerID <> B.CustomerID
    -> AND A.City = B.City
    -> ORDER BY A.City;
+------------------------------------+------------------------------------+-------------+
| CustomerName1                      | CustomerName2                      | City        |
+------------------------------------+------------------------------------+-------------+
| Antonio Moreno Taqueria            | Ana Trujillo Emparedados y helados | Mexico D.F. |
| Ana Trujillo Emparedados y helados | Antonio Moreno Taqueria            | Mexico D.F. |
+------------------------------------+------------------------------------+-------------+
2 rows in set (0.00 sec)
```

REFERENCES

- https://www.w3schools.com/sql/sql_groupby.asp
- https://www.tutorialspoint.com/sql/sql-order-by.htm
- https://www.w3schools.com/sql/sql_union.asp
- https://www.w3resource.com/mysql/mysql-union.php
- https://www.w3schools.com/sql/sql_join.asp
- http://www.mysqltutorial.org/mysql-join/