

MySQL SELF JOIN

MySQL Self JOIN is an SQL statement that is used to intersect or join a table in the database to itself. Basically, a JOIN function combines two or more tables in SQL, but here Self Join is a regular Join that allows to combine records within a table based on a certain conditional expression. This type of joining of records with other records in the same table when a condition is matched is also called as a Unary relationship. Specially, here the Foreign key that exists in the table refers to the tables own Primary Key.

This means that a table row will join with every other row under a condition and this process is queried by a Self-Join SQL command. This Self Join in MySQL is a special kind of Join that is useful to query the hierarchical data and comparing records in the rows within a table. In the SQL statement query, we use Inner Join or Left Join Clauses with the Self Join to return a result set using any condition that satisfies. This is because it is difficult to refer more than one query for the same table. So, we have to use a different table alias to add a different name for the table when we use Self Join in MySQL otherwise the query displays an error when executed.

```
SELECT A.Column_Name, B.Column_Name,...FROM TableA T1, TableA T2  
WHERE A.Common_Field = B.Common_Field;
```

Here, **T1** and **T2** are two aliases used for the same table named Table A in the database and the WHERE clause denotes any given conditional expression.

How MySQL Self Join works?

- In MySQL, Self-Join is important to query ordered data by comparing rows in the same table. The SQL query for Self-Join is applied in a table to itself showing as if there are two tables where using temporary names as alias for the same table in SQL statement.
- Therefore, the main application of this query in MySQL is when we have a table reference data rows in itself. Suppose, we have a table named Employee

and there is a Supervisor_ID column that refers to the employee which is the head of the current employees in any particular company.

- Thus, there is a requirement of any relationship between the rows data that is stored in the table to apply the Self Join in MySQL within the same table.
- Let us consider a table named Persons with columns (PersonID, Name, Contact, Address, City).
- Suppose, we want to match persons that are from the same city in the table then, we use the following SQL Statement for Self-Join command in MySQL:

```
SELECT A.Name AS Name1, B.Name AS Name2, A.City FROM Persons A, Persons  
B WHERE A.PersonID <> B.PersonID AND A.City = B.City ORDER BY A.City;
```

A SELF JOIN is a join that is used to join a table with **itself**. In the previous sections, we have learned about the joining of the table with the other tables using different JOINS, such as INNER, LEFT, RIGHT, and CROSS JOIN. However, there is a need to combine data with other data in the same table itself. In that case, we use Self Join.

We can perform Self Join using **table aliases**. The table aliases allow us not to use the same table name twice with a single statement. If we use the same table name more than one time in a single query without table aliases, it will throw an error.

The table aliases enable us to use the **temporary name** of the table that we are going to use in the query. Let us understand the table aliases with the following explanation.

Suppose we have a table named "**student**" that is going to use twice in the single query. To aliases the student table, we can write it as:

```
Select ... FROM student AS S1  
INNER JOIN student AS S2;
```

SELF JOIN Syntax

The syntax of self-join is the same as the syntax of joining two different tables. Here, we use aliases name for tables because both the table name are the same. The following are the syntax of a SELF JOIN in MySQL:

```
SELECT s1.col_name, s2.col_name...
FROM table1 s1, table1 s2
WHERE s1.common_col_name = s2.common_col_name;
```

NOTE: You can also use another condition instead of WHERE clause according to your requirements.

SELF JOIN Example

Let us create a table "**student**" in a database that contains the following data:

student_id	name	course_id	duration
1	Adam	1	3
2	Peter	2	4
1	Adam	2	4
3	Brian	3	2
2	Shane	3	5

Now, we are going to get all the result (student_id and name) from the table where **student_id** is equal, and **course_id** is not equal. Execute the following query to understand the working of self-join in MySQL:

```
SELECT s1.student_id, s1.name
FROM student AS s1, student s2
WHERE s1.student_id=s2.student_id
AND s1.course_id<>s2.course_id;
```

After the successful execution, we will get the following output:

student_id	name
1	Adam
2	Shane
1	Adam
2	Peter

SELF JOIN using INNER JOIN clause

The following example explains how we can use Inner Join with Self Join. This query returns the student id and name when the student_id of both tables is equals, and course_id are not equal.

```
SELECT s1.student_id, s1.name
FROM student s1
INNER JOIN student s2
ON s1.student_id=s2.student_id AND s1.course_id<>s2.course_id
GROUP BY student;
```

After executing the above statement, we will get the following example:

student_id	name
1	Adam
2	Shane

SELF JOIN using LEFT JOIN clause

The following example explains how we can use LEFT Join with Self Join. This query returns the student name as **monitor** and city when the student_id of both tables are equals.

1. **SELECT** (CONCAT(s1.stud_lname, ' ', s2.stud_fname)) **AS** 'Monitor', s1.city
2. **FROM** students s1
3. **LEFT JOIN** students s2 **ON** s1.student_id=s2.student_id
4. **ORDER BY** s1.city **DESC**;

After executing the above statement, we will get the following example:

Monitor	city
Putin Devine	France
Miller Ethon	England
Clark Michael	Australiya
Strauss Mark	America

SQL Self Join Example

The following SQL statement matches customers that are from the same city:

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico

Example

```
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;
```

Examples of MySQL Self Join

Given below are the examples of MySQL self Join: Let us consider a table named Persons.

Example #1 – MySQL Self Join using Inner Join Clause

Let us create a table Persons and insert some data as a sample to execute the query on them using Self Join in MYSQL with the following SQL commands:

```
CREATE TABLE Persons (PersonID int NOT NULL PRIMARY KEY,
LastName varchar(255) NOT NULL, FirstName varchar(255),
ReportsTo int, Title varchar(255));
```

Output:

Showing rows 0 - 4 (5 total, Query took 0.0062 seconds)

```
SELECT * FROM `persons`;
```

Options

	PersonID	LastName	FirstName	ReportsTo	Title
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	101	Eichsen	John	104	Manager
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	102	Shah	Anita	101	Sales Rep
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	103	Sharma	Karuna	101	IT Rep
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	104	Eichsen	John	102	VP Sales
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	105	Pandit	Neha	0	Marketing

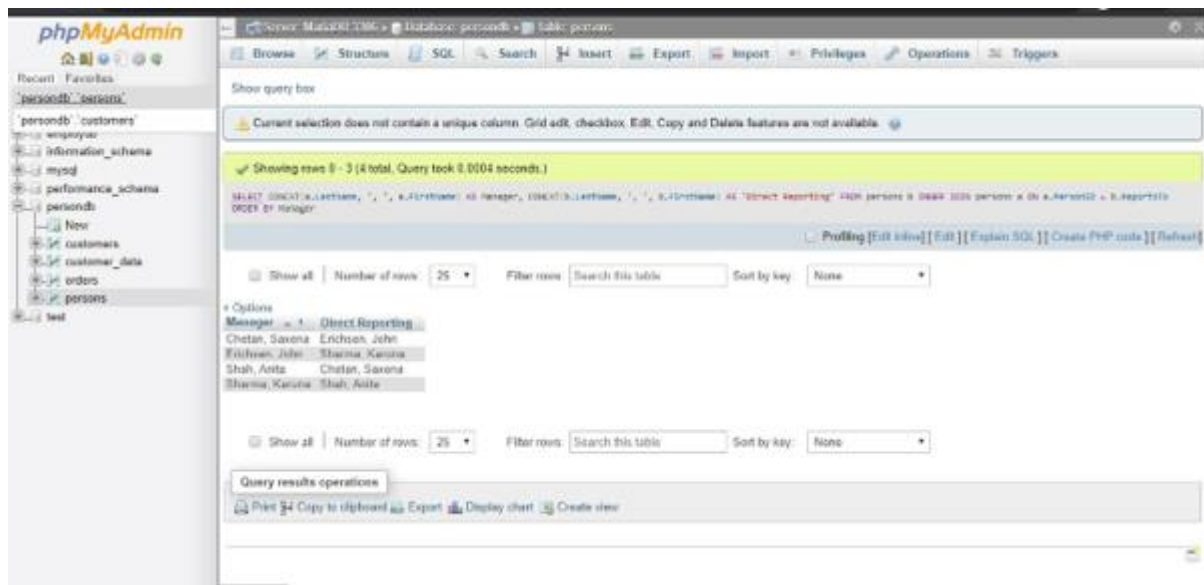
Query results operations

Print ☐ Copy to clipboard ☐ Export ☐ Display chart ☐ Create view

Now, let us fetch the whole organization structure details. We are joining the Person table with itself based on the PersonID and ReportsTo column from the table which determines the job title with employees.

```
SELECT CONCAT (a.LastName, ', ', a.FirstName) AS Manager,  
CONCAT(b.LastName, ', ', b.FirstName) AS 'Direct Reporting' FROM  
persons b INNER JOIN persons a ON a.PersonID = b.ReportsTo  
ORDER BY Manager;
```

Output:



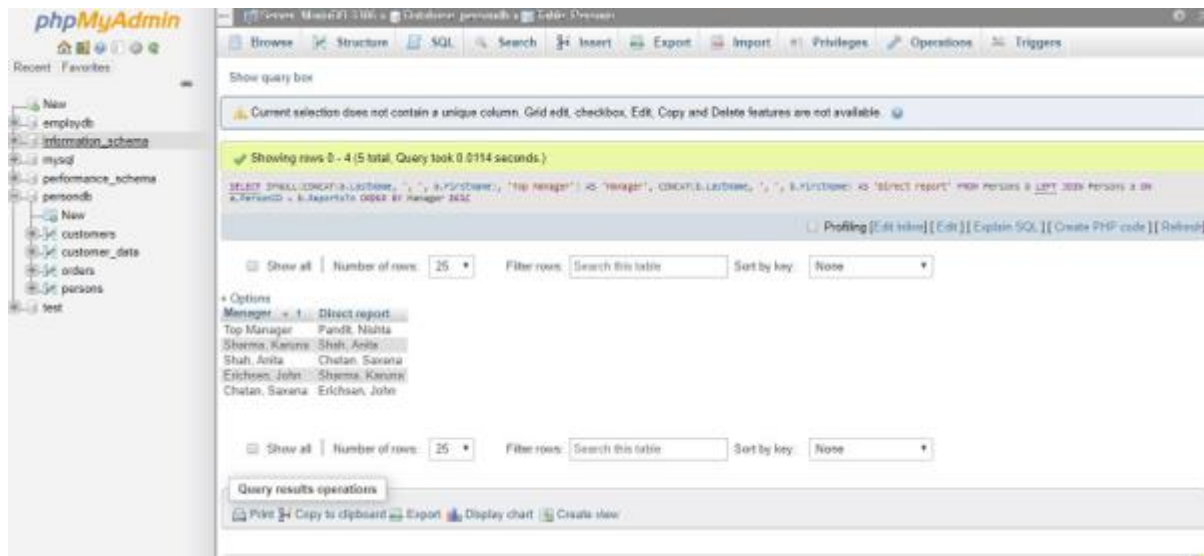
The result set will define the table of Persons with two different roles: 'Manager' and 'Direct Reporting' and only those employees who have a manager. However, the 'Marketing' is not displayed because the name is filtered out by the Inner Join clause as Reports to column is NULL with no value to match the condition applied.

Example #2 – MySQL Self Join using Left Join Clause

Since the Marketing titled person has no Manager and is not included in the above result set but in the Left Join, we can display it even if it has no value for Manager.

```
SELECT IFNULL (CONCAT (a.LastName, ' ', a.FirstName), 'Top  
Manager') AS 'Manager', CONCAT(b.LastName, ' ', b.FirstName) AS  
'Direct report' FROM Persons b LEFT JOIN Persons a ON a.PersonID =  
b.ReportsTo ORDER BY Manager DESC;
```

Output:



Here, the Marketing person is provided with 'Top Manager' as 'Manager' as its value for Reports To was NULL.

Example #3 – Use MySQL Self Join for comparing successive rows

For example, let us use Self Join in MySQL to fetch the rows by comparing the records within the same table. Let us consider another table named 'Customer_Data' with fields (ID, name, Age, Address, Salary) for this SQL query:

The SQL statement is as follows:

```
SELECT k.ID, l.Name, k.Salary FROM customer_data k,  
customer_data l WHERE k.Salary < l.Salary;
```


phpMyAdmin

Recent Favorites

- New
- employdb
- information_schema
- mysql
- performance_schema
- persondb
 - New
 - customers
 - customer_data
 - orders
 - persons
- test

Showing rows 0 - 4 (5 total, Query took 0.0002 seconds)

```
DELETE FROM `customer_data`
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

```
UPDATE `customer_data` SET `salary` = `name` WHERE `customer_data`.`ID` = 4;
```

[Edit inline] [Edit] [Create PHP code]

Show all | Number of rows: 25 | Filter rows: Search this table | Sort by key: None

Options

ID	Name	Age	Address	Salary
1	Erica Smith	21	Norway	7500
2	Rajesh Tripathi	30	Dharwad	8000
3	Smithi	22	Delhi	9500
4	Hardik	22	Bhopal	8500
5	Komal	24	MP	4500

Check all | With selected | Edit | Copy | Delete | Export

Show all | Number of rows: 25 | Filter rows: Search this table | Sort by key: None

Query results operations

Print | Copy to clipboard | Export | Display chart | Create view

Console

Output:

phpMyAdmin

Recent Favorites

- New
- employdb
- information_schema
- mysql
- performance_schema
- persondb
 - New
 - customers
 - customer_data
 - orders
 - persons
- test

Show query box

Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available.

Showing rows 0 - 9 (10 total, Query took 0.0003 seconds)

```
DELETE k.ID, l.name, k.salary FROM customer_data k, customer_data l WHERE k.salary < l.salary
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 | Filter rows: Search this table

Options

ID	Name	Salary
5	Erica Smith	4500
1	Rajesh Tripathi	7000
5	Rajesh Tripathi	4500
1	Smithi	7000
2	Smithi	6000
4	Smithi	8500
5	Smithi	4500
1	Hardik	7000
2	Hardik	6000
5	Hardik	4500

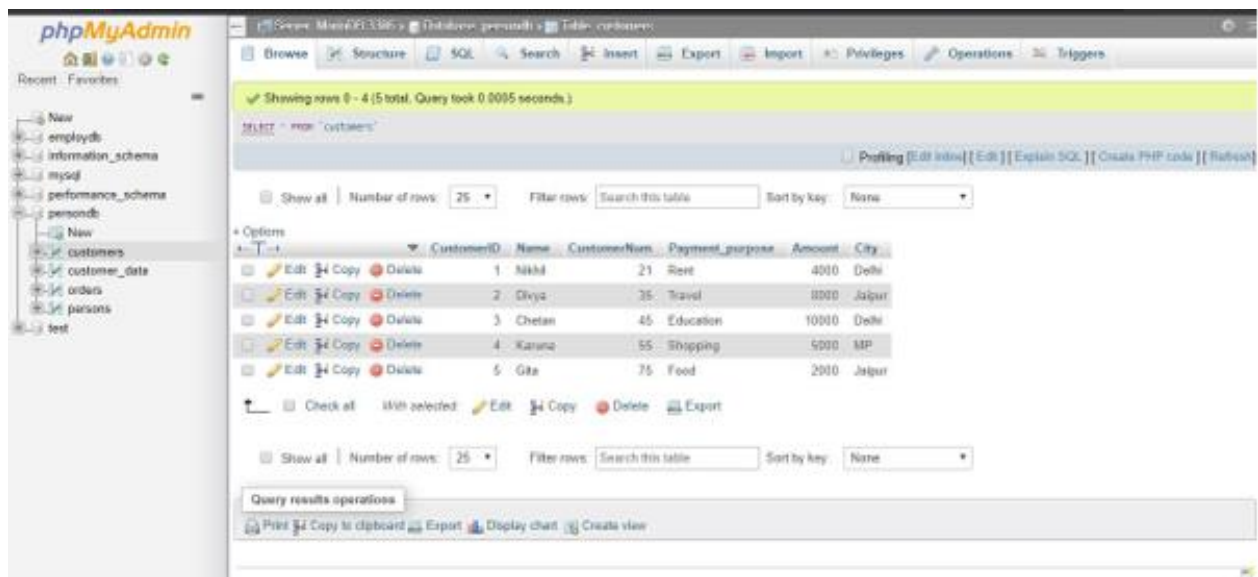
Show all | Number of rows: 25 | Filter rows: Search this table

Query results operations

Print | Copy to clipboard | Export | Display chart | Create view

Example #4 – Use MySQL Self Join with Inner Join for comparing successive rows

We have taken the 'Customers' table with fields (CustomerID, Name, CustomerNum, Payment_purpose, Amount, City) as an example or sample table to run the query using Self Join and comparing the rows in the table.



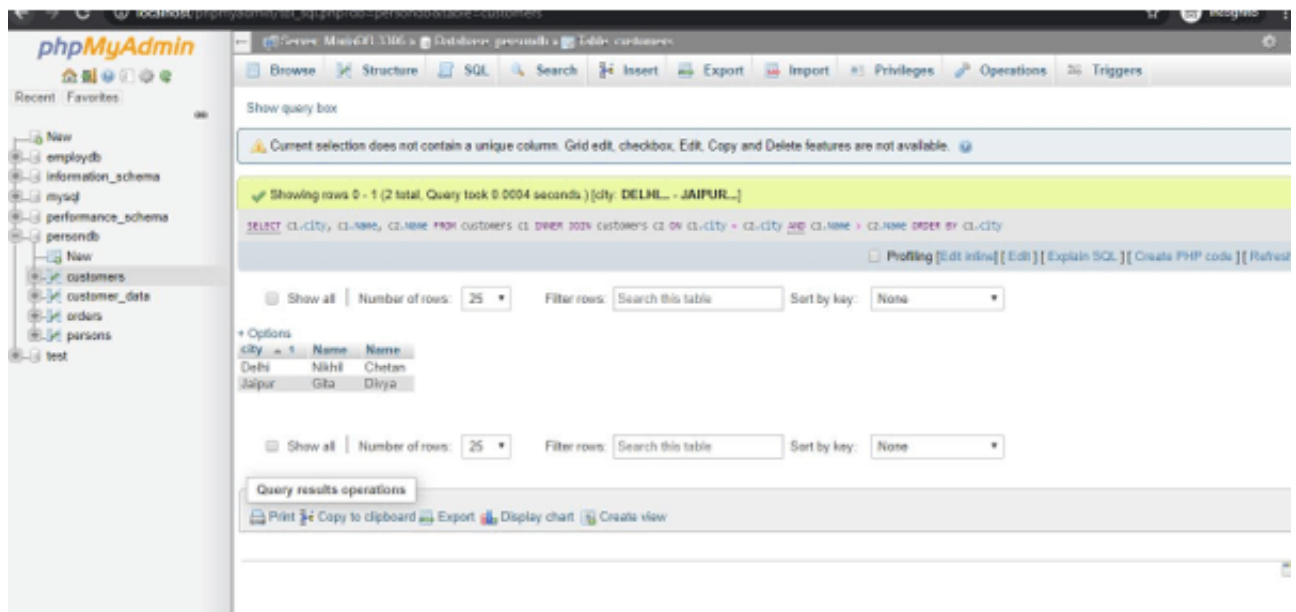
The screenshot shows the phpMyAdmin interface with the 'customers' table selected. The table contains 5 rows of data. The columns are CustomerID, Name, CustomerNum, Payment_purpose, Amount, and City. The data is as follows:

CustomerID	Name	CustomerNum	Payment_purpose	Amount	City
1	Nikhil	21	Rent	4000	Delhi
2	Divya	35	Travel	8000	Jaipur
3	Chetan	45	Education	10000	Delhi
4	Karuna	55	Shopping	5000	MP
5	Gita	75	Food	2000	Jaipur

Here, the list of persons from Customers table is fetched who are located in the same city by joining the table to itself and using comparing operators like $<$, $>$ and $=$ for City column.

```
SELECT c1.city, c1.Name, c2.Name FROM customers c1 INNER JOIN  
customers c2 ON c1.city = c2.city AND c1.Name > c2.Name ORDER  
BY c1.city;
```

Output:



You can see there the result table a city column and two Name columns which contain different names of customers residing in the same place or city using the Join conditions where:

- `city = c2.city` ensures that both customers have the same city.
- `Name > c2.Name ORDER BY c1.city` checks that no same name of customers are included in the result rows.

Thus, likewise, we can apply it to fetch results from the same table under certain joining predicates and using Inner or Left Joins as well as comparison operators to query correct data from the table in the database.

Conclusion

Hence, we can say that a Self-Join in MySQL takes a 'Selfie' to itself but producing results considering the table as two using an alias so that the table name is not repeated twice which may produce an error. This type of regular

join is useful for modelling an organized structure data form from a table and also allows comparing the rows and does needful to produce the required result.