The Employee table holds all employees including their managers. Every employee has an Id, and there is also a column for the manager Id.

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

| Id | Name | Salary | ManagerId |

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

| 1 | Joe | 70000 | 3 |

| 2 | Henry | 80000 | 4 |

| 3 | Sam | 60000 | NULL |

| 4 | Max | 90000 | NULL |

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

Given the Employee table, write a SQL query that finds out employees who earn more than their managers. For the above table, Joe is the only employee who earns more than his manager.

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

| Employee |

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

| Joe |

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

**SOLUTION**:

**SELECT** employee.Name

**FROM** Employee employee

**JOIN** Employee manager **ON** employee.ManagerId = manager.Id

**AND** employee.Salary **>** manager.Salary

Table Alias

**KEY WORD**

**Technique:** [**SELF JOIN**](http://www.mysqltutorial.org/mysql-self-join/) , you use self-join when you want to combine rows with other rows in the same table. To perform the self-join operation, you must use a[**table alias**](http://www.mysqltutorial.org/mysql-alias/) to help MySQL distinguish the left table from the right table of the same table.

Write a SQL query to find all duplicate emails in a table named Person.

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

| Id | Email |

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

| 1 | a@b.com |

| 2 | c@d.com |

| 3 | a@b.com |

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

For example, your query should return the following for the above table:

+---------+

| Email |

+---------+

| a@b.com |

+---------+

**Note**: All emails are in lowercase.

**SOLUTION**:

**SELECT** Email

**FROM** Person

**GROUP BY** Email **HAVING** COUNT(Email) > 1

Table Alias

**KEY WORD**

**Technique:** Group by and count

Table: Person

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

| Column Name | Type |

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

| PersonId | int |

| FirstName | varchar |

| LastName | varchar |

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

PersonId is the primary key column for this table.

Table: Address

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

| Column Name | Type |

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

| AddressId | int |

| PersonId | int |

| City | varchar |

| State | varchar |

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

AddressId is the primary key column for this table.

Write a SQL query for a report that provides the following information for each person in the Person table, regardless if there is an address for each of those people:

FirstName, LastName, City, State

**SOLUTION**:

**SELECT** p.FirstName, p.LastName, a.City, a.State

**FROM** Person p **LEFT JOIN** Address a ON a.PersonId = p.PersonId

OR

**SELECT** p.FirstName, p.LastName, a.City, a.State

**FROM** Person p **NATURAL LEFT JOIN** Address a

**Technique: LEFT JOIN or NATURAL LEFT JOIN**

**--------------------------------------------------------------------------------------------------------------------------------**

Suppose that a website contains two tables, the Customers table and the Orders table. Write a SQL query to find all customers who never order anything.

Table: Customers.

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

| Id | Name |

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

| 1 | Joe |

| 2 | Henry |

| 3 | Sam |

| 4 | Max |

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

Table: Orders.

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

| Id | CustomerId |

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

| 1 | 3 |

| 2 | 1 |

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

Using the above tables as example, return the following:

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

| Customers |

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

| Henry |

| Max |

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

**SOLUTION**:

**SELECT** Customers.Name **AS** Customers

**FROM** Customers

**AND** Customers.Id **NOT IN** (**SELECT** CustomerId **FROM** Orders)

**Technique: Nested query**