JOINS

Eng. Lina Hammad

August 16th, 2022

Agenda

Joins

joins

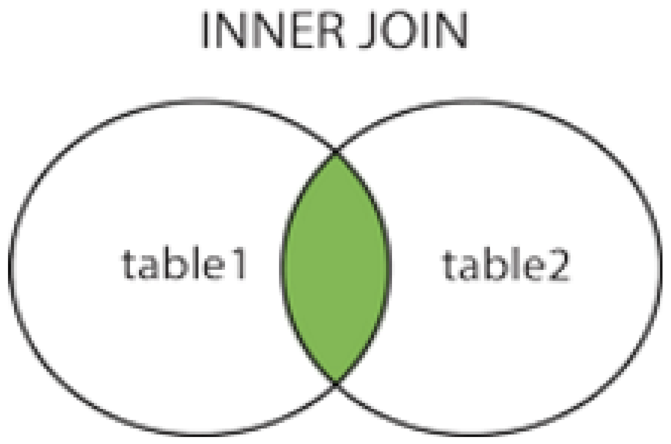
- JOIN clause is used to combine rows from two or more tables, based on a common field between them.
- type of join:
  - **Inner Join (simple join) > 99%**
  - **Left Join**
  - **Right join**
  - **Full Outer Join**
  - Left Excluding JOIN
  - Right Excluding JOIN
  - Outer Excluding JOIN
  - Self join

Inner Join (simple join)

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

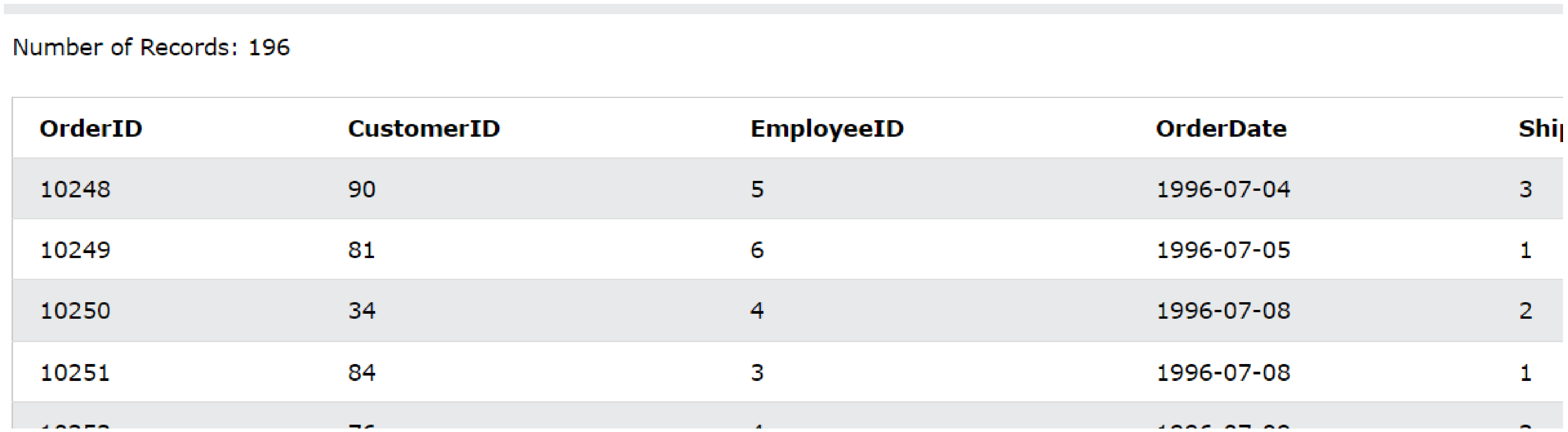
- Syntax:

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



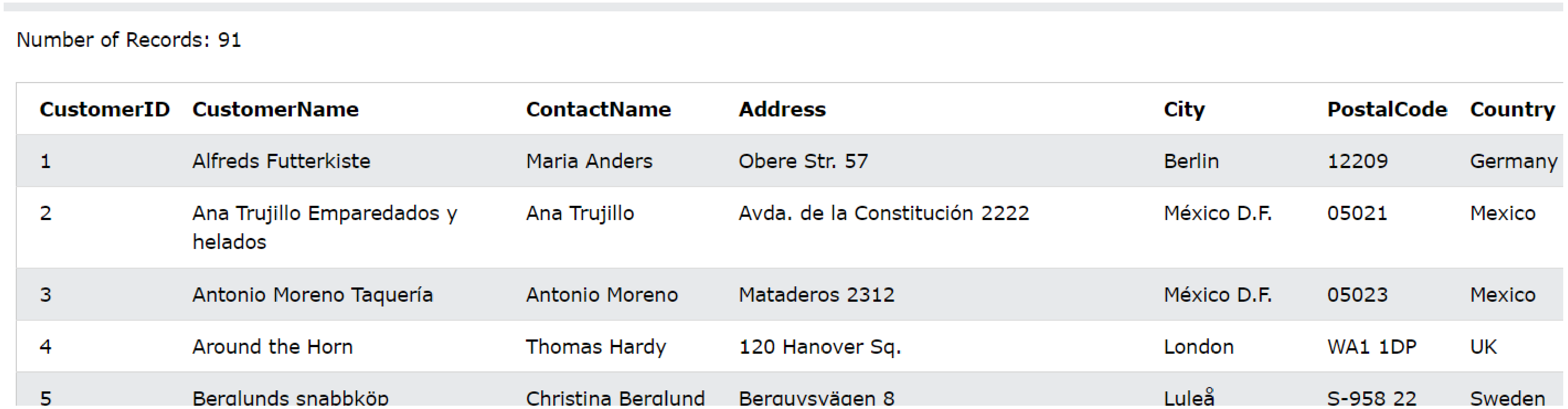
- **Note:** INNER JOIN is the same as JOIN.

Inner Join Example



Number of Records: 196

OrderID	CustomerID	EmployeeID	OrderDate	Shi
10248	90	5	1996-07-04	3
10249	81	6	1996-07-05	1
10250	34	4	1996-07-08	2
10251	84	3	1996-07-08	1

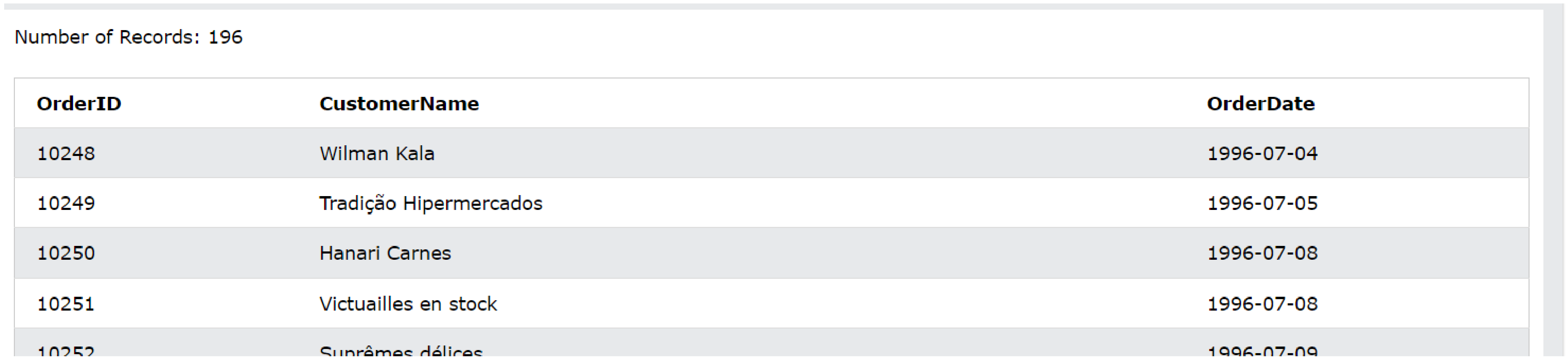


Number of Records: 91

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
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

For each oder, show the order ID, Customer Name, and order date

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



Number of Records: 196

OrderID	CustomerName	OrderDate
10248	Wilman Kala	1996-07-04
10249	Tradição Hipermercados	1996-07-05
10250	Hanari Carnes	1996-07-08
10251	Victuailles en stock	1996-07-08
10252	Suprêmes délices	1996-07-09

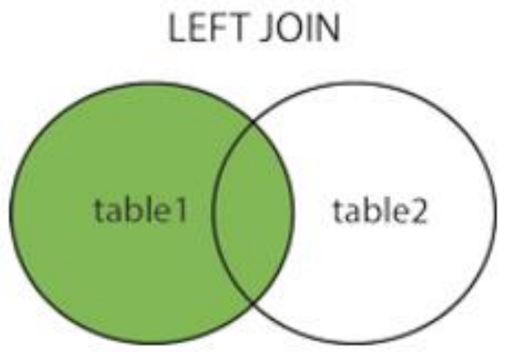
Left Join

- The LEFT JOIN keyword returns all rows from the left table (table1), with the matching rows in the right table (table2). The result is NULL in the right side when there is no match.

- Syntax:

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

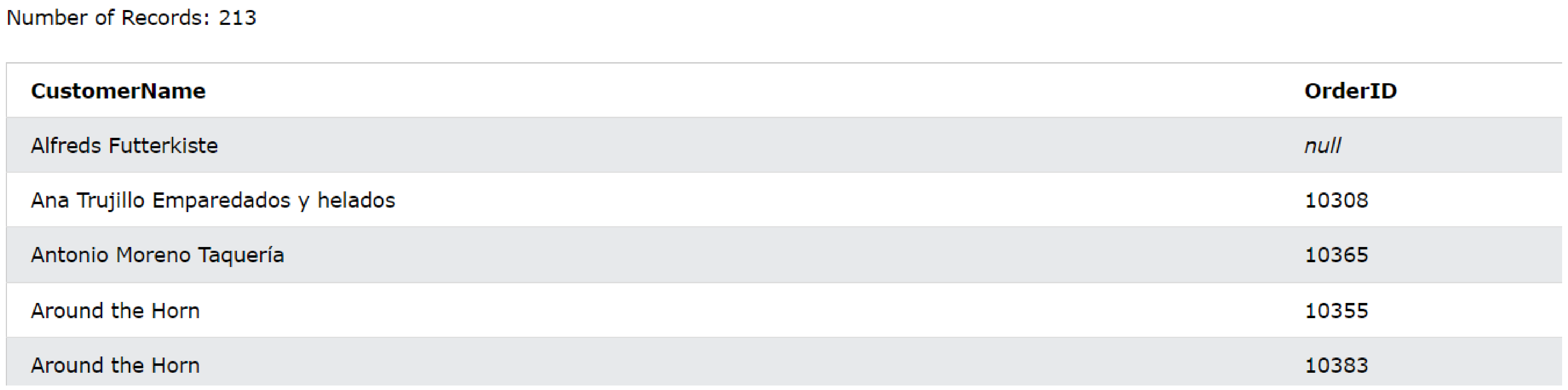
- **Note:** In some databases LEFT JOIN is called LEFT OUTER JOIN.



Left Join Example

For each customer, show customer name and orders MAY have

```sql
SELECT Customers.CustomerName, Orders.OrderID
FROM Customers
LEFT JOIN Orders ON Customers.CustomerID = Orders.CustomerID
ORDER BY Customers.CustomerName;
```



Number of Records: 213

CustomerName	OrderID
Alfreds Futterkiste	*null*
Ana Trujillo Emparedados y helados	10308
Antonio Moreno Taquería	10365
Around the Horn	10355
Around the Horn	10383

The `LEFT JOIN` keyword returns all records from the left table (Customers), even if there are no matches in the right table (Orders).

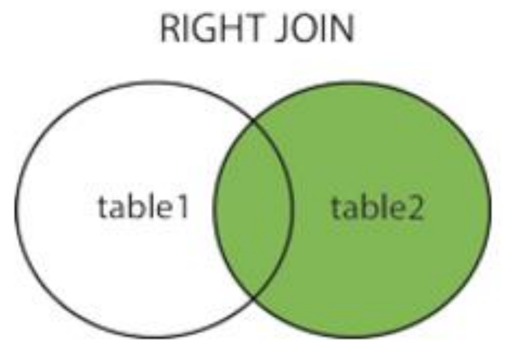
Right Join

- returns all rows from the right table (table2), with the matching rows in the left table (table1). The result is NULL in the left side when there is no match.

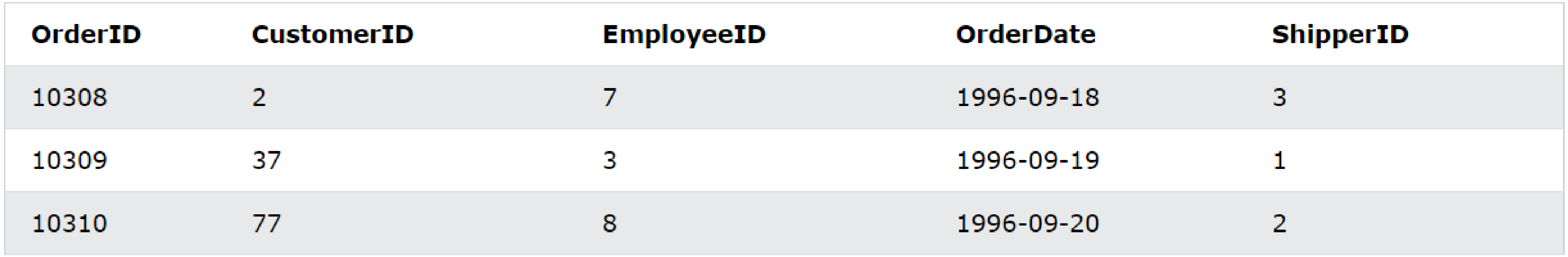
- Syntax:

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

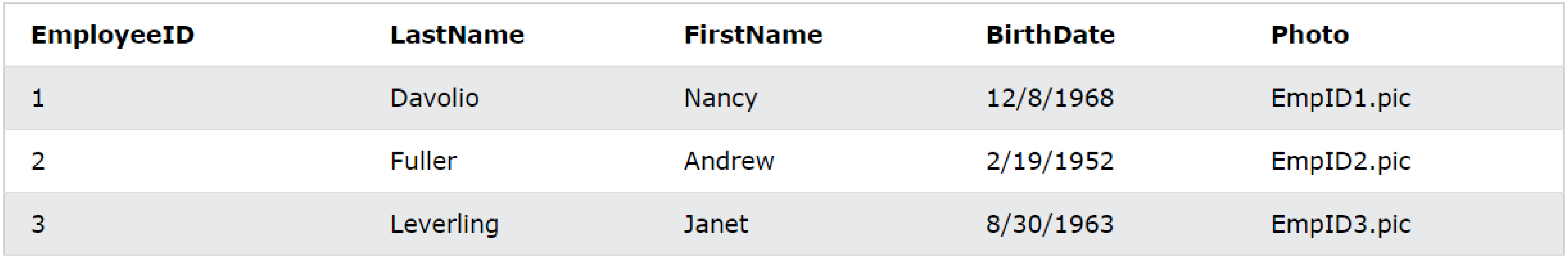
- **Note:** In some databases RIGHT JOIN is called RIGHT OUTER JOIN.



Right Join Example



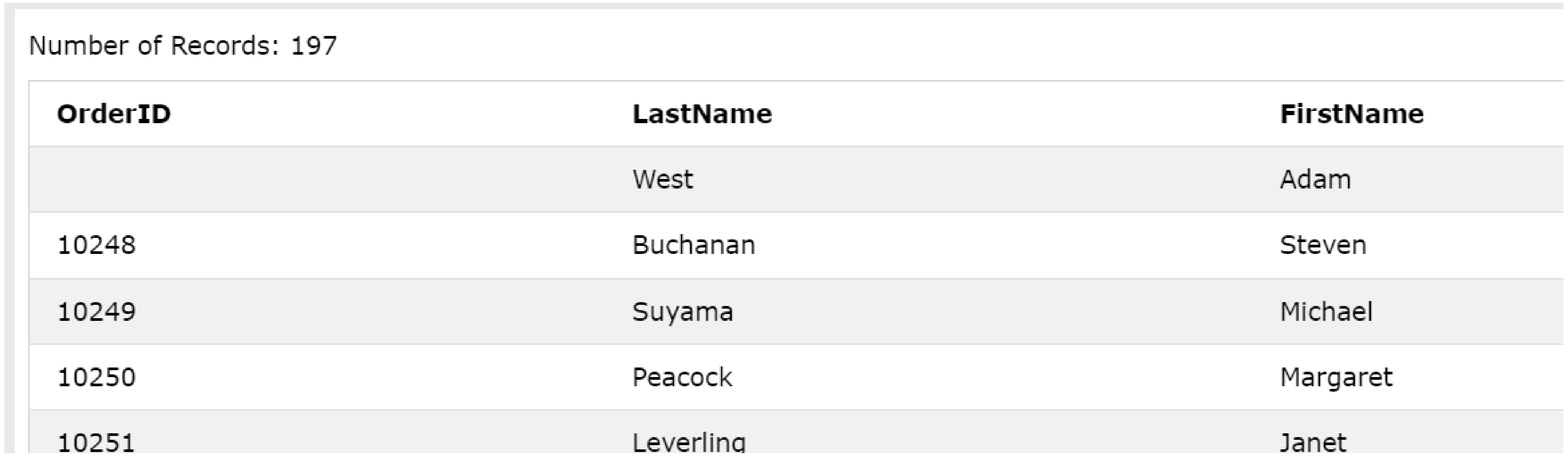
OrderID	CustomerID	EmployeeID	OrderDate	ShipperID
10308	2	7	1996-09-18	3
10309	37	3	1996-09-19	1
10310	77	8	1996-09-20	2



EmployeeID	LastName	FirstName	BirthDate	Photo
1	Davolio	Nancy	12/8/1968	EmpID1.pic
2	Fuller	Andrew	2/19/1952	EmpID2.pic
3	Leverling	Janet	8/30/1963	EmpID3.pic

Return all employees fname and lname, and any orders they might have placed

```
SELECT Orders.OrderID, Employees.LastName, Employees.FirstName
FROM Orders
RIGHT JOIN Employees ON Orders.EmployeeID = Employees.EmployeeID
ORDER BY Orders.OrderID;
```



Number of Records: 197

OrderID	LastName	FirstName
	West	Adam
10248	Buchanan	Steven
10249	Suyama	Michael
10250	Peacock	Margaret
10251	Leverling	Janet

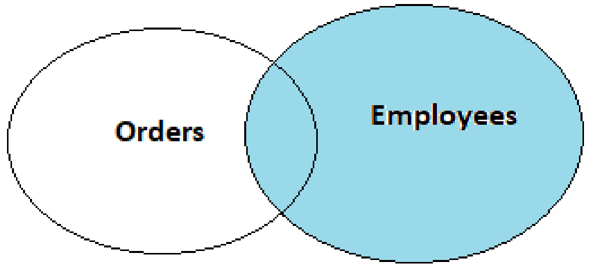
Note: The RIGHT JOIN keyword returns all records from the right table (Employees), even if there are no matches in the left table (Orders).

Two possible solutions Example

Return all employees fname and lname, and any orders they might have placed

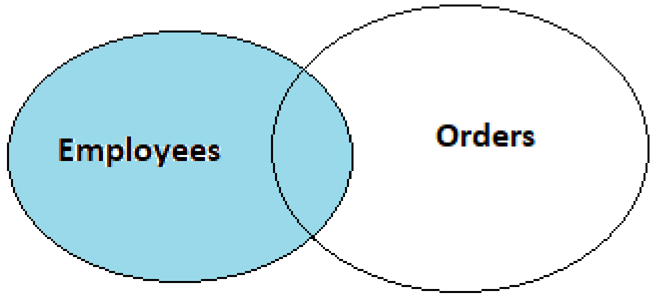
Solution 1: Right Join

```
SELECT Orders.OrderID, Employees.LastName, Employees.FirstName
FROM Orders
RIGHT JOIN Employees ON Orders.EmployeeID = Employees.EmployeeID
ORDER BY Orders.OrderID;
```



Solution 2: Left Join

```
SELECT Orders.OrderID, Employees.LastName, Employees.FirstName
FROM  Employees
LEFT JOIN  Orders  ON Orders.EmployeeID = Employees.EmployeeID
ORDER BY Orders.OrderID;
```

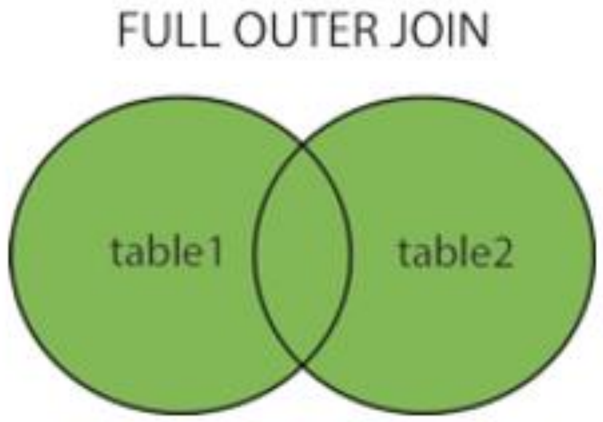


Full outer join

- The FULL OUTER JOIN keyword returns all records when there is a match in left (table1) or right (table2) table records.
- **Note**: FULL OUTER JOIN can potentially return very large result-sets!
- **Tip**: FULL OUTER JOIN and FULL JOIN are the same.

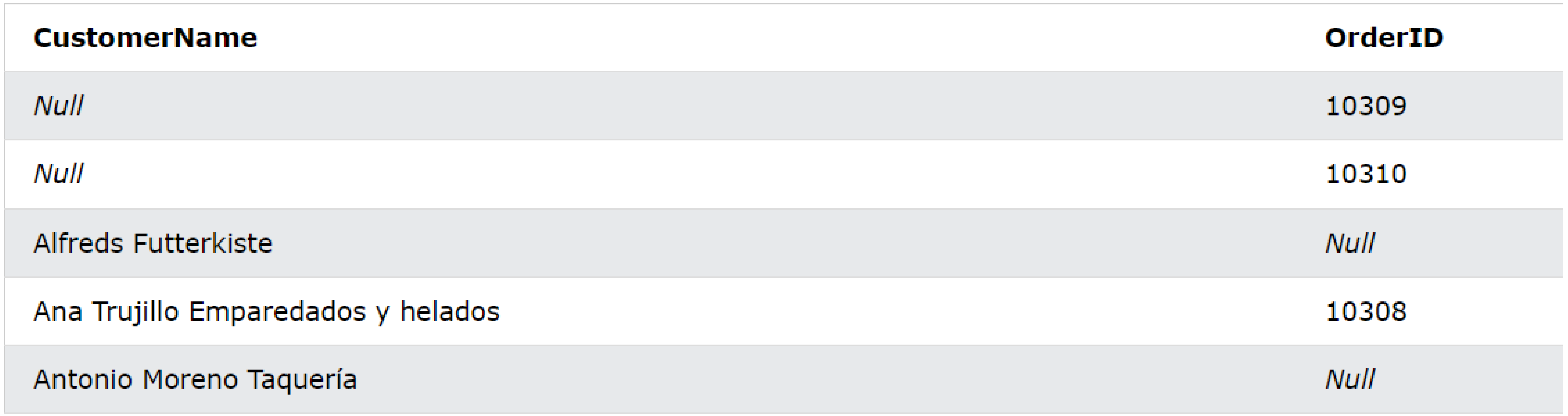
- Syntax:

```
SELECT column_name(s)
FROM table1
FULL OUTER JOIN table2
ON table1.column_name = table2.column_name
WHERE condition;
```



Full Join Example

```
SELECT Customers.CustomerName, Orders.OrderID
FROM Customers
FULL OUTER JOIN Orders ON Customers.CustomerID=Orders.CustomerID
ORDER BY Customers.CustomerName;
```



CustomerName	OrderID
Null	10309
Null	10310
Alfreds Futterkiste	*Null*
Ana Trujillo Emparedados y helados	10308
Antonio Moreno Taquería	*Null*

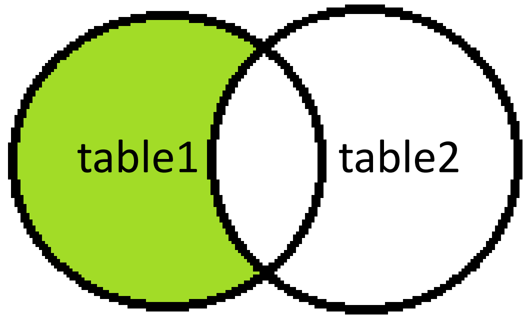
Left Excluding JOIN

- Returns all of the records in the left table (table 1) that do not match any records in the right table (table 2).

- Syntax:

```
SELECT column_name(s)
FROM table1
LEFT JOIN table2
ON table1.column_name = table2.column_name
WHERE table2.key IS NULL;
```

Left Excluding JOIN



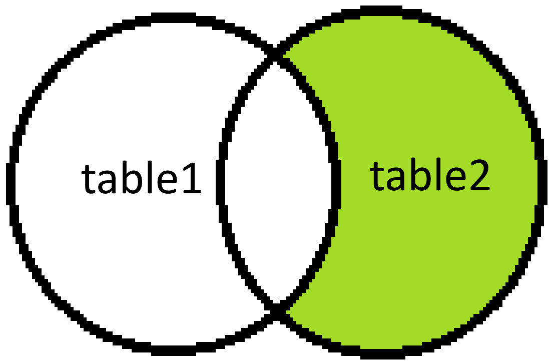
Right Excluding JOIN

- Returns all of the records in the left table (table 2) that do not match any records in the left table (table 1).

- Syntax:

```
SELECT column_name(s)
FROM table1
Right JOIN table2
ON table1.column_name = table2.column_name
WHERE table1.key IS NULL;
```

Right Excluding JOIN



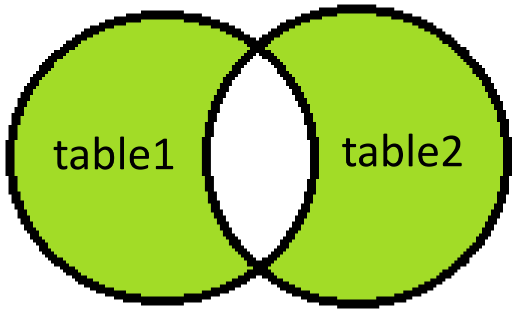
Outer Excluding JOIN

- return all of the records in the left table (table A) and all of the records in the right table (table B) that do not match. I have yet to have a need for using this type of Join, but all of the others, I use quite frequently.

- Syntax:

```
SELECT column_name(s)
FROM table1
FULL OUTER JOIN table2
ON table1.column_name = table2.column_name
WHERE table1.key IS NULL OR table2.key IS NULL;
```

Outer Excluding JOIN



SQL Aliases

- SQL aliases are used to give a table, or a column in a table, a temporary name.
- Aliases are often used to make column names more readable.
- An alias only exists for the duration of the query.

- Alias Column Syntax:

```
SELECT column_name AS alias_name
FROM table_name;
```

- Alias Table Syntax:

```
SELECT column_name(s)
FROM table_name AS alias_name;
```

Self join

- A **self join** is a join in which a table is joined with itself.
- A self join must have aliases.
- Example: when you require details about an employee and his manager (also an employee).

- Syntax

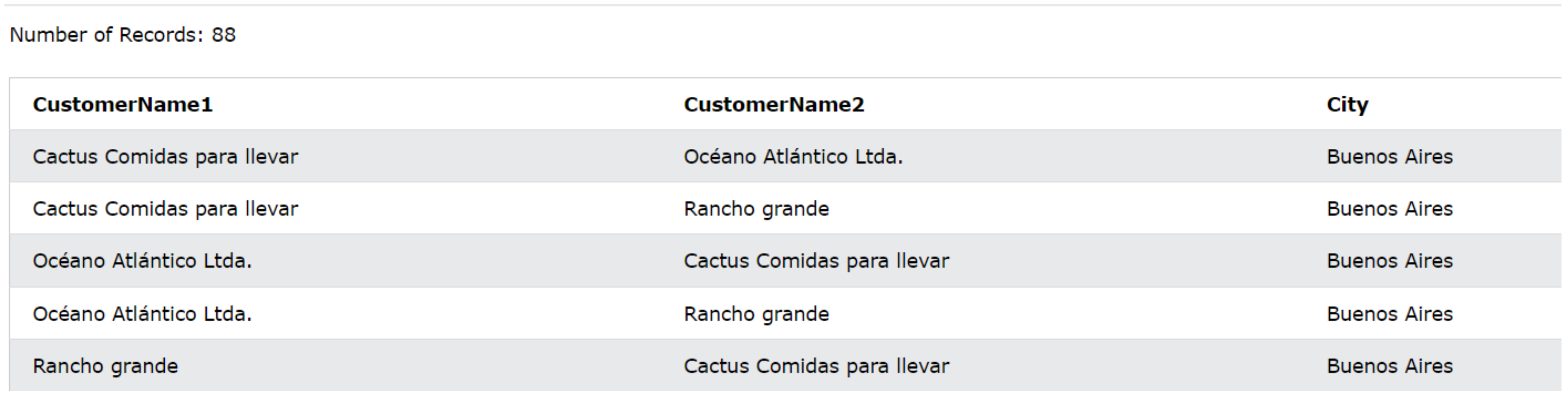
```
SELECT column_name(s)
FROM table1 AS T1, table1 AS T2
WHERE condition;
```

- **Note**: T1 and T2 are different table aliases for the same table.

Self Join Example

Show customers names that are from the same city

```sql
SELECT A.CustomerName AS CustomerName1, B.CustomerName AS CustomerName2, A.City
FROM Customers AS  A, Customers AS B
WHERE A.CustomerID <> B.CustomerID
AND A.City = B.City
ORDER BY A.City;
```



Number of Records: 88

CustomerName1	CustomerName2	City
Cactus Comidas para llevar	Océano Atlántico Ltda.	Buenos Aires
Cactus Comidas para llevar	Rancho grande	Buenos Aires
Océano Atlántico Ltda.	Cactus Comidas para llevar	Buenos Aires
Océano Atlántico Ltda.	Rancho grande	Buenos Aires
Rancho grande	Cactus Comidas para llevar	Buenos Aires

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.

- **UNION Syntax**:

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

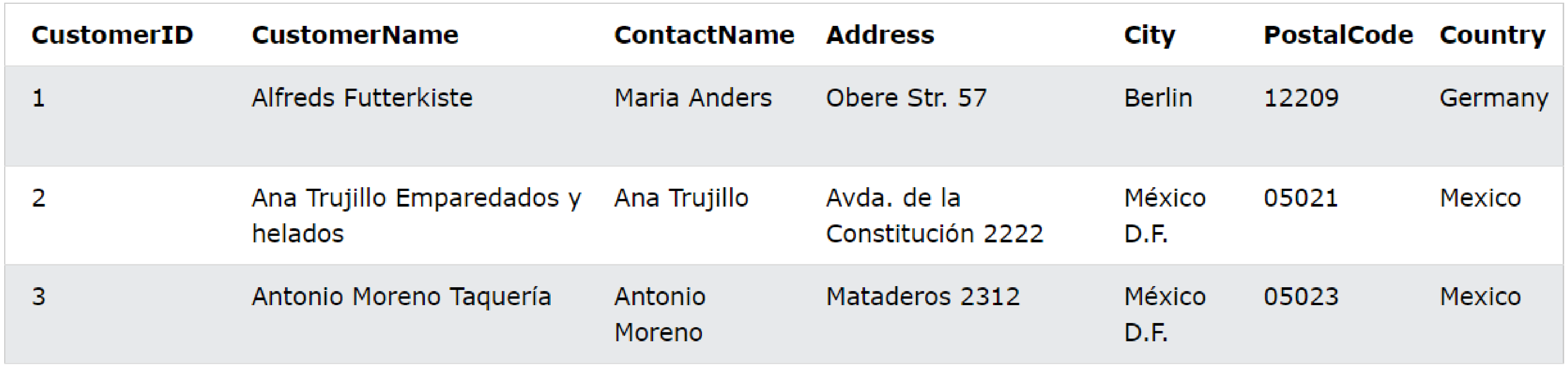
- The UNION operator selects only distinct values by default. To allow duplicate values, use UNION ALL:

- **UNION ALL Syntax**:

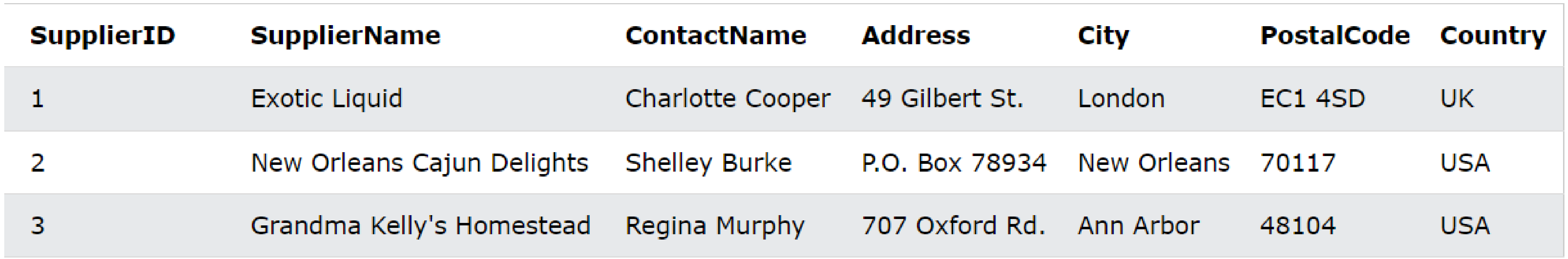
```
SELECT column_name(s) FROM table1
UNION ALL
SELECT column_name(s) FROM table2;
```

- **Note**: The column names in the result-set are usually equal to the column names in the first SELECT statement in the UNION.

Union Example



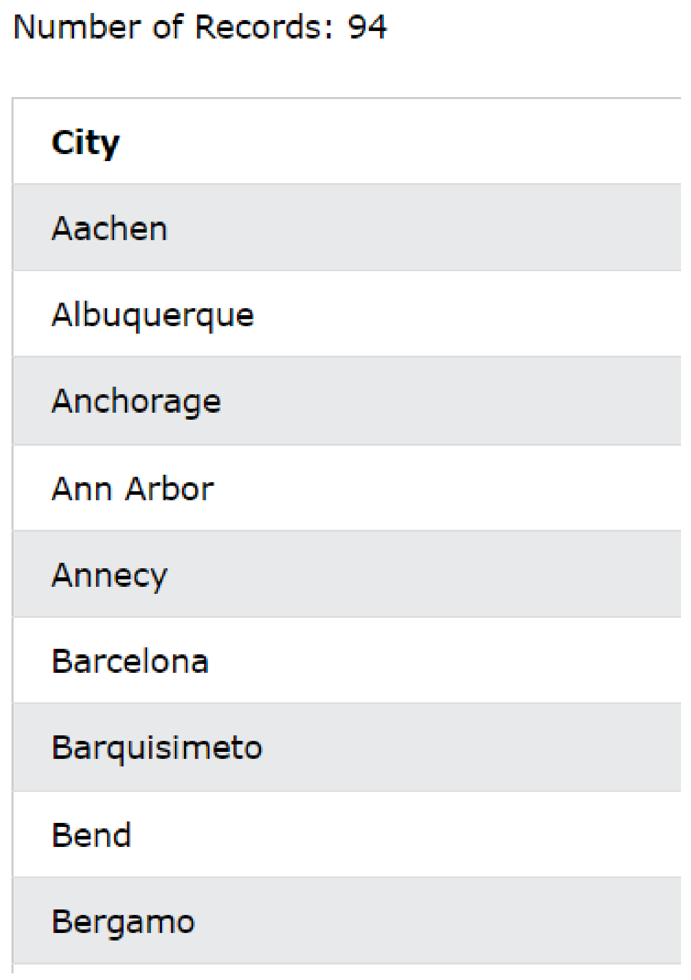
CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México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



SupplierID	SupplierName	ContactName	Address	City	PostalCode	Country
1	Exotic Liquid	Charlotte Cooper	49 Gilbert St.	London	EC1 4SD	UK
2	New Orleans Cajun Delights	Shelley Burke	P.O. Box 78934	New Orleans	70117	USA
3	Grandma Kelly's Homestead	Regina Murphy	707 Oxford Rd.	Ann Arbor	48104	USA

Show the cities (only distinct values) from both the "Customers" and the "Suppliers" table

```
SELECT City FROM Customers
UNION
SELECT City FROM Suppliers
ORDER BY City;
```



Number of Records: 94

City
Aachen
Albuquerque
Anchorage
Ann Arbor
Annecy
Barcelona
Barquisimeto
Bend
Bergamo