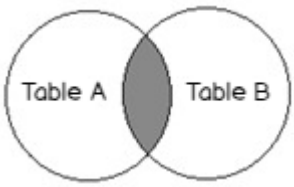
SQL multiple joins for beginners with examples

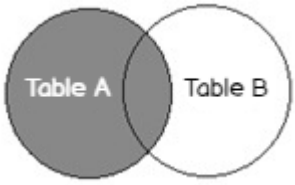
October 16, 2019 by Esat Erkec

In this article, we will learn the SQL multiple joins concept and reinforce our learnings with pretty simple examples, which are explained with illustrations. In relational databases, data is stored in tables. Without a doubt, and most of the time, we need a result set that is formed combining data from several tables. The joins allow us to combine data from two or more tables so that we are able to join data of the tables so that we can easily retrieve data from multiple tables. You might ask yourself how many different types of join exist in SQL Server. The answer is there are four main types of joins that exist in SQL Server. First of all, we will briefly describe them using Venn diagram illustrations:

- **Inner join** returns the rows that match in both tables

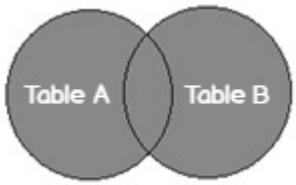


- **Left join** returns all rows from the left table



- **Right join** returns all rows from the right table

- **Full join** returns whole rows from both tables



If you lack knowledge about the SQL join concept in the SQL Server, you can see the SQL Join types overview and tutorial article.

After this short explanatory about the SQL joins types, we will go through the multiple joins.

What are SQL multiple joins?

Multiple joins can be described as follows; multiple join is a query that contains the same or different join types, which are used more than once**.** Thus, we gain the ability to combine multiple tables of data in order to overcome relational database issues.

Example scenario

Green-Tree company launched a new campaign for the New Year and made different offers to its on-line customers. As a result of their campaign, they succeeded in converting some offers to sales. In the following examples, we will uncover the new year campaign data details of the Green-Tree company.

The company stores these campaign data details in the following tables. Now, we will create these tables through the following query and populate them with some dummy data:

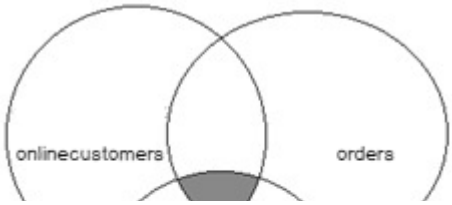
```
DROP TABLE IF EXISTS sales
GO
DROP TABLE IF EXISTS orders
GO
DROP TABLE IF EXISTS onlinecustomers
GO
CREATE TABLE onlinecustomers (customerid INT PRIMARY KEY IDENTITY(1,1) ,Customer-
Name VARCHAR(100)
,CustomerCity VARCHAR(100) ,Customermail VARCHAR(100))
GO
CREATE TABLE orders (orderId INT PRIMARY KEY IDENTITY(1,1) , customerid INT  ,
ordertotal float ,discountrate float ,orderdate DATETIME)
GO
CREATE TABLE sales (salesId INT PRIMARY KEY IDENTITY(1,1) ,
orderId INT  ,
salestotal FLOAT)
```

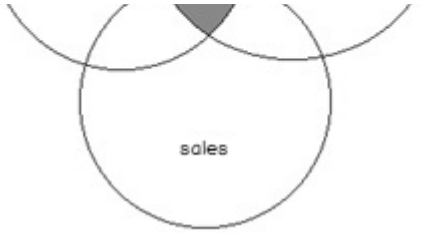
```
GO


INSERT INTO [dbo].[onlinecustomers]([CustomerName],[CustomerCity],[Customermail])
VALUES (N'Salvador',N'Philadelphia',N'tyiptqo.wethls@chttw.org')
INSERT INTO [dbo].[onlinecustomers]([CustomerName],[CustomerCity],[Customermail])
VALUES (N'Gilbert',N'San Diego',N'rrvyy.wdumos@lklkj.org')
INSERT INTO [dbo].[onlinecustomers]([CustomerName],[CustomerCity],[Customermail])
VALUES (N'Ernest',N'New York',N'ymuea.pnxkukf@dwv.org')
INSERT INTO [dbo].[onlinecustomers]([CustomerName],[CustomerCity],[Customermail])
VALUES (N'Stella',N'Phoenix',N'xvsfzp.rjhtni@rdn.com')
INSERT INTO [dbo].[onlinecustomers]([CustomerName],[CustomerCity],[Customermail])
VALUES (N'Jorge',N'Los Angeles',N'oykbo.vlxopp@nmwhv.org')
INSERT INTO [dbo].[onlinecustomers]([CustomerName],[CustomerCity],[Customermail])
VALUES (N'Jerome',N'San Antonio',N'wkabc.ofmhetq@gtmh.co')
INSERT INTO [dbo].[onlinecustomers]([CustomerName],[CustomerCity],[Customermail])
VALUES (N'Edward',N'Chicago',N'wguexiymy.nnbdgpc@juc.co')

GO

INSERT INTO [dbo].[orders]([customerid],[ordertotal],[discountrate],[orderdate]) V
ALUES (3,1910.64,5.49,CAST('03-Dec-2019' AS DATETIME))
INSERT INTO [dbo].[orders]([customerid],[ordertotal],[discountrate],[orderdate]) V
ALUES (4,150.89,15.33,CAST('11-Jun-2019' AS DATETIME))
INSERT INTO [dbo].[orders]([customerid],[ordertotal],[discountrate],[orderdate]) V
ALUES (5,912.55,13.74,CAST('15-Sep-2019' AS DATETIME))
INSERT INTO [dbo].[orders]([customerid],[ordertotal],[discountrate],[orderdate]) V
ALUES (7,418.24,14.53,CAST('28-May-2019' AS DATETIME))
INSERT INTO [dbo].[orders]([customerid],[ordertotal],[discountrate],[orderdate]) V
ALUES (55,512.55,13.74,CAST('15-Jun-2019' AS DATETIME))
INSERT INTO [dbo].[orders]([customerid],[ordertotal],[discountrate],[orderdate]) V
ALUES (57,118.24,14.53,CAST('28-Dec-2019' AS DATETIME))
GO

INSERT INTO [dbo].[sales]([orderId],[salestotal]) VALUES (3,370.95)
INSERT INTO [dbo].[sales]([orderId],[salestotal]) VALUES (4,882.13)
INSERT INTO [dbo].[sales]([orderId],[salestotal]) VALUES (12,370.95)
INSERT INTO [dbo].[sales]([orderId],[salestotal]) VALUES (13,882.13)
INSERT INTO [dbo].[sales]([orderId],[salestotal]) VALUES (55,170.95)
INSERT INTO [dbo].[sales]([orderId],[salestotal]) VALUES (57,382.13)
```

How SQL multiple joins work?

Business problem: Which customers were interested in this New Year campaign?

In order to answer this question, we need to find out the matched rows for all the tables because some customers did not receive an email offer, and some offers could not be converted into a sale. The following Venn diagram will help us to figure out the matched rows which we need. In short, the result of this query should be the intersecting rows of all tables in the query. The grey-colored area specifies these rows in the Venn diagram:





The SQL multiple joins approach will help us to join **onlinecustomers**, **orders,** and **sales** tables. As shown in the Venn diagram, we need to matched rows of all tables. For this reason, we will combine all tables with an **inner join** clause. The following query will return a result set that is desired from us and will answer the question:

```
SELECT customerName, customercity, customermail, salestotal
FROM onlinecustomers AS oc
   INNER JOIN
   orders AS o
   ON oc.customerid = o.customerid
   INNER JOIN
   sales AS s
   ON o.orderId = s.orderId
```

At first, we will analyze the query. An inner join clause that is between **onlinecustomers** and **orders** tables derived the matched rows between these two tables. The second inner join clause that combines the **sales** table derived the matched rows from the previous result set. The following colored tables illustration will help us to understand the joined tables data matching in the query. The yellow-colored rows specify matched data between **onlinecustomers** and **orders.** On the other hand, only the blue colored rows exist in the **sales** tables so the query result will be blue colored rows:

customerid	CustomerName	CustomerCity	Customermail
1	Salvador	Philadelphia	tyiptqo.wethls@chttw.org
2	Gilbert	San Diego	rrvyy.wdumos@lklkj.org
3	Ernest	New York	ymuea.pnxkukf@dwv.org
4	Stella	Phoenix	xvsfzp.rjhtni@rdn.com
5	Jorge	Los Angeles	oykbo.vlxopp@nmwhv.org
6	Jerome	San Antonio	wkabc.ofmhetq@gtmh.co
7	Edward	Chicago	wguexiymy.nnbdgpc@juc.co

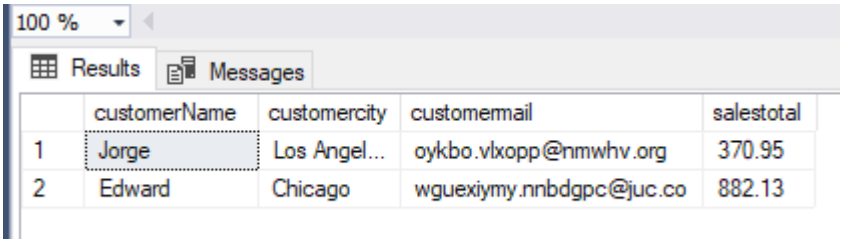
onlinecustomers AS oc
INNER JOIN
orders AS o
ON oc.customerid = o.customerid

orderId	customerid	ordertotal	discountrate	orderdate
1	3	1910.64	5.49	2019-12-03 00:00:00.000
2	4	150.89	15.33	2019-06-11 00:00:00.000
3	5	912.55	13.74	2019-09-15 00:00:00.000
4	7	418.24	14.53	2019-05-28 00:00:00.000
5	55	512.55	13.74	2019-06-15 00:00:00.000
6	57	118.24	14.53	2019-12-28 00:00:00.000

INNER JOIN
sales AS s
ON o.orderId = s.orderId

salesId	orderId	salestotal
1	3	370.95
2	4	882.13
3	12	370.95
4	13	882.13
5	55	170.95
6	57	382.13

The result of the query will look like this:



Different join types usage in SQL multiple joins

Business problem: Which offers could not be converted into a sell?

We can use the different types of joins in a single query so that we can overcome different relational database issues. In this example, we need all rows of the **orders** table, which are matched to **onlinecustomers** tables. On the other hand, these rows do not exist in the **sales** table. The following Venn diagram will help us to figure out the matched rows which we need. The grey-colored area indicates rows which will be the output of the query:

In the first step, we should combine the **onlinecustomers** and **orders** tables through the inner join clause because inner join returns all the matched rows between **onlinecustomers** and **orders** tables. In the second step, we will combine the **orders** table to the **sales** table through the left join and then filter the null values because we need to eliminate the rows which are stored by the **sales** table:

```
SELECT customerName, customercity, customermail, ordertotal,salestotal
FROM onlinecustomers AS c
   INNER JOIN
   orders AS o
   ON c.customerid = o.customerid
   LEFT JOIN
   sales AS s
   ON o.orderId = s.orderId
   WHERE s.salesId IS NULL
```

The result of the query will look like this:

Results Messages

	customerName	customercity	customermail	ordertotal	salestotal
1	Ernest	New York	ymuea.pnxkukf@dwv.org	1910.64	NULL
2	Stella	Phoenix	xvsfzp.rjhtni@rdn.com	150.89	NULL

Quiz

Question: Please generate the proper query according to the below Venn diagram.

Answer: As we learned, the full join allows us to return all rows from the combined tables. The answered query will be like the following:

```
SELECT customerName, customercity, customermail, ordertotal,salestotal
FROM onlinecustomers AS c
   FULL JOIN
   orders AS o
   ON c.customerid = o.customerid
   FULL JOIN
   sales AS s
   ON o.orderId = s.orderId
```

Results Messages

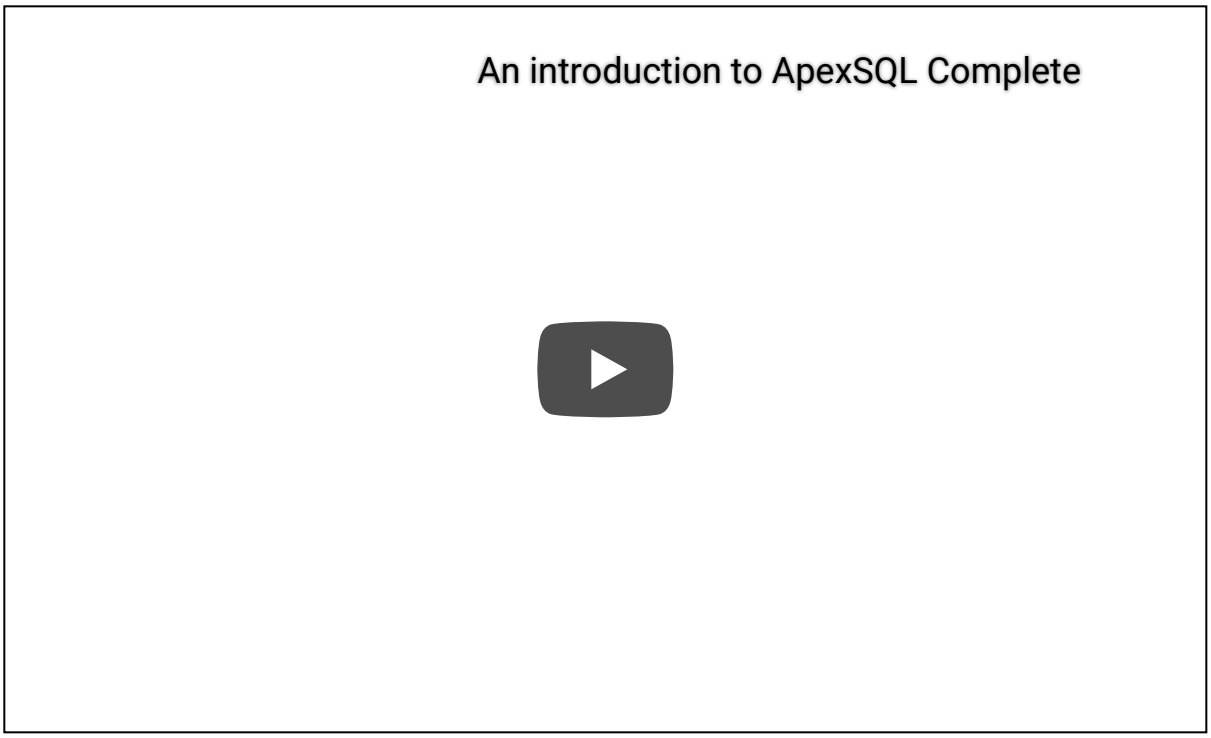
	customerName	customercity	customermail	ordertotal	salestotal
1	Salvador	Philadelphia	tyiptqo.wethls@chttw.org	NULL	NULL
2	Gilbert	San Diego	rrvyy.wdumos@lkllkj.org	NULL	NULL
3	Ernest	New York	ymuea.pnxkukf@dwv.org	1910.64	NULL
4	Stella	Phoenix	xvsfzp.rjhtni@rdn.com	150.89	NULL
5	Jorge	Los Angeles	oykbo.vlxopp@nmwhv.org	912.55	370.95
6	Jerome	San Antonio	wkabc.ofmhetq@gtmh.co	NULL	NULL
7	Edward	Chicago	wguexiymy.nnbdgpc@juc.co	418.24	882.13
8	NULL	NULL	NULL	512.55	NULL
9	NULL	NULL	NULL	118.24	NULL
10	NULL	NULL	NULL	NULL	370.95
11	NULL	NULL	NULL	NULL	170.95
12	NULL	NULL	NULL	NULL	882.13
13	NULL	NULL	NULL	NULL	382.13

Conclusion

In this article, we focused on the SQL multiple joins approach and learned it with detailed examples. Multiple joins allow us to combine more than two tables so that we can overcome different issues in the relational database system. Furthermore, we saw how we could use different join types in a single query.

See more

ApexSQL Complete is a SQL code complete tool that includes features like code snippets, SQL auto-replacements, tab navigation, saved queries and more for SSMS and Visual Studio



Esat Erkec

Esat Erkec is a SQL Server professional who began his career 8+ years ago as a Software Developer. He is a SQL Server Microsoft Certified Solutions Expert.

Most of his career has been focused on SQL Server Database Administration and Development. His current interests are in database administration and Business Intelligence. You can find him on LinkedIn.

View all posts by Esat Erkec

Related Posts:

1. **SQL PARTITION BY Clause overview**
2. **Descripción general de la cláusula PARTITION BY de SQL**
3. **An overview of the SQL Server Update Join**
4. **Term Extraction Transformation in SSIS**
5. **Implementing a Modular ETL in SSIS**

Development, SQL commands, T-SQL

244,812 Views

ALSO ON **SQL SHACK**

5 months ago • 1 comment
Test-driven database hotfix development …

4 months ago • 2 comments
How to prepare for the Exam DP-201: …

a month ag
Best a award

0 Comments **SQL Shack** 1 **Login**

Recommend 9 Tweet Share **Sort by Best**

Start the discussion…

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

Be the first to comment.