Joins and subqueries are both used to combine data from different tables into a single result. They share many similarities and differences. Subqueries can be used to return either a scalar (single) value or a row set; whereas, joins are used to **return** rows.

**Joins are advantageous over subqueries if the SELECT list in a query contains columns from more than one table. Example 2 shows this.**

**Subqueries are advantageous over joins when you have to calculate an aggregate value on-the-fly and use it in the outer query for comparison. Example 1 shows this.**

### When to Use a Subquery vs. a JOIN

We’ve reviewed some common uses of subqueries and the situations in which some subqueries might be rewritten with JOINs instead.

A JOIN is more efficient in most cases, but there are cases in which constructs other than a subquery is not possible.

While subqueries may be more readable for beginners,

JOINs are more readable for experienced SQL coders as the queries become more complex.

It is a good practice to avoid multiple levels of nested subqueries, since they are not easily readable and do not have good performance

. In general, it is better to write a query with JOINs rather than with subqueries if possible, especially if the subqueries are correlated.

What is subquery in SQL?

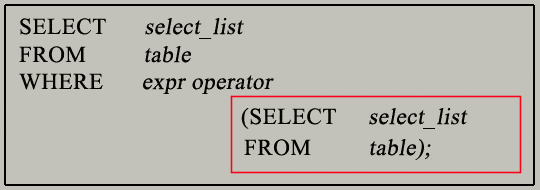
A subquery is a SQL query nested inside a larger query.

* A subquery may occur in :
  + - A SELECT clause
  + - A FROM clause
  + - A WHERE clause
* The subquery can be nested inside a SELECT, INSERT, UPDATE, or DELETE statement or inside another subquery.
* A subquery is usually added within the WHERE Clause of another SQL SELECT statement.
* You can use the comparison operators, such as >, <, or =. The comparison operator can also be a multiple-row operator, such as IN, ANY, or ALL.
* A subquery is also called an inner query or inner select, while the statement containing a subquery is also called an outer query or outer select.
* The inner query executes first before its parent query so that the results of an inner query can be passed to the outer query.

You can use a subquery in a SELECT, INSERT, DELETE, or UPDATE statement to perform the following tasks:

* Compare an expression to the result of the query.
* Determine if an expression is included in the results of the query.
* Check whether the query selects any rows.

**Syntax :**



* **The subquery (inner query) executes once before the main query (outer query) executes.**
* **The main query (outer query) use the subquery result.**
* ***IN* - Equal to anything in the list**
* ***ANY* - Compares value to *each* value returned by the sub query.**
* ***ALL* - Compares value to *every* value returned by the sub query.**
* **Using the > comparison operator as an example, >ALL means greater than every value--in other words, greater than the maximum value. For example, >ALL (1, 2, 3) means greater than 3.**
* **>ANY means greater than at least one value, that is, greater than the minimum. So >ANY (1, 2, 3) means greater than 1.**
* **For example:**
* **SELECT \***
* **FROM employee**
* **WHERE salary > ANY (2000, 3000, 4000);**

## ANY:

## Similarly, >ANY means that for a row to satisfy the condition specified in the outer query, the value in the column that introduces the subquery must be greater than at least one of the values in the list of values returned by the subquery.

< ANY means less than the maximum value in the list.

Get the details of all employees who are earning less than the highest earning manager?

Select Empno, Ename, Job, Sal

From Emp

Where Sal < Any (Select Distinct MGR

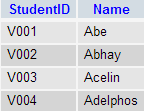
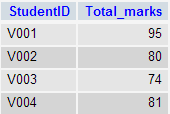
From Emp);

> ANY means more than the minimum value in the list.

* **While using all**
* SELECT empno, sal FROM emp WHERE sal > ALL (2000, 3000, 4000);
* EMPNO SAL
* 7839 5000
* It will return result equivalent to query:
* SELECT empno, sal FROM emp WHERE sal > 2000 AND sal > 3000 AND sal > 4000;
* **While using any**
* SELECT empno, sal FROM emp WHERE sal > ANY (2000, 3000, 4000);
* EMPNO SAL
* 7566 2975
* 7698 2850
* 7782 2450
* 7788 3000
* 7839 5000
* 7902 3000
* Returns a result same as
* SELECT empno, sal FROM emp WHERE sal > 2000 OR sal > 3000 OR sal > 4000;

SQL Subqueries Example :

In this section, you will learn the requirements of using subqueries. We have the following two tables 'student' and 'marks' with common field 'StudentID'.

  
           
            student                                        marks

Now we want to write a query to identify all students who get better marks than that of the student who's StudentID is 'V002', but we do not know the marks of 'V002'.  
- To solve the problem, we require two queries. One query returns the marks (stored in Total\_marks field) of 'V002' and a second query identifies the students who get better marks than the result of the first query.

**First query:**

SELECT \*

FROM `marks`

WHERE studentid = 'V002';

Copy

**Query result:**

student query

The result of the query is 80.  
- Using the result of this query, here we have written another query to identify the students who get better marks than 80. Here is the query :

**Second query:**

SELECT a.studentid, a.name, b.total\_marks

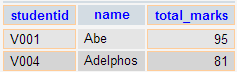
FROM student a, marks b

WHERE a.studentid = b.studentid

AND b.total\_marks >80;

Copy

**Query result:**



Above two queries identified students who get the better number than the student who's StudentID is 'V002' (Abhay).

You can combine the above two queries by placing one query inside the other. The subquery (also called the 'inner query') is the query inside the parentheses. See the following code and query result :

**SQL Code:**

SELECT a.studentid, a.name, b.total\_marks

FROM student a, marks b

WHERE a.studentid = b.studentid AND b.total\_marks >

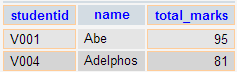
(SELECT total\_marks

FROM marks

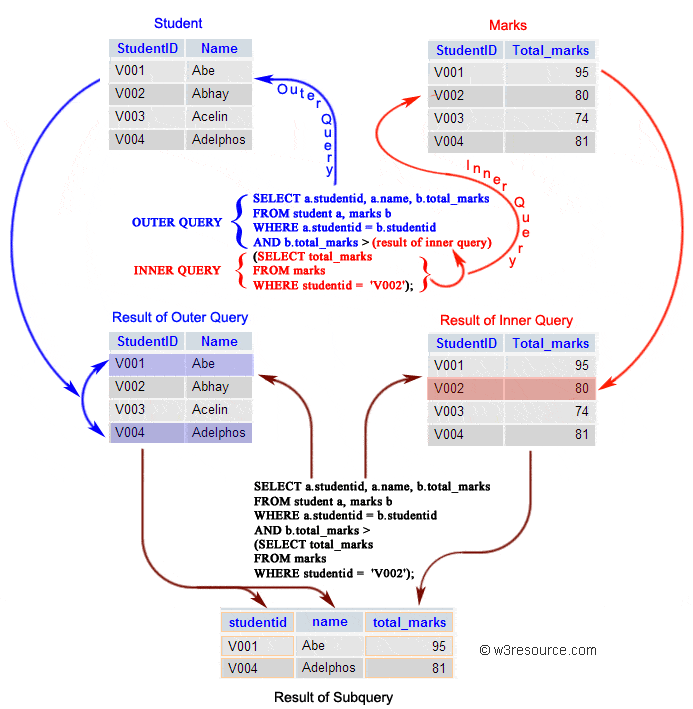
WHERE studentid = 'V002');

Copy

**Query result:**



**Pictorial Presentation of SQL Subquery:**

[](https://www.w3resource.com/sql/subqueries/sql-subqueries.gif)

Subqueries: General Rules

A subquery SELECT statement is almost similar to the SELECT statement and it is used to begin a regular or outer query. Here is the syntax of a subquery:

**Syntax:**

(SELECT [DISTINCT] subquery\_select\_argument

FROM {table\_name | view\_name}

{table\_name | view\_name} ...

[WHERE search\_conditions]

[GROUP BY aggregate\_expression [, aggregate\_expression] ...]

[HAVING search\_conditions])

Subqueries: Guidelines

There are some guidelines to consider when using subqueries :

* A subquery must be enclosed in parentheses.
* A subquery must be placed on the right side of the comparison operator.
* Subqueries cannot manipulate their results internally, therefore ORDER BY clause cannot be added into a subquery. You can use an ORDER BY clause in the main SELECT statement (outer query) which will be the last clause.
* Use single-row operators with single-row subqueries.
* If a subquery (inner query) returns a null value to the outer query, the outer query will not return any rows when using certain comparison operators in a WHERE clause.

Type of Subqueries

* Single row subquery : Returns zero or one row.
* Multiple row subquery : Returns one or more rows.
* Multiple column subqueries : Returns one or more columns.
* Correlated subqueries : Reference one or more columns in the outer SQL statement. The subquery is known as a correlated subquery because the subquery is related to the outer SQL statement.
* Nested subqueries : Subqueries are placed within another subquery.

In the next session, we have thoroughly discussed the above topics. Apart from the above type of subqueries, you can use a subquery inside INSERT, UPDATE and DELETE statement. Here is a brief discussion :

Subqueries with INSERT statement

INSERT statement can be used with subqueries. Here are the syntax and an example of subqueries using INSERT statement.

**Syntax:**

INSERT INTO table\_name [ (column1 [, column2 ]) ]

SELECT [ \*|column1 [, column2 ]

FROM table1 [, table2 ]

[ WHERE VALUE OPERATOR ];

If we want to insert those orders from 'orders' table which have the advance\_amount 2000 or 5000 into 'neworder' table the following SQL can be used:

Sample table: orders

**SQL Code:**

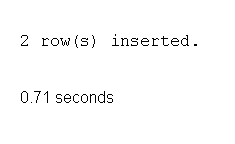
INSERT INTO neworder

SELECT \* FROM orders

WHERE advance\_amount in(2000,5000);

Copy

Output:



To see more details of subqueries using INSERT statement [click here](https://www.w3resource.com/sql/insert-statement/insert-using-subqueries.php).

Subqueries with UPDATE statement

In a UPDATE statement, you can set new column value equal to the result returned by a single row subquery. Here are the syntax and an example of subqueries using UPDATE statement.

**Syntax:**

UPDATE table SET column\_name = new\_value

[ WHERE OPERATOR [ VALUE ]

(SELECT COLUMN\_NAME

FROM TABLE\_NAME)

[ WHERE) ]

If we want to update that ord\_date in 'neworder' table with '15-JAN-10' which have the difference of ord\_amount and advance\_amount is less than the minimum ord\_amount of 'orders' table the following SQL can be used:

Sample table: neworder

**SQL Code:**

UPDATE neworder

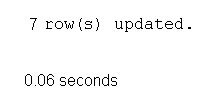
SET ord\_date='15-JAN-10'

WHERE ord\_amount-advance\_amount<

(SELECT MIN(ord\_amount) FROM orders);

Copy

Output:



To see more details of subqueries using UPDATE statement [click here](https://www.w3resource.com/sql/update-statement/update-using-subqueries.php).

Subqueries with DELETE statement

DELETE statement can be used with subqueries. Here are the syntax and an example of subqueries using DELETE statement.

**Syntax:**

DELETE FROM TABLE\_NAME

[ WHERE OPERATOR [ VALUE ]

(SELECT COLUMN\_NAME

FROM TABLE\_NAME)

[ WHERE) ]

If we want to delete those orders from 'neworder' table which advance\_amount are less than the maximum advance\_amount of 'orders' table, the following SQL can be used:

Sample table: neworder

**SQL Code:**

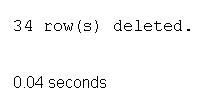
DELETE FROM neworder

WHERE advance\_amount<

(SELECT MAX(advance\_amount) FROM orders);

Copy

Output:



# SQL Delete records using subqueries

## SQL Deleting records with subqueries

, we are going to discuss, how SUBQUERIES (A SELECT statement within another SELECT statement can be used as a subquery )along with SQL DELETE command can be used to perform a deletion.

Sample tables associated with this page have shown bellow:

**Sample table: customer1**

**Sample table: agents**

**Sample table: customer**

**Sample table: agent1**

**Sample table: orders**

**Example:**

To remove rows from the table 'customer1' with following conditions -

**1.** 'agent\_code' should be any 'agent\_code' from 'agents' table which satisfies the condition bellow :

**2.** 'working\_area' of 'agents' table must be 'London',

the following SQL statement can be used:

**SQL Code:**

DELETE FROM customer1

WHERE agent\_code=ANY(

SELECT agent\_code FROM agents

WHERE working\_area='London');

Copy

Output:



## SQL delete records using subqueries with alias

In this page, we are going to discuss, how table aliases( when two or more tables used in a query, then alias makes it easy to read and write with a short name which comes after the table name after the FROM keyword) can be used with SUBQUERIES (A SELECT statement within another SELECT statement can be used as a subquery ), and with the help of subqueries SQL DELETE command can be used to delete records.

**Example:**

To remove rows from the table 'agent1' with following conditions -

**1.** **'da'** and **'cu'** are the aliases for the table 'agent1' and 'customer'

**2.** check the existence of the subquery is true or false. which satisfies the condition bellow :

**3.** 'grade' of 'customer' table must be 3,

**4.** 'agent\_code' of 'agent1' table and 'agent\_code' of 'customer' table should not be same,

the following SQL statement can be used:

**SQL Code:**

DELETE FROM agent1 da

WHERE EXISTS(

SELECT \* FROM customer cu

WHERE grade=3

AND da.agent\_code<>cu.agent\_code);

Copy

Output:



## SQL delete records using subqueries with alias and IN

In this page we are going to discuss, how rows can be removed from a table by SQL DELETE statement with the use of IN operator and SUBQUERIES.

**Example:**

To remove rows from the table 'agent1' with following conditions -

**1.** 'da' and 'cu' are the aliases of 'agent1' and 'customer' table,

**2.** check the number 3 is in the result of the subquery which satisfies the condition bellow :

**3.** 'agent\_code' of 'agent1' table and 'agent\_code' of 'customer' table should not be same,

the following SQL statement can be used:

**SQL Code:**

DELETE FROM agent1 da

WHERE 3 IN(

SELECT grade FROM customer cu

WHERE agent1.agent\_code<>customer.agent\_code);

Copy

Output:



## SQL delete records using subqueries with alias and MIN

In this page, we are going to discuss, how rows can be removed from a table by SQL DELETE statement along with the SQL MIN() function.

**Example:**

To remove rows from the table 'agent1' with following conditions -

**1.** 'orders' table used as alias **'a'** and alias **'b',**

**2.** 'agent\_code' of 'agent1' should be within the 'agent\_code' in alias **'a'** which satisfies the condition bellow :

**i)** 'ord\_amount' of alias **'a'** must be equal to the minimum 'ord\_amount' of alias **'b'** which satisfies the condition bellow :

**a)** 'ord\_date' of alias **'a'** and alias **'b'** must be equal,

the following SQL statement can be used :

**SQL Code:**

DELETE FROM agent1

WHERE agent\_code IN

(SELECT agent\_code FROM orders a

WHERE ord\_amount=(

SELECT MIN(ord\_amount) FROM orders b

WHERE a.ord\_date=b.ord\_date));

Copy

Output:



## SQL delete records using subqueries with alias and MIN and COUNT

In this page, we are going to discuss, how rows can be removed from a table by SQL DELETE statement along with the SQL MIN() and COUNT() function.

**Example:**

To remove rows from the table 'agent1' with following conditions -

**1.** 'orders' table used as alias **'a'** and alias **'b'**

**2.** 'agent\_code' of 'agent1' should be within the 'agent\_code' in alias **'a'** which satisfies the condition bellow:

**i)** 'ord\_amount' of alias **'a'** must be equal to the minimum 'ord\_amount' of alias **'b'** which satisfies the condition bellow :

**a)** 'ord\_date' of alias **'a'** and alias **'b'** must be equal

**ii)** the number 1 should be less than the number of 'ord\_num' form alias **'b'** which satisfies the condition bellow :

**a)** 'ord\_date' of alias **'a'** and alias **'b'** must be equal,

the following SQL statement can be used :

**SQL Code:**

DELETE FROM agent1

WHERE agent\_code IN(

SELECT agent\_code FROM orders a

WHERE ord\_amount=(

SELECT MIN(ord\_amount) FROM orders b

WHERE a.ord\_date=b.ord\_date)

AND 1<(

SELECT COUNT(ord\_num) FROM orders b

WHERE a.ord\_date=b.ord\_date));

Copy

Output:



[**#**](https://www.dofactory.com/sql/subquery#syntax)

There is no general syntax. Subqueries are regular queries placed inside parenthesis. Subqueries can be used in different ways and at different locations inside a query.  
  
A subquery with the IN operator.

1. **SELECT column-names**
2. **FROM table-name1**
3. **WHERE value IN (SELECT column-name**
4. **FROM table-name2**
5. **WHERE condition)**

Subqueries can also assign column values to each record.

1. **SELECT column1 = (SELECT column-name**
2. **FROM table-name**
3. **WHERE condition),**
4. **column-names**
5. **FROM table-name**
6. **WHERE condition**

**More Examples**

[**#**](https://www.dofactory.com/sql/subquery#examples)

Subquery with IN

|  |
| --- |
| **ORDERITEM** |
| Id |
| OrderId |
| ProductId |
| UnitPrice |
| Quantity |

|  |
| --- |
| **PRODUCT** |
| Id |
| ProductName |
| SupplierId |
| UnitPrice |
| Package |
| IsDiscontinued |

Problem: List products with order quantities greater than 100.

**SELECT ProductName**

**FROM Product**

**WHERE Id IN (SELECT ProductId**

**FROM OrderItem**

**WHERE Quantity > 100)**

Try it live

Result: 12 records.

|  |
| --- |
| PRODUCTNAME |
| Guaraná Fantástica |
| Schoggi Schokolade |
| Chartreuse verte |
| Jack's New England Clam Chowder |
| Rogede sild |
| Manjimup Dried Apples |
| Perth Pasties |
| https://www.dofactory.com/img/ellipsis.png |

Subquery assigning column value

|  |
| --- |
| **CUSTOMER** |
| Id |
| FirstName |
| LastName |
| City |
| Country |
| Phone |

|  |
| --- |
| **ORDER** |
| Id |
| OrderDate |
| OrderNumber |
| CustomerId |
| TotalAmount |

Problem: List all customers with their total number of orders.

**SELECT FirstName, LastName,**

**OrderCount = (SELECT COUNT(O.id)**

**FROM [Order] O**

**WHERE O.CustomerId = C.Id)**

**FROM Customer C**

Try it live

This is called a correlated subquery because the subquery references the enclosing query, specifically, the C.Id in the WHERE clause.

Result:  91 records

|  |  |  |
| --- | --- | --- |
| FIRSTNAME | LASTNAME | ORDERCOUNT |
| Maria | Anders | 6 |
| Ana | Trujillo | 4 |
| Antonio | Moreno | 7 |
| Thomas | Hardy | 13 |
| Christina | Berglund | 18 |
| Hanna | Moos | 7 |
| Frédérique | Citeaux | 11 |
| Martín | Sommer | 3 |