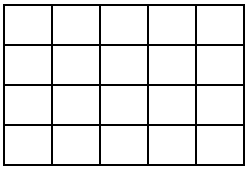
Table of Contents

Another shape we can have for a subquery is a multi-column, multi-row table



The subquery returns a virtual table that we can use in places where the query expects a table. We have seen this type of subquery in the Set operations. A Union join two subqueries.

The subquery could be used as a table expression in the From clause. In that case the subquery is sometimes called an in-line view. The subquery does not have to be the only table expression in the From clause; you can join the subquery result to a regular table to do a join. We have seen examples of this earlier; these are more interesting when we also consider using aggregates.

One advantage of using an inline view instead of creating a view is that you do not clutter up your database with a bunch of views that are used only once and that need to be documented and maintained.

1. Use a subquery in a From clause

When you embed a sub query in the From clause, it serves as a table expression.

- You should give the derived table an alias..
- Each column needs a name; calculated columns need an alias
- The column names must be unique

Demo 01: This query is not very interesting but it shows using a subquery in the From clause; I am including a table alias- tbl.
I do not want to see you writing queries this simplistic. The subquery is providing no value.

```
Select *
From (
   Select *
   From a_emp.jobs
   ) tbl;
```

This is also an inappropriate use of a subquery. Sometimes people get the idea that using subqueries makes your code more efficient- but that is not always the case.

```
Select job_Id, job_title, max_salary
From (
   Select job_Id, job_title, max_salary
   From a_emp.jobs
   Where max_salary is not null
   ) tbl
;
```

The previous examples used a subquery as part of a where filter. There are other places to use a subquery. Here we discuss using a subquery as a data source by using it in the From clause.

When you embed a sub query in the From clause, it serves as a table expression.

- You need to give the derived table an alias.
- Each column needs a name; calculated columns need an alias
- The column names must be unique

2. Counting in a subquery

You can use the COUNT (DISTINCT) feature to count the different numbers of shipping modes on all orders.

Demo 02: A standard COUNT DISTINCT- this does not count a null shipping_mode.

```
Select COUNT(DISTINCT shipping_mode) as num_diff_ship_modes
From a_oe.order_headers;
+---------------------+
| num_diff_ship_modes |
+---------------------+
|                   4 |
+---------------------+
```

Demo 03: If you wish to count nulls, you can use Coalesce to force a value to be counted.

```
Select count(distinct coalesce(shipping_mode, 'none')) as num_diff_ship_modes
From a_oe.order_headers;
+---------------------+
| num_diff_ship_modes |
+---------------------+
|                   5 |
+---------------------+
```

Demo 04: Another way to do this is to use a subquery in the FROM clause to return the distinct shipping methods- which does return a "null" group, and then use the parent query to count those rows. You need to provide a table alias for the subquery

```
Select count(*) as num_diff_ship_modes
From (
   Select distinct shipping_mode
   From a_oe.order_headers
   ) tbl;
+---------------------+
| num_diff_ship_modes |
+---------------------+
|                   5 |
+---------------------+
```

What if you want to count the number of distinct pairs of column values- for example, the number of pairs of shipping and order modes?

Demo 05: For a check, what are the distinct pairs of shipping and order modes values

```
Select distinct shipping_mode, ord_mode
From a_oe.order_headers
Order by shipping_mode, ord_mode;
+---------------+----------+
| shipping_mode | ord_mode |
+---------------+----------+
| NULL          | DIRECT   |
| NULL          | ONLINE   |
| FEDEX1        | DIRECT   |
| FEDEX1        | ONLINE   |
| FEDEX2        | DIRECT   |
| FEDEX2        | ONLINE   |
```

```
| UPSGR          | DIRECT   |
| USPS1          | DIRECT   |
| USPS1          | ONLINE   |
+----------------+----------+
```

Demo 06: You can do this with a subquery.

```
Select count(*) as num_diff_modes
From (
   Select distinct shipping_mode, ord_mode
   From a_oe.order_headers
   ) tbl;
+-----------------+
| num_diff_modes |
+-----------------+
|               9 |
+-----------------+
```

Demo 07: This approach yields a different result:

```
Select count(distinct shipping_mode, ord_mode) as  num_diff_modes
From a_oe.order_headers;
+-----------------+
| num_diff_modes |
+-----------------+
|               7 |
+-----------------+
```

With MySQL, Count (distinct expr1, expr2,…) counts the number of rows with different non-null values for the expressions.

3. Joining a regular table and a subquery

We want to display the number of employees in each department. We can start this as a simple query.

Demo 08: Since we have departments with no employees we need an outer join. This is incorrect. Try to figure out the error before you go to the next demo.

```
Select D.dept_id, count(*)
From  A_emp.departments  D
Left join a_emp.employees E on D.dept_id = E.dept_id
Group by D.dept_id;
+---------+----------+
| dept_id | count(*) |
+---------+----------+
|      10 |        1 |
|      20 |        1 |
|      30 |        8 |
|      35 |        3 |
|      80 |        3 |
|      90 |        1 |
|      95 |        1 |
|     210 |        2 |
|     215 |        4 |
+---------+----------+
```

Demo 09: The previous query used count(*) and counted the nulled-rows that the outer join generates. So every department row returned a value of at least 1. We want to count employees- not rows.

```
Select D.dept_id, count(E.emp_id)
From a_emp.departments  D
Left join a_emp.employees E on D.dept_id = E.dept_id
Group by D.dept_id
;
+---------+-----------------+
| dept_id | count(E.emp_id) |
+---------+-----------------+
|      10 |               1 |
|      20 |               1 |
|      30 |               8 |
|      35 |               3 |
|      80 |               3 |
|      90 |               0 |
|      95 |               0 |
|     210 |               2 |
|     215 |               4 |
+---------+-----------------+
```

Demo 10: If we want to see the department name , we can add that attribute to the select without adding it to the group by since we know there is only one department name per department id.

```
Select D.dept_id, D.dept_name, count(E.emp_id)
From a_emp.departments  D
Left join a_emp.employees E on D.dept_id = E.dept_id
Group by D.dept_id;
+---------+-----------------+-----------------+
| dept_id | dept_name       | count(E.emp_id) |
+---------+-----------------+-----------------+
|      10 | Administration  |               1 |
|      20 | Marketing       |               1 |
|      30 | Development     |               8 |
|      35 | Cloud Computing |               3 |
|      80 | Sales           |               3 |
|      90 | Shipping        |               0 |
|      95 | Logistics       |               0 |
|     210 | IT Support      |               2 |
|     215 | IT Support      |               4 |
+---------+-----------------+-----------------+
```

Demo 11: Alternately we could write a query that counts employees grouping by the department ID. This does not give us departments with no employees.

```
Select dept_id, count(*) as EmpCount
From a_emp.employees E
Group by dept_id;
+---------+----------+
| dept_id | EmpCount |
+---------+----------+
|      10 |        1 |
|      20 |        1 |
|      30 |        8 |
|      35 |        3 |
|      80 |        3 |
|     210 |        2 |
|     215 |        4 |
+---------+----------+
```

We could use that query and do an outer join to the department table. The subquery needs to supply an alias for the calculated (count) column because we want to refer to it.
Note that the subquery in enclosed in parentheses and given a table alias. Then the join is done as we usually do joins: on `D.dept_id = EC.dept_id`
We can use the subquery to contain the details of the aggregation.

Demo 12:

```
Select D.dept_id, D.dept_name, EC.EmpCount
From a_emp.departments D
Left join (
   Select dept_id, count(*) as EmpCount
   From a_emp.employees
   Group by dept_id
   ) EC  on D.dept_id = EC.dept_id
;
+---------+-----------------+----------+
| dept_id | dept_name       | EmpCount |
+---------+-----------------+----------+
|      10 | Administration  |        1 |
|      20 | Marketing       |        1 |
|      30 | Development     |        8 |
|      35 | Cloud Computing |        3 |
|      80 | Sales           |        3 |
|      90 | Shipping        |     NULL |
|      95 | Logistics       |     NULL |
|     210 | IT Support      |        2 |
|     215 | IT Support      |        4 |
+---------+-----------------+----------+
```

3.1. Joining subqueries

Demo 13: Joining a subquery to a base table

```
Select
  D.dept_id
, D.dept_name
, concat(l.loc_city, ' ',  l.loc_state_province)  as Location
, coalesce(EmpCount,0) as EmpCount
From a_emp.departments D
Join a_emp.locations L on D.loc_id = l.loc_id
Left join (
   Select dept_id, count(emp_id) as EmpCount
   From a_emp.employees E
   Group by dept_id
   ) EC on D.dept_id = EC.dept_id
;
+---------+----------------+---------------------------------+----------+
| dept_id | dept_name      | Location                        | EmpCount |
+---------+----------------+---------------------------------+----------+
|      20 | Marketing      | Southlake Texas                 |        1 |
|      80 | Sales          | Southlake Texas                 |        3 |
|      30 | Development    | South San Francisco California  |        8 |
|      10 | Administration | San Francisco California        |        1 |
|     210 | IT Support     | Toronto Ontario                 |        2 |
|     215 | IT Support     | Munich Bavaria                  |        4 |
+---------+----------------+---------------------------------+----------+
```

This uses two subqueries as data sources.

Demo 14: How many employees do we have in each department and what percent is that of all employees? Use each of these as a virtual table in the FROM clause to get the percent of each department over the entire employee table. Notice that we do not need a joining clause since the overall count has only one return row.

```
Select
  dept_id
, dept_count
, Round((dept_count / Count_All),2) AS Percnt
From
   (Select dept_id, count(1) AS dept_count  /* get count by department */
    From a_emp.employees
    Group by dept_id)  vt1,   /* get total count for all employees */
   (Select count(*) AS Count_All
    From a_emp.employees)   vt2
Order by Dept_id;
+---------+------------+--------+
| dept_id | dept_count | Percnt |
+---------+------------+--------+
|      10 |          1 |   0.05 |
|      20 |          1 |   0.05 |
|      30 |          8 |   0.36 |
|      35 |          3 |   0.14 |
|      80 |          4 |   0.14 |
|     210 |          2 |   0.09 |
|     215 |          4 |   0.18 |
+---------+------------+--------+
```

Demo 15: To get a percent value, multiply by 100 . You can also do some formatting to get a value that looks more like a percent.

```
Select
  dept_id
, dept_count
, concat(Round((100. * dept_count / Count_All),0), '%') AS Percnt
From
   (Select dept_id, count(1) AS dept_count
    From a_emp.employees
    Group by dept_id)  vt1,
   (Select count(*) AS Count_All
    From a_emp.employees)   vt2
Order by Dept_id;
+---------+------------+--------+
| dept_id | dept_count | Percnt |
+---------+------------+--------+
|      10 |          1 | 5%     |
|      20 |          1 | 5%     |
|      30 |          8 | 36%    |
|      35 |          3 | 14%    |
```

Demo 16: One way to find customers who bought both an appliance and a houseware item.
(oe_cust_orders is a view you should have created earlier)
The first subquery picks up the appliances
The second subquery picks up the houseware items
The join of the two subqueries checks that we are looking at the same customer id

```
Select distinct tblapl.custid
From
    (Select CustID
     From a_oe.cust_orders
     Where category ='APL'
     )  tblapl
 Join
    (Select CustID
     From  a_oe.cust_orders
     Where Category ='HW'
     )  tblhw
   On tblapl.custid = tblhw.custid
;
```

4. Generating data in the subquery

We have a table Credit ratings that we can join to the customers table.

Demo 17:

```
Select CS.cust_id, CS.credit_limit, CR.rating
From a_oe.customers  CS
Join a_oe.credit_ratings CR on CS.credit_limit
      between CR.low_limit and CR.high_limit;
+---------+--------------+-----------+
| cust_id | credit_limit | rating    |
+---------+--------------+-----------+
|  400300 |         6000 | Excellent |
|  400801 |          750 | Standard  |
|  401250 |          750 | Standard  |
|  401890 |         1750 | Good      |
|  402110 |          750 | Standard  |
|  402120 |          750 | Standard  |
|  403500 |         6000 | Excellent |
|  403750 |         6000 | Excellent |
|  404100 |         3500 | High      |
```

Demo 18: We do not have such a table for salary but maybe we want to do a similar rating for salaries. We can generate the rating data with a union

```
  Select 'under paid' as catg, 0 as low, 19999.99 as high
  union all
  Select 'medium paid' as catg, 20000.00 as low, 79999.99 as high
  union all
  Select 'over paid' as catg, 80000.00 as low, 9999999.99 as high ;
+-------------+----------+------------+
| catg        | low      | high       |
+-------------+----------+------------+
| under paid  |     0.00 |   19999.99 |
| medium paid | 20000.00 |   79999.99 |
| over paid   | 80000.00 | 9999999.99 |
+-------------+----------+------------+
```

Demo 19: Now join that union to the employees table

```
Select emp_id, name_last,salary, catg
From a_emp.employees E
Join (
     Select 'under paid' as catg, 0 as low, 19999.99 as high   union all
     Select 'medium paid' as catg, 20000.00 as low, 79999.99 as high
   union all
     Select 'over paid' as catg, 80000.00 as low, 9999999.99 as high
  ) Ratings on E.salary between Ratings.low and Ratings.high
Order by salary
;
-- selected rows
+--------+----------+----------+-------------+
| emp_id | name_last | salary   | catg        |
+--------+----------+----------+-------------+
|    150 | Tuck      |  6500.00 | under paid  |
|    103 | Hunol     |  9000.00 | under paid  |
|    108 | Green     | 12000.00 | under paid  |
|    109 | Fiet      | 15000.00 | under paid  |
|    200 | Whale     | 65000.00 | medium paid |
|    207 | Russ      | 65000.00 | medium paid |
|    155 | Hiller    | 80000.00 | over paid   |
|   1995 | Prince    | 85000.00 | over paid   |
|    146 | Partne    | 88954.00 | over paid   |
```

Demo 20: You could do this with a case

```
Select emp_id, name_last,salary
,  case
     when salary between 0 and 19999.99 then 'under paid'
     when salary between 20000.00 and 79999.99 then 'medium paid'
     when salary between 80000.00 and 9999999.99 then 'over paid'
   end as catg
From a_emp.employees E
Order by salary;
```

Demo 21: we may want just the aggregates

```
Select catg, count(*)as NumEmployees
From   a_emp.employees E
Join (
  Select 'under paid' as catg, 0 as low, 19999.99 as high
  union all
  Select 'medium paid' as catg, 20000.00 as low, 79999.99 as high
  union all
  Select 'over paid' as catg, 80000.00 as low, 9999999.99 as high
  ) Ratings on E.salary between Ratings.low and Ratings.high
Group by catg;
+-------------+--------------+
| catg        | NumEmployees |
+-------------+--------------+
| medium paid |            8 |
| over paid   |            5 |
| under paid  |            9 |
+-------------+--------------+
```