Table of Contents

There are two places in a Select statement where you might create something referred to as an alias. One is a column alias that you could use in the Select clause.

1. Column Alias

One use for column aliases is to make the output easier to understand.

This query does not use any column aliases; With MySQL if you have an expression that does not have a column alias then that column header is uses the expression.

```
Select ord_id, prod_id, quoted_price, quantity_ordered
, quoted_price * quantity_ordered
from  a_oe.order_details;
+--------+---------+--------------+------------------+-------------------------------+
| ord_id | prod_id | quoted_price | quantity_ordered | quoted_price * quantity_ordered |
+--------+---------+--------------+------------------+-------------------------------+
|    105 |    1030 |        25.00 |               12 |                        300.00 |
|    105 |    1020 |        12.95 |               12 |                        155.40 |
|    105 |    1010 |       150.00 |                5 |                        750.00 |
|    106 |    1060 |       255.95 |                1 |                        255.95 |
+--------+---------+--------------+------------------+-------------------------------+
```

Now we have an alias for several of the columns; the column alias is use as the column header; it can be used in the Order By clause- but not in the Where clause. When we get to subqueries we will see other uses for column aliases.

```
Select ord_id, prod_id
, quoted_price  AS PRICE
, quantity_ordered AS QUANTITY
, quoted_price * quantity_ordered AS ExtCost
from  a_oe.order_details;
+--------+---------+--------+----------+---------+
| ord_id | prod_id | PRICE  | QUANTITY | ExtCost |
+--------+---------+--------+----------+---------+
|    105 |    1030 |  25.00 |       12 |  300.00 |
|    105 |    1020 |  12.95 |       12 |  155.40 |
|    105 |    1010 | 150.00 |        5 |  750.00 |
|    106 |    1060 | 255.95 |        1 |  255.95 |
+--------+---------+--------+----------+---------+
```

2. Table Alias/Correlation Name/ Tuple Variable/Range Variable

We can also designate an alternate name for a table. This is most commonly called a table alias; SQL Server may use the term range variable names; you may also see this called a tuple variable or a correlation name. I'll generally use the term table alias since it is the most commonly used.

The table alias is defined in the From clause of the SQL statement and is limited in scope to that statement. The table alias is not saved on the server. Once you establish a table alias in a query, you need to use that alias, not the table name, in the other clauses of that query.

The use of table aliases is very common in SQL and you need to be aware of its use. However, poorly defined table aliases can make a query harder to read and understand.

The table alias is commonly a single letter but a short alias is often difficult to understand and remember. You should not have to keep referring to the From clause to interpret the Select clause. Table aliases can be longer than one character to make them more meaningful.

Some types of queries require the use of table aliases since the same table is included in the query more than once. In other queries, it is optional.

Each of these queries will run. Each of these queries uses a single table and do not require the use of qualification of the columns in the Select or Where clause.

```
select ord_id
, prod_id
, quoted_price
from  a_oe.order_details
where quoted_price = 150.00;
```

This query qualifies each column name and uses the full table name; this is rather long and harder to read.

```
select a_oe.order_details.ord_id
, a_oe.order_details.prod_id
, a_oe.order_details.quoted_price
from  a_oe.order_details
where a_oe.order_details.quoted_price = 150.00;
```

This query qualifies each column name and uses a table alias; this is easier to read.

```
select od.ord_id
, od.prod_id
, od.quoted_price
from  a_oe.order_details  od
where od.quoted_price = 150.00;
```

In this statement we use the token "od" for two different purposes. In the From clause, it identifies the table and stands for the set of rows that make up that table at the present time. In the Select and the Where clauses, "od" refers to a row at a time.

When we say

```
where od.quoted_price = 150.00
```

we are not testing that the entire table set of rows has a value for quoted_price of 150.00, or that at least one of the rows has that value; we are testing for each row- one row at a time- if that row has the value 150.00 for the quoted_price column in that row.

Likewise in the Select clause, the expression od.ord_id stands for the ord_id column of each row- one row at a time.

(This understanding of the token 'od' is the basis for calling this a tuple variable or a range variable.)