A simple rule to remember when using aggregates in a selection is if a column isn't aggregated it **must** be included in the group by clause.

The previous example would work perfectly once a group by is added for the stock number …

Worksheet | Query Builder

```
select sum(od_quantity)
,od_stock_number
from order_details
group by od_stock_number
```

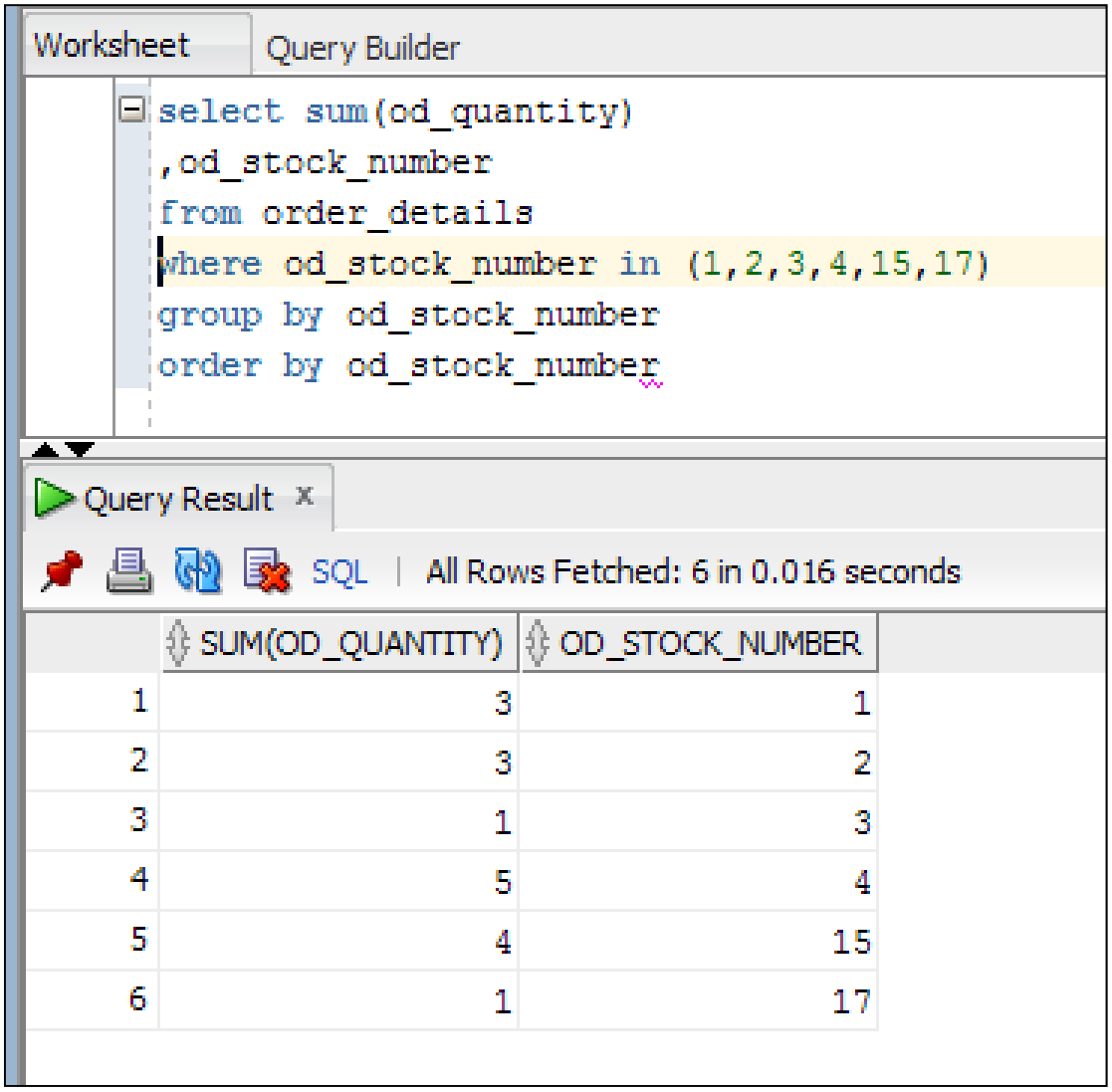
Query Result | SQL | All Rows Fetched: 13 in 0 seconds

	SUM(OD_QUANTITY)	OD_STOCK_NUMBER
1	5	25
2	3	1
3	3	13
4	1	14
5	10	26
6	1	21
7	3	2
8	1	24
9	5	4
10	10	23
11	1	17
12	1	3
13	4	15

The User should be aware that the Where clause in a Group By will limit records before the records are grouped because a where clause works on the individual rows, if the User needs to limit records after they have been 'grouped' then the Having clause needs to be used.

The next examples will show the usage of the where clause and the order by, note that order by can be used on any column including the aggregated column …

Aggregating Values – Using Where with Group By



	SUM(OD_QUANTITY)	OD_STOCK_NUMBER
1	3	1
2	3	2
3	1	3
4	5	4
5	4	15
6	1	17

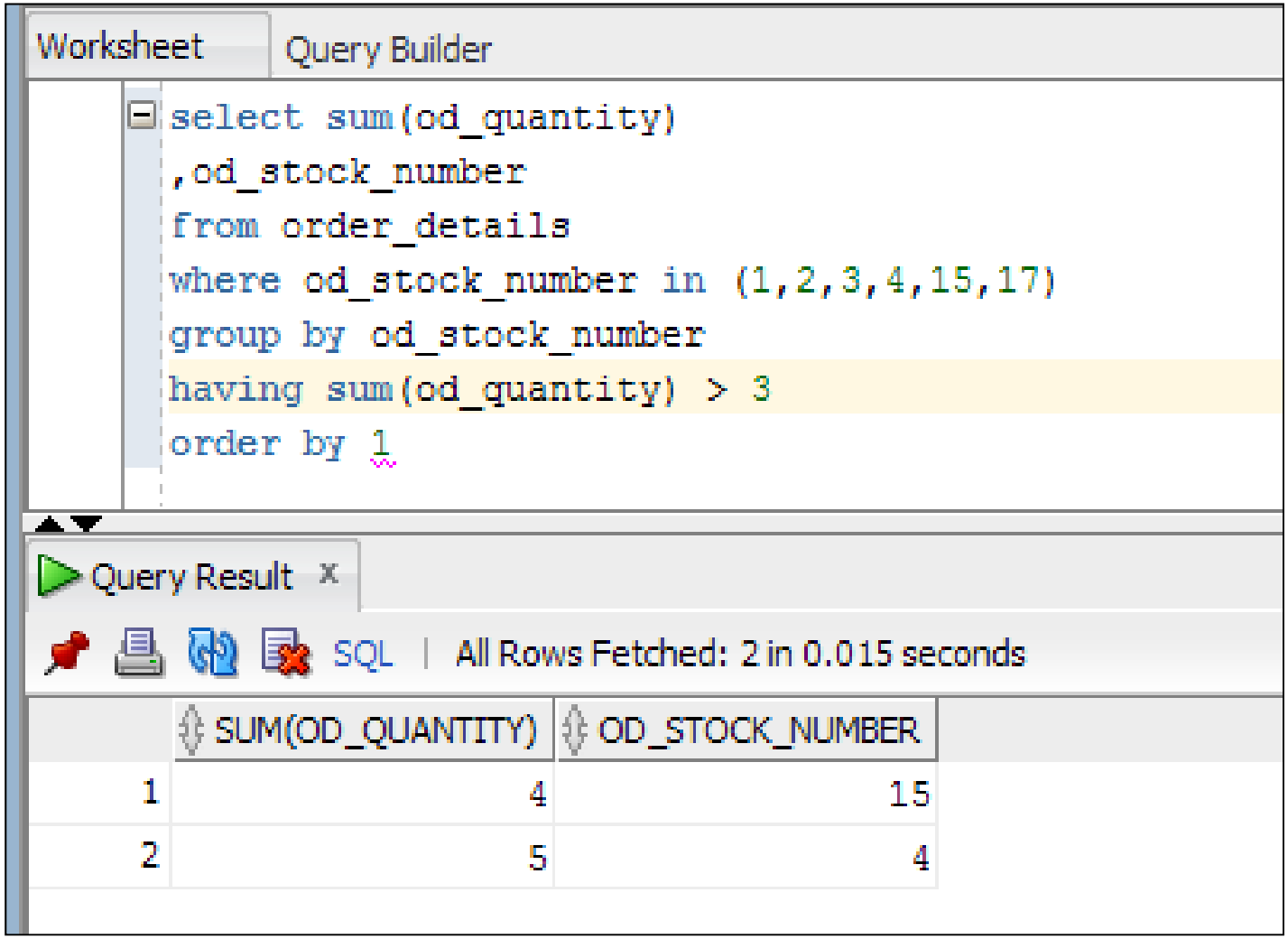
The Having clause is used when the User wishes to limit records after they have been grouped. Therefore Having is the group by equivalent of a where clause.

Having can only be used when the Group By clause has been used and must follow the group by in the SQL syntax.

The Having clause must work on the aggregated column(s)

The following example has both Having and Order By …

Aggregating Values – Using Having with Group By



Tree Walking is a technique which involves Tables with a foreign key to a column within itself.

This is called a self-joined Table and is typically found in a Personnel table when a person work for another person within the same Table, or items of Stock making up other items of Stock.

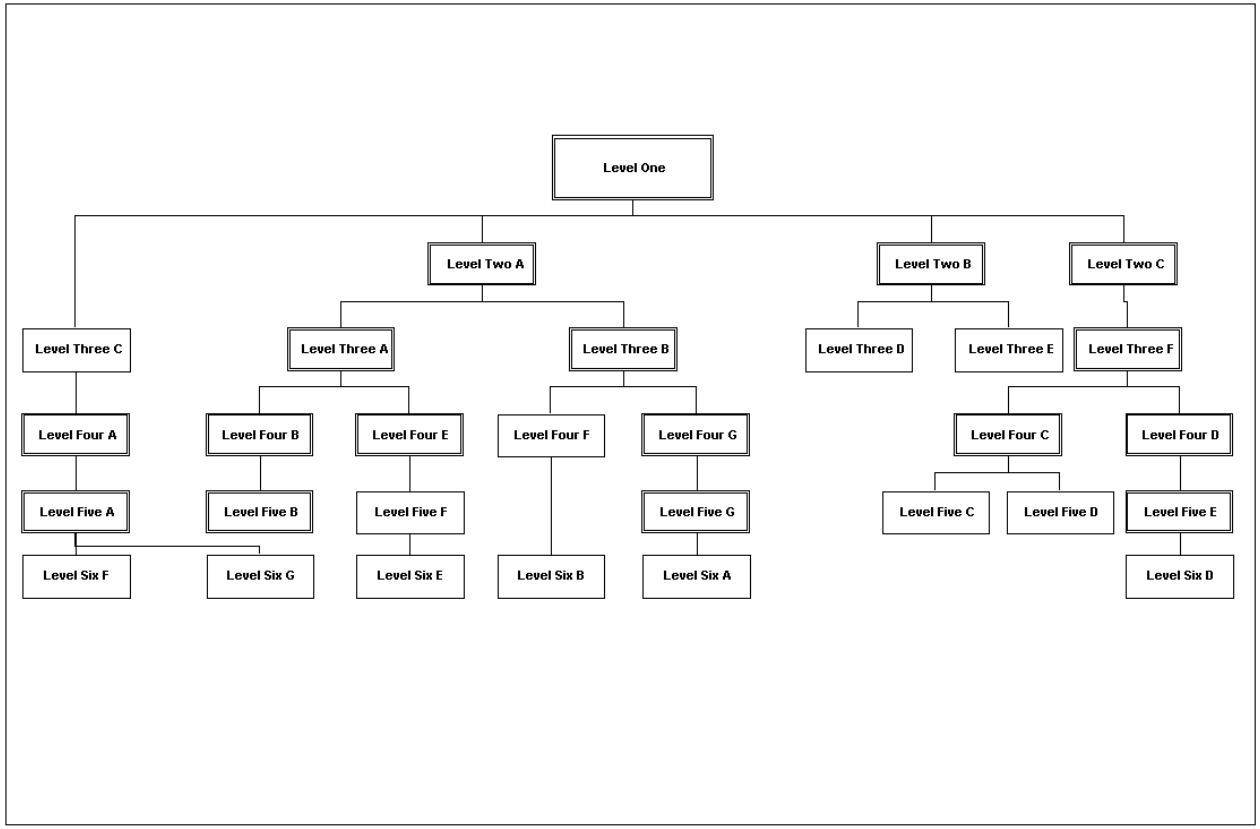
The Tree Walking SQL must be written in the SQL Worksheet, it cannot be created anywhere else in SQL Developer.

Typically the hierarchy will be defined in a table by the use of a foreign key or 'pigs ear', where a column perhaps called Parent ID would have a foreign key to the Primary Key column.

ID	Level	Name	Parent ID
1	One	Level 1	
3	Two	Level 2 A	1
5	Two	Level 2 B	1

Tree walking is the technique of taking the Parent ID and traversing either up or down the hierarchy or 'tree'.

Tree Walking is the ability to report from within a hierarchy, on either the record's parents or its children…..



select <column_name>
from <table_name>
start with *<column_name> = '<value>'*
connect by prior *<primary key column> = <foreign key name>;*

In the example this is how to work down the tree….

start with *name = 'LEVEL 1'*
connect by prior *ID = Parent_ID;*

To work up the tree the 'prior' clause is moved to the right …

start with *name = 'LEVEL 6A'*
connect by ID = ***prior*** *Parent_ID;*