This is an Oracle feature; this is not found in MySQL nor in SQL Server. Be very careful when you use this. People make a lot of mistakes with this approach (particularly on the final exam).

1. Two Level Nested Aggregates

Suppose we want to find the average amount of our orders. To do this, we need to group each order into a sub-table- so we need a group by order_id. We then need to find the sum of the price * quantity of those groups. Finally we need to find the average of that calculated data. Oracle allows nesting the aggregate functions with some limitations.

Demo 01: Step 1: What is the amount due for each order?

```
select order_id
, sum( quantity_ordered  * quoted_price) AS AmntDue
from oe_orderDetails
group by order_id;
```

```
   ORDER_ID    AMNTDUE
---------- ----------
       105     1205.4
       106     255.95
       107      49.99
       108       22.5
       109     149.99
. . . rows omitted
```

Demo 02: Step 2 What is the average of those sums? This is nesting the sum function inside the avg function.

```
select avg( sum( quantity_ordered  * quoted_price) ) AS "AvgOrderSize"
from oe_orderDetails
group by order_id;
```

```
AvgOrderSize
------------
  904.148723
```

We can find the highest average salary of any department and we can find the average of the highest salary in each department. Again this is not a matter of which query is correct- the question is what do you want to know.

Demo 03: What is the highest average salary of any department? We get a single row returned.

first what is the average salary by department?

```
select to_char(avg(salary), '999,999.00') as "AvgSalary"
from emp_employees
group by dept_id;
```

```
AvgSalary
-----------
  76,599.88
  15,000.00
  67,000.00
  83,563.50
  64,333.33
  36,000.00
 100,000.00
```

```
select max (avg(salary))
from emp_employees
group by dept_id;
```

```
MAX(AVG(SALARY))
----------------
          100000
```

Demo 04: What is the average of the highest salary of any department? We get a single row returned.

```
Step 1 highest salary by department
select max(salary)
from emp_employees
group by dept_id;
```

```
MAX(SALARY)
-----------
      99090
      15000
      69000
     120000
      98000
      59000
     100000
```

```
select avg ( max (salary))
from emp_employees
group by dept_id;
```

```
AVG(MAX(SALARY))
----------------
      80012.8571
```

See another version in the demo using a CTE.

Demo 05: If the company fired the highest paid person in each department, what would they save in salary?

```
select sum (max( salary))
from emp_employees
group by dept_id;
```

```
SUM(MAX(SALARY))
----------------
          560090
```

Now consider a demo we had recently which calculated the amount due for each order. This is that query, with fewer columns.

Demo 06: These are customers and the amounts of their orders. We group by the order id so that we can get the amount for each order and add the customer id to the grouping so that we can display that. Each order has only one customer id, so we are not changing the grouping logic.

```
select customer_id, order_id
, to_char( sum( quantity_ordered * quoted_price), '999,999.00') as "AmntDue"
from oe_orderHeaders
join oe_orderDetails  using (order_id)
group by order_id, customer_id
order by "AmntDue" desc
;
```

```
   CUSTOMER_ID     ORDER_ID AmntDue
---------- ---------- -----------
    900300        609    9,630.00
    903000        312    9,405.00
    400300        378    4,500.00
    900300        307    4,500.00
    903000       2121    3,800.25
    903000        551    3,500.00
    403000        395    2,925.00
    409150        415    2,879.95
. . .
```

Demo 07: Now we nest the aggregates to find the largest amount due for any of the orders. The group by clause affects the inner aggregate - in this case the sum and the outer aggregate works over that result set.

```
select max( sum( quantity_ordered * quoted_price)) as "LargestAmntDue"
from oe_orderHeaders
join oe_orderDetails  using (order_id)
group by order_id, customer_id
order by customer_id, order_id
;
```

```
LargestAmntDue
--------------
          9630
```

Now you want to see the largest order per customer- It would seem to make sense to try to use max(sum(quantity_ordered * quoted_price) but if you did that, what is the group by clause? We want to group by the order id to get the amt due per order but we want one row per customer so it looks like the group by clause should be just the customer id. Oracle does not allow the following query.

```
select
  customer_id
, order_id
, max(sum(quantity_ordered * quoted_price)) as "AmntDue"
from oe_orderHeaders
join oe_orderDetails Using (order_id)
group by customer_id;
```

The error message is rather clumsy.

```
Error report:
SQL Error: ORA-00937: not a single-group group function
```

What you really need to do is say do this aggregate -sum- grouping by the order id and then another aggregate -max- grouping by the customer id. We can do that with a CTE- isolating the first aggregate in the CTE and then using Max in the main query.

Demo 08:

```
with amtDueByOrder as (
   select customer_id, order_id
   ,sum( quantity_ordered * quoted_price) as AmntDue
   from oe_orderHeaders
   Join oe_orderDetails  using (order_id)
   group by order_id, customer_id
)
select customer_id, max(amntDue) as "LargestOrderByCustomer"
from amtDueByOrder
group by customer_id
order by customer_id;
```

```
CUSTOMER_ID LargestOrderByCustomer
----------- ----------------------
     400300                   4500
     401250                 255.95
     401890                 114.74
     402100                   2305
     403000                   2925
     403010                   1900
. . .
```