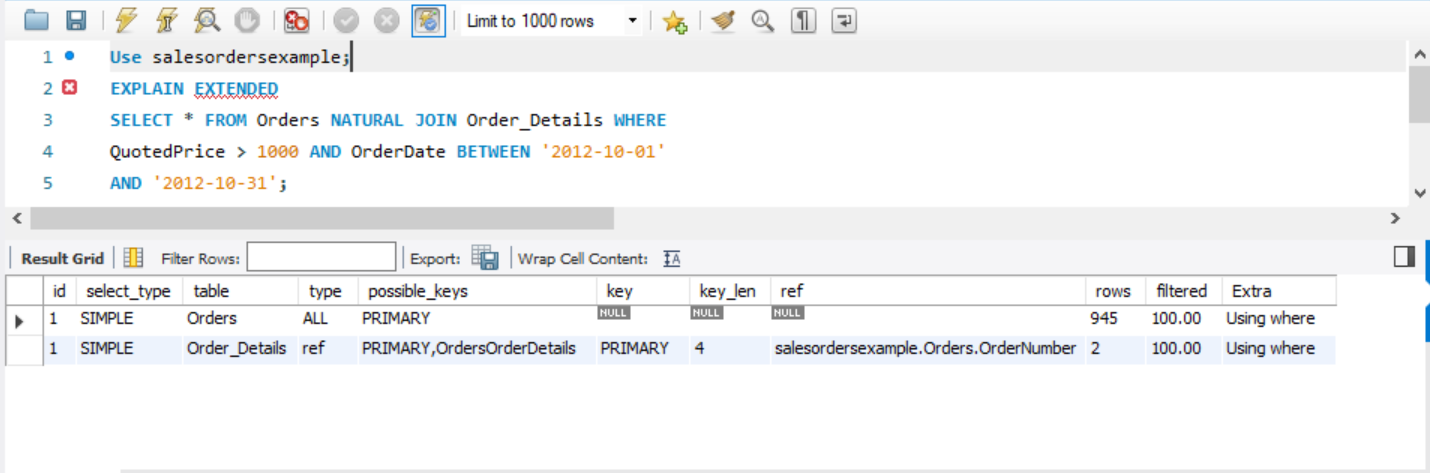
Output for the command given in question



Here two different queries were run in total.

At first, a simple query was run in Order table where the entire table had been searched though sequentially, giving 945 possible rows to look at (according to the DBMS statistics) and an extra condition of where was applied. Also based on the where clause, 100% of the rows will be examined.

Furthermore it is possible to use its primary keys as an index in this table, but it is not actually using it for this query and instead is looking though all data rows.

After that query, another simple query was run. But this time it was in the table Order\_details. It is of ref type which means the DBMS is using an index on key column to find the matching rows.The keys possible to be used here are the primary keys(from order table) and which is also the foreign key in orderdetails. Here the index used was the primary key and keylength was of 4 digits. Here the column compared with the index was ordernumber. The DBMS believes it needs to look at 2 rows and based on where condition (as stated in extra) it will see 100% of the table rows.

Because the order table had the primary key ordernumber while the order\_details had the foreign key for it. Thus it decided to go though order table first and use the primary keys as an index (as it is unique in order table). Then it used this information to check orderdate (which had the higher selectivity) in the order table. Once this was done, it checked in order\_details using the primary key from orders as an index where the “where” condition saw only two states, more than 1000 or not.

(indexes present in Order\_details)

