Output for the command given in question

```
Use salesordersexample;
EXPLAIN EXTENDED
SELECT * FROM Orders NATURAL JOIN Order_Details WHERE
QuotedPrice > 1000 AND OrderDate BETWEEN '2012-10-01'
AND '2012-10-31';
```

id	select_type	table	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	Orders	ALL	PRIMARY	NULL	NULL	NULL	945	100.00	Using where
1	SIMPLE	Order_Details	ref	PRIMARY,OrdersOrderDetails	PRIMARY	4	salesordersexample.Orders.OrderNumber	2	100.00	Using where

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)

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment
order_details	0	PRIMARY	1	OrderNumber	A	1987	NULL	NULL		BTREE		
order_details	0	PRIMARY	2	ProductNumber	A	3974	NULL	NULL		BTREE		
order_details	1	OrdersOrderDetails	1	OrderNumber	A	1987	NULL	NULL		BTREE		
order_details	1	ProductsOrderDetails	1	ProductNumber	A	76	NULL	NULL		BTREE		