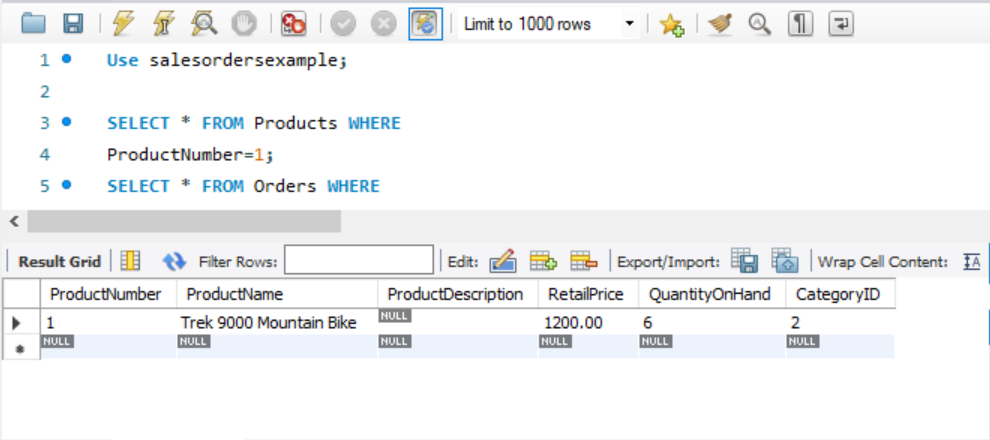
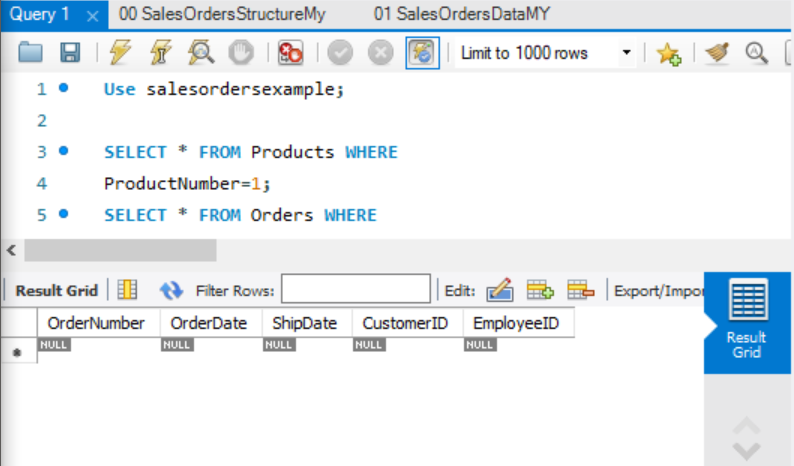
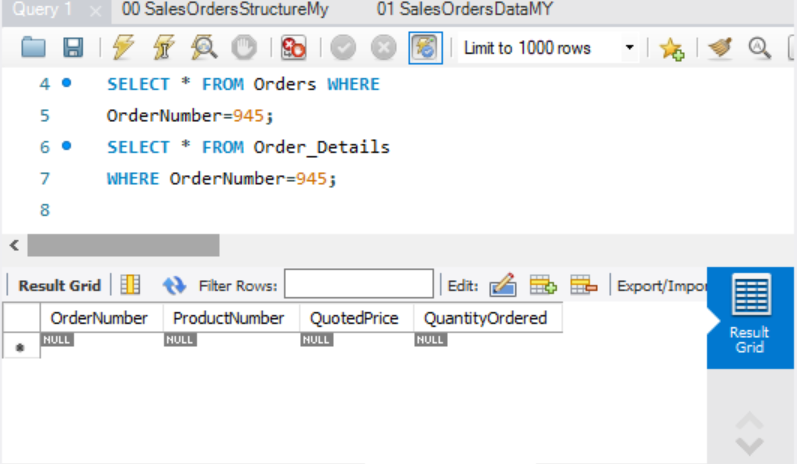
1)It gives null values for 2nd and 3rd query and the 1st query doesn’t change the productonhand number

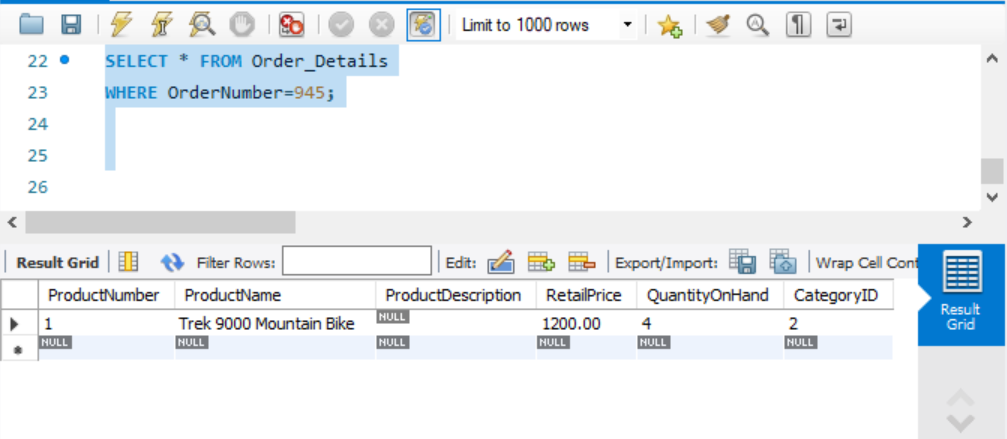


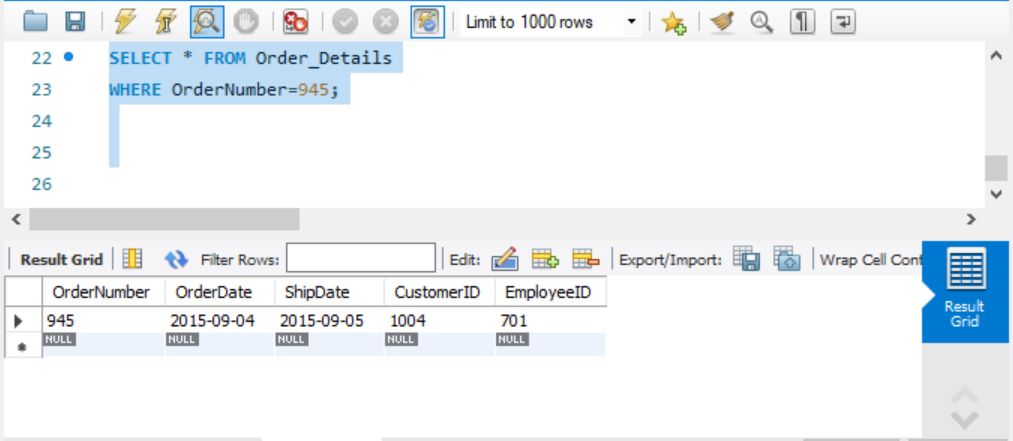


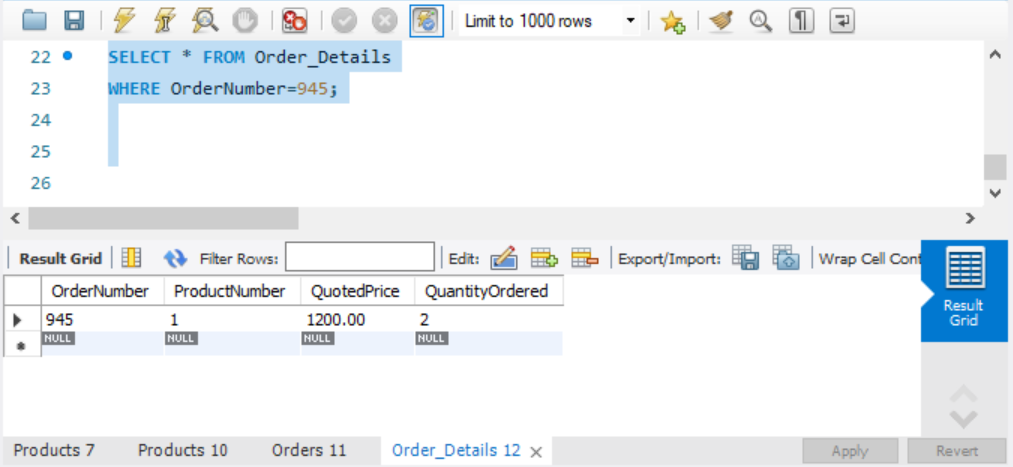


2)Still no change in output

3)Now all changes can be seen in their respective tables in right instance







4) Now all of the query results come as expected (giving the exact same results which had been seen in part 3)

Previously, the changes would only appear on transaction1 if it was committed in both transaction2 and transaction1. But now it can appear if only transaction 2 has been committed, but transaction1 has not!

This can lead to a lost update in the following way. At first the transaction1 read in the value before trasaction2 wrote something to it. Thus it will not know the presence of the write part in trasaction2. Thus when transaction1 will write something to it, it will have overridden the one written by transaction2’s write. Thus when transaction1 commits, the transaction2 modification have been lost and hence a “lost update” has been formed.

|  |  |
| --- | --- |
| Transaction1 | Transaction2 |
| R(k)  W(k)  C | R(k)  W(k)  C |

Or in line syntax: R2(k) R1(k) W2(k) C2 W1(k)C1

Where r is read (ie any sort of select)

And w is write (ie any sort of update or insert)