

Recommended Answer: Use `returned` in `orders.status`

Why?

Because:

1. `orders.status` represents high-level order lifecycle

Once the return is complete, the order as a whole is now considered "returned" from a business point of view — whether you refunded, replaced, or closed it.

So:

Order's journey: delivered → returned 

2. `return_processed` is more of a return-level detail

That should belong to the `returns.status`, not `orders.status`.

Your existing schema already does this well:

```
returns.status = 'requested' → 'approved' → 'processed'
```

So when `returns.status = 'processed'`, it's logical to:

```
UPDATE orders SET status = 'returned'
```

Final Design Summary

Table	Status Field	Example Value	Purpose
<code>orders</code>	<code>status</code>	<code>returned</code> 	High-level summary of order status
<code>returns</code>	<code>status</code>	<code>processed</code> 	Detailed state of return process

Status Flow Example

For a successful return:

orders.status: pending → confirmed → delivered → returned ✓ returns.status:
→ requested → approved → processed ✓

💡 Pro Tip:

If you ever need more detail (like refund completed), you can store it in:

- A `refunds` table
- Or more fields in the `returns` table (`refunded_at`, `refund_method`, etc.)

✓ Final Recommendation:

- ✓ Keep `orders.status = 'returned'` once return is completed.
- ⌚ Let `returns.status = 'processed'` indicate the detailed step-by-step flow.

Let me know if you want to automatically trigger this status update when a return is marked as processed.

'requested' → 'approved' → 'processed' to mujhe return ke konse status pe order ka status returned karna chahiye

Great! You're asking:

- At which `returns.status` should I update `orders.status` to `'returned'` ?

✓ Short Answer:

When `returns.status = 'processed'`, then update `orders.status = 'returned'` ✓

💡 Why?

Each return status represents a **stage**:

returns.status	Meaning
requested	Customer has asked to return the product
approved	Admin has approved the return
processed	Return is fully handled — item received, refund/replacement is done 

So until the return is **processed**, the order is still technically in a **transitional state**.

Your Status Update Logic Should Be:

When admin or system marks a return as `'processed'`, then:

```
UPDATE orders SET status = 'returned' WHERE id = <order_id>;
```

You can do this in your API:

```
if (returnStatus === 'processed') { await pool.query(` UPDATE orders SET
status = 'returned', updated_at = CURRENT_TIMESTAMP WHERE id = $1 `,
[orderId]); }
```

Final Flow Summary

returns.status	orders.status	Action
requested	delivered	No change
approved	delivered	No change
processed 	returned 	 Update order status here
rejected	delivered	No change