You're a database detective investigating suspicious orders in an e-commerce database.

Customers(cust_id PRIMARY KEY, name, email)

Products(prod_id PRIMARY KEY, name, price)

Orders(order_id PRIMARY KEY, cust_id FOREIGN KEY REFERENCES Customers(cust_id), order_date)

OrderItems(order_id FOREIGN KEY REFERENCES Orders(order_id),

prod_id FOREIGN KEY REFERENCES Products(prod_id),

quantity,

PRIMARY KEY(order_id, prod_id))

Returns(order_id FOREIGN KEY REFERENCES Orders(order_id),

prod_id FOREIGN KEY REFERENCES Products(prod_id),

return_date,

PRIMARY KEY(order_id, prod_id))

Clues

1. **Some orders exist in the `Orders` table without any corresponding items in `OrderItems`.** That shouldn't happen.
2. **Some returns are recorded for products that were never ordered.** Suspicious!
3. A trigger should prevent returns if the product wasn't actually in the order.
4. A view named `ValidOrders` should show only those orders that have at least one valid item.
5. A stored procedure `ProcessReturn` should:
    - Take `order_id` and `prod_id`.
    - Check if the product exists in that order.
    - If yes, insert into `Returns`.
    - If not, raise an error.

Your Tasks (Puzzle Steps)

1. **Identify the "phantom" orders** — orders with no items.
2. **Find invalid returns** — returns for non-ordered products.
3. **Write a view `ValidOrders` to only include orders with at least one product.**
4. **Design the `BEFORE INSERT` trigger on `Returns` to block invalid returns.**
5. **Implement the `ProcessReturn` stored procedure to safely handle returns.**

Bonus Challenge

Imagine someone deleted a customer from `Customers`. What happens to their orders and returns?

- Use **referential integrity** rules (CASCADE, SET NULL, RESTRICT).
- Propose what behavior the database should enforce and why.