**Assignment 4:** Compose SQL statements to BEGIN a transaction, INSERT a new record into the 'orders' table, COMMIT the transaction, then UPDATE the 'products' table, and ROLLBACK the transaction.

Below are the SQL statements to perform the described operations using transactions, assuming you have tables named orders and products.

**SQL Statements**

**1. BEGIN Transaction:** Start a new transaction.

BEGIN TRANSACTION;

**2. INSERT Into `order` Table:** Insert a new record into the **`order`** Table.

INSERT INTO orders (order\_id, customer\_id, order\_date, order\_amount)

VALUES (nextval('orders\_order\_id\_seq'), 123, CURRENT\_DATE, 100.00);

**-** This assumes you are using a sequence for generating **`order\_id`**, and **`123`** is a placeholder for **`customer\_id`**, **`CURRENT\_DATE`** is used to insert the current date as **`order\_date`**, and **`100.00`** is a placeholder for **`order\_amount`**.

**3. COMMIT Transaction:** Commit the transaction to make the changes permanent.

COMMIT;

**4. UPDATE `products` Table**: Perfome an update operation on the `products` Table.

UPDATE products

SET quantity = quantity - 1

WHERE product\_id = 456;

- This example assumes you are updating the **`products`** table to decrese the quantity of a product (**`product\_id = 456`**) by 1

**5. ROLLBACK Transaction:** Rollback the transaction to undo changes if needed (for example, in case of an error).

ROLLBACK;

**Explanation**

**- BEGIN TRANSACTION:** Starts a new transaction. All subsequentoperations(insert, update) are part of this transaction until it is either committed or rolled back.

**- INSERT INTO `orders`:** Inserts a new record into the **`orders`** table with dummy data (**`customer\_id`, `order\_date`, `order\_amount`**). Replace these values with actual data as per your requirements.

**-COMMIT**: Commits the transaction, making all changes (inserts and updates) permanent in the database.

**- UPDATE `products`:** Updates the **`products`** table, in this case, reducing the quantity of a product identified by **`product\_id = 456`**. Adjust the **`SET`** clause and **`WHERE`** clause according to your specific update requirements.

**- ROLLBACK:** Rolls back (undoes) the transaction, reverting any changes made since the transaction began. This is useful if there was an error or if you decide to cancel the changes for any reason before committing.

**Notes**

- Ensure that your database system supports transactions (**`BEGIN`, `COMMIT`, `ROLLBACK`**).

- Replace placeholder values (**`123`, `456`, `100.00`**, etc.) with actual data as per your database schema and requirements.

- Transactions ensure that multiple operations (inserts, updates, etc.) can be treated as a single unit of work, ensuring consistency and data integrity.