TASK-2

DML (Data Manipulation Language): commands are used to manipulate data stored in database tables. They modify the content of the tables without altering the table structure. DML operations are not auto-committed; they can be rolled back using TCL. Common DML commands:

1. *INSERT:* Adds new rows into a table
2. *UPDATE:* Modifies existing records
3. *DELETE:* Removes records from a table
4. *SELECT:* Retrieves data from the database

TCL (Transaction Control Language): commands manage transactions, which are groups of operations executed as a single unit. TCL ensures data integrity during such operations. Use TCL when handling tasks like deleting orders or inserting data with dependencies, to ensure atomicity. Common TCL commands:

1. *COMMIT:* Saves all changes made in the current transaction.
2. *ROLLBACK:* Reverts changes made in the current transaction.
3. *SAVEPOINT:* Sets a savepoint to which you can rollback.

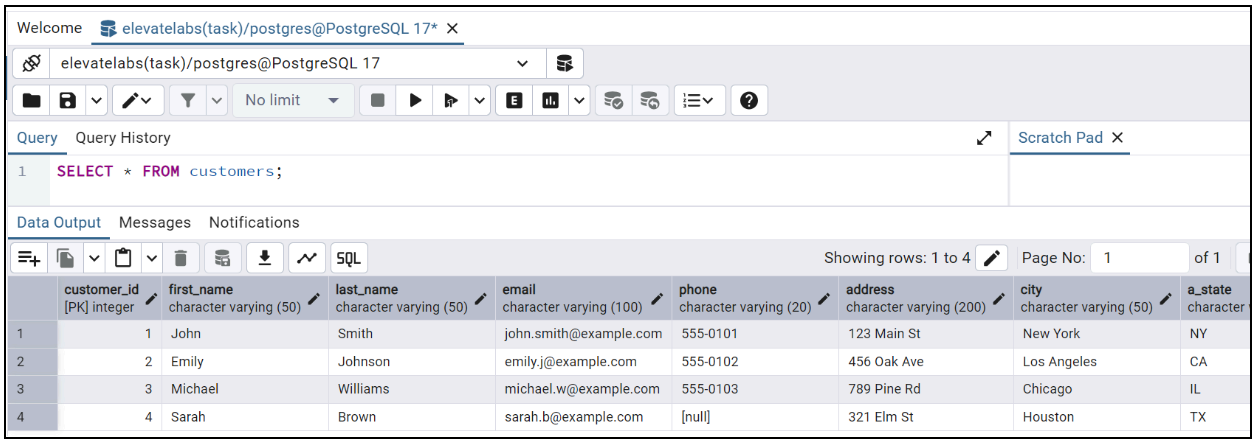
WHERE Clause: Used to filter rows before grouping or aggregation. Applies to individual rows in a table.

GROUP BY Clause: Used to group rows that have the same values in specified columns. Often used with aggregate functions like COUNT(), SUM(), AVG(), etc.

HAVING Clause: Used to filter groups after aggregation (unlike WHERE, which filters before). Must be used with GROUP BY.

BEFORE ***INSERT*** QUERY EXECUTION

1. TABLE → CUSTOMERS

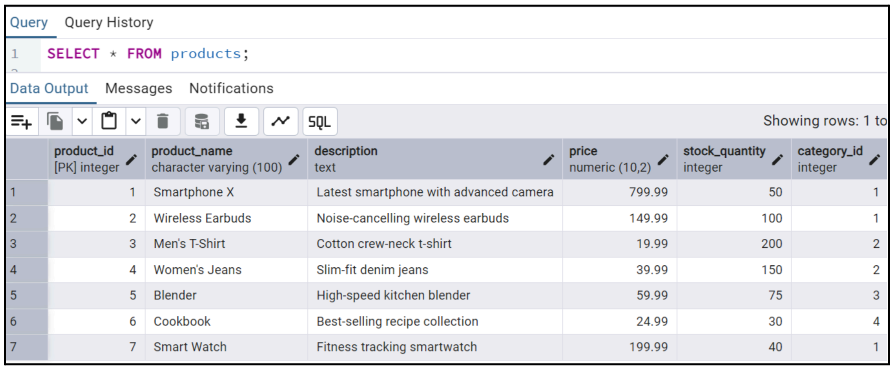


```
SELECT * FROM customers;
```

Showing rows: 1 to 4 Page No: 1 of 1

	customer_id [PK] integer	first_name character varying (50)	last_name character varying (50)	email character varying (100)	phone character varying (20)	address character varying (200)	city character varying (50)	a_state character
1	1	John	Smith	john.smith@example.com	555-0101	123 Main St	New York	NY
2	2	Emily	Johnson	emily.j@example.com	555-0102	456 Oak Ave	Los Angeles	CA
3	3	Michael	Williams	michael.w@example.com	555-0103	789 Pine Rd	Chicago	IL
4	4	Sarah	Brown	sarah.b@example.com	[null]	321 Elm St	Houston	TX

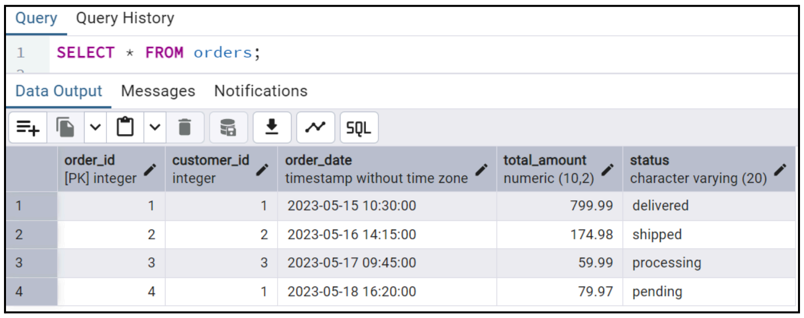
2. TABLE → PRODUCTS



```sql
SELECT * FROM products;
```

	product_id [PK] integer	product_name character varying (100)	description text	price numeric (10,2)	stock_quantity integer	category_id integer
1	1	Smartphone X	Latest smartphone with advanced camera	799.99	50	1
2	2	Wireless Earbuds	Noise-cancelling wireless earbuds	149.99	100	1
3	3	Men's T-Shirt	Cotton crew-neck t-shirt	19.99	200	2
4	4	Women's Jeans	Slim-fit denim jeans	39.99	150	2
5	5	Blender	High-speed kitchen blender	59.99	75	3
6	6	Cookbook	Best-selling recipe collection	24.99	30	4
7	7	Smart Watch	Fitness tracking smartwatch	199.99	40	1

3. TABLE → ORDERS

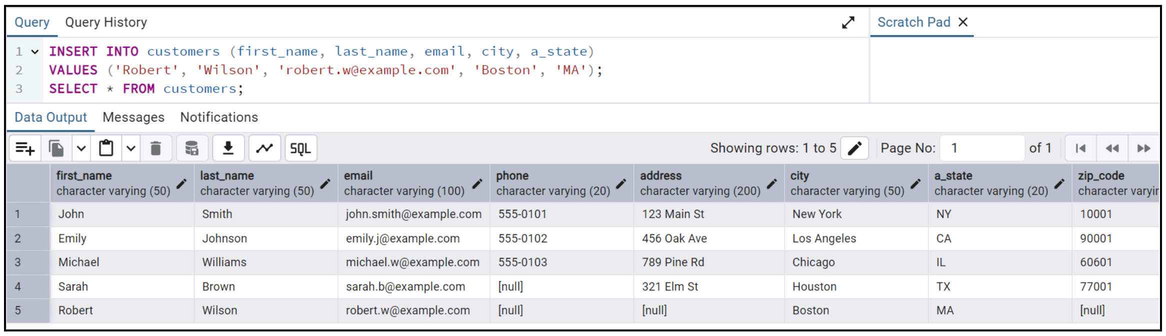


```sql
SELECT * FROM orders;
```

	order_id [PK] integer	customer_id integer	order_date timestamp without time zone	total_amount numeric (10,2)	status character varying (20)
1	1	1	2023-05-15 10:30:00	799.99	delivered
2	2	2	2023-05-16 14:15:00	174.98	shipped
3	3	3	2023-05-17 09:45:00	59.99	processing
4	4	1	2023-05-18 16:20:00	79.97	pending

AFTER ***INSERT*** QUERY EXECUTION

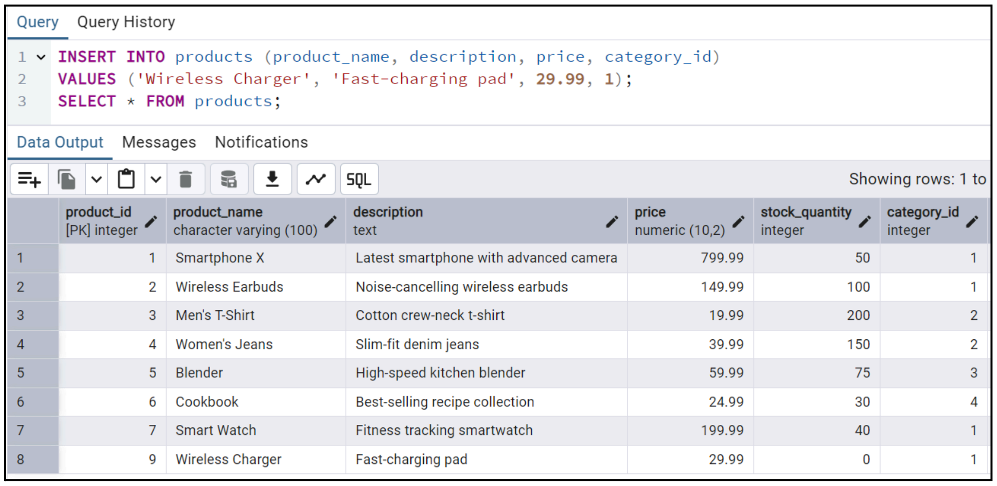
1. TABLE → CUSTOMERS



```sql
INSERT INTO customers (first_name, last_name, email, city, a_state)
VALUES ('Robert', 'Wilson', 'robert.w@example.com', 'Boston', 'MA');
SELECT * FROM customers;
```

	first_name character varying (50)	last_name character varying (50)	email character varying (100)	phone character varying (20)	address character varying (200)	city character varying (50)	a_state character varying (20)	zip_code character varyir
1	John	Smith	john.smith@example.com	555-0101	123 Main St	New York	NY	10001
2	Emily	Johnson	emily.j@example.com	555-0102	456 Oak Ave	Los Angeles	CA	90001
3	Michael	Williams	michael.w@example.com	555-0103	789 Pine Rd	Chicago	IL	60601
4	Sarah	Brown	sarah.b@example.com	[null]	321 Elm St	Houston	TX	77001
5	Robert	Wilson	robert.w@example.com	[null]	[null]	Boston	MA	[null]

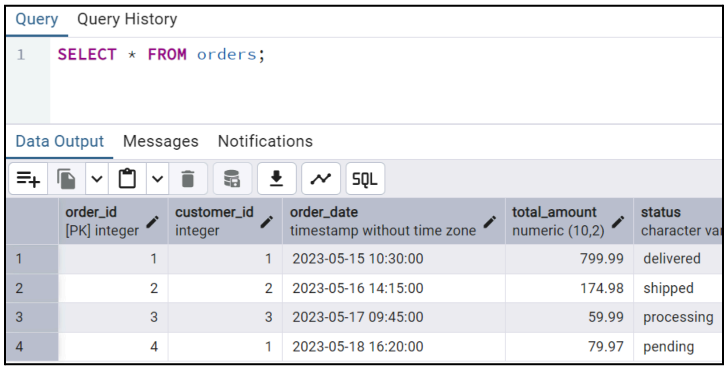
2. TABLE → PRODUCTS



```sql
INSERT INTO products (product_name, description, price, category_id)
VALUES ('Wireless Charger', 'Fast-charging pad', 29.99, 1);
SELECT * FROM products;
```

	product_id [PK] integer	product_name character varying (100)	description text	price numeric (10,2)	stock_quantity integer	category_id integer
1	1	Smartphone X	Latest smartphone with advanced camera	799.99	50	1
2	2	Wireless Earbuds	Noise-cancelling wireless earbuds	149.99	100	1
3	3	Men's T-Shirt	Cotton crew-neck t-shirt	19.99	200	2
4	4	Women's Jeans	Slim-fit denim jeans	39.99	150	2
5	5	Blender	High-speed kitchen blender	59.99	75	3
6	6	Cookbook	Best-selling recipe collection	24.99	30	4
7	7	Smart Watch	Fitness tracking smartwatch	199.99	40	1
8	9	Wireless Charger	Fast-charging pad	29.99	0	1

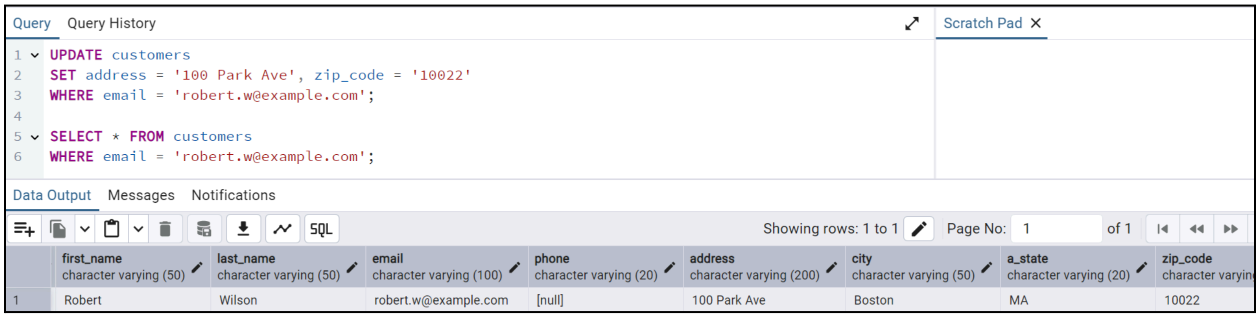
3. TABLE → ORDERS



```sql
SELECT * FROM orders;
```

	order_id [PK] integer	customer_id integer	order_date timestamp without time zone	total_amount numeric (10,2)	status character va
1	1	1	2023-05-15 10:30:00	799.99	delivered
2	2	2	2023-05-16 14:15:00	174.98	shipped
3	3	3	2023-05-17 09:45:00	59.99	processing
4	4	1	2023-05-18 16:20:00	79.97	pending

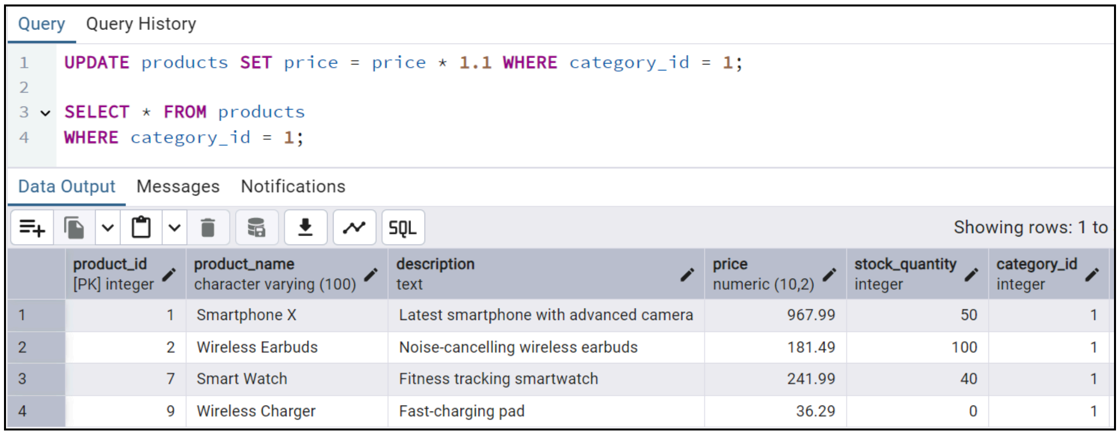
AFTER ***UPDATE*** QUERY EXECUTION



```sql
UPDATE customers
SET address = '100 Park Ave', zip_code = '10022'
WHERE email = 'robert.w@example.com';

SELECT * FROM customers
WHERE email = 'robert.w@example.com';
```

Showing rows: 1 to 1 Page No: 1 of 1

	first_name character varying (50)	last_name character varying (50)	email character varying (100)	phone character varying (20)	address character varying (200)	city character varying (50)	a_state character varying (20)	zip_code character varyin
1	Robert	Wilson	robert.w@example.com	[null]	100 Park Ave	Boston	MA	10022

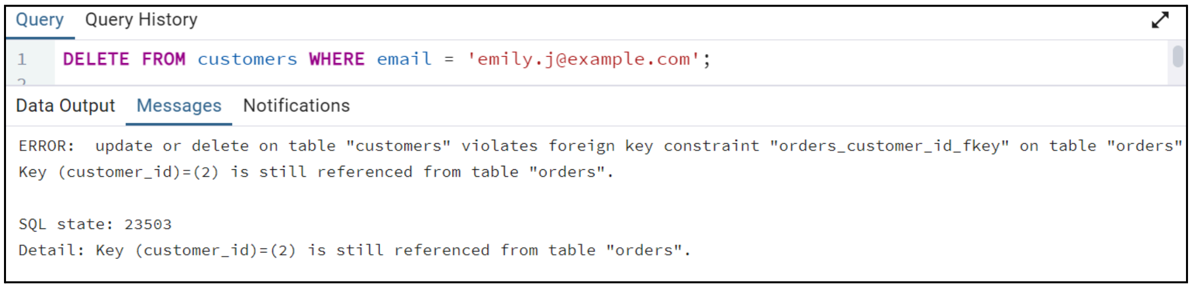


```sql
UPDATE products SET price = price * 1.1 WHERE category_id = 1;

SELECT * FROM products
WHERE category_id = 1;
```

Showing rows: 1 to

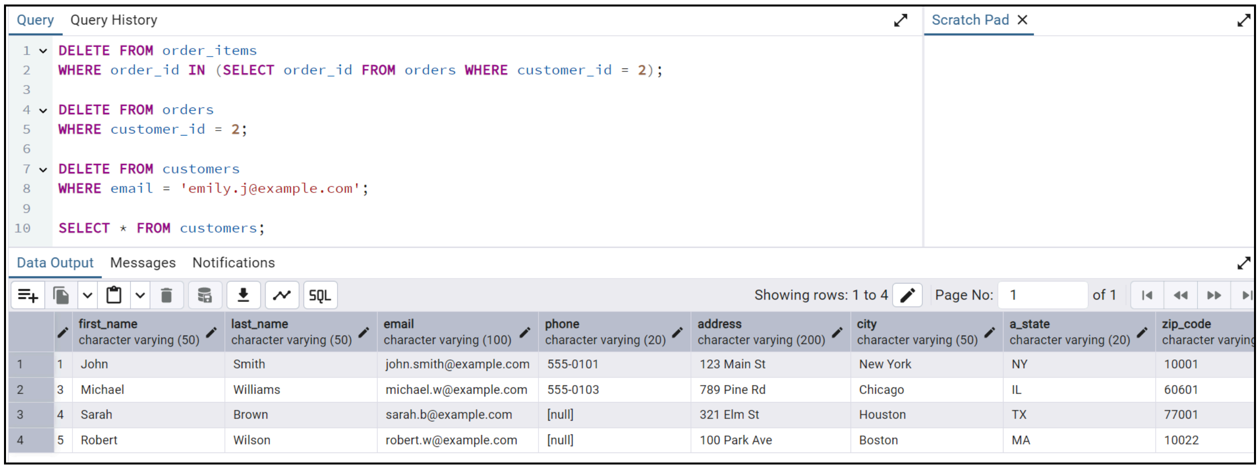
	product_id [PK] integer	product_name character varying (100)	description text	price numeric (10,2)	stock_quantity integer	category_id integer
1	1	Smartphone X	Latest smartphone with advanced camera	967.99	50	1
2	2	Wireless Earbuds	Noise-cancelling wireless earbuds	181.49	100	1
3	7	Smart Watch	Fitness tracking smartwatch	241.99	40	1
4	9	Wireless Charger	Fast-charging pad	36.29	0	1

AFTER ***DELETE*** QUERY EXECUTION



```sql
DELETE FROM customers WHERE email = 'emily.j@example.com';
```

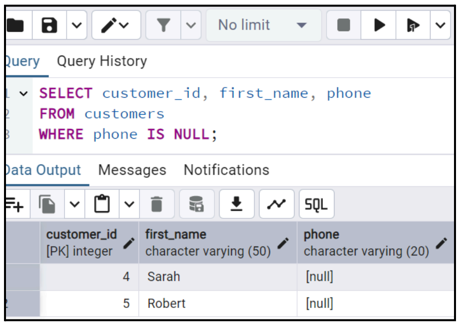
```
ERROR:  update or delete on table "customers" violates foreign key constraint "orders_customer_id_fkey" on table "orders"
Key (customer_id)=(2) is still referenced from table "orders".

SQL state: 23503
Detail: Key (customer_id)=(2) is still referenced from table "orders".
```



```sql
DELETE FROM order_items
WHERE order_id IN (SELECT order_id FROM orders WHERE customer_id = 2);

DELETE FROM orders
WHERE customer_id = 2;

DELETE FROM customers
WHERE email = 'emily.j@example.com';

SELECT * FROM customers;
```

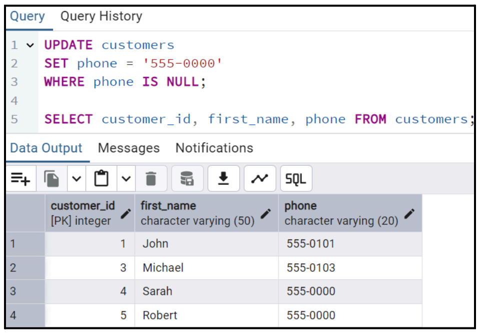
		first_name character varying (50)	last_name character varying (50)	email character varying (100)	phone character varying (20)	address character varying (200)	city character varying (50)	a_state character varying (20)	zip_code character varying
1	1	John	Smith	john.smith@example.com	555-0101	123 Main St	New York	NY	10001
2	3	Michael	Williams	michael.w@example.com	555-0103	789 Pine Rd	Chicago	IL	60601
3	4	Sarah	Brown	sarah.b@example.com	[null]	321 Elm St	Houston	TX	77001
4	5	Robert	Wilson	robert.w@example.com	[null]	100 Park Ave	Boston	MA	10022

AFTER ***HANDLING NULL ENTRY*** QUERY EXECUTION



```sql
SELECT customer_id, first_name, phone
FROM customers
WHERE phone IS NULL;
```

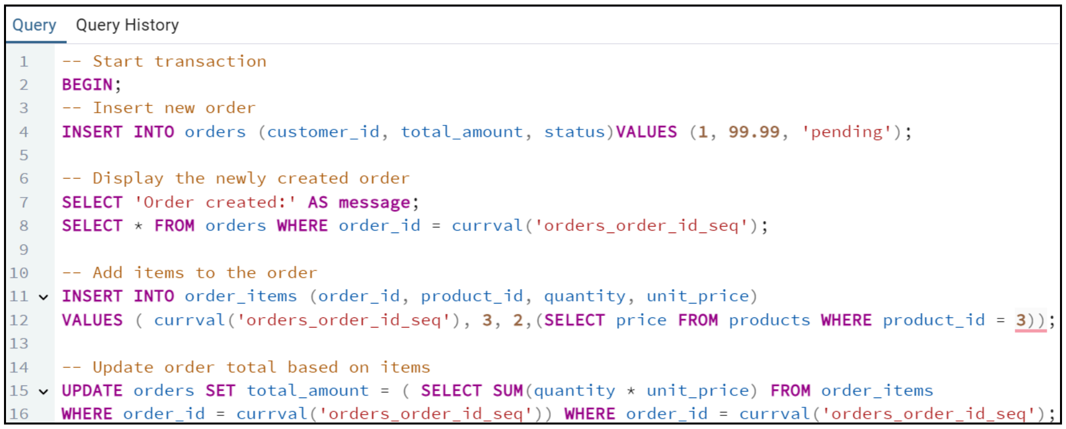
	customer_id [PK] integer	first_name character varying (50)	phone character varying (20)
	4	Sarah	[null]
	5	Robert	[null]



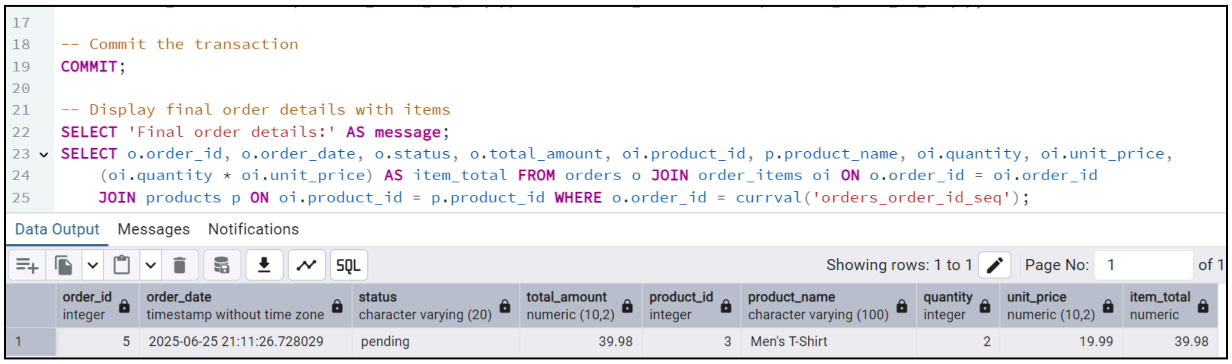
```sql
UPDATE customers
SET phone = '555-0000'
WHERE phone IS NULL;

SELECT customer_id, first_name, phone FROM customers;
```

	customer_id [PK] integer	first_name character varying (50)	phone character varying (20)
1	1	John	555-0101
2	3	Michael	555-0103
3	4	Sarah	555-0000
4	5	Robert	555-0000

AFTER ***TRANSACTION*** QUERY EXECUTION



```sql
-- Start transaction
BEGIN;
-- Insert new order
INSERT INTO orders (customer_id, total_amount, status)VALUES (1, 99.99, 'pending');

-- Display the newly created order
SELECT 'Order created:' AS message;
SELECT * FROM orders WHERE order_id = currval('orders_order_id_seq');

-- Add items to the order
INSERT INTO order_items (order_id, product_id, quantity, unit_price)
VALUES ( currval('orders_order_id_seq'), 3, 2,(SELECT price FROM products WHERE product_id = 3));

-- Update order total based on items
UPDATE orders SET total_amount = ( SELECT SUM(quantity * unit_price) FROM order_items
WHERE order_id = currval('orders_order_id_seq')) WHERE order_id = currval('orders_order_id_seq');
```



```sql

-- Commit the transaction
COMMIT;

-- Display final order details with items
SELECT 'Final order details:' AS message;
SELECT o.order_id, o.order_date, o.status, o.total_amount, oi.product_id, p.product_name, oi.quantity, oi.unit_price,
    (oi.quantity * oi.unit_price) AS item_total FROM orders o JOIN order_items oi ON o.order_id = oi.order_id
    JOIN products p ON oi.product_id = p.product_id WHERE o.order_id = currval('orders_order_id_seq');
```

	order_id integer	order_date timestamp without time zone	status character varying (20)	total_amount numeric (10,2)	product_id integer	product_name character varying (100)	quantity integer	unit_price numeric (10,2)	item_total numeric
1	5	2025-06-25 21:11:26.728029	pending	39.98	3	Men's T-Shirt	2	19.99	39.98