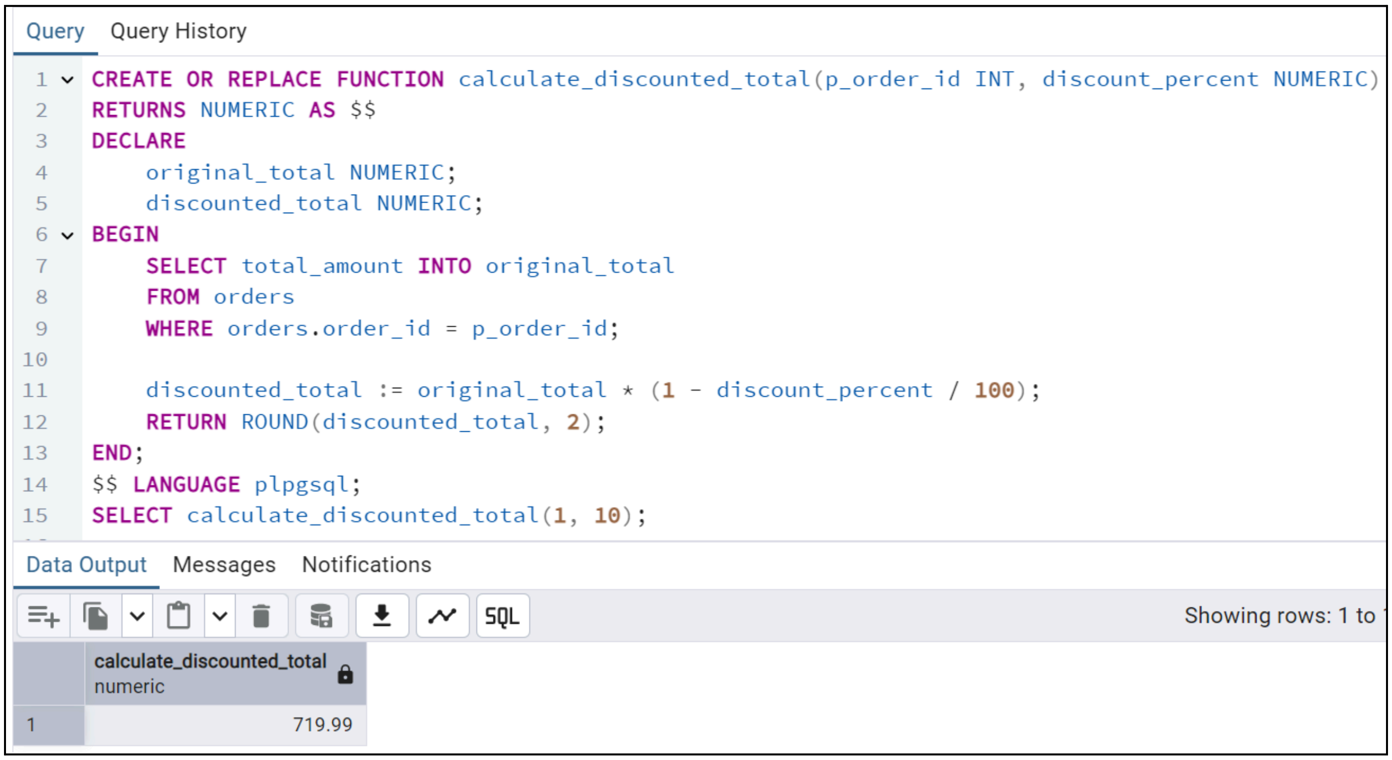
TASK-8

STORED PROCEDURE

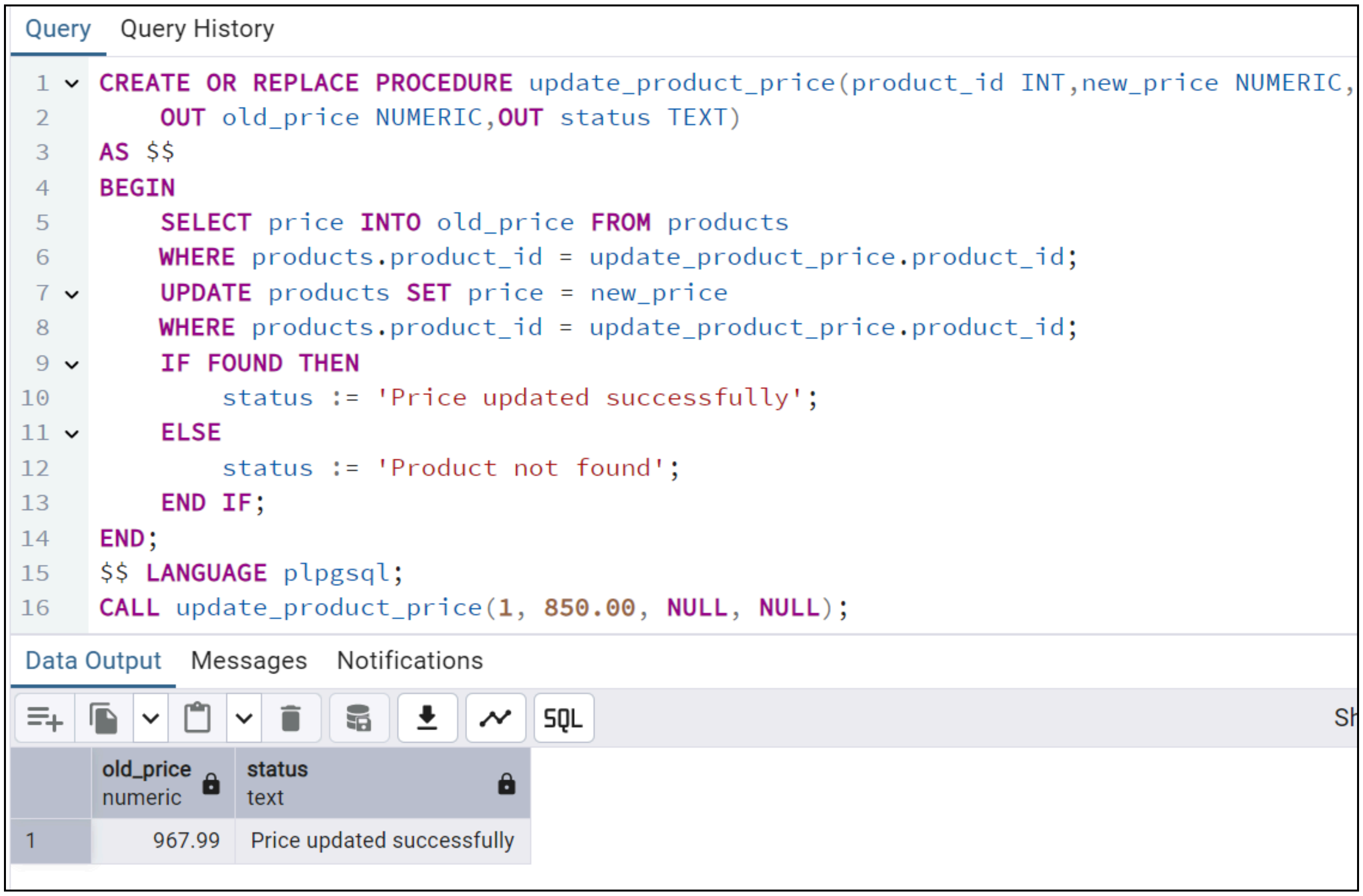
A stored procedure is a precompiled set of one or more SQL statements that are stored and executed on the database server. It acts like a function or method in programming languages and is used to perform a specific task, such as updating records, inserting data, or performing calculations. Unlike regular SQL queries written ad hoc by a user or application, stored procedures are saved in the database and can be reused by calling them whenever needed.

Stored procedures support input (IN) parameters, output (OUT) parameters, and input/output (INOUT) parameters, allowing them to process dynamic values and return results. They can also contain control-flow constructs like IF, WHILE, and CASE, and can include error handling and transaction management to ensure reliable execution. This makes them ideal for encapsulating business logic within the database, improving performance, security, and maintainability by reducing client-server communication and centralizing complex logic. For example, a stored procedure could be written to process a new customer order checking inventory, applying discounts, and updating multiple tables—all within a single transactional unit.



```
CREATE OR REPLACE FUNCTION calculate_discounted_total(p_order_id INT, discount_percent NUMERIC)
RETURNS NUMERIC AS $$
DECLARE
    original_total NUMERIC;
    discounted_total NUMERIC;
BEGIN
    SELECT total_amount INTO original_total
    FROM orders
    WHERE orders.order_id = p_order_id;

    discounted_total := original_total * (1 - discount_percent / 100);
    RETURN ROUND(discounted_total, 2);
END;
$$ LANGUAGE plpgsql;
SELECT calculate_discounted_total(1, 10);
```

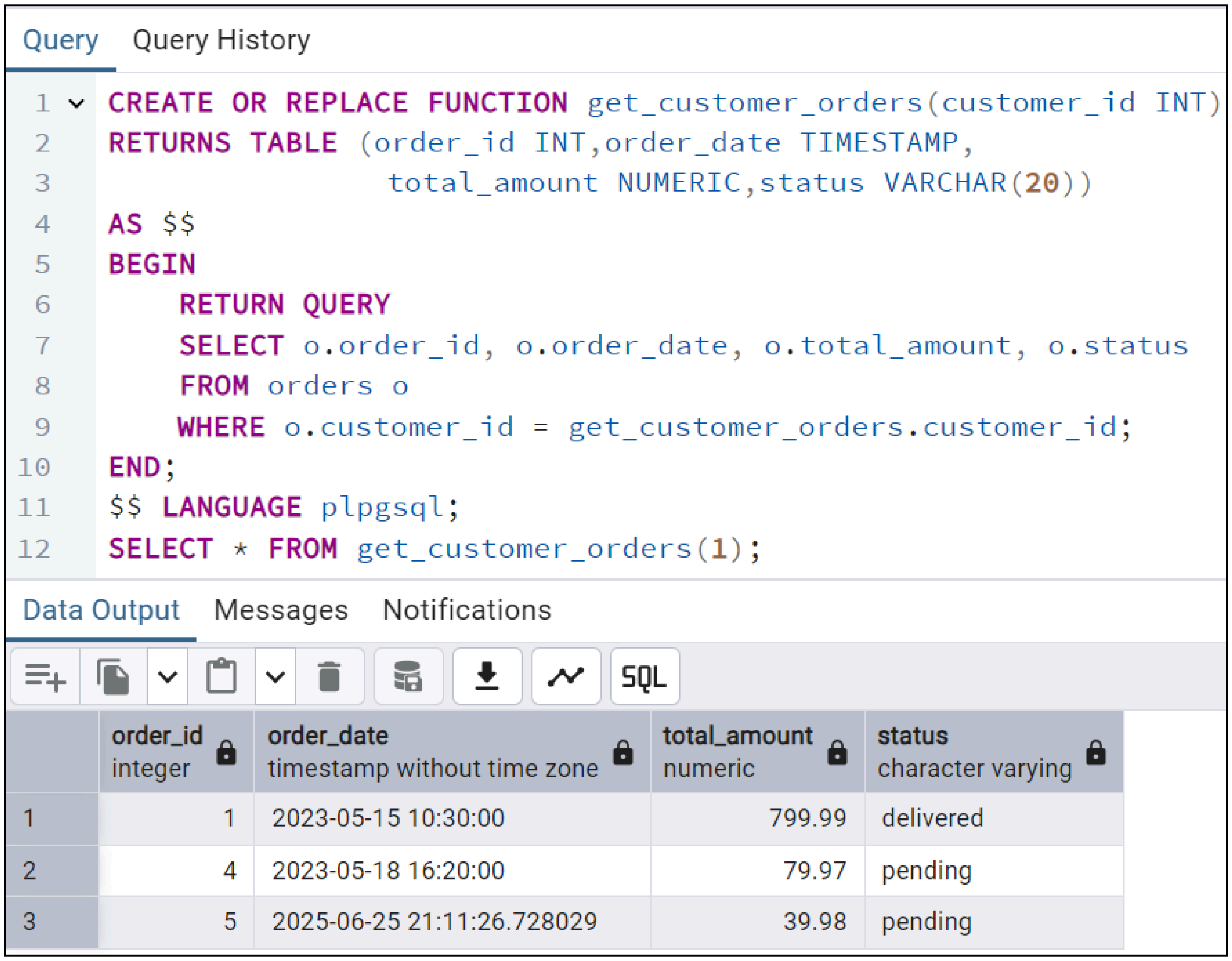
	calculate_discounted_total numeric
1	719.99


Query Query History

```
CREATE OR REPLACE PROCEDURE update_product_price(product_id INT,new_price NUMERIC,
    OUT old_price NUMERIC,OUT status TEXT)
AS $$
BEGIN
    SELECT price INTO old_price FROM products
    WHERE products.product_id = update_product_price.product_id;
    UPDATE products SET price = new_price
    WHERE products.product_id = update_product_price.product_id;
    IF FOUND THEN
        status := 'Price updated successfully';
    ELSE
        status := 'Product not found';
    END IF;
END;
$$ LANGUAGE plpgsql;
CALL update_product_price(1, 850.00, NULL, NULL);
```

Data Output Messages Notifications

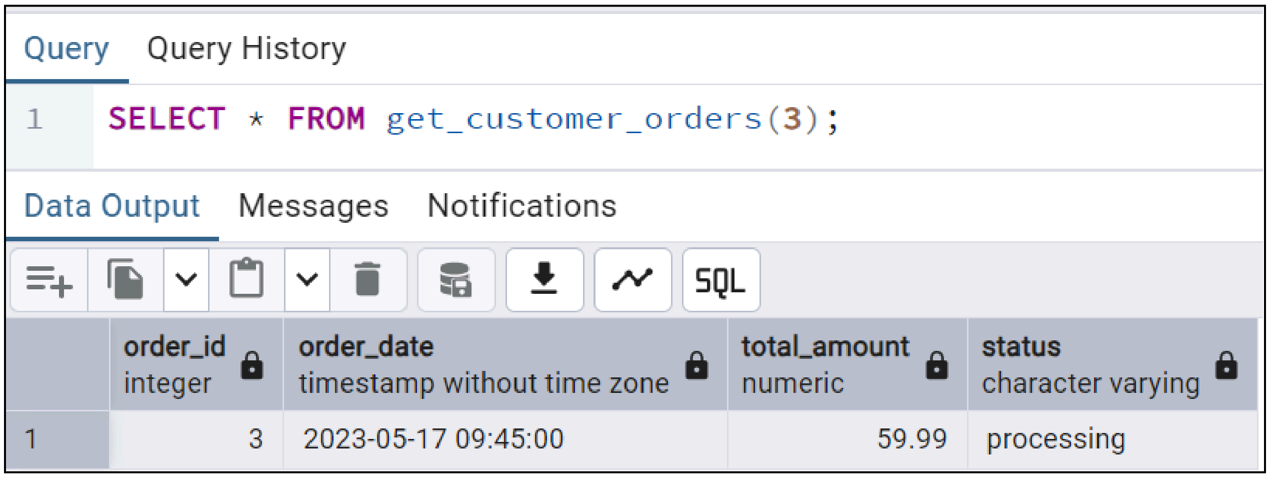
	old_price numeric	status text
1	967.99	Price updated successfully


Query Query History

```
CREATE OR REPLACE FUNCTION get_customer_orders(customer_id INT)
RETURNS TABLE (order_id INT,order_date TIMESTAMP,
                    total_amount NUMERIC,status VARCHAR(20))
AS $$
BEGIN
    RETURN QUERY
    SELECT o.order_id, o.order_date, o.total_amount, o.status
    FROM orders o
    WHERE o.customer_id = get_customer_orders.customer_id;
END;
$$ LANGUAGE plpgsql;
SELECT * FROM get_customer_orders(1);
```

Data Output Messages Notifications

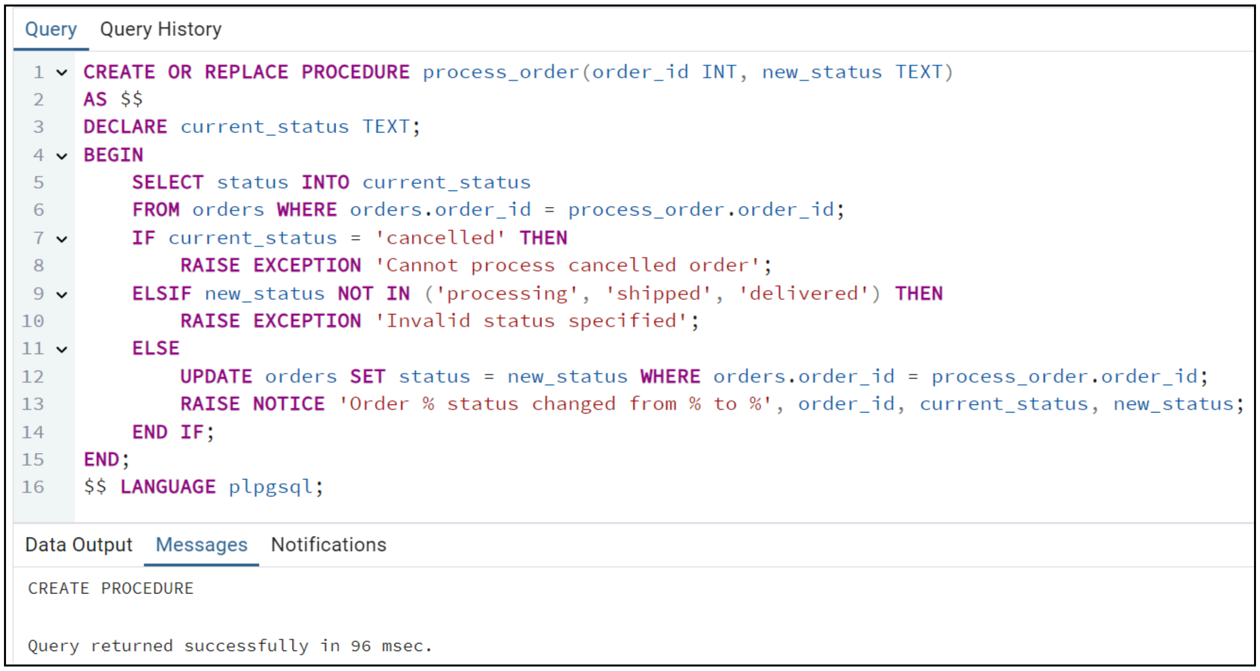
	order_id integer	order_date timestamp without time zone	total_amount numeric	status character varying
1	1	2023-05-15 10:30:00	799.99	delivered
2	4	2023-05-18 16:20:00	79.97	pending
3	5	2025-06-25 21:11:26.728029	39.98	pending


Query Query History

```
SELECT * FROM get_customer_orders(3);
```

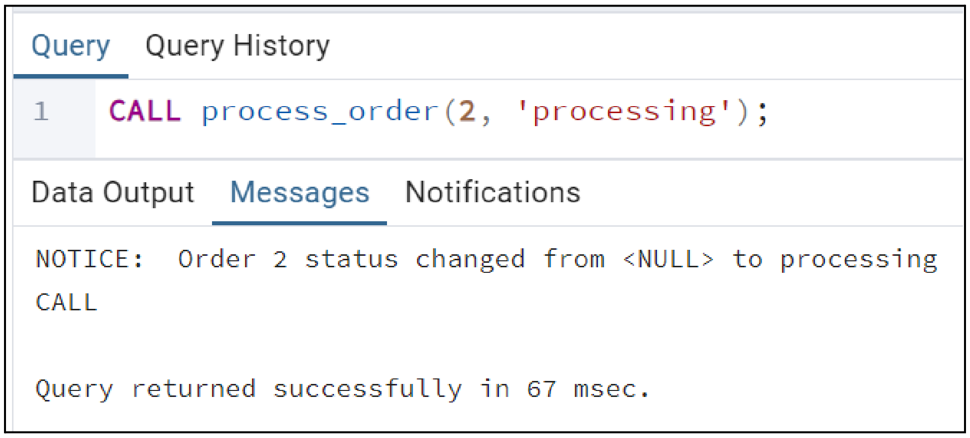
Data Output Messages Notifications

	order_id integer	order_date timestamp without time zone	total_amount numeric	status character varying
1	3	2023-05-17 09:45:00	59.99	processing



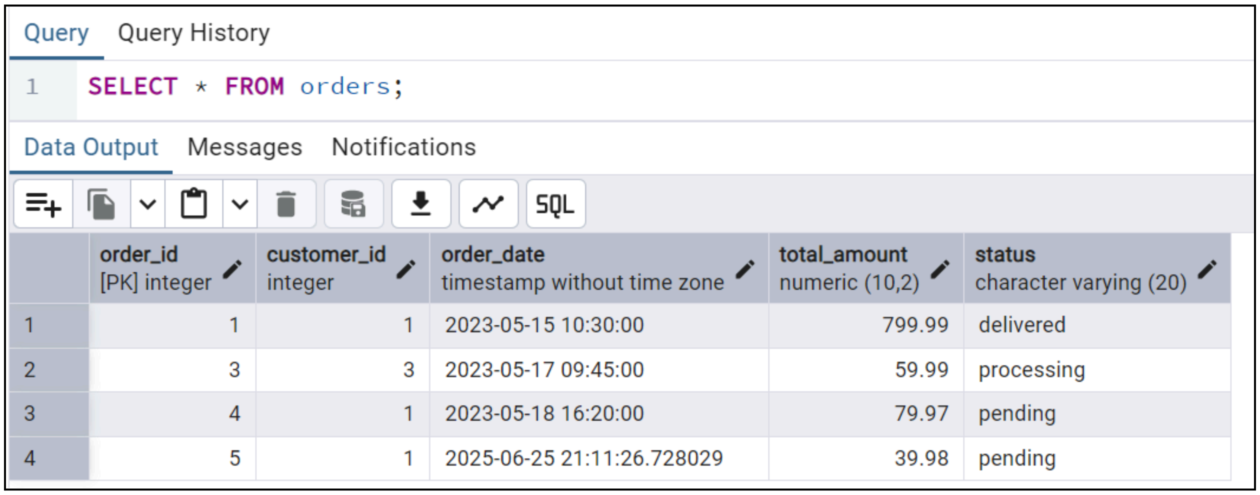
```
CREATE OR REPLACE PROCEDURE process_order(order_id INT, new_status TEXT)
AS $$
DECLARE current_status TEXT;
BEGIN
    SELECT status INTO current_status
    FROM orders WHERE orders.order_id = process_order.order_id;
    IF current_status = 'cancelled' THEN
        RAISE EXCEPTION 'Cannot process cancelled order';
    ELSIF new_status NOT IN ('processing', 'shipped', 'delivered') THEN
        RAISE EXCEPTION 'Invalid status specified';
    ELSE
        UPDATE orders SET status = new_status WHERE orders.order_id = process_order.order_id;
        RAISE NOTICE 'Order % status changed from % to %', order_id, current_status, new_status;
    END IF;
END;
$$ LANGUAGE plpgsql;
```

```
CREATE PROCEDURE

Query returned successfully in 96 msec.
```



```
CALL process_order(2, 'processing');
```

```
NOTICE:  Order 2 status changed from <NULL> to processing
CALL

Query returned successfully in 67 msec.
```



```
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	3	3	2023-05-17 09:45:00	59.99	processing
3	4	1	2023-05-18 16:20:00	79.97	pending
4	5	1	2025-06-25 21:11:26.728029	39.98	pending