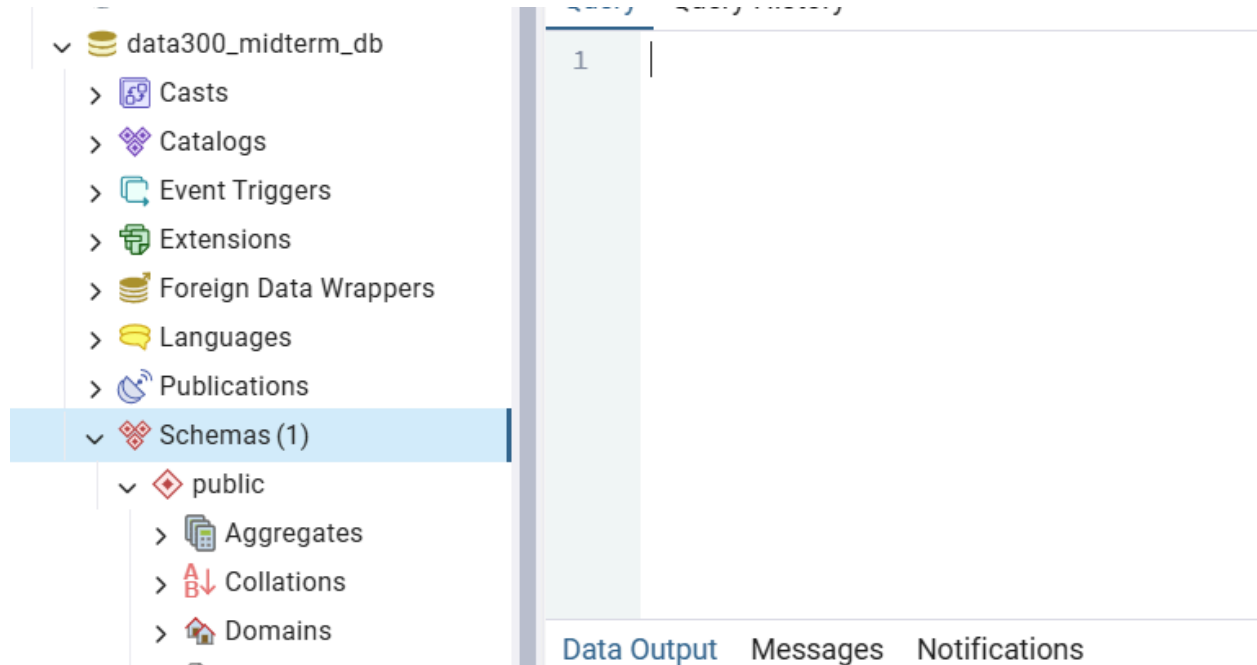


Name: Min Thant Hein  
ID: PIUS20230001

Created the database “data300\_midterm\_db”:



#

##

## Creating tables:

### Creating Customer table:

The screenshot shows a database management interface. On the left is a sidebar with a tree view of database objects. The 'public' schema is expanded, and the 'customer' table is selected under the 'Tables (1)' category. The main panel displays a SQL query editor with the following code:

```
1 -- 1) Customer
2 CREATE TABLE Customer (
3     customer_id SERIAL PRIMARY KEY,
4     full_name VARCHAR(100) NOT NULL,
5     email VARCHAR(100) UNIQUE NOT NULL,
6     phone VARCHAR(20) UNIQUE,
7     date_of_birth DATE NOT NULL CHECK (date_of_birth <= CURRENT_DATE - INTERVAL '16 years'),
8     created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP NOT NULL,
9     CONSTRAINT created_at_immutable CHECK (created_at IS NOT NULL)
10 );
11
12 -- 2) Product table
13 CREATE TABLE Product (
14     product_id INT PRIMARY KEY,
```

Below the query editor, the 'Data Output' tab is active, showing the message: 'Query returned successfully in 206 msec.'

#

## Creating Product table:

The screenshot shows a database management tool interface. On the left, a tree view displays the database structure for 'data300\_midterm\_db'. The 'public' schema is expanded, showing various database objects. The 'Tables (2)' folder is selected, showing 'customer' and 'product' tables. The 'product' table is highlighted. The main pane shows the SQL query for creating the 'Product' table. The query is as follows:

```
-- 2) Product table
CREATE TABLE Product (
    product_id INT PRIMARY KEY,
    sku VARCHAR(50) UNIQUE NOT NULL,
    name VARCHAR(200) NOT NULL,
    unit_price DECIMAL(10,2) NOT NULL CHECK (unit_price > 0),
    stock_qty INT NOT NULL CHECK (stock_qty >= 0),
    status VARCHAR(15) NOT NULL CHECK (status IN ('ACTIVE', 'DISCONTINUED')),
    CONSTRAINT discontinued_stock_zero CHECK (
        (status = 'DISCONTINUED' AND stock_qty = 0) OR status = 'ACTIVE'
    )
);
```

Below the query editor, the 'Messages' tab is active, showing the execution result: 'CREATE TABLE' and 'Query returned successfully in 49 msec.'

#

## Creating "Order" Table:

The screenshot shows the same database management tool interface. The 'Tables (3)' folder is selected, showing 'customer', 'product', and 'Order' tables. The 'Order' table is highlighted. The main pane shows the SQL query for creating the 'Order' table. The query is as follows:

```
-- 3) Order Table
CREATE TABLE "Order" (
    order_id SERIAL PRIMARY KEY,
    customer_id INT NOT NULL REFERENCES Customer(customer_id) ON DELETE RESTRICT,
    order_status VARCHAR(15) DEFAULT 'PENDING' NOT NULL CHECK (order_status IN ('PENDING', 'PAI',
    payment_method VARCHAR(10) NOT NULL CHECK (payment_method IN ('CARD', 'TRANSFER', 'COD')),
    ordered_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP NOT NULL,
    shipping_country CHAR(2) NOT NULL,
    shipping_city VARCHAR(100) NOT NULL
);
```

Below the query editor, the 'Messages' tab is active, showing the execution result: 'CREATE TABLE' and 'Query returned successfully in 50 msec.'

#

## Creating OrderItem Table:

The screenshot shows a database management interface. On the left is a tree view of database objects. The 'Tables (4)' folder is expanded, showing 'Order', 'customer', 'orderitem', and 'product'. The 'customer' table is currently selected. The main area on the right is a query editor with a toolbar at the top. The 'Query' tab is active, displaying a SQL script to create the 'OrderItem' table. The script includes foreign key references to the 'Order' and 'Product' tables, a primary key on 'order\_id' and 'line\_no', and a unique constraint on 'order\_id' and 'product\_id'. Below the query editor, the 'Data Output' tab is selected, showing the message 'Query returned successfully in 77 msec.'

```
35
36 -- 4)OrderItem
37 CREATE TABLE OrderItem (
38     order_id INT NOT NULL REFERENCES "Order"(order_id) ON DELETE CASCADE,
39     line_no INT NOT NULL CHECK (line_no >= 1),
40     product_id INT NOT NULL REFERENCES Product(product_id) ON DELETE RESTRICT,
41     quantity INT NOT NULL CHECK (quantity >= 1),
42     unit_price_at_order DECIMAL(10,2) NOT NULL CHECK (unit_price_at_order > 0),
43     PRIMARY KEY (order_id, line_no),
44     CONSTRAINT unique_product_per_order UNIQUE (order_id, product_id)
45 );
46
47
48
```

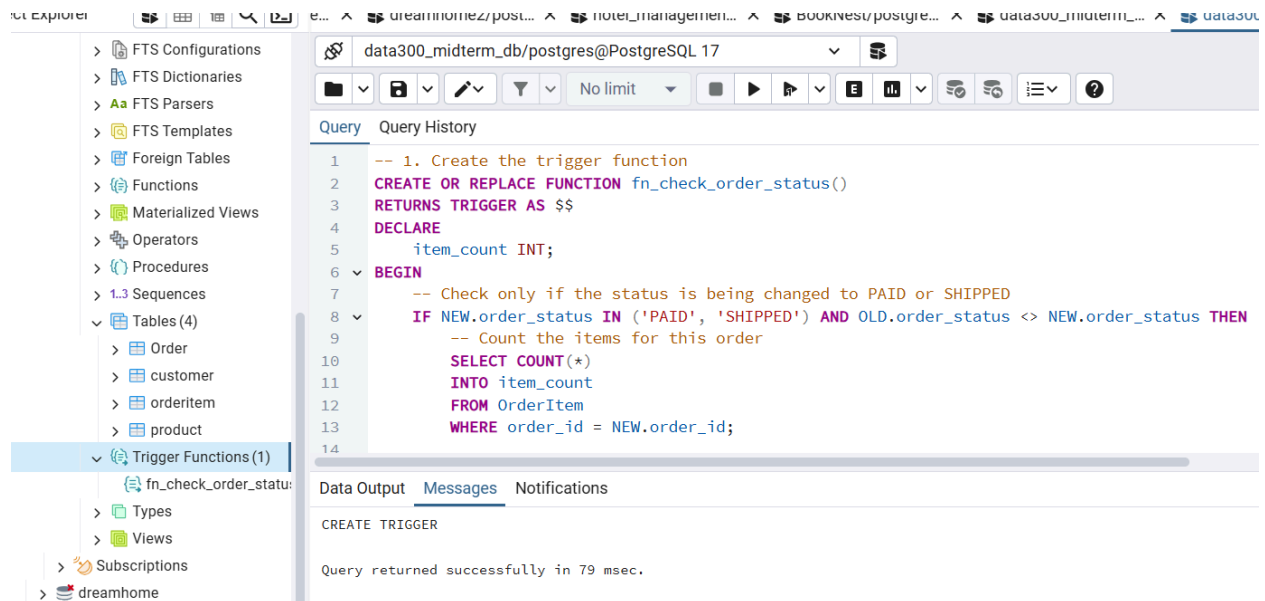
Query returned successfully in 77 msec.

#

## Cross-table business rules to enforce:

Running the trigger function, “fn\_check\_order\_status”:

This trigger handles the UPDATE key on the Order table!



To prevent deleting the last item from a 'PAID' or 'SHIPPED' order, I also add a similar trigger on the OrderItem table.

#

Running the trigger for rule2 and rule3 for OrderItem table:

The screenshot shows a database management tool interface. On the left is a tree view of database objects. The 'Trigger Functions (2)' folder is expanded, showing 'fn\_check\_order\_statu:' and 'fn\_check\_orderitem\_r'. The main pane displays a SQL query with line numbers 1 through 15. The query creates a trigger function 'fn\_check\_orderitem\_rules()' that returns a trigger. It declares variables 'v\_stock\_qty' and 'v\_product\_status', and includes a 'BEGIN' block with a 'SELECT' statement to get product info. The query is partially cut off at line 15. Below the query, the 'Messages' tab is active, showing the message 'CREATE TRIGGER' and 'Query returned successfully in 53 msec.'

Aggregates  
Collations  
Domains  
FTS Configurations  
FTS Dictionaries  
FTS Parsers  
FTS Templates  
Foreign Tables  
Functions  
Materialized Views  
Operators  
Procedures  
Sequences  
Tables (4)  
Order  
customer  
orderitem  
product  
Trigger Functions (2)  
fn\_check\_order\_statu:  
fn\_check\_orderitem\_r

Query Query History

```
1 -- 1. Create the trigger function
2 CREATE OR REPLACE FUNCTION fn_check_orderitem_rules()
3 RETURNS TRIGGER AS $$
4 DECLARE
5     v_stock_qty INT;
6     v_product_status VARCHAR(20);
7 BEGIN
8     -- Get product info
9     SELECT stock_qty, product_status
10    INTO v_stock_qty, v_product_status
11    FROM Product
12    WHERE product_id = NEW.product_id;
13
14     -- Rule 3: Check if product is DISCONTINUED (only on INSERT)
15 IF (TG_OP = 'INSERT' AND v_product_status = 'DISCONTINUED') THEN
```

Data Output Messages Notifications

CREATE TRIGGER

Query returned successfully in 53 msec.

#

Inserting Data:

Insertion Data into Customer table:

Catalogs

Event Triggers

Extensions

Foreign Data Wrappers

Languages

Publications

Schemas (1)

public

Aggregates

Collations

Domains

FTS Configurations

FTS Dictionaries

FTS Parsers

FTS Templates

Foreign Tables

Functions

Materialized Views

Operators

Procedures

Sequences

Query

Query History

1

2

3

4

5

6

INSERT INTO Customer (customer\_id, full\_name, email, phone, date\_of\_birth, created\_at)

VALUES

(1, 'Alice Tan', 'alice.tan@example.com', '+66812345678', '1998-04-15', CURRENT\_TIMESTAMP),

(2, 'David Lee', 'david.lee@example.com', '+66987654321', '1995-09-02', CURRENT\_TIMESTAMP),

(3, 'Maya Wong', 'maya.wong@example.com', NULL, '2000-01-25', CURRENT\_TIMESTAMP);

Data Output

Messages

Notifications

INSERT 0 3

Query returned successfully in 54 msec.

Table view:

Languages

Publications

Schemas (1)

public

Aggregates

Collations

Domains

FTS Configurations

FTS Dictionaries

FTS Parsers

FTS Templates

Foreign Tables

Functions

Materialized Views

Operators

Procedures

Sequences

Tables (4)

Order

customer

orderitem

product

Trigger Functions (2)

Query

Query History

1

2

SELECT \* FROM public.customer

ORDER BY customer\_id ASC

Data Output

Messages

Notifications

Showing rows: 1 to 3

Page 1

	customer_id	full_name	email	phone	date_of_birth	created_at
	[PK] integer	character varying (100)	character varying (100)	character varying (20)	date	timestamp without time zone
1	1	Alice Tan	alice.tan@example.com	+66812345678	1998-04-15	2025-10-30 20:18:18.22431
2	2	David Lee	david.lee@example.com	+66987654321	1995-09-02	2025-10-30 20:18:18.22431
3	3	Maya Wong	maya.wong@example.com	[null]	2000-01-25	2025-10-30 20:18:18.22431

###

Insertion Data into Product table:

The screenshot displays the SQL Server Enterprise Manager interface. On the left, the 'public' schema is expanded, showing various database objects. The 'Tables (4)' folder is selected, and the 'product' table is highlighted. The central pane shows the execution of an SQL INSERT statement. The 'Query' tab is active, displaying the following code:

```
1 INSERT INTO Product (product_id, sku, name, unit_price, stock_qty, status)
2 VALUES
3 (101, 'BK001', 'Python for Beginners', 350.00, 25, 'ACTIVE'),
4 (102, 'BK002', 'Database Design Essentials', 420.00, 15, 'ACTIVE'),
5 (103, 'BK003', 'Advanced SQL Queries', 500.00, 0, 'DISCONTINUED'),
6 (104, 'BK004', 'AI in Practice', 650.00, 10, 'ACTIVE');
7
```

The 'Messages' tab is also active, showing the execution results:

```
INSERT 0 4
Query returned successfully in 47 msec.
```

#

Table view:



Query

Query History

1

SELECT \* FROM public.product

2

ORDER BY product\_id ASC

Data Output

Messages

Notifications

Showing rows: 1 to 4

	product_id [PK] integer	sku character varying (50)	name character varying (200)	unit_price numeric (10,2)	stock_qty integer	status character varying (15)
1	101	BK001	Python for Beginners	350.00	25	ACTIVE
2	102	BK002	Database Design Essentials	420.00	15	ACTIVE
3	103	BK003	Advanced SQL Queries	500.00	0	DISCONTINUED
4	104	BK004	AI in Practice	650.00	10	ACTIVE

##

Insertion Data into Order table:

Languages

Publications

Schemas (1)

public

Aggregates

Collations

Domains

FTS Configurations

FTS Dictionaries

FTS Parsers

FTS Templates

Foreign Tables

Functions

Materialized Views

Operators

Procedures

Sequences

Tables (4)

Order

customer

orderitem

product

Trigger Functions (2)

Query

Query History

```
1 INSERT INTO "Order" (order_id, customer_id, order_status, payment_method, ordered_at, shipping_co
2 VALUES
3 (1001, 1, 'PAID', 'CARD', CURRENT_TIMESTAMP, 'TH', 'Bangkok'),
4 (1002, 2, 'PENDING', 'TRANSFER', CURRENT_TIMESTAMP, 'TH', 'Chiang Mai'),
5 (1003, 3, 'SHIPPED', 'CARD', CURRENT_TIMESTAMP, 'TH', 'Phuket');
6
```

Data Output

Messages

Notifications

INSERT 0 3

Query returned successfully in 47 msec.

Table view:

Query

Query History

```
1 SELECT * FROM public."Order"
2 ORDER BY order_id ASC
```

Data Output

Messages

Notifications

Showing rows: 1 to 3

Page No: 1

	order_id [PK] integer	customer_id integer	order_status character varying (15)	payment_method character varying (10)	ordered_at timestamp without time zone	shipping_country character (2)	shipping_city character varying (100)
1	1001	1	PAID	CARD	2025-10-30 20:21:41.091083	TH	Bangkok
2	1002	2	PENDING	TRANSFER	2025-10-30 20:21:41.091083	TH	Chiang Mai
3	1003	3	SHIPPED	CARD	2025-10-30 20:21:41.091083	TH	Phuket

##

Insertion Data into OrderItem table:

The screenshot displays a PostgreSQL client interface with the following components:

- Left Panel (Schema Explorer):** Shows the database structure. The 'Schemas (1)' section is expanded, revealing the 'public' schema. Under 'public', several database objects are listed, including 'Tables (4)', which contains 'Order', 'customer', 'orderitem', and 'product'.
- Top Bar:** Indicates the current database is 'data300\_midterm\_db/postgres@PostgreSQL 17'. It includes standard icons for file operations, query execution, and settings.
- Query Editor:** Contains the following SQL query:

```
1 INSERT INTO OrderItem (order_id, line_no, product_id, quantity, unit_price_at_order)
2 VALUES
3 (1001, 1, 101, 2, 350.00),
4 (1001, 2, 102, 1, 420.00),
5 (1002, 1, 104, 1, 650.00),
6 (1003, 1, 101, 1, 350.00),
7 (1003, 2, 102, 2, 420.00);
8
```
- Bottom Panel:** Features three tabs: 'Data Output', 'Messages', and 'Notifications'. The 'Messages' tab is active, showing the message 'INSERT 0 5' and a status message: 'Query returned successfully in 48 msec.'

Table view:

