Northern University

ASSIGNMENT

Department: Department of Computer Science & Engineering

Program: ECSE

ID: 42230301042 & 41

Section: 3A

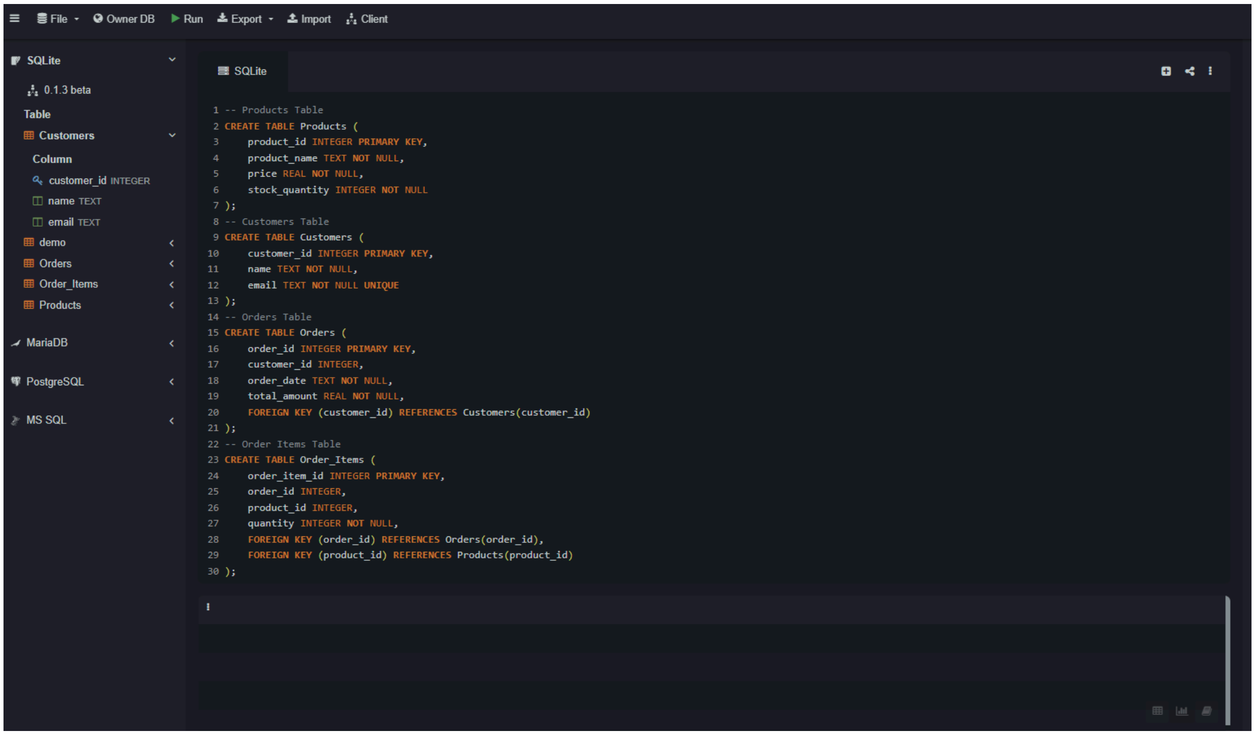
Course Code: CSE 2367

Subject: Database Management System.

Assignment Name: E-commerce Product Catalog.

Create Tables

```
-- Products Table
CREATE TABLE Products (
    product_id INTEGER PRIMARY KEY,
    product_name TEXT NOT NULL,
    price REAL NOT NULL,
    stock_quantity INTEGER NOT NULL
);



-- Customers Table
CREATE TABLE Customers (
    customer_id INTEGER PRIMARY KEY,
    name TEXT NOT NULL,
    email TEXT NOT NULL UNIQUE
);



-- Orders Table
CREATE TABLE Orders (
    order_id INTEGER PRIMARY KEY,
    customer_id INTEGER,
    order_date TEXT NOT NULL,
    total_amount REAL NOT NULL,
    FOREIGN KEY (customer_id) REFERENCES Customers(customer_id)
);




-- Order Items Table
CREATE TABLE Order_Items (
    order_item_id INTEGER PRIMARY KEY,
    order_id INTEGER,
    product_id INTEGER,
    quantity INTEGER NOT NULL,
    FOREIGN KEY (order_id) REFERENCES Orders(order_id),
    FOREIGN KEY (product_id) REFERENCES Products(product_id)
);
```

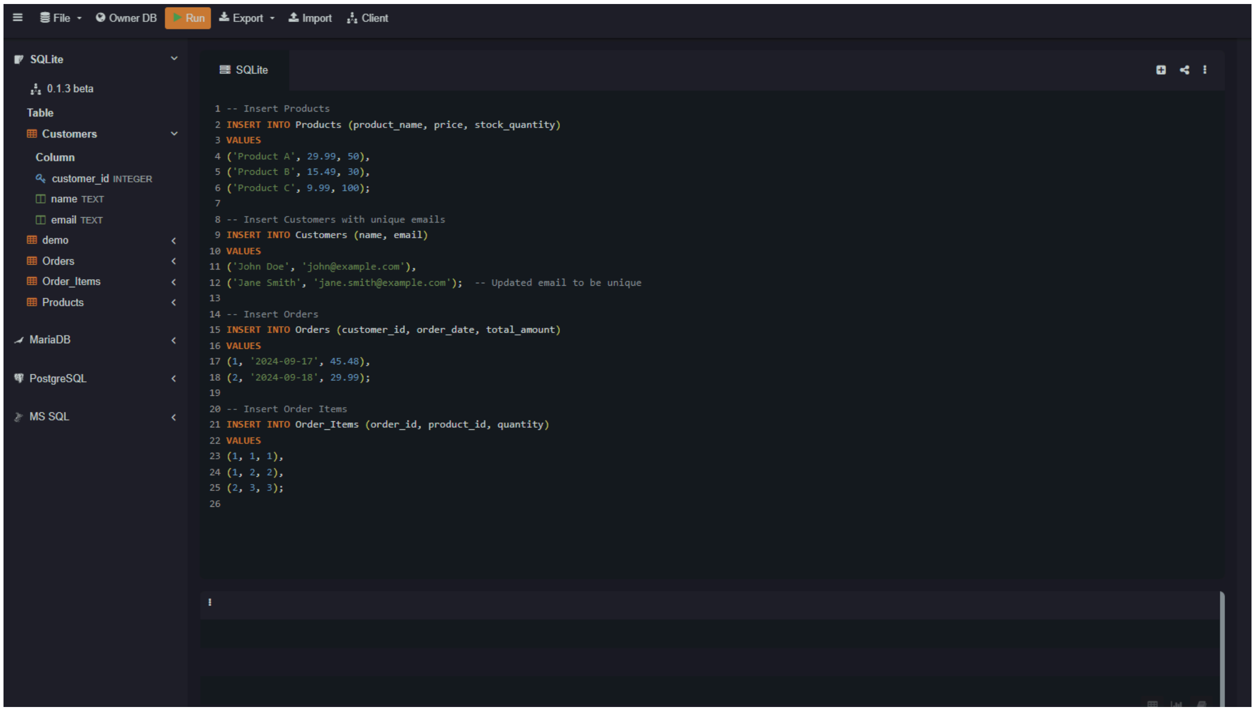


```
-- Products Table
CREATE TABLE Products (
    product_id INTEGER PRIMARY KEY,
    product_name TEXT NOT NULL,
    price REAL NOT NULL,
    stock_quantity INTEGER NOT NULL
);
-- Customers Table
CREATE TABLE Customers (
    customer_id INTEGER PRIMARY KEY,
    name TEXT NOT NULL,
    email TEXT NOT NULL UNIQUE
);
-- Orders Table
CREATE TABLE Orders (
    order_id INTEGER PRIMARY KEY,
    customer_id INTEGER,
    order_date TEXT NOT NULL,
    total_amount REAL NOT NULL,
    FOREIGN KEY (customer_id) REFERENCES Customers(customer_id)
);
-- Order Items Table
CREATE TABLE Order_Items (
    order_item_id INTEGER PRIMARY KEY,
    order_id INTEGER,
    product_id INTEGER,
    quantity INTEGER NOT NULL,
    FOREIGN KEY (order_id) REFERENCES Orders(order_id),
    FOREIGN KEY (product_id) REFERENCES Products(product_id)
);
```

Insert Sample Data

```
-- Insert Products
INSERT INTO Products (product_name, price, stock_quantity)
VALUES
('Product A', 29.99, 50),
('Product B', 15.49, 30),
('Product C', 9.99, 100);
-- Insert Customers with unique emails
INSERT INTO Customers (name, email)
VALUES
('John Doe', 'john@example.com'),
('Jane Smith', 'jane.smith@example.com');  -- Updated email to be unique
-- Insert Orders
INSERT INTO Orders (customer_id, order_date, total_amount)
VALUES
(1, '2024-09-17', 45.48),
(2, '2024-09-18', 29.99);
-- Insert Order Items
INSERT INTO Order_Items (order_id, product_id, quantity)
VALUES
```

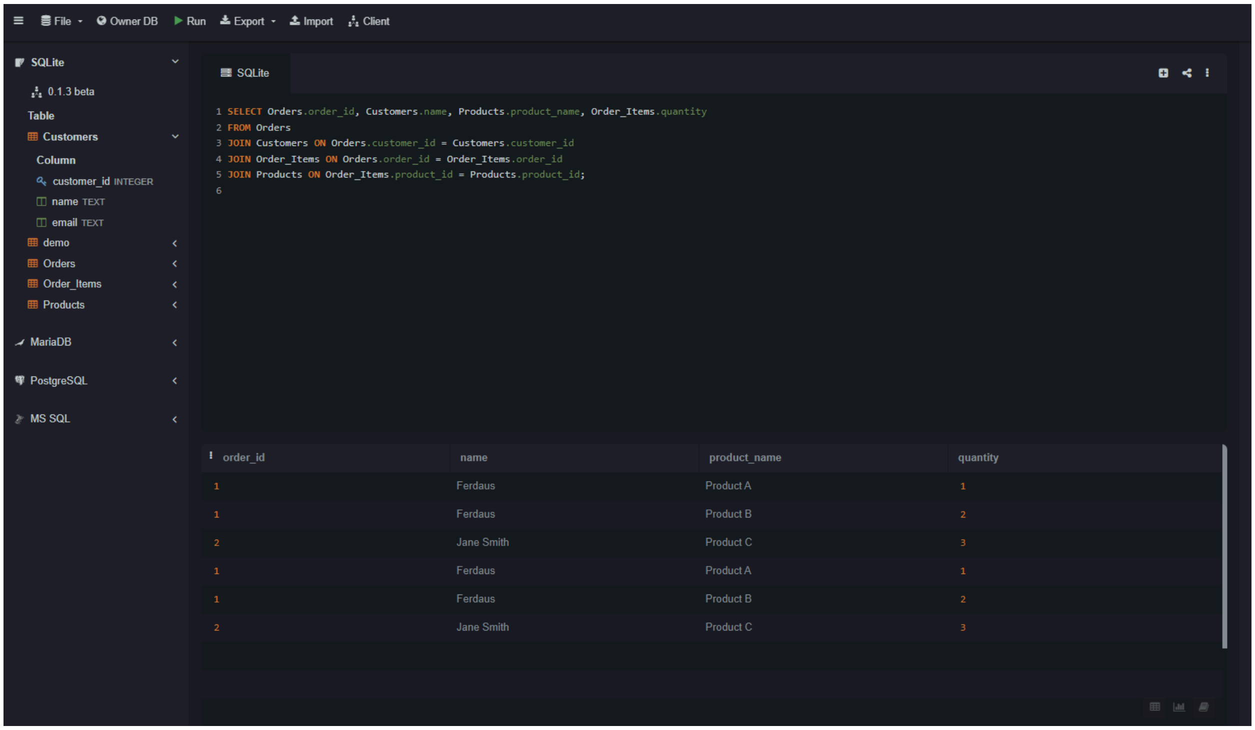
(1, 1, 1),
(1, 2, 2),
(2, 3, 3);



```sql
-- Insert Products
INSERT INTO Products (product_name, price, stock_quantity)
VALUES
('Product A', 29.99, 50),
('Product B', 15.49, 30),
('Product C', 9.99, 100);

-- Insert Customers with unique emails
INSERT INTO Customers (name, email)
VALUES
('John Doe', 'john@example.com'),
('Jane Smith', 'jane.smith@example.com');  -- Updated email to be unique

-- Insert Orders
INSERT INTO Orders (customer_id, order_date, total_amount)
VALUES
(1, '2024-09-17', 45.48),
(2, '2024-09-18', 29.99);

-- Insert Order Items
INSERT INTO Order_Items (order_id, product_id, quantity)
VALUES
(1, 1, 1),
(1, 2, 2),
(2, 3, 3);
```

Retrieve Orders with Products

-- Retrieve Orders with Products (JOIN Example)
SELECT Orders.order_id, Customers.name, Products.product_name, Order_Items.quantity
FROM Orders
JOIN Customers ON Orders.customer_id = Customers.customer_id
JOIN Order_Items ON Orders.order_id = Order_Items.order_id
JOIN Products ON Order_Items.product_id = Products.product_id;

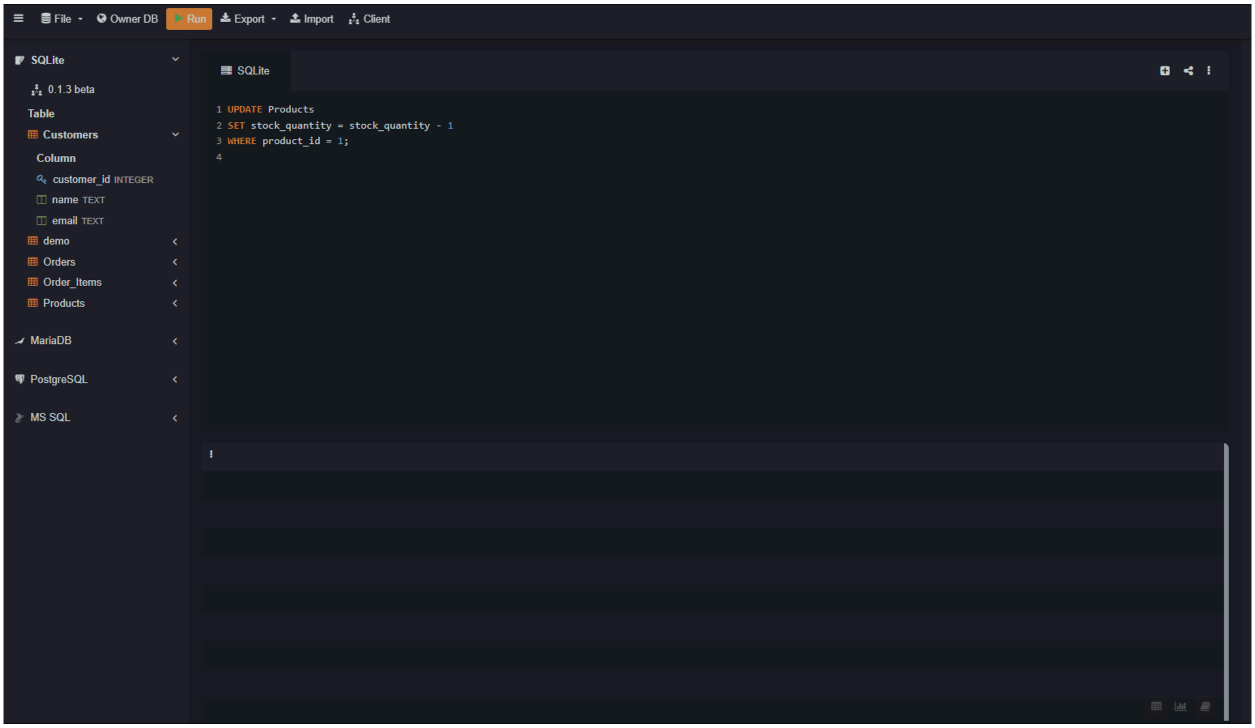


```
SELECT Orders.order_id, Customers.name, Products.product_name, Order_Items.quantity
FROM Orders
JOIN Customers ON Orders.customer_id = Customers.customer_id
JOIN Order_Items ON Orders.order_id = Order_Items.order_id
JOIN Products ON Order_Items.product_id = Products.product_id;
```

order_id	name	product_name	quantity
1	Ferdaus	Product A	1
1	Ferdaus	Product B	2
2	Jane Smith	Product C	3
1	Ferdaus	Product A	1
1	Ferdaus	Product B	2
2	Jane Smith	Product C	3

Update Stock

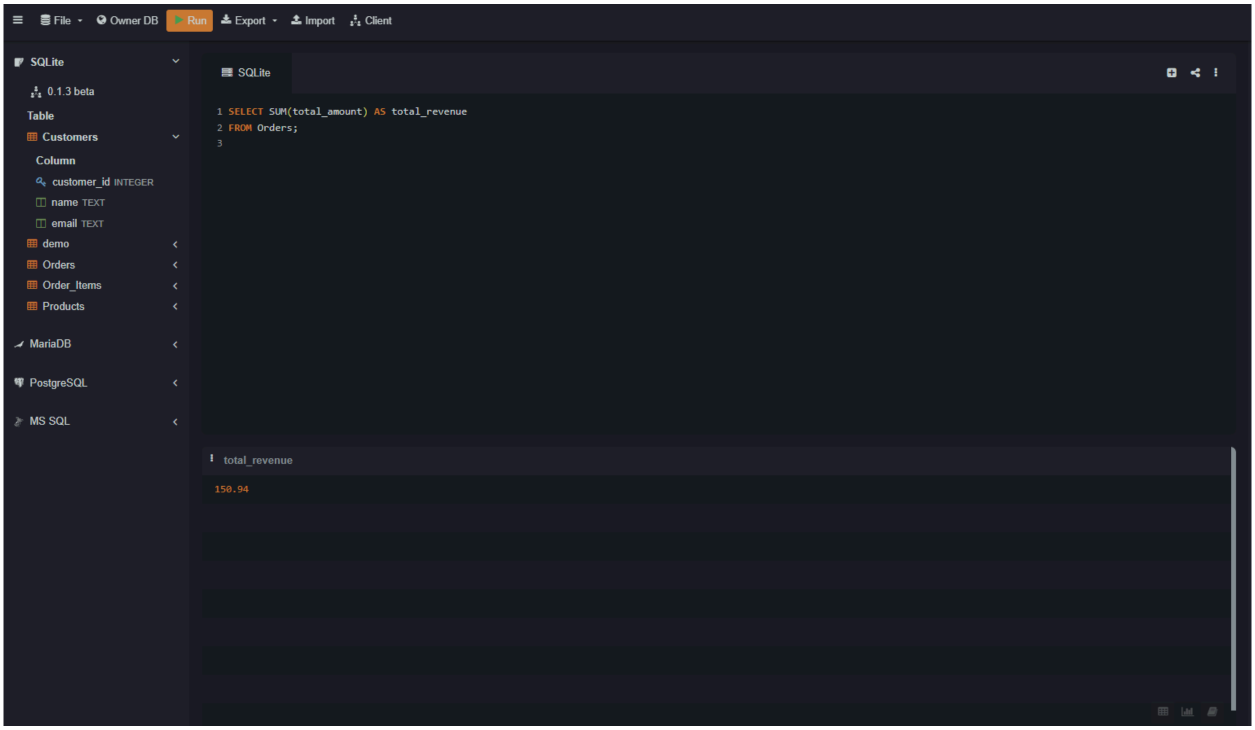
```
-- Update Stock After an Order
UPDATE Products
SET stock_quantity = stock_quantity - 1
WHERE product_id = 1;
```



```sql
UPDATE Products
SET stock_quantity = stock_quantity - 1
WHERE product_id = 1;
```

Calculate Total Revenue

```
-- Calculate Total Revenue
SELECT SUM(total_amount) AS total_revenue
FROM Orders;
```



```sql
SELECT SUM(total_amount) AS total_revenue
FROM Orders;
```

```
total_revenue
150.94
```