This SQL code creates tables for products, customers, orders, and order items. It also inserts sample data to demonstrate the queries. The queries retrieve order information with customer details, calculate total revenue by category, and find the top 5 products with the highest revenue. This e-commerce system provides essential functionalities for managing products, customers, orders, and analyzing sales data.

```
-- Create tables
CREATE TABLE products (
    product_id INT PRIMARY KEY,
    product_name VARCHAR(100) NOT NULL,
    category VARCHAR(50),
    price DECIMAL(10, 2) NOT NULL,
    stock_quantity INT NOT NULL
);

CREATE TABLE customers (
    customer_id INT PRIMARY KEY,
    first_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    address VARCHAR(200) NOT NULL,
    city VARCHAR(50) NOT NULL,
    country VARCHAR(50) NOT NULL
);

CREATE TABLE orders (
    order_id INT PRIMARY KEY,
    customer_id INT NOT NULL,
    order_date DATE NOT NULL,
    total_amount DECIMAL(10, 2) NOT NULL,
    FOREIGN KEY (customer_id) REFERENCES customers (customer_id)
```

```
);

CREATE TABLE order_items (
    order_item_id INT PRIMARY KEY,
    order_id INT NOT NULL,
    product_id INT NOT NULL,
    quantity INT NOT NULL,
    item_price DECIMAL(10, 2) NOT NULL,
    total_price DECIMAL(10, 2) NOT NULL,
    FOREIGN KEY (order_id) REFERENCES orders (order_id),
    FOREIGN KEY (product_id) REFERENCES products (product_id)
);

-- Insert sample data
INSERT INTO products (product_id, product_name, category, price, stock_quantity)
VALUES
    (1, 'Laptop', 'Electronics', 800.00, 20),
    (2, 'Smartphone', 'Electronics', 600.00, 30),
    (3, 'Headphones', 'Electronics', 100.00, 50),
    (4, 'T-shirt', 'Clothing', 20.00, 100),
    (5, 'Jeans', 'Clothing', 50.00, 80);

INSERT INTO customers (customer_id, first_name, last_name, email, address, city, country)
VALUES
    (1, 'John', 'Doe', 'john.doe@example.com', '123 Main St', 'New York', 'USA'),
    (2, 'Jane', 'Smith', 'jane.smith@example.com', '456 Elm St', 'Los Angeles', 'USA'),
    (3, 'Michael', 'Johnson', 'michael.johnson@example.com', '789 Oak St', 'Chicago', 'USA');

-- Let's assume we have two orders
```

```
INSERT INTO orders (order_id, customer_id, order_date, total_amount)
VALUES
  (1, 1, '2023-08-01', 1400.00),
  (2, 2, '2023-08-01', 200.00);

-- Add order items for Order 1
INSERT INTO order_items (order_item_id, order_id, product_id, quantity, item_price, total_price)
VALUES
  (1, 1, 1, 1, 800.00, 800.00),
  (2, 1, 3, 2, 100.00, 200.00),
  (3, 1, 4, 3, 20.00, 60.00);

-- Add order items for Order 2
INSERT INTO order_items (order_item_id, order_id, product_id, quantity, item_price, total_price)
VALUES
  (4, 2, 4, 5, 20.00, 100.00),
  (5, 2, 5, 2, 50.00, 100.00);

-- Query to get all orders with customer details
SELECT o.order_id, c.first_name, c.last_name, o.order_date, o.total_amount
FROM orders o
JOIN customers c ON o.customer_id = c.customer_id;

-- Query to get the total revenue generated by each category
SELECT p.category, SUM(oi.total_price) AS total_revenue
FROM products p
JOIN order_items oi ON p.product_id = oi.product_id
GROUP BY p.category;
```

```
-- Query to get the top 5 products with the highest revenue
SELECT p.product_name, p.category, SUM(oi.total_price) AS total_revenue
FROM products p
JOIN order_items oi ON p.product_id = oi.product_id
GROUP BY p.product_id, p.product_name, p.category
ORDER BY total_revenue DESC
LIMIT 5;
```