ASSIGNMENT 2

```
use l7tutorial;

CREATE TABLE customers (
   CustomerID INT PRIMARY KEY,
   Name VARCHAR(100) NOT NULL,
   Email VARCHAR(100) NOT NULL UNIQUE,
   City VARCHAR(50),
   SignupDate DATE NOT NULL
);

CREATE TABLE orders (
   OrderID INT PRIMARY KEY,
   CustomerID INT NOT NULL,
   OrderDate DATE NOT NULL,
   TotalAmount DECIMAL(10, 2) NOT NULL,
   FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
);

CREATE TABLE products (
   ProductID INT PRIMARY KEY,
   ProductName VARCHAR(100) NOT NULL,
   Category VARCHAR(50),
   Price DECIMAL(10, 2) NOT NULL
);

CREATE TABLE orderDetails (
   OrderDetailID INT PRIMARY KEY,
   OrderID INT NOT NULL,
    ProductID INT NOT NULL,

    Quantity INT NOT NULL,

    Price DECIMAL(10, 2) NOT NULL,

    FOREIGN KEY (OrderID) REFERENCES Orders(OrderID),

    FOREIGN KEY (ProductID) REFERENCES Products(ProductID)

);


INSERT INTO customers (CustomerID, Name, Email, City, SignupDate) VALUES

(1, 'Anita Chadwell', 'achadwell0@ebay.co.uk', 'Haljala', '2025-02-21'),

(2, 'Seka Arnell', 'sarnell1@nsw.gov.au', 'Ban Ko Lan', '2025-02-15'),

(3, 'Jewell Agneau', 'jagneau2@photobucket.com', 'Don Tan', '2025-04-09'),

(4, 'Thacher Riseley', 'triseley3@joomla.org', 'Mumbai', '2025-06-01');



INSERT INTO products (ProductID, ProductName, Category, Price) VALUES

(101, 'Zephyrus G15', 'Electronics', 1200.00),

(102, 'Samsung S25', 'Electronics', 800.00),

(103, 'Sennheiser', 'Electronics', 750.00),

(104, 'Corelle Plate', 'Household', 70.00),

(105, 'Notebook', 'Stationery', 5.00),

(106, 'BMW M7', 'Vehicle', 150000.00);


INSERT INTO orders (OrderID, CustomerID, OrderDate, TotalAmount) VALUES

(1001, 1, '2025-03-01', 1200.00),

(1002, 1, '2025-03-05', 70.00),

(1003, 2, '2025-03-02', 800.00),

(1004, 3, '2025-04-10', 15.00),

(1005, 1, '2025-05-28', 750.00),

(1006, 4, '2025-06-02', 150000.00);


INSERT INTO orderDetails (OrderDetailID, OrderID, ProductID, Quantity, Price) VALUES
```

(1, 1001, 101, 1, 1200.00),

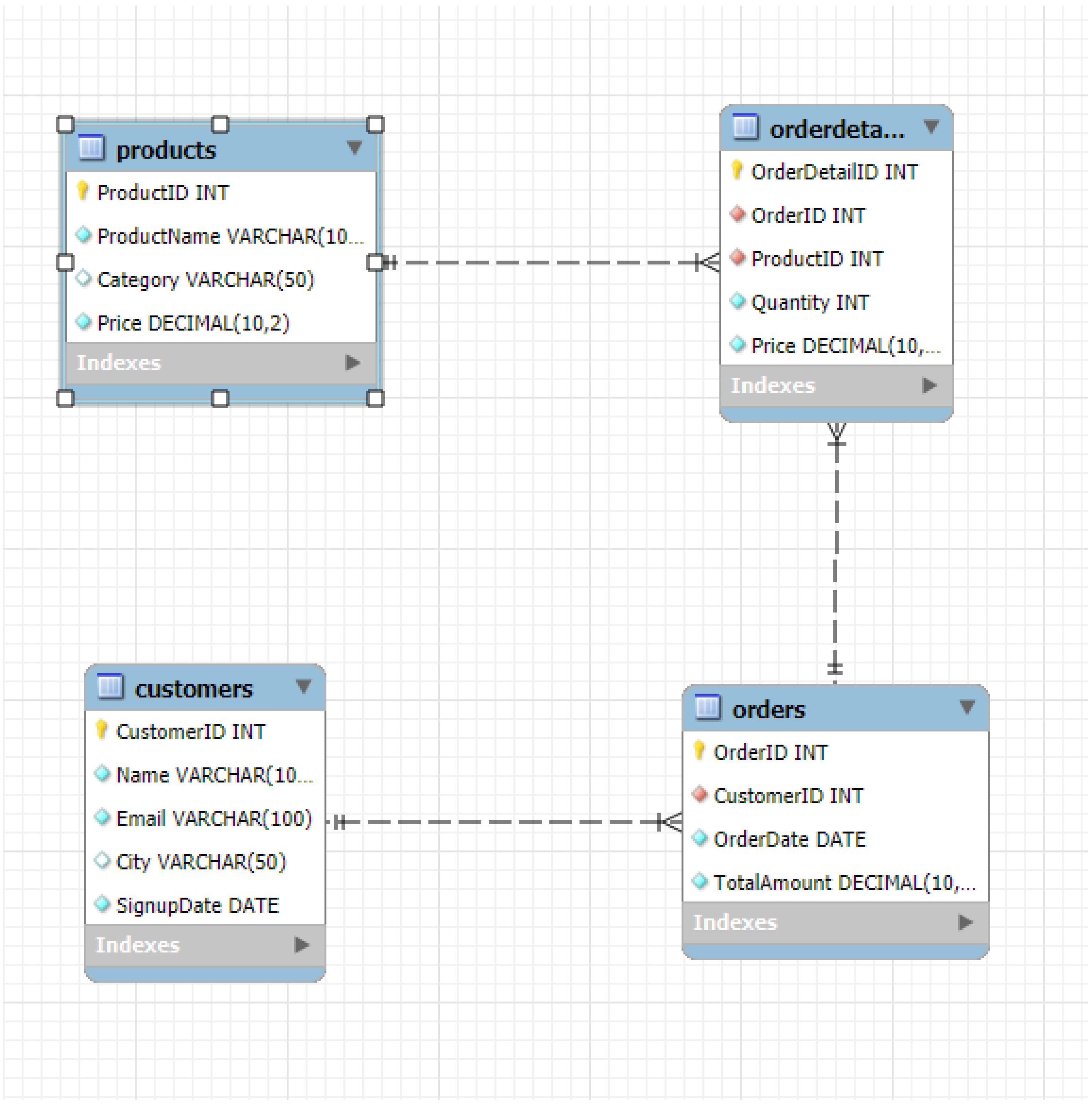
(2, 1002, 104, 1, 70.00),

(3, 1003, 102, 1, 800.00),

(4, 1004, 105, 1, 5.00),

(5, 1005, 103, 1, 750.00),

(6, 1006, 106, 1, 150000.00);



```
use l7tutorial;


-- Basic Queries


-- Get the list of all customers

SELECT * FROM customers;


-- Find all orders placed in the last 30 days

SELECT * FROM orders

	WHERE OrderDate >= DATE_SUB(CURDATE(), INTERVAL 30 DAY);


-- Show product names and their prices

SELECT ProductName, Price FROM products;


-- Find the total number of products in each category

SELECT Category, COUNT(*) AS TotalProducts

	FROM products

	GROUP BY Category;




-- Filtering and Conditions


-- Get all customers from the city 'Mumbai'

SELECT * FROM customers

	WHERE City = 'Mumbai';


-- Find orders with a total amount greater than 5000

SELECT * FROM orders

	WHERE TotalAmount > 5000;


-- List customers who signed up after '2024-01-01'
```

```
SELECT * FROM customers

        WHERE SignupDate > '2024-01-01';




-- Joins


-- Show all orders along with the customer's name

SELECT orders.OrderID, orders.OrderDate, orders.TotalAmount, customers.Name

        FROM orders

        INNER JOIN customers ON orders.CustomerID = customers.CustomerID;


-- List products purchased in each order

SELECT orders.OrderID, products.ProductName, orderDetails.Quantity

        FROM orders

        INNER JOIN orderDetails ON orders.OrderID = orderDetails.OrderID

        INNER JOIN Products ON orderDetails.ProductID = products.ProductID;


-- Find customers who have never placed an order

SELECT customers.*

        FROM customers

        LEFT JOIN Orders ON customers.CustomerID = orders.CustomerID

        WHERE orders.OrderID IS NULL;




-- Aggregation and Grouping


-- Find the total amount spent by each customer

SELECT customers.CustomerID, customers.Name, SUM(orders.TotalAmount) AS TotalSpent

        FROM customers

        LEFT JOIN Orders ON customers.CustomerID =orders.CustomerID

        GROUP BY customers.CustomerID, customers.Name;
```

```
-- Which product has been sold the most (by quantity)? though all of mine were qty 1
SELECT products.ProductName, SUM(orderDetails.Quantity) AS TotalQuantity
        FROM products
        INNER JOIN orderDetails ON products.ProductID = orderDetails.ProductID
        GROUP BY products.ProductName
        ORDER BY TotalQuantity DESC
        LIMIT 1;

-- Find the average order value for each customer
SELECT customers.CustomerID, customers.Name, AVG(orders.TotalAmount) AS AverageOrderValue
        FROM customers
        LEFT JOIN orders ON customers.CustomerID = orders.CustomerID
        GROUP BY customers.CustomerID, customers.Name;

-- Total sales amount per product category
SELECT products.Category, SUM(orderDetails.Quantity * orderDetails.Price) AS TotalSales
        FROM products
        INNER JOIN orderDetails ON products.ProductID = orderDetails.ProductID
        GROUP BY products.Category;



-- Subqueries

-- Find customers who spent more than the average spending
SELECT customers.CustomerID, customers.Name, SUM(orders.TotalAmount) AS TotalSpent
        FROM customers
        INNER JOIN orders ON customers.CustomerID = orders.CustomerID
        GROUP BY customers.CustomerID, customers.Name
        HAVING TotalSpent > (
                SELECT AVG(TotalAmount)
                FROM orders
        );

-- List products that have never been ordered
SELECT products.*
        FROM products
        LEFT JOIN orderDetails ON products.ProductID = orderDetails.ProductID
        WHERE orderDetails.OrderID IS NULL;

-- Find the most recent order for each customer
SELECT customers.CustomerID, customers.Name, orders.OrderID, orders.OrderDate
        FROM customers
        INNER JOIN orders ON customers.CustomerID = orders.CustomerID
        WHERE orders.OrderDate = (
                SELECT MAX(orderDate)
                FROM orders
                WHERE orders.CustomerID = customers.CustomerID
        );


-- Advanced Queries

-- Rank customers by total spending (highest first)
SELECT customers.CustomerID, customers.Name, SUM(orders.TotalAmount) AS TotalSpent
        FROM customers
        LEFT JOIN orders ON customers.CustomerID = orders.CustomerID
        GROUP BY customers.CustomerID, customers.Name
        ORDER BY TotalSpent DESC;

-- Get the top 3 customers based on the number of orders placed
```

```
SELECT customers.CustomerID, customers.Name, COUNT(orders.OrderID) AS TotalOrders
        FROM customers
        LEFT JOIN orders ON customers.CustomerID = orders.CustomerID
        GROUP BY customers.CustomerID, customers.Name
        ORDER BY TotalOrders DESC
        LIMIT 3;

-- For each product, find how many unique customers have purchased it
SELECT products.ProductName, COUNT(DISTINCT orders.CustomerID) AS UniqueCustomers
        FROM products
        INNER JOIN orderDetails ON products.ProductID = orderDetails.ProductID
        INNER JOIN orders ON orderDetails.OrderID = orders.OrderID
        GROUP BY products.ProductName;
```