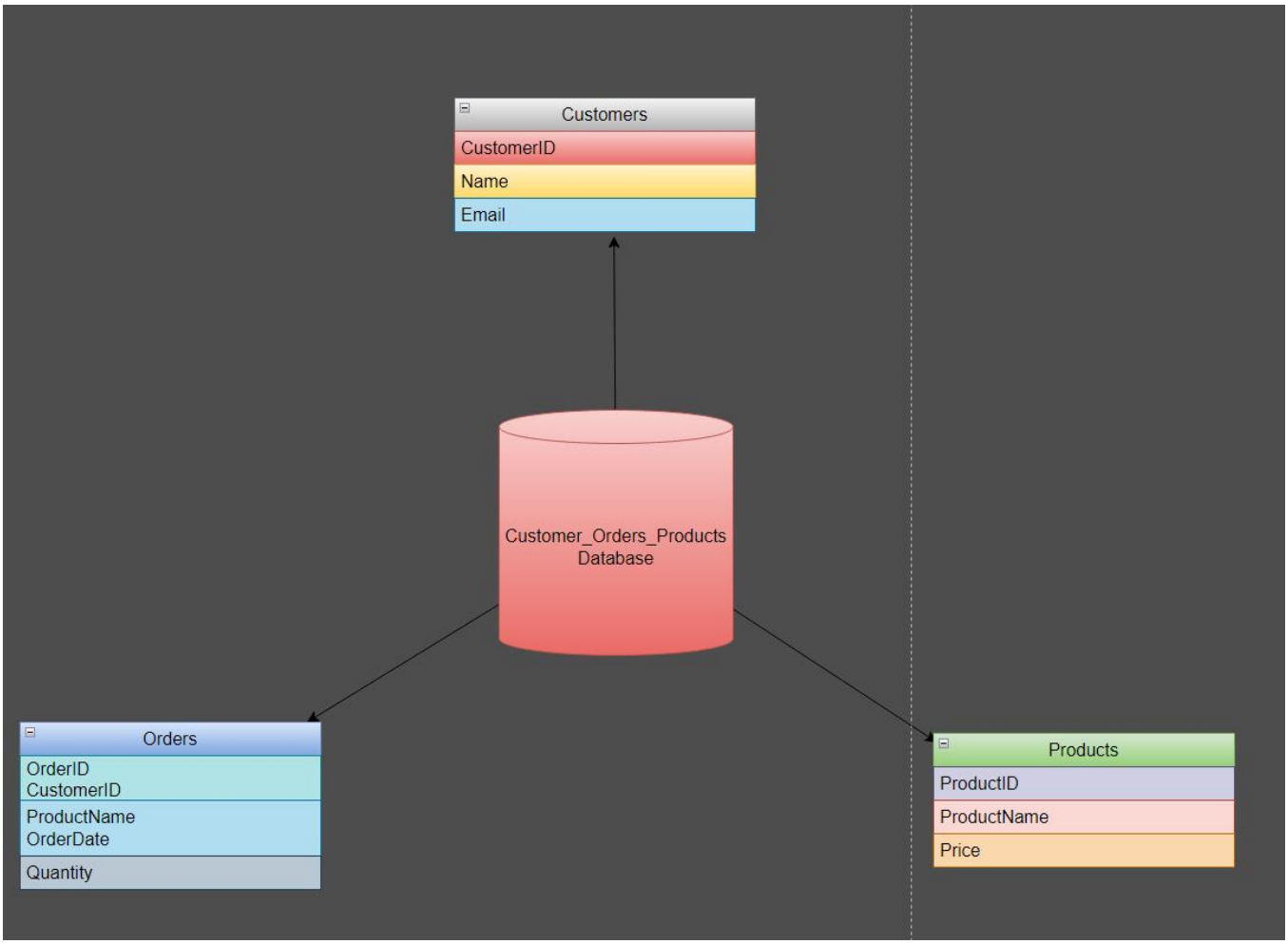
(Customer_Orders_Products Database)



```
CREATE DATABASE Customers_Orders_Products;

USE Customers_Orders_Products;


-- Create the Customers table
CREATE TABLE Customers (
    CustomerID INT PRIMARY KEY,
    Name VARCHAR(50),
    Email VARCHAR(100)
);
-- Create the Orders table
CREATE TABLE Orders (
    OrderID INT PRIMARY KEY,
    CustomerID INT,
    ProductName VARCHAR(50),
    OrderDate DATE,
    Quantity INT,
    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
);
-- Create the Products table
CREATE TABLE Products (
    ProductID INT PRIMARY KEY,
    ProductName VARCHAR(50),
    Price DECIMAL(10, 2)
);
```

```
-- Insert records into the Customers table

INSERT INTO Customers (CustomerID, Name, Email)

VALUES

(1, 'John Doe', 'johndoe@example.com'),

(2, 'Jane Smith', 'janesmith@example.com'),

(3, 'Robert Johnson', 'robertjohnson@example.com'),

(4, 'Emily Brown', 'emilybrown@example.com'),

(5, 'Michael Davis', 'michaeldavis@example.com'),

(6, 'Sarah Wilson', 'sarahwilson@example.com'),

(7, 'David Thompson', 'davidthompson@example.com'),

(8, 'Jessica Lee', 'jessicalee@example.com'),

(9, 'William Turner', 'williamturner@example.com'),

(10, 'Olivia Martinez', 'oliviamartinez@example.com'),

(11, 'James Anderson', 'jamesanderson@example.com'),

(12, 'Kelly Clarkson', 'kellyclarkson@example.com');


-- Insert records into the Products table

INSERT INTO Products (ProductID, ProductName, Price)

VALUES

(1, 'Product A', 10.99),

(2, 'Product B', 8.99),

(3, 'Product C', 5.99),

(4, 'Product D', 12.99),

(5, 'Product E', 7.99),

(6, 'Product F', 6.99),
```

(7, 'Product G', 9.99),

(8, 'Product H', 11.99),

(9, 'Product I', 14.99),

(10, 'Product J', 4.99),

(11, 'Product K', 3.99),

(12, 'Product L', 15.99);

-- Insert records into the Orders table

INSERT INTO Orders (OrderID, CustomerID, ProductName, OrderDate, Quantity)

VALUES

(1, 1, 'Product A', '2023-07-01', 5),

(2, 2, 'Product B', '2023-07-02', 3),

(3, 3, 'Product C', '2023-07-03', 2),

(4, 4, 'Product A', '2023-07-04', 1),

(5, 5, 'Product B', '2023-07-05', 4),

(6, 6, 'Product C', '2023-07-06', 2),

(7, 7, 'Product A', '2023-07-07', 3),

(8, 8, 'Product B', '2023-07-08', 2),

(9, 9, 'Product C', '2023-07-09', 5),

(10, 10, 'Product A', '2023-07-10', 1),

(11, 11, 'Product D', '2023-07-10', 3),

(12, 12, 'Product E', '2023-07-11', 6),

(13, 5, 'Product G', '2023-07-12', 2),

(14, 4, 'Product H', '2023-07-13', 4),

(15, 6, 'Product I', '2023-07-14', 3)

After Creating tables Solve Following tasks:

Task 1 :-

1. Write a query to retrieve all records from the Customers table..

2. Write a query to retrieve the names and email addresses of customers whose names start with 'J'.

3. Write a query to retrieve the order details (OrderID, ProductName, Quantity) for all orders..

4.Write a query to calculate the total quantity of products ordered.

5. Write a query to retrieve the names of customers who have placed an order.

6. Write a query to retrieve the products with a price greater than $10.00.

7. Write a query to retrieve the customer name and order date for all orders placed on or after '2023-07-05'.

8. Write a query to calculate the average price of all products.

9. Write a query to retrieve the customer names along with the total quantity of products they have ordered.

10. Write a query to retrieve the products that have not been ordered.

CoachX LIVE

Task 2 :-

1. **Write a query to retrieve the top 5 customers who have placed the highest total quantity of orders.**

2. **Write a query to calculate the average price of products for each product category.**

3. **Write a query to retrieve the customers who have not placed any orders.**

4. **Write a query to retrieve the order details (OrderID, ProductName, Quantity) for orders placed by customers whose names start with 'M'.**

5. **Write a query to calculate the total revenue generated from all orders.**

6. **Write a query to retrieve the customer names along with the total revenue generated from their orders.**

7. **Write a query to retrieve the customers who have placed at least one order for each product category.**

8. **Write a query to retrieve the customers who have placed orders on consecutive days.**

9. **Write a query to retrieve the top 3 products with the highest average quantity ordered.**

10. **Write a query to calculate the percentage of orders that have a quantity greater than the average quantity.**

Hints to solve the Queries

Task 1:

1. **Retrieve all records from the Customers table:**
   - Hint: Use the `SELECT` statement with a wildcard to fetch all columns from the `Customers` table.

2. **Retrieve the names and email addresses of customers whose names start with 'J':**
   - Hint: Use the `LIKE` operator with a wildcard `%` after 'J' to filter names.

3. **Retrieve the order details (OrderID, ProductName, Quantity) for all orders:**
   - Hint: Select specific columns from the `Orders` table.

4. **Calculate the total quantity of products ordered:**
   - Hint: Use the `SUM` function on the `Quantity` column in the `Orders` table.

5. **Retrieve the names of customers who have placed an order:**
   - Hint: Perform a `JOIN` between the `Customers` and `Orders` tables on the `CustomerID`.

6. **Retrieve the products with a price greater than $10.00:**
    - Hint: Use the `WHERE` clause to filter `Price` in the `Products` table.
7. **Retrieve the customer name and order date for all orders placed on or after '2023-07-05':**
    - Hint: Use a `JOIN` and the `WHERE` clause to filter orders based on the `OrderDate`.
8. **Calculate the average price of all products:**
    - Hint: Use the `AVG` function on the `Price` column in the `Products` table.
9. **Retrieve the customer names along with the total quantity of products they have ordered:**
    - Hint: Perform a `JOIN` and use the `SUM` function with a `GROUP BY` on `CustomerID`.
10. **Retrieve the products that have not been ordered:**
    - Hint: Use a `LEFT JOIN` between `Products` and `Orders` and filter with `NULL` check.

Task 2:

1. **Retrieve the top 5 customers who have placed the highest total quantity of orders:**
   - Hint: Use `GROUP BY` on `CustomerID`, `SUM` on `Quantity`, and order by the sum in descending order with a `LIMIT`.

2. **Calculate the average price of products for each product category:**
   - Hint: If you have categories, use `GROUP BY` on the category column and apply the `AVG` function on `Price`.

3. **Retrieve the customers who have not placed any orders:**
   - Hint: Use a `LEFT JOIN` between `Customers` and `Orders` and filter with `NULL` check.

4. **Retrieve the order details (OrderID, ProductName, Quantity) for orders placed by customers whose names start with 'M':**
   - Hint: Use a `JOIN` between `Customers` and `Orders` and filter `Name` with `LIKE`.

5. **Calculate the total revenue generated from all orders:**
   - Hint: Multiply `Quantity` by `Price` and sum it up across all orders.

6. **Retrieve the customer names along with the total revenue generated from their orders:**
    - Hint: Use `JOIN` and aggregate `SUM` of `Quantity * Price` with a `GROUP BY` on `CustomerID`.

7. **Retrieve the customers who have placed at least one order for each product category:**
    - Hint: Use a `HAVING` clause after `GROUP BY` to ensure orders exist for all categories.

8. **Retrieve the customers who have placed orders on consecutive days:**
    - Hint: Use `DATEDIFF` function or a `LAG`/`LEAD` window function to compare consecutive order dates for each customer.

9. **Retrieve the top 3 products with the highest average quantity ordered:**
    - Hint: Use `GROUP BY` on `ProductName` and `AVG` on `Quantity`, then order by the average quantity and limit the result to 3

10. **Calculate the percentage of orders that have a quantity greater than the average quantity:**
    - Hint: Calculate the average quantity first using `AVG`, then filter orders with quantity greater than this average, and calculate the percentage of such orders.