1. Create a Database for the following specifications with at least 10 entries:

Database Name: Product_Management	
Table	***Columns***
Rickshaw	registration_number, name, speed
Driver	driving_license_number, name
rickshaw_driver_mappi ng	rickshaw_id,driver_id, date_of_joining

Create the following API's:

i. Read only the **rickshaw_id** and **name** that has speed greater than 10
ii. Read all the driver information who has joined after June 24, 2023
iii. Update the **speed** of all the rickshaws to 15 whose **speed** is less than 5
iv. A Homepage to show all the previous tasks link

DB-

```
-- Create the database
CREATE DATABASE Product_Management;

-- Use the database
USE Product_Management;

-- Create the Rickshaw table
CREATE TABLE Rickshaw (
    registration_number VARCHAR(20) PRIMARY KEY,
    name VARCHAR(100),
    speed INT
);

-- Create the Driver table
CREATE TABLE Driver (
    driving_license_number VARCHAR(20) PRIMARY KEY,
    name VARCHAR(100)
);

-- Create the Rickshaw-Driver Mapping table
```

```
CREATE TABLE rickshaw_driver_mapping (
    rickshaw_id VARCHAR(20),
    driver_id VARCHAR(20),
    date_of_joining DATE,
    PRIMARY KEY (rickshaw_id, driver_id),
    FOREIGN KEY (rickshaw_id) REFERENCES Rickshaw(registration_number),
    FOREIGN KEY (driver_id) REFERENCES Driver(driving_license_number)
);

-- Insert sample data into Rickshaw
INSERT INTO Rickshaw VALUES
('R001', 'Rickshaw A', 12),
('R002', 'Rickshaw B', 3),
('R003', 'Rickshaw C', 15),
('R004', 'Rickshaw D', 8),
('R005', 'Rickshaw E', 5),
('R006', 'Rickshaw F', 20),
('R007', 'Rickshaw G', 10),
('R008', 'Rickshaw H', 18),
('R009', 'Rickshaw I', 4),
('R010', 'Rickshaw J', 6);

-- Insert sample data into Driver
INSERT INTO Driver VALUES
('DL001', 'Driver A'),
('DL002', 'Driver B'),
('DL003', 'Driver C'),
('DL004', 'Driver D'),
('DL005', 'Driver E'),
('DL006', 'Driver F'),
('DL007', 'Driver G'),
('DL008', 'Driver H'),
('DL009', 'Driver I'),
('DL010', 'Driver J');

-- Insert sample data into rickshaw_driver_mapping
INSERT INTO rickshaw_driver_mapping VALUES
('R001', 'DL001', '2023-05-01'),
('R002', 'DL002', '2023-06-25'),
('R003', 'DL003', '2023-07-10'),
('R004', 'DL004', '2023-06-20'),
('R005', 'DL005', '2023-07-15'),
('R006', 'DL006', '2023-04-22'),
('R007', 'DL007', '2023-08-01'),
```

```
('R008', 'DL008', '2023-03-30'),
('R009', 'DL009', '2023-06-28'),
('R010', 'DL010', '2023-09-10');
```

PHP-

```
const express = require('express');
const mysql = require('mysql2');
const app = express();
const port = 3000;

// Create a MySQL connection
const db = mysql.createConnection({
    host: 'localhost',
    user: 'root',    // Use your MySQL username
    password: '',    // Use your MySQL password
    database: 'Product_Management'
});

// Connect to MySQL
db.connect((err) => {
    if (err) throw err;
    console.log('Connected to the database');
});

// API 1: Read only the rickshaw_id and name that has speed greater than 10
app.get('/rickshaws/speed_greater_than_10', (req, res) => {
    const query = 'SELECT registration_number AS rickshaw_id, name FROM Rickshaw
WHERE speed > 10';
    db.query(query, (err, results) => {
        if (err) {
            res.status(500).send('Error retrieving data');
        } else {
            res.json(results);
        }
    });
});

// API 2: Read all the driver information who has joined after June 24, 2023
app.get('/drivers/joined_after_2023_06_24', (req, res) => {
    const query = `SELECT * FROM Driver
                WHERE driving_license_number IN
                (SELECT driver_id FROM rickshaw_driver_mapping WHERE date_of_joining >
'2023-06-24')`;
```

```
    db.query(query, (err, results) => {
        if (err) {
            res.status(500).send('Error retrieving data');
        } else {
            res.json(results);
        }
    });
});

// API 3: Update the speed of all the rickshaws to 15 whose speed is less than 5
app.put('/rickshaws/update_speed', (req, res) => {
    const query = 'UPDATE Rickshaw SET speed = 15 WHERE speed < 5';
    db.query(query, (err, results) => {
        if (err) {
            res.status(500).send('Error updating data');
        } else {
            res.send('Rickshaw speeds updated successfully');
        }
    });
});

// Homepage to show links to all the previous tasks
app.get('/', (req, res) => {
    res.send(`
        <h1>Product Management APIs</h1>
        <ul>
            <li><a href="/rickshaws/speed_greater_than_10">Rickshaws with speed greater than
10</a></li>
            <li><a href="/drivers/joined_after_2023_06_24">Drivers who joined after June 24,
2023</a></li>
            <li><a href="/rickshaws/update_speed">Update speed of rickshaws with speed less
than 5</a></li>
        </ul>
    `);
});

// Start the server
app.listen(port, () => {
    console.log(`Server is running on http://localhost:${port}`);
});
```

1. Create a Database for the following specifications with at least 10 entries:

Database Name: Product_Management	
Table	***Columns***
Product_Info	Product_id, Product_name, Stock, Price
Customer	Customer_id, Customer_name
Purchased_items	Customer_id, product_id, date

Create the following API's:

i. Read only the **Product_id** and **Product_name** that has a stock greater than 10

ii. Read all the customer information who has bought the products whose **Purchased_Product_id** is 5 and 6

iii. Update the **Stock** of all the products to 15 whose **Price** is less than 500

iv. A Homepage to show all the previous tasks link

DB-

```
-- Create the database
CREATE DATABASE Product_Management;

-- Use the database
USE Product_Management;

-- Create the Product_Info table
CREATE TABLE Product_Info (
    Product_id INT PRIMARY KEY,
    Product_name VARCHAR(100),
    Stock INT,
```

```
    Price DECIMAL(10, 2)
);

-- Create the Customer table
CREATE TABLE Customer (
    Customer_id INT PRIMARY KEY,
    Customer_name VARCHAR(100)
);

-- Create the Purchased_items table
CREATE TABLE Purchased_items (
    Customer_id INT,
    Product_id INT,
    Date DATE,
    PRIMARY KEY (Customer_id, Product_id),
    FOREIGN KEY (Customer_id) REFERENCES Customer(Customer_id),
    FOREIGN KEY (Product_id) REFERENCES Product_Info(Product_id)
);

-- Insert sample data into Product_Info
INSERT INTO Product_Info VALUES
(1, 'Product A', 20, 300),
(2, 'Product B', 8, 450),
(3, 'Product C', 15, 600),
(4, 'Product D', 5, 150),
(5, 'Product E', 12, 200),
(6, 'Product F', 25, 700),
(7, 'Product G', 9, 350),
(8, 'Product H', 30, 1000),
(9, 'Product I', 18, 120),
(10, 'Product J', 3, 400);

-- Insert sample data into Customer
INSERT INTO Customer VALUES
(1, 'Customer A'),
(2, 'Customer B'),
(3, 'Customer C'),
(4, 'Customer D'),
(5, 'Customer E');

-- Insert sample data into Purchased_items
INSERT INTO Purchased_items VALUES
(1, 5, '2023-07-01'),
(2, 6, '2023-08-15'),
```

```
(3, 4, '2023-06-20'),
(4, 5, '2023-09-10'),
(5, 6, '2023-10-05');
PHP-
const express = require('express');
const mysql = require('mysql2');
const app = express();
const port = 3000;

// Database connection
const db = mysql.createConnection({
    host: 'localhost',
    user: 'root',     // Replace with your MySQL username
    password: '',     // Replace with your MySQL password
    database: 'Product_Management'
});

// Connect to the database
db.connect((err) => {
    if (err) throw err;
    console.log('Connected to the database');
});

// API 1: Read only the Product_id and Product_name with stock greater than 10
app.get('/products/stock_greater_than_10', (req, res) => {
    const query = 'SELECT Product_id, Product_name FROM Product_Info WHERE Stock > 10';
    db.query(query, (err, results) => {
        if (err) {
            res.status(500).send('Error retrieving data');
        } else {
            res.json(results);
        }
    });
});

// API 2: Read customer information who bought products with Purchased_Product_id 5 and 6
app.get('/customers/purchased_product_5_6', (req, res) => {
    const query = `
        SELECT DISTINCT Customer.Customer_id, Customer_name
        FROM Customer
        INNER JOIN Purchased_items ON Customer.Customer_id =
Purchased_items.Customer_id
        WHERE Product_id IN (5, 6)
    `;
```

```
    db.query(query, (err, results) => {
        if (err) {
            res.status(500).send('Error retrieving data');
        } else {
            res.json(results);
        }
    });
});

// API 3: Update the stock of products to 15 where the price is less than 500
app.put('/products/update_stock', (req, res) => {
    const query = 'UPDATE Product_Info SET Stock = 15 WHERE Price < 500';
    db.query(query, (err, results) => {
        if (err) {
            res.status(500).send('Error updating data');
        } else {
            res.send('Stock updated successfully for products with price less than 500');
        }
    });
});

// Homepage with links to the APIs
app.get('/', (req, res) => {
    res.send(`
        <h1>Product Management APIs</h1>
        <ul>
            <li><a href="/products/stock_greater_than_10">Products with stock greater than
10</a></li>
            <li><a href="/customers/purchased_product_5_6">Customers who purchased products
5 and 6</a></li>
            <li><a href="/products/update_stock">Update stock of products with price less than
500</a></li>
        </ul>
    `);
});

// Start the server
app.listen(port, () => {
    console.log(`Server is running on http://localhost:${port}`);
});
```

1. Create a Database for the following specifications with at least 10 entries:

Database Name: Product_Management
Table Columns
Product_Info Product_id, Product_name, Stock, Price
Customer Customer_id, Customer_name, Purchased_Product_id, Total_ Purchase_amount

Create the following API's:

i. Read only the Product_id and Product_name that has a stock greater than 10
ii. Read all the customer information who has bought the products whose Purchased_Product_id is 5 and 6
iii. Update the Stock of all the products to 15 whose Price is less than 500
iv. A Homepage to show all the previous tasks link

DB-

```
-- Create the database
CREATE DATABASE Product_Management;

-- Use the database
USE Product_Management;

-- Create the Product_Info table
CREATE TABLE Product_Info (
    Product_id INT PRIMARY KEY,
    Product_name VARCHAR(100),
    Stock INT,
    Price DECIMAL(10, 2)
);

-- Create the Customer table
CREATE TABLE Customer (
    Customer_id INT PRIMARY KEY,
    Customer_name VARCHAR(100),
    Purchased_Product_id INT,
    Total_Purchase_amount DECIMAL(10, 2),
    FOREIGN KEY (Purchased_Product_id) REFERENCES Product_Info(Product_id)
);

-- Insert sample data into Product_Info
INSERT INTO Product_Info VALUES
(1, 'Product A', 20, 300),
```

```
(2, 'Product B', 8, 450),
(3, 'Product C', 15, 600),
(4, 'Product D', 5, 150),
(5, 'Product E', 12, 200),
(6, 'Product F', 25, 700),
(7, 'Product G', 9, 350),
(8, 'Product H', 30, 1000),
(9, 'Product I', 18, 120),
(10, 'Product J', 3, 400);

-- Insert sample data into Customer
INSERT INTO Customer VALUES
(1, 'Customer A', 5, 500),
(2, 'Customer B', 6, 700),
(3, 'Customer C', 4, 150),
(4, 'Customer D', 5, 200),
(5, 'Customer E', 6, 1200);
```

PHP

```
const express = require('express');
const mysql = require('mysql2');
const app = express();
const port = 3000;

// Database connection
const db = mysql.createConnection({
    host: 'localhost',
    user: 'root',     // Replace with your MySQL username
    password: '',     // Replace with your MySQL password
    database: 'Product_Management'
});

// Connect to the database
db.connect((err) => {
    if (err) throw err;
    console.log('Connected to the database');
});

// API 1: Read Product_id and Product_name with stock greater than 10
app.get('/products/stock_greater_than_10', (req, res) => {
    const query = 'SELECT Product_id, Product_name FROM Product_Info WHERE Stock > 10';
    db.query(query, (err, results) => {
        if (err) {
            res.status(500).send('Error retrieving data');
```

```
        } else {
            res.json(results);
        }
    });
});

// API 2: Read all customer information who bought products with Purchased_Product_id 5 and
6
app.get('/customers/purchased_product_5_6', (req, res) => {
    const query = `
        SELECT * FROM Customer
        WHERE Purchased_Product_id IN (5, 6)
    `;
    db.query(query, (err, results) => {
        if (err) {
            res.status(500).send('Error retrieving data');
        } else {
            res.json(results);
        }
    });
});

// API 3: Update the stock of products to 15 where price < 500
app.put('/products/update_stock', (req, res) => {
    const query = 'UPDATE Product_Info SET Stock = 15 WHERE Price < 500';
    db.query(query, (err, results) => {
        if (err) {
            res.status(500).send('Error updating data');
        } else {
            res.send('Stock updated successfully for products with price less than 500');
        }
    });
});

// API 4: Homepage with links to the tasks
app.get('/', (req, res) => {
    res.send(`
        <h1>Product Management APIs</h1>
        <ul>
            <li><a href="/products/stock_greater_than_10">Products with stock greater than
10</a></li>
            <li><a href="/customers/purchased_product_5_6">Customers who purchased products
5 and 6</a></li>
```

```
            <li><a href="/products/update_stock">Update stock of products with price less than
500</a></li>
        </ul>
    `);
});

// Start the server
app.listen(port, () => {
    console.log(`Server is running on http://localhost:${port}`);
});
```