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\_mapping | rickshaw\_id,driver\_id, date\_of\_joining |

Create the following API’s:

1. Read only the **rickshaw\_id** and **name** that has speed greater than 10
2. Read all the driver information who has joined after June 24, 2023
3. Update the **speed** of all the rickshaws to 15 whose **speed** is less than 5
4. 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:

1. Read only the **Product\_id** and **Product\_name** that has a stock greater than 10
2. Read all the customer information who has bought the products whose **Purchased\_Product\_id** is 5 and 6
3. Update the **Stock** of all the products to 15 whose **Price** is less than 500
4. 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}`);

});