students:

student_id	first_name	last_name	age	course_id
1	John	Doe	19	101
2	Alice	Smith	22	102
3	Bob	Johnson	21	103
4	Emma	White	23	101
5	Mark	Green	20	102

courses:

course_id	course_name	college_name
101	BCA	ABC College
102	MCA	XYZ College
103	BBA	ABC College

colleges:

college_name	location
ABC College	New York
XYZ College	San Francisco

1. **List all students who are older than 20.**
2. **Update the course name of a student based on their courseupdate ID.**
3. List all colleges that offer the course 'BCA'.
4. Add a new student to the table
5. Find the course name for all students based on their first name and last name.
6. **Delete a student record based on `studentid`**
7. Find the average age of students
8. **1**
9. Add a new course to the college table for a specific college

fruitid	name	price	stock
1	Apple	1.50	100
2	Banana	0.75	200
3	Orange	1.00	150
4	Mango	2.00	50
5	Pineapple	3.00	30

`sales` table:

sales_id	fruit_id	quantity	sales_date
101	1	10	2025-01-01
102	2	20	2025-01-02
103	3	15	2025-01-03
104	1	30	2025-01-04
105	4		2025-01-05

Mysql problem to create a database of fruit shop

1. Create a database named fruit shop
2. Create table fruits where there is fruit id, name, price and stock of fruits
3. Create a table sales.it include sales id, fruit id, quantity, sales date, and make foreign key of fruit id is same as in the table fruits (fruit id)
4. Insert the records or values into the fruits table
5. Insert the values into sales
6. List all fruits along with their prices and stocks.
7. Update the stock of a fruit after a sales
8. List all fruits that have been sold on '2024-12-02'
9. Find the fruit with the highest stock available
10. Find the fruit with lower price
11. Delete a fruit record after it is discontinued from the shop
12. Find the total quantity sold of each fruit
13. Find the total revenue generated by the shop
14. Add a new fruit to the fruits table

Employee management system

1. Create database company
2. Create table employees and add these data (employee_id,first_name,last_name,email,salary,department_id)
3. Insert 5 values into employee table
4. Create table department
5. Insert values into department
6. Now show the details of employees and department.
7. Filter the employees with salary greater than 60000
8. Select employees from IT department with salary > 60000 or from sales department.

1. Employee Table

employee_id	first_name	last_name	email	salary	department_id
1	John	Doe	john.doe@example.com	75000	1
2	Jane	Smith	jane.smith@example.com	65000	2
3	Mark	Johnson	mark.johnson@example.com	55000	1
4	Emily	Davis	emily.davis@example.com	85000	3
5	Michael	Williams	michael.williams@example.com	40000	2

2. Department Table

department_id	department_name
1	IT
2	Sales
3	HR

products:

id	product_name	category	price
1	Laptop	Electronics	800
2	Phone	Electronics	500
3	Shirt	Clothing	30
4	Shoes	Clothing	60
5	Headphones	Electronics	100

Questions:

1. Write an SQL query to find the names and prices of all products in the "Electronics" category.
2. Write an SQL query to find the product with the highest price.
3. Write an SQL query to update the price of "Shirt" to 40.
4. Write an SQL query to insert a new product, "Watch", in the "Electronics" category with a price of 150.
5. Write an SQL query to delete all products that belong to the "Clothing" category.

students:

id	name	age	grade	department
1	John	20	B	Math
2	Alice	22	A	Science
3	Bob	21	C	History
4	Sarah	23	B	Math
5	Emma	20	A	History

courses:

id	course_name	department
1	Calculus	Math
2	Algebra	Math
3	Biology	Science
4	Chemistry	Science
5	World History	History

Questions:

1. Write an SQL query to find the names of students and the courses they are taking, based on the department.
2. Write an SQL query to update the grade of student "Bob" to "A".
3. Write an SQL query to insert a new student "Liam", aged 24, with a grade of "B" and enrolled in the "Science" department.
4. Write an SQL query to find the average age of students who have a grade of "A".
5. Write an SQL query to delete all students who are in the "Math" department.
6. Write an SQL query to find the department with the highest number of students.

Online Store Database

1. Create database online store
2. Create table customers
   (Customer id, name, email)
3. Create table products
   (product id, name, price, quantity)
4. Create table orders
   (order id, customer id (Foreign Key Referencing Customers), order date)
5. Create table order item
   (order item id, order id (Foreign Key Referencing Orders), product id (Foreign Key referencing products), quantity)
6. Insert the values into tables
7. List all products
8. Find customers who placed the orders
9. Find the total number of orders
10. Show the product with stock 5
11. Show any customers last order date

Sure! Below are the complete tables based on the simplified problem statement:

1. Customers Table

This table stores information about customers.

Column	Data Type	Description
customer_id	INT	Primary Key, Auto Increment
name	VARCHAR	Customer's name
email	VARCHAR	Unique customer email

SQL Statement to Create Customers Table:

```
sql
Copy code
CREATE TABLE Customers (
    customer_id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100),
    email VARCHAR(100) UNIQUE
);
```

---

2. Products Table

This table stores information about the products available for sale.

Column	Data Type	Description
product_id	INT	Primary Key, Auto Increment
name	VARCHAR	Product name
price	DECIMAL	Product price (e.g., 500.00 for $500.00)
stock_quantity	INT	Available stock for the product

SQL Statement to Create Products Table:

```
sql
Copy code
CREATE TABLE Products (
    product_id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100),
    price DECIMAL(10, 2),
    stock_quantity INT
);
```

---

3. Orders Table

This table stores details about customer orders.

Column	Data Type	Description
order_id	INT	Primary Key, Auto Increment
customer_id	INT	Foreign Key referencing Customers

Column	Data Type	Description
order_date	DATE	Date when the order was placed

SQL Statement to Create Orders Table:

```
sql
Copy code
CREATE TABLE Orders (
    order_id INT AUTO_INCREMENT PRIMARY KEY,
    customer_id INT,
    order_date DATE,
    FOREIGN KEY (customer_id) REFERENCES Customers(customer_id)
);
```

---

4. Order_Items Table

This table stores information about the products in each order.

Column	Data Type	Description
order_item_id	INT	Primary Key, Auto Increment
order_id	INT	Foreign Key referencing Orders
product_id	INT	Foreign Key referencing Products
quantity	INT	Quantity of the product in the order

SQL Statement to Create Order_Items Table:

```
sql
Copy code
CREATE TABLE Order_Items (
    order_item_id INT AUTO_INCREMENT PRIMARY KEY,
    order_id INT,
    product_id INT,
    quantity INT,
    FOREIGN KEY (order_id) REFERENCES Orders(order_id),
    FOREIGN KEY (product_id) REFERENCES Products(product_id)
);
```

Example Data for Each Table

1. Customers Table:

```
sql
Copy code
INSERT INTO Customers (name, email) VALUES
('John Doe', 'john@example.com'),
('Jane Smith', 'jane@example.com');
```

2. Products Table:

```
sql
Copy code
INSERT INTO Products (name, price, stock_quantity) VALUES
('Laptop', 1000.00, 20),
```

```
('Phone', 500.00, 50),
('Headphones', 100.00, 30);
```

3. Orders Table:

```
sql
Copy code
INSERT INTO Orders (customer_id, order_date) VALUES
(1, '2024-12-01'),
(2, '2024-12-05');
```

4. Order_Items Table:

```
sql
Copy code
INSERT INTO Order_Items (order_id, product_id, quantity) VALUES
(1, 1, 1),  -- John Doe ordered 1 Laptop
(1, 2, 2),  -- John Doe ordered 2 Phones
(2, 3, 3);  -- Jane Smith ordered 3 Headphones
```

Summary of Table Structures:

1. **Customers**
   - `customer_id` (INT, Primary Key, Auto Increment)
   - `name` (VARCHAR)
   - `email` (VARCHAR, UNIQUE)
2. **Products**
   - `product_id` (INT, Primary Key, Auto Increment)
   - `name` (VARCHAR)
   - `price` (DECIMAL)
   - `stock_quantity` (INT)
3. **Orders**
   - `order_id` (INT, Primary Key, Auto Increment)
   - `customer_id` (INT, Foreign Key referencing Customers)
   - `order_date` (DATE)
4. **Order_Items**
   - `order_item_id` (INT, Primary Key, Auto Increment)
   - `order_id` (INT, Foreign Key referencing Orders)
   - `product_id` (INT, Foreign Key referencing Products)
   - `quantity` (INT)

This is the full table structure and example data. You can now run queries against this schema to explore how to retrieve and manipulate data in an online store database.