*Imagine you have been hired by a small retail business that wants to streamline its operations by creating a new database system. This database will be used to manage inventory, sales, and customer information. The business is a small corner shop that sells a range of groceries and domestic products. It might help to picture your local convenience store and think of what they sell. They also have a loyalty program, which you will need to consider when deciding what tables to create.*

*Write a 500-word essay explaining the steps you would take to set up and create this database. Your essay should cover the following points:*

1. ***Understanding the Business Requirements****:*
   1. *What kind of data will the database need to store?*
   2. *Who will be the users of the database, and what will they need to accomplish?*
2. ***Designing the Database Schema****:*
   1. *How would you structure the database tables to efficiently store inventory, sales, and customer information?*
   2. *What relationships between tables are necessary (e.g., how sales relate to inventory and customers)?*
3. ***Implementing the Database****:*
   1. *What SQL commands would you use to create the database and its tables?*
   2. *Provide examples of SQL statements for creating tables and defining relationships between them.*
4. ***Populating the Database****:*
   1. *How would you input initial data into the database? Give examples of SQL INSERT statements.*
5. ***Maintaining the Database****:*
   1. *What measures would you take to ensure the database remains accurate and up to date?*
   2. *How would you handle backups and data security?*

*Your essay should include specific examples of SQL commands and explain why each step is necessary for creating a functional and efficient database for the retail business.*

|  |  |
| --- | --- |
| Please write your 500-word essay here | **1. Understanding the Business Requirements**  **a. Data to Store:** The database must store details about products (name, category, price, quantity), sales transactions (date, products sold, quantities, total price), customer information (name, contact, loyalty points), and staff users (login credentials and roles). The loyalty program will also require tracking customers' purchases and points earned.  **b. Users and Their Needs:** Shop staff will use the database to check inventory levels, record sales, and manage customer accounts. The store manager might analyse sales trends, monitor inventory restocking, and manage loyalty rewards.  **Designing the Database Schema**  **a. Structuring Tables:** A relational database is ideal for efficiency and scalability. Key tables would include:   * **Products**: product\_id, name, category, price, stock\_quantity * **Customers**: customer\_id, name, email, phone, loyalty\_points * **Sales**: sale\_id, customer\_id (foreign key), sale\_date, total\_amount * **Sale\_Items**: sale\_item\_id, sale\_id (FK), product\_id (FK), quantity, item\_total * **Users**: user\_id, username, password\_hash, role   **b. Relationships:**   * Each sale links to one customer (optional if walk-in) but can include many products. * Each product can appear in many sale items. * The Sale\_Items table establishes a many-to-many relationship between sales and products.   **Implementing the Database**  **a. SQL Commands to Create Tables**  CREATE TABLE Products (  product\_id INT PRIMARY KEY AUTO\_INCREMENT,  name VARCHAR(100),  category VARCHAR(50),  price DECIMAL(10,2),  stock\_quantity INT  );  CREATE TABLE Customers (  customer\_id INT PRIMARY KEY AUTO\_INCREMENT,  name VARCHAR(100),  email VARCHAR(100),  phone VARCHAR(20),  loyalty\_points INT DEFAULT 0  );  CREATE TABLE Sales (  sale\_id INT PRIMARY KEY AUTO\_INCREMENT,  customer\_id INT,  sale\_date DATE,  total\_amount DECIMAL(10,2),  FOREIGN KEY (customer\_id) REFERENCES Customers(customer\_id)  );  CREATE TABLE Sale\_Items (  sale\_item\_id INT PRIMARY KEY AUTO\_INCREMENT,  sale\_id INT,  product\_id INT,  quantity INT,  item\_total DECIMAL(10,2),  FOREIGN KEY (sale\_id) REFERENCES Sales(sale\_id),  FOREIGN KEY (product\_id) REFERENCES Products(product\_id)  );  CREATE TABLE Users (  user\_id INT PRIMARY KEY AUTO\_INCREMENT,  username VARCHAR(50) UNIQUE,  password\_hash VARCHAR(255),  role VARCHAR(20)  );  **Populating the Database**  **a. Example SQL INSERT Statements:**  INSERT INTO Products (name, category, price, stock\_quantity)  VALUES ('Milk', 'Dairy', 1.20, 50);  INSERT INTO Customers (name, email, phone)  VALUES ('Mark Johnson', 'mark@example.com', '555-1234');  INSERT INTO Sales (customer\_id, sale\_date, total\_amount)  VALUES (1, '2025-04-15', 3.60);  INSERT INTO Sale\_Items (sale\_id, product\_id, quantity, item\_total)  VALUES (1, 1, 3, 3.60);  **Maintaining the Database**  **a. Ensuring Accuracy:** Use constraints (e.g., NOT NULL, CHECK) and triggers to update stock levels after sales. Regular audits and admin access restrictions will also help maintain integrity.  **b. Backups and Security:** Automate daily backups and store them securely offsite or in the cloud. Encrypt sensitive data like passwords and implement user roles for data access control. Regular updates and patches to the database system are essential to guard against vulnerabilities. |