**Conceptual ERD Model:**

- Correctly identified entities and relationships.

- Consider adding an entity for "Category" to further normalize the data.

**Logical ERD Model:**

- Well-defined attributes for each entity.

- Consider adding a "Status" attribute to the Enrollment entity (e.g., "Enrolled", "Completed", "InProgress").

**Physical ERD Model (Database Schema):**

- Proper use of primary keys, foreign keys, and data types.

- Consider indexing columns used in WHERE, JOIN, and ORDER BY clauses for improved performance.

**Additional Suggestions:**

1. Consider adding a "Rating" attribute to the Course entity to store user ratings.

2. Add a "Timestamp" attribute to the Completion entity to track completion time.

3. Create a separate table for "Questions" to store individual quiz questions.

4. Use ENUM data type for "Category" and "Status" attributes.

5. Use VARCHAR(255) for "Password" to accommodate hashed passwords.

**SQL Enhancements:**

1. Use CONSTRAINTS to enforce data integrity.

2. Create INDEXES on columns used in queries.

3. Consider using VIEWs to simplify complex queries.

**Next Steps:**

1. Implement the database schema.

2. Create CRUD (Create, Read, Update, Delete) operations for each entity.

3. Develop queries for reporting and analytics.

**Would you like me to:**

1. Review and refine the database schema?

2. Assist with CRUD operation implementation?

3. Develop queries for reporting and analytics?

4. Discuss database security and backup strategies?

**Conceptual ER Diagram**

**Products:**

| - Product\_ID |

| - Name |

| - Description |

| - Price |

| - Stock\_Level|

**Customers:**

| - Customer\_ID|

| - Name |

| - Email |

| - Address |

| - Phone |

**Orders:**

|- Order\_ID |

| - Customer\_ID|

| - Order\_Date |

| - Total |

| - Status |

**Order\_Items:**

| - Order\_Item\_ID|

| - Order\_ID |

| - Product\_ID |

| - Quantity |

**Payments:**

| - Payment\_ID |

| - Order\_ID |

| - Payment\_Date|

| - Amount |

| - Method |

**Logical ER Diagram:**

CREATE TABLE Products (

Product\_ID INT PRIMARY KEY,

Name VARCHAR(100),

Description VARCHAR(200),

Price DECIMAL(10, 2),

Stock\_Level INT

);

CREATE TABLE Customers (

Customer\_ID INT PRIMARY KEY,

Name VARCHAR(100),

Email VARCHAR(100),

Address VARCHAR(200),

Phone VARCHAR(20)

);

CREATE TABLE Orders (

Order\_ID INT PRIMARY KEY,

Customer\_ID INT,

Order\_Date DATE,

Total DECIMAL(10, 2),

Status VARCHAR(20),

FOREIGN KEY (Customer\_ID) REFERENCES Customers(Customer\_ID)

);

CREATE TABLE Order\_Items (

Order\_Item\_ID INT 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)

);

CREATE TABLE Payments (

Payment\_ID INT PRIMARY KEY,

Order\_ID INT,

Payment\_Date DATE,

Amount DECIMAL(10, 2),

Method VARCHAR(20),

FOREIGN KEY (Order\_ID) REFERENCES Orders(Order\_ID)

);

Physical ER Diagram (SQL)

CREATE TABLE Products (

Product\_ID INT PRIMARY KEY,

Name VARCHAR(100),

Description VARCHAR(200),

Price DECIMAL(10, 2),

Stock\_Level INT

);

CREATE TABLE Customers (

Customer\_ID INT PRIMARY KEY,

Name VARCHAR(100),

Email VARCHAR(100),

Address VARCHAR(200),

Phone VARCHAR(20)

);

CREATE TABLE Orders (

Order\_ID INT PRIMARY KEY,

Customer\_ID INT,

Order\_Date DATE,

Total DECIMAL(10, 2),

Status VARCHAR(20),

FOREIGN KEY (Customer\_ID) REFERENCES Customers(Customer\_ID)

);

CREATE TABLE Order\_Items (

Order\_Item\_ID INT 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)

);

CREATE TABLE Payments (

Payment\_ID INT PRIMARY KEY,

Order\_ID INT,

Payment\_Date DATE,

Amount DECIMAL(10, 2),

Method VARCHAR(20),

FOREIGN KEY (Order\_ID) REFERENCES Orders(Order\_ID)

);

Stored Procedures

CREATE PROCEDURE sp\_place\_order

@Customer\_ID INT,

@Order\_Date DATE,

@Total DECIMAL(10, 2)

AS

BEGIN

INSERT INTO Orders (Customer\_ID, Order\_Date, Total)

VALUES (@Customer\_ID, @Order\_Date, @Total);

END;

CREATE PROCEDURE sp\_update\_order

@Order\_ID INT,

@Status VARCHAR(20)

AS

BEGIN

UPDATE Orders

SET Status = @Status

WHERE Order\_ID = @Order\_ID;

END;

CREATE PROCEDURE sp\_process\_refund

@Order\_ID INT,

@Amount DECIMAL(10, 2)

AS

BEGIN

INSERT INTO Payments (Order\_ID, Payment\_Date, Amount, Method)

VALUES (@Order\_ID, GETDATE(), @Amount, 'Refund');

END;

Triggers

CREATE TRIGGER tr\_update\_product\_stock

ON Order\_Items

AFTER INSERT, UPDATE

AS

BEGIN

UPDATE Products

SET Stock\_Level = Stock\_Level - INSERTED.Quantity

WHERE Product\_ID = INSERTED.Product\_ID;

END;

CREATE TRIGGER tr\_update\_payment\_status

ON Payments

AFTER INSERT, UPDATE

AS

BEGIN

UPDATE Orders

SET Status = 'Paid'

WHERE Order\_ID = INSERTED.Order\_ID;

END;