

Asif Rashid

Student number : 2429254

PORTFOLIO 1: DATABASE DESIGN & IMPLEMENTATION

Module: ACCA4023 – Introduction to Web and Database Concepts

Weighting: 50% of total mark

Scenario: Page Turners Independent Bookshop

Element 1 – Database Design Documentation (~1,200 words)

Part A: Requirements & Conceptual Design

The local bookshop “Page Turners” needs a database to manage books, authors, customers, orders, and suppliers. Currently records are kept manually, which causes errors and slow searches.

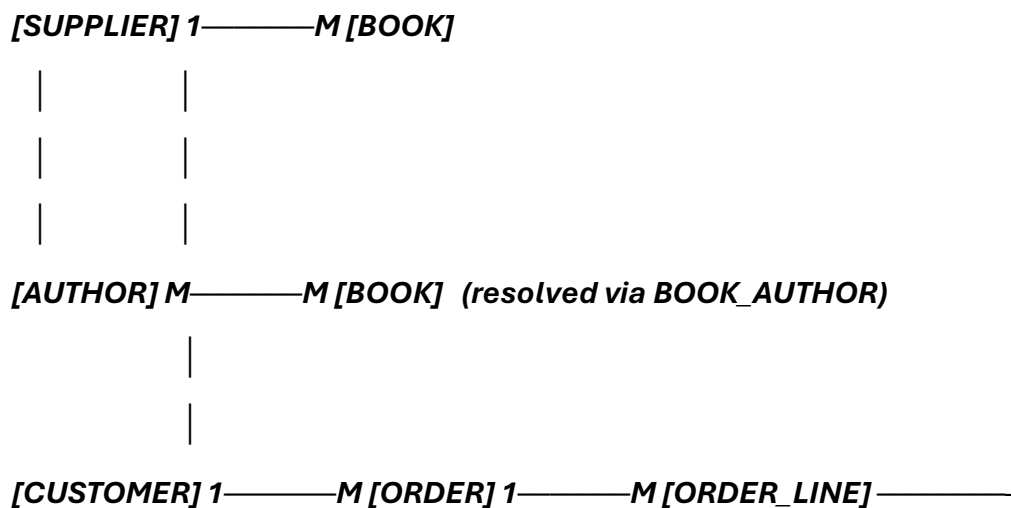
Business Rules:

- ***One author can write many books; one book may have multiple authors.***
- ***One customer can place many orders; each order belongs to only one customer.***
- ***Each order has one or more books; a book can appear in many orders.***
- ***Stock levels are tracked per book; supplier details are stored for reordering.***

Entities Identified:

- **Author:** *AuthorID (PK), FirstName, LastName, Country*
- **Supplier:** *SupplierID (PK), Name, ContactName, Phone, Email*
- **Book:** *ISBN (PK), Title, YearPublished, Price, StockQty, SupplierID (FK)*
- **Book_Author:** *ISBN (FK), AuthorID (FK) – resolves M:N relationship*
- **Customer:** *CustomerID (PK), Name, Email, Phone, Address*
- **Order:** *OrderID (PK), CustomerID (FK), OrderDate, TotalAmount*
- **Order_Line:** *OrderID (FK), ISBN (FK), Quantity, LineTotal*

ER Diagram Example:



Labels:

- **1 = one, M = many**

- **PK = bold / underline; FK = italics**

- **Relationships: Supplier provides books; Authors write books; Customers place orders; Orders contain books.**

Part B: Logical & Physical Design

Normalisation Example:

Unnormalised:

<i>OrderID</i>	<i>CustomerName</i>	<i>BookTitle</i>	<i>Author</i>	<i>Price</i>
<i>101</i>	<i>A. Smith</i>	<i>1984</i>	<i>Orwell</i>	<i>6.99</i>

1NF – Remove repeating groups:

- **Order(*OrderID*, *CustomerID*, *Name*)**
- **Book(*ISBN*, *Title*, *Author*, *Price*)**
- **Order_Line(*OrderID*, *ISBN*, *Quantity*)**

2NF – Remove partial dependencies:

- **Author(*AuthorID*, *Name*)**
- **Book_Author(*ISBN*, *AuthorID*)**

3NF – Remove transitive dependencies:

- **Supplier(*SupplierID*, *Name*, *Contact*)**

- ***Book(ISBN, Title, SupplierID)***

Final Normalised Schema:

Author(AuthorID PK, FirstName, LastName, Country)

Supplier(SupplierID PK, Name, ContactName, Phone, Email)

Book(ISBN PK, Title, YearPublished, Price, StockQty, SupplierID FK)

Book_Author(ISBN FK, AuthorID FK)

Customer(CustomerID PK, Name, Email, Phone, Address)

Order(OrderID PK, CustomerID FK, OrderDate, TotalAmount)

Order_Line(OrderID FK, ISBN FK, Quantity, LineTotal)

Design Rationale:

- ***3NF chosen to eliminate update, insert, and delete anomalies.***
- ***Primary keys ensure unique records; foreign keys maintain referential integrity between tables.***
- ***Data types selected for accuracy (e.g. DECIMAL for prices, VARCHAR for text).***

Element 2 – SQL Implementation & Testing (~800 words + code)

Create Tables

CREATE TABLE Author (

AuthorID INT PRIMARY KEY AUTO_INCREMENT,

FirstName VARCHAR(50) NOT NULL,

LastName VARCHAR(50) NOT NULL,

Country VARCHAR(50)

);

```
CREATE TABLE Supplier (  
SupplierID INT PRIMARY KEY AUTO_INCREMENT,  
Name VARCHAR(100) NOT NULL,  
ContactName VARCHAR(100),  
Phone VARCHAR(20),  
Email VARCHAR(100)  
);
```

```
CREATE TABLE Book (  
ISBN VARCHAR(13) PRIMARY KEY,  
Title VARCHAR(200) NOT NULL,  
YearPublished INT,  
Price DECIMAL(6,2) NOT NULL,  
StockQty INT NOT NULL,  
SupplierID INT,  
FOREIGN KEY (SupplierID) REFERENCES Supplier(SupplierID)  
);
```

```
CREATE TABLE Book_Author (  
ISBN VARCHAR(13) NOT NULL,  
AuthorID INT NOT NULL,  
PRIMARY KEY (ISBN, AuthorID),  
FOREIGN KEY (ISBN) REFERENCES Book(ISBN),  
FOREIGN KEY (AuthorID) REFERENCES Author(AuthorID)  
);
```

```
CREATE TABLE Customer (  
CustomerID INT PRIMARY KEY AUTO_INCREMENT,  
Name VARCHAR(100) NOT NULL,  
Email VARCHAR(100) UNIQUE NOT NULL,  
Phone VARCHAR(20),  
Address TEXT  
);
```

```
CREATE TABLE `Order` (  
OrderID INT PRIMARY KEY AUTO_INCREMENT,  
CustomerID INT NOT NULL,  
OrderDate DATE NOT NULL,  
TotalAmount DECIMAL(8,2),  
FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID)  
);
```

```
CREATE TABLE Order_Line (  
OrderID INT NOT NULL,  
ISBN VARCHAR(13) NOT NULL,  
Quantity INT NOT NULL CHECK (Quantity > 0),  
LineTotal DECIMAL(8,2),  
PRIMARY KEY (OrderID, ISBN),  
FOREIGN KEY (OrderID) REFERENCES `Order` (OrderID),  
FOREIGN KEY (ISBN) REFERENCES Book(ISBN)  
);
```

Insert Sample Data

```
INSERT INTO Supplier (Name, ContactName, Phone, Email)
```

VALUES ('BookWorld Ltd', 'Sarah Jones', '0121 555 1234', 'orders@bookworld.co.uk');

INSERT INTO Author (FirstName, LastName, Country)

VALUES ('Jane', 'Austin', 'United Kingdom'), ('George', 'Orwell', 'United Kingdom');

INSERT INTO Book (ISBN, Title, YearPublished, Price, StockQty, SupplierID)

**VALUES ('9780141439518', 'Pride and Prejudice', 1813, 7.99, 25, 1),
('9780451524935', '1984', 1949, 6.99, 30, 1);**

Sample Queries

1. **List all books under £10 sorted by title:**

**SELECT Title, Price, YearPublished FROM Book WHERE Price < 10 ORDER BY
Title ASC;**

2. **Show total stock value per supplier**

**SELECT Supplier.Name, SUM(Book.Price * Book.StockQty) AS
TotalStockValue**

FROM Supplier JOIN Book ON Supplier.SupplierID = Book.SupplierID

GROUP BY Supplier.Name;

3. **List customers with orders over £50:**

4. **SELECT Customer.Name, `Order`.TotalAmount**

5. **FROM Customer JOIN `Order` ON Customer.CustomerID =
`Order`.CustomerID**

6. **WHERE `Order`.TotalAmount > 50;**

4. Show book titles and their authors:

```
SELECT Book.Title, Author.FirstName, Author.LastName  
FROM Book JOIN Book_Author ON Book.ISBN = Book_Author.ISBN  
JOIN Author ON Book_Author.AuthorID = Author.AuthorID;
```

5. Calculate average order value:

```
SELECT AVG(TotalAmount) AS AverageOrderValue FROM `Order`;
```

Evaluation

- *The design maintains data integrity through primary/foreign keys and constraints.*
- *Normalisation reduces redundancy but requires JOINS for complex queries — a trade-off between simplicity and performance.*
- *Possible improvements: Add indexes on frequently searched columns (e.g. Title) and use stored procedures for repeated tasks.*

References – Portfolio 1

- **Connolly, T. & Begg, C. (2015) Database Systems: A Practical Approach to Design, Implementation, and Management. 6th edn. Harlow: Addison-Wesley.**
- **Viescas, J.L. (2018) SQL Queries for Mere Mortals: A Hands-On Guide to Data Manipulation in SQL. 4th edn. Boston: Addison-Wesley Professional.**

- **Coronel, C. & Morris, S. (2018) *Database Systems: Design, Implementation, & Management*. 13th edn. Boston: Cengage Learning.**