```sql
-- Stores different book categories (e.g., Fiction, Non-fiction, Science,
etc.)
CREATE TABLE category (
    CategoryID TEXT PRIMARY KEY,        -- Unique identifier for each
category
    CategoryName TEXT NOT NULL          -- Descriptive name of the category
);

-- Stores publisher details for books
CREATE TABLE publisher (
    PublisherID TEXT PRIMARY KEY,       -- Unique identifier for each
publisher
    Name TEXT NOT NULL,                 -- Publisher's name
    Address TEXT,                       -- Publisher's address (optional)
    ContactInfo TEXT                    -- Contact details for the
publisher
);

-- Stores details of authors who write books
CREATE TABLE author (
    AuthorID TEXT PRIMARY KEY,          -- Unique identifier for each
author
    Name TEXT NOT NULL                  -- Full name of the author
);

-- Stores book details and links to their publisher and category
CREATE TABLE book (
    ISBN TEXT PRIMARY KEY,              -- International Standard Book
Number
    Title TEXT NOT NULL,                -- Title of the book
    Year INTEGER,                       -- Year the book was published
    Price DECIMAL(10,2),                -- Price of the book
    PublisherID TEXT,                   -- References the publisher of the
book
    CategoryID TEXT,                    -- References the category of the
book
    FOREIGN KEY (PublisherID) REFERENCES publisher(PublisherID) ON DELETE
SET NULL,
    FOREIGN KEY (CategoryID) REFERENCES category(CategoryID) ON DELETE SET
NULL
);
```

```sql
-- Associates books with authors (many-to-many relationship)
CREATE TABLE book_author (
    ISBN TEXT,                          -- Book identifier
    AuthorID TEXT,                      -- Author identifier
    PRIMARY KEY (ISBN, AuthorID),       -- Ensures unique book-author pairs
    FOREIGN KEY (ISBN) REFERENCES book(ISBN) ON DELETE CASCADE,
    FOREIGN KEY (AuthorID) REFERENCES author(AuthorID) ON DELETE CASCADE
);

-- Stores basic customer information
CREATE TABLE customer (
    CustomerID TEXT PRIMARY KEY,        -- Unique identifier for each
customer
    Name TEXT NOT NULL,                 -- Full name of the customer
    Address TEXT                        -- Customer's address (optional)
);

-- Stores customer contact details separately
CREATE TABLE customer_contact (
    CustomerID TEXT PRIMARY KEY,        -- References a customer
    Email TEXT UNIQUE NOT NULL,         -- Unique email for the customer
    PhoneNumber TEXT UNIQUE NOT NULL,   -- Unique phone number for the
customer
    FOREIGN KEY (CustomerID) REFERENCES customer(CustomerID) ON DELETE
CASCADE
);

-- Stores orders placed by customers
CREATE TABLE customer_order (
    OrderID TEXT PRIMARY KEY,           -- Unique identifier for each order
    CustomerID TEXT NOT NULL,           -- Customer who placed the order
    OrderDate DATE NOT NULL DEFAULT CURRENT_DATE,  -- Date of the order
    FOREIGN KEY (CustomerID) REFERENCES customer(CustomerID) ON DELETE
CASCADE
);

-- Stores specific books included in each order
CREATE TABLE orderItem (
    OrderItemID TEXT PRIMARY KEY,       -- Unique identifier for each order
item
    OrderID TEXT NOT NULL,              -- References the order
    ISBN TEXT NOT NULL,                 -- References the purchased book
    Quantity INTEGER NOT NULL CHECK (Quantity > 0),  -- Number of copies
```

```
ordered
    Price DECIMAL(10,2) NOT NULL,       -- Price per copy at the time of
order
    FOREIGN KEY (OrderID) REFERENCES customer_order(OrderID) ON DELETE
CASCADE,
    FOREIGN KEY (ISBN) REFERENCES book(ISBN) ON DELETE CASCADE
);

-- Tracks available inventory for each book
CREATE TABLE inventory (
    ISBN TEXT PRIMARY KEY,              -- Book identifier
    StockQuantity INTEGER NOT NULL CHECK (StockQuantity >= 0),  --
Available stock
    FOREIGN KEY (ISBN) REFERENCES book(ISBN) ON DELETE CASCADE
);

-- Tracks the popularity score of each book
CREATE TABLE bookDemand (
    ISBN TEXT PRIMARY KEY,              -- Book identifier
    Popularity INTEGER NOT NULL CHECK (Popularity >= 0),  -- Popularity
score
    FOREIGN KEY (ISBN) REFERENCES book(ISBN) ON DELETE CASCADE
);

-- Tracks total sales and costs for each book
CREATE TABLE profitMargin (
    ISBN TEXT PRIMARY KEY,              -- Book identifier
    SalesTotal DECIMAL(10,2) DEFAULT 0, -- Total sales revenue
    CostTotal DECIMAL(10,2) DEFAULT 0,  -- Total cost incurred
    FOREIGN KEY (ISBN) REFERENCES book(ISBN) ON DELETE CASCADE
);

-- ================================================
-- INDEXES TO IMPROVE DATABASE PERFORMANCE
-- ================================================

-- Indexes for foreign key relationships to improve JOIN operations
CREATE INDEX idx_book_publisher ON book(PublisherID);
CREATE INDEX idx_book_category ON book(CategoryID);
CREATE INDEX idx_book_author_isbn ON book_author(ISBN);
CREATE INDEX idx_book_author_author ON book_author(AuthorID);
CREATE INDEX idx_customer_order_customer ON customer_order(CustomerID);
CREATE INDEX idx_order_item_order ON orderItem(OrderID);
```

```sql
CREATE INDEX idx_order_item_book ON orderItem(ISBN);

-- ================================================
-- VIEWS FOR COMMON DATA RETRIEVAL OPERATIONS
-- ================================================

-- View for calculating total sales revenue by book category
CREATE VIEW CategorySales AS
SELECT c.CategoryID, c.CategoryName, SUM(oi.Quantity * oi.Price) AS
TotalSales
FROM category c
JOIN book b ON c.CategoryID = b.CategoryID
JOIN orderItem oi ON b.ISBN = oi.ISBN
GROUP BY c.CategoryID, c.CategoryName;

-- View for ranking authors by the popularity of their books
CREATE VIEW AuthorPopularity AS
SELECT a.AuthorID, a.Name, SUM(bd.Popularity) AS TotalPopularity
FROM author a
JOIN book_author ba ON a.AuthorID = ba.AuthorID
JOIN bookDemand bd ON ba.ISBN = bd.ISBN
GROUP BY a.AuthorID, a.Name;

-- View for books with low inventory (less than 5 copies)
CREATE VIEW LowStockBooks AS
SELECT b.ISBN, b.Title, i.StockQuantity
FROM book b
JOIN inventory i ON b.ISBN = i.ISBN
WHERE i.StockQuantity < 5;

-- View for profit margin calculation
CREATE VIEW BookProfitability AS
SELECT b.ISBN, b.Title, pm.SalesTotal, pm.CostTotal,
       (pm.SalesTotal - pm.CostTotal) AS Profit,
       CASE
         WHEN pm.CostTotal > 0 THEN ROUND((pm.SalesTotal - pm.CostTotal) *
100.0 / pm.CostTotal, 2)
         ELSE 0
       END AS ProfitMarginPercent
FROM book b
JOIN profitMargin pm ON b.ISBN = pm.ISBN;

-- View for customer purchase history
```

```sql
CREATE VIEW CustomerPurchaseHistory AS
SELECT c.CustomerID, c.Name, co.OrderID, co.OrderDate,
       b.ISBN, b.Title, oi.Quantity, oi.Price,
       (oi.Quantity * oi.Price) AS TotalPrice
FROM customer c
JOIN customer_order co ON c.CustomerID = co.CustomerID
JOIN orderItem oi ON co.OrderID = oi.OrderID
JOIN book b ON oi.ISBN = b.ISBN;
```