1. Entities and Normalisation Table

To design a database for the Smith and Co Second-Hand Bookshop and normalize it to 3NF:

Entities:

1. Customer
    1. Attributes: CustomerID (PK), Name, Email, Address1, Address2, Address3, Postcode.
2. Author
    1. Attributes: AuthorID (PK), AuthorName.
3. Book
    1. Attributes: BookID (PK), Title, AuthorID (FK), Genre, ISBN.
4. Transaction
    1. Attributes: TransactionID (PK), CustomerID (FK), BookID (FK), PurchaseDate, SalePrice.

Normalisation Steps:

1. Unnormalized Form (UNF):
    1. Data is not structured; information about customers, books, and transactions is stored in one table.
2. First Normal Form (1NF):
    1. Remove repeating groups; each piece of data is atomic.
    2. Table:
        1. Customer (CustomerID, Name, Email, Address1, Address2, Address3, Postcode)
        2. Book (BookID, Title, AuthorID, Genre, ISBN)
        3. Author (AuthorID, AuthorName)

        4. Transaction (TransactionID, CustomerID, BookID, PurchaseDate, SalePrice)
3. Second Normal Form (2NF):
    1. Remove partial dependencies; all non-key attributes depend on the whole primary key.
    2. Table:
        1. Customer remains unchanged.
        2. Book now links AuthorID as FK.
        3. Transaction links CustomerID and BookID as FK.
4. Third Normal Form (3NF):
    1. Remove transitive dependencies; non-key attributes must depend only on the primary key.
    2. Ensured AuthorName relates only to AuthorID, and ISBN is uniquely tied to BookID.

Final Tables:

Entity	Attributes	Primary Key (PK)	Foreign Key (FK)
Customer	CustomerID, Name, Email, Address1, Address2, Address3, Postcode	CustomerID	None
Author	AuthorID, AuthorName	AuthorID	None
Book	BookID, Title, AuthorID, Genre, ISBN	BookID	AuthorID

Transaction	TransactionID, CustomerID, BookID, PurchaseDate, SalePrice	TransactionID	CustomerID, BookID

---

2. Short Report: Database Attacks

Attack 1: SQL Injection

Reason for Targeting: The database holds sensitive customer data (e.g., emails, addresses) and financial transactions, making it valuable to attackers.

Type of Attack: SQL injection occurs when attackers manipulate input fields to execute malicious SQL commands. For example:

```
SELECT * FROM Customers WHERE Email = 'x' OR '1'='1';
```

This command could bypass login systems or expose entire customer tables.

Data Extracted:

- Customer emails and addresses.
- Transaction details like book purchases and sales.

Prevention:

- Use parameterized queries or prepared statements.
- Validate and sanitize user inputs.

Attack 2: Data Breach via Weak Authentication

Reason for Targeting: Attackers may exploit weak authentication protocols to gain unauthorized access to the database.

Type of Attack: Credential stuffing or brute force attacks. If admin passwords are weak or reused, attackers could access the system and steal data.

Data Extracted:

- Complete customer details.
- Purchase history and transaction records.
- Author information and book details.

Prevention:

- Implement multi-factor authentication (MFA).
- Enforce strong password policies and use hashed passwords.

References:

1. Garfinkel, S. (2021). *Database Security: Concepts and Practices*. Addison-Wesley.
2. OWASP. (2024). *SQL Injection Prevention Cheat Sheet*. [Online] Available at: OWASP SQL Injection.

Here are the SQL queries to create the tables for the Smith and Co Second-Hand Bookshop database normalized to 3NF:

---

1. Create `Customer` Table

```
CREATE TABLE Customer (
    CustomerID INT AUTO_INCREMENT PRIMARY KEY,
    Name VARCHAR(100) NOT NULL,
    Email VARCHAR(150) NOT NULL UNIQUE,
    Address1 VARCHAR(255),
    Address2 VARCHAR(255),
    Address3 VARCHAR(255),
    Postcode VARCHAR(10)
);
```

---

2. Create `Author` Table

```
CREATE TABLE Author (
    AuthorID INT AUTO_INCREMENT PRIMARY KEY,
    AuthorName VARCHAR(100) NOT NULL UNIQUE
);
```

---

3. Create `Book` Table

```
CREATE TABLE Book (
    BookID INT AUTO_INCREMENT PRIMARY KEY,
    Title VARCHAR(150) NOT NULL,
    AuthorID INT NOT NULL,
    Genre VARCHAR(100),
    ISBN VARCHAR(13) UNIQUE,
    FOREIGN KEY (AuthorID) REFERENCES Author(AuthorID)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);
```

---

4. Create `Transaction` Table

```
CREATE TABLE Transaction (
    TransactionID INT AUTO_INCREMENT PRIMARY KEY,
    CustomerID INT NOT NULL,
    BookID INT NOT NULL,
    PurchaseDate DATE NOT NULL,
    SalePrice DECIMAL(10, 2) NOT NULL,
    FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID)
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    FOREIGN KEY (BookID) REFERENCES Book(BookID)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);
```

---

5. Insert Sample Data

Insert Data into `Customer`

```
INSERT INTO Customer (Name, Email, Address1, Address2, Address3, Postcode)
VALUES
```

```
    ('Julie Jones', 'j.jones@gmail.com', '33 Main Crescent', 'South Street',
'Brighton', 'BN6 7AD');
```

Insert Data into `Author`

```
INSERT INTO Author (AuthorName)
VALUES
    ('James Baldwin'),
    ('Zadie Smith'),
    ('Jane Austen'),
    ('Stephen King');
```

Insert Data into `Book`

```
INSERT INTO Book (Title, AuthorID, Genre, ISBN)
VALUES
    ('Giovanni\'s Room', 1, 'Fiction', '9780345806567'),
    ('White Teeth', 2, 'Fiction', '9780140276336'),
    ('Emma', 3, 'Classic', '9780141439587'),
    ('Carrie', 4, 'Horror', '9780307743664'),
    ('On Beauty', 2, 'Fiction', '9780141019451');
```

Insert Data into `Transaction`

```
INSERT INTO Transaction (CustomerID, BookID, PurchaseDate, SalePrice)
VALUES
    (1, 1, '2019-04-13', 9.99),
    (1, 2, '2001-01-08', 7.99),
    (1, 3, '2015-06-22', 4.00),
    (1, 4, '2015-12-02', 3.50),
    (1, 5, '2020-08-07', 3.50);
```

---

6. Sample Queries

a) Retrieve Customer Details and Purchase History

```
SELECT
    Customer.Name AS CustomerName,
    Customer.Email,
    Book.Title AS BookTitle,
    Author.AuthorName,
    Transaction.PurchaseDate,
    Transaction.SalePrice
FROM
    Transaction
JOIN Customer ON Transaction.CustomerID = Customer.CustomerID
JOIN Book ON Transaction.BookID = Book.BookID
JOIN Author ON Book.AuthorID = Author.AuthorID
WHERE Customer.CustomerID = 1;
```

b) List All Books by an Author

```
SELECT
    Book.Title,
    Book.Genre,
    Author.AuthorName
FROM
    Book
JOIN Author ON Book.AuthorID = Author.AuthorID
WHERE Author.AuthorName = 'Zadie Smith';
```

c) Total Sales by a Customer

```
SELECT
    Customer.Name AS CustomerName,
    SUM(Transaction.SalePrice) AS TotalSpent
FROM
    Transaction
JOIN Customer ON Transaction.CustomerID = Customer.CustomerID
WHERE Customer.CustomerID = 1
GROUP BY Customer.Name;
```

These queries handle creating the normalized database schema and performing common operations like retrieving purchase history, listing books by an author, and calculating total sales.